Numerical Analysis for Scientists and Engineers: Theory and C Programs

Madhumangal Pal
Department of Applied Mathematics with
Oceanology and Computer Programming
Vidyasagar University
Midnapore - 721102

.

Dedicated to my parents

Preface

Numerical Analysis is a multidisciplinary subject. It has formed an integral part of the undergraduate and postgraduate curriculum in Mathematics, Computer Science, Physics, Commerce and different Engineering streams. Numerical Analysis shows the way to obtain numerical answers to applied problems. Numerical methods stand there where analytical methods may fail or complicated to solve the problem. For example, in finding the roots of transcendental equations or in solving non-linear differential equations. So, it is quite impossible to train the students in applied sciences or engineering without an adequate knowledge of numerical methods.

The book is suitable for undergraduate as well as for postgraduate students and advanced readers. Ample material is presented so that instructors will able to select topics appropriate to their needs. The book contains ten chapters.

In Chapter 1, different types of errors and their sources in numerical computation are presented. The representation of floating point numbers and their arithmetic are studied in this chapter.

Finite difference operators, relations among them are well studied in Chapter 2. The difference equations and their solution methods are also introduced here.

Chapter 3 is devoted to single and bivariate interpolations. Different types of interpolation methods such as Lagrange, Newton, Bessal, Stirling, Hermite, Everette are incorporated here. Inverse and cubic spline interpolation techniques are also presented in this chapter. Several bivariate methods are presented here.

Several methods such as graphical, tabulation, bisection, regula-falsi, fixed point iteration, Newton-Raphson, Aitken, secant, Chebyshev and Muller are well studied to solve an algebraic and transcendental equation in Chapter 4. The geometrical meaning and the rate of convergence of the above methods are also presented. The very new method, modified Newton-Raphson with cubic convergence is introduced here. Birge-Vieta, Bairstow and Graeffe's root squaring methods are deduced and illustrated to find the roots of a polynomial equation. The methods to solve a system of non-linear equations are introduced here.

Chapter 5 deals to solve a system of linear equations. Different direct and iterative methods such as matrix inverse, Gauss-Jordon, Gauss elimination, LU decomposition, Cholesky, matrix partition, Jacobi, Gauss-Seidal and relaxation are well studied here. The very new methods to find tri-diagonal determinant and to solve a tri-diagonal system of equations are incorporated. A method to solve ill-condition system is discussed. The generalised inverse of a matrix is introduced. Also, least squares solution method for an inconsistent system is illustrated here.

Determination of eigenvalues and eigenvectors of a matrix is very important problem in applied science and engineering. In Chapter 6, different methods, viz., Leverrier-Faddeev, Rotishauser, Power, Jacobi, Givens and Householder are presented to find the eigenvalues and eigenvectors for arbitrary and symmetric matrices.

Chapter 7 contains indepth presentation of several methods to find derivative and integration of a functions. Three types of integration methods, viz., Newton-Cotes (trapezoidal, Simpson, Boole, Weddle), Gaussian (Gauss-Legendre, Lobatto, Radau, Gauss-Chebyshev, Gauss-Hermite, Gauss-Leguerre, Gauss-Jacobi) and Monte Carlo are well studies here. Euler-Maclaurin sum formula, Romberg integration are also studied in this chapter. An introduction to find double integration is also given here.

To solve ordinary differential equations, Taylor series, Picard, Euler, Runge-Kutta, Runge-Kutta-Fehlberg, Runge-Kutta-Butcher, Adams-Bashforth-Moulton, Milne, finite-difference, shooting and finite element methods are discussed in Chapter 8. Stability analysis of some methods are also done.

An introduction to solve partial differential equation is given in Chapter 9. The finite difference methods to solve parabolic, hyperbolic and elliptic PDEs are discussed here.

Least squares approximation techniques are discussed in Chapter 10. The method to fit straight line, parabolic, geometric, etc., curves are illustrated here. Orthogonal polynomials, their applications and Chebyshev approximation are discussed in this chapter.

The algorithms and programmes in C are supplied for the most of the important methods discussed in this book.

At first I would like to thank Prof. N. Dutta and Prof. R.N. Jana, as from their book I learnt my first lessons in the subject.

In writing this book I have taken help from several books, research articles and some websites mentioned in the bibliography. So, I acknowledge them gratefully.

This book could not have been complete without the moral and loving support and also continuous encouragement of my wife *Anita* and my son *Aniket*.

Also, I would like to express my sincere appreciation to my teachers and colleagues specially Prof. M. Maiti, Prof. T.K. Pal and Prof. R.N. Jana as they have taken all the academic and administrative loads of the department on their shoulders, providing me with sufficient time to write this book. I would also like to acknowledge my other colleagues Dr. K. De and Dr. S. Mondal for their encouragement.

I express my sincerest gratitude to my teacher Prof. G.P.Bhattacharjee, Indian Institute of Technology, Kharagpur, for his continuous encouragement.

I feel great reverence for my parents, sisters, sister-in-law and relatives for their blessings and being a constant source of inspiration.

I would like to thank to Sk. Md. Abu Nayeem, Dr. Amiya K. Shyamal, Dr. Anita Saha, for scrutinizing the manuscript.

I shall feel great to receive constructive criticisms for the improvement of the book from the experts as well as the learners.

I thank the Narosa Publishing House Pvt. Ltd. for their sincere care in the publication of the book.

Madhumangal Pal

Contents

List of Algorithms and Programs

Chapter 1

Errors in Numerical Computations

The solutions of mathematical problems are of two types: analytical and numerical. The analytical solutions can be expressed in closed form and these solutions are error free. On the other hand, numerical method is a division of mathematics which solves problems using computational machine (computer, calculator, etc.). But, for some classes of problems it is very difficult to obtain an analytical solution. For example, the Indian populations are known at the years 1951, 1961, 1971, 1981, 1991, 2001. There is no analytical method available to determine the population in the year, say, 2000. But, using numerical method one can determine the population in the said year. Again, sometimes we observed that the solutions of non-linear differential equations cannot be determined by analytical methods, but, such problems can easily be solved by numerical methods. Numerical computations are almost invariably contaminated by errors, and it is important to understand the source, propagation, magnitude, and rate of growth of these errors.

In this age of computer, many complicated and large problems are solved in significantly less time. But, without using numerical methods we cannot solve any mathematical problem using computer, as analytical methods are not suitable to solve a problem by computer. Thus, the numerical methods are highly appreciated and extensively used by Mathematicians, Computer Scientists, Statisticians, Engineers and others.

1.1 Sources of Errors

The solution of a problem obtained by numerical method contains some errors. To minimize the errors, it is most essential to identify the causes or sources of the errors and their growth and propagation in numerical computation. Three types of errors, viz., inherent errors, round-off errors and truncation errors, occur in finding the solution of a problem using numerical method. These three type of errors are discussed below.

(i) **Inherent errors:** This type of errors is present in the statement of the problem itself, before determining its solution. Inherent errors occur due to the simplified assumptions made in the process of mathematical modelling of a problem. It can also arise when the data is obtained from certain physical measurements of the parameters of the proposed problem.

(ii) **Round-off errors:** Generally, the numerical methods are carried out using calculator or computer. In numerical computation, all the numbers are represented by decimal fraction. Some numbers such as 1/3, 2/3, 1/7 etc. can not be represented by decimal fraction in finite numbers of digits. Thus, to get the result, the numbers should be rounded-off into some finite number of digits.

Again, most of the numerical computations are carried out using calculator and computer. These machines can store the numbers up to some finite number of digits. So in arithmetic computation, some errors will occur due to the finite representation of the numbers; these errors are called round-off error. Thus, round-off errors occur due to the finite representation of numbers during arithmetic computation. These errors depend on the word length of the computational machine.

(iii) **Truncation errors:** These errors occur due to the finite representation of an inherently infinite process. For example, the use of a finite number of terms in the infinite series to compute the value of $\cos x, \sin x, e^x$, etc.

The Taylor's series expansion of $\sin x$ is

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots .$$

This is an infinite series expansion. If only first five terms are taken to compute the value of $\sin x$ for a given x, then we obtain an approximate result. Here, the error occurs due to the truncation of the series. Suppose, we retain the first n terms, the truncation error (E_{trunc}) is given by

$$E_{\text{trunc}} \leq \frac{x^{2n+1}}{(2n+1)!}.$$

It may be noted that the truncation error is independent of the computational machine.

1.2 Exact and Approximate Numbers

To solve a problem, two types of numbers are used. They are exact and approximate. Exact number gives a true value of a result and approximate number gives a value which is closed to the true value.

For example, in the statements ‘a triangle has three sides’, ‘there are 2000 people in a locality’, ‘a book has 450 pages’ the numbers 3, 2000 and 450 are exact numbers. But, in the assertions ‘the height of a pupil is 178 cm’, ‘the radius of the Earth is 6400 km’, ‘the mass of a match box is ten gram’, the numbers 178, 6400 and 10 are approximate numbers.

This is due to the imperfection of measuring instruments we use. There are no absolutely exact measuring instruments; each of them has its own accuracy. Thus, the height of a pupil is 178 cm is not absolute measurement. In the second example, the radius of the Earth is very concept; actually, the Earth is not a sphere at all, and we can use its radius only in approximate terms. In the last example, the approximation of the number is also defined by the fact that different boxes may have different masses and the number 10 defines the mass of a particular box.

One important observation is that, same number may be exact as well as approximate. For example, the number 3 is exact when it represents the number of sides of a triangle and approximate if we use it to represent the number π when calculating the area of a circle using the formula πr^2.

Independently, the numbers $1, 2, 3, \frac{1}{2}, \frac{5}{3}, \sqrt{2}, \pi, e$, etc. written in this manner are exact. An approximate value of π is 3.1416, a better approximation of it is 3.14159265. But one cannot write the exact value of π.

The accuracy of calculations is defined by the number of digits in the result which enjoy confidence. The **significant digits** or **significant figures** of a number are all its digits, except for zeros which appear to the left of the first non-zero digit. Zeros at the end of a number are always significant digit. For example, the numbers 0.001205 and 356.800 have 4 and 6 significant digits respectively.

In practical calculations, some numbers occur containing large number of digits, and it will be necessary to cut them to a usable number of figures. This process is called **rounding-off** of numbers. That is, in rounding process the number is replaced by another number consisting of a smaller number of digits. In that case, one or several digits keep with the number, taken from left to right, and discard all others.

The following rules of rounding-off are commonly used:

(i) *If the discarded digits constitute a number which is larger than half the unit in the last decimal place that remains, then the last digit that is left is increased by one.*

If the discarded digits constitute a number which is smaller than half the unit in the last decimal place that remains, then the digits that remain do not change.

(ii) *If the discarded digits constitute a number which is equal to half the unit in the last decimal place that remains, then the last digit that is half is increased by one, if it is odd, and is unchanged if it is even.*

This rule is often called a rule of an **even digit.** If a number is rounded using the above rule then the number is called correct up to some (say n) significant figures.

The following numbers are rounded-off correctly to *five* significant figures:

Exact number	Round-off number
$25.367\underline{8}35$	25.368
$28.353\underline{2}15$	28.353
$3.7853\underline{5}3$	3.7854
$5.8354\underline{5}3$	5.8355
$6.7354\underline{5}$	6.7354
$4.8327\underline{5}$	4.8328
$0.00583457\underline{7}8$	0.0058346
$38567\underline{5}4$	38568×10^2
2.37	2.3700
$8.9999\underline{7}$	9.0000
$9.9999\underline{8}$	10.000

From above examples, it is easy to observe that, while rounding a number, an error is generated and this error is sometimes called **round-off** error.

1.3 Absolute, Relative and Percentage Errors

Let x_T be the exact value of a number and x_A be its approximate value. If $x_A < x_T$, then we say that the number x_A is an approximate value of the number x_T by **defect** and if $x_A > x_T$, then it is an approximate value of x_T by **excess**.

The difference between the exact value x_T and its approximate value x_A is an **error**. As a rule, it is not possible to determine the value of the error $x_T - x_A$ and even its sign, since the exact number x_T is unknown.

The errors are represented in three ways, viz., absolute error, relative error and percentage error.

Absolute error:
The absolute error of the approximate number x_A is a quantity (Δx) which satisfies the inequality

$$\Delta x \geq |x_T - x_A|.$$

The absolute error is the upper bound of the deviation of the exact number x_T from its approximation, i.e.,

$$x_A - \Delta x \leq x_T \leq x_A + \Delta x.$$

The above result can be written in the form

$$x_T = x_A \pm \Delta x. \tag{1.1}$$

In other words, the absolute error of the number x is the difference between true value and approximate value, i.e.,

$$\Delta x = |x_T - x_A|.$$

It may be noted from the rounding process that, if a number be rounded to m decimal places then

$$\text{absolute error } \leq \frac{1}{2} \times 10^{-m}. \tag{1.2}$$

The absolute error measures only the **quantitative** aspect of the error but not the qualitative one, i.e., does not show whether the measurement and calculation were accurate. For example, the length and the width of a table are measured with a scale (whose division is 1 cm) and the following results are obtained: the width $w = 5 \pm 0.5$ cm and the length $l = 100 \pm 0.5$ cm. In both cases the absolute error is same and it is 0.5 cm. It is obvious that the second measurement was more accurate than the first. To estimate the quality of calculations or measurements, the concept of a relative error is introduced.

Relative error:
The relative error (δx) of the number x_A is

$$\delta x = \frac{\Delta x}{|x_A|} \text{ or } \frac{\Delta x}{|x_T|}, \quad |x_T| \neq 0 \text{ and } |x_A| \neq 0.$$

This expression can be written as

$$x_T = x_A(1 \pm \delta x) \text{ or } x_A = x_T(1 \pm \delta x).$$

Note that relative error is the absolute error when measuring 1 unit.

For the measurements of the length and the width of the table (discussed earlier) the relative errors are

$$\delta w = \frac{0.5}{5} = 0.1 \text{ and } \delta l = \frac{0.5}{100} = 0.005.$$

In these cases, one can conclude that the measurement of the length of the table has been relatively more accurate than that of its width. So one conclusion can be drawn: *the relative error measures the quantity and quality of the calculation and measurement. Thus, the relative error is a better measurement of error than absolute error.*

Percentage error:
The percentage error of an approximate number x_A is $\delta x \times 100\%$.

It is a particular type of relative error. This error is sometimes called **relative percentage error**. The percentage error gives the total error while measuring 100 unit instead of 1 unit. This error also calculates the quantity and quality of measurement. When relative error is very small then the percentage error is calculated.

Note 1.3.1 The absolute error of a number correct to n significant figures cannot be greater than half a unit in the nth place.

Note 1.3.2 The relative error and percentage error are independent of the unit of measurement, while absolute error depends on the measuring unit.

Difference between relative error and absolute error:

Absolute error measures only quantity of error and it is the total amount of error incurred by approximate value. While the relative error measures both the quantity and quality of the measurement. It is the total error while measuring one unit. The absolute error depends on the measuring unit, but, relative error does not depend on measuring unit.

Example 1.3.1 Find the absolute, relative and percentage error in x_A when $x_T = \frac{1}{3}$ and $x_A = 0.333$.

Solution. The absolute error

$$\begin{aligned}\Delta x = |x_T - x_A| &= \frac{1}{3} - 0.333 = \frac{1-0.999}{3}\\ &= \frac{0.001}{3} = 0.00033.\end{aligned}$$

The relative error

$$\delta x = \frac{\Delta x}{x_T} = \frac{0.00033}{1/3} = 0.00099 \simeq 0.001.$$

The percentage error is $\delta x \times 100\% = 0.00099 \times 100\% = 0.099\% \simeq 0.1\%$.

Example 1.3.2 An exact number x_T is in the interval [28.03, 28.08]. Assuming an approximate value, find the absolute and the percentage errors.

Solution. The middle of the given interval is taken as its approximate value, i.e., $x_A = 28.055$. The absolute error is half of its length, i.e., $\Delta x = 0.025$. The relative error $\delta x = \dfrac{\Delta x}{x_A} = 0.000891\cdots$.
It is conventional to round-off the error to one or two non-zero digits. Therefore, $\delta x = 0.0009$ and the percentage error is 0.09%.

Example 1.3.3 Determine the absolute error and the exact number corresponding to the approximate number $x_A = 5.373$ if percentage error is 0.01%.

Solution. Here the relative error $\delta x = 0.01\% = 0.0001$.
The absolute error $\Delta x = |x_A \times \delta x| = 5.373 \times 0.0001 = 0.0005373 \simeq 0.00054$.
The exact value $= 5.373 \pm 0.00054$.

Example 1.3.4 Find out in which of the following cases, the quality of calculations is better: $x_T = \dfrac{15}{17} \simeq 0.8824$ and $y_T = \sqrt{51} \simeq 7.141$.

Solution. To find the absolute error, we take the numbers x_A and y_A with a larger number of decimal digits as $x_A \simeq 0.882353, y_A = \sqrt{51} \simeq 7.141428$.
Therefore, the absolute error in x_T is $|0.882353 \cdots - 0.8824| \simeq 0.000047$,
and in y_T is $|7.141428 \cdots - 7.141| \simeq 0.00043$.
The relative error in x_A is $0.00047/0.8824 \simeq 0.00053 = 0.05\%$
and relative error in y_A is $0.00043/7.141 = 0.0000602 = 0.006\%$.

In the second case the quality of calculation is better than the first case as relative error in $x_T >$ relative error in y_T.

1.4 Valid Significant Digits

A real number can be represented by many different ways. For example, the number 840000 can be represented as two factors: 840×10^3 or 84.0×10^4 or 0.840×10^6. (Note that in these representations the last three significant zeros are lost). The later form of the notation is known as **normalize form** and it is commonly used. In this case, we say that 840 is the mantissa of the number and 6 is its **order**.

Every positive decimal number, exact as well as approximate, can be expressed as

$$a = d_1 \times 10^m + d_2 \times 10^{m-1} + \cdots + d_n \times 10^{m-n+1} + \cdots,$$

where d_i are the digits constituting the number $(i = 1, 2, \ldots)$ with $d_1 \neq 0$ and 10^{m-i+1} is the value of the ith decimal position (counting from left).

The digit d_n of the approximate number a is **valid significant digit** (or simply a **valid digit**) if it satisfies the following inequality.

$$\Delta a \leq 0.5 \times 10^{m-n+1}, \tag{1.3}$$

i.e., absolute error does not exceed half the unit of the decimal digit in which d_n appears.

If inequality (1.3) is not satisfied, then the digit d_n is said to be **doubtful**. It is obvious that if the digit d_n is valid, then all the preceding digits, to the left of it, are also valid.

Theorem 1.1 *If a number is correct up to n significant figures and the first significant digit of the number is k, then the relative error is less than*

$$\frac{1}{k \times 10^{n-1}}.$$

Proof. Let x_A be the approximate value of the exact number x_T. Also, let x_A is correct up to n significant figures and m decimal places. Then there are three possibilities may occur:
(i) $m < n$
(ii) $m = n$ and
(iii) $m > n$.

We have by (1.2), the absolute error $\Delta x \leq 0.5 \times 10^{-m}$.

Case I. When $m < n$.
In this case, the total number of digits in integral part is $n - m$. If k be the first significant digit in x_T, then

$$\Delta x \leq 0.5 \times 10^{-m} \text{ and } |x_T| \geq k \times 10^{n-m-1} - 0.5 \times 10^{-m}.$$

Therefore, the relative error

$$\begin{aligned}\delta x = \frac{\Delta x}{|x_T|} &\leq \frac{0.5 \times 10^{-m}}{k \times 10^{n-m-1} - 0.5 \times 10^{-m}} \\ &= \frac{1}{2k \times 10^{n-1} - 1}.\end{aligned}$$

Since, n is a positive integer and k is an integer lies between 1 and 9,

$$2k \times 10^{n-1} - 1 > k \times 10^{n-1}$$

for all k and n except $k = n = 1$.

Hence,

$$\delta x < \frac{1}{k \times 10^{n-1}}.$$

Case II. When $m = n$.
In this case, the first significant digit is the first digit after decimal point, i.e., the integral part is zero.

As before,

$$\begin{aligned}\delta x &= \frac{0.5 \times 10^{-m}}{k \times 10^{n-m-1} - 0.5 \times 10^{-m}} \\ &= \frac{1}{2k \times 10^{n-1} - 1} < \frac{1}{k \times 10^{n-1}}.\end{aligned}$$

Case III. When $m > n$.
In this case, the first significant digit k is at the $(n-m+1) = -(m-n-1)$th position. Also, the integer part is zero. Then $\Delta x \leq 0.5 \times 10^{-m}$ and $|x_T| \geq k \times 10^{-(m-n+1)} - 0.5 \times 10^{-m}$.

Therefore,
$$\begin{aligned}\delta x &= \frac{0.5 \times 10^{-m}}{k \times 10^{-(m-n+1)} - 0.5 \times 10^{-m}}\\ &= \frac{1}{2k \times 10^{n-1} - 1} < \frac{1}{k \times 10^{n-1}}.\end{aligned}$$

Hence the theorem.

1.5 Propagation of Errors in Arithmetic Operations

1.5.1 The errors in sum and difference

Consider the exact numbers $X_1, X_2, \ldots, X_n$ and their approximations be respectively $x_1, x_2, \ldots, x_n$. Let $\Delta x_1, \Delta x_2, \ldots, \Delta x_n$ be the errors in $x_1, x_2, \ldots, x_n$, i.e., $X_i = x_i \pm \Delta x_i, i = 1, 2, \ldots, n$. Also, let $X = X_1 + X_2 + \cdots + X_n$ and $x = x_1 + x_2 + \cdots + x_n$.

Therefore, the total absolute error is
$$\begin{aligned}|X - x| &= |(X_1 - x_1) + (X_2 - x_2) + \cdots + (X_n - x_n)|\\ &\leq |X_1 - x_1| + |X_2 - x_2| + \cdots + |X_n - x_n|.\end{aligned}$$

Thus the absolute error in the sum is
$$|\Delta x| = |\Delta x_1| + |\Delta x_2| + \cdots + |\Delta x_n|. \tag{1.4}$$

Thus the absolute error in sum of approximate numbers is equal to the sum of the absolute errors of the numbers.

From (1.4), it follows that the absolute error of the algebraic sum must not be smaller than the absolute error of the least exact term.

The following points should be kept in mind when adding numbers of different absolute accuracy.

(i) identify a number (or numbers) of the least accuracy (i.e., a number which has the maximum absolute error),

(ii) round-off more exact numbers so as to retain in them one digit more than in the identified number (i.e., retain one reserve digit),

(iii) perform addition taking into account all the retained digits,

(iv) round-off the result by discarding one digit.

Subtraction
Consider x_1 and x_2 be two approximate values of the corresponding exact numbers X_1 and X_2. Let $X = X_1 - X_2$ and $x = x_1 - x_2$.

Then $X_1 = x_1 \pm \Delta x_1$ and $X_2 = x_2 \pm \Delta x_2$, where Δx_1 and Δx_2 are the errors in x_1 and x_2 respectively.

Therefore, $|X - x| = |(X_1 - x_1) - (X_2 - x_2)| \le |X_1 - x_1| + |X_2 - x_2|$. Hence,

$$|\Delta x| = |\Delta x_1| + |\Delta x_2|. \tag{1.5}$$

Thus the absolute error in difference of two numbers is equal to the sum of individual absolute errors.

1.5.2 The error in product

Let us consider two exact numbers X_1 and X_2 and their approximate values x_1 and x_2. Also, let Δx_1 and Δx_2 be the errors in x_1 and x_2, i.e., $X_1 = x_1 \pm \Delta x_1$ and $X_2 = x_2 \pm \Delta x_2$.

Now, $X_1 X_2 = x_1 x_2 \pm x_1 \Delta x_2 \pm x_2 \Delta x_1 \pm \Delta x_1 \cdot \Delta x_2$.

Then $|X_1 X_2 - x_1 x_2| \le |x_1 \Delta x_2| + |x_2 \Delta x_1| + |\Delta x_1 \cdot \Delta x_2|$. The last term of right hand side is small, so we discard it and dividing both sides by $|x| = |x_1 x_2|$.

Thus the relative error in the product is

$$\left|\frac{X_1 X_2 - x_1 x_2}{x_1 x_2}\right| = \left|\frac{\Delta x_2}{x_2}\right| + \left|\frac{\Delta x_1}{x_1}\right|. \tag{1.6}$$

Thus the relative errors in product of two numbers is equal to the sum of individual relative errors.

The result (1.6) can be easily extended to the product of several numbers so that, if $X = X_1 X_2 \cdots X_n$ and $x = x_1 x_2 \cdots x_n$, then

$$\left|\frac{X - x}{x}\right| = \left|\frac{\Delta x_1}{x_1}\right| + \left|\frac{\Delta x_2}{x_2}\right| + \cdots + \left|\frac{\Delta x_n}{x_n}\right|. \tag{1.7}$$

That is, the total relative error in product of n numbers is equal to the sum of individual relative errors.

A particular case
Let the approximate numbers $x_1, x_2, \ldots, x_n$ be all positive and $x = x_1 x_2 \cdots x_n$.

Then $\log x = \log x_1 + \log x_2 + \cdots + \log x_n$.

Therefore, $\dfrac{\Delta x}{x} = \dfrac{\Delta x_1}{x_1} + \dfrac{\Delta x_2}{x_2} + \cdots + \dfrac{\Delta x_n}{x_n}$.

That is, $\left|\dfrac{\Delta x}{x}\right| = \left|\dfrac{\Delta x_1}{x_1}\right| + \left|\dfrac{\Delta x_2}{x_2}\right| + \cdots + \left|\dfrac{\Delta x_n}{x_n}\right|$.

Usually, the following steps are followed when multiplying two numbers:

(i) identify a number with the least number of valid digits,

(ii) round off the remaining factors so that they would contain one significant digit more than the valid significant digits there are in the isolated number,

(iii) retain as many significant digits in the product as there are valid significant digits in the least exact factor (the identified number).

Example 1.5.1 Show that when an approximate number x is multiplied by an exact factor k, the relative error of the product is equal to the relative error of the approximate number x and the absolute error is $|k|$ times the absolute error of the absolute number.

Solution. Let $x = kx_1$, where k is an exact factor other than zero. Then

$$\delta x = \left|\frac{\Delta x}{x}\right| = \left|\frac{k\ \Delta x_1}{k\ x_1}\right| = \left|\frac{\Delta x_1}{x_1}\right| = \delta x_1.$$

But, the absolute error $|\Delta x| = |k\ \Delta x_1| = |k|\ |\Delta x_1| = |k|$ times the absolute error in x_1.

1.5.3 The error in quotient

Let us consider two exact numbers X_1 and X_2 and their approximate values x_1 and x_2. Also, let $X = \dfrac{X_1}{X_2}$ and $x = \dfrac{x_1}{x_2}$.

Then $X_1 = x_1 + \Delta x_1, X_2 = x_2 + \Delta x_2$, where Δx_1 and Δx_2 are the errors.

Let $x_1 \neq 0$ and $x_2 \neq 0$.

Now,

$$X - x = \frac{x_1 + \Delta x_1}{x_2 + \Delta x_2} - \frac{x_1}{x_2} = \frac{x_2\,\Delta x_1 - x_1\,\Delta x_2}{x_2(x_2 + \Delta x_2)}.$$

Dividing both sides by x and taking absolute values:

$$\left|\frac{X - x}{x}\right| = \left|\frac{x_2\,\Delta x_1 - x_1\,\Delta x_2}{x_1(x_2 + \Delta x_2)}\right| = \left|\frac{x_2}{x_2 + \Delta x_2}\right| \left|\frac{\Delta x_1}{x_1} - \frac{\Delta x_2}{x_2}\right|.$$

The error Δx_2 is small as compared to x_2, then approximately $\dfrac{x_2}{x_2 + \Delta x_2} \simeq 1$. Therefore, above relation becomes

$$\delta x = \left|\frac{\Delta x}{x}\right| = \left|\frac{X - x}{x}\right| = \left|\frac{\Delta x_1}{x_1} - \frac{\Delta x_2}{x_2}\right| \leq \left|\frac{\Delta x_1}{x_1}\right| + \left|\frac{\Delta x_2}{x_2}\right|, \tag{1.8}$$

i.e., $\delta x = \delta x_1 + \delta x_2$. Hence, the total relative error in quotient is equal to the sum of their individual relative errors.

The relation (1.8) can also be written as

$$\left|\frac{\Delta x}{x}\right| = \left|\frac{\Delta x_1}{x_1} - \frac{\Delta x_2}{x_2}\right| \geq \left|\frac{\Delta x_1}{x_1}\right| - \left|\frac{\Delta x_2}{x_2}\right|. \tag{1.9}$$

From this relation one can conclude that the relative error in quotient is greater than or equal to the difference of their individual relative errors.

A particular case
For positive approximate numbers x_1 and x_2, the equation (1.8) can easily be deduced.
Let $x = x_1/x_2$. Then $\log x = \log x_1 - \log x_2$. Therefore,

$$\frac{\Delta x}{x} = \frac{\Delta x_1}{x_1} - \frac{\Delta x_2}{x_2} \text{ i.e., } \left|\frac{\Delta x}{x}\right| \leq \left|\frac{\Delta x_1}{x_1}\right| + \left|\frac{\Delta x_2}{x_2}\right|.$$

While dividing two numbers the following points should be followed.

(i) identify the least exact number, i.e., the number with the least number of valid digits,

(ii) round-off the other number, leaving in it on significant digit more than there are digits in the identified number,

(iii) retain as many significant digits in the quotient as there were in the least exact number.

Example 1.5.2 Find the sum of the approximate numbers 0.543, 0.1834, 17.45, 0.000234, 205.2, 8.35, 185.3, 0.0863, 0.684, 0.0881 in each of which all the written digits are valid. Find the absolute error in sum.

Solution. The least exact numbers (those possessing the maximum absolute error) are 205.2 and 185.3. The error of each of them is 0.05. Now, rounding off the other numbers, leaving one digit more and adding all of them.
0.54+0.18+17.45+0.00+205.2+8.35+185.3+0.09+0.68+0.09=417.88.
Discarding one digit by round-off the sum and we obtained 417.9.
The absolute error in the sum consists of two terms:

(i) the initial error, i.e., the sum of the errors of the least exact numbers and the rounding errors of the other numbers: $0.05 \times 2 + 0.0005 \times 8 = 0.104 \simeq 0.10$.

(ii) the error in rounding-off the sum is $417.9 - 417.88 = 0.02$.

Thus the absolute error of the sum is $0.10 + 0.02 = 0.12$.
So, the sum can be written as 417.9 ± 0.12.

Example 1.5.3 Find the difference of the approximate numbers 27.5 and 35.8 having absolute errors 0.02 and 0.03 respectively. Evaluate the absolute and the relative errors of the result.

Solution. Let $x_1 = 27.5$ and $x_2 = 35.8$. Then $x = x_1 - x_2 = -8.3$. The total absolute error $\Delta x = 0.02 + 0.03 = 0.05$.
Thus the difference $x_1 - x_2$ is -8.3 with absolute error 0.05.
The relative error is $0.05/|-8.3| \simeq 0.006 = 0.6\%$.

Example 1.5.4 Find the product of the approximate numbers $x_1 = 8.6$ and $x_2 = 34.359$ all of whose digits are valid. Also find the relative and the absolute errors.

Solution. In the first number, there are two valid significant digits and in the second there are five digits. Therefore, round-off the second number to three significant digits. After rounding-off the numbers x_1 and x_2 become $x_1 = 8.6$ and $x_2 = 34.4$.
Hence the product is

$$x = x_1 x_2 = 8.6 \times 34.4 = 295.84 \simeq 3.0 \times 10^2.$$

In the result, there are two significant digits, because the least number of valid significant digits of the given numbers is 2.
The relative error in product is

$$\delta x = \left|\frac{\Delta x}{x}\right| = \left|\frac{\Delta x_1}{x_1}\right| + \left|\frac{\Delta x_2}{x_2}\right| = \frac{0.05}{8.6} + \frac{0.0005}{34.359} = 0.00583 \simeq 0.58\%.$$

The absolute error is $(3.0 \times 10^2) \times 0.00583 = 1.749 \simeq 1.7$.

Example 1.5.5 Calculate the quotient x/y of the approximate numbers $x = 6.845$ and $y = 2.53$ if all the digits of the numbers are valid. Find the relative and the absolute errors.

Solution. Here the dividend $x = 6.845$ has four valid significant digits and the divisor has three, so we perform division without rounding-off. Thus

$$\frac{x}{y} = \frac{6.845}{2.53} = 2.71.$$

Three significant digits are retained in the result, since, the least exact number (the divisor y) contains three valid significant digits.
The absolute error in x and y are respectively

$$\Delta x = 0.0005 \text{ and } \Delta y = 0.005.$$

Therefore the relative error in quotient is

$$\left|\frac{\Delta x}{x}\right| + \left|\frac{\Delta y}{y}\right| = \frac{0.0005}{6.845} + \frac{0.005}{2.53} = 0.000073 + 0.00198$$
$$\simeq 0.002 = 0.2\%.$$

The absolute error is

$$\left|\frac{x}{y}\right| \times 0.002 = 2.71 \times 0.002 = 0.00542 = 0.005.$$

1.5.4 The errors in power and in root

Let us consider an approximate number x_1 which has a relative error δx_1. Now, the problem is to find the relative error of $x = x_1^m$.

Then

$$x = x_1^m = \underbrace{x_1 \cdot x_1 \cdots x_1}_{m \text{ times}}.$$

By (1.7), the relative error δx in the product is

$$\delta x = \underbrace{\delta x_1 + \delta x_1 + \cdots + \delta x_1}_{m \text{ times}} = m \ \delta x_1. \tag{1.10}$$

Thus, when the approximate number x is raised to the power m, its relative error increases m times.

Similarly, one can calculate the relative error of the number $x = \sqrt[m]{x_1}$.

Here $x_1 > 0$. Therefore,

$$\log x = \frac{1}{m} \log x_1.$$

That is,

$$\frac{\Delta x}{x} = \frac{1}{m}\frac{\Delta x_1}{x_1} \quad \text{or} \quad \left|\frac{\Delta x}{x}\right| = \frac{1}{m}\left|\frac{\Delta x_1}{x_1}\right|.$$

Hence the relative error is

$$\delta x = \frac{1}{m}\delta x_1,$$

where δx and δx_1 respectively represent the relative errors in x and x_1.

Example 1.5.6 Calculate $A = \dfrac{X^3\sqrt{Y}}{Z^2}$ where $X = 8.36, Y = 80.46, Z = 25.8$. The absolute errors in X, Y, Z are respectively 0.01, 0.02 and 0.03. Find the error of the result.

Solution. Here the absolute error $\Delta x = 0.01, \Delta y = 0.02$ and $\Delta z = 0.03$. To calculate intermediate result, retain one reserve digit. The approximate intermediate values are $x^3 = 584.3, \sqrt{y} = 8.9699, z^2 = 665.6$, where x, y, z are approximate values of X, Y, Z respectively.
Thus the approximate value of the expression is

$$a = \frac{584.3 \times 8.9699}{665.6} = 7.87.$$

Three significant digits are taken in the result, since, the least number of significant digits in the numbers is 3.
Now, the relative error δa in a is given by

$$\begin{aligned}\delta a &= 3\,\delta x + \frac{1}{2}\,\delta y + 2\,\delta z = 3 \times \frac{0.01}{8.36} + \frac{1}{2} \times \frac{0.02}{80.46} + 2 \times \frac{0.03}{25.8}\\ &\simeq 0.0036 + 0.00012 + 0.0023 \simeq 0.006 = 0.6\%.\end{aligned}$$

The absolute error Δa in a is $7.87 \times 0.006 = 0.047$.
Hence, $A = 7.87 \pm 0.047$ and the relative error is 0.006.

1.5.5 Error in evaluation of a function of several variables

Let $y = f(x_1, x_2, \ldots, x_n)$ be a differentiable function containing n variables $x_1, x_2, \ldots, x_n$ and let Δx_i be the error in x_i, for all $i = 1, 2, \ldots, n$.

Then the error Δy in y is given by

$$\begin{aligned}y + \Delta y &= f(x_1 + \Delta x_1, x_2 + \Delta x_2, \ldots, x_n + \Delta x_n)\\ &= f(x_1, x_2, \ldots, x_n) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\Delta x_i + \cdots\\ &\quad \text{(by Taylor's series expansion)}\\ &= y + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\Delta x_i\\ &\quad \text{(neglecting second and higher powers terms of } \Delta x_i\text{)}\\ \textit{i.e.,}\quad \Delta y &= \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\Delta x_i\end{aligned}$$

This formula gives the total error for computing a function containing several variables.

The relative error is given by

$$\frac{\Delta y}{y} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \frac{\Delta x_i}{y}.$$

1.6 Significant Error

Significant error occurs due to the loss of significant digits during arithmetic computation. This error occurs mainly due to the finite representation of the numbers in computational machine (computer or calculator). The loss of significant digits occurs due to the following two reasons:

(i) when two nearly equal numbers are subtracted and

(ii) when division is made by a very small divisor compared to the dividend.

Significant error is more serious than round-off error, which are illustrated in the following examples:

Example 1.6.1 Find the difference $X = \sqrt{5.36} - \sqrt{5.35}$ and evaluate the relative error of the result.

Solution. Let $X_1 = \sqrt{5.36} \simeq 2.315 = x_1$ and $X_2 = \sqrt{5.35} \simeq 2.313 = x_2$.
The absolute errors $\Delta x_1 = 0.0005$ and $\Delta x_2 = 0.0005$. Then the approximate difference is $x = 2.315 - 2.313 = 0.002$.
The total absolute error in the subtraction is $\Delta x = 0.0005 + 0.0005 = 0.001$.
The relative error $\delta x = \dfrac{0.001}{0.002} = 0.5 = 50\%$.
However, by changing the scheme of calculation we get a more accurate result.

$$\begin{aligned} X = \sqrt{5.36} - \sqrt{5.35} &= \frac{(\sqrt{5.36} - \sqrt{5.35})(\sqrt{5.36} + \sqrt{5.35})}{\sqrt{5.36} + \sqrt{5.35}} \\ &= \frac{5.36 - 5.35}{\sqrt{5.36} + \sqrt{5.35}} = \frac{0.01}{\sqrt{5.36} + \sqrt{5.35}} \simeq 0.002 = x \quad \text{(say)}. \end{aligned}$$

In this case the relative error is

$$\delta x = \frac{\Delta x_1 + \Delta x_2}{x_1 + x_2} = \frac{0.001}{2.315 + 2.313} = 0.0002 = 0.02\%.$$

Thus, when calculating x_1 and x_2 with the same four digits we get a better result in the sense of a relative error.

Example 1.6.2 Calculate the values of the function $y = 1 - \cos x$ at $x = 82^o$ and at $x = 1^o$. Also, calculate the absolute and the relative errors of the results.

Solution. $\underline{y \text{ at } x = 82^o}$
The value of $\cos 82^o \simeq 0.1392 = a_1$ (say) (correct up to four digits) and $\Delta a_1 = 0.00005$. Then $y_1 = 1 - 0.1392 = 0.8608$ and $\Delta y_1 = 0.00005$ (from an exact number equal to unity we subtract an approximate number with an absolute error not exceeding 0.00005).
Consequently, the relative error is

$$\delta y_1 = \frac{0.00005}{0.8608} = 0.000058 = 0.006\%.$$

$\underline{y \text{ at } x = 1^o}$
We have $\cos 1^o \simeq 0.9998 = a_2$ (say). $\Delta a_2 = 0.00005$.
$y_2 = 1 - 0.9998 = 0.0002$. $\Delta y_2 = 0.00005$.
Hence

$$\delta y_2 = \frac{0.00005}{0.0002} = 0.25 = 25\%.$$

From this example it is observed that for small values of x, a direct calculation of $y = 1 - \cos x$ gives a relative error of the order 25%. But at $x = 82^o$ the relative error is only 0.006%.
Now, change the calculation procedure and use the formula $y = 1 - \cos x = 2\sin^2 \frac{x}{2}$ to calculate the value of y for small values of x.
Let $a = \sin 0^o 30' \simeq 0.0087$. Then $\Delta a = 0.00005$ and

$$\delta a = \frac{0.00005}{0.0087} = 0.0058 = 0.58\%.$$

Thus $y_2 = 2 \times 0.0087^2 = 0.000151$ and relative error
$\delta y_2 = 0.0058 + 0.0058 = 0.012 = 1.2\%$ (using the formula $\delta a = \delta x + \delta y$ if $a = x.y$).
The absolute error is
$\Delta y_2 = y_2 \times \delta y_2 = 0.000151 \times 0.012 = 0.000002$.
Thus a simple transformation, of the computing formula, gives a more accurate result for the same data.

Example 1.6.3 Find the roots of the equation $x^2 - 1000x + 0.25 = 0$.

Solution. For simplicity, it is assumed that all the calculations are performed using four significant digits. The roots of this equation are

$$\frac{1000 \pm \sqrt{10^6 - 1}}{2}.$$

Now, $10^6 - 1 = 0.1000 \times 10^7 - 0.0000 \times 10^7 = 0.1000 \times 10^7$.
Thus $\sqrt{10^6 - 1} = 0.1000 \times 10^4$.
Therefore the roots are $\dfrac{0.1000 \times 10^4 \pm 0.1000 \times 10^4}{2}$

which are respectively 0.1000×10^4 and 0.0000×10^4. One of the roots becomes zero due to the finite representation of the numbers. But, the transformed formula gives the smaller root more accurately.
The smaller root of the equation may be calculated using the transformed formula

$$\begin{aligned}\frac{1000 - \sqrt{10^6 - 1}}{2} &= \frac{(1000 - \sqrt{10^6 - 1})(1000 + \sqrt{10^6 - 1})}{2(1000 + \sqrt{10^6 - 1})} \\ &= \frac{1}{2(1000 + \sqrt{10^6 - 1})} = 0.00025.\end{aligned}$$

Thus the roots of the given equation are 0.1000×10^4 and 0.00025.
Such a situation may be recognized by checking $|4ac| \ll b^2$.

It is not always possible to transform the computing formula. Therefore, when nearly equal numbers are subtracted, they must be taken with a sufficient number of reserve valid digits. If it is known that the first m significant digits may be lost during computation and if we need a result with n valid significant digits then the initial data should be taken with $m + n$ valid significant digits.

1.7 Representation of Numbers in Computer

Today, generally, numerical computations are carried out by calculator or computer. Due to the limitations of calculator, computers are widely used in numerical computation. In this book, computer is taken as the computational machine. Now, the representation and computation of numbers in computer are discussed below.

In computer, the numbers are stored mainly in two forms: (i) integer or fixed point form, and (ii) real or floating point form. Before storing to the computer, all the numbers are converted into binary numbers (consisting two bits 0 and 1) and then these converted numbers are stored into computer memory. Generally, two bytes (two bytes equal to 16 bits, one bit can store either 0 or 1) memory space is required to store an integer and four bytes space is required to store a floating point number. So, there is a limitation to store the numbers into computers.

Storing of integers is straight forward while representation of floating point numbers is different from our conventional technique. The main aim of this new technique is to preserve the maximum number of significant digits in a real number and also increase the range of values of the real numbers. This representation is called the **normalized**

floating point mode. In this mode of representation, the whole number is converted to a proper fraction in such a way that the first digit after decimal point should be non-zero and is adjusted by multiplying a number which is some powers of 10. For example, the number 375.3×10^4 is represented in this mode as $.3753 \times 10^7 = .3753E7$ (E7 is used to represent 10^7). From this example, it is observed that in normalized floating point representation, a number is a combination of two parts – **mantissa** and **exponent**. In the above example, .3753 is the mantissa and 7 is the exponent. It may be noted that the mantissa is always greater than or equal to .1 and exponent is an integer.

For simplicity, it is assume that the computer (hypothetical) uses *four* digits to store mantissa and *two* digits for exponent. The mantissa and the exponent have their own signs.

The number .0003783 would be stored as .3783E–3. The leading zeros in this number serve only to indicate the decimal point. Thus, in this notation the range of numbers (magnitudes) is $.9999 \times 10^{99}$ to $.1000 \times 10^{-99}$.

1.8 Arithmetic of Normalized Floating Point Numbers

In this section, the arithmetic operations on normalized floating point numbers are discussed.

1.8.1 Addition

If two numbers have same exponent, then the mantissas are added directly and the exponents are adjusted, if required.

If the exponents are different then lower exponent is shifted to higher exponent by adjusting mantissa. The details about addition are discussed in the following examples.

Example 1.8.1 Add the following normalized floating point numbers.
(i) .3456E3 and .4325E3 (same exponent)
(ii) .8536E5 and .7381E5
(iii) .3758E5 and .7811E7 (different exponent)
(iv) .2538E2 and .3514E7
(v) .7356E99 and .3718E99 (overflow condition)

Solution. (i) In this case, the exponents are equal, so the mantissa are added directly. Thus the sum is .7781E3.

(ii) In this case, the exponent are equal and the sum is 1.5917E5. Here the mantissa has 5 significant digits, but our computer (hypothetical) can store only four significant figures. So, the number is shifted right one place before it is stored. The exponent is increased by 1 and the last digit is truncated. The final result is .1591E6.

(iii) Here, the numbers are .3758E5 and .7811E7. The exponent of the first number is less than that of the second number. The difference of the exponents is $7 - 5 = 2$. So the mantissa of the smaller number (here first number) is shifted right by 2 places (the difference of the exponents) and the last 2 digits of the mantissa are discarded as our hypothetical computer can store only 4 digits. Then the first number becomes .0037E7. Then the result is .0037E7 + .7811E7 = .7848E7.

(iv) Here also the exponents are different and the difference is $7 - 2 = 5$. The mantissa of first number (smaller exponent) is shifted 5 places and the number becomes .0000E7. The final result is .0000E7 + .3514E7 = .3514E7.

(v) Here the numbers are .7356E99 and .3718E99 and they have equal exponent. So the sum of them is 1.1074E99. In this case mantissa has five significant digits. Thus the mantissa is shifted right and the exponent is increased by 1. Then the exponent becomes 100. As the exponent cannot store more than two digits, in our hypothetical computer, the number is larger than the largest number that can be stored in our computer. This situation is called an **overflow** condition and the machine will give an error message.

1.8.2 Subtraction

The subtraction is same as addition. In subtraction one positive number and one negative number are added. The following example shows the details about subtraction.

Example 1.8.2 Subtract the normalized floating point numbers indicated below:
(i) .3628E6 from .8321E6
(ii) .3885E5 from .3892E5
(iii) .3253E–7 from .4123E–6
(iv) .5321E–99 from .5382E–99.

Solution. (i) Here the exponents are equal, and the result is
.8321E6 – .3628E6 = .4693E6.

(ii) Here the result is .3892E5 – .3885E5 = .0007E5. The most significant digit in the mantissa is 0, so the mantissa is shifted left till the most significant digit becomes non-zero and in each left shift of the mantissa the exponent is reduced by 1. Hence the final result is .7000E2.

(iii) The numbers are .4123E–6 and .3253E–7. The exponents are not equal, so the number with smaller exponent is shifted right and the exponent increased by 1 for every right shift. Then the second number becomes .0325E–6. Thus the result is .4123E–6 – .0325E–6 = .3798E–6.

(iv) The result is .5382E–99 – .5321E–99 = .0061E–99. For normalization, the mantissa is shifted left twice and in this process the exponent is reduced by 1. In first shift, the exponent becomes –100, but our hypothetical computer can store only two digits as exponent. So –100 cannot be accommodated in the exponent part of the number. In this case, the result is smaller than the smallest number which could be stored in our computer. This condition is called an **underflow** condition and the computer will give an error message.

1.8.3 Multiplication

Two numbers in normalized floating point mode are multiplied by multiplying the mantissa and adding the exponents. After multiplication, the mantissa is converted into normalized floating point form and the exponent is converted appropriately. The following example shows the steps of multiplication.

Example 1.8.3 Multiply the following numbers indicated below:
(i) .5321E5 by .4387E10
(ii) .1234E10 by .8374E–10
(iii) .1139E50 by .8502E51
(iv) .3721E–52 by .3205E-53.

Solution. (i) Here, .5321E5 × .4387E10 = $.2334\underbrace{3227}_{\text{discarded}}\text{E}15$.

The mantissa has 8 significant figures, so the last four digits are discarded. The final result is .2334E15.

(ii) Here, .1234E10 × .8374E–10 = $.1033\underbrace{3516}_{\text{discarded}}\text{E}0$ = .1033E0.

(iii) .1139E50 × .8502E51 = .09683778E101.
Here, the mantissa has one 0 as most significant digit, so the mantissa is shifted left one digit and the exponent is adjusted. The product is .9683E100. But our hypothetical computer cannot store 3 digits as exponent. Hence, in this case, the overflow condition occurs.

(iv) .3721E–52 × .3205E–53 = .11925805E–105 = .1192E–105.
In this case, the product is very small (as the exponent is –105). Hence the underflow condition occurs.

1.8.4 Division

In the division, the mantissa of the numerator is divided by that of the denominator. The exponent is obtained by subtracting exponent of denominator from the exponent of numerator. The quotient mantissa is converted to normalized form and the exponent is adjusted appropriately.

Example 1.8.4 Perform the following divisions
(i) .9938E5 ÷ .3281E2
(ii) .9999E2 ÷ .1230E–99
(iii) .3568E–10 ÷ .3456E97.

Solution. (i) .9938E5 ÷ .3281E2 = .3028E4.

(ii) .9999E2 ÷ .1230E–99 = .8129E102.
The result overflows.

(iii) .3568E–10 ÷ .3456E97 = .1032E–106.
In this case the result underflows.

1.9 Effect of Normalized Floating Point Representations

The truncation of mantissa leads to very interesting results. For example, $\frac{1}{6} \times 12 = 2$ is well known. But, when the arithmetic is performed with floating point numbers, .1667 being added 12 times yields .1996E1, whereas, .1667 × 12 gives .2000E1. That is, $12x = \underbrace{x + x + \cdots + x}_{\text{12 times}}$ is not true.

It is very surprising that due to the truncation of mantissa, the associative and distributive laws do not hold always in normalized floating point numbers.

That is,
(i) $(a + b) + c \neq a + (b + c)$
(ii) $(a + b) - c \neq (a - c) + b$
(iii) $a(b - c) \neq ab - ac$.

These results are illustrated in the following examples:

(i) $a =$.6878E1, $b =$.7898E1 and $c =$.1007E1.
Now, $a + b =$.1477E2
$(a + b) + c =$ .1477E2 + .1007E1 = .1477E2 + .0100E2 = .1577E2.
Again, $b + c =$.8905E1.
$a + (b + c)$=.6878E1+.8905E1=.1578E2.
Thus, $(a + b) + c \neq a + (b + c)$.

(ii) Let $a =$.6573E1, $b =$.5857E–1, $c =$.6558E1.
Then $a + b =$.6631E1 and $(a + b) - c =$.6631E1 − .6558E1 = .7300E-1.
Again, $a - c =$.1500E–1 and $(a - c) + b =$.1500E–1 + .5857E–1 = .7357E–1.
Thus, $(a + b) - c \neq (a - c) + b$.

(iii) Let $a =$.5673E1, $b =$.3583E1, $c =$.3572E1.

$b - c =$.1100E–1.
$a(b - c) =$.5673E1 × .1100E–1 = .0624E0 = .6240E–1.
$ab =$.2032E2, $ac =$.2026E2.
$ab - ac =$.6000E–1.
Thus, $a(b - c) \neq ab - ac$.

The above examples are intentionally chosen to point out the occurrence of inaccuracies in normalized floating point arithmetic due to the shifting and truncation of numbers during arithmetic operations. But these situations always do not happen. Here, we assume that the computer can store only four digits in mantissa, but actually computer can store **seven** digits as mantissa (in single precision). The larger length of mantissa gives more accurate result.

1.9.1 Zeros in floating point numbers

The number zero has a definite meaning in mathematics, but, in computer exact equality of a number to zero can never be guaranteed. The cause behind this situation is that most of the numbers in floating point representation are approximate. One interesting example is presented below to illustrate the behaviour of zero.

The roots of the quadratic equation $x^2 + 2x - 5 = 0$ are $x = -1 \pm \sqrt{6}$.

The roots in floating point representation (4 digits mantissa) are .1449E1 and –.3449E1.

But, at $x =$.1449E1 the left hand side of the equation is –.003 clearly which is not equal to zero, while at $x =$–.3449E1, the left hand side of the equation is
(–.3449E1) × (–.3449E1) + .2000E1 × (–.3449E1) – .5000E1
= .1189E2 – .6898E1 – .5000E1 = .1189E2 – .0689E2 – .0500E2 = .0000E2, which is equal to 0.

Thus, one root satisfies the equation completely but other root does not, though they are roots of the equation. By the property of the root of an equation the number 0.003 be zero. Depending on the result of this example we may note the following.

Note 1.9.1 In any computational algorithm, it is not advisable to give any instruction based on testing whether a floating point number is zero or not.

1.10 Exercise

1. What do you mean by the terms in numerical analysis ?
   (i) truncation error, (ii) round-off error, (iii) significant error.

2. What are the different sources of computational errors in a numerical computational work ?

3. Explain what do you understand by an approximate number and significant figures of a number.

4. What convention are used in rounding-off a number ?

5. When a number is said to be correct to n significant figures ?
Round-off the following numbers to three significant figures.
(i) 0.01302, (ii) –349.87, (iii) 0.005922, (iv) 87678, (v) 64.8523, (vi) 6380.7, (vii) 0.0000098, (viii) .2345, (ix) 0.4575, (x) 34.653, (xi) 21.752, (xii) 1.99999.

6. Define absolute, relative and percentage errors.

7. Explain when relative error is a better indicator of the accuracy of a computation than the absolute error.

8. Find out which of the following two equalities is more exact:
(i) $6/25 \simeq 1.4$ or $1/3 \simeq 0.333$, (ii) $1/9 \simeq 0.1$ or $1/3 \simeq 0.33$, (iii) $\pi \simeq 3.142$ or $\sqrt{10} \simeq 3.1623$.

9. Find the absolute, relative and percentage errors when (i) 2/3 is approximated to 0.667, (ii) 1/3 is approximated to 0.333, and (iii) true value is 0.50 and its calculated value was 0.49.

10. (i) If π is approximated as 3.14 instead of 3.14156, find the absolute, relative and percentage errors.
(ii) Round-off the number $x = 3.4516$ to three significant figures and find the absolute and the relative errors.

11. The numbers 23.982 and 3.4687 are both approximate and correct only to their last digits. Find their difference and state how many figures in the result are trustworthy.

12. Two lengths X and Y are measured approximately up to three significant figures as $X = 3.32$ cm and $Y = 5.39$ cm. Estimate the error in the computed value of $X + Y$.

13. Let x_T and x_A denote respectively the true and approximate values of a number. Prove that the relative error in the product $x_A y_A$ is approximately equal to the sum of the relative errors in x_A and y_A.

14. Show that the relative error in the product of several approximate non-zero numbers does not exceed the sum of the relative errors of the numbers.

15. Show that the maximum relative error in the quotient of two approximate numbers is approximately equal to the algebraic sum of the maximum relative errors of the individual numbers.

16. Let $x = 5.234 \pm 0.0005$ and $y = 5.123 \pm 0.0005$. Find the percentage error of the difference $a = x - y$ when relative errors $\delta_x = \delta_y = 0.0001$.

17. What do you mean by the statement that x_A (approximate value) has m significant figures with respect to x_T (true value) ? If the first significant figure of x_A is k and x_A is correct up to n significant figures, prove that the relative error is less than $10^{1-n}/k$.

18. Given $a = 11 \pm 0.5$, $b = 0.04562 \pm 0.0001$, $c = 17200 \pm 100$. Find the maximum value of the absolute error in the following expressions
(i) $a + 2b - c$, (ii) $2a - 5b + c$ and (iii) a^2.

19. Calculate the quotient $a = x/y$ of the approximate numbers $x = 5.762$ and $y = 1.24$ if all the digits of the dividend and the divisor are valid. Find the relative and the absolute errors.

20. (i) Establish the general formula for absolute and relative errors for the function $v = f(u_1, u_2, \ldots, u_n)$ when absolute errors Δu_i of each independent quantity u_i are known. Use this result for the function $v = \dfrac{u_1^p u_2^q u_3^r}{u_4^s u_5^t}$ to find the upper bound of the relative error.
(ii) Find the relative error in computing $f(x) = 2x^5 - 3x + 2$ at $x = 1$, if the error in x is 0.005.
(iii) If $y = \dfrac{1.42x + 3.45}{x + 0.75}$ here the coefficients are rounded-off, find the absolute and relative errors in y when $x = 0.5 \pm 0.1$.

21. Given $y = x^4 y^{5/2}$, if x_0, y_0 be the approximate values of x, y respectively and $\Delta x_0, \Delta y_0$ be the absolute errors in them, determine the relative error in u.

22. Calculate $x = \dfrac{(a+b)c}{(d-e)^2}$, where $a = 1.562 \pm 0.001$, $b = 10.3 \pm 0.02, c = 0.12 \pm 0.04, d = 10.541 \pm 0.004, e = 2.34 \pm 0.006$. Find the absolute and the relative errors in the result.

23. (i) Determine the number of correct digits in the number 0.2318 if the relative error is 0.3×10^{-1}.
(ii) Find the number of significant figures in the approximate number 0.4785 given that the relative error is 0.2×10^{-2}.

24. Find the smaller root of the equation $x^2 - 500x + 1 = 0$ using four-digit arithmetic.

25. Find the value of $\sqrt{103} - \sqrt{102}$ correct up to four significant figures.

26. Find an example where in an approximate computation
(i) $(a + b) + c \neq a + (b + c)$, (ii) $(a + b) - c \neq (a - c) + b$, (iii) $a(b - c) \neq ab - ac$.

Chapter 2

Calculus of Finite Differences and Difference Equations

Let us consider a function $y = f(x)$ defined on $[a, b]$. The variables x and y are called independent and dependent variables respectively. The points $x_0, x_1, \ldots, x_n$ are taken as equidistance, i.e., $x_i = x_0 + ih$, $i = 0, 1, 2, \ldots, n$. Then the value of y, when $x = x_i$, is denoted by y_i, where $y_i = f(x_i)$. The values of x are called **arguments** and that of y are called **entries**. The interval h is called the difference interval. In this chapter, some important difference operators, viz., forward difference (Δ), backward difference (∇), central difference (δ), shift (E) and mean (μ) are introduced.

2.1 Finite Difference Operators

2.1.1 Forward differences

The **forward difference** or simply **difference** operator is denoted by Δ and is defined by

$$\Delta f(x) = f(x+h) - f(x). \tag{2.1}$$

In terms of y, at $x = x_i$ the above equation gives

$$\Delta f(x_i) = f(x_i + h) - f(x_i), \text{ i.e., } \Delta y_i = y_{i+1} - y_i, i = 0, 1, 2, \ldots, n-1. \tag{2.2}$$

Explicitly, $\Delta y_0 = y_1 - y_0, \Delta y_1 = y_2 - y_1, \ldots, \Delta y_{n-1} = y_n - y_{n-1}$.

The differences of the first differences are called second differences and they are denoted by $\Delta^2 y_0, \Delta^2 y_1, \ldots$. Similarly, one can define third differences, fourth differences, etc.

Thus,

$$\begin{aligned}
\Delta^2 y_0 &= \Delta y_1 - \Delta y_0 = (y_2 - y_1) - (y_1 - y_0) = y_2 - 2y_1 + y_0 \\
\Delta^2 y_1 &= \Delta y_2 - \Delta y_1 = (y_3 - y_2) - (y_2 - y_1) = y_3 - 2y_2 + y_1 \\
\Delta^3 y_0 &= \Delta^2 y_1 - \Delta^2 y_0 = (y_3 - 2y_2 + y_1) - (y_2 - 2y_1 + y_0) = y_3 - 3y_2 + 3y_1 - y_0 \\
\Delta^3 y_1 &= y_4 - 3y_3 + 3y_2 - y_1
\end{aligned}$$

and so on.

In general,

$$\Delta^{n+1} f(x) = \Delta[\Delta^n f(x)], \text{ i.e., } \Delta^{n+1} y_i = \Delta[\Delta^n y_i], n = 0, 1, 2, \ldots. \tag{2.3}$$

Also, $\Delta^{n+1} f(x) = \Delta^n [f(x+h) - f(x)] = \Delta^n f(x+h) - \Delta^n f(x)$
and

$$\Delta^{n+1} y_i = \Delta^n y_{i+1} - \Delta^n y_i, n = 0, 1, 2, \ldots, \tag{2.4}$$

where $\Delta^0 \equiv$ identity operator, i.e., $\Delta^0 f(x) = f(x)$ and $\Delta^1 \equiv \Delta$.

The different forward differences for the arguments $x_0, x_1, \ldots, x_4$ are shown in Table 2.1.

Table 2.1: Forward difference table.

x	y	Δ	Δ^2	Δ^3	Δ^4
x_0	y_0				
		Δy_0			
x_1	y_1		$\Delta^2 y_0$		
		Δy_1		$\Delta^3 y_0$	
x_2	y_2		$\Delta^2 y_1$		$\Delta^4 y_0$
		Δy_2		$\Delta^3 y_1$	
x_3	y_3		$\Delta^2 y_2$		
		Δy_3			
x_4	y_4				

Table 2.1 is called **forward difference** or **diagonal difference** table.

2.1.2 Backward differences

The backward difference operator is denoted by ∇ and it is defined as

$$\nabla f(x) = f(x) - f(x-h). \tag{2.5}$$

In terms of y, the above relation transforms to

$$\nabla y_i = y_i - y_{i-1}, \qquad i = n, n-1, \ldots, 1. \tag{2.6}$$

That is,

$$\nabla y_1 = y_1 - y_0, \nabla y_2 = y_2 - y_1, \ldots, \nabla y_n = y_n - y_{n-1}. \tag{2.7}$$

These differences are called first differences. The second differences are denoted by $\nabla^2 y_2, \nabla^2 y_3, \ldots, \nabla^2 y_n$. That is,

$$\nabla^2 y_2 = \nabla(\nabla y_2) = \nabla(y_2 - y_1) = \nabla y_2 - \nabla y_1 = (y_2 - y_1) - (y_1 - y_0) = y_2 - 2y_1 + y_0.$$

Similarly, $\nabla^2 y_3 = y_3 - 2y_2 + y_1, \nabla^2 y_4 = y_4 - 2y_3 + y_2$, and so on.
In general,

$$\nabla^k y_i = \nabla^{k-1} y_i - \nabla^{k-1} y_{i-1}, \qquad i = n, n-1, \ldots, k, \tag{2.8}$$

where $\nabla^0 y_i = y_i, \nabla^1 y_i = \nabla y_i$.

These backward differences can be written in a tabular form and this table is known as **backward difference** or **horizontal** table.

Table 2.2 is the backward difference table for the arguments $x_0, x_1, \ldots, x_4$.

Table 2.2: Backward difference table.

x	y	∇	∇^2	∇^3	∇^4
x_0	y_0				
x_1	y_1	∇y_1			
x_2	y_2	∇y_2	$\nabla^2 y_2$		
x_3	y_3	∇y_3	$\nabla^2 y_3$	$\nabla^3 y_3$	
x_4	y_4	∇y_4	$\nabla^2 y_4$	$\nabla^3 y_4$	$\nabla^4 y_4$

2.1.3 Central differences

The **central difference** operator is denoted by δ and is defined by

$$\delta f(x) = f(x + h/2) - f(x - h/2). \tag{2.9}$$

In terms of y, the first central difference is

$$\delta y_i = y_{i+1/2} - y_{i-1/2} \tag{2.10}$$

where $y_{i+1/2} = f(x_i + h/2)$ and $y_{i-1/2} = f(x_i - h/2)$.

Thus $\delta y_{1/2} = y_1 - y_0, \delta y_{3/2} = y_2 - y_1, \ldots, \delta y_{n-1/2} = y_n - y_{n-1}$.

The second central differences are

$\delta^2 y_i = \delta y_{i+1/2} - \delta y_{i-1/2} = (y_{i+1} - y_i) - (y_i - y_{i-1}) = y_{i+1} - 2y_i + y_{i-1}$.

In general,

$$\delta^n y_i = \delta^{n-1} y_{i+1/2} - \delta^{n-1} y_{i-1/2}. \tag{2.11}$$

The central difference table for the five arguments $x_0, x_1, \ldots, x_4$ is shown in Table 2.3.

Table 2.3: Central difference table.

x	y	δ	δ^2	δ^3	δ^4
x_0	y_0				
		$\delta y_{1/2}$			
x_1	y_1		$\delta^2 y_1$		
		$\delta y_{3/2}$		$\delta^3 y_{3/2}$	
x_2	y_2		$\delta^2 y_2$		$\delta^4 y_2$
		$\delta y_{5/2}$		$\delta^3 y_{5/2}$	
x_3	y_3		$\delta^2 y_3$		
		$\delta y_{7/2}$			
x_4	y_4				

It is observed that all odd differences have fraction suffices and all the even differences are with integral suffices.

2.1.4 Shift, Average and Differential operators

Shift operator, E:

The shift operator is defined by

$$Ef(x) = f(x + h). \tag{2.12}$$

This gives,

$$Ey_i = y_{i+1}. \tag{2.13}$$

That is, shift operator shifts the function value y_i to the next higher value y_{i+1}.

The second shift operator gives

$$E^2 f(x) = E[Ef(x)] = E[f(x + h)] = f(x + 2h). \tag{2.14}$$

In general,

$$E^n f(x) = f(x+nh) \text{ or } E^n y_i = y_{i+nh}. \tag{2.15}$$

The inverse shift operator E^{-1} is defined as

$$E^{-1} f(x) = f(x-h). \tag{2.16}$$

Similarly, second and higher inverse operators are

$$E^{-2} f(x) = f(x-2h) \qquad \text{and} \qquad E^{-n} f(x) = f(x-nh). \tag{2.17}$$

More general form of E operator is

$$E^r f(x) = f(x+rh), \tag{2.18}$$

where r is positive as well as negative rationals.

Average operator, μ:

The average operator μ is defined as

$$\begin{aligned} \mu f(x) &= \frac{1}{2}\big[f(x+h/2) + f(x-h/2)\big] \\ \text{i.e.,} \qquad \mu y_i &= \frac{1}{2}\big[y_{i+1/2} + y_{i-1/2}\big]. \end{aligned} \tag{2.19}$$

Differential operator, D:

The differential operator is usually denoted by D, where

$$\begin{aligned} Df(x) &= \frac{d}{dx} f(x) = f'(x) \\ D^2 f(x) &= \frac{d^2}{dx^2} f(x) = f''(x). \end{aligned} \tag{2.20}$$

2.1.5 Factorial notation

The factorial notation has many uses in calculus of finite difference. This is used to find different differences and anti-differences. The nth factorial of x, denoted by $x^{(n)}$, is defined by

$$x^{(n)} = x(x-h)(x-2h)\cdots(x-\overline{n-1}h), \tag{2.21}$$

where, each factor is decreased from the earlier by h; and $x^{(0)} = 1$.

Similarly, the nth negative factorial of x is defined by

$$x^{(-n)} = \frac{1}{x(x+h)(x+2h)\cdots(x+\overline{n-1}h)}. \tag{2.22}$$

It may be noted that $x^{(n)}.x^{(-n)} \neq 1$.

2.2 Properties of Forward Differences

Property 2.2.1 $\Delta c = 0$, *where c is a constant.*

Property 2.2.2 $\Delta[f_1(x) + f_2(x) + \cdots + f_n(x)]$
$= \Delta f_1(x) + \Delta f_2(x) + \cdots + \Delta f_n(x)$.

Property 2.2.3 $\Delta[cf(x)] = c\Delta f(x)$.

Combining properties (2.2.2) and (2.2.3), one can generalise the property (2.2.2) as

Property 2.2.4 $\Delta[c_1 f_1(x) + c_2 f_2(x) + \cdots + c_n f_n(x)]$
$= c_1\Delta f_1(x) + c_2\Delta f_2(x) + \cdots + c_n\Delta f_n(x)$.

Property 2.2.5 $\Delta^m\Delta^n f(x) = \Delta^{m+n} f(x) = \Delta^n\Delta^m f(x) = \Delta^k\Delta^{m+n-k} f(x)$, $k = 0, 1, 2, \ldots, m$ *or* n.

Property 2.2.6

$$\begin{aligned}
\Delta[f(x)g(x)] &= f(x+h)g(x+h) - f(x)g(x) \\
&= f(x+h)g(x+h) - f(x+h)g(x) + f(x+h)g(x) - f(x)g(x) \\
&= f(x+h)[g(x+h) - g(x)] + g(x)[f(x+h) - f(x)] \\
&= f(x+h)\Delta g(x) + g(x)\Delta f(x).
\end{aligned}$$

Also, it can be shown that

$$\begin{aligned}
\Delta[f(x)g(x)] &= f(x)\Delta g(x) + g(x+h)\Delta f(x) \\
&= f(x)\Delta g(x) + g(x)\Delta f(x) + \Delta f(x)\Delta g(x).
\end{aligned}$$

Property 2.2.7 $\Delta\left[\dfrac{f(x)}{g(x)}\right] = \dfrac{g(x)\Delta f(x) - f(x)\Delta g(x)}{g(x+h)g(x)}, g(x) \neq 0.$

Proof.

$$\begin{aligned}
\Delta\left[\frac{f(x)}{g(x)}\right] &= \frac{f(x+h)}{g(x+h)} - \frac{f(x)}{g(x)} \\
&= \frac{f(x+h)g(x) - g(x+h)f(x)}{g(x+h)g(x)} \\
&= \frac{g(x)[f(x+h) - f(x)] - f(x)[g(x+h) - g(x)]}{g(x+h)g(x)} \\
&= \frac{g(x)\Delta f(x) - f(x)\Delta g(x)}{g(x+h)g(x)}.
\end{aligned}$$

Property 2.2.8 *In particular, when the numerator is 1, then*

$$\Delta\left[\frac{1}{f(x)}\right] = -\frac{\Delta f(x)}{f(x+h)f(x)}.$$

Property 2.2.9 $\Delta[c^x] = c^{x+h} - c^x = c^x(c^h - 1)$, *for some constant c.*

Property 2.2.10 $\Delta[{}^xC_r] = {}^xC_{r-1}$, *where r is fixed and* $h = 1$.
Proof. $\Delta[{}^xC_r] = {}^{x+1}C_r - {}^xC_r = {}^xC_{r-1}$ *as* $h = 1$.

Property 2.2.11 $\Delta x^{(n)} = nhx^{(n-1)}$.
Proof.

$$\begin{aligned}\Delta x^{(n)} &= (x+h)(x+h-h)(x+h-2h)\cdots(x+h-\overline{n-1}h)\\ &\quad -x(x-h)(x-2h)\cdots(x-\overline{n-1}h)\\ &= x(x-h)(x-2h)\cdots(x-\overline{n-2}h)[x+h-\{x-(n-1)h\}]\\ &= nhx^{(n-1)}.\end{aligned}$$

This property is analogous to the differential formula $D(x^n) = nx^{n-1}$ when $h = 1$.
Most of the above formulae are similar to the corresponding formulae in differential calculus.

Property 2.2.12 *The above formula can also be used to find anti-difference (like integration in integral calculus), as*

$$\Delta^{-1}x^{(n-1)} = \frac{1}{nh}x^{(n)}. \tag{2.23}$$

2.2.1 Properties of shift operators

Property 2.2.13 $Ec = c$, *where c is a constant.*

Property 2.2.14 $E\{cf(x)\} = cEf(x)$.

Property 2.2.15 $E\{c_1f_1(x) + c_2f_2(x) + \cdots + c_nf_n(x)]$
$= c_1Ef_1(x) + c_2Ef_2(x) + \cdots + c_nEf_n(x)$.

Property 2.2.16 $E^mE^nf(x) = E^nE^mf(x) = E^{m+n}f(x)$.

Property 2.2.17 $E^nE^{-n}f(x) = f(x)$.
In particular, $EE^{-1} \equiv I, I$ *is the identity operator and it is some times denoted by 1.*

Property 2.2.18 $(E^n)^mf(x) = E^{mn}f(x)$.

Property 2.2.19 $E\left\{\dfrac{f(x)}{g(x)}\right\} = \dfrac{Ef(x)}{Eg(x)}$.

Property 2.2.20 $E\{f(x)\, g(x)\} = Ef(x)\, Eg(x)$.

Property 2.2.21 $E\Delta f(x) = \Delta E f(x)$.

Property 2.2.22 $\Delta^m f(x) = \nabla^m E^m f(x) = E^m \nabla^m f(x)$
and $\nabla^m f(x) = \Delta^m E^{-m} f(x) = E^{-m}\Delta^m f(x)$.

2.3 Relations Among Operators

It is clear from the forward, backward and central difference tables that in a definite numerical case, the same values occur in the same positions, practically there are no differences among the values of the tables, but, different symbols have been used for the theoretical importance.

Thus

$$\begin{aligned}\Delta y_i &= y_{i+1} - y_i = \nabla y_{i+1} = \delta y_{i+1/2}\\ \Delta^2 y_i &= y_{i+2} - 2y_{i+1} + y_i = \nabla^2 y_{i+2} = \delta^2 y_{i+1}\end{aligned}$$

etc.

In general,

$$\Delta^n y_i = \nabla^n y_{i+n}, \qquad i = 0, 1, 2, \ldots. \tag{2.24}$$

Again,

$$\Delta f(x) = f(x+h) - f(x) = Ef(x) - f(x) = (E-1)f(x).$$

This relation indicates that the effect of the operator Δ on $f(x)$ is the same as that of the operator $E - 1$ on $f(x)$. Thus

$$\Delta \equiv E - 1 \qquad \text{or} \qquad E \equiv \Delta + 1. \tag{2.25}$$

Also,

$$\nabla f(x) = f(x) - f(x-h) = f(x) - E^{-1}f(x) = (1 - E^{-1})f(x).$$

That is,

$$\nabla \equiv 1 - E^{-1}. \tag{2.26}$$

The higher order forward difference can be expressed in terms of the given function values in the following way:

$$\Delta^3 y_i = (E-1)^3 y_i = (E^3 - 3E^2 + 3E - 1)y_i = y_3 - 3y_2 + 3y_1 - y_0.$$

There is a relation among the central difference, δ, and the shift operator E, as

$\delta f(x) = f(x+h/2) - f(x-h/2) = E^{1/2}f(x) - E^{-1/2}f(x) = (E^{1/2} - E^{-1/2})f(x)$.

That is,

$$\delta \equiv E^{1/2} - E^{-1/2}. \tag{2.27}$$

Again,

$$\begin{aligned}\mu f(x) &= \frac{1}{2}\big[f(x+h/2) + f(x-h/2)\big] \\ &= \frac{1}{2}\big[E^{1/2}f(x) + E^{-1/2}f(x)\big] = \frac{1}{2}(E^{1/2} + E^{-1/2})f(x).\end{aligned}$$

Thus,

$$\mu \equiv \frac{1}{2}\big[E^{1/2} + E^{-1/2}\big]. \tag{2.28}$$

The average operator μ can also be expressed in terms of the central difference operator.

$$\begin{aligned}\mu^2 f(x) &= \frac{1}{4}\big[E^{1/2} + E^{-1/2}\big]^2 f(x) \\ &= \frac{1}{4}\big[(E^{1/2} - E^{-1/2})^2 + 4\big]f(x) = \frac{1}{4}\big[\delta^2 + 4\big]f(x).\end{aligned}$$

Hence,

$$\mu \equiv \sqrt{1 + \frac{1}{4}\delta^2}. \tag{2.29}$$

Some more relations among the operators Δ, ∇, E and δ are deduced in the following.

$$\nabla E f(x) = \nabla f(x+h) = f(x+h) - f(x) = \Delta f(x).$$

Also,

$$\delta E^{1/2} f(x) = \delta f(x+h/2) = f(x+h) - f(x) = \Delta f(x).$$

Thus,

$$\Delta \equiv \nabla E \equiv \delta E^{1/2}. \tag{2.30}$$

From the definition of E,

$$\begin{aligned}Ef(x) &= f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \cdots \\ &\quad \text{[by Taylor's series]} \\ &= f(x) + hDf(x) + \frac{h^2}{2!}D^2 f(x) + \frac{h^3}{3!}D^3 f(x) + \cdots \\ &= \left[1 + hD + \frac{h^2}{2!}D^2 + \frac{h^3}{3!}D^3 + \cdots\right]f(x) \\ &= e^{hD}f(x).\end{aligned}$$

Hence,

$$E \equiv e^{hD}. \tag{2.31}$$

Also,

$$hD \equiv \log E. \tag{2.32}$$

This relation is used to separate the effect of E into that of the powers of Δ and this method of separation is called the **method of separation of symbols**.

The operators μ and δ can be expressed in terms of D, as shown below

$$\begin{aligned} \mu f(x) &= \frac{1}{2}[E^{1/2} + E^{-1/2}]f(x) = \frac{1}{2}\big[e^{hD/2} + e^{-hD/2}\big]f(x) \\ &= \cosh\left(\frac{hD}{2}\right)f(x) \\ \text{and} \quad \delta f(x) &= [E^{1/2} - E^{-1/2}]f(x) = \big[e^{hD/2} - e^{-hD/2}\big]f(x) \\ &= 2\sinh\left(\frac{hD}{2}\right)f(x). \end{aligned}$$

Thus,

$$\mu \equiv \cosh\left(\frac{hD}{2}\right) \text{ and } \delta \equiv 2\sinh\left(\frac{hD}{2}\right). \tag{2.33}$$

Again,

$$\mu\delta \equiv 2\cosh\left(\frac{hD}{2}\right)\sinh\left(\frac{hD}{2}\right) = \sinh(hD). \tag{2.34}$$

The inverse relation

$$hD \equiv \sinh^{-1}(\mu\delta) \tag{2.35}$$

is also useful.

Since $E \equiv 1 + \Delta$ and $E^{-1} \equiv 1 - \nabla$, [from (2.25) and (2.26)] from (2.32), it is obtained that

$$hD \equiv \log E \equiv \log(1+\Delta) \equiv -\log(1-\nabla) \equiv \sinh^{-1}(\mu\delta). \tag{2.36}$$

The operators μ and E are commutative, as

$$\mu E f(x) = \mu f(x+h) = \frac{1}{2}\big[f(x+3h/2) + f(x+h/2)\big],$$

while

$$E\mu f(x) = E\Big[\frac{1}{2}\big\{f(x+h/2) + f(x-h/2)\big\}\Big] = \frac{1}{2}\big[f(x+3h/2) + f(x+h/2)\big].$$

Hence,

$$\mu E \equiv E\mu. \tag{2.37}$$

Example 2.3.1 Prove the following relations.
(i) $1+\delta^2\mu^2 \equiv \left(1+\dfrac{\delta^2}{2}\right)^2$, (ii) $E^{1/2} \equiv \mu + \dfrac{\delta}{2}$, (iii) $\Delta \equiv \dfrac{\delta^2}{2} + \delta\sqrt{1+\dfrac{\delta^2}{4}}$,
(iv) $(1+\Delta)(1-\nabla) \equiv 1$, (v) $\mu\delta \equiv \dfrac{\Delta E^{-1}}{2} + \dfrac{\Delta}{2}$, (vi) $\mu\delta \equiv \dfrac{\Delta+\nabla}{2}$,
(vii) $\Delta\nabla \equiv \nabla\Delta \equiv \delta^2$.

Solution. (i) $\delta\mu f(x) = \frac{1}{2}(E^{1/2}+E^{-1/2})(E^{1/2}-E^{-1/2})f(x) = \frac{1}{2}[E-E^{-1}]f(x)$.
Therefore,

$$(1+\delta^2\mu^2)f(x) = \left[1+\frac{1}{4}(E-E^{-1})^2\right]f(x)$$
$$= \left[1+\frac{1}{4}(E^2-2+E^{-2})\right]f(x) = \frac{1}{4}(E+E^{-1})^2 f(x)$$
$$= \left[1+\frac{1}{2}(E^{1/2}-E^{-1/2})^2\right]^2 f(x) = \left[1+\frac{\delta^2}{2}\right]^2 f(x).$$

Hence

$$1+\delta^2\mu^2 \equiv \left(1+\frac{\delta^2}{2}\right)^2. \tag{2.38}$$

(ii) $\left(\mu+\dfrac{\delta}{2}\right)f(x) = \left\{\dfrac{1}{2}[E^{1/2}+E^{-1/2}] + \dfrac{1}{2}[E^{1/2}-E^{-1/2}]\right\}f(x) = E^{1/2}f(x)$.
Thus

$$E^{1/2} \equiv \mu + \frac{\delta}{2}. \tag{2.39}$$

(iii)

$$\left[\frac{\delta^2}{2}+\delta\sqrt{1+\frac{\delta^2}{4}}\right]f(x)$$
$$= \frac{1}{2}(E^{1/2}-E^{-1/2})^2 f(x) + \left[(E^{1/2}-E^{-1/2})\sqrt{1+\frac{1}{4}(E^{1/2}-E^{-1/2})^2}\right]f(x)$$
$$= \frac{1}{2}[E+E^{-1}-2]f(x) + \frac{1}{2}(E^{1/2}-E^{-1/2})(E^{1/2}+E^{-1/2})f(x)$$
$$= \frac{1}{2}[E+E^{-1}-2]f(x) + \frac{1}{2}(E-E^{-1})f(x)$$
$$= (E-1)f(x).$$

Hence,

$$\frac{\delta^2}{2}+\delta\sqrt{1+\frac{\delta^2}{4}} \equiv E-1 \equiv \Delta. \tag{2.40}$$

(iv) $(1+\Delta)(1-\nabla)f(x) = (1+\Delta)[f(x) - f(x) + f(x-h)]$
$= (1+\Delta)f(x-h) = f(x-h) + f(x) - f(x-h)$
$= f(x).$

Therefore,

$$(1+\Delta)(1-\nabla) \equiv 1. \tag{2.41}$$

(v)
$$\left[\frac{\Delta E^{-1}}{2} + \frac{\Delta}{2}\right] f(x) = \frac{1}{2}[\Delta f(x-h) + \Delta f(x)]$$
$$\begin{aligned}
&= \frac{1}{2}[f(x) - f(x-h) + f(x+h) - f(x)]\\
&= \frac{1}{2}[f(x+h) - f(x-h)] = \frac{1}{2}[E - E^{-1}]f(x)\\
&= \frac{1}{2}(E^{1/2} + E^{-1/2})(E^{1/2} - E^{-1/2})f(x)\\
&= \mu\delta f(x).
\end{aligned}$$

Hence

$$\frac{\Delta E^{-1}}{2} + \frac{\Delta}{2} \equiv \mu\delta. \tag{2.42}$$

(vi)
$$\begin{aligned}
\left[\frac{\Delta + \nabla}{2}\right] f(x) &= \frac{1}{2}[\Delta f(x) + \nabla f(x)]\\
&= \frac{1}{2}[f(x+h) - f(x) + f(x) - f(x-h)]\\
&= \frac{1}{2}[f(x+h) - f(x-h)] = \frac{1}{2}[E - E^{-1}]f(x)\\
&= \mu\delta f(x) \qquad \text{(as in previous case).}
\end{aligned}$$

Thus,
$$\mu\delta \equiv \frac{\Delta + \nabla}{2}. \tag{2.43}$$

(vii) $\Delta\nabla f(x) = \Delta[f(x) - f(x-h)] = f(x+h) - 2f(x) + f(x-h).$
Again,

$$\begin{aligned}
\nabla\Delta f(x) &= f(x+h) - 2f(x) + f(x-h) = (E - 2 + E^{-1})f(x)\\
&= (E^{1/2} - E^{-1/2})^2 f(x) = \delta^2 f(x).
\end{aligned}$$

Hence,
$$\Delta\nabla \equiv \nabla\Delta \equiv (E^{1/2} - E^{-1/2})^2 \equiv \delta^2. \tag{2.44}$$

The relations among the various operators are shown in Table 2.4.

Table 2.4: Relationship between the operators.

	E	Δ	∇	δ	hD
E	E	$\Delta+1$	$(1-\nabla)^{-1}$	$1+\frac{\delta^2}{2}+\delta\sqrt{1+\frac{\delta^2}{4}}$	e^{hD}
Δ	$E-1$	Δ	$(1-\nabla)^{-1}-1$	$\frac{\delta^2}{2}+\delta\sqrt{1+\frac{\delta^2}{4}}$	$e^{hD}-1$
∇	$1-E^{-1}$	$1-(1+\Delta)^{-1}$	∇	$-\frac{\delta^2}{2}+\delta\sqrt{1+\frac{\delta^2}{4}}$	$1-e^{-hD}$
δ	$E^{1/2}-E^{-1/2}$	$\Delta(1+\Delta)^{-1/2}$	$\nabla(1-\nabla)^{-1/2}$	δ	$2\sinh(hD/2)$
μ	$\frac{E^{1/2}+E^{-1/2}}{2}$	$(1+\Delta/2)$ $\times(1+\Delta)^{-1/2}$	$(1-\nabla/2)(1-\nabla)^{-1/2}$	$1+\frac{\delta^2}{4}$	$\cosh(hD/2)$
hD	$\log E$	$\log(1+\Delta)$	$-\log(1-\nabla)$	$2\sinh^{-1}(\delta/2)$	hD

From definition of derivative, we have

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}=\lim_{h\to 0}\frac{\Delta f(x)}{h}.$$

Thus one can write,

$$\Delta f(x)\simeq hf'(x).$$

Again,

$$\begin{aligned}f''(x)&=\lim_{h\to 0}\frac{f'(x+h)-f'(x)}{h}\\&\simeq\lim_{h\to 0}\frac{\dfrac{\Delta f(x+h)}{h}-\dfrac{\Delta f(x)}{h}}{h}\\&=\lim_{h\to 0}\frac{\Delta f(x+h)-\Delta f(x)}{h^2}=\lim_{h\to 0}\frac{\Delta^2 f(x)}{h^2}.\end{aligned}$$

Therefore, $h^2f''(x)\simeq\Delta^2 f(x)$.

In general, $\Delta^n f(x)\simeq h^n f^n(x)$. That is, for small h, the operators Δ and hD are almost equal.

2.4 Representation of Polynomial using Factorial Notation

From the definition of factorial notation,

$$\begin{aligned} x^{(0)} &= 1 \\ x^{(1)} &= x \\ x^{(2)} &= x(x-h) \\ x^{(3)} &= x(x-h)(x-2h) \\ x^{(4)} &= x(x-h)(x-2h)(x-3h) \end{aligned} \tag{2.45}$$

and so on.

The above relations show that $x^{(n)}$, $n = 1, 2, \ldots$ is a polynomial of degree n in x. Also, $x, x^2, x^3, \ldots$ can be expressed in terms of factorial notations $x^{(1)}, x^{(2)}, x^{(3)}, \ldots$, as shown below.

$$\begin{aligned} 1 &= x^{(0)} \\ x &= x^{(1)} \\ x^2 &= x^{(2)} + hx^{(1)} \\ x^3 &= x^{(3)} + 3hx^{(2)} + h^2x^{(1)} \\ x^4 &= x^{(4)} + 6hx^{(3)} + 7h^2x^{(2)} + h^3x^{(1)} \end{aligned} \tag{2.46}$$

and so on.

These relations show that x^n can be expressed as a polynomial of $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$, of degree n. Once a polynomial is expressed in a factorial notation, its differences can be obtained by using the formula like differential calculus.

Example 2.4.1 Express $f(x) = 2x^4 + x^3 - 5x^2 + 8$ in factorial notation and find its first and second differences.

Solution. Here we assume that $h = 1$.
Then by (2.46), $x = x^{(1)}, x^2 = x^{(2)} + x^{(1)}, x^3 = x^{(3)} + 3x^{(2)} + x^{(1)}$,
$x^4 = x^{(4)} + 6x^{(3)} + 7x^{(2)} + x^{(1)}$.
Using these values, the function $f(x)$ becomes

$$\begin{aligned} f(x) &= 2\big[x^{(4)} + 6x^{(3)} + 7x^{(2)} + x^{(1)}\big] + \big[x^{(3)} + 3x^{(2)} + x^{(1)}\big] - 5\big[x^{(2)} + x^{(1)}\big] + 8 \\ &= 2x^{(4)} + 13x^{(3)} + 12x^{(2)} - 2x^{(1)} + 8. \end{aligned}$$

Now, the Property 2.2.11, i.e., $\Delta x^{(n)} = nx^{(n-1)}$ is used to find the differences.
Therefore,
$\Delta f(x) = 2.4x^{(3)} + 13.3x^{(2)} + 12.2x^{(1)} - 2.1x^{(0)} = 8x^{(3)} + 39x^{(2)} + 24x^{(1)} - 2$
and $\Delta^2 f(x) = 24x^{(2)} + 78x^{(1)} + 24$.
In terms of x,
$\Delta f(x) = 8x(x-1)(x-2) + 39x(x-1) + 24x - 2$
and $\Delta^2 f(x) = 24x(x-1) + 78x + 24$.

From the relations of (2.46) one can conclude the following result.

Lemma 2.4.1 *Any polynomial $f(x)$ in x of degree n can be expressed in factorial notation with same degree, n.*

This means, in conversion to the factorial notation, the degree of a polynomial remains unchanged.

The above process to convert a polynomial in a factorial form is a labourious technique when the degree of the polynomial is large. The other systematic process, like Maclaurin's formula in differential calculus, is used to convert a polynomial, even a function, in factorial notation.

Let $f(x)$ be a polynomial in x of degree n. In factorial notation, let it be

$$f(x) = a_0 + a_1x^{(1)} + a_2x^{(2)} + \cdots + a_nx^{(n)}, \tag{2.47}$$

where a_i's are unknown constants to be determined, $a_n \neq 0$.

Now, one can find the differences of (2.47) as follows.

$$\begin{aligned}
\Delta f(x) &= a_1 + 2a_2x^{(1)} + 3a_3x^{(2)} + \cdots + na_nx^{(n-1)}\\
\Delta^2 f(x) &= 2.1a_2 + 3.2a_3x^{(1)} + \cdots + n(n-1)a_nx^{(n-2)}\\
\Delta^3 f(x) &= 3.2.1a_3 + 4.3.2.x^{(1)} + \cdots + n(n-1)(n-2)a_nx^{(n-3)}\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
\Delta^n f(x) &= n(n-1)(n-2)\cdots 3\cdot 2\cdot 1a_n = n!a_n.
\end{aligned}$$

When $x = 0$, the above relations give

$$\begin{array}{lll}
a_0 = f(0), & & \Delta f(0) = a_1,\\
\Delta^2 f(0) = 2.1.a_2 & \text{or,} & a_2 = \dfrac{\Delta^2 f(0)}{2!}\\
\Delta^3 f(0) = 3.2.1.a_3 & \text{or,} & a_3 = \dfrac{\Delta^3 f(0)}{3!}\\
\cdots\cdots\cdots\cdots\cdots & & \\
\Delta^n f(0) = n!a_n & \text{or,} & a_n = \dfrac{\Delta^n f(0)}{n!}.
\end{array}$$

Using these results equation (2.47) becomes

$$f(x) = f(0) + \Delta f(0)x^{(1)} + \frac{\Delta^2 f(0)}{2!}x^{(2)} + \frac{\Delta^3 f(0)}{3!}x^{(3)} + \cdots + \frac{\Delta^n f(0)}{n!}x^{(n)}. \tag{2.48}$$

This formula is similar to Maclaurin's formula of differential calculus and it is also used to expand a function in terms of factorial notation. To perform the formula (2.48), different forward differences are to be evaluated at $x = 0$, and this can be done using forward difference table. This is a systematic process and easy to implement as computer program.

Example 2.4.2 Express $f(x) = 2x^4 - 5x^3 + 8x^2 + 2x - 1$ in factorial notation.

Solution. Taking $h = 1$. $f(0) = -1, f(1) = 6, f(2) = 27, f(3) = 104, f(4) = 327$.

x	$f(x)$	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$	$\Delta^4 f(x)$
0	−1				
		7			
1	6		14		
		21		42	
2	27		56		48
		77		90	
3	104		146		
		223			
4	327				

Thus using formula (2.48)

$$f(x) = f(0) + \Delta f(0)x^{(1)} + \frac{\Delta^2 f(0)}{2!}x^{(2)} + \frac{\Delta^3 f(0)}{3!}x^{(3)} + \frac{\Delta^4 f(0)}{4!}x^{(4)}$$
$$= 2x^{(4)} + 7x^{(3)} + 7x^{(2)} + 7x^{(1)} - 1.$$

Example 2.4.3 If $\Delta f(x) = x^4 + 2x^3 + 8x^2 + 3$, find $f(x)$.

Solution. Applying synthetic division to express $\Delta f(x)$ in factorial notation.

$$\begin{array}{c|ccccc}
1 & 1 & 2 & 8 & 0 & 3 \\
 & & 1 & 3 & 11 & \\
\hline
2 & 1 & 3 & 11 & 11 & \\
 & & 2 & 10 & & \\
\hline
3 & 1 & 5 & 21 & & \\
 & & 3 & & & \\
\hline
4 & 1 & 8 & & & \\
\hline
 & 1 & & & &
\end{array}$$

Therefore, $\Delta f(x) = x^{(4)} + 8x^{(3)} + 21x^{(2)} + 11x^{(1)} + 3$.
Hence,

$$f(x) = \frac{1}{5}x^{(5)} + \frac{8}{4}x^{(4)} + \frac{21}{3}x^{(3)} + \frac{11}{2}x^{(2)} + 3x^{(1)} + c, \text{[using Property 2.2.12]}$$
$$= \frac{1}{5}x(x-1)(x-2)(x-3)(x-4) + 2x(x-1)(x-2)(x-3)$$
$$+7x(x-1)(x-2) + \frac{11}{2}x(x-1) + 3x + c, \text{ where } c \text{ is arbitrary constant.}$$

2.5 Difference of a Polynomial

Let $f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ be a polynomial of degree n. From this expression one can find the successive differences of $f(x)$.

Now, $\Delta f(x) = f(x+h) - f(x)$, where h is the spacing of x
$= a_0[(x+h)^n - x^n] + a_1[(x+h)^{n-1} - x^{n-1}] + \cdots + a_{n-1}[(x+h) - x]$.

Expanding the terms within parenthesis using binomial theorem and obtain

$$\begin{aligned}\Delta f(x) = a_0\Big[&x^n + nhx^{n-1} + \frac{n(n-1)}{2!}h^2x^{n-2} + \cdots + h^n - x^n\Big]\\ &+a_1\Big[x^{n-1} + (n-1)hx^{n-2} + \frac{(n-1)(n-2)}{2!}h^2x^{n-3} + \cdots + h^{n-1} - x^{n-1}\Big]\\ &+ \cdots + ha_{n-1}\\ &= a_0nhx^{n-1} + \Big[a_0\frac{n(n-1)}{2!}h^2 + a_1(n-1)h\Big]x^{n-2} + \cdots + a_{n-1}h.\end{aligned}$$

If h is constant, then the coefficients of $x^{n-1}, x^{n-2}, \ldots, x$ and $a_{n-1}h$ are constants. The coefficients of $x^{n-1}, x^{n-2}, \ldots, x$ and the constant term are denoted by $b_0, b_1, \ldots, b_{n-2}$ and b_{n-1} respectively. In this notation first difference can be written as

$$\Delta f(x) = b_0x^{n-1} + b_1x^{n-2} + \cdots + b_{n-2}x + b_{n-1}, \text{ where } b_0 = a_0nh.$$

It may be noted that $\Delta f(x)$ is a polynomial of degree $n-1$, i.e., first difference reduces the degree of $f(x)$ by 1.

The second difference of $f(x)$ is

$$\begin{aligned}&\Delta^2 f(x)\\ &= \Delta f(x+h) - \Delta f(x)\\ &= b_0[(x+h)^{n-1} - x^{n-1}] + b_1[(x+h)^{n-2} - x^{n-2}] + \cdots + b_{n-2}[(x+h) - x]\\ &= b_0\Big[x^{n-1} + (n-1)hx^{n-2} + \frac{(n-1)(n-2)}{2!}h^2x^{n-3} + \cdots + h^{n-1} - x^{n-1}\Big]\\ &\quad +b_1\Big[x^{n-2} + (n-2)hx^{n-3} + \frac{(n-2)(n-3)}{2!}h^2x^{n-4} + \cdots + h^{n-2} - x^{n-2}\Big]\\ &\quad + \cdots + b_{n-2}h\\ &= b_0(n-1)hx^{n-2} + \Big[\frac{1}{2!}(n-1)(n-2)b_0h^2 + (n-2)b_1h\Big]x^{n-3} + \cdots + b_{n-2}h\\ &= c_0x^{n-2} + c_1x^{n-3} + \cdots + c_{n-3}x + c_{n-2}\end{aligned}$$

where $c_0 = b_0(n-1)h, c_1 = \frac{1}{2!}(n-1)(n-2)b_0h^2 + (n-2)b_1h$, etc.

This expression shows that $\Delta^2 f(x)$ is a polynomial of degree $n-2$.

It may be noted that the coefficient of the leading term is $c_0 = b_0h(n-1) = n(n-1)h^2a_0$ and it is a constant quantity.

In this way, one can find $\Delta^{n-1}f(x)$ is a polynomial of degree one and let it be p_0x+p_1, i.e., $\Delta^{n-1}f(x) = p_0x + p_1$.

Then $\Delta^n f(x) = p_0(x+h)+p_1-p_0x-p_1 = p_0h$, which is a constant. And $\Delta^{n+1}f(x) = 0$.

It can be shown that $\Delta^n f(x) = n(n-1)(n-2)\cdots 2\cdot 1\cdot h^n a_0 = n!h^n a_0$.

Thus finally,

$$\begin{aligned}&\Delta^k f(x), k < n \text{ is a polynomial of degree } n-k,\\ &\Delta^n f(x) \text{ is constant, and}\\ &\Delta^k f(x), k > n \text{ is zero.}\end{aligned}$$

Alternative proof.

It is observed that, a polynomial in x of degree n can be expressed as a polynomial in factorial notation with same degree n.

Thus, if $f(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-1}x + a_n$ be the given polynomial then it can be written as $f(x) = b_0x^{(n)} + b_1x^{(n-1)} + b_2x^{(n-2)} + \cdots + b_{n-1}x^{(1)} + b_n$.

Therefore,
$\Delta f(x) = b_0nhx^{(n-1)} + b_1h(n-1)x^{(n-2)} + b_2h(n-2)x^{(n-3)} + \cdots + b_{n-1}h$.
Clearly this is a polynomial of degree $n-1$.

Similarly,

$$\begin{aligned}\Delta^2 f(x) &= b_0n(n-1)h^2x^{(n-2)} + b_1(n-1)(n-2)h^2x^{(n-3)} + \cdots + b_{n-2}h^2,\\ \Delta^3 f(x) &= b_0n(n-1)(n-2)h^3x^{(n-3)} + b_1(n-1)(n-2)(n-3)h^3x^{(n-4)}\\ &\quad + \cdots + b_{n-3}h^3.\end{aligned}$$

In this way,

$$\begin{aligned}\Delta^n f(x) &= b_0n(n-1)(n-2)\cdots 2\cdot 1\cdot h^n x^{(n-n)}\\ &= b_0n!h^n, \qquad \text{a constant quantity.}\end{aligned}$$

Hence $\Delta^{n+1}f(x) = 0$.

Difference of factorial power function

From definition of factorial notation, we have

$$\begin{aligned}\Delta x^{(n)} &= nhx^{(n-1)}\\ \Delta^2 x^{(n)} &= nh\Delta x^{(n-1)} = nh.(n-1)hx^{(n-2)} = n(n-1)h^2x^{(n-2)}\\ \Delta^3 x^{(n)} &= n(n-1)h^2.(n-2)hx^{(n-2)} = n(n-1)(n-2)h^3x^{(n-3)}.\end{aligned}$$

In this way,

$$\Delta^n x^{(n)} = n(n-1)(n-2)\cdots 2\cdot 1\cdot h^n x^{(n-n)} = n!h^2.$$

Example 2.5.1 Given $x_i = x_0 + ih, i = 0, 1, 2, \ldots, n; h > 0$
and $u_i(x) = (x - x_0)(x - x_1)\cdots(x - x_i)$,
prove that
$\Delta^k u_i(x) = (i+1)i(i-1)\cdots(i-k+2)h^k(x-x_0)(x-x_1)\cdots(x-x_{i-k})$.

Solution. Here $u_i(x) = (x - x_0)(x - x_1)\cdots(x - x_i) = (x - x_0)^{(i+1)}$ (say).
Therefore,

$$\begin{aligned}
\Delta u_i(x) &= (x+h-x_0)(x+h-x_1)\cdots(x+h-x_i) - (x-x_0)\cdots(x-x_i)\\
&= (x+h-x_0)(x-x_0)(x-x_1)\cdots(x-x_{i-1})\\
&\quad -(x-x_0)(x-x_1)\cdots(x-x_i)\\
&= (x-x_0)(x-x_1)\cdots(x-x_{i-1})[(x+h-x_0)-(x-x_i)]\\
&= (x-x_0)(x-x_1)\cdots(x-x_{i-1})(h+x_i-x_0)\\
&= (x-x_0)(x-x_1)\cdots(x-x_{i-1})(i+1)h \qquad [\text{since } x_i = x_0+ih]\\
&= (i+1)h(x-x_0)^{(i)}.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\Delta^2 u_i(x) &= (i+1)h[(x+h-x_0)(x+h-x_1)\cdots(x+h-x_{i-1})\\
&\quad -(x-x_0)(x-x_1)\cdots(x-x_{i-1})]\\
&= (i+1)h(x-x_0)(x-x_1)\cdots(x-x_{i-2})[(x+h-x_0)-(x-x_{i-1})]\\
&= (i+1)h(x-x_0)^{(i-1)}ih\\
&= (i+1)ih^2(x-x_0)^{(i-1)}.
\end{aligned}$$

In similar way,

$$\Delta^3 u_i(x) = (i+1)i(i-1)h^3(x-x_0)^{(i-2)}.$$

Hence,

$$\begin{aligned}
\Delta^k u_i(x) &= (i+1)i(i-1)\cdots(i-\overline{k-2})h^k(x-x_0)^{(i-\overline{k-1})}\\
&= (i+1)i(i-1)\cdots(i-k+2)h^k(x-x_0)(x-x_1)\cdots(x-x_{i-k}).
\end{aligned}$$

2.6 Summation of Series

The finite difference method is also used to find the sum of a finite series. Two important results are presented here.

Theorem 2.1 *If $f(x)$ be defined only for integral values of independent variable x, then*

$$\sum_{x=a}^{b} f(x) = f(a) + f(a+h) + f(a+2h) + \cdots + f(b) = \Big[F(x)\Big]_a^{b+h}$$

$$\text{i.e.,} \sum_{x=a}^{b} f(x) = F(x+h) - F(x), \qquad b = a + nh \text{ for some } n \tag{2.49}$$

where $F(x)$ is an anti-difference (instead of anti-derivative) of $f(x)$,
i.e., $\Delta F(x) = f(x)$.

Proof. Since $\Delta F(x) = f(x)$, therefore,

$$\begin{aligned}\sum_{x=a}^{b} f(x) &= \sum_{x=a}^{b} \Delta F(x) = \sum_{x=a}^{b}[F(x+h) - F(x)]\\ &= [F(b+h) - F(b)] + [F(b) - F(b-h)] + \cdots + [F(a+h) - F(a)]\\ &= F(b+h) - F(a).\end{aligned}$$

Thus, if $F(x)$ is anti-difference of $f(x)$, i.e.,
$\Delta^{-1} f(x) = F(x)$, then $\sum_{x=a}^{b} f(x) = F(b+h) - F(a)$.

Example 2.6.1 Use finite difference method to find the sum of the series $\sum_{x=1}^{n} f(x)$, where $f(x) = x(x-1) + \dfrac{3}{x(x+1)(x+2)}$.

Solution. Let

$$\begin{aligned}F(x) = \Delta^{-1} f(x) &= \Delta^{-1}\left[x(x-1) + \frac{3}{x(x+1)(x+2)}\right]\\ &= \Delta^{-1} x^{(2)} + 3\Delta^{-1} x^{(-3)} = \frac{x^{(3)}}{3} + 3\frac{x^{(-2)}}{-2}\\ &= \frac{1}{3}x(x-1)(x-2) - \frac{3}{2}\frac{1}{x(x+1)}.\end{aligned}$$

Therefore,

$$\begin{aligned}\sum_{x=1}^{n} f(x) = [F(n+1) - F(1)] &= \frac{1}{3}(n+1)n(n-1) - \frac{3}{2}\frac{1}{(n+1)(n+2)} - 0 + \frac{3}{2}\frac{1}{1.2}\\ &= \frac{1}{3}n(n^2-1) - \frac{3}{2}\frac{1}{(n+1)(n+2)} + \frac{3}{4}.\end{aligned}$$

Summation by parts

Like the formula 'integration by parts' of integral calculus there is a similar formula in finite difference calculus. If $f(x)$ and $g(x)$ are two functions defined only for integral values of x between a and b, then

$$\sum_{x=a}^{b} f(x)\Delta g(x) = \Big[f(x)g(x)\Big]_a^{b+h} - \sum_{x=a}^{b} g(x+h)\Delta f(x). \tag{2.50}$$

Example 2.6.2 Find the sum of the series $\sum\limits_{x=1}^{n} x4^x$.

Solution. Let $f(x) = x$, $\Delta g(x) = 4^x$. Then $g(x) = 4^x/3$. Hence for $h = 1$,

$$\sum_{x=1}^{n} x4^x = \Big[x.\frac{4^x}{3}\Big]_1^{n+1} - \sum_{x=1}^{n} \frac{4^{x+1}}{3}.\Delta x = \Big[(n+1)\frac{4^{n+1}}{3} - \frac{4}{3}\Big] - \frac{4}{3}\sum_{x=1}^{n} 4^x.1$$
$$= \Big[(n+1)\frac{4^{n+1}}{3} - \frac{4}{3}\Big] - \frac{4}{3}\frac{4(4^n-1)}{4} = n\frac{4^{n+1}}{3}.$$

2.7 Worked out Examples

Example 2.7.1 If n is a positive integer then show that
$x^{(-n)} = \dfrac{1}{(x+nh)^{(n)}}$, where h is the spacing.

Solution. From the definition of $x^{(n)}$, we have,
$x^{(n)} = x(x-h)(x-2h)\cdots(x-\overline{n-2}h)(x-\overline{n-1}h) = x^{(n-1)}(x-\overline{n-1}h)$.
This can be written as $x^{(n-1)} = \dfrac{x^{(n)}}{x-(n-1)h}$.
Substituting $n = 0, -1, -2, \ldots, -(n-1)$ we obtain

$$x^{(-1)} = \frac{x^{(0)}}{x+h} = \frac{1}{x+h}$$
$$x^{(-2)} = \frac{x^{(-1)}}{x+2h} = \frac{1}{(x+h)(x+2h)}$$
$$x^{(-3)} = \frac{x^{(-2)}}{x+3h} = \frac{1}{(x+h)(x+2h)(x+3h)}.$$

In this way,

$$x^{(-\overline{n-1})} = \frac{1}{(x+h)(x+2h)\cdots(x+\overline{n-1}h)}$$

$$\text{and hence } x^{(-n)} = \frac{1}{(x+nh)(x+nh-h)(x+nh-2h)\cdots(x+nh-\overline{n-1}h)} = \frac{1}{(x+nh)^{(n)}}.$$

Example 2.7.2 Prove the following identities

(i) $u_0+u_1+u_2+\cdots+u_n = {}^{n+1}C_1 u_0 + {}^{n+1}C_2 \Delta u_0 + {}^{n+1}C_3 \Delta^2 u_0 + \cdots + \Delta^n u_0.$

(ii) $u_0 + xu_1 + \frac{x^2}{2!}u_2 + \frac{x^3}{3!}u_3 + \cdots = e^x\left[u_0 + x\Delta u_0 + \frac{x^2}{2!}\Delta^2 u_0 + \cdots\right]$

(iii) $u_0 - u_1 + u_2 - u_3 + \cdots = \frac{1}{2}u_0 - \frac{1}{4}\Delta u_0 + \frac{1}{8}\Delta^2 u_0 - \frac{1}{16}\Delta^3 u_0 + \cdots.$

Solution. (i) $u_0+u_1+u_2+\cdots+u_n$

$$\begin{aligned}
&= u_0 + Eu_0 + E^2u_0 + \cdots + E^n u_0\\
&= (1+E+E^2+\cdots+E^n)u_0 = \left(\frac{E^{n+1}-1}{E-1}\right)u_0\\
&= \left(\frac{(1+\Delta)^{n+1}-1}{\Delta}\right)u_0 \qquad [\text{since } E \equiv 1+\Delta]\\
&= \frac{1}{\Delta}\left[{}^{n+1}C_1\Delta + {}^{n+1}C_2\Delta^2 + \cdots + \Delta^{n+1}\right]u_0\\
&= {}^{n+1}C_1 u_0 + {}^{n+1}C_2\Delta u_0 + {}^{n+1}C_3\Delta^2 u_0 + \cdots + \Delta^n u_0.
\end{aligned}$$

(ii)

$$\begin{aligned}
& u_0 + xu_1 + \frac{x^2}{2!}u_2 + \frac{x^3}{3!}u_3 + \cdots\\
&= u_0 + xEu_0 + \frac{x^2}{2!}E^2u_0 + \frac{x^3}{3!}E^3u_0 + \cdots\\
&= \left[1 + xE + \frac{(xE)^2}{2!} + \frac{(xE)^3}{3!} + \cdots\right]u_0\\
&= e^{xE}u_0 = e^{x(1+\Delta)}u_0 = e^x e^{x\Delta}u_0\\
&= e^x\left[1 + x\Delta + \frac{(x\Delta)^2}{2!} + \frac{(x\Delta)^3}{3!} + \cdots\right]u_0\\
&= e^x\left[u_0 + x\Delta u_0 + \frac{x^2}{2!}\Delta^2u_0 + \frac{x^3}{3!}\Delta^3u_0 + \cdots\right]
\end{aligned}$$

(iii)

$$\begin{aligned}
&u_0 - u_1 + u_2 - u_3 + \cdots \\
&= u_0 - Eu_0 + E^2u_0 - E^3u_0 + \cdots = [1 - E + E^2 - E^3 + \cdots]u_0 \\
&= (1+E)^{-1}u_0 = (2+\Delta)^{-1}u_0 = \frac{1}{2}\left(1+\frac{\Delta}{2}\right)^{-1}u_0 \\
&= \frac{1}{2}\left[1 - \frac{\Delta}{2} + \frac{\Delta^2}{4} - \frac{\Delta^3}{8} + \cdots\right]u_0 = \frac{1}{2}u_0 - \frac{\Delta u_0}{4} + \frac{\Delta^2 u_0}{8} - \frac{\Delta^3 u_0}{16} + \cdots
\end{aligned}$$

Example 2.7.3 Let $f_i = f(x_i)$ where $x_i = x_0 + ih$, $i = 1, 2, \ldots$. Prove that

$$f_i = E^i f_0 = \sum_{j=0}^{i} \binom{i}{j} \Delta^j f_0.$$

Solution. From definition of Δ,
$\Delta f(x_i) = f(x_i + h) - f(x_i) = Ef(x_i) - f(x_i)$,
i.e., $Ef(x_i) = \Delta f(x_i) + f(x_i) = (\Delta + 1)f(x_i)$.

Hence, $E \equiv \Delta + 1$ and also $E^i \equiv (1 + \Delta)^i$.
Therefore,

$$f_i = E^i f_0 = (1+\Delta)^i f_0 = 1 + {}^iC_1 \Delta f_0 + {}^iC_2 \Delta^2 f_0 + \cdots = \sum_{j=0}^{i} \binom{i}{j} \Delta^j f_0.$$

Example 2.7.4 Prove that

$$\Delta^n f(x) = \sum_{i=0}^{n} (-1)^i \, {}^nC_i f[x + (n-i)h],$$

where h is step-length.

Solution. We know that, $\Delta \equiv E - 1$ and $\Delta^n \equiv (E-1)^n$.

Therefore, $\Delta^n f(x) = (E-1)^n f(x) = \sum_{i=0}^{n} (-1)^i \, {}^nC_i E^{n-i} f(x)$.

[by Binomial theorem]
Now, $Ef(x) = f(x+h), E^2 f(x) = f(x+2h), \ldots, E^{n-i}f(x) = f[x + (n-i)h]$.

Hence $\Delta^n f(x) = \sum_{i=0}^{n} (-1)^i \, {}^nC_i f[x + (n-i)h]$.

Example 2.7.5 Find the polynomial $f(x)$ which satisfies the following data and hence find the value of $f(1.5)$.

x	:	1	2	3	4
$f(x)$	:	3	5	10	30

Solution. The difference table of the given data is

x	$f(x)$	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$
1	3			
		2		
2	5		3	
		5		12
3	10		15	
		20		
4	30			

It is known that, $f(x_0 + nh) = E^n f(x_0) = (1 + \Delta)^n f(x_0)$.
Here $x_0 = 1, h = 1$. Let $x_0 + nh = 1 + n = x$, i.e., $n = x - 1$. Therefore,

$$\begin{aligned}
f(x) &= E^{x-1} f(x_0) = (1+\Delta)^{x-1} f(x_0) \\
&= f(0) + (x-1)\Delta f(0) + \frac{(x-1)(x-2)}{2!}\Delta^2 f(0) \\
&\quad + \frac{(x-1)(x-2)(x-3)}{3!}\Delta^3 f(0) + \cdots \\
&= 3 + 2(x-1) + \frac{3(x-1)(x-2)}{2!} + \frac{12(x-1)(x-2)(x-3)}{3!} \\
&= 2x^3 - \frac{21}{2}x^2 + \frac{39}{2}x - 8.
\end{aligned}$$

Thus $f(1.5) = 4.375$.

Example 2.7.6 Find the missing term in the following table:

x	:	1	2	3	4	5
$f(x)$	:	–2	3	8	–	21

Solution. Here four values of $f(x)$ are given. So, we consider $f(x)$ be a polynomial of degree 3. Thus the fourth differences of $f(x)$ vanish, i.e.,
$\Delta^4 f(x) = 0$ or, $(E - 1)^4 f(x) = 0$
or, $(E^4 - 4E^3 + 6E^2 - 4E + 1)f(x) = 0$
or, $E^4 f(x) - 4E^3 f(x) + 6E^2 f(x) - 4Ef(x) + f(x) = 0$

or, $f(x+4) - 4f(x+3) + 6f(x+2) - 4f(x+1) + f(x) = 0$.
Here, $h = 1$ as the values are in spacing of 1 unit.
For $x = 1$ the above equation becomes
$f(5) - 4f(4) + 6f(3) - 4f(2) + f(1) = 0$ or, $21 - 4f(4) + 6 \times 8 - 4 \times 3 - 2 = 0$
or, $f(4) = 13.75$.

Example 2.7.7 Use finite difference method to find the values of a and b in the following table.

x	:	0	2	4	6	8	10
$f(x)$	:	–5	a	8	b	20	32

Solution. Here, four values of $f(x)$ are known, so we can assume that $f(x)$ is a polynomial of degree 3. Then, $\Delta^4 f(x) = 0$.
or, $(E-1)^4 f(x) = 0$
or, $E^4 f(x) - 4E^3 f(x) + 6E^2 f(x) - 4Ef(x) + f(x) = 0$
or, $f(x+8) - 4f(x+6) + 6f(x+4) - 4f(x+2) + f(x) = 0$
[Here $h = 2$, because the values of x are given in 2 unit interval]
In this problem, two unknowns a and b are to be determined and needs two equations. Therefore, the following equations are obtained by substituting $x = 2$ and $x = 0$ to the above equation.
$f(10) - 4f(8) + 6f(6) - 4f(4) + f(2) = 0$ and
$f(8) - 4f(6) + 6f(4) - 4f(2) + f(0) = 0$.
These equations are simplifies to
$32 - 4 \times 20 + 6b - 4 \times 8 + a = 0$ and $20 - 4b + 6 \times 8 - 4a - 5 = 0$.
That is, $6b + a - 80 = 0$ and $-4b - 4a + 63 = 0$. Solution of these equations is $a = 2.9, b = 12.85$.

Example 2.7.8 Find the value of $\left(\dfrac{\Delta^2}{E}\right)x^2$.

Solution.

$$\left(\frac{\Delta^2}{E}\right)x^2 = \left[\frac{(E-1)^2}{E}\right]x^2$$
$$= \left[\frac{E^2 - 2E + 1}{E}\right]x^2$$
$$= (E - 2 + E^{-1})x^2 = Ex^2 - 2x^2 + E^{-1}x^2$$
$$= (x+1)^2 - 2x^2 + (x-1)^2$$
$$= 2.$$

Example 2.7.9 Show that $\nabla \log f(x) = -\log\left[1 - \frac{\nabla f(x)}{f(x)}\right]$.

Solution.

$$\begin{aligned}\nabla \log f(x) &= \log f(x) - \log f(x-h) = \log f(x) - \log E^{-1}f(x)\\ &= \log \frac{f(x)}{E^{-1}f(x)} = -\log \frac{E^{-1}f(x)}{f(x)} = -\log \frac{(1-\nabla)f(x)}{f(x)}\\ &= -\log \frac{f(x)-\nabla f(x)}{f(x)} = -\log\left[1 - \frac{\nabla f(x)}{f(x)}\right].\end{aligned}$$

The following formulae for anti-difference can easily be verified, for $h = 1$.

(i) $\Delta^{-1}cf(x) = c\Delta^{-1}f(x)$, c being a constant.

(ii) $\Delta^{-1}x^{(n)} = \frac{1}{n+1}x^{(n+1)}$, n being a positive integer.

(iii) $\Delta^{-1}a^x = \dfrac{1}{a-1}a^x, \qquad a \neq 1.$

(iv) $\Delta^{-1}\sin ax = -\dfrac{1}{2\sin a/2}\cos(ax - a/2).$

(v) $\Delta^{-1}\cos ax = \dfrac{1}{2\sin a/2}\sin(ax - a/2).$

2.8 Difference Equations

Like differential equations, difference equations have many applications in different branches of mathematics, statistics and other field of science and engineering.

A difference equation is an equation containing an independent variable, a dependent variable and successive differences of dependent variable. The difference equations are some times called **recurrence equations**.

For example,

$$a\Delta^2 f(x) + b\Delta f(x) + cf(x) = g(x), \tag{2.51}$$

where a, b, c are constants, $g(x)$ is a known function and $f(x)$ is the unknown function. The solution of a difference equation is the value of the unknown function.

The difference equation can also be expressed as a relation among the independent variable and the successive values, i.e., $f(x)$, $f(x+h)$, $f(x+2h)$, $\ldots$, of dependent variable. For example, the difference equation

$$2\Delta^2 f(x) - \Delta f(x) + 5f(x) = x^2 + 3x, \tag{2.52}$$

is same as $2E^2f(x) - 5Ef(x) + 8f(x) = x^2 + 3x$,
or, it can be written as

$$2f(x+2h) - 5f(x+h) + 8f(x) = x^2 + 3x.$$

If $f(x)$ is denoted by u_x, then this equation can be written as

$$2u_{x+2h} - 5u_{x+h} + 8u_x = x^2 + 3x.$$

When $h = 1$, the above equation is simplified as

$$2u_{x+2} - 5u_{x+1} + 8u_x = x^2 + 3x. \tag{2.53}$$

The difference between the largest and the smallest arguments appearing in the difference equation with unit interval is called the **order** of the difference equation.

The order of the equation (2.53) is $(x+2) - x = 2$, while the order of the equation $u_{x+3} - 8u_{x+1} + 5u_{x-1} = x^3 + 2$ is $(x+3) - (x-1) = 4$. The order of the difference equation $\Delta f(x+2) - 3f(x) = 0$ is 3 as it is equivalent to $u_{x+3} - u_{x+2} - 3u_x = 0$.

A difference equation in which $u_x, u_{x+1}, \ldots, u_{x+n}$ occur to the first degree only and there are no product terms is called **linear difference equation**. Its general form is

$$a_0u_{x+n} + a_1u_{x+n-1} + \cdots + a_nu_x = g(x). \tag{2.54}$$

If the coefficients $a_0, a_1, \ldots, a_n$ are constants, then the equation (2.54) is called **linear difference equation with constant coefficients**. If $g(x) = 0$ then the equation is called **homogenous** otherwise it is called **non-homogeneous** difference equation.

The linear homogeneous equation with constant coefficients of order n is

$$a_0u_{x+n} + a_1u_{x+n-1} + \cdots + a_nu_x = 0. \tag{2.55}$$

This can be written as

$$(a_0E^n + a_1E^{n-1} + \cdots + a_n)u_x = 0 \qquad \text{or,} \qquad f(E)u_x = 0, \tag{2.56}$$

where $f(E) \equiv a_0E^n + a_1E^{n-1} + \cdots + a_n$ is known as the **characteristic function** of (2.56).

The equation $f(m) = 0$ is called the **auxiliary equation (A.E.)** for the difference equation (2.56).

The solution of a difference equation is a relation between the independent variable and the dependent variable satisfying the equation.

2.8.1 Formation of difference equations

Let $\{1, 2, 4, 8, 16, \ldots\}$ be a sequence and its general term is $2^n, n = 0, 1, 2, \ldots$. Let us denote the nth term by a_n. Then $a_n = 2^n$. Also, $a_{n+1} = 2^{n+1}$. Thus $a_{n+1} = 2.2^n = 2a_n$, i.e., $a_{n+1} - 2a_n = 0$ is the difference equation of the above sequence.

Let us consider another example of a sequence whose xth term is given by

$$u_x = a2^x + b3^x \tag{2.57}$$

where a, b are arbitrary constants.

Then,

$$u_{x+1} = a2^{x+1} + b3^{x+1} \tag{2.58}$$

and

$$u_{x+2} = a2^{x+2} + b3^{x+2}. \tag{2.59}$$

Now $u_{x+1} - 2u_x = b3^x$ [using (2.57) and (2.58)]
and $u_{x+2} - 2u_{x+1} = b3^{x+1}$. [using (2.58) and (2.59)]

Therefore, $u_{x+2} - 2u_{x+1} = 3(b3^x) = 3(u_{x+1} - 2u_x)$.

That is, $u_{x+2} - 5u_{x+1} + 6u_x = 0$ is the required difference equation for the relation (2.57).

It may be noted that the equation (2.57) contains two arbitrary constants and the corresponding difference equation is of order 2.

A large number of counting problems can be modelled by using recurrence relations. Some of them are presented here.

Example 2.8.1 (Rabbits on an island). This problem was originally posed by the Italian mathematician Leonardo Fibonacci in the thirteenth century in his book 'Liber abaci'. The problem is stated below.
A pair of new-born rabbits (one male and other female) is kept on an island where there is no other rabbit. A pair of rabbits does not breed until they are 2 months old. After a pair becomes 2 months old, each pair of rabbits (of opposite sexes) produces another pair (of opposite sexes) each month. It is assuming that no rabbits ever die. This problem can be modelled as a recurrence relation as follows.

Let x_n denote the number of pairs of rabbits on the island just after n months. At the end of first month the number of pairs of rabbits is 1, i.e., $x_1 = 1$. Since this pair does not breed during second month, so $x_2 = 1$. Now,

$$\begin{aligned} x_n = {} & \text{number of pairs of rabbits just after } n \text{ months} \\ = {} & \text{number of pairs after } (n-1) \text{ months} + \text{number of new born pairs at} \\ & \text{the end of } n\text{th month.} \end{aligned}$$

The number of new born pairs = number of pairs just after the $(n-2)$th month, since each new-born pair is produced by a pair of at least 2 months old, and it is equal to x_{n-2}.

Hence

$$x_n = x_{n-1} + x_{n-2}, \qquad n \geq 3, \qquad x_1 = x_2 = 1. \tag{2.60}$$

This is the difference equation of the above stated problem and the solution is $x_1 = 1, x_2 = 1, x_2 = 2, x_3 = 3, x_4 = 5, x_5 = 8, \ldots$, i.e., the sequence is $\{1, 1, 2, 3, 5, 8 \ldots\}$ and this sequence is known as **Fibonacci sequence**.

Example 2.8.2 (The Tower of Hanoi). The Tower of Hanoi problem is a famous problem of the late nineteenth century. The problem is stated below.
Let there be three pegs, numbered 1, 2, 3 and they are on a board and n discs of difference sizes with holes in their centres. Initially, these n discs are placed on one peg, say peg 1, in order of decreasing size, with the largest disc at the bottom. The rules of the puzzle are that the discs can be moved from one peg to another only one at a time and no discs can be placed on the top of a smaller disc. The problem is to transfer all the discs from peg 1 to another peg 2, in order of size, with the largest disc at the bottom, in minimum number of moves.

Let x_n be the number of moves required to solve the problem with n discs. If $n = 1$, i.e., if there is only one disc on peg 1, we simply transfer it to peg 2 by one move. Hence $x_1 = 1$. Now, if $n > 1$, starting with n discs on peg 1 we can transfer the top $n - 1$ discs, following the rules of this problem to peg 3 by x_{n-1} moves. During these moves the largest disc at the bottom on peg 1 remains fixed. Next, we use one move to transfer the largest disc from peg 1 to peg 2, which was empty. Finally, we again transfer the $n-1$ discs on peg 3 to peg 2 by x_{n-1} moves, placing them on top of the largest disc on peg 2 which remains fixed during these moves. Thus, when $n > 1$, $(n - 1)$ discs are transferred twice and one additional move is needed to move the largest disc at the bottom from peg 1 to peg 2. Thus the recurrence relation is

$$x_n = 2x_{n-1} + 1 \text{ for } n \geq 2 \text{ and } x_1 = 1. \tag{2.61}$$

This is a first order non-homogeneous difference equation.

2.9 Solution of Difference Equations

Several methods are used to solve difference equations. Among them the widely used methods are iterative method, solution using operators, solution using generating function, etc.

2.9.1 Iterative method

In this method the successive terms are substituted until the terms reduce to initial term. The method is illustrated by example.

Example 2.9.1 Solve the difference equation

$$x_n = x_{n-1} + (n-1) \text{ for } n \geq 2 \text{ and } x_1 = 0.$$

Solution.

$$\begin{aligned} x_n &= x_{n-1} + (n-1) = \{x_{n-2} + (n-2)\} + (n-1) \\ &= x_{n-2} + 2n - (1+2) = \{x_{n-3} + (n-3)\} + 2n - (1+2) \\ &= x_{n-3} + 3n - (1+2+3). \end{aligned}$$

In this way, after $(n-1)$ steps

$$\begin{aligned} x_n &= x_{n-(n-1)} + (n-1)n - (1 + 2 + \cdots + \overline{n-1}) \\ &= x_1 + (n-1)n - \frac{n(n-1)}{2} \\ &= 0 + \frac{1}{2}n(n-1) \text{ [since } x_1 = 0] \\ &= \frac{1}{2}n(n-1). \end{aligned}$$

Example 2.9.2 Solve the difference equation for the Tower of Hanoi problem: $x_n = 2x_{n-1} + 1, n \geq 2$ with $x_1 = 1$.

Solution.

$$\begin{aligned} x_n &= 2x_{n-1} + 1 = 2(2x_{n-2} + 1) + 1 \\ &= 2^2 x_{n-2} + (2+1) = 2^2(2x_{n-3} + 1) + (2+1) \\ &= 2^3 x_{n-3} + (2^2 + 2 + 1) = 2^3(2x_{n-4} + 1) + (2^2 + 2 + 1) \\ &= 2^4 x_{n-4} + (2^3 + 2^2 + 2 + 1). \end{aligned}$$

In this way, after $(n-1)$ steps,

$$\begin{aligned} x_n &= 2^{n-1} x_{n-(n-1)} + (2^{n-2} + 2^{n-3} + \cdots + 2^2 + 2 + 1) \\ &= 2^{n-1} x_1 + (2^{n-2} + 2^{n-3} + \cdots + 2^2 + 2 + 1) \\ &= 2^{n-1} + 2^{n-2} + 2^{n-3} + \cdots + 2^2 + 2 + 1 \qquad \text{[since } x_1 = 1] \\ &= 2^n - 1. \end{aligned}$$

2.9.2 Solution using symbolic operators

This method is used to solve homogeneous as well as non-homogeneous difference equations. First we consider the homogeneous linear difference equations with constant coefficients.

Solution of homogeneous equations with constant coefficients

To explain the method, a general second order difference equation is considered. Let

$$u_{x+2} + au_{x+1} + bu_x = 0. \tag{2.62}$$

Using shift operator E, this equation can be written as

$$(E^2 + aE + b)u_x = 0. \tag{2.63}$$

Let $u_x = cm^x$ be a solution of (2.63), c is a non-zero constant.

Then, $E^2u_x = u_{x+2} = cm^{x+2}$, $Eu_x = u_{x+1} = cm^{x+1}$.

Using these values, the equation (2.63) reduces to $cm^x(m^2 + am + b) = 0$. Since $cm^x \neq 0$,

$$m^2 + am + b = 0. \tag{2.64}$$

This equation is called the **auxiliary equation (A.E.)** for the difference equation (2.62). Since (2.64) is a quadratic equation, three types of roots may occur.

Case I. Let m_1 and m_2 be two distinct real roots of (2.64). In this case, the general solution is $u_x = c_1m_1^x + c_2m_2^x$, where c_1 and c_2 are arbitrary constants.

Case II. Let m_1, m_1 be two real and equal roots of (2.64). In this case $(c_1m_1^x + c_2m_1^x) = (c_1 + c_2)m_1^x = cm_1^x$ is the only one solution of (2.62). To get the other solution (as a second order difference equation should have two independent solutions, like differential equation), let us consider $u_x = m_1^xv_x$ be its solution.

Since m_1, m_1 are two equal roots of (2.64), the equation (2.63) may be written as $(E^2 - 2m_1E + m_1^2)u_x = 0$.

Substituting $u_x = m_1^xv_x$ to this equation, we obtain
$m_1^{x+2}u_{x+2} - 2m_1^{x+2}v_{x+1} + m_1^{x+2}v_x = 0$
or, $m_1^{x+2}(v_{x+2} - 2v_{x+1} + v_x) = 0$ or, $m_1^{x+2}\Delta^2v_x = 0$.

That is, $\Delta^2v_x = 0$. Since second difference is zero, the first difference is constant and hence v_x is linear. Let $v_x = c_1 + c_2x$, where c_1, c_2 are arbitrary constants.

Hence, in this case the general solution is

$$u_x = (c_1 + c_2x)m_1^x.$$

Case III. If the roots m_1, m_2 are complex, then m_1, m_2 should be conjugate complex and let them be $(\alpha + i\beta)$ and $(\alpha - i\beta)$, where α, β are reals. Then the general solution is

$$u_x = c_1(\alpha + i\beta)^x + c_2(\alpha - i\beta)^x.$$

To simplify the above expression, substituting $\alpha = r\cos\theta, \beta = r\sin\theta$, where $r = \sqrt{\alpha^2 + \beta^2}$ and $\tan\theta = \beta/\alpha$.

Therefore,

$$\begin{aligned}u_x &= c_1 r^x(\cos\theta + i\sin\theta)^x + c_2 r^x(\cos\theta - i\sin\theta)^x\\ &= r^x\{c_1(\cos\theta x + i\sin\theta x) + c_2(\cos\theta x - i\sin\theta x)\}\\ &= r^x\{(c_1+c_2)\cos\theta x + i(c_1-c_2)\sin\theta x\}\\ &= r^x(A\cos\theta x + B\sin\theta x),\ \text{where } A = c_1+c_2 \text{ and } B = i(c_1-c_2).\end{aligned}$$

Example 2.9.3 Solve $u_{x+1} - 8u_x = 0$.

Solution. This equation is written as $(E-8)u_x = 0$. Let $u_x = cm^2$ be a solution.
The A.E. is $m - 8 = 0$ or, $m = 8$.
Then $u_x = c8^x$, where c is an arbitrary constant, is the general solution.

Example 2.9.4 Solve the difference equation $u_x = u_{x-1} + u_{x-2}, x \geq 2$, $u_0 = 1, u_1 = 1$. Also, find the approximate value of u_x when x tends to a large number.

Solution. Let $u_x = cm^x$ be a solution. The A.E. is $m^2 - m - 1 = 0$
or, $m = \dfrac{1 \pm \sqrt{5}}{2}$.
Therefore, general solution is

$$u_x = c_1\left(\frac{1+\sqrt{5}}{2}\right)^x + c_2\left(\frac{1-\sqrt{5}}{2}\right)^x,$$

where c_1, c_2 are arbitrary constants.
Given that $u_0 = 1, u_1 = 1$, therefore,

$$1 = c_1 + c_2 \text{ and } 1 = c_1\left(\frac{1+\sqrt{5}}{2}\right) + c_2\left(\frac{1-\sqrt{5}}{2}\right).$$

Solution of these equations is

$$c_1 = \frac{\sqrt{5}+1}{2\sqrt{5}} \text{ and } c_2 = -\frac{1-\sqrt{5}}{2\sqrt{5}}.$$

Hence, the particular solution is

$$u_x = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{x+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{x+1}\right].$$

When $x \to \infty$ then $\left(\dfrac{1-\sqrt{5}}{2}\right)^{x+1} \to 0$ and therefore,

$$u_x \simeq \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{x+1}.$$

Solution of non-homogeneous difference equation

The general form of non-homogeneous linear difference equation with constant coefficients is

$$(a_0E^n + a_1E^{n-1} + a_2E^{n-2} + \cdots + a_n)u_x = g(x), \text{ or, } f(E)u_x = g(x).$$

The solution of this equation is the combination of two solutions – **complementary function (C.F.)** and **particular integral** or **particular solution (P.S.)**. The C.F. is the solution of the homogeneous equation $f(E)u_x = 0$ and the P.I. is given by $\dfrac{1}{f(E)}g(x)$. Then the general solution is

$$u_x = \text{C.F.}+\text{P.I.}$$

The method to find C.F. is discussed in previous section.

Rules to find particular integral

Case I. $g(x) = a^x, f(a) \neq 0$.
Since $f(E) = a_0E^n + a_1E^{n-1} + \cdots + a_n$,

$$\begin{aligned} f(E)a^x &= a_0a^{x+n} + a_1a^{x+n-1} + \cdots + a_na^x = a^x(a_0a^n + a_1a^{n-1} \cdots + a_n) \\ &= a^x f(a). \end{aligned}$$

Thus P.I. $= \dfrac{1}{f(E)}a^x = \dfrac{1}{f(a)}a^x$, provided $f(a) \neq 0$.

Case II. $g(x) = a^x\phi(x)$. Then,

$$\begin{aligned} f(E)a^x\phi(x) &= a_0E^na^x\phi(x) + a_1E^{n-1}a^x\phi(x) + \cdots + a_na^x\phi(x) \\ &= a^x[a_0a^n\phi(x+n) + a_1a^{n-1}\phi(x+n-1) + \cdots + a_n\phi(x)] \\ &= a^x[(a_0a^nE^n + a_1a^{n-1}E^{n-1} + \cdots + a_n)\phi(x)] \\ &= a^x f(aE)\phi(x). \end{aligned}$$

This gives P.I. $= \dfrac{1}{f(E)}a^x\phi(x) = a^x\dfrac{1}{f(aE)}\phi(x)$, where $f(aE) \neq 0$.

Case III. $g(x) = a^x$, $f(a) = 0$.
In this case, P.I. $= a^x\dfrac{1}{f(aE)}1$ [by Case II]

Case IV. $g(x) = x^m$ (m is zero or positive integer)
Then, P.I. $= \dfrac{1}{f(E)}x^m = \dfrac{1}{f(1+\Delta)}x^m$.

Now, this expression is evaluated by expanding $[f(1+\Delta)]^{-1}$ as an infinite series of Δ and applying different differences on x^m.

Case V. $g(x) = \sin ax$ or $\cos ax$.

(i) P.I. $= \dfrac{1}{f(E)} \sin ax =$ Imaginary part of $\dfrac{1}{f(E)} e^{iax}$.

(ii) P.I. $= \dfrac{1}{f(E)} \cos ax =$ Real part of $\dfrac{1}{f(E)} e^{iax}$.

Example 2.9.5 Solve the equation $u_{x+2} - 5u_{x+1} + 6u_x = 5^x + 2^x$.

Solution. The given equation is $(E^2 - 5E + 6)u_x = 5^x + 2^x$.
Let $u_x = cm^x$ be a solution of $(E^2 - 5E + 6)u_x = 0$. Therefore, A.E. is $m^2 - 5m + 6 = 0$ or, $m = 2, 3$.
Hence C.F. is $c_1 2^x + c_2 3^x$.

$$\text{P.I. of } 5^x = \frac{1}{E^2 - 5E + 6} 5^x = \frac{1}{25 - 25 + 6} 5^x = \frac{5^x}{6}.$$

$$\begin{aligned}
\text{P.I. of } 2^x &= \frac{1}{E^2 - 5E + 6} 2^x = 2^x \frac{1}{(2E)^2 - 5(2E) + 6} 1 \\
&= 2^x \frac{1}{4E^2 - 10E + 6} 1 = 2^x \frac{1}{4(1+\Delta)^2 - 10(1+\Delta) + 6} 1 \\
&= 2^x \frac{1}{4\Delta^2 - 2\Delta} 1 = 2^x \frac{1}{-2\Delta} (1 - 2\Delta)^{-1} 1 \\
&= 2^x \frac{1}{-2\Delta} 1 = -2^{x-1} x^{(1)} = -2^{x-1}.x
\end{aligned}$$

Therefore, the general solution is $u_x = c_1 2^x + c_2 3^x + \dfrac{5^x}{6} - 2^{x-1}.x$, where c_1, c_2 are arbitrary constants.

Example 2.9.6 Solve $u_{x+2} - 4u_{x+1} + 3u_x = 2^x.x^{(3)}$.

Solution. The equation can be written as $(E^2 - 4E + 3)u_x = 2^x x^{(3)}$. Let $u_x = cm^x$ be a solution. The A.E. is $m^2 - 4m + 3 = 0$ or, $m = 1, 3$.
Therefore, C.F. is $c_1 1^x + c_2 3^x = c_1 + c_2 3^x$.

$$\begin{aligned}
\text{P.I. } &= \frac{1}{E^2 - 4E + 3} 2^x x^{(3)} = 2^x \frac{1}{(2E)^2 - 4(2E) + 3} x^{(3)} \\
&= 2^x \frac{1}{4(1+\Delta)^2 - 8(1+\Delta) + 3} x^{(3)} \\
&= 2^x \frac{1}{4\Delta^2 - 1} x^{(3)} = -2^x (1 - 4\Delta^2)^{-1} x^{(3)} \\
&= -2^x (1 + 4\Delta^2 + 16\Delta^4 + \cdots) x^{(3)} \\
&= -2^x [x^{(3)} + 4.3.2x^{(1)}] = -2^x [x^{(3)} + 24x^{(1)}].
\end{aligned}$$

Hence, the general solution is $u_x = c_1 + c_2 3^x - 2^x [x^{(3)} + 24x^{(1)}]$.

Example 2.9.7 Solve $u_x - u_{x-1} + 2u_{x-2} = x^2 + 5^x$.

Solution. The given equation can be written as

$$(E^2 - E + 2)u_{x-2} = x^2 + 5^x.$$

Let $u_{x-2} = cm^x$ be a solution of $(E^2 - E + 2)u_{x-2} = 0$.

Then A.E. is $m^2 - m + 2 = 0$ or, $m = \dfrac{1 \pm i\sqrt{7}}{2}$.

Here the roots are complex. Let $\dfrac{1}{2} = r\cos\theta$ and $\dfrac{\sqrt{7}}{2} = r\sin\theta$.

Therefore, $r = \sqrt{2}, \tan\theta = \sqrt{7}$.

The C.F. is $(\sqrt{2})^x[c_1 \cos\theta x + c_2 \sin\theta x]$ where $\tan\theta = \sqrt{7}$.

$$\begin{aligned}
\text{P.I.} &= \frac{1}{E^2 - E + 2}(x^2 + 5^x) \\
&= \frac{1}{(1+\Delta)^2 - (1+\Delta) + 2}\{x^{(2)} + x^{(1)}\} + \frac{1}{25 - 5 + 2}5^x \\
&= \frac{1}{\Delta^2 + \Delta + 2}\{x^{(2)} + x^{(1)}\} + \frac{5^x}{22} \\
&= \frac{1}{2}\left(1 + \frac{\Delta^2 + \Delta}{2}\right)^{-1}\{x^{(2)} + x^{(1)}\} + \frac{5^x}{22} \\
&= \frac{1}{2}\left[1 - \frac{\Delta^2 + \Delta}{2} + \left(\frac{\Delta^2 + \Delta}{2}\right)^2 - \cdots\right]\{x^{(2)} + x^{(1)}\} + \frac{5^x}{22} \\
&= \frac{1}{2}\left[x^{(2)} + x^{(1)} - \frac{1}{2}(2 + 2x^{(1)} + 1) + \frac{1}{4}(2)\right] + \frac{5^x}{22} \\
&= \frac{1}{2}[x^{(2)} - 1] + \frac{5^x}{22} = \frac{1}{2}(x^2 - x - 1) + \frac{5^x}{22}.
\end{aligned}$$

Therefore, the general solution is

$$u_{x-2} = (\sqrt{2})^x[c_1 \cos\theta x + c_2 \sin\theta x] + \frac{1}{2}(x^2 - x - 1) + \frac{5^x}{22},$$

where $\theta = \sqrt{7}$ and c_1, c_2 are arbitrary constants.

Example 2.9.8 Show that the solution of the equation $u_{x+2} + u_x = 2\sin\dfrac{\pi x}{2}$ is given by $u_x = a\cos\left(\dfrac{\pi x}{2} + \varepsilon\right) - x\sin\dfrac{\pi x}{2}$.

Solution. Let $u_x = cm^x$ be a solution of $u_{x+2} + u_x = 0$.
Then A.E. is $m^2 + 1 = 0$ or, $m = \pm i$.
Therefore, C.F. is $A(i)^x + B(-i)^x$.
Substituting $0 = r\cos\theta, 1 = r\sin\theta$, where $r = 1, \theta = \pi/2$.

Then C.F. reduces to

$$\begin{aligned}
&A\{r(\cos\theta + i\sin\theta)\}^x + B\{r(\cos\theta - i\sin\theta)\}^x\\
&= A\Big(\cos\frac{\pi x}{2} + i\sin\frac{\pi x}{2}\Big) + B\Big(\cos\frac{\pi x}{2} - i\sin\frac{\pi x}{2}\Big)\\
&= (A+B)\cos\frac{\pi x}{2} - (B-A)i\sin\frac{\pi x}{2}\\
&= a\cos\varepsilon\cos\frac{\pi x}{2} - a\sin\varepsilon\sin\frac{\pi x}{2},\ \text{where } A+B = a\cos\varepsilon, (B-A)i = a\sin\varepsilon\\
&= a\cos\left(\frac{\pi x}{2} + \varepsilon\right).
\end{aligned}$$

$$\begin{aligned}
\text{P.I.} &= \frac{1}{E^2+1}2\sin\frac{\pi x}{2} = \text{ Imaginary part of } \frac{1}{E^2+1}2e^{i\pi x/2}\\
&= \text{ I.P. of } 2e^{i\pi x/2}\frac{1}{(e^{i\pi/2}E)^2+1}1 \qquad [\text{since } (e^{i\pi/2})^2+1=0]\\
&= \text{ I.P. of } 2e^{i\pi x/2}\frac{1}{e^{i\pi}E^2+1}1 = \text{ I.P. of } 2e^{i\pi x/2}\frac{1}{1-(1+\Delta)^2}1\\
&= \text{ I.P. of } 2e^{i\pi x/2}\frac{1}{-2\Delta}\left(1-\frac{\Delta}{2}+\cdots\right)1\\
&= \text{ I.P. of } 2e^{i\pi x/2}\frac{1}{-2\Delta}1 = \text{ I.P. of } 2e^{i\pi x/2}\frac{1}{-2}x\\
&= \text{ I.P. of } (-x)\Big(\cos\frac{\pi x}{2} + i\sin\frac{\pi x}{2}\Big)\\
&= -x\sin\frac{\pi x}{2}.
\end{aligned}$$

Therefore, the general solution is $u_x = a\cos\left(\frac{\pi x}{2} + \varepsilon\right) - x\sin\frac{\pi x}{2}$.

Example 2.9.9 Solve the following difference equation

$$y_{n+2} - 4y_{n+1} + 4y_n = n^2 + 3^n.$$

Solution. Let $y_n = cm^n$ be a trial solution of $(E^2 - 4E + 4)y_n = 0$.
Then A.E. is $m^2 - 4m + 4 = 0$ or, $m = 2, 2$.
Therefore, C.F. is $(c_1 + c_2n)2^n$.

P.I. of $3^n = \dfrac{1}{E^2-4E+4}3^n = \dfrac{1}{(E-2)^2}3^n = 3^n$.

To find the P.I. of n^2, let $y_n = an^2+bn+c$ be the solution of $y_{n+2}-4y_{n+1}+4y_n = n^2$.

Then $\{a(n+2)^2 + b(n+2) + c\} - 4\{a(n+1)^2 + b(n+1) + c\}$
$+4\{an^2 + bn + c\} = n^2$.

That is, $an^2 + (b - 4a)n + (c - 2b) = n^2$. Equating both sides we have
$a = 1$, $b - 4a = 0$, $c - 2b = 0$, i.e., $a = 1, b = 4, c = 8$.
Hence, the P.I. of n^2 is $n^2 + 4n + 8$.
Therefore, the general solution is

$$y_n = (c_1 + c_2 n)2^n + 3^n + n^2 + 4n + 8.$$

Example 2.9.10 Solve the difference equation $\Delta f(n) - 4f(n) = 3$, $n \geq 2$ and $f(1) = 2$.

Solution. The equation can be written using E operator as

$$(E - 1)f(n) - 4f(n) = 3 \qquad \text{or,} \qquad (E - 5)f(n) = 3.$$

Let $f(n) = cm^n$ be a trial solution.
The A.E. is $m - 5 = 0$. Therefore, C.F. is $c5^n$.
To find P.I. let, $f(n) = a$ be the solution of the given equation.
Then $(E - 5)a = 3$ or, $a - 5a = 3$ or, $a = -3/4$.
Hence, the general solution is

$$f(n) = c5^n - \frac{3}{4}.$$

But, given $f(1) = 2$. Therefore, $2 = 5c - 3/4$ or $c = 11/20$.
Hence the particular solution is

$$f(n) = \frac{11}{4}5^{n-1} - \frac{3}{4}.$$

2.9.3 Generating function

Generating function is used to solve different kind of problems including combinatorial problems. This function may also be used to solve difference equation.

Def. 2.9.1 *Let $\{a_0, a_1, a_2, \ldots\}$ be a sequence of real numbers. The power series*

$$G(x) = \sum_{n=0}^{\infty} a_n x^n \tag{2.65}$$

is called the generating function for the sequence $\{a_0, a_1, a_2, \ldots\}$.

In other words, a_n, the nth term of the sequence $\{a_n\}$ is the coefficient of x^n in the expansion of $G(x)$. That is, if the generating function of a sequence is known, then one can determine all the terms of the sequence.

Example 2.9.11 Use generating function to solve the difference equation

$$u_n = 2u_{n-1} + 3, n \geq 1, u_0 = 2.$$

Solution. From the definition of the generating function,

$$G(x) = \sum_{n=0}^{\infty} u_n x^n = u_0 + \sum_{n=1}^{\infty} u_n x^n = 2 + \sum_{n=1}^{\infty} u_n x^n.$$

That is, $\quad G(x) - 2 = \sum_{n=1}^{\infty} u_n x^n.$

Now, $u_n = 2u_{n-1} + 3, n \geq 1$. Multiplying both sides by x^n, we obtain

$$u_n x^n = 2u_{n-1} x^n + 3x^n.$$

Taking summation for all $n = 1, 2, \ldots$, we have

$$\sum_{n=1}^{\infty} u_n x^n = 2\sum_{n=1}^{\infty} u_{n-1} x^n + 3\sum_{n=1}^{\infty} x^n.$$

That is,

$$\begin{aligned}
G(x) - 2 &= 2x\sum_{n=1}^{\infty} u_{n-1} x^{n-1} + 3\sum_{n=1}^{\infty} x^n \\
&= 2x\sum_{n=0}^{\infty} u_n x^n + 3\Big(\sum_{n=0}^{\infty} x^n - 1\Big) \\
&= 2x\,G(x) + 3\Big(\frac{1}{1-x} - 1\Big) \qquad \Big[\text{since } \sum_{n=0}^{\infty} x^n = \frac{1}{1-x}\Big]
\end{aligned}$$

Thus, $\quad (1-2x)G(x) = \dfrac{3}{1-x} - 1.$

Therefore, the generating function for this difference equation or for the sequence $\{u_n\}$ is

$$\begin{aligned}
G(x) &= \frac{3}{(1-x)(1-2x)} - \frac{1}{1-2x} \\
&= \frac{5}{1-2x} - \frac{3}{1-x} = 5(1-2x)^{-1} - 3(1-x)^{-1} \\
&= 5\sum_{n=0}^{\infty}(2x)^n - 3\sum_{n=0}^{\infty} x^n = \sum_{n=0}^{\infty}(5.2^n - 3)x^n.
\end{aligned}$$

The coefficient of x^n in the expansion of $G(x)$ is $5.2^n - 3$ and hence $u_n = 5.2^n - 3$ is the required solution.

Example 2.9.12 Using generating function solve the difference equation $a_n - 5a_{n-1} + 6a_{n-2} = 0, n \geq 2$ with initial conditions $a_0 = 2$ and $a_1 = 3$.

Solution. Let $G(x)$ be the generating function of the sequence $\{a_n\}$. Then

$$G(x) = a_0 + a_1 x + \sum_{n=2}^{\infty} a_n x^n = 2 + 3x + \sum_{n=2}^{\infty} a_n x^n.$$

That is, $\displaystyle\sum_{n=2}^{\infty} a_n x^n = G(x) - 2 - 3x$.

Multiplying given equation by x^n,

$$a_n x^n - 5a_{n-1}x^n + 6a_{n-2}x^n = 0.$$

Taking summation for $n = 2, 3, \ldots,$

$$\begin{aligned}
&\sum_{n=2}^{\infty} a_n x^n - 5\sum_{n=2}^{\infty} a_{n-1}x^n + 6\sum_{n=2}^{\infty} a_{n-2}x^n = 0\\
\text{or,}\quad & G(x) - 2 - 3x - 5x\sum_{n=2}^{\infty} a_{n-1}x^{n-1} + 6x^2\sum_{n=2}^{\infty} a_{n-2}x^{n-2} = 0\\
\text{or,}\quad & G(x) - 2 - 3x - 5x\left(\sum_{n=0}^{\infty} a_n x^n - a_0\right) + 6x^2 G(x) = 0\\
\text{or,}\quad & G(x) - 2 - 3x - 5x[G(x) - 2] + 6x^2 G(x) = 0.
\end{aligned}$$

Therefore, $G(x) = \dfrac{2-7x}{1-5x+6x^2}$. This is the generating function for the given difference equation.

Let $$\frac{2-7x}{1-5x+6x^2} = \frac{A}{1-2x} + \frac{B}{1-3x} = \frac{(A+B)-(3A+2B)x}{(1-2x)(1-3x)}.$$

The unknown A and B are related by the equations

$$A + B = 2 \text{ and } 3A + 2B = 7,$$

whose solution is $A = 3, B = -1$. Thus, $G(x) = \dfrac{3}{1-2x} - \dfrac{1}{1-3x}$.

Now,

$$\begin{aligned}
G(x) &= 3(1-2x)^{-1} - (1-3x)^{-1}\\
&= 3\sum_{n=0}^{\infty}(2x)^n - \sum_{n=0}^{\infty}(3x)^n = \sum_{n=0}^{\infty}(3.2^n - 3^n)x^n.
\end{aligned}$$

Hence a_n = coefficient of x^n in the expansion of $G(x) = 3.2^n - 3^n$, which is the required solution.

2.10 Exercise

1. Define the operators: forward difference (Δ), backward difference (∇), shift (E), central difference (δ) and average (μ).

2. Prove the following relations among the operators
(i) $1+\delta^2\mu^2 \equiv \left(1+\dfrac{\delta^2}{2}\right)^2$, (ii) $E^{1/2} \equiv \mu + \dfrac{\delta}{2}$
(iii) $\mu\delta \equiv \dfrac{\Delta+\nabla}{2}$, (iv) $\mu\delta \equiv \dfrac{\Delta E^{-1}}{2} + \dfrac{\Delta}{2}$,
(v) $\Delta \equiv \dfrac{\delta^2}{2} + \delta\sqrt{1+(\delta^2/4)}$, (vi) $hD \equiv \sinh^{-1}(\mu\delta)$,
(vii) $hD \equiv \log(1+\Delta) \equiv -\log(1-\nabla) \equiv \sinh^{-1}(\mu\delta)$, (viii) $E \equiv e^{hD}$,
(ix) $\mu \equiv \left(1+\dfrac{\Delta}{2}\right)(1+\Delta)^{-1/2}$, (x) $\mu \equiv \cosh(hD/2)$,
(xi) $\nabla \equiv -\dfrac{\delta^2}{2} + \delta\sqrt{1+\dfrac{\delta^2}{4}}$, (xii) $E \equiv 1 + \dfrac{\delta^2}{2} + \delta\sqrt{1+\dfrac{\delta^2}{4}}$,
(xiii) $E\nabla \equiv \Delta \equiv \delta E^{1/2}$, (xiv) $\nabla \equiv E^{-1}\Delta$.

3. Show that
(i) $\Delta^i y_k = \nabla^i y_{k+i} = \delta^i y_{k+i/2}$, (ii) $\Delta(y_i^2) = (y_i + y_{i+1})\Delta y_i$,
(iii) $\Delta\nabla y_i = \nabla\Delta y_i = \delta^2 y_i$, (iv) $\Delta\left(\dfrac{1}{y_i}\right) = -\dfrac{\Delta y_i}{y_i y_{i+1}}$.

4. Prove the following
(i) $\Delta^n f(x) = \displaystyle\sum_{i=0}^{n} \frac{(-1)^i n! f(x+(n-i)h))}{i!(n-i)!}$
(ii) $\nabla^n f(x) = \displaystyle\sum_{i=0}^{n} \frac{(-1)^i n! f(x-ih)}{i!(n-i)!}$
(iii) $\delta^{2n} f(x) = \displaystyle\sum_{i=0}^{2n} \frac{(-1)^i (2n)! f(x+(n-i)h))}{i!(2n-i)!}$.

5. Prove the following
(i) $hD \equiv \Delta - \dfrac{\Delta^2}{2} + \dfrac{\Delta^3}{3} - \cdots$
(ii) $h^2D^2 \equiv \Delta^2 - \Delta^3 + 11\dfrac{\Delta^4}{12} - 5\dfrac{\Delta^5}{6} + \cdots$
(iii) $h^4D^4 \equiv \Delta^4 - 2\Delta^5 + 17\dfrac{\Delta^6}{6} - 7\dfrac{\Delta^7}{2} + \cdots$.

6. Prove that
(i) $\Delta^n(e^{ax+b}) = (e^{ah}-1)^n e^{ax+b}$

(ii) $e^x = \left(\dfrac{\Delta^2}{E}\right)e^x \dfrac{Ee^x}{\Delta^2 e^x}$
(iii) $\Delta \log f(x) = \log\left[1 + \dfrac{\Delta f(x)}{f(x)}\right]$
(iv) $\left(\dfrac{\Delta^2}{E}\right)x^3 = 6x$.

7. Prove that, if the spacing h is very small then the forward difference operator is almost equal to differential operator, i.e., for small h, $\Delta^n f(x) \approx h^n D^n f(x)$.

8. Show that the operators δ, μ, E, Δ and ∇ are commute with one another.

9. Express $\Delta^3 y_i$ and $\nabla^4 y_4$ in terms of y.

10. Prove the following relations:
(i) $u_x = u_{x-1} + \Delta u_{x-2} + \Delta^2 u_{x-3} + \cdots + \Delta^{n-1} u_{x-n} + \Delta^n u_{x-n-1}$
(ii) $u_1 + u_2 + u_3 + \cdots + u_n = ^n C_1 u_0 + ^n C_2 \Delta u_0 + ^n C_3 \Delta^2 u_0 + \cdots + \Delta^{n-1} u_0$
(iii) $\Delta^n y_x = y_{n+x} - ^n C_1 y_{x+n-1} + ^n C_2 y_{x+n-2} - \cdots + (-1)^n y_x$
(iv) $u_1 x + u_2 x^2 + u_3 x^3 + \cdots$
$$= \frac{x}{1-x}u_1 + \frac{x^2}{(1-x)^2}\Delta u_1 + \frac{x^3}{(1-x)^3}\Delta^2 u_1 + \cdots.$$

11. Show that
(i) $\Delta[f(x)g(x)] = f(x)\Delta g(x) + g(x+h)\Delta f(x)$
(ii) $\Delta^n f(x) = \nabla^n f(x+nh)$, where n is a positive integer
(iii) $\Delta\nabla f(x) = \Delta f(x) - \nabla f(x)$
(iv) $\Delta f(x) + \nabla f(x) = (\Delta/\nabla) f(x) - (\nabla/\Delta) f(x)$.

12. The nth difference Δ^n be defined as $\Delta^n f(x) = \Delta^{n-1} f(x+h) - \Delta^{n-1} f(x)$, $(n \geq 1)$. If $f(x)$ is a polynomial of degree n show that $\Delta f(x)$ is a polynomial of degree $n-1$. Hence deduce that the nth difference of $f(x)$ is a constant.

13. If $\phi_r(x) = (x - x_0)(x - x_1)\cdots(x - x_r)$ where $x_r = x_0 + rh$, $r = 0, 1, 2, \ldots, n$, calculate $\Delta^k \phi_r(x)$.

14. If f_i is the value of $f(x)$ at x_i where $x_i = x_0 + ih$, $i = 1, 2, \ldots$ prove that
$$f_i = E^i f_0 = \sum_{j=0}^{i} \binom{i}{j} \Delta^j f_0.$$

15. For equally spaced points $x_0, x_1, \ldots, x_n$, where $x_k = x_0 + kh$, $(h > 0, k = 0, 1, \ldots, n)$ express $\Delta^k y_0$ in terms of the ordinates.

16. Taking $h = 1$, compute the second, third and fourth differences of $f(x) = 3x^4 - 2x^2 + 5x - 1$.

17. Construct the forward difference table for the following tabulated values of $f(x)$ and hence find the values of $\Delta^2 f(3), \Delta^3 f(2), \Delta^4 f(0)$.

x	:	0	1	2	3	4	5	6
$f(x)$	:	4	7	10	20	45	57	70

18. Use finite difference method to find a polynomial which takes the following values:

x	:	–2	–1	0	1	2
$f(x)$	:	–12	–6	0	6	10

19. Compute the missing term in the following table.

x	:	0	2	4	6	8
$f(x)$	:	12	6	0	?	–25

20. Use finite difference method to find the value of $f(2.2)$ from the following data.

x	:	1	2	3	4	5
$f(x)$	:	3	24	99	288	675

21. Find the functions, whose first differences are
(i) $3x^2 + 9x + 2$, (ii) $x^4 - 3x^3 + x^2 - 11x + 20$.

22. Use iteration method to solve the following difference equations
(i) $a_n = a_{n-1} + 4$, for all $n \geq 2$ with $a_1 = 2$.
(ii) $u_n = u_{n-1} + (n-1)$, $n \geq 2$ and $a_1 = 0$.
(iii) $x_n = 5x_{n-1} + 3$ for $n \geq 2$ and $x_1 = 2$.

23. Find the first five terms of the sequence defined by the following recurrence relations:
(i) $x_n = x_{n-1}^2$ for $n \geq 2$ and $x_1 = 1$.
(ii) $x_n = nx_{n-1} + n^2 x_{n-2}$ for $n \geq 2$ and $x_0 = 1, x_1 = 1$.
(iii) Let $x_1 = 1$ and for $n \geq 2$, $x_n = x_1 x_{n-1} + x_2 x_{n-2} + \cdots + x_{n-1} x_1$.
(The numbers of this sequence are called Catalan numbers).

24. Mr. Das deposits Rs. 1,000 in a bank account yielding 5% compound interest yearly.
(i) Find a recurrence relation for the amount in the account after n years.
(ii) Find an explicit formula for the amount in the account after n years.
(iii) How much will be in the account after 10 years ?

25. Solve the following difference equations.
(i) $u_n - 5u_{n-1} + 6u_{n-2} = n^2 + 7^n + 3^n$
(ii) $u_{x+2} - 7u_{x-1} + 10u_x = 12e^{3x} + 4^x$
(iii) $u_{n+2} - 4u_{n+1} + 4u_n = n$ for $n \geq 1$ and $u_1 = 1, u_2 = 4$
(iv) $u_n - 5u_{n-1} + 6u_{n-2} = n^2 + 5^n + 2^n$
(v) $6u_{n+2} - 7u_{n+1} - 20u_n = 3n^2 - 2n + 8$
(vi) $f(n+2) - 8f(n+1) + 25f(n) = 2n^2 + n + 1$
(vii) $u_x - u_{x-1} + 2u_{x-2} = x + 2^x$
(viii) $y_{n+2} - 4y_{n+1} + 4y_n = n + 3^n$
(ix) $u_n - 5u_{n-1} + 6u_{n-2} = n^2$
(x) $S_{n+1} = S_n + S_{n-1}$, $n \geq 3$ and $S_1 = 1, S_2 = 1$. Find S_8.

26. Assuming $u_n = an + b$, show that the particular solution of
$u_n - 5u_{n-1} + 6u_{n-2} = n$ is $\frac{1}{4}(2n+7)$.

27. Show that the general solution of $u_x - u_{x-1} - u_{x-2} = x^2$
is $\frac{1}{2^x}[A(1+\sqrt{5})^x + B(1-\sqrt{5})^x] - (x^2 + 6x + 13)$.

28. Show that the solution of $u_{x+2} + a^2 u_x = \cos ax$ is
$u_x = a^x\left(A\cos\frac{\pi x}{2} + B\sin\frac{\pi x}{2}\right) + \frac{a^2\cos ax + \cos a(2-x)}{1 + 2a^2\cos 2a + a^4}$.

29. If u_n satisfies the difference equation $u_n = 11u_{n-1} + 8u_{n-2}, n \geq 2$ where $u_0 = 1$ and $u_1 = 11$ then show that u_n is given by $u_n = \frac{1}{a}\left\{\left(\frac{11+a}{2}\right)^{n+1} - \left(\frac{11-a}{2}\right)^{n+1}\right\}$ where $a = 3\sqrt{17}$.

30. The seeds of a certain plant when one year old produce eighteen fold. A seed is planted and every seed subsequently produced is planted as soon as it is produced. Prove that the number of grain at the end of nth year is

$$u_n = \frac{1}{a}\left\{\left(\frac{11+a}{2}\right)^{n+1} - \left(\frac{11-a}{2}\right)^{n+1}\right\}$$

where $a = 3\sqrt{17}$.

31. The first term of a sequence $\{u_n\}$ is 1, the second is 4 and every other term is the arithmetic mean of the two preceding terms. Find u_n and show that u_n tends to a definite limit as $n \to \infty$.

32. The first term of a sequence is 1, the second is 2 and every term is the sum of the two proceeding terms. Find the nth term.

33. If u_r satisfies the difference equation $u_r - 4u_{r-1} + u_{r-2} = 0, 2 \leq r \leq n$, where $u_n = 0$ and $u_0 = A$, show that, if $\alpha = \log(2+\sqrt{3})$ then $u_r = \dfrac{A\sinh(n-r)\alpha}{\sinh n\alpha}$.

34. Show that the general solution of the national income equation $y_{n+2} - \dfrac{1}{2}y_{n+1} - \dfrac{1}{4}y_n = nh + A$ where h, A are constants, is given by $y_n = c_1 m_1^n + c_2 m_2^n + 4hn + 4(A - 6h)$ where c_1, c_2 are arbitrary constants and the values of m_1 and m_2 you are to actually find out. Also show that y_n/n tends to finite limit as $n \to \infty$.

35. Use generating functions to solve the following difference equations.
(i) $x_n = 3x_{n-1}$, $n \geq 1$ and $x_0 = 2$.
(ii) $x_n = 5x_{n-1} + 2^n$, $n \geq 1$ and $x_0 = 2$.
(iii) $x_n = x_{n-1} + n$ for $n \geq 1$ and $x_0 = 1$.
(iv) $u_n - u_{n-1} - u_{n-2} = 0$ for $n \geq 2$ and $u_0 = 0, u_1 = 1$.
(v) $a_n + a_{n-2} = 2a_{n-1}$, $n \geq 2$ and $a_0 = 0, a_1 = 1$.

Chapter 3

Interpolation

Sometimes we have to compute the value of the dependent variable for a given independent variable, but the explicit relation between them is not known. For example, the Indian population are known to us for the years 1951, 1961, 1971, 1981, 1991 and 2001. There is no exact mathematical expression available which will give the population for any given year. So one can not determine the population of India in the year 2000 analytically. But, using interpolation one can determine the population (obviously approximate) for any year.

The general interpolation problem is stated below:
Let $y = f(x)$ be a function whose analytic expression is not known, but a table of values of y is known only at a set of values x_0, x_1, x_2, ..., x_n of x. There is no other information available about the function $f(x)$. That is,

$$f(x_i) = y_i,\ i = 0, 1, \ldots, n. \tag{3.1}$$

The problem of interpolation is to find the value of $y(= f(x))$ for an argument, say, x'. The value of y at x' is not available in the table.

A large number of different techniques are used to determine the value of y at $x = x'$. But one common step is "*find an approximate function, say, $\psi(x)$, corresponding to the given function $f(x)$ depending on the tabulated value.*" The approximate function should be simple and easy to handle. The function $\psi(x)$ may be a polynomial, exponential, geometric function, Taylor's series, Fourier series, etc. When the function $\psi(x)$ is a polynomial, then the corresponding interpolation is called polynomial interpolation. The polynomial interpolation is widely used interpolation technique, because, polynomials are continuous and can be differentiated and integrated term by term within any range.

A polynomial $\phi(x)$ is called **interpolating polynomial** if $y_i = f(x_i) = \phi(x_i), i = 0, 1, 2, \ldots, n$ and $\left.\dfrac{d^k f}{dx^k}\right)_{x'} = \left.\dfrac{d^k \phi}{dx^k}\right)_{x'}$ for some finite k, and x' is one of the values of x_0,

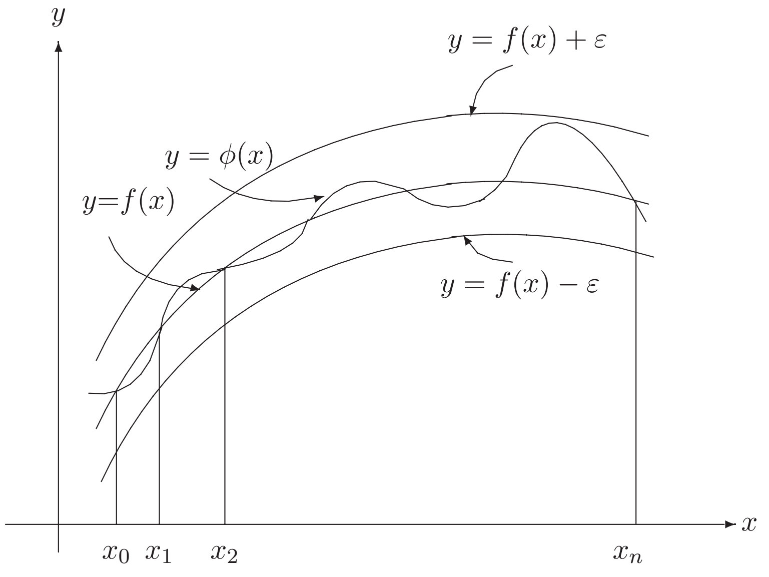


Figure 3.1: Interpolation of a function.

x_1, ..., x_n.

The following theorem justifies the approximation of an unknown function $f(x)$ to a polynomial $\phi(x)$.

Theorem 3.1 *If the function $f(x)$ is continuous in $[a, b]$, then for any pre-assigned positive number $\varepsilon > 0$, there exists a polynomial $\phi(x)$ such that*

$$|f(x) - \phi(x)| < \varepsilon \text{ for all } x \in (a, b).$$

This theorem ensures that the interpolating polynomial $\phi(x)$ is bounded by $y = f(x) - \varepsilon$ and $y = f(x) + \varepsilon$, for a given ε. This is shown in Figure 3.1.

Depending on the tabulated points, several interpolation methods are developed. Among them finite-difference interpolating formulae, Lagrange's interpolation are widely used polynomial interpolation methods.

For the sake of convenience, a polynomial of degree n, means a polynomial of degree not higher than n.

3.1 Lagrange's Interpolation Polynomial

Let $y = f(x)$ be a real valued function defined on an interval $[a, b]$. Let $x_0, x_1, \ldots, x_n$ be $n + 1$ distinct points in the interval $[a, b]$ and $y_0, y_1, \ldots, y_n$ be the corresponding values of y at these points, i.e., $y_i = f(x_i), i = 0, 1, \ldots, n$, are given.

Now, we construct an algebraic polynomial $\phi(x)$ of degree less than or equal to n which attains the assigned values at the points x_i, that is,

$$\phi(x_i) = y_i, \quad i = 0, 1, \ldots, n. \tag{3.2}$$

The polynomial $\phi(x)$ is called the **interpolation polynomial** and the points x_i, $i = 0, 1, \ldots, n$ are called **interpolation points**.

Let the polynomial $\phi(x)$ be of the form

$$\phi(x) = \sum_{i=0}^{n} L_i(x)\, y_i, \tag{3.3}$$

where each $L_i(x)$ is polynomial in x, of degree less than or equal to n, called the **Lagrangian function**.

The polynomial $\phi(x)$ satisfies the equation (3.2) if

$$L_i(x_j) = \begin{cases} 0, & \text{for } i \neq j \\ 1, & \text{for } i = j. \end{cases}$$

That is, the polynomial $L_i(x)$ vanishes only at the points $x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. So it should be of the form

$$L_i(x) = a_i(x - x_0)(x - x_1)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n),$$

where a_i, a constant whose value is determined by using the relation

$$L_i(x_i) = 1.$$

Then $a_i(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n) = 1$.
or, $a_i = 1/\{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)\}$.
Therefore,

$$L_i(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)}. \tag{3.4}$$

Thus, the required Lagrange's interpolation polynomial $\phi(x)$ is

$$\phi(x) = \sum_{i=0}^{n} L_i(x)\, y_i,$$

where $L_i(x)$ is given in (3.4).

The polynomial $L_i(x)$ can be written as

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j}\right).$$

In this notation, the polynomial $\phi(x)$ is

$$\phi(x) = \sum_{i=0}^{n} \prod_{\substack{j=0 \\ j\neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j}\right) y_i.$$

The function $L_i(x)$ can also be expressed in another form as follows.

Let

$$w(x) = (x - x_0)(x - x_1)\cdots(x - x_n) \tag{3.5}$$

be a polynomial of degree $n+1$ and vanishes at $x = x_0, x_1, \ldots, x_n$.

Now, the derivative of $w(x)$ with respect to x is given by

$$\begin{aligned} w'(x) &= (x - x_1)(x - x_2)\cdots(x - x_n) + (x - x_0)(x - x_2)\cdots(x - x_n) \\ &\quad + \cdots + (x - x_0)(x - x_1)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n) \\ &\quad + \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1}). \end{aligned}$$

Therefore, $w'(x_i) = (x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)$, which is the denominator of $L_i(x)$.

Using $w(x)$, $L_i(x)$ becomes

$$L_i(x) = \frac{w(x)}{(x - x_i)w'(x_i)}.$$

So the Lagrange's interpolation polynomial in terms of $w(x)$ is

$$\phi(x) = \sum_{i=0}^{n} \frac{w(x)}{(x - x_i)w'(x_i)} y_i. \tag{3.6}$$

Example 3.1.1 Obtain Lagrange's interpolating polynomial for $f(x)$ and find an approximate value of the function $f(x)$ at $x = 0$, given that $f(-2) = -5, f(-1) = -1$ and $f(1) = 1$.

Solution. Here $x_0 = -2, x_1 = -1, x_2 = 1$ and $f(x_0) = -5, f(x_1) = -1, f(x_2) = 1$.

Then $f(x) \simeq \displaystyle\sum_{i=0}^{2} L_i(x) f(x_i)$.

Now,

$$\begin{aligned} L_0(x) &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x+1)(x-1)}{(-2+1)(-2-1)} = \frac{x^2 - 1}{3}. \\ L_1(x) &= \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(x+2)(x-1)}{(-1+2)(-1-1)} = \frac{x^2 + x - 2}{-2}. \\ L_2(x) &= \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{(x+2)(x+1)}{(1+2)(1+1)} = \frac{x^2 + 3x + 2}{6}. \end{aligned}$$

Therefore,

$$\begin{aligned} f(x) &\simeq \frac{x^2-1}{3}\times(-5)+\frac{x^2+x-2}{-2}\times(-1)+\frac{x^2+3x+2}{6}\times 1 \\ &= 1+x-x^2. \end{aligned}$$

Thus, $f(0)=1$.

The Lagrangian coefficients can be computed from the following scheme. The differences are computed, row-wise, as shown below:

$$\begin{array}{lllll} \mathbf{x-x_0}^* & x_0-x_1 & x_0-x_2 & \cdots & x_0-x_n \\ x_1-x_0 & \mathbf{x-x_1}^* & x_1-x_2 & \cdots & x_1-x_n \\ x_2-x_0 & x_2-x_1 & \mathbf{x-x_2}^* & \cdots & x_2-x_n \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_n-x_0 & x_n-x_1 & x-x_2 & \cdots & \mathbf{x-x_n}^* \end{array}$$

From this table, it is observed that the product of diagonal elements is $w(x)$. The product of the elements of first row is $(x-x_0)w'(x_0)$, product of elements of second row is $(x-x_1)w'(x_1)$ and so on. Then the Lagrangian coefficient can be computed using the formula

$$L_i(x)=\frac{w(x)}{(x-x_i)w'(x_i)}.$$

Linear Lagrangian Interpolation

Let x_0 and x_1 be two points and y_0 and y_1 be the corresponding values of y. In this case,

$$\begin{aligned} \phi(x) &= L_0(x)y_0+L_1(x)y_1 \\ &= \frac{x-x_1}{x_0-x_1}y_0+\frac{x-x_0}{x_1-x_0}y_1 = y_0+\frac{x-x_0}{x_1-x_0}(y_1-y_0). \end{aligned} \qquad (3.7)$$

This polynomial is known as **linear interpolation polynomial**.

3.1.1 Lagrangian interpolation formula for equally spaced points

For the equally spaced points, $x_i=x_0+ih$, $i=0,1,2\ldots,n$, where h is the spacing. Now, a new variable s is introduced, defined by $x=x_0+sh$.

Then $x-x_i=(s-i)h$ and $x_i-x_j=(i-j)h$.
Therefore,

$$\begin{aligned} w(x) &= (x-x_0)(x-x_1)\cdots(x-x_n) = sh(s-1)h(s-2)h\cdots(s-n)h \\ &= h^{n+1}s(s-1)(s-2)\cdots(s-n). \end{aligned}$$

Also,

$$\begin{aligned} w'(x_i) &= (x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})(x_i - x_{i+2})\cdots(x_i - x_n) \\ &= (ih)(i-1)h\cdots(i - \overline{i-1})h(i - \overline{i+1})h(i - \overline{i+2})h\cdots(i-n)h \\ &= h^n i(i-1)\cdots 1\cdot(-1)(-2)\cdots(\{-(n-i)\} \\ &= h^n i!(-1)^{n-i}(n-i)!. \end{aligned}$$

Using these values, the relation (3.5) becomes

$$\begin{aligned} \phi(x) &= \sum_{i=0}^{n} \frac{h^{n+1}s(s-1)(s-2)\cdots(s-n)}{(-1)^{n-i}h^n i!(n-i)!(s-i)h} y_i \\ &= \sum_{i=0}^{n} (-1)^{n-i} \frac{s(s-1)(s-2)\cdots(s-n)}{i!(n-i)!(s-i)} y_i, \qquad (3.8) \\ &\quad \text{where } x = x_0 + sh. \end{aligned}$$

For given tabulated values, the Lagrange's interpolation polynomial exists and unique. These are proved in the following theorem.

Theorem 3.2 *The Lagrange's interpolation polynomial exists and unique.*

Proof. The Lagrange's interpolation formula satisfied the condition

$$y_i = \phi(x_i), i = 0, 1, \ldots, n. \qquad (3.9)$$

For $n = 1$,

$$\phi(x) = \frac{x - x_1}{x_0 - x_1} y_0 + \frac{x - x_0}{x_1 - x_0} y_1. \qquad (3.10)$$

For $n = 2$,

$$\begin{aligned} \phi(x) &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} y_0 + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} y_1 \\ &\quad + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} y_2. \qquad (3.11) \end{aligned}$$

In general, for any positive integer n,

$$\phi(x) = \sum_{i=0}^{n} L_i(x) y_i, \qquad (3.12)$$

where

$$L_i(x) = \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}, \quad i=0,1,\ldots,n. \tag{3.13}$$

Expression (3.10) is a linear function, i.e., a polynomial of degree one and also, $\phi(x_0)=y_0$ and $\phi(x_1)=y_1$.

Also, expression (3.11) is a second degree polynomial and $\phi(x_0)=y_0, \phi(x_1)=y_1, \phi(x_2)=y_2$, i.e., satisfy (3.9). Thus, the condition (3.13) for $n=1,2$ is fulfilled.

The functions (3.13) expressed in the form of a fraction whose numerator is a polynomial of degree n and whose denominator is a non-zero number. Also, $L_i(x_i)=1$ and $L_i(x_j)=0$ for $j\neq i, j=0,1,\ldots,n$. That is, $\phi(x_i)=y_i$. Thus, the conditions of (3.9) are satisfied. Hence, the Lagrange's polynomial exists.

Uniqueness of the polynomial

Let $\phi(x)$ be a polynomial of degree n, where

$$\phi(x_i)=y_i, i=0,1,\ldots,n. \tag{3.14}$$

Also, let $\phi^*(x)$ be another polynomials of degree n satisfying the conditions

$$\phi^*(x_i)=y_i, i=0,1,\ldots,n. \tag{3.15}$$

Then from (3.14) and (3.15),

$$\phi^*(x_i)-\phi(x_i)=0, \quad i=0,1,\ldots,n. \tag{3.16}$$

If $\phi^*(x)-\phi(x)\neq 0$, then this difference is a polynomial of degree at most n and it has at most n zeros, which contradicts (3.16), whose number of zeros is $n+1$. Consequently, $\phi^*(x)=\phi(x)$. Thus $\phi(x)$ is unique.

3.2 Properties of Lagrangian Functions

1. *The Lagrangian functions depend only on x_i's and independent of y_i's or $f(x_i)$'s.*

2. *The form of Lagrangian functions remain unchanged (invariant) under linear transformation.*

Proof. Let $x=az+b$, where a,b are arbitrary constants.

Then $x_j=az_j+b$ and $x-x_j=a(z-z_j)$. Also, $x_i-x_j=a(z_i-z_j) \qquad (i\neq j)$.

Therefore,

$$\begin{aligned}
w(x) &= (x-x_0)(x-x_1)\cdots(x-x_n)\\
&= a^{n+1}(z-z_0)(z-z_1)\cdots(z-z_n).\\
w'(x_i) &= (x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)\\
&= a^n(z_i-z_0)(z_i-z_1)\cdots(z_i-z_{i-1})(z_i-z_{i+1})\cdots(z_i-z_n).
\end{aligned}$$

Thus,

$$\begin{aligned}
L_i(x) &= \frac{w(x)}{(x-x_i)w'(x_i)}\\
&= \frac{a^{n+1}(z-z_0)(z-z_1)\cdots(z-z_n)}{a(z-z_i)a^n(z_i-z_0)(z_i-z_1)\cdots(z_i-z_{i-1})(z_i-z_{i+1})\cdots(z_i-z_n)}\\
&= \frac{w(z)}{(z-z_i)w'(z_i)} = L_i(z).
\end{aligned}$$

Thus $L_i(x)$'s are invariant.

3. *Sum of Lagrangian functions is 1, i.e.,* $\sum_{i=0}^{n} L_i(x) = 1$.

Proof. Sum of Lagrangian functions is

$$\sum_{i=0}^{n} L_i(x) = \sum_{i=0}^{n} \frac{w(x)}{(x-x_i)w'(x_i)} \tag{3.17}$$

where $w(x) = (x-x_0)(x-x_1)\cdots(x-x_n)$.

Let

$$\frac{1}{w(x)} = \frac{A_0}{x-x_0} + \frac{A_1}{x-x_1} + \cdots + \frac{A_i}{x-x_i} + \cdots + \frac{A_n}{x-x_n} \tag{3.18}$$

$$\begin{aligned}
\textit{i.e.},\ 1 &= A_0(x-x_1)(x-x_2)\cdots(x-x_n) + A_1(x-x_0)(x-x_2)\cdots(x-x_n)\\
&\quad + \cdots + A_i(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)\\
&\quad + \cdots + A_n(x-x_0)(x-x_1)\cdots(x-x_{n-1}).
\end{aligned} \tag{3.19}$$

When $x = x_0$ is substituted in (3.19) then the value of A_0 is given by

$$1 = A_0(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)$$

That is,

$$A_0 = \frac{1}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)} = \frac{1}{w'(x_0)}.$$

Similarly, $x = x_1$ gives

$A_1 = \dfrac{1}{w'(x_1)}$.

Also, $A_i = \dfrac{1}{w'(x_i)}$ and $A_n = \dfrac{1}{w'(x_n)}$.

Using these results, equation (3.18) becomes

$$\begin{aligned}\frac{1}{w(x)} &= \frac{1}{(x-x_0)w'(x_0)} + \frac{1}{(x-x_1)w'(x_1)} + \cdots \\ &\quad + \frac{1}{(x-x_i)w'(x_i)} + \cdots + \frac{1}{(x-x_n)w'(x_n)} \\ \textit{i.e.,}\ \ 1 &= \sum_{i=0}^{n} \frac{w(x)}{(x-x_i)w'(x_i)}.\end{aligned}$$

Hence the sum of Lagrangian functions is 1, that is,

$$\sum_{i=0}^{n} L_i(x) = \sum_{i=0}^{n} \frac{w(x)}{(x-x_i)w'(x_i)} = 1. \tag{3.20}$$

3.3 Error in Interpolating Polynomial

It is obvious that if $f(x)$ is approximated by a polynomial $\phi(x)$, then there should be some error at the non-tabular points. The following theorem gives the amount of error in interpolating polynomial.

Theorem 3.3 *Let I be an interval contains all interpolating points $x_0, x_1, \ldots, x_n$. If $f(x)$ is continuous and have continuous derivatives of order $n+1$ for all x in I then the error at any point x is given by*

$$E_n(x) = (x-x_0)(x-x_1)\cdots(x-x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}, \tag{3.21}$$

where $\xi \in I$.

Proof. Let the error $E_n(x) = f(x) - \phi(x)$, where $\phi(x)$ is a polynomial of degree less than or equal to n, which approximates the function $f(x)$.

Now, $E_n(x_i) = f(x_i) - \phi(x_i) = 0$ for $i = 0, 1, \ldots, n$.

By virtue of the above result, it is assumed that $E_n(x) = w(x)k$, where $w(x) = (x-x_0)(x-x_1)\ldots(x-x_n)$.

The error at any point, say, $x = t$, other than $x_0, x_1, \ldots, x_n$ is $E_n(t) = w(t)k$

$$\text{or,}\ \ f(t) - \phi(t) = kw(t). \tag{3.22}$$

Let us construct an auxiliary function

$$F(x) = f(x) - \phi(x) - kw(x). \tag{3.23}$$

The function vanishes at $x = x_0, x_1, \ldots, x_n$ because $f(x_i) = \phi(x_i)$ and $w(x_i) = 0$. Also, $F(t) = 0$, by (3.22).

Hence, $F(x) = 0$ has $n+2$ roots in I. By Roll's theorem, $F'(x) = 0$ has $n+1$ roots in I. $F''(x) = 0$ has n roots in I and finally, $F^{(n+1)}(x) = 0$ must have at least one root in I. Let ξ be one such root. Then $F^{(n+1)}(\xi) = 0$. That is, $f^{(n+1)}(\xi) - 0 + k(n+1)! = 0$ [$\phi(x)$ is a polynomial of degree n so $\phi^{(n+1)}(x) = 0$ and $w(x)$ is a polynomial of degree $n+1$, so $w^{(n+1)}(x) = (n+1)!$]. Thus

$$k = \frac{f^{(n+1)}(\xi)}{(n+1)!}.$$

Therefore, the error at $x = t$ is

$$\begin{aligned} E_n(t) &= kw(t) \quad [\text{by } (3.22)] \\ &= (t - x_0)(t - x_1) \cdots (t - x_n) \frac{f^{(n+1)}(\xi)}{(n+1)!}. \end{aligned}$$

Hence, the error at any point x is

$$E_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n) \frac{f^{(n+1)}(\xi)}{(n+1)!}. \qquad \square$$

Note 3.3.1 The above expression gives the error at any point x.

But practically it has little utility, because, in many cases $f^{(n+1)}(\xi)$ cannot be determined.

If M_{n+1} be the upper bound of $f^{(n+1)}(\xi)$ in I, i.e., if $|f^{(n+1)}(\xi)| \le M_{n+1}$ in I then the upper bound of $E_n(x)$ is

$$|E_n(x)| \le \frac{M_{n+1}}{(n+1)!} |w(x)|. \tag{3.24}$$

Note 3.3.2 (Error for equispaced points)
Let $x_i = x_0 + ih$, $i = 0, 1, \ldots, n$ and $x = x_0 + sh$ then $x - x_i = (s - i)h$.

Then the error is

$$E_n(x) = s(s-1)(s-2) \cdots (s-n) h^{n+1} \frac{f^{(n+1)}(\xi)}{(n+1)!}. \tag{3.25}$$

Note 3.3.3 (Error bounds for equally spaced points, particular cases)
Assume that, $f(x)$ is defined on $[a, b]$ that contains the equally spaced points. Suppose, $f(x)$ and the derivatives up to $n+1$ order are continuous and bounded on the intervals $[x_0, x_1], [x_0, x_2]$ and $[x_0, x_3]$ respectively. That is, $|f^{(n+1)(\xi)}| \leq M_{n+1}$ for $x_0 \leq \xi \leq x_n$, for $n = 1, 2, 3$. Then

$$(i)\ \ |E_1(x)| \leq \frac{h^2\, M_2}{8}, \qquad x_0 \leq x \leq x_1 \tag{3.26}$$

$$(ii)\ \ |E_2(x)| \leq \frac{h^3\, M_3}{9\sqrt{3}}, \qquad x_0 \leq x \leq x_2 \tag{3.27}$$

$$(iii)\ \ |E_3(x)| \leq \frac{h^4\, M_4}{24}, \qquad x_0 \leq x \leq x_3. \tag{3.28}$$

Proof. (i) From (3.25),

$$|E_1(x)| = |s(s-1)|h^2 \frac{|f^{(2)}(\xi)|}{2!}.$$

Let $g_1(s) = s(s-1)$. $g_1'(s) = 2s - 1$. Then $s = 1/2$, which is the solution of $g_1'(s) = 0$. The extreme value of $g_1(s)$ is $1/4$.

Therefore,

$$|E_1(x)| \leq \frac{1}{4} h^2 \frac{M_2}{2!} = \frac{h^2 M_2}{8}.$$

(ii) $|E_2(x)| = |s(s-1)(s-2)|h^3 \dfrac{|f^{(3)}(\xi)|}{3!}$.

Let $g_2(s) = s(s-1)(s-2)$. Then $g_2'(s) = 3s^2 - 6s + 2$. At $g_2'(s) = 0, s = 1 \pm \dfrac{1}{\sqrt{3}}$.

Again $g_2''(s) = 6(s-1) < 0$ at $s = 1 - \frac{1}{\sqrt{3}}$.

Therefore, the maximum value of $g_2(s)$ is

$$\Big(1 - \frac{1}{\sqrt{3}}\Big)\Big(-\frac{1}{\sqrt{3}}\Big)\Big(-\frac{1}{\sqrt{3}} - 1\Big) = \frac{2}{3\sqrt{3}}.$$

Thus,

$$|E_2(x)| \leq \frac{2}{3\sqrt{3}} h^3 \frac{M_3}{6} = \frac{h^3 M_3}{9\sqrt{3}}.$$

(iii) $|E_3(x)| = |s(s-1)(s-2)(s-3)|h^4 \dfrac{|f^{(4}(\xi)|}{4!}$.

Let $g_3(s) = s(s-1)(s-2)(s-3)$. Then $g_3'(s) = 4s^3 - 18s^2 + 22s - 6$.

At extrema, $g_3'(s) = 0$, i.e., $2s^3 - 9s^2 + 11s - 3 = 0$.

This gives $s = \dfrac{3}{2},\ \dfrac{3 \pm \sqrt{5}}{2}$.

$g_3''(s) = 6s^2 - 18s + 11$. Then $g_3''(3/2) < 0$ and $g_3''\left(\frac{3 \pm \sqrt{5}}{2}\right) > 0$.
But, $|g_3(s)| = 1$ at $s = \frac{3 \pm \sqrt{5}}{2}$ and $|g_3(s)| = \frac{9}{16}$ at $x = \frac{3}{2}$.
Therefore the maximum value of $|g_3(s)|$ is 1.
Hence,

$$|E_3(x)| \leq 1.h^4 \frac{M_4}{24} = \frac{h^4 M_4}{24}. \qquad \square$$

Comparison of accuracy and $O(h^{n+1})$

The equations (3.26), (3.27) and (3.28) give the bounds of errors for linear, quadratic and cubic interpolation polynomials. In each of these cases the error bound $|E_n(x)|$ depends on h in two ways.
Case I. h^{n+1} is present explicitly in $|E_n(x)|$ and $E_n(x)$ is proportional to h^{n+1}.

Case II. The value of M_{n+1} generally depends on the choice of h and tend to $|f^{(n+1)}(x_0)|$ as h goes to zero.

Therefore, as h tends to zero $|E_n(x)|$ converges to zero with the same rate that h^{n+1} converges to zero. Thus, one can say that $|E_n(x)| = O(h^{n+1})$. In particular, $|E_1(x)| = O(h^2), |E_2(x)| = O(h^3), |E_3(x)| = O(h^4)$ and so on.

As a consequence, if the derivatives of $f(x)$ are uniformly bounded on the interval and $|h| < 1$, then there is a scope to choose n sufficiently large to make h^{n+1} very small, and the higher degree polynomial will have less error.

Example 3.3.1 Consider $f(x) = \cos x$ over [0, 1.5]. Determine the error bounds for linear, quadratic and cubic Lagrange's polynomials.

Solution. $|f'(x)| = |\sin x|$, $|f''(x)| = |\cos x|$, $|f'''(x)| = |\sin x|$,
$|f^{iv}(x)| = |\cos x|$.
$|f''(x)| \leq |\cos 0| = 1.0$, so that $M_2 = 1.0$,
$|f'''(x)| \leq |\sin 1.5| = 0.997495$, so that $M_3 = 0.997495$,
$|f^{iv}(x)| \leq |\cos 0| = 1.0$, so that $M_4 = 1.0$.
For linear polynomial the spacing h of the points is $1.5 - 0 = 1.5$ and its error bound is

$$|E_1(x)| \leq \frac{h^2 M_2}{8} \leq \frac{(1.5)^2 \times 1.0}{8} = 0.28125.$$

For quadratic polynomial the spacing of the points is $h = (1.5 - 0)/2 = 0.75$ and its error bound is

$$|E_2(x)| \leq \frac{h^3 M_5}{9\sqrt{3}} \leq \frac{(0.75)^3 \times 0.997495}{9\sqrt{3}} = 0.0269955.$$

The spacing for cubic polynomial is $h = (1.5 - 0)/3 = 0.5$ and thus the error bound is

$$|E_3(x)| \leq \frac{h^4 M_4}{24} \leq \frac{(0.5)^4 \times 1.0}{24} = 0.0026042.$$

Example 3.3.2 A function $f(x)$ defined on the interval (0, 1) is such that $f(0) = 0, f(1/2) = -1, f(1) = 0$. Find the quadratic polynomial $p(x)$ which agrees with $f(x)$ for $x = 0, 1/2, 1$.
If $\left|\frac{d^3 f}{dx^3}\right| \leq 1$ for $0 \leq x \leq 1$, show that $|f(x) - p(x)| \leq \frac{1}{12}$ for $0 \leq x \leq 1$.

Solution. Given $x_0 = 0, x_1 = 1/2, x_2 = 1$ and $f(0) = 0, f(1/2) = -1,\ f(1) = 0$.
From Lagrange's interpolating formula, the required quadratic polynomial is

$$\begin{aligned}
p(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) \\
&\quad + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2) \\
&= \frac{(x-1/2)(x-1)}{(0-1/2)(0-1)} \times 0 + \frac{(x-0)(x-1)}{(1/2-0)(1/2-1)} \times (-1) + \frac{(x-0)(x-1/2)}{(1-0)(1-1/2)} \times 0 \\
&= 4x(x-1).
\end{aligned}$$

The error $E(x) = f(x) - p(x)$ is given by

$$\begin{aligned}
E(x) &= (x-x_0)(x-x_1)(x-x_2)\frac{f'''(\xi)}{3!} \\
\text{or, } |E(x)| &= |x-x_0||x-x_1||x-x_2|\left|\frac{f'''(\xi)}{3!}\right| \\
&\leq |x-0||x-1/2||x-1|1.\frac{1}{3!} \quad \left[\text{as } \left|\frac{d^3 f}{dx^3}\right| \leq 1 \quad \text{in} \quad 0 \leq x \leq 1\right].
\end{aligned}$$

Now, $|x - 0| \leq 1, |x - 1/2| \leq 1/2$ and $|x - 1| \leq 1$ in $0 \leq x \leq 1$.
Hence, $|E(x)| \leq 1.\frac{1}{2}.1.\frac{1}{6} = \frac{1}{12}$.
That is, $|f(x) - p(x)| \leq \frac{1}{12}$.

Example 3.3.3 Determine the step size h (and number of points n) to be used in the tabulation of $f(x) = \cos x$ in the interval [1, 2] so that the quadratic interpolation will be correct to six decimal places.

Solution. The upper bound of error in quadratic polynomial is

$$|E_2(x)| \le \frac{h^3 M_3}{9\sqrt{3}}, \qquad M_3 = \max_{1\le x\le 2} f'''(x).$$

$f(x) = \cos x, \qquad f'(x) = -\sin x, \qquad f''(x) = -\cos x, \qquad f'''(x) = \sin x.$

$$\max_{1\le x\le 2} |f'''(x)| = \max_{1\le x\le 2} |\sin x| = 1.$$

Hence $\dfrac{h^3}{9\sqrt{3}} \times 1 \le 5\times 10^{-6}$, i.e., $h^3 \le 45\sqrt{3}\times 10^{-6}$.

This gives $h \le 0.0427$ and $n = \dfrac{2-1}{h} = 23.42 \simeq 24$.

Advantage and disadvantage of Lagrangian interpolation

In Lagrangian interpolation, there is no restriction on the spacing and order of the tabulating points $x_0, x_1, \ldots, x_n$. Also, the value of y (the dependent variable) can be calculated at any point x within the minimum and maximum values of $x_0, x_1, \ldots, x_n$.

But its main disadvantage is, if the number of interpolating points decreases or increases then fresh calculation is required, the previous computations are of little help. This disadvantage is not in Newton's difference interpolation formulae, which are discussed in Section 3.5.

Example 3.3.4 Obtain a quadratic polynomial approximation to $f(x) = e^{-x}$ using Lagrange's interpolation method, taking three points $x = 0, 1/2, 1$.

Solution. Here $x_0 = 0, x_1 = 1/2, x_2 = 1$ and $f(x_0) = 1, f(x_1) = e^{-1/2}$, $f(x_2) = e^{-1}$.
The quadratic polynomial $\phi(x)$ is

$$\begin{aligned}
\phi(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) \\
&\quad + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2) \\
&= \frac{(x-1/2)(x-1)}{(0-1/2)(0-1)} \times 1 + \frac{(x-0)(x-1)}{(1/2-0)(1/2-1)} \times e^{-1/2} + \frac{(x-0)(x-1/2)}{(1-0)(1-1/2)} \times e^{-1} \\
&= (2x-1)(x-1) - 4e^{-1/2}x(x-1) + e^{-1}x(2x-1) \\
&= 2x^2(1 - 2e^{-1/2} + e^{-1}) - x(3 - 4e^{-1/2} + e^{-1}) + 1 \\
&= 0.309636243x^2 - 0.941756802x + 1.0.
\end{aligned}$$

The functions $f(x)$ and $\phi(x)$ are shown in Figure 3.2.

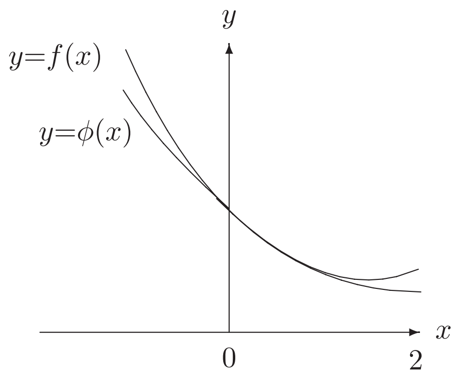


Figure 3.2: The graph of the function $f(x) = e^{-x}$ and the polynomial $\phi(x) = 0.309636243x^2 - 0.941756802x + 1$.

Algorithm 3.1 (Lagrange's interpolation). This algorithm determines the value of y at $x = x'$, say, from a given table of points $(x_i, y_i), i = 0, 1, 2, \dots, n$, using Lagrange's interpolation method.

```
**Algorithm Lagrange_Interpolation**
Step 1: Read $x, n$ // $n$ represents the number of points minus one//
                    // $x$ is the interpolating point//
Step 2: for $i = 0$ to $n$ do       //reading of tabulated values//
            Read $x_i, y_i$;
        endfor;
Step 3: Set $sum = 0$;
Step 4: for $i = 0$ to $n$ do
            Step 4.1: Set $prod = 1$;
            Step 4.2: for $j = 0$ to $n$ do
                if $(i \neq j)$ then $prod = prod \times \frac{x - x_j}{x_i - x_j}$;
            Step 4.3: Compute $sum = sum + y_i \times prod$;
        endfor;
Step 5: Print $x, sum$;
end Lagrange_Interpolation
```

Program 3.1

```
/* Program Lagrange Interpolation
  This program implements Lagrange's interpolation
  formula for one dimension; xg is the interpolating points */
```

```
#include <stdio.h>
#include <math.h>
void main()
{
 int n, i, j; float xg, x[20], y[20], sum=0, prod=1;
 printf("Enter the value of n and the data
         in the form x[i],y[i] ");
 scanf("%d",&n);
 for(i=0;i<=n;i++) scanf("%f %f",&x[i],&y[i]);
 printf("\nEnter the interpolating point x ");
 scanf("%f",&xg);
 for(i=0;i<=n;i++)
    {
      prod=1;
      for(j=0;j<=n;j++)
        {
         if(i!=j) prod*=(xg-x[j])/(x[i]-x[j]);
        }
      sum+=y[i]*prod;
    }
 printf("\nThe given data is ");
 for(i=0;i<=n;i++) printf("\n(%6.4f,%6.4f)",x[i],y[i]);
 printf("\nThe value of y at x= %5.2f is %8.5f ", xg, sum);
} /* main */
```

A sample of input/output:

```
Enter the value of n and the data in the form x[i],y[i] 4
1   5
1.5 8.2
2   9.2
3.2 11
4.5 16
Enter the interpolating point x 1.75
The given data is
(1.0000,5.0000)
(1.5000,8.2000)
(2.0000,9.2000)
(3.2000,11.0000)
(4.5000,16.0000)
The value of y at x=  1.75 is  8.85925
```

3.4 Finite Differences

Different types of finite differences are introduced in Chapter 2. Some of them are recapitulated here.

Let a function $y = f(x)$ be known as (x_i, y_i) at $(n+1)$ points x_i, $i = 0, 1, \ldots, n$, where x_i's are equally spaced, i.e., $x_i = x_0 + ih$, h is the spacing between any two successive points x_i's. That is, $y_i = f(x_i), i = 0, 1, \ldots, n$.

3.4.1 Forward differences

The first forward difference of $f(x)$ is defined as

$$\Delta f(x) = f(x+h) - f(x),$$

Δ is called forward difference operator.

Then $\Delta f(x_0) = f(x_0 + h) - f(x_0) = f(x_1) - f(x_0)$.

That is, $\Delta y_0 = y_1 - y_0$ using $y_i = f(x_i)$.

Similarly, $\Delta y_1 = y_2 - y_1$, $\Delta y_2 = y_3 - y_2$, ..., $\Delta y_{n-1} = y_n - y_{n-1}$.

The second order differences are

$$\begin{aligned} \Delta^2 y_0 &= \Delta(\Delta y_0) = \Delta(y_1 - y_0) \\ &= \Delta y_1 - \Delta y_0 = (y_2 - y_1) - (y_1 - y_0) = y_2 - 2y_1 + y_0. \end{aligned}$$

Similarly, $\Delta^2 y_1 = y_3 - 2y_2 + y_1$, etc.

The third order differences

$$\begin{aligned} \Delta^3 y_0 &= \Delta(\Delta^2 y_0) = \Delta(y_2 - 2y_1 + y_0) = y_3 - 3y_2 + 3y_1 - y_0, \\ \Delta^3 y_1 &= y_4 - 3y_3 + 3y_2 - y_1, \text{ etc.} \end{aligned}$$

In general,

$$\Delta^k y_0 = y_k - {}^kC_1 y_{k-1} + {}^kC_2 y_{k-2} - \cdots - (-1)^k y_0 \tag{3.29}$$

$$\Delta^k y_i = y_{k+i} - {}^kC_1 y_{k+i-1} + {}^kC_2 y_{k+i-2} - \cdots - (-1)^k y_i. \tag{3.30}$$

It is observed that difference of any order can easily be expressed in terms of the ordinates y_i's with binomial coefficients.

All orders forward differences can be written in a tabular form shown in Table 3.1.

This difference table is called **forward difference table** or **diagonal difference table**.

Table 3.1: Forward difference table.

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
x_0	y_0				
		Δy_0			
x_1	y_1		$\Delta^2 y_0$		
		Δy_1		$\Delta^3 y_0$	
x_2	y_2		$\Delta^2 y_1$		$\Delta^4 y_0$
		Δy_2		$\Delta^3 y_1$	
x_3	y_3		$\Delta^2 y_2$		
		Δy_3			
x_4	y_4				

3.4.2 Backward differences

The first order backward difference of $f(x)$ is defined as

$$\nabla f(x) = f(x) - f(x-h),$$

where ∇ is the backward difference operator.

Thus, $\nabla f(x_1) = f(x_1) - f(x_0)$, or, $\nabla y_1 = y_1 - y_0$.

Similarly, $\nabla y_2 = y_2 - y_1$, $\nabla y_3 = y_3 - y_2$, …, $\nabla y_n = y_n - y_{n-1}$.

The second order differences are

$$\begin{aligned} \nabla^2 y_2 &= \nabla(\nabla y_2) = \nabla(y_2 - y_1) = y_2 - 2y_1 + y_0, \\ \nabla^2 y_3 &= y_3 - 2y_2 + y_1, \text{ etc.} \end{aligned}$$

The third order differences are

$$\begin{aligned} \nabla^3 y_3 &= y_3 - 3y_2 + 3y_1 - y_0, \\ \nabla^3 y_4 &= y_4 - 3y_3 + 3y_2 - y_1, \text{ etc.} \end{aligned}$$

In general, $$\nabla^k y_i = y_i - {}^kC_1 y_{i-1} + {}^kC_2 y_{i-2} - \cdots - (-1)^k y_{i-k}. \tag{3.31}$$

Table 3.2 shows how the backward differences of all orders can be formed.

The backward difference table is sometimes called **horizontal difference table**.

3.4.3 Error propagation in a difference table

If there is an error in any entry among the tabulated values of a function, then this error propagates to other entries of higher order differences. To illustrate the behaviour of propagation of error, we assume that an error ε is present in the number, say, y_3.

Table 3.2: Backward difference table.

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$
x_0	y_0				
x_1	y_1	∇y_1			
x_2	y_2	∇y_2	$\nabla^2 y_2$		
x_3	y_3	∇y_3	$\nabla^2 y_3$	$\nabla^3 y_3$	
x_4	y_4	∇y_4	$\nabla^2 y_4$	$\nabla^3 y_4$	$\nabla^4 y_4$

Table 3.3: Error propagation in a finite difference table.

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
x_0	y_0					
		Δy_0				
x_1	y_1		$\Delta^2 y_0$			
		Δy_1		$\Delta^3 y_0 + \varepsilon$		
x_2	y_2		$\Delta^2 y_1 + \varepsilon$		$\Delta^4 y_0 - 4\varepsilon$	
		$\Delta y_2 + \varepsilon$		$\Delta^3 y_1 - 3\varepsilon$		$\Delta^5 y_0 + 10\varepsilon$
x_3	$y_3 + \varepsilon$		$\Delta^2 y_2 - 2\varepsilon$		$\Delta^4 y_1 + 6\varepsilon$	
		$\Delta y_3 - \varepsilon$		$\Delta^3 y_2 + 3\varepsilon$		$\Delta^5 y_1 - 10\varepsilon$
x_4	y_4		$\Delta^2 y_3 + \varepsilon$		$\Delta^4 y_2 - 4\varepsilon$	
		Δy_4		$\Delta^3 y_3 - \varepsilon$		
x_5	y_5		$\Delta^2 y_4$			
		Δy_5				
x_6	y_6					

Table 3.3 shows the propagation of error in a difference table and how the error affects the differences. From this table, the following observations are noted.

(i) The effect of the error increases with the order of the differences.

(ii) The error is maximum (in magnitude) along the horizontal line through the erroneous tabulated value.

(iii) The second difference column has the errors $\varepsilon, -2\varepsilon, \varepsilon$, in the third difference column, the errors are $\varepsilon, -3\varepsilon, 3\varepsilon, -\varepsilon$. In the fourth difference column the expected errors $\varepsilon, -4\varepsilon, 6\varepsilon, -4\varepsilon, \varepsilon$ (this column is not sufficient to show all of the expected errors). Thus, in the pth difference column, the coefficients of errors are the binomial coefficients in the expansion of $(1 - x)^p$.

(iv) The algebraic sum of errors in any column (complete) is zero. If there is any error in a single entry of a table, then from the difference table one can detect and correct such error.

Detection of errors using difference table

Difference table may be used to detect errors in a set of tabular values. From Table 3.3, it follows that if an error is present in a given data, the differences of some order will become alternating in sign. Thus, higher order differences should be formed till the error is revealed.

To detect the position of the error in an entry, the following steps may be proceed.

(i) Form the difference table. If at any stage, the differences do not follow a smooth pattern, then one can conclude that there is an error.

(ii) If the differences of some order (it is generally happens in higher order) becomes alternating in sign then the middle entry has an error.

Finite Difference Interpolations

3.5 Newton's Forward Difference Interpolation Formula

Let $y = f(x)$ be a function whose explicit form is unknown. But, the values of y at the equispaced points $x_0, x_1, \ldots, x_n$, i.e., $y_i = f(x_i), i = 0, 1, \ldots, n$ are known. Since $x_0, x_1, \ldots, x_n$ are equispaced then $x_i = x_0 + ih, i = 0, 1, \ldots, n$, where h is the spacing. It is required to construct a polynomial $\phi(x)$ of degree less than or equal to n satisfying the conditions

$$y_i = \phi(x_i), \qquad i = 0, 1, \ldots, n. \tag{3.32}$$

Since $\phi(x)$ is a polynomial of degree at most n, so $\phi(x)$ can be taken in the following form

$$\begin{aligned} \phi(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2) \\ &\quad + \cdots + a_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}), \end{aligned} \tag{3.33}$$

where $a_0, a_1, \ldots, a_n$ are constants whose values are to be determined using (3.32).

To determine the values of a_i's, substituting $x = x_i, i = 0, 1, 2, \ldots, n$.

When $x = x_0$ then

$\phi(x_0) = a_0$ or, $a_0 = y_0$.

For $x = x_1$, $\phi(x_1) = a_0 + a_1(x_1 - x_0)$

or, $y_1 = y_0 + a_1 h$ or, $a_1 = \dfrac{y_1 - y_0}{h} = \dfrac{\Delta y_0}{h}$.

For $x = x_2$, $\phi(x_2) = a_0 + a_1(x_2 - x_1) + a_2(x_2 - x_0)(x_2 - x_1)$

or, $y_2 = y_0 + \dfrac{y_1 - y_0}{h}.2h + a_2(2h)(h)$

or, $a_2 = \dfrac{y_2 - 2y_1 + y_0}{2!h^2} = \dfrac{\Delta^2 y_0}{2!h^2}$.

In this way,

$$a_3 = \frac{\Delta^3 y_0}{3!h^3}, a_4 = \frac{\Delta^4 y_0}{4!h^4}, \ldots, a_n = \frac{\Delta^n y_0}{n!h^n}.$$

Using these values, (3.33) becomes

$$\begin{aligned}\phi(x) &= y_0 + (x - x_0)\frac{\Delta y_0}{h} + (x - x_0)(x - x_1)\frac{\Delta^2 y_0}{2!h^2} \\ &\quad +(x - x_0)(x - x_1)(x - x_2)\frac{\Delta^3 y_0}{3!h^3} \\ &\quad + \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1})\frac{\Delta^n y_0}{n!h^n}. \end{aligned} \tag{3.34}$$

Introducing the condition $x_i = x_0 + ih, i = 0, 1, \ldots, n$ for equispaced points and a new variable u as $x = x_0 + uh$.

Therefore, $x - x_i = (u - i)h$.

So the equation (3.34) becomes

$$\begin{aligned}\phi(x) &= y_0 + (uh)\frac{\Delta y_0}{h} + (uh)(u-1)h\frac{\Delta^2 y_0}{2!h^2} + (uh)(u-1)h(u-2)h\frac{\Delta^3 y_0}{3!h^3} \\ &\quad + \cdots + (uh)(u-1)h(u-2)h\cdots(u - \overline{n-1})h\frac{\Delta^n y_0}{n!h^n} \\ &= y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 \\ &\quad + \cdots + \frac{u(u-1)(u-2)\cdots(u - \overline{n-1})}{n!}\Delta^n y_0, \end{aligned} \tag{3.35}$$

where $u = \dfrac{x - x_0}{h}$.

This is known as **Newton** or **Newton-Gregory** forward difference interpolating polynomial.

Example 3.5.1 The following table gives the values of e^x for certain equidistant values of x. Find the value of e^x when $x = 0.612$ using Newton's forward difference formulae.

x	:	0.61	0.62	0.63	0.64	0.65
y	:	1.840431	1.858928	1.877610	1.896481	1.915541

Solution. The forward difference table is

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
0.61	1.840431			
		0.018497		
0.62	1.858928		0.000185	
		0.018682		0.000004
0.63	1.877610		0.000189	
		0.018871		0.0
0.64	1.896481		0.000189	
		0.019060		
0.65	1.915541			

Here, $x_0 = 0.61, x = 0.612, h = 0.01, u = \dfrac{x - x_0}{h} = \dfrac{0.612 - 0.61}{0.01} = 0.2.$
Then,

$$\begin{aligned}
y(0.612) &= y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 \\
&= 1.840431 + 0.2 \times 0.018497 + \frac{0.2(0.2-1)}{2} \times 0.000185 \\
&\quad + \frac{0.2(0.2-1)(0.2-2)}{6} \times 0.000004 \\
&= 1.840431 + 0.003699 - 0.000015 + 0.00000019 \\
&= 1.844115.
\end{aligned}$$

3.5.1 Error in Newton's forward formula

The error in any polynomial interpolation formula is

$$\begin{aligned}
E(x) &= (x - x_0)(x - x_1)\cdots(x - x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!} \\
&= u(u-1)(u-2)\cdots(u-n)h^{n+1}\frac{f^{(n+1)}(\xi)}{(n+1)!} \quad \text{(using } x = x_0 + uh\text{)}
\end{aligned}$$

where ξ lies between $\min\{x_0, x_1, \ldots, x_n, x\}$ and $\max\{x_0, x_1, \ldots, x_n, x\}$.

Also, $f^{(n+1)}(\xi) \simeq h^{n+1}\Delta^{n+1} y_0$.

Therefore,

$$E(x) \simeq \frac{u(u-1)(u-2)\cdots(u-n)}{(n+1)!}\Delta^{n+1} y_0.$$

A particular case:
If $0 < u < 1$ then

$|u(u-1)| = (1-u)u = u - u^2 = \frac{1}{4} - \left(\frac{1}{2} - u\right)^2 \leq \frac{1}{4}$ and
$|(u-2)(u-3)\cdots(u-n)| \leq |(-2)(-3)\cdots(-n)| = n!$.
Then,

$$|E(x)| \leq \frac{1}{4}\frac{n!}{(n+1)!}|\Delta^{n+1}y_0| = \frac{1}{4(n+1)}|\Delta^{n+1}y_0|.$$

Also, $|\Delta^{n+1}y_0| \leq 9$ in the last significant figure.
Thus, $|E(x)| \leq \frac{9}{4(n+1)} < 1$ for $n > 2$ and $0 < u < 1$.
That is, the maximum error in Newton's forward interpolation is 1 when $|x - x_0| < h$.

Newton's forward formula is used to compute the approximate value of $f(x)$ when the argument x is near the beginning of the table. But this formula is not appropriate to compute $f(x)$ when x at the end of the table. In this situation Newton's backward formula is appropriate.

Algorithm 3.2 (Newton's forward interpolation). This algorithm determines the value of y when the value of x is given, by Newton's forward interpolation method. The values of $x_i, y_i, i = 0, 1, 2, \ldots, n$ are given and assumed that $x_i = x_0 + ih$, i.e., the data are equispaced.

```
**Algorithm Newton_Forward_Intepolation**
//Assume that the data are equispaced.//
Read $(x_i, y_i), i = 0, 1, 2, \ldots, n$;
Read $xg$; //the value of $x$ at which $y$ is to be determined.//
Compute $h = x_1 - x_0$; //compute spacing.//
Compute $u = (xg - x_0)/h$;
for $j = 0$ to $n$ do
            $dy_j = y_j$; //copy of $y$ to $dy$//
Set $prod = 1$, $sum = y_0$;
for $i = 1$ to $n$ do
    for $j = 0$ to $(n - i)$ do $dy_j = dy_{j+1} - dy_j$;
            //$dy$ represents the difference.//
    Compute $prod = prod \times \frac{u - i + 1}{i}$;
    Compute $sum = sum + prod \times dy_0$;
endfor;
Print 'The value of $y$ at $x =$',$xg$, 'is ', $sum$;
**end Newton_Forward_Intepolation**
```

Program 3.2

```
/* Program Newton Forward Interpolation
   This program finds the value of y=f(x) at a given x when
   the function is supplied as (x[i],y[i]), i=0, 1, ..., n.
   Assumed that x's are equispaced.*/
#include<stdio.h>
#include<math.h>
void main()
{
 int i,j,n; float x[20],y[20],xg,sum,prod=1,u,dy[20],h;
 printf("Enter number of subintervals ");
 scanf("%d",&n);
 printf("Enter x and y values ");
 for(i=0;i<=n;i++) scanf("%f %f",&x[i],&y[i]);
 printf("Enter interpolating point x ");
 scanf("%f",&xg);
 h=x[1]-x[0];
 u=(xg-x[0])/h;
 for(j=0;j<=n;j++) dy[j]=y[j];
 prod=1;  sum=y[0];
 for(i=1;i<=n;i++)
   {
     for(j=0;j<=n-i;j++) dy[j]=dy[j+1]-dy[j];
     prod*=(u-i+1)/i;
     sum+=prod*dy[0];
   }
 printf("The value of y at x=%f is %f ",xg,sum);
}
```

A sample of input/output:

```
Enter number of subintervals 4
Enter x and y values
140 3.685
150 5.854
160 6.302
170 8.072
180 10.225
Enter interpolating point x 142
The value of y at x=142.000000 is 4.537069
```

3.6 Newton's Backward Difference Interpolation Formula

Suppose, a set of values $y_0, y_1, \dots, y_n$ of the function $y = f(x)$ is given, at $x_0, x_1, \dots, x_n$, i.e., $y_i = f(x_i), i = 0, 1, \dots, n$. Let x_i's are equispaced with spacing h, i.e., $x_i = x_0 + ih$.

Let us consider the polynomial $\phi(x)$ in the following form:

$$\begin{aligned} f(x) \simeq \phi(x) &= a_0 + a_1(x - x_n) + a_2(x - x_n)(x - x_{n-1}) \\ &\quad + a_3(x - x_n)(x - x_{n-1})(x - x_{n-2}) \\ &\quad + \cdots + a_n(x - x_n)(x - x_{n-1}) \cdots (x - x_1). \end{aligned} \tag{3.36}$$

The constants a_i's are to be determined using the conditions

$$y_i = \phi(x_i), i = 0, 1, \dots, n. \tag{3.37}$$

Substituting $x = x_n, n_{n-1}, \dots, x_1$ in (3.36), we obtain
$\phi(x_n) = a_0$ or, $a_0 = y_n$.
$\phi(x_{n-1}) = a_0 + a_1(x_{n-1} - x_n)$ or, $y_{n-1} = y_n + a_1(-h)$ or, $a_1 = \dfrac{y_n - y_{n-1}}{h} = \dfrac{\nabla y_n}{h}$.
$\phi(x_{n-2}) = a_0 + a_1(x_{n-2} - x_n) + a_2(x_{n-2} - x_n)(x_{n-2} - x_{n-1})$
$\quad = y_n + \dfrac{y_n - y_{n-1}}{h}(-2h) + a_2(-2h)(-h)$
or, $y_{n-2} = 2y_{n-1} - y_n + a_2.2!h^2$ or, $a_2 = \dfrac{y_n - 2y_{n-1} + y_{n-2}}{2!h^2} = \dfrac{\nabla^2 y_n}{2!h^2}$.

In this way, the others values are obtained as,

$$a_3 = \frac{\nabla^3 y_n}{3!h^3}, \quad a_4 = \frac{\nabla^4 y_n}{4!h^4}, \dots, a_n = \frac{\nabla^n y_n}{n!h^n}.$$

When the values of a_i's are substituted in (3.36) then the polynomial $\phi(x)$ becomes

$$\begin{aligned} \phi(x) &= y_n + (x - x_n)\frac{\nabla y_n}{h} + (x - x_n)(x - x_{n-1})\frac{\nabla^2 y_n}{2!h^2} \\ &\quad + (x - x_n)(x - x_{n-1})(x - x_{n-2})\frac{\nabla^3 y_n}{3!h^3} + \cdots \\ &\quad + (x - x_n)(x - x_{n-1})(x - x_{n-2}) \cdots (x - x_1)\frac{\nabla^n y_n}{n!h^n}. \end{aligned} \tag{3.38}$$

Now, a unit less variable v is introduced which is defined as $x = x_n + vh$, i.e., $v = \dfrac{x - x_n}{h}$. This substitution simplifies the formula.

Also, for equispaced points $x_i = x_0 + ih$.

Then $x - x_{n-i} = (x_n + vh) - (x_0 + \overline{n - i}h) = (x_n - x_0) + (v - \overline{n - i})h = (v + i)h$, $i = 0, 1, \dots, n$.

Using above results, (3.38) becomes

$$\begin{aligned}\phi(x) &= y_n + vh\frac{\nabla y_n}{h} + vh(v+1)h\frac{\nabla^2 y_n}{2!h^2} + vh(v+1)h(v+2)h\frac{\nabla^3 y_n}{3!h^3} + \cdots \\ &\quad + vh(v+1)h(v+2)h\cdots(v+n-1)h\frac{\nabla^n y_n}{n!h^n} \\ &= y_n + v\nabla y_n + \frac{v(v+1)}{2!}\nabla^2 y_n + \frac{v(v+1)(v+2)}{3!}\nabla^3 y_n + \cdots \\ &\quad + \frac{v(v+1)(v+2)\cdots(v+n-1)}{n!}\nabla^n y_n. \end{aligned} \tag{3.39}$$

This formula is known as **Newton's backward** or **Newton-Gregory backward interpolation formula**.

Example 3.6.1 From the following table of values of x and $f(x)$ determine the value of $f(0.29)$ using Newton's backward interpolation formula.

x	:	0.20	0.22	0.24	0.26	0.28	0.30
$f(x)$	:	1.6596	1.6698	1.6804	1.6912	1.7024	1.7139

Solution. The difference table is

x	$f(x)$	$\Delta f(x)$	$\Delta^2 f(x)$	$\Delta^3 f(x)$
0.20	1.6596			
0.22	1.6698	0.0102		
0.24	1.6804	0.0106	0.0004	
0.26	1.6912	0.0108	0.0002	−0.0002
0.28	1.7024	0.0112	0.0004	0.0002
0.30	1.7139	0.0115	0.0003	−0.0001

Here, $x_n = 0.30, x = 0.29, h = 0.02, v = \dfrac{x - x_n}{h} = \dfrac{0.29 - 0.30}{0.02} = -0.5.$
Then,

$$\begin{aligned} f(0.29) &= f(x_n) + v\nabla f(x_n) + \frac{v(v+1)}{2!}\nabla^2 f(x_n) + \frac{v(v+1)(v+2)}{3!}\nabla^3 f(x_n) + \cdots \\ &= 1.7139 - 0.5 \times 0.0115 + \frac{-0.5(-0.5+1)}{2} \times 0.0003 \\ &\quad + \frac{-0.5(-0.5+1)(-0.5+2)}{6} \times (-0.0001) \\ &= 1.7139 - 0.00575 - 0.0000375 + 0.00000625 \\ &= 1.70811875 \simeq 1.7081. \end{aligned}$$

Example 3.6.2 The population of a town in decennial census were given in the following table.

Year	:	1921	1931	1941	1951	1961
Population (in thousand)	:	46	66	81	93	101

Estimate the population for the year 1955 using Newton's backward and forward formulae and compare the results.

Solution.
Using Newton's backward formula
The backward difference table is

Year (x)	Population (y)	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$
1921	46				
1931	66	20			
1941	81	15	-5		
1951	93	12	-3	2	
1961	101	8	-4	-1	-3

Here, $x_n = 1961, x = 1955, h = 10, v = \frac{x-x_n}{h} = \frac{1955-1961}{10} = -0.6.$
By Newton's backward formula

$$\begin{aligned} y(1955) &= y_n + v\nabla y_n + \frac{v(v+1)}{2!}\nabla^2 y_n + \frac{v(v+1)(v+2)}{3!}\nabla^3 y_n \\ &\quad + \frac{v(v+1)(v+2)(v+3)}{4!}\nabla^4 y_n \\ &= 101 - 0.6 \times 8 + \frac{-0.6(-0.6+1)}{2} \times (-4) \\ &\quad + \frac{-0.6(-0.6+1)(-0.6+2)}{6} \times (-1) \\ &\quad + \frac{-0.6(-0.6+1)(-0.6+2)(-0.6+3)}{24} \times (-3) \\ &= 96.8368 \simeq 97. \end{aligned}$$

Hence the approximate population of the town was 97 thousand.

Using Newton's forward formula
The given table is written in reverse order as

Year	:	1961	1951	1941	1931	1921
Population	:	101	93	81	66	46

The forward difference table is

Year x	Population y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
1961	101				
		-8			
1951	93		-4		
		-12		1	
1941	81		-3		-3
		-15		-2	
1931	66		-5		
		-20			
1921	46				

Here $x_0 = 1961, x = 1955, h = -10, u = \dfrac{x - x_0}{h} = \dfrac{1955 - 1961}{-10} = 0.6.$
Then

$$\begin{aligned}
y(1955) &= y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 \\
&\quad + \frac{u(u-1)(u-2)(u-3)}{4!}\Delta^4 y_0 \\
&= 101 + 0.6 \times (-8) + \frac{0.6(0.6-1)}{2} \times (-4) + \frac{0.6(0.6-1)(0.6-2)}{6} \times 1 \\
&\quad + \frac{0.6(0.6-1)(0.6-2)(0.6-3)}{24} \times (-3) \\
&= 101 - 4.8 + 0.48 + 0.056 + 0.1008 \\
&= 96.8368 \simeq 97.
\end{aligned}$$

Therefore the population of the town in the year 1955 was 97 thousand and this result is same with the result obtained by Newton's backward difference formula.

3.6.1 Error in Newton's backward interpolation formula

The error in this interpolation formula is

$$\begin{aligned}
E(x) &= (x - x_n)(x - x_{n-1}) \cdots (x - x_1)(x - x_0)\frac{f^{(n+1)}(\xi)}{(n+1)!} \\
&= v(v+1)(v+2)\cdots(v+n)h^{n+1}\frac{f^{(n+1)}(\xi)}{(n+1)!}, \qquad (3.40)
\end{aligned}$$

where $v = \dfrac{x - x_n}{h}$ and ξ lies between $\min\{x_0, x_1, \ldots, x_n, x\}$ and $\max\{x_0, x_1, \ldots, x_n, x\}$.

Note 3.6.1 The Newton's backward difference interpolation formula is used to compute the value of $f(x)$ when x is near to x_n, i.e., when x is at the end of the table.

Central Difference Interpolation Formulae

3.7 Gaussian Interpolation Formulae

Newton's forward and Newton's backward formulae does not give accurate value of $f(x)$ when x is in the middle of the table. To get more accurate result another formula may be used. There are several methods available to solve this type of problem. Among them Gaussian forward and backward, Stirling's and Bessel's interpolation formulae are widely used.

3.7.1 Gauss's forward difference formula

Case I: For $2n+1$ (odd) arguments

Suppose the values of the function $y = f(x)$ are known at $2n+1$ equally spaced points $x_{-n}, x_{-(n-1)}, \dots, x_{-1}, x_0, x_1, \dots, x_{n-1}, x_n$, i.e., $y_i = f(x_i)$, $i = 0, \pm 1, \pm 2, \dots, \pm n$.

The problem is to construct a polynomial $\phi(x)$ of degree at most $2n$ such that

$$\phi(x_i) = y_i, i = 0, \pm 1, \pm 2, \dots, \pm n, \tag{3.41}$$

where $x_i = x_0 + ih$, h is the spacing.

Let us consider $\phi(x)$ as

$$\begin{aligned}\phi(x) &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_{-1})(x - x_0)(x - x_1) \\ &\quad + a_4(x - x_{-1})(x - x_0)(x - x_1)(x - x_2) + \cdots \\ &\quad + a_{2n-1}(x - x_{-n+1})(x - x_{-n+2}) \cdots (x - x_{-1})(x - x_0) \cdots (x - x_{n-1}) \\ &\quad + a_{2n}(x - x_{-n+1})(x - x_{-n+2}) \cdots (x - x_{-1})(x - x_0) \cdots (x - x_n),\end{aligned} \tag{3.42}$$

where a_i's are unknown constants and their values are to be determined by substituting $x = x_0, x_1, x_{-1}, x_2, x_{-2}, \dots, x_n, x_{-n}$.

Therefore,

$$\begin{aligned} y_0 &= a_0 \\ y_1 &= a_0 + a_1(x_1 - x_0) \;\; i.e., \;\; y_1 = y_0 + a_1 h, \\ \text{i.e.,} \quad a_1 &= \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y_0}{h}. \\ y_{-1} &= y_0 + a_1(-h) + a_2(-h)(-2h) \\ &= y_0 - h\frac{\Delta y_0}{h} + a_2 h^2 \cdot 2! \end{aligned}$$

$$\begin{aligned}
\text{i.e.,}\quad a_2 &= \frac{y_{-1}-2y_0+y_1}{2!\,h^2} = \frac{\Delta^2 y_{-1}}{2!\,h^2}.\\
y_2 &= a_0 + a_1(x_2-x_0) + a_2(x_2-x_0)(x_2-x_1)\\
&\quad + a_3(x_2-x_{-1})(x_2-x_0)(x_2-x_1)\\
&= y_0 + \frac{y_1-y_0}{h}(2h) + \frac{y_{-1}-2y_0+y_1}{2!h^2}(2h)(h) + a_3(3h)(2h)(h)\\
\text{or,}\quad a_3 &= \frac{y_2-3y_1+3y_0-y_{-1}}{3!h^3} = \frac{\Delta^3 y_{-1}}{3!h^3}.
\end{aligned}$$

In this manner, the remaining values are obtained as

$$a_4 = \frac{\Delta^4 y_{-2}}{4!h^4}, a_5 = \frac{\Delta^5 y_{-2}}{5!h^5}, \ldots, a_{2n-1} = \frac{\Delta^{2n-1} y_{-(n-1)}}{(2n-1)!h^{2n-1}}, a_{2n} = \frac{\Delta^{2n} y_{-n}}{(2n)!h^{2n}}.$$

Thus the Gauss's forward difference formula is

$$\begin{aligned}
\phi(x) &= y_0 + (x-x_0)\frac{\Delta y_0}{h} + (x-x_0)(x-x_1)\frac{\Delta^2 y_{-1}}{2!h^2}\\
&\quad + (x-x_{-1})(x-x_0)(x-x_1)\frac{\Delta^3 y_{-1}}{3!h^3} + \cdots\\
&\quad + (x-x_{-(n+1)})\cdots(x-x_{n-1})\frac{\Delta^{2n-1} y_{-(n-1)}}{(2n-1)!h^{2n-1}}\\
&\quad + (x-x_{-(n+1)})\cdots(x-x_{n-1})(x-x_n)\frac{\Delta^{2n} y_{-n}}{(2n)!h^{2n}}. \qquad (3.43)
\end{aligned}$$

To simplify the above equation, a new variable s is introduced, where $s = \dfrac{x-x_0}{h}$ i.e, $x = x_0 + sh$.

Also, $x_{\pm\, i} = x_0 \pm ih, i = 0, 1, 2, \ldots, n$.

Therefore, $x - x_{\pm i} = (s \pm i)h$.

Then $x - x_0 = sh, x - x_1 = (s-1)h, x - x_{-1} = (s+1)h$, $x - x_2 = (s-2)h, x - x_{-2} = (s+2)h$ and so on.

Making use of these results, (3.43) becomes

$$\begin{aligned}
\phi(x) &= y_0 + sh\frac{\Delta y_0}{h} + sh(s-1)h\frac{\Delta^2 y_{-1}}{2!h^2} + (s+1)hsh(s-1)h\frac{\Delta^3 y_{-1}}{3!h^3} + \cdots\\
&\quad + (s+\overline{n-1})h\cdots sh(s-1)h\cdots(s-\overline{n-1})h\frac{\Delta^{2n-1} y_{-(n-1)}}{(2n-1)!h^{2n-1}}\\
&\quad + (s+\overline{n-1})h\cdots sh(s-1)h\cdots(s-\overline{n-1})h(s-n)h\frac{\Delta^{2n} y_{-n}}{(2n)!h^{2n}}
\end{aligned}$$

$$
\begin{aligned}
&= y_0 + s\Delta y_0 + s(s-1)\frac{\Delta^2 y_{-1}}{2!} + (s+1)s(s-1)\frac{\Delta^3 y_{-1}}{3!} + \cdots \\
&\quad + (s+\overline{n-1})\cdots s(s-1)\cdots(s-\overline{n-1})\frac{\Delta^{2n-1} y_{-(n-1)}}{(2n-1)!} \\
&\quad + (s+\overline{n-1})\cdots s(s-1)\cdots(s-\overline{n-1})(s-n)\frac{\Delta^{2n} y_{-n}}{(2n)!} \\
&= y_0 + s\Delta y_0 + s(s-1)\frac{\Delta^2 y_{-1}}{2!} + s(s^2-1^2)\frac{\Delta^3 y_{-1}}{3!} \\
&\quad + s(s^2-1^2)(s-2)\frac{\Delta^4 y_{-2}}{4!} + \cdots \\
&\quad + s(s^2-\overline{n-1}^2)(s^2-\overline{n-2}^2)\cdots(s^2-1^2)\frac{\Delta^{2n-1} y_{-(n-1)}}{(2n-1)!} \\
&\quad + s(s^2-\overline{n-1}^2)(s^2-\overline{n-2}^2)\cdots(s^2-1^2)(s-n)\frac{\Delta^{2n} y_{-n}}{(2n)!}. \qquad (3.44)
\end{aligned}
$$

The formula (3.43) or (3.44) is known as **Gauss's forward central difference formula** or **the first interpolation formula of Gauss**.

Case II: For $2n$ (even) arguments

In this case the arguments are $x_0, x_{\pm 1}, \ldots, x_{\pm(n-1)}$ and x_n.

For these points the Gauss's forward interpolation takes the following form.

$$
\begin{aligned}
\phi(x) &= y_0 + s\Delta y_0 + s(s-1)\frac{\Delta^2 y_{-1}}{2!} + (s+1)s(s-1)\frac{\Delta^3 y_{-1}}{3!} \\
&\quad + (s+1)s(s-1)(s-2)\frac{\Delta^4 y_{-2}}{4!} \\
&\quad + (s+2)(s+1)s(s-1)(s-2)\frac{\Delta^5 y_{-2}}{5!} + \cdots \\
&\quad + (s+\overline{n-1})\cdots s\cdots(s-\overline{n-1})\frac{\Delta^{2n-1} y_{-(n-1)}}{(2n-1)!} \\
&= y_0 + s\Delta y_0 + s(s-1)\frac{\Delta^2 y_{-1}}{2!} + s(s^2-1^2)\frac{\Delta^3 y_{-1}}{3!} \\
&\quad + s(s^2-1^2)(s-2)\frac{\Delta^4 y_{-2}}{4!} + (s^2-2^2)(s^2-1^2)s\frac{\Delta^5 y_{-2}}{5!} + \cdots \\
&\quad + (s^2-\overline{n-1}^2)\cdots(s^2-1^2)s\frac{\Delta^{2n-1} y_{-(n-1)}}{(2n-1)!}. \qquad (3.45)
\end{aligned}
$$

3.7.2 Remainder in Gauss's forward central difference formula

The remainder of Gauss's forward central difference interpolation for $2n+1$ arguments is

$$\begin{aligned}
E(x) &= (x-x_{-n})(x-x_{-(n-1)})\cdots(x-x_{-1})(x-x_0)\cdots(x-x_n)\frac{f^{2n+1}(\xi)}{(2n+1)!}\\
&= (s+n)(s+n-1)\cdots(s+1)s(s-1)\cdots(s-n+1)(s-n)\\
&\qquad\qquad\qquad\qquad \times\ h^{2n+1}\frac{f^{2n+1}(\xi)}{(2n+1)!}\\
&= s(s^2-1^2)\cdots(s^2-n^2).h^{2n+1}\frac{f^{2n+1}(\xi)}{(2n+1)!} \qquad (3.46)
\end{aligned}$$

where $s=\dfrac{x-x_0}{h}$ and ξ lies between $\min\{x_{-n},x_{-(n-1)},\ldots,x_0,\ x_1,\ldots,x_{n-1},x_n\}$ and $\max\{x_{-n},x_{-(n-1)},\ldots,x_0,x_1,\ldots,x_{n-1},x_n\}$.

In case of $2n$ arguments, the error is

$$\begin{aligned}
E(x) &= (s+n-1)\cdots(s+1)s(s-1)\cdots(s-n+1)(s-n)h^{2n}\frac{f^{2n}(\xi)}{(2n)!}\\
&= s(s^2-1^2)\cdots(s^2-\overline{n-1}^2)(s-n).h^{2n}\frac{f^{2n}(\xi)}{(2n)!}, \qquad (3.47)
\end{aligned}$$

where, $\min\{x_{-n},x_{-(n-1)},\ldots,x_0,x_1,\ldots,x_{n-1},x_n\}<\xi$
$<\max\{x_{-n},x_{-(n-1)},\ldots,x_0,x_1,\ldots,x_{n-1},x_n\}$.

3.7.3 Gauss's backward difference formula

Case I: For $2n+1$ (odd) number of arguments

Let a function $y=f(x)$ is known for $2n+1$ equispaced arguments $x_{\pm i}, i=0,1,2,\ldots,n$, such that

$$x_{\pm i}=x_0\pm ih, i=0,1,2,\ldots,n.$$

Let $y_{\pm i}=f(x_{\pm i}), i=0,1,2,\ldots,n$.

Our aim is to determine a polynomial $\phi(x)$ of degree not more than $2n$ which satisfies the conditions

$$\phi(x_{\pm i})=y_{\pm i}, i=0,1,\ldots,n. \qquad (3.48)$$

The polynomial $\phi(x)$ is considered in the following form.

$$\begin{aligned}\phi(x) &= a_0 + a_1(x-x_0) + a_2(x-x_{-1})(x-x_0) + a_3(x-x_{-1})(x-x_0)(x-x_1)\\ &\quad +a_4(x-x_{-2})(x-x_{-1})(x-x_0)(x-x_1)\\ &\quad +a_5(x-x_{-2})(x-x_{-1})(x-x_0)(x-x_1)(x-x_2)+\cdots\\ &\quad +a_{2n-1}(x-x_{-(n-1)})\cdots(x-x_{-1})(x-x_0)\cdots(x-x_{n-1})\\ &\quad +a_{2n}(x-x_{-n})(x-x_{-(n-1)})\cdots(x-x_{-1})(x-x_0)\cdots(x-x_{n-1}). \qquad (3.49)\end{aligned}$$

The coefficients a_i's are unknown constants. These values are determined by using the relations (3.48). Substituting $x = x_0, x_{-1}, x_1, x_{-2}, x_2, \ldots, x_{-n}, x_n$ to (3.49) in succession. Note that $x_i - x_{-j} = (i+j)h$ and $(x_{-i} - x_j) = -(i+j)h$. Then it is found that

$$\begin{aligned} y_0 &= a_0\\ \phi(x_{-1}) &= a_0 + a_1(x_{-1}-x_0)\\ \text{i.e.,}\quad y_{-1} &= y_0 + a_1(-h),\\ \text{i.e.,}\ a_1 &= \frac{y_0 - y_{-1}}{h} = \frac{\Delta y_{-1}}{h}\end{aligned}$$

$$\begin{aligned}\phi(x_1) &= a_0 + a_1(x_1-x_0) + a_2(x_1-x_{-1})(x_1-x_0)\\ y_1 &= y_0 + h.\frac{\Delta y_{-1}}{h} + a_2(2h)(h)\\ \text{i.e.,}\quad a_2 &= \frac{y_1 - y_0 - (y_0 - y_{-1})}{2!h^2} = \frac{\Delta^2 y_{-1}}{2!h^2}\end{aligned}$$

$$\begin{aligned}\phi(x_{-2}) &= a_0 + a_1(x_{-2}-x_0) + a_2(x_{-2}-x_{-1})(x_{-2}-x_0)\\ &\quad +a_3(x_{-2}-x_{-1})(x_{-2}-x_0)(x_{-2}-x_1)\\ \text{i.e.,}\quad y_{-2} &= y_0 + \frac{\Delta y_{-1}}{h}(-2h) + \frac{\Delta^2 y_{-1}}{2!h^2}(-h)(-2h) + a_3(-h)(-2h)(-3h)\\ &= y_0 - 2(y_0 - y_{-1}) + (y_1 - 2y_0 + y_{-1}) + a_3(-1)^3(3!)h^3\\ \text{or,}\quad a_3 &= \frac{y_1 - 3y_0 + 3y_{-1} - y_{-2}}{3!h^3} = \frac{\Delta^3 y_{-2}}{3!h^3}.\end{aligned}$$

In this manner, the other values are obtained as

$$a_4 = \frac{\Delta^4 y_{-2}}{4!h^4}, a_5 = \frac{\Delta^5 y_{-3}}{5!h^5}, \ldots, a_{2n-1} = \frac{\Delta^{2n-1} y_{-n}}{(2n-1)!h^{2n-1}}, a_{2n} = \frac{\Delta^{2n} y_{-n}}{(2n)!h^{2n}}.$$

Making use of a_i's, equation (3.49) becomes

$$\begin{aligned}
\phi(x) &= y_0 + (x-x_0)\frac{\Delta y_{-1}}{1!h} + (x-x_{-1})(x-x_0)\frac{\Delta^2 y_{-1}}{2!h^2} \\
&\quad +(x-x_{-1})(x-x_0)(x-x_1)\frac{\Delta^3 y_{-2}}{3!h^3} \\
&\quad +(x-x_{-2})(x-x_{-1})(x-x_0)(x-x_1)\frac{\Delta^4 y_{-2}}{4!h^4} + \cdots \\
&\quad +(x-x_{-(n-1)})\cdots(x-x_{-1})(x-x_0)\cdots(x-x_{n-1})\frac{\Delta^{2n-1} y_{-n}}{(2n-1)!h^{2n-1}} \\
&\quad +(x-x_{-n})(x-x_{-1})(x-x_0)(x-x_1)\cdots(x-x_{n-1})\frac{\Delta^{2n} y_{-n}}{(2n)!h^{2n}}. \qquad (3.50)
\end{aligned}$$

As in previous case, a new unit less variable s is introduced to reduce the above formula into a simple form, where $s = \dfrac{x-x_0}{h}$ i.e, $x = x_0 + sh$.

Then

$$\begin{aligned}
\frac{x-x_i}{h} &= \frac{x-x_0-ih}{h} = s-i \text{ and} \\
\frac{x-x_{-i}}{h} &= \frac{x-x_0+ih}{h} = s+i, \quad i = 0,1,2,\ldots,n.
\end{aligned}$$

Then the above formula is transferred to

$$\begin{aligned}
\phi(x) &= y_0 + s\Delta y_{-1} + \frac{(s+1)s}{2!}\Delta^2 y_{-1} + \frac{(s+1)s(s-1)}{3!}\Delta^3 y_{-2} \\
&\quad +\frac{(s+2)(s+1)s(s-1)}{4!}\Delta^4 y_{-2} + \cdots \\
&\quad +\frac{(s+n-1)\cdots(s+1)s(s-1)\cdots(s-n+1)}{(2n-1)!}\Delta^{2n-1} y_{-n} \\
&\quad +\frac{(s+n)(s+n-1)\cdots(s+1)s(s-1)\cdots(s-n+1)}{(2n)!}\Delta^{2n} y_{-n}. \qquad (3.51)
\end{aligned}$$

The above formula (3.51) is known as **Gauss's backward interpolation formula** or **second interpolation formula of Gauss**.

Case II: For $2n$ (even) number of arguments

In this case the arguments are taken as $x_0, x_{\pm 1}, \ldots, x_{\pm(n-1)}$ and x_{-n}, where $x_{\pm i} = x_0 \pm ih$, $i = 0, 1, \ldots, n-1$ and $x_{-n} = x_0 - nh$.

For these equispaced points the Gauss's backward interpolation formula is

$$\begin{aligned}\phi(x) &= y_0 + s\Delta y_{-1} + \frac{(s+1)s}{2!}\Delta^2 y_{-1} + \frac{(s+1)s(s-1)}{3!}\Delta^3 y_{-2} \\ &\quad + \frac{(s+2)(s+1)s(s-1)}{4!}\Delta^4 y_{-2} + \cdots \\ &\quad + \frac{(s+n-1)\cdots(s+1)s(s-1)\cdots(s-n+1)}{(2n-1)!}\Delta^{2n-1} y_{-n}, \qquad (3.52)\end{aligned}$$

where $s = \dfrac{x-x_0}{h}$.

3.7.4 Remainder of Gauss's backward central difference formula

The remainder for $(2n+1)$ equispaced points is

$$\begin{aligned}E(x) &= (x-x_{-n})(x-x_{-(n-1)})\cdots(x-x_{-1})(x-x_0)\cdots(x-x_n)\frac{f^{2n+1}(\xi)}{(2n+1)!} \\ &= (s+n)(s+n-1)\cdots(s+1)s(s-1)\cdots(s-n+1)(s-n) \\ &\qquad\qquad \times\ h^{2n+1}\frac{f^{2n+1}(\xi)}{(2n+1)!},\end{aligned}$$

where $\min\{x_{-n}, x_{-(n-1)}, \ldots, x_0,\, x_1, \ldots, x_{n-1}, x_n\} < \xi$
$< \max\{x_{-n}, x_{-(n-1)}, \ldots, x_0, x_1, \ldots, x_{n-1}, x_n\}$.

The remainder for the case of $2n$ equispaced points is

$$\begin{aligned}E(x) &= (x-x_{-n})(x-x_{-(n-1)})\cdots(x-x_{-1})(x-x_0)\cdots(x-x_{n-1})\frac{f^{2n}(\xi)}{(2n)!} \\ &= (s+n)(s+n-1)\cdots(s+1)s(s-1)\cdots(s-n+1)h^{2n}\frac{f^{2n}(\xi)}{(2n)!},\end{aligned}$$

$\min\{x_{-n}, x_{-n}, \ldots, x_0,\, x_1, \ldots, x_{n-1}, x_{n-1}\} < \xi$
$< \max\{x_{-n}, x_{-(n-1)}, \ldots, x_0, x_1, \ldots, x_{n-1}\}$.

3.8 Stirling's Interpolation Formula

Stirling's interpolation formula is used for odd number of equispaced arguments.

This formula is obtained by taking the arithmetic mean of the Gauss's forward and backward difference formulae given by (3.44) and (3.51).

Therefore Stirling's formula is

$$\begin{aligned}
\phi(x) &= \frac{\phi(x)_{\text{forward}} + \phi(x)_{\text{backward}}}{2} \\
&= y_0 + \frac{s}{1!}\frac{\Delta y_{-1} + \Delta y_0}{2} + \frac{s^2}{2!}\Delta^2 y_{-1} + \frac{s(s^2-1^2)}{3!}\frac{\Delta^3 y_{-2} + \Delta^3 y_{-1}}{2} \\
&\quad + \frac{s^2(s^2-1^2)}{4!}\Delta^4 y_{-2} + \frac{s(s^2-1^2)(s^2-2^2)}{5!}\frac{\Delta^5 y_{-3} + \Delta^5 y_{-2}}{2} + \cdots \\
&\quad + \frac{s^2(s^2-1^2)(s^2-2^2)\cdots(s^2-\overline{n-1}^2)}{(2n)!}\Delta^{2n} y_{-n}. \qquad (3.53)
\end{aligned}$$

The remainder of this formula is

$$E(x) = \frac{s(s^2-1^2)(s^2-2^2)\cdots(s^2-n^2)}{(2n+1)} h^{2n+1} f^{2n+1}(\xi), \qquad (3.54)$$

where $\min\{x_{-n}, \ldots, x_0, \ldots, x_n\} < \xi < \max\{x_{-n}, \ldots, x_0, \ldots, x_n\}$.

The formula (3.53) is known as **Stirling's central difference interpolation formula**.

Note 3.8.1 (a) The Stirling's interpolation formula (3.53) gives the best approximate result when $-0.25 < s < 0.25$. So we choose x_0 in such a way that $s = \dfrac{x - x_0}{h}$ satisfy this condition.

(b) The Stirling's interpolation formula is used when the point x, for which $f(x)$ to be determined, is at the centre of the table and the number of points at which the values of $f(x)$ known is *odd*.

3.9 Bessel's Interpolation Formula

This central difference formula is also obtained by taking the arithmetic mean of Gauss's forward and backward interpolation formulae. But, one difference is that the backward formula taken after one modification.

Let us consider $2n$ equispaced points $x_{-(n-1)}, \ldots, x_{-1}, x_0, x_1, \ldots, x_{n-1}, x_n$ as arguments, where $x_{\pm i} = x_0 \pm ih$, h is the spacing.

If x_0, y_0 be the initial values of x and y respectively, then the Gauss's backward difference interpolation formula (3.52) is

$$\begin{aligned}
\phi(x) &= y_0 + s\Delta y_{-1} + \frac{s(s+1)}{2!}\Delta^2 y_{-1} + \frac{(s+1)s(s-1)}{3!}\Delta^3 y_{-2} \\
&\quad + \frac{(s+2)(s+1)s(s-1)}{4!}\Delta^4 y_{-2} + \cdots \\
&\quad + \frac{(s+n-1)\cdots(s+1)s(s-1)\cdots(s-n+1)}{(2n-1)!}\Delta^{2n-1} y_{-n}. \qquad (3.55)
\end{aligned}$$

Suppose x_1, y_1 be the initial values of x and y. Then

$$\frac{x-x_1}{h} = \frac{x-(x_0+h)}{h} = \frac{x-x_0}{h} - 1 = s-1.$$

Also, the indices of all the differences of (3.55) will increase by 1. Now, replacing s by $s-1$ and increasing the indices of (3.55) by 1, then the above equation becomes

$$\begin{aligned}\phi_1(x) &= y_1 + (s-1)\Delta y_0 + \frac{s(s-1)}{2!}\Delta^2 y_0 + \frac{s(s-1)(s-2)}{3!}\Delta^3 y_{-1}\\ &\quad + \frac{(s+1)s(s-1)(s-2)}{4!}\Delta^4 y_{-1}\\ &\quad + \frac{(s+1)s(s-1)(s-2)(s-3)}{5!}\Delta^5 y_{-2} + \cdots\\ &\quad + \frac{(s+n-2)\cdots(s+1)s(s-1)(s-2)\cdots(s-n)}{(2n-1)!}\Delta^{2n-1} y_{-n+1}. \qquad (3.56)\end{aligned}$$

Taking arithmetic mean of (3.56) and Gauss's forward interpolation formula given by (3.45),

$$\begin{aligned}\phi(x) &= \frac{\phi_1(x) + \phi(x)_{\text{forward}}}{2}\\ &= \frac{y_0+y_1}{2} + \Big(s-\frac{1}{2}\Big)\Delta y_0 + \frac{s(s-1)}{2!}\frac{\Delta^2 y_0 + \Delta^2 y_{-1}}{2}\\ &\quad + \frac{(s-\frac{1}{2})s(s-1)}{3!}\Delta^3 y_{-1} + \frac{s(s-1)(s+1)(s-2)}{4!}\frac{\Delta^4 y_{-2} + \Delta^4 y_{-1}}{2}\\ &\quad + \frac{(s-\frac{1}{2})s(s-1)(s+1)(s-2)}{5!}\Delta^5 y_{-2} + \cdots\\ &\quad + \frac{(s-\frac{1}{2})s(s-1)(s+1)\cdots(s+n-2)(s-\overline{n-1})}{(2n-1)!}\Delta^{2n-1} y_{-(n-1)}, \qquad (3.57)\end{aligned}$$

where $s = \dfrac{x-x_0}{h}$.

Introducing $u = s - \dfrac{1}{2} = \dfrac{x-x_0}{h} - \dfrac{1}{2}$ and then the above formula reduces to

$$\begin{aligned}\phi(x) &= \frac{y_0+y_1}{2} + u\Delta y_0 + \frac{u^2-\frac{1}{4}}{2!}\cdot\frac{\Delta^2 y_{-1} + \Delta^2 y_0}{2} + \frac{u(u^2-\frac{1}{4})}{3!}\Delta^3 y_{-1}\\ &\quad + \frac{(u^2-\frac{1}{4})(u^2-\frac{9}{4})\cdots(u^2-\frac{(2n-3)^2}{4})}{(2n-1)!}\Delta^{2n-1} y_{-(n-1)}. \qquad (3.58)\end{aligned}$$

The formulae given by (3.57) and (3.58) are known as Bessel's central difference interpolation formula.

Note 3.9.1 (a) The Bessel's formula gives the best result when the starting point x_0 be so chosen such that $-0.25 < u < 0.25$ i.e., $-0.25 < s-0.5 < 0.25$ or, $0.25 < s < 0.75$.

(b) This central difference formula is used when the interpolating point is near the middle of the table and the number of arguments is even.

Example 3.9.1 Use the central difference interpolation formula of Stirling or Bessel to find the values of y at (i) $x = 1.40$ and (ii) $x = 1.60$ from the following table

x	:	1.0	1.25	1.50	1.75	2.00
y	:	1.0000	1.0772	1.1447	1.2051	2.2599

Solution. The central difference table is

i	x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$
–2	1.00	1.0000			
			0.0772		
–1	1.25	1.0772		–0.0097	
			0.0675		0.0026
0	1.50	1.1447		–0.0071	
			0.0604		0.0015
1	1.75	1.2051		–0.0056	
			0.0548		
2	2.00	1.2599			

(i) For $x = 1.40$, we take $x_0 = 1.50$, then $s = (1.40 - 1.50)/0.25 = -0.4$.
The Bessel's formula gives

$$\begin{aligned} y(1.40) &= \frac{y_0 + y_1}{2} + \left(s - \frac{1}{2}\right)\Delta y_0 + \frac{s(s-1)}{2!}\frac{\Delta^2 y_0 + \Delta^2 y_{-1}}{2} \\ &\quad + \frac{1}{3!}\left(s - \frac{1}{2}\right)s(s-1)\Delta^3 y_{-1} \\ &= \frac{1.1447 + 1.2051}{2} + (-0.4 - 0.5) \times 0.0604 \\ &\quad + \frac{-0.4(-0.4-1)}{2!}\frac{-0.0071 - 0.0056}{2} \\ &\quad + \frac{1}{6}(-0.4 - 0.5)(-0.4)(-0.4-1) \times 0.0015 \\ &= 1.118636. \end{aligned}$$

(ii) For $x = 1.60$, we take $x_0 = 1.50$, then $s = (1.60 - 1.50)/0.25 = 0.4$.

Using Stirling's formula

$$\begin{aligned} y(1.60) &= y_0 + s\frac{\Delta y_{-1}+\Delta y_0}{2} + \frac{s^2}{2!}\Delta^2 y_{-1} + \frac{s(s^2-1^2)}{3!}\frac{\Delta^3 y_{-2}+\Delta^3 y_{-1}}{2} \\ &= 1.1447 + 0.4\frac{0.0675+0.0604}{2} + \frac{(0.4)^2}{2}\times(-0.0071) \\ &\quad + \frac{0.4(0.16-1)}{6}\frac{0.0026+0.0015}{2} \\ &= 1.1447 + 0.02558 - 0.000568 - 0.0001148 = 1.1695972. \end{aligned}$$

3.10 Everett's Interpolation Formula

In this interpolation formula, $2n$ (even) equispaced arguments $x_{-(n-1)}$, $x_{-(n-2)}$, $\ldots$, x_{-1}, x_0, x_1, $\ldots, x_{n-1}$ and x_n are considered. The Everett's interpolation formula is obtained from Gauss's forward interpolation formula (3.44) by replacing the odd order differences using the lower order even differences. That is, by the substitution

$$\Delta^{2k+1} y_{-k} = \Delta^{2k} y_{-(k-1)} - \Delta^{2k} y_{-k}.$$

That is,

$$\begin{aligned} \Delta y_0 &= y_1 - y_0; \\ \Delta^3 y_{-1} &= \Delta^2 y_0 - \Delta^2 y_{-1}; \\ \Delta^5 y_{-2} &= \Delta^4 y_{-1} - \Delta^4 y_{-2}, \\ \vdots \quad & \quad \vdots \qquad\qquad \vdots \\ \Delta^{2n-1} y_{-(n-1)} &= \Delta^{2n-2} y_{-(n-2)} - \Delta^{2n-2} y_{-(n-1)}. \end{aligned}$$

On substitution of these relations, equation (3.44) yields

$$\begin{aligned} \phi(x) &= y_0 + s(y_1 - y_0) + \frac{s(s-1)}{2!}\Delta^2 y_{-1} + \frac{(s+1)s(s-1)}{3!}(\Delta^2 y_0 - \Delta^2 y_{-1}) \\ &\quad + \frac{(s+1)s(s-1)(s-2)}{4!}\Delta^4 y_{-2} \\ &\quad + \frac{(s+2)(s+1)s(s-1)(s-2)}{5!}(\Delta^4 y_{-1} - \Delta^4 y_{-2}) + \cdots \\ &\quad + \frac{(s+n-1)(s+n-2)\cdots(s+1)s(s-1)(s-n+1)}{(2n-1)!} \\ &\qquad \times(\Delta^{2n-2} y_{-(n-2)} - \Delta^{2n-2} y_{-(n-1)}) \\ &= \left[sy_1 + \frac{(s+1)s(s-1)}{3!}\Delta^2 y_0 + \frac{(s+2)(s+1)s(s-1)(s-2)}{5!}\Delta^4 y_{-1} + \cdots \right. \end{aligned}$$

$$+\frac{(s+n-1)(s+n-2)\cdots(s+1)s(s-1)\cdots(s-n+1)}{(2n-1)!}\Delta^{2n-2}y_{-(n-2)}\Bigg]$$
$$+\Bigg[(1-s)y_0+\left\{\frac{s(s-1)}{2!}-\frac{(s+1)s(s-1)}{3!}\right\}\Delta^2 y_{-1}$$
$$+\left\{\frac{(s+1)s(s-1)(s-2)}{4!}-\frac{(s+2)(s+1)s(s-1)(s-2)}{5!}\right\}\Delta^4 y_{-2}+\cdots$$
$$+\frac{(s+n-1)(s+n-2)\cdots(s+1)s(s-1)\cdots(s-n+1)}{(2n-1)!}\Delta^{2n-2}y_{-(n-1)}\Bigg]$$
$$=\Bigg[sy_1+\frac{s(s^2-1^2)}{3!}\Delta^2 y_0+\frac{s(s^2-1^1)(s^2-2^2)}{5!}\Delta^4 y_{-1}+\cdots$$
$$+\frac{s(s^2-1^2)(s^2-2^2)\cdots(s^2-(n-1)^2)}{(2n-1)!}\Delta^{2n-2}y_{-(n-2)}\Bigg]$$
$$+\Bigg[uy_0+\frac{u(u^2-1^1)}{3!}\Delta^2 y_{-1}+\frac{u(u^2-1^2)(u^2-2^2)}{5!}\Delta^4 y_{-2}+\cdots$$
$$+\frac{u(u^2-1^2)(u^2-2^2)\cdots(u^2-(n-1)^2)}{(2n-1)!}\Delta^{2n-2}y_{-(n-1)}\Bigg], \quad (3.59)$$

where $s=\dfrac{x-x_0}{h}$ and $u=1-s$.

Example 3.10.1 Use Everett's interpolation formula to find the value of y when $x = 1.60$ from the following table.

x	:	1.0	1.25	1.50	1.75	2.00	2.25
y	:	1.0000	1.1180	1.2247	1.3229	1.4142	1.5000

Solution. The difference table is

i	x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$
−2	1.00	1.0000				
			0.1180			
−1	1.25	1.1180		−0.0113		
			0.1067		0.0028	
0	1.50	1.2247		−0.0085		−0.0012
			0.0982		0.0016	
1	1.75	1.3229		−0.0069		−0.0002
			0.0913		0.0014	
2	2.00	1.4142		−0.0055		
			0.0858			
3	2.25	1.5000				

We take $x_0 = 1.50$. Here $h = 0.25$.
Then $s = \dfrac{x - x_0}{h} = \dfrac{1.60 - 1.50}{0.25} = 0.4$ and $u = 1 - s = 0.6$.
Using Everett's formula, the value of $y(1.60)$ is given by

$$\begin{aligned}
y(1.60) &= \left[sy_1 + \frac{s(s^2-1^2)}{3!}\Delta^2 y_0 + \frac{s(s^2-1^2)(s^2-2^2)}{5!}\Delta^4 y_{-1}\right] \\
&\quad + \left[uy_0 + \frac{u(u^2-1^2)}{3!}\Delta^2 y_{-1} + \frac{u(u^2-1^2)(u^2-2^2)}{5!}\Delta^4 y_{-2}\right] \\
&= \left[0.4 \times 1.3229 + \frac{0.4(0.16-1)}{6}(-0.0069)\right. \\
&\quad \left. + \frac{0.4(0.16-1)(0.16-4)}{120}(-0.0002)\right] + \left[0.6 \times 1.2247\right. \\
&\quad \left. + \frac{0.6(0.36-1)}{6}(-0.0085) + \frac{0.6(0.36-1)(0.36-4)}{120}(-0.0012)\right] \\
&= 0.5292 + 0.0004 - 0.0000025 + 0.7348 + 0.0005 - 0.00001 \\
&= 1.2649.
\end{aligned}$$

3.10.1 Relation between Bessel's and Everett's formulae

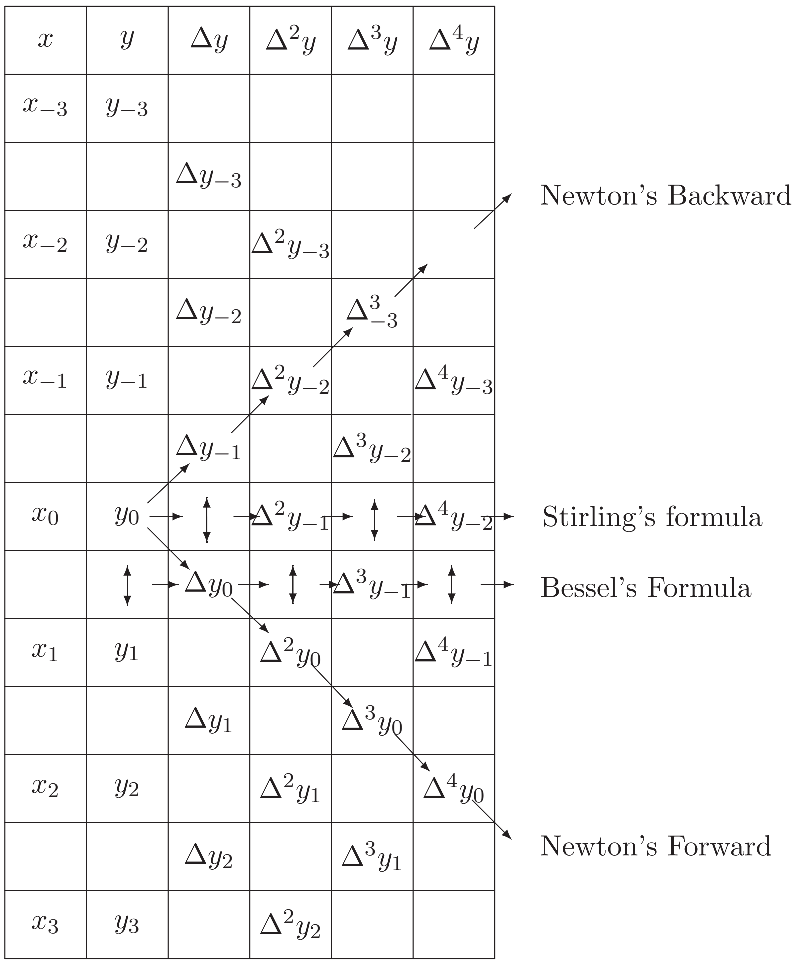
These two formulae are closely related and one formula can be deduced from the other one. Now, starting from Bessel's formula (3.57):

$$\begin{aligned}
\phi(x) &= \frac{y_0 + y_1}{2} + \left(s - \frac{1}{2}\right)\Delta y_0 + \frac{s(s-1)}{2!}\frac{\Delta^2 y_0 + \Delta^2 y_{-1}}{2} \\
&\quad + \frac{\left(s - \frac{1}{2}\right)s(s-1)}{3!}\Delta^3 y_{-1} + \cdots, \text{ where } s = \frac{x - x_0}{h}. \\
&= \frac{y_0 + y_1}{2} + \left(s - \frac{1}{2}\right)(y_1 - y_0) + \frac{s(s-1)}{2!}\frac{\Delta^2 y_0 + \Delta^2 y_{-1}}{2} \\
&\quad + \frac{\left(s - \frac{1}{2}\right)s(s-1)}{3!}(\Delta^2 y_0 - \Delta^2 y_{-1}) + \cdots \\
&= (1-s)y_0 + \left[\frac{s(s-1)}{4} - \frac{s(s-1)(s-\frac{1}{2})}{6}\right]\Delta^2 y_{-1} + \cdots \\
&\quad + sy_1 + \left[\frac{s(s-1)}{4} + \frac{s(s-1)(s-\frac{1}{2})}{6}\right]\Delta^2 y_0 + \cdots \\
&= sy_1 + \frac{s(s^2-1^2)}{3!}\Delta^2 y_0 + \cdots + uy_0 + \frac{u(u^2-1^2)}{3!}\Delta^2 y_{-1} + \cdots \\
&\quad \text{where } u = 1 - s.
\end{aligned}$$

This is the Everett's formula up to second order differences. From this deduction it is clear that the Everett's formula truncated after second order differences is equivalent to the Bessel's formula truncated after third differences. Conversely, the Bessel's formula may be deduced from Everett's formula.

Central difference table

The difference Table 3.3 shows how the different order of differences are used in different interpolation formulae.



x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
x_{-3}	y_{-3}				
		Δy_{-3}			
x_{-2}	y_{-2}		$\Delta^2 y_{-3}$		
		Δy_{-2}		Δ^3_{-3}	
x_{-1}	y_{-1}		$\Delta^2 y_{-2}$		$\Delta^4 y_{-3}$
		Δy_{-1}		$\Delta^3 y_{-2}$	
x_0	y_0		$\Delta^2 y_{-1}$		$\Delta^4 y_{-2}$
		Δy_0		$\Delta^3 y_{-1}$	
x_1	y_1		$\Delta^2 y_0$		$\Delta^4 y_{-1}$
		Δy_1		$\Delta^3 y_0$	
x_2	y_2		$\Delta^2 y_1$		$\Delta^4 y_0$
		Δy_2		$\Delta^3 y_1$	
x_3	y_3		$\Delta^2 y_2$		

Figure 3.3: Central difference table.

Note 3.10.1 In Newton's forward and backward interpolation formulae the first or the last interpolating point is taken as initial point x_0. But, in central difference interpolation formulae, a middle point is taken as the initial point x_0.

3.11 Interpolation by Iteration (Aitken's Interpolation)

Newton's interpolation formula generates successively higher order interpolation formula. The **Aitken** interpolation formula served the same purpose. But it has one advantage that it can be easily programmed for a computer.

Let $y = f(x)$ be given for $n+1$ distinct points $x_0, x_1, \ldots, x_n$, i.e., $y_i = f(x_i)$, $i = 0, 1, \ldots, n$ are given, where the points $x_i, i = 0, 1, \ldots, n$ need not be equispaced. To compute the value of y for a given x the iterations proceed as follows:
to find the value of y obtain a first approximation by taking first two points; then obtain its second approximation by taking the first approximations and so on.

The linear polynomial for the points x_0 and x_1 is

$$\begin{aligned} p_{01}(x) &= \frac{x-x_1}{x_0-x_1}y_0 + \frac{x-x_0}{x_1-x_0}y_1 = \frac{1}{x_1-x_0}[(x_1-x)y_0 - (x_0-x)y_1] \\ &= \frac{1}{x_1-x_0}\begin{vmatrix} y_0 & x_0-x \\ y_1 & x_1-x \end{vmatrix}. \end{aligned} \tag{3.60}$$

In general,

$$p_{0j}(x) = \frac{1}{x_j-x_0}\begin{vmatrix} y_0 & x_0-x \\ y_j & x_j-x \end{vmatrix}, j = 1, 2, \ldots, n. \tag{3.61}$$

Here $p_{0j}(x)$ is a polynomial of degree less than or equal to 1, for the points x_0 and x_j.

The polynomial

$$p_{01j}(x) = \frac{1}{x_j-x_1}\begin{vmatrix} p_{01}(x) & x_1-x \\ p_{0j}(x) & x_j-x \end{vmatrix}, j = 2, 3, \ldots, n. \tag{3.62}$$

is a polynomial of degree less than or equal to 2.

The polynomial $p_{01j}(x)$ interpolates the points x_0, x_1 and x_j.

In general, the polynomial for the $(k+1)$ points $x_0, x_1, \ldots, x_k$ and x_j is

$$p_{012\cdots kj}(x) = \frac{1}{x_j-x_k}\begin{vmatrix} p_{012\cdots k}(x) & x_k-x \\ p_{012\cdots(k-1)j}(x) & x_j-x \end{vmatrix}, j = k+1, \ldots, n. \tag{3.63}$$

This polynomial is of degree $(k+1)$ and interpolates the points $x_0, x_1, \ldots, x_{k-1}, x_k$ and x_j.

The tabular representation of this form is

x_j	y_j	p_{0j}	p_{01j}	p_{012j}	p_{0123j}	$x_j - x$
x_0	y_0					$x_0 - x$
x_1	y_1	p_{01}				$x_1 - x$
x_2	y_2	p_{02}	p_{012}			$x_2 - x$
x_3	y_3	p_{03}	p_{013}	p_{0123}		$x_3 - x$
x_4	y_4	p_{04}	p_{014}	p_{0124}	p_{01234}	$x_4 - x$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
x_n	y_n	p_{0n}	p_{01n}	p_{012n}	p_{0123n}	$x_n - x$

Example 3.11.1 Find the value of $y(1.52)$ by iterated linear interpolation using the following table.

x	:	1.4	1.5	1.6	1.7
$y(x)$	:	1.8330	1.9365	2.0396	2.1424

Solution. Here $x = 1.52$.
The calculations are shown below.

x_j	y_j	p_{0j}	p_{01j}	p_{012j}	$x_j - x$
1.4	1.8330				−0.12
1.5	1.9365	1.9572			−0.02
1.6	2.0396	1.9570	1.9572		0.08
1.7	2.1424	1.9568	1.9572	1.9572	0.18

Now

$$p_{0j} = \frac{1}{x_j - x_0}\begin{vmatrix} y_0 & x_0 - x \\ y_j & x_j - x \end{vmatrix}, \qquad j = 1, 2, 3.$$

$$p_{01} = \frac{1}{0.1}\begin{vmatrix} 1.8330 & -0.12 \\ 1.9365 & -0.02 \end{vmatrix} = 1.9572.$$

$$p_{02} = \frac{1}{0.2}\begin{vmatrix} 1.8330 & -0.12 \\ 2.0396 & 0.08 \end{vmatrix} = 1.9570.$$

$$p_{03} = \frac{1}{0.3}\begin{vmatrix} 1.8330 & -0.12 \\ 2.1424 & 0.18 \end{vmatrix} = 1.9568.$$

$$p_{01j} = \frac{1}{x_j - x_1}\begin{vmatrix} p_{01} & x_1 - x \\ p_{0j} & x_j - x \end{vmatrix}, \qquad j = 2, 3.$$

$$p_{012} = \frac{1}{0.1}\begin{vmatrix} 1.9572 & -0.02 \\ 1.9570 & 0.08 \end{vmatrix} = 1.9572.$$

$$p_{013} = \frac{1}{0.2}\begin{vmatrix} 1.9572 & -0.02 \\ 1.9568 & 0.18 \end{vmatrix} = 1.9572.$$

$$p_{012j} = \frac{1}{x_j - x_2}\begin{vmatrix} p_{012} & x_2 - x \\ p_{01j} & x_j - x \end{vmatrix}, \qquad j = 3.$$

$$p_{0123} = \frac{1}{0.1}\begin{vmatrix} 1.9572 & 0.08 \\ 1.9572 & 0.18 \end{vmatrix} = 1.9572.$$

Therefore the interpolated value of y at $x = 1.52$ is 1.9572.

Algorithm 3.3 (Aitken Interpolation). This algorithm determines the value of y at a given x from the table of $(x_i, y_i), i = 0, 1, 2, \ldots, n$, by Aitken's iterative interpolation formula.

```
**Algorithm Aitken_Interpolation**
**Step 1:** Read $n$, $x_0, x_1, \ldots, x_n$; $y_0, y_1, \ldots, y_n$ and $xg$;
          // $n$ represents the number of points, $xg$ is the value of
          $x$ at which $y$ is to be calculated.//
**Step 2:** for $j = 0$ to $n$ do      // reading of tabulated values//
          Step 2.1. Set $p(j, 0) = y(j)$;
          Step 2.2. Compute $xd(j) = x(j) - xg$;
      endfor;
**Step 3:** for $k = 0$ to $n$ do
          for $j = k + 1$ to $n$ do
            $p(j, k+1) = [p(k, k) * xd(j) - p(j, k) * xd(k)]/[x(j) - x(k)]$;
      endfor;
**Step 4:** // printing of the table//
      for $j = 0$ to $n$ do
          Print $x(j)$, $(p(j, k), k = 0$ to $j)$, $xd(j)$;
      endfor;
**Step 5:** //Printing of the value of the polynomial.//
      Print $p(n, n)$;
**end Aitken_Interpolation**
```

Program 3.3

```
/* Program Aitken Interpolation
   This program implements Aitken's  interpolation
   formula; xg is the interpolating points. */
#include <stdio.h>
#include <math.h>
void main()
{
 int n, j, k;
 float x[20], y[20], xg, p[20][20], xd[20];
 printf("Enter the value of n and the data points
         in the form x[i],y[i] ");
 scanf("%d", &n);
 for(j=0;j<=n;j++) scanf("%f %f", &x[j],&y[j]);
 printf("\nEnter the value of x ");
 scanf("%f",&xg);
 for(j=0;j<=n;j++){
      p[j][0]=y[j]; xd[j]=x[j]-xg;
    }
 for(k=0;k<=n;k++)
    for(j=k+1;j<=n;j++)
       p[j][k+1]=(p[k][k]*xd[j]-p[j][k]*xd[k])/(x[j]-x[k]);
 for(j=0;j<=n;j++){
      printf("%6.4f ",x[j]);
      for(k=0;k<=j;k++) printf(" %6.4f",p[j][k]);
      for(k=j+1;k<=n;k++) printf("        ");
      printf(" %6.4f\n",xd[j]);
    }
 printf("The value of y at x= %6.3f is %8.5f ",xg,p[n][n]);
} /* main */
```

A sample of input/output:

```
Enter the value of n and the data points in the form x[i],y[i] 4
1921 46
1931 68
1941 83
1951 95
1961 105
Enter the value of x 1955
1921.0000  46.0000                                  -34.0000
1931.0000  68.0000 120.8000                         -24.0000
1941.0000  83.0000 108.9000 92.2400                 -14.0000
1951.0000  95.0000 101.5333 97.6800 99.8560         -4.0000
1961.0000  105.0000 96.1500 101.0800 98.4280 99.2848 6.0000
The value of y at x= 1955.000 is 99.28481
```

3.12 Divided Differences and their Properties

The Lagrange's interpolation formula has a disadvantage that if a new interpolation point is added or removed then the Lagrangian functions $L_i(x)$ will have to be re-computed. But the **Newton's general interpolation formula**, based on **divided differences** removes this drawback.

Let $y_i = f(x_i)$, $i = 0, 1, \ldots, n$ be known at $n + 1$ points $x_0, x_1, \ldots, x_n$. The points $x_0, x_1, \ldots, x_n$ are not necessarily be equispaced. Then the divided differences of different orders are defined as follows:

Zeroth order divided difference

$$f[x_0] = f(x_0).$$

First order divided difference

$$f[x_0, x_1] = \frac{f(x_0) - f(x_1)}{x_0 - x_1}.$$

In general, $f[x_i, x_j] = \dfrac{f(x_i) - f(x_j)}{x_i - x_j}$.

Second order divided difference

$$f[x_0, x_1, x_2] = \frac{f[x_0, x_1] - f[x_1, x_2]}{x_0 - x_2}.$$

In general, $f[x_i, x_j, x_k] = \dfrac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k}$.

nth order divided differences

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_0, x_1, \ldots, x_{n-1}] - f[x_1, x_2, \ldots, x_n]}{x_0 - x_n}.$$

3.12.1 Properties of divided differences

1. *Divided difference of a constant is zero*

   Let $f(x) = c$.

   Then $f[x_0, x_1] = \dfrac{f(x_0) - f(x_1)}{x_0 - x_1} = \dfrac{c - c}{x_0 - x_1} = 0$.

2. *Divided difference of $cf(x)$, c is constant, is the divided difference of $f(x)$ multiplied by c*

   Let $g(x) = cf(x)$.

   Therefore,

$$g[x_0, x_1] = \frac{g(x_0) - g(x_1)}{x_0 - x_1} = \frac{cf(x_0) - cf(x_1)}{x_0 - x_1} = c\frac{f(x_0) - f(x_1)}{x_0 - x_1} = cf[x_0, x_1].$$

3. *Divided difference is linear*

   Let $h(x) = af(x) + bg(x)$.

   Now,

$$\begin{aligned} h[x_0, x_1] &= \frac{h(x_0) - h(x_1)}{x_0 - x_1} = \frac{af(x_0) + bg(x_0) - af(x_1) - bg(x_1)}{x_0 - x_1} \\ &= a\frac{f(x_0) - f(x_1)}{x_0 - x_1} + b\frac{g(x_0) - g(x_1)}{x_0 - x_1} \\ &= af[x_0, x_1] + bg[x_0, x_1]. \end{aligned}$$

   That is, divided difference obeys linear property.

4. *Divided differences are symmetric functions*

   The first order divided difference is

$$f[x_0, x_1] = \frac{f(x_0) - f(x_1)}{x_0 - x_1} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f[x_1, x_0].$$

   That is, first order difference is symmetric.

   Also, $f[x_0, x_1] = \dfrac{1}{x_0 - x_1}f(x_0) + \dfrac{1}{x_1 - x_0}f(x_1)$.

   The second order difference is

$$\begin{aligned} f[x_0, x_1, x_2] &= \frac{f[x_0, x_1] - f[x_1, x_2]}{x_0 - x_2} \\ &= \frac{1}{x_0 - x_2}\Big[\Big\{\frac{1}{x_0 - x_1}f(x_0) + \frac{1}{x_1 - x_0}f(x_1)\Big\} \\ &\qquad - \Big\{\frac{1}{x_1 - x_2}f(x_1) + \frac{1}{x_2 - x_1}f(x_2)\Big\}\Big] \\ &= \frac{1}{(x_0 - x_2)(x_0 - x_1)}f(x_0) \\ &\quad + \frac{1}{(x_1 - x_0)(x_1 - x_2)}f(x_1) + \frac{1}{(x_2 - x_0)(x_2 - x_1)}f(x_2). \end{aligned}$$

Similarly, it can be shown that

$$\begin{aligned} f[x_0, x_1, \ldots, x_n] &= \frac{1}{(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_n)} f(x_0) \\ &+ \frac{1}{(x_1 - x_0)(x_1 - x_2)\cdots(x_1 - x_n)} f(x_1) + \cdots \\ &+ \frac{1}{(x_n - x_0)(x_n - x_1)\cdots(x_n - x_{n-1})} f(x_n) \\ &= \sum_{i=0}^{n} \frac{f(x_i)}{(x_i - x_0)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)}. \end{aligned} \tag{3.64}$$

From these relations it is easy to observe that the divided differences are symmetric.

5. *For equispaced arguments, the divided differences can be expressed in terms of forward differences.*

   That is,

$$f[x_0, x_1, \ldots, x_n] = \frac{1}{h^n . n!} \Delta^n y_0.$$

   In this case, $x_i = x_0 + ih, i = 0, 1, \ldots, n$.

$$\begin{aligned} f[x_0, x_1] &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y_0}{h}. \\ f[x_0, x_1, x_2] &= \frac{f[x_0, x_1] - f[x_1, x_2]}{x_0 - x_2} = \frac{1}{-2h}\Big(\frac{\Delta y_0}{h} - \frac{\Delta y_1}{h}\Big) \\ &= \frac{\Delta y_1 - \Delta y_0}{2h^2} = \frac{\Delta^2 y_0}{2!h^2}. \end{aligned}$$

   The result is true for $n = 1, 2$. Let the result be true for

$$n = k, \text{ i.e., } f[x_0, x_1, \ldots, x_k] = \frac{\Delta^k y_0}{k!\; h^k}.$$

   Now,

$$\begin{aligned} f[x_0, x_1, \ldots, x_k, x_{k+1}] &= \frac{f[x_0, x_1, \ldots, x_k] - f[x_1, x_2, \ldots, x_{k+1}]}{x_0 - x_{k+1}} \\ &= \frac{1}{-(x_{k+1} - x_0)}\Big[\frac{\Delta^k y_0}{k!\; h^k} - \frac{\Delta^k y_1}{k!\; h^k}\Big] \\ &= \frac{1}{(k+1)k!\; h^{k+1}}[\Delta^k y_1 - \Delta^k y_0] \\ &= \frac{\Delta^{k+1} y_0}{(k+1)!\; h^{k+1}}. \end{aligned}$$

Hence, by mathematical induction

$$f[x_0, x_1, \ldots, x_n] = \frac{1}{h^n.n!}\Delta^n y_0. \tag{3.65}$$

6. *Divided differences for equal arguments or divided differences for confluent arguments.*

If the arguments are equal then the divided differences have a meaning. If two arguments are equal then the divided difference has no meaning as denominator becomes zero. But, by limiting process one can define the divided differences for equal arguments which is known as **confluent divided differences**.

$$f[x_0, x_0] = \lim_{\varepsilon\to 0} f[x_0, x_0+\varepsilon] = \lim_{\varepsilon\to 0}\frac{f(x_0+\varepsilon)-f(x_0)}{\varepsilon} = f'(x_0),$$

provided $f(x)$ is differentiable.

$$\begin{aligned} f[x_0, x_0, x_0] &= \lim_{\varepsilon\to 0} f[x_0, x_0, x_0+\varepsilon] = \lim_{\varepsilon\to 0}\frac{f[x_0,x_0]-f[x_0,x_0+\varepsilon]}{\varepsilon} \\ &= \lim_{\varepsilon\to 0}\frac{f'(x_0)-\frac{f(x_0+\varepsilon)-f(x_0)}{\varepsilon}}{\varepsilon} \\ &= \lim_{\varepsilon\to 0}\frac{\varepsilon f'(x_0)-f(x_0+\varepsilon)+f(x_0)}{\varepsilon^2} \quad \left(\frac{0}{0}\text{ form}\right) \\ &= \lim_{\varepsilon\to 0}\frac{f'(x_0)-f'(x_0+\varepsilon)}{2\varepsilon} \quad \text{(by L'Hospital rule)} \\ &= \frac{f''(x_0)}{2!}. \end{aligned}$$

Similarly, it can be shown that $f[x_0, x_0, x_0, x_0] = \dfrac{f'''(x_0)}{3!}$.

In general,

$$f[\overbrace{x_0, x_0, \ldots, x_0}^{(k+1)\ \text{times}}] = \frac{f^k(x_0)}{k!}. \tag{3.66}$$

In other words,

$$\frac{d^k}{dx^k}f(x_0) = k!\ f[\overbrace{x_0, x_0, \ldots, x_0}^{(k+1)\ \text{times}}]. \tag{3.67}$$

7. *The nth order divided difference of a polynomial of degree n is constant*

Let $f(x) = a_0x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-1}x + a_n$, $(a_0 \neq 0)$ be a polynomial of degree n.

Then

$$\begin{aligned}
f[x,x_0] &= \frac{f(x)-f(x_0)}{x-x_0}\\
&= a_0\frac{x^n-x_0^n}{x-x_0}+a_1\frac{x^{n-1}-x_0^{n-1}}{x-x_0}+a_2\frac{x^{n-2}-x_0^{n-2}}{x-x_0}+\cdots+a_{n-1}\frac{x-x_0}{x-x_0}\\
&= a_0[x^{n-1}+x^{n-2}x_0+x^{n-3}x_0^2+\cdots+xx_0^{n-2}+x_0^{n-1}]\\
&\quad +a_1[x^{n-2}+x^{n-3}x_0+x^{n-4}x_0^2+\cdots+xx_0^{n-3}+x_0^{n-2}]+\cdots+a_{n-1}\\
&= a_0x^{n-1}+(a_0x_0+a_1)x^{n-2}+(a_0x_0^2+a_1x_0+a_2)x^{n-3}+\cdots+a_{n-1}\\
&= b_0x^{n-1}+b_1x^{n-2}+b_2x^{n-3}+\cdots+b_{n-1},
\end{aligned}$$

where $b_0=a_0, b_1=a_0x_0+a_1, b_2=a_0x_0^2+a_1x_0+a_2,\ldots,b_{n-1}=a_{n-1}$.

Thus, first order divided difference of a polynomial of degree n is a polynomial of degree $n-1$.

Again, the second order divided difference is

$$\begin{aligned}
f[x,x_0,x_1] &= \frac{f[x_0,x]-f[x_0,x_1]}{x-x_1}\\
&= b_0\frac{x^{n-1}-x_1^{n-1}}{x-x_1}+b_1\frac{x^{n-2}-x_1^{n-2}}{x-x_1}+b_2\frac{x^{n-3}-x_1^{n-3}}{x-x_1}\\
&\quad +\cdots+b_{n-2}\frac{x-x_1}{x-x_1}\\
&= b_0[x^{n-2}+x^{n-2}x_1+x^{n-4}x_1^2+\cdots+xx_1^{n-2}]\\
&\quad +b_1[x^{n-3}+x^{n-4}x_1+x^{n-5}x_1^2\\
&\quad +\cdots+xx_1^{n-4}+x_1^{n-3}]+\cdots+b_{n-2}\\
&= b_0x^{n-2}+(b_0x_1+b_1)x^{n-3}+(b_0x_1^2+b_1x_1+b_2)x^{n-4}\\
&\quad +\cdots+b_{n-2}\\
&= c_0x^{n-2}+c_1x^{n-3}+c_2x^{n-4}+\cdots+c_{n-2},
\end{aligned}$$

where $c_0=b_0, c_1=b_0x_1+b_1, c_2=b_0x_1^2+b_1x_1+b_2,\ldots,c_{n-2}=b_{n-2}$.

This is a polynomial of degree $n-2$. So, the second order divided difference is a polynomial of degree $n-2$.

In this way, it can be shown that the n order divided difference of a polynomial of degree n is constant and which is equal to a_0.

3.13 Newton's Fundamental Interpolation Formula

Suppose the function $y=f(x)$ is known at the points $x_0,x_1,\ldots,x_n$ and $y_i=f(x_i)$, $i=0,1,\ldots,n$. The points x_i, $i=0,1,\ldots,n$ need not be equispaced.

Now, from the definition of divided difference of second order

$$f[x_0, x] = \frac{f(x) - f(x_0)}{x - x_0}.$$

Then $f(x) = f(x_0) + (x - x_0)f[x_0, x]$.

From third order divided difference,

$$\begin{aligned} f[x_0, x_1, x] &= \frac{f[x_0, x_1] - f[x_0, x]}{x_1 - x} \\ \textit{i.e.,}\ \ f[x_0, x] &= f[x_0, x_1] + (x - x_1)f[x_0, x_1, x] \\ \textit{i.e.,}\quad \frac{f(x) - f(x_0)}{x - x_0} &= f[x_0, x_1] + (x - x_1)f[x_0, x_1, x] \\ \text{Thus,}\ \ f(x) &= f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x]. \end{aligned}$$

Similarly, for the arguments x_0, x_1, x_2, x,

$$\begin{aligned} f(x) &= f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] \\ &+ (x - x_0)(x - x_1)(x - x_2)f[x_0, x_1, x_2, x_3] + \cdots \\ &+ (x - x_0)(x - x_1) \cdots (x - x_n)f[x_0, x_1, x_2, \ldots, x_n, x]. \end{aligned} \tag{3.68}$$

This formula is known as **Newton's fundamental** or **Newton's general** interpolation formula including error term.

Tabular form of divided differences are shown in Table 3.4.

Table 3.4: Divided difference table.

		x	$f(x)$	First	Second	Third
		x_0	$f(x_0)$			
	$x_0 - x_1$			$f[x_0, x_1]$		
$x_0 - x_2$		x_1	$f(x_1)$		$f[x_0, x_1, x_2]$	
	$x_1 - x_2$			$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
$x_1 - x_3$		x_2	$f(x_2)$		$f[x_1, x_2, x_3]$	
	$x_2 - x_3$			$f[x_2, x_3]$		
		x_3	$f(x_3)$			

Error term

The error term

$$\begin{aligned} E(x) &= (x-x_0)(x-x_1)\cdots(x-x_n)f[x_0,x_1,\ldots,x_n,x] \\ &= (x-x_0)(x-x_1)\cdots(x-x_n)\frac{f^{n+1}(\xi)}{(n+1)!}, \text{[using (3.66)]} \end{aligned} \tag{3.69}$$

where $\min\{x_0,x_1,\ldots,x_n,x\} < \xi < \max\{x_0,x_1,\ldots,x_n,x\}$.

Example 3.13.1 Find the value of y when $x = 1.5$ from the following table:

x :	1	5	7	10	12
y :	0.6931	1.7918	2.0794	2.3979	2.5649

using Newton's divided difference formula.

Solution. The divided difference table is

				x	y	1st	2nd	3rd	4th
				1	0.6931				
			–4			0.2747			
		–6		5	1.7918		–0.0218		
	–9		–2			0.1438		0.0016	
–11		–5		7	2.0794		–0.0075		–0.0001
	–7		–3			0.1062		0.0004	
		–5		10	2.3979		–0.0045		
			–2			0.0835			
				12	2.5649				

Here $x = 1.5, x_0 = 1, x_1 = 5, x_2 = 7, x_3 = 10, x_4 = 12$,
$f[x_0] = 0.6931, f[x_0,x_1] = 0.2747, f[x_0,x_1,x_2] = -0.0218$,
$f[x_0,x_1,x_2,x_3] = 0.0016, f[x_0,x_1,x_2,x_3,x_4] = -0.0001$.
The Newton's divided difference formula is

$$\begin{aligned} y(1.5) &= f[x_0] + (x-x_0)f[x_0,x_1] + (x-x_0)(x-x_1)f[x_0,x_1,x_2] \\ &\quad +(x-x_0)(x-x_1)(x-x_2)f[x_0,x_1,x_2,x_3] \\ &\quad +(x-x_0)(x-x_1)(x-x_2)(x-x_3)f[x_0,x_1,x_2,x_3,x_4] \\ &= 0.6931 + 0.1374 + 0.0382 + 0.0154 + 0.0082 \\ &= 0.8922. \end{aligned}$$

3.14 Deductions of other Interpolation Formulae from Newton's Divided Difference Formula

3.14.1 Newton's forward difference interpolation formula

If the arguments $x_0, x_1, \ldots, x_n$ are equispaced, then $x_i = x_0 + ih, i = 0, 1, \ldots, n$, and

$$f[x_0, x_1, \ldots, x_n] = \frac{\Delta^n f(x_0)}{n!\, h^n}. \text{ [see (3.65)]}$$

In this particular case, the Newton's divided difference formula becomes

$$\begin{aligned}\phi(x) &= f(x_0) + (x - x_0)\frac{\Delta f(x_0)}{1!h} + (x - x_0)(x - x_1)\frac{\Delta^2 f(x_0)}{2!h^2} \\ &+ (x - x_0)(x - x_1)(x - x_2)\frac{\Delta^3 f(x_0)}{3!h^3} + \cdots \\ &+ (x - x_0)(x - x_1)\cdots(x - x_{n-1})\frac{\Delta^n f(x_0)}{n!h^n} + E(x),\end{aligned}$$

where

$$\begin{aligned}E(x) &= (x - x_0)(x - x_1)\cdots(x - x_n)f[x, x_0, x_1, \ldots, x_n] \\ &= (x - x_0)(x - x_1)\cdots(x - x_n)\frac{\Delta^{n+1} f(\xi)}{(n+1)!\, h^{n+1}}.\end{aligned}$$

Now, a unit less quantity $u = \dfrac{x - x_0}{h}$, i.e., $x = x_0 + uh$ is introduced to simplify the formula.

So, $x - x_i = (u - i)h$.

Then

$$\begin{aligned}\phi(x) &= f(x_0) + u\Delta f(x_0) + \frac{u(u-1)}{2!}\Delta^2 f(x_0) + \cdots \\ &+ \frac{u(u-1)(u-2)\cdots(u-n+1)}{n!}\Delta^n f(x_0) + E(x), \\ E(x) &= u(u-1)(u-2)\cdots(u-n)\frac{f^{n+1}(\xi)}{(n+1)!}, \\ &\min\{x, x_0, x_1, \ldots, x_n\} < \xi < \max\{x, x_0, x_1, \ldots, x_n\}.\end{aligned}$$

3.14.2 Newton's backward difference interpolation formula

Let

$$\begin{aligned}\phi(x) &= f(x_n) + (x - x_n)f[x_n, x_{n-1}] + (x - x_n)(x - x_{n-1})f[x_n, x_{n-1}, x_{n-2}] \\ &+ \cdots + (x - x_n)(x - x_{n-1})\cdots(x - x_1)f[x_n, x_{n-1}, \ldots, x_1, x_0] \\ &+ E(x),\end{aligned}$$

where

$$E(x) \;=\; (x-x_n)(x-x_{n-1})\cdots(x-x_1)(x-x_0)f[x,x_n,x_{n-1},\ldots,x_1,x_0].$$

From the relation (3.65), we have

$$f[x_n,x_{n-1},\ldots,x_{n-k}] = \frac{\Delta^k f(x_{n-k})}{k!h^k} = \frac{\nabla^k f(x_n)}{k!h^k}.$$

Therefore,

$$\begin{aligned}\phi(x) \;&=\; f(x_n)+(x-x_n)\frac{\nabla f(x_n)}{1!h}+(x-x_n)(x-x_{n-1})\frac{\nabla^2 f(x_n)}{2!h^2}+\cdots\\ &+\;(x-x_n)(x-x_{n-1})\cdots(x-x_1)(x-x_0)\frac{\nabla^n f(x_n)}{n!h^n}+E(x),\end{aligned}$$

where

$$E(x) \;=\; (x-x_n)(x-x_{n-1})\cdots(x-x_1)(x-x_0)\frac{\nabla^{n+1} f(\xi)}{(n+1)!h^{n+1}},$$

where $\min\{x,x_0,x_1,\ldots,x_n\} < \xi < \max\{x,x_0,x_1,\ldots,x_n\}$.

3.14.3 Lagrange's interpolation formula

From the definition of $(n+1)$ order divided difference (3.64) we have

$$f[x_0,x_1,\ldots,x_n] = \sum_{i=0}^{n}\frac{f(x_i)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}.$$

For $(n+2)$ arguments $x,x_0,\ldots,x_n$,

$$\begin{aligned}f[x,x_0,x_1,\ldots,x_n] \;&=\; \frac{f(x)}{(x-x_0)(x-x_1)\cdots(x-x_n)}\\ &+\;\sum_{i=0}^{n}\frac{f(x_i)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}\\ &=\;\frac{f(x)}{w(x)}+\sum_{i=0}^{n}\frac{f(x_i)}{(x_i-x)w'(x_i)},\end{aligned}$$

where $w(x) = (x-x_0)(x-x_1)\cdots(x-x_n)$.

Therefore,

$$\begin{aligned}f(x) \;&=\; \sum_{i=0}^{n}\frac{w(x)f(x_i)}{(x-x_i)w'(x_i)}+w(x)f[x,x_0,x_1,\ldots,x_n]\\ &=\;\sum_{i=0}^{n}L_i(x)f(x_i)+w(x)\frac{f^{n+1}(\xi)}{(n+1)!}\;\;\text{[using (3.65)]}\end{aligned}$$

where $\min\{x_0, x_1, \ldots, x_n, x\} < \xi < \max\{x_0, x_1, \ldots, x_n, x\}$ and

$L_i(x) = \dfrac{w(x)}{(x - x_i)w'(x_i)}$.

This is the Lagrange's interpolation formula with error term.

Note 3.14.1 It is observed that the divided differences for equispaced arguments produce the Newton forward and backward difference formulae. Also, this interpolation gives Lagrange's interpolation formula.

Actually, Lagrange's interpolation formula and Newton's divided difference interpolation formula are equivalent. This fact is proved in the following.

3.15 Equivalence of Lagrange's and Newton's divided difference formulae

In the following, it is proved that the Lagrange's interpolation formula and Newton's divided difference interpolation formula are equivalent.

The Lagrange's interpolation polynomial for the points (x_i, y_i), $i = 0, 1, \ldots, n$ of degree n is

$$\phi(x) = \sum_{i=0}^{n} L_i(x) y_i, \tag{3.70}$$

where

$$L_i(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)}. \tag{3.71}$$

The Newton's interpolation formula with divided difference is given by

$$\begin{aligned}
\phi(x) &= f(x_0) + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + \cdots \\
&\quad +(x - x_0)(x - x_1)\cdots(x - x_{n-1})f[x_0, x_1, \cdots, x_n] \\
&= f(x_0) + (x - x_0)\left[\frac{f(x_0)}{x_0 - x_1} + \frac{f(x_1)}{x_1 - x_0}\right] \\
&\quad +(x - x_0)(x - x_1) \times \\
&\quad \left[\frac{f(x_0)}{(x_0 - x_1)(x_0 - x_2)} + \frac{f(x_1)}{(x_1 - x_0)(x_1 - x_2)} + \frac{f(x_2)}{(x_2 - x_0)(x_2 - x_1)}\right] \\
&\quad + \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1}) \times \\
&\quad \left[\frac{f(x_0)}{(x_0 - x_1)\cdots(x_0 - x_n)} + \cdots + \frac{f(x_n)}{(x_n - x_0)\cdots(x_n - x_{n-1})}\right].
\end{aligned} \tag{3.72}$$

The coefficient of $f(x_0)$ in the above expression is

$$\begin{aligned}
& 1+\frac{x-x_0}{x_0-x_1}+\frac{(x-x_0)(x-x_1)}{(x_0-x_1)(x_0-x_2)}+\cdots \\
&\qquad +\frac{(x-x_0)(x-x_1)\cdots(x-x_{n-1})}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_n)} \\
=\ & \frac{(x-x_1)}{(x_0-x_1)}\left[1+\frac{x-x_0}{x_0-x_2}+\frac{(x-x_0)(x-x_2)}{(x_0-x_2)(x_0-x_3)}+\right. \\
&\qquad \left.+\cdots+\frac{(x-x_0)(x-x_2)\cdots(x-x_{n-1})}{(x_0-x_2)(x_0-x_3)\cdots(x_0-x_n)}\right] \\
=\ & \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}\left[1+\frac{x-x_0}{x_0-x_2}+\cdots+\frac{(x-x_0)(x-x_3)\cdots(x-x_{n-1})}{(x_0-x_3)(x_0-x_4)\cdots(x_0-x_n)}\right] \\
=\ & \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_{n-1})}\left[1+\frac{x-x_0}{x_0-x_n}\right] \\
=\ & \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})(x-x_n)}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_{n-1})(x_0-x_n)} \\
=\ & L_0(x).
\end{aligned}$$

Similarly, it can be shown that the coefficient of $f(x_1)$ is $L_1(x)$, coefficient of $f(x_2)$ is $L_2(x)$ and so on.

Thus, (3.72) becomes

$$\phi(x)=L_0(x)f(x_0)+L_1(x)f(x_1)+\cdots+L_n(x)f(x_n)=\sum_{i=1}^{n}L_i(x)f(x_i).$$

Thus the Lagrange's interpolation and Newton's divided difference interpolation formulae are equivalent.

Example 3.15.1 A function $y=f(x)$ is given at the points $x=x_0,x_1,x_2$. Show that the Newton's divided difference interpolation formula and the corresponding Lagrange's interpolation formula are identical.

Solution. The Newton's divided difference formulae is given as

$$\begin{aligned}
y &= f(x)=y_0+(x-x_0)f[x_0,x_1]+(x-x_0)(x-x_1)f[x_0,x_1,x_2] \\
&= y_0+(x-x_0)\frac{f(x_1)-f(x_0)}{x_1-x_0}+(x-x_0)(x-x_1) \\
&\quad\times\left[\frac{f(x_0)}{(x_0-x_1)(x_0-x_2)}+\frac{f(x_1)}{(x_1-x_0)(x_1-x_2)}+\frac{f(x_2)}{(x_2-x_0)(x_2-x_1)}\right]
\end{aligned}$$

$$= \left[1 - \frac{(x_0 - x)}{(x_0 - x_1)} + \frac{(x - x_0)(x - x_1)}{(x_0 - x_1)(x_0 - x_2)}\right] f(x_0)$$
$$+\left[\frac{(x - x_0)}{(x_1 - x_0)} + \frac{(x - x_0)(x - x_1)}{(x_1 - x_0)(x_1 - x_2)}\right] f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2)$$
$$= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1)$$
$$+ \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2)$$

which is the Lagrange's interpolation polynomial. Hence Newton's divided difference interpolation polynomial and Lagrange's interpolation polynomial are equivalent.

Example 3.15.2 For the following table, find the interpolation polynomial using (i) Lagrange's formula and (ii) Newton's divided difference formula, and hence show that both represent same interpolating polynomial.

x	:	0	2	4	8
$f(x)$	:	3	8	11	18

Solution. (i) The Lagrange's interpolation polynomial is

$$\begin{aligned}
\phi(x) &= \frac{(x-2)(x-4)(x-8)}{(0-2)(0-4)(0-8)} \times 3 + \frac{(x-0)(x-4)(x-8)}{(2-0)(2-4)(2-8)} \times 8 \\
&\quad + \frac{(x-0)(x-2)(x-8)}{(4-0)(4-2)(4-8)} \times 11 + \frac{(x-0)(x-2)(x-4)}{(8-0)(8-2)(8-4)} \times 19 \\
&= \frac{x^3 - 14x^2 + 56x - 64}{-64} \times 3 + \frac{x^3 - 12x^2 + 32x}{24} \times 8 \\
&\quad + \frac{x^3 - 10x^2 + 16x}{-32} \times 11 + \frac{x^3 - 6x^2 + 8x}{192} \times 19 \\
&= \frac{1}{24}x^3 - \frac{1}{2}x^2 + \frac{10}{3}x + 3.
\end{aligned}$$

(ii) The divided difference table is

x	$f(x)$	1st divided difference	2nd divided difference	3rd divided difference
0	3			
2	8	5/2		
4	11	3/2	–1/4	
8	19	2	1/12	1/24

Newton's divided difference polynomial is

$$\begin{aligned}\phi(x) &= 3+(x-0)\times\frac{5}{2}+(x-0)(x-2)\times(-\frac{1}{4})\\ &\quad +(x-0)(x-2)(x-4)\times\frac{1}{24}\\ &= 3+\frac{5}{2}x-\frac{1}{4}(x^2-2x)+\frac{1}{24}(x^3-6x^2+8x)\\ &= \frac{1}{24}x^3-\frac{1}{2}x^2+\frac{10}{3}x+3.\end{aligned}$$

Thus, it is observed that the interpolating polynomial by both Lagrange's and Newton's divided difference formulae are one and same.

It may be noted that Newton's formula involves less number of arithmetic operations than that of Lagrange's formula.

Algorithm 3.4 (Divided difference). This algorithm finds the value of y at a given point x from a table $(x_i, y_i), i = 0, 1, 2, \dots, n$, by Newton's divided difference interpolation formula.

Let

$$\begin{aligned} y_k &= f(x_k) = d_{k,0}.\\ f[x_k, x_{k-1}] &= d_{k,1} = \frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}} = \frac{d_{k,0}-d_{k-1,0}}{x_k-x_{k-1}}\\ f[x_k,x_{k-1},x_{k-2}] &= d_{k,2} = \frac{f[x_k,x_{k-1}]-f[x_{k-1},x_{k-2}]}{x_k-x_{k-2}} = \frac{d_{k,1}-d_{k-1,1}}{x_k-x_{k-2}}\\ f[x_k,x_{k-1},x_{k-2},x_{k-3}] &= d_{k,3} = \frac{d_{k,2}-d_{k-1,2}}{x_k-x_{k-3}}\end{aligned}$$

In general,

$$d_{k,i} = \frac{d_{k,i-1}-d_{k-1,i-1}}{x_k-x_{k-i}}, \quad i = 1, 2, \dots, n. \tag{3.73}$$

Using the above notations, the Newton's divided difference formula (3.68) can be written in the following form.

$$\begin{aligned} f(x) &= d_{0,0}+d_{1,1}(x-x_0)+d_{2,2}(x-x_0)(x-x_1)\\ &\quad +d_{3,3}(x-x_0)(x-x_1)(x-x_2)+\cdots\\ &\quad +d_{n,n}(x-x_0)(x-x_1)\cdots(x-x_{n-1})\end{aligned}$$

$$\begin{aligned}
&= d_{n,n}(x-x_{n-1})(x-x_{n-2})(x-x_{n-3})\cdots(x-x_0)\\
&\quad +d_{n-1,n-1}(x-x_{n-2})(x-x_{n-3})\cdots(x-x_0)\\
&\quad +d_{n-2,n-2}(x-x_{n-3})\cdots(x-x_0)+d_{1,1}(x-x_0)+d_{0,0}\\
&= (\cdots(((d_{n,n}(x-x_{n-1})+d_{n-1,n-1})(x-x_{n-2})\\
&\quad +d_{n-2,n-2})(x-x_{n-3})+d_{n-3,n-3})(x-x_{n-4})+\cdots)+d_{0,0}
\end{aligned}$$

```
Algorithm Divided_difference
 Step 1: Read $n$; //the degree of the polynomial//
         Read $(x(i), y(i)), i = 0, 1, \ldots, n$ //the points//
 Step 2: for $k = 0$ to $n$ do
             Set $d(k, 0) = y(k)$;
 Step 3: for $i = 0$ to $n$ do      //compute all divided differences//
             for $k = i$ to $n$ do
                 $d(k, i) = (d(k, i-1) - d(k-1, i-1))/(x(k) - x(k-i))$;
 Step 4: Read $xg$;     //for which the polynomial is to be calculated//
 Step 5: Set $sum = d(n, n)$      //initialization of sum//
 Step 6: for $k = n-1$ to 0 do
             Compute $sum = sum * (xg - x(k)) + d(k, k)$;
 Step 7: Print $xg$, $sum$;
end Divided_difference
```

Program 3.4

```
/* Program Divided Difference Interpolation
   This program finds the value of y=f(x) at a given x when
   the function is supplied as (x[i],y[i]), i=0, 1, ..., n.
*/
#include<stdio.h>
#include<math.h>
void main()
{
 int i,k,n;
 float x[20],y[20],xg,sum,d[20][20];
 printf("Enter number of subintervals ");
 scanf("%d",&n);
 printf("Enter x and y values ");
 for(i=0;i<=n;i++)
     scanf("%f %f",&x[i],&y[i]);
 printf("Enter interpolating point x ");
 scanf("%f",&xg);
 printf("The given values of x and y are\nx-value  y-value\n");
 for(i=0;i<=n;i++) printf("%f   %f\n",x[i],y[i]);
 for(k=0;k<=n;k++) d[k][0]=y[k];
 for(i=1;i<=n;i++)
     for(k=i;k<=n;k++)
         d[k][i]=(d[k][i-1]-d[k-1][i-1])/(x[k]-x[k-i]);
 sum=d[n][n];
 for(k=n-1;k>=0;k--) sum=sum*(xg-x[k])+d[k][k];
 printf("The interpolated value at x=%f is %f ",xg,sum);
}
```

A sample of input/output:

```
Enter number of subintervals 4
Enter x and y values
0.10   1.1052
0.15   1.1618
0.20   1.2214
0.25   1.2840
0.30   1.3499
Enter interpolating point x 0.12
The given values of x and y are
x-value    y-value
0.100000   1.105200
0.150000   1.161800
0.200000   1.221400
0.250000   1.284000
0.300000   1.349900
The interpolated value at x=0.120000 is 1.127468
```

3.16 Inverse Interpolation

In interpolation, for a given set of values of x and y, the value of y is determined for a given value of x. But the inverse interpolation is the process which finds the value of x for a given y. Commonly used inverse interpolation formulae are based on successive iteration.

In the following, three inverse interpolation formulae based on Lagrange, Newton forward and Newton backward interpolation formulae are described. The inverse interpolation based on Lagrange's formula is a direct method while the formulae based on Newton's interpolation formulae are iterative.

3.16.1 Inverse interpolation based on Lagrange's formula

The Lagrange's interpolation formula of y on x is

$$y = \sum_{i=0}^{n} \frac{w(x)\, y_i}{(x - x_i)w'(x_i)}.$$

When x and y are interchanged then the above relation changes to

$$x = \sum_{i=0}^{n} \frac{w(y)x_i}{(y - y_i)w'(y_i)} = \sum_{i=0}^{n} L_i(y)x_i,$$

where

$$L_i(y) = \frac{w(y)}{(y-y_i)w'(y_i)} = \frac{(y-y_0)(y-y_1)\cdots(y-y_{i-1})(y-y_{i+1})\cdots(y-y_n)}{(y_i-y_0)(y_i-y_1)\cdots(y_i-y_{i-1})(y_i-y_{i+1})\cdots(y_i-y_n)}.$$

This formula gives the value of x for given value of y and the formula is known as **Lagrange's inverse interpolation** formula.

3.16.2 Method of successive approximations

Based on Newton's forward difference interpolation formula

The Newton's forward difference interpolation formula is

$$\begin{aligned} y &= y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 + \cdots \\ &\quad + \frac{u(u-1)(u-2)\cdots(u-\overline{n-1})}{n!}\Delta^n y_0, \end{aligned}$$

where $u = \dfrac{x - x_0}{h}$.

The above formula can be written as

$$\begin{aligned} u &= \frac{1}{\Delta y_0}\Big[y - y_0 - \frac{u(u-1)}{2!}\Delta^2 y_0 - \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 - \cdots \\ &\quad - \frac{u(u-1)(u-2)\cdots(u-\overline{n-1})}{n!}\Delta^n y_0\Big]. \end{aligned}$$

Let the first approximation of u be denoted by $u^{(1)}$ and it is obtained by neglecting the second and higher differences as

$$u^{(1)} = \frac{1}{\Delta y_0}(y - y_0).$$

Next, the second approximation, $u^{(2)}$, is obtained by neglecting third and higher order differences as follows:

$$u^{(2)} = \frac{1}{\Delta y_0}\Big[y - y_0 - \frac{u^{(1)}(u^{(1)}-1)}{2!}\Delta^2 y_0\Big].$$

Similarly, the third approximation $u^{(3)}$ is given by

$$u^{(3)} = \frac{1}{\Delta y_0}\Big[y - y_0 - \frac{u^{(2)}(u^{(2)}-1)}{2!}\Delta^2 y_0 - \frac{u^{(2)}(u^{(2)}-1)(u^{(2)}-2)}{3!}\Delta^3 y_0\Big].$$

In general,

$$\begin{aligned} u^{(k+1)} &= \frac{1}{\Delta y_0}\Bigg[y - y_0 - \frac{u^{(k)}(u^{(k)}-1)}{2!}\Delta^2 y_0 - \frac{u^{(k)}(u^{(k)}-1)(u^{(k)}-2)}{3!}\Delta^3 y_0 \\ &\quad - \cdots - \frac{u^{(k)}(u^{(k)}-1)\cdots(u^{(k)}-k)}{(k+1)!}\Delta^{k+1} y_0\Bigg], \end{aligned}$$

$k = 0, 1, 2, \ldots$.

This process of approximation should be continued till two successive approximations $u^{(k+1)}$ and $u^{(k)}$ be equal up to desired number of decimal places. Then the value of x is obtained from the relation $x = x_0 + u^{(k+1)}h$.

Example 3.16.1 From the table of values

x	:	1.8	2.0	2.2	2.4	2.6
y	:	3.9422	4.6269	5.4571	6.4662	7.6947

find x when $y = 5.0$ using the method of successive approximations.

Solution. The difference table is

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
1.8	3.9422			
		0.6847		
2.0	**4.6269**		0.1455	
		0.8302		0.0334
2.2	5.4571		**0.1789**	
		1.0091		**0.0405**
2.4	6.4662		0.2194	
		1.2285		
2.6	7.6947			

Let $x_0 = 2.0, h = 0.2$. The value of u is determined by successive approximation. The first approximation is

$$\begin{aligned}
u^{(1)} &= \frac{1}{\Delta y_0}(y - y_0) = \frac{1}{0.8302}(5.0 - 4.6269) = 0.4494.\\
u^{(2)} &= \frac{1}{\Delta y_0}\left[y - y_0 - \frac{u^{(1)}(u^{(1)} - 1)}{2!}\Delta^2 y_0\right] = u^{(1)} - \frac{u^{(1)}(u^{(1)} - 1)}{2!}\frac{\Delta^2 y_0}{\Delta y_0}\\
&= 0.4494 - \frac{0.4494(0.4494 - 1)}{2}\frac{0.1789}{0.8302} = 0.4761.\\
u^{(3)} &= u^{(1)} - \frac{u^{(2)}(u^{(2)} - 1)}{2}\frac{\Delta^2 y_0}{\Delta y_0} - \frac{u^{(2)}(u^{(2)} - 1)(u^{(2)} - 2)}{3!}\frac{\Delta^3 y_0}{\Delta y_0}\\
&= 0.4494 - \frac{0.4761(0.4761 - 1)}{2}\cdot\frac{0.1789}{0.8302}\\
&\quad - \frac{0.4761(0.4761 - 1)(0.4761 - 2)}{6}\cdot\frac{0.0405}{0.8302}\\
&= 0.4494 + 0.0269 - 0.0031 = 0.4732.
\end{aligned}$$

Thus the value of x is given by $x = x_0 + u^{(3)}h = 2.0 + 0.4732 \times 0.2 = 2.0946$.

3.16.3 Based on Newton's backward difference interpolation formula

The Newton's backward formula is

$$\begin{aligned}
y &= y_n + v\nabla y_n + \frac{v(v+1)}{2!}\nabla^2 y_n + \frac{v(v+1)(v+2)}{3!}\nabla^3 y_n + \cdots\\
&\quad + \frac{v(v+1)(v+2)\cdots(v+n-1)}{n!}\nabla^n y_n,
\end{aligned}$$

where $v = \dfrac{x - x_n}{h}$ or $x = x_n + vh$.

That is,

$$\begin{aligned}
v &= \frac{1}{\nabla y_n}\Big[y - y_n - \frac{v(v+1)}{2!}\nabla^2 y_n - \frac{v(v+1)(v+2)}{3!}\nabla^3 y_n - \cdots\\
&\quad - \frac{v(v+1)\cdots(v+n-1)}{n!}\nabla^n y_n\Big].
\end{aligned}$$

Neglecting second and higher order differences, the first approximation is given by

$$v^{(1)} = \frac{1}{\nabla y_n}(y - y_n).$$

Similarly,

$$v^{(2)} = \frac{1}{\nabla y_n}\left[y - y_n - \frac{v^{(1)}(v^{(1)} + 1)}{2!}\nabla^2 y_n\right].$$

$$v^{(3)} = \frac{1}{\nabla y_n}\left[y - y_n - \frac{v^{(2)}(v^{(2)}+1)}{2!}\nabla^2 y_n - \frac{v^{(2)}(v^{(2)}+1)(v^{(2)}+2)}{3!}\nabla^3 y_n\right]$$

and so on.

In general,

$$\begin{aligned}v^{(k+1)} &= \frac{1}{\nabla y_n}\left[y - y_n - \frac{v^{(k)}(v^{(k)}+1)}{2!}\nabla^2 y_n - \frac{v^{(k)}(v^{(k)}+1)(v^{(k)}+2)}{3!}\nabla^3 y_n\right. \\ &\quad \left. - \cdots - \frac{v^{(k)}(v^{(k)}+1)\cdots(v^{(k)}+k)}{(k+1)!}\nabla^{k+1} y_n\right],\end{aligned}$$

$k = 0, 1, 2, \ldots$.

This iteration continues until two consecutive values $v^{(k)}$ and $v^{(k+1)}$ become equal up to a desired number of significant figures.

The value of x is given by $x = x_n + v^{(k+1)}h$.

3.16.4 Use of inverse interpolation to find a root of an equation

Suppose $x = \alpha$ be a root of the equation $f(x) = 0$ and let it lies between a and b, i.e., $a < \alpha < b$. Now, a table is constructed for some values of x, within (a, b), and the corresponding values of y. Then by inverse interpolation, the value of x is determined when $y = 0$. This value of x is the required root.

Example 3.16.2 Find a real root of the equation $x^3 - 3x + 1 = 0$.

Solution. Let $y = x^3 - 3x + 1$. One root of this equation lies between 1/4 and 1/2. Let us consider the points $x = 0.25, 0.30, 0.35, 0.40, 0.45, 0.50$. The table is shown below.

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
0.25	0.265625			
0.30	0.127000	–0.138625		
0.35	–0.007125	–0.134125	0.00450	
0.40	–0.136000	–0.128875	0.00525	0.00075
0.45	–0.258875	–0.122875	0.00600	0.00075
0.50	–0.375000	–0.116125	0.00675	0.00075

Here the interpolating point is $y = 0$.

Now,

$$
\begin{aligned}
u^{(1)} &= -\frac{y_0}{\Delta y_0} = \frac{0.265625}{0.138625} = 1.916140.\\
u^{(2)} &= -\frac{1}{\Delta y_0}\left[y_0 + \frac{u^{(1)}(u^{(1)}-1)}{2}\Delta^2 y_0\right]\\
&= \frac{1}{0.138625}\left[0.265625 + \frac{1.916140 \times 0.916140}{2} \times 0.00450\right]\\
&= 1.944633.\\
u^{(3)} &= -\frac{1}{\Delta y_0}\left[y_0 + \frac{u^{(2)}(u^{(2)}-1)}{2!}\Delta^2 y_0 + \frac{u^{(2)}(u^{(2)}-1)(u^{(2)}-2)}{3!}\Delta^3 y_0\right]\\
&= \frac{1}{0.138625}\left[0.265625 + \frac{1.944633 \times 0.944633}{2} \times 0.00450\right.\\
&\quad \left. + \frac{1.944633 \times 0.944633 \times (-0.055367)}{6} \times 0.000750\right]\\
&= 1.945864.
\end{aligned}
$$

Thus $x = x_0 + u^{(3)} \times h = 0.25 + 1.945864 \times 0.05 = 0.347293$ which is a required root.

3.17 Choice and use of Interpolation Formulae

If the interpolating points are not equally spaced then Lagrange's, Newton's divided difference or Aitken's iterated interpolation formulae may be used. Newton's forward formula is appropriate for interpolation at the beginning of the table, Newton's backward formula for interpolation at the end of the table, Stirling's or Bessel's formula for interpolation at the centre of the table. It is well known that the interpolation polynomial is unique and the above formulae are just different forms of one and the same interpolation polynomial and the results obtained by the different formulae should be identical. Practically, only a subset of the set of given interpolating points in the table is used. For interpolation at the beginning of the table, it is better to take this subset from the beginning of the table. This reason recommends the use of Newton's forward formula for interpolation at the beginning of the table. For interpolation, near the end of the table, interpolating points should be available at the end of the table and hence Newton's backward formula is used for interpolation at the end of the table. For the same reasons the central difference formulae like Stirling's, Bessel's, Everett's etc. are used for interpolation near the centre of the table. The proper choice of a central interpolation formulae depends on the error terms of the different formulae.

For interpolation near the centre of the table, Stirling's formula gives the most accurate result for $-1/4 \leq s \leq 1/4$, and Bessel's formula gives most accurate result near

$s = 1/2$, i.e., for $1/4 \leq s \leq 3/4$. If all the terms of the formulae are considered, then both the formulae give identical result. But, if some terms are discarded to evaluate the polynomial, then Stirling's and Bessel's formulae, in general, do not give the same result and then a choice must be made between them. The choice depends on the order of the highest difference that could be neglected so that contributions from it and further differences would be less than half a unit in the last decimal place. If the highest difference is of odd order, then Stirling's formula is used and if it is of even order, then, generally, Bessel's formula is used. This conclusion is drawn from the following comparison.

The term of Stirling's formula containing the third differences is

$$\frac{s(s^2-1^2)}{6}\frac{\Delta^3 y_{-1}+\Delta^3 y_{-2}}{2}.$$

This term may be neglected if its magnitude is less than half a unit in the last place, i.e., if

$$\left|\frac{s(s^2-1^2)}{6}\frac{\Delta^3 y_{-1}+\Delta^3 y_{-2}}{2}\right| \leq \frac{1}{2}.$$

The maximum value of $\left|\dfrac{s(s^2-1)}{6}\right|$ is 0.064 at $s = -1/\sqrt{3}$. Then

$$\left|0.064 \times \frac{\Delta^3 y_{-1}+\Delta^3 y_{-2}}{2}\right| < \frac{1}{2}, \text{ i.e., } \left|\frac{\Delta^3 y_{-1}+\Delta^3 y_{-2}}{2}\right| < 7.8.$$

The term containing third order difference of Bessel's formula will be less than half a unit in the last place if

$$\left|\frac{s(s-1)(s-1/2)}{6}\Delta^3 y_{-1}\right| < \frac{1}{2}.$$

The maximum value of $\left|\dfrac{s(s-1)(s-1/2)}{6}\right|$ is 0.008 and so that $|\Delta^3 y_{-1}| < 62.5$.

Thus, if the third difference is ignored, Bessel's formula gives about eight times more accurate result than Stirling's formula. But, if the third differences need to be retained and when the magnitude is more than 62.5, then Everett's formula is more appropriate. It may be reminded that the Bessel's formula with third differences is equivalent to Everett's formula with second differences.

Depending on these discussions the following working rules are recommended for use of interpolation formulae.

(i) If the interpolating point is at the beginning of the table, then use Newton's forward formula with a suitable starting point x_0 such that $0 < u < 1$.

(ii) If the interpolating point is at the end of the table, then use Newton's backward formula with a suitable starting point x_n such that $-1 < u < 0$.

(iii) If the interpolating point is at the centre of the table and the difference table ends with odd order differences, then use Stirling's formula.

(iv) If the interpolating point is at the centre of the table and the difference table ends with even order differences, then use Bessel's or Everett's formula.

(v) If the arguments are not equispaced then use Lagrange's formula or Newton's divided difference formula or Aitken's iterative formula.

In some functions, a higher degree interpolation polynomial does not always give the best result compared to a lower degree polynomial. This fact is illustrated in the following example.

Let $f(x) = \dfrac{1}{1+x^2}$. In $[-3, 3]$ the second degree polynomial $y = \phi_2(x) = 1-0.1x^2$ and fourth degree polynomial $y = \phi_4(x) = 0.15577u^4 - 1.24616u^3 + 2.8904u^3 - 1.59232u + 0.1$ where $u = (x+3)/1.5$ are derived. The graph of the curves $y = f(x)$, $y = \phi_2(x)$ and $y = \phi_4(x)$ are shown in Figure 3.4.

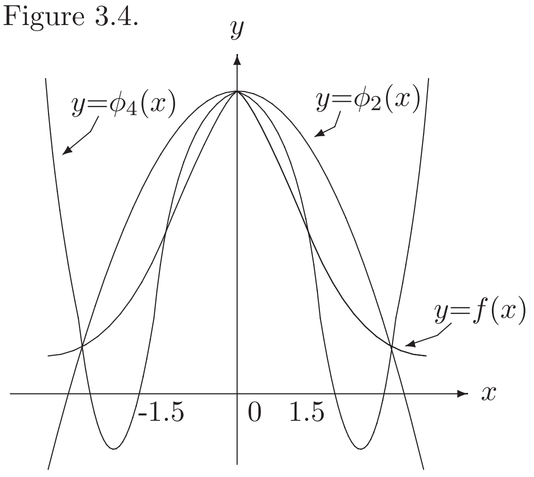


Figure 3.4: The graph of the curves $y = f(x)$, $y = \phi_2(x)$ and $y = \phi_4(x)$.

For this function $y = f(x)$, the fourth degree polynomial gives an absurd result at $x = 2$. At this point $f(2) = 0.2$, $\phi_2(2) = 0.6$ and $\phi_4(2) = -0.01539$. It may be noted that the functions $y = f(x)$ and $y = \phi_2(x)$ are positive for all values of x, but $y = \phi_4(x)$ is negative for some values of x. This example indicates that the higher degree polynomial does always not give more accurate result.

3.18 Hermite's Interpolation Formula

The interpolation formulae considered so far make use of the function values at some number of points, say, $n+1$ number of points and an nth degree polynomial is obtained. But, if the values of the function $y = f(x)$ and its first derivatives are known at $n+1$ points then it is possible to determine an interpolating polynomial $\phi(x)$ of degree $(2n+1)$ which satisfies the $(2n+2)$ conditions

$$\left.\begin{aligned} \phi(x_i) &= f(x_i) \\ \phi'(x_i) &= f'(x_i), i = 0, 1, 2, \ldots, n. \end{aligned}\right\} \tag{3.74}$$

This formula is known as **Hermite's interpolation formula**. Here, the number of conditions is $(2n+2)$, the number of coefficients to be determined is $(2n+2)$ and the degree of the polynomial is $(2n+1)$.

Let us assume the Hermite's interpolating polynomial in the form

$$\phi(x) = \sum_{i=0}^{n} h_i(x)f(x_i) + \sum_{i=0}^{n} H_i(x)f'(x_i), \tag{3.75}$$

where $h_i(x)$ and $H_i(x)$, $i = 0, 1, 2, \ldots, n$, are polynomial in x of degree at most $(2n+1)$.

Using conditions (3.74), we get

$$h_i(x_j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} ; \; H_i(x_j) = 0, \text{ for all } i \tag{3.76}$$

$$h_i'(x_j) = 0, \text{ for all } i; \; H_i'(x_j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \tag{3.77}$$

Let us consider the Lagrangian function

$$\begin{aligned} L_i(x) &= \frac{(x - x_0)(x - x_1)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)}, \\ & i = 0, 1, 2, \ldots, n. \end{aligned} \tag{3.78}$$

Obviously,

$$L_i(x_j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j. \end{cases} \tag{3.79}$$

Since each $L_i(x)$ is a polynomial of degree n, $[L_i(x)]^2$ is a polynomial of degree $2n$. Again, each $L_i(x)$ satisfies (3.79) and $[L_i(x)]^2$ also satisfies (3.79). Since $h_i(x)$ and $H_i(x)$ are polynomials in x of degree $(2n+1)$, their explicit form may be taken as

$$\begin{aligned} h_i(x) &= (a_i x + b_i)[L_i(x)]^2 \\ H_i(x) &= (c_i x + d_i)[L_i(x)]^2 \end{aligned} \tag{3.80}$$

Using the conditions (3.77), we obtain

$$\begin{aligned} a_i x_i + b_i &= 1 \\ c_i x_i + d_i &= 0 \\ a_i + 2L_i'(x_i) &= 0 \\ c_i &= 1 \end{aligned} \tag{3.81}$$

These relations give

$$\begin{aligned} a_i &= -2L_i'(x_i), & b_i &= 1 + 2x_i L_i'(x_i) \\ c_i &= 1 \quad \text{and} & d_i &= -x_i \end{aligned} \tag{3.82}$$

Hence,

$$\begin{aligned} h_i(x) &= [-2xL_i'(x_i) + 1 + 2x_i L_i'(x_i)][L_i(x_i)]^2 \\ &= [1 - 2(x - x_i)L_i'(x_i)][L_i(x)]^2 \text{ and} \\ H_i(x) &= (x - x_i)[L_i(x)]^2. \end{aligned} \tag{3.83}$$

Then finally (3.75) becomes

$$\begin{aligned} \phi(x) &= \sum_{i=0}^{n} [1 - 2(x - x_i)L_i'(x_i)][L_i(x)]^2 f(x_i) \\ &\quad + \sum_{i=0}^{n} (x - x_i)[L_i(x)]^2 f'(x_i), \end{aligned} \tag{3.84}$$

which is the required **Hermite's interpolation** formula.

Example 3.18.1 Determine the Hermite's polynomial of degree 5 which satisfies the following data and hence find an approximate value of $\sqrt[3]{2.8}$.

x	:	1.5	2.0	2.5
$y = \sqrt[3]{x}$	:	1.14471	1.25992	1.35721
$y' = 1/(3x^{2/3})$	:	0.25438	0.20999	0.18096

Solution.

$$\begin{aligned} L_0(x) &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x - 2.0)(x - 2.5)}{(1.5 - 2.0)(1.5 - 2.5)} = 2x^2 - 9x + 10, \\ L_1(x) &= \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(x - 1.5)(x - 2.5)}{(2.0 - 1.5)(2.0 - 2.5)} = -4x^2 + 16x - 15, \\ L_2(x) &= \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{(x - 1.5)(x - 2.0)}{(2.5 - 1.5)(2.5 - 2.0)} = 2x^2 - 7x + 6. \end{aligned}$$

Therefore
$L_0'(x) = 4x - 9, L_1'(x) = -8x + 16, L_2'(x) = 4x - 7.$
Hence $L_0'(x_0) = -3, L_1'(x_1) = 0, L_2'(x_2) = 3.$

$$\begin{aligned}
h_0(x) &= [1 - 2(x - x_0)L_0'(x_0)][L_0(x)]^2 = [1 - 2(x - 1.5)(-3)][L_0(x)]^2\\
&= (6x - 8)(2x^2 - 9x + 10)^2,\\
h_1(x) &= [1 - 2(x - x_1)L_1'(x_1)][L_1(x)]^2 = (4x^2 - 16x + 15)^2,\\
h_2(x) &= [1 - 2(x - x_2)L_2'(x_2)][L_2(x)]^2 = [1 - 2(x - 2.5)(3)](2x^2 - 7x + 6)^2\\
&= (16 - 6x)(2x^2 - 7x + 6)^2.\\
H_0(x) &= (x - x_0)[L_0(x)]^2 = (x - 1.5)(2x^2 - 9x + 10)^2,\\
H_1(x) &= (x - x_1)[L_1(x)]^2 = (x - 2.0)(4x^2 - 16x + 15)^2,\\
H_2(x) &= (x - x_2)[L_2(x)]^2 = (x - 2.5)(2x^2 - 7x + 6)^2.
\end{aligned}$$

The required Hermite polynomial is

$$\begin{aligned}
\phi(x) &= (6x - 8)(2x^2 - 9x + 10)^2(1.14471) + (4x^2 - 16x + 15)^2(1.25992)\\
&\quad +(16 - 6x)(2x^2 - 7x + 6)^2(1.35721)\\
&\quad +(x - 1.5)(2x^2 - 9x + 10)^2(0.25438)\\
&\quad +(x - 2)(4x^2 - 16x + 15)^2(0.20999)\\
&\quad +(x - 2.5)(2x^2 - 7x + 6)^2(0.18096).
\end{aligned}$$

To find the value of $\sqrt[3]{2.8}$, substituting $x = 2.8$ to the above relation.
Therefore,

$$\begin{aligned}
\sqrt[3]{2.8} &= 10.07345 \times 0.23040 + 1.25992 \times 2.43360 - 1.08577 \times 4.32640\\
&\quad +0.33069 \times 0.23040 + 0.16799 \times 2.43360 + 0.05429 \times 4.32640\\
&= 1.40948.
\end{aligned}$$

3.19 Spline Interpolation

Spline interpolation is very powerful and widely used method and has many applications in numerical differentiation, integration, solution of boundary value problems, two and three - dimensional graph plotting etc. Spline interpolation method, interpolates a function between a given set of points by means of piecewise smooth polynomials. In this interpolation, the curve passes through the given set of points and also its slope and its curvature are continuous at each point. The splines with different degree are found in literature, among them cubic splines are widely used.

3.19.1 Cubic spline

A function $y(x)$ is called **cubic spline** in $[x_0, x_n]$ if there exist cubic polynomials $p_0(x), p_1(x), \ldots, p_{n-1}(x)$ such that

$$\begin{aligned} y(x) &= p_i(x) \text{ on } [x_i, x_{i+1}], i = 0, 1, 2, \ldots, n-1. && (3.85)\\ p'_{i-1}(x_i) &= p'_i(x_i), i = 1, 2, \ldots, n-1 \text{ (equal slope)}. && (3.86)\\ p''_{i-1}(x_i) &= p''_i(x_i), i = 1, 2, \ldots, n-1 \text{ (equal curvature)}. && (3.87)\\ \text{and } p_i(x_i) &= y_i, \qquad p_i(x_{i+1}) = y_{i+1}, i = 0, 1, \ldots, n-1. && (3.88) \end{aligned}$$

It may be noted that, at the endpoints x_0 and x_n, no continuity on slope and curvature are assigned. The conditions at these points are assigned, generally, depending on the applications.

Let the interval $[x_i, x_{i+1}]$, $i = 0, 1, \ldots, n-1$ be denoted by ith interval.

Let $h_i = x_i - x_{i-1}, i = 1, 2, \ldots, n$ and $M_i = y''(x_i), i = 0, 1, 2, \ldots, n$.

Let the cubic spline for the ith interval be

$$y(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad \text{in } [x_i, x_{i+1}]. \tag{3.89}$$

Since it passes through the end points x_i and x_{i+1}, therefore,

$$y_i = d_i \tag{3.90}$$

and

$$\begin{aligned} y_{i+1} &= a_i(x_{i+1} - x_i)^3 + b_i(x_{i+1} - x_i)^2 + c_i(x_{i+1} - x_i) + d_i \\ &= a_i h_{i+1}^3 + b_i h_{i+1}^2 + c_i h_{i+1} + d_i. && (3.91) \end{aligned}$$

Equation (3.89) is differentiated twice and obtained the following equations.

$$\begin{aligned} y'(x) &= 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i. && (3.92)\\ \text{and } y''(x) &= 6a_i(x - x_i) + 2b_i. && (3.93) \end{aligned}$$

From (3.93), $y''_i = 2b_i$ and $y''_{i+1} = 6a_i h_{i+1} + 2b_i$,
that is, $M_i = 2b_i$, $M_{i+1} = 6a_i h_{i+1} + 2b_i$.

Therefore,

$$\begin{aligned} b_i &= \frac{M_i}{2}, && (3.94)\\ a_i &= \frac{M_{i+1} - M_i}{6h_{i+1}}. && (3.95) \end{aligned}$$

Using (3.90), (3.94) and (3.95), equation (3.91) becomes

$$\begin{aligned} y_{i+1} &= \frac{M_{i+1}-M_i}{6h_{i+1}}h_{i+1}^3 + \frac{M_i}{2}h_{i+1}^2 + c_i h_{i+1} + y_i \\ i.e., \;\; c_i &= \frac{y_{i+1}-y_i}{h_{i+1}} - \frac{2h_{i+1}M_i + h_{i+1}M_{i+1}}{6}. \end{aligned} \tag{3.96}$$

Thus, the coefficients a_i, b_i, c_i and d_i of (3.89) are determined in terms of $n+1$ unknowns $M_0, M_1, \ldots, M_n$. These unknowns are determined in the following way.

The condition of equation (3.86) state that the slopes of the two cubics p_{i+1} and p_i are same at x_i.

Now, for the ith interval

$$y_i' = 3a_i(x_i - x_i)^2 + 2b_i(x_i - x_i) + c_i = c_i, \tag{3.97}$$

and for the $(i-1)$th interval

$$y_i' = 3a_{i-1}(x_i - x_{i-1})^2 + 2b_{i-1}(x_i - x_{i-1}) + c_{i-1}. \tag{3.98}$$

Equating (3.97) and (3.98), we obtain

$$c_i = 3a_{i-1}h_i^2 + 2b_{i-1}h_i + c_{i-1}.$$

The values of $a_{i-1}, b_{i-1}, c_{i-1}$ and c_i are substituted to the above equation and obtained the following equation.

$$\begin{aligned} \frac{y_{i+1}-y_i}{h_{i+1}} &- \frac{2h_{i+1}M_i + h_{i+1}M_{i+1}}{6} \\ &= 3\Big(\frac{M_i - M_{i-1}}{6h_i}\Big)h_i^2 + \Big(\frac{M_i}{2}\Big)h_i + \frac{y_i - y_{i-1}}{h_i} - \frac{2h_iM_{i-1} + h_iM_i}{6}. \end{aligned}$$

After simplification the above equation reduces to

$$h_iM_{i-1} + 2(h_i + h_{i+1})M_i + h_{i+1}M_{i+1} = 6\Big(\frac{y_{i+1}-y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\Big). \tag{3.99}$$

This relation is true for $i = 1, 2, \ldots, n-1$. Thus $n-1$ equations are available for the $n+1$ unknown quantities $M_0, M_1, \ldots, M_n$. Now, two more conditions are required to solve these equations uniquely. These conditions can be assumed to take one of the following forms:

(i) $M_0 = M_n = 0$. If this conditions are satisfied then the corresponding spline is called **natural spline**.

(ii) $M_0 = M_n, M_1 = M_{n+1}, y_0 = y_n, y_1 = y_{n+1}, h_1 = h_{n+1}$. The corresponding spline satisfying these conditions is called **periodic spline**.

(iii) $y'(x_0) = y'_0, y'(x_n) = y'_n$, i.e.,

$$\begin{aligned} 2M_0 + M_1 &= \frac{6}{h_1}\Big(\frac{y_1 - y_0}{h_1} - y'_0\Big) \\ \text{and}\quad M_{n-1} + 2M_n &= \frac{6}{h_n}\Big(y'_n - \frac{y_n - y_{n-1}}{h_n}\Big). \end{aligned} \tag{3.100}$$

The corresponding spline satisfying the above conditions is called **non-periodic spline** or **clamped cubic spline**.

(iv) If $M_0 = M_1 - \dfrac{h_1(M_2 - M_1)}{h_2}$ and $M_n = M_{n-1} + \dfrac{h_n(M_{n-1} - M_{n-2})}{h_{n-1}}$. The corresponding spline is called **extrapolated spline**.

(v) If $M_0 = y''_0$ and $M_n = y''_n$ are specified. If a spline satisfy these conditions then it is called **endpoint curvature-adjusted spline**.

Let $A_i = h_i, B_i = 2(h_i + h_{i+1}), C_i = h_{i+1}$ and $D_i = 6\Big(\dfrac{y_{i+1} - y_i}{h_{i+1}} - \dfrac{y_i - y_{i-1}}{h_i}\Big)$.

Then (3.99) becomes

$$A_iM_{i-1} + B_iM_i + C_iM_{i+1} = D_i, i = 1, 2, \ldots, n-1. \tag{3.101}$$

The system of equations (3.101) is basically a tri-diagonal system. Depending on different conditions, the tri-diagonal systems are different and they are stated below.

For natural spline, the tri-diagonal system for $M_1, M_2, \ldots, M_{n-1}$ is

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_3 & B_3 & C_3 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & A_{n-1} & B_{n-1} \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_{n-1} \end{bmatrix} \tag{3.102}$$

and $M_0 = M_n = 0$.

Imposing the conditions for non-periodic spline, we find

$$2M_0 + M_1 = D_0 \tag{3.103}$$

$$\text{and } M_{n-1} + 2M_n = D_n, \tag{3.104}$$

$$\begin{aligned} \text{where } D_0 &= \frac{6}{h_1}\left(\frac{y_1 - y_0}{h_1} - y'_0\right) \\ \text{and } D_n &= \frac{6}{h_n}\left(y'_n - \frac{y_n - y_{n-1}}{h_n}\right). \end{aligned} \tag{3.105}$$

Then equations (3.101), (3.103), (3.104) and (3.105) result the following tri-diagonal system for the unknowns $M_0, M_1, \ldots, M_n$.

$$\begin{bmatrix} 2 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ A_1 & B_1 & C_1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_2 & B_2 & C_2 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & A_{n-1} & B_{n-1} & C_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ \vdots \\ D_{n-1} \\ D_n \end{bmatrix}. \qquad (3.106)$$

For the extrapolated spline the values of M_0 and M_n are given by the relations

$$M_0 = M_1 - \frac{h_1(M_2 - M_1)}{h_2} \text{ and } M_n = M_{n-1} + \frac{h_n(M_{n-1} - M_{n-2})}{h_{n-1}}. \qquad (3.107)$$

The first expression is rewritten as

$$M_1\left[A_1 + B_1 + \frac{A_1h_1}{h_2}\right] + M_2\left[C_1 - \frac{A_1h_1}{h_2}\right] = D_1 \text{ or, } M_1B_1' + M_2C_1' = D_1$$

where $B_1' = A_1 + B_1 + \dfrac{A_1h_1}{h_2}$ and $C_1' = C_1 - \dfrac{A_1h_1}{h_2}$.

Similarly, the second expression is transferred to $M_{n-2}A_{n-1}' + M_{n-1}B_{n-1}' = D_{n-1}$ where

$$A_{n-1}' = A_{n-1} - \frac{C_{n-1}h_n}{h_{n-1}} \text{ and } B_{n-1}' = B_{n-1} + C_{n-1} + \frac{h_nC_{n-1}}{h_{n-1}}.$$

For this case, the tri-diagonal system of equations for $M_1, M_2, \ldots, M_{n-1}$ is

$$\begin{bmatrix} B_1' & C_1' & 0 & 0 & \cdots & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_3 & B_3 & C_3 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & A_{n-1}' & B_{n-1}' \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_{n-1} \end{bmatrix}. \qquad (3.108)$$

The values of M_0 and M_n are obtained from the equation (3.107).

For the endpoint curvature-adjusted spline, the values of M_0 and M_n are respectively y_0'' and y_n'', where y_0'' and y_n'' are specified. The values of $M_1, M_2, \ldots, M_{n-1}$ are given by solving the following tri-diagonal system of equations

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & A_3 & B_3 & C_3 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & A_{n-1} & B_{n-1} \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} D_1' \\ D_2 \\ \vdots \\ D_{n-2} \\ D_{n-1}' \end{bmatrix}, \qquad (3.109)$$

where $D_1' = D_1 - A_1y_0''$, $\quad D_{n-1}' = D_{n-1} - C_{n-1}y_n''$.

Example 3.19.1 Fit a cubic spline curve that passes through (0, 0.0), (1, 0.5), (2, 2.0) and (3, 1.5) with the natural end boundary conditions, $y''(0) = y''(3) = 0$.

Solution. Here the intervals are (0, 1) (1, 2) and (2, 3), i.e., three intervals of x, in each of which we can construct a cubic spline. These piecewise cubic spline polynomials together gives the cubic spline curve $y(x)$ in the entire interval (0, 3). Here $h_1 = h_2 = h_3 = 1$.
Then equation (3.99) becomes

$$M_{i-1} + 4M_i + M_{i+1} = 6(y_{i+1} - 2y_i + y_{i-1}), \quad i = 1, 2, 3.$$

That is,

$$\begin{aligned} M_0 + 4M_1 + M_2 &= 6(y_2 - 2y_1 + y_0) = 6 \times (2.0 - 2 \times 0.5 + 0.0) = 6 \\ M_1 + 4M_2 + M_3 &= 6(y_3 - 2y_2 + y_1) = 6 \times (1.5 - 2 \times 2.0 + 0.5) = -12. \end{aligned}$$

Imposing the conditions $M_0 = y''(0) = 0$ and $M_3 = y''(3) = 0$ to the above equations, and they simplify as

$$4M_1 + M_2 = 6, \qquad M_1 + 4M_2 = -12.$$

These equations give $M_1 = \frac{12}{5}$ and $M_2 = -\frac{18}{5}$.
Let the natural cubic spline be given by

$$p_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

where the coefficients a_i, b_i, c_i and d_i are given by the relations

$$\begin{aligned} a_i &= \frac{M_{i+1} - M_i}{6h_{i+1}}, \qquad b_i = \frac{M_i}{2}, \\ c_i &= \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{2h_{i+1}M_i + h_{i+1}M_{i+1}}{6}, \qquad d_i = y_i, \end{aligned}$$

for $i = 0, 1, 2$.
Therefore,

$$\begin{aligned} a_0 &= \frac{M_1 - M_0}{6} = 0.4, & b_0 &= \frac{M_0}{2} = 0, \\ c_0 &= \frac{y_1 - y_0}{1} - \frac{2M_0 + M_1}{6} = 0.1, & d_0 &= y_0 = 0. \\ a_1 &= \frac{M_2 - M_1}{6} = -1, & b_1 &= \frac{M_1}{2} = \frac{6}{5}, \end{aligned}$$

$$\begin{aligned} c_1 &= \frac{y_2 - y_1}{1} - \frac{2M_1 + M_2}{6} = 1.3, \qquad d_1 = y_1 = 0.5. \\ a_2 &= \frac{M_3 - M_2}{6} = \frac{3}{5}, \qquad b_2 = \frac{M_2}{2} = -\frac{9}{5}, \\ c_2 &= \frac{y_3 - y_2}{1} - \frac{2M_2 + M_3}{6} = 0.7, \qquad d_2 = y_2 = 2.0. \end{aligned}$$

Hence the required piecewise cubic splines in each interval are given by

$$\begin{aligned} p_0(x) &= 0.4x^3 + 0.1x, \qquad 0 \le x \le 1 \\ p_1(x) &= -(x-1)^3 + 1.2(x-1)^2 + 1.3(x-1) + 0.5, \qquad 1 \le x \le 2 \\ p_2(x) &= 0.6(x-2)^3 - 1.8(x-2)^2 + 0.7(x-2) + 2.0, \qquad 2 \le x \le 3. \end{aligned}$$

Example 3.19.2 Fit a cubic spline curve for the following data with end conditions $y'(0) = 0.2$ and $y'(3) = -1$.

x	:	0	1	2	3
y	:	0	0.5	3.5	5

Solution. Here, the three intervals (0, 1) (1, 2) and (2, 3) are given in each of which the cubic splines are to be constructed. These cubic spline functions are denoted by y_0, y_1 and y_2. In this example, $h_1 = h_2 = h_3 = 1$.
For the boundary conditions, equation (3.99) is used. That is,

$$\begin{aligned} M_0 + 4M_1 + M_2 &= 6(y_2 - 2y_1 + y_0) \\ M_1 + 4M_2 + M_3 &= 6(y_3 - 2y_2 + y_1). \end{aligned}$$

Also, from the equations (3.100)

$$2M_0 + M_1 = 6(y_1 - y_0 - y_0') \text{ and } M_2 + 2M_3 = 6(y_3' - y_3 + y_2)$$

i.e.,

$$\begin{aligned} M_0 + 4M_1 + M_2 &= 15 \\ M_1 + 4M_2 + M_3 &= -9 \\ 2M_0 + M_1 &= 1.8 \\ M_2 + 2M_3 &= 6(-1 - 5 + 3.5) = -15. \end{aligned}$$

These equations give $M_0 = -1.36, M_1 = 4.52, M_2 = -1.72$ and $M_3 = -6.64$.
Let the cubic spline in each interval be given by

$$y_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i.$$

The coefficients are computed as

$$\begin{aligned} a_i &= \frac{M_{i+1}-M_i}{6h_{i+1}}, \qquad b_i = \frac{M_i}{2}, \\ c_i &= \frac{y_{i+1}-y_i}{h_{i+1}} - \frac{2h_{i+1}M_i + h_{i+1}M_{i+1}}{6}, \qquad d_i = y_i, \text{ for } i = 0,1,2. \end{aligned}$$

Therefore,

$$\begin{aligned} a_0 &= \frac{M_1-M_0}{6} = 0.98, \ \ b_0 = -0.68, \ \ c_0 = 0.2, \ \ d_0 = 0. \\ a_1 &= \frac{M_2-M_1}{6} = -1.04, \ \ b_1 = 2.26, \ \ c_1 = 1.78, \ \ d_1 = 0.5. \\ a_2 &= \frac{M_3-M_2}{6} = -0.82, \ \ b_2 = -0.86, \ \ c_2 = 3.18, \ \ d_2 = 3.5. \end{aligned}$$

Hence, the required piecewise cubic spline polynomials in each interval are given by

$$\begin{aligned} y_0(x) &= 0.98x^3 - 0.68x^2 + 0.2x, \qquad 0 \le x \le 1 \\ y_1(x) &= -1.04(x-1)^3 + 2.26(x-1)^2 + 1.78(x-1) + 0.5, \qquad 1 \le x \le 2 \\ y_2(x) &= -0.82(x-2)^3 - 0.86(x-2)^2 + 3.18(x-2) + 3.5, \qquad 2 \le x \le 3. \end{aligned}$$

Example 3.19.3 Consider the function

$$f(x) = \begin{cases} -\frac{11}{2}x^3 + 26x^2 - \frac{75}{2}x + 18, & 1 \le x \le 2, \\ \frac{11}{2}x^3 - 40x^2 + \frac{189}{2}x - 70, & 2 \le x \le 3. \end{cases}$$

Show that $f(x)$ is a cubic spline.

Solution. Let

$$\begin{aligned} p_0(x) &= -\frac{11}{2}x^3 + 26x^2 - \frac{75}{2}x + 18, \qquad 1 \le x \le 2, \\ \text{and } p_1(x) &= \frac{11}{2}x^3 - 40x^2 + \frac{189}{2}x - 70, \qquad 2 \le x \le 3. \end{aligned}$$

Here $x_0 = 1, x_1 = 2$ and $x_2 = 3$. The function $f(x)$ will be a cubic spline if

$$\begin{aligned} &(a) \quad p_i(x_i) = f(x_i), \qquad p_i(x_{i+1}) = f(x_{i+1}), \qquad i = 0,1 \text{ and} \\ &(b) \quad p'_{i-1}(x_i) = p'_i(x_i), \qquad p''_{i-1}(x_i) = p''_i(x_i), \qquad i = 1. \end{aligned}$$

But, here the values of $f(x_0), f(x_1)$ and $f(x_2)$ are not supplied, so only the conditions of (b) are to be checked.
Now,

$$\begin{aligned} p_0'(x) &= -\frac{33}{2}x^2 + 52x - \frac{75}{2}, & p_1'(x) &= \frac{33}{2}x^2 - 80x + \frac{189}{2} \\ p_0''(x) &= -33x + 52, & p_1''(x) &= 33x - 80. \end{aligned}$$

$p_0'(x_1) = p_0'(2) = 0.5$, $p_1'(x_1) = p_1'(2) = 0.5$, i.e., $p_0'(x_1) = p_1'(x_1)$.

$p_0''(x_1) = p_0''(2) = -14$ and $p_1''(x_1) = p_1''(2) = -14$. Thus $p_0''(x_1) = p_1''(x_1)$.

Hence $f(x)$ is a spline.

Algorithm 3.5 (Cubic spline). This algorithm finds the cubic spline for each of the intervals $[x_i, x_{i+1}], i = 0, 1, \ldots, n-1$. The (i) Natural spline, (ii) Non-periodic spline or clamped cubic spline, (iii) Extrapolated spline, and (iv) endpoint curvature-adjusted spline are incorporated here. The spacing for x_i need not be equal and assume that $x_0 < x_1 < \cdots < x_n$.

```
**Algorithm Cubic_Spline**
Read $x_i, y_i, i = 0, 1, 2, \ldots, n$;//$x$'s are not necessary equispaced//
Read $xg$; //the value of $x$ at which $y$ is to be computed.//
//Computation of $h_i$.//
for $i = 1$ to $n$ do
    $h_i = x_i - x_{i-1}$;
endfor;
//Compututation of $A_i, B_i, C_i$ and $D_i$//
for $i = 1$ to $n-1$ do
    $A_i = h_i$;    $B_i = 2(h_i + h_{i+1})$;    $C_i = h_{i+1}$;
    $D_i = 6\left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\right)$;
endfor;
Case :
(i) *Natural spline*
    To find $M_1, M_2, \ldots, M_{n-1}$, solve the system of tri-diagonal equation
    defined in (3.102).
    Set $M_0 = M_n = 0$.
(ii) *Non-periodic spline*
    Read $y_0', y_n'$; //first derivative of $y$ at $x = x_0, x_n$//
    Compute $D_0 = \frac{6}{h_1}\left(\frac{y_1 - y_0}{h_1} - y_0'\right)$, $D_n = \frac{6}{h_n}\left(y_n' - \frac{y_n - y_{n-1}}{h_n}\right)$.
    To find $M_0, M_1, M_2, \ldots, M_n$, solve the system of tri-diagonal equation
    defined in (3.106).
```

```
(iii) Extrapolated spline
      Compute $B_1' = A_1 + B_1 + \frac{A_1 h_1}{h_2}, C_1' = C_1 - \frac{A_1 h_1}{h_2}$,
      $A_{n-1}' = A_{n-1} - \frac{C_{n-1} h_n}{h_{n-1}}, B_{n-1}' = B_{n-1} + C_{n-1} + \frac{h_n C_{n-1}}{h_{n-1}}$
      To find $M_1, M_2, \ldots, M_{n-1}$, solve the system of tri-diagonal equation
      defined in (3.108).
      Compute $M_0 = M_1 - \frac{h_1(M_2 - M_1)}{h_2}, M_n = M_{n-1} + \frac{h_n(M_{n-1} - M_{n-2})}{h_{n-1}}$.
(iv) Endpoint curvature-adjusted spline
      Read $y_0'', y_n''$; //double derivative of $y$ at $x = x_0, x_n$//
      Compute $D_1' = D_0 - A_1 y_0'', D_{n-1}' = D_{n-1} - C_{n-1} y_n''$
      To find $M_1, M_2, \ldots, M_{n-1}$, solve the system of tri-diagonal equation
      defined in (3.109).
      Set $M_0 = y_0'', M_n = y_n''$.
endcase;

//Compututation of the coefficients $a_i, b_i, c_i, d_i, i = 0, 1, 2, \ldots, n$.//
for $i = 1$ to $n$ do      $a_i = \frac{M_{i+1} - M_i}{6h_{i+1}}$,
      $b_i = \frac{M_i}{2}$
      $c_i = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{2h_{i+1} M_i + h_{i+1} M_{i+1}}{6}$,      $d_i = y_i$;
endfor;
//Printing of splines//
for $i = 0$ to $n - 1$ do
      Print 'Coefficient of ', $i$, 'th spline is' $a_i, b_i, c_i, d_i$;
endfor;
//Computation of $y$ at $x = xg$//
if $(xg < x_0)$ or $(xg > x_n)$ then
      Print '$x$ outside the range';
      Stop;
endif;
for $i = 0$ to $n - 1$ do
      if $(xg < x_{i+1})$ then
            $j = i$;
            exit from for loop;
      endif;
endfor;
Compute $y_c = a_j(xg - x_j)^3 + b_j(xg - x_j)^2 + c_j(xg - x_j) + d_j$;
Print 'The value of $y$ at $x =$', $xg$, 'is', $y_c$;
end Cubic_Spline
```

Program 3.5

```
/* Program Cubic Spline
   This program construct cubic splines at each interval
   [x[i-1],x[i]], i=1, 2, ..., n and finds the value of
   y=f(x) at a given x when the function is supplied as
   (x[i],y[i]), i=0, 1, ..., n. */
#include<stdio.h>
#include<math.h>
#include<ctype.h>
#include<stdlib.h>
float M[21];
void main()
{
 int i,n;
 char opt,s[5];
 float x[20],y[20],h[20],A[20],B[20],C[20],D[20];
 float a[20],b[20],c[20],d[20],xg,yd0,ydn,temp,yc;
 float TriDiag(float a[],float b[],float c[],float d[],int n);
 printf("\nEnter number of subintervals ");
 scanf("%d",&n);
 printf("Enter x and y values ");
 for(i=0;i<=n;i++) scanf("%f %f",&x[i],&y[i]);
 printf("Enter interpolating point x ");
 scanf("%f",&xg);
 printf("The given values of x and y are\nx-value   y-value\n");
 for(i=0;i<=n;i++) printf("%f   %f\n",x[i],y[i]);

 for(i=0;i<=n;i++) h[i]=x[i]-x[i-1]; /* computation of h[i] */
 for(i=1;i<n;i++) /* computation of A,B,C,D's */
     {
         A[i]=h[i];
         B[i]=2*(h[i]+h[i+1]);
         C[i]=h[i+1];
         D[i]=6*((y[i+1]-y[i])/h[i+1]-(y[i]-y[i-1])/h[i]);
     }
 printf("\nN  Natural spline\n");
 printf("P  Non-Periodic spline\n");
 printf("E  Extrapolated spline\n");
 printf("C  End point Curvature adjusted spline\n");
 printf("   Enter your choice ");
 opt=getche();

 switch(toupper(opt))
    {
     case 'N': /* Natural spline */
               temp=TriDiag(A,B,C,D,n-1);
               M[0]=0; M[n]=0;
               break;
     case 'P': /* Non-periodic spline */
               printf("\nEnter the values of y'[0] and y'[n] ");
               scanf("%f %f",&yd0,&ydn);
               D[0]=6*((y[1]-y[0])/h[1]-yd0)/h[1];
               D[n]=6*(ydn-(y[n]-y[n-1])/h[n])/h[n];
               for(i=n+1;i>=1;i--) D[i]=D[i-1];
               A[n+1]=1; B[n+1]=2;
               for(i=n;i>=2;i--){
                  A[i]=A[i-1]; B[i]=B[i-1]; C[i]=C[i-1];}
               B[1]=2; C[1]=1;
               temp=TriDiag(A,B,C,D,n+1);
               for(i=0;i<=n;i++) M[i]=M[i+1];
               break;
    case 'E': /* Extrapolated spline */
              B[1]=A[1]+B[1]+A[1]*h[1]/h[2];
              C[1]=C[1]-A[1]*h[1]/h[2];
              A[n-1]=A[n-1]-C[n-1]*h[n]/h[n-1];
              B[n-1]=B[n-1]+C[n-1]+C[n-1]*h[n]/h[n-1];
              temp=TriDiag(A,B,C,D,n-1);
              M[0]=M[1]-h[1]*(M[2]-M[1])/h[2];
              M[n]=M[n-1]+h[n]*(M[n-1]-M[n-2])/h[n-1];
              break;
    case 'C': /* End point Curvature adjusted spline */
              printf("\nEnter the values of y''[0] and y''[n] ");
              scanf("%f %f",&yd0,&ydn);
              D[1]=D[1]-A[1]*yd0;
              D[n-1]=D[n-1]-C[n-1]*ydn;
              temp=TriDiag(A,B,C,D,n-1);
              M[0]=yd0; M[n]=ydn;
              break;
   default : printf("\n No choice \n");
             exit(0);
 } /* switch */
 /* Computation of the coefficients of the splines */
 for(i=0;i<=n-1;i++){
       a[i]=(M[i+1]-M[i])/(6*h[i+1]); b[i]=M[i]/2;
       c[i]=(y[i+1]-y[i])/h[i+1]-(2*h[i+1]*M[i]+h[i+1]*M[i+1])/6;
       d[i]=y[i];
     }
 /* printing of splines */
 printf("\nThe cubic splines are \n");
 for(i=0;i<n;i++)
   {
    s[1]=(x[i]>0) ? '-':'+';
    s[2]=(b[i]<0) ? '-':'+';
    s[3]=(c[i]<0) ? '-':'+';
    s[4]=(d[i]<0) ? '-':'+';
    temp=fabs(x[i]);
    printf("p%1d(x)=%7.4f(x%c%7.4f)^3%c%7.4f(x%c%7.4f)^2%c%7.4f
         (x%c%7.4f)%c%7.4f\n",i,a[i],s[1],temp,s[2],fabs(b[i]),
         s[1],temp,s[3],fabs(c[i]),s[1],temp,s[4],fabs(d[i]));
    printf("      in [%7.4f,%7.4f]\n",x[i],x[i+1]);
  }
 /* computation of y at x=xg */
 if((xg<x[0]) || (xg>x[n])){
      printf("\nx outside the range ");
      exit(0);
    }
 for(i=0;i<=n-1;i++) /* determination of y */
     {
       if(xg<x[i+1])
         {
           temp=xg-x[i];
           yc=a[i]*temp*temp*temp+b[i]*temp*temp+c[i]*temp+d[i];
           printf("The value of y at x=%f is %f ",xg,yc);
           exit(0);
         }
     }
} /* main */
```

```
float TriDiag(float a[20],float b[20],float c[20],float d[20],int n)
 {
 /* output M[i], i=1, 2,..., n, is a global variable.*/
 int i;
 float gamma[10],z[10];
 gamma[1]=b[1];
 for(i=2;i<=n;i++)
    {
     if(gamma[i-1]==0.0)
        {
          printf("A minor is zero: Method fails ");
          exit(0);
        }
     gamma[i]=b[i]-a[i]*c[i-1]/gamma[i-1];
    }
 z[1]=d[1]/gamma[1];
 for(i=2;i<=n;i++)
     z[i]=(d[i]-a[i]*z[i-1])/gamma[i];
 /* Computation of M */
 M[n]=z[n];
 for(i=n-1;i>=1;i--)
     M[i]=z[i]-c[i]*M[i+1]/gamma[i];
  return(M[0]);
} /*end of TriDiag */
```

A sample of input/output:

```
Enter number of subintervals 3
Enter x and y values
-1   1.0
 1   0.5
 2   3.5
 3   5.0
Enter interpolating point x 1.2
The given values of x and y are
x-value    y-value
-1.000000   1.000000
1.000000   0.500000
2.000000   3.500000
3.000000   5.000000
```

```
N  Natural spline
P  Non-Periodic spline
E  Extrapolated spline
C  End point Curvature adjusted spline
   Enter your choice n

The cubic splines are
p0(x)= 0.3152(x+ 1.0000)^3+ 0.0000(x+ 1.0000)^2- 1.5109(x+ 1.0000)
       + 1.0000 in [-1.0000, 1.0000]
p1(x)=-1.1630(x- 1.0000)^3+ 1.8913(x- 1.0000)^2+ 2.2717(x- 1.0000)
       + 0.5000 in [ 1.0000, 2.0000]
p2(x)= 0.5326(x- 2.0000)^3- 1.5978(x- 2.0000)^2+ 2.5652(x- 2.0000)
       + 3.5000 in [ 2.0000, 3.0000]
The value of y at x=1.200000 is 1.020696
```

Another input/output:

```
Enter number of subintervals 3
Enter x and y values
-1   1.0
 1   0.5
 2   3.5
 3   5.0
Enter interpolating point x 1.2
The given values of x and y are
x-value    y-value
-1.000000   1.000000
1.000000   0.500000
2.000000   3.500000
3.000000   5.000000

N  Natural spline
P  Non-Periodic spline
E  Extrapolated spline
C  End point Curvature adjusted spline
   Enter your choice p
Enter the values of y'[0] and y'[n] 0 1
The cubic splines are
p0(x)= 0.6250(x+ 1.0000)^3- 1.3750(x+ 1.0000)^2+ 0.0000(x+ 1.0000)
       + 1.0000 in [-1.0000, 1.0000]
```

```
p1(x)=-1.3750(x- 1.0000)^3+ 2.3750(x- 1.0000)^2+ 2.0000(x- 1.0000)
       + 0.5000 in [ 1.0000, 2.0000]
p2(x)= 0.6250(x- 2.0000)^3- 1.7500(x- 2.0000)^2+ 2.6250(x- 2.0000)
       + 3.5000 in [ 2.0000, 3.0000]
The value of y at x=1.200000 is 0.984000
```

3.20 Bivariate Interpolation

Like single valued interpolation, recently bivariate interpolations become important due to their extensive uses in a wide range of fields e.g., digital image processing, digital filter design, computer-aided design, solution of non-linear simultaneous equations etc.

In this section, some of the important methods are described to construct interpolation formulae that can be efficiently evaluated.

To construct the formulae, the following two approaches are followed.

(i) Constructing a function that matches exactly the functional values at all the data points.

(ii) Constructing a function that approximately fits the data. This approach is desirable when the data likely to have errors and require smooth functions.

On the basis of these approaches, one can use four types of methods (i) local matching methods, (ii) local approximation methods, (iii) global matching methods and (iv) global approximation methods. In the local methods, the constructed function at any point depends only on the data at relatively nearby points. In global methods, the constructed function at any points depends on all or most of the data points.

In the matching method, the matching function matches exactly the given values, but in the approximate method the function approximately fits the data.

Here, the local and the global matching methods are discussed only.

3.20.1 Local matching methods

Here two local matching methods are described, viz., triangular interpolation and rectangular grid or bilinear interpolation.

Triangular interpolation

The simplest local interpolating surface is of the form

$$F(x, y) = a + bx + cy.$$

The data at the three corners of a triangle determine the coefficients. This procedure generates a piecewise linear surface which is global continuous.

Suppose the function $f(x,y)$ be known at the points (x_1,y_1), (x_2,y_2) and (x_3,y_3). Let $f_1=f(x_1,y_1)$, $f_2=f(x_2,y_2)$ and $f_3=f(x_3,y_3)$.

Let the constructed function be

$$F(x,y)=a+bx+cy \tag{3.110}$$

such that $F(x_i,y_i)=f(x_i,y_i), i=1,2,3$.

Then

$$\begin{aligned} f_1 &= a+bx_1+cy_1 \\ f_2 &= a+bx_2+cy_2 \\ f_3 &= a+bx_3+cy_3. \end{aligned}$$

These equations give the values of a,b and c as

$$\begin{aligned} a &= \frac{f_1(x_2y_3-x_3y_2)-f_2(x_1y_3-x_3y_1)+f_3(x_1y_2-x_2y_1)}{\Delta} \\ b &= \frac{(f_2-f_1)(y_3-y_1)-(f_3-f_1)(y_2-y_1)}{\Delta} \\ c &= \frac{(f_3-f_1)(x_2-x_1)-(f_2-f_1)(x_3-x_1)}{\Delta} \end{aligned}$$

where

$$\Delta=(x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1).$$

The values of a,b,c give the required polynomial.

But, the function $F(x,y)$ can be written in the following form

$$F(x,y) = Af_1+Bf_2+Cf_3, \tag{3.111}$$

$$\text{where } A = \frac{(x_2-x)(y_3-y)-(x_3-x)(y_2-y)}{\Delta} \tag{3.112}$$

$$B = \frac{(x_3-x)(y_1-y)-(x_1-x)(y_3-y)}{\Delta} \tag{3.113}$$

$$C = \frac{(x_1-x)(y_2-y)-(x_2-x)(y_1-y)}{\Delta}. \tag{3.114}$$

Note 3.20.1 (i) If $A+B+C=1$ then $\Delta\neq 0$.

(ii) If $\Delta=0$ then the points (x_i,y_i), $i=1,2,3$ are collinear.

(iii) Let (x_i,y_i) and $f(x_i,y_i)$, $i=1,2,\dots,n$ be given. If we choose non-overlapping triangles which cover the region containing all these points, then a function that is continuous in this region can be determined.

Example 3.20.1 For a function $f(x,y)$, let $f(1,1)=8, f(2,1)=12$ and $f(2,2)=20$. Find the approximate value of $f(3/2,5/4)$ using triangular interpolation.

Solution. Here given that

$$\begin{aligned}
x_1 &= 1, & y_1 &= 1, & f_1 &= f(x_1,y_1)=8\\
x_2 &= 2, & y_2 &= 1, & f_2 &= f(x_2,y_2)=12\\
x_3 &= 2, & y_3 &= 2, & f_3 &= f(x_3,y_3)=20\\
x &= \frac{3}{2}, & y &= \frac{5}{4}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\Delta &= (x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1)\\
&= (2-1)(2-1)-(2-1)(1-1)=1.\\
A &= \frac{(x_2-x)(y_3-y)-(x_3-x)(y_2-y)}{\Delta}=\frac{1}{2}\\
B &= \frac{(x_3-x)(y_1-y)-(x_1-x)(y_3-y)}{\Delta}=\frac{1}{4}\\
C &= \frac{(x_1-x)(y_2-y)-(x_2-x)(y_1-y)}{\Delta}=\frac{1}{4}.
\end{aligned}$$

Thus $f(3/2,5/4)\simeq F(3/2,5/4)=Af_1+Bf_2+Cf_3=\frac{1}{2}\times 8+\frac{1}{4}\times 12+\frac{1}{4}\times 20=12.$

Bilinear interpolation

Let a function $f(x,y)$ be known at the points (x_1,y_1), (x_1+h,y_1), (x_1,y_1+k) and (x_1+h,y_1+k).

A function $F(x,y)$ is to be constructed within the rectangle formed by these points.

Let $f_1=f(x_1,y_1), f_2=f(x_1+h,y_1), f_3=f(x_1,y_1+k)$ and $f_4=f(x_1+h,y_1+k)$.

Let us construct a function $F(x,y)$ of the form

$$F(x,y)=a+b(x-x_1)+c(y-y_1)+d(x-x_1)(y-y_1) \tag{3.115}$$

such that

$$\begin{aligned}
F(x_1,y_1)=f(x_1,y_1)=f_1, &\quad F(x_1+h,y_1)=f(x_1+h,y_1)=f_2,\\
F(x_1,y_1+k)=f(x_1,y_1+k)=f_3, &\quad F(x_1+h,y_1+k)=f(x_1+h,y_1+k)=f_4.
\end{aligned}$$

The unknowns a,b,c,d can be obtained by solving the following equations
$f_1=a, f_2=a+bh, f_3=a+ck$ and $f_4=a+hb+kc+hkd.$

Thus

$$\begin{aligned} a &= f_1, \qquad b = \frac{f_2 - f_1}{h}, \\ c &= \frac{f_3 - f_1}{k} \quad \text{and } d = \frac{f_4 + f_1 - f_2 - f_3}{hk}. \end{aligned} \tag{3.116}$$

Example 3.20.2 Find a bilinear interpolation polynomial $F(x, y)$ for the function $f(x, y)$ where $f(1, 1) = 8, f(2, 1) = 10, f(1, 2) = 12$ and $f(2, 2) = 20$. Also, find an approximate value of $f(4/3, 5/3)$.

Solution. Here

$x_1 = 1,$	$y_1 = 1,$	$f_1 = f(x_1, y_1) = 8$
$x_1 + h = 2,$	$y_1 = 1,$	$f_2 = f(x_1 + h, y_1) = 10$
$x_1 = 1,$	$y_1 + k = 2,$	$f_3 = f(x_1, y_1 + k) = 12$
$x_1 + h = 2,$	$y_1 + k = 2,$	$f_4 = f(x_1 + h, y_1 + k) = 20.$

Obviously, $h = 1, k = 1$.

Thus,

$$a = f_1 = 8, \quad b = \frac{f_2 - f_1}{h} = \frac{10 - 8}{1} = 2,$$

$$c = \frac{f_3 - f_1}{k} = \frac{12 - 8}{1} = 4, \quad d = \frac{f_4 + f_1 - f_2 - f_3}{hk} = 6.$$

Hence,

$$\begin{aligned} f(x, y) &\simeq F(x, y) = a + b(x - x_1) + c(y - y_1) + d(x - x_1)(y - y_1) \\ &= 8 + 2(x - 1) + 4(y - 1) + 6(x - 1)(y - 1). \end{aligned}$$

Therefore, $\quad f(4/3, 5/3) = \dfrac{38}{3}.$

3.20.2 Global methods

Variables-separation method (bilinear form)

Let the interpolating points are evenly distributed over a rectangular grid. Let the polynomial be of the form

$$F(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x^i y^j. \tag{3.117}$$

This function can be written as

$$F(x, y) = [1 \;\; y \;\; y^2 \;\; \cdots \;\; y^{n-1}] \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \vdots \\ x^{n-1} \end{bmatrix} \tag{3.118}$$

That is, $F(x,y) = \mathbf{Y}^t(y)\mathbf{A}\mathbf{X}(x)$ (say),
where $\mathbf{Y}(y) = [1\ \ y\ \ y^2\ \cdots\ y^{n-1}]^t$, $\mathbf{X}(x) = [1\ \ x\ \ x^2\ \cdots\ x^{n-1}]^t$, $\mathbf{A} = [a_{ij}]_{n\times n}$.

Now, the function $\mathbf{F}(x,y)$ is constructed in such a way that

$$\mathbf{F}(x,y) = \mathbf{Y}(y_j)\mathbf{A}\mathbf{X}(x_i),$$

where $\mathbf{F}$ is the rearranged array form of the column vector $F(x_i,y_j)$, and $\mathbf{Y}(y_j), \mathbf{X}(x_i)$ are the matrices derived from $\mathbf{Y}^t(y), \mathbf{X}(x)$ by introducing the points (x_i, y_i).

That is,

$$\mathbf{F} = \begin{bmatrix} F(x_1,y_1) & F(x_1,y_2) & \cdots & F(x_1,y_n) \\ F(x_2,y_1) & F(x_2,y_2) & \cdots & F(x_2,y_n) \\ \cdots & \cdots & \cdots & \cdots \\ F(x_n,y_1) & F(x_n,y_2) & \cdots & F(x_n,y_n) \end{bmatrix},$$

$$\mathbf{X}(x_i) = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix}, \mathbf{Y}^t(y_j) = \begin{bmatrix} 1 & y_1 & y_1^2 & \cdots & y_1^{n-1} \\ 1 & y_2 & y_2^2 & \cdots & y_2^{n-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & y_n & y_n^2 & \cdots & y_n^{n-1} \end{bmatrix}.$$

Since the matrices $\mathbf{X}, \mathbf{Y}$ and $\mathbf{F}$ are known, one can calculate the matrix $\mathbf{A}$ as (assuming $\mathbf{X}$ and $\mathbf{Y}$ are non-singular)

$$\mathbf{A}^* = (\mathbf{Y}^{-1})^t\mathbf{F}\mathbf{X}^{-1}. \tag{3.119}$$

Thus $F(x,y) = \mathbf{Y}^t(y)\mathbf{A}^*\mathbf{X}(x)$ is the required interpolating polynomial.

Example 3.20.3 Obtain a bilinear interpolating polynomial using the following data

y / x	1	2
1	6	10
2	10	18

Solution. Here $\mathbf{X}^t(x) = (1\ \ x)$, $\quad \mathbf{Y}^t(y) = (1\ \ y)$, $\quad n = 2$.

$$\mathbf{X}(x_i) = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}, \mathbf{Y}^t(y_j) = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix},$$

$$\begin{aligned} \mathbf{A}^* &= \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 6 & 10 \\ 10 & 18 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 6 & 10 \\ 10 & 18 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}. \end{aligned}$$

Therefore,

$$F(x,y) = [1 \;\; y]\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ x \end{bmatrix} = 2 + 4xy.$$

Lagrange's bivariate interpolation

Let $f(x,y)$ be a function defined at $(m+1)(n+1)$ distinct points (x_i, y_i), $i = 0, 1, \dots, m; j = 0, 1, \dots, n$. Let us construct a polynomial $F(x,y)$ of degree at most m in x and n in y, such that

$$F(x_i, y_j) = f(x_i, y_j), \qquad i = 0, 1, \dots, m; \;\; j = 0, 1, \dots, n. \tag{3.120}$$

As in Lagrange's polynomial (3.3) for single variable, we define

$$L_{x,i}(x) = \frac{w_x(x)}{(x - x_i)w'_x(x_i)}, \qquad i = 0, 1, \dots, m \tag{3.121}$$

$$L_{y,j}(y) = \frac{w_y(y)}{(y - y_j)w'_y(y_j)}, \qquad j = 0, 1, \dots, n \tag{3.122}$$

where $w_x(x) = (x - x_0)(x - x_1)\cdots(x - x_m)$ and $w_y(y) = (y - y_0)(y - y_1)\cdots(y - y_n)$.

The functions $L_{x,i}(x)$ and $L_{y,j}(y)$ are the polynomials of degree m in x and n in y respectively and also

$$L_{x,i}(x_k) = \begin{cases} 0, & \text{if } x_i \neq x_k \\ 1, & \text{if } x_i = x_k \end{cases} \text{ and } L_{y,j}(y_k) = \begin{cases} 0, & \text{if } y_i \neq y_k \\ 1, & \text{if } y_i = y_k \end{cases} \tag{3.123}$$

Thus the Lagrange's bivariate polynomial is

$$F(x,y) = \sum_{i=0}^{m}\sum_{j=0}^{n} L_{x,i}(x)L_{y,j}(y) \;\; f(x_i, y_j). \tag{3.124}$$

Example 3.20.4 The following data for a function $f(x,y)$ is given:

y / x	0	1
0	1	1.414214
1	1.732051	2

Find the Lagrange's bivariate polynomial and hence find an approximate value of $f(0.25, 0.75)$.

Solution. $m = 1, n = 1, x_0 = 0, y_0 = 0, x_1 = 1, y_1 = 1, f(x_0, y_0) = 1,\; f(x_0, y_1) = 1.414214, f(x_1, y_0) = 1.732051, f(x_1, y_1) = 2$.

Then

$$\begin{aligned}F(x,y) &= \sum_{i=0}^{1}\sum_{j=0}^{1} L_{x,i}(x)L_{y,j}(y)\, f(x_i,y_j)\\ &= L_{x,0}(x)\{L_{y,0}(y)f(x_0,y_0)+L_{y,1}(y)f(x_0,y_1)\}\\ &\quad +L_{x,1}(x)\{L_{y,0}(y)f(x_1,y_0)+L_{y,1}(y)f(x_1,y_1)\}.\end{aligned}$$

Now,

$$\begin{aligned}&L_{x,0}(x)=\frac{x-1}{0-1}=1-x, \qquad L_{y,0}(y)=1-y\\ &L_{x,1}(x)=x, \qquad L_{y,1}(y)=y.\end{aligned}$$

Therefore,

$$\begin{aligned}F(x,y) &= (1-x)(1-y)\times 1+(1-x)y\times 1.414214+x(1-y)\times 1.732051+2xy\\ &= 1+0.732051x+0.414214y-0.146265xy.\end{aligned}$$

Thus,

$$\begin{aligned}F(0.25,0.75) &= 1+0.732051\times 0.25+0.414214\times 0.75-0.146265\times 0.25\times 0.75\\ &= 1.466248563.\end{aligned}$$

Algorithm 3.6 (Lagrange's bivariate interpolation). This algorithm finds the value of $f(x,y)$ by Lagrange's interpolation method when a table of values of x_i, y_j, $f(x_i,y_j)$, $i=0,1,\ldots,m$; $j=0,1,\ldots,n$, is given.

Algorithm Lagrange_Bivariate

```
Read $x_i, i=0,1,\ldots,m$; $y_j, j=0,1,\ldots,n$;// $x$ and $y$ values //
Read $f_{ij}$, $i=0,1,\ldots,m; j=0,1,\ldots,n$;// $f_{ij}=f(x_i,y_j)$//
Read $xg, yg$; //the values of $x$ and $y$ at which $f(x,y)$ is to be determined.//
Set $wx=1, wy=1$;
for $i=0$ to $m$ do //computation of $w_x(x_i)$//
    $wx=wx*(xg-x_i)$;
endfor;
for $j=0$ to $n$ do //find $w_y(y_j)$//
    $wy=wy*(yg-y_j)$;
endfor;
Set $sum=0$;
for $i=0$ to $m$ do
    for $j=0$ to $n$ do
```

```
            Compute $sum = sum + \frac{wx}{(xg - x_i)wdx(i)} * \frac{wy}{(yg - y_j)wdy(j)} * f_{ij}$;
      endfor;
endfor;
Print ‘The value of $f(x, y)$ is ’, $sum$;
end Lagrange_Bivariate

function $wdx(j)$
$sum = 0$;
for $i = 0$ to $m$ do
      if $(i \neq j)$ $sum = sum + (x_j - x_i)$;
endfor;
return $sum$;
end $wdx(j)$

function $wdy(j)$
$sum = 0$;
for $i = 0$ to $n$ do
      if $(i \neq j)$ $sum = sum + (y_j - y_i)$;
endfor;
return $sum$;
end $wdy(j)$
```

Program 3.6

```
/* Program Lagrange bivariate
   This program is used to find the value of a function
   f(x,y) at a given point (x,y) when a set of values of
   f(x,y) is given for different values of x and y, by
   Lagrange bivariate interpolation formula. */
#include<stdio.h>
#include<math.h>
float x[20],y[20];
void main()
 {
    int i,j,n,m;
    float xg,yg,f[20][20],wx=1,wy=1,sum=0;
    float wdx(int j,int m); float wdy(int j,int n);
    printf("Enter the number of subdivisions along x and y ");
    scanf("%d %d",&m,&n);
    printf("Enter x values ");
    for(i=0;i<=m;i++) scanf("%f",&x[i]);
    printf("Enter y values ");
    for(i=0;i<=n;i++) scanf("%f",&y[i]);
    printf("Enter function f(x,y) values \n");
    for(i=0;i<=m;i++) for(j=0;j<=n;j++)
        {printf("f(%f,%f)= ",x[i],y[j]);scanf("%f",&f[i][j]); }
    printf("Enter the interpolating point ");
    scanf("%f %f",&xg,&yg);
    for(i=0;i<=m;i++) wx*=(xg-x[i]);
    for(j=0;j<=n;j++) wy*=(yg-y[j]);
    for(i=0;i<=m;i++)for(j=0;j<=n;j++)
      sum+=wx*wy*f[i][j]/((xg-x[i])*wdx(i,n)*(yg-y[j])*wdy(j,m));
    printf("The interpolated value at
            (%8.5f,%8.5f) is %8.5f ",xg,yg,sum);
 } /* main */
/* function to find w'(x[j]) */
float wdx(int j,int m)
 {
    int i; float prod=1;
    for(i=0;i<=m;i++) if(i!=j) prod*=(x[j]-x[i]);
    return prod;
 }
/* function to find w'(y[j]) */
float wdy(int j,int n)
 {
    int i; float prod=1;
    for(i=0;i<=n;i++) if(i!=j) prod*=(y[j]-y[i]);
    return prod;
 }
```

A sample of input/output:

```
Enter the number of subdivisions along x and y 2 2
Enter x values 0 1 2
Enter y values 0 1 2
Enter function f(x,y) values
f(0.000000,0.000000)= 2
f(0.000000,1.000000)= 3
f(0.000000,2.000000)= 6
f(1.000000,0.000000)= 3
f(1.000000,1.000000)= 5
```

```
f(1.000000,2.000000)= 9
f(2.000000,0.000000)= 6
f(2.000000,1.000000)= 9
f(2.000000,2.000000)= 14
Enter the interpolating point 0.5 0.5
The interpolated value at ( 0.50000, 0.50000) is  2.75000
```

Newton's bivariate interpolation formula

Let $f(x, y)$ be defined at $(m+1)(n+1)$ distinct points (x_i, y_j), $i = 0, 1, \ldots, m; j = 0, 1, \ldots, n$. Also, let $x_s = x_0 + sh$, $y_t = y_0 + tk$, $x = x_0 + mh$ and $y = y_0 + nk$.

Some notations are defined in the following:

$$\begin{aligned}
\Delta_x f(x,y) &= f(x+h,y) - f(x,y) = E_x f(x,y) - f(x,y) \\
&= (E_x - 1) f(x,y) \\
\Delta_y f(x,y) &= f(x,y+k) - f(x,y) = E_y f(x,y) - f(x,y) \\
&= (E_y - 1) f(x,y) \\
\Delta_{xx} f(x,y) &= (E_x^2 - 2E_x + 1) f(x,y) = (E_x - 1)^2 f(x,y) \\
\Delta_{yy} f(x,y) &= (E_y^2 - 1)^2 f(x,y) \\
\Delta_{xy} f(x,y) &= \Delta_x \{f(x,y+k) - f(x,y)\} \\
&= \{f(x+h,y+k) - f(x,y+k)\} - \{f(x+h,y) - f(x,y)\} \\
&= E_x E_y f(x,y) - E_y f(x,y) - E_x f(x,y) + f(x,y) \\
&= (E_x - 1)(E_y - 1) f(x,y)
\end{aligned}$$

and so on.

Then,

$$\begin{aligned}
f(x,y) &= f(x_0 + mh, y_0 + nk) = E_x^m E_y^n f(x_0, y_0) \\
&= (1 + \Delta_x)^m (1 + \Delta_y)^n f(x_0, y_0) \\
&= \left\{1 + m\Delta_x + \frac{m(m-1)}{2!}\Delta_{xx} + \cdots\right\} \\
&\quad \times \left\{1 + n\Delta_y + \frac{n(n-1)}{2!}\Delta_{yy} + \cdots\right\} f(x_0, y_0) \\
&= \left[1 + m\Delta_x + n\Delta_y + \frac{m(m-1)}{2!}\Delta_{xx} + \frac{n(n-1)}{2!}\Delta_{yy}\right. \\
&\quad \left. + mn\Delta_{xy} + \cdots\right] f(x_0, y_0).
\end{aligned}$$

Substituting $m = \dfrac{x - x_0}{h}$ and $n = \dfrac{y - y_0}{k}$.

Then $m-1=\dfrac{x-x_0-h}{h}=\dfrac{x-x_1}{h}$ and $n-1=\dfrac{y-y_1}{k}$.
Thus

$$\begin{aligned}F(x,y) &= f(x_0,y_0)+\left[\frac{x-x_0}{h}\Delta_x+\frac{y-y_0}{k}\Delta_y\right]f(x_0,y_0)\\ &+\frac{1}{2!}\left[\frac{(x-x_0)(x-x_1)}{h^2}\Delta_{xx}+\frac{2(x-x_0)(y-y_0)}{hk}\Delta_{xy}\right.\\ &\left.+\frac{(y-y_0)(y-y_1)}{k^2}\Delta_{yy}\right]f(x_0,y_0)+\cdots \qquad (3.125)\end{aligned}$$

which is called **Newton's bivariate interpolating polynomial**.

Now, introducing unit less quantities u and v defined as $x=x_0+uh$ and $y=y_0+vk$.
Then $x-x_s=(u-s)h$ and $y-y_t=(v-t)k$.
Hence, finally (3.125) becomes

$$\begin{aligned}F(x,y) &= f(x_0,y_0)+[u\Delta_x+v\Delta_y]f(x_0,y_0)+\frac{1}{2!}[u(u-1)\Delta_{xx}\\ &+2uv\Delta_{xy}+v(v-1)\Delta_{yy}]f(x_0,y_0)+\cdots \qquad (3.126)\end{aligned}$$

Example 3.20.5 For the following data obtain Newton's bivariate interpolating polynomial and hence calculate the values of $f(0.75,0.25)$ and $f(1.25,1.5)$.

y / x	0	1	2
0	1	3	5
1	–1	2	5
2	–5	–1	3

Solution.

$$\begin{aligned}\Delta_x f(x_0,y_0) &= f(x_0+h,y_0)-f(x_0,y_0)\\ &= f(x_1,y_0)-f(x_0,y_0)=-1-1=-2\\ \Delta_y f(x_0,y_0) &= f(x_0,y_0+k)-f(x_0,y_0)\\ &= f(x_0,y_1)-f(x_0,y_0)=3-1=2\\ \Delta_{xx} f(x_0,y_0) &= f(x_0+2h,y_0)-2f(x_0+h,y_0)+f(x_0,y_0)\\ &= f(x_2,y_0)-2f(x_1,y_0)+f(x_0,y_0)=-5-2\times(-1)+1=-2\\ \Delta_{yy} f(x_0,y_0) &= f(x_0,y_0+2k)-2f(x_0,y_0+k)+f(x_0,y_0)\\ &= f(x_0,y_2)-2f(x_0,y_1)+f(x_0,y_0)=5-2\times 3+1=0\\ \Delta_{xy} f(x_0,y_0) &= f(x_0+h,y_0+k)-f(x_0,y_0+k)-f(x_0+h,y_0)+f(x_0,y_0)\\ &= f(x_1,y_1)-f(x_0,y_1)-f(x_1,y_0)+f(x_0,y_0)=1.\end{aligned}$$

Here $h = k = 1$. $u = \dfrac{x - x_0}{h} = x, v = \dfrac{y - y_0}{k} = y$.
Thus,

$$\begin{aligned} F(x, y) &= 1 + [x \times (-2) + y \times 2] \\ &\quad + \frac{1}{2!}[x(x-1) \times (-2) + 2xy \times 1 + y(y-1) \times 0] \\ &= 1 - x + 2y - x^2 + xy. \end{aligned}$$

Hence $f(0.75, 0.25) \simeq F(0.75, 0.25) = 0.375$ and $f(1.25, 1.5) \simeq F(1.25, 1.5) = 3.0625$.

3.21 Worked out Examples

Example 3.21.1 Using Lagrange's interpolation formula, express the function

$$\frac{x^2 + 2x + 3}{(x+1)x(x-1)}$$

as sums of partial fractions.

Solution. Let $f(x) = x^2 + 2x + 3$. Now, $f(x)$ is tabulated for $x = -1, 0, 1$ as follows:

x	:	-1	0	1
$f(x)$	:	2	3	6

The Lagrange's functions are

$$\begin{aligned} L_0(x) &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{x(x-1)}{2}. \\ L_1(x) &= \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(x+1)(x-1)}{-1}. \\ L_2(x) &= \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{(x+1)x}{2}. \end{aligned}$$

By Lagrange's interpolation formula the polynomial $f(x)$ is given by

$$\begin{aligned} f(x) &= \frac{x(x-1)}{2} \times 2 + \frac{(x+1)(x-1)}{-1} \times 3 + \frac{(x+1)x}{2} \times 6 \\ &= x(x-1) - 3(x+1)(x-1) + 3x(x+1). \end{aligned}$$

Hence

$$\frac{x^2 + 2x + 3}{(x+1)x(x-1)} = \frac{f(x)}{(x+1)x(x-1)} = \frac{1}{x+1} - \frac{3}{x} + \frac{3}{x-1}.$$

Example 3.21.2 Find the missing term in the following table

x	:	0	1	2	3	4
y	:	1	2	4	?	16

Solution.
Method 1.
Using Lagrange's formula

$$\begin{aligned}
L_0(x) &= \frac{(x-1)(x-2)(x-4)}{(0-1)(0-2)(0-4)} = \frac{x^3-7x^2+14x-8}{-8}.\\
L_1(x) &= \frac{(x-0)(x-2)(x-4)}{(1-0)(1-2)(1-4)} = \frac{x^3-6x^2+8x}{3}.\\
L_2(x) &= \frac{(x-0)(x-1)(x-4)}{(2-0)(2-1)(2-4)} = \frac{x^3-5x^2+4x}{-4}.\\
L_3(x) &= \frac{(x-0)(x-1)(x-2)}{(4-0)(4-1)(4-2)} = \frac{x^3-3x^2+2x}{24}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
y(x) &\simeq y_0L_0(x)+y_1L_1(x)+y_2L_2(x)+y_3L_3(x)\\
&= \frac{x^3-7x^2+14x-8}{-8}\times 1+\frac{x^3-6x^2+8x}{3}\times 2\\
&\quad+\frac{x^3-5x^2+4x}{-4}\times 4+\frac{x^3-3x^2+2x}{24}\times 16\\
&= \frac{5}{24}x^3-\frac{1}{8}x^2+\frac{11}{12}x+1.
\end{aligned}$$

Thus, $y(3) = 8.25$.
Hence the missing term is 8.25.
Method 2.
Let us construct a polynomial of degree 3 in the form

$$y(x) = a+bx+cx^2+dx^3.$$

If the curve passes through the points $x = 0, 1, 2, 4$, then
$a = 1, \qquad a+b+c+d = 2,$
$a+2b+4c+8d = 4, \qquad a+4b+16c+64d = 16.$
Solution of these equations is

$$a = 1, b = \frac{11}{12}, c = -\frac{1}{8} \text{ and } d = \frac{5}{24}.$$

Therefore,

$$y(x) = 1 + \frac{11}{12}x - \frac{1}{8}x^2 + \frac{5}{24}x^3.$$

Thus $y(3) = 8.25$.

Example 3.21.3 Let $f(x) = \log x$, $x_0 = 2$ and $x_1 = 2.1$. Use linear interpolation to calculate an approximate value for $f(2.05)$ and obtain a bound on the truncation error.

Solution. Let $f(x) = \log x$. The table is

x	:	2.0	2.1
y	:	0.693147	0.741937

The linear interpolation polynomial is

$$\begin{aligned}\phi(x) &= \frac{(x-2.1)}{(2.0-2.1)} \times 0.693147 + \frac{(x-2.0)}{(2.1-2.0)} \times 0.741937 \\ &= 0.487900x - 0.282653.\end{aligned}$$

Thus, $f(2.05) \simeq \phi(2.05) = 0.717542$.
The error in linear interpolation is

$$|E_1(x)| = |(x-x_0)(x-x_1)|\left|\frac{f''(\xi)}{2!}\right|, \qquad 2 \leq \xi \leq 2.1.$$

The maximum value of $f''(x) = -\dfrac{1}{x^2}$ in $2 \leq x \leq 2.1$ is $|f''(2.0)| = 0.25$.
Then

$$|E_1(x)| \leq \left|(2.05-2)(2.05-2.1)\frac{0.25}{2}\right| = 0.000313.$$

Thus the upper bound of truncation error is 0.000313.

Example 3.21.4 For the following table find the value of y at $x = 2.5$, using piecewise linear interpolation.

x	:	1	2	3	4	5
y	:	35	40	65	72	80

Solution. The point $x = 2.5$ lies between 2 and 3. Then

$$y(2.5) = \frac{2.5-3}{2-3} \times 40 + \frac{2.5-2}{3-2} \times 65 = 52.5.$$

Example 3.21.5 Deduce the following interpolation formula taking three points $x_0, x_0+\varepsilon, \varepsilon \to 0$ and x_1 using Lagrange's formula.

$$\begin{aligned} f(x) &= \frac{(x_1-x)(x+x_1-2x_0)}{(x-x_0)^2} f(x_0) + \frac{(x-x_0)(x_1-x)}{(x_1-x_0)} f'(x_0) \\ &\quad + \frac{(x-x_0)^2}{(x_1-x_0)^2} f(x_1) + E(x) \end{aligned}$$

where

$$E(x) = \frac{1}{6}(x-x_0)^2(x-x_1)f'''(\xi) \text{ and } \min\{x_0, x_0+\varepsilon, x_1\} \le \xi \le \max\{x_0, x_0+\varepsilon, x_1\}.$$

Solution. The Lagrange's interpolating polynomial for the points $x_0, x_0+\varepsilon$ and x_1 is

$$\begin{aligned} f(x) &\simeq \frac{(x-x_0-\varepsilon)(x-x_1)}{(x_0-x_0-\varepsilon)(x_0-x_1)} f(x_0) + \frac{(x-x_0)(x-x_1)}{(x_0+\varepsilon-x_0)(x_0+\varepsilon-x_1)} f(x_0+\varepsilon) \\ &\quad + \frac{(x-x_0)(x-x_0-\varepsilon)}{(x_1-x_0)(x_1-x_0-\varepsilon)} f(x_1) + E(x) \\ &= \left[\frac{(x-x_0-\varepsilon)(x-x_1)}{-\varepsilon(x_0-x_1)} f(x_0) + \frac{(x-x_0)(x-x_1)}{\varepsilon(x_0+\varepsilon-x_1)} f(x_0)\right] \\ &\quad + \frac{(x-x_0)(x-x_1)}{(x_0-x_1+\varepsilon)} \left[\frac{f(x_0+\varepsilon)-f(x_0)}{\varepsilon}\right] \\ &\quad + \frac{(x-x_0)(x-x_0-\varepsilon)}{(x_1-x_0)(x_1-x_0-\varepsilon)} f(x_1) + E(x) \\ &= \frac{(2x_0-x_1-x)(x-x_1)}{(x_0-x_1)(x_0-x_1+\varepsilon)} f(x_0) + \frac{(x-x_0)(x-x_1)}{(x_0-x_1+\varepsilon)} \left[\frac{f(x_0+\varepsilon)-f(x_0)}{\varepsilon}\right] \\ &\quad + \frac{(x-x_0)(x-x_0-\varepsilon)}{(x_1-x_0)(x_1-x_0-\varepsilon)} f(x_0) + E(x) \\ &= \frac{(x_1-x)(x+x_1-2x_0)}{(x_1-x_0)^2} f(x_0) + \frac{(x-x_0)(x_1-x)}{(x_1-x_0)} f'(x_0) \\ &\quad + \frac{(x-x_0)^2}{(x_1-x_0)^2} f(x_1) + E(x) \text{ as } \varepsilon \to 0. \end{aligned}$$

The error term is

$$\begin{aligned} E(x) &= (x-x_0)(x-x_0-\varepsilon)(x-x_1)\frac{f'''(\xi)}{3!} \\ &= \frac{1}{6}(x-x_0)^2(x-x_1)f'''(\xi), \text{ as } \varepsilon \to 0 \end{aligned}$$

and $\min\{x_0, x_0+\varepsilon, x_1\} \le \xi \le \max\{x_0, x_0+\varepsilon, x_1\}$.

Example 3.21.6 The standard normal probability integral

$$P(x) = \sqrt{\frac{2}{\pi}} \int_0^x exp\Big(-\frac{1}{2}t^2\Big)\ dt$$

has the following values

x	:	1.00	1.05	1.10	1.15	1.20	1.25
$P(x)$	:	0.682689	0.706282	0.728668	0.749856	0.769861	0.788700

Calculate $P(1.235)$.

Solution. The backward difference table is

x	$P(x)$	∇P	$\nabla^2 P$	$\nabla^3 P$
1.00	0.682689			
1.05	0.706282	0.023593		
1.10	0.728668	0.022386	−0.001207	
1.15	0.749856	0.021188	−0.001198	0.000009
1.20	0.769861	0.020005	−0.001183	0.000015
1.25	0.788700	0.018839	−0.001166	0.000017

Here $x = 1.235, h = 0.05, x_n = 1.25, v = (x - x_n)/h = (1.235 - 1.25)/0.05 = -0.3$.

$$\begin{aligned} P(1.235) &= P(x_n) + v\nabla P(x_n) + \frac{v(v+1)}{2!}\nabla^2 P(x_n) + \frac{v(v+1)(v+2)}{3!}\nabla^3 P(x_n) \\ &= 0.788700 - 0.3 \times 0.018839 + \frac{-0.3(-0.3+1)}{2} \times (-0.001166) \\ &\quad + \frac{-0.3(-0.3+1)(-0.3+2)}{6} \times 0.000017 \\ &= 0.783169. \end{aligned}$$

Example 3.21.7 Find the seventh and the general terms of the series 3, 9, 20, 38, 65, ...

Solution. Let $x_i = i, i = 1, 2, 3, 4, 5$ and $y_0 = 3, y_1 = 9, y_2 = 20, y_3 = 38$, $y_4 = 65$. We construct the Newton's backward interpolation polynomial using these values.

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$
1	3			
2	9	6		
3	20	11	5	
4	38	18	7	2
5	65	27	9	2

Here $x_n = 5, v = (x - x_n)/h = x - 5$.

$$\begin{aligned}
\phi(x) &= y_n + v\nabla y_n + \frac{v(v+1)}{2!}\nabla^2 y_n + \frac{v(v+1)(v+2)}{3!}\nabla^3 y_n \\
&= 65 + 27v + 9\frac{v(v+1)}{2!} + 2\frac{v(v+1)(v+2)}{3!} \\
&= \frac{1}{6}(2v^3 + 33v^2 + 193v + 390) \\
&= \frac{1}{6}[2(x-5)^3 + 33(x-5)^2 + 193(x-5) + 390] \\
&= \frac{1}{6}(2x^3 + 3x^2 + 13x).
\end{aligned}$$

The seventh term is

$$\phi(7) = \frac{1}{6}(2 \times 7^3 + 3 \times 7^2 + 13 \times 7) = 154.$$

[Other interpolation formulae may also be used to solve this problem.]

Example 3.21.8 From the following table of $\sin x$ compute $\sin 12^0$ and $\sin 45^0$.

x	:	10^0	20^0	30^0	40^0	50^0
$y = \sin x$	:	0.17365	0.34202	0.50000	0.64279	0.76604

Solution. The difference table is

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
10^0	0.17365				
		0.16837			
20^0	0.34202		−0.01039		
		0.15798		−0.00480	
30^0	0.50000		−0.01519		0.00045
		0.14279		−0.00435	
40^0	0.64279		−0.01954		
		0.12325			
50^0	0.76604				

(i) To find $\sin 12^0$.
Here $x_0 = 10^0, x = 12^0, h = 10^0, u = \frac{(x-x_0)}{h} = \frac{(12^0-10^0)}{10^0} = 0.2$.
By Newton's forward formula

$$y(12^0) = \sin 12^0 \simeq y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0$$

$$
\begin{aligned}
& +\frac{u(u-1)(u-2)(u-3)}{4!}\Delta^4 y_0 \\
= {} & 0.17365 + 0.2 \times 0.16837 + \frac{0.2(0.2-1)}{2} \times (-0.01039) \\
& +\frac{0.2(0.2-1)(0.2-2)}{6} \times (-0.00480) \\
& +\frac{0.2(0.2-1)(0.2-2)(0.2-3)}{24} \times (0.000450) \\
= {} & 0.20791.
\end{aligned}
$$

(ii) To find $\sin 45^0$.
Here $x_n = 50^0, x = 45^0, h = 10^0, v = (x - x_n)/h = (45^0 - 50^0)/10^0 = -0.5$.
By Newton's backward formula

$$
\begin{aligned}
y(45^0) &= \sin 45^0 \\
&= 0.76604 - 0.5 \times 0.12325 + \frac{-0.5(-0.5+1)}{2} \times (-0.01954) \\
&\quad +\frac{-0.5(-0.5+1)(-0.5+2)}{6} \times (-0.00435) \\
&\quad +\frac{-0.5(-0.5+1)(-0.5+2)(-0.5+3)}{24} \times (0.00045) \\
&= 0.70711.
\end{aligned}
$$

Example 3.21.9 Use Stirling's formula to find u_{32} from the following table

$$
\begin{aligned}
& u_{20} = 14.035,\ u_{25} = 13.674,\ u_{30} = 13.257, \\
& u_{35} = 12.734,\ u_{40} = 12.089,\ u_{45} = 11.309.
\end{aligned}
$$

Solution. The finite difference table is shown below.

i	x_i	u_{x_i}	Δu_{x_i}	$\Delta^2 u_{x_i}$	$\Delta^3 u_{x_i}$
-2	20	14.035			
			-0.361		
-1	25	13.674		-0.056	
			-0.417		-0.050
0	30	13.257		-0.106	
			-0.523		-0.016
1	35	12.734		-0.122	
			-0.645		-0.013
2	40	12.089		-0.135	
			-0.780		
3	45	11.309			

Here $x = 32, x_0 = 30, h = 5, s = (x - x_0)/h = 0.4$.
Therefore,

$$\begin{aligned}
u_{32} &= y_0 + s\frac{\Delta y_{-1} + \Delta y_0}{2} + \frac{s^2}{2!}\Delta^2 y_{-1} + \frac{s(s^2-1)}{3!}\frac{\Delta^3 y_{-2} + \Delta^3 y_{-1}}{2} \\
&= 13.257 + 0.4\frac{-0.417 - 0.523}{2} + \frac{0.4^2}{2}(-0.106) \\
&\quad + \frac{0.4(0.4^2-1)}{6}\frac{-0.050 - 0.016}{2} \\
&= 13.059.
\end{aligned}$$

Example 3.21.10 The function $y = \sqrt[3]{x}$ is tabulated below.

x :	5600	5700	5800	5900	6000
y :	17.75808	17.86316	17.96702	18.06969	18.17121

Compute $\sqrt[3]{5860}$ by (i) Bessel's formula, and (ii) Stirling's formula.

Solution. The finite difference table is

i	x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
−2	5600	17.75808				
			0.10508			
−1	5700	17.86316		−0.00122		
			0.10386		0.00003	
0	5800	17.96702		−0.00119		0.00001
			0.10267		0.00004	
1	5900	18.06969		−0.00115		
			0.10152			
2	6000	18.17121				

(i) For $x = 5860$, let us take $x_0 = 5800$, then $s = (5860 - 5800)/100 = 0.6$.
By Bessel's formula

$$\begin{aligned}
y(5860) &= \frac{y_0 + y_1}{2} + (s - 0.5)\Delta y_0 + \frac{s(s-1)}{2!}\frac{\Delta^2 y_0 + \Delta^2 y_{-1}}{2} \\
&\quad + \frac{1}{3!}(s - 0.5)s(s-1)\Delta^3 y_{-1} \\
&= \frac{17.96702 + 18.06969}{2} + (0.6 - 0.5) \times 0.10267
\end{aligned}$$

$$
\begin{aligned}
&+\frac{0.6(0.6-1)}{2!}\frac{-0.00115-0.00119}{2}\\
&+\frac{1}{6}(0.6-0.5)(0.6)(0.6-1)\times 0.00004\\
=\ & 18.02877.
\end{aligned}
$$

(ii) By Stirling's formula

$$
\begin{aligned}
y(5860) &= y_0 + s\frac{\Delta y_{-1}+\Delta y_0}{2} + \frac{s^2}{2!}\Delta^2 y_{-1} + \frac{s(s^2-1)}{3!}\frac{\Delta^3 y_{-2}+\Delta^3 y_{-1}}{2}\\
&= 17.96702 + 0.6\frac{0.10386+0.10267}{2} + \frac{0.6^2}{2}(-0.00119)\\
&\quad+\frac{0.6(0.6^2-1)}{6}\frac{0.00003+0.00004}{2}\\
&= 18.02877.
\end{aligned}
$$

Thus $\sqrt[3]{5860} = 18.02877$.

Example 3.21.11 Prove that the third order divided difference of the function $f(x) = \frac{1}{x}$ with arguments a, b, c, d is $-\frac{1}{abcd}$.

Solution. Here $f(x) = \frac{1}{x}$.

$$f[a,b] = \frac{f(b)-f(a)}{b-a} = \frac{\frac{1}{b}-\frac{1}{a}}{b-a} = -\frac{1}{ab}.$$

$$f[a,b,c] = \frac{f[a,b]-f[b,c]}{a-c} = \frac{-\frac{1}{ab}+\frac{1}{bc}}{a-c} = \frac{1}{abc}.$$

The third order divided difference is

$$f[a,b,c,d] = \frac{f[a,b,c]-f[b,c,d]}{a-d} = \frac{\frac{1}{abc}-\frac{1}{bcd}}{a-d} = -\frac{1}{abcd}.$$

Example 3.21.12 If $f(x) = \frac{1}{x}$, prove that $f[x_0, x_1, \ldots, x_n] = \frac{(-1)^n}{x_0 x_1 \cdots x_n}$.

Solution. The first order divided difference is

$$f[x_0,x_1] = \frac{f(x_1)-f(x_0)}{x_1-x_0} = \frac{\frac{1}{x_1}-\frac{1}{x_0}}{x_1-x_0} = -\frac{1}{x_0x_1} = \frac{(-1)^1}{x_0x_1}.$$

The second order divided difference is

$$f[x_0,x_1,x_2]=\frac{f[x_0,x_1]-f[x_1,x_2])}{x_0-x_2}=\frac{-\frac{1}{x_0x_1}+\frac{1}{x_1x_2}}{x_0-x_2}=\frac{(-1)^2}{x_0x_1x_2}.$$

Thus the result is true for $n=1,2$.
Let the result be true for $n=k$, i.e.,

$$f[x_0,x_1,\ldots,x_k]=\frac{(-1)^k}{x_0x_1\cdots x_k}.$$

Now,

$$\begin{aligned}
f[x_0,x_1,\ldots,x_k,x_{k+1}] &= \frac{f[x_0,x_1,\ldots,x_k]-f[x_1,x_2,\ldots,x_{k+1}]}{x_0-x_{k+1}}\\
&= \frac{1}{x_0-x_{k+1}}\left[\frac{(-1)^k}{x_0x_1\cdots x_k}-\frac{(-1)^k}{x_1x_2\cdots x_{k+1}}\right]\\
&= \frac{(-1)^k}{x_1x_2\cdots x_k}\;\frac{1}{x_0-x_{k+1}}\left(\frac{1}{x_0}-\frac{1}{x_{k+1}}\right)\\
&= \frac{(-1)^{k+1}}{x_0x_1x_2\cdots x_{k+1}}.
\end{aligned}$$

Therefore, the result is true for $n=k+1$. Hence by mathematical induction the result is true for $n=1,2,\ldots$.

Example 3.21.13 If $f(x)=u(x)v(x)$ then show that

$$f[x_0,x_1]=u(x_0)v[x_0,x_1]+v(x_1)u[x_0,x_1]$$

and hence deduce that if $g(x)=w^2(x)$ then

$$g[x_0,x_1]=w[x_0,x_1][w(x_0)+w(x_1)].$$

Solution.

$$\begin{aligned}
f[x_0,x_1] &= \frac{f(x_0)-f(x_1)}{x_0-x_1}=\frac{u(x_0)v(x_0)-u(x_1)v(x_1)}{x_0-x_1}\\
&= \frac{u(x_0)[v(x_0)-v(x_1)]+v(x_1)[u(x_0)-u(x_1)]}{x_0-x_1}\\
&= u(x_0)v[x_0,x_1]+v(x_1)u[x_0,x_1].
\end{aligned}$$

If $g(x)=w^2(x)$ then let $u(x)=v(x)=w(x)$.
Then

$$\begin{aligned}
g[x_0,x_1] &= w(x_0)w[x_0,x_1]+w(x_1)w[x_0,x_1]\\
&= w[x_0,x_1][w(x_0)+w(x_1)].
\end{aligned}$$

Example 3.21.14 Show that the nth divided difference $f[x_0, x_1, \ldots, x_n]$ can be expressed as

$$f[x_0, x_1, \ldots, x_n] = D \div V, \text{ where}$$

$$D = \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_0 & x_1 & x_2 & \cdots & x_n \\ x_0^2 & x_1^2 & x_2^2 & \cdots & x_n^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_0^{n-1} & x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \\ y_0 & y_1 & y_2 & \cdots & y_n \end{vmatrix}$$

and

$$V = \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_0 & x_1 & x_2 & \cdots & x_n \\ x_0^2 & x_1^2 & x_2^2 & \cdots & x_n^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_0^{n-1} & x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \\ x_0^n & x_1^n & x_2^n & \cdots & x_n^n \end{vmatrix}.$$

Solution. The **Vandermonde's** determinant

$$V(x_0, x_1, \ldots, x_n) = \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ x_0 & x_1 & x_2 & \cdots & x_n \\ x_0^2 & x_1^2 & x_2^2 & \cdots & x_n^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_0^{n-1} & x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \\ x_0^n & x_1^n & x_2^n & \cdots & x_n^n \end{vmatrix}.$$

Let

$$V(x_0, x_1, \ldots, x_{n-1}, x) = \begin{vmatrix} 1 & 1 & \cdots & 1 & 1 \\ x_0 & x_1 & \cdots & x_{n-1} & x \\ x_0^2 & x_1^2 & \cdots & x_{n-1}^2 & x^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_0^n & x_1^n & \cdots & x_{n-1}^n & x^n \end{vmatrix}.$$

When $x = x_0, x_1, \ldots, x_{n-1}$ then $V = 0$,
i.e., $(x - x_0), (x - x_1), (x - x_2), \ldots, (x - x_{n-1})$ are the factors of V.
Then one can write

$$V(x_0, x_1, \ldots, x_{n-1}, x) = (x - x_0)(x - x_1) \cdots (x - x_{n-1})\Delta,$$

where Δ is a constant.

Equating the coefficient of x^n, the value of Δ is given by

$$\Delta = \begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_{n-1} \\ x_0^2 & x_1^2 & \cdots & x_{n-1}^2 \\ \cdots & \cdots & \cdots & \cdots \\ x_0^n & x_1^n & \cdots & x_{n-1}^n \end{vmatrix} = V(x_0, x_1, \ldots, x_{n-1}).$$

Therefore,

$$V(x_0, x_1, \ldots, x_n) = V(x_0, x_1, \ldots, x_{n-1}) \prod_{i=0}^{n-1} (x_n - x_i).$$

Applying this result successively, the explicit expression for V is given by

$$\begin{aligned} V(x_0, x_1, \ldots, x_n) &= \{(x_0 - x_1)(x_0 - x_2) \cdots (x_0 - x_n)\} \times \\ &\quad \{(x_1 - x_2)(x_1 - x_3) \cdots (x_1 - x_n)\} \times \\ &\quad \{(x_2 - x_3)(x_2 - x_4) \cdots (x_2 - x_n)\} \times \\ &\quad \cdots \{(x_{n-2} - x_{n-1})(x_{n-2} - x_n)\}\{(x_0 - x_n)\} \\ &= \prod_{i=0}^{n} \prod_{\substack{j=0 \\ i>j}}^{n} (x_i - x_j) \end{aligned}$$

Let us consider the determinant

$$C(x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \begin{vmatrix} 1 & 1 & \cdots & 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_{i-1} & x_{i+1} & \cdots & x_n \\ x_0^2 & x_1^2 & \cdots & x_{i-1}^2 & x_{i+1}^2 & \cdots & x_n^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_0^{n-1} & x_1^{n-1} & \cdots & x_{i-1}^{n-1} & x_{i+1}^{n-1} & \cdots & x_n^{n-1} \end{vmatrix}.$$

It can be shown that

$$C(x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = (-1)^n V(x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n).$$

Therefore,

$$\begin{aligned} &\frac{V(x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)}{V(x_0, x_1, x_2, \ldots, x_n)} \\ &= \frac{1}{(x_0 - x_i)(x_1 - x_i) \cdots (x_{i-1} - x_i)(x_i - x_{i+1}) \cdots (x_i - x_n)} \\ &= \frac{(-1)^i}{(x_i - x_0)(x_i - x_1) \cdots (x_i - x_{i+1})(x_i - x_{i+1}) \cdots (x_i - x_n)}. \end{aligned}$$

Now,

$$\begin{aligned} D &= (-1)^n y_0 C(x_1, x_2, \ldots, x_n) + (-1)^{n+1} y_1 C(x_0, x_2, \ldots, x_n) + \cdots \\ &\quad +(-1)^{n+n} y_n C(x_0, x_1, \ldots, x_{n-1}) \\ &= \sum_{i=0}^{n} (-1)^i y_i V(x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n). \end{aligned}$$

Thus,

$$\begin{aligned} D \div V &= \sum_{i=0}^{n} (-1)^i y_i V(x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)/V(x_0, x_1, \ldots, x_n) \\ &= \sum_{i=0}^{n} (-1)^i y_i \frac{(-1)^i}{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)} \\ &= \sum_{i=0}^{n} \frac{y_i}{(x_i - x_0)(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)} \\ &= f[x_0, x_1, \ldots, x_n]. \end{aligned}$$

Example 3.21.15 Given $y = f(x)$ in the following table,

x	:	10	15	17
y	:	3	7	11

Find the values of x when $y = 10$ and $y = 5$.

Solution. Here, inverse Lagrange's interpolation formula is used in the following form

$$\phi(y) = \sum_{i=0}^{n} L_i(y) x_i.$$

The Lagrangian functions

$$\begin{aligned} L_0(y) &= \frac{(y - y_1)(y - y_2)}{(y_0 - y_1)(y_0 - y_2)} = \frac{(y-7)(y-11)}{(3-7)(3-11)} = \frac{y^2 - 18y + 77}{32}, \\ L_1(y) &= \frac{(y - y_0)(y - y_2)}{(y_1 - y_0)(y_1 - y_2)} = \frac{(y-3)(y-11)}{(7-3)(7-11)} = \frac{y^2 - 14y + 33}{-16}, \\ \text{and } L_2(y) &= \frac{(y - y_0)(y - y_1)}{(y_2 - y_0)(y_2 - y_1)} = \frac{(y-3)(y-7)}{(11-3)(11-7)} = \frac{y^2 - 10y + 21}{32}. \end{aligned}$$

Then

$$\begin{aligned}\phi(y) &= \frac{y^2-18y+77}{32}\times 10 - \frac{y^2-14y+33}{16}\times 15 + \frac{y^2-10y+21}{32}\times 17\\ &= \frac{1}{32}(137+70y-3y^2).\end{aligned}$$

Hence,

$$\begin{aligned} x(10) &\simeq \phi(10) = \frac{1}{32}(137+700-300) = 16.78125\\ \text{and } x(5) &\simeq \phi(5) = \frac{1}{32}(137+350-75) = 12.87500.\end{aligned}$$

Example 3.21.16 Use inverse Lagrange's interpolation to find a root of the equation $y \equiv x^3 - 3x + 1 = 0$.

Solution. Here $y(0) = 1 > 0$ and $y(1) = -1 < 0$. One root lies between 0 and 1. Now, x and y are tabulated, considering five points of x as $0, 0.25, 0.50, 0.75$ and 1.

x :	0	0.25	0.50	0.75	1.00
y :	1	0.26563	−0.37500	−0.82813	−1.0000

Solution. Here $y = 0$. Then

$$\begin{aligned}L_0(y) &= \frac{(y-y_1)(y-y_2)(y-y_3)(y-y_4)}{(y_0-y_1)(y_0-y_2)(y_0-y_3)(y_0-y_4)}\\ &= \frac{y_1y_2y_3y_4}{(y_0-y_1)(y_0-y_2)(y_0-y_3)(y_0-y_4)} = \frac{-0.08249}{3.69194} = -0.02234.\\ L_1(y) &= \frac{y_0y_2y_3y_4}{(y_1-y_0)(y_1-y_2)(y_1-y_3)(y_1-y_4)} = \frac{-0.31054}{-0.65125} = 0.47684.\\ L_2(y) &= \frac{y_0y_1y_3y_4}{(y_2-y_0)(y_2-y_1)(y_2-y_3)(y_2-y_4)} = \frac{0.21997}{0.24947} = 0.88176.\\ L_3(y) &= \frac{y_0y_1y_2y_4}{(y_3-y_0)(y_3-y_1)(y_3-y_2)(y_3-y_4)} = \frac{0.09961}{-0.15577} = -0.63966.\\ L_4(y) &= \frac{y_0y_1y_2y_3}{(y_4-y_0)(y_4-y_1)(y_4-y_2)(y_4-y_3)} = \frac{0.08249}{0.27190} = 0.30338.\end{aligned}$$

Therefore,

$$\begin{aligned} x \simeq \phi(0) &= \sum_{i=0}^{4} L_i(y)x_i = -0.02234\times 0 + 0.47684\times 0.25\\ &\quad +0.88176\times 0.50 + 0.06014\times 0.75 + 0.30338\times 1\\ &= 0.38373.\end{aligned}$$

Hence, the approximate root is 0.38373.

3.22 Exercise

1. Show that
$$\sum_{i=1}^{n} \frac{w(x)}{(x-x_i)w'(x_i)} = 1.$$

2. Show that $L_0(x) + L_1(x) + L_2(x) = 1$ for all x.

3. Find a polynomial for $f(x)$ where $f(0) = 1, f(1) = 2$ and $f(3) = 5$, using Lagrange's method.

4. Show that the truncation error on quadratic interpolation in an equidistant table is bounded by
$$\left(\frac{h^2}{9\sqrt{3}}\right) \max |f'''(\xi)|.$$

5. Suppose that $f(x) = e^x \cos x$ is to be approximated on [0,1] by an interpolating polynomial on $n+1$ equally spaced points $0 = x_0 < x_1 < \cdots < x_n = 1$. Determine n so that the truncation error will be less than 0.0001 in this interval.

6. Determine an appropriate step size to use, in the construction of a table of $f(x) = (1+x)^4$ on [0,1]. The truncation error for linear interpolation is to be bounded by 5×10^{-5}.

7. The function defined by $f(x) = \displaystyle\int_1^x \frac{dt}{2\sqrt{t}}$ is tabulated for equally spaced values of x with $h = 0.1$. What is the maximum error encountered if piecewise quadratic interpolation is to be used to calculate $f(a)$ where $a \in [1, 2]$?

8. Find the formula for the upper bound of the error involved in linearly interpolating $f(x)$ between a and b. Use the formula to find the maximum error encountered when $f(x) = \displaystyle\int_0^x e^{-t^2}\, dt$ is interpolated between $x = 0$ and $x = 1$.

9. From the following table, find the number of students who obtain less than 35 marks

Marks :	20-30	30-40	40-50	50-60	60-70
No. of Students :	32	53	50	38	33

10. If $y(1) = -3, y(3) = 9, y(4) = 30$ and $y(6) = 132$, find the Lagrange's interpolation polynomial that takes the same values as the function y at the given points.

11. Let the following observation follows the law of a cubic polynomial

x	:	0	1	2	3	4
$f(x)$	:	1	-2	1	16	49

Find the extrapolated value of $f(5)$.

12. Use Lagrange's interpolation formula to express the function

$$\frac{3x^2+2x-5}{(x-1)(x-2)(x-3)}$$

as sums of partial fractions.

13. Express the function

$$\frac{x^3+6x+2}{(x+1)x(x-2)(x-4)}$$

as sums of partial fractions.

14. Using Lagrange's interpolation formula, express the function $f(x) = 3x^2 - 2x + 5$ as the sum of products of the factors $(x-1), (x-2)$ and $(x-3)$ taken two at a time.

15. Compute the missing values of y_n and Δy_n in the following table

y_n	Δy_n	$\Delta^2 y_n$
–		
	–	
–		1
	–	
–		4
	5	
6		13
	–	
–		18
	–	
–		24
	–	
–		

16. The following table gives pressure of a steam plant at a given temperature. Using Newton's formula, compute the pressure for a temperature of 142^{0}C.

Temperature ^{0}C	:	140	150	160	170	180
Pressure, kgf/cm^2	:	3.685	4.854	6.302	8.076	10.225

17. The following data gives the melting point of an alloy of lead and zinc; where T is the temperature in ^{0}C and P is the percentage of lead in the alloy. Find the melting point of the alloy containing 84% of lead using Newton's interpolation method.

P :	50	60	70	80
T :	222	246	272	299

18. Using a polynomial of third degree, complete the record of the export of a certain commodity during five years, as given below.

Year, x :	1988	1989	1990	1991	1992
Export in tons, y :	450	388	–	400	470

19. Find the polynomial which attains the following values at the given points.

x :	-1	0	1	2	3	4
$f(x)$:	-16	-7	-4	-1	8	29

20. Compute $\log_{10} 2.5$ using Newton's forward difference interpolation formula, given that

x :	2.0	2.2	2.4	2.6	2.8	3.0
$\log_{10} x$:	0.30103	0.34242	0.38021	0.41497	0.44716	0.47721

21. Find the missing term in the following table:

x :	0	1	2	3	4
y :	1	3	9	–	81

Why the result differs from $3^3 = 27$?

22. In the following table, the value of y are consecutive terms of a series of which the number 36 is the fifth term. Find the first and the tenth terms of the series. Find also the polynomial which approximates these values.

x :	3	4	5	6	7	8	9
y :	18	26	36	48	62	78	96

23. From the following table determine (a) $f(0.27)$, and (b) $f(0.33)$.

x :	0.24	0.26	0.28	0.30	0.32	0.34
$f(x)$:	1.6804	1.6912	1.7024	1.7139	1.7233	1.7532

24. The population of a town in decennial census were as under. Estimate the population for the year 1955.

Year	:	1921	1931	1941	1951	1961
Population (in core)	:	46	68	83	95	105

25. Using Gauss's forward formula, find the value of $f(32)$ given that

    $f(25) = 0.2707, f(30) = 0.3027, f(35) = 0.3386, f(40) = 0.3794.$

26. Using Gauss's backward formula, find the value of $\sqrt{518}$ given that

$$\sqrt{500} = 22.360680, \ \sqrt{510} = 22.583100,$$
$$\sqrt{520} = 22.803509, \ \sqrt{530} = 23.021729.$$

27. Use a suitable central difference formula of either Stirling's or Bessel's to find the values of $f(x)$ from the following tabulated function at $x = 1.35$ and at $x = 1.42$.

x	:	1.0	1.2	1.4	1.6	1.8
$f(x)$	:	1.17520	1.50946	1.90430	2.37557	2.94217

28. From Bessel's formula, derive the following formula for midway interpolation

$$y_{1/2} = \frac{1}{2}(y_0 + y_1) - \frac{1}{16}(\Delta^2 y_{-1} + \Delta^2 y_0) + \frac{3}{256}(\Delta^4 y_{-2} + \Delta^4 y_{-1}) - \cdots$$

    Also deduce this formula from Everett's formula.

29. The function $\log E$ where $E = \int_0^{\pi/2} \sqrt{1 - \sin^2 \alpha \sin^2 \theta}\, d\theta$ is tabulated below:

α^0	:	0	5	10	15	20
$\log E$	:	0.196120	0.195293	0.192815	0.188690	0.182928

    Compute $\log 12^0$ by (a) Bessel's formula and (b) Stirling's formula and compare the results.

30. The value of the elliptic integral

$$E(\alpha) = \int_0^{\pi/2} (1 - \alpha \sin^2 \theta)^{-1/2}\, d\theta$$

    for certain equidistance values of α are given below. Use Everett's or Bessel's formula to determine $E(0.25)$.

α	0.20	0.22	0.24	0.26	0.28	0.30
$E(\alpha)$	1.659624	1.669850	1.680373	1.691208	1.702374	1.713889

31. Using Everett's formula, evaluate $f(20)$ from the following table.

x	14	18	22	26
$f(x)$	2877	3162	3566	3990

32. Using Aitken's method evaluate y when $x = 2$ from the following table.

x	1	3	4	6
y	-3	9	30	132

33. Use the Aitken's procedure to determine the value of $f(0.2)$ as accurately as possible from the following table.

x	0.17520	0.25386	0.33565	0.42078	0.50946
$f(x)$	0.84147	0.86742	0.89121	0.91276	0.93204

34. Show that the first order divided difference of a linear polynomial is independent of the arguments.

35. Show that the second order divided difference of a quadratic polynomial is constant.

36. If $f'(x)$ is continuous for $x_0 \le x \le x_1$, show that $f[x_0, x_1] = f'(\xi)$ where $x_0 \le \xi \le x_1$ and hence show that

$$f[x_0, x_0] \equiv \lim_{x_1 \to x_0} f[x_0, x_1] = f'(x_0).$$

37. For the equidistant values x_0, x_1, x_2, x_3 i.e., $x_i = x_0 + ih$, establish the following relations

$$\begin{aligned} f[x_0, x_1] &= \frac{1}{h}[f(x_1) - f(x_0)], \\ f[x_0, x_1, x_2] &= \frac{1}{2!h^2}[f(x_2) - 2f(x_1) + f(x_0)], \\ \text{and } f[x_0, x_1, x_2, x_3] &= \frac{1}{3!h^3}[f(x_3) - 3f(x_2) + 3f(x_1) - f(x_0)]. \end{aligned}$$

38. If $f(x) = \dfrac{ax + b}{cx + d}$ obtain expressions for $f[p, q], f[p, p, q]$ and $f[p, p, q, q]$.

39. If $f(x) = x^4$ obtain expressions for $f[a, b, c]$, $f[a, a, b]$ and $f[a, a, a]$ where $a \neq b \neq c$.

40. If $f(x) = 1/(a - x)$, show that

$$f[x_0, x_1, \ldots, x_n] = \frac{1}{(a - x_0)(a - x_1) \cdots (a - x_n)}.$$

41. Use Newton's divided difference interpolation to find the interpolation polynomial for the function $y = f(x)$ given by the table:

x :	-1	1	4	6
y :	1	-3	21	127

42. Use Newton's divided difference interpolation to find the interpolating polynomial for the function $y = f(x)$ given by

x :	-1	1	4	6
$f(x)$:	5	2	26	132

43. Using the given table of value of Bessel's function $y = J_0(x)$, find the root of the equation $J_0(x) = 0$ lying in $(2.4, 2.6)$ correct up to three significant digits.

x :	2.4	2.5	2.6
y :	0.0025	-0.0484	-0.0968

44. Given below is a table of values of the probability integral

$$y = \frac{2}{\pi} \int_0^x e^{-x^2}\, dx.$$

Determine the value of x for which the value of y is 0.49.

x :	0.45	0.46	0.47	0.48	0.49
y :	0.4754818	0.4846555	0.497452	0.5027498	0.5116683

45. Find x for which $\cosh x = 1.285$, by using the inverse interpolation technique of successive approximation of Newton's forward difference interpolation formula, given the table.

x :	0.738	0.739	0.740	0.741	0.752
$\cosh x$:	1.2849085	1.2857159	1.2865247	1.2873348	1.2881461

46. Use the technique of inverse interpolation to find x for which $\sinh x = 5.5$ from the following table.

x	:	2.2	2.4	2.6	2.8
$\sinh x$	:	4.457	5.466	6.095	8.198

47. Given the following table of $f(x)$ between $x = 1.1$ and $x = 1.5$, find the zero of $f(x)$.

x	:	1.1	1.2	1.3	1.4	1.5
$f(x)$	:	1.769	1.472	1.103	−1.344	−1.875

48. Use the technique of inverse interpolation to find a real root of the equation $x^3 - 2x - 4 = 0$.

49. Using Hermite's interpolation formula, estimate the value of $\log 3.2$ from the following table

x	:	3.0	3.5	4.0
$y = \log x$	:	1.09861	1.25276	1.38629
$y' = \frac{1}{x}$	:	0.33333	0.28571	0.25000

50. Find the Hermite polynomial of the third degree approximating the function $y(x)$ such that

$$y(x_0) = 1, y(x_1) = 0 \text{ and } y'(x_0) = y'(x_1) = 0.$$

51. The following values of x and y are calculated from the relation $y = x^3 + 10$

x	:	1	2	3	4	5
y	:	11	18	37	74	135

Determine the cubic spline $p(x)$ for the interval [2, 3] given that
(a) $p'(1) = y'(1)$ and $p'(5) = y'(5)$, (b) $p''(1) = y''(1)$ and $p''(5) = y''(5)$.

52. Fit a cubic spline to the function defined by the set of points given in the following table.

x	:	0.10	0.15	0.20	0.25	0.30
$y = e^x$	:	1.1052	1.1618	1.2214	1.2840	1.3499

Use the end conditions

(a) $M_0 = M_N = 0$
(b) $p'(0.10) = y'(0.10)$ and $p'(0.30) = y'(0.30)$ and
(c) $p''(0.10) = y''(0.10)$ and $p''(0.30) = y''(0.30)$.

Interpolate in each case for $x = 0.12$ and state which of the end conditions gives the best fit.

53. The distance d_i that a car has travelled at time t_i is given below.

time t_i	:	0	2	4	6	8
distance d_i	:	0	40	160	300	480

Use the values $p'(0)$ and $p'(8) = 98$, and find the clamped spline for the points.

54. Fit a cubic spline for the points (0,1), (1,0), (2,0), (3,1), (4,2), (5,2) and (6,1) and $p'(0) = -0.6, p'(6) = -1.8$ and $p''(0) = 1$ and $p''(6) = -1$.

55. A function $f(x)$ is defined as follows:

$$f(x) = \begin{cases} 1 + x, & 0 \le x \le 3 \\ 1 + x + (x-3)^2, & 3 \le x \le 4. \end{cases}$$

Show that $f(x)$ is a cubic spline in [0, 4].

56. Tabulate the values of the function

$$f(x, y) = e^x \sin y + y + 1$$

for
$x = 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5$
and $y = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6$.

Hence find the value of $f(1.6, 0.33)$ by two-dimensional interpolation.

57. Using the following data obtain the Lagrange and Newton's bivariate interpolating polynomials

x	:	0	0	0	1	1	1	2	2	2
y	:	0	1	2	0	1	2	0	1	2
$f(x,y)$	:	1	3	7	3	6	11	7	11	17

Chapter 4

Solution of Algebraic and Transcendental Equations

Determination of roots of algebraic and transcendental equations is a very important problem in science and engineering.

A function $f(x)$ is called **algebraic** if, to get the values of the function starting from the given value of x, we have to perform arithmetic operations between some real numbers and rational power of x. On the other hand, **transcendental** functions include all non-algebraic functions, i.e., an exponential function e^x, a^x, a logarithmic function $\log x$, trigonometric functions $\sin x, \cos x, \tan x, \cot x$, etc., inverse trigonometric functions $\sin^{-1} x, \cos^{-1} x$, etc. and others.

An equation $f(x) = 0$ is called algebraic or transcendental according as $f(x)$ is algebraic or transcendental.

The equations $x^3 + 7x + 3 = 0, x^5 - 7x^2 + 3 = 0$ are algebraic equations where as $e^x + \sin x = 0, 5 \log x + 3x - 2 = 0$ are the transcendental equations.

The definition of roots of an equation can be given in two different ways:
Algebraically, a number c is called a root of the equation $f(x) = 0$ iff $f(c) = 0$ and geometrically, the real roots of the equation $f(x) = 0$ are the values of x where the graph of $y = f(x)$ meets the x-axis.

Development of numerical methods to solve algebraic or transcendental equations are very much essential because the analytic method fails to solve the polynomial equations of degree greater than four.

Most of the numerical methods, used to solve an equation are based on iterative techniques. Different numerical methods are available to solve the equation $f(x) = 0$. But, each method has some advantages and disadvantages over another method. Generally, the following aspects are considered to compare the methods:

convergence or divergence, rate of convergence, applicability of the method, amount of pre-calculation needed before application of the method, etc.

The process of finding the approximate values of the roots of an equation can be divided into two stages: (i) location of the roots, and (ii) computation of the values of the roots with the specified degree of accuracy.

4.1 Location of Roots

An interval $[a, b]$ is said to be the **location of a real root** c if $f(c) = 0$ for $a < c < b$.

Mainly, two methods are used to locate the real roots of an equation, one is **graphical method** and other is an analytic method known as **method of tabulation**.

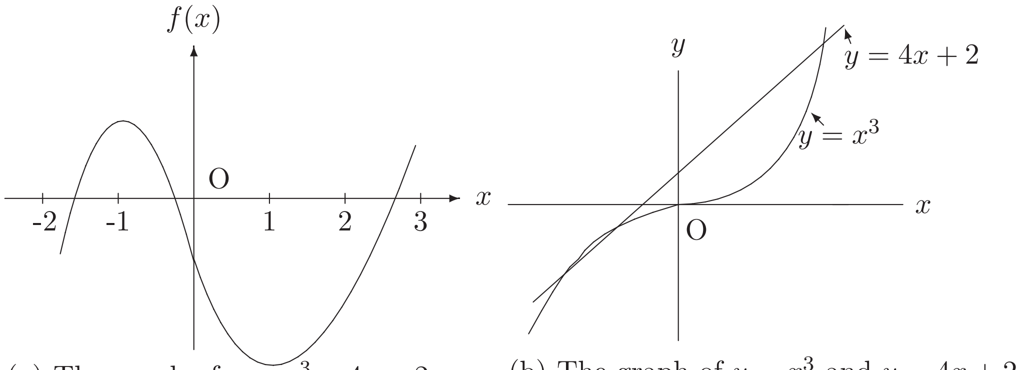
4.1.1 Graphical method

First method:
In this method, the graph of $y = f(x)$ is drawn in a rectangular co-ordinates system. It is obvious that the abscissas of the points where the graph intersects the x-axis are the roots of the equation $f(x) = 0$. But, practically, it is most difficult to determine the exact value of x where the graph intersects the x-axis. For example, if $x = 1.27831$ is a root of an equation $f(x) = 0$ then we can not determine 1.2783 (four digits after decimal point) from the graph. We can measure the value of x up to one or two decimal places. But, the approximate value of the root can be determined using this method.

Second method:
Some times, the approximate roots of $f(x) = 0$ can be determined by dividing all the terms of the equation into groups, one of them is written on the left-hand side of the equation and the other on the right hand side, i.e., the equation is represented as $g(x) = h(x)$. Then the graph of two functions $y = g(x)$ and $y = h(x)$, are drawn. The abscissas of the points of intersection of these graphs are the roots of the equation.

Example 4.1.1 Use graphical method to locate the roots of the equation $x^3 - 4x - 2 = 0$.

Solution. First method:
The graph of the function $y = x^3 - 4x - 2$ is drawn in Figure 4.1(a). The curve cuts the x-axis at three points and, consequently, the equation has three real roots. From figure, it is observed that the roots belong to the intervals $[-2, -1]$, $[-1, 0]$ and $[2, 3]$.
Second method:
The given equation can be written as $x^3 = 4x + 2$. The graph of the functions $y = x^3$ and $y = 4x + 2$ are drawn (Figure 4.1(b)). The abscissas of the points of intersection of the graphs of these functions are roots of the equations. The intervals of the roots are $[-2, -1]$, $[-1, 0]$ and $[2, 3]$.



(a) The graph of $y = x^3 - 4x - 2$. (b) The graph of $y = x^3$ and $y = 4x + 2$.
Figure 4.1: Illustration of location of roots.

The graphical method to locate the roots is not very useful, because, the drawing of the function $y = f(x)$ is itself a complicated problem. But, it makes possible to roughly determine the intervals to locate the roots. Then an analytic method is used to locate the root.

4.1.2 Method of tabulation

This method depends on the continuity of the function $f(x)$. Before applying the tabulation method, following result should be noted.

Theorem 4.1 *If $f(x)$ is continuous in the interval (a, b) and if $f(a)$ and $f(b)$ have the opposite signs, then at least one real root of the equation $f(x) = 0$ lies within the interval (a, b).*

If $f(a)$ and $f(b)$ have same signs then $f(x) = 0$ has no real roots or has an even number of real roots.

If the curve $y = f(x)$ touches the x-axis at some point, say, $x = c$ then c is a root of $f(x) = 0$ though $f(a)$ and $f(b)$, $a < c < b$ may have same sign. For example, $f(x) = (x - 2)^2$ touches the x-axis at $x = 2$, also $f(1.5) > 0$ and $f(2.5) > 0$, but, $x = 2$ is the root of the equation $f(x) = (x - 2)^2 = 0$.

A trial method for tabulation is as follows:
Form a table of signs of $f(x)$ setting $x = 0, \pm 1, \pm 2, \ldots$. If the sign $f(x)$ changes it signs for two consecutive values of x then at least one root lies between these two values, i.e., if $f(a)$ and $f(b)$ have opposite signs then a root lies between a and b.

Example 4.1.2 Find the location of roots of the equation $8x^3 - 20x^2 - 2x + 5 = 0$ by tabulation method.

Solution. We form a table of sign of $f(x)$, taken $x = 0, -1, 1, -2, 2, \dots$ as follows:

x	0	-1	1	-2	2	3
Sign of $f(x)$	$+$	$-$	$-$	$-$	$-$	$+$

The equation has three real roots as its degree is 3. The location of the roots of the given equation are $(-1, 0), (0, 1)$ and $(2, 3)$.

A systematic process for tabulation
The following sequence of steps to be performed to locate the roots of an equation $f(x) = 0$ by tabulation method:

1. find the first derivative $f'(x)$,
2. prepare a table of signs of the function $f(x)$ by setting x equal to
   (a) the roots of $f'(x) = 0$ or the values close to them,
   (b) the boundary values (preceding from the domain of permissible values of the variable),
3. determine the intervals at the endpoints of which the function assumes values of opposite signs. These intervals contain one and only one root each in its interior.

Example 4.1.3 Find the number of real roots of the equation $3^x - 3x - 2 = 0$ and locate them.

Solution. Let $f(x) = 3^x - 3x - 2$. The domain of definition of the function $f(x)$ is $(-\infty, \infty)$.
Now, $f'(x) = 3^x \log 3 - 3$.
The roots of $f'(x) = 0$ is given by $3^x \log 3 - 3 = 0$

or, $3^x = \dfrac{3}{\log 3}$ or, $x = \dfrac{\log 3 - \log\log 3}{\log 3} = 0.914$.
A table of signs of $f(x)$ is then form by setting x equal to
(a) the values close of the roots of $f'(x) = 0$, i.e., $x = 0, x = 1$ and
(b) boundary values of domain, i.e., $x = \pm\infty$.

x	$-\infty$	0	1	∞
Sign of $f(x)$	$+$	$-$	$-$	$+$

The equation $3^x - 3x - 2 = 0$ has two real roots since the function twice changes sign, among them one is negative root and other is greater than 1.

A new table with small intervals of the location of the root is constructed in the following.

x	0	-1	1	2
Sign of $f(x)$	$-$	$+$	$-$	$+$

The roots of the given equation are in $(-1, 0)$ [as $f(0).f(-1) < 0$] and $(1, 2)$.

This section is devoted to locate the roots which is the first stage of solution of algebraic and transcendental equations.

The second stage is the computation of roots with the specified degree of accuracy. In the following sections some methods are discussed to determine the roots of an algebraic or a transcendental equation. Before presenting the solution methods we define the order of convergence of a sequence of numbers in the following.

Order of Convergence

Assume that the sequence $\{x_n\}$ of numbers converges to ξ and let $\varepsilon_n = \xi - x_n$ for $n \geq 0$. If two positive constants $A \neq 0$ and $p > 0$ exist and

$$\lim_{n\to\infty} \frac{\varepsilon_{n+1}}{\varepsilon_n^p} = A \tag{4.1}$$

then the sequence is said to converge to ξ with order of convergence p. The number A is called the asymptotic error constant.

If $p = 1$, the order of convergence of $\{x_n\}$ is called **linear** and if $p = 2$, the order of convergence is called **quadratic**, etc.

In the next section, one of the bracketing method called bisection method is introduced.

4.2 Bisection Method

Let ξ be a root of the equation $f(x) = 0$ lies in the interval $[a, b]$, i.e., $f(a).f(b) < 0$, and $(b-a)$ is not sufficiently small. The interval $[a, b]$ is divided into two equal intervals $[a, c]$ and $[c, b]$, each of length $\dfrac{b-a}{2}$, and $c = \dfrac{a+b}{2}$ (Figure 4.2). If $f(c) = 0$, then c is an exact root.

Now, if $f(c) \neq 0$, then the root lies either in the interval $[a, c]$ or in the interval $[c, b]$. If $f(a).f(c) < 0$ then the interval $[a, c]$ is taken as new interval, otherwise $[c, b]$ is taken as the next interval. Let the new interval be $[a_1, b_1]$ and use the same process to select the next new interval. In the next step, let the new interval be $[a_2, b_2]$. The process of bisection is continued until either the midpoint of the interval is a root, or the length $(b_n - a_n)$ of the interval $[a_n, b_n]$ (at nth step) is sufficiently small. The number a_n and

b_n are the approximate roots of the equation $f(x) = 0$. Finally, $x_n = \dfrac{a_n + b_n}{2}$ is taken as the approximate value of the root ξ.

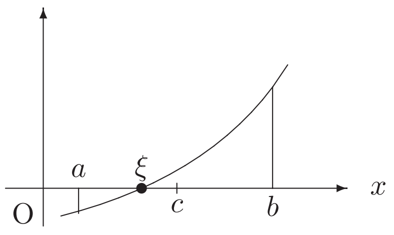


Figure 4.2: Illustration of bisection method.

It may be noted that when the reduced interval be $[a_1, b_1]$ then the length of the interval is $(b - a)/2$, when the interval be $[a_2, b_2]$ then the length is $(b - a)/2^2$. At the nth step the length of the interval being $(b-a)/2^n$. In the final step, when $\xi = \dfrac{a_n + b_n}{2}$ is chosen as a root then the length of the interval being $(b-a)/2^{n+1}$ and hence the error does not exceed $(b - a)/2^{n+1}$.

Thus, if ε be the error at the nth step then the lower bound of n is obtained from the following relation

$$\frac{|b - a|}{2^n} \le \varepsilon. \tag{4.2}$$

The lower bound of n is obtained by rewriting this inequation as

$$n \ge \frac{\log(|b - a|) - \log \varepsilon}{\log 2}. \tag{4.3}$$

Hence the minimum number of iterations required to achieve the accuracy ε is

$$\frac{\log_e \left(\frac{|b-a|}{\varepsilon}\right)}{\log 2}. \tag{4.4}$$

For example, if the length of the interval is $|b-a| = 1$ and $\varepsilon = 0.0001$, then n is given by $n \ge 14$.

The minimum number of iterations required to achieved the accuracy ε for $|b-a| = 1$ are shown in Table 4.1.

Theorem 4.2 *Assume that $f(x)$ is a continuous function on $[a, b]$ and that there exists a number $\xi \in [a, b]$ such that $f(\xi) = 0$. If $f(a)$ and $f(b)$ have opposite signs, and $\{x_n\}$ represents the sequence of midpoints generated by the bisection method, then*

$$|\xi - x_n| \le \frac{b - a}{2^{n+1}} \quad \textit{for } n = 0, 1, 2, \ldots \tag{4.5}$$

Table 4.1: Number of iterations for given ε.

ε	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
n	7	10	14	17	20	24

and therefore the sequence $\{x_n\}$ converges to the root ξ i.e.,

$$\lim_{n\to\infty} x_n = \xi.$$

Proof. The root ξ and the midpoint x_n both lie in the interval $[a_n, b_n]$, the distance between x_n and ξ cannot be greater than half the width of the interval $[a_n, b_n]$. Thus

$$|\xi - x_n| \le \frac{|b_n - a_n|}{2} \quad \text{for all } n. \tag{4.6}$$

From the bisection method, it is observed that the successive interval widths form the following pattern.

$$\begin{aligned}
|b_1 - a_1| &= \frac{|b_0 - a_0|}{2^1}, \quad \text{where } b_0 = b \text{ and } a_0 = a,\\
|b_2 - a_2| &= \frac{|b_1 - a_1|}{2} = \frac{|b_0 - a_0|}{2^2},\\
|b_3 - a_3| &= \frac{|b_2 - a_2|}{2} = \frac{|b_0 - a_0|}{2^3}.
\end{aligned}$$

In this way, $|b_n - a_n| = \dfrac{|b_0 - a_0|}{2^n}$.
Hence

$$|\xi - x_n| \le \frac{|b_0 - a_0|}{2^{n+1}} \quad \text{[using (4.6)]}.$$

Now, the limit gives

$$|\xi - x_n| \to 0 \text{ as } n \to \infty \quad \text{i.e., } \lim_{n\to\infty} x_n = \xi.$$

□

Note 4.2.1 If the function $f(x)$ is continuous on $[a, b]$ then the bisection method is applicable. This is justified in Figure 4.3. For the function $f(x)$ of the graph of Figure 4.3, $f(a) \cdot f(b) < 0$, but the equation $f(x) = 0$ has no root between a and b as the function is not continuous at $x = c$.

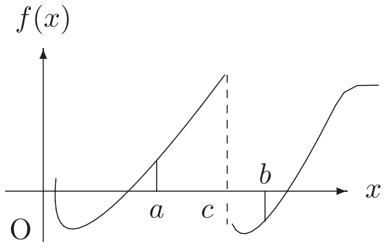


Figure 4.3: The function has no root between a and b, though $f(a) \cdot f(b) < 0$.

Note 4.2.2 This method is very slow, but it is very simple and will converge surely to the exact root. So the method is applicable for any function only if the function is continuous within the interval $[a, b]$, where the root lies.

In this method derivative of the function $f(x)$ and pre-manipulation of function are not required.

Note 4.2.3 This method is also called bracketing method since the method successively reduces the two endpoints (brackets) of the interval containing the real root.

Example 4.2.1 Find a root of the equation $x^2 + x - 7 = 0$ by bisection method, correct up to two decimal places.

Solution. Let $f(x) = x^2 + x - 7$.
$f(2) = -1 < 0$ and $f(3) = 5 > 0$. So, a root lies between 2 and 3.

	Left end point	Right end point	Midpoint	
n	a_n	b_n	x_{n+1}	$f(x_{n+1})$
0	2	3	2.5	1.750
1	2	2.5	2.250	0.313
2	2	2.250	2.125	-0.359
3	2.125	2.250	2.188	-0.027
4	2.188	2.250	2.219	0.143
5	2.188	2.219	2.204	0.062
6	2.188	2.204	2.196	0.018
7	2.188	2.196	2.192	-0.003
8	2.192	2.196	2.194	0.008
9	2.192	2.194	2.193	0.002
10	2.192	2.193	2.193	0.002

Therefore, the root is 2.19 correct up to two decimal places.

Algorithm 4.1 (Bisection method). This algorithm finds a real root of the equation $f(x) = 0$ which lies in $[a, b]$ by bisection method.

```
Algorithm Bisection
Input function $f(x)$;
// Assume that $f(x)$ is continuous within $[a, b]$ and a root lies on $[a, b]$.//
Read $\varepsilon$;          //tolerance for width of the interval//
Read $a, b$;          //input of the interval//
Compute $fa = f(a)$; $fb = f(b)$;     //compute the function values//
if $sign(fa) = sign(fb)$ then
//$sign(fa)$ gives the sign of the value of $fa$.//
    Print '$f(a) \cdot f(b) > 0$, so there is no guarantee for a root within $[a, b]$';
    Stop;
endif;
do
    Compute $c = (a + b)/2$;
    Compute $fc = f(c)$;
    if $fc = 0$ or $|fc| < \varepsilon$ then
        $a = c$ and $b = c$;
    else if $sign(fb) = sign(fc)$ then
        $b = c$; $fb = fc$;
    else
        $a = c$; $fa = fc$;
    endif;
while ($|b - a| > \varepsilon$);
Print 'the desired root is' $c$;
end Bisection
```

Program 4.1

```
/* Program Bisection
   Program to find a root of the equation x*x*x-2x-1=0 by
   bisection method.
   Assume that a root lies between a and b. */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#define f(x) x*x*x-2*x-1 /* definition of the function f(x) */
void main()
{
 float a,b,fa,fb,c,fc;
 float eps=1e-5; /* error tolerance */
 printf("\nEnter the value of a and b ");
 scanf("%f %f",&a,&b);
 fa=f(a);  fb=f(b);
 if(fa*fb>0)
    {
     printf("There is no guarantee for a root within [a,b]");
     exit(0);
    }
 do
 {
    c=(a+b)/2.;
    fc=f(c);
    if((fc==0) || (fabs(fc)<eps))
       {
         a=c;b=c;
       }
    else if(fb*fc>0)
       {
         b=c; fb=fc;
       }
    else
       {
         a=c; fa=fc;
       }
 }while(fabs(b-a)>eps);
 printf("\nThe desired root is %8.5f ",c);
} /* main */
```

A sample of input/output:

```
Enter the value of a and b 0 2
The desired root is  1.61803
```

Another popular method is the method of false position or the regula falsi method. This is also a bracketing method. This method was developed because the bisection method converges at a fairly slow speed. In general, the regula falsi method is faster than bisection method.

4.3 Regula-Falsi Method (Method of False Position)

The Regula-Falsi method is one of the most widely used methods of solving algebraic and transcendental equations. This method is also known as ‘method of false position’, 'method of chords' and 'the method of linear interpolation'.

Let a root of the equation $f(x) = 0$ be lies in the interval $[a, b]$, i.e., $f(a) \cdot f(b) < 0$.

The idea of this method is that on a sufficiently small interval $[a, b]$ the arc of the curve $y = f(x)$ is replaced by the chord joining the points $(a, f(a))$ and $(b, f(b))$. The abscissa of the point of intersection of the chord and the x-axis is taken as the approximate value of the root.

Let $x_0 = a$ and $x_1 = b$. The equation of the chord joining the points $(x_0, f(x_0))$ and $(x_1, f(x_1))$ is

$$\frac{y - f(x_0)}{f(x_0) - f(x_1)} = \frac{x - x_0}{x_0 - x_1}. \tag{4.7}$$

To find the point of intersection, set $y = 0$ in (4.7) and let $(x_2, 0)$ be such point. Thus,

$$x_2 = x_0 - \frac{f(x_0)}{f(x_1) - f(x_0)}(x_1 - x_0). \tag{4.8}$$

This is the second approximation of the root. Now, if $f(x_2)$ and $f(x_0)$ are of opposite signs then the root lies between x_0 and x_2 and then we replace x_1 by x_2 in (4.8). The next approximation is obtained as

$$x_3 = x_0 - \frac{f(x_0)}{f(x_2) - f(x_0)}(x_2 - x_0).$$

If $f(x_2)$ and $f(x_1)$ are of opposite signs then the root lies between x_1 and x_2 and the new approximation x_3 is obtain as

$$x_3 = x_2 - \frac{f(x_2)}{f(x_1) - f(x_2)}(x_1 - x_2).$$

The procedure is repeated till the root is obtained to the desired accuracy.

If the nth approximate root (x_n) lies between a_n and b_n then the next approximate root is thus obtained as

$$x_{n+1} = a_n - \frac{f(a_n)}{f(b_n) - f(a_n)}(b_n - a_n). \tag{4.9}$$

The illustration of the method is shown in Figure 4.4.

This method is also very slow and not suitable for hand calculation. The advantage of this method is that it is very simple and the sequence $\{x_n\}$ is sure to converge. The another advantage of this method is that it does not require the evaluation of derivatives and pre-calculation.

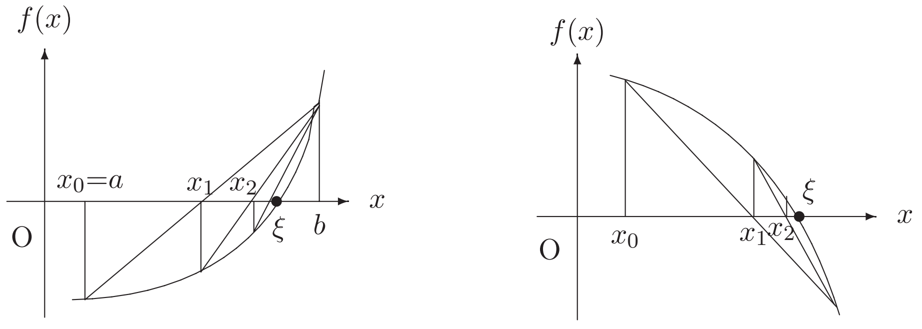


Figure 4.4: Illustration of Regula-falsi method.

To estimate the error of approximation, the following formula may be used

$$|\xi - x_n| < |x_n - x_{n-1}| \tag{4.10}$$

where ξ is an exact root and x_{n-1} and x_n are its approximations obtained at the $(n-1)$th and nth iterations. This relation can be used when

$$M \leq 2m, \text{ where } M = \max|f'(x)| \text{ and } m = \min|f'(x)| \text{ in } [a,b]. \tag{4.11}$$

Example 4.3.1 Find a root of the equation $x^3 + 2x - 2 = 0$ using Regula-Falsi method, correct up to three decimal places.

Solution. Let $f(x) = x^3 + 2x - 2$. $f(0) = -2 < 0$ and $f(1) = 1 > 0$. Thus, one root lies between 0 and 1. The calculations are shown in the following table.

	left end	right end				
n	point a_n	point b_n	$f(a_n)$	$f(b_n)$	x_{n+1}	$f(x_{n+1})$
0	0.0000	1.0	–2.0000	1.0	0.6700	–0.3600
1	0.6700	1.0	–0.3600	1.0	0.7570	–0.0520
2	0.7570	1.0	–0.0520	1.0	0.7690	–0.0072
3	0.7690	1.0	–0.0072	1.0	0.7707	–0.0010
4	0.7707	1.0	–0.0010	1.0	0.7709	–0.0001

Therefore, a root of the equation is 0.771 correct up to three decimal places.

Algorithm 4.2 (Regula-Falsi). This algorithm finds a root of the equation $f(x) = 0$ which lies in $[x_0, x_1]$, by Regula-Falsi method.

Algorithm Regula-Falsi
Input function $f(x)$;

```
Read $x_0, x_1, \varepsilon$; //interval for the root and error tolerance//
Compute $fx_0 = f(x_0)$; $fx_1 = f(x_1)$; // Compute the function values at $x_0$ and $x_1$//
do
    Compute $x_2 = \frac{x_0 \cdot fx_1 - x_1 \cdot fx_0}{fx_1 - fx_0}$;
    Compute $fx_2 = f(x_2)$;
    if $|fx_2| \le \varepsilon$ then
        Print 'The root is', $x_2$;
        Stop;
    endif;
    if $sign(fx_2) \neq sign(fx_0)$ then
        Set $x_1 = x_2$; $fx_1 = fx_2$;
    else
        Set $x_0 = x_2$; $fx_0 = fx_2$;
    endif;
while ($|fx2| > \varepsilon$);
end Regula-Falsi
```

Program 4.2

```
/* Program Regula-Falsi
   Program to find a root of the equation x*x-2x-3=0 by regula
   falsi method. Assumed that a root lies between x0 and x1. */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#define f(x) x*x-2*x-3 /* definition of the function f(x) */
void main()
{
 float x0,x1,x2,fx0,fx1,fx2;
 float eps=1e-5; /* error tolerance */
 printf("\nEnter the value of x0 and x1 ");
 scanf("%f %f",&x0,&x1);
 fx0=f(x0);  fx1=f(x1);

 if(fx0*fx1>0)
  {
   printf("There is no guarantee for a root within
           [%6.3f,%6.3f]",x0,x1);
   exit(0);
  }
 do
 {
    x2=(x0*fx1-x1*fx0)/(fx1-fx0);
    fx2=f(x2);
    if(fabs(fx2)<eps)
       {
         printf("The root is %8.5f ",x2);
         exit(0);
       }
    if(fx2*fx0<0)
       {
         x1=x2; fx1=fx2;
       }
    else
       {
         x0=x2; fx0=fx2;
       }
 }while(fabs(fx2)>eps);
} /* main */
```

A sample of input/output:

```
Enter the value of x0 and x1 0 3
The root is  3.00000
```

4.4 Iteration Method or Fixed Point Iteration

The **iteration method** or the **method of successive approximations**, is one of the most important methods in numerical mathematics. This method is also known as **fixed-point iteration**.

Let $f(x)$ be a function continuous on the interval $[a, b]$ and the equation $f(x) = 0$ has at least one root on $[a, b]$. The equation $f(x) = 0$ can be written in the form

$$x = \phi(x). \tag{4.12}$$

Suppose $x_0 \in [a, b]$ be an initial guess to the desired root ξ. Then $\phi(x_0)$ is evaluated and this value is denoted by x_1. It is the first approximation of the root ξ. Again, x_1 is substituted for x to the right side of (4.12) and obtained a new value $x_2 = \phi(x_1)$. This process is continued to generate the sequence of numbers $x_0, x_1, x_2, \ldots, x_n, \ldots$, those are defined by the following relation:

$$x_{n+1} = \phi(x_n), \qquad n = 0, 1, 2, \ldots \tag{4.13}$$

This successive iterations are repeated till the approximate numbers x_n's converges to the root with desired accuracy, i.e., $|x_{n+1} - x_n| < \varepsilon$, where ε is a sufficiently small number. The function $\phi(x)$ is called the iteration function.

Note 4.4.1 There is no guarantee that this sequence $x_0, x_1, x_2, \ldots$ will converge. The function $f(x) = 0$ can be written as $x = \phi(x)$ in many different ways. This is very significant since the form of the function $\phi(x)$ is very important both for the convergence and for its rate.

For example, the equation $x^3 + x^2 - 1 = 0$ has a root lies between 0 and 1. This equation can be rewritten in the following ways:

$$x = \frac{1-x^2}{x^2}; x = (1-x^2)^{1/3}; x = \sqrt{\frac{1-x^2}{x}}; x = \sqrt{1-x^3}; x = \frac{1}{\sqrt{1+x}}, \text{ etc.}$$

The following theorem gives the sufficient condition for convergence of the iteration process.

Theorem 4.3 *Let ξ be a root of the equation $f(x) = 0$ and it can be written as $x = \phi(x)$ and further that*

1. *the function $\phi(x)$ is defined and differentiable on the interval $[a, b]$,*
2. *$\phi(x) \in [a, b]$ for all $x \in [a, b]$,*
3. *there is a number $l < 1$ such that*

$$|\phi'(x)| \le l < 1 \quad \text{for } x \in [a, b]. \tag{4.14}$$

Then the sequence $\{x_n\}$ given by (4.13) converges to the desired root ξ irrespective of the choice of the initial approximation $x_0 \in [a, b]$ and the root ξ is unique.

Proof. Since ξ is a root of the equation $x = \phi(x)$, therefore

$$\xi = \phi(\xi). \tag{4.15}$$

Also from (4.13),

$$x_{i+1} = \phi(x_i). \tag{4.16}$$

Subtracting (4.16) from (4.15),

$$\begin{aligned} \xi - x_{i+1} &= \phi(\xi) - \phi(x_i) \\ &= (\xi - x_i)\phi'(\xi_i) \text{ (by mean value theorem)} \\ &\quad \text{(where } \xi_i \text{ lies between } \xi \text{ and } x_i\text{)} \end{aligned}$$

For $i = 0, 1, 2, \ldots, n$,

$$\begin{aligned}
\xi - x_1 &= (\xi - x_0)\phi'(\xi_0), \ \ \xi_0 \text{ lies between } \xi \text{ and } x_0 \\
\xi - x_2 &= (\xi - x_1)\phi'(\xi_1), \ \ \xi_1 \text{ lies between } \xi \text{ and } x_1 \\
&\ \ \vdots \qquad\qquad\qquad\qquad \vdots \\
\xi - x_{n+1} &= (\xi - x_n)\phi'(\xi_n), \ \ \xi_n \text{ lies between } \xi \text{ and } x_n.
\end{aligned}$$

Multiplying all these equations we obtain the following result

$$\begin{aligned}
(\xi - x_{n+1}) &= (\xi - x_0)\phi'(\xi_0)\phi'(\xi_1)\cdots\phi'(\xi_n) \\
\text{or } |\xi - x_{n+1}| &= |\xi - x_0||\phi'(\xi_0)||\phi'(\xi_1)|\cdots|\phi'(\xi_n)| \\
&\le l^{n+1}|\xi - x_0|, \qquad (4.17) \\
&\quad \text{where } |\phi'(x)| \le l \text{ for all } x \in [a, b].
\end{aligned}$$

Now, if $l < 1$ then the right hand side of (4.17) tends to zero as $n \to \infty$.
Therefore, $\lim\limits_{n\to\infty} x_{n+1} = \xi$.
Hence the sequence $\{x_n\}$ converge to ξ if $|\phi'(x)| < 1$, for all $x \in [a, b]$.
Now to prove the uniqueness.
Let ξ_1 and ξ_2 be two roots of $x = \phi(x)$, i.e., $\xi_1 = \phi(\xi_1)$ and $\xi_2 = \phi(\xi_2)$. Then

$$|\xi_2 - \xi_1| = |\phi(\xi_2) - \phi(\xi_1)| = |\phi'(c)||\xi_1 - \xi_2|, \qquad (4.18)$$

where $c \in (\xi_1, \xi_2)$.

Equation (4.18) reduces to

$$|\xi_1 - \xi_2|(1 - |\phi'(c)|) = 0$$

and by condition (iii) $\xi_1 = \xi_2$, i.e., the two roots are not distinct, they are equal. □

4.4.1 Estimation of error

Let ξ be an exact root of the equation $x = \phi(x)$ and $x_{n+1} = \phi(x_n)$.
Therefore,

$$\begin{aligned}
|\xi - x_n| &= |\phi(\xi) - \phi(x_{n-1})| = |\xi - x_{n-1}|\,|\phi'(c)|, c \text{ lies between } x_{n-1} \text{ and } \xi \\
&\le l|\xi - x_{n-1}| \\
&= l|\xi - x_n + x_n - x_{n-1}| \le l|\xi - x_n| + l|x_n - x_{n-1}|.
\end{aligned}$$

After rearrangement, this relation becomes

$$|\xi - x_n| \le \frac{l}{1-l}|x_n - x_{n-1}| \le \frac{l^n}{1-l}|x_1 - x_0|. \qquad (4.19)$$

Let the maximum number of iterations needed to achieve the accuracy ε be $N(\varepsilon)$. Thus from (4.19)

$$\frac{l^N}{1-l}|x_1 - x_0| \le \varepsilon.$$

This gives

$$N(\varepsilon) \ge \frac{\log \frac{\varepsilon(1-l)}{|x_1-x_0|}}{\log l}. \tag{4.20}$$

For $l \le \frac{1}{2}$ the estimation of the error is given by the following simple form:

$$|\xi - x_n| \le |x_n - x_{n-1}|. \tag{4.21}$$

Order of convergence

The convergence of an iteration method depends on the suitable choice of the iteration function $\phi(x)$ and x_0, the initial guess.

Let x_n converges to the exact root ξ, so that $\xi = \phi(\xi)$.

Thus $x_{n+1} - \xi = \phi(x_n) - \phi(\xi)$.

Let $\varepsilon_{n+1} = x_{n+1} - \xi$. Note that $\phi'(x) \ne 0$. Then the above relation becomes

$$\begin{aligned}
\varepsilon_{n+1} &= \phi(\varepsilon_n + \xi) - \phi(\xi) \\
&= \varepsilon_n \phi'(\xi) + \frac{1}{2}\varepsilon_n^2 \phi'(\xi) + \cdots \\
&= \varepsilon_n \phi'(\xi) + O(\varepsilon_n^2) \\
\text{i.e.,} \qquad \varepsilon_{n+1} &\simeq \varepsilon_n \phi'(\xi).
\end{aligned}$$

Hence the order of convergence of iteration method is linear.

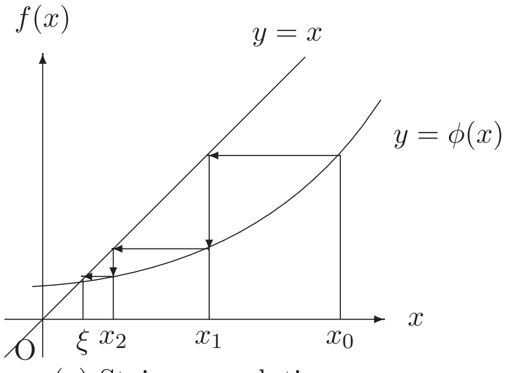
Geometric interpretation

Geometrically, the point of intersection of the line $y = x$ and the curve $y = \phi(x)$ is a root of the equation $f(x) = 0$. Depending on the value of $\phi'(\xi)$ the convergence and divergence cases are illustrated in Figures 4.5-4.6.

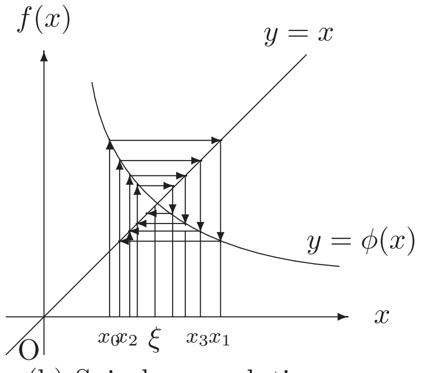
Merit and Demerit

The disadvantage of iteration method is that a pre-calculation is required to rewrite $f(x) = 0$ into $x = \phi(x)$ in such a way that $|\phi'(x)| < 1$. But, the main advantage of this method is that the operation carried out at each stage are of the same kind, and this makes easier to develop computer program.

This method is some times called a **linear iteration** due to its linear order of convergence.

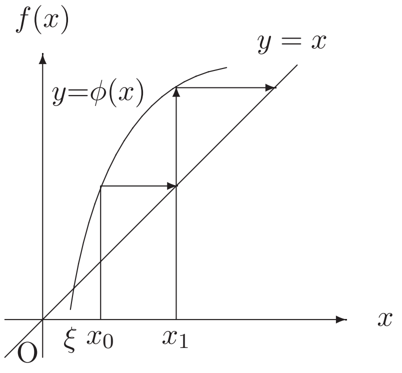


(a) Stair case solution, $0 < \phi'(\xi) < 1$

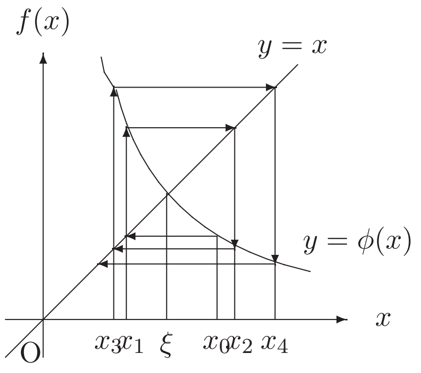


(b) Spiral case solution, $-1 < \phi'(\xi) < 0$.

Figure 4.5: Convergent for $|\phi'(\xi)| < 1$.



(a) Divergent for $\phi'(\xi) > 1$.



(b) Divergent for $\phi'(\xi) < -1$.

Figure 4.6: Divergent for $|\phi'(\xi)| > 1$.

Example 4.4.1 Consider the equation $5x^3 - 20x + 3 = 0$. Find the root lying on the interval [0,1] with an accuracy of 10^{-4}.

Solution. The given equation is written as $x = \frac{5x^3+3}{20} = \phi(x)$ (say).
Now, $\phi'(x) = \frac{15x^2}{20} = \frac{3x^2}{4} < 1$ on [0,1]. Let $x_0 = 0.5$. The calculations are shown in the following table.

n	x_n	$\phi(x_n) = x_{n+1}$
0	0.5	0.18125
1	0.18125	0.15149
2	0.15149	0.15087
3	0.15087	0.15086
4	0.15086	0.15086

At this stage the iteration process is terminated and $\xi = 0.1509$ is taken as the required root.

Example 4.4.2 Find a root of the equation

$$\cos x - xe^x = 0$$

correct up to three decimal places.

Solution. It is easy to see that one root of the given equation lies between 0 and 1. Let $x_0 = 0$. The equation can be written as $x = e^{-x}\cos x = \phi(x)$ (say).
The calculations are shown in the following table.

n	0	1	2	3	4	5	6
x_n	0.50000	0.53228	0.50602	0.52734	0.51000	0.52408	0.51263
x_{n+1}	0.53228	0.50602	0.52734	0.51000	0.52408	0.51263	0.52193

n	7	8	9	10	11	12	13
x_n	0.52193	0.51437	0.52051	0.51552	0.51958	0.51628	0.51896
x_{n+1}	0.51437	0.52051	0.51552	0.51958	0.51628	0.51896	0.51678

n	14	15	16	17	18	19	20
x_n	0.51678	0.51855	0.51711	0.51828	0.51733	0.51810	0.51748
x_{n+1}	0.51855	0.51711	0.51828	0.51733	0.51810	0.51748	0.51798

Therefore, the required root is 0.518 correct up to three decimal places.

Algorithm 4.3 (Fixed point iteration). This algorithm computes a root of the equation $f(x) = 0$ by rewriting the equation as $x = \phi(x)$, provided $|\phi'(x)| < 1$ in the interval $[a, b]$, by fixed point iteration method. $x_0 \in [a, b]$ be the initial guess and ε is the error tolerance.

```
Algorithm Iteration
Input function $\phi(x)$;
Read $x_0, \varepsilon$; //initial guess and error tolerance.//
Set $x_1 = x_0$;
do
    Set $x_0 = x_1$;
    Compute $x_1 = \phi(x_0)$;
while ($|x_1 - x_0| > \varepsilon$);
Print 'The root is', $x_1$;
end Iteration
```

Program 4.3

```
/* Program Fixed-Point Iteration
   Program to find a root of the equation x*x*x-3x+1=0
   by fixed point iteration method. phi(x) is obtained
   by rewrite f(x)=0 as x=phi(x), which is to be supplied.*/
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#define phi(x) (3*x-1)/(x*x)
/* definition of the function phi(x) and it to be
   changed accordingly */
void main()
{
 int k=0; /* counts number of iterations */
 float x1,x0; /* initial guess */
 float eps=1e-5; /* error tolerance */
 printf("\nEnter the initial guess x0 ");
 scanf("%f",&x0);
 x1=x0;
 do
 {
    k++;
    x0=x1;
    x1=phi(x0);
 }while(fabs(x1-x0)>eps);
 printf("One root is %8.5f obtained at %d th iteration ",x1,k);
} /* main */
```

A sample of input/output:

```
Enter the initial guess x0 1
One root is  1.53209 obtained at 37th iteration
```

4.5 Acceleration of Convergence: Aitken's Δ^2-Process

The rate of convergence of iteration method is linear. But, this slow rate can be accelerated by using Aitken's method.

The iteration scheme of this method is obtained from fixed point iteration method as

$$x_{n+1} = \phi(x_n) \text{ with } |\phi'(x)| < 1.$$

If ξ be the root of the equation $f(x) = 0$ then

$$\xi - x_{n+1} = \phi(\xi) - \phi(x_n) = \phi'(\xi_0)(\xi - x_n)$$

where ξ_0 lies between ξ and x_n.

Let x_{n-1}, x_n and x_{n+1} be three successive approximations to the root ξ.

Then

$$\xi - x_n = a(\xi - x_{n-1}) \text{ where } a = \phi'(\xi_0)$$
$$\xi - x_{n+1} = a(\xi - x_n).$$

Eliminating a from these equations, we find the relation

$$\frac{\xi - x_n}{\xi - x_{n+1}} = \frac{\xi - x_{n-1}}{\xi - x_n}$$

which gives

$$\xi = x_{n+1} - \frac{(x_{n+1} - x_n)^2}{x_{n+1} - 2x_n + x_{n-1}}. \tag{4.22}$$

Now, introduce the forward difference operator as

$$\Delta x_n = x_{n+1} - x_n, \qquad \Delta^2 x_{n-1} = x_{n+1} - 2x_n + x_{n-1}.$$

Then (4.22) is simplified as

$$\xi = x_{n+1} - \frac{(\Delta x_n)^2}{\Delta^2 x_{n-1}}, \tag{4.23}$$

which is known as Aitken's Δ^2-process.

Example 4.5.1 Find a root of the equation $\cos x - xe^x = 0$ using Aitken's Δ^2-process.

Solution. Let $x = e^{-x}\cos x = \phi(x)$ (say) and $x_0 = 0$.
$x_1 = \phi(x_0) = 0.54030$, $x_2 = \phi(x_1) = 0.49959$.
$\Delta x_1 = x_2 - x_1 = -0.04071$, $\Delta^2 x_0 = x_2 - 2x_1 + x_0 = -0.58101$.

Then $x_3 = x_2 - \dfrac{(\Delta x_1)^2}{\Delta^2 x_0} = 0.49959 - \dfrac{(-0.04071)^2}{-0.58101} = 0.50244$.
The results for $n = 1, 2, 3, 4$ are shown below.

n	x_{n-1}	x_n	x_{n+1}	Δx_n	$\Delta^2 x_{n-1}$	x_{n+2}
1	0.00000	0.54030	0.49959	–0.04071	–0.58101	0.50244
2	0.54030	0.49959	0.50244	0.00285	0.04356	0.50225
3	0.49958	0.50244	0.50225	–0.00019	–0.00304	0.50226
4	0.50244	0.50225	0.50226	0.00001	0.00021	0.50226

Therefore, a root is 0.5023 correct up to four decimal places.

4.6 Newton-Raphson Method or Method of Tangent

Let x_0 be an approximate root of the equation $f(x) = 0$. Suppose $x_1 = x_0 + h$ be the exact root of the equation, where h is the correction of the root (error). Then $f(x_1) = 0$.

Using Taylor's series, $f(x_1) = f(x_0 + h)$ is expanded in the following form

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

Neglecting the second and higher order derivatives the above equation reduces to

$$f(x_0) + hf'(x_0) = 0 \text{ or, } h = -\frac{f(x_0)}{f'(x_0)}.$$

Hence,

$$x_1 = x_0 + h = x_0 - \frac{f(x_0)}{f'(x_0)}. \tag{4.24}$$

To compute the value of h, the second and higher powers of h are neglected so the value of $h = -\dfrac{f(x_0)}{f'(x_0)}$ is not exact, it is an approximate value. So, x_1, obtained from (4.24) is not a root of the equation, but it is a better approximation of x than x_0.

In general,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{4.25}$$

This expression generates a sequence of approximate values $x_1, x_2, \ldots, x_n, \ldots$ each successive term of which is closer to the exact value of the root ξ than its predecessor. The method will terminate when $|x_{n+1} - x_n|$ becomes very small.

In Newton-Raphson method the arc of the curve $y = f(x)$ is replaced by a tangent to the curve, hence, this method is sometimes called the **method of tangents**.

Note 4.6.1 The Newton-Raphson method may also be used to find a complex root of an equation when the initial guess is taken as a complex number.

Geometrical interpretation

The geometrical interpretation of Newton-Raphson method is shown in Figure 4.7.

In this method, a tangent is drawn at $(x_0, f(x_0))$ to the curve $y = f(x)$. The tangent cuts the x-axis at $(x_1, 0)$. Again, a tangent is drawn at $(x_1, f(x_1))$ and this tangent cuts x-axis at $(x_2, 0)$. This process is continued until $x_n = \xi$ as $n \to \infty$.

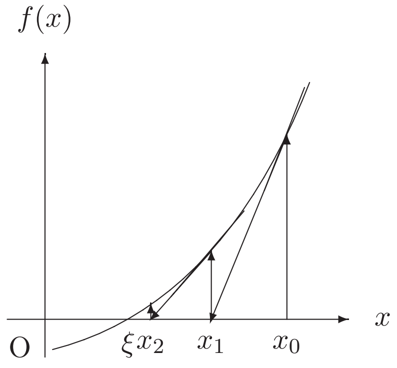


Figure 4.7: Geometrical interpretation of Newton-Raphson method.

The choice of initial guess of Newton-Raphson method is very important. If the initial guess is near the root then the method converges very fast. If it is not so near the root or if the starting point is wrong, then the method may lead to an endless cycle. This is illustrated in Figure 4.8. In this figure, the initial guess x_0 gives the fast convergence to the root, the initial guess y_0 leads to an endless cycle and the initial guess z_0 gives a divergent solution as $f'(z_0)$ is very small.

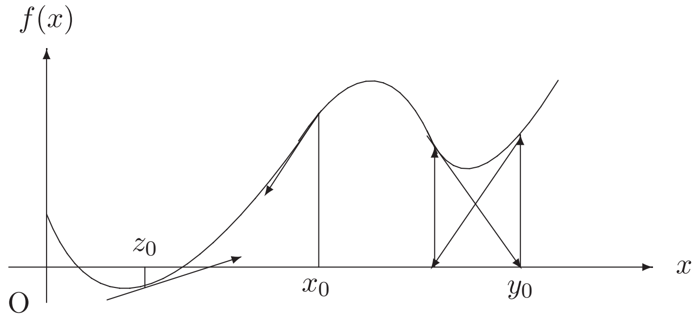


Figure 4.8: Illustration of the choice of initial guess in Newton-Raphson method.

Even if the initial guess is not close to the exact root, the method may diverges. To choose the initial guess the following rule may be followed:
The endpoint of the interval $[a, b]$ at which the sign of the function coincides with the sign of the second derivative must be taken as the initial guess. When $f(b) \cdot f''(x) > 0$, the initial guess is $x_0 = b$, and when $f(a) \cdot f''(x) > 0$ then $x_0 = a$ be the initial guess.

Three different cases - divergent, cyclic and oscillation of Newton-Raphson method are discussed in the following by examples.

Sometimes, if the initial guess x_0 is far away from the exact root then the sequence

$\{x_n\}$ may converges to some other root. This situation happens when the slope $f'(x_0)$ is small and the tangent to the curve $y = f(x)$ is nearly horizontal. For example, if $f(x) = \cos x$ and we try to find the root $\xi = \pi/2$ starting with $x_0 = 3$ then $x_0 = -4.01525, x_2 = -4.85266, \ldots$ and the sequence $\{x_n\}$ will converge to a different root $-4.71239 \simeq -3\pi/2$.

We consider another example which will produces a divergent sequence. Let

$$f(x) = xe^{-x} \qquad \text{and} \qquad x_0 = 2.$$

Then $x_1 = 4.0, x_2 = 5.33333, \ldots, x_{15} = 19.72255, \ldots$ and clearly $\{x_n\}$ diverges slowly to ∞ (Figure 4.9).

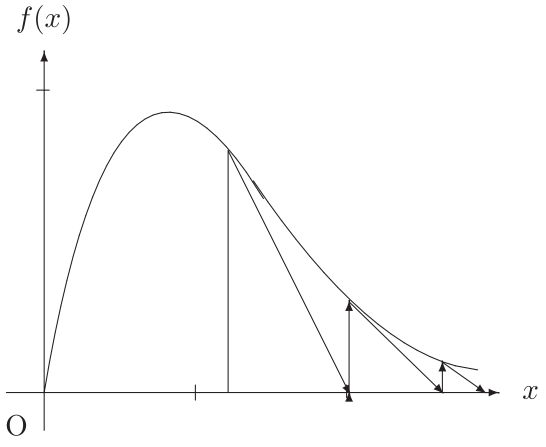


Figure 4.9: Newton-Raphson method produces a divergent sequence for $f(x) = xe^{-x}$.

Now consider a function $f(x) = x^3 - x - 3$ which will produce a cyclic sequence when initial guess is $x_0 = 0$. The sequence is
$x_1 = -3.0, x_2 = -1.961538, x_3 = -1.147176, x_4 = -0.006579,$
$x_5 = -3.000389, x_6 = -1.961818, x_7 = -1.147430, \ldots$
and it may be noted that $x_{k+4} \simeq x_k, k = 0, 1, 2, \ldots$ (Figure 4.10).

But, the initial guess $x_0 = 2$ gives the convergent sequence $x_1 = 1.72727, x_2 = 1.67369, x_3 = 1.67170, x_4 = 1.67170$.

The function $f(x) = \tan^{-1} x$ and $x_0 = 1.45$ gives a divergent oscillating sequence. If the initial guess is $x_0 = 1.45$ then

$$x_1 = -1.55026, x_2 = 1.84593, x_3 = -2.88911, \ldots$$

(Figure 4.11). But, if $x_0 = 1.5$ then $x_1 = -0.07956, x_2 = 0.00034, x_3 = 0.00000$.

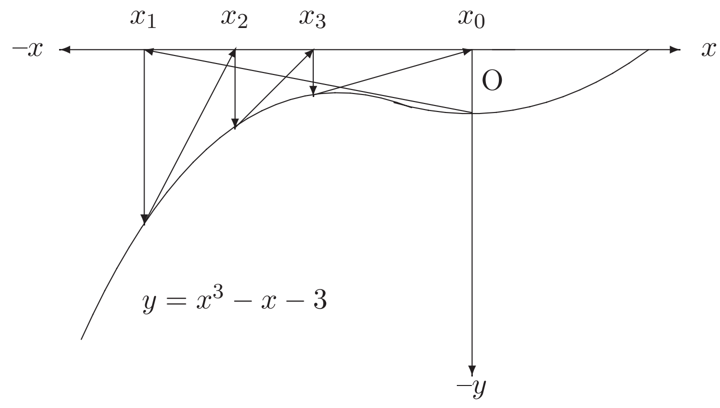


Figure 4.10: Newton-Raphson method produces a cyclic sequence for $f(x) = x^3 - x - 3$ when $x_0 = 0$.

4.6.1 Convergence of Newton-Raphson method

The Newton-Raphson iteration formula (4.25) is

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

Comparing this expression with fixed point iteration formula $x_{n+1} = \phi(x_n)$ and we obtain

$$\phi(x_n) = x_n - \frac{f(x_n)}{f'(x_n)}.$$

This can be written as

$$\phi(x) = x - \frac{f(x)}{f'(x)}.$$

It is already proved that the iteration method converges if $|\phi'(x)| < 1$. Therefore, Newton-Raphson method converges, if

$$\left|\frac{d}{dx}\left\{x - \frac{f(x)}{f'(x)}\right\}\right| < 1 \qquad \text{or} \qquad |f(x) \cdot f''(x)| < |f'(x)|^2 \tag{4.26}$$

within the interval under consideration. Newton-Raphson method converges if the initial guess x_0 is chosen sufficiently close to the root and the functions $f(x)$, $f'(x)$ and $f''(x)$ are continuous and bounded in any small interval containing the root. The rate of convergent of Newton-Raphson method is stated in the following theorem.

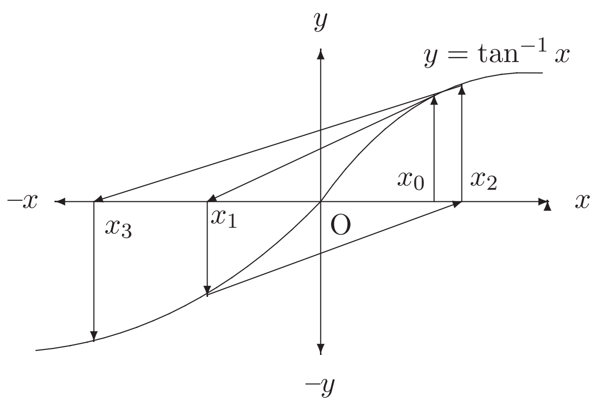


Figure 4.11: Newton-Raphson method produces a divergent oscillating sequence for $f(x) = \tan^{-1} x$ when $x_0 = 1.45$.

Theorem 4.4 *The rate of convergence of Newton-Raphson method is quadratic.*

Proof. Let ξ be a root of the equation $f(x) = 0$. Then $f(\xi) = 0$. The iteration scheme for Newton-Raphson method is

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

Let $x_n = \varepsilon_n + \xi$.

Therefore, above relation becomes

$$\begin{aligned}
\varepsilon_{n+1} + \xi &= \varepsilon_n + \xi - \frac{f(\varepsilon_n + \xi)}{f'(\varepsilon_n + \xi)} \\
\varepsilon_{n+1} &= \varepsilon_n - \frac{f(\xi) + \varepsilon_n f'(\xi) + (\varepsilon_n^2/2) f''(\xi) + \cdots}{f'(\xi) + \varepsilon_n f''(\xi) + \cdots} \quad \text{[by Taylor's series]} \\
&= \varepsilon_n - \frac{f'(\xi)\left[\varepsilon_n + \frac{\varepsilon_n^2}{2}\frac{f''(\xi)}{f'(\xi)} + \cdots\right]}{f'(\xi)\left[1 + \varepsilon_n \frac{f''(\xi)}{f'(\xi)} + \cdots\right]} \quad \text{[as } f(\xi) = 0\text{]} \\
&= \varepsilon_n - \left[\varepsilon_n + \frac{\varepsilon_n^2}{2}\frac{f''(\xi)}{f'(\xi)} + \cdots\right]\left[1 - \varepsilon_n \frac{f''(\xi)}{f'(\xi)} + \cdots\right] \\
&= -\frac{\varepsilon_n^2}{2}\frac{f''(\xi)}{f'(\xi)} + \varepsilon_n^2 \frac{f''(\xi)}{f'(\xi)} + O(\varepsilon_n^3) \\
&= \frac{1}{2}\varepsilon_n^2 \frac{f''(\xi)}{f'(\xi)} + O(\varepsilon_n^3).
\end{aligned}$$

Neglecting the terms of order ε_n^3 and higher powers the above expression becomes

$$\varepsilon_{n+1} = A\varepsilon_n^2, \text{ where } A = \frac{f''(\xi)}{2f'(\xi)}. \tag{4.27}$$

This relation shows that Newton-Raphson method has quadratic convergence or second order convergence.

Example 4.6.1 Use Newton-Raphson method to find a root of the equation $x^3 + x - 1 = 0$.

Solution. Let $f(x) = x^3 + x - 1$. Then $f(0) = -1 < 0$ and $f(1) = 1 > 0$. So one root lies between 0 and 1. Let $x_0 = 0$ be the initial root.
The iteration scheme is

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \\ &= x_n - \frac{x_n^3 + x_n - 1}{3x_n^2 + 1} = \frac{2x_n^3 + 1}{3x_n^2 + 1}. \end{aligned}$$

The sequence $\{x_n\}$ for different values of n is shown below.

n	x_n	x_{n+1}
0	0	1
1	1	0.7500
2	0.7500	0.6861
3	0.6861	0.6823
4	0.6823	0.6823

Therefore, a root of the equation is 0.682 correct up to three decimal places.

Example 4.6.2 Find an iteration scheme to find the kth root of a number a.

Solution. Let x be the kth root of a. That is, $x = a^{1/k}$ or $x^k - a = 0$.
Let $f(x) = x^k - a$. The iteration scheme is

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \\ \text{or, } x_{n+1} &= x_n - \frac{x_n^k - a}{k\,x_n^{k-1}} = \frac{k\,x_n^k - x_n^k + a}{k\,x_n^{k-1}} \\ &= \frac{1}{k}\Big[(k-1)x_n + \frac{a}{x_n^{k-1}}\Big]. \end{aligned}$$

Example 4.6.3 Write down an iteration scheme for finding square root of a positive number N. Hence find the square root of the number 2.

Solution. Let the square root of N be x. That is, $x = \sqrt{N}$, or, $x^2 - N = 0$. Thus the root of this equation is the required value of $\sqrt{N}$. Let $f(x) = x^2 - N$. By Newton-Raphson method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - N}{2x_n} = \frac{1}{2}\Big(x_n + \frac{N}{x_n}\Big).$$

This is the required iteration scheme.
<u>*Second Part.*</u> Let $x = \sqrt{2}$ or, $x^2 - 2 = 0$. Also, let $f(x) = x^2 - 2$. Then $f'(x) = 2x$. The Newton-Raphson iteration scheme is

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

Let $x_0 = 1$. The successive calculations are shown in the following.

n	x_n	x_{n+1}
0	1	1.5000
1	1.50000	1.41667
2	1.41667	1.41422
3	1.41422	1.41421

Therefore, the value of $\sqrt{2}$ is 1.4142, correct up to five significant figures.

Example 4.6.4 Find a root of the equation $x \log_{10} x = 4.77$ by Newton-Raphson method correct up to five decimal places.

Solution. Let $f(x) = x \log_{10} x - 4.77$. Here $f(6) = -0.10109 < 0$ and $f(7) = 1.14569 > 0$.
Therefore, one root lies between 6 and 7. Let the initial guess be $x_0 = 6$.
The iteration scheme is

$$x_{n+1} = x_n - \frac{x_n \log_{10} x_n - 4.77}{\log_{10} x_n + \log_{10} e} = \frac{0.43429x_n + 4.77}{\log_{10}(2.71828x_n)}.$$

The values of x_0, x_1, x_2 are shown in the following table.

n	x_n	x_{n+1}
0	6.000000	6.083358
1	6.083358	6.083152
2	6.083152	6.083153

Therefore, one root is 6.08315 correct up to five decimal places.

Example 4.6.5 The following expression $x_{n+1} = \dfrac{3\,x_n^2 + 2}{8}$ is an iteration scheme to find a root of the equation $f(x) = 0$. Find the function $f(x)$.

Solution. Let α be the root obtained by performing the iteration scheme

$$x_{n+1} = \frac{3\,x_n^2 + 2}{8}.$$

Therefore, $\lim\limits_{n\to\infty} x_n = \alpha$.
Thus, $\lim\limits_{n\to\infty} x_{n+1} = \dfrac{1}{8}\Big[3 \lim\limits_{n\to\infty} x_n^2 + 2\Big]$.
This gives $\alpha = \dfrac{1}{8}[3\alpha^2 + 2]$, i.e., $3\alpha^2 + 2 = 8\alpha$ or, $3\alpha^2 - 8\alpha + 2 = 0$.
Thus the required equation is $3x^2 - 8x + 2 = 0$ and hence $f(x) = 3x^2 - 8x + 2$.

Example 4.6.6 Discuss the Newton-Raphson method to find the root of the equation $x^{10} - 1 = 0$ starting with $x_0 = 0.5$.

Solution. The real roots of this equation are ± 1.
Here $f(x) = x^{10} - 1$.
Therefore,

$$x_{n+1} = x_n - \frac{x_n^{10} - 1}{10x_n^9} = \frac{9x_n^{10} + 1}{10x_n^9}.$$

When $x_0 = 0.5$ then $x_1 = \dfrac{9 \times (0.5)^{10} + 1}{10 \times (0.5)^9} = 51.65$, which is far away from the root 1.
This is because 0.5 was not close enough to the root $x = 1$.
But the sequence $\{x_n\}$ will converge to the root 1, although very slowly.
The initial root $x_0 = 0.9$ gives the first approximate root $x_1 = 1.068$, which is close to the root 1.
This example points out the role of initial approximation in Newton-Raphson method.

Example 4.6.7 Find a complex root of the equation $z^3 + 2z^2 + 2z + 1 = 0$ starting with the initial guess $-0.5 + 0.5i$.

Solution. Let $z_0 = -0.5 + 0.5i = (-0.5, 0.5)$ be the initial guess and $f(z) = z^3 + 2z^2 + 2z + 1$. Then $f'(z) = 3z^2 + 4z + 2$. The iteration scheme is

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}.$$

All the calculations are shown below.

n	z_n	$f(z_n)$	$f'(z_n)$	z_{n+1}
0	(-0.50000, 0.50000)	(0.25000, 0.25000)	(0.00000, 0.50000)	(-1.00000, 1.00000)
1	(-1.00000, 1.00000)	(1.00000,0.00000)	(-2.00000,-2.00000)	(-0.75000, 0.75000)
2	(-0.75000, 0.75000)	(0.34375, 0.09375)	(-1.00000,-0.37500)	(-0.41781, 0.71918)
3	(-0.41781, 0.71918)	(0.05444, 0.24110)	(-0.69919, 1.07384)	(-0.55230, 0.85744)
4	(-0.55230, 0.85744)	(0.08475,-0.02512)	(-1.49971, 0.58836)	(-0.49763, 0.86214)
5	(-0.49763, 0.86214)	(-0.00014, 0.00785)	(-1.47746, 0.87439)	(-0.50003, 0.86603)
6	(-0.50003, 0.86603)	(0.00004,-0.00003)	(-1.50005, 0.86587)	(-0.50000, 0.86603)

Thus one complex root is $(-0.5000, 0.8660)$ i.e., $-0.5 + 0.866\,i$.

4.7 Newton-Raphson Method for Multiple Root

The Newton-Raphson method can be used to find the multiple root of an equation. But, its generalized form

$$x_{n+1} = x_n - p\frac{f(x_n)}{f'(x_n)} \tag{4.28}$$

gives a faster convergent sequence. The term $\frac{1}{p}f'(x_n)$ is the slope of the straight line passing through $(x_n, f(x_n))$ and intersecting the x-axis at the point $(x_{n+1}, 0)$. The formula (4.28) reduces to Newton-Raphson formula when $p = 1$.

If ξ is a root of $f(x) = 0$ with multiplicity p, then ξ is also a root of $f'(x) = 0$ with multiplicity $(p-1)$, of $f''(x) = 0$ with multiplicity $(p-2)$ and so on. Hence the expression

$$x_0 - p\frac{f(x_0)}{f'(x_0)}, \quad x_0 - (p-1)\frac{f'(x_0)}{f''(x_0)}, \quad x_0 - (p-2)\frac{f''(x_0)}{f'''(x_0)}, \ldots$$

should have the same value if there is a root with multiplicity p, when the initial guess is very close to the exact root ξ.

Theorem 4.5 *The rate of convergence of the formula (4.28) is quadratic.*

Proof. Let ξ be a multiple root of multiplicity p, of the equation $f(x) = 0$. Then $f(\xi) = f'(\xi) = f''(\xi) = \cdots = f^{p-1}(\xi) = 0$ and $f^p(\xi) \neq 0$. Let $\varepsilon_n = x_n - \xi$. Then from (4.28),

$$\begin{aligned}
\varepsilon_{n+1} &= \varepsilon_n - p\frac{f(\varepsilon_n + \xi)}{f'(\varepsilon_n + \xi)} \\
&= \varepsilon_n - p\frac{f(\xi) + \varepsilon_n f'(\xi) + \cdots + \frac{\varepsilon_n^{p-1}}{(p-1)!}f^{p-1}(\xi) + \frac{\varepsilon_n^p}{p!}f^p(\xi) + \frac{\varepsilon_n^{p+1}}{(p+1)!}f^{p+1}(\xi) + \cdots}{f'(\xi) + \varepsilon_n f''(\xi) + \cdots + \frac{\varepsilon_n^{p-2}}{(p-2)!}f^{p-1}(\xi) + \frac{\varepsilon_n^{p-1}}{(p-1)!}f^p(\xi) + \frac{\varepsilon_n^p}{p!}f^{p+1}(\xi) + \cdots}
\end{aligned}$$

$$
\begin{aligned}
&= \varepsilon_n - p\frac{\frac{\varepsilon_n^p}{p!}f^p(\xi) + \frac{\varepsilon_n^{p+1}}{(p+1)!}f^{p+1}(\xi) + \cdots}{\frac{\varepsilon_n^{p-1}}{(p-1)!}f^p(\xi) + \frac{\varepsilon_n^p}{p!}f^{p+1}(\xi) + \cdots}\\
&= \varepsilon_n - p\left[\frac{\varepsilon_n}{p} + \frac{\varepsilon_n^2}{p(p+1)}\frac{f^{p+1}(\xi)}{f^p(\xi)} + \cdots\right]\left[1 + \frac{\varepsilon_n}{p}\frac{f^{p+1}(\xi)}{f^p(\xi)} + \cdots\right]^{-1}\\
&= \varepsilon_n - p\left[\frac{\varepsilon_n}{p} + \frac{\varepsilon_n^2}{p(p+1)}\frac{f^{p+1}(\xi)}{f^p(\xi)} + \cdots\right]\left[1 - \frac{\varepsilon_n}{p}\frac{f^{p+1}(\xi)}{f^p(\xi)} + \cdots\right]\\
&= \varepsilon_n - \left[\varepsilon_n + \frac{\varepsilon_n^2}{p+1}\frac{f^{p+1}(\xi)}{f^p(\xi)} - \frac{\varepsilon_n^2}{p}\frac{f^{p+1}(\xi)}{f^p(\xi)} + \cdots\right]\\
&= \varepsilon_n^2\left[\frac{1}{p(p+1)}\frac{f^{p+1}(\xi)}{f^p(\xi)}\right] + O(\varepsilon_n^3).
\end{aligned}
$$

Thus $\varepsilon_{n+1} = A\varepsilon_n^2$, where $A = \dfrac{1}{p(p+1)}\dfrac{f^{p+1}(\xi)}{f^p(\xi)}$.

This shows that the rate of convergence is quadratic.

Example 4.7.1 Find the double root of the equation $x^3 - 3x^2 + 4 = 0$.

Solution. Let $x_0 = 1.5$ and $f(x) = x^3 - 3x^2 + 4$.
$f'(x) = 3x^2 - 6x$, $f''(x) = 6x - 6$.
$x_1 = x_0 - 2\dfrac{f(x_0)}{f'(x_0)} = 1.5 - 2\dfrac{0.625}{-2.25} = 2.05556$ and
$x_1 = x_0 - 2\dfrac{f'(x_0)}{f''(x_0)} = 1.5 - \dfrac{-2.25}{3} = 2.25000$.
The close values of x_1 indicates that there is a double root near 2. Let $x_1 = 2.05556$.
Then $x_2 = x_1 - 2\dfrac{f(x_1)}{f'(x_1)} = 2.05556 - 2.\dfrac{0.00943}{0.34262} = 2.00051$
$x_2 = x_1 - 2\dfrac{f'(x_1)}{f''(x_1)} = 2.05556 - \dfrac{0.34262}{6.33336} = 2.00146$.
Thus there is a double root at $x = 2.00051$ which is sufficiently close to the actual root 2.
The Newton-Raphson method with same initial guess $x_0 = 1.5$ produces the sequence
$x_1 = 1.77778, x_2 = 1.89352, x_3 = 1.94776, x_4 = 1.97410, x_5 = 1.98714,$
$x_6 = 1.99353, x_7 = 1.99689, x_8 = 1.99850, x_9 = 1.99961, x_{10} = 1.99980$.
Thus at 10th iteration the Newton-Raphson method produces the root 2.000 correct up to three decimal places, while the formula (4.28) needs only two iterations.

4.8 Modification on Newton-Raphson Method

In the Newton-Raphson method, the derivative of the function $f(x)$ is calculated at each point x_n. That is, at each iteration two functions are evaluated at x_n, $n = 0, 1, 2, \ldots$.

But, some functions take much time to evaluate the derivative. To save this time one can change the iteration scheme of Newton-Raphson method as

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}. \tag{4.29}$$

That is, the derivative of $f(x)$ is calculated only at the initial guess instead of several different points x_n. This method reduces the time for calculating the derivatives. But, the rate of convergence of this method is linear, which is proved in Theorem 4.6.

Theorem 4.6 *The rate of convergence of the formula (4.29) is linear.*

Solution. Let ξ be the root of the equation $f(x) = 0$. Then $f(\xi) = 0$, and $\varepsilon_n = x_n - \xi$.
Therefore from (4.29),

$$\begin{aligned}\varepsilon_{n+1} &= \varepsilon_n - \frac{f(\varepsilon_n + \xi)}{f'(x_0)} = \varepsilon_n - \frac{f(\xi) + \varepsilon_n f'(\xi) + \cdots}{f'(x_0)} \\ &= \varepsilon_n\left(1 - \frac{f'(\xi)}{f'(x_0)}\right) + O(\varepsilon_n^2).\end{aligned}$$

Neglecting ε_n^2 and higher powers of ε_n^2 and denoting $A = 1 - \dfrac{f'(\xi)}{f'(x_0)}$ the above error term becomes

$$\varepsilon_{n+1} = A\varepsilon_n. \tag{4.30}$$

This proved that the rate of convergence of the formula (4.29) is linear.

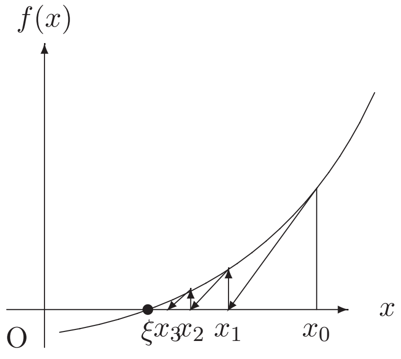


Figure 4.12: Geometrical meaning of the formula (4.29).

Geometrical interpretation

The gradient of tangent at the point x_n is $f'(x_0)$ for all n.

Thus the line passing through the point $(x_n, f(x_n))$ is parallel to the tangent drawn at $(x_0, f(x_0))$, i.e., the tangent at $(x_n, f(x_n))$ in Newton-Raphson method is replaced by a line parallel to the tangent drawn at $(x_0, f(x_0))$ and passing through the point $(x_n, f(x_n))$. This phenomena is shown in Figure 4.12.

Example 4.8.1 Find a root of the equation $x^3 - x + 1 = 0$ using formula (4.29) and Newton-Raphson method up to four decimal places.

Solution. One root of this equation lies between -2 and -1. Let $x_0 = -1.5$ and $f(x) = x^3 - x + 1$. Then $f'(x_0) = 5.75$.
The iteration scheme of the formula (4.29) is

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_0)} \\ &= x_n - \frac{x_n^3 - x_n + 1}{5.75} = -\frac{1}{5.75}(x_n^3 - 6.75x_n + 1). \end{aligned}$$

All the calculations are shown in the following table.

n	x_n	x_{n+1}
0	–1.50000	–1.34783
1	–1.34783	–1.33032
2	–1.33032	–1.32614
3	–1.32614	–1.32508
4	–1.32508	–1.32481
5	–1.32481	–1.32474
6	–1.32474	–1.32472
7	–1.32472	–1.32472

Therefore, one root of the given equation is -1.3247, correct up to four decimal places attained at 7th iteration.

Using Newton-Raphson method
The iteration scheme for Newton-Raphson method is

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \\ &= x_n - \frac{x_n^3 - x_n + 1}{3x_n^2 - 1} = \frac{2x_n^3 - 1}{3x_n^2 - 1}. \end{aligned}$$

Let $x_0 = -1.5$. The successive calculations are shown below.

n	x_n	x_{n+1}
0	–1.50000	–1.34783
1	–1.34783	–1.32520
2	–1.32520	–1.32472
3	–1.32472	–1.32472

Therefore, a root is -1.3247 correct up to four decimal places, attained at 3rd iteration.

This example shows that Newton-Raphson method is more faster than the method given by (4.29).

4.9 Modified Newton-Raphson Method [1]

In this method, the iteration scheme is taken as

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n + a(x_n)f(x_n))} = \phi(x_n) \quad \text{(say)} \tag{4.31}$$

That is,

$$\phi(x) = x - \frac{f(x)}{f'(x + a(x)f(x))},$$

where $a(x)$ is a smooth function.

Now,

$$\begin{aligned}\phi'(x) &= 1 - \frac{f'(x)}{f'(x+a(x)f(x))} \\ &\quad + \frac{f(x)f''(x+a(x)f(x))(1+a'(x)f(x)+a(x)f'(x))}{\{f'(x+a(x)f(x))\}^2}\end{aligned} \tag{4.32}$$

$$\begin{aligned}\text{and} \quad \phi''(x) &= -\frac{f''(x)}{f'(x+a(x)f(x))} \\ &+ 2\frac{f'(x)f''(x+a(x)f(x))(1+a'(x)f(x)+a(x)f'(x))}{\{f'(x+a(x)f(x))\}^2} \\ &- 2\frac{f(x)\{f''(x+a(x)f(x))\}^2\{1+a'(x)f(x)+a(x)f'(x)\}^2}{\{f'(x+a(x)f(x))\}^3}\end{aligned}$$

[1] H.H.H.Homeier, A modified Newton method for root finding with cubic convergence, J. Computational and Applied Mathematics, 157 (2003) 227-230.

$$+ \quad \frac{f(x)f'''(x+a(x)f(x))\{1+a'(x)f(x)+a(x)f'(x)\}^2}{\{f'(x+a(x)f(x))\}^2}$$
$$+ \quad \frac{\{f(x)\}^2f''(x+a(x)f(x))a''(x)}{\{f'(x+a(x)f(x))\}^2}$$
$$+ \quad \frac{f(x)f''(x+a(x)f(x))\{2a'(x)f'(x)+a(x)f''(x)\}}{\{f'(x+a(x)f(x))\}^2}.$$

If ξ be the root of the equation $f(x)=0$ then $f(\xi)=0$ and hence $\phi(\xi)=\xi$, $\phi'(\xi)=0$ [from (4.32)] and

$$\begin{aligned}\phi''(\xi) &= -\frac{f''(\xi)}{f'(\xi)} + \frac{2f'(\xi)f''(\xi)\{1+a(\xi)f'(\xi)\}}{\{f'(\xi)\}^2} \\ &= \frac{f''(\xi)}{f'(\xi)}\{1+2a(\xi)f'(\xi)\}. \qquad (4.33)\end{aligned}$$

Now, if $a(\xi) = -\frac{1}{2f'(\xi)}$ then $\phi''(\xi)=0$. This is the easiest way to put $a(x) = -\frac{1}{2f'(x)}$.
Hence the iteration scheme for modified Newton-Raphson method is

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n+a(x_n)f(x_n))}, \text{ where } a(x_n) = -\frac{1}{2f'(x_n)}. \qquad (4.34)$$

Also we have

$$\phi(\xi)=\xi, \quad \phi'(\xi)=0, \quad \phi''(\xi)=0. \qquad (4.35)$$

The rate of convergence of this method is evaluated below:

Theorem 4.7 *The rate of convergence of modified Newton-Raphson method is cubic.*

Proof. Let ξ be the root of the equation $f(x)=0$. Also, let $\varepsilon_n = x_n - \xi$.
Therefore, $x_{n+1} = \phi(x_n) = \phi(\varepsilon_n + \xi)$
or, $\varepsilon_{n+1} + \xi = \phi(\xi) + \varepsilon_n\phi'(\xi) + \frac{\varepsilon_n^2}{2!}\phi''(\xi) + \frac{\varepsilon_n^3}{3!}\phi'''(\xi) + \cdots$

or, $\varepsilon_{n+1} = \frac{\varepsilon_n^3}{3!}\phi'''(\xi) + O(\varepsilon_n^4)$.

[Using the facts $\phi(\xi)=\xi, \phi'(\xi)=0, \phi''(\xi)=0$ from (4.35)]
Neglecting the terms ε_n^4 and higher powers of ε_n^4 the above equation reduces to

$$\varepsilon_{n+1} = A\varepsilon_n^3, \text{ where } A = \frac{1}{3!}\phi'''(\xi). \qquad (4.36)$$

This shows that the rate of modified Newton-Raphson method is cubic.

This method evaluates three functions $f(x), x + a(x)f'(x)$ and $f'(x + a(x)f(x))$ at each iteration.

Example 4.9.1 Find a root of the equation $x^3-3x^2+4 = 0$ using modified Newton-Raphson method, starting with $x_0 = 1.5$.

Solution. Let $f(x) = x^3 - 3x^2 + 4$. $f'(x) = 3x^2 - 6x$,
$a(x) = -\dfrac{1}{2f'(x)} = -\dfrac{1}{(6x^2 - 12x)}$.
Let $g(x) = x + a(x)f(x) = x - \dfrac{x^3 - 3x^2 + 4}{6x^2 - 12x} = \dfrac{5x^3 - 9x^2 - 4}{6x^2 - 12x}$.
Then the iteration scheme is $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(g(x_n))}$.
The calculations for each value of n is listed below.

n	x_n	$f(x_n)$	$g(x_n)$	$f'(g(x_n))$	x_{n+1}
0	1.50000	0.62500	1.63889	–1.77546	1.85202
1	1.85202	0.06245	1.89000	–0.62370	1.95215
2	1.95215	0.00676	1.96421	–0.21090	1.98420
3	1.98420	0.00074	1.98816	–0.07062	1.99468
4	1.99468	0.00008	1.99601	–0.02389	1.99803
5	1.99803	0.00001	1.99852	–0.00887	1.99916
6	1.99916	0.000002	1.99937	–0.00378	1.99969

Therefor, a root of the given equation is 2.000, correct up to three decimal places, and this value is attained at 6th iteration, while Newton-Raphson method takes 10 iterations (see Example 4.7.1).

Algorithm 4.4 (Newton-Raphson method). This algorithm finds a root of the equation $f(x) = 0$ by Newton-Raphson method, when $f(x), f'(x)$ and initial guess x_0 are supplied.

```
**Algorithm Newton-Raphson**
// $fd(x)$ is the derivative of $f(x)$ and $\varepsilon$ is the error tolerance.//
Input function $f(x), fd(x)$;
Read $x_0, \varepsilon$;
Set $x_1 = x_0$;
do
    Set $x_0 = x_1$;
    Compute $x_1 = x_0 - f(x_0)/fd(x_0)$;
while ($|x_1 - x_0| > \varepsilon$);
Print 'The root is', $x_1$;
**end Newton-Raphson**
```

Program 4.4

```
/* Program Newton-Raphson
   Program to find a root of the equation x*x*x-3x+1=0 by Newton-
   Raphson method. f(x) and its derivative fd(x) are to be supplied. */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void main()
{
 int k=0; /* counts number of iterations */
 float x1,x0; /* x0 is the initial guess */
 float eps=1e-5; /* error tolerance */
 float f(float x);
 float fd(float x);
 printf("\nEnter the initial guess x0 ");
 scanf("%f",&x0);
 x1=x0;
 do
 {
     k++;
     x0=x1;
     x1=x0-f(x0)/fd(x0);
 }while(fabs(x1-x0)>eps);
 printf("One root is %8.5f obtained at %d th iteration ",x1,k);
} /* main */
/* definition of the function f(x) */
float f(float x)
 {
   return(x*x*x-3*x+1);
 }
/* definition of the function fd(x) */
float fd(float x)
 {
   return(3*x*x-3);
 }
```

A sample of input/output:

```
Enter the initial guess x0 1.1
One root is  1.53209 obtained at 7 th iteration
```

4.10 Secant Method

The main drawback of Newton-Raphson method is to determine the derivatives at several points. In many cases, calculation of derivatives takes much time. In some cases closed form expression for $f'(x)$ is not available.

To remove this drawback, the derivative $f'(x)$ is approximated by the backward difference

$$f'(x_i) \simeq \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

where x_i and x_{i-1} are two approximations to the root but need not require the condition $f(x_i) \cdot f(x_{i-1}) < 0$.

Then from the Newton-Raphson method

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}. \tag{4.37}$$

This formula is same as the formula for Regula-falsi method and this formula needs two initial guess x_0 and x_1 of the root.

Note 4.10.1 Regula-falsi method need an interval where the root belongs to, i.e., if $[x_0, x_1]$ is the interval then $f(x_0) \cdot f(x_1) < 0$. But, secant method needs two nearest values x_0 and x_1 of the exact root and not necessarily $f(x_0) \cdot f(x_1) < 0$.

Geometrical interpretation

A geometrical interpretation of secant method is illustrated in Figure 4.13.

A secant is drawn connecting $f(x_{i-1})$ and $f(x_i)$. The point where it cuts the x-axis is x_{i+1}. Another secant is drawn connecting $f(x_i)$ and $f(x_{i+1})$ to obtain x_{i+2}, and so on.

4.10.1 Convergence of secant method

The formula for secant method is

$$x_{n+1} = x_n - \frac{(x_n - x_{n-1})f(x_n)}{f(x_n) - f(x_{n-1})}. \tag{4.38}$$

Let ξ be the exact root of the equation $f(x) = 0$ and the error at the nth iteration is $\varepsilon_n = x_n - \xi$. Also $f(\xi) = 0$.

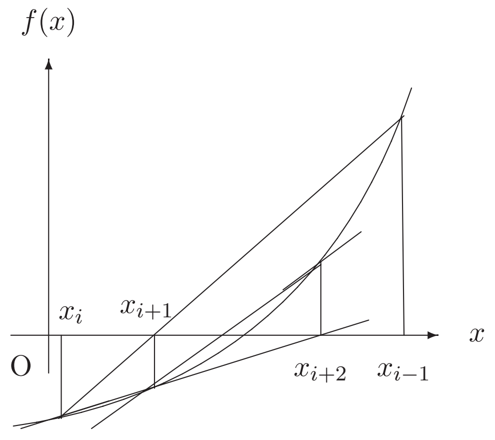


Figure 4.13: Geometrical interpretation of secant method.

Then (4.38) becomes

$$\begin{aligned}
\varepsilon_{n+1} &= \varepsilon_n - \frac{(\varepsilon_n - \varepsilon_{n-1})f(\varepsilon_n + \xi)}{f(\varepsilon_n+\xi) - f(\varepsilon_{n-1}+\xi)} \\
&= \varepsilon_n - \frac{(\varepsilon_n - \varepsilon_{n-1})[f(\xi) + \varepsilon_n f'(\xi) + (\varepsilon_n^2/2)f''(\xi) + \cdots]}{(\varepsilon_n - \varepsilon_{n-1})f'(\xi) + \frac{1}{2}(\varepsilon_n^2 - \varepsilon_{n-1}^2)f''(\xi) + \cdots} \\
&= \varepsilon_n - \left[\varepsilon_n + \frac{\varepsilon_n^2}{2}\frac{f''(\xi)}{f'(\xi)} + \cdots\right]\left[1 + \frac{1}{2}(\varepsilon_n + \varepsilon_{n-1})\frac{f''(\xi)}{f'(\xi)} + \cdots\right]^{-1} \\
&= \frac{1}{2}\varepsilon_n\varepsilon_{n-1}\frac{f''(\xi)}{f'(\xi)} + O(\varepsilon_n^2\varepsilon_{n-1} + \varepsilon_n\varepsilon_{n-1}^2).
\end{aligned}$$

Thus $\varepsilon_{n+1} = c\varepsilon_n\varepsilon_{n-1}$ where $c = \dfrac{1}{2}\dfrac{f''(\xi)}{f'(\xi)}$. This is a non-linear difference equation and to solve it, let $\varepsilon_{n+1} = A\varepsilon_n^p$. Then $\varepsilon_n = A\varepsilon_{n-1}^p$. This gives $\varepsilon_{n-1} = \varepsilon_n^{1/p}A^{-1/p}$.

Therefore, $A\varepsilon_n^p = c\varepsilon_n\varepsilon_n^{1/p}A^{-1/p}$, i.e., $\varepsilon_n^p = cA^{-(1+1/p)}\varepsilon_n^{1+1/p}$.

Equating the power of ε_n on both sides, obtain the equation for p

$$p = 1 + \frac{1}{p} \qquad \text{or} \qquad p = \frac{1}{2}(1 \pm \sqrt{5}).$$

Positive sign gives $p = 1.618$. Hence $\varepsilon_{n+1} = A\varepsilon_n^{1.618}$.

Thus the rate of convergence of secant method is 1.618, which is smaller than the Newton-Raphson method. Thus this method converges at a slower rate. However, this method evaluates function only once in each iteration, but Newton-Raphson method evaluates two functions f and f' in each iteration. In this context, the secant method is more efficient as compared to Newton-Raphson method.

Example 4.10.1 Find a root of the equation $x^3 - 8x - 4 = 0$ using secant method.

Solution. Let $f(x) = x^3 - 8x - 4 = 0$. One root lies between 3 and 4. Let the initial approximation be $x_0 = 3, x_1 = 3.5$. The formula for x_2 is $x_2 = \dfrac{x_0 f(x_1) - x_1 f(x_0)}{f(x_1) - f(x_0)}$.
The calculations are shown below:

x_0	$f(x_0)$	x_1	$f(x_1)$	x_2	$f(x_2)$
3.0000	–1.0000	3.5000	10.8750	3.0421	–0.1841
3.5000	10.8750	3.0421	–0.1841	3.0497	–0.0333
3.0421	–0.1841	3.0497	–0.0333	3.0514	0.0005
3.0497	–0.0333	3.0514	0.0005	3.0514	0.0005

Therefore, a root is 3.051 correct up to four significant figures.

Algorithm 4.5 (Secant method). This algorithm finds a root of the equation $f(x) = 0$ by secant method when two initial guesses x_0 and x_1 are supplied.

Algorithm Secant

```
// The iteration terminates when $|f(x_1) - f(x_0)|$ is very small (in this case slope of
the secant is very small) and $|f(x_2)| < \varepsilon$, $\varepsilon$ is the error tolerance, $\delta$ is a very small
quantity, taken as 0.//
Read $\varepsilon$ and $\delta$;
Input function $f(x)$;
1. $fx_0 = f(x_0); fx_1 = f(x_1)$;
if $|fx_1 - fx_0| < \delta$ then
    Print ‘Slope too small, the method does not give correct root or fail’
    Stop;
endif;
Compute $x_2 = \dfrac{x_0 \cdot fx_1 - x_1 \cdot fx_0}{fx_1 - fx_0}$;
Compute $f_2 = f(x_2)$;
if $|f_2| < \varepsilon$ then
    Print ‘A root is’, $x_2$;
    Stop;
endif;
Set $fx_0 = fx_1$; $fx_1 = fx_2$;
Set $x_0 = x_1$; $x_1 = x_2$;
Go to 1;
end Secant
```

Program 4.5

```
/* Program Secant
   Program to find a root of the equation x*sin(x)-1=0 by
   secant method. It is assumed that a root lies between x0 and x1.*/
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void main()
{
 float x0,x1,x2,fx0,fx1,fx2;
 float eps=1e-5; /* error tolerance */
 float delta=1e-5; /* slope */
 float f(float x);
 printf("\nEnter the values of x0 and x1 ");
 scanf("%f %f",&x0,&x1);
 fx0=f(x0);
 fx1=f(x1);
 if(fabs(fx1-fx0)<delta){
    printf("Slope too small
            the method does not give correct root or fail");
    exit(0);
   }
 do
 {
    x2=(x0*fx1-x1*fx0)/(fx1-fx0);
    fx2=f(x2);
    if(fabs(fx2)<eps){
        printf("One root is %8.5f ",x2);
        exit(0);
      }
    fx0=fx1; fx1=fx2;
    x0=x1;   x1=x2;
 }while(fabs(fx2)>eps);
} /* main */
/* definition of function f(x), it may change accordingly */
float f(float x)
{
  return(x*sin(x)-1);
}
```

A sample of input/output:

```
Enter the values of x0 and x1 0 1
One root is  1.11416
```

4.11 Chebyshev Method

Let us consider the equation $f(x)=0$. The function $f(x)$ is expanded by Taylor's series in the neighbourhood of x_n as $0=f(x)=f(x_n)+(x-x_n)f'(x_n)+\cdots$.

This relation gives $\qquad x = x_n - \dfrac{f(x_n)}{f'(x_n)}$.

This is the $(n+1)$th approximation to the root. Therefore,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{4.39}$$

Again, expanding $f(x)$ by Taylor's series and retaining up to second order term, shown below.

$$0 = f(x) = f(x_n) + (x-x_n)f'(x_n) + \frac{(x-x_n)^2}{2}f''(x_n)$$

Therefore, $\qquad f(x_{n+1}) = f(x_n) + (x_{n+1}-x_n)f'(x_n) + \dfrac{(x_{n+1}-x_n)^2}{2}f''(x_n) = 0.$

Substituting the value of $x_{n+1}-x_n$ from (4.39) to the last term and we find

$$f(x_n) + (x_{n+1}-x_n)f'(x_n) + \frac{1}{2}\frac{[f(x_n)]^2}{[f'(x_n)]^2}f''(x_n) = 0.$$

Thus,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{1}{2}\frac{[f(x_n)]^2}{[f'(x_n)]^3}f''(x_n). \tag{4.40}$$

This formula is the extended form of Newton-Raphson formula and it is known as Chebyshev's formula.

The rate of convergence of this method is cubic.

4.12 Muller Method

The main idea of this method is, the function $f(x)$ is approximated by a quadratic polynomial passing through the three points in the neighbour of the root. The root of this quadratic is assumed to approximate the root of the equation $f(x)=0$.

Let x_{n-2}, x_{n-1}, x_n be any three distinct approximation to a root of the equation $f(x)=0$. We denote $f(x_{n-2})=f_{n-2}, f(x_{n-1})=f_{n-1}$ and $f(x_n)=f_n$.

Let the quadratic polynomial be

$$f(x) = ax^2 + bx + c. \tag{4.41}$$

Suppose, (4.41) passes through the points (x_{n-2}, f_{n-2}), (x_{n-1}, f_{n-1}) and (x_n, f_n), then

$$ax_{n-2}^2 + bx_{n-2} + c = f_{n-2} \tag{4.42}$$
$$ax_{n-1}^2 + bx_{n-1} + c = f_{n-1} \tag{4.43}$$
$$ax_n^2 + bx_n + c = f_n \tag{4.44}$$

Eliminating a, b, c from (4.41)-(4.44), we obtain the following determinant

$$\begin{vmatrix} f(x) & x^2 & x & 1 \\ f_{n-2} & x_{n-2}^2 & x_{n-2} & 1 \\ f_{n-1} & x_{n-1}^2 & x_{n-1} & 1 \\ f_n & x_n^2 & x_n & 1 \end{vmatrix} = 0.$$

By expanding this determinant the function $f(x)$ can be written as

$$\begin{aligned} f(x) = {} & \frac{(x - x_{n-1})(x - x_n)}{(x_{n-2} - x_{n-1})(x_{n-2} - x_n)} f_{n-2} + \frac{(x - x_{n-2})(x - x_n)}{(x_{n-1} - x_{n-2})(x_{n-1} - x_n)} f_{n-1} \\ & + \frac{(x - x_{n-2})(x - x_{n-1})}{(x_n - x_{n-2})(x_n - x_{n-1})} f_n. \end{aligned} \tag{4.45}$$

This is a quadratic polynomial passing through the given points.

Let $h = x - x_n, h_n = x_n - x_{n-1}$ and $h_{n-1} = x_{n-1} - x_{n-2}$. Then above equation reduces to

$$\begin{aligned} & \frac{h(h + h_n)}{h_{n-1}(h_{n-1} + h_n)} f_{n-2} - \frac{h(h + h_n + h_{n-1})}{h_n h_{n-1}} f_{n-1} \\ & \qquad + \frac{(h + h_n)(h + h_n + h_{n-1})}{h_n(h_n + h_{n-1})} f_n = 0, \end{aligned} \tag{4.46}$$

since $f(x) = 0$.

Now, introducing

$$\lambda = \frac{h}{h_n}, \qquad \lambda_n = \frac{h_n}{h_{n-1}} \qquad \text{and} \qquad \delta_n = 1 + \lambda_n.$$

Therefore, the equation (4.46) reduces to the following form

$$\begin{aligned} & \lambda^2(f_{n-2}\lambda_n^2 - f_{n-1}\lambda_n\delta_n + f_n\lambda_n)\delta_n^{-1} \\ & + \lambda\{f_{n-2}\lambda_n^2 - f_{n-2}\delta_n^2 + f_n(\lambda_n + \delta_n)\}\delta_n^{-1} + f_n = 0 \end{aligned} \tag{4.47}$$

or,

$$\lambda^2 c_n + \lambda g_n + \delta_n f_n = 0, \tag{4.48}$$

where
$$g_n = \lambda_n^2 f_{n-2} - \delta_n^2 f_{n-1} + (\lambda_n + \delta_n) f_n$$
$$c_n = \lambda_n(\lambda_n f_{n-2} - \delta_n f_{n-1} + f_n).$$

The equation (4.48) now becomes

$$\delta_n f_n \left(\frac{1}{\lambda^2}\right) + \frac{g_n}{\lambda} + c_n = 0.$$

When solving for $\frac{1}{\lambda}$ this equation gives

$$\lambda = -\frac{2\delta_n f_n}{g_n \pm \sqrt{g_n^2 - 4\delta_n f_n c_n}}. \tag{4.49}$$

The sign in the denominator of (4.49) is $\pm$ according as $g_n > 0$ or $g_n < 0$. Thus

$$\lambda = \frac{x - x_n}{x_n - x_{n-1}} \qquad \text{or} \qquad x = x_n + (x_n - x_{n-1})\lambda. \tag{4.50}$$

Now, replacing x on left hand side by x_{n+1} and obtain the formula

$$x_{n+1} = x_n + (x_n - x_{n-1})\lambda, \tag{4.51}$$

which is called the **Muller method**.

This method is also an iterative method and free from evaluation of derivative as in Newton-Raphson method.

Example 4.12.1 Find a root of the equation $x^3 - 3x - 5 = 0$ using Muller's method which lies between 2 and 3.

Solution. Let $x_0 = 2.0, x_1 = 2.5$ and $x_2 = 3.0$, $f(x) = x^3 - 3x - 5$.

$$h_n = x_n - x_{n-1}, \quad \lambda_n = \frac{h_n}{h_{n-1}}, \quad \delta_n = 1 + \lambda_n$$
$$g_n = \lambda_n^2 f_{n-2} - \delta_n^2 f_{n-1} + (\lambda_n + \delta_n) f_n$$
$$c_n = \lambda_n(\lambda_n f_{n-2} - \delta_n f_{n-1} + f_n)$$
$$\lambda = -\frac{2\delta_n f_n}{g_n \pm \sqrt{(g_n^2 - 4\delta_n f_n c_n)}}$$
$$x_{n+1} = x_n + h_n\lambda.$$

All the calculations are shown in the following.

n	x_{n-2}	x_{n-1}	x_n	g_n	c_n	λ	x_{n+1}
2	2.00000	2.50000	3.00000	23.50000	3.75000	–1.43497	2.28252
3	2.50000	3.00000	2.28252	3.89279	-1.74262	0.00494	2.27897
4	3.00000	2.28252	2.27897	-0.04477	0.00010	-0.01263	2.27902
5	2.28252	2.27897	2.27902	0.00056	0.00000	-0.00338	2.27902

Therefore, one root is 2.2790 correct up to four decimal places.

Summary of the Methods

Method	Formula	Order of Convergent	Evaluation of function in each step
1. Bisection	$x_{n+1} = \dfrac{x_n + x_{n-1}}{2}$	Gain of one bit per iteration	1
2. False Position	$x_{n+1} = \dfrac{x_{n-1}f_n - x_n f_{n-1}}{f_n - f_{n-1}}$	1	1
3. Iteration	$x_{n+1} = \phi(x_n)$	1	1
4. Newton-Raphson	$x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$	2	2
5. Secant	$x_{n+1} = \dfrac{x_{n-1}f_n - x_n f_{n-1}}{f_n - f_{n-1}}$	1.62	1
6. Modified Newton-Raphson	$x_{n+1} = x_n - \dfrac{f_n}{f'(x_n - \frac{1}{2}f_n/f'_n)}$	3	3
7. Chebyshev	$x_{n+1} = x_n - \dfrac{f_n}{f'_n} - \dfrac{1}{2}\dfrac{f_n^2}{f'^3_n}f''_n$	3	3
8. Muller	$x_{n+1} = x_n + (x_n - x_{n-1})\lambda$	1.84	1

Example 4.12.2 Consider the iteration method $x_{n+1} = \phi(x_n), n = 0, 1, 2, \ldots$ for solving the equation $f(x) = 0$. If the iteration function is in the form

$$\phi(x) = x - \alpha f(x) - \beta\{f(x)\}^2 - \gamma\{f(x)\}^3$$

where α, β and γ are arbitrary parameters then find the values of α, β, γ such that the iteration method has (i) third and (ii) fourth order convergence.

Solution. Let ξ be an exact root of the equation $f(x)=0$, i.e., $f(\xi)=0$.
Now,

$$\begin{aligned}
\phi'(x) &= 1-\alpha f'(x)-2\beta f(x)f'(x)-3\gamma\{f(x)\}^2 f'(x)\\
\phi''(x) &= -\alpha f''(x)-2\beta\{f'(x)\}^2-2\beta f(x)f''(x)-6\gamma f(x)\{f'(x)\}^2\\
&\quad -3\gamma\{f(x)\}^2 f''(x)\\
\phi'''(x) &= -\alpha f'''(x)-6\beta f'(x)f''(x)-2\beta f(x)f'''(x)-6\gamma\{f'(x)\}^3\\
&\quad -18\gamma f(x)f'(x)f''(x)-3\gamma\{f(x)\}^2 f'''(x).
\end{aligned}$$

Substituting $f(\xi)=0$ to the above equations.
$\phi(\xi)=\xi$
$\phi'(\xi)=1-\alpha f'(\xi)$
$\phi''(\xi)=-\alpha f''(\xi)-2\beta\{f'(\xi)\}^2$
$\phi'''(\xi)=-\alpha f'''(\xi)-6\beta f'(\xi)f''(\xi)-6\gamma\{f'(\xi)\}^3$.

Let $\varepsilon_n=x_n-\xi$.
Then $\varepsilon_{n+1}+\xi=\phi(\varepsilon_n+\xi)=\phi(\xi)+\varepsilon_n\phi'(\xi)+\frac{1}{2}\varepsilon_n^2\phi''(\xi)+\frac{\varepsilon_n^3}{6}\phi'''(\xi)+\frac{\varepsilon_n^4}{24}\phi^{iv}(\xi)+\cdots$
(i) For third order convergence, the value of $\phi'(\xi)$ and $\phi''(\xi)$ should be zero. In this case,

$$\varepsilon_{n+1}=\frac{\varepsilon_n^3}{6}\phi'''(\xi)+\cdots \qquad \text{or,} \qquad \varepsilon_{n+1}=A\varepsilon_n^3.$$

Thus, $1-\alpha f'(\xi)=0$ and $-\alpha f''(\xi)-2\beta\{f'(\xi)\}^2=0$.
That is, $\alpha=\dfrac{1}{f'(\xi)}, \qquad \beta=-\dfrac{f''(\xi)}{2\{f'(\xi)\}^3}$.
Hence, for third order convergence, the values of α and β are given by

$$\alpha=\frac{1}{f'(x)}, \qquad \beta=-\frac{f''(x)}{2\{f'(x)\}^3}.$$

(ii) For the fourth order convergence, $\phi'(\xi)=\phi''(\xi)=\phi'''(\xi)=0$.
In this case $\varepsilon_{n+1}=A\varepsilon_n^4$.
Then $\alpha=-\dfrac{1}{f'(x)}, \qquad \beta=-\dfrac{f''(x)}{2\{f'(x)\}^3}$

and $-\alpha f'''(\xi)-6\beta f'(\xi)f''(\xi)-6\gamma\{f'(\xi)\}^3=0$.
That is,

$$6\gamma\{f'(x)\}^3=-\alpha f'''(x)-6\beta f'(x)f''(x)=-\frac{f'''(x)}{f'(x)}+6\frac{f''(x)}{2\{f'(x)\}^3}f'(x)f''(x)$$

$$\text{or,} \qquad \gamma=-\frac{f'''(x)}{6\{f'(x)\}^4}+\frac{\{f''(x)\}^2}{2\{f'(x)\}^5}.$$

Example 4.12.3 The equation $x^3 - 5x^2 + 4x - 3 = 0$ has one root near $x = 4$, which is to be computed by the iteration

$$\begin{aligned} x_0 &= 4 \\ x_{n+1} &= \frac{3 + (k-4)x_n + 5x_n^2 - x_n^3}{k}, \qquad k \text{ is integer.} \end{aligned}$$

Find the value of k such that the iteration scheme has fast convergence.

Solution. Let $\lim\limits_{n\to\infty} x_n = \lim\limits_{n\to\infty} x_{n+1} = \xi$.
Since ξ is a root of the given equation,

$$\xi^3 - 5\xi^2 + 4\xi - 3 = 0.$$

Substituting, $x_n = \xi + \varepsilon_n$, $x_{n+1} = \xi + \varepsilon_{n+1}$ to the given iteration scheme. Then

$$\begin{aligned} k(\varepsilon_{n+1} + \xi) &= 3 + (k-4)(\varepsilon_n + \xi) + 5(\varepsilon_n + \xi)^2 - (\varepsilon_n + \xi)^3 \\ k\varepsilon_{n+1} &= (3 - 4\xi + 5\xi^2 - \xi^3) + \varepsilon_n\{(k-4) + 10\xi - 3\xi^2\} \\ &\quad + \varepsilon_n^2(5 - 3\xi) - \varepsilon_n^3 \end{aligned}$$

$$\text{or,} \quad k\varepsilon_{n+1} = \varepsilon_n\{(k-4) + 10\xi - 3\xi^2\} + \varepsilon_n^2(5 - 3\xi) - \varepsilon_n^3.$$

If $k - 4 + 10\xi - 3\xi^2 = 0$ or, $k = 3\xi^2 - 10\xi + 4$ then

$$\varepsilon_{n+1} = A\varepsilon_n^2, \text{ where } A = \frac{1}{k}(5 - 3\xi)$$

i.e., the rate of convergence is 2.
The values of ξ is determined from the equation

$$\xi^3 - 5\xi^2 + 4\xi - 3 = 0.$$

The Newton-Raphson method is used to determine the approximate value of ξ as $\xi = 4.22069$. Thus the required value of k is $k = 3\xi^2 - 10\xi + 4 = 15.23577$.

Example 4.12.4 Consider the following iteration scheme

$$x_{n+1} = x_n - \frac{af(x_n)}{f'(x_n - bf(x_n)/f'(x_n))}$$

where a, b are arbitrary parameters, for solving the equation $f(x) = 0$. Determine a and b such that the iteration method is of order as high as possible for finding a simple root of $f(x) = 0$.

Solution. Here the iteration scheme is $x_{n+1} = \phi(x_n)$ where

$$\phi(x) = x - \frac{af(x)}{f'(x - bf(x)/f'(x))} = x - \frac{af(x)}{f'(g(x))}$$

where

$$g(x) = x - \frac{bf(x)}{f'(x)}, \qquad g'(x) = 1 - \frac{b\{f'(x)\}^2 - bf(x)f''(x)}{\{f'(x)\}^2}.$$

If ξ be a root of $f(x) = 0$ then $f(\xi) = 0$, $g(\xi) = \xi, g'(\xi) = 1 - b$.

$$\begin{aligned}\phi'(x) &= 1 - \frac{af'(x)f'(g(x)) - af(x)f''(g(x))g'(x)}{\{f'(g(x))\}^2} \\ \phi'(\xi) &= 1 - \frac{af'(\xi)f'(\xi)}{\{f'(\xi)\}^2} = 1 - a.\end{aligned}$$

Now, we choose $1 - a = 0$ so that $\phi'(\xi) = 0$.
Then

$$\begin{aligned}\phi'(x) &= 1 - \frac{f'(x)}{f'(g(x))} + \frac{f(x)f''(g(x))g'(x)}{\{f'(g(x))\}^2} \\ \phi''(x) &= -\frac{f''(x)f'(g(x)) - f'(x)f''(g(x))g'(x)}{\{f'(g(x))\}^2} \\ &\quad + \frac{f'(x)f''(g(x))g'(x)}{\{f'(g(x))\}^2} \\ &\quad + \frac{\{f(x)f'''(g(x))\{g'(x)\}^2 + f(x)f''(g(x))g''(x)\}\{f'(g(x))\}^2}{\{f'(g(x))\}^4} \\ &\quad - \frac{f(x)f''(g(x))g'(x) \cdot 2f'(g(x)) \cdot f''(g(x))g'(x)}{\{f'(g(x))\}^4} \\ \phi''(\xi) &= -\frac{f''(\xi)f'(\xi) - f'(\xi)f''(\xi)(1-b)}{\{f'(\xi)\}^2} \\ &\quad + \frac{f'(\xi)f''(\xi)(1-b)}{\{f'(\xi)\}^2} = \frac{f''(\xi)}{f'(\xi)}\{-1 + 2(1-b)\}.\end{aligned}$$

Now, $\phi''(\xi) = 0$ if $-1 + 2(1-b) = 0$ or, $b = 1/2$.
From the relation, $x_{n+1} = \phi(x_n)$ one can write

$$\varepsilon_{n+1} + \xi = \phi(\varepsilon_n + \xi) = \phi(\xi) + \varepsilon_n\phi'(\xi) + \frac{\varepsilon_n^2}{2}\phi''(\xi) + \frac{\varepsilon_n^3}{6}\phi'''(\xi) + \cdots$$

$$\text{or,} \quad \varepsilon_{n+1} = \varepsilon_n\phi'(\xi) + \frac{\varepsilon_n^2}{2}\phi''(\xi) + \frac{\varepsilon_n^3}{6}\phi'''(\xi) + \cdots$$

If $a = 1$ and $b = 1/2$ then $\phi'(\xi) = 0$ and $\phi''(\xi) = 0$. Then

$$\varepsilon_{n+1} \simeq \frac{\varepsilon_n^3}{6}\phi'''(\xi).$$

Hence the iteration scheme will have a third order convergence when $a = 1$ and $b = 1/2$.

4.13 Roots of Polynomial Equations

The polynomial of degree n is generally denoted by $P_n(x)$ and is defined as

$$P_n(x) \equiv a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n = 0 \tag{4.52}$$

where $a_0, a_1, \ldots, a_n$ are real coefficients.

A number ξ is a root of the polynomial $P_n(x)$ iff $P_n(x)$ is exactly divisible by $x - \xi$. If $P_n(x)$ is exactly divisible by $(x - \xi)^k$ $(k \geq 1)$, but is not divisible by $(x - \xi)^{k+1}$, then ξ is a k-fold root or a root of multiplicity k of the polynomial $P_n(x)$. The roots of multiplicity $k = 1$ are called **simple root** or **single root**.

The following theorem assures the existence of the roots of a polynomial equation.

Theorem 4.8 (The fundamental theorem of algebra). *Every polynomial equation with any numerical coefficients whose degree is not lower than unity has at least one root, real or complex.*

From this theorem it can be proved that:
Every polynomial $P_n(x)$ of degree n $(n \geq 1)$ with any numerical coefficients has exactly n roots, real or complex.

The roots of the equation (4.52) may be real or complex. If the coefficients of (4.52) are real and has a complex root $\alpha + i\beta$ of multiplicity k then (4.52) has a complex root $\alpha - i\beta$ also of multiplicity k.

The number of positive and negative roots of a polynomial equation can be determined by using **Descartes' rule** of signs:
The number of positive real roots of the algebraic equation $P_n(x) = 0$ with real coefficients either is equal to the number of sign changes in the sequence of the coefficients of the equation $P_n(x) = 0$ or is less than the number of sign changes by an even integer. The number of negative roots of the equation is equal to the number of sign changes in the sequence of coefficients of $P_n(-x)$ or is smaller by an even integer.

The **Sturm's theorem** gives the exact number of positive real roots of a polynomial equation.

Let $f(x)$ be a given polynomial of degree n and let $f_1(x)$ be its first order derivative. Let $f_2(x)$ be the remainder of $f(x)$ when it is divided by $f_1(x)$ taken with reverse sign.

Similarly, $f_3(x)$ is the remainder of $f_1(x)$ when it is divided by $f_2(x)$ with the reverse sign and so on. This division process is terminated when the quotient becomes a constant. Thus we obtain a sequence of function $f(x), f_1(x), f_2(x), \ldots, f_n(x)$ called the **Sturm functions** or the **Sturm sequences**.

Theorem 4.9 (Sturm). *The number of real roots of the equation $f(x) = 0$ on $[a, b]$ is equal to the difference between the number of changes of sign in the Sturm sequence at $x = a$ and $x = b$, provided $f(a) \neq 0$ and $f(b) \neq 0$.*

4.13.1 Domains of roots

Let the polynomial of degree n be

$$a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n = 0, \tag{4.53}$$

where $a_0, a_1, \ldots, a_n$ are real coefficients, and let $A = \max\{|a_1|, |a_2|, \ldots, |a_n|\}$ and $B = \max\{|a_0|, |a_1|, \ldots, |a_{n-1}|\}$. Then the roots of the equation (4.53) lie in the interval $r < |x| < R$ where

$$r = \frac{1}{1 + B/|a_n|} \qquad \text{and} \qquad R = 1 + \frac{A}{|a_0|}. \tag{4.54}$$

Here r is the lower bound and R is the upper bound of the positive roots of the equation (4.53) and $-R$ and $-r$ are the lower and the upper bounds of the negative roots respectively.

The Lagrange's or Newton's method may also be used to find the upper bound of the positive roots of the equation (4.53).

Theorem 4.10 (Lagrange's). *If the coefficients of the polynomial*

$$a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n = 0$$

satisfy the conditions $a_0 > 0, a_1, a_2, \ldots, a_{m-1} \geq 0, a_m < 0$, for some $m \leq n$, then the upper bound of the positive roots of the equation is $1 + \sqrt[m]{B/a_0}$, where B is the greatest of the absolute values of the negative coefficients of the polynomial.

Theorem 4.11 (Newton's). *If for $x = c$ the polynomial*

$$f(x) \equiv a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n = 0$$

and its derivatives $f'(x), f''(x), \ldots$ assume positive values then c is the upper bound of the positive roots of the equation.

The roots of a polynomial equation can be determined in two techniques – iteration methods and direct methods. In this section, two iteration methods, viz., Birge-Vieta and Bairstow methods, and one direct method – Graeffe's root squaring method are discussed.

Iterative Methods

4.14 Birge-Vieta Method

This method is based on the Newton-Raphson method. Here a real number ξ is determined such that $(x-\xi)$ is a factor of the polynomial

$$P_n(x) = x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-1}x + a_n = 0. \tag{4.55}$$

Let $Q_{n-1}(x)$ and R be the quotient and remainder when $P_n(x)$ is divided by the factor $(x-\xi)$, where $Q_{n-1}(x)$ is a polynomial of degree $(n-1)$ of the form

$$Q_{n-1}(x) = x^{n-1} + b_1x^{n-2} + b_2x^{n-3} + \cdots + b_{n-2}x + b_{n-1}. \tag{4.56}$$

Thus

$$P_n(x) = (x-\xi)Q_{n-1}(x) + R. \tag{4.57}$$

The value of R depends on ξ. Now, the problem is to find the value of ξ starting from an initial guess x_0 such that $R(\xi) = 0$ or it is equivalent to

$$R(\xi) = P_n(\xi) = 0. \tag{4.58}$$

The value of ξ can be determined by Newton-Raphson method or any other method. The Newton-Raphson methods for (4.58) is

$$x_{k+1} = x_k - \frac{P_n(x_k)}{P_n'(x_k)}, \quad k = 0, 1, 2, \ldots. \tag{4.59}$$

The values of $P_n(x_k)$ and $P_n'(x_k)$ can be determined by **synthetic division**. To determine the values of $b_1, b_2, \ldots, b_{n-1}$ and R, comparing the coefficient of like powers of x on both sides of (4.57) and obtain the following relations.

$$\begin{array}{ll}
a_1 = b_1 - \xi & b_1 = a_1 + \xi \\
a_2 = b_2 - \xi b_1 & b_2 = a_2 + \xi b_1 \\
\quad\vdots & \quad\vdots \\
a_k = b_k - \xi b_{k-1} & b_k = a_k + \xi b_{k-1} \\
\quad\vdots & \quad\vdots \\
a_n = R - \xi b_{n-1} & R = a_n + \xi b_{n-1}
\end{array}$$

From (4.57),

$$P_n(\xi) = R = b_n \text{ (say)}. \tag{4.60}$$

Hence,

$$b_k = a_k + \xi b_{k-1}, k = 1, 2, \ldots, n, \text{ with } b_0 = 1. \tag{4.61}$$

Thus b_n is the value of P_n.

To determine the value of P_n', differentiating (4.57) with respect to x

$$P_n'(x) = (x - \xi)Q_{n-1}'(x) + Q_{n-1}(x)$$

That is,

$$P_n'(\xi) = Q_{n-1}(\xi) = \xi^{n-1} + b_1\xi^{n-2} + \cdots + b_{n-2}\xi + b_{n-1}. \tag{4.62}$$

Thus

$$P_n'(x_i) = x_i^{n-1} + b_1 x_i^{n-2} + \cdots + b_{n-2}x_i + b_{n-1}. \tag{4.63}$$

$P_n'(x)$ can be evaluated as $P_n(x)$ is evaluated. Differentiating (4.61) with respect to ξ and obtain

$$\frac{db_k}{d\xi} = b_{k-1} + \xi\frac{db_{k-1}}{d\xi}.$$

Let

$$\frac{db_k}{d\xi} = c_{k-1}. \tag{4.64}$$

Then the above relation becomes

$$c_{k-1} = b_{k-1} + \xi c_{k-2}$$

This gives

$$c_k = b_k + \xi c_{k-1}, \quad k = 1, 2, \ldots, n-1. \tag{4.65}$$

The relation $P_n(\xi) = R = b_n$ is differentiated (equation (4.60)) to get

$$P_n'(\xi) = \frac{dR}{d\xi} = \frac{db_n}{d\xi} = c_{n-1} \qquad \text{[using (4.64)]}.$$

Thus Newton-Raphson method becomes

$$x_{k+1} = x_k - \frac{b_n}{c_{n-1}}, \quad k = 0, 1, 2, \ldots. \tag{4.66}$$

This method is known as **Birge-Vieta** method.

The Table 4.2 is useful to determine b_k and c_k for hand calculations.

Table 4.2: Scheme to calculate b's and c's.

x_0	1	a_1	a_2	$\cdots$	a_{n-2}	a_{n-1}	a_n
		x_0	x_0b_1	$\cdots$	x_0b_{n-3}	x_0b_{n-2}	x_0b_{n-1}
x_0	1	b_1	b_2	$\cdots$	b_{n-2}	b_{n-1}	$b_n = R$
		x_0	x_0c_1	$\cdots$	x_0c_{n-3}	x_0c_{n-2}	
	1	c_1	c_2	$\cdots$	c_{n-2}	$c_{n-1} = P_n'(x_0)$	

Example 4.14.1 Find all the roots of the polynomial equation $x^4 - 8x^3 + 14.91x^2 + 9.54x - 25.92 = 0$. One root of the equation lies between 1 and 2.

Solution. Let the polynomial be denoted by $P_4(x)$. Also, let the initial guess be $x_0 = 1.2$.

1.2	1	–8	14.91	9.54	–25.92
		1.2	–8.16	8.10	21.168
1.2	1	–6.8	6.75	17.64	–4.752=$b_4 = P_4(x_0)$
		1.2	–6.72	0.036	
	1	–5.6	0.03	17.676=$c_3 = P_4'(x_0)$	

Therefore,

$$x_1 = x_0 - \frac{b_4}{c_3} = 1.2 - \frac{-4.752}{17.676} = 1.46884.$$

1.46884	1	–8	14.91	9.54	–25.92
		1.46884	–9.59323	7.80949	25.48362
1.46884	1	–6.53116	5.31677	17.34949	–0.43638=b_4
		1.46884	–7.43574	–3.11243	
	1	–5.06232	–2.11897	14.23706=c_3	

Then $x_2 = x_1 - \dfrac{b_4}{c_3} = 1.46884 - \dfrac{-0.43638}{14.23706} = 1.49949.$

1.49949	1	–8	14.91	9.54	–25.92
		1.49949	–9.74745	7.74119	25.91298
1.49949	1	–6.50051	5.16255	17.28119	–0.00702=b_4
		1.49949	–7.49898	–3.50345	
	1	–5.00102	–2.33643	13.77774=c_3	

Then $x_3 = x_2 - \dfrac{b_4}{c_3} = 1.49949 - \dfrac{-0.00702}{13.77774} = 1.50000.$

Therefore, one root is 1.50000, which is an exact root. The reduce polynomial is

$$x^3 - 6.50051x^2 + 5.16255x + 17.28119 = 0.$$

One root of this equation lies between 4 and 5. Let $x_0 = 4.0$.

$$\begin{array}{r|rrrr}
4 & 1 & -6.50051 & 5.16255 & 17.28119 \\
 & & 4 & -10.00204 & -19.35796 \\ \hline
4 & 1 & -2.50051 & -4.83949 & -2.07677 = b_3 \\
 & & 4 & 5.99796 & \\ \hline
 & 1 & 1.49949 & 1.15847 = c_2 &
\end{array}$$

Therefore,

$$x_1 = x_0 - \frac{b_3}{c_2} = 4 - \frac{-2.07677}{1.15847} = 5.79268.$$

$$\begin{array}{r|rrrr}
5.79268 & 1 & -6.50051 & 5.16255 & 17.28119 \\
 & & 5.79268 & -4.10023 & -6.15366 \\ \hline
5.79268 & 1 & -0.70783 & 1.06232 & 23.43485 = b_3 \\
 & & 5.79268 & 29.45491 & \\ \hline
 & 1 & 5.08485 & 30.51723 = c_2 &
\end{array}$$

$$x_2 = x_1 - \frac{b_3}{c_2} = 5.79268 - \frac{23.43485}{30.51723} = 5.02476.$$

$$\begin{array}{r|rrrr}
5.02476 & 1 & -6.50051 & 5.16255 & 17.28119 \\
 & & 5.02476 & -7.41529 & -11.31948 \\ \hline
5.02476 & 1 & -1.47575 & -2.25274 & 5.96171 = b_3 \\
 & & 5.02476 & 17.83292 & \\ \hline
 & 1 & 3.54901 & 15.58018 = c_2 &
\end{array}$$

$$x_3 = x_2 - \frac{b_3}{c_2} = 5.02476 - \frac{5.96171}{15.58018} = 4.64211.$$

$$\begin{array}{r|rrrr}
4.64211 & 1 & -6.50051 & 5.16255 & 17.28119 \\
 & & 4.64211 & -8.62690 & -16.08188 \\ \hline
4.64211 & 1 & -1.85840 & -3.46435 & 1.19931 = b_3 \\
 & & 4.64211 & 12.92229 & \\ \hline
 & 1 & 2.78371 & 9.45794 = c_2 &
\end{array}$$

$$x_4 = x_3 - \frac{b_3}{c_2} = 4.64211 - \frac{1.19931}{9.45794} = 4.51531.$$

We take $x = 4.5$ as another root. The next reduced equation is

$$x^2 - 1.85840x - 3.46435 = 0$$

and roots of this equation are

$$x = 3.00953, -1.15113.$$

Hence the roots of the given equation are

$$1.5, 4.5, 3.00953, -1.15113.$$

But, the roots 3.00953 and 1.15113 contain some errors, because we approximate 4.51531 as 4.5. Some more iterations are required to achieve the root 4.5 and the correct quotient polynomial.

Algorithm 4.6 (Birge-Vieta method). This algorithm finds a root of a polynomial equation when the coefficients $a_1, a_2, \dots, a_n$ and an initial guess x_0 are given.

Algorithm Birge-Vieta

//n is the degree of the polynomial and N is the maximum number of iterations to be performed. Assume that the leading coefficient is one. ε is the error tolerance.//

```
Read $x_0, \varepsilon, n, N$;
Read $a_i, i = 1, 2, \dots, n$;

for $i = 1$ to $N$ do
    Set $b_0 = 1, c_0 = 1$;
    for $k = 1$ to $n$ do
        Compute $b_k = a_k + x_0 b_{k-1}$;
    endfor;
    for $k = 1$ to $n - 1$ do
        Compute $c_k = b_k + x_0 c_{k-1}$;
    endfor;
    Compute $x_1 = x_0 - \frac{b_n}{c_{n-1}}$;
    if $|x_1 - x_0| < \varepsilon$ then
        Print'One root is', $x_1$;
        Stop;
    else
        Set $x_0 = x_1$;
    endif;
endfor;
Print 'Root not found in $N$ iterations';
end Birge-Vieta
```

Program 4.6

```
/* Program Birge-Vieta for polynomial equation
   Program to find a root of the polynomial equation by
   Birge-Vieta method. Leading coefficient is 1 */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void main()
{
 int n, N,i,k;
 float x0,x1,a[10],b[10],c[10];
 float epp=1e-5; /* error tolerance */
 printf("\nEnter the degree of the polynomial ");
 scanf("%d",&n);
 printf("Enter the coefficients of the polynomial,
         except leading coeff.");
 for(i=1;i<=n;i++) scanf("%f",&a[i]);
 printf("Enter the initial guess x0 ");
 scanf("%f",&x0);
 printf("Enter maximum number of iterations to be done ");
 scanf("%d",&N);
 b[0]=1; c[0]=1;
 for(i=1;i<=N;i++)
  {
   for(k=1;k<=n;k++) b[k]=a[k]+x0*b[k-1];
   for(k=1;k<=n-1;k++) c[k]=b[k]+x0*c[k-1];
   x1=x0-b[n]/c[n-1];
   if(fabs(x1-x0)<epp)
    {
     printf("One root is %8.5f obtained at %d iterations",x1,i);
     printf("\nCoefficients of the reduced polynomial are\n ");
     for(k=0;k<=n-1;k++) printf("%f ",b[k]);
     exit(0);
    }
  else
    x0=x1;
 } /* i loop */
printf("\n Root not found at %d iterations ",N);
} /* main */
```

A sample of input/output:

```
Enter the degree of the polynomial 4
Enter the coefficients of the polynomial, except leading coeff.
-3 0 6 -4
Enter the initial guess x0 1
Enter maximum number of iterations to be done 100
One root is  1.00000 obtained at 1 iterations
Coefficients of the reduced polynomial are
 1.000000 -2.000000 -2.000000 4.000000
```

4.15 Bairstow Method

The roots of a polynomial equation can also be determined by extracting a quadratic factor from the polynomial. The roots (real or complex) of a quadratic equation can determine using a known closed form formula. The **Bairstow method** is used to extract a quadratic factor.

Let the polynomial $P_n(x)$ of degree n be

$$x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-1}x + a_n = 0. \qquad (4.67)$$

Let x^2+px+q be a factor of (4.67). If (4.67) is divided by the factor x^2+px+q then we obtain a polynomial $Q_{n-2}(x)$ of degree $(n-2)$ and a remainder $Rx+S$ of degree one, where R and S are independent of x. Thus the polynomial $P_n(x)$ can be written as

$$P_n(x) = (x^2+px+q)Q_{n-2}(x) + Rx + S \qquad (4.68)$$

where

$$Q_{n-2}(x) = x^{n-2} + b_1x^{n-3} + \cdots + b_{n-3}x + b_{n-2}. \qquad (4.69)$$

The values of R and S depends on p and q. If x^2+px+q is a factor of $P_n(x)$ then R and S should be zero. Thus our problem is to determine the values of p and q such that

$$R(p,q) = 0 \qquad \text{and} \qquad S(p,q) = 0. \qquad (4.70)$$

These equations are two non-linear equations in p and q. The values of p and q can then be determined by Newton-Raphson method for two variables (see Section 4.17.3).

Let (p_t,q_t) be the true values of p and q and Δp, Δq be the corrections to p and q. Then

$$p_t = p + \Delta p \qquad \text{and} \qquad q_t = q + \Delta q.$$

Thus

$$R(p_t,q_t) = R(p+\Delta p, q+\Delta q) = 0 \qquad \text{and} \qquad S(p_t,q_t) = S(p+\Delta p, q+\Delta q) = 0.$$

Therefore, by Taylor's series

$$R(p+\Delta p, q+\Delta q) = R(p,q) + \Delta p\frac{\partial R}{\partial p} + \Delta q\frac{\partial R}{\partial q} + \cdots = 0$$
$$\text{and } S(p+\Delta p, q+\Delta q) = S(p,q) + \Delta p\frac{\partial S}{\partial p} + \Delta q\frac{\partial S}{\partial q} + \cdots = 0.$$

The derivatives are evaluated at (p,q). Neglecting the square and higher powers of Δp and Δq the above equations reduce to

$$\Delta p R_p + \Delta q R_q = -R \tag{4.71}$$
$$\Delta p S_p + \Delta q S_q = -S. \tag{4.72}$$

The values of Δp and Δq are thus given by

$$\Delta p = -\frac{RS_q - SR_q}{R_pS_q - R_qS_p}, \quad \Delta q = -\frac{SR_p - RS_p}{R_pS_q - R_qS_p}. \tag{4.73}$$

Now, we determine the coefficient of the polynomial $Q_{n-2}(x)$ and the expression for R and S in terms of p and q.

From equations (4.67)-(4.69)

$$\begin{aligned} &x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-1}x + a_n \\ &\quad = (x^2+px+q)(x^{n-2} + b_1x^{n-3} + \cdots + b_{n-3}x + b_{n-2}). \end{aligned} \tag{4.74}$$

Equating the coefficients of $x^n, x^{n-1}, \ldots$ on both sides, we get

$$\begin{array}{ll} a_1 = b_1 + p & b_1 = a_1 - p \\ a_2 = b_2 + pb_1 + q & b_2 = a_2 - pb_1 - q \\ \qquad\vdots & \qquad\vdots \\ a_k = b_k + pb_{k-1} + qb_{k-2} & b_k = a_k - pb_{k-1} - qb_{k-2} \\ \qquad\vdots & \qquad\vdots \\ a_{n-1} = R + pb_{n-2} + qb_{n-3} & R = a_{n-1} - pb_{n-2} - qb_{n-3} \\ a_n = S + qb_{n-2} & S = a_n - qb_{n-2}. \end{array} \tag{4.75}$$

In general,

$$b_k = a_k - pb_{k-1} - qb_{k-2}, \qquad k = 1, 2, \ldots, n \tag{4.76}$$

where $b_0 = 1, b_{-1} = 0$.

In this notation,

$$R = b_{n-1}, \qquad S = b_n + pb_{n-1}. \tag{4.77}$$

Thus R and S are available when b's are known. To determine the partial derivatives R_p, R_q, S_p and S_q, differentiating (4.76) with respect to p and q.

$$\frac{\partial b_k}{\partial p} = -b_{k-1} - p\frac{\partial b_{k-1}}{\partial p} - q\frac{\partial b_{k-2}}{\partial p}, \qquad \frac{\partial b_0}{\partial p} = \frac{\partial b_{-1}}{\partial p} = 0 \tag{4.78}$$

$$\frac{\partial b_k}{\partial q} = -b_{k-2} - p\frac{\partial b_{k-1}}{\partial q} - q\frac{\partial b_{k-2}}{\partial q}, \qquad \frac{\partial b_0}{\partial q} = \frac{\partial b_{-1}}{\partial q} = 0 \tag{4.79}$$

Denoting

$$\frac{\partial b_k}{\partial p} = -c_{k-1}, \qquad k = 1, 2, \ldots, n \tag{4.80}$$

$$\text{and } \frac{\partial b_k}{\partial q} = -c_{k-2}. \tag{4.81}$$

Then (4.78) becomes

$$c_{k-1} = b_{k-1} - pc_{k-2} - qc_{k-3} \tag{4.82}$$

and (4.79) becomes

$$c_{k-2} = b_{k-2} - pc_{k-3} - qc_{k-4}. \tag{4.83}$$

Thus the recurrence relation to determine c_k using b_k is

$$c_k = b_k - pc_{k-1} - qc_{k-2}, k = 1, 2, \ldots, n-1 \text{ and } c_0 = 1, c_{-1} = 0. \tag{4.84}$$

Therefore,

$$\begin{aligned}
R_p &= \frac{\partial b_{n-1}}{\partial p} = -c_{n-2} \\
S_p &= \frac{\partial b_n}{\partial p} + p\frac{\partial b_{n-1}}{\partial p} + b_{n-1} = b_{n-1} - c_{n-1} - pc_{n-2} \\
R_q &= \frac{\partial b_{n-1}}{\partial q} = -c_{n-3} \\
S_q &= \frac{\partial b_n}{\partial q} + p\frac{\partial b_{n-1}}{\partial q} = -(c_{n-2} + pc_{n-3}).
\end{aligned}$$

To find the explicit expression for Δp and Δq, substituting the above values in (4.73). Therefore,

$$\begin{aligned}
\Delta p &= -\frac{b_n c_{n-3} - b_{n-1}c_{n-2}}{c_{n-2}^2 - c_{n-3}(c_{n-1} - b_{n-1})} \\
\Delta q &= -\frac{b_{n-1}(c_{n-1} - b_{n-1}) - b_n c_{n-2}}{c_{n-2}^2 - c_{n-3}(c_{n-1} - b_{n-1})}.
\end{aligned} \tag{4.85}$$

Therefore, the improved values of p and q are $p + \Delta p$ and $q + \Delta q$. Thus if p_0, q_0 be the initial values of p and q then the improved values are

$$p_1 = p_0 + \Delta p \qquad \text{and} \qquad q_1 = q_0 + \Delta q. \tag{4.86}$$

The values of b_k's and c_k's may be calculated using the following scheme: (when p_0 and q_0 are taken as initial values of p and q).

$$\begin{array}{r|cccccccc}
 & 1 & a_1 & a_2 & \cdots & a_k & \cdots & a_{n-1} & a_n \\
-p_0 & & -p_0 & -p_0b_1 & \cdots & -p_0b_{k-1} & \cdots & -p_0b_{n-2} & -p_0b_{n-1} \\
-q_0 & & & -q_0 & \cdots & -q_0b_{k-2} & \cdots & -q_0b_{n-3} & -q_0b_{n-2} \\
\hline
 & 1 & b_1 & b_2 & \cdots & b_k & \cdots & b_{n-1} & b_n \\
-p_0 & & -p_0 & -p_0c_1 & \cdots & -p_0c_{k-1} & \cdots & -p_0c_{n-2} & \\
-q_0 & & & -q_0 & \cdots & -q_0c_{k-2} & \cdots & -q_0c_{n-3} & \\
\hline
 & 1 & c_1 & c_2 & \cdots & c_k & \cdots & c_{n-1} &
\end{array}$$

Once p_1 and q_1 are evaluated, the next improved values p_2, q_2 are determined from the relation

$$p_2 = p_1 + \Delta p, \qquad q_2 = q_1 + \Delta q.$$

In general,

$$p_{k+1} = p_k + \Delta p, \qquad q_{k+1} = q_k + \Delta q, \tag{4.87}$$

the values of Δp and Δq are determined at $p = p_k$ and $q = q_k$.

The repetition is to be terminated when p and q have been obtained to the desired accuracy.

The polynomial

$$Q_{n-2}(x) = P_n(x)/(x^2 + px + q) = x^{n-2} + b_1x^{n-3} + \cdots + b_{n-3}x + b_{n-2}$$

is called the **deflated polynomial**. The next quadratic polynomial can be obtained in similar process from the deflated polynomial.

The rate of convergence of this method is quadratic as the computations of Δp and Δq are based on Newton-Raphson method.

Example 4.15.1 Extract a quadratic factor using the Bairstow method from the equation

$$x^4 + 4x^3 - 7x^2 - 22x + 24 = 0.$$

Solution. Let the initial guess of p and q be $p_0 = 0.5$ and $q_0 = 0.5$. Then

$$\begin{array}{r|rrrrl}
 & 1.00000 & 4.00000 & -7.00000 & -22.00000 & 24.00000 \\
-0.5 & & -0.50000 & -1.75000 & 4.62500 & 9.56250 \\
-0.5 & & & -0.50000 & -1.75000 & 4.62500 \\
\hline
 & 1.00000 & 3.50000 = b_1 & -9.25000 & -19.12500 & 38.18750 = b_4 \\
-0.5 & & -0.50000 & -1.50000 & 5.62500 & \\
-0.5 & & & -0.50000 & -1.50000 & \\
\hline
 & 1.00000 & 3.00000 & -11.25000 & -15.00000 & \\
 & & = c_1 & = c_2 & = c_3 &
\end{array}$$

$$\Delta p = -\frac{b_4c_1 - b_3c_2}{c_2^2 - c_1(c_3 - b_3)} = 0.88095, \qquad \Delta q = -\frac{b_3(c_3 - b_3) - b_4c_2}{c_2^2 - c_1(c_3 - b_3)} = -3.07143$$

Therefore, $p_1 = p_0 + \Delta p = 1.38095$, $q_1 = q_0 + \Delta q = -2.57143$.

Second iteration

	1.00000	4.00000	−7.00000	−22.00000	24.00000
−1.38095		−1.38095	−3.61678	11.11025	5.73794
2.57143			2.57143	6.73469	−20.68805
	1.00000	2.61905	−8.04535	−4.15506	9.04989
−1.38095		−1.38095	−1.70975	9.92031	
2.57143			2.57143	3.18367	
	1.00000	1.23810	−7.18367	8.94893	

$\Delta p = 0.52695$, $\Delta q = -0.29857$.
$p_2 = p_1 + \Delta p = 1.90790, q_2 = q_1 + \Delta q = -2.86999$.

Third iteration

	1.00000	4.00000	−7.00000	−22.00000	24.00000
−1.90790		−1.90790	−3.99152	15.49504	0.95517
2.86999		2.86999	6.00432	−23.30873	
	1.00000	2.09210	−8.12152	−0.50064	1.64644
−1.90790		−1.90790	−0.35144	10.68990	
2.86999			2.86999	0.52866	
	1.00000	0.18420	−5.60297	10.71793	

$\Delta p = 0.08531$, $\Delta q = -0.12304$.
$p_3 = p_2 + \Delta p = 1.99321, q_3 = q_2 + \Delta q = -2.99304$.

Fourth iteration

	1.00000	4.00000	−7.00000	−22.00000	24.00000
−1.99321		−1.99321	−3.99995	15.95943	0.06808
2.99304			2.99304	6.00642	−23.96501
	1.00000	2.00679	−8.00692	−0.03416	0.10307
−1.99321		−1.99321	−0.02709	10.04768	
2.99304			2.99304	0.04067	
	1.00000	0.01359	−5.04097	10.05419	

$\Delta p = 0.00676$, $\Delta q = -0.00692$.
$p_4 = p_3 + \Delta p = 1.99996, q_4 = q_3 + \Delta q = -2.99996$.

<u>Fifth iteration</u>

	1.00000	4.00000	−7.00000	−22.00000	24.00000
−1.99996		−1.99996	−4.00000	15.99978	0.00037
2.99996			2.99996	6.00004	−23.99981
	1.00000	2.00004	−8.00004	−0.00018	0.00055
−1.99996		−1.99996	−0.00015	10.00027	
2.99996			2.99996	0.00023	
	1.00000	0.00008	−5.00023	10.00031	

$\Delta p = 0.00004, \quad \Delta q = -0.00004.$
$p_5 = p_4 + \Delta p = 2.00000, q_5 = q_4 + \Delta q = -3.00000.$
Therefore, a quadratic factor is $x^2 + 2x - 3$ which is equal to $(x-1)(x+3)$. The deflated polynomial is $Q_2(x) = x^2 + 2.00004x - 8.00004 \simeq x^2 + 2x - 8$.
Thus $P_4(x) = (x-1)(x+3)(x^2+2x-8) = (x-1)(x+3)(x-2)(x+4)$.
Hence the roots of the given equation are $1, -3, 2, -4$.

Algorithm 4.7 (Bairstow method). This algorithm extracts a quadratic factor from a polynomial of degree n and also determines the deflated polynomial, by Bairstow method.

Algorithm Bairstow

```
// Extract a quadratic factor $x^2 + px + q$ from a polynomial $P_n(x) = x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ of degree $n$ and determines the deflated polynomial $Q_{n-2}(x) = x^{n-2} + b_1x^{n-3} + b_2x^{n-4} + \cdots + b_{n-2}$.//
Read $n, a_1, a_2, \ldots, a_n$;      //the degree and the coefficients of the polynomial.//
Read $p, q, \varepsilon$;      //the initial guess of $p, q$ and error tolerance.//
Set $b_0 = 1, b_{-1} = 0, c_0 = 1, c_{-1} = 0$;
//Compute $b_k$ and $c_k$
1. for $k = 1$ to $n$ do
       Compute $b_k = a_k - pb_{k-1} - qb_{k-2}$;
endfor;
for $k = 1$ to $n-1$ do
       Compute $c_k = b_k - pc_{k-1} - qc_{k-2}$;
endfor;
Compute $\Delta p = -\frac{b_n c_{n-3} - b_{n-1}c_{n-2}}{c_{n-2}^2 - c_{n-3}(c_{n-1} - b_{n-1})}$;      $\Delta q = -\frac{b_{n-1}(c_{n-1} - b_{n-1}) - b_n c_{n-2}}{c_{n-2}^2 - c_{n-3}(c_{n-1} - b_{n-1})}$;
Compute $p_{new} = p + \Delta p, \quad q_{new} = q + \Delta q$;
if ($|p_{new} - p| < \varepsilon$) and ($|q_{new} - q| < \varepsilon$) then
       Print 'The values of $p$ and $q$ are', $p_{new}, q_{new}$;
       Stop;
endif;
Set $p = p_{new}, q = q_{new}$;
```

```
go to 1.
Print 'The coefficients of deflated polynomial are',$b_1, b_2, \ldots, b_{n-2}$;
end Bairstow
```

Program 4.7

```
/* Program Bairstow for polynomial equation
   Program to find all the roots of a polynomial equation by
   Bairstow method. Leading coefficient is 1. Assume
   initial guess for all p and q are 0.5, 0.5.  */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void main()
{
 int n,i,k;
 float p,q,pnew,qnew,a[10],b[10],c[10],bm1,cm1,delp,delq;
 float epp=1e-5; /* error tolerance */
 void findroots(float p, float q);
 printf("\nEnter the degree of the polynomial ");
 scanf("%d",&n);
 printf("Enter the coefficients of the polynomial,
         except leading coeff.");
 for(i=1;i<=n;i++) scanf("%f",&a[i]);
 q=0.5;
 p=0.5;
 printf("The roots are \n");
 do
 {
    b[0]=1; bm1=0; c[0]=1; cm1=0;
    pnew=p; qnew=q;
    do{
       p=pnew; q=qnew;
       b[1]=a[1]-p*b[0]-q*bm1;
       c[1]=b[1]-p*c[0]-q*cm1;
       for(k=2;k<=n;k++) b[k]=a[k]-p*b[k-1]-q*b[k-2];
       for(k=2;k<=n;k++) c[k]=b[k]-p*c[k-1]-q*c[k-2];
       delp=-(b[n]*c[n-3]-b[n-1]*c[n-2])/
               (c[n-2]*c[n-2]-c[n-3]*(c[n-1]-b[n-1]));
       delq=-(b[n-1]*(c[n-1]-b[n-1])-b[n]*c[n-2])/
               (c[n-2]*c[n-2]-c[n-3]*(c[n-1]-b[n-1]));
```

```
        pnew=p+delp;
        qnew=q+delq;
     }while((fabs(pnew-p)>epp || fabs(qnew-q)>epp));
     findroots(p,q);
     n-=2;
     for(i=1;i<=n;i++) a[i]=b[i];
 }while(n>2);

   /* deflated polynomial is quadratic */
 if(n==2) findroots(b[1],b[2]);

   /* deflated polynomial is linear */
 if(n==1) printf("%f ",-b[1]);
} /* main */

/* finds the roots of the quadratic x*x+px+q=0 */
void findroots(float p, float q)
{
 float dis;
 dis=p*p-4*q;
 if(dis>=0)
    {
       printf("%f  %f\n",(-p+sqrt(dis))/2,(-p-sqrt(dis))/2);
    }
 else
    {
       printf("(%f,%f), (%f,%f)\n",-p/2,sqrt(fabs(dis))/2,
         -p/2,-sqrt(fabs(dis))/2);
    }
} /* findroots */
```

A sample of input/output:

```
Enter the degree of the polynomial 5
Enter the coefficients of the polynomial, except leading coeff.
3 4 -5 6 1
The roots are
(0.604068,0.729697), (0.604068,-0.729697)
(-2.030660,1.861934), (-2.030660,-1.861934)
-0.146816
```

Direct Method

4.16 Graeffe's Root Squaring Method

This method may be used to find all the roots of all types (real, equal or complex) of a polynomial equation with real coefficients. In this method, an equation is constructed whose roots are squares of the roots of the given equation, then another equation whose roots are squares of the roots of this new equation and so on, the process of root-squaring being continued as many times as necessary.

Let the given equation be

$$x^n + a_1x^{n-1} + a_2x^{n-2} + \cdots + a_{n-1}x + a_n = 0 \tag{4.88}$$

whose roots are $\xi_1, \xi_2, \ldots, \xi_n$.

Now, an equation is constructed whose roots are $-\xi_1^2, -\xi_2^2, \ldots, -\xi_n^2$ using the following technique.

Separating the even and odd powers of x in (4.88), and squaring both sides

$$(x^n + a_2x^{n-2} + a_4x^{n-4} + \cdots)^2 = (a_1x^{n-1} + a_3x^{n-3} + \cdots)^2.$$

After simplification the above equation becomes

$$x^{2n} - (a_1^2 - 2a_2)x^{2n-2} + (a_2^2 - 2a_1a_3 + 2a_4)x^{2n-4} + \cdots + (-1)^n a_n^2 = 0. \tag{4.89}$$

Setting $x^2 = z$ to the above equation and let it be

$$z^n + b_1z^{n-1} + \cdots + b_{n-1}z + b_n = 0$$

where

$$\begin{aligned} b_1 &= a_1^2 - 2a_2 \\ b_2 &= a_2^2 - 2a_1a_3 + 2a_4 \\ &\vdots \\ b_k &= a_k^2 - 2a_{k-1}a_{k+1} + 2a_{k-2}a_{k+2} - \cdots \qquad (4.90) \\ &\vdots \\ b_n &= a_n^2. \end{aligned}$$

The roots of the equation (4.89) are $-\xi_1^2, -\xi_2^2, \ldots, -\xi_n^2$. The coefficients b_k's can be obtained from Table 4.3.

The $(k+1)$th column i.e., b_k of Table 4.3 can be obtained as follows:
The terms alternate in sign starting with a positive sign. The first term is a_k^2. The second term is twice the product of a_{k-1} and a_{k+1}. The third term is twice the product

Table 4.3: Graeffe's root-squaring scheme.

1	a_1	a_2	a_3	a_4	$\cdots$	a_n
1	a_1^2	a_2^2	a_3^2	a_4^2	$\cdots$	a_n^2
	$-2a_2$	$-2a_1a_3$	$-2a_2a_4$	$-2a_3a_5$	$\cdots$	
		$2a_4$	$2a_1a_5$	$2a_2a_6$	$\cdots$	
			$-2a_6$	$-a_1a_7$	$\cdots$	
				$2a_8$	$\cdots$	
1	b_1	b_2	b_3	b_4	$\cdots$	b_n

of a_{k-2} and a_{k+2}. This process is continued until there are no available coefficients to form the cross product terms.

The root-squaring process is repeated to a sufficient number of times, say m times and we obtain the equation

$$x^n + c_1x^{n-1} + c_2x^{n-2} + \cdots + c_{n-1}x + c_n = 0. \tag{4.91}$$

Let the roots of (4.91) be $\alpha_1, \alpha_2, \ldots, \alpha_n$. This roots are the 2^mth power of the roots of the equation (4.88) with opposite signs, i.e.,

$$\alpha_i = -\xi_i^{2^m}, \quad i = 1, 2, \ldots, n.$$

The relation between roots and coefficients gives

$$\begin{aligned}
-c_1 &= \alpha_1 + \alpha_2 + \cdots + \alpha_n = \sum \alpha_i = -\sum \xi_i^p \\
c_2 &= \alpha_1\alpha_2 + \alpha_1\alpha_3 + \cdots + \alpha_{n-1}\alpha_n = \sum_{i<j} \alpha_i\alpha_j = \sum_{i<j} \xi_i^p\xi_j^p \\
-c_3 &= \sum_{i<j<k} \alpha_i\alpha_j\alpha_k = -\sum_{i<j<k} \xi_i^p\xi_j^p\xi_k^p \\
\cdots\ \cdots &\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
(-1)^n c_n &= \alpha_1\alpha_2\cdots\alpha_n = (-1)^n\xi_1^p\xi_2^p\cdots\xi_n^p, \text{ where } p = 2^m.
\end{aligned} \tag{4.92}$$

In the following different cases are considered separately.

Case I. *Roots are real and unequal in magnitudes.*
Let us consider

$$|\xi_n| < |\xi_{n-1}| < \cdots < |\xi_2| < |\xi_1|$$

then

$$|\alpha_n| \ll |\alpha_{n-1}| \ll \cdots \ll |\alpha_2| \ll |\alpha_1|.$$

That is,

$$|\xi_n|^{2^m} \ll |\xi_{n-1}|^{2^m} \ll \cdots \ll |\xi_2|^{2^m} \ll |\xi_1|^{2^m} \tag{4.93}$$

since all the roots are widely separated in magnitude at the final stage.

Then from (4.92),

$$c_1 = \xi_1^p \left[1 + \left(\frac{\xi_2}{\xi_1}\right)^p + \left(\frac{\xi_3}{\xi_1}\right)^p + \cdots + + \left(\frac{\xi_n}{\xi_1}\right)^p \right] \simeq \xi_1^p$$

as $\left(\dfrac{\xi_i}{\xi_1}\right)^p \to 0$, $i > 1$ at the desired level of accuracy.

Similarly,

$$c_2 = (\xi_1\xi_2)^p \left[1 + \left(\frac{\xi_1\xi_3}{\xi_1\xi_2}\right)^p + \cdots + \left(\frac{\xi_{n-1}\xi_n}{\xi_1\xi_2}\right)^p \right] \simeq (\xi_1\xi_2)^p$$

and so on, finally,

$$c_n = (\xi_1\xi_2\cdots\xi_n)^p.$$

Thus at the desired level of accuracy

$$|\xi_1| = c_1^{1/p}, \;\; |\xi_2| = (c_2/c_1)^{1/p}, \;\; |\xi_n| = (c_n/c_{n-1})^{1/p}, \;\; p = 2^m. \tag{4.94}$$

This determines the absolute values of the roots. By substituting these values in the original equation (4.88) one can determine the sign of the roots. The squaring process is terminated when another squaring process produces new coefficients that are almost the squares of the corresponding coefficients c_k's i.e., when the cross product terms become negligible with respect to square terms. Thus the final stage is identified by the fact that on root-squaring at that stage all the cross products will vanish.

Case II. *All roots are real with one pair of equal magnitude.*
Let $\xi_1, \xi_2, \ldots, \xi_n$ be the roots of the given equation, if a pair of roots are equal in magnitude then this pair is conveniently called a double root. A double root can be identified in the following way:

If the magnitude of the coefficient c_k is about half the square of the magnitude of the corresponding coefficient in the previous equation, then it indicates that ξ_k is a double root. The double root is determined by the following process.

We have

$$\alpha_k \simeq -\frac{c_k}{c_{k-1}} \qquad \text{and} \qquad \alpha_{k+1} \simeq -\frac{c_{k+1}}{c_k}.$$

Then

$$\alpha_k\alpha_{k+1} \simeq \alpha_k^2 \simeq \left|\frac{c_{k+1}}{c_{k-1}}\right|.$$

Therefore,

$$|\alpha_k^2| = |\xi_k|^{2(2^m)} = \left|\frac{c_{k+1}}{c_{k-1}}\right|.$$

Thus the roots are given by

$$\begin{aligned}&|\xi_1| = c_1^{1/p}, |\xi_2| = (c_2/c_1)^{1/p}, \ldots, |\xi_{k-1}| = (c_{k-1}/c_{k-2})^{1/p}, \ldots,\\ &|\xi_k| = (c_{k+1}/c_{k-1})^{1/(2p)}, \ldots, |\xi_n| = (c_n/c_{n-1})^{1/p},\\ &\text{where } p = 2^m.\end{aligned} \tag{4.95}$$

This gives the magnitude of the double root. The sign is determined by substituting the root to the equation.

The double root can also be determined directly since α_k and α_{k+1} converge to the same root after sufficient squaring. Generally, the rate of convergence to the double root is slow.

Case III. *One pair of complex roots and other roots are distinct in magnitude.*

Let ξ_k and ξ_{k+1} form a complex pair and let

$$\xi_k, \xi_{k+1} = \rho_k e^{\pm i\theta_k}$$

where $\rho_k = |\xi_k| = |\xi_{k+1}|$.

For sufficiently large m, ρ_k can be determined from the previous case,

$$|\alpha_k^2| \simeq \left|\frac{c_{k+1}}{c_{k-1}}\right| \qquad \text{or,} \qquad \rho_k^{2p} \simeq \left|\frac{c_{k+1}}{c_{k-1}}\right|, \text{ where } p = 2^m.$$

and θ_k is determined from the relation

$$2\rho_k^m \cos m\theta_k \simeq \frac{c_{k+1}}{c_{k-1}}.$$

Thus the roots are given by

$$|\xi_1| = c_1^{1/p}, |\xi_2| = (c_2/c_1)^{1/p}, \ldots, |\xi_{k-1}| = (c_{k-1}/c_{k-2})^{1/p}, \rho_k^2 = \left(\frac{c_{k+1}}{c_{k-1}}\right)^{1/p},$$

$$|\xi_{k+2}| = (c_{k+2}/c_{k+1})^{1/p}, \ldots, |\xi_n| = (c_n/c_{n-1})^{1/p}, p = 2^m.$$

The real roots $\xi_1, \xi_2, \ldots, \xi_{k-1}, \xi_{k+2}, \ldots, \xi_n$ are then corrected for sign.

If the equation has only one pair of complex roots $\xi_k, \xi_{k+1} = u \pm iv$ then the sum of the roots is

$$\xi_1 + \xi_2 + \cdots + \xi_{k-1} + 2u + \xi_{k+2} + \cdots + \xi_n = -a_1.$$

From this relation one can determine the value of u. Then the value of v can be determined from the relation

$$v^2 = \rho_k^2 - u^2.$$

The presence of complex roots in $(k+1)$th column is identified by the following technique:
If the coefficients of x^{n-k} in the successive squaring to fluctuate both in magnitude and sign, a complex pair can be detected by this oscillation.

Merits and Demerits

1. All roots are found at the end of the method, i.e., at one execution of the method all the roots are determined, including complex roots.
2. No initial guess is required.
3. As a direct method, there is no scope for correcting the error generated in any stage. If any error is generated at any stage, then the error propagates to all the subsequent computations and ultimately gives a wrong result.
4. The method is laborious, and to get a very accurate result the method has to be repeated for a large number of times.
5. There is a chance for data overflow in computer.

In the following, three examples are considered to discuss the three possible cases of Graeffe's method. The following table is the Graeffe's root squaring scheme for four degree equation.

1	a_1	a_2	a_3	a_4
1	a_1^2	a_2^2	a_3^2	a_4^2
	$-2a_2$	$-2a_1a_3$	$-2a_2a_4$	
		$2a_4$		
1	c_1	c_2	c_3	c_4

Example 4.16.1 Find the roots of the equation

$$2x^4 - 15x^3 + 40x^2 - 45x + 18 = 0$$

correct up to four decimal places by Graeffe's root squaring method.

Solution. All the calculations are shown in the following table. The number within the parenthesis represents the exponent of the adjacent number. i.e., 0.75(02) means 0.75×10^2.

m	2^m	x^4	x^3	x^2	x	1
0	1	1.00000	−0.75000(01)	0.20000(02)	−0.22500(02)	0.90000(01)
		1.00000	0.56250(02)	0.40000(03)	0.50625(03)	0.81000(02)
			−0.40000(02)	−0.33750(03)	−0.36000(03)	
				0.18000(02)		
1	2	1.00000	0.16250(02)	0.80500(02)	0.14625(03)	0.81000(02)
		1.00000	0.26406(03)	0.64803(04)	0.21389(05)	0.65610(04)
			−0.16100(03)	−0.47531(04)	−0.13041(05)	
				0.16200(03)		
2	4	1.00000	0.10306(03)	0.18891(04)	0.83481(04)	0.65610(04)
		1.00000	0.10622(05)	0.35688(07)	0.69690(08)	0.43047(08)
			−0.37783(04)	−0.17207(07)	−0.24789(08)	
				0.13122(05)		
3	8	1.00000	0.68436(04)	0.18612(07)	0.44901(08)	0.43047(08)
		1.00000	0.46835(08)	0.34640(13)	0.20161(16)	0.18530(16)
			−0.37223(07)	−0.61457(12)	−0.16023(15)	
				0.86093(08)		
4	16	1.00000	0.43113(08)	0.28495(13)	0.18559(16)	0.18530(16)
		1.00000	0.18587(16)	0.81195(25)	0.34443(31)	0.34337(31)
			−0.56989(13)	−0.16002(24)	−0.10560(29)	
				0.37060(16)		
5	32	1.0000	0.18530(16)	0.79595(25)	0.34337(31)	0.34337(31)

This is the final equation since all the cross products vanish at the next step and all the roots are real and distinct in magnitude.
Therefore,

$$
\begin{aligned}
|\xi_1| &= (0.18530 \times 10^{16})^{1/32} = 3.0000,\\
|\xi_2| &= \left(\frac{0.79595 \times 10^{25}}{0.18530 \times 10^{16}}\right)^{1/32} = 2.0000,\\
|\xi_3| &= \left(\frac{0.34337 \times 10^{31}}{0.79595 \times 10^{25}}\right)^{1/32} = 1.5000,\\
|\xi_4| &= \left(\frac{0.34337 \times 10^{31}}{0.34337 \times 10^{31}}\right)^{1/32} = 1.0000.
\end{aligned}
$$

Hence the roots of the given equation are 3, 2, 1.5, 1.

Example 4.16.2 Find the roots of the equation $x^4 - 3x^3 + 6x - 4 = 0$ correct up to four decimal places by Graeffe's root squaring method.

Solution. The necessary calculations are shown below.

m	2^m	x^4	x^3	x^2	x	1
0	1	1.00000	-0.30000(01)	0.00000(00)	0.60000(01)	-0.40000(01)
		1.00000	0.90000(01)	0.00000(00)	0.36000(02)	0.16000(02)
			0.00000(00)	0.36000(02)	0.00000(00)	
				-0.80000(01)		
1	2	1.00000	0.90000(01)	0.28000(02)	0.36000(02)	0.16000(02)
		1.00000	0.81000(02)	0.78400(03)	0.12960(04)	0.25600(03)
			-0.56000(02)	-0.64800(03)	-0.89600(03)	
				0.32000(02)		
2	4	1.00000	0.25000(02)	0.16800(03)	0.40000(03)	0.25600(03)
		1.00000	0.62500(03)	0.28224(05)	0.16000(06)	0.65536(05)
			-0.33600(03)	-0.20000(05)	-0.86016(05)	
				0.51200(03)		
3	8	1.00000	0.28900(03)	0.87360(04)	0.73984(05)	0.65536(05)
		1.00000	0.83521(05)	0.76318(08)	0.54736(10)	0.42950(10)
			-0.17472(05)	-0.42763(08)	-0.11450(10)	
				0.13107(06)		
4	16	1.00000	0.66049(05)	0.33686(08)	0.43286(10)	0.42950(10)
		1.00000	0.43625(10)	0.11347(16)	0.18737(20)	0.18447(20)
			-0.67372(08)	-0.57180(15)	-0.28936(18)	
				0.85899(10)		
5	32	1.00000	0.42951(10)	0.56296(15)	0.18447(20)	0.18447(20)

The diminishing double products vanish at the next step and hence this is the final equation and since we find the characteristic behaviour of a double root in the second column.
Therefore,

$$\begin{aligned}
|\xi_1| &= (0.42951 \times 10^{10})^{1/32} = 2.0000, \\
|\xi_2| &= |\xi_3| = \left(\frac{0.18447 \times 10^{20}}{0.42951 \times 10^{10}}\right)^{1/64} = 1.4142, \\
|\xi_4| &= \left(\frac{0.18447 \times 10^{20}}{0.18447 \times 10^{20}}\right)^{1/32} = 1.0000.
\end{aligned}$$

Here 1.4142 as well as -1.4142 satisfied the given equation, hence the roots of the given equation are 2, ± 1.4142, 1.

Example 4.16.3 Find the roots of the equation $x^4-5x^3+x^2-7x+10=0$ correct up to four decimal places by Graeffe's root squaring method.

Solution. The necessary calculations are shown below.

m	2^m	x^4	x^3	x^2	x	1
0	1	1.00000	-0.50000(01)	0.10000(01)	-0.70000(01)	0.10000(02)
		1.00000	0.25000(02)	0.10000(01)	0.49000(02)	0.10000(03)
			-0.20000(01)	-0.70000(02)	-0.20000(02)	
				0.20000(02)		
1	2	1.00000	0.23000(02)	-0.49000(02)	0.29000(02)	0.10000(03)
		1.00000	0.52900(03)	0.24010(04)	0.84100(03)	0.10000(05)
			0.98000(02)	-0.13340(04)	0.98000(04)	
				0.20000(03)		
2	4	1.00000	0.62700(03)	0.12670(04)	0.10641(05)	0.10000(05)
		1.00000	0.39313(06)	0.16053(07)	0.11323(09)	0.10000(09)
			-0.25340(04)	-0.13344(08)	-0.25340(08)	
				0.20000(05)		
3	8	1.00000	0.39060(06)	-0.11719(08)	0.87891(08)	0.10000(09)
		1.00000	0.15256(12)	0.13732(15)	0.77248(16)	0.10000(17)
			0.23437(08)	-0.68659(14)	0.23437(16)	
				0.20000(09)		
4	16	1.00000	0.15259(12)	0.68665(14)	0.10069(17)	0.10000(17)
		1.00000	0.23283(23)	0.47148(28)	0.10137(33)	0.10000(33)
			-0.13733(15)	-0.30727(28)	-0.13733(31)	
				0.20000(17)		
5	32	1.00000	0.23283(23)	0.16422(28)	0.10000(33)	0.10000(33)

Since c_2 alternates in sign, it indicates that there is a pair of complex roots.

$$\begin{aligned}
|\xi_1| &= (.23283\times 10^{23})^{1/32} = 5.0000,\\
|\xi_4| &= \left(\frac{0.10000\times 10^{33}}{0.10000\times 10^{33}}\right)^{1/64} = 1.0000.
\end{aligned}$$

These two roots are positive.

$$\rho_2^2 = \left(\frac{0.10000\times 10^{33}}{0.23283\times 10^{23}}\right)^{1/32} = 2.0000.$$

If $\xi_2,\xi_3 = u\pm v$, then $2u+5+1=5$ (sum of the roots). Therefore, $u=-0.5$. Then $v=\sqrt{\rho_2^2-u^2}=\sqrt{2-0.25}=1.3229$.
Hence the roots are 5, 1, $-0.5\pm 1.3229i$.

4.17 Solution of Systems of Nonlinear Equations

To solve a system of nonlinear equations the following methods are discussed in this section.

1. The method of iteration (fixed point iteration)
2. Seidal iteration
3. Newton-Raphson method.

4.17.1 The method of iteration

Let the system of nonlinear equations be

$$\begin{aligned} f(x,y) &= 0 \\ \text{and} \qquad g(x,y) &= 0. \end{aligned} \tag{4.96}$$

whose real roots are required within a specified accuracy. The above system can be rewritten as

$$\begin{aligned} x &= F(x,y) \\ \text{and} \qquad y &= G(x,y). \end{aligned} \tag{4.97}$$

The function F and G may be obtained in many different ways.

Let (x_0, y_0) be the initial guess to a root (ξ, η) of the system (4.96). Then we obtain the following sequence $\{(x_n, y_n)\}$ of roots.

$$\begin{aligned} x_1 &= F(x_0,y_0), \quad y_1 = G(x_0,y_0) \\ x_2 &= F(x_1,y_1), \quad y_2 = G(x_1,y_1) \\ &\cdots\cdots\cdots\cdots \quad \cdots\cdots\cdots\cdots \\ x_{n+1} &= F(x_n,y_n), \; y_{n+1} = G(x_n,y_n). \end{aligned} \tag{4.98}$$

If the sequence (4.98) converges, i.e.,

$$\lim_{n\to\infty} x_n = \xi \qquad \text{and} \qquad \lim_{n\to\infty} y_n = \eta$$

then

$$\xi = F(\xi,\eta) \qquad \text{and} \qquad \eta = G(\xi,\eta). \tag{4.99}$$

Like the iteration process for single variable, the above sequence surely converge to a root under certain condition. The sufficient condition is stated below.

Theorem 4.12 *Assume that the functions $x = F(x,y), y = G(x,y)$ and their first order partial derivatives are continuous on a region R that contains a root (ξ,η). If the starting point (x_0,y_0) is sufficiently close to (ξ,η) and if*

$$\left|\frac{\partial F}{\partial x}\right| + \left|\frac{\partial F}{\partial y}\right| < 1 \qquad \textit{and} \qquad \left|\frac{\partial G}{\partial x}\right| + \left|\frac{\partial G}{\partial y}\right| < 1, \tag{4.100}$$

for all $(x,y) \in R$, then the iteration scheme (4.98) converges to a root (ξ,η).

The condition for the functions $x = F(x,y,z), y = G(x,y,z), z = H(x,y,z)$ is

$$\begin{aligned}
\left|\frac{\partial F}{\partial x}\right| + \left|\frac{\partial F}{\partial y}\right| + \left|\frac{\partial F}{\partial z}\right| &< 1,\\
\left|\frac{\partial G}{\partial x}\right| + \left|\frac{\partial G}{\partial y}\right| + \left|\frac{\partial G}{\partial z}\right| &< 1\\
\text{and} \left|\frac{\partial H}{\partial x}\right| + \left|\frac{\partial H}{\partial y}\right| + \left|\frac{\partial H}{\partial z}\right| &< 1
\end{aligned}$$

for all $(x,y,z) \in R$.

4.17.2 Seidal method

An improvement of the iteration method can be made by using the recently computed values of x_i while computing y_i, i.e., x_{i+1} is used in the calculation of y_i. Therefore, the iteration scheme becomes

$$\begin{aligned}
x_{n+1} &= F(x_n, y_n)\\
y_{n+1} &= G(x_{n+1}, y_n).
\end{aligned} \tag{4.101}$$

This method is called Seidal iteration. In case of three variables the scheme is

$$\begin{aligned}
x_{n+1} &= F(x_n, y_n, z_n)\\
y_{n+1} &= G(x_{n+1}, y_n, z_n)\\
\text{and } z_{n+1} &= H(x_{n+1}, y_{n+1}, z_n).
\end{aligned} \tag{4.102}$$

Example 4.17.1 Solve the following system of equations

$$x = \frac{8x - 4x^2 + y^2 + 1}{8} \text{ and } y = \frac{2x - x^2 + 4y - y^2 + 3}{4}$$

starting with $(x_0, y_0) = (1.1, 2.0)$, using (i) iteration method, and (ii) Seidal iteration method.

Solution.
(i) Iteration method
Let

$$F(x,y)=\frac{8x-4x^2+y^2+1}{8} \qquad \text{and} \qquad G(x,y)=\frac{2x-x^2+4y-y^2+3}{4}.$$

The iteration scheme is

$$\begin{aligned} x_{n+1} &= F(x_n,y_n)=\frac{8x_n-4x_n^2+y_n^2+1}{8}, \\ \text{and } y_{n+1} &= G(x_n,y_n)=\frac{2x_n-x_n^2+4y_n-y_n^2+3}{4}. \end{aligned}$$

The value of x_n, y_n, x_{n+1} and y_{n+1} for $n=0,1,\ldots$ are shown in the following table.

n	x_n	y_n	x_{n+1}	y_{n+1}
0	1.10000	2.00000	1.12000	1.99750
1	1.12000	1.99750	1.11655	1.99640
2	1.11655	1.99640	1.11641	1.99660
3	1.11641	1.99660	1.11653	1.99661
4	1.11653	1.99661	1.11652	1.99660

Therefore, a root correct up to four decimal place is $(1.1165, 1.9966)$.

(ii) Seidal method
The iteration scheme for Seidal method is

$$\begin{aligned} x_{n+1} &= F(x_n,y_n)=\frac{8x_n-4x_n^2+y_n^2+1}{8} \\ \text{and} \qquad y_{n+1} &= G(x_{n+1},y_n)=\frac{2x_{n+1}-x_{n+1}^2+4y_n-y_n^2+3}{4}. \end{aligned}$$

The calculations are shown below.

n	x_n	y_n	x_{n+1}	y_{n+1}
0	1.10000	2.00000	1.12000	1.99640
1	1.12000	1.99640	1.11600	1.99663
2	1.11600	1.99663	1.11659	1.99660
3	1.11659	1.99660	1.11650	1.99660

Therefore, root correct up to four decimal places is

$$(1.1165, 1.9966).$$

Algorithm 4.8 (Seidal iteration). This algorithm used to solve two non-linear equations by Seidal iteration, when the initial guess is given.

```
Algorithm Seidal-Iteration-2D
// Let $(x_0, y_0)$ be the initial guess of the system of equations $x = F(x, y)$, $y = G(x, y)$.
$\varepsilon$ be the error tolerance, $maxiteration$ represents the maximum number of repetitions
to be done.//
Input functions $F(x, y), G(x, y)$.
Read $\varepsilon$, $maxiteration$, $x_0, y_0, z_0$;
Set $k = 0, error = 1$;
While $k < maxiteration$ and $error > \varepsilon$ do
    Set $k = k + 1$;
    Compute $x_1 = F(x_0, y_0), y_1 = G(x_1, y_0)$;
    Compute $error = |x_1 - x_0| + |y_1 - y_0|$;
    Set $x_0 = x_1, y_0 = y_1$;
endwhile;
if $error < \varepsilon$ then
    Print ‘The sequence converge to the root’, $x_1, y_1$;
    Stop;
else
    Print ‘The iteration did not converge after’, $k$,’iterations’;
    Stop;
endif;
end Seidal-Iteration-2D
```

Program 4.8
```
/* Program Seidal for a pair of non-linear equations
   Program to find a root of a pair of non-linear equations
   by Seidal method. Assumed that the equations are given
   in the form x=f(x,y) and y=g(x,y).
   The equations taken are x*x+4y*y-4=0, x*x-2x-y+1=0. */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>

void main()
{
 int k=0,maxiteration;
 float error=1,eps=1e-5,x0,y0; /*initial guesses for x and y*/
 float x1,y1;
 float f(float x, float y);
 float g(float x, float y);
 printf("Enter initial guess for x and y ");
 scanf("%f %f",&x0,&y0);
 printf("Maximum iterations to be allowed ");
 scanf("%d",&maxiteration);
 while((k<maxiteration) && (error>eps))
   {
     k++;
     x1=f(x0,y0); y1=g(x1,y0);
     error=fabs(x1-x0)+fabs(y1-y0);
     x0=x1; y0=y1;
   }
 if(error<eps)
  {
   printf("The sequence converges to the\n");
   printf("root (%7.5f, %7.5f) at %d iterations",x1,y1,k);
   exit(0);
  }
 else
  {
   printf("The iteration did not converge after %d iterations",k);
   exit(0);
  }
} /* main */

/* definition of f(x,y) */
float f(float x, float y)
{
 float f1;
 f1=sqrt(4-4*y*y);
 return f1;
}

/* definition of g(x,y) */
float g(float x, float y)
{
 float g1;
 g1=x*x-2*x+1;
 return g1;
}
```

A sample of input/output:

```
Enter initial guess for x and y 0.5 0.5
Maximum iterations to be allowed 100
The sequence converges to the root (0.00000, 1.00000) at 68 iterations
```

4.17.3 Newton-Raphson method

Let (x_0, y_0) be an initial guess to the root (ξ, η) of the equations

$$f(x,y) = 0 \text{ and } g(x,y) = 0. \tag{4.103}$$

If $(x_0 + h, y_0 + k)$ is the root of the above system then

$$\begin{aligned} f(x_0+h, y_0+k) &= 0 \\ g(x_0+h, y_0+k) &= 0. \end{aligned} \tag{4.104}$$

Assume that f and g are differentiable. Expanding (4.104) by Taylor's series.

$$\begin{aligned} f(x_0,y_0) + h\left(\frac{\partial f}{\partial x}\right)_{(x_0,y_0)} + k\left(\frac{\partial f}{\partial y}\right)_{(x_0,y_0)} + \cdots = 0 \\ g(x_0,y_0) + h\left(\frac{\partial g}{\partial x}\right)_{(x_0,y_0)} + k\left(\frac{\partial g}{\partial y}\right)_{(x_0,y_0)} + \cdots = 0 \end{aligned} \tag{4.105}$$

Neglecting square and higher order terms, the above equations simplified as

$$\begin{aligned} h\frac{\partial f_0}{\partial x} + k\frac{\partial f_0}{\partial y} &= -f_0 \\ h\frac{\partial g_0}{\partial x} + k\frac{\partial g_0}{\partial y} &= -g_0 \end{aligned} \tag{4.106}$$

where $f_0 = f(x_0, y_0)$, $\dfrac{\partial f_0}{\partial x} = \left(\dfrac{\partial f}{\partial x}\right)_{(x_0,y_0)}$ etc.

The above system can be written as

$$\begin{bmatrix} \dfrac{\partial f_0}{\partial x} & \dfrac{\partial f_0}{\partial y} \\ \\ \dfrac{\partial g_0}{\partial x} & \dfrac{\partial g_0}{\partial y} \end{bmatrix} \begin{bmatrix} h \\ k \end{bmatrix} = \begin{bmatrix} -f_0 \\ -g_0 \end{bmatrix} \text{ or } \begin{bmatrix} h \\ k \end{bmatrix} = J^{-1} \begin{bmatrix} -f_0 \\ -g_0 \end{bmatrix}.$$

Alternatively, h and k can be evaluated as

$$h = \frac{1}{J}\begin{vmatrix} -f_0 & \dfrac{\partial f_0}{\partial y} \\ & \\ -g_0 & \dfrac{\partial g_0}{\partial y} \end{vmatrix}, \quad k = \frac{1}{J}\begin{vmatrix} \dfrac{\partial f_0}{\partial x} & -f_0 \\ & \\ \dfrac{\partial g_0}{\partial x} & -g_0 \end{vmatrix}, \text{ where } J = \begin{vmatrix} \dfrac{\partial f_0}{\partial x} & \dfrac{\partial f_0}{\partial y} \\ & \\ \dfrac{\partial g_0}{\partial x} & \dfrac{\partial g_0}{\partial y} \end{vmatrix}. \tag{4.107}$$

Thus h and k are determined by one of the above two ways. Therefore, the new approximations are then given by

$$x_1 = x_0 + h, \qquad y_1 = y_0 + k. \tag{4.108}$$

The process is to be repeated until the roots are achieved to the desired accuracy. The general formula is $x_{n+1} = x_n + h, y_{n+1} = y_n + k$; h, k are evaluated at (x_n, y_n) instead at (x_0, y_0).

If the iteration converges (the condition is stated below) then the rate of convergence is quadratic.

Theorem 4.13 *Let (x_0, y_0) be an initial guess to a root (ξ, η) of the system $f(x, y) = 0, g(x, y) = 0$ in a closed neighbourhood R containing (ξ, η). If*

1. *f, g and their first order partial derivatives are continuous and bounded in R, and*
2. *$J \neq 0$ in R, then the sequence of approximation $x_{n+1} = x_n + h, y_{n+1} = y_n + k$, where h and k are given by (4.107), converges to the root (ξ, η).*

Example 4.17.2 Use Newton-Raphson method to solve the system $x^2 - 2x - y + 0.5 = 0, x^2 + 4y^2 - 4 = 0$ with the starting value $(x_0, y_0) = (2.00, 0.25)$.

Solution. Let $f(x, y) = x^2 - 2x - y + 0.5$ and $g(x, y) = x^2 + 4y^2 - 4$.

$$\frac{\partial f}{\partial x} = 2x - 2, \frac{\partial f}{\partial y} = -1, \frac{\partial g}{\partial x} = 2x, \frac{\partial g}{\partial y} = 8y.$$

Therefore,

$$J = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x-2 & -1 \\ 2x & 8y \end{bmatrix}, \qquad \begin{bmatrix} -f_0 \\ -g_0 \end{bmatrix} = \begin{bmatrix} -0.25 \\ -0.25 \end{bmatrix}.$$

At (x_0, y_0), $J_0 = \begin{bmatrix} 2 & -1 \\ 4 & 2 \end{bmatrix}$.

Therefore,

$$\begin{bmatrix} 2 & -1 \\ 4 & 2 \end{bmatrix}\begin{bmatrix} h \\ k \end{bmatrix} = \begin{bmatrix} -0.25 \\ -0.25 \end{bmatrix}$$

$$\text{or, } \begin{bmatrix} h \\ k \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 2 & 1 \\ -4 & 2 \end{bmatrix}\begin{bmatrix} -0.25 \\ -0.25 \end{bmatrix} = \begin{bmatrix} -0.09375 \\ 0.06250 \end{bmatrix}$$

Thus, $x_1 = x_0 + h = 2.00 - 0.09375 = 1.90625$,
$y_1 = y_0 + k = 0.25 + 0.0625 = 0.31250$.
At (x_1, y_1), $J_1 = \begin{bmatrix} 1.81250 & -1.00000 \\ 3.81250 & 2.50000 \end{bmatrix}$, $\begin{bmatrix} -f_1 \\ -g_1 \end{bmatrix} = \begin{bmatrix} -0.00879 \\ -0.02441 \end{bmatrix}$.

$$J_1 \begin{bmatrix} h \\ k \end{bmatrix} = \begin{bmatrix} -f_1 \\ -g_1 \end{bmatrix}$$

$$\text{or, } \begin{bmatrix} h \\ k \end{bmatrix} = \frac{1}{8.34375}\begin{bmatrix} 2.50000 & 1.00000 \\ -3.81250 & 1.81250 \end{bmatrix}\begin{bmatrix} -0.00879 \\ -0.02441 \end{bmatrix} = \begin{bmatrix} -0.00556 \\ -0.00129 \end{bmatrix}.$$

Therefore, $x_2 = x_1 + h = 1.90625 - 0.00556 = 1.90069$,
$y_2 = y_1 + k = 0.31250 - 0.00129 = 0.31121$.
At (x_2, y_2), $J_2 = \begin{bmatrix} 1.80138 & -1.00000 \\ 3.80138 & 2.48968 \end{bmatrix}$, $\begin{bmatrix} -f_2 \\ -g_2 \end{bmatrix} = \begin{bmatrix} -0.00003 \\ -0.00003 \end{bmatrix}$.

$$J_2 \begin{bmatrix} h \\ k \end{bmatrix} = \begin{bmatrix} -f_2 \\ -g_2 \end{bmatrix}$$

$$\text{or, } \begin{bmatrix} h \\ k \end{bmatrix} = \frac{1}{8.28624}\begin{bmatrix} 2.48968 & 1.00000 \\ -3.80138 & 1.80138 \end{bmatrix}\begin{bmatrix} -0.00003 \\ -0.00003 \end{bmatrix} = \begin{bmatrix} -0.00001 \\ 0.00001 \end{bmatrix}.$$

Hence, $x_3 = x_2 + h = 1.90069 - 0.00001 = 1.90068$,
$y_3 = y_2 + k = 0.31121 + 0.00001 = 0.31122$.
Thus, one root is $x = 1.9007, y = 0.3112$ correct up to four decimal places.

Algorithm 4.9 (Newton-Raphson method for pair of equations). This algorithm solves a pair of non-linear equations by Newton-Raphson method. The initial guess of a root is to be supplied.

```
Algorithm Newton-Raphson -2D
//$(x_0, y_0)$ is initial guess, $\varepsilon$ is the error tolerance.//
Input functions $f(x,y), g(x,y), f_x(x,y), f_y(x,y), g_x(x,y), g_y(x,y)$.
Read $x_0, y_0, \varepsilon$, $maxiteration$;
for $i = 1$ to $maxiteration$ do
    Compute $f_0 = f(x_0, y_0)$, $g_0 = g(x_0, y_0)$;
    if ($|f_0| < \varepsilon$ and $|g_0| < \varepsilon$) then
        Print 'A root is', $x_0, y_0$;
        Stop;
    endif;
    Compute $delfx = \left(\frac{\partial f}{\partial x}\right)_{(x_0,y_0)}$,   $delfy = \left(\frac{\partial f}{\partial y}\right)_{(x_0,y_0)}$
```

```
        Compute $delgx = \left(\frac{\partial g}{\partial x}\right)_{(x_0,y_0)}$,  $delgy = \left(\frac{\partial g}{\partial y}\right)_{(x_0,y_0)}$
        Compute $J = delfx * delgy - delgx * delfy$;
        Compute $h = (-f_0 * delgy + g_0 * delfy)/J$;
        Compute $k = (-g_0 * delfx + f_0 * delgx)/J$;
        Compute $x_0 \leftarrow x_0 + h, y_0 \leftarrow y_0 + k$;
endfor;
Print ‘Solution does not converge in’, $maxiteration$, ‘iteration’;
end Newton-Raphson -2D
```

Program 4.9

```
/* Program Newton-Raphson (for a pair of non-linear equations)
   Program to find a root of a pair of non-linear equations
   by Newton-Raphson method. Partial derivative of f and g
   w.r.t. x and y are to be supplied.
   The equations taken are 3x*x-2y*y-1=0, x*x-2x+2y-8=0. */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void main()
{
 int i,maxiteration;
 float eps=1e-5,x0,y0; /* initial guesses for x and y */
 float J,k,h,delfx,delfy,delgx,delgy,f0,g0;
 float f(float x, float y);
 float fx(float x, float y);
 float fy(float x, float y);
 float g(float x, float y);
 float gx(float x, float y);
 float gy(float x, float y);
 printf("Enter initial guess for x and y ");
 scanf("%f %f",&x0,&y0);
 printf("Maximum iterations to be allowed ");
 scanf("%d",&maxiteration);

 for(i=1;i<=maxiteration;i++)
   {
    f0=f(x0,y0);
    g0=g(x0,y0);
    if(fabs(f0)<eps && fabs(g0)<eps){
```

```
    printf("One root is (%7.5f, %7.5f) obtained at %d iterations",
             x0,y0,i);
       exit(0);
      }
    delfx=fx(x0,y0); delfy=fy(x0,y0);
    delgx=gx(x0,y0); delgy=gy(x0,y0);
    J=delfx*delgy-delgx*delfy;
    h=(-f0*delgy+g0*delfy)/J;
    k=(-g0*delfx+f0*delgx)/J;
    x0+=h; y0+=k;
  }
printf("Iteration does not converge in %d iterations",
         maxiteration);
} /* main */

/* definition of f(x,y) and its partial derivative w.r.t x and y
   fx(x,y) and fy(x,y) */
float f(float x, float y)
{
  return (3*x*x-2*y*y-1);
}
float fx(float x, float y)
{
  return (6*x);
}
float fy(float x, float y)
{
  return (-4*y);
}
/*definition of g(x,y) and its partial derivative w.r.t x and y
   gx(x,y) and gy(x,y) */
float g(float x, float y)
{
  return (x*x-2*x+2*y-8);
}
float gx(float x, float y)
{
  return (2*x-2);
}
```

```
float gy(float x, float y)
{
  return (2);
}
```

A sample of input/output:

```
Enter initial guess for x and y 2.5 3
Maximum iterations to be allowed 50
One root is (2.64005, 3.15512) obtained at 4 iterations
```

4.18 Exercise

1. Find the number of real roots of the equation (i) $2^x - 3 = 0$, (ii) $x^{10} - 4x^4 - 100x + 200 = 0$.

2. Describe the methods to locate the roots of the equation $f(x) = 0$.

3. Obtain a root for each of the following equations using bisection method, regula-falsi method, iteration method, Newton-Raphson method, secant method.
(i) $x^3 + 2x^2 - x + 7 = 0$, (ii) $x^3 - 4x - 9 = 0$, (iii) $\cos x = 3x - 1$
(iv) $e^{-x} = 10x$, (v) $\sin x = 10(x - 1)$, (vi) $\sin^2 x = x^2 - 1$
(vii) $\tan x - \tanh x = 0$, (viii) $x^3 + 0.5x^2 - 7.5x - 9.0 = 0$, (ix) $\tan x + x = 0$,
(x) $x^3 - 5.2x^2 - 17.4x + 21.6 = 0$, (xi) $x^7 + 28x^4 - 480 = 0$,
(xii) $(x - 1)(x - 2)(x - 3) = 0$, (xiii) $x - \cos x = 0$, (xiv) $x + \log x = 2$,
(xv) $\sin x = \frac{1}{2}x$, (xvi) $x^3 - \sin x + 1 = 0$, (xvii) $2x = \cos x + 3$,
(xviii) $x \log_{10} x = 4.77$, (xix) $x^2 - \sin \pi x = 0$, (xx) $2^x - 2x^2 - 1 = 0$,
(xxi) $2 \log x - \frac{x}{2} + 1 = 0$, (xxii) $\sqrt{1 + x} = 1/x$, (xxiii) $\log x + (x + 1)^3 = 0$.

4. Find a root of the equation $x = \frac{1}{2} + \sin x$ by using the fixed point iteration method
$$x_{n+1} = \frac{1}{2} + \sin x_n, x_0 = 1$$
correct to six decimal places.

5. Use Newton-Raphson method for multiple roots to find the roots of the following equations.
(i) $x^3 - 3x + 2 = 0$, (ii) $x^4 + 2x^3 - 2x - 1 = 0$.

6. Describe the bisection method to find a root of the equation $f(x) = 0$ when $f(a) \cdot f(b) < 0$, a, b be two specified numbers. Is this condition necessary to get a root using this method ?

7. Describe regula-falsi method for finding a real root of an equation. Why this method is called the bracketing method ? Give a geometric interpretation of this method. What is the rate of convergence of regula-falsi method ? Compare this method with Newton-Raphson method. Discuss the advantages and disadvantages of this method.

8. Compare bisection method and regula-falsi method. Also compare bracketing methods and iterative methods.

9. What is the main difference between regula-falsi method and secant method ?

10. Explain how an equation $f(x) = 0$ can be solved by the method of iteration (fixed point iteration) and deduce the condition of convergence. Show that the rate of convergence of iteration method is one. Give a geometric interpretation of this method. Discuss the advantages and disadvantages of this method.

11. Find the iteration schemes to solve the following equations using fixed point iteration method:
(i) $2x - \sin x - 1 = 0$, (ii) $x^3 - 2x - 1 = 0$, (iii) $x + \sin x = 0$.

12. Describe Newton-Raphson method for computing a simple real root of an equation $f(x) = 0$. Give a geometrical interpretation of the method. What are the advantages and disadvantages of this method. Prove that the Newton-Raphson method has a second order convergence.

13. Find the iteration schemes to solve the following equations using Newton-Raphson method:
(i) $2x - \cos x - 1 = 0$, (ii) $x^5 + 3x^2 - 1 = 0$, (iii) $x^2 - 2 = 0$.

14. Use Newton-Raphson method to find the value of the following terms:
(i) $1/15$, (ii) $\sqrt[3]{11}$, (iii) $\sqrt{5}$.

15. Show that an iterative method for computing $\sqrt[k]{a}$ is given by

$$x_{n+1} = \frac{1}{k}\left[(k-1)x_n + \frac{a}{x_n^{k-1}}\right]$$

and also deduce that

$$\varepsilon_{n+1} \simeq -\frac{k-1}{2\sqrt[k]{a}}\varepsilon_n^2$$

where ε_n is the error at the nth iteration. What is the order of convergence for this iterative method ?

16. Use Newton-Raphson method and modified Newton-Raphson method to find the roots of the following equations and compare the methods.
(i) $x^3 - 3x + 2 = 0$, (ii) $x^4 + 2x^3 - 2x - 1 = 0$.

17. Solve the following equation using Newton-Raphson method:
(i) $z^3 - 4iz^2 - 3e^z = 0$, $z_0 = -0.53 - 0.36i$, (ii) $1 + z^2 + z^3 = 0$, $z_0 = 1 + i$.

18. Compare Newton-Raphson method and modified Newton-Raphson method to find a root of the equation $f(x) = 0$.

19. Use modified Newton-Raphson method to find a root of the equation
$$\int_0^x e^{-t^2} dt = \frac{1}{10}$$
with six decimal places.

20. Device a scheme to find the value of $\sqrt[5]{a}$ using modified Newton-Raphson method.

21. The formula
$$x_{n+1} = x_n - \frac{1}{2}\frac{f(x_n + 2f(x_n)/f'(x_n))}{f'(x_n)}$$
is used to find a multiple root of multiplicity two of the equation $f(x) = 0$. Show that the rate of convergence of this method is cubic.

22. The iteration scheme
$$x_{n+1} = x_n - \frac{3\log_e x_n - e^{-x_n}}{p}$$
is used to find the root of the equation $e^{-x} - 3\log_e x = 0$. Show that $p = 3$ gives rapid convergence.

23. Use Muller's method to find a root of the equations
(i) $x^3 - 2x - 1 = 0$, (ii) $x^4 - 6x^2 + 3x - 1 = 0$.

24. Determine the order of convergence of the iterative method
$$x_{n+1} = \frac{x_0 f(x_n) - x_n f(x_0)}{f(x_n) - f(x_0)}$$
for finding the simple root of the equation $f(x) = 0$.

25. Find the order of convergence of the Steffensen method
$$x_{n+1} = x_n - \frac{f(x_n)}{g(x_n)}, g(x_n) = \frac{f(x_n + f(x_n)) - f(x_n)}{f(x_n)}, n = 0, 1, 2, \ldots.$$
Use this method to find a root of the equation $x - 1 + e^x = 0$.

26. Show that the iteration scheme to find the value of $\sqrt{a}$ using Chebyshev third order method is given by

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) - \frac{1}{8x_n}\left(x_n - \frac{a}{x_n}\right)^2.$$

Use this scheme to find the value of $\sqrt{17}$.

27. An iteration scheme is given by

$$x_0 = 5, \qquad x_{n+1} = \frac{1}{16}x_n^4 - \frac{1}{2}x_n^3 + 8x_n - 12.$$

Show that it gives cubic convergence to $\xi = 4$.

28. Using Graeffe's root squaring method, find the roots of the following equations
(i) $x^3 - 4x^2 + 3x + 1 = 0$, (ii) $x^3 - 3x - 5 = 0$,
(iii) $x^4 - 2x^3 + 1.99x^2 - 2x + 0.99 = 0$, (iv) $x^3 + 5x^2 - 44x - 60 = 0$.

29. Use Birge-Vieta method to find the roots of the equations to 3 decimal places.
(i) $x^4 - 8x^3 + 14.91x^2 + 9.54x - 25.92 = 0$, (ii) $x^3 - 4x + 1 = 0$,
(iii) $x^4 - 1.1x^3 - 0.2x^2 - 1.2x + 0.9 = 0$.

30. Find the quadratic factors of the following polynomial equations using Bairstow's method.
(i) $x^4 - 8x^3 + 39x^2 - 62x + 50 = 0$, (ii) $x^3 - 2x^2 + x - 2 = 0$,
(iii) $x^4 - 6x^3 + 18x^2 - 24x + 16 = 0$, (iv) $x^3 - 2x + 1 = 0$.

31. Solve the following systems of nonlinear equations using iteration method
(i) $x^2 + y = 11$, $y^2 + x = 7$, (ii) $2xy - 3 = 0, x^2 - y - 2 = 0$.

32. Solve the following systems of nonlinear equations using Seidal method
(i) $x^2 + 4y^2 - 4 = 0$, $x^2 - 2x - y + 1 = 0$, start with $(1.5, 0.5)$,
(ii) $3x^2 - 2y^2 - 1 = 0, x^2 - 2x + y^2 + 2y - 8 = 0$, start with $(-1, 1)$.

33. Solve the following systems of nonlinear equations using Newton-Raphson method
(i) $3x^2 - 2y^2 - 1 = 0, x^2 - 2x + 2y - 8 = 0$ start with initial guess (2.5,3),
(ii) $x^2 - x + y^2 + z^2 - 5 = 0, x^2 + y^2 - y + z^2 - 4 = 0, x^2 + y^2 + z^2 + z - 6 = 0$ start with $(-0.8, 0.2, 1.8)$ and $(1.2, 2.2, -0.2)$.

34. Use Newton's method to find all nine solutions to $7x^3 - 10x - y - 1 = 0, 8y^3 - 11y + x - 1 = 0$. Use the starting points $(0,0), (1,0), (0,1)$, $(-1,0)$, $(0,-1)$, $(1,1)$, $(-1,1)$, $(1,-1)$ and $(-1,-1)$.

35. What are the differences between direct method and iterative method ?

Chapter 5

Solution of System of Linear Equations

A system of m linear equations in n unknowns (variables) is written as

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots &\quad \cdots \\
a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n &= b_i \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots &\quad \cdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m.
\end{aligned} \tag{5.1}$$

The quantities $x_1, x_2, \ldots, x_n$ are the **unknowns** (**variables**) of the system and $a_{11}, a_{12}, \ldots, a_{mn}$ are the **coefficients** of the unknowns of the system. The numbers $b_1, b_2, \ldots, b_m$ are **constant** or **free terms** of the system.

The above system of equations (5.1) can be written as

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \qquad i = 1, 2, \ldots, m. \tag{5.2}$$

Also, the system of equations (5.1) can be written in matrix form as

$$\mathbf{AX} = \mathbf{b}, \tag{5.3}$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{i1} & a_{i2} & \cdots & a_{in} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_m \end{bmatrix} \text{ and } X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_m \end{bmatrix}. \tag{5.4}$$

The system of linear equation (5.1) is **consistent** if it has a solution. If a system of linear equations has no solution, then it is **inconsistent** (or **incompatible**). A consistent system of linear equations may have one solution or several solutions and is said to be **determinate** if there is one solution and **indeterminate** if there are more than one solution.

Generally, the following three types of the **elementary transformations** to a system of linear equations are used.

Interchange: The order of two equations can be changed.

Scaling: Multiplication of both sides of an equation of the system by any non-zero number.

Replacement: Addition to (subtraction from) both sides of one equation of the corresponding sides of another equation multiplied by any number.

A system in which the constant terms $b_1, b_2, \ldots, b_m$ are zero is called a **homogeneous** system.

Two basic techniques are used to solve a system of linear equations:

(i) direct method, and (ii) iteration method.

Several direct methods are used to solve a system of equations, among them following are most useful.

(i) Cramer's rule, (ii) matrix inversion, (iii) Gauss elimination, (iv) decomposition, etc.

The most widely used iteration methods are (i) Jacobi's iteration, (ii) Gauss-Seidal's iteration, etc.

Direct Methods

5.1 Cramer's Rule

To solve a system of linear equations, a simple method (but, not efficient) was discovered by Gabriel Cramer in 1750.

Let the determinant of the coefficients of the system (5.2) be $D = |a_{ij}|; i,j = 1,2,\ldots,n$, i.e., $D = |A|$. In this method, it is assumed that $D \neq 0$. The Cramer's rule is described in the following. From the properties of determinant

$$x_1 D = x_1 \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} = \begin{vmatrix} x_1 a_{11} & a_{12} & \cdots & a_{1n} \\ x_1 a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_1 a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix}$$

$$= \begin{vmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & a_{12} & \cdots & a_{1n} \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n & a_{n2} & \cdots & a_{nn} \end{vmatrix} \quad \text{[Using the operation } C_1' = C_1 + x_2C_2 + \cdots + x_nC_n.\text{]}$$

$$= \begin{vmatrix} b_1 & a_{12} & \cdots & a_{1n} \\ b_2 & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ b_n & a_{n2} & \cdots & a_{nn} \end{vmatrix} \text{[Using (5.1)]}$$
$$= D_{x_1}(say).$$

Therefore, $x_1 = D_{x_1}/D$.
Similarly, $x_2 = \dfrac{D_{x_2}}{D}, \ldots, x_n = \dfrac{D_{x_n}}{D}$.
In general, $x_i = \dfrac{D_{x_i}}{D}$, where

$$D_{x_i} = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1\,i-1} & b_1 & a_{1\,i+1} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2\,i-1} & b_2 & a_{2\,i+1} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{n\,i-1} & b_n & a_{n\,i+1} & \cdots & a_{nn} \end{vmatrix},$$

$i = 1, 2, \cdots, n$

Example 5.1.1 Use Cramer's rule to solve the following systems of equations

$$\begin{aligned} x_1 + x_2 + x_3 &= 2 \\ 2x_1 + x_2 - x_3 &= 5 \\ x_1 + 3x_2 + 2x_3 &= 5. \end{aligned}$$

Solution. The determinant D of the system is

$$D = \begin{vmatrix} 1 & 1 & 1 \\ 2 & 1 & -1 \\ 1 & 3 & 2 \end{vmatrix} = 5.$$

The determinants D_1, D_2 and D_3 are shown below:

$$D_1 = \begin{vmatrix} 2 & 1 & 1 \\ 5 & 1 & -1 \\ 5 & 3 & 2 \end{vmatrix} = 5, \qquad D_2 = \begin{vmatrix} 1 & 2 & 1 \\ 2 & 5 & -1 \\ 1 & 5 & 2 \end{vmatrix} = 10, \qquad D_3 = \begin{vmatrix} 1 & 1 & 2 \\ 2 & 1 & 5 \\ 1 & 3 & 5 \end{vmatrix} = -5.$$

Thus, $x_1 = \dfrac{D_1}{D} = \dfrac{5}{5} = 1, x_2 = \dfrac{D_2}{D} = \dfrac{10}{5} = 2, x_3 = \dfrac{D_3}{D} = -\dfrac{5}{5} = -1.$
Therefore the solution is $x_1 = 1, x_2 = 2, x_3 = -1$.

5.1.1 Computational aspect of Cramer's rule

It may be noted that the Cramer's rule involves to compute $(n + 1)$ determinants each of order n (for a system of n equations and n variables). Again, the Laplace's expansion method (the conventional method to find the value of a determinant) to evaluate a determinant of order n requires $(n! - 1)$ additions and $n!(n-1)$ multiplications. Thus, to compute a determinant of order 10, needs 32 million multiplications and 4 millions additions. So, the Cramer's rule is only of theoretical interest due to the computational inefficiency. But, the time complexity can be reduced by triangularizing the determinant and this method is a polynomial time bound algorithm.

5.2 Evaluation of Determinant

Triangularization, is also known as **Gauss reduction** method, is the best way to evaluate a determinant. The basic principle of this method is to convert the given determinant into a lower or upper triangular form by using only elementary row operations. If the determinant is reduced to a triangular form (D') then the value of D is obtained by multiplying the diagonal elements of D'.

Let $D = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix}$ be a determinant of order n.

Using the elementary row operations, D can be reduced to the following form:

$$D' = \begin{vmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n-1)} \end{vmatrix},$$

where

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} a_{kj}^{(k-1)}; \tag{5.5}$$

$i, j = k+1, \ldots, n;\;\; k = 1, 2, \ldots, n-1$ and $a_{ij}^{(0)} = a_{ij},\;\; i, j = 1, 2, \cdots, n$.

Then the value of D is equal to $a_{11}a_{22}^{(1)}a_{33}^{(2)} \cdots a_{nn}^{(n-1)}$.

To compute the value of $a_{ij}^{(k)}$ one division is required. If $a_{kk}^{(k-1)}$ is zero then further reduction is not possible. If $a_{kk}^{(k-1)}$ is small then the division leads to the loss of significant digits. To prevent the loss of significant digits, the pivoting techniques are used.

A **pivot** is the largest magnitude element in a row or a column or the principal diagonal or the leading or trailing sub-matrix of order i $(2 \leq i \leq n)$.

For example, for the matrix

$$A = \begin{bmatrix} 5 & -1 & 0 & 5 \\ -10 & 8 & 3 & 10 \\ 20 & 3 & -30 & 8 \\ 3 & 50 & 9 & 10 \end{bmatrix}$$

20 is the pivot for the first column, -30 is the pivot for the principal diagonal, 50 is the pivot for this matrix and -10 is the pivot for the trailing sub-matrix $\begin{bmatrix} 5 & -1 \\ -10 & 8 \end{bmatrix}$.

In the elementary row operation, if the any one of the pivot element is zero or very small relative to other elements in that row then we rearrange the remaining rows in such a way that the pivot becomes non-zero or not a very small number. The method is called pivoting. The pivoting are of two types- partial pivoting and complete pivoting. Partial and complete pivoting are discussed in the following.

Partial pivoting

In the first stage, the first pivot is determined by finding the largest element in magnitude among the elements of first column and let it be a_{i1}. Then rows i and 1 are interchanged. In the second stage, the second pivot is determined by finding the largest element in magnitude among the elements of second column leaving first element and let it be a_{j2}. The second and jth rows are interchanged. This process is repeated for $(n-1)$th times. In general, at ith stage, the smallest index j is chosen for which

$$|a_{ij}^{(k)}| = \max\{|a_{kk}^{(k)}|, |a_{k+1\,k}^{(k)}|, \ldots, |a_{nk}^{(k)}|\} = \max\{|a_{ik}^{(k)}|, i = k, k+1, \ldots, n\}$$

and the rows k and j are interchanged.

Complete pivoting or full pivoting

The largest element in magnitude is determined among all the elements of the determinant and let it be $|a_{lm}|$.

Taking a_{lm} as the first pivot by interchanging first row and the lth row and of first column and mth column. In second stage, the largest element in magnitude is determined among all elements leaving the first row and first column and taking this element as second pivot.

In general, we choose l and m for which

$$|a_{lm}^{(k)}| = \max\{|a_{ij}^{(k)}|, i, j = k, k+1, \ldots, n\}.$$

Then the rows k, l and columns k, m are interchanged, and a_{kk} becomes a pivot.

The complete pivoting is more complicated than the partial pivoting. The partial pivoting is preferred for hand computation.

Note 5.2.1 If the coefficient matrix **A** is diagonally dominant, i.e.,

$$\sum_{\substack{j=1\\j\neq i}}^{n} |a_{ij}| < |a_{ii}| \qquad \text{or} \qquad \sum_{\substack{j=1\\j\neq i}}^{n} |a_{ji}| < |a_{ii}|, \qquad \text{for } i = 1, 2, \ldots, n. \tag{5.6}$$

or real symmetric and positive definite then no pivoting is necessary.

Note 5.2.2 Every diagonally dominant matrix is non-singular.

To illustrate the partial pivoting, let us consider the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 7 & 3 \\ 4 & 5 & 1 \\ -8 & 1 & 6 \end{bmatrix}.$$

The largest element (in magnitude) in the first column is -8. Then interchanging first and third rows i.e.,

$$\mathbf{A} \sim \begin{bmatrix} -8 & 1 & 6 \\ 4 & 5 & 1 \\ 1 & 7 & 3 \end{bmatrix}.$$

The largest element in second column leaving the first row is 7, so interchanging second and third rows.

The matrix after partial pivoting is

$$A \sim \begin{bmatrix} -8 & 1 & 6 \\ 1 & 7 & 3 \\ 4 & 5 & 1 \end{bmatrix}.$$

Let us consider the matrix $\mathbf{B} = \begin{bmatrix} 1 & 7 & 3 \\ 4 & -8 & 5 \\ 2 & 6 & 1 \end{bmatrix}$ to illustrate the complete pivoting. The largest element (in magnitude) is determined among all the elements of the matrix. It is -8 attained at the (2, 2) position. Therefore, first and second columns, and first and second rows are interchanged. The matrix transferred to

$$\mathbf{B} \sim \begin{bmatrix} -8 & 4 & 5 \\ 7 & 1 & 3 \\ 6 & 2 & 1 \end{bmatrix}.$$

The number -8 is the first pivot. To find second pivot, largest element is determined from the trailing sub-matrix $\begin{bmatrix} 1 & 3 \\ 2 & 1 \end{bmatrix}$. The largest element is 3 and it is at the position (2, 3).

Interchanging second and third columns. The final matrix (after complete pivoting) is $\begin{bmatrix} -8 & 5 & 4 \\ 7 & 3 & 1 \\ 6 & 1 & 2 \end{bmatrix}$.

Example 5.2.1 Compute the determinant of the following matrix by a triangularization algorithm using (i) partial pivoting, and (ii) complete pivoting:

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 4 \\ 4 & 6 & 1 \\ 5 & 1 & -2 \end{bmatrix}.$$

Solution. (i) The largest element in the first column is 5, which is the first pivot of **A**.
Interchanging first and third rows, we obtain

$$\begin{bmatrix} 5 & 1 & -2 \\ 4 & 6 & 1 \\ 2 & 0 & 4 \end{bmatrix}.$$

and *sign* $= -1$.
Adding $-\frac{4}{5}$ times the first row to the second row, $-\frac{2}{5}$ times the first row to the third row i.e., $R_2' = R_2 - \frac{4}{5}R_1$ and $R_3' = R_3 - \frac{2}{5}R_1$, we get

$$\begin{bmatrix} 5 & 1 & -2 \\ 0 & 26/5 & 13/5 \\ 0 & -2/5 & 24/5 \end{bmatrix}.$$

The second pivot is $\frac{26}{5}$, which is the largest element (in magnitude) among the elements of second column except first row. Since this element is in the (2,2) position, so no interchange of rows is required.
Adding $\frac{2/5}{26/5}$ times the second row to the third row i.e., $R_3' = R_3 + \frac{2/5}{26/5}R_2$ we obtain

$$\begin{bmatrix} 5 & 1 & -2 \\ 0 & 26/5 & 13/5 \\ 0 & 0 & 5 \end{bmatrix}.$$

Hence the determinant is $sign \times (5)(26/5)(5) = -1(5)(26/5)(5) = -130$.

(ii) The largest element in $\mathbf{A}$ is 6. Interchanging first and second columns and setting $sign = -1$; and then interchanging first and second rows and setting $sign = -sign = 1$, we have

$$\begin{bmatrix} 6 & 4 & 1 \\ 0 & 2 & 4 \\ 1 & 5 & -2 \end{bmatrix}.$$

Adding $-\frac{1}{6}$ times the first row to the third row i.e., $R_3' = R_3 - \frac{1}{6}R_1$ we obtain

$$\begin{bmatrix} 6 & 4 & 1 \\ 0 & 2 & 4 \\ 0 & 13/3 & -13/6 \end{bmatrix}.$$

The pivot of the trailing sub-matrix $\begin{bmatrix} 2 & 4 \\ 13/3 & -13/6 \end{bmatrix}$ is 13/3. Interchanging the second and third rows, we have $\begin{bmatrix} 6 & 1 & 4 \\ 0 & 13/3 & -13/6 \\ 0 & 2 & 4 \end{bmatrix}$ and $sign = -1$.

Adding $-\dfrac{2}{13/3}$ times the second row to the third row i.e., $R_3' = R_3 - \dfrac{6}{13}R_2$ we obtain

$$\begin{bmatrix} 6 & 1 & 4 \\ 0 & 13/3 & -13/6 \\ 0 & 0 & 5 \end{bmatrix}.$$

Therefore, $|\mathbf{A}| = sign \times (6)(13/3)(5) = -130$.

The algorithm for triangularization i.e., to find the value of a determinant using partial and complete pivoting are presented below.

Algorithm 5.1 (Evaluation of determinant using partial pivoting). This algorithm finds the value of a determinant of order $n \times n$ using partial pivoting.

```
Algorithm Det_Partial_Pivoting.
//The value of determinant using partial pivoting.//
Let $\mathbf{A} = [a_{ij}]$ be an $n \times n$ matrix.
 Step 1. Read the matrix $\mathbf{A} = [a_{ij}], i, j = 1, 2, \ldots, n$.
 Step 2. Set $k = 1$ and $sign = 1$// sign indicates the sign of the determinant
         when interchanges two rows.//
 Step 3. Find a pivot from the elements $a_{kk}, a_{k+1k}, \cdots, a_{nk}$ in the $k$th
         column of $\mathbf{A}$, and let $a_{jk}$ be the pivot. That is, $|a_{jk}| =$
         $\max\{|a_{kk}|, |a_{k+1\,k}|, \ldots, |a_{nk}|\}$.
 Step 4. If $a_{jk} = 0$ then $|\mathbf{A}| = 0$; print the value of $|\mathbf{A}|$ and Stop.
 Step 5. If $j = k$ then go to Step 6.
         Otherwise interchange the $k$th and $j$th rows and set $sign = -sign$.
```

```
Step 6. Subtract $\frac{a_{jk}}{a_{kk}}$ times the kth row from the jth row for $j = k+1, k+2, \ldots, n$
        i.e., for $j = k+1, k+2, \ldots, n$ do the following $R_j' = R_j - \frac{a_{jk}}{a_{kk}}.R_k$, where
        $R_j, R_j'$ are the old and new jth rows respectively.
        // This step makes $a_{k+1k}, a_{k+2k}, \cdots, a_{nk}$ zero.//
Step 7. Increment k by 1 i.e., set $k = k+1$. If $k < n$ then goto Step 3. Otherwise,
        //Triangularization is complete.//
        Compute $|\mathbf{A}| = sign \times$ product of diagonal elements.
        Print $|\mathbf{A}|$ and Stop.
end Det_Partial_Pivoting.
```

Program 5.1

```
/* Program Partial Pivoting
   Program to find the value of a determinant using partial pivoting */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
void main()
{
 int n,k,i,j,sign=1;
 float a[10][10],b[10],prod,temp;
 int max1(float b[],int k, int n);
 printf("\nEnter the size of the determinant ");
 scanf("%d",&n);
 printf("Enter the elements rowwise ");
 for(i=1;i<=n;i++) for(j=1;j<=n;j++)
    scanf("%f",&a[i][j]);
 for(k=1;k<=n;k++)
  {
    for(i=k;i<=n;i++) b[i]=a[i][k];
       /* copy from a[k][k] to a[n][k] into b */
    j=max1(b,k,n); /* finds pivot position */
    if(a[j][k]==0)
       {
         printf("The value of determinant is 0");
         exit(0);
       }
    if(j!=k) /* interchange k and j rows */
       {
       sign=-sign;
```

```
        for(i=1;i<=n;i++){
             temp=a[j][i]; a[j][i]=a[k][i]; a[k][i]=temp;
          }
        }
     for(j=k+1;j<=n;j++) /* makes a[k+1][k] to a[n][k] zero */
         {
            temp=a[j][k]/a[k][k];
            for(i=1;i<=n;i++)   a[j][i]-=temp*a[k][i];
         }
   } /* end of k loop */
  prod=sign;
  /* product of diagonal elements */
  for(i=1;i<=n;i++) prod*=a[i][i];
  printf("The value of the determinant is %f ",prod);
}/* main */
/* finds position of maximum element among n numbers */
int max1(float b[],int k, int n)
   {
     float temp; int i,j;
     temp=fabs(b[k]);  j=k; /* initial maximum */
     for(i=k+1;i<=n;i++)
         if(temp<fabs(b[i])) {temp=fabs(b[i]); j=i;}
     return j;
   }
```

A sample of input/output:

```
Enter the size of the determinant 3
Enter the elements rowwise
0 2 5
1 3 -8
6 5 1
The value of the determinant is -163.000000
```

Algorithm 5.2 (Evaluation of determinant using complete pivoting). This algorithm finds the value of a determinant of order $n \times n$ using complete pivoting.

Algorithm Det_Complete_Pivoting.
Let $\mathbf{A} = [a_{ij}]$ be an $n \times n$ matrix.

```
Step 1. Read the matrix $\mathbf{A} = [a_{ij}], i, j = 1, 2, \ldots, n$.
Step 2. Set $k = 1$ and $sign = 1$.
Step 3. Find a pivot from the elements of the trailing sub-matrix
        $\begin{bmatrix} a_{kk} & a_{kk+1} & \cdots & a_{kn} \\ a_{k+1k} & a_{k+1k+1} & \cdots & a_{k+1n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{nk} & a_{nk+1} & \cdots & a_{nn} \end{bmatrix}$ of $\mathbf{A}$. Let $a_{pq}$ be the pivot.
        i.e., $|a_{pq}| = \max\{|a_{kk}|, \ldots, |a_{kn}|; \quad |a_{k+1k}|, \ldots, |a_{k+1n}|; |a_{nk}|, \ldots, |a_{nn}|\}$.
Step 4. If $a_{pq} = 0$ then $|\mathbf{A}| = 0$; print the value of $|\mathbf{A}|$ and Stop.
Step 5. If $p = k$ then goto Step 6. Otherwise, interchange the $k$th and the $p$th
        rows and set $sign = -sign$.
Step 6. If $q = k$ then goto Step 7. Otherwise, interchange the $k$th and the $q$th
        columns and set $sign = -sign$.
Step 7. Subtract $\frac{a_{jk}}{a_{kk}}$ times the $k$th row to the $j$th row for $j = k+1, k+2, \ldots, n$
        i.e., for $j = k+1, k+2, \ldots, n$ do the following
        $R_j' = R_j - \frac{a_{jk}}{a_{kk}}.R_k$, where $R_j, R_j'$ are the old and new $j$th rows respectively.
Step 8. Increment $k$ by 1 i.e., set $k = k + 1$.
        If $k < n$ then goto Step 3. Otherwise,
        compute $|\mathbf{A}| = sign \times$ product of diagonal elements.
        Print $|\mathbf{A}|$ and Stop.
end Det_Complete_Pivoting.
```

Program 5.2

```
/*Program Complete Pivoting
  Program to find the value of a determinant using complete pivoting */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
void main()
{
int n,k,i,j,sign=1,p,q;
float a[10][10],prod,max,temp;
printf("\nEnter the size of the determinant ");
scanf("%d",&n);
printf("Enter the elements rowwise ");
for(i=1;i<=n;i++) for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
for(k=1;k<=n;k++)
 {
    /* finds the position of the pivot element */
    max=fabs(a[k][k]); p=k; q=k; /* set initial maximum */
    for(i=k;i<=n;i++)
      for(j=k;j<=n;j++)
        if(max<fabs(a[i][j])) { max=fabs(a[i][j]); p=i; q=j;}
    if(a[p][q]==0)
       {
         printf("The value of determinant is 0");
         exit(0);
       }
    if(p!=k) /* interchange k and p rows */
       {
         sign=-sign;
         for(i=1;i<=n;i++)
         {
           temp=a[p][i]; a[p][i]=a[k][i]; a[k][i]=temp;
         }
       }
    if(q!=k) /* interchange k and q columns */
       {
         sign=-sign;
         for(i=1;i<=n;i++)
             {
                temp=a[i][q]; a[i][q]=a[i][k]; a[i][k]=temp;
             }
       }
    for(j=k+1;j<=n;j++) /* makes a[k+1][k] to a[n][k] zero */
        {
           temp=a[j][k]/a[k][k];
           for(i=1;i<=n;i++)   a[j][i]-=temp*a[k][i];
        }
 } /* end of k loop */
 prod=sign;
 for(i=1;i<=n;i++) /* product of diagonal elements*/
    prod*=a[i][i];
 printf("The value of the determinant is %f ",prod);
}/* main */
```

A sample of input/output:

```
Enter the size of the determinant 4
Enter the elements rowwise
```

```
-2 3 8 4
6 1 0 5
-8 3 1 2
3 8 7 10
The value of the determinant is -1273.000000
```

Advantages and disadvantages of partial and complete pivoting

The general disadvantages of the pivoting is that the symmetry or regularity of the original matrix may be lost. Partial pivoting requires less time in terms of interchanges and search for the pivot than the complete pivoting. A combination of partial and complete pivoting is expected to be very effective not only for computing a determinant but also for solving system of linear equations. The pivoting brings in stability where a method becomes unstable for a problem. The pivoting reduces the error due to the loss of significant digits.

5.3 Inverse of a Matrix

From the theory of matrices, it is well known that every square non-singular matrix has unique inverse. The inverse of a matrix **A** is defined by

$$\mathbf{A}^{-1} = \frac{\mathbf{adj\ A}}{|\mathbf{A}|}. \tag{5.7}$$

The matrix **adj A** is called adjoint of **A** and defined as

$$\mathbf{adj\ A} = \begin{bmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & A_{22} & \cdots & A_{n2} \\ \cdots & \cdots & \cdots & \cdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{bmatrix},$$

where A_{ij} being the cofactor of a_{ij} in $|\mathbf{A}|$.

The main difficulty of this method is to compute the inverse of the matrix **A**. From the definition of **adj A** it is easy to observe that to compute the matrix **adj A**, we have to determine n^2 determinants each of order $(n-1)$. So, it is very much time consuming. Many efficient methods are available to find the inverse of a matrix, among them **Gauss-Jordan** is most popular. In the following Gauss-Jordan method is discussed to find the inverse of a square non-singular matrix.

5.3.1 Gauss-Jordan Method

In this method, the given matrix **A** is augmented with a unit matrix of same size, i.e., if the order of **A** is $n \times n$ then the order of the augmented matrix $[\mathbf{A}\vdots\mathbf{I}]$ will be $n \times 2n$.

The augmented matrix looks like

$$[\mathbf{A}\vdots\mathbf{I}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & \vdots & 1 & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & a_{2n} & \vdots & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \vdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & \vdots & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{5.8}$$

Then the inverse of **A** is computed in two stages. In the first stage, **A** is converted into an upper triangular form, using only elementary row operations (Gauss elimination method discussed in Section 5.5). In the second stage, the upper triangular matrix (obtained in first stage) is reduced to an identity matrix by row operations. All these operations are operated on the augmented matrix $[\mathbf{A}\vdots\mathbf{I}]$. After completion of these stages, the augmented matrix $[\mathbf{A}\vdots\mathbf{I}]$ is turned to $[\mathbf{I}\vdots\mathbf{A}^{-1}]$, i.e., the inverse of **A** is obtained from the right half of augmented matrix.

Thus

$$[\mathbf{A}\vdots\mathbf{I}] \xrightarrow{\text{Gauss} - \text{Jordan}} [\mathbf{I}\vdots\mathbf{A}^{-1}].$$

At the end of the operations the matrix shown in (5.8) reduces to the following form:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & \vdots & a'_{11} & a'_{12} & \cdots & a'_{1n} \\ 0 & 1 & \cdots & 0 & \vdots & a'_{21} & a'_{22} & \cdots & a'_{2n} \\ \cdots & \cdots & \cdots & \cdots & \vdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & \vdots & a'_{n1} & a'_{n2} & \cdots & a'_{nn} \end{bmatrix}. \tag{5.9}$$

Example 5.3.1 Find the inverse of the following matrix $\mathbf{A} = \begin{bmatrix} 2 & 4 & 5 \\ 1 & -1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$.

Solution. The augmented matrix $[\mathbf{A}\vdots\mathbf{I}]$ can be written as

$$[\mathbf{A}\vdots\mathbf{I}] = \begin{bmatrix} 2 & 4 & 5 & \vdots & 1 & 0 & 0 \\ 1 & -1 & 2 & \vdots & 0 & 1 & 0 \\ 3 & 4 & 5 & \vdots & 0 & 0 & 1 \end{bmatrix}. \tag{5.10}$$

Stage I. *(Reduction to upper triangular form):*
In the first column 3 is the largest element, thus interchanging first (R_1) and third (R_3) rows to bring the pivot element 3 to the a_{11} position. Then (5.10) becomes

$$\left[\begin{array}{ccc:ccc} 3 & 4 & 5 & 0 & 0 & 1 \\ 1 & -1 & 2 & 0 & 1 & 0 \\ 2 & 4 & 5 & 1 & 0 & 0 \end{array}\right].$$

$$\sim \left[\begin{array}{ccc:ccc} 1 & 4/3 & 5/3 & 0 & 0 & 1/3 \\ 1 & -1 & 2 & 0 & 1 & 0 \\ 2 & 4 & 5 & 1 & 0 & 0 \end{array}\right] R_1' = \tfrac{1}{3}R_1$$

$$\sim \left[\begin{array}{ccc:ccc} 1 & 4/3 & 5/3 & 0 & 0 & 1/3 \\ 0 & -7/3 & 1/3 & 0 & 1 & -1/3 \\ 0 & 4/3 & 5/3 & 1 & 0 & -2/3 \end{array}\right] R_2' = R_2 - R_1; \quad R_3' = R_3 - 2R_1$$

(The largest element (in magnitude) in the second column is $-\dfrac{7}{3}$, which is at the a_{22} position and so there is no need to interchange any rows).

$$\sim \left[\begin{array}{ccc:ccc} 1 & 4/3 & 5/3 & 0 & 0 & 1/3 \\ 0 & 1 & -1/7 & 0 & -3/7 & 1/7 \\ 0 & 0 & 13/7 & 1 & 4/7 & -6/7 \end{array}\right] R_2' = -\tfrac{3}{7}R_2\,; \quad R_3' = R_3 - \tfrac{4}{3}R_2'$$

$$\sim \left[\begin{array}{ccc:ccc} 1 & 4/3 & 5/3 & 0 & 0 & 1/3 \\ 0 & 1 & -1/7 & 0 & -3/7 & 1/7 \\ 0 & 0 & 1 & 7/13 & 4/13 & -6/13 \end{array}\right] R_3' = \tfrac{7}{13}R_3$$

Stage II. *(Make the left half a unit matrix):*

$$\sim \left[\begin{array}{ccc:ccc} 1 & 0 & 13/7 & 0 & 4/7 & 1/7 \\ 0 & 1 & -1/7 & 0 & -3/7 & 1/7 \\ 0 & 0 & 1 & 7/13 & 4/13 & -6/13 \end{array}\right] R_1' = R_1 - \tfrac{4}{3}R_2$$

$$\sim \left[\begin{array}{ccc:ccc} 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 1/13 & -5/13 & 1/13 \\ 0 & 0 & 1 & 7/13 & 4/13 & -6/13 \end{array}\right] R_1' = R_1 - \tfrac{13}{7}R_3\,; \quad R_2' = R_2 + \tfrac{1}{7}R_3$$

The left hand becomes a unit matrix, thus the inverse of the given matrix is

$$\begin{bmatrix} -1 & 0 & 1 \\ 1/13 & -5/13 & 1/13 \\ 7/13 & 4/13 & -6/13 \end{bmatrix}.$$

Algorithm 5.3 (Matrix inverse). The following algorithm computes the inverse of a non-singular square matrix of order $n \times n$ and if the matrix is singular it prints the message 'the matrix is singular and hence not invertible'.

Algorithm Matrix_Inversion (using partial pivoting).

```
Let $\mathbf{A} = [a_{ij}]$ be an $n \times n$ matrix.
   Step 1. Read the matrix $\mathbf{A} = [a_{ij}], i, j = 1, 2, \ldots, n$.
   Step 2. //Augment the matrix $\mathbf{A}$.//
           Augment the matrix $\mathbf{A}$ by a unit matrix of order $n \times n$. The resultant
           matrix $\mathbf{A}$ becomes of order $n \times 2n$.
           i.e., $a_{i\,n+j} = \begin{cases} 0, & \text{for } i \neq j \\ 1, & \text{for } i = j \end{cases}$
           for $i, j = 1, 2, \ldots, n$.

  Stage I. Make upper triangular form.
   Step 3. Set $k = 1$.
   Step 4. Find a pivot from the elements $a_{kk}, a_{k+1\,k}, \ldots, a_{nk}$ in the $k$th column of
           $\mathbf{A}$ and let $a_{jk}$ be the pivot.
   Step 5. If $a_{jk} = 0$ then print 'the matrix is singular and hence not invertible'
           and Stop.
   Step 6. If $j = k$ then goto Step 7.
           Otherwise interchange the $k$th and $j$th rows.
   Step 7. If $a_{kk} \neq 1$ then divide all the elements of $k$th row by $a_{kk}$.
           Subtract $a_{jk}$ times the $k$th row to the $j$th row for
           $j = k+1, k+2, \cdots, 2n$;
           i.e., $R'_j = R_j - a_{jk}R_k$.
           //This step makes $a_{k+1\,k}, a_{k+2\,k}, \ldots, a_{nk}$ zero.//
   Step 8. Increase $k$ by 1 i.e., set $k = k + 1$.
           If $k < n$ then goto Step 4. Otherwise goto Step 9.
           //Stage I is completed.//

 Stage II. //Make the left half of $\mathbf{A}$ a unit matrix.//
   Step 9. Set $k = 2$.

  Step 10. Subtract $a_{jk}$ times the $k$th row to the $j$th row for
           $j = k-1, k-2, \ldots, 1$.
  Step 11. Increase $k$ by 1.
           If $k < n$ then goto Step 10.
           Otherwise, print the right half of $\mathbf{A}$ as inverse of $\mathbf{A}$ and Stop.

end Matrix_Inversion
```

Program 5.3

```
/* Program Matrix Inverse
   Program to find the inverse of a square matrix using
   partial pivoting */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define zero 0.00001
void main()
{
int n,m,k,i,j;
float a[10][20],temp;
printf("\nEnter the size of the matrix ");
scanf("%d",&n);
printf("Enter the elements rowwise ");
for(i=1;i<=n;i++) for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
/* augment the matrix A */
for(i=1;i<=n;i++) for(j=1;j<=n;j++) a[i][n+j]=0;
for(i=1;i<=n;i++) a[i][n+i]=1;
m=2*n;
for(k=1;k<=n;k++)
  {
    /* finds pivot element and its position */
    temp=fabs(a[k][k]);  j=k; /* initial maximum */
    for(i=k+1;i<=n;i++)
        if(temp<fabs(a[i][k])){
             temp=fabs(a[i][k]); j=i;
           }
    if(fabs(a[j][k])<=zero) /* if a[j][k]=0 */
       {
         printf("The matrix is singular and is not invertible");
         exit(0);
       }
    if(j!=k) /* interchange k and j rows */
       {
       for(i=1;i<=m;i++){
            temp=a[j][i]; a[j][i]=a[k][i]; a[k][i]=temp;
         }
       }
    if(a[k][k]!=1)
       {
         temp=a[k][k];
         for(i=1;i<=m;i++) a[k][i]/=temp;
       }
    for(j=k+1;j<=n;j++) /* makes a[k+1][k] to a[n][k] zero */
        {
           temp=a[j][k];
           for(i=1;i<=m;i++)   a[j][i]-=temp*a[k][i];
        }
 } /* end of k loop */

 /* make left half of A to a unit matrix */
 for(k=2;k<=n;k++)
     {
        for(j=k-1;j>=1;j--)
            {
               temp=a[j][k];
               for(i=1;i<=m;i++) a[j][i]-=temp*a[k][i];
            }
       }
 printf("\nThe inverse matrix is \n");
 for(i=1;i<=n;i++)
    {
      for(j=n+1;j<=m;j++)
        printf("%f ",a[i][j]); printf("\n");
    }
}/* main */
```

A sample of input/output:

```
Enter the size of the matrix 3
Enter the elements rowwise
0 1 2
3 -2 1
4 3 2
The inverse matrix is
-0.218750  0.125000  0.156250
-0.062500 -0.250000  0.187500
 0.531250  0.125000 -0.093750
```

Complexity of the algorithm

Step 4 determines the maximum among $n-k+1$ elements and takes $n-k+1$ comparisons. Step 6 takes $O(n)$ operations if the kth and jth rows need to interchange. For a fixed j, Step 7 needs $n-k$ additions and $n-k$ multiplications. Since j is running from $k+1$ to n, so the total time taken by Step 7 is $(n-k)^2$. The Step 4 to Step 7 are repeated for n times $(k = 1, 2, \ldots, n)$, therefore, Stage I takes $\displaystyle\sum_{k=1}^{n}(n-k)^2 + O(n) + n - k + 1 = O(n^3)$ operations.

Similarly, Stage II takes $O(n^3)$ time. Hence the time complexity to compute the inverse of a non-singular matrix is $O(n^3)$.

Since the Stage I is similar to the algorithm **Det_Partial_Pivoting**, so the time complexity to compute the determinant is $O(n^3)$.

5.4 Matrix Inverse Method

The system of equations (5.1) can be written in the matrix form (5.3) as

$$\mathbf{Ax} = \mathbf{b}$$

where $\mathbf{A}, \mathbf{b}$ and $\mathbf{x}$ are defined in (5.4).

The solution of $\mathbf{Ax} = \mathbf{b}$ is given

$$\mathbf{x} = \mathbf{A^{-1}b}, \tag{5.11}$$

where $\mathbf{A^{-1}}$ is the inverse of the matrix $\mathbf{A}$.

Once the inverse of $\mathbf{A}$ is known then post multiplication of it with $\mathbf{b}$ gives the solution vector $\mathbf{x}$.

Example 5.4.1 Solve the following system of equations by matrix inverse method $x + 2y + 3z = 10,\ \ x + 3y - 2z = 7,\ \ 2x - y + z = 5.$

Solution. The given system of equations is $\mathbf{Ax} = \mathbf{b}$, where

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 3 & -2 \\ 2 & -1 & 1 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 10 \\ 7 \\ 5 \end{bmatrix}.$$

Now, $|\mathbf{A}| = \begin{vmatrix} 1 & 2 & 3 \\ 1 & 3 & -2 \\ 2 & -1 & 1 \end{vmatrix} = -30 \neq 0.$

That is, $\mathbf{A}$ is non-singular and hence $\mathbf{A^{-1}}$ exists.

$$\textbf{adj A} = \begin{bmatrix} 1 & -5 & -13 \\ -5 & -5 & 5 \\ -7 & 5 & 1 \end{bmatrix}.$$

Thus, $\mathbf{A}^{-1} = \dfrac{\textbf{adj A}}{|\mathbf{A}|} = \dfrac{1}{-30}\begin{bmatrix} 1 & -5 & -13 \\ -5 & -5 & 5 \\ -7 & 5 & 1 \end{bmatrix}.$

Therefore, $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} = \dfrac{1}{-30}\begin{bmatrix} 1 & -5 & -13 \\ -5 & -5 & 5 \\ -7 & 5 & 1 \end{bmatrix}\begin{bmatrix} 10 \\ 7 \\ 5 \end{bmatrix} = \dfrac{1}{30}\begin{bmatrix} 90 \\ 60 \\ 30 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}.$

Hence the required solution is $x = 3, y = 2, z = 1$.

Algorithm 5.4 (Matrix inverse method). This algorithm is used to solve a system of linear equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} = [a_{ij}]_{n\times n}$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$, $\mathbf{b} = (b_1, b_2, \ldots, b_n)^t$, by matrix inverse method.

```
Algorithm Matrix_Inverse_Method
 Step 1. Read the coefficient matrix $\mathbf{A} = [a_{ij}], i, j = 1, 2, \ldots, n$ and the right
         hand vector $\mathbf{b} = (b_1, b_2, \ldots, b_n)^t$.
 Step 2. Compute the inverse of A by the algorithm Matrix_Inverse.
 Step 3. If A is invertible then compute $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ and print $\mathbf{x} =$
         $(x_1, x_2, \ldots, x_n)^t$.
         Otherwise, print 'A is singular and the system has either no solution or
         has infinitely many solutions'.
end Matrix_Inverse_Method
```

Program 5.4

```
/* Program Matrix Inverse Method
   Program to find the solution of a system of linear
   equation by matrix inverse method. Partial pivoting
   is used to find matrix inverse. */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define zero 0.00001
float a[10][20],ai[10][10];
int n;
int matinv();
void main()
{
 int i,j;
 float b[10],x[10];
 printf("\nEnter the size of the coefficient matrix ");
 scanf("%d",&n);
 printf("Enter the elements rowwise ");
 for(i=1;i<=n;i++) for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
 printf("Enter the right hand vector ");
 for(i=1;i<=n;i++) scanf("%f",&b[i]);
 i=matinv();
 if(i==0)
  {
   printf("Coefficient matrix is singular: ");
   printf("The system has either no solution or many solutions");
   exit(0);
  }
 for(i=1;i<=n;i++)
   {
     x[i]=0;
     for(j=1;j<=n;j++) x[i]+=ai[i][j]*b[j];
   }
 printf("Solution of the system is\n ");
 for(i=1;i<=n;i++) printf("%8.5f ",x[i]);
} /* main */
/* function to find matrix inverse */
int matinv()
{
 int i,j,m,k; float temp;
/* augment the matrix A */
for(i=1;i<=n;i++) for(j=1;j<=n;j++) a[i][n+j]=0;
for(i=1;i<=n;i++) a[i][n+i]=1;
m=2*n;

for(k=1;k<=n;k++)
 {
    /* finds pivot element and its position */
    temp=fabs(a[k][k]);  j=k; /* initial maximum */
    for(i=k+1;i<=n;i++)
       if(temp<fabs(a[i][k]))
         {
           temp=fabs(a[i][k]); j=i;
         }
    if(fabs(a[j][k])<=zero) /* if a[j][k]=0 */
       {
         printf("The matrix is singular and is not invertible");
         return(0);
       }
    if(j!=k) /* interchange k and j rows */
       {
       for(i=1;i<=m;i++)
         {
            temp=a[j][i]; a[j][i]=a[k][i]; a[k][i]=temp;
         }
       }
    if(a[k][k]!=1)
       {
         temp=a[k][k];
         for(i=1;i<=m;i++) a[k][i]/=temp;
       }
    for(j=k+1;j<=n;j++) /* makes a[k+1][k] to a[n][k] zero */
        {
          temp=a[j][k];
          for(i=1;i<=m;i++)   a[j][i]-=temp*a[k][i];
        }
 } /* end of k loop */
 /* make left half of A to a unit matrix */
 for(k=2;k<=n;k++)
    for(j=k-1;j>=1;j--){
         temp=a[j][k];
         for(i=1;i<=m;i++) a[j][i]-=temp*a[k][i];
       }
 printf("\nThe inverse matrix is \n");
 for(i=1;i<=n;i++)
    for(j=n+1;j<=m;j++) ai[i][j-n]=a[i][j];
 return(1);
}/* matinv */
```

A sample of input/output:

```
Enter the size of the matrix 3
Enter the elements rowwise
```

```
0 1 2
3 -2 1
4 3 2
The inverse matrix is
-0.218750  0.125000  0.156250
-0.062500 -0.250000  0.187500
 0.531250  0.125000 -0.093750
```

Complexity

The time complexity of this method is $O(n^3)$ as the method involves computation of $\mathbf{A^{-1}}$.

5.5 Gauss Elimination Method

In this method, the variables are eliminated by a process of systematic elimination. Suppose the system has n variables and n equations of the form (5.1). This procedure reduces the system of linear equations to an equivalent upper triangular system which can be solved by back–substitution. To convert an upper triangular system, x_1 is eliminated from second equation to nth equation, x_2 is eliminated from third equation to nth equation, x_3 is eliminated from fourth equation to nth equation, and so on and finally, x_{n-1} is eliminated from nth equation.

To eliminate x_1, from second, third, $\cdots$, and nth equations the first equation is multiplied by $-\frac{a_{21}}{a_{11}}, -\frac{a_{31}}{a_{11}}, \ldots, -\frac{a_{n1}}{a_{11}}$ respectively and successively added with the second, third, $\cdots$, nth equations (assuming that $a_{11} \neq 0$). This gives

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\
a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \cdots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\
a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \cdots + a_{3n}^{(1)}x_n &= b_3^{(1)} \qquad (5.12)\\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \cdots \cdots \\
a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \cdots + a_{nn}^{(1)}x_n &= b_n^{(1)},
\end{aligned}$$

where

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}}a_{1j}; \quad i, j = 2, 3, \ldots, n.$$

Again, to eliminate x_2 from the third, forth, $\ldots$, and nth equations the second equation is multiplied by $-\frac{a_{32}^{(1)}}{a_{22}^{(1)}}, -\frac{a_{42}^{(1)}}{a_{22}^{(1)}}, \ldots, -\frac{a_{n2}^{(1)}}{a_{22}^{(1)}}$ respectively (assuming that $a_{22}^{(1)} \neq 0$), and successively added to the third, fourth, ..., and nth equations to get the new system of equations as

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \cdots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\ a_{33}^{(2)}x_3 + \cdots + a_{3n}^{(2)}x_n &= b_3^{(2)} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots & \cdots \cdots \\ a_{n3}^{(2)}x_3 + \cdots + a_{nn}^{(2)}x_n &= b_n^{(2)}, \end{aligned} \tag{5.13}$$

where

$$a_{ij}^{(2)} = a_{ij}^{(1)} - \frac{a_{i2}^{(1)}}{a_{22}^{(1)}} a_{2j}^{(1)}; \quad i, j = 3, 4, \ldots, n.$$

Finally, after eliminating x_{n-1}, the above system of equations become

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \cdots + a_{2n}^{(1)}x_n &= b_2^{(1)} \\ a_{33}^{(2)}x_3 + \cdots + a_{3n}^{(2)}x_n &= b_3^{(2)} \\ \cdots\cdots\cdots\cdots & \cdots \cdots \\ a_{nn}^{(n-1)}x_n &= b_n^{(n-1)}, \end{aligned} \tag{5.14}$$

where,

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} a_{kj}^{(k-1)};$$

$i, j = k+1, \ldots, n;\;\; k = 1, 2, \ldots, n-1$, and $a_{pq}^{(0)} = a_{pq};\;\; p, q = 1, 2, \ldots, n.$

Now, by back substitution, the values of the variables can be found as follows:

From last equation we have, $x_n = \dfrac{b_n^{(n-1)}}{a_{nn}^{(n-1)}}$, from the last but one equation, i.e., $(n-1)$th equation, one can find the value of x_{n-1} and so on. Finally, from the first equation we obtain the value of x_1.

The evaluation of the elements $a_{ij}^{(k)}$'s is a **forward substitution** and the determination of the values of the variables x_i's is a **back substitution** since we first determine the value of the last variable x_n.

Note 5.5.1 The method described above assumes that the diagonal elements are non-zero. If they are zero or nearly zero then the above simple method is not applicable to solve a linear system though it may have a solution. If any diagonal element is zero or very small then partial pivoting should be used to get a solution or a better solution.

It is mentioned earlier that if the system is diagonally dominant or real symmetric and positive definite then no pivoting is necessary.

Example 5.5.1 Solve the equations by Gauss elimination method.
$2x_1 + x_2 + x_3 = 4,\ \ x_1 - x_2 + 2x_3 = 2,\ \ 2x_1 + 2x_2 - x_3 = 3.$

Solution. Multiplying the second and third equations by 2 and 1 respectively and subtracting them from first equation we get

$$\begin{aligned} 2x_1 + x_2 + x_3 &= 4 \\ 3x_2 - 3x_3 &= 0 \\ -x_2 + 2x_3 &= 1. \end{aligned}$$

Multiplying third equation by –3 and subtracting from second equation we obtain

$$\begin{aligned} 2x_1 + x_2 + x_3 &= 4 \\ 3x_2 - 3x_3 &= 0 \\ 3x_3 &= 3. \end{aligned}$$

From the third equation $x_3 = 1$, from the second equations $x_2 = x_3 = 1$ and from the first equation $2x_1 = 4 - x_2 - x_3 = 2$ or, $x_1 = 1$.
Therefore the solution is $x_1 = 1, x_2 = 1, x_3 = 1$.

Example 5.5.2 Solve the following system of equations by Gauss elimination method (use partial pivoting).

$$\begin{aligned} x_2 + 2x_3 &= 5 \\ x_1 + 2x_2 + 4x_3 &= 11 \\ -3x_1 + x_2 - 5x_3 &= -12. \end{aligned}$$

Solution. The largest element (the pivot) in the coefficients of the variable x_1 is -3, attained at the third equation. So we interchange first and third equations

$$\begin{aligned} -3x_1 + x_2 - 5x_3 &= -12 \\ x_1 + 2x_2 + 4x_3 &= 11 \\ x_2 + 2x_3 &= 5. \end{aligned}$$

Multiplying the second equation by 3 and adding with the first equation we get,

$$\begin{aligned} -3x_1 + x_2 - 5x_3 &= -12 \\ x_2 + x_3 &= 3 \\ x_2 + 2x_3 &= 5 \end{aligned}$$

The second pivot is 1, which is at the positions a_{22} and a_{32}. Taking $a_{22} = 1$ as pivot to avoid interchange of rows. Now, subtracting the third equation from second equation, we obtain

$$\begin{aligned} -3x_1 + x_2 - 5x_3 &= -12 \\ x_2 + x_3 &= 3 \\ -x_3 &= -2. \end{aligned}$$

Now by back substitution, the values of x_3, x_2, x_1 are obtained as

$x_3 = 2$, $x_2 = 3 - x_3 = 1, x_1 = -\frac{1}{3}(-12 - x_2 + 5x_3) = 1.$

Hence the solution is $x_1 = 1$, $x_2 = 1$, $x_3 = 2$.

Algorithm 5.5 (Gauss elimination). This algorithm solves a system of equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} =[a_{ij}]_{n\times n}$, $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t, \mathbf{b} = (b_1, b_2, \ldots, b_n)^t$, by Gauss elimination method.

```
Algorithm Gauss_Elimination (using partial pivoting)
  Step 1. Read the matrix $\mathbf{A} = [a_{ij}], i, j = 1, 2, \ldots, n$.
  Step 2. Set $a_{in+1} = b_i$ for $i = 1, 2, \ldots, n$.
          Then the augmented matrix $\mathbf{A}$ becomes of order $n \times (n + 1)$.
  Step 3. Set $k = 1$.
  Step 4. Find the pivot from the elements $a_{kk}, a_{k+1k}, \ldots, a_{nk}$. Let $a_{jk}$ be the
          pivot.
  Step 5. If $a_{jk} = 0$ then // $\mathbf{A}$ is singular.//
          Print 'the system has either no solution or infinite many solutions', and
          Stop.
  Step 6. If $j \neq k$ then interchange the rows $j$ and $k$.
  Step 7. Subtract $a_{jk}/a_{kk}$ times the $k$th row to the $j$th row for $j = k + 1, k +
          2, \ldots, n$.
  Step 8. Increase $k$ by 1.
          If $k = n$ then //forward substitution is over.//
          go to Step 9.
          Otherwise go to Step 4.
  Step 9. //Back substitution.//
          $x_n = a_{n\,n+1}/a_{nn}$.
          Compute $x_i$'s using the expression
          $x_i = \frac{1}{a_{ii}}\left(a_{i\,n+1} - \sum_{j=i+1}^{n} a_{ij}x_j\right)$, for $i = n - 1, n - 2, \ldots, 1$.
  Step 10. Print $\mathbf{x} =(x_1, x_2, \ldots, x_n)^t$ as solution.
end Gauss_Elimination
```

Program 5.5

```
/*  Program Gauss-elimination
    Program to find the solution of a system of linear equations by
    Gauss elimination method. Partial pivoting is used. */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define zero 0.00001
void main()
{
int i,j,k,n,m;
float a[10][10],b[10],x[10],temp;
printf("\nEnter the size of the coefficient matrix ");
scanf("%d",&n);
printf("Enter the elements rowwise ");
for(i=1;i<=n;i++) for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
printf("Enter the right hand vector ");
for(i=1;i<=n;i++) scanf("%f",&b[i]);
/* augment A with b[i], i.e., a[i][n+1]=b[i] */
m=n+1;
for(i=1;i<=n;i++) a[i][m]=b[i];
for(k=1;k<=n;k++)
{
 /* finds pivot element and its position */
 temp=fabs(a[k][k]);  j=k; /* initial maximum */
 for(i=k+1;i<=n;i++)
    if(temp<fabs(a[i][k]))
       {
         temp=fabs(a[i][k]); j=i;
       }
 if(fabs(a[j][k])<=zero) /* if a[j][k]=0 */
   {
    printf("The matrix is singular:");
    printf("The system has either no solution or many solutions");
    exit(0);
   }
 if(j!=k) /* interchange k and j rows */
   {
     for(i=1;i<=m;i++)
```

```
      {
         temp=a[j][i]; a[j][i]=a[k][i]; a[k][i]=temp;
      }
    }
  for(j=k+1;j<=n;j++) /* makes a[k+1][k] to a[n][k] zero */
     {
        temp=a[j][k]/a[k][k];
        for(i=1;i<=m;i++)   a[j][i]-=temp*a[k][i];
     }
  } /* end of k loop */
/* forward substitution is over */
/* backward substitution */
x[n]=a[n][m]/a[n][n];
for(i=n-1;i>=1;i--)
   {
     x[i]=a[i][m];
     for(j=i+1;j<=n;j++) x[i]-=a[i][j]*x[j];
     x[i]/=a[i][i];
   }
printf("Solution of the system is\n ");
for(i=1;i<=n;i++) printf("%8.5f ",x[i]);
} /* main */
```

A sample of input/output:

```
Enter the size of the coefficient matrix 3
Enter the elements rowwise
1 1 1
2 -1 3
3 1 -1
Enter the right hand vector
3 16 -3
Solution of the system is
  1.00000 -2.00000  4.00000
```

5.6 Gauss-Jordan Elimination Method

In Gauss elimination method, the coefficient matrix is reduced to an upper triangular form and the solution is obtained by back substitution. But, the Gauss-Jordan method reduces the coefficient matrix to a diagonal matrix rather than upper triangular matrix and produces the solution of the system without using the back substitution. At the end of Gauss-Jordan method the system of equations (5.2) reduces to the following form:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1' \\ b_2' \\ \vdots \\ b_n' \end{bmatrix}. \tag{5.15}$$

The solution of the system is given by

$$x_1 = b_1', x_2 = b_2', \ldots, x_n = b_n'.$$

Thus the Gauss-Jordan method gives

$$\left[\mathbf{A} \vdots \mathbf{b}\right] \xrightarrow{\text{Gauss} - \text{Jordan}} \left[\mathbf{I} \vdots \mathbf{b}'\right]. \tag{5.16}$$

Generally, the Gauss-Jordan method is not used to solve a system of equations, because it is more costly than the Gauss elimination method. But, this method is used to find the matrix inverse (discussed in Section 5.3).

Example 5.6.1 Solve the following equations by Gauss-Jordan elimination method.

$$\begin{aligned} x_1 + x_2 + x_3 &= 3 \\ 2x_1 + 3x_2 + x_3 &= 6 \\ x_1 - x_2 - x_3 &= -3. \end{aligned}$$

Solution. Here $\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 1 \\ 1 & -1 & -1 \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 3 \\ 6 \\ -3 \end{bmatrix}$.

The augmented matrix $\left[\mathbf{A} \vdots \mathbf{b}\right]$ is

$$\begin{aligned} \left[\mathbf{A} \vdots \mathbf{b}\right] &= \left[\begin{array}{ccc|c} 1 & 1 & 1 & 3 \\ 2 & 3 & 1 & 6 \\ 1 & -1 & -1 & -3 \end{array}\right] \\ &\sim \left[\begin{array}{ccc|c} 1 & 1 & 1 & 3 \\ 0 & 1 & -1 & 0 \\ 0 & -2 & -2 & -6 \end{array}\right] \quad \begin{array}{l} R_2' = R_2 - 2R_1, \\ R_3' = R_3 - R_1 \end{array} \end{aligned}$$

$$\sim \left[\begin{array}{ccc:c} 1 & 1 & 1 & 3 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & -4 & -6 \end{array}\right] R_3' = R_3 + 2R_2$$

$$\sim \left[\begin{array}{ccc:c} 1 & 1 & 1 & 3 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 3/2 \end{array}\right] R_3' = \frac{1}{-4}R_3$$

$$\sim \left[\begin{array}{ccc:c} 1 & 0 & 2 & 3 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 3/2 \end{array}\right] R_1' = R_1 - R_2$$

$$\sim \left[\begin{array}{ccc:c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3/2 \\ 0 & 0 & 1 & 3/2 \end{array}\right] \begin{array}{l} R_1' = R_1 - 2R_3, \\ R_2' = R_2 + R_3 \end{array}$$

The equivalent system of equations are

$$\begin{aligned} x_1 &= 0 \\ x_2 &= 3/2 \\ x_3 &= 3/2. \end{aligned}$$

Hence the required solution is $x_1 = 0$, $x_2 = 3/2$, $x_3 = 3/2$.

5.7 Method of Matrix Factorization

5.7.1 LU Decomposition Method

This method is also known as factorization or **LU decomposition method** or **Crout's reduction method**.

Let the system of linear equations be

$$\mathbf{Ax} = \mathbf{b} \tag{5.17}$$

where $\mathbf{A}, \mathbf{x}, \mathbf{b}$ are given by (5.4).

The matrix $\mathbf{A}$ can be factorized into the form $\mathbf{A} = \mathbf{LU}$, where $\mathbf{L}$ and $\mathbf{U}$ are the lower and upper triangular matrices respectively. *If the principal minors of* $\mathbf{A}$ *are non-singular, i.e.,*

$$a_{11} \neq 0, \quad \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \neq 0, \quad \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \neq 0, \quad \cdots, \quad |\mathbf{A}| \neq 0 \tag{5.18}$$

then this factorization is possible and it is unique.

The matrices **L** and **U** are of the form

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix} \text{ and } \mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{bmatrix}. \tag{5.19}$$

The equation $\mathbf{Ax} = \mathbf{b}$ becomes $\mathbf{LUx} = \mathbf{b}$. Let $\mathbf{Ux} = \mathbf{z}$ then $\mathbf{Lz} = \mathbf{b}$, where $\mathbf{z} = (z_1, z_2, \ldots, z_n)^t$ is an intermediate variable vector. The value of $\mathbf{z}$ i.e., $z_1, z_2, \ldots, z_n$ can be determined by forward substitution in the following equations.

$$\begin{aligned} &l_{11}z_1 &&= b_1 \\ &l_{21}z_1 + l_{22}z_2 &&= b_2 \\ &l_{31}z_1 + l_{32}z_2 + l_{33}z_3 &&= b_3 \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots &&\cdots \\ &l_{n1}z_1 + l_{n2}z_2 + l_{n3}z_3 + \cdots + l_{nn}z_n &&= b_n. \end{aligned} \tag{5.20}$$

After determination of $\mathbf{z}$, one can compute the value of $\mathbf{x}$ i.e., $x_1, x_2, \ldots, x_n$ from the equation $\mathbf{Ux} = \mathbf{z}$ i.e., from the following equations by the backward substitution.

$$\begin{aligned} u_{11}x_1 + u_{12}x_2 + u_{13}x_3 + \cdots + u_{1n}x_n &= z_1 \\ u_{22}x_2 + u_{23}x_3 \cdots + z_{2n}x_n &= z_2 \\ u_{33}x_3 + u_{23}x_3 \cdots + u_{3n}x_n &= z_3 \\ \cdots\cdots\cdots\cdots\cdots\cdots &\cdots\ \cdots \\ u_{n-1n-1}x_{n-1} + u_{n-1n}x_n &= z_{n-1} \\ u_{nn}x_n &= z_n. \end{aligned} \tag{5.21}$$

When $u_{ii} = 1$, for $i = 1, 2, \ldots, n$, then the method is known as **Crout's decomposition method**. When $l_{ii} = 1$, for $i = 1, 2, \ldots, n$ then the method is known as **Doolittle's method** for decomposition. In particular, when $l_{ii} = u_{ii}$ for $i = 1, 2, \ldots, n$ then the corresponding method is called **Cholesky's decomposition method**.

Procedure to compute L and U

Here, we assume that $u_{ii} = 1$ for $i = 1, 2, \ldots, n$. From the relation $\mathbf{LU} = \mathbf{A}$, i.e., from

$$\begin{bmatrix} l_{11} & l_{11}u_{12} & l_{11}u_{13} & \cdots & l_{11}u_{1n} \\ l_{21} & l_{21}u_{12} + l_{22} & l_{21}u_{13} + l_{22}u_{23} & \cdots & l_{21}u_{1n} + l_{22}u_{2n} \\ l_{31} & l_{31}u_{12} + l_{32} & l_{31}u_{13} + l_{32}u_{23} + l_{33} & \cdots & l_{31}u_{1n} + l_{32} + u_{2n} + l_{33}u_{3n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ l_{n1} & l_{n1}u_{12} + l_{n2} & l_{n1}u_{13} + l_{n2}u_{23} + l_{n3} & \cdots & l_{n1}u_{1n} + l_{n2}u_{2n} + \cdots + l_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{31} & a_{32} & a_{33} & \cdots & a_{nn} \end{bmatrix}$$

we obtain

$l_{i1} = a_{i1}$, $i = 1, 2, \ldots, n$ and $u_{1j} = \dfrac{a_{1j}}{l_{11}}$, $j = 2, 3, \ldots, n$.

The second column of **L** and the second row of **U** are determined from the relations

$$\begin{aligned} l_{i2} &= a_{i2} - l_{i1}u_{12}, \quad \text{for } i = 2, 3, \ldots, n, \\ u_{2j} &= \frac{a_{2j} - l_{21}u_{1j}}{l_{22}} \quad \text{for} \quad j = 3, 4, \ldots, n. \end{aligned}$$

Next, third column of **L** and third row of **U** are determined in a similar way.

In general, l_{ij} and u_{ij} are given by

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}, \quad i \geq j \tag{5.22}$$

$$u_{ij} = \frac{a_{ij} - \sum\limits_{k=1}^{i-1} l_{ik}u_{kj}}{l_{ii}}, \quad i < j \tag{5.23}$$

$$u_{ii} = 1, \quad l_{ij} = 0, \quad j > i \quad \text{and} \quad u_{ij} = 0, \quad i > j.$$

Alternatively, the vectors **z** and **x** can be determined from the equations

$$\mathbf{z} = \mathbf{L}^{-1}\mathbf{b} \tag{5.24}$$

$$\text{and} \quad \mathbf{x} = \mathbf{U}^{-1}\mathbf{z}. \tag{5.25}$$

It may be noted that the computation of inverse of a triangular matrix is easier than an arbitrary matrix.

The inverse of **A** can also be determined from the relation

$$\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}. \tag{5.26}$$

Some properties of triangular matrices

Let $\mathbf{L} = [l_{ij}]$ and $\mathbf{U} = [u_{ij}]$ denote respectively the lower and upper triangular matrices.

- The determinant of a triangular matrix is the product of the diagonal elements i.e., $|\mathbf{L}| = \prod\limits_{i=1}^{n} l_{ii}$ and $|\mathbf{U}| = \prod\limits_{i=1}^{n} u_{ii}$.
- Product of two lower (upper) triangular matrices is a lower (upper) triangular matrix.

- The inverse of lower (upper) triangular matrix is also a lower (upper) triangular matrix.
- Since $\mathbf{A} = \mathbf{LU}$, $|\mathbf{A}| = |\mathbf{LU}| = |\mathbf{L}||\mathbf{U}| = \left(\prod_{i=1}^{n} l_{ii}\right)\left(\prod_{i=1}^{n} u_{ii}\right)$.

It may be remembered that the computation of determinant by LU decomposition is not a better method since it may fail or become unstable due to vanishing or near-vanishing leading minors.

Example 5.7.1 Factorize the matrix

$$A = \begin{bmatrix} 2 & -2 & 1 \\ 5 & 1 & -3 \\ 3 & 4 & 1 \end{bmatrix}$$

into the form **LU**, where **L** and **U** are lower and upper triangular matrices and hence solve the system of equations $2x_1-2x_2+x_3 = 2$, $5x_1+x_2-3x_3 = 0$, $3x_1+4x_2+x_3 = 9$. Determine $\mathbf{L^{-1}}$ and $\mathbf{U^{-1}}$ and hence find $\mathbf{A^{-1}}$. Also determine $|\mathbf{A}|$.

Solution. Let $\begin{bmatrix} 2 & -2 & 1 \\ 5 & 1 & -3 \\ 3 & 4 & 1 \end{bmatrix}$

$$= \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} l_{11} & l_{11}u_{12} & l_{11}u_{13} \\ l_{21} & l_{21}u_{12}+l_{22} & l_{21}u_{13}+l_{22}u_{23} \\ l_{31} & l_{31}u_{12}+l_{32} & l_{31}u_{13}+l_{32}u_{23}+l_{33} \end{bmatrix}.$$

Comparing both sides, we have

$l_{11} = 2,\ l_{21} = 5 \qquad l_{31} = 3$

$l_{11}u_{12} = -2$ or, $u_{12} = -2/l_{11} = -1$

$l_{11}u_{13} = 1$ or, $u_{13} = 1/l_{11} = 1/2$

$l_{21}u_{12} + l_{22} = 1$ or, $l_{22} = 1 - l_{21}u_{12} = 6$

$l_{31}u_{12} + l_{32} = 4$ or, $l_{32} = 4 - l_{31}u_{12} = 7$

$l_{21}u_{13} + l_{22}u_{23} = -3$ or, $u_{23} = (-3 - l_{21}u_{13})/l_{22} = -11/12$

$l_{31}u_{13} + l_{32}u_{23} + l_{33} = 1$ or, $l_{33} = 1 - l_{31}u_{13} - l_{32}u_{23} = 71/12.$

Hence **L** and **U** are given by

$$\mathbf{L} = \begin{bmatrix} 2 & 0 & 0 \\ 5 & 6 & 0 \\ 3 & 7 & \frac{71}{12} \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} 1 & -1 & \frac{1}{2} \\ 0 & 1 & -\frac{11}{12} \\ 0 & 0 & 1 \end{bmatrix}.$$

Second Part. The given system of equations can be written as $\mathbf{Ax} = \mathbf{b}$, where

$$\mathbf{A} = \begin{bmatrix} 2 & -2 & 1 \\ 5 & 1 & -3 \\ 3 & 4 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 0 \\ 9 \end{bmatrix}.$$

Using $\mathbf{A} = \mathbf{LU}$, the equation $\mathbf{Ax} = \mathbf{b}$ reduces to $\mathbf{LUx} = \mathbf{b}$. Let $\mathbf{Ux} = \mathbf{y}$. Then $\mathbf{Ly} = \mathbf{b}$.
From the relation $\mathbf{Ly} = \mathbf{b}$, we have

$$\begin{bmatrix} 2 & 0 & 0 \\ 5 & 6 & 0 \\ 3 & 7 & \frac{71}{12} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 9 \end{bmatrix},$$

That is,

$$\begin{aligned} 2y_1 \qquad\qquad &= 2, \\ 5y_1 + 6y_2 \qquad &= 0, \\ 3y_1 + 7y_2 + \frac{71}{12}y_3 &= 9. \end{aligned}$$

Solution of this system is $y_1 = 1, \;\; y_2 = -\frac{5}{6}, \;\; y_3 = 2$.
Thus $\mathbf{y} = (1, -5/6, 2)^t$.
Now, from the relation $\mathbf{Ux} = \mathbf{y}$ we have

$$\begin{bmatrix} 1 & -1 & \frac{1}{2} \\ 0 & 1 & -\frac{11}{12} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{5}{6} \\ 2 \end{bmatrix},$$

i.e.,

$$\begin{aligned} x_1 - x_2 + \frac{1}{2}x_3 &= 1 \\ x_2 - \frac{11}{12}x_3 &= -\frac{5}{6} \\ x_3 &= 2. \end{aligned}$$

Solution of this system of equations is

$$x_3 = 2, \;\; x_2 = -\frac{5}{6} + \frac{11}{12} \times 2 = 1, \;\; x_1 = 1 + x_2 - \frac{1}{2}x_3 = 1.$$

Hence the required solution is $x_1 = 1, \;\; x_2 = 1, \;\; x_3 = 2$.
Third Part. Applying the Gauss-Jordan method to find $\mathbf{L^{-1}}$. The necessary augmented matrix is

$$[\mathbf{L} \vdots \mathbf{I}] = \left[\begin{array}{ccc:ccc} 2 & 0 & 0 & 1 & 0 & 0 \\ 5 & 6 & 0 & 0 & 1 & 0 \\ 3 & 7 & \frac{71}{12} & 0 & 0 & 1 \end{array}\right]$$

$$\sim \left[\begin{array}{ccc:ccc} 1 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 6 & 0 & -\frac{5}{2} & 1 & 0 \\ 0 & 7 & \frac{71}{12} & -\frac{3}{2} & 0 & 1 \end{array}\right] R_1' = \frac{1}{2}R_1,\ R_2' = R_2 - 5R_1',\ R_3' = R_3 - 3R_1'$$

$$\sim \left[\begin{array}{ccc:ccc} 1 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & -\frac{5}{12} & \frac{1}{6} & 0 \\ 0 & 0 & \frac{71}{12} & \frac{17}{12} & -\frac{7}{6} & 1 \end{array}\right] R_2' = \frac{1}{6}R_2,\ R_3' = R_3 - 7R_2'$$

$$\sim \left[\begin{array}{ccc:ccc} 1 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 1 & 0 & -\frac{5}{12} & \frac{1}{6} & 0 \\ 0 & 0 & 1 & \frac{17}{71} & -\frac{14}{71} & \frac{12}{71} \end{array}\right].$$

Hence $\mathbf{L^{-1}} = \begin{bmatrix} 1/2 & 0 & 0 \\ -5/12 & 1/6 & 0 \\ 17/71 & -14/71 & 12/71 \end{bmatrix}$.

The value of $\mathbf{U^{-1}}$ can also be determined in a similar way. Here we apply another method based on the property *'inverse of a triangular matrix is a triangular matrix of same shape'*.

Let $\mathbf{U^{-1}} = \begin{bmatrix} 1 & b_{12} & b_{13} \\ 0 & 1 & b_{23} \\ 0 & 0 & 1 \end{bmatrix}$.

Then $\mathbf{U^{-1}U} = \mathbf{I}$ gives

$$\begin{bmatrix} 1 & b_{12} & b_{13} \\ 0 & 1 & b_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1/2 \\ 0 & 1 & -11/12 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

i.e.,
$$\begin{bmatrix} 1 & -1+b_{12} & \frac{1}{2} - \frac{11}{12}b_{12} + b_{13} \\ 0 & 1 & -\frac{11}{12} + b_{23} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Equating both sides

$-1 + b_{12} = 0$ or, $b_{12} = 1$, $\frac{1}{2} - \frac{11}{12}b_{12} + b_{13} = 0$ or, $b_{13} = -\frac{1}{2} + \frac{11}{12}b_{12} = \frac{5}{12}$

$-\frac{11}{12} + b_{23} = 0$ or, $b_{23} = \frac{11}{12}$.

Hence

$$\mathbf{U^{-1}} = \begin{bmatrix} 1 & 1 & 5/12 \\ 0 & 1 & 11/12 \\ 0 & 0 & 1 \end{bmatrix}.$$

Therefore,

$$\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1} = \begin{bmatrix} 1 & 1 & 5/12 \\ 0 & 1 & 11/12 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ -5/12 & 1/6 & 0 \\ 17/71 & -14/71 & 12/71 \end{bmatrix}$$

$$= \begin{bmatrix} 13/71 & 6/71 & 5/71 \\ -14/71 & -1/71 & 11/71 \\ 17/71 & -14/71 & 12/71 \end{bmatrix}.$$

Last Part. $|\mathbf{A}| = |\mathbf{L}||\mathbf{U}| = \left(2 \times 6 \times \frac{71}{12}\right) \times 1 = 71.$

Algorithm 5.6 (LU decomposition). This algorithm finds the solution of a system of linear equations using LU decomposition method. Assume that the principal minors of all order are non-zero.

```
Algorithm LU-decomposition
Let $\mathbf{Ax} = \mathbf{b}$ be the systems of equations and $\mathbf{A} = [a_{ij}]$, $\mathbf{b} = (b_1, b_2, \ldots, b_n)^t$,
$\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$.
//Assume that the principal minors of all order are non-zero.//
//Determine the matrices L and U.//
Step 1. Read the matrix $\mathbf{A} = [a_{ij}], i, j = 1, 2, \ldots, n$ and the right
        hand vector $\mathbf{b} = (b_1, b_2, \ldots, b_n)^t$.
Step 2. $l_{i1} = a_{i1}$ for $i = 1, 2, \ldots, n$; $u_{1j} = \frac{a_{1j}}{l_{11}}$ for $j = 2, 3, \ldots, n$;
        $u_{ii} = 1$ for $i = 1, 2, \ldots, n$.
Step 3. For $i, j = 2, 3, \ldots, n$ compute the following
        $l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}, \quad i \geq j$
        $u_{ij} = \left(a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}\right)/l_{ii}, \quad i < j.$
Step 4. //Solve the system $\mathbf{Lz} = \mathbf{b}$ by forward substitution.//
        $z_1 = \frac{b_1}{l_{11}}, \quad z_i = \frac{1}{l_{ii}}\left(b_i - \sum_{j=1}^{i-1} l_{ij}z_j\right)$ for $i = 2, 3, \ldots, n$.
Step 5. //Solve the system $\mathbf{Ux} = \mathbf{z}$ by backward substitution.//
        Set $x_n = z_n$;
        $x_i = z_i - \sum_{j=i+1}^{n} u_{ij}x_j$ for $i = n-1, n-2, \ldots, 1$.
        Print $x_1, x_2, \ldots, x_n$ as solution.
end LU-decomposition
```

Program 5.6

```
/* Program LU-decomposition
   Solution of a system of equations by LU decomposition method.
   Assume that all order principal minors are non-zero. */
#include<stdio.h>
void main()
{
 float a[10][10],l[10][10],u[10][10],z[10],x[10],b[10];
 int i,j,k,n;
 printf("\nEnter the size of the coefficient matrix ");
 scanf("%d",&n);
 printf("Enter the elements rowwise ");
 for(i=1;i<=n;i++) for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
 printf("Enter the right hand vector ");
 for(i=1;i<=n;i++) scanf("%f",&b[i]);
 /* computations of L and U matrices */
 for(i=1;i<=n;i++) l[i][1]=a[i][1];
 for(j=2;j<=n;j++) u[1][j]=a[1][j]/l[1][1];
 for(i=1;i<=n;i++) u[i][i]=1;
 for(i=2;i<=n;i++)
     for(j=2;j<=n;j++)
        if(i>=j)
          {
            l[i][j]=a[i][j];
            for(k=1;k<=j-1;k++) l[i][j]-=l[i][k]*u[k][j];
          }
        else
          {
            u[i][j]=a[i][j];
            for(k=1;k<=i-1;k++) u[i][j]-=l[i][k]*u[k][j];
            u[i][j]/=l[i][i];
          }
 printf("\nThe lower triangular matrix L\n");
 for(i=1;i<=n;i++)
    {
      for(j=1;j<=i;j++) printf("%f ",l[i][j]);
      printf("\n");
    }
 printf("\nThe upper triangular matrix U\n");
 for(i=1;i<=n;i++)
     {
       for(j=1;j<i;j++) printf("         ");
       for(j=i;j<=n;j++) printf("%f ",u[i][j]);
       printf("\n");
     }
 /* solve Lz=b by forward substitution */
 z[1]=b[1]/l[1][1];
 for(i=2;i<=n;i++)
     {
       z[i]=b[i];
       for(j=1;j<=i-1;j++) z[i]-=l[i][j]*z[j];
       z[i]/=l[i][i];
     }

 /* solve Ux=z by backward substitution */
 x[n]=z[n];
 for(i=n-1;i>=1;i--)
    {
      x[i]=z[i];
      for(j=i+1;j<=n;j++) x[i]-=u[i][j]*x[j];
    }
 printf("The solution is ");
 for(i=1;i<=n;i++) printf("%f ",x[i]);
} /* main */
```

A sample of input/output:

```
Enter the size of the coefficient matrix 3
Enter the elements rowwise
4 2 1
2 5 -2
1 -2 7
Enter the right hand vector
3 4 5

The lower triangular matrix L
4.000000
2.000000 4.000000
1.000000 -2.500000 5.187500
```

```
The upper triangular matrix U
1.000000 0.500000 0.250000
         1.000000 -0.625000
                  1.000000
The solution is -0.192771 1.325301 1.120482
```

5.8 Gauss Elimination Method to the Find Inverse of a Matrix

Conventionally, the Gauss elimination is applied to the augmented matrix $\left[\mathbf{A} \vdots \mathbf{b}\right]$. This method can also be applied to augmented matrix $\left[\mathbf{A} \vdots \mathbf{I}\right]$. In this method, the matrix $\mathbf{A}(= \mathbf{LU})$ becomes an upper triangular matrix $\mathbf{U}$ and the unit matrix $\mathbf{I}$ becomes the lower triangular matrix, which is the inverse of $\mathbf{L}$. Then the relation $\mathbf{AA^{-1}} = \mathbf{I}$ becomes $\mathbf{LUA^{-1}} = \mathbf{I}$ i.e.,

$$\mathbf{UA^{-1}} = \mathbf{L^{-1}}. \tag{5.27}$$

The left hand side of (5.27) is a lower triangular matrix and also the matrices $\mathbf{U}$ and $\mathbf{L^{-1}}$ are known. Hence by back substitution the matrix $\mathbf{A^{-1}}$ can be determined easily. This method is illustrated by an example below.

Example 5.8.1 Find the inverse of the matrix $\mathbf{A}$ using Gauss elimination method where $\mathbf{A} = \begin{bmatrix} 1 & 3 & 4 \\ 1 & 0 & 2 \\ -2 & 3 & 1 \end{bmatrix}$.

Solution. The augmented matrix $\left[\mathbf{A} \vdots \mathbf{I}\right]$ is

$$\left[\mathbf{A} \vdots \mathbf{I}\right] = \begin{bmatrix} 1 & 3 & 4 & \vdots & 1 & 0 & 0 \\ 1 & 0 & 2 & \vdots & 0 & 1 & 0 \\ -2 & 3 & 1 & \vdots & 0 & 0 & 1 \end{bmatrix}$$

$$\xrightarrow[R_3' \leftarrow R_3 + 2R_1]{R_2' \leftarrow R_2 - R_1} \begin{bmatrix} 1 & 3 & 4 & \vdots & 1 & 0 & 0 \\ 0 & -3 & -2 & \vdots & -1 & 1 & 0 \\ 0 & 9 & 9 & \vdots & 2 & 0 & 1 \end{bmatrix}$$

$$\xrightarrow{R_3' \leftarrow R_3 + 3R_2} \begin{bmatrix} 1 & 3 & 4 & \vdots & 1 & 0 & 0 \\ 0 & -3 & -2 & \vdots & -1 & 1 & 0 \\ 0 & 0 & 3 & \vdots & -1 & 3 & 1 \end{bmatrix}$$

Here $\mathbf{U} = \begin{bmatrix} 1 & 3 & 4 \\ 0 & -3 & -2 \\ 0 & 0 & 3 \end{bmatrix}$, $\mathbf{L^{-1}} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 3 & 1 \end{bmatrix}$.

Let $\mathbf{A^{-1}} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$.

Since, $\mathbf{UA^{-1}} = \mathbf{L^{-1}}$,

$$\begin{bmatrix} 1 & 3 & 4 \\ 0 & -3 & -2 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 3 & 1 \end{bmatrix}.$$

This implies

$x_{11} + 3x_{21} + 4x_{31} = 1$

$- 3x_{21} - 2x_{31} = -1$

$3x_{31} = -1$

These equations give $x_{31} = -\frac{1}{3}$, $x_{21} = \frac{5}{9}$, $x_{11} = \frac{2}{3}$.

Again, $x_{12} + 3x_{22} + 4x_{32} = 0$

$- 3x_{22} - 2x_{32} = 1$

$3x_{32} = 3$

Solution of this system is $x_{32} = 1$, $x_{22} = -1$, $x_{12} = -1$

$x_{13} + 3x_{23} + 4x_{33} = 0$

$- 3x_{23} - 2x_{33} = 0$

$3x_{33} = 1$

Then, $x_{33} = \frac{1}{3}$, $x_{23} = -\frac{2}{9}$, $x_{13} = -\frac{2}{3}$ is the solution of the above equations.

Hence $\mathbf{A^{-1}} = \begin{bmatrix} 2/3 & -1 & -2/3 \\ 5/9 & -1 & -2/9 \\ -1/3 & 1 & 1/3 \end{bmatrix}$.

5.9 Cholesky Method

If the coefficient matrix $\mathbf{A}$ is symmetric and positive definite then this method is applicable to solve the system $\mathbf{Ax} = \mathbf{b}$. This method is also known as **square-root** method.

Since $\mathbf{A}$ is symmetric then $\mathbf{A}$ can be written as

$$\mathbf{A} = \mathbf{LL^t}, \tag{5.28}$$

where $\mathbf{L} = [l_{ij}]$, $l_{ij} = 0$, $i < j$, a lower triangular matrix.

Also, $\mathbf{A}$ can be decomposed as

$$\mathbf{A} = \mathbf{UU^t}, \tag{5.29}$$

in terms of upper triangular form.

For (5.28), the equation $\mathbf{Ax} = \mathbf{b}$ becomes

$$\mathbf{LL^t x} = \mathbf{b}. \tag{5.30}$$

Let
$$\mathbf{L^t x} = \mathbf{z} \tag{5.31}$$
then
$$\mathbf{Lz} = \mathbf{b}. \tag{5.32}$$

The vector $\mathbf{z}$ can be obtained from (5.32) by forward substitution and the solution vector $\mathbf{x}$ are determined from the equation (5.31) by back substitution. Also, $\mathbf{z}$ and $\mathbf{x}$ can be determined by computing the inverse of $\mathbf{L}$ only as

$$\mathbf{z} = \mathbf{L^{-1}b} \quad \text{and} \quad \mathbf{x} = (\mathbf{L^t})^{-1}\mathbf{z} = (\mathbf{L^{-1}})^t\mathbf{z}. \tag{5.33}$$

The inverse of $\mathbf{A}$ can be determined as

$$\mathbf{A^{-1}} = (\mathbf{L^{-1}})^{\mathbf{t}}\mathbf{L^{-1}}.$$

Procedure to determine L

Since $\mathbf{A} = \mathbf{LL^t}$, then

$$\mathbf{A} = \begin{bmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ l_{i1} & l_{i2} & l_{i3} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{j1} & \cdots & l_{n1} \\ 0 & l_{22} & \cdots & l_{j2} & \cdots & l_{n2} \\ 0 & 0 & \cdots & l_{j3} & \cdots & l_{n3} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & \cdots & l_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} l_{11}^2 & l_{21}l_{11} & \cdots & l_{j1}l_{11} & \cdots & l_{n1}l_{11} \\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & \cdots & l_{j1}l_{21} + l_{j2}l_{22} & \cdots & l_{n1}l_{21} + l_{n2}l_{22} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ l_{i1}l_{11} & l_{21}l_{i1} + l_{22}l_{i2} & \cdots & l_{j1}l_{i1} + \cdots + l_{jj}l_{ij} & \cdots & l_{n1}l_{i1} + \cdots + l_{ni}l_{ii} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ l_{n1}l_{11} & l_{21}l_{n1} + l_{22}l_{n2} & \cdots & l_{j1}l_{n1} + \cdots + l_{jj}l_{nj} & \cdots & l_{n1}^2 + l_{n2}^2 + \cdots + l_{nn}^2 \end{bmatrix}.$$

Equating both sides, we find the following equations.

$$\begin{aligned} & l_{11} = (a_{11})^{1/2} \\ & l_{i1}^2 + l_{i2}^2 + \cdots + l_{ii}^2 = a_{ii} \quad \text{or} \quad l_{ii} = \Big(a_{ii} - \sum_{j=1}^{i-1} l_{ij}\Big)^{1/2}, \ i = 2, 3, \ldots, n \\ & l_{i1} = a_{i1}/l_{11}, \ \ i = 2, 3, \ldots, n \\ & l_{i1}l_{j1} + l_{i2}l_{j2} + \cdots + l_{ij}l_{jj} = a_{ij} \\ & \text{or,} \ \ l_{ij} = \frac{1}{l_{jj}}\Big(a_{ij} - \sum_{k=1}^{j-1} l_{jk}l_{ik}\Big)^{1/2}, \quad \text{for } i = j+1, j+2, \ldots, n \\ & l_{ij} = 0, \ i < j. \end{aligned} \tag{5.34}$$

Similarly, for the system of equations (5.29), the elements u_{ij} of $\mathbf{U}$ are given by

$$\begin{aligned}
&u_{nn} = (a_{nn})^{1/2}\\
&u_{in} = a_{in}/u_{nn}, \;\; i = 1, 2, \ldots, n-1\\
&u_{ij} = \tfrac{1}{u_{jj}}\left(a_{ij} - \sum_{k=j+1}^{n} u_{ik}u_{jk}\right),\\
&\;\text{for } i = n-2, n-3, \ldots, 1; \;\; j = i+1, i+2, \ldots, n-1\\
&u_{ii} = \left(a_{ii} - \sum_{k=i+1}^{n} u_{ik}^2\right)^{1/2}, \; i = n-1, n-2, \ldots, 1\\
&u_{ij} = 0, \; i > j.
\end{aligned} \tag{5.35}$$

Example 5.9.1 Solve the following system of equations by Cholesky method.

$$\begin{aligned}
2x_1 + x_2 - x_3 &= 6\\
x_1 - 3x_2 + 5x_3 &= 11\\
-x_1 + 5x_2 + 4x_3 &= 13.
\end{aligned}$$

Solution. The given system of equations is

$\mathbf{Ax} = \mathbf{b}$ where $\mathbf{x} = (x_1, x_2, x_3)^t$, $\mathbf{b} = (6, 11, 13)^t$, and $\mathbf{A} = \begin{bmatrix} 2 & 1 & 1\\ 1 & 3 & 2\\ 1 & 2 & 4 \end{bmatrix}$.

It is observed that $\mathbf{A}$ is symmetric and positive definite.

Let $\mathbf{L} = \begin{bmatrix} l_{11} & 0 & 0\\ l_{21} & l_{22} & 0\\ l_{31} & l_{32} & l_{33} \end{bmatrix}$.

Therefore, $\mathbf{LL^t} = \begin{bmatrix} l_{11}^2 & l_{11}l_{21} & l_{11}l_{31}\\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & l_{21}l_{31} + l_{22}l_{32}\\ l_{31}l_{11} & l_{31}l_{21} + l_{32}l_{22} & l_{31}^2 + l_{32}^2 + l_{33}^2 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1\\ 1 & 3 & 2\\ 1 & 2 & 4 \end{bmatrix}$.

Comparing both sides, we have

$l_{11}^2 = 2 \quad \text{or} \quad l_{11} = \sqrt{2}$

$l_{11}l_{21} = 1 \quad \text{or} \quad l_{21} = 1/\sqrt{2}$

$l_{11}l_{31} = 1 \quad \text{or} \quad l_{31} = 1/\sqrt{2}$

$l_{21}^2 + l_{22}^2 = 3 \quad \text{or} \quad l_{22} = (3 - \frac{1}{2})^{1/2} = \sqrt{\frac{5}{2}}$

$l_{31}l_{21} + l_{32}l_{22} = 2 \quad \text{or} \quad l_{32} = \frac{1}{l_{22}}(2 - l_{31}l_{21}) = 3/\sqrt{10}$

$l_{31}^2 + l_{32}^2 + l_{33}^2 = 4 \quad \text{or} \quad l_{33} = (4 - l_{31}^2 - l_{32}^2)^{1/2} = \sqrt{\frac{13}{5}}.$

Therefore, $\mathbf{L} = \begin{bmatrix} \sqrt{2} & 0 & 0\\ 1/\sqrt{2} & \sqrt{5/2} & 0\\ 1/\sqrt{2} & 3/\sqrt{10} & \sqrt{13/5} \end{bmatrix} = \begin{bmatrix} 1.41421 & 0 & 0\\ 0.70711 & 1.58114 & 0\\ 0.70711 & 0.94868 & 1.61245 \end{bmatrix}$.

From the relation $\mathbf{Lz} = \mathbf{b}$, we have

$$\begin{aligned} 1.41421z_1 &= 6 \\ 0.70711z_1 + 1.58114z_2 &= 11 \\ 0.70711z_1 + 0.94868z_2 + 1.61245z_3 &= 13. \end{aligned}$$

This gives $z_1 = 4.24265,\ \ z_2 = 5.05963,\ \ z_3 = 3.22491$.
Now, from the relation $\mathbf{L^t x} = \mathbf{z}$, we have

$$\begin{bmatrix} 1.41421 & 0.70711 & 0.70711 \\ 0 & 1.58114 & 0.94868 \\ 0 & 0 & 1.61245 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4.24265 \\ 5.05963 \\ 3.22491 \end{bmatrix}.$$

$$\text{i.e.,}\qquad \begin{aligned} 1.41421x_1 + 0.70711x_2 + 0.70711x_3 &= 4.24265 \\ 1.58114x_2 + 0.94868x_3 &= 5.05963 \\ 1.61245x_3 &= 3.22491 \end{aligned}$$

Solution of these equations is $x_3 = 2.00001,\ \ x_2 = 1.99993,\ \ x_1 = 0.99998$.
Hence the solution is $x_1 = 1.0000,\ \ x_2 = 2.0000,\ \ x_3 = 2.0000$, correct up to four decimal places.

5.10 Matrix Partition Method

When a matrix is very large and it is not possible to store the entire matrix into the primary memory of a computer at a time, then matrix partition method is used to find the inverse of a matrix. When a few more variables and consequently a few more equations are added to the original system then also this method is very useful.

Let the coefficient matrix $\mathbf{A}$ be partitioned as

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} & \vdots & \mathbf{C} \\ \cdots & \cdots & \cdots \\ \mathbf{D} & \vdots & \mathbf{E} \end{bmatrix} \tag{5.36}$$

where $\mathbf{B}$ is an $l \times l$ matrix, $\mathbf{C}$ is an $l \times m$ matrix, $\mathbf{D}$ is an $m \times l$ and $\mathbf{E}$ is an $m \times m$ matrix; and l, m are positive integers with $l + m = n$.

Let $\mathbf{A^{-1}}$ be partitioned as

$$\mathbf{A^{-1}} = \begin{bmatrix} \mathbf{P} & \vdots & \mathbf{Q} \\ \cdots & \cdots & \cdots \\ \mathbf{R} & \vdots & \mathbf{S} \end{bmatrix} \tag{5.37}$$

where the matrices $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ and $\mathbf{S}$ are of the same orders as those of the matrices $\mathbf{B}, \mathbf{C}, \mathbf{D}$ and $\mathbf{E}$ respectively. Then

$$\mathbf{AA}^{-1} = \begin{bmatrix} \mathbf{B} & \vdots & \mathbf{C} \\ \cdots & \cdots & \cdots \\ \mathbf{D} & \vdots & \mathbf{E} \end{bmatrix} \begin{bmatrix} \mathbf{P} & \vdots & \mathbf{Q} \\ \cdots & \cdots & \cdots \\ \mathbf{R} & \vdots & \mathbf{S} \end{bmatrix} = \begin{bmatrix} \mathbf{I_1} & \vdots & \mathbf{0} \\ \cdots & \cdots & \cdots \\ \mathbf{0} & \vdots & \mathbf{I_2} \end{bmatrix}, \tag{5.38}$$

where $\mathbf{I_1}$ and $\mathbf{I_2}$ are identity matrices of order l and m respectively. From (5.38), we have

$$\begin{aligned} \mathbf{BP} + \mathbf{CR} &= \mathbf{I_1} \\ \mathbf{BQ} + \mathbf{CS} &= \mathbf{0} \\ \mathbf{DP} + \mathbf{ER} &= \mathbf{0} \\ \mathbf{DQ} + \mathbf{ES} &= \mathbf{I_2}. \end{aligned}$$

Now, $\mathbf{BQ} + \mathbf{CS} = \mathbf{0}$ gives $\mathbf{Q} = -\mathbf{B}^{-1}\mathbf{CS}$ i.e., $\mathbf{DQ} = -\mathbf{DB}^{-1}\mathbf{CS}$.
Also, from $\mathbf{DQ} + \mathbf{ES} = \mathbf{I_2}$, we have $(\mathbf{E} - \mathbf{DB}^{-1}\mathbf{C})\mathbf{S} = \mathbf{I_2}$.
Therefore, $\mathbf{S} = (\mathbf{E} - \mathbf{DB}^{-1}\mathbf{C})^{-1}$.
Similarly, the other matrices are

$$\begin{aligned} \mathbf{Q} &= -\mathbf{B}^{-1}\mathbf{CS} \\ \mathbf{R} &= -(\mathbf{E} - \mathbf{DB}^{-1}\mathbf{C})^{-1}\mathbf{DB}^{-1} = -\mathbf{SDB}^{-1} \\ \mathbf{P} &= \mathbf{B}^{-1}(\mathbf{I_1} - \mathbf{CR}) = \mathbf{B}^{-1} - \mathbf{B}^{-1}\mathbf{CR}. \end{aligned}$$

It may be noted that, to find the inverse of $\mathbf{A}$, it is required to determine the inverses of two matrices $\mathbf{B}$ and $(\mathbf{E} - \mathbf{DB}^{-1}\mathbf{C})$ of order $l \times l$ and $m \times m$ respectively.

That is, to compute the inverse of the matrix $\mathbf{A}$ of order $n \times n$, the inverses of two lower order (roughly half) matrices are to be determined. If the matrices $\mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}$ are still large to fit in the computer memory, then further partition them.

Example 5.10.1 Find the inverse of the matrix $\mathbf{A} = \begin{bmatrix} 3 & 3 & 4 \\ 2 & 1 & 1 \\ 1 & 3 & 5 \end{bmatrix}$ using the matrix partition method. Hence find the solution of the system of equations

$$\begin{aligned} 3x_1 + 3x_2 + 4x_3 &= 5 \\ 2x_1 + x_2 + x_3 &= 7 \\ x_1 + 3x_2 + 5x_3 &= 6. \end{aligned}$$

Solution. Let the matrix $\mathbf{A}$ be partitioned as

$$\mathbf{A} = \left[\begin{array}{cc:c} 3 & 3 & 4 \\ 2 & 1 & 1 \\ \hdashline 1 & 3 & 5 \end{array}\right] = \left[\begin{array}{c:c} \mathbf{B} & \mathbf{C} \\ \hdashline \mathbf{D} & \mathbf{E} \end{array}\right], \text{ where } \mathbf{B} = \begin{bmatrix} 3 & 3 \\ 2 & 1 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 1 & 3 \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} 5 \end{bmatrix}$$

and $\mathbf{A}^{-1} = \left[\begin{array}{c:c} \mathbf{P} & \mathbf{Q} \\ \hdashline \mathbf{R} & \mathbf{S} \end{array}\right]$, where $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ and $\mathbf{S}$ are given by

$$\mathbf{S} = (\mathbf{E} - \mathbf{D}\mathbf{B}^{-1}\mathbf{C})^{-1}, \ \mathbf{R} = -\mathbf{S}\mathbf{D}\mathbf{B}^{-1}, \ \mathbf{P} = \mathbf{B}^{-1} - \mathbf{B}^{-1}\mathbf{C}\mathbf{R}, \ \mathbf{Q} = -\mathbf{B}^{-1}\mathbf{C}\mathbf{S}.$$

Now,

$$\mathbf{B}^{-1} = -\frac{1}{3}\begin{bmatrix} 1 & -3 \\ -2 & 3 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} -1 & 3 \\ 2 & -3 \end{bmatrix}.$$

$$\begin{aligned}
\mathbf{E} - \mathbf{D}\mathbf{B}^{-1}\mathbf{C} &= 5 - \begin{bmatrix} 1 & 3 \end{bmatrix} \frac{1}{3}\begin{bmatrix} -1 & 3 \\ 2 & -3 \end{bmatrix}\begin{bmatrix} 4 \\ 1 \end{bmatrix} = \frac{1}{3}. \\
\mathbf{S} &= 3 \\
\mathbf{R} &= -3\begin{bmatrix} 1 & 3 \end{bmatrix}\frac{1}{3}\begin{bmatrix} -1 & 3 \\ 2 & -3 \end{bmatrix} = \begin{bmatrix} -5 & 6 \end{bmatrix} \\
\mathbf{P} &= \mathbf{B}^{-1} - \mathbf{B}^{-1}\mathbf{C}\mathbf{R} = \frac{1}{3}\begin{bmatrix} -1 & 3 \\ 2 & -3 \end{bmatrix} - \frac{1}{3}\begin{bmatrix} -1 & 3 \\ 2 & -3 \end{bmatrix}\begin{bmatrix} 4 \\ 1 \end{bmatrix}\begin{bmatrix} -5 & 6 \end{bmatrix} \\
&= \begin{bmatrix} -2 & 3 \\ 9 & -11 \end{bmatrix}. \\
\mathbf{Q} &= -\frac{1}{3}\begin{bmatrix} -1 & 3 \\ 2 & -3 \end{bmatrix}\begin{bmatrix} 4 \\ 1 \end{bmatrix} 3 = \begin{bmatrix} 1 \\ -5 \end{bmatrix}
\end{aligned}$$

Therefore,

$$\mathbf{A}^{-1} = \begin{bmatrix} -2 & 3 & 1 \\ 9 & -11 & -5 \\ -5 & 6 & 3 \end{bmatrix}.$$

Hence,

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} = \begin{bmatrix} -2 & 3 & 1 \\ 9 & -11 & -5 \\ -5 & 6 & 3 \end{bmatrix}\begin{bmatrix} 5 \\ 7 \\ 6 \end{bmatrix} = \begin{bmatrix} 17 \\ -62 \\ 35 \end{bmatrix}.$$

Hence the required solution is $x_1 = 17, \ x_2 = -62, \ x_3 = 35.$

5.11 Solution of Tri-diagonal Systems

If the system of equations is of the form

$$\begin{aligned} b_1x_1 + c_1x_2 &= d_1 \\ a_2x_1 + b_2x_2 + c_2x_3 &= d_2 \\ a_3x_2 + b_3x_3 + c_3x_4 &= d_3 \\ \cdots\cdots\cdots\cdots\cdots\cdots & \quad \cdots \\ a_nx_{n-1} + b_nx_n &= d_n, \end{aligned} \tag{5.39}$$

then the coefficient matrix is

$$\mathbf{A} = \begin{bmatrix} b_1 & c_1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \cdots & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & a_n & b_n \end{bmatrix} \text{ and } \mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \tag{5.40}$$

It may be noted that the main diagonal and the adjacent coefficients on either side of it consist of only non-zero elements and all other elements are zero. The matrix is called tri-diagonal matrix and the system of equations is called a **tri-diagonal system**. These type of matrices occur frequently in the solution of ordinary and partial differential equations by finite difference method.

A tri-diagonal system can be solved using **LU** decomposition method.

Let $\mathbf{A} = \mathbf{LU}$ where

$$\mathbf{L} = \begin{bmatrix} \gamma_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \beta_2 & \gamma_2 & 0 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & \beta_{n-1} & \gamma_{n-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \beta_n & \gamma_n \end{bmatrix},$$

and
$$\mathbf{U} = \begin{bmatrix} 1 & \alpha_1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & \alpha_2 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & \alpha_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}.$$

Then

$$\mathbf{LU} = \begin{bmatrix} \gamma_1 & \gamma_1\alpha_1 & 0 & \cdots & 0 & 0 & 0 \\ \beta_2 & \alpha_1\beta_2 + \gamma_2 & \alpha_2\gamma_2 & \cdots & 0 & 0 & 0 \\ 0 & \beta_3 & \alpha_2\beta_3 + \gamma_3 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 & \beta_n & \beta_n\alpha_{n-1} + \gamma_n \end{bmatrix}.$$

Now, comparing the matrix **LU** with **A** and obtain the non-zero elements of **L** and **U** as

$$\begin{aligned}\gamma_1 &= b_1, \quad \gamma_i\alpha_i = c_i, \quad \text{or,} \quad \alpha_i = c_i/\gamma_i, \quad i = 1, 2, \ldots, n-1\\ \beta_i &= a_i, \;\; i = 2, \ldots, n\\ \gamma_i &= b_i - \alpha_{i-1}\beta_i = b_i - a_i\frac{c_{i-1}}{\gamma_{i-1}}, \quad i = 2, 3, \ldots, n.\end{aligned}$$

Thus the elements of **L** and **U** are given by the following relations.

$$\begin{aligned}\gamma_1 &= b_1,\\ \gamma_i &= b_i - \frac{a_ic_{i-1}}{\gamma_{i-1}}, \;\; i = 2, 3, \ldots, n \qquad (5.41)\\ \beta_i &= a_i, \;\; i = 2, 3, \ldots, n \qquad (5.42)\\ \alpha_i &= c_i/\gamma_i, \;\; i = 1, 2, \ldots, n-1.\end{aligned}$$

The solution of the equation (5.39) i.e., $\mathbf{Ax} = \mathbf{d}$ where $\mathbf{d} = (d_1, d_2, \ldots, d_n)^t$ can be obtained by solving $\mathbf{Lz} = \mathbf{d}$ using forward substitution and then solving $\mathbf{Ux} = \mathbf{z}$ using back substitution. The solution of $\mathbf{Lz} = \mathbf{d}$ is given by

$$z_1 = \frac{d_1}{b_1}, \;\; z_i = \frac{d_i - a_iz_{i-1}}{\gamma_i}, \;\; i = 2, 3, \ldots, n. \qquad (5.43)$$

The solution of the equation $\mathbf{Ux} = \mathbf{z}$ is

$$x_n = z_n, \;\; x_i = z_i - \alpha_ix_{i+1} = z_i - \frac{c_i}{\gamma_i}x_{i+1}, \qquad i = n-1, n-2, \ldots, 1. \qquad (5.44)$$

Example 5.11.1 Solve the following tri-diagonal system of equation.
$x_1 + x_2 = 3, \quad -x_1 + 2x_2 + x_3 = 6, \quad 3x_2 + 2x_3 = 12.$

Solution. Here $b_1 = c_1 = 1, a_2 = -1, b_2 = 2, c_2 = 1, a_3 = 3, b_3 = 2,$
$d_1 = 3, d_2 = 6, d_3 = 12.$
Therefore,

$$\begin{aligned}\gamma_1 &= b_1 = 1\\ \gamma_2 &= b_2 - a_2\frac{c_1}{\gamma_1} = 2 - (-1).1 = 3\\ \gamma_3 &= b_3 - a_3\frac{c_2}{\gamma_2} = 2 - 3.\frac{1}{3} = 1\\ z_1 &= \frac{d_1}{b_1} = 3, \quad z_2 = \frac{d_2 - a_2z_1}{\gamma_2} = 3, \quad z_3 = \frac{d_3 - a_3z_2}{\gamma_3} = 3\\ x_3 &= z_3 = 3, \quad x_2 = z_2 - \frac{c_2}{\gamma_2}x_3 = 2, \quad x_1 = z_1 - \frac{c_1}{\gamma_1}x_2 = 1.\end{aligned}$$

Hence the required solution is $x_1 = 1, x_2 = 2, x_3 = 3$.

It may be noted that the equation (5.43) and also (5.44) are valid only if $\gamma_i \neq 0$ for all $i = 1, 2, \ldots, n$. If any one of γ_i becomes zero at any stage then the method is not applicable. Actually, this method is based on LU decomposition technique and LU decomposition method is applicable and unique if principal minors of coefficient matrix of all orders are non-zero. But, if a minor becomes zero then a modification on (5.43) and (5.44) gives the solution of the tri-diagonal system.

Suppose $\gamma_k = 0$ and $\gamma_i \neq 0, i = 1, 2, \ldots, k-1$. Then let $\gamma_k = s$, a symbolic value of γ_k. Then the remaining $\gamma_i, i = k+1, \ldots, n$ are calculated using the equation (5.41). These γ's are used to calculate z_i and x_i using the formulae (5.43) and (5.44). The values of x_i's are obtained in terms of s. Lastly, the final solution is obtained by substituting $s = 0$. The following example illustrates this case.

Example 5.11.2 Solve the following system of equations.

$$\begin{aligned} x_1 + x_2 \qquad &= 3, \\ x_1 + x_2 - 3x_3 &= -3, \\ -2x_2 + 3x_3 &= 4. \end{aligned}$$

Solution. Here $b_1 = c_1 = 1, a_2 = b_2 = 1, c_2 = -3, a_3 = -2, b_3 = 3,$
$d_1 = 3, d_2 = -3, d_3 = 4.$
Therefore,

$$\begin{aligned} \gamma_1 &= b_1 = 1 \\ \gamma_2 &= b_2 - a_2\frac{c_1}{\gamma_1} = 1 - 1 = 0 \end{aligned}$$

Since $\gamma_2 = 0$, let $\gamma_2 = s$. Therefore,

$$\begin{aligned} \gamma_3 &= b_3 - a_3\frac{c_2}{\gamma_2} = 3 + 2\frac{-3}{s} = 3 - \frac{6}{s} \\ z_1 &= \frac{d_1}{b_1} = 3, \qquad z_2 = \frac{d_2 - a_2 z_1}{\gamma_2} = -\frac{6}{s}, \qquad z_3 = \frac{d_3 - a_3 z_2}{\gamma_3} = \frac{4s - 12}{3s - 6} \\ x_3 &= z_3 = \frac{4s - 12}{3s - 6}, \\ x_2 &= z_2 - \frac{c_2}{\gamma_2}x_3 = \frac{-6}{3s - 6}, \\ x_1 &= z_1 - \frac{c_1}{\gamma_1}x_2 = \frac{9s - 12}{3s - 6}. \end{aligned}$$

Substituting $s = 0$ to find the solution. The required solution is
$x_1 = 2, x_2 = 1, x_3 = 2.$

Algorithm 5.7 (Solution of a tri-diagonal system). This algorithm solves a tri-diagonal system of linear equations. Assume that the principal minors are non-zero.

```
Algorithm Tridiagonal
//Let the tri-diagonal system is of the form $\mathbf{Ax} = \mathbf{d}$ where $\mathbf{A}$ and $\mathbf{d}$ are given by (5.40).//
 Step 1. Read the matrix $\mathbf{A}$, i.e., the arrays $a_i, b_i, c_i, i = 2, 3, \ldots, n-1$
         and $b_1, c_1, a_n, b_n$.
 Step 2. Compute $\gamma_1 = b_1$, $\gamma_i = b_i - \dfrac{a_i c_{i-1}}{\gamma_{i-1}}$, $i = 2, 3, \ldots, n$.
 Step 3. Compute $z_1 = \dfrac{d_1}{b_1}$, $z_i = \dfrac{d_i - a_i z_{i-1}}{\gamma_i}$, $i = 2, 3, \ldots, n$.
 Step 4. Compute $x_n = z_n$, $x_i = z_i - \dfrac{c_i}{\gamma_i} x_{i+1}$, $i = n-1, n-2, \ldots, 1$.
 Step 5. Print $x_i$, $i = 1, 2, \ldots, n$ and Stop.
end Tridiagonal
```

Program 5.7

```
/* Program TriDiagonal
  Program to solve a tri-diagonal system of equations.
  The coefficient matrix are taken as a[i],b[i],c[i],
  i=2, 3, ..., n-1, b[1],c[1],a[n],b[n]. The right
  hand vector is d[i], i=1, 2, ..., n.*/
#include<stdio.h>
#include<stdlib.h>
float x[10]; /* x[i]is the solution of the tri-diagonal system */
void main()
{
 float a[10],b[10],c[10],d[10];
 int i,n; float y;
 float TriDiag(float [],float [],float [],float [],int);
 printf("Enter the size of the coefficient matrix ");
 scanf("%d",&n);
 printf("Enter first row (only non-zero elements) ");
 scanf("%f %f",&b[1],&c[1]);
 printf("Enter rows 2 to n-1 ");
 for(i=2;i<=n-1;i++) scanf("%f %f %f",&a[i],&b[i],&c[i]);
 printf("Enter last row ");
 scanf("%f %f",&a[n],&b[n]);
```

```
  printf("Enter the right hand vector ");
  for(i=1;i<=n;i++) scanf("%f",&d[i]);
  y=TriDiag(a,b,c,d,n);/* call of TriDiag to a dummy variable */
  printf("The solution is \n");
  for(i=1;i<=n;i++) printf("%f ",x[i]);
 } /* end of main */

float TriDiag(float a[10],float b[10],float c[10],float d[10],int n)
  {
  /* output x[i], i=1, 2,..., n, is a global variable.*/
  int i; float gamma[10],z[10];
  gamma[1]=b[1];
  for(i=2;i<=n;i++)
     {
      if(gamma[i-1]==0.0)
         {
            printf("A minor is zero: Method fails ");
            exit(0);
         }
      gamma[i]=b[i]-a[i]*c[i-1]/gamma[i-1];
     }
  z[1]=d[1]/gamma[1];
  for(i=2;i<=n;i++)
      z[i]=(d[i]-a[i]*z[i-1])/gamma[i];
  x[n]=z[n];
  for(i=n-1;i>=1;i--)
      x[i]=z[i]-c[i]*x[i+1]/gamma[i];
 /*  for(i=1;i<=n;i++) printf("%f ",x[i]); */
  return(x[0]);
 } /*end of TriDiag */
```

A sample of input/output:

```
Enter the size of the coefficient matrix 4
Enter first row (only non-zero elements) 1 2
Enter rows 2 to n-1
3 2 1
2 0 -1
Enter last row
1 2
```

```
Enter the right hand vector
3 2 1 1
The solution is
0.500000 1.250000 -2.000000 1.500000
```

5.12 Evaluation of Tri-diagonal Determinant [1]

For $n \geq 3$, a general tri-diagonal matrix $\mathbf{T} = [t_{ij}]_{n\times n}$ is of the form

$$\mathbf{T} = \begin{bmatrix} b_1 & c_1 & 0 & \cdots & \cdots & \cdots & 0 \\ a_2 & b_2 & c_2 & \cdots & \cdots & \cdots & 0 \\ 0 & a_3 & b_3 & \cdots & \cdots & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & a_n & b_n \end{bmatrix}$$

for any $t_{ij} = 0$ for $|i-j| \geq 2$.

The entire matrix can be stored using only three vectors $\mathbf{c} = (c_1, c_2, \ldots, c_{n-1})$, $\mathbf{a} = (a_2, a_3, \ldots, a_n)$, and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$. We define a vector $\mathbf{d} = (d_1, d_2, \ldots, d_n)$ as

$$d_i = \begin{cases} b_1 & \text{if } i = 1 \\ b_i - \dfrac{a_i}{d_{i-1}} c_{i-1} & \text{if } i = 2, 3, \ldots, n. \end{cases} \tag{5.45}$$

If $d_i = 0$ for any $i \leq n$, then, set $d_i = x$ (x is just a symbolic name) and continue to compute $d_{i+1}, d_{i+2}, \ldots, d_n$ in terms of x by using (5.45).

The product $P = \prod_{i=1}^{n} d_i$ (in general, this is a polynomial in x) evaluated at $x = 0$ is the value of $|\mathbf{T}|$. If P is free from x then the product P directly gives the value of $|\mathbf{T}|$.

Example 5.12.1 Find the values of the determinants of the following tri-diagonal matrices.

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & -2 \\ 0 & -3 & 4 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}.$$

Solution. For the matrix **A**,
$d_1 = 1, d_2 = 0$, so, set $d_2 = x$, $d_3 = b_3 - \dfrac{a_3}{d_2}c_2 = \dfrac{4x-6}{x}$.
Therefore, $P = d_1 d_2 d_2 = 4x - 6$, gives $|\mathbf{A}| = -6$.

[1] M.E.A.El-Mikkawy, A fast algorithm for evaluating nth order tri-diagonal determinants, J. Computational and Applied Mathematics, 166 (2004) 581-584.

<u>For the matrix **B**</u>,
$d_1 = 1, d_2 = b_2 - \frac{a_2}{d_1} c_1 = 1,\ d_3 = b_3 - \frac{a_3}{d_2} c_2 = 1.$
Therefore, $P = d_1 d_2 d_2 = 1.1.1 = 1$, gives $|\mathbf{B}| = 1$.

5.13 Vector and Matrix Norms

The **norm** of a vector is the size or length of that vector. The norm of a vector $\mathbf{x}$ is denoted by $\|\mathbf{x}\|$. This is a real number and satisfies the following conditions

$$\text{(i)}\ \ \|\mathbf{x}\| \geq 0 \quad \text{and} \quad \|\mathbf{x}\| = \mathbf{0} \quad \text{iff} \quad \mathbf{x} = \mathbf{0} \tag{5.46}$$

$$\text{(ii)}\ \ \|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\| \quad \text{for any real scalar } \alpha \tag{5.47}$$

$$\text{(iii)}\ \ \|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \ \ \text{(triangle inequality).} \tag{5.48}$$

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ be any vector. The commonly used norms are

$$\text{(i)}\ \ \|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i| \tag{5.49}$$

$$\text{(ii)}\ \ \|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} \ \ \text{(Euclidean norm)} \tag{5.50}$$

$$\text{(iii)}\ \ \|\mathbf{x}\|_\infty = \max_i |x_i| \ \text{(maximum norm or uniform norm).} \tag{5.51}$$

Let $\mathbf{A}$ and $\mathbf{B}$ be two matrices such that $\mathbf{A}+\mathbf{B}$ and $\mathbf{AB}$ are defined. The **norm of a matrix** $\mathbf{A} = [a_{ij}]$ is denoted by $\|\mathbf{A}\|$, which satisfies the following conditions

$$\text{(i)}\ \ \|\mathbf{A}\| \geq 0 \text{ and } \|\mathbf{A}\| = 0 \text{ iff } \mathbf{A} = \mathbf{0} \tag{5.52}$$

$$\text{(ii)}\ \ \|\alpha\mathbf{A}\| = |\alpha|\|\mathbf{A}\|,\ \ \alpha \text{ is a real scalar} \tag{5.53}$$

$$\text{(iii)}\ \ \|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\| \tag{5.54}$$

$$\text{(iv)}\ \ \|\mathbf{AB}| \leq \|\mathbf{A}\|\|\mathbf{B}\|. \tag{5.55}$$

From (5.55), it can be verified that

$$\|\mathbf{A}^{\mathbf{k}}\| \leq \|\mathbf{A}\|^k, \tag{5.56}$$

for any positive integer k.

Like the vector norms, the matrix norms may be defined as

$$\text{(i)}\quad \|\mathbf{A}\|_1 = \max_j \sum_i |a_{ij}| \text{ (the column norm)} \tag{5.57}$$

$$\text{(ii)}\quad \|\mathbf{A}\|_2 = \sqrt{\sum_i \sum_j |a_{ij}|^2} \quad \text{(the Euclidean norm)} \tag{5.58}$$

$$\text{(iii)}\quad \|\mathbf{A}\|_\infty = \max_i \sum_j |a_{ij}| \quad \text{(the row norm).} \tag{5.59}$$

The Euclidean norm is also known as **Erhard-Schmidt norm** or **Schur norm** or the **Frobenius norm**.

The concept of matrix norm is used to study the **stability** of a system of equations. It is also used to study the convergence of iterative methods to solve the linear system of equations.

Example 5.13.1 Find the matrix norms $\|\mathbf{A}\|_1, \|A\|_2$ and $\|\mathbf{A}\|_\infty$ for the matrix $\mathbf{A} = \begin{bmatrix} 2 & 3 & 4 \\ 0 & -1 & 5 \\ 3 & 2 & 6 \end{bmatrix}$.

Solution.

$$\begin{aligned} \|\mathbf{A}\|_1 &= \max\{2+0+3, 3-1+2, 4+5+6\} = 15 \\ \|\mathbf{A}\|_2 &= \sqrt{2^2+3^2+4^2+0^2+(-1)^2+5^2+3^2+2^2+6^2} = \sqrt{104} \text{ and} \\ \|\mathbf{A}\|_\infty &= \max\{2+3+4, 0-1+5, 3+2+6\} = 11. \end{aligned}$$

5.14 Ill-Conditioned Linear Systems

Before introduction of ill-conditioned system, let us consider the following system of equations.

$$\begin{aligned} x + 3y &= 4 \\ \frac{1}{3}x + y &= 1.33. \end{aligned} \tag{5.60}$$

Note that this system of equations has no solution. But, if we take the approximate value of $\frac{1}{3}$ as 0.3 then (5.60) becomes

$$\begin{aligned} x + 3y &= 4 \\ 0.3x + y &= 1.33. \end{aligned} \tag{5.61}$$

The solution of (5.61) is $x = 0.1,\ y = 1.3$.

If the approximation of $\frac{1}{3}$ is taken as 0.33 then the solution of the system of equations

$$\begin{aligned} x + 3y &= 4 \\ 0.33x + y &= 1.33 \end{aligned} \tag{5.62}$$

is $x = 1,\ y = 1$.

The approximations 0.333 and 0.3333 of $\frac{1}{3}$ give the following systems

$$\begin{aligned} x + 3y &= 4 \\ 0.333x + y &= 1.33. \end{aligned} \tag{5.63}$$

whose solution is $x = 10,\ y = -2$ and

$$\begin{aligned} x + 3y &= 4 \\ 0.3333x + y &= 1.33. \end{aligned} \tag{5.64}$$

with solution $x = 100,\ y = -32$.

The systems (5.60)-(5.64) and their solutions indicate a dangerous situation. It may be noted that the different approximations of $\frac{1}{3}$ give high variations in their solutions. What is conclusion about the above systems? We may conclude that the systems are **unstable**. That is, a small change in the coefficients of the system produces large change in the solution. These systems are called **ill-conditioned** or **ill-posed** system. On the other hand, if the change in the solution is small for small changes in the coefficients, then the system is called **well-conditioned** or **well-posed** system.

Let the system of equations be

$$\mathbf{Ax} = \mathbf{b}. \tag{5.65}$$

Let $\mathbf{A}'$ and $\mathbf{b}'$ be matrices obtained from $\mathbf{A}$ and $\mathbf{b}$ by introducing small changes in $\mathbf{A}$ and $\mathbf{b}$ and let $\mathbf{y}$ be the solution of the new system. That is,

$$\mathbf{A}'\mathbf{y} = \mathbf{b}'. \tag{5.66}$$

The system (5.65) is called ill-conditioned when the changes in $\mathbf{y}$ are too large compared to those in $\mathbf{x}$. Otherwise, the system is called well-conditioned. If a system is ill-conditioned then the corresponding coefficient matrix is called an **ill-conditioned matrix**.

The system (5.62) is ill-conditioned and the corresponding coefficient matrix $\begin{bmatrix} 1 & 3 \\ 0.33 & 1 \end{bmatrix}$ is an ill-conditioned matrix.

Generally, ill-condition occurs when $|\mathbf{A}|$ is small. To measure the ill-condition of a matrix, different methods are available. One of the useful method is introduced here.

The quantity $Cond(\mathbf{A})$, called the **condition number** of the matrix, defined by

$$Cond(\mathbf{A}) = \|\mathbf{A}\| \, \|\mathbf{A^{-1}}\| \tag{5.67}$$

where $\|\mathbf{A}\|$ is any matrix norm, gives a **measure of the condition of the matrix A**. The large value of $Cond(\mathbf{A})$ indicates the ill-condition of a matrix or the associated system of equations.

Let $\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 0.33 & 1 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 4 & 3 \\ 3 & 5 \end{bmatrix}$ be two matrices.

Then $\mathbf{A^{-1}} = \begin{bmatrix} 100 & -300 \\ -33 & 100 \end{bmatrix}$ and $\mathbf{B^{-1}} = \frac{1}{11}\begin{bmatrix} 5 & -3 \\ -3 & 4 \end{bmatrix}$.

The Euclidean norms, $\|\mathbf{A}\|_2 = \sqrt{1 + 9 + 0.10890 + 1} = 3.3330$ and $\|\mathbf{A^{-1}}\|_2 = 333.3001$.

Therefore, $Cond(\mathbf{A}) = \|\mathbf{A}\|_2 \times \|\mathbf{A^{-1}}\|_2 = 1110.88945$, a very large number. Hence $\mathbf{A}$ is ill-conditioned.

Where as $\|\mathbf{B}\|_2 = 7.68115$ and $\|\mathbf{B^{-1}}\|_2 = 0.69829$
Then $Cond(\mathbf{B}) = 5.36364$, a relatively small quantity.

Therefore, $\mathbf{B}$ is well-conditioned matrix.

Another indicator of ill-conditioning matrix is presented below.

Let $\mathbf{A} = [a_{ij}]$ be the matrix and $r_i = \left(\sum\limits_{j=1}^{n} a_{ij}^2\right)^{1/2}$. The quantity

$$\nu(\mathbf{A}) = \frac{|\mathbf{A}|}{r_1 r_2 \cdots r_n} \tag{5.68}$$

measures the smallness of the determinant of $\mathbf{A}$. If ν is very small compared to 1, then the matrix $\mathbf{A}$ is ill-conditioned, otherwise $\mathbf{A}$ is well-conditioned.

For the matrix $\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 0.33 & 1 \end{bmatrix}$, $r_1 = \sqrt{10},\ r_2 = 1.05304,\ |\mathbf{A}| = 0.01,$
$\nu(\mathbf{A}) = \dfrac{0.01}{\sqrt{10} \times 1.05304} = 0.003$ and for $\mathbf{B} = \begin{bmatrix} 4 & 3 \\ 3 & 5 \end{bmatrix}$, $r_1 = 5,\ r_2 = \sqrt{34},$
$|\mathbf{B}| = 11,\ \nu(\mathbf{B}) = \dfrac{11}{5 \times \sqrt{34}} = 0.37730.$

Hence $\mathbf{A}$ is ill-conditioned while $\mathbf{B}$ is well-conditioned.

5.14.1 Method to solve ill-conditioned system

Some methods are available to solve an ill-conditioned system of linear equations. One straight forward technique is to carry out the calculations with more number of significant digits. But, the calculations with more significant digits is time-consuming. One suggested method is to improve upon the accuracy of the approximate solution by an iterative method. This iterative method is discussed below.

Let the system of equations be

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, 2, \ldots, n. \tag{5.69}$$

Let $\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_n$ be an approximate solution of (5.69). Since this is an approximate solution, $\sum\limits_{j=1}^{n} a_{ij}\widetilde{x}_j$ is not necessarily equal to b_i. Let $b_i = \widetilde{b}_i$ for this approximate solution.

Then, for this solution, (5.69) becomes

$$\sum_{j=1}^{n} a_{ij}\widetilde{x}_j = \widetilde{b}_i, \quad i = 1, 2, \ldots, n. \tag{5.70}$$

Subtracting (5.70) from (5.69), we obtain

$$\sum_{j=1}^{n} a_{ij}(x_j - \widetilde{x}_j) = (b_i - \widetilde{b}_i)$$

$$\text{i.e.,} \quad \sum_{j=1}^{n} a_{ij}\varepsilon_i = d_i \tag{5.71}$$

where $\varepsilon_i = x_i - \widetilde{x}_i, d_i = b_i - \widetilde{b}_i, i = 1, 2, \ldots, n$.

Now, the solution for ε_i's is obtained by solving the system (5.71). Hence the new solution is given by $x_i = \varepsilon_i + \widetilde{x}_i$ and these values are better approximations to x_i's. This technique can be repeated again to improve the accuracy.

5.15 Generalized Inverse (g-inverse)

The conventional matrix inverse (discussed in Section 5.3) is widely used in many areas of science and engineering. It is also well known that conventional inverses can be determined only for square non-singular matrices. But, in many areas of science and engineering such as statistics, data analysis etc. some kind of weak inverses of singular square and rectangular matrices are very much essential. The inverses of such types of matrices are known as **generalized inverse** or **g-inverse**. A number of works have been done during last three decades on g-inverse. The generalized inverse of an $m \times n$ matrix $\mathbf{A}$ is a matrix $\mathbf{X}$ of size $n \times m$. But, different types of generalized inverses are defined by various authors. The following matrix equations are used to classify the different types of generalized inverses for the matrix $\mathbf{A}$:

$$\begin{array}{ll} \text{(i)} & \mathbf{AXA} = \mathbf{A} \\ \text{(ii)} & \mathbf{XAX} = \mathbf{X} \\ \text{(iii)} & \mathbf{AX} = (\mathbf{AX})^* \\ \text{(iv)} & \mathbf{XA} = (\mathbf{XA})^*, \quad (* \text{ denotes the conjugate transpose}). \end{array} \tag{5.72}$$

The matrix $\mathbf{X}$ is called
(a) a generalized inverse of $\mathbf{A}$, denoted by $\mathbf{A}^-$, if (i) holds;
(b) a reflexive generalized inverse of $\mathbf{A}$, denoted by $\mathbf{A}_\mathbf{r}^-$, if (i) and (ii) hold ;
(c) a minimum norm inverse of $\mathbf{A}$, denoted by $\mathbf{A}_\mathbf{m}^-$, if (i) and (iv) hold;
(d) a least-square inverse of $\mathbf{A}$, denoted by $\mathbf{A}_\mathbf{l}^-$, if (i) and (iii) hold;
(e) the Moore-Penrose inverse of $\mathbf{A}$, denoted by $\mathbf{A}^+$, if (i), (ii), (iii) and (iv) hold.

Only the Moore-Penrose inverse $\mathbf{A}^+$ is unique and other all inverses are not unique. One interesting result is that, when $\mathbf{A}$ is non-singular then all these inverses reduce to $\mathbf{A}^{-\mathbf{1}}$. Due to the uniqueness of $\mathbf{A}^+$, this generalized inverse is widely used.

Some important properties are presented below:

$$\text{(i)}\ \ (\mathbf{A}^+)^+ = \mathbf{A}; \tag{5.73}$$
$$\text{(ii)}\ \ (\mathbf{A}^+)^\mathbf{t} = (\mathbf{A}^\mathbf{t})^+; \tag{5.74}$$
$$\text{(iii)}\ \ \mathbf{A}^{-\mathbf{1}} = \mathbf{A}^+,\ \text{if } \mathbf{A} \text{ is non-singular} \tag{5.75}$$
$$\text{(iv)}\ \ \text{In general, } (\mathbf{AB})^+ \neq \mathbf{B}^+\mathbf{A}^+. \tag{5.76}$$

Let $\mathbf{A}$ be a matrix of order $m \times n$.

$$\text{(v)}\quad \text{If rank of } \mathbf{A} \text{ is } 0,\ \text{then } \mathbf{A}^+ \text{ is a null matrix of order } n \times m; \tag{5.77}$$
$$\text{(vi)}\quad \text{If rank of } \mathbf{A} \text{ is } 1,\ \text{then } \mathbf{A}^+ = \frac{\mathbf{1}}{\mathbf{trace}(\mathbf{AA^t})}\mathbf{A^t}; \tag{5.78}$$
$$\text{(vii)}\quad \text{If rank of } \mathbf{A} \text{ is } n,\ \text{then } \mathbf{A}^+ = (\mathbf{A^tA})^{-\mathbf{1}}\mathbf{A^t}; \tag{5.79}$$
$$\text{(viii)}\quad \text{If rank of } \mathbf{A} \text{ is } m,\ \text{then } \mathbf{A}^+ = \mathbf{A^t}(\mathbf{AA^t})^{-\mathbf{1}}. \tag{5.80}$$

The g-inverse $\mathbf{A}^+$ of the matrix $\mathbf{A}$ is used to solve the system of equations $\mathbf{Ax} = \mathbf{b}$, $(\mathbf{b} \neq \mathbf{0})$ where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{x}$ and $\mathbf{b}$ are respectively $n \times 1$ and $m \times 1$ vectors. The solution of $\mathbf{Ax} = \mathbf{b}$ is given by

$$\mathbf{x} = \mathbf{A}^+\mathbf{b} \tag{5.81}$$

and this solution is known as minimum norm[2] and least-squares[3] solution.

5.15.1 Greville's algorithm for Moore-Penrose inverse

Greville[4] presented a recursive algorithm to find the Moore-Penrose inverse of a matrix.

[2]The Euclidean norm $\|\mathbf{x}\| = \sqrt{\mathbf{x} * \mathbf{x}}$ is minimum for any choice of arbitrary inverse.

[3]Least-squares solution minimizes $\|\mathbf{Ax} - \mathbf{b}\|$ for an inconsistent system.

[4]For the proof of the algorithm see Greville, T.N.E, The pseudo-inverse of a rectangular or singular matrix and its application to the solution of system of linear equations, SIAM Review, 1 (1959) 38-43.

Let

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1k} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2k} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mk} & \cdots & a_{mn} \end{bmatrix} = (\alpha_\mathbf{1} \;\; \alpha_\mathbf{2} \;\; \ldots \;\; \alpha_\mathbf{k} \;\; \ldots \;\; \alpha_\mathbf{n}) \tag{5.82}$$

where $\alpha_\mathbf{k} = \begin{bmatrix} a_{1k} \\ a_{2k} \\ \vdots \\ a_{mk} \end{bmatrix}$, the kth column of the matrix $\mathbf{A}$.

Also, let $\mathbf{A_k}$ be the matrix formed by the first k columns, i.e., $\mathbf{A_k} = (\alpha_\mathbf{1} \;\; \alpha_\mathbf{2} \;\; \ldots \;\; \alpha_\mathbf{k})$. Then

$$\mathbf{A_k} = (\mathbf{A_{k-1}} \quad \alpha_\mathbf{k}).$$

The proposed algorithm is recursive and recursion is started with

$$\begin{aligned} &\mathbf{A_1^+} = \mathbf{0} \text{ if } \alpha_\mathbf{1} = \mathbf{0} \text{ (null column) else} \\ &\mathbf{A_1^+} = (\alpha_\mathbf{1}^\mathbf{t}\alpha_\mathbf{1})^{-\mathbf{1}}\alpha_\mathbf{1}^\mathbf{t}. \end{aligned} \tag{5.83}$$

Let the column vectors be

$$\delta_\mathbf{k} = \mathbf{A_{k-1}^+}\alpha_\mathbf{k} \tag{5.84}$$

$$\text{and } \gamma_\mathbf{k} = \alpha_\mathbf{k} - \mathbf{A_{k-1}}\delta_\mathbf{k}. \tag{5.85}$$

If $\gamma_\mathbf{k} \neq \mathbf{0}$, then compute

$$\beta_\mathbf{k} = \gamma_\mathbf{k}^+ = (\gamma_\mathbf{k}^\mathbf{t}\gamma_\mathbf{k})^{-\mathbf{1}}\gamma_\mathbf{k}^\mathbf{t}; \tag{5.86}$$

$$\text{else} \qquad \beta_\mathbf{k} = (\mathbf{1} + \delta_\mathbf{k}^\mathbf{t}\delta_\mathbf{k})^{-\mathbf{1}}\delta_\mathbf{k}^\mathbf{t}\mathbf{A_{k-1}^+}. \tag{5.87}$$

Then, the matrix $\mathbf{A_k^+}$ is given by

$$\mathbf{A_k^+} = \begin{bmatrix} \mathbf{A_{k-1}^+} - \delta_\mathbf{k}\beta_\mathbf{k} \\ \beta_\mathbf{k} \end{bmatrix}. \tag{5.88}$$

The process is repeated for $k = 1, 2, \ldots, n$.

Example 5.15.1 Obtain the g-inverse (Moore-Penrose inverse) of $\begin{bmatrix} 2 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 2 \end{bmatrix}$
and use it to solve the following system of equations

$$\begin{aligned} 2x_1 + x_2 \quad\;\; + x_4 &= 4 \\ x_1 \quad\;\; - x_3 \quad\;\; &= 0 \\ x_2 + x_3 + 2x_4 &= 4. \end{aligned}$$

Solution. Here $\alpha_1 = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$, $\alpha_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, $\alpha_3 = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$, $\alpha_4 = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$, $\mathbf{A_1} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$.

$$\mathbf{A_1^+} = (\alpha_1^t\alpha_1)^{-1}\alpha_1^t = \left(\begin{bmatrix} 2 & 1 & 0 \end{bmatrix}\begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}\right)^{-1}\begin{bmatrix} 2 & 1 & 0 \end{bmatrix} = \frac{1}{5}\begin{bmatrix} 2 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{2}{5} & \frac{1}{5} & 0 \end{bmatrix}$$

$$\delta_2 = \mathbf{A_1^+}\alpha_2 = \begin{bmatrix} \frac{2}{5} & \frac{1}{5} & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{5} \end{bmatrix}$$

$$\gamma_2 = \alpha_2 - \mathbf{A_1}\delta_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}\begin{bmatrix} \frac{2}{5} \end{bmatrix} = \begin{bmatrix} \frac{1}{5} \\ -\frac{2}{5} \\ 1 \end{bmatrix} \neq \mathbf{0} \text{ (the null column vector).}$$

$$\text{Hence } \beta_2 = \gamma_2^+ = (\gamma_2^t\gamma_2)^{-1}\gamma_2^t = \left(\begin{bmatrix} \frac{1}{5} & -\frac{2}{5} & 1 \end{bmatrix}\begin{bmatrix} \frac{1}{5} \\ -\frac{2}{5} \\ 1 \end{bmatrix}\right)^{-1}\begin{bmatrix} \frac{1}{5} & -\frac{2}{5} & 1 \end{bmatrix}$$

$$= \frac{5}{6}\begin{bmatrix} \frac{1}{5} & -\frac{2}{5} & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & -\frac{1}{3} & \frac{5}{6} \end{bmatrix}$$

$$\delta_2\beta_2 = \frac{2}{5}\begin{bmatrix} \frac{1}{6} & -\frac{1}{3} & \frac{5}{6} \end{bmatrix} = \begin{bmatrix} \frac{1}{15} & -\frac{2}{15} & \frac{1}{3} \end{bmatrix}$$

$$\mathbf{A_2^+} = \begin{bmatrix} \mathbf{A_1^+} - \delta_2\beta_2 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{6} & -\frac{1}{3} & \frac{5}{6} \end{bmatrix}$$

$$\text{Now, } \delta_3 = \mathbf{A_2^+}\alpha_3 = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{6} & -\frac{1}{3} & \frac{5}{6} \end{bmatrix}\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{2}{3} \\ \frac{7}{6} \end{bmatrix}$$

$$\gamma_3 = \alpha_3 - \mathbf{A_2}\delta_3 = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} -\frac{2}{3} \\ \frac{7}{6} \end{bmatrix} = \begin{bmatrix} \frac{1}{6} \\ -\frac{1}{3} \\ -\frac{1}{6} \end{bmatrix} \neq \mathbf{0}.$$

$$\text{Hence } \beta_3 = \gamma_3^+ = (\gamma_3^t\gamma_3)^{-1}\gamma_3^t = 6\begin{bmatrix} \frac{1}{6} & -\frac{1}{3} & -\frac{1}{6} \end{bmatrix} = \begin{bmatrix} 1 & -2 & -1 \end{bmatrix}$$

$$\delta_3\beta_3 = \begin{bmatrix} -\frac{2}{3} \\ \frac{7}{6} \end{bmatrix}\begin{bmatrix} 1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} -\frac{2}{3} & \frac{4}{3} & \frac{2}{3} \\ \frac{7}{6} & -\frac{7}{3} & -\frac{7}{6} \end{bmatrix}$$

$$\mathbf{A_3^+} = \begin{bmatrix} \mathbf{A_2^+} - \delta_3\beta_3 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 2 & 2 \\ 1 & -2 & -1 \end{bmatrix}$$

$$\text{Now, } \delta_4 = \mathbf{A_3^+}\alpha_4 = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 2 & 2 \\ 1 & -2 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \\ -1 \end{bmatrix},$$

$$\gamma_4 = \alpha_4 - \mathbf{A_3}\delta_4 = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} - \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix}\begin{bmatrix} -1 \\ 3 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

(the null column vector)

So that

$$\beta_4 = (1+\delta_4^t\delta_4)^{-1}\delta_4^t\mathbf{A}_3^+ = \tfrac{1}{12}\begin{bmatrix}-1 & 3 & -1\end{bmatrix}\begin{bmatrix}1 & -1 & -1\\ -1 & 2 & 2\\ 1 & -2 & -1\end{bmatrix} = \begin{bmatrix}-\tfrac{5}{12} & \tfrac{3}{4} & \tfrac{2}{3}\end{bmatrix}$$

$$\delta_4\beta_4 = \begin{bmatrix}-1\\ 3\\ -1\end{bmatrix}\begin{bmatrix}-\tfrac{5}{12} & \tfrac{3}{4} & \tfrac{2}{3}\end{bmatrix} = \begin{bmatrix}5/12 & -3/4 & -2/3\\ -5/4 & 9/4 & 2\\ 5/12 & -3/4 & -2/3\end{bmatrix}$$

$$\mathbf{A}_4^+ = \begin{bmatrix}\mathbf{A}_3^+ - \delta_4\beta_4\\ \beta_4\end{bmatrix} = \begin{bmatrix}7/12 & -1/4 & -1/3\\ 1/4 & -1/4 & 0\\ 7/12 & -5/4 & -1/3\\ -5/12 & 3/4 & 2/3\end{bmatrix} = \mathbf{A}^+$$

The given system is $\mathbf{Ax} = \mathbf{b}$ and its solution is $\mathbf{x} = \mathbf{A}^+\mathbf{b}$.

Therefore, $\mathbf{x} = \begin{bmatrix}7/12 & -1/4 & -1/3\\ 1/4 & -1/4 & 0\\ 7/12 & -5/4 & -1/3\\ -5/12 & 3/4 & 2/3\end{bmatrix}\begin{bmatrix}4\\ 0\\ 4\end{bmatrix} = \begin{bmatrix}1\\ 1\\ 1\\ 1\end{bmatrix}$

Therefore the required solution is
$x_1 = 1,\ x_2 = 1,\ x_3 = 1,\ x_4 = 1.$

Note 5.15.1 It may be verified that A^+ satisfies all the conditions (5.72). Again, $\mathbf{A}_3^+ = \mathbf{A}_3^{-1}$ as $|\mathbf{A}_3| = 1$ i.e., $\mathbf{A}_3$ is non-singular. In addition to this, for this $\mathbf{A}$, $\mathbf{AA}^+ = \mathbf{I}_4$, but $\mathbf{A}^+\mathbf{A} \neq \mathbf{I}_4$, the unit matrix of order 4.

The g-inverse $\mathbf{A}^+$ of $\mathbf{A}$ can also be computed using the formula (5.80).

5.16 Least Squares Solution for Inconsistent Systems

Let the system of linear equations be

$$\mathbf{Ax} = \mathbf{b} \tag{5.89}$$

where $\mathbf{A}, \mathbf{x}$ and $\mathbf{b}$ are of order $m\times n$, $n\times 1$ and $m\times 1$ respectively. Here, we assume that (5.89) is inconsistent. Since (5.89) is inconsistent, the system has no solution. Again, since there may be more than one $\mathbf{x_l}$ (least-squares solutions) for which $\|\mathbf{Ax} - \mathbf{b}\|$ is minimum, there exist one such $\mathbf{x_l}$ (say $\mathbf{x_m}$) whose norm is minimum. That is, $\mathbf{x_m}$ is called minimum norm least squares solution if

$$\|\mathbf{x_m}\| \le \|\mathbf{x_l}\| \tag{5.90}$$

for any $\mathbf{x_l}$ such that

$$\|\mathbf{Ax_l} - \mathbf{b}\| \le \|\mathbf{Ax} - \mathbf{b}\| \qquad \text{for all } \mathbf{x}. \tag{5.91}$$

The minimum norm least squares solution can be determined using the relation

$$\mathbf{x} = \mathbf{A}^+\mathbf{b}. \tag{5.92}$$

Since $\mathbf{A}^+$ is unique, the minimum norm least squares solution is unique.

The least squares solution can also be determined by the following way.

The vector $\mathbf{Ax} - \mathbf{b}$ in terms of the elements of $\mathbf{A}$, $\mathbf{x}$ and $\mathbf{b}$ is

$$\begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_n - b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots a_{2n}x_n - b_2 \\ \cdots\cdots \quad \cdots\cdots \quad \cdots \quad \cdots\cdots\cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots a_{mn}x_n - b_m \end{bmatrix}.$$

Let square of $\|\mathbf{Ax} - \mathbf{b}\|$ be denoted by S. Then

$$\begin{aligned} S &= (a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_n - b_1)^2 \\ &\quad +(a_{21}x_1 + a_{22}x_2 + \cdots a_{2n}x_n - b_2)^2 + \cdots \\ &\quad +(a_{m1}x_1 + a_{m2}x_2 + \cdots a_{mn}x_n - b_n)^2 \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij}x_j - b_i)^2. \end{aligned} \tag{5.93}$$

Also, S is called the sum of square of residues. To solve (5.89), find $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ from (5.93) in such a way that S is minimum. The conditions for S to be minimum are

$$\frac{\partial S}{\partial x_1} = 0, \ \frac{\partial S}{\partial x_2} = 0, \ \cdots, \ \frac{\partial S}{\partial x_n} = 0 \tag{5.94}$$

The system (5.94) is non-homogeneous and consists of n equations with n unknowns $x_1, x_2, \ldots, x_n$. This system can be solved by any method. Let $x_1 = x_1^*, x_2 = x_2^*, \ldots, x_n = x_n^*$ be a solution of (5.94). Therefore, the least-squares solution of (5.89) is

$$x^* = (x_1^*, x_2^*, \ldots, x_n^*)^t \tag{5.95}$$

and the sum of square of residues is given by

$$S^* = \sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij}x_j^* - b_i)^2. \tag{5.96}$$

This method is not suitable for a large system of equations, while the method stated in equation (5.92) is applicable for a large system also.

Example 5.16.1 Find g-inverse of the singular matrix $\mathbf{A} = \begin{bmatrix} 3 & 6 \\ 2 & 4 \end{bmatrix}$ and hence find a least squares solution of the inconsistent system

$$\begin{aligned} 3x + 6y &= 9 \\ 2x + 4y &= 5. \end{aligned}$$

Solution. Let $\alpha_{\mathbf{1}} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$, $\alpha_{\mathbf{2}} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$, $\mathbf{A_1} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$.

$\mathbf{A_1^+} = (\alpha_{\mathbf{1}}^{\mathbf{t}}\alpha_{\mathbf{1}})^{-\mathbf{1}}\alpha_{\mathbf{1}}^{\mathbf{t}} = \frac{1}{13}\begin{bmatrix} 3 & 2 \end{bmatrix} = \begin{bmatrix} \frac{3}{13} & \frac{2}{13} \end{bmatrix}$,

$\delta_{\mathbf{2}} = \mathbf{A_1^+}\alpha_{\mathbf{2}} = \begin{bmatrix} \frac{3}{13} & \frac{2}{13} \end{bmatrix}\begin{bmatrix} 6 \\ 4 \end{bmatrix} = 2$,

$\gamma_{\mathbf{2}} = \alpha_{\mathbf{2}} - \mathbf{A_1}\delta_{\mathbf{2}} = \begin{bmatrix} 6 \\ 4 \end{bmatrix} - \begin{bmatrix} 3 \\ 2 \end{bmatrix}.2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \mathbf{0}$ (a null vector),

$\beta_{\mathbf{2}} = (1 + \delta_{\mathbf{2}}^{\mathbf{t}}\delta_{\mathbf{2}})^{-\mathbf{1}}\delta_{\mathbf{2}}^{\mathbf{t}}\mathbf{A_1^+} = \frac{1}{5}.2.\begin{bmatrix} \frac{3}{13} & \frac{2}{13} \end{bmatrix} = \begin{bmatrix} \frac{6}{65} & \frac{4}{65} \end{bmatrix}$

$\delta_{\mathbf{2}}\beta_{\mathbf{2}} = \begin{bmatrix} \frac{12}{65} & \frac{8}{65} \end{bmatrix}$

Therefore, $\mathbf{A_2^+} = \begin{bmatrix} \mathbf{A_1^+} - \delta_{\mathbf{2}}\beta_{\mathbf{2}} \\ \beta_{\mathbf{2}} \end{bmatrix} = \begin{bmatrix} \frac{3}{65} & \frac{2}{65} \\ \frac{6}{65} & \frac{4}{65} \end{bmatrix} = \mathbf{A}^+$, which is the g-inverse of $\mathbf{A}$.

Second Part: The given equations can be written as

$\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} = \begin{bmatrix} 3 & 6 \\ 2 & 4 \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$, $\mathbf{A} = \begin{bmatrix} 9 \\ 5 \end{bmatrix}$.

Then the least squares solution is given by

$\mathbf{x} = \mathbf{A}^+\mathbf{b}$, i.e., $\mathbf{x} = \frac{1}{65}\begin{bmatrix} 3 & 2 \\ 6 & 4 \end{bmatrix}\begin{bmatrix} 9 \\ 5 \end{bmatrix} = \frac{37}{65}\begin{bmatrix} 1 \\ 2 \end{bmatrix}$.

Hence the least squares solution is $x = \dfrac{37}{65}$, $y = \dfrac{74}{65}$.

Example 5.16.2 Find the least squares solution of the following equations $x + y = 3.0, 2x - y = 0.03, x + 3y = 7.03$, and $3x + y = 4.97$. Also, estimate the residue.

Solution. Let x, y be least squares solution of the given system. Then the square of residues S is

$S = (x + y - 3.0)^2 + (2x - y - 0.03)^2 + (x + 3y - 7.03)^2 + (3x + y - 4.97)^2$.

We choose x and y in such a way that S is minimum. Therefore

$$\frac{\partial S}{\partial x} = 0 \text{ and } \frac{\partial S}{\partial y} = 0.$$

That is,

$$\begin{aligned} 2(x + y - 3.0) + 4(2x - y - 0.03) + 2(x + 3y - 7.03) + 6(3x + y - 4.97) &= 0 \\ \text{and } 2(x + y - 3.0) - 2(2x - y - 0.03) + 6(x + 3y - 7.03) + 2(3x + y - 4.97) &= 0. \end{aligned}$$

These equations reduce to $3x + y - 5 = 0$ and $5x + 12y - 29.03 = 0$.
Solution of these equations is $x = 30.97/31 = 0.9990322$ and $y = 62.09/31 = 2.0029032$, which is the required solution of the given equations.
The sum of the square of residues is $S = (3.0019354 - 3.0)^2 + (-0.0048388 - 0.03)^2 + (7.0077418 - 7.03)^2 + (4.9999998 - 4.97)^2 = 0.0026129$.

Iteration Methods

If the system of equations has a large number of variables, then the direct methods are not much suitable. In this case, the approximate numerical methods are used to determine the variables of the system.

The approximate methods for solving system of linear equations make it possible to obtain the values of the roots of the system with the specified accuracy as the limit of the sequence of some vectors. The process of constructing such a sequence is known as the **iterative process**.

The efficiency of the application of approximate methods depends on the choice of the initial vector and the rate of convergence of the process.

The following two approximate methods are widely used to solve a system of linear equations:
(i) method of iteration (Jacobi's iteration method), and
(ii) Gauss-Seidal's iteration method.

Before presenting the iteration methods, some terms are introduced to analyse the methods.

Let $x_i^{(k)}$, $i = 1, 2, \ldots, n$ be the kth ($k = 1, 2, \ldots$) iterated value of the variable x_i and $\mathbf{x^{(k)}} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})^t$ be the solution vector obtained at the kth iteration.

The sequence $\{\mathbf{x^{(k)}}\}$, $k = 1, 2, \ldots$ is said to converge to a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^t$ if for each i $(= 1, 2, \ldots, n)$

$$x_i^{(k)} \longrightarrow x_i \quad \text{as} \quad k \longrightarrow \infty. \tag{5.97}$$

Let $\xi = (\xi_1, \xi_2, \ldots, \xi_n)^t$ be the exact solution of the system of linear equations. Then the error $\varepsilon_i^{(k)}$ in the ith variable x_i committed in the kth iteration is given by

$$\varepsilon_i^{(k)} = \xi_i - x_i^{(k)}. \tag{5.98}$$

The error vector $\varepsilon^{(\mathbf{k})}$ at the kth iteration is then given by

$$\varepsilon^{(\mathbf{k})} = (\varepsilon_1^{(k)}, \varepsilon_2^{(k)}, \ldots, \varepsilon_n^{(k)})^t. \tag{5.99}$$

The error difference $\mathbf{e^{(k)}}$ at two consecutive iterations is given by

$$\mathbf{e^{(k)}} = \mathbf{x^{(k+1)}} - \mathbf{x^{(k)}} = \varepsilon^{(\mathbf{k})} - \varepsilon^{(\mathbf{k+1})}, \tag{5.100}$$

where $e_i^{(k)} = x_i^{(k+1)} - x_i^{(k)}$.

An iteration method is said to be of order $p \geq 1$ if there exists a positive constant A such that for all k

$$\|\varepsilon^{(\mathbf{k+1})}\| \leq A\|\varepsilon^{(\mathbf{k})}\|^p. \tag{5.101}$$

5.17 Jacobi's Iteration Method

Let us consider a system of n linear equations containing n variables:

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\; \cdots &\;\cdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n.
\end{aligned} \tag{5.102}$$

Also, we assume that the quantities a_{ii} are pivot elements.

The above equations can be written as

$$\begin{aligned}
x_1 &= \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n) \\
x_2 &= \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n) \\
\cdots\; &\cdots\;\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_n &= \frac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n\,n-1}x_{n-1}).
\end{aligned} \tag{5.103}$$

Let $x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}$ be the initial guess to the variables $x_1, x_2, \ldots, x_n$ respectively (initial guess may be taken as zeros). Substituting these values in the right hand side of (5.103), which yields the first approximation as follows.

$$\begin{aligned}
x_1^{(1)} &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)} - \cdots - a_{1n}x_n^{(0)}) \\
x_2^{(1)} &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(0)} - a_{23}x_3^{(0)} - \cdots - a_{2n}x_n^{(0)}) \\
\cdots\; &\cdots\;\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_n^{(1)} &= \frac{1}{a_{nn}}(b_n - a_{n1}x_1^{(0)} - a_{n2}x_2^{(0)} - \cdots - a_{n\,n-1}x_{n-1}^{(0)}).
\end{aligned} \tag{5.104}$$

Again, substituting $x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}$ in the right hand side of (5.103) and obtain the second approximation $x_1^{(2)}, x_2^{(2)}, \ldots, x_n^{(2)}$.

In general, if $x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}$ be the kth approximate roots then the next approximate roots are given by

$$\begin{aligned}
x_1^{(k+1)} &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)}) \\
x_2^{(k+1)} &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_n^{(k)}) \\
\cdots \; \cdots & \; \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_n^{(k+1)} &= \frac{1}{a_{nn}}(b_n - a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \cdots - a_{n\,n-1}x_{n-1}^{(k)}). \\
& \quad k = 0, 1, 2, \ldots.
\end{aligned} \tag{5.105}$$

The iteration process is continued until all the roots converge to the required number of significant figures. This iteration method is called **Jacobi's iteration** or simply the **method of iteration**.

The Jacobi's iteration method surely converges if the coefficient matrix is diagonally dominant.

5.17.1 Convergence of Gauss-Jacobi's iteration

The Gauss-Jacobi's iteration scheme (5.105) can also be written as

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\Bigg(b_i - \sum_{\substack{j=1\\j\neq i}}^{n} a_{ij}x_j^{(k)}\Bigg), i = 1, 2, \ldots, n. \tag{5.106}$$

If ξ be the exact solution of the system, then

$$\xi_i = \frac{1}{a_{ii}}\Bigg(b_i - \sum_{\substack{j=1\\j\neq i}}^{n} a_{ij}\xi_j\Bigg). \tag{5.107}$$

Therefore, equations (5.106) and (5.107) yield

$$\begin{aligned}
\xi_i - x_i^{(k+1)} &= -\frac{1}{a_{ii}}\sum_{\substack{j=1\\j\neq i}}^{n} a_{ij}\Big(\xi_j - x_j^{(k)}\Big) \\
\text{or} \qquad \varepsilon_i^{(k+1)} &= -\frac{1}{a_{ii}}\sum_{\substack{j=1\\j\neq i}}^{n} a_{ij}\varepsilon_j^{(k)}.
\end{aligned}$$

That is,

$$\|\varepsilon_i^{(k+1)}\| \;\leq\; \frac{1}{|a_{ii}|}\sum_{\substack{j=1\\j\neq i}}^{n} |a_{ij}| \; \|\varepsilon_j^{(k)}\| \leq \frac{1}{|a_{ii}|}\sum_{\substack{j=1\\j\neq i}}^{n} |a_{ij}| \; \|\varepsilon^{(\mathbf{k})}\|.$$

Let $\quad A = \max_i \left\{ \frac{1}{|a_{ii}|} \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \right\}.$

Then the above relation becomes

$$\|\varepsilon^{(\mathbf{k+1})}\| \leq A \|\varepsilon^{(\mathbf{k})}\|. \tag{5.108}$$

This relation shows that the rate of convergence of Gauss-Jacobi's method is linear.

Again, $\quad \|\varepsilon^{(\mathbf{k+1})}\| \leq A\|\varepsilon^{(\mathbf{k})}\| \leq A^2 \|\varepsilon^{(\mathbf{k-1})}\| \leq \cdots \leq A^{k+1} \|\varepsilon^{(\mathbf{0})}\|.$

That is,

$$\|\varepsilon^{(\mathbf{k})}\| \leq A^k \|\varepsilon^{(\mathbf{0})}\|. \tag{5.109}$$

If $A < 1$ then $A^k \to 0$ as $k \to \infty$ and consequently $\|\varepsilon^{(\mathbf{k})}\| \to 0$ as $k \to \infty$, i.e., the iteration converges.

Hence the sufficient condition for convergent of Gauss-Jacobi's method is $A < 1$ i.e., $\sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| < |a_{ii}|$ for all i, i.e., the coefficient matrix is diagonally dominant.

The relation (5.108) can be written as

$$\begin{aligned}
\|\varepsilon^{(\mathbf{k+1})}\| &\leq A\|\varepsilon^{(\mathbf{k})}\| = A\, \|\mathbf{e}^{(\mathbf{k})} + \varepsilon^{(\mathbf{k+1})}\| \qquad \text{[by (5.100)]} \\
&\leq A\|\mathbf{e}^{(\mathbf{k})}\| + A\|\varepsilon^{(\mathbf{k+1})}\| \\
\text{or,} \qquad \|\varepsilon^{(\mathbf{k+1})}\| &\leq \frac{A}{A-1}\|\mathbf{e}^{(\mathbf{k})}\|.
\end{aligned} \tag{5.110}$$

This relation gives the absolute error at the $(k+1)$th iteration in terms of the error difference at kth and $(k+1)$th iterations.

Example 5.17.1 Solve the following system of linear equations by Gauss-Jacobi's method correct up to four decimal places and calculate the upper bound of absolute errors.

$$\begin{aligned}
27x + 6y - z &= 54 \\
6x + 15y + 2z &= 72 \\
x + y + 54z &= 110.
\end{aligned}$$

Solution. Obviously, the system is diagonally dominant as

$$|6| + |-1| < |27|, \qquad |6| + |2| < |15|, \qquad |1| + |1| < |54|.$$

The Gauss-Jacobi's iteration scheme is

$$\begin{aligned} x^{(k+1)} &= \frac{1}{27}\Big(54 - 6y^{(k)} + z^{(k)}\Big) \\ y^{(k+1)} &= \frac{1}{15}\Big(27 - 6x^{(k)} - 2z^{(k)}\Big) \\ z^{(k+1)} &= \frac{1}{54}\Big(110 - x^{(k)} - y^{(k)}\Big). \end{aligned}$$

Let the initial solution be (0, 0, 0). The next iterations are shown in the following table.

k	x	y	z
0	0	0	0
1	2.00000	4.80000	2.03704
2	1.00878	3.72839	1.91111
3	1.24225	4.14167	1.94931
4	1.15183	4.04319	1.93733
5	1.17327	4.08096	1.94083
6	1.16500	4.07191	1.93974
7	1.16697	4.07537	1.94006
8	1.16614	4.07454	1.93996
9	1.16640	4.07488	1.93999
10	1.16632	4.07477	1.93998
11	1.16635	4.07481	1.93998

The solution correct up to four decimal places is

$$x = 1.1664, \quad y = 4.0748, \quad z = 1.9400.$$

Here

$$A = \max_i \left\{ \frac{1}{a_{ii}} \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \right\} = \max\left\{ \frac{7}{27}, \frac{8}{15}, \frac{2}{54} \right\} = \frac{8}{15}.$$

$\mathbf{e}^{(\mathbf{0})} = (3 \times 10^{-5}, 4 \times 10^{-5}, 0)$. Therefore, the upper bound of absolute error is

$$\|\varepsilon^{(\mathbf{0})}\| \le \frac{A}{1-A}\|\mathbf{e}^{(\mathbf{0})}\| = 5.71 \times 10^{-5}.$$

Algorithm 5.8 (Gauss-Jacobi's). This algorithm finds the solution of a system of linear equations by Gauss-Jacobi's iteration method. The method will terminate when $|x_i^{(k+1)} - x_i^{(k)}| < \varepsilon$, where ε is the supplied error tolerance, for all i.

```
Algorithm Gauss_Jacobi
 Step 1. Read the coefficients $a_{ij}, i, j = 1, 2, \ldots, n$ and the right hand vector
         $b_i, i = 1, 2, \ldots, n$ of the system of equations and error tolerance $\varepsilon$.
 Step 2. Rearrange the given equations, if possible, such that the system becomes
         diagonally dominant.
 Step 3. Rewrite the $i$th equation as
         $x_i = \frac{1}{a_{ii}}\Bigl(b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j\Bigr)$,   for $i = 1, 2, \ldots, n$.
 Step 4. Set the initial solution as
         $x_i = 0,\ \ i = 1, 2, 3, \ldots, n$.
 Step 5. Calculate the new values $xn_i$ of $x_i$ as
         $xn_i = \frac{1}{a_{ii}}\Bigl(b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j\Bigr)$,   for $i = 1, 2, \ldots, n$.
 Step 6. If $|x_i - xn_i| < \varepsilon$ ($\varepsilon$ is an error tolerance) for all $i$, then goto Step 7 else
         set $x_i = xn_i$ for all $i$ and goto Step 5.
 Step 7. Print $xn_i,\ \ i = 1, 2, \ldots, n$ as solution.
end Gauss_Jacobi
```

Program 5.8

```
/*Program Gauss_Jacobi
 Solution of a system of linear equations by Gauss-Jacobi's iteration
 method. Testing of diagonal dominance is also incorporated.*/
#include<stdio.h>
#include<math.h>
#include<stdlib.h>

void main()
{
 float a[10][10],b[10],x[10],xn[10],epp=0.00001,sum;
 int i,j,n,flag;
 printf("Enter number of variables ");
 scanf("%d",&n);
 printf("\nEnter the coefficients rowwise ");
 for(i=1;i<=n;i++)
     for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
 printf("\nEnter right hand vector ");
 for(i=1;i<=n;i++)
     scanf("%f",&b[i]);
 for(i=1;i<=n;i++) x[i]=0; /* initialize */
 /* checking for row dominance */
 flag=0;
 for(i=1;i<=n;i++)
     {
        sum=0;
        for(j=1;j<=n;j++)
            if(i!=j) sum+=fabs(a[i][j]);
        if(sum>fabs(a[i][i])) flag=1;
     }
 /* checking for column dominance */
 if(flag==1)
    {
     flag=0;
     for(j=1;j<=n;j++)
         {
           sum=0;
           for(i=1;i<=n;i++)
               if(i!=j) sum+=fabs(a[i][j]);
           if(sum>fabs(a[j][j])) flag=1;
         }
    }

if(flag==1)
  {
   printf("The coefficient matrix is not diagonally dominant\n");
   printf("The Gauss-Jacobi method does not converge surely");
   exit(0);
  }
 for(i=1;i<=n;i++) printf("  x[%d]  ",i);printf("\n");
 do
 {
 for(i=1;i<=n;i++)
   {
       sum=b[i];
       for(j=1;j<=n;j++)
           if(j!=i) sum-=a[i][j]*x[j];
       xn[i]=sum/a[i][i];
    }
 for(i=1;i<=n;i++) printf("%8.5f ",xn[i]);printf("\n");
 flag=0; /* indicates |x[i]-xn[i]|<epp for all i */
 for(i=1;i<=n;i++) if(fabs(x[i]-xn[i])>epp) flag=1;
 if(flag==1) for(i=1;i<=n;i++) x[i]=xn[i]; /* reset x[i] */
 }while(flag==1);

 printf("Solution is \n");
 for(i=1;i<=n;i++) printf("%8.5f ",xn[i]);
} /* main */
```

A sample of input/output:

```
Enter number of variables 3
Enter the coefficients rowwise
9 2 4
1 10 4
2 -4 10
Enter right hand vector
20 6 -15
    x[1]   x[2]    x[3]
 2.22222  0.60000 -1.50000
 2.75556  0.97778 -1.70444
 2.76247  1.00622 -1.66000
 2.73640  0.98775 -1.65000
 2.73606  0.98636 -1.65218
 2.73733  0.98727 -1.65267
 2.73735  0.98733 -1.65256
 2.73729  0.98729 -1.65254
 2.73729  0.98729 -1.65254
Solution is
 2.73729  0.98729 -1.65254
```

5.18 Gauss-Seidal's Iteration Method

A simple modification of Jacobi's iteration sometimes give faster convergence. The modified method is known as Gauss-Seidal's iteration method.

Let us consider a system of n linear equations with n variables.

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \qquad (5.111)\\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \cdots \ \cdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n.
\end{aligned}$$

Assume that the diagonal coefficients a_{ii}, $i = 1, 2, \ldots, n$ are diagonally dominant. If this is not the case then the above system of equations are re-arranged in such a way that the above condition holds.

The equations (5.111) are rewritten in the following form:

$$\begin{aligned}
x_1 &= \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n) \\
x_2 &= \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n) \\
\cdots\ \cdots &\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_n &= \frac{1}{a_{nn}}(b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n\,n-1}x_{n-1}).
\end{aligned} \tag{5.112}$$

To solve these equations an initial approximation $x_2^{(0)}, x_3^{(0)}, \ldots, x_n^{(0)}$ for the variables $x_2, x_3, \ldots, x_n$ respectively is considered. Substituting these values to the above system and get the first approximate value of x_1, denoted by $x_1^{(1)}$. Now, substituting $x_1^{(1)}$ for x_1 and $x_3^{(0)}, x_4^{(0)}, \ldots, x_n^{(0)}$ for $x_3, x_4, \ldots, x_n$ respectively and we find $x_2^{(1)}$ from second equation of (5.112), the first approximate value of x_2. Then substituting $x_1^{(1)}, x_2^{(1)}, \ldots, x_{i-1}^{(1)}, x_{i+1}^{(0)}, \ldots, x_n^{(0)}$ for $x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ to the ith equation of (5.112) respectively and obtain $x_i^{(1)}$, and so on.

If $x_i^{(k)}$, $i = 1, 2, \ldots, n$ be the kth approximate value of x_i, then the $(k+1)$th approximate value of $x_1, x_2, \ldots, x_n$ are given by

$$\begin{aligned}
x_1^{(k+1)} &= \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)}) \\
x_2^{(k+1)} &= \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_n^{(k)}) \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_i^{(k+1)} &= \frac{1}{a_{ii}}(b_i - a_{i1}x_1^{(k+1)} - \cdots - a_{i\,i-1}x_{i-1}^{(k+1)} - a_{i\,i+1}x_{i+1}^{(k)} - \cdots - a_{n\,n-1}x_{n-1}^{(k)}) \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_n^{(k+1)} &= \frac{1}{a_{nn}}(b_n - a_{n1}x_1^{(k+1)} - a_{n2}x_2^{(k+1)} - \cdots - a_{n\,n-1}x_{n-1}^{(k+1)}). \\
& k = 0, 1, 2, \ldots.
\end{aligned} \tag{5.113}$$

That is,

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\Big(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\Big), \quad i = 1, 2, \ldots, n \text{ and } k = 0, 1, 2, \ldots.$$

The method is repeated until $|x_i^{(k+1)} - x_i^{(k)}| < \varepsilon$ for all $i = 1, 2, \ldots, n$, where $\varepsilon > 0$ is any pre-assigned number called the error tolerance. This method is called Gauss-Seidal's iteration method.

Example 5.18.1 Solve the following system of equations by Gauss-Seidal's iteration method, correct up to four decimal places.

$$\begin{aligned} 27x + 6y - z &= 54 \\ 6x + 15y + 2z &= 72 \\ x + y + 54z &= 110 \end{aligned}$$

Solution. The iteration scheme is

$$\begin{aligned} x^{(k+1)} &= \frac{1}{27}(54 - 6y^{(k)} + z^{(k)}) \\ y^{(k+1)} &= \frac{1}{15}(72 - 6x^{(k+1)} - 2z^{(k)}) \\ z^{(k+1)} &= \frac{1}{54}(110 - x^{(k+1)} - y^{(k+1)}). \end{aligned}$$

Let $y = 0,\ z = 0$ be the initial solution. The successive iterations are shown below.

k	x	y	z
0	–	0	0
1	2.00000	4.00000	1.92593
2	1.18244	4.07023	1.93977
3	1.16735	4.07442	1.93997
4	1.16642	4.07477	1.93998
5	1.16635	4.07480	1.93998
6	1.16634	4.07480	1.93998

The solution correct up to four decimal places is $x = 1.1663,\ y = 4.0748,\ z = 1.9400$.

Note 5.18.1 This solution is achieved in eleven iterations using Gauss-Jacobi's method while only six iterations are used in Gauss-Seidal's method.

The sufficient condition for convergence of this method is that the diagonal elements of the coefficient matrix are diagonally dominant. This is justified in the following.

5.18.1 Convergence of Gauss-Seidal's method

Let $A = \max_i \left\{ \frac{1}{a_{ii}} \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \right\}$ and let $A_i = \frac{1}{a_{ii}} \sum_{j=1}^{i-1} |a_{ij}|,\quad i = 1, 2, \ldots, n.$

It can easily be verified that $0 \le A_i \le A < 1$.

Then by Gauss-Jacobi's case

$$\begin{aligned}
|\varepsilon_i^{(k+1)}| &\le \frac{1}{|a_{ii}|}\Big[\sum_{j<i}|a_{ij}|\,|\varepsilon_j^{(k+1)}| + \sum_{j>i}|a_{ij}|\,|\varepsilon_j^{(k)}|\Big] \\
&\le \frac{1}{|a_{ii}|}\Big[\sum_{j<i}|a_{ij}|\,\|\varepsilon^{(\mathbf{k+1})}\| + \sum_{j>i}|a_{ij}|\,\|\varepsilon^{(\mathbf{k})}\|\Big] \\
&\le A_i\|\varepsilon^{(\mathbf{k+1})}\| + (A-A_i)\|\varepsilon^{(\mathbf{k})}\|.
\end{aligned}$$

Then for some i,

$$\|\varepsilon^{(\mathbf{k+1})}\| \le A_i\|\varepsilon^{(\mathbf{k+1})}\| + (A-A_i)\|\varepsilon^{(\mathbf{k})}\|$$

That is,

$$\|\varepsilon^{(\mathbf{k+1})}\| \le \frac{A-A_i}{1-A_i}\|\varepsilon^{(\mathbf{k})}\|.$$

Since $0 \le A_i \le A < 1$ then $\dfrac{A-A_i}{1-A_i} \le A$.

Therefore the above relation reduces to

$$\|\varepsilon^{(\mathbf{k+1})}\| \le A\|\varepsilon^{(\mathbf{k})}\|. \tag{5.114}$$

This shows that the rate of convergence of Gauss-Seidal's iteration is also linear. The successive substitutions give

$$\|\varepsilon^{(\mathbf{k})}\| \le A^k\|\varepsilon^{(\mathbf{0})}\|.$$

Now, if $A < 1$ then $\|\varepsilon^{(\mathbf{k})}\| \to 0$ as $k \to \infty$, i.e., the sequence $\{\mathbf{x}^{(\mathbf{k})}\}$ is sure to converge when $A < 1$ i.e.,

$$\sum_{\substack{j=1\\ j\ne i}}^{n} |a_{ij} < |a_{ii}| \text{ for all } i.$$

In other words the sufficient condition for Gauss-Seidal's iteration is that the coefficient matrix is diagonally dominant. The absolute error at the $(k+1)$th iteration is given by

$$\|\varepsilon^{(\mathbf{k+1})}\| \le \frac{A}{1-A}\|\mathbf{e}^{(\mathbf{k})}\| \quad \text{when } \ A < 1,$$

as in previous section.

Note 5.18.2 Usually, the Gauss-Seidal's method converges rapidly than the Gauss-Jacobi's method. But, this is not always true. There are some examples in which the Gauss-Jacobi's method converges faster than the Gauss-Seidal's method.

Example 5.18.2 Solve the following system of equations by Gauss-Seidal's method correct to four significant figures: $3x + y + z = 3, 2x + y + 5z = 5, x + 4y + z = 2$.

Solution. It may be noted that the given system is not diagonally dominant, but, the rearranged system $3x + y + z = 3, x + 4y + z = 2, 2x + y + 5z = 5$ is diagonally dominant.
Then the Gauss-Seidal's iteration scheme is

$$\begin{aligned} x^{(k+1)} &= \frac{1}{3}(3 - y^{(k)} - z^{(k)}) \\ y^{(k+1)} &= \frac{1}{4}(2 - x^{(k+1)} - z^{(k)}) \\ z^{(k+1)} &= \frac{1}{5}(5 - 2x^{(k+1)} - y^{(k+1)}). \end{aligned}$$

Let $y = 0,\ z = 0$ be the initial values. The successive iterations are shown below.

k	x	y	z
0	–	0	0
1	1.00000	0.25000	0.55000
2	0.73333	0.17917	0.67083
3	0.71667	0.15313	0.68271
4	0.72139	0.14898	0.68165
5	0.72312	0.14881	0.68099
6	0.72340	0.14890	0.68086
7	0.72341	0.14893	0.68085

Therefore, the solution correct up to four significant figures is
$x = 0.7234,\ y = 0.1489,\ z = 0.6808$.

Example 5.18.3 Solve the following system of equations using Gauss-Seidal's method: $3x + y + 2z = 6, -x + 4y + 2z = 5, 2x + y + 4z = 7$.

Solution. The iteration scheme is

$$\begin{aligned} x^{(k+1)} &= \frac{1}{3}(6 - y^{(k)} - 2z^{(k)}) \\ y^{(k+1)} &= \frac{1}{4}(5 + x^{(k+1)} - 2z^{(k)}) \\ z^{(k+1)} &= \frac{1}{4}(7 - 2x^{(k+1)} - y^{(k+1)}). \end{aligned}$$

Let $y = 0, z = 0$ be the initial solution and other approximate values are shown below.

k	x	y	z
0	–	0	0
1	2.00000	1.75000	0.31250
2	1.20833	1.39583	0.79688
3	1.00347	1.10243	1.10243
4	0.89757	0.92328	1.07042
5	0.97866	0.95946	1.02081
6	0.99964	0.98951	1.00280
7	1.00163	0.99901	0.99943
8	1.00071	1.00046	0.99953
9	1.00016	1.00028	0.99985
10	1.00001	1.00008	0.99998

Therefore, the solution correct up to four decimal places is

$$x = 1.0000,\ y = 1.0000,\ z = 1.0000.$$

It may be noted that the given system is not diagonally dominant while the iteration scheme converges to the exact solution.

Another interesting problem is considered in the following. The system of equations

$$x_1 + x_2 = 2, x_1 - 3x_2 = 1$$

converges when the iteration scheme is taken as

$$x_1^{(k+1)} = 2 - x_2^{(k)}, \qquad x_2^{(k+1)} = \frac{1}{3}(-1 + x_1^{(k+1)}) \tag{5.115}$$

While the Gauss-Seidal's iteration method diverges when the iteration scheme is

$$x_1^{(k+1)} = 1 + 3x_2^{(k)}, \qquad x_2^{(k+1)} = 2 - x_1^{(k+1)} \tag{5.116}$$

(It may be noted that the system is not diagonally dominant).

The calculations for these two schemes are shown below and the behaviour of the solutions are shown in the figures 5.1 and 5.2.

For the scheme (5.115)

k	x_1	x_2
0	-	0
1	2	0.333
2	1.667	0.222
3	1.778	0.259
4	1.741	0.247
5	1.753	0.251
6	1.749	0.250
7	1.750	0.250

For the scheme (5.116)

k	x_1	x_2
0	-	0
1	1	1
2	4	-2
3	-5	7
4	22	-20
5	-59	61
6	184	-182
7	547	-545

The exact solution of the equations is

$$x_1 = 1.75,\ x_2 = 0.25.$$

This example shows that the condition 'diagonally dominant' is a sufficient condition, not necessary for Gauss-Seidal's iteration method.

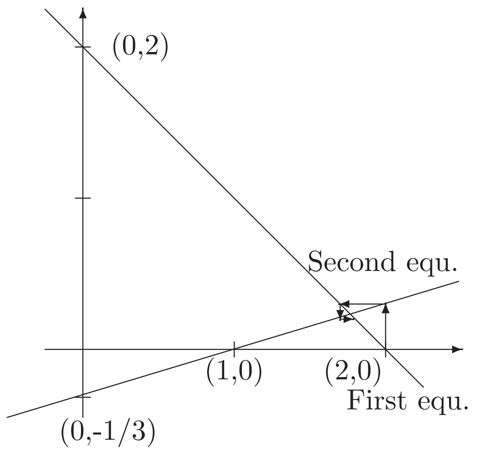


Figure 5.1: Illustration of Gauss-Seidal's method for the convergent scheme (5.115.)

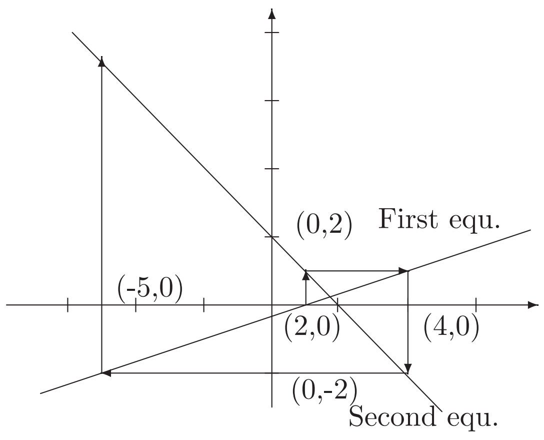


Figure 5.2: Illustration of Gauss-Seidal's method for the divergent scheme (5.116).

Algorithm 5.9 (Gauss-Seidal's). This algorithm finds the solution of a system of linear equations by Gauss-Seidal's iteration method. The method will terminate when $|x_i^{(k+1)} - x_i^{(k)}| < \varepsilon$, where ε is the supplied error tolerance, for all i.

Algorithm Gauss_Seidal

```
Step 1. Read the coefficients $a_{ij}, i, j = 1, 2, \ldots, n$ and the right hand vector
        $b_i, i = 1, 2, \ldots, n$ of the system of equations and error tolerance $\varepsilon$.
Step 2. Rearrange the given equations, if possible, such that the system becomes
        diagonally dominant.
Step 3. Rewrite the $i$th equation as
        $x_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j<i} a_{ij}x_j - \sum_{j>i} a_{ij}x_j\Big)$,   for $i = 1, 2, \ldots, n$.
Step 4. Set the initial solution as
        $x_i = 0,\ \ i = 1, 2, 3, \ldots, n$.
Step 5. Calculate the new values $xn_i$ of $x_i$ as
        $xn_i = \frac{1}{a_{ii}}\Big(b_i - \sum_{j<i} a_{ij}xn_j - \sum_{j>i} a_{ij}x_j\Big)$,   for $i = 1, 2, \ldots, n$.
Step 6. If $|x_i - xn_i| < \varepsilon$ ($\varepsilon$ is an error tolerance) for all $i$ then goto Step 7 else
        set $x_i = xn_i$ for all $i$ and goto Step 5.
Step 7. Print $xn_i,\ \ i = 1, 2, \ldots, n$ as solution.
end Gauss_Seidal
```

Program 5.9

```
/* Program Gauss-Seidal
   Solution of a system of linear equations by Gauss-Seidal's
   iteration method. Assume that the coefficient matrix satisfies
   the condition of convergence. */
#include<stdio.h>
#include<math.h>
void main()
{
 float a[10][10],b[10],x[10],xn[10],epp=0.00001,sum;
 int i,j,n,flag;
 printf("Enter number of variables ");
 scanf("%d",&n);
 printf("\nEnter the coefficients rowwise ");
 for(i=1;i<=n;i++)
    for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
 printf("\nEnter right hand vector ");
 for(i=1;i<=n;i++)
     scanf("%f",&b[i]);
 for(i=1;i<=n;i++) x[i]=0; /* initialize */
 /* testing of diagonal dominance may be included here
     from the program of Gauss-Jacobi's method */
 do
 {
 for(i=1;i<=n;i++)
   {
    sum=b[i];
    for(j=1;j<=n;j++)
      {
    if(j<i)
      sum-=a[i][j]*xn[j];
    else if(j>i)
      sum-=a[i][j]*x[j];
    xn[i]=sum/a[i][i];
      }
    }
 flag=0; /* indicates |x[i]-xn[i]|<epp for all i */
 for(i=1;i<=n;i++) if(fabs(x[i]-xn[i])>epp) flag=1;
 if(flag==1) for(i=1;i<=n;i++) x[i]=xn[i]; /* reset x[i] */
 }
 while(flag==1);
 printf("Solution is \n");
 for(i=1;i<=n;i++) printf("%8.5f ",xn[i]);
} /* main */
```

A sample of input/output:

```
Enter number of variables 3
Enter the coefficients rowwise
3 1 -1
2 5 2
2 4 6

Enter right hand vector
7 9 8
Solution is
 2.00000  1.00000  0.00000
```

5.19 The Relaxation Method

The relaxation method is also an iterative method invented by Southwell in 1946.

Let $\mathbf{x^{(k)}} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})^t$ be the solution obtained at the kth iteration of the system of equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \;\; i = 1, 2, \ldots, n. \tag{5.117}$$

Then $\sum_{j=1}^{n} a_{ij}x_j^{(k)} = b_i, \;\; i = 1, 2, \ldots, n.$

If $r_i^{(k)}$ denotes the residual of the ith equation, then

$$r_i^{(k)} = b_i - \sum_{j=1}^{n} a_{ij}x_j^{(k)}. \tag{5.118}$$

Now, the solution can be improved successively by reducing the largest residual to zero at that iteration.

To achieve the fast convergence of the method, the equations are rearranged in such a way that the largest coefficients in the equations appear on the diagonals. Now, the largest residual (in magnitude) is determined and let it be r_p. Then the value of the variable x_p be increased by dx_p where $dx_p = -\frac{r_p}{a_{pp}}$.

In other words, x_p is changed to $x_p + dx_p$ to relax r_p, i.e., to reduce r_p to zero. Then the new solution after this iteration becomes

$$\mathbf{x^{(k+1)}} = \left(x_1^{(k)}, x_2^{(k)}, \ldots, x_{p-1}^{(k)}, x_p + dx_p, x_{p+1}^{(k)}, \ldots, x_n^{(k)}\right).$$

The method is repeated until all the residuals become zero or very small.

Example 5.19.1 Solve the following system of equations
$8x_1 + x_2 - x_3 = 8, \quad 2x_1 + x_2 + 9x_3 = 12, \quad x_1 - 7x_2 + 2x_3 = -4$
by relaxation method taking $(0, 0, 0)$ as initial solution.

Solution. We rearrange the equations to get the largest coefficients in the diagonals as

$$\begin{aligned} 8x_1 + x_2 - x_3 &= 8 \\ x_1 - 7x_2 + 2x_3 &= -4 \\ 2x_1 + x_2 + 9x_3 &= 12. \end{aligned}$$

The residuals r_1, r_2, r_3 can be computed from the equations

$$\begin{aligned} r_1 &= 8 - 8x_1 - x_2 + x_3 \\ r_2 &= -4 - x_1 + 7x_2 - 2x_3 \\ r_3 &= 12 - 2x_1 - x_2 - 9x_3. \end{aligned}$$

The method is started with the initial solution $(0, 0, 0)$, i.e., $x_1 = x_2 = x_3 = 0$. Then the residuals $r_1 = 8, r_2 = -4, r_3 = 12$ of which the largest residual in magnitude is r_3. This indicates that the third equation has more error and needs to improve x_3. Then the increment $dx_3 = -\frac{r_3}{a_{33}} = \frac{12}{9} = 1.333$. The new solution then becomes $(0, 0, 0 + 1.333)$ i.e., $(0, 0, 1.333)$.

Similarly, we find the new residuals of large magnitudes and relax it to zero and we repeat the process until all the residuals become zero or very small.
The detail calculations are shown in the following table.

	residuals			max		increment	solution		
k	r_1	r_2	r_3	(r_1, r_2, r_3)	p	dx_p	x_1	x_2	x_3
0	–	–	–	–	–	–	0	0	0
1	8	–4	12	12	3	1.333	0	0	1.333
2	9.333	–6.666	0.003	9.333	1	1.167	1.167	0	1.333
3	–0.003	–7.833	–2.331	–7.833	2	1.119	1.167	1.119	1.333
4	–1.122	0	–3.450	–3.450	3	–0.383	1.167	1.119	0.950
5	–1.505	0.766	–0.003	–1.505	1	–0.188	0.979	1.119	0.950
6	–0.001	0.954	0.373	0.954	2	–0.136	0.979	0.983	0.950
7	0.135	0.002	0.509	0.509	3	0.057	0.979	0.983	1.007
8	0.192	–0.112	–0.004	0.192	1	0.024	1.003	0.983	1.007
9	0	–0.136	–0.052	–0.136	2	0.019	1.003	1.002	1.007
10	–0.019	–0.003	–0.071	–0.071	3	–0.008	1.003	1.002	0.999
11	–0.027	0.013	0.001	–0.027	1	0.003	1.000	1.002	0.999
12	–0.003	0.016	0.007	0.016	2	–0.002	1.000	1.000	0.999
13	–0.001	0.002	0.009	0.009	3	0.001	1.000	1.000	1.000
14	0	0	0	0	–	0	1.000	1.000	1.000

At this stage all the residuals are zero and therefore the solution of the given system of equations is $x_1 = 1.000,\ x_2 = 1.000,\ x_3 = 1.000$, which is the exact solution of the equations.

5.20 Successive Overrelaxation (S.O.R.) Method

The convergence can be accelerated by introducing a suitable relaxation factor w. In this method, the ith equation $\sum_{j=1}^{n} a_{ij}x_j = b_i$ can be written as

$$\sum_{j=1}^{i-1} a_{ij}x_j + \sum_{j=i}^{n} a_{ij}x_j = b_i. \tag{5.119}$$

Like Gauss-Seidal's method, for the solution

$$\left(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_i^{(k)}, x_{i+1}^{(k)}, \ldots, x_n^{(k)}\right),$$

the equation (5.119) becomes

$$\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i}^{n} a_{ij} x_j^{(k)} = b_i. \tag{5.120}$$

The residual at the ith equation is then computed as

$$r_i = b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} x_j^{(k)}. \tag{5.121}$$

Let $d_i^{(k)} = x_i^{(k+1)} - x_i^{(k)}$ denote the differences of x_i at two consecutive iterations.

In the successive overrelaxation (S.O.R. or SOR) method, assume that

$$a_{ii} d_i^{(k)} = w r_i, \quad i = 1, 2, \ldots, n, \tag{5.122}$$

where w is a suitable factor, called the **relaxation factor**.

Using (5.121), equation (5.122) becomes

$$a_{ii} x_i^{(k+1)} = a_{ii} x_i^{(k)} - w\left[\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} + \sum_{j=i}^{n} a_{ij} x_j^{(k)} - b_i\right], \tag{5.123}$$
$$i = 1, 2, \ldots, n; \quad k = 0, 1, 2, \ldots$$

and $\left(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}\right)^t$ is the initial solution. The method is repeated until desired accuracy is achieved.

This method is called the **overrelaxation method** when $1 < w < 2$, and is called the **under relaxation** method when $0 < w < 1$. When $w = 1$, the method becomes Gauss-Seidal's method.

Best relaxation factor w_b

The proper choice of w_b can speed up the convergence of the system. In a problem, Carré took $w_b = 1.9$ and found that the convergence is 40 times faster than when $w = 1$ (Gauss-Seidal's method). He also observed that when $w = 1.875$ (a minor variation of 1.9), the convergence is only two times faster than the Gauss-Seidal's method. In general, the choice of w_b is not a simple task. It depends on the spectral radius of the coefficient matrix.

Example 5.20.1 Solve the following system of equations

$$\begin{aligned} 3x_1 + x_2 + 2x_3 &= 6 \\ -x_1 + 4x_2 + 2x_3 &= 5 \\ 2x_1 + x_2 + 4x_3 &= 7 \end{aligned}$$

by SOR method taken $w = 1.01$

Solution. The iteration scheme for SOR method is

$$\begin{aligned} a_{11}x_1^{(k+1)} &= a_{11}x_1^{(k)} - w\Big[a_{11}x_1^{(k)} + a_{12}x_2^{(k)} + a_{13}x_3^{(k)} - b_1\Big] \\ a_{22}x_2^{(k+1)} &= a_{22}x_2^{(k)} - w\Big[a_{21}x_1^{(k+1)} + a_{22}x_2^{(k)} + a_{23}x_3^{(k)} - b_2\Big] \\ a_{33}x_3^{(k+1)} &= a_{33}x_3^{(k)} - w\Big[a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{33}x_3^{(k)} - b_3\Big] \end{aligned}$$

or

$$\begin{aligned} 3x_1^{(k+1)} &= 3x_1^{(k)} - 1.01\Big[3x_1^{(k)} + x_2^{(k)} + 2x_3^{(k)} - 6\Big] \\ 4x_2^{(k+1)} &= 4x_2^{(k)} - 1.01\Big[- x_1^{(k+1)} + 4x_2^{(k)} + 2x_3^{(k)} - 5\Big] \\ 4x_3^{(k+1)} &= 4x_3^{(k)} - 1.01\Big[2x_1^{(k+1)} + x_2^{(k+1)} + 4x_3^{(k)} - 7\Big]. \end{aligned}$$

Let $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$.
The detail calculations are shown in the following table.

k	x_1	x_2	x_3
0	0	0	0
1	2.02000	1.77255	0.29983
2	1.20116	1.39665	0.80526
3	0.99557	1.09326	0.98064
4	0.98169	1.00422	1.00838
5	0.99312	0.99399	1.00491
6	0.99879	0.99728	1.00125
7	1.00009	0.99942	1.00009
8	1.00013	0.99999	0.99993
9	1.00005	1.00005	0.99997

Therefore the required solution is

$$x_1 = 1.0000, \quad x_2 = 1.0000, \quad x_3 = 1.0000$$

correct up to four decimal places.

Program 5.10

```
/* Program Gauss-Seidal SOR
   Solution of a system of linear equations by Gauss-Seidal
   successive overrelaxation (SOR) method. The relaxation factor w
   lies between 1 and 2. Assume that the coefficient matrix
   satisfies the condition of convergence. */
#include<stdio.h>
#include<math.h>
void main()
{
 float a[10][10],b[10],x[10],xn[10],epp=0.00001,sum,w;
 int i,j,n,flag;
 printf("Enter number of variables ");
 scanf("%d",&n);
 printf("\nEnter the coefficients rowwise ");
 for(i=1;i<=n;i++)
     for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
 printf("\nEnter right hand vector ");
 for(i=1;i<=n;i++) scanf("%f",&b[i]);
 printf("Enter the relaxation factor w ");
 scanf("%f",&w);
 for(i=1;i<=n;i++) x[i]=0; /* initialize */
 for(i=1;i<=n;i++) printf("  x[%d]  ",i);printf("\n");
 do
  {
   for(i=1;i<=n;i++)
      {
       sum=b[i]*w+a[i][i]*x[i];
       for(j=1;j<=n;j++)
       {
        if(j<i)
           sum-=a[i][j]*xn[j]*w;
        else if(j>=i)
           sum-=a[i][j]*x[j]*w;
        xn[i]=sum/a[i][i];
       }
      }
 for(i=1;i<=n;i++)  printf("%8.5f ",xn[i]);
 printf("\n");
 flag=0; /* indicates |x[i]-xn[i]|<epp for all i */
 for(i=1;i<=n;i++) if(fabs(x[i]-xn[i])>epp) flag=1;
 if(flag==1) for(i=1;i<=n;i++) x[i]=xn[i]; /* reset x[i] */
 }while(flag==1);
 printf("Solution is \n");
 for(i=1;i<=n;i++) printf("%8.5f ",xn[i]);
} /* main */
```

A sample of input/output:

```
Enter number of variables 3
Enter the coefficients rowwise
9 2 4
1 10 4
2 -4 10

Enter right hand vector
20 6 -15
Enter the relaxation factor w 1.01
   x[1]    x[2]    x[3]
 2.24444  0.37931 -1.81514
 2.95166  1.03740 -1.67397
 2.73352  0.99583 -1.64812
 2.73342  0.98581 -1.65241
 2.73760  0.98722 -1.65264
 2.73734  0.98732 -1.65254
 2.73728  0.98729 -1.65254
 2.73729  0.98729 -1.65254
Solution is
 2.73729  0.98729 -1.65254
```

5.21 Comparison of Direct and Iterative Methods

The direct and iterative, both the methods have some advantages and also some disadvantages and a choice between them is based on the given system of equations.

(i) The direct method is applicable for all types of problems (when the coefficient determinant is not zero) where as iterative methods are useful only for particular types of problems.

(ii) The rounding errors may become large particularly for ill-conditioned systems while in iterative method the rounding error is small, since it is committed in the last iteration. Thus for ill-conditioned systems an iterative method is a good choice.

(iii) In each iteration, the computational effect is large in direct method (it is $2n^3/3$ for elimination method) and it is low in iteration method ($2n^2$ in Gauss-Jacobi's and Gauss-Seidal's methods).

(iv) Most of the direct methods are applied on the coefficient matrix and for this purpose, the entire matrix to be stored into primary memory of the computer. But, the iteration methods are applied in a single equation at a time, and hence only a single equation is to be stored at a time in primary memory. Thus iterative methods are efficient then direct method with respect to space.

5.22 Exercise

1. Find the values of the following determinants using partial pivoting.

$$\text{(i)} \quad \begin{vmatrix} 0 & 2 & 5 \\ 1 & 3 & -8 \\ 6 & 5 & 1 \end{vmatrix} \qquad \text{(ii)} \quad \begin{vmatrix} -2 & 3 & 8 & 4 \\ 6 & 1 & 0 & 5 \\ -8 & 3 & 1 & 2 \\ 3 & 8 & 7 & 10 \end{vmatrix}.$$

2. Solve the following systems of equations by Cramer's rule.

$$\text{(i)} \quad \begin{aligned} 3x_1 + 2x_2 + x_3 &= 5 \\ 2x_1 + 5x_2 + x_3 &= -3 \\ 2x_1 + x_2 + 3x_3 &= 11 \end{aligned} \qquad \text{(ii)} \quad \begin{aligned} 7.6x_1 + 0.5x_2 + 2.4x_3 &= 1.9 \\ 2.2x_1 + 9.1x_2 + 4.4x_3 &= 9.7 \\ -1.3x_1 + 0.2x_2 + 5.8x_3 &= -1.4 \end{aligned}$$

3. Find the inverse of the matrix

$$\begin{bmatrix} 11 & 3 & -1 \\ 2 & 5 & 5 \\ 1 & 1 & 1 \end{bmatrix}$$

using Gauss-Jordan method and solve the following system of equations.

$$\begin{aligned} 11x_1 + 3x_2 - x_3 &= 15 \\ 2x_1 + 5x_2 + 5x_3 &= -11 \\ x_1 + x_2 + x_3 &= 1. \end{aligned}$$

4. Find the inverses of the following matrices (using partial pivoting).

$$\text{(i)} \quad \begin{bmatrix} 0 & 1 & 2 \\ 3 & 5 & 1 \\ 6 & 8 & 9 \end{bmatrix} \qquad \text{(ii)} \quad \begin{bmatrix} -1 & 2 & 0 \\ 1 & 0 & 5 \\ 3 & 8 & 7 \end{bmatrix}.$$

5. Solve the following systems of equations by Gauss elimination method.

(i) $$\begin{aligned} x_1+\tfrac{1}{2}x_2+\tfrac{1}{3}x_3 &= 1\\ \tfrac{1}{2}x_1+\tfrac{1}{3}x_2+\tfrac{1}{4}x_3 &= 0\\ \tfrac{1}{3}x_1+\tfrac{1}{4}x_2+\tfrac{1}{5}x_3 &= 0 \end{aligned}$$

(ii) $$\begin{aligned} 4x+y+z &= 4\\ x+4y-2z &= 4\\ 3x+2y-4z &= 6 \end{aligned}$$

(iii) $$\begin{aligned} x_1-4x_2-x_4 &= 6\\ x_1+x_2+2x_3+3x_4 &= -1\\ 2x_1+3x_2-x_3-x_4 &= -1\\ x_1+2x_2+3x_3-x_4 &= 3 \end{aligned}$$

(iv) $$\begin{aligned} 1.14x_1-2.15x_2-5.11x_3 &= 2.05\\ 0.42x_1-1.13x_2+7.05x_3 &= 0.80\\ -0.71x_1+0.81x_2-0.02x_3 &= -1.07 \end{aligned}$$

6. Use Gauss elimination method to find the values of the determinants.

$$\text{(i)}\quad \begin{vmatrix} 1 & 4 & 1 & 3\\ 0 & -1 & 3 & -1\\ 3 & 1 & 0 & 2\\ 1 & -2 & 5 & 1 \end{vmatrix} \qquad \text{(ii)}\quad \begin{vmatrix} -1.6 & 5.4 & -7.7 & 3.1\\ 8.2 & 1.4 & -2.3 & 0.2\\ 5.3 & -5.9 & 2.7 & -8.9\\ 0.7 & 1.9 & -8.5 & 4.8 \end{vmatrix}.$$

7. Using LU decomposition method, solve the following systems of equations

(i) $$\begin{aligned} x_1+x_2+x_3 &= 3\\ 2x_1-x_2+3x_3 &= 16\\ 3x_1+x_2-x_3 &= -3 \end{aligned}$$

(ii) $$\begin{aligned} x-2y+7z &= 6\\ 4x+2y+z &= 7\\ 2x+5y-2z &= 5. \end{aligned}$$

8. Find the triangular factorization $\mathbf{A}=\mathbf{LU}$ for the matrices

$$\text{(i)}\quad \begin{bmatrix} 4 & 2 & 1\\ 2 & 5 & -2\\ 1 & -2 & 7 \end{bmatrix} \qquad \text{(ii)}\quad \begin{bmatrix} -5 & 2 & -1\\ 1 & 0 & 3\\ 3 & 1 & 6 \end{bmatrix}.$$

9. Solve $\mathbf{LY}=\mathbf{B}$, $\mathbf{UX}=\mathbf{Y}$ and verify that $\mathbf{B}=\mathbf{AX}$ for $\mathbf{B}=(8,-4,10,-4)^t$, where $\mathbf{A}=\mathbf{LU}$ is given by

$$\begin{bmatrix} 4 & 8 & 4 & 0\\ 1 & 5 & 4 & -3\\ 1 & 4 & 7 & 2\\ 1 & 3 & 0 & -2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0\\ 1/4 & 1 & 0 & 0\\ 1/4 & 2/3 & 1 & 0\\ 1/4 & 1/3 & -1/2 & 1 \end{bmatrix} \begin{bmatrix} 4 & 8 & 4 & 0\\ 0 & 3 & 3 & -3\\ 0 & 0 & 4 & 4\\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

10. Prove that the product of two upper-triangular matrices is an upper-triangular matrix.

11. Prove that the inverse of a non-singular upper-triangular matrix is an upper-triangular matrix.

12. Use Cholesky method to solve the following systems of linear equations.

(i) $$\begin{aligned} x+2y+3z &= 0\\ 2x+y+2z &= 1\\ 3x+2y+z &= 4 \end{aligned}$$

(ii) $$\begin{aligned} x_1+3x_2+4x_3 &= 8\\ 3x_1-x_2+5x_3 &= 7\\ 4x_1+5x_2-7x_3 &= 2. \end{aligned}$$

13. Find the inverses of the following matrices using partition.

$$\text{(i)}\quad \begin{bmatrix} 20 & 1 & -2 \\ 3 & 20 & -1 \\ 2 & -3 & 20 \end{bmatrix} \qquad \text{(ii)}\quad \begin{bmatrix} 8 & 1 & -1 \\ 2 & 1 & 9 \\ 1 & -7 & 2 \end{bmatrix}.$$

14. Find the solution of the following tri-diagonal system:

$$\begin{aligned} 2x_1 - 2x_2 &= 1 \\ -x_1 + 2x_2 - 3x_3 &= -2 \\ -2x_2 + 2x_3 - 4x_4 &= -1 \\ x_3 - x_4 &= 3. \end{aligned}$$

15. Test the following system for ill-condition.

$$\begin{aligned} 10x + 7y + 8z + 7w &= 32 \\ 7x + 5y + 6z + 5w &= 23 \\ 8x + 6y + 10z + 9w &= 33 \\ 7x + 5y + 9z + 10w &= 31. \end{aligned}$$

16. Find the g-inverses of the following matrices

$$\text{(i)}\quad \begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix} \qquad \text{(ii)}\quad \begin{bmatrix} 2 & 3 & 5 \\ 1 & -1 & 0 \\ 3 & 1 & 2 \end{bmatrix}.$$

17. Find the g-inverse of the matrix

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 5 \end{bmatrix}$$

and hence solve the following system of equations.

$$\begin{aligned} x + y + z &= 3 \\ 2x + 3y + 5z &= 10. \end{aligned}$$

18. Find the least squares solution of the equations

$$\begin{aligned} x_1 + 2x_2 &= 3 \\ 2x_1 + 4x_2 &= 7 \end{aligned}.$$

19. Find the solution of the following equations using least squares method:

$$\begin{aligned} x+y+3z &= 0\\ 2x-y+4z &= 8\\ x+5y+z &= 10\\ x+y-2z &= 2. \end{aligned}$$

20. Solve the following equations by (i) Gauss-Jordan's and (ii) Gauss-Seidal's methods, correct up to four significant figures:

$$\begin{aligned} 9x+2y+4z &= 20\\ x+10y+4z &= 6\\ 2x-4y+10z &= -15. \end{aligned}$$

21. Test if the following systems of equations are diagonally dominant and hence solve them using Gauss-Jacobi's and Gauss-Seidal's methods.

$$\text{(i)}\quad \begin{aligned} 10x+15y+3z &= 14\\ -30x+y+5z &= 17\\ x+y+4z &= 3 \end{aligned} \qquad \text{(ii)}\quad \begin{aligned} x+3y+4z &= 7\\ 3x+2y+5z &= 10\\ x-5y+7z &= 3. \end{aligned}$$

22. Solve the following simultaneous equations

$$\begin{aligned} 2.5x_1+5.2x_2 &= 6.2\\ 1.251x_1+2.605x_2 &= 3.152 \end{aligned}$$

by Gauss elimination method and get your answer to 4 significant figures. Improve the solution by iterative refinement.

23. Consider the following linear system

$$5x_1+3x_2=6, \qquad 2x_1-x_2=4.$$

Can either Gauss-Jacobi's or Gauss-Seidal's iteration be used to solve this linear system? Why?

24. Consider the following tri-diagonal linear system and assume that the coefficient matrix is diagonally dominant.

$$\begin{array}{lllllllll} d_1x_1 & + & c_1x_2 & & & & & = & b_1\\ a_1x_1 & + & d_2x_2 & + & c_2x_3 & & & = & b_2\\ & & a_2x_2 & + & d_3x_3 & + & c_4x_4 & = & b_3\\ & & \ddots & & \ddots & & \ddots & & \vdots\\ & & & & a_{n-2}x_{n-2} & + & d_{n-1}x_{n-1} \;+\; c_{n-1}x_n & = & b_{n-1}\\ & & & & & & a_{n-1}x_{n-1} \;+\; d_nx_n & = & b_n. \end{array}$$

Write an iterative algorithm that will solve this system.

25. Use Gauss-Seidal's iteration to solve the following band system.

$$\begin{aligned}
12x_1 - 2x_2 + x_3 &= 5\\
-2x_1 + 12x_2 - 2x_3 + x_4 &= 5\\
x_1 - 2x_2 + 12x_3 - 2x_4 + x_5 &= 5\\
x_2 - 2x_3 + 12x_4 - 2x_5 + x_6 &= 5\\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots &\cdots\ \cdots\\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots &\cdots\ \cdots\\
x_{46} - 2x_{47} + 12x_{48} - 2x_{49} + x_{50} &= 5\\
x_{47} - 2x_{48} + 12x_{49} - 2x_{50} &= 5\\
x_{48} - 2x_{49} + 12x_{50} &= 5.
\end{aligned}$$

26. Solve the following systems of equations by successive relaxation method.

(i)
$$\begin{aligned}
2x - 3y + z &= -1\\
x + 4y - 3z &= 0\\
x + y + z &= 6
\end{aligned}$$
(ii)
$$\begin{aligned}
x + y - z &= 0\\
2x + 3y + 8z &= -1\\
5x - 4y + 10z &= 9.
\end{aligned}$$

27. Solve the following systems of equations by successive overrelaxation method.

(i)
$$\begin{aligned}
5x - 2y + z &= 4\\
7x + y - 5z &= 8\\
3x + 7y + 4z &= 10
\end{aligned}$$
(ii)
$$\begin{aligned}
x_1 + x_2 + x_3 &= 3\\
2x_1 - x_2 + 3x_3 &= 16\\
3x_1 + x_2 - x_3 &= -3.
\end{aligned}$$

Choose appropriate relaxation factor w.

28. The Hilbert matrix is a classical ill-conditioned matrix, and small changes in its coefficients will produce a large changes in the solution to the perturbed system.

(i) Solve $\mathbf{AX} = \mathbf{b}$ using the Hilbert matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 & 1/5\\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6\\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7\\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8\\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1\\0\\0\\0\\0 \end{bmatrix}.$$

(ii) Solve $\mathbf{Cx} = \mathbf{b}$ where

$$\mathbf{C} = \begin{bmatrix} 1.0 & 0.5 & 0.33333 & 0.25 & 0.2\\ 0.5 & 0.33333 & 0.25 & 0.2 & 0.16667\\ 0.33333 & 0.25 & 0.2 & 0.16667 & 0.14286\\ 0.25 & 0.2 & 0.16667 & 0.14286 & 0.125\\ 0.2 & 0.16667 & 0.14286 & 0.125 & 0.11111, \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1\\0\\0\\0\\0 \end{bmatrix}.$$

[Note that the two matrices **A** and **C** are different.]

Chapter 6

Eigenvalues and Eigenvectors of a Matrix

6.1 Eigenvalue of a Matrix

Let $\mathbf{A}$ be a square matrix of order n. Suppose $\mathbf{X}$ is a column matrix and λ is a scalar such that

$$\mathbf{AX} = \lambda\mathbf{X}. \tag{6.1}$$

The scalar λ is called an **eigenvalue** or **characteristic value** of the matrix $\mathbf{A}$ and $\mathbf{X}$ is called the corresponding **eigenvector**. The equation (6.1) can be written as $(\mathbf{A} - \lambda\mathbf{I})\mathbf{X} = 0$.

The equation

$$|\mathbf{A} - \lambda\mathbf{I}| = 0 \tag{6.2}$$

that is,

$$\begin{vmatrix} a_{11} - \lambda & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & a_{23} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} - \lambda \end{vmatrix} = 0 \tag{6.3}$$

is a polynomial in λ of degree n, called **characteristic equation** of the matrix $\mathbf{A}$. The roots $\lambda_i, i = 1, 2, \ldots, n$, of the equation (6.2) are the eigenvalues of $\mathbf{A}$. For each value of λ_i, there exists an $\mathbf{X_i}$ such that

$$\mathbf{AX_i} = \lambda_i\mathbf{X_i}. \tag{6.4}$$

The eigenvalues λ_i may be either distinct or repeated, they may be real or complex. If the matrix is real symmetric then all the eigenvalues are real. If the matrix is skew-symmetric then the eigenvalues are either zero or purely imaginary. Sometimes, the set of all eigenvalues, λ_i, of a matrix $\mathbf{A}$ is called the **spectrum** of $\mathbf{A}$ and the largest value of $|\lambda_i|$ is called the **spectral radius** of $\mathbf{A}$.

Example 6.1.1 Find the eigenvalues and eigenvectors of the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 2 \end{bmatrix}.$$

Solution. The characteristic equation of $\mathbf{A}$ is $|\mathbf{A} - \lambda\mathbf{I}| = 0$.
Therefore

$$\begin{vmatrix} 2-\lambda & -1 & 1 \\ -1 & 2-\lambda & -1 \\ 1 & -1 & 2-\lambda \end{vmatrix} = 0.$$

By direct expansion this gives $(1-\lambda)^2(4-\lambda) = 0$.
Hence the characteristic equation is $(1-\lambda)^2(4-\lambda) = 0$ and the eigenvalues of $\mathbf{A}$ are 1, 1, and 4. The two distinct eigenvectors corresponding to two eigenvalues $\lambda = 1$ and 4 are calculated below.

Eigenvector corresponding to the eigenvalue 1.
Let $(x_1, x_2, x_3)^T$ be the eigenvector corresponding to 1. Then

$$\begin{bmatrix} 2-1 & -1 & 1 \\ -1 & 2-1 & -1 \\ 1 & -1 & 2-1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Thus

$$\begin{aligned} x_1 - x_2 + x_3 &= 0 \\ -x_1 + x_2 - x_3 &= 0 \\ x_1 - x_2 + x_3 &= 0. \end{aligned}$$

The solution of this system of equations is $x_3 = 0, x_1 = x_2$. We take $x_2 = 1$. Then the eigenvector is $(1, 1, 0)^T$.
Let $(y_1, y_2, y_3)^T$ be the eigenvector corresponding to $\lambda = 4$.
Then

$$\begin{bmatrix} 2-4 & -1 & 1 \\ -1 & 2-4 & -1 \\ 1 & -1 & 2-4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Thus the system of equations are

$$\begin{aligned}
-2y_1 - y_2 + y_3 &= 0\\
-y_1 - 2y_2 - y_3 &= 0\\
y_1 - y_2 - 2y_3 &= 0.
\end{aligned}$$

The solution is $y_1 = k, y_2 = -k, y_3 = k$ for arbitrary k. Let $k = 1$. Then the eigenvector is $(1, -1, 1)^T$.

The upper bound of the eigenvalue of a matrix is stated below.

Theorem 6.1 *The largest eigenvalue (in magnitude) of a square matrix* **A** *cannot exceed the largest sum of the moduli of the elements along any row or any column, i.e.,*

$$|\lambda| \le \max_i \left[\sum_{j=1}^{n} |a_{ij}|\right] \text{ and } |\lambda| \le \max_j \left[\sum_{i=1}^{n} |a_{ij}|\right]. \tag{6.5}$$

Proof. Let λ be an eigenvalue of **A** and $\mathbf{X} = (x_1, x_2, \ldots, x_n)^T$ be the corresponding eigenvector. Then $\mathbf{AX} = \lambda\mathbf{X}$.

This equation may be written as

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= \lambda x_1\\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= \lambda x_2\\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \cdots \cdots\\
a_{k1}x_1 + a_{k2}x_2 + \cdots + a_{kn}x_n &= \lambda x_k\\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \cdots \cdots\\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= \lambda x_n.
\end{aligned}$$

Let $|x_k| = \max_i |x_i|$. Now, the kth equation is divided by x_k. Therefore,

$$\lambda = a_{k1}\frac{x_1}{x_k} + a_{k2}\frac{x_2}{x_k} + \cdots + a_{kk} + \cdots + a_{kn}\frac{x_n}{x_k}.$$

Since $\left|\dfrac{x_i}{x_k}\right| \le 1$ for $i = 1, 2, \ldots, n$,

$$|\lambda| \le |a_{k1}| + |a_{k2}| + \cdots + |a_{kk}| + \cdots + |a_{kn}| = \sum_{j=1}^{n} |a_{kj}|.$$

This is true for all rows k, hence

$$|\lambda| \le \max_i \left[\sum_{j=1}^{n} |a_{ij}|\right].$$

The theorem is also true for columns, as the eigenvalues of $\mathbf{A^T}$ and **A** are same. □

Theorem 6.2 (Shifting eigenvalues). *Suppose λ be an eigenvalue and $\mathbf{X}$ be its corresponding eigenvector of $\mathbf{A}$. If c is any constant, then $\lambda - c$ is an eigenvalue of the matrix $\mathbf{A} - c\mathbf{I}$ with same eigenvector $\mathbf{X}$.*

Let the characteristic polynomial of the matrix $\mathbf{A}$ be

$$det(\mathbf{A} - \lambda\mathbf{I}) = \lambda^n + c_1\lambda^{n-1} + c_2\lambda^{n-2} + \cdots + c_n, \tag{6.6}$$

where

$-c_1 = \sum_{i=1}^{n} a_{ii} = Tr.\ \mathbf{A}$, which is the sum of all diagonal elements of $\mathbf{A}$, called the **trace.**

$c_2 = \sum_{i<j} \begin{vmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{vmatrix}$, is the sum of all principal minors of order two of $\mathbf{A}$,

$-c_3 = \sum_{i<j<k} \begin{vmatrix} a_{ii} & a_{ij} & a_{ik} \\ a_{ji} & a_{jj} & a_{jk} \\ a_{ki} & a_{kj} & a_{kk} \end{vmatrix}$, is the sum of all principal minors of order three of $\mathbf{A}$,

and finally $(-1)^n c_n = det\mathbf{A}$, is the determinant of the matrix $\mathbf{A}$.

If the coefficients of the polynomial (6.6) are known then solving this polynomial using the method discussed in Chapter 4, one can determine all the eigenvalues. But, the direct expansion is very labourious and is only used for low order matrices. One efficient method to compute the coefficients of (6.6) is Leverrier-Faddeev method.

6.2 Leverrier-Faddeev Method to Construct Characteristic Equation

This method was developed by Leverrier and latter modified by Faddeev.

Let

$$det(\mathbf{A} - \lambda\mathbf{I}) = \lambda^n + c_1\lambda^{n-1} + c_2\lambda^{n-2} + \cdots + c_n \tag{6.7}$$

be the characteristic polynomial of the matrix $\mathbf{A}$ and its roots be $\lambda_1, \lambda_2, \ldots, \lambda_n$. Now, the sum of kth power of the eigenvalues is denoted by S_k, i.e.,

$$\begin{array}{lcl} S_1 &=& \lambda_1 + \lambda_2 + \cdots + \lambda_n = Tr\ \mathbf{A}, \\ S_2 &=& \lambda_1^2 + \lambda_2^2 + \cdots + \lambda_n^2 = Tr\ \mathbf{A^2}, \\ \cdots & \cdots & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ S_n &=& \lambda_1^n + \lambda_2^n + \cdots + \lambda_n^n = Tr\ \mathbf{A^n}. \end{array} \tag{6.8}$$

Then by Newton's formula (on polynomial) for $k \le n$

$$S_k + c_1S_{k-1} + c_2S_{k-2} + \cdots + c_{k-1}S_1 = -kc_k. \tag{6.9}$$

For $k = 1, 2, \ldots, n$, the coefficients are given by

$$\begin{aligned} c_1 &= -S_1 \\ c_2 &= -\frac{1}{2}(S_2 + c_1 S_1) \\ \cdots & \cdots \cdots\cdots\cdots \\ c_n &= -\frac{1}{n}(S_n + c_1 S_{n-1} + c_2 S_{n-2} + \cdots + c_{n-1} S_1). \end{aligned}$$

Thus computation of the coefficients $c_1, c_2, \ldots, c_n$ depends on the trace of $\mathbf{A}$, $\mathbf{A^2}$, …, $\mathbf{A^n}$.

The Leverrier method was modified by Faddeev by generating a sequence of matrices $\mathbf{B_1}, \mathbf{B_2}, \ldots, \mathbf{B_n}$, shown below.

$$\begin{array}{llll} \mathbf{B_1} = \mathbf{A}, & d_1 = Tr.\, \mathbf{B_1}, & \mathbf{D_1} = \mathbf{B_1} - d_1\mathbf{I} \\ \mathbf{B_2} = \mathbf{AD_1}, & d_2 = \frac{1}{2}\, Tr.\, \mathbf{B_2}, & \mathbf{D_2} = \mathbf{B_2} - d_2\mathbf{I} \\ \mathbf{B_3} = \mathbf{AD_2}, & d_3 = \frac{1}{3}\, Tr.\, \mathbf{B_3}, & \mathbf{D_3} = \mathbf{B_3} - d_3\mathbf{I} \\ \cdots \quad \cdots \cdots & \cdots \quad \cdots\cdots & \cdots\cdots \quad \cdots\cdots\cdots \\ \mathbf{B_{n-1}} = \mathbf{AD_{n-2}}, & d_{n-1} = \frac{1}{n-1}\, Tr.\, \mathbf{B_{n-1}}, & \mathbf{D_{n-1}} = \mathbf{B_{n-1}} - d_{n-1}\mathbf{I} \\ \mathbf{B_n} = \mathbf{AD_{n-1}}, & d_n = \frac{1}{n}\, Tr.\, \mathbf{B_n}, & \mathbf{D_n} = \mathbf{B_n} - d_n\mathbf{I} \end{array} \tag{6.10}$$

Thus the coefficients of the characteristic polynomial are

$$c_1 = -d_1, c_2 = -d_2, \ldots, c_n = -d_n.$$

It can be verified that $\mathbf{D_n} = \mathbf{0}$.

Example 6.2.1 Use the Leverrier-Faddeev method to find the characteristic polynomial of the matrix $\begin{bmatrix} 2 & 1 & -1 \\ 0 & 2 & 3 \\ 1 & 5 & 4 \end{bmatrix}$.

Solution.

$$\mathbf{B_1} = \mathbf{A} = \begin{bmatrix} 2 & 1 & -1 \\ 0 & 2 & 3 \\ 1 & 5 & 4 \end{bmatrix}, d_1 = Tr.\ \mathbf{B_1} = 2 + 2 + 4 = 8$$

$$\mathbf{B_2} = \mathbf{A}(\mathbf{B_1} - d_1\mathbf{I}) = \begin{bmatrix} 2 & 1 & -1 \\ 0 & 2 & 3 \\ 1 & 5 & 4 \end{bmatrix} \begin{bmatrix} -6 & 1 & -1 \\ 0 & -6 & 3 \\ 1 & 5 & -4 \end{bmatrix} = \begin{bmatrix} -13 & -9 & 5 \\ 3 & 3 & -6 \\ -2 & -9 & -2 \end{bmatrix}$$

$$d_2 = \frac{1}{2} Tr.\ \mathbf{B_2} = \frac{1}{2}(-13 + 3 - 2) = -6$$

$$\mathbf{B_3} = \mathbf{A}(\mathbf{B_2} - d_2\mathbf{I}) = \begin{bmatrix} 2 & 1 & -1 \\ 0 & 2 & 3 \\ 1 & 5 & 4 \end{bmatrix} \begin{bmatrix} -7 & -9 & 5 \\ 3 & 9 & -6 \\ -2 & -9 & 4 \end{bmatrix} = \begin{bmatrix} -9 & 0 & 0 \\ 0 & -9 & 0 \\ 0 & 0 & -9 \end{bmatrix}$$

$$d_3 = \frac{1}{3}Tr.\ \mathbf{B_3} = \frac{1}{3}(-9-9-9) = -9.$$

Thus $c_1 = -d_1 = -8, c_2 = -d_2 = 6, c_3 = -d_3 = 9$. Hence the characteristic polynomial is $\lambda^3 - 8\lambda^2 + 6\lambda + 9 = 0$.

Note 6.2.1 Using this method one can compute the inverse of the matrix **A**. It is mentioned that $\mathbf{D_n} = \mathbf{0}$. That is, $\mathbf{B_n} - d_n\mathbf{I} = \mathbf{0}$ or, $\mathbf{AD_{n-1}} = d_n\mathbf{I}$. From this relation one can write $\mathbf{D_{n-1}} = d_n\mathbf{A^{-1}}$. This gives,

$$\mathbf{A^{-1}} = \frac{\mathbf{D_{n-1}}}{d_n} = \frac{\mathbf{D_{n-1}}}{-c_n}. \tag{6.11}$$

Algorithm 6.1 (Leverrier-Faddeev method). This algorithm determines the characteristic polynomial $\lambda^n + c_1\lambda^{n-1} + \cdots + c_{n-1}\lambda + c_n = 0$ of a square matrix **A**.

```
Algorithm Leverrier-Faddeev
 Step 1: Read the matrix $\mathbf{A} = [a_{ij}], i, j = 1, 2, \ldots, n$.
 Step 2: Set $\mathbf{B_1} = \mathbf{A}, d_1 = \sum_{i=1}^{n} a_{ii}$.
 Step 3: for $i = 2, 3, \ldots, n$ do
             Compute
                 (a) $\mathbf{B_i} = \mathbf{A}(\mathbf{B_{i-1}} - d_{i-1}\mathbf{I})$
                 (b) $d_i = \frac{1}{i}$ (sum of the diagonal elements of $\mathbf{B_i}$)
 Step 4: Compute $c_i = -d_i$ for $i = 1, 2, \ldots, n$.
             //$c_i$'s are the coefficients of the polynomial.//
end Leverrier-Faddeev.
```

Program 6.1

```
/* Program Leverrier-Faddeev
   This program finds the characteristic polynomial of
   a square matrix. From which we can determine all the
   eigenvalues of the matrix. */
#include<stdio.h>
#include<math.h>
void main()
{
 int n,i,j,k,l;
 float a[10][10],b[10][10],c[10][10],d[11];
 printf("Enter the size of the matrix ");
 scanf("%d",&n);
 printf("Enter the elements row wise ");
 for(i=1;i<=n;i++) for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
 printf("The given matrix is\n");
 for(i=1;i<=n;i++) /* printing of A */
     {
       for(j=1;j<=n;j++) printf("%f\t ",a[i][j]); printf("\n");
     }
   printf("\n");
 for(i=1;i<=n;i++) for(j=1;j<=n;j++) b[i][j]=a[i][j];
 d[1]=0;for(i=1;i<=n;i++) d[1]+=a[i][i];
 for(i=2;i<=n;i++)
    {
       /* construction of B[i-1]-d[i-1] I=C (say) */
       for(j=1;j<=n;j++) for(k=1;k<=n;k++) c[j][k]=b[j][k];
       for(j=1;j<=n;j++) c[j][j]=c[j][j]-d[i-1];
       /* product of A and B[i-1]-d[i-1]I */
       for(j=1;j<=n;j++) for(k=1;k<=n;k++)
           {
             b[j][k]=0;
             for(l=1;l<=n;l++) b[j][k]+=a[j][l]*c[l][k];
           }
       /* trace of B */
       d[i]=0;
       for(j=1;j<=n;j++)
               d[i]+=b[j][j];
       d[i]/=i;
    } /* end of i loop */
printf("The coefficients of the characteristic polynomial are \n");
printf("1.00000 ");
for(i=1;i<=n;i++) printf("%8.5f ",-d[i]);
}/* main */
```

A sample of input/output:

```
Enter the size of the matrix 5
Enter the elements row wise
2 3 0 1 2
3 5 6 2 1
2 3 0 1 0
0 0 2 3 8
1 2 3 0 2
```

```
The given matrix is
2.000000     3.000000     0.000000     1.000000     2.000000
3.000000     5.000000     6.000000     2.000000     1.000000
2.000000     3.000000     0.000000     1.000000     0.000000
0.000000     0.000000     2.000000     3.000000     8.000000
1.000000     2.000000     3.000000     0.000000     2.000000

The coefficients of the characteristic polynomial are
1.00000 -12.00000 18.00000 -29.00000 -142.00000 14.00000
```

6.2.1 Eigenvectors using Leverrier-Faddeev method

The Leverrier-Faddeev method may also be used to determine all the eigenvectors. Suppose the matrices $\mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_{n-1}}$ and the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ are known. Then the eigenvectors $\mathbf{x^{(i)}}$ can be determined using the formula

$$\mathbf{x^{(i)}} = \lambda_i^{n-1}\mathbf{e_0} + \lambda_i^{n-2}\mathbf{e_1} + \lambda_i^{n-3}\mathbf{e_2} + \cdots + \mathbf{e_{n-1}}, \tag{6.12}$$

where $\mathbf{e_0}$ is a unit vector and $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_{n-1}}$ are column vectors of the matrices $\mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_{n-1}}$ of the same order as $\mathbf{e_0}$.

Example 6.2.2 Use Leverrier-Faddeev method to find characteristic equation and all eigenvectors of the matrix $\mathbf{A} = \begin{bmatrix} 9 & -1 & 9 \\ 3 & -1 & 3 \\ -7 & 1 & -7 \end{bmatrix}$.

Solution.

$$\begin{aligned}
\mathbf{B_1} &= \mathbf{A} = \begin{bmatrix} 9 & -1 & 9 \\ 3 & -1 & 3 \\ -7 & 1 & -7 \end{bmatrix} \\
d_1 &= Tr.\,\mathbf{B_1} = 9 - 1 - 7 = 1 \\
\mathbf{D_1} &= \mathbf{B_1} - d_1\mathbf{I} = \begin{bmatrix} 9 & -1 & 9 \\ 3 & -1 & 3 \\ -7 & 1 & -7 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 8 & -1 & 9 \\ 3 & -2 & 3 \\ -7 & 1 & -8 \end{bmatrix} \\
\mathbf{B_2} &= \mathbf{A}\mathbf{D_1} = \begin{bmatrix} 9 & -1 & 9 \\ 3 & -1 & 3 \\ -7 & 1 & -7 \end{bmatrix} \begin{bmatrix} 8 & -1 & 9 \\ 3 & -2 & 3 \\ -7 & 1 & -8 \end{bmatrix} = \begin{bmatrix} 6 & 2 & 6 \\ 0 & 2 & 0 \\ -4 & -2 & -4 \end{bmatrix} \\
d_2 &= \frac{1}{2}Tr.\,\mathbf{B_2} = \frac{1}{2}(6 + 2 - 4) = 2 \\
\mathbf{D_2} &= \mathbf{B_2} - d_2\mathbf{I} = \begin{bmatrix} 4 & 2 & 6 \\ 0 & 0 & 0 \\ -4 & -2 & -6 \end{bmatrix}
\end{aligned}$$

$$\mathbf{B_3} = \mathbf{AD_2} = \begin{bmatrix} 9 & -1 & 9 \\ 3 & -1 & 3 \\ -7 & 1 & -7 \end{bmatrix} \begin{bmatrix} 4 & 2 & 6 \\ 0 & 0 & 0 \\ -4 & -2 & -6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$d_3 = \frac{1}{3} Tr.\, \mathbf{B_3} = \frac{1}{3}(0+0+0) = 0.$$

Thus $c_1 = -d_1 = -1, c_2 = -d_2 = -2, c_3 = -d_3 = 0$.
The characteristic equation is $\lambda^3 - \lambda^2 - 2\lambda = 0$.
The eigenvalues are $\lambda_1 = 0, \lambda_2 = -1, \lambda_3 = 2$.

Let $\mathbf{e_0} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, and then $\mathbf{e_1} = \begin{bmatrix} 8 \\ 3 \\ -7 \end{bmatrix}$, $\mathbf{e_2} = \begin{bmatrix} 4 \\ 0 \\ -4 \end{bmatrix}$.

($\mathbf{e_1}, \mathbf{e_2}$ are the first columns of the matrices $\mathbf{D_1}, \mathbf{D_2}$).
The formula

$$\mathbf{x^{(i)}} = \lambda_i^2 \mathbf{e_0} + \lambda_i \mathbf{e_1} + \mathbf{e_2},$$

for $\lambda_1 = 0$ gives $\mathbf{x^{(1)}} = \begin{bmatrix} 4 \\ 0 \\ -4 \end{bmatrix}$.
Similarly, for $\lambda_2 = -1$,

$$\mathbf{x^{(2)}} = (-1)^2 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + (-1) \begin{bmatrix} 8 \\ 3 \\ -7 \end{bmatrix} + \begin{bmatrix} 4 \\ 0 \\ -4 \end{bmatrix} = \begin{bmatrix} -3 \\ -3 \\ 3 \end{bmatrix}$$

and for $\lambda_2 = 2$, $\mathbf{x^{(3)}} = 2^2 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 8 \\ 3 \\ -7 \end{bmatrix} + \begin{bmatrix} 4 \\ 0 \\ -4 \end{bmatrix} = \begin{bmatrix} 24 \\ 6 \\ -18 \end{bmatrix}$.

Thus the eigenvectors are $\begin{bmatrix} 4 \\ 0 \\ -4 \end{bmatrix}$, $\begin{bmatrix} -3 \\ -3 \\ 3 \end{bmatrix}$ and $\begin{bmatrix} 24 \\ 6 \\ -18 \end{bmatrix}$.

A square matrix $\mathbf{B}$ is said to be **similar** to another square matrix $\mathbf{A}$ if there exists a non-singular matrix $\mathbf{P}$ such that $\mathbf{B} = \mathbf{P^{-1}AP}$. The *similar matrices have the same eigenvalues.* A square matrix $\mathbf{A}$ is said to be **diagonalisable** if $\mathbf{A}$ is similar to a square diagonal matrix. A square matrix $\mathbf{A}$ of order n is diagonalisable iff $\mathbf{A}$ has n linearly independent eigenvectors. If $\mathbf{P^{-1}AP}$ is a diagonal matrix and $\mathbf{P}$ is orthogonal then $\mathbf{A}$ is said to be **orthogonally diagonalisable**. It can be proved that a matrix $\mathbf{A}$ is orthogonally diagonalisable iff $\mathbf{A}$ is real and symmetric.

If a matrix is either diagonal or lower triangular or upper triangular then its eigenvalues are the diagonal elements.

6.3 Eigenvalues for Arbitrary Matrices

Several methods are available to determine the eigenvalues and eigenvectors of a matrix. Here, two methods – Rutishauser and Power methods are introduced.

6.3.1 Rutishauser method

Let $\mathbf{A}$ be a square matrix. In this method, a convergent sequence of upper triangular matrices $\mathbf{A_1}, \mathbf{A_2}, \ldots$ are generated. The diagonal elements of the convergent matrix are the eigenvalues, if they are real.

The conversion is based on $\mathbf{LU}$ decomposition technique, where $\mathbf{L}$ and $\mathbf{U}$ are lower and upper triangular matrices. Initially, let $\mathbf{A} = \mathbf{A_1}$ and $\mathbf{A_1} = \mathbf{L_1U_1}$ with $l_{ii} = 1$. Then form $\mathbf{A_2} = \mathbf{U_1L_1}$. The matrices $\mathbf{A_1}$ and $\mathbf{A_2}$ are similar as $\mathbf{A_2} = \mathbf{U_1L_1} = \mathbf{U_1A_1U_1^{-1}}$ and they have same eigenvalues. Now, $\mathbf{A_2}$ is factorized in the form $\mathbf{A_2} = \mathbf{L_2U_2}$ with $l_{ii} = 1$. Then form $\mathbf{A_3} = \mathbf{U_2L_2}$. Proceeding this way we generate a sequence of similar matrices $\mathbf{A_1}, \mathbf{A_2}, \mathbf{A_3}, \ldots$. In general, this sequence converge to an upper triangular matrix or a near-triangular matrix $\overline{\mathbf{A}}$. If the eigenvalues are real, then they all lie on the leading diagonal of the matrix $\overline{\mathbf{A}}$. But, practically this method is complicated. Sometimes, the lower triangular matrix $\mathbf{L}$ is replaced by $\mathbf{Q}$, where $\mathbf{Q}$ is an unitary or orthogonal matrix. The $\mathbf{QU}$ decomposition is also not simple for practical computation.

Since the sequence $\{\mathbf{A_i}\}$ converges slowly the *shifting* technique may be used to accelerate its convergence. This technique is not discussed here.

Example 6.3.1 Find all the eigenvalues of the matrix $\begin{bmatrix} 4 & 2 \\ -1 & 1 \end{bmatrix}$ using Rutishauser method.

Solution. Let $\mathbf{A} = \mathbf{A_1} = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$.

That is, $\begin{bmatrix} 4 & 2 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{11}l_{21} & u_{12}l_{21} + u_{22} \end{bmatrix}$.

This gives $u_{11} = 4, u_{12} = 2, l_{21} = -1/4, u_{22} = 3/2$.

Therefore, $\mathbf{L_1} = \begin{bmatrix} 1 & 0 \\ -1/4 & 1 \end{bmatrix}, \mathbf{U_1} = \begin{bmatrix} 4 & 2 \\ 0 & 3/2 \end{bmatrix}$.

Form $\mathbf{A_2} = \mathbf{U_1L_1} = \begin{bmatrix} 4 & 2 \\ 0 & 3/2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1/4 & 1 \end{bmatrix} = \begin{bmatrix} 7/2 & 2 \\ -3/8 & 3/2 \end{bmatrix}$.

Again, let $\mathbf{A_2} = \mathbf{L_2U_2} = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$.

$\begin{bmatrix} 7/2 & 2 \\ -3/8 & 3/2 \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ u_{11}l_{21} & u_{12}l_{21} + u_{22} \end{bmatrix}$.

Solution is $u_{11} = 7/2, u_{12} = 2, l_{21} = -3/28, u_{22} = 12/7$.

Therefore, $\mathbf{L_2} = \begin{bmatrix} 1 & 0 \\ -3/28 & 1 \end{bmatrix}$, $\mathbf{U_2} = \begin{bmatrix} 7/2 & 2 \\ 0 & 12/7 \end{bmatrix}$.

Form $\mathbf{A_3} = \mathbf{U_2 L_2} = \begin{bmatrix} 7/2 & 2 \\ 0 & 12/7 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -3/28 & 1 \end{bmatrix} = \begin{bmatrix} 23/7 & 2 \\ -9/49 & 12/7 \end{bmatrix}$.

In this way, we find

$\mathbf{A_4} = \begin{bmatrix} 3.17391 & 2 \\ -0.10208 & 1.82609 \end{bmatrix}$, $\mathbf{A_5} = \begin{bmatrix} 3.10959 & 2 \\ -0.06080 & 1.89041 \end{bmatrix}$,

$\mathbf{A_6} = \begin{bmatrix} 3.07049 & 2 \\ -0.03772 & 1.92951 \end{bmatrix}$, $\mathbf{A_7} = \begin{bmatrix} 3.04591 & 2 \\ -0.02402 & 1.95409 \end{bmatrix}$,

$\mathbf{A_8} = \begin{bmatrix} 3.04591 & 2 \\ -0.00788 & 1.96986 \end{bmatrix}$, $\mathbf{A_9} = \begin{bmatrix} 3.04073 & 2 \\ -0.00512 & 1.97504 \end{bmatrix}$

and so on.

The sequence $\{\mathbf{A_i}\}$ converges slowly to an upper triangular matrix and the diagonal elements converge to the eigenvalues of $\mathbf{A}$. The exact eigenvalues are 3 and 2, which are approximated by the diagonal elements of $\mathbf{A_9}$.

6.3.2 Power method

Power method is generally used to find the eigenvalue, largest in magnitude, (sometimes called first eigenvalue) of a matrix $\mathbf{A}$. Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be all eigenvalues of the matrix $\mathbf{A}$. We assume that

$$|\lambda_1| > |\lambda_2| > |\lambda_3| > \cdots > |\lambda_n|,$$

i.e., λ_1 is largest in magnitude and $\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_n}$ be the eigenvectors corresponding to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ respectively. The method is applicable if the matrix $\mathbf{A}$ has n independent eigenvectors. Then any vector $\mathbf{X}$ in the (vector) space of eigenvectors $\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_n}$ can be written as

$$\mathbf{X} = \mathbf{c_1 X_1} + \mathbf{c_2 X_2} + \cdots + \mathbf{c_n X_n}. \tag{6.13}$$

Multiplying this relation by $\mathbf{A}$ and using the results $\mathbf{AX_1} = \boldsymbol{\lambda_1}\mathbf{X_1}$, $\mathbf{AX_2} = \boldsymbol{\lambda_2}\mathbf{X_2}$, $\ldots$, $\mathbf{AX_n} = \boldsymbol{\lambda_n}\mathbf{X_n}$, we obtain

$$\begin{aligned} \mathbf{AX} &= c_1\lambda_1\mathbf{X_1} + \mathbf{c_2}\boldsymbol{\lambda_2}\mathbf{X_2} + \cdots + \mathbf{c_n}\boldsymbol{\lambda_n}\mathbf{X_n} \\ &= \lambda_1\Big[c_1\mathbf{X_1} + \mathbf{c_2}\Big(\frac{\boldsymbol{\lambda_2}}{\boldsymbol{\lambda_1}}\Big)\mathbf{X_2} + \cdots + \mathbf{c_n}\Big(\frac{\boldsymbol{\lambda_n}}{\boldsymbol{\lambda_1}}\Big)\mathbf{X_n}\Big]. \end{aligned} \tag{6.14}$$

Again, multiplying this relation by $\mathbf{A}, \mathbf{A^2}, \ldots, \mathbf{A^k}$ successively and obtain the relations

$$\mathbf{A^2X} = \lambda_1^2\Big[c_1\mathbf{X_1} + c_2\Big(\frac{\lambda_2}{\lambda_1}\Big)^2\mathbf{X_2} + \cdots + c_n\Big(\frac{\lambda_n}{\lambda_1}\Big)^2\mathbf{X_n}\Big]. \tag{6.15}$$

$$\vdots \qquad\qquad \vdots$$

$$\mathbf{A^kX} = \lambda_1^k\Big[c_1\mathbf{X_1} + c_2\Big(\frac{\lambda_2}{\lambda_1}\Big)^k\mathbf{X_2} + \cdots + c_n\Big(\frac{\lambda_n}{\lambda_1}\Big)^k\mathbf{X_n}\Big]. \tag{6.16}$$

$$\mathbf{A^{k+1}X} = \lambda_1^{k+1}\Big[c_1\mathbf{X_1} + c_2\Big(\frac{\lambda_2}{\lambda_1}\Big)^{k+1}\mathbf{X_2} + \cdots + \mathbf{c_n}\Big(\frac{\lambda_\mathbf{n}}{\lambda_\mathbf{1}}\Big)^{\mathbf{k+1}}\mathbf{X_n}\Big]. \tag{6.17}$$

When $k \to \infty$, then right hand sides of (6.16) and (6.17) tend to $\lambda_1^k c_1\mathbf{X_1}$ and $\lambda_1^{k+1}c_1\mathbf{X_1}$, since $\left|\frac{\lambda_i}{\lambda_1}\right| < 1$ for $i = 2, \ldots, n$. Thus for $k \to \infty, \mathbf{A^kX} = \lambda_1^k c_1\mathbf{X_1}$ and $\mathbf{A^{k+1}X} = \lambda_1^{k+1}c_1\mathbf{X_1}$. That is, for $k \to \infty, \lambda_1\mathbf{A^k}\lambda = \mathbf{A^{k+1}X}$. It is well known that two vectors are equal if their corresponding components are same. That is,

$$\lambda_1 = \lim_{k\to\infty}\frac{\Big(\mathbf{A^{k+1}X}\Big)_r}{\Big(\mathbf{A^kX}\Big)_r},\ r = 1, 2, \ldots, n. \tag{6.18}$$

The symbol $(\mathbf{A^kX})_r$ denotes the rth component of the vector $\mathbf{A^kX}$.

If $|\lambda_2| \ll |\lambda_1|$, then the term within square bracket of (6.17) tend faster to $c_1\mathbf{X_1}$, i.e., the rate of convergence is fast.

To reduce the round off error, the method is carried out by normalizing (reducing the largest element to unity) the eigenvector at each iteration. Let $\mathbf{X_0}$ be a non-null initial (arbitrary) vector (non-orthogonal to $\mathbf{X_1}$) and we compute

$$\begin{aligned}\mathbf{Y_{i+1}} &= \mathbf{AX_i}\\ \mathbf{X_{i+1}} &= \frac{\mathbf{Y_{i+1}}}{\lambda^{(i+1)}}, \qquad \text{for } i = 0, 1, 2, \ldots.\end{aligned} \tag{6.19}$$

where $\lambda^{(i+1)}$ is the largest element in magnitude of $\mathbf{Y_{i+1}}$ and it is the $(i+1)$th approximate value of λ_1. Then

$$\lambda_1 = \lim_{k\to\infty}\frac{(\mathbf{Y_{k+1}})_r}{(\mathbf{X_k})_r}, \qquad r = 1, 2, \ldots, n. \tag{6.20}$$

and $\mathbf{X_{k+1}}$ is the eigenvector corresponding to the eigenvalue λ_1.

Note 6.3.1 The initial vector $\mathbf{X_0}$ is usually chosen as $\mathbf{X_0} = (1, 1, \cdots, 1)^T$. But, if the initial vector $\mathbf{X_0}$ is poor, then the formula (6.20) does not give λ_1, i.e., the limit of the ratio $\frac{(\mathbf{Y_{k+1}})_r}{(\mathbf{X_k})_r}$ may not exist. If this situation occurs, then the initial vector must be changed.

Note 6.3.2 The power method is also used to find the least eigenvalue of a matrix **A**. If **X** is the eigenvector corresponding to the eigenvalue λ then $\mathbf{AX} = \lambda\mathbf{X}$. If **A** is non-singular then $\mathbf{A^{-1}}$ exist. Therefore, $\mathbf{A^{-1}}(\mathbf{AX}) = \lambda\mathbf{A^{-1}X}$ or, $\mathbf{A^{-1}X} = \frac{1}{\lambda}\mathbf{X}$. This means that if λ is an eigenvalue of **A** then $\frac{1}{\lambda}$ is an eigenvalue of $\mathbf{A^{-1}}$ and the same eigenvector **X** corresponds to the eigenvalue $1/\lambda$ of the matrix $\mathbf{A^{-1}}$. Thus, if λ is largest (in magnitude) eigenvalue of **A** then $1/\lambda$ is the least eigenvalue of $\mathbf{A^{-1}}$.

Note 6.3.3 We observed that the coefficient X_j in (6.16) goes to zero in proposition to $(\lambda_j/\lambda_1)^k$ and that the speed of convergence is governed by the terms $(\lambda_2/\lambda_1)^k$. Consequently, the rate of convergence is linear.

Example 6.3.2 Find the largest eigenvalue in magnitude and corresponding eigenvector of the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 2 \\ -1 & 0 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

Solution. Let the initial vector be $\mathbf{X_0} = (1, 1, 1)^T$.
The first iteration is given by

$$\mathbf{Y_1} = \mathbf{AX_0} = \begin{bmatrix} 1 & 3 & 2 \\ -1 & 0 & 2 \\ 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 1 \\ 12 \end{bmatrix}.$$

Therefore $\lambda^{(1)} = 12$ and $\mathbf{X_1} = \dfrac{\mathbf{Y_1}}{12} = \begin{bmatrix} 0.50000 \\ 0.08333 \\ 1.0000 \end{bmatrix}.$

$$\mathbf{Y_2} = \mathbf{AX_1} = \begin{bmatrix} 1 & 3 & 2 \\ -1 & 0 & 2 \\ 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 0.50000 \\ 0.08333 \\ 1.00000 \end{bmatrix} = \begin{bmatrix} 2.75 \\ 1.5 \\ 6.83333 \end{bmatrix}$$

$$\lambda^{(2)} = 6.83333, \quad \mathbf{X_2} = \frac{\mathbf{Y_2}}{6.83333} = \begin{bmatrix} 0.40244 \\ 0.21951 \\ 1.0000 \end{bmatrix}.$$

$$\mathbf{Y_3} = \mathbf{AX_2} = \begin{bmatrix} 1 & 3 & 2 \\ -1 & 0 & 2 \\ 3 & 4 & 5 \end{bmatrix} \begin{bmatrix} 0.40244 \\ 0.21951 \\ 1.00000 \end{bmatrix} = \begin{bmatrix} 3.06098 \\ 1.59756 \\ 7.08537 \end{bmatrix}$$

$$\lambda^{(3)} = 7.08537, \quad \mathbf{X_3} = \begin{bmatrix} 0.43201 \\ 0.22547 \\ 1.00000 \end{bmatrix}.$$

$$\mathbf{Y_4} = \begin{bmatrix} 3.10843 \\ 1.56799 \\ 7.19793 \end{bmatrix}, \quad \mathbf{X_4} = \begin{bmatrix} 0.43185 \\ 0.21784 \\ 1. \end{bmatrix}, \quad \lambda^{(4)} = 7.19793.$$

$$\mathbf{Y_5} = \begin{bmatrix} 3.08691 \\ 1.56950 \\ 7.16672 \end{bmatrix}, \quad \mathbf{X_5} = \begin{bmatrix} 0.43050 \\ 0.21880 \\ 1. \end{bmatrix}, \quad \lambda^{(5)} = 7.16691.$$

$$\mathbf{Y_6} = \begin{bmatrix} 3.08691 \\ 1.56950 \\ 7.16672 \end{bmatrix}, \quad \mathbf{X_6} = \begin{bmatrix} 0.43073 \\ 0.21900 \\ 1. \end{bmatrix}, \quad \lambda^{(6)} = 7.16672.$$

$$\mathbf{Y_7} = \begin{bmatrix} 3.08772 \\ 1.56927 \\ 7.16818 \end{bmatrix}, \quad \mathbf{X_7} = \begin{bmatrix} 0.43075 \\ 0.21892 \\ 1.0 \end{bmatrix}, \quad \lambda^{(7)} = 7.16818.$$

$$\mathbf{Y_8} = \begin{bmatrix} 3.08752 \\ 1.56925 \\ 7.16795 \end{bmatrix}, \quad \mathbf{X_8} = \begin{bmatrix} 0.43074 \\ 0.21893 \\ 1.0 \end{bmatrix}, \quad \lambda^{(8)} = 7.16795.$$

$$\mathbf{Y_9} = \begin{bmatrix} 3.08752 \\ 1.56926 \\ 7.16792 \end{bmatrix}, \quad \mathbf{X_9} = \begin{bmatrix} 0.43074 \\ 0.21893 \\ 1.0 \end{bmatrix}, \quad \lambda^{(9)} = 7.16792.$$

$$\mathbf{Y_{10}} = \begin{bmatrix} 3.08753 \\ 1.56926 \\ 7.16794 \end{bmatrix}, \quad \mathbf{X_{10}} = \begin{bmatrix} 0.43074 \\ 0.21893 \\ 1.0 \end{bmatrix}, \quad \lambda^{(10)} = 7.16794.$$

The required largest eigenvalue is 7.1679 correct up to four decimal places and the corresponding eigenvector is

$$\begin{bmatrix} 0.43074 \\ 0.21893 \\ 1.00000 \end{bmatrix}.$$

Algorithm 6.2 (Power method). This method determines the largest eigenvalue (in magnitude) and its corresponding eigenvector of a square matrix **A**.

```
Algorithm Power_Method
 Step 1. Read the matrix $\mathbf{A} = [a_{ij}], i, j = 1, 2, \ldots, n$.
 Step 2. Set initial vector $\mathbf{X_0} = (1, 1, 1, \ldots, 1)^T$ of $n$ components.
 Step 3. Find the product $\mathbf{Y} = \mathbf{AX_0}$.
 Step 4. Find the largest element (in magnitude) of the vector $\mathbf{Y}$ and let it be $\lambda$.
 Step 5. Divide all the elements of $\mathbf{Y}$ by $\lambda$ and take it as $\mathbf{X_1}$, i.e., $\mathbf{X_1} = \mathbf{Y}/\lambda$.
 Step 6. //Let $\mathbf{X_0} = (x_{01}, x_{02}, \ldots, x_{0n})$ and $\mathbf{X_1} = (x_{11}, x_{12}, \ldots, x_{1n})$.//
         If $|x_{oi} - x_{1i}| > \varepsilon$ for at least $i$ then
             set $\mathbf{X_0} = \mathbf{X_1}$ and goto Step 3.
 Step 7. Print $\lambda$ as largest eigenvalue and corresponding eigenvector $\mathbf{X_1}$ of $\mathbf{A}$.
end Power_Method
```

Program 6.2

```
/* Program Power Method
   This program finds the largest eigenvalue (in magnitude)
   of a square matrix.*/
#include<stdio.h>
#include<math.h>
void main()
{
 int n,i,j,flag;
 float a[10][10],x0[10],x1[10],y[10],lambda,eps=1e-5;
 printf("Enter the size of the matrix ");
 scanf("%d",&n);
 printf("Enter the elements row wise ");
 for(i=1;i<=n;i++) for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
 printf("The given matrix is\n");
 for(i=1;i<=n;i++){ /* printing of A */
       for(j=1;j<=n;j++) printf("%f ",a[i][j]); printf("\n");
     }
   printf("\n");
 for(i=1;i<=n;i++) {
      x0[i]=1; x1[i]=1; /* initialization */
    }
 do
 {
    flag=0;
    for(i=1;i<=n;i++) x0[i]=x1[i]; /* reset x0 */
    for(i=1;i<=n;i++) /* product of A and X0 */
        {
           y[i]=0;
           for(j=1;j<=n;j++) y[i]+=a[i][j]*x0[j];
        }
    lambda=y[1]; /* finds maximum among y[i] */
    for(i=2;i<=n;i++) if(lambda<y[i]) lambda=y[i];
    for(i=1;i<=n;i++) x1[i]=y[i]/lambda;
    for(i=1;i<=n;i++) if(fabs(x0[i]-x1[i])>eps) flag=1;
 }while(flag==1);
 printf("The largest eigenvalue is %8.5f \n",lambda);
 printf("The corresponding eigenvector is \n");
 for(i=1;i<=n;i++) printf("%8.5f ",x1[i]);
}/* main */
```

A sample of input/output:

```
Enter the size of the matrix 5
Enter the elements row wise
3 4  5  6 7
0 0  2  1 3
3 4  5 -2 3
3 4 -2  3 4
0 1  2  0 0
The given matrix is
3.000000 4.000000 5.000000 6.000000 7.000000
0.000000 0.000000 2.000000 1.000000 3.000000
3.000000 4.000000 5.000000 -2.000000 3.000000
3.000000 4.000000 -2.000000 3.000000 4.000000
0.000000 1.000000 2.000000 0.000000 0.000000
The largest eigenvalue is 10.41317
The corresponding eigenvector is
 1.00000  0.20028  0.62435  0.41939  0.13915
```

6.3.3 Power method for least eigenvalue

It is mentioned earlier that if λ is the largest eigenvalue of $\mathbf{A}$ then $1/\lambda$ is the smallest eigenvalue of $\mathbf{A}$ and $1/\lambda$ can be obtained by finding the largest eigenvalue of $\mathbf{A^{-1}}$. But, computation of $\mathbf{A^{-1}}$ is a labourious process, so a simple process is needed. One simple method is introduced here.

If the largest eigenvalue (in magnitude) λ_1, of an $n \times n$ matrix $\mathbf{A}$ is known then the smallest magnitude eigenvalue can be computed by using power method for the matrix $\mathbf{B} = (\mathbf{A} - \lambda_1\mathbf{I})$ instead of $\mathbf{A}$. The eigenvalues of the matrix $\mathbf{B}$ are $\lambda_i^{'} = (\lambda_i - \lambda_1)$ (called shifting eigenvalues), $i = 1, 2, \ldots, n$, where λ_i are the eigenvalues of $\mathbf{A}$. Obviously, $\lambda_n^{'}$ is the largest magnitude eigenvalue of $\mathbf{B}$. Again, if $\mathbf{X_n}$ is the corresponding eigenvector, then $\mathbf{BX_n} = \lambda_n^{'}\mathbf{X_n}$ or, $(\mathbf{A} - \lambda_1\mathbf{I})\mathbf{X_n} = (\lambda_n - \lambda_1)\mathbf{X_n}$, i.e., $\mathbf{AX_n} = \lambda_n\mathbf{X_n}$. Hence $\mathbf{X_n}$ is also the eigenvector of $\mathbf{A}$, corresponding to the eigenvalue λ_n.

6.4 Eigenvalues for Symmetric Matrices

The methods discussed earlier are also applicable for symmetric matrices, but, due to the special properties of symmetric matrices some efficient methods are developed here. Among them three commonly used methods, viz., Jacobi, Givens and Householder are discussed here.

In linear algebra it is established that *all the eigenvalues of a real symmetric matrix are real.*

6.4.1 Jacobi's method

This method is widely used to find the eigenvalues and eigenvectors of a real symmetric matrix. Since all the eigenvalues of **A** are real and there exist a real orthogonal matrix **S** such that $\mathbf{S^{-1}AS}$ is a diagonal matrix **D**. As **D** and **A** are similar, the diagonal elements of **D** are the eigenvalues of **A**. But, the computation of the matrix **S** is not a simple task. It is obtained by a series of orthogonal transformations $\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_n}, \ldots$ as discussed below.

Let $|a_{ij}|$ be the largest element among the off-diagonal elements of **A**. Now, we construct an orthogonal matrix $\mathbf{S_1}$ whose elements are defined as

$$s_{ij} = -\sin\theta,\ s_{ji} = \sin\theta,\ s_{ii} = \cos\theta,\ s_{jj} = \cos\theta, \tag{6.21}$$

all other off-diagonal elements are zero and all other diagonal elements are unity. Thus $\mathbf{S_1}$ is of the form

$$\mathbf{S_1} = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & \cos\theta & \cdots & -\sin\theta & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & \sin\theta & \cdots & \cos\theta & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \tag{6.22}$$

where $\cos\theta,\ -\sin\theta,\ \sin\theta$ and $\cos\theta$ are at the positions $(i,i), (i,j), (j,i)$ and (j,j) respectively.

Let $\mathbf{A_1} = \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix}$ be a sub-matrix of **A** formed by the elements a_{ii}, a_{ij}, a_{ji} and a_{jj}. To reduce $\mathbf{A_1}$ to a diagonal matrix, an orthogonal transformation is applied which is defined as $\overline{\mathbf{S_1}} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$, where θ is an unknown quantity and it will be selected in such a way that $\mathbf{A_1}$ becomes diagonal.

Now,

$$\begin{aligned} & \overline{\mathbf{S_1}}^{-1}\mathbf{A_1}\overline{\mathbf{S_1}} \\ &= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \\ &= \begin{bmatrix} a_{ii}\cos^2\theta + a_{ij}\sin 2\theta + a_{jj}\sin^2\theta & (a_{jj} - a_{ii})\sin\theta\cos\theta + a_{ij}\cos 2\theta \\ (a_{jj} - a_{ii})\sin\theta\cos\theta + a_{ij}\cos 2\theta & a_{ii}\sin^2\theta - a_{ij}\sin 2\theta + a_{jj}\cos^2\theta \end{bmatrix}. \end{aligned}$$

This matrix becomes a diagonal matrix if $(a_{jj} - a_{ii})\sin\theta\cos\theta + a_{ij}\cos 2\theta = 0$, That is , if

$$\tan 2\theta = \frac{2a_{ij}}{a_{ii} - a_{jj}}. \tag{6.23}$$

The value of θ can be obtained from the following relation.

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{2a_{ij}}{a_{ii} - a_{jj}}\right). \tag{6.24}$$

This expression gives four values of θ, but, to get smallest rotation, θ should lie in $-\pi/4 \le \theta \le \pi/4$. The equation (6.24) is valid for all i, j if $a_{ii} \ne a_{jj}$. If $a_{ii} = a_{jj}$ then

$$\theta = \begin{cases} \dfrac{\pi}{4}, & \text{if} \quad a_{ij} > 0 \\ \\ -\dfrac{\pi}{4}, & \text{if} \quad a_{ij} < 0. \end{cases} \tag{6.25}$$

Thus the off-diagonal elements s_{ij} and s_{ji} of $\overline{\mathbf{S_1}}^{-1}\mathbf{A_1}\overline{\mathbf{S_1}}$ vanish and the diagonal elements are modified. The first diagonal matrix is obtained by computing $\mathbf{D_1} = \mathbf{S_1^{-1}A_1S_1}$. In the next step largest off-diagonal (in magnitude) element is selected from the matrix $\mathbf{D_1}$ and the above process is repeated to generate another orthogonal matrix $\mathbf{S_2}$ to compute $\mathbf{D_2}$. That is,

$$\mathbf{D_2} = \mathbf{S_2^{-1}D_1S_2} = \mathbf{S_2^{-1}}(\mathbf{S_1^{-1}AS_1})\mathbf{S_2} = (\mathbf{S_1S_2})^{-1}\mathbf{A}(\mathbf{S_1S_2}).$$

In this way, a series of two-dimensional rotations are performed. At the end of k transformations the matrix $\mathbf{D_k}$ is obtained as

$$\begin{aligned} \mathbf{D_k} &= \mathbf{S_k^{-1}S_{k-1}^{-1}\cdots S_1^{-1}AS_1S_2\cdots S_{k-1}S_k} \\ &= (\mathbf{S_1S_2\cdots S_k})^{-1}\mathbf{A}(\mathbf{S_1S_2\cdots S_k}) \\ &= \mathbf{S^{-1}AS} \end{aligned} \tag{6.26}$$

where $\mathbf{S} = \mathbf{S_1S_2\cdots S_k}$.

As $k \to \infty$, $\mathbf{D_k}$ tends to a diagonal matrix. The diagonal elements of $\mathbf{D_k}$ are the eigenvalues and the columns of $\mathbf{S}$ are the corresponding eigenvectors.

The method has a drawback. The elements those are transferred to zero during diagonalisation may not necessarily remain zero during subsequent rotations. The value of θ must be verified for its accuracy by checking whether $|\sin^2\theta + \cos^2\theta - 1|$ is sufficiently small.

Note 6.4.1 It may be noted that for orthogonal matrix $\mathbf{S}$, $\mathbf{S^{-1}} = \mathbf{S^T}$.

Note 6.4.2 It can be shown that the minimum number of rotations required to transform a real symmetric matrix into a diagonal matrix is $n(n-1)/2$.

Example 6.4.1 Find the eigenvalues and eigenvectors of the symmetric matrix $\mathbf{A} = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$ using Jacobi's method.

Solution. The largest off-diagonal element is 2 at $(1,2)$, $(1,3)$ and (2, 3) positions. The rotational angle θ is given by $\tan 2\theta = \dfrac{2a_{12}}{a_{11}-a_{22}} = \dfrac{4}{0} = \infty$ i.e., $\theta = \dfrac{\pi}{4}$.
Thus the orthogonal matrix $\mathbf{S_1}$ is

$$\mathbf{S_1} = \begin{bmatrix} \cos\pi/4 & -\sin\pi/4 & 0 \\ \sin\pi/4 & \cos\pi/4 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then the first rotation yields

$$\begin{aligned} \mathbf{D_1} &= \mathbf{S_1^{-1}AS_1} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 3 & 0 & 4/\sqrt{2} \\ 0 & -1 & 0 \\ 4/\sqrt{2} & 0 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 2.82843 \\ 0 & -1 & 0 \\ 2.82843 & 0 & 1 \end{bmatrix}. \end{aligned}$$

The largest off-diagonal element of $\mathbf{D_1}$ is now 2.82843 situated at $(1,3)$ position and hence the rotational angle is

$$\theta = \frac{1}{2}\tan^{-1}\frac{2a_{13}}{a_{11}-a_{33}} = 0.61548.$$

The second orthogonal matrix $\mathbf{S_2}$ is

$$\mathbf{S_2} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} = \begin{bmatrix} 0.81650 & 0 & -0.57735 \\ 0 & 1 & 0 \\ 0.57735 & 0 & 0.81650 \end{bmatrix}.$$

Then second rotation gives

$$\begin{aligned} \mathbf{D_2} &= \mathbf{S_2^{-1}D_1S_2} \\ &= \begin{bmatrix} 0.81650 & 0 & 0.57735 \\ 0 & 1 & 0 \\ -0.577351 & 0 & 0.81650 \end{bmatrix} \begin{bmatrix} 3 & 0 & 2.82843 \\ 0 & -1 & 0 \\ 2.82843 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.81650 & 0 & -0.57735 \\ 0 & 1 & 0 \\ 0.57735 & 0 & 0.81650 \end{bmatrix} \\ &= \begin{bmatrix} 5 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \end{aligned}$$

Thus $\mathbf{D_2}$ becomes a diagonal matrix and hence the eigenvalues are $5, -1, -1$.
The eigenvectors are the columns of $\mathbf{S}$, where

$$\begin{aligned}\mathbf{S} \;&=\; \mathbf{S_1 S_2} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.81650 & 0 & -0.57735 \\ 0 & 1 & 0 \\ 0.57735 & 0 & 0.81650 \end{bmatrix} \\ &= \begin{bmatrix} 0.57735 & -0.70711 & -0.40825 \\ 0.57735 & 0.70711 & -0.40825 \\ 0.57735 & 0 & 0.81650 \end{bmatrix}.\end{aligned}$$

Hence the eigenvalues are $5, -1, -1$ and the corresponding eigenvectors are $(0.57735, 0.57735, 0.57735)^T$, $(-0.70711, -0.70711, 0)^T$, $(-0.40825, -0.40825, 0.81650)^T$ respectively.
Note that the eigenvectors are normalized and two independent eigenvectors (last two vectors) for the eigenvalue -1 are obtained by this method.

In this problem, two rotations are used to convert $\mathbf{A}$ into a diagonal matrix. But, this does not happen in general. The following example shows that at least six rotations are needed to diagonalise a symmetric matrix.

Example 6.4.2 Find all the eigenvalues and eigenvectors of the matrix $\begin{bmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ by Jacobi's method.

Solution. Let $\mathbf{A} = \begin{bmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 1 & 2 & 1 \end{bmatrix}$.
It is a real symmetric matrix and the Jacobi's method is applicable.
The largest off-diagonal element is at $a_{12} = a_{21}$ and it is 3.

Then $\tan 2\theta = \dfrac{2a_{12}}{a_{11} - a_{22}} = \dfrac{6}{0} = \infty$, and this gives $\theta = \dfrac{\pi}{4}$.

Thus the orthogonal matrix $\mathbf{S_1}$ is

$$\mathbf{S_1} = \begin{bmatrix} \cos\pi/4 & -\sin\pi/4 & 0 \\ \sin\pi/4 & \cos\pi/4 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The first rotation gives

$$\mathbf{D_1} \;=\; \mathbf{S_1^{-1} A S_1} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 1 \\ 3 & 2 & 2 \\ 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & 0 & 3/\sqrt{2} \\ 0 & -1 & 1/\sqrt{2} \\ 3/\sqrt{2} & 1/\sqrt{2} & 1 \end{bmatrix}.$$

The largest off-diagonal element of $\mathbf{D_1}$ is $3/\sqrt{2}$, situated at $(1,3)$ position. Then

$\tan 2\theta = \dfrac{2a_{13}}{a_{11}-a_{33}} = \dfrac{6/\sqrt{2}}{5-1} = 1.06066.$ or, $\theta = \frac{1}{2}\tan^{-1}(0.69883) = 0.407413$

So, the next orthogonal matrix $\mathbf{S_2}$ is $\mathbf{S_2} = \begin{bmatrix} 0.91815 & 0 & -0.39624 \\ 0 & 1 & 0 \\ 0.39624 & 0 & 0.91815 \end{bmatrix}$.

$$\begin{aligned}\mathbf{D_2} &= \mathbf{S_2^{-1}D_1S_2} \\ &= \begin{bmatrix} 0.91815 & 0 & 0.39624 \\ 0 & 1 & 0 \\ -0.39624 & 0 & 0.91815 \end{bmatrix} \begin{bmatrix} 5 & 0 & 2.12132 \\ 0 & -1 & 0.70711 \\ 2.12132 & 0.70711 & 1 \end{bmatrix} \begin{bmatrix} 0.91815 & 0 & -0.39624 \\ 0 & 1 & 0 \\ 0.39624 & 0 & 0.91815 \end{bmatrix} \\ &= \begin{bmatrix} 5.91548 & 0.28018 & 0 \\ 0.28018 & -1.0 & 0.64923 \\ 0 & 0.64923 & 0.08452 \end{bmatrix}.\end{aligned}$$

The largest off-diagonal element of $\mathbf{D_2}$ is 0.64923, present at the position $(2,3)$. Then

$$\tan 2\theta = \frac{2a_{23}}{a_{22}-a_{33}} = -1.19727 \quad \text{or,} \quad \theta = \frac{1}{2}\tan^{-1}(-1.19727) = -0.43747.$$

Therefore, $\mathbf{S_3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.90583 & 0.42365 \\ 0 & -0.42365 & 0.90583 \end{bmatrix}$.

$$\mathbf{D_3} = \mathbf{S_3^{-1}D_2S_3} = \begin{bmatrix} 5.91548 & 0.25379 & 0.11870 \\ 0.25379 & -1.30364 & 0 \\ 0.11870 & 0 & 0.38816 \end{bmatrix}.$$

Again, largest off-diagonal element is 0.25379, located at $(1,2)$ position.

Therefore, $\theta = \dfrac{1}{2}\tan^{-1}\dfrac{2a_{12}}{a_{11}-a_{22}} = 0.03510.$

$$\mathbf{S_4} = \begin{bmatrix} 0.99938 & -0.03509 & 0 \\ 0.03509 & 0.99938 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$$\mathbf{D_4} = \mathbf{S_4^{-1}D_3S_4} = \begin{bmatrix} 5.92439 & 0 & 0.11863 \\ 0 & -1.31255 & -0.00417 \\ 0.11863 & -0.00417 & 0.38816 \end{bmatrix}.$$

Here largest off-diagonal element is 0.11863 at $(1, 3)$ position. In this case

$$\theta = \frac{1}{2}\tan^{-1}\frac{2a_{13}}{a_{11} - a_{33}} = 0.02141.$$

$$\mathbf{S_5} = \begin{bmatrix} 0.99977 & 0 & -0.02141 \\ 0 & 1 & 0 \\ 0.02141 & 0 & 0.99977 \end{bmatrix}.$$

$$\mathbf{D_5} = \mathbf{S_5^{-1}D_4S_5} = \begin{bmatrix} 5.92693 & -0.00009 & 0 \\ -0.00009 & -1.31255 & -0.00417 \\ 0 & -0.00417 & 0.38562 \end{bmatrix}.$$

The largest off-diagonal element in magnitude is -0.00417 situated at $(2, 3)$ position. Then

$$\theta = \frac{1}{2}\tan^{-1}\frac{2a_{23}}{a_{22} - a_{33}} = 0.00246.$$

$$\mathbf{S_6} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -0.00246 \\ 0 & 0.00246 & 1 \end{bmatrix}.$$

$$\mathbf{D_6} = \mathbf{S_6^{-1}D_5S_6} = \begin{bmatrix} 5.92693 & -0.00009 & 0 \\ -0.00009 & -1.31256 & 0 \\ 0 & 0 & 0.38563 \end{bmatrix}.$$

This matrix is almost diagonal and hence the eigenvalues are $5.9269, -1.3126$ and 0.3856 correct up to four decimal places.
The eigenvectors are the columns of

$$\mathbf{S} = \mathbf{S_1S_2}\ldots\mathbf{S_6} = \begin{bmatrix} 0.61852 & -0.54567 & -0.56540 \\ 0.67629 & 0.73604 & 0.02948 \\ 0.40006 & -0.40061 & 0.82430 \end{bmatrix}.$$

That is, the eigenvectors corresponding to the eigenvalues 5.9269, -1.3126 and 0.3856 are respectively $(0.61825, 0.67629, 0.40006)^T$,
$(-0.54567, 0.73604, -0.40061)^T$ and $(-0.56540, 0.02948, 0.82430)^T$.

Note 6.4.3 This example shows that the elements which were annihilated by a rotation may not remain zero during the next rotations.

Algorithm 6.3 (Jacobi's method). This method determines the eigenvalues and eigenvectors of a real symmetric matrix **A**, by converting **A** into a diagonal matrix by similarity transformation.

```
Algorithm Jacobi
 Step 1. Read the symmetric matrix $\mathbf{A} = [a_{ij}], i, j = 1, 2, \ldots, n$.
 Step 2. Initialize $\mathbf{D} = \mathbf{A}$ and $\mathbf{S} = \mathbf{I}$, a unit matrix.
Step 3. Find the largest off-diagonal element (in magnitude) from $\mathbf{D} = [d_{ij}]$ and
        let it be $d_{ij}$.
Step 4. //Find the rotational angle $\theta$.//
        If $d_{ii} = d_{jj}$ then
            if $d_{ij} > 0$ then $\theta = \pi/4$ else $\theta = -\pi/4$ endif;
        else
            $\theta = \frac{1}{2}\tan^{-1}\left(\frac{2d_{ij}}{d_{ii} - d_{jj}}\right)$;
        endif;
Step 5. //Compute the matrix $\mathbf{S_1} = [s_{pq}]$//
        Set $s_{pq} = 0$ for all $p, q = 1, 2, \ldots, n$
        $s_{kk} = 1, k = 1, 2, \ldots, n$
        and $s_{ii} = s_{jj} = \cos\theta, s_{ij} = -\sin\theta, s_{ji} = \sin\theta$.
Step 6. Find $\mathbf{D} = \mathbf{S_1^T} * \mathbf{D} * \mathbf{S_1}$ and $\mathbf{S} = \mathbf{S} * \mathbf{S_1}$;
Step 7. Repeat steps 3 to 6 until $\mathbf{D}$ becomes diagonal.
Step 8. Diagonal elements of $\mathbf{D}$ are the eigenvalues and the columns of $\mathbf{S}$ are
        the corresponding eigenvectors.
end Jacobi
```

Program 6.3

```c
/* Program Jacobi's Method to find eigenvalues
   This program finds all the eigenvalues and the corresponding
   eigenvectors of a real symmetric matrix. Assume that the
   given matrix is real symmetric. */
#include<stdio.h>
#include<math.h>
void main()
{
 int n,i,j,p,q,flag;
 float a[10][10],d[10][10],s[10][10],s1[10][10],s1t[10][10];
 float temp[10][10],theta,zero=1e-4,max,pi=3.141592654;
 printf("Enter the size of the matrix ");
 scanf("%d",&n);
 printf("Enter the elements row wise ");
 for(i=1;i<=n;i++) for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
 printf("The given matrix is\n");
 for(i=1;i<=n;i++) /* printing of A */
     {
       for(j=1;j<=n;j++) printf("%8.5f ",a[i][j]); printf("\n");
     }
   printf("\n");
```

```
/* initialization of D and S */
for(i=1;i<=n;i++) for(j=1;j<=n;j++){
      d[i][j]=a[i][j]; s[i][j]=0;
    }
for(i=1;i<=n;i++) s[i][i]=1;
do
{
   flag=0;
   /* find largest off-diagonal element */
   i=1; j=2; max=fabs(d[1][2]);
   for(p=1;p<=n;p++) for(q=1;q<=n;q++)
      { if(p!=q)  /* off-diagonal element */
          if(max<fabs(d[p][q])){
               max=fabs(d[p][q]); i=p; j=q;
            }
      }
   if(d[i][i]==d[j][j]){
        if(d[i][j]>0) theta=pi/4; else theta=-pi/4;
      }
   else
      {
        theta=0.5*atan(2*d[i][j]/(d[i][i]-d[j][j]));
      }
   /* construction of the matrix S1 and S1T */
   for(p=1;p<=n;p++) for(q=1;q<=n;q++)
       {s1[p][q]=0; s1t[p][q]=0;}
   for(p=1;p<=n;p++) {s1[p][p]=1; s1t[p][p]=1;}
   s1[i][i]=cos(theta);  s1[j][j]=s1[i][i];
   s1[j][i]=sin(theta);  s1[i][j]=-s1[j][i];
   s1t[i][i]=s1[i][i];   s1t[j][j]=s1[j][j];
   s1t[i][j]=s1[j][i];   s1t[j][i]=s1[i][j];
   /* product of S1T and D  */
   for(i=1;i<=n;i++)
      for(j=1;j<=n;j++){
              temp[i][j]=0;
              for(p=1;p<=n;p++) temp[i][j]+=s1t[i][p]*d[p][j];
           }
   /* product of temp and S1 i.e., D=S1T*D*S1 */
   for(i=1;i<=n;i++)
      for(j=1;j<=n;j++){
```

```
               d[i][j]=0;
               for(p=1;p<=n;p++) d[i][j]+=temp[i][p]*s1[p][j];
            }
    /* product of S and S1 i.e., S=S*S1 */
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            {
               temp[i][j]=0;
               for(p=1;p<=n;p++) temp[i][j]+=s[i][p]*s1[p][j];
            }
    for(i=1;i<=n;i++) for(j=1;j<=n;j++) s[i][j]=temp[i][j];
    for(i=1;i<=n;i++) for(j=1;j<=n;j++) /* is D diagonal ? */
        {
          if(i!=j) if(fabs(d[i][j]>zero)) flag=1;
        }
 }while(flag==1);
 printf("The eigenvalues are\n");
 for(i=1;i<=n;i++) printf("%8.5f ",d[i][i]);
 printf("\nThe corresponding eigenvectors are \n");
 for(j=1;j<=n;j++){
        printf("(");
        for(i=1;i<n;i++) printf("%8.5f,",s[i][j]);
        printf("%8.5f)\n",s[n][j]);
     }
}/* main */
```

A sample of input/output:

```
Enter the size of the matrix 4
Enter the elements row wise
1  2 3 4
2 -3 3 4
3  3 4 5
4  4 5 0
The given matrix is
1.000000 2.000000 3.000000 4.000000
2.000000 -3.000000 3.000000 4.000000
3.000000 3.000000 4.000000 5.000000
4.000000 4.000000 5.000000 0.000000
The eigenvalues are
-0.73369 -5.88321 11.78254 -3.16564
```

```
The corresponding eigenvectors are
( 0.74263, 0.04635,-0.65234, 0.14421)
( 0.13467, 0.74460, 0.06235,-0.65081)
( 0.43846, 0.33395, 0.64097, 0.53422)
(-0.48797, 0.57611,-0.39965, 0.51987)
```

6.4.2 Eigenvalues of a Symmetric Tri-diagonal Matrix

Let

$$\mathbf{A} = \begin{bmatrix} a_1 & b_2 & 0 & 0 & \cdots & 0 & 0 \\ b_2 & a_2 & b_3 & 0 & \cdots & 0 & 0 \\ 0 & b_3 & a_3 & b_4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & b_n & a_n \end{bmatrix}$$

be a symmetric tri-diagonal matrix. The characteristic equation of this matrix is

$$p_n(\lambda) = \begin{bmatrix} a_1-\lambda & b_2 & 0 & 0 & \cdots & 0 & 0 \\ b_2 & a_2-\lambda & b_3 & 0 & \cdots & 0 & 0 \\ 0 & b_3 & a_3-\lambda & b_4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & b_n & a_n-\lambda \end{bmatrix} = 0.$$

Expanding by minors, the sequence $\{p_n(\lambda)\}$ satisfies the following equations.

$$\begin{aligned} p_0(\lambda) &= 1 \\ p_1(\lambda) &= a_1 - \lambda \\ p_{k+1}(\lambda) &= (a_{k+1}-\lambda)p_k(\lambda) - b_{k+1}^2 p_{k-1}(\lambda),\ k = 1, 2, \ldots, n \end{aligned} \tag{6.27}$$

The polynomial $p_n(\lambda)$ is the characteristic polynomial of $\mathbf{A}$.

If none of $b_2, b_3, \ldots, b_n$ vanish, then $\{p_n(\lambda)\}$ is a Sturm sequence. Then using the property of Sturm sequence, one can determine the intervals (containing the eigenvalue), by substituting different values of λ. That is, if $N(\lambda)$ denotes the number of changes in sign of the Sturm sequence for a given value of λ, then $\mid N(a) - N(b) \mid$ represents the number of zeros (eigenvalues) lie in $[a, b]$, provided $p_n(a)$ and $p_n(b)$ are not zero. Once the location of an eigenvalue is identified then using any iterative method such as bisection method, Newton-Raphson method etc. one can determine it.

Having computed the eigenvalues of $\mathbf{A}$, the eigenvectors of $\mathbf{A}$ can be directly computed by solving the resulting homogeneous linear equations $(\mathbf{A} - \lambda\mathbf{I})\mathbf{X} = \mathbf{0}$.

Example 6.4.3 Find the eigenvalues of the following tri-diagonal matrix $\begin{bmatrix} 3 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix}$.

Solution. The Sturm sequence $\{p_n(\lambda)\}$ is given by

$$\begin{aligned} p_0(\lambda) &= 1 \\ p_1(\lambda) &= 3-\lambda \\ p_2(\lambda) &= (2-\lambda)p_1(\lambda) - 1p_0(\lambda) = \lambda^2 - 5\lambda + 5 \\ p_3(\lambda) &= (1-\lambda)p_2(\lambda) - 1p_1(\lambda) = -\lambda^3 + 6\lambda^2 - 9\lambda + 2. \end{aligned}$$

Now tabulate the values of p_0, p_1, p_2, p_3 for different values of λ.

λ	p_0	p_1	p_2	p_3	$N(\lambda)$
–1	+	+	+	+	0
0	+	+	+	+	0
1	+	+	+	–	1
2	+	+	–	0	1
3	+	0	–	+	2
4	+	–	+	–	3

Here $p_3(2) = 0$, so $\lambda = 2$ is an eigenvalue. The other two eigenvalues lie in the intervals (0, 1) and (3, 4) as $N(1) - N(0) = 1$ and $N(4) - N(3) = 1$.

To find the eigenvalue within (0, 1) using Newton-Raphson method

Let $\lambda^{(i)}$ be the ith approximate value of the eigenvalue in (0, 1). The Newton-Raphson iteration scheme is

$$\lambda^{(i+1)} = \lambda^{(i)} - \frac{p_3(\lambda^{(i)})}{p_3'(\lambda^{(i)})}.$$

Initially, let $\lambda^{(0)} = 0.5$. The successive iterations are shown below.

$\lambda^{(i)}$	$p_3(\lambda^{(i)})$	$p_3'(\lambda^{(i)})$	$\lambda^{(i+1)}$
0.5	–1.12500	–3.75000	0.20000
0.2	0.43200	–6.72000	0.26429
0.26429	0.02205	–6.03811	0.26794
0.26794	0.00007	–6.00012	0.26795
0.26795	0.00000	–6.00000	0.26795

Hence the other eigenvalue is 0.26795.

To find the eigenvalue within (3, 4) using Newton-Raphson method

Let the initial eigenvalue be $\lambda^{(0)} = 3.5$.

The calculations are shown in the following table.

$\lambda^{(i)}$	$p_3(\lambda^{(i)})$	$p_3'(\lambda^{(i)})$	$\lambda^{(i+1)}$
3.5	1.12500	–3.75000	3.80000
3.80000	–0.43200	–6.72000	3.73572
3.73572	–0.02206	–6.03812	3.73206
3.73206	–0.00007	–6.00012	3.73205
3.73205	0.00000	–6.00000	3.73205

The other eigenvalue is 3.73205.
Hence all the eigenvalues are 2, 0.26795 and 3.73205, the exact values are 2, $2 \pm \sqrt{3}$.

6.4.3 Givens method

The Givens method is used to find eigenvalues of a real symmetric matrix $\mathbf{A} = [a_{ij}]$. This method preserves the zeros in the off-diagonal elements, once they are created. This method works into two steps. In first step, the given symmetric matrix is converted to a symmetric tri-diagonal matrix using plane rotations. In second step, the eigenvalues of this new matrix are determined by the method discussed in previous section.

The conversion to a tri-diagonal form is done by using orthogonal transformations as in Jacobi's method. In this case, the rotation is performed with the elements a_{22}, a_{23}, a_{32} and a_{33}.

Let $\mathbf{S_1} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \cos\theta & -\sin\theta & 0 & \cdots & 0 \\ 0 & \sin\theta & \cos\theta & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$ be the orthogonal matrix, where θ is unknown.

Let $\mathbf{B_1}$ be the transformed matrix under orthogonal transformation $\mathbf{S_1}$, Then $\mathbf{B_1} = \mathbf{S_1^{-1} A S_1}$.

The elements of (1, 3) and (3, 1) positions are equal and they are $-a_{12}\sin\theta + a_{13}\cos\theta$. The angle θ is now obtained by putting this value to zero.

That is,

$$-a_{12}\sin\theta + a_{13}\cos\theta = 0 \qquad \text{or,} \qquad \tan\theta = \frac{a_{13}}{a_{12}}. \tag{6.28}$$

This transformation is now considered as a rotation in the (2,3)-plane. It may be noted that this computation is more simple than Jacobi's method. The matrix $\mathbf{B_1}$ has the form

$$\mathbf{B_1} = \begin{bmatrix} a'_{11} & a'_{12} & 0 & a'_{14} & \cdots & a'_{1n} \\ a'_{21} & a'_{22} & a'_{23} & a'_{24} & \cdots & a'_{2n} \\ 0 & a'_{32} & a'_{33} & a'_{34} & \cdots & a'_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a'_{n1} & a'_{n2} & a'_{n3} & a'_{n4} & \cdots & a'_{nn} \end{bmatrix}.$$

Then apply the rotation in (2,4)-plane to convert a'_{14} and a'_{41} to zero. This would not effect zeros that have been created earlier. Successive rotations are carried out in the planes (2, 3), (2, 4), $\ldots, (2, n)$ where θ's are so chosen that the new elements at the positions (1, 3), (1, 4), $\ldots, (1, n)$ vanish. After $(n-2)$ such rotations, all elements of first row and column (except first two) become zero. Then the transformed matrix $\mathbf{B_{n-2}}$ after $(n-2)$ rotations reduces to the following form:

$$\mathbf{B_{n-2}} = \begin{bmatrix} a''_{11} & a''_{12} & 0 & 0 & \cdots & 0 \\ a''_{21} & a''_{22} & a''_{23} & a''_{24} & \cdots & a''_{2n} \\ 0 & a''_{32} & a''_{33} & a''_{34} & \cdots & a''_{3n} \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & a''_{n2} & a''_{n3} & a''_{n4} & \cdots & a''_{nn} \end{bmatrix}.$$

The second row of $\mathbf{B_{n-2}}$ is taken in the same way as of the first row. The rotations are made in the planes (3,4), (3,5), $\ldots, (3, n)$. Thus, after $(n-2) + (n-3) + \cdots + 1$ $= \dfrac{(n-1)(n-2)}{2}$ rotations the matrix $\mathbf{A}$ becomes a tri-diagonal matrix $\mathbf{B}$ of the form

$$\mathbf{B} = \begin{bmatrix} a_1 & b_2 & 0 & 0 & \cdots & 0 & 0 \\ b_2 & a_2 & b_3 & 0 & \cdots & 0 & 0 \\ 0 & b_3 & a_3 & b_4 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & b_n & a_n \end{bmatrix}.$$

In this process, the previously created zeros are not effected by successive rotations. The eigenvalues of $\mathbf{B}$ and $\mathbf{A}$ are same as they are similar matrices.

Example 6.4.4 Find the eigenvalues of the symmetric matrix
$\mathbf{A} = \begin{bmatrix} 2 & 3 & -1 \\ 3 & 1 & 2 \\ -1 & 2 & -1 \end{bmatrix}$ using Givens method.

Solution. Let the orthogonal matrix $\mathbf{S_1}$ be

$$\mathbf{S_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix},$$

where $\tan\theta = \dfrac{a_{13}}{a_{12}} = -\dfrac{1}{3}$, i.e., $\theta = -0.32175$.

Therefore, $\mathbf{S_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.94868 & 0.31623 \\ 0 & -0.31623 & 0.94868 \end{bmatrix}$.

The reduced matrix is

$$\mathbf{B} = \mathbf{S_1^{-1}AS_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.94868 & -0.31623 \\ 0 & 0.31623 & 0.94868 \end{bmatrix} \begin{bmatrix} 2 & 3 & -1 \\ 3 & 1 & 2 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.94868 & 0.31623 \\ 0 & -0.31623 & 0.94868 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 3.16228 & 0.00001 \\ 3.16228 & -0.40001 & 2.2 \\ 0.00001 & 2.2 & 0.40001 \end{bmatrix}.$$

Let $\mathbf{B} = \begin{bmatrix} 2 & 3.1623 & 0 \\ 3.1623 & -0.4 & 2.2 \\ 0 & 2.2 & 0.4 \end{bmatrix}$.

The Sturm sequence is

$$\begin{aligned} p_0(\lambda) &= 1 \\ p_1(\lambda) &= 2-\lambda \\ p_2(\lambda) &= (-0.4-\lambda)p_1(\lambda) - 3.1623^2 p_0(\lambda) = \lambda^2 - 1.6\lambda - 10.8 \\ p_3(\lambda) &= (0.4-\lambda)p_2(\lambda) - 2.2^2 p_1(\lambda) = -\lambda^3 + 2\lambda^2 + 15\lambda - 14. \end{aligned}$$

Now, we tabulate the values of p_0, p_1, p_2, p_3 for different values of λ.

λ	p_0	p_1	p_2	p_3	$N(\lambda)$
-4	+	+	+	+	0
-3	+	+	+	−	1
-1	+	+	−	−	1
0	+	+	−	−	1
1	+	+	−	+	2
2	+	0	−	+	2
4	+	−	−	+	2
5	+	−	+	−	3

From this table, it is observed that the eigenvalues are located in the intervals $(-4, -3)$, (0, 1) and (4, 5). Any iterative method may be used to find them. Using Newton-Raphson method, we find the eigenvalues $-3.47531, 0.87584$ and 4.59947 of **B**. These are also the eigenvalues of **A**.

6.4.4 Householder's method

This method is applicable to a real symmetric matrix of order $n \times n$. It is more economic and efficient than the Givens method. Here also a sequence of orthogonal (similarity) transformations is used on **A** to get a tri-diagonal matrix. Each transformation produces a complete row of zeros in appropriate positions, without affecting the previous rows.

Thus, $(n-2)$ Householder transformations are needed to produce the tri-diagonal form.

The orthogonal transformation used in this method is of the form

$$\mathbf{S} = \mathbf{I} - 2\mathbf{V}\mathbf{V}^{\mathbf{T}} \tag{6.29}$$

where $\mathbf{V} = (s_1, s_2, \ldots, s_n)^T$ is a column vector containing n components, such that

$$\mathbf{V}^{\mathbf{T}}\mathbf{V} = s_1^2 + s_2^2 + \cdots + s_n^2 = 1. \tag{6.30}$$

The matrix $\mathbf{S}$ is symmetric and orthogonal, since

$$\mathbf{S}^{\mathbf{T}} = (\mathbf{I} - 2\mathbf{V}\mathbf{V}^{\mathbf{T}})^{\mathbf{T}} = \mathbf{I} - 2\mathbf{V}\mathbf{V}^{\mathbf{T}} = \mathbf{S}$$

and

$$\begin{aligned}\mathbf{S}^{\mathbf{T}}\mathbf{S} &= (\mathbf{I} - 2\mathbf{V}\mathbf{V}^{\mathbf{T}})(\mathbf{I} - 2\mathbf{V}\mathbf{V}^{\mathbf{T}})\\ &= \mathbf{I} - 4\mathbf{V}\mathbf{V}^{\mathbf{T}} + 4\mathbf{V}\mathbf{V}^{\mathbf{T}}\mathbf{V}\mathbf{V}^{\mathbf{T}}\\ &= \mathbf{I} - 4\mathbf{V}\mathbf{V}^{\mathbf{T}} + 4\mathbf{V}\mathbf{V}^{\mathbf{T}} = \mathbf{I}. \qquad \text{[using (6.30)]}.\end{aligned} \tag{6.31}$$

Thus

$$\mathbf{S}^{-1}\mathbf{A}\mathbf{S} = \mathbf{S}^{\mathbf{T}}\mathbf{A}\mathbf{S} = \mathbf{S}\mathbf{A}\mathbf{S}, \tag{6.32}$$

since $\mathbf{S}$ is orthogonal and symmetric.

Let $\mathbf{A_1} = \mathbf{A}$ and form a sequence of transformations

$$\mathbf{A_r} = \mathbf{S_r}\mathbf{A_{r-1}}\mathbf{S_r}, r = 2, 3, \ldots, n-1, \tag{6.33}$$

where $\mathbf{S_r} = \mathbf{I} - 2\mathbf{V_r}\mathbf{V_r^T}, \mathbf{V_r} = (0, 0, \ldots, 0, s_r, s_{r+1}, \ldots, s_n)^{\mathbf{T}}$ and $s_r^2 + s_{r+1}^2 + \cdots + s_n^2 = 1$.

At the first transformation, we find s_r's in such a way that the elements in the positions $(1,3), (1,4), \ldots, (1,n)$ of $\mathbf{A_2}$ become zero. Also, the elements in the corresponding positions in the first column becomes zero. Therefore, one rotation creates $n-2$ zeros in the first row and first column. In the second rotation, the elements in the positions $(2,4), (2,5), \ldots, (2,n)$ and $(4,2), (5,2), \ldots, (n,2)$ reduce to zeros.

Thus $(n-2)$ Householder transformations are required to obtain the tri-diagonal matrix $\mathbf{A_{n-1}}$. This method is illustrated using a 4×4 matrix $\mathbf{A} = [a_{ij}]_{4\times 4}$.

In the first transformation, let $\mathbf{V_2} = (0, s_2, s_3, s_4)^T$ such that

$$s_2^2 + s_3^2 + s_4^2 = 1. \tag{6.34}$$

Now,

$$\mathbf{S_2} = \mathbf{I} - 2\mathbf{V}\mathbf{V}^{\mathbf{T}} = \begin{bmatrix} 1 & 0 & 0 & 0\\ 0 & 1-2s_2^2 & -2s_2s_3 & -2s_2s_4\\ 0 & -2s_2s_3 & 1-2s_3^2 & -2s_3s_4\\ 0 & -2s_2s_4 & -2s_3s_4 & 1-2s_4^2 \end{bmatrix}. \tag{6.35}$$

The first rows of $\mathbf{A_1}$ and $\mathbf{S_2A_1}$ are same. The elements in the first row of $\mathbf{A_2} = \mathbf{S_2A_1S_2}$ are given by

$$\begin{aligned}
a'_{11} &= a_{11}\\
a'_{12} &= (1-2s_2^2)a_{12} - 2s_2s_3a_{13} - 2s_2s_4a_{14} = a_{12} - 2s_2p_1\\
a'_{13} &= -2s_2s_3a_{12} + (1-2s_3^2)a_{13} - 2s_3s_4a_{14} = a_{13} - 2s_3p_1\\
\text{and}\quad a'_{14} &= -2s_2s_4a_{12} - 2s_3s_4a_{13} + (1-2s_4^2)a_{14} = a_{14} - 2s_4p_1
\end{aligned}$$

where $p_1 = s_2a_{12} + s_3a_{13} + s_4a_{14}$.

It can be verified that

$$a'^2_{11} + a'^2_{12} + a'^2_{13} + a'^2_{14} = a_{11}^2 + a_{12}^2 + a_{13}^2 + a_{14}^2.$$

That is,

$$a'^2_{12} + a'^2_{13} + a'^2_{14} = a_{12}^2 + a_{13}^2 + a_{14}^2 = q^2 \text{ (say)} \tag{6.36}$$

and q is a known quantity.

Since the elements at the positions (1, 3) and (1, 4) of $\mathbf{A_2}$ need to be zeros, $a'_{13} = 0, a'_{14} = 0$.

Hence

$$a_{13} - 2p_1s_3 = 0 \tag{6.37}$$

$$a_{14} - 2p_1s_4 = 0 \tag{6.38}$$

$$\text{and}\quad a'_{12} = \pm q \text{ or, } a_{12} - 2p_1s_2 = \pm q. \tag{6.39}$$

Multiplying equations (6.39), (6.37) and (6.38) by s_2, s_3 and s_4 respectively and adding them to obtain the equation

$$p_1 - 2p_1(s_2^2 + s_3^2 + s_4^2) = \pm qs_2, \text{ or, } p_1 = \pm s_2q. \tag{6.40}$$

Thus from (6.39), (6.37) and (6.38) the values of s_2, s_3 and s_4 are obtained as

$$s_2^2 = \frac{1}{2}\left(1 \mp \frac{a_{12}}{q}\right), \qquad s_3 = \mp\frac{a_{13}}{2s_2q}, \qquad s_4 = \mp\frac{a_{14}}{2s_2q}. \tag{6.41}$$

It is noticed that the values of s_3 and s_4 depend on s_2, so the better accuracy can be achieved if s_2 becomes large. This can be obtained by taking suitable sign in (6.41). Choosing

$$s_2^2 = \frac{1}{2}\left(1 + \frac{a_{12} \times sign(a_{12})}{q}\right). \tag{6.42}$$

The sign of the square root is irrelevant and positive sign is taken. Hence

$$s_3 = \frac{a_{13} \times sign(a_{12})}{2q \times s_2}, \qquad s_4 = \frac{a_{14} \times sign(a_{12})}{2q \times s_2}.$$

Thus first transformation generates two zeros in the first row and first column. The second transformation is required to create zeros at the positions (2, 4) and (4, 2).

In the second transformation, let $V_3 = (0, 0, s_3, s_4)^T$ and the matrix

$$\mathbf{S_3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 - 2s_3^2 & -2s_3s_4 \\ 0 & 0 & -2s_3s_4 & 1 - 2s_4^2 \end{bmatrix}. \qquad (6.43)$$

The values of s_3 and s_4 are to be computed using the previous technique. The new matrix $\mathbf{A_3} = \mathbf{S_3A_2S_3}$ is obtained. The zeros in first row and first column remain unchanged while computing $\mathbf{A_3}$. Thus, $\mathbf{A_3}$ becomes to a tri-diagonal form in this case. The application of this method for a general $n \times n$ matrix is obvious. The elements of the vector $\mathbf{V_k} = (0, \cdots, 0, s_k, s_{k+1}, \cdots, s_n)$ at the kth transformation are given by

$$s_k^2 = \frac{1}{2}\left(1 + \frac{a_{kr} \times sign(a_{kr})}{q}\right), \qquad r = k+1, \text{where } q = \sqrt{\sum_{i=k+1}^{n} a_{ki}^2}$$

$$s_i = \frac{a_{ki} \times sign(a_{kr})}{2qs_k}, \qquad i = k+1, \ldots, n.$$

Since the tri-diagonal matrix $\mathbf{A_{n-1}}$ is similar to the original matrix $\mathbf{A}$, they have identical eigenvalues. The eigenvalues of $\mathbf{A_{n-1}}$ are computed in the same way as in the Givens method. Once the eigenvalues become available, the eigenvectors are obtained by solving the homogeneous system of equations $(\mathbf{A} - \lambda\mathbf{I})\mathbf{X} = \mathbf{0}$.

Example 6.4.5 Use the Householder method to reduce the matrix $\mathbf{A} = \begin{bmatrix} 2 & -1 & -1 & 1 \\ -1 & 4 & 1 & -1 \\ -1 & 1 & 3 & -1 \\ 1 & -1 & -1 & 2 \end{bmatrix}$ into the tri-diagonal form.

Solution. First rotation.

Let $\mathbf{V_2} = (0, s_2, s_3, s_4)^T$, $q = \sqrt{a_{12}^2 + a_{13}^2 + a_{14}^2} = \sqrt{3}$,

$$s_2^2 = \frac{1}{2}\left(1 + \frac{(-1)(-1)}{\sqrt{3}}\right) = 0.78868, \qquad s_2 = 0.88807,$$

$$s_3 = \frac{(-1)(-1)}{2\sqrt{3} \times 0.88807} = 0.32506, \qquad s_4 = \frac{1 \times (-1)}{2\sqrt{3} \times 0.88807} = -0.32506.$$

$\mathbf{V_2} = (0, 0.88807, 0.32506, -0.32506)^T$.

$$\begin{aligned}
\mathbf{S_2} &= \mathbf{I} - 2\mathbf{V_2}\mathbf{V_2^T} \\
&= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.57734 & -0.57735 & 0.57735 \\ 0 & -0.57735 & 0.78867 & 0.21133 \\ 0 & 0.57735 & 0.21133 & 0.78867 \end{bmatrix}. \\
\mathbf{A_2} &= \mathbf{S_2}\mathbf{A_1}\mathbf{S_2} = \begin{bmatrix} 2 & 1.73204 & 0 & 0 \\ 1.73204 & 5.0 & 0.21132 & -0.78867 \\ 0 & 0.21132 & 2.28867 & -0.5 \\ 0 & -0.78867 & -0.5 & 1.71133 \end{bmatrix}.
\end{aligned}$$

Second transformation.

$\mathbf{V_3} = (0, 0, s_3, s_4)^T$. $q = \sqrt{a_{23}^2 + a_{24}^2} = 0.81649$,

$$s_3^2 = \frac{1}{2}\left(1 + \frac{a_{23} \times sign(a_{23})}{q}\right) = 0.62941, \qquad s_3 = 0.79335,$$

$$s_4 = \frac{a_{24} \times sign(a_{23})}{2qs_3} = -0.60876.$$

$\mathbf{V_3} = (0, 0, 0.79335, -0.60876)^T$.

$$\begin{aligned}
\mathbf{S_3} &= \mathbf{I} - 2\mathbf{V_3}\mathbf{V_3^T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.25881 & 0.96592 \\ 0 & 0 & 0.96592 & 0.25882 \end{bmatrix}. \\
\mathbf{A_3} &= \mathbf{S_3}\mathbf{A_2}\mathbf{S_3} = \begin{bmatrix} 2 & 1.73204 & 0 & 0 \\ 1.73204 & 5.0 & -0.81648 & 0 \\ 0 & -0.81648 & 2 & -0.57731 \\ 0 & 0 & -0.57731 & 2 \end{bmatrix}.
\end{aligned}$$

This is the required tri-diagonal matrix similar to $\mathbf{A}$.

Algorithm 6.4 (Householder method). This method converts a real symmetric matrix $\mathbf{A}$ of order $n \times n$ into a real symmetric tri-diagonal form.

```
Algorithm Householder
 Step 1. Read the symmetric matrix $\mathbf{A} = [a_{ij}], i, j = 1, 2, \ldots, n$.
 Step 2. Set $k = 1, r = 2$.
 Step 3. //Compute the vector $\mathbf{V} = (v_1, v_2, \cdots, v_n)^T$//
         Step 3.1.   Compute $q = \sqrt{\sum_{i=k+1}^{n} a_{ki}^2}$.
         Step 3.2.   Set $v_i = 0$ for $i = 1, 2, \ldots, r - 1$.
```

```
        Step 3.3.   Compute $v_r^2 = \frac{1}{2}\left(1 + \frac{a_{kr} \times sign(a_{kr})}{q}\right)$.
        Step 3.4.   Compute $v_i = \frac{a_{ki} \times sign(a_{kr})}{2qv_r}$
                    for $i = r+1, \ldots, n$.
 Step 4. Compute the transformation matrix $\mathbf{S} = \mathbf{I} - 2\mathbf{V} * \mathbf{V^T}$.
 Step 5. Compute $\mathbf{A} = \mathbf{S} * \mathbf{A} * \mathbf{S}$.
 Step 6. Set $k = k + 1$, $r = r + 1$.
 Step 7. Repeat steps 3 to 6 until $k \le n - 2$.
end Householder
```

Program 6.4

```
/* Program Householder method
   This program reduces the given real symmetric matrix
   into a real symmetric tri-diagonal matrix. Assume that
   the given matrix is real symmetric. */
#include<stdio.h>
#include<math.h>
void main()
{
 int n,i,j,r=2,k,l,sign;
 float a[10][10],v[10],s[10][10],temp[10][10],q;
 printf("Enter the size of the matrix ");
 scanf("%d",&n);
 printf("Enter the elements row wise ");
 for(i=1;i<=n;i++) for(j=1;j<=n;j++) scanf("%f",&a[i][j]);
 printf("The given matrix is\n");
 for(i=1;i<=n;i++) /* printing of A */
   {
     for(j=1;j<=n;j++) printf("%8.5f ",a[i][j]); printf("\n");
   }
   for(k=1;k<=n-2;k++)
      {
       q=0;
       for(i=k+1;i<=n;i++) q+=a[k][i]*a[k][i];
       q=sqrt(q);
       for(i=1;i<=r-1;i++) v[i]=0;
       sign=1; if(a[k][r]<0) sign=-1;
       v[r]=sqrt(0.5*(1+a[k][r]*sign/q));
       for(i=r+1;i<=n;i++) v[i]=a[k][i]*sign/(2*q*v[r]);
       /* construction of S */
       for(i=1;i<=n;i++) for(j=1;j<=n;j++) s[i][j]=-2*v[i]*v[j];
```

```
        for(i=1;i<=n;i++) s[i][i]=1+s[i][i];
        for(i=1;i<=n;i++) for(j=1;j<=n;j++)
           {
             temp[i][j]=0;
             for(l=1;l<=n;l++) temp[i][j]+=s[i][l]*a[l][j];
           }
        for(i=1;i<=n;i++) for(j=1;j<=n;j++)
           {
             a[i][j]=0;
             for(l=1;l<=n;l++) a[i][j]+=temp[i][l]*s[l][j];
           }
        r++;
      } /* end of loop k */
 printf("The reduced symmetric tri-diagonal matrix is\n");
 for(i=1;i<=n;i++)
    {
      for(j=1;j<=n;j++) printf("%8.5f ",a[i][j]);
      printf("\n");
    }
}/* main */
```

A sample of input/output:

```
Enter the size of the matrix 5
Enter the elements row wise
 1 -1 -2  1  1
-1  0  1  3  2
-2  1  3  1  1
 1  3  1  4  0
 1  2  1  0  5
The given matrix is
 1.00000 -1.00000 -2.00000  1.00000  1.00000
-1.00000  0.00000  1.00000  3.00000  2.00000
-2.00000  1.00000  3.00000  1.00000  1.00000
 1.00000  3.00000  1.00000  4.00000  0.00000
 1.00000  2.00000  1.00000  0.00000  5.00000

The reduced symmetric tri-diagonal matrix is
 1.00000  2.64575 -0.00000  0.00000 -0.00000
 2.64575  1.00000  2.03540 -0.00000  0.00000
```

```
-0.00000  2.03540 -0.58621  0.94489  0.00000
 0.00000  0.00000  0.94489  5.44237 -1.26864
-0.00000  0.00000  0.00000 -1.26864  6.14384
```

6.5 Exercise

1. Compare Jacobi, Givens and Householder methods to find the eigenvalues of a real symmetric matrix.

2. If $\mathbf{X}$ is any vector and $\mathbf{P} = \mathbf{I} - 2\mathbf{X}\mathbf{X}^{\mathbf{T}}$, show that $\mathbf{P}$ is symmetric. What additional condition is necessary in order that $\mathbf{P}$ is orthogonal ?

3. Use the Leverrier-Faddeev method to find the characteristic equations of the following matrices.
(a) $\begin{bmatrix} 2 & 5 & 7 \\ 6 & 3 & 4 \\ 5 & -2 & -3 \end{bmatrix}$, (b) $\begin{bmatrix} 2 & -1 & 3 & -4 \\ 3 & -2 & 4 & 1 \\ 5 & -3 & -2 & 2 \\ 3 & -3 & -1 & 1 \end{bmatrix}$, (c) $\begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$.

4. Use the Leverrier-Faddeev method to find the eigenvalues and eigenvectors of the matrices
(a) $\begin{bmatrix} 2 & 3 \\ -1 & 2 \end{bmatrix}$, (b) $\begin{bmatrix} 5 & 6 & -3 \\ -1 & 0 & 1 \\ 1 & 2 & -1 \end{bmatrix}$, (c) $\begin{bmatrix} 1 & 2 & -3 \\ 3 & -1 & 2 \\ 1 & 0 & -1 \end{bmatrix}$, (d) $\begin{bmatrix} 1 & -2 & 1 & -2 \\ 2 & -1 & 2 & -1 \\ 1 & 1 & -2 & -2 \\ -2 & -2 & 1 & 1 \end{bmatrix}$.

5. Find all eigenvalues of the following matrices using Rutishauser method.
(a) $\begin{bmatrix} 4 & 5 \\ -1 & 1 \end{bmatrix}$, (b) $\begin{bmatrix} 3 & 2 & -3 \\ 0 & 1 & 1 \\ 1 & -2 & 1 \end{bmatrix}$.

6. Use power method to find the largest and the least (in magnitude) eigenvalues of the following matrices.
(a) $\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$, (b) $\begin{bmatrix} 4 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix}$, (c) $\begin{bmatrix} 2 & -1 & 2 \\ 5 & -3 & 3 \\ -1 & 0 & -2 \end{bmatrix}$, (d) $\begin{bmatrix} 3 & 1 & 0 \\ 1 & 2 & 2 \\ 0 & 1 & 1 \end{bmatrix}$.

7. Use Jacobi's method to find the eigenvalues of the following matrices.
(a) $\begin{bmatrix} 3 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 3 \end{bmatrix}$, (b) $\begin{bmatrix} -2 & -2 & 6 \\ -2 & 5 & 4 \\ 6 & 4 & 1 \end{bmatrix}$, (c) $\begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}$.

8. Use Givens method to find the eigenvalues of the following symmetric matrices.
(a) $\begin{bmatrix} 3 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 3 \end{bmatrix}$, (b) $\begin{bmatrix} 2 & 2 & 6 \\ 2 & 5 & 4 \\ 6 & 4 & 1 \end{bmatrix}$, (c) $\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & 3 \\ -1 & 3 & 1 \end{bmatrix}$.

9. Use Householder method to convert the above matrices to tri-diagonal form.

10. Find the eigenvalues of the following matrices using Householder method.
(a) $\begin{bmatrix} 4 & 3 & 2 \\ 3 & 4 & 3 \\ 2 & 3 & 4 \end{bmatrix}$, (b) $\begin{bmatrix} 5 & 4 & 3 \\ 4 & 5 & 4 \\ 3 & 4 & 5 \end{bmatrix}$.

11. Find the eigenvalues of the following tri-diagonal matrices.
(a) $\begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 2 \end{bmatrix}$, (b) $\begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix}$, (c) $\begin{bmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{bmatrix}$.

12. Find the spectral radii of the following matrices.
(a) $\begin{bmatrix} -2 & 1 & 1 \\ -6 & 1 & 3 \\ -12 & -2 & 8 \end{bmatrix}$, (b) $\begin{bmatrix} -3 & -7 & -2 \\ 12 & 20 & 6 \\ -20 & -31 & -9 \end{bmatrix}$.

13. Transform the symmetric matrix $\mathbf{A} = \begin{bmatrix} 1 & \sqrt{2} & \sqrt{2} & 2 \\ \sqrt{2} & -\sqrt{2} & -1 & \sqrt{2} \\ \sqrt{2} & -1 & \sqrt{2} & \sqrt{2} \\ 2 & \sqrt{2} & \sqrt{2} & -3 \end{bmatrix}$ to a tri-diagonal form, using Givens method, by a sequence of orthogonal transformations. Use exact arithmetic.

Chapter 7

Differentiation and Integration

7.1 Differentiation

Numerical differentiation is a method to find the derivatives of a function at some values of independent variable x, when the function $f(x)$ is not known explicitly, but is known only for a set of arguments.

Like interpolation, a number of formulae for differentiation are derived. The choice of formula depends on the point at which the derivative is to be determined. So, to find the derivative at a point at the beginning of the table, the formula based on Newton's forward interpolation is used, but, at a point which is at the end of the table, the formula based on Newton's backward interpolation is used. If the given values of x_i are not equispaced then the formula based on Lagrange's interpolation is appropriate.

7.1.1 Error in Numerical Differentiation

The error in polynomial interpolation is

$$E(x) = (x - x_0)(x - x_1) \cdots (x - x_n) \frac{f^{n+1}(\xi)}{(n+1)!} = w(x) \frac{f^{n+1}(\xi)}{(n+1)!}$$

where $\min\{x, x_0, \ldots, x_n\} < \xi < \max\{x, x_0, \ldots, x_n\}$ and $w(x) = (x - x_0)(x - x_1) \cdots (x - x_n)$. Obviously, $\xi = \xi(x)$ is an unknown function of x.

Therefore,

$$E'(x) = w'(x) \frac{f^{n+1}(\xi)}{(n+1)!} + w(x) \frac{f^{n+2}(\xi)}{(n+1)!} \xi'(x). \tag{7.1}$$

The bound of the second term is unknown due to the presence of the unknown quantity $\xi'(x)$.

But, at $x = x_i$, $w(x) = 0$. Thus,

$$E'(x_i) = w'(x_i)\frac{f^{n+1}(\xi_i)}{(n+1)!}, \tag{7.2}$$

where $\min\{x, x_0, \ldots, x_n\} < \xi_i < \max\{x, x_0, \ldots, x_n\}$. The error can also be expressed in terms of divided difference.

Let $E(x) = w(x)f[x, x_0, x_1, \ldots, x_n]$ where $f[x, x_0, x_1, \ldots, x_n] = \dfrac{f^{n+1}(\xi)}{(n+1)!}$.

Then $E'(x) = w'(x)f[x, x_0, x_1, \ldots, x_n] + w(x)f[x, x, x_0, x_1, \ldots, x_n]$.

Now, this expression is differentiated $(k-1)$ times by Leibnitz's theorem.

$$\begin{aligned}
E^k(x) &= \sum_{i=0}^{k} {}^kC_r\ w^{(i)}(x)\frac{d^{k-1}}{dx^{k-i}}(f[x, x_0, \ldots, x_n]) \\
&= \sum_{i=0}^{k} {}^kC_r\ w^{(i)}(x)(k-i)!f[\overbrace{x, x, \ldots, x}^{k-i+1}, x_0, \ldots, x_n] \\
&= \sum_{i=0}^{k} \frac{k!}{i!}w^{(i)}(x)(k-i)!f[\overbrace{x, x, \ldots, x}^{k-i+1}, x_0, \ldots, x_n],
\end{aligned} \tag{7.3}$$

where $w^{(i)}(x)$ denotes the ith derivative of $w(x)$.

Note 7.1.1 If a function $f(x)$ is well-approximated by a polynomial $\phi(x)$ of degree at most n, the slope $f'(x)$ can also be approximated by the slope $\phi'(x)$. But, the error committed in $\phi'(x)$ is more than the error committed in $\phi(x)$.

7.2 Differentiation Based on Newton's Forward Interpolation Polynomial

Suppose the function $y = f(x)$ is known at $(n+1)$ equispaced points $x_0, x_1, \ldots, x_n$ and they are $y_0, y_1, \ldots, y_n$ respectively, i.e., $y_i = f(x_i), i = 0, 1, \ldots, n$. Let $x_i = x_0 + ih$ and $u = \dfrac{x - x_0}{h}$, h is the spacing.

The Newton's forward interpolation formula is

$$\begin{aligned}
\phi(x) &= y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \cdots + \frac{u(u-1)\cdots(u-\overline{n-1})}{n!}\Delta^n y_0 \\
&= y_0 + u\Delta y_0 + \frac{u^2-u}{2!}\Delta^2 y_0 + \frac{u^3-3u^2+2u}{3!}\Delta^3 y_0 + \frac{u^4-6u^3+11u^2-6u}{4!}\Delta^4 y_0 \\
&\quad + \frac{u^5 - 10u^4 + 35u^3 - 50u^2 + 24u}{5!}\Delta^5 y_0 + \cdots
\end{aligned} \tag{7.4}$$

with error

$$E(x) = \frac{u(u-1)\cdots(u-n)}{(n+1)!}h^{n+1}f^{(n+1)}(\xi),$$

where $\min\{x, x_0, \cdots, x_n\} < \xi < \max\{x, x_0, \ldots, x_n\}$.

Differentiating (7.4) successively with respect to x, we obtain

$$\phi'(x) = \frac{1}{h}\Big[\Delta y_0 + \frac{2u-1}{2!}\Delta^2 y_0 + \frac{3u^2-6u+2}{3!}\Delta^3 y_0 + \frac{4u^3-18u^2+22u-6}{4!}\Delta^4 y_0 + \frac{5u^4-40u^3+105u^2-100u+24}{5!}\Delta^5 y_0 + \cdots\Big] \quad (7.5)$$

$\left(\text{as } \frac{du}{dx} = \frac{1}{h}\right)$

$$\phi''(x) = \frac{1}{h^2}\Big[\Delta^2 y_0 + \frac{6u-6}{3!}\Delta^3 y_0 + \frac{12u^2-36u+22}{4!}\Delta^4 y_0 + \frac{20u^3-120u^2+210u-100}{5!}\Delta^5 y_0 + \cdots\Big] \quad (7.6)$$

$$\phi'''(x) = \frac{1}{h^3}\Big[\Delta^3 y_0 + \frac{24u-36}{4!}\Delta^4 y_0 + \frac{60u^2-240u+210}{5!}\Delta^5 y_0 + \cdots\Big] \quad (7.7)$$

and so on.

It may be noted that Δy_0, $\Delta^2 y_0$, $\Delta^3 y_0, \cdots$ are constants.

The above equations give the approximate derivative of $f(x)$ at arbitrary point x ($= x_0 + uh$).

When $x = x_0$, $u = 0$, the above formulae become

$$\phi'(x_0) = \frac{1}{h}\Big[\Delta y_0 - \frac{1}{2}\Delta^2 y_0 + \frac{1}{3}\Delta^3 y_0 - \frac{1}{4}\Delta^4 y_0 + \frac{1}{5}\Delta^5 y_0 - \cdots\Big] \quad (7.8)$$

$$\phi''(x_0) = \frac{1}{h^2}\Big[\Delta^2 y_0 - \Delta^3 y_0 + \frac{11}{12}\Delta^4 y_0 - \frac{5}{6}\Delta^5 y_0 + \cdots\Big] \quad (7.9)$$

$$\phi'''(x_0) = \frac{1}{h^3}\Big[\Delta^3 y_0 - \frac{3}{2}\Delta^4 y_0 + \frac{7}{4}\Delta^5 y_0 - \cdots\Big] \quad (7.10)$$

and so on.

Error in differentiation formula based on Newton's forward interpolation polynomial

The error in Newton's forward interpolation formula is

$$E(x) = u(u-1)\cdots(u-n)h^{n+1}\frac{f^{n+1}(\xi)}{(n+1)!}.$$

Then

$$E'(x) = h^{n+1}\frac{f^{n+1}(\xi)}{(n+1)!}\frac{d}{du}\big[u(u-1)\cdots(u-n)\big]\frac{1}{h} + \frac{u(u-1)\cdots(u-n)}{(n+1)!}h^{n+1}\frac{d}{dx}[f^{n+1}(\xi)]$$
$$= h^n\frac{f^{n+1}(\xi)}{(n+1)!}\frac{d}{du}[u(u-1)\cdots(u-n)] + \frac{u(u-1)\cdots(u-n)}{(n+1)!}h^{n+1}f^{n+2}(\xi_1), \quad (7.11)$$

where $\min\{x, x_0, x_1, \ldots, x_n\} < \xi, \xi_1 < \max\{x, x_0, x_1, \ldots, x_n\}$.

Error at the point $x = x_0$, i.e., $u = 0$ is

$$E'(x_0) = h^n\frac{f^{n+1}(\xi)}{(n+1)!}\frac{d}{du}[u(u-1)\cdots(u-n)]_{u=0} = \frac{h^n(-1)^n\, n!\, f^{n+1}(\xi)}{(n+1)!}$$
$$\left[\text{as } \frac{d}{du}[u(u-1)\cdots(u-n)]_{u=0} = (-1)^n n!\right]$$
$$= \frac{(-1)^n h^n f^{n+1}(\xi)}{n+1}, \quad (7.12)$$

where $\min\{x, x_0, x_1, \ldots, x_n\} < \xi < \max\{x, x_0, x_1, \ldots, x_n\}$.

Example 7.2.1 From the following table find the value of $\frac{dy}{dx}$ and $\frac{d^2y}{dx^2}$ at the point $x = 1.5$.

x :	1.5	2.0	2.5	3.0	3.5	4.0
y :	3.375	7.000	13.625	24.000	38.875	59.000

Solution. The forward difference table is

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
1.5	3.375			
		3.625		
2.0	7.000		3.000	
		6.625		0.750
2.5	13.625		3.750	
		10.375		0.750
3.0	24.000		4.500	
		14.875		0.750
3.5	38.875		5.250	
		20.125		
4.0	59.000			

Here $x_0 = 1.5$ and $h = 0.5$. Then $u = 0$ and hence

$$\begin{aligned}
y'(1.5) &= \frac{1}{h}\Big(\Delta y_0 - \frac{1}{2}\Delta^2 y_0 + \frac{1}{3}\Delta^3 y_0 - \cdots\Big)\\
&= \frac{1}{0.5}\Big(3.625 - \frac{1}{2}\times 3.000 + \frac{1}{3}\times 0.750\Big) = 4.750.\\
y''(1.5) &= \frac{1}{h^2}(\Delta^2 y_0 - \Delta^3 y_0 + \cdots) = \frac{1}{(0.5)^2}(3.000 - 0.750) = 9.000.
\end{aligned}$$

7.3 Differentiation Based on Newton's Backward Interpolation Polynomial

Suppose the function $y = f(x)$ is known at $(n+1)$ points $x_0, x_1, \ldots, x_n$, i.e., $y_i = f(x_i)$, $i = 0, 1, 2, \ldots, n$ are known. Let $x_i = x_0 + ih$, $i = 0, 1, 2, \ldots, n$ and $v = \dfrac{x - x_n}{h}$. Then Newton's backward interpolation formula is

$$\begin{aligned}
\phi(x) &= y_n + v\nabla y_n + \frac{v(v+1)}{2!}\nabla^2 y_n + \frac{v(v+1)(v+2)}{3!}\nabla^3 y_n\\
&\quad + \frac{v(v+1)(v+2)(v+3)}{4!}\nabla^4 y_n + \frac{v(v+1)(v+2)(v+3)(v+4)}{5!}\nabla^5 y_n + \cdots
\end{aligned}$$

The above equation is differentiated with respect to x successively.

$$\begin{aligned}
\phi'(x) &= \frac{1}{h}\Big[\nabla y_n + \frac{2v+1}{2!}\nabla^2 y_n + \frac{3v^2+6v+2}{3!}\nabla^3 y_n + \frac{4v^3+18v^2+22v+6}{4!}\nabla^4 y_n\\
&\quad + \frac{5v^4+40v^3+105v^2+100v+24}{5!}\nabla^5 y_n + \cdots\Big] \qquad (7.13)\\
\phi''(x) &= \frac{1}{h^2}\Big[\nabla^2 y_n + \frac{6v+6}{3!}\nabla^3 y_n + \frac{12v^2+36v+22}{4!}\nabla^4 y_n\\
&\quad + \frac{20v^3+120v^2+210v+100}{5!}\nabla^5 y_n + \cdots\Big] \qquad (7.14)\\
\phi'''(x) &= \frac{1}{h^3}\Big[\nabla^3 y_n + \frac{24v+36}{4!}\nabla^4 y_n + \frac{60v^3+240v+210}{5!}\nabla^5 y_n + \cdots\Big] \qquad (7.15)
\end{aligned}$$

and so on.

The above formulae are used to determine the approximate differentiation of first, second and third, etc. order at any point x where $x = x_n + vh$.

If $x = x_n$ then $v = 0$. In this case, the above formulae become

$$\phi'(x_n) = \frac{1}{h}\Big[\nabla y_n + \frac{1}{2}\nabla^2 y_n + \frac{1}{3}\nabla^3 y_n + \frac{1}{4}\nabla^4 y_n + \frac{1}{5}\nabla^5 y_n + \cdots\Big] \tag{7.16}$$

$$\phi''(x_n) = \frac{1}{h^2}\Big[\nabla^2 y_n + \nabla^3 y_n + \frac{11}{12}\nabla^4 y_n + \frac{5}{6}\nabla^5 y_n + \cdots\Big] \tag{7.17}$$

$$\phi'''(x_n) = \frac{1}{h^3}\Big[\nabla^3 y_n + \frac{3}{2}\nabla^4 y_n + \frac{7}{4}\nabla^5 y_n + \cdots\Big] \tag{7.18}$$

Error in differentiation formula based on Newton's backward interpolation polynomial

The error in Newton's backward interpolation formula is

$$E(x) = v(v+1)(v+2)\cdots(v+n)h^{n+1}\frac{f^{n+1}(\xi)}{(n+1)!},$$

where $v = \dfrac{x - x_n}{h}$ and $\min\{x, x_0, x_1, \dots, x_n\} < \xi < \max\{x, x_0, x_1, \dots, x_n\}$.

Then

$$\begin{aligned} E'(x) &= h^n \frac{d}{dv}[v(v+1)(v+2)\cdots(v+n)]\frac{f^{n+1}(\xi)}{(n+1)!} \\ &\quad + h^{n+1}\frac{v(v+1)(v+2)\cdots(v+n)}{(n+1)!}f^{n+2}(\xi_1), \end{aligned}$$

where $\min\{x, x_0, x_1, \dots, x_n\} < \xi, \xi_1 < \max\{x, x_0, x_1, \dots, x_n\}$.

Error at $x = x_n$, i.e., at $v = 0$ is

$$\begin{aligned} E'(x_n) &= h^n \frac{d}{dv}[v(v+1)(v+2)\cdots(v+n)]\frac{f^{n+1}(\xi)}{(n+1)!} \\ &= h^n \frac{n!}{(n+1)!}f^{n+1}(\xi) \qquad \Big[\text{as } \frac{d}{dv}[v(v+1)\cdots(v+n)]_{v=0} = n!\Big] \\ &= \frac{h^n f^{n+1}(\xi)}{n+1}. \end{aligned} \tag{7.19}$$

Example 7.3.1 A particle is moving along a straight line. The displacement x at some time instances t are given below:

t :	0	1	2	3	4
x :	5	8	12	17	26

Find the velocity and acceleration of the particle at $t = 4$.

Solution. The backward difference table is

t	x	∇x	$\nabla^2 x$	$\nabla^3 x$	$\nabla^4 x$
0	5				
1	8	3			
2	12	4	1		
3	17	5	1	0	
4	26	9	4	3	3

The velocity is

$$\begin{aligned}\frac{dx}{dt} &= \frac{1}{h}\Big[\nabla x_n + \frac{1}{2}\nabla^2 x_n + \frac{1}{3}\nabla^3 x_n + \frac{1}{4}\nabla^4 x_n + \cdots\Big]\\ &= \frac{1}{1}\Big[9 + \frac{1}{2}\times 4 + \frac{1}{3}\times 3 + \frac{1}{4}\times 3\Big]\\ &= 12.75.\end{aligned}$$

The acceleration is

$$\begin{aligned}\frac{d^2x}{dt^2} &= \frac{1}{h^2}\Big[\nabla^2 x_n + \nabla^3 x_n + \frac{11}{12}\nabla^4 x_n + \cdots\Big]\\ &= \frac{1}{1^2}\Big[4 + 3 + \frac{11}{12}\times 3\Big] = 9.75.\end{aligned}$$

Example 7.3.2 A slider in a machine moves along a fixed straight rod. Its distance x cm along the rod are given in the following table for various values of the time t (in second).

t (sec)	:	1.0	1.1	1.2	1.3	1.4	1.5
x (cm)	:	16.40	19.01	21.96	25.29	29.03	33.21

Find the velocity and the acceleration of the slider at time $t = 1.5$.

Solution. The backward difference table is

t	x	∇x	$\nabla^2 x$	$\nabla^3 x$	$\nabla^4 x$	$\nabla^5 x$
1.0	16.40					
1.1	19.01	2.61				
1.2	21.96	2.95	0.34			
1.3	25.29	3.33	0.38	0.04		
1.4	29.03	3.74	0.41	0.03	–0.01	
1.5	33.21	4.18	0.44	0.03	0.00	0.01

Here $h = 0.1$.

$$\begin{aligned}
\frac{dx}{dt} &= \frac{1}{h}\left[\nabla + \frac{1}{2}\nabla^2 + \frac{1}{3}\nabla^3 + \frac{1}{4}\nabla^4 + \frac{1}{5}\nabla^5 + \cdots\right]x_n \\
&= \frac{1}{0.1}\left[4.18 + \frac{1}{2}\times 0.44 + \frac{1}{3}\times 0.03 + \frac{1}{4}\times 0.00 + \frac{1}{5}\times 0.01\right] \\
&= 44.12. \\
\frac{d^2x}{dt^2} &= \frac{1}{h^2}\left[\nabla^2 + \nabla^3 + \frac{11}{12}\nabla^5 + \cdots\right]x_n \\
&= \frac{1}{(0.1)^2}\left[0.44 + 0.03 + \frac{11}{12}\times 0.00 + \frac{5}{6}\times 0.01\right] \\
&= 47.83.
\end{aligned}$$

Hence velocity and acceleration are respectively 44.12 cm/sec and 47.83 cm/sec^2.

7.4 Differentiation Based on Stirling's Interpolation Formula

Suppose $y_{\pm i} = f(x_{\pm i}), i = 0, 1, \ldots, n$ are given for $2n+1$ equispaced points x_0, $x_{\pm 1}$, $x_{\pm 2}$, …, $x_{\pm n}$, where $x_{\pm i} = x_0 \pm ih, i = 0, 1, \ldots, n$.

The Stirling's interpolation polynomial is

$$\begin{aligned}
\phi(x) &= y_0 + \frac{u}{1!}\cdot\frac{\Delta y_{-1} + \Delta y_0}{2} + \frac{u^2}{2!}\cdot\Delta^2 y_{-1} + \frac{u^3 - u}{3!}\cdot\frac{\Delta^3 y_{-2} + \Delta^3 y_{-1}}{2} \\
&\quad + \frac{u^4 - u^2}{4!}\cdot\Delta^4 y_{-2} + \frac{u^5 - 5u^3 + 4u}{5!}\cdot\frac{\Delta^5 y_{-3} + \Delta^5 y_{-2}}{2} + \cdots \\
&\quad \text{where } u = \frac{x - x_0}{h}.
\end{aligned} \tag{7.20}$$

This equation is differentiated with respect to x successively.

$$\begin{aligned}
\phi'(x) &= \frac{1}{h}\left[\frac{\Delta y_{-1} + \Delta y_0}{2} + u\Delta^2 y_{-1} + \frac{3u^2 - 1}{6}\cdot\frac{\Delta^3 y_{-2} + \Delta^3 y_{-1}}{2}\right. \\
&\quad \left. + \frac{2u^3 - u}{12}\Delta^4 y_{-2} + \frac{5u^4 - 15u^2 + 4}{120}\cdot\frac{\Delta^5 y_{-3} + \Delta^5 y_{-2}}{2} + \cdots\right].
\end{aligned} \tag{7.21}$$

$$\begin{aligned}
\phi''(x) &= \frac{1}{h^2}\left[\Delta^2 y_{-1} + u\frac{\Delta^3 y_{-2} + \Delta^3 y_{-1}}{2} + \frac{6u^2 - 1}{12}\cdot\Delta^4 y_{-2}\right. \\
&\quad \left. + \frac{2u^3 - 3u}{12}\cdot\frac{\Delta^5 y_{-3} + \Delta^5 y_{-2}}{2} + \cdots\right].
\end{aligned} \tag{7.22}$$

At $x = x_0$, $u = 0$. Then

$$\phi'(x_0) = \frac{1}{h}\left[\frac{\Delta y_0 + \Delta y_{-1}}{2} - \frac{1}{6}\frac{\Delta^3 y_{-1} + \Delta^3 y_{-2}}{2} + \frac{1}{30}\frac{\Delta^5 y_{-2} + \Delta^5 y_{-3}}{2} + \cdots\right]. \tag{7.23}$$

and $$\phi''(x_0) = \frac{1}{h^2}\left[\Delta^2 y_{-1} - \frac{1}{12}\Delta^4 y_{-2} + \cdots\right]. \tag{7.24}$$

Error in differentiation formula based on Stirling's interpolation polynomial

The error of Stirling's interpolation formula is

$$E(x) = u(u^2-1^2)(u^2-2^2)\cdots(u^2-n^2)h^{2n+1}\frac{f^{2n+1}(\xi)}{(2n+1)!},$$

where $\min\{x, x_{-n}, \ldots, x_0, \ldots, x_n\} < \xi < \max\{x, x_{-n}, \ldots, x_0, \ldots, x_n\}$.

Then

$$\begin{aligned}E'(x) &= \frac{d}{du}[u(u^2-1^2)(u^2-2^2)\cdots(u^2-n^2)]\frac{du}{dx}h^{2n+1}\frac{f^{2n+1}(\xi)}{(2n+1)!}\\ &\quad+h^{2n+1}[u(u^2-1^2)(u^2-2^2)\cdots(u^2-n^2)]\frac{d}{dx}\left[\frac{f^{2n+1}(\xi)}{(2n+1)!}\right]\\ &= h^{2n}\frac{d}{du}[u(u^2-1^2)(u^2-2^2)\cdots(u^2-n^2)]\frac{f^{2n+1}(\xi)}{(2n+1)!}\\ &\quad+h^{2n+1}[u(u^2-1^2)(u^2-2^2)\cdots(u^2-n^2)]\frac{f^{2n+2}(\xi_1)}{(2n+1)!},\end{aligned} \tag{7.25}$$

where $\min\{x, x_{-n}, \ldots, x_0, \ldots, x_n\} < \xi, \xi_1 < \max\{x, x_{-n}, \ldots, x_0, \ldots, x_n\}$.

At $x = x_0$, $u = 0$. Then $\frac{d}{du}[u(u^2-1^2)(u^2-2^2)\cdots(u^2-n^2)] = (-1)^n(n!)^2$.

In this case,

$$E'(x_0) = \frac{(-1)^n(n!)^2}{(2n+1)!}h^{2n}f^{2n+1}(\xi). \tag{7.26}$$

Example 7.4.1 Compute the values of (i) $f'(3)$, (ii) $f''(3)$, (iii) $f'(3.1)$, (iv) $f''(3.1)$ using the following table.

x	:	1	2	3	4	5
$f(x)$	:	0.0000	1.3863	3.2958	5.5452	8.0472

Solution. The central difference table is

x	$y=f(x)$	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
$x_{-2}=1$	0.0000				
		1.3863			
$x_{-1}=2$	1.3863		0.5232		
		1.9095		–0.1833	
$x_0=3$	3.2958		0.3399		0.0960
		2.2494		–0.0873	
$x_1=4$	5.5452		0.2526		
		2.5020			
$x_2=5$	8.0472				

Since $x=3$ and $x=3.1$ are the middle of the table, so the formula based on central difference may be used. Here Stirling's formula is used to find the derivatives.
(i) Here $x_0=3, h=1, u=0$.
Then

$$f'(3)=\frac{1}{h}\left[\frac{\Delta y_{-1}+\Delta y_0}{2}-\frac{\Delta^3 y_{-2}+\Delta^3 y_{-1}}{12}+\cdots\right]$$
$$=\frac{1}{1}\left[\frac{1.9095+2.2494}{2}-\frac{-0.1833-0.0873}{12}\right]=2.1020.$$

(ii) $f''(3)=\frac{1}{h^2}\left[\Delta^2 y_{-1}-\frac{1}{12}\Delta^4 y_{-2}+\cdots\right]=\frac{1}{1^2}\left[0.3399-\frac{1}{12}\times 0.0960\right]=0.3319.$
(iii) Let $x_0=3, h=1, u=\frac{3.1-3}{1}=0.1$. Then

$$f'(3.1)=\frac{1}{h}\left[\frac{\Delta y_{-1}+\Delta y_0}{2}+u\Delta^2 y_{-1}+\frac{3u^2-1}{6}\frac{\Delta^3 y_{-2}+\Delta^3 y_{-1}}{2}\right.$$
$$\left.+\frac{2u^3-u}{12}\Delta^4 y_{-2}+\cdots\right]$$
$$=\frac{1}{1}\left[\frac{1.9095+2.2494}{2}+0.1\times 0.3399+\frac{3\times(0.1)^2-1}{6}\frac{-0.1833-0.0873}{2}\right.$$
$$\left.+\frac{2\times(0.1)^3-0.1}{12}\times 0.0960\right]=2.1345.$$

(iv)
$$f''(3.1)=\frac{1}{h^2}\left[\Delta^2 y_{-1}+u\frac{\Delta^3 y_{-2}+\Delta^3 y_{-1}}{2}+\frac{6u^2-1}{12}\Delta^4 y_{-2}+\cdots\right]$$
$$=\frac{1}{1^2}\left[0.3399+0.1\times\frac{-0.1833-0.0873}{2}+\frac{6(0.1)^2-1}{12}\times 0.0960\right]$$
$$=0.31885.$$

7.5 Differentiation Based on Lagrange's Interpolation Polynomial

The Lagrange's interpolation formula is

$$\phi(x) = w(x)\sum_{i=0}^{n} \frac{y_i}{(x-x_i)w'(x_i)}, \text{ where } w(x) = (x-x_0)(x-x_1)\cdots(x-x_n).$$

Then

$$\phi'(x) = w'(x)\sum_{i=0}^{n} \frac{y_i}{(x-x_i)w'(x_i)} - w(x)\sum_{i=0}^{n} \frac{y_i}{(x-x_i)^2 w'(x_i)}. \tag{7.27}$$

$$\text{and} \quad \phi''(x) = w''(x)\sum_{i=0}^{n} \frac{y_i}{(x-x_i)w'(x_i)} - 2w'(x)\sum_{i=0}^{n} \frac{y_i}{(x-x_i)^2 w'(x_i)} + 2w(x)\sum_{i=0}^{n} \frac{y_i}{(x-x_i)^3 w'(x_i)}. \tag{7.28}$$

The value of $w'(x)$ can be determined as

$$w'(x) = \sum_{j=0}^{n}(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n).$$

The formulae (7.27) and (7.28) are valid for all x except $x = x_i, i = 0, 1, \ldots, n$.

To find the derivatives at the points $x_0, x_1, \ldots, x_n$, the Lagrange's polynomial is rearranged as

$$\phi(x) = w(x)\sum_{\substack{i=0\\i\neq j}}^{n} \frac{y_i}{(x-x_i)w'(x_i)} + \frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)}y_j.$$

Therefore,

$$\phi'(x_j) = w'(x_j)\sum_{\substack{i=0\\i\neq j}}^{n} \frac{y_i}{(x_j-x_i)w'(x_i)} + \sum_{\substack{i=0\\i\neq j}}^{n} \frac{y_j}{x_j-x_i}, \tag{7.29}$$

where
$$w'(x_j) = (x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n) = \prod_{\substack{i=0\\i\neq j}}^{n}(x_j-x_i).$$

Note that

$$\frac{d}{dx}\left[\frac{(x-x_0)(x-x_1)\cdots(x-x_{j-1})(x-x_{j+1})\cdots(x-x_n)}{(x_j-x_0)(x_j-x_1)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)}\right]_{x=x_j}$$
$$=\sum_{\substack{i=0\\i\neq j}}^{n}\frac{1}{x_j-x_i}.$$

This formula is used to find the derivative at the points $x = x_0, x_1, \ldots, x_n$.

Error in differentiation formula based on Lagrange's interpolation polynomial

The error term is

$$E(x) = w(x)\frac{f^{n+1}(\xi)}{(n+1)!},\ \text{where } w(x) = (x-x_0)(x-x_1)\cdots(x-x_n).$$

Thus

$$E'(x) = w'(x)\frac{f^{n+1}(\xi)}{(n+1)!} + w(x)\frac{f^{n+2}(\xi_1)}{(n+1)!}, \tag{7.30}$$

where $\min\{x, x_0, x_1, \ldots, x_n\} < \xi, \xi_1 < \max\{x, x_0, x_1, \ldots, x_n\}$.

At $x = x_i$, $i = 0, 1, 2, \ldots, n$,

$$E'(x_i) = w'(x_i)\frac{f^{n+1}(\xi)}{(n+1)!}. \tag{7.31}$$

Example 7.5.1 Use the differentiation formula based on Lagrange's interpolation to find the value of $f'(2)$ and $f'(2.5)$ from the following table.

x :	2	3	5	6
y :	13	34	136	229

Solution. Here $x_0 = 2, x_1 = 3, x_2 = 5, x_3 = 6$.
$w(x) = (x-x_0)(x-x_1)(x-x_2)(x-x_3) = (x-2)(x-3)(x-5)(x-6)$.
$w'(x) = (x-3)(x-5)(x-6) + (x-2)(x-5)(x-6) + (x-2)(x-3)(x-6)$
$\quad + (x-2)(x-3)(x-5)$.
By the formula (7.29),

$$f'(x) \simeq \phi'(x) = w'(x_0)\sum_{i=1}^{3}\frac{y_i}{(x_0-x_i)w'(x_i)} + \sum_{i=1}^{3}\frac{y_0}{x_0-x_i}$$
$$= w'(2)\left[\frac{y_1}{(2-3)w'(3)} + \frac{y_2}{(2-5)w'(5)} + \frac{y_3}{(2-6)w'(6)}\right]$$
$$+y_0\left[\frac{1}{2-3} + \frac{1}{2-5} + \frac{1}{2-6}\right].$$

$w'(2) = -12, w'(3) = 6, w'(5) = -6, w'(6) = 12.$
Thus

$$f'(2) \simeq -12\left[\frac{34}{(-1)\times 6} + \frac{136}{(-3)\times(-6)} + \frac{229}{(-4)\times 12}\right] + 13\left[-1 - \frac{1}{3} - \frac{1}{4}\right] = 14.$$

Also

$$\begin{aligned}
&f'(2.5)\\
&\simeq w'(2.5)\sum_{i=0}^{3}\frac{y_i}{(2.5-x_i)w'(x_i)} - w(2.5)\sum_{i=0}^{3}\frac{y_i}{(2.5-x_i)^2 w'(x_i)}\\
&= w'(2.5)\left[\frac{y_0}{(2.5-2)w'(2)} + \frac{y_1}{(2.5-3)w'(3)} + \frac{y_2}{(2.5-5)w'(5)} + \frac{y_3}{(2.5-6)w'(6)}\right]\\
&\quad -w(2.5)\left[\frac{y_0}{(2.5–2)^2 w'(2)} + \frac{y_1}{(2.5–3)^2 w'(3)} + \frac{y_2}{(2.5–5)^2 w'(5)} + \frac{y_3}{(2.5–6)^2 w'(6)}\right]
\end{aligned}$$

Now, $w'(2.5) = 1.5, w(2.5) = -2.1875.$
Therefore,

$$\begin{aligned}
f'(2.5) &\simeq 1.5\left[\frac{13}{0.5\times(-12)} + \frac{34}{(-0.5)\times 6} + \frac{136}{(-2.5)\times(-6)} + \frac{229}{(-3.5)\times 12}\right]\\
&+ 2.1875\left[\frac{13}{(0.5)^2\times(-12)} + \frac{34}{(-0.5)^2\times 6} + \frac{136}{(-2.5)^2\times(-6)} + \frac{229}{(-3.5)^2\times 12}\right]\\
&= 20.75.
\end{aligned}$$

Algorithm 7.1 (Derivative). This algorithm determines the first order derivative of a function given in tabular form $(x_i, y_i), i = 0, 1, 2, \ldots, n$, at a given point xg, xg may or may not be equal to the given nodes x_i, based on Lagrange's interpolation.

```
Algorithm Derivative_Lagrange
Read $x_i, y_i, i = 0, 1, 2, \ldots, n$.
Read $xg$; //the point at which the derivative is to be evaluated.//
Compute $w'(x_j), j = 0, 1, 2, \ldots, n$, using the function $wd(j)$.
Set $sum1 = sum2 = 0$;
Check $xg$ is equal to given nodes $x_i$.
If $xg$ is not equal to any node then
    for $i = 0$ to $n$ do
        Compute $t = y_i/((xg - x_i) * wd(i))$;
        Compute $sum1 = sum1 + t$;
        Compute $sum2 = sum2 + t/(xg - x_i)$;
    endfor;
    //compute $w'(xg)$//
    set $t = 0$;
    for $j = 0$ to $n$ do
        set $prod = 1$;
        for $i = 0$ to $n$ do
            if $(i \neq j)$ then $prod = prod * (xg - x_i)$;
        endfor;
    Compute $t = t + prod$;
    endfor;
    //compute $w(xg)$ //
    set $t1 = 1$;
    for $i = 0$ to $n$ do
        Compute $t1 = t1 * (xg - x_i)$;
    endfor;
    Compute $result = t * sum1 - t1 * sum2$;
else //$xg$ is equal to $x_j$//
    for $i = 0$ to $n$ do
        if $i \neq j$ then
            Compute $sum1 = sum1 + y_i/((x_j - x_i) * wd(i))$;
            Compute $sum2 = sum2 + 1/(x_j - x_i)$;
        endif;
    endfor;
    Compute $result = wd(j) * sum1 + y_j * sum2$;
endif;
Print 'The value of the derivative', $result$;
function $wd(j)$
    //This function determines $w'(x_j)$.//
    Set $prod = 1$;
    for $i = 0$ to $n$ do
        if $(i \neq j)$ $prod = prod * x_i$;
    endfor;
    return $prod$;
end $wd$
end Derivative_Lagrange
```

Program 7.1

```
/* Program Derivative
   Program to find the first order derivative of a function
   y=f(x) given as (xi,yi), i=0, 1, 2, ..., n, using formula
   based on Lagrange's interpolation. */
#include<stdio.h>
```

```
int n; float x[20],xg;
float wd(int);
void main()
{
 int i,j,flag=-1;
 float y[20],sum1=0,sum2=0,prod,t,t1=1,result;
 printf("Enter the value of n and the data in the form
         (x[i],y[i]) \n");
 scanf("%d",&n);
 for(i=0;i<=n;i++) scanf("%f %f",&x[i],&y[i]);
 printf("Enter the value of x at which derivative
         is required \n");
 scanf("%f",&xg);
 for(i=0;i<=n;i++) if(x[i]==xg) flag=i;
 if(flag==-1) /* xg is not equal to xi, i=0, 1, ..., n */
   {
    for(i=0;i<=n;i++)
       {
         t=y[i]/((xg-x[i])*wd(i));
         sum1+=t;
         sum2+=t/(xg-x[i]);
       }
   /* Computation of w'(xg) */
   t=0;
   for(j=0;j<=n;j++)
      {
        prod=1;
        for(i=0;i<=n;i++) if(i!=j) prod*=(xg-x[i]);
        t+=prod;
      }
  /* computation of w(xg) */
  for(i=0;i<=n;i++) t1*=(xg-x[i]);
  result=t*sum1-t1*sum2;
  } /* end of if part */
 else
  {
    j=flag;
    for(i=0;i<=n;i++)
    if(i!=j)
```

```
       {
         sum1+=y[i]/((x[j]-x[i])*wd(i));
         sum2+=1/(x[j]-x[i]);
       }
     result=wd(j)*sum1+y[j]*sum2;
  } /* end of else part */
printf("The value of derivative at x= %6.4f is
        %8.5f",xg,result);
}
 /* this function determines w'(xj) */
 float wd(int j)
  {
     int i;float prod=1;
     for(i=0;i<=n;i++) if(i!=j) prod*=(x[j]-x[i]);
     return prod;
  }
```

A sample of input/output:

```
Enter the value of n and the data in the form (x[i],y[i])
3
1 0.54030
2 -0.41615
3 -0.98999
4 -0.65364
Enter the value of x at which derivative is required
1.2
The value of derivative at x= 1.2000 is -0.99034
```

Table of derivatives

The summary of the formulae of derivatives based on finite differences.

$$f'(x_0) \simeq \frac{1}{h}\left(\Delta - \frac{1}{2}\Delta^2 + \frac{1}{3}\Delta^3 - \frac{1}{4}\Delta^4 + \frac{1}{5}\Delta^5 - \frac{1}{6}\Delta^6 + \cdots\right)y_0 \tag{7.32}$$

$$f''(x_0) \simeq \frac{1}{h^2}\left(\Delta^2 - \Delta^3 + \frac{11}{12}\Delta^4 - \frac{5}{6}\Delta^5 + \frac{137}{180}\Delta^6 - \frac{7}{10}\Delta^7 + \frac{363}{560}\Delta^8 + \cdots\right)y_0 \tag{7.33}$$

$$f'(x_n) \simeq \frac{1}{h}\left(\nabla + \frac{1}{2}\nabla^2 + \frac{1}{3}\nabla^3 + \frac{1}{4}\nabla^4 + \frac{1}{5}\nabla^5 + \frac{1}{6}\nabla^6 + \cdots\right)y_n \tag{7.34}$$

$$f''(x_n) \simeq \frac{1}{h^2}\left(\nabla^2 + \nabla^3 + \frac{11}{12}\nabla^4 + \frac{5}{6}\nabla^5 + \frac{137}{180}\nabla^6 + \frac{7}{10}\nabla^7 + \frac{363}{560}\nabla^8 + \cdots\right)y_n \tag{7.35}$$

$$f'(x_0) \simeq \frac{1}{h}\left(\frac{\Delta y_{-1}+\Delta y_0}{2} - \frac{\Delta^3 y_{-2}+\Delta^3 y_{-1}}{12}+\frac{\Delta^5 y_{-3}+\Delta^5 y_{-2}}{60}+\cdots\right)y_0 \tag{7.36}$$

$$f''(x_0) \simeq \frac{1}{h^2}\left(\Delta^2 y_{-1} - \frac{1}{12}\Delta^4 y_{-2} + \frac{1}{90}\Delta^6 y_{-3} - \cdots\right)y_0 \tag{7.37}$$

7.6 Two-point and Three-point Formulae

Only the first term of (7.32), gives a simple formula for the first order derivative

$$f'(x_i) \simeq \frac{\Delta y_i}{h} = \frac{y_{i+1} - y_i}{h} = \frac{y(x_i + h) - y(x_i)}{h}. \tag{7.38}$$

Similarly, the equation (7.34), gives

$$f'(x_i) \simeq \frac{\nabla y_i}{h} = \frac{y_i - y_{i-1}}{h} = \frac{y(x_i) - y(x_i - h)}{h}. \tag{7.39}$$

Adding equations (7.38) and (7.39), we obtain the central difference formula for first order derivative, as

$$f'(x_i) \simeq \frac{y(x_i + h) - y(x_i - h)}{2h}. \tag{7.40}$$

Equations (7.38)-(7.40) give two-point formulae to find first order derivative at $x = x_i$. Similarly, from equation (7.33)

$$f''(x_i) \simeq \frac{\Delta^2 y_i}{h^2} = \frac{y_{i+2} - 2y_{i+1} + y_i}{h^2} = \frac{y(x_i+2h) - 2y(x_i + h)+y(x_i)}{h^2}. \tag{7.41}$$

From equation (7.35)

$$f''(x_i) \simeq \frac{\nabla^2 y_i}{h^2} = \frac{y_i - 2y_{i-1}+y_{i-2}}{h^2} = \frac{y(x_i)-2y(x_i - h)+y(x_i - 2h)}{h^2}. \tag{7.42}$$

Equation (7.37) gives

$$f''(x_0) \simeq \frac{\Delta^2 y_{-1}}{h^2} = \frac{y_1 - 2y_0 + y_{-1}}{h^2} = \frac{y(x_0 + h) - 2y(x_0) + y(x_0 - h)}{h^2}.$$

In general,

$$f''(x_i) \simeq \frac{y(x_i + h) - 2y(x_i) + y(x_i - h)}{h^2}. \tag{7.43}$$

Equations (7.41)-(7.43) give the three-point formulae for second order derivative.

7.6.1 Error analysis and optimum step size

The **truncation error** of the two-point formula (7.40) is $O(h^2)$. Assume that $f \in C^3[a,b]$ (i.e., f is continuously differentiable up to third order within $[a,b]$) and $x-h, x, x+h \in [a,b]$. Then by Taylor's series

$$\begin{aligned} f(x_i+h) &= f(x_i) + hf'(x_i) + \frac{h^2}{2!}f''(x_i) + \frac{h^3}{3!}f'''(\xi_1) \\ \text{and } f(x_i-h) &= f(x_i) - hf'(x_i) + \frac{h^2}{2!}f''(x_i) - \frac{h^3}{3!}f'''(\xi_2) \end{aligned}$$

By subtraction

$$f(x_i+h) - f(x_i-h) = 2hf'(x_i) + \frac{f'''(\xi_1)+f'''(\xi_2)}{3!}h^3. \tag{7.44}$$

Since f''' is continuous, by the intermediate value theorem there exist a number ξ so that

$$\frac{f'''(\xi_1)+f'''(\xi_2)}{2} = f'''(\xi).$$

Thus, after rearrangement the equation (7.44) becomes

$$f'(x_i) = \frac{f(x_i+h)-f(x_i-h)}{2h} - \frac{f'''(\xi)h^2}{3!}. \tag{7.45}$$

It may be noted that the first term of right hand side is two-point formula while second term is the truncation error and it is of $O(h^2)$.

To find the computer's **round-off error**, it is assumed that $f(x_0-h) = y(x_0-h)+\varepsilon_{-1}$ and $f(x_0+h) = y(x_0+h)+\varepsilon_1$ where $y(x_0-h)$ and $y(x_0+h)$ are the approximate values of the original function f at the points (x_0-h) and (x_0+h) respectively and ε_{-1} and ε_1 are the round-off errors.

Thus

$$f'(x_i) = \frac{y(x_i+h)-y(x_i-h)}{2h} + E_{\text{trunc}}$$

and

$$\begin{aligned} f'(x_i) &= \frac{y(x_i+h)-y(x_i-h)}{2h} + E_{\text{trunc}} + E_{\text{round}} \\ &= \frac{y(x_i+h)-y(x_i-h)}{2h} + E \end{aligned}$$

where

$$E = E_{\text{round}} + E_{\text{trunc}} = \frac{\varepsilon_1 - \varepsilon_{-1}}{2h} - \frac{h^2 f'''(\xi)}{6} \tag{7.46}$$

is the **total error** accumulating the round-off error (E_{round}) and the truncation error (E_{trunc}).

Let $|\varepsilon_{-1}| \le \varepsilon$, $|\varepsilon_1| \le \varepsilon$ and $M_3 = \max\limits_{a\le x\le b} |f'''(x)|$.

Then from (7.46), the upper bound of the total error is given by

$$|E| \le \frac{|\varepsilon_1| + |\varepsilon_{-1}|}{2h} + \frac{h^2}{6}|f'''(\xi)| \le \frac{\varepsilon}{h} + \frac{M_3h^2}{6}. \tag{7.47}$$

Now, $|E|$ will be minimum for a given h if $\dfrac{d|E|}{dh} = 0$ i.e., $-\dfrac{\varepsilon}{h^2} + \dfrac{hM_3}{3} = 0$. Thus the optimum value of h to minimize $|E|$ is

$$h = \left(\frac{3\varepsilon}{M_3}\right)^{1/3} \tag{7.48}$$

and the minimum total error is

$$|E| = \varepsilon\left(\frac{3\varepsilon}{M_3}\right)^{-1/3} + \frac{M_3}{6}\left(\frac{3\varepsilon}{M_3}\right)^{2/3}.$$

Example 7.6.1 If $f \in C^5[a,b]$ (i.e., the function f is continuously differentiable up to fifth order on $[a,b]$) and $x-2h, x-h, x+h, x+2h \in [a,b]$, then show that

$$f'(x) \simeq \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} + \frac{h^4 f^v(\xi)}{30}, \tag{7.49}$$

where ξ lies between $x-2h$ and $x+2h$. Determine the optimal value of h when
(i) $|E_{\text{round}}| = |E_{\text{trunc}}|$,
(ii) total error $|E_{\text{round}}| + |E_{\text{trunc}}|$ is minimum.

Solution. By Taylor's series expansion for step length h and $-h$,

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{iv}(x) + \frac{h^5}{5!}f^v(\xi_1)$$

and

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \frac{h^4}{4!}f^{iv}(x) - \frac{h^5}{5!}f^v(\xi_2).$$

Then by subtraction

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{2f'''(x)h^3}{3!} + \frac{2f^v(\xi_3)h^5}{5!}.$$

Similarly, when step size is $2h$ then

$$f(x+2h)-f(x-2h)=4hf'(x)+\frac{16f'''(x)h^3}{3!}+\frac{64f^v(\xi_4)h^5}{5!}.$$

All ξ's lie between $x-2h$ and $x+2h$.
Now,

$$\begin{aligned}&f(x-2h)-f(x+2h)+8f(x+h)-8f(x-h)\\&=12hf'(x)+\frac{16f^v(\xi_3)-64f^v(\xi_4)}{120}h^5.\end{aligned}$$

Since $f^v(x)$ is continuous, $f^v(\xi_3)\simeq f^v(\xi_4)=f^v(\xi)$ (say).
Then $16f^v(\xi_3)-64f^v(\xi_4)=-48f^v(\xi)$.
Using this result the above equation becomes

$$f(x-2h)-f(x+2h)+8f(x+h)-8f(x-h)=12hf'(x)-\frac{2}{5}h^5f^v(\xi).$$

Hence, the value of $f'(x)$ is given by

$$f'(x)\simeq\frac{-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)}{12h}+\frac{f^v(\xi)h^4}{30}. \qquad (7.50)$$

The first term on the right hand side is a four-point formula to find $f'(x)$ and second term is the corresponding truncation error.
Let $f(x+2h)=y_2+\varepsilon_2$, $f(x+h)=y_1+\varepsilon_1$, $f(x-h)=y_{-1}+\varepsilon_{-1}$ and $f(x-2h)=y_{-2}+\varepsilon_{-2}$, where y_i and ε_i are the approximate values of $f(x+ih)$ and the corresponding round-off errors respectively. Also let

$$M_5=\max_{x-2h\le\xi\le x+2h}|f^v(\xi)|.$$

Then (7.50) becomes

$$\begin{aligned}f'(x)&=\frac{-y_2+8y_1-8y_{-1}+y_{-2}}{12h}+\frac{-\varepsilon_2+8\varepsilon_1-8\varepsilon_{-1}+\varepsilon_{-2}}{12h}+\frac{f^v(\xi)h^4}{30}\\&=\frac{-y_2+8y_1-8y_{-1}+y_{-2}}{12h}+E_{\text{round}}+E_{\text{trunc}}.\end{aligned}$$

Let $\varepsilon=\max\{|\varepsilon_{-2}|,|\varepsilon_{-1}|,|\varepsilon_1|,|\varepsilon_2|\}$. Then

$$|E_{\text{round}}|\le\frac{|\varepsilon_2|+8|\varepsilon_1|+8|\varepsilon_{-1}|+|\varepsilon_{-2}|}{12h}\le\frac{18\varepsilon}{12h}=\frac{3\varepsilon}{2h}$$

and $|E_{\text{trunc}}|\le\dfrac{M_5h^4}{30}$.

(i) If $|E_{\text{round}}| = |E_{\text{trunc}}|$ then $\dfrac{3\varepsilon}{2h} = \dfrac{M_5 h^4}{30}$ or, $h^4 = \dfrac{45\varepsilon}{M_5}$.

Thus the optimum value of h is $\left(\dfrac{45\varepsilon}{M_5}\right)^{1/4}$ and

$$|E_{\text{round}}| = |E_{\text{trunc}}| = \frac{3\varepsilon}{2}\left(\frac{M_5}{45\varepsilon}\right)^{1/4} = (M_5\varepsilon^3)^{1/4}\left(\frac{9}{80}\right)^{1/4}.$$

(ii) When total error $|E| = |E_{\text{round}}| + |E_{\text{trunc}}|$ is minimum then $\dfrac{d|E|}{dh} = 0$

or, $-\dfrac{3\varepsilon}{2h^2} + \dfrac{4M_5h^3}{30} = 0$, i.e., $h^5 = \dfrac{45\varepsilon}{4M_5}$.

Hence, in this case, the optimum value of h is $\left(\dfrac{45\varepsilon}{4M_5}\right)^{1/5}$.

Corollary 7.6.1 *If $f \in C^5[a,b]$ and $x_{-2}, x_{-1}, x_1, x_2 \in [a,b]$ then*

$$f'(x_0) = \frac{-f(x_2) + 8f(x_1) - 8f(x_{-1}) + f(x_{-2})}{12h} + \frac{h^4 f^v(\xi)}{30}, \tag{7.51}$$

where $x_{-2} < \xi < x_2$.

Example 7.6.2 The value of x and $f(x) = x\cos x$ are tabulated as follows:

x :	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$f(x)$:	0.19601	0.28660	0.36842	0.43879	0.49520	0.53539	0.55737	0.55945	0.54030

Find the value of $f'(0.6)$ using the two- and four-point formulae

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h}$$

and
$$f'(x_0) = \frac{-f(x_0+2h) + 8f(x_0+h) - 8f(x_0-h) + f(x_0-2h)}{12h}$$

with step size $h = 0.1$.

Solution. By the two-point formula,

$$f'(0.6) \simeq \frac{f(0.7) - f(0.5)}{0.2} = \frac{0.53539 - 0.43879}{0.2} = 0.48300$$

and by the four-point formula,

$$\begin{aligned} f'(0.6) &\simeq \frac{-f(0.8) + 8f(0.7) - 8f(0.5) + f(0.4)}{12 \times 0.1} \\ &= \frac{-0.55737 + 8 \times 0.53539 - 8 \times 0.43879 + 0.36842}{1.2} = 0.48654. \end{aligned}$$

The exact value is $f'(0.6) = \cos 0.6 - 0.6 \times \sin 0.6 = 0.48655$.
Therefore, error in two-point formula is 0.00355 and that in four-point formula is 0.00001. Clearly, four-point formula gives better result than two-point formula.

7.7 Richardson's Extrapolation Method

The improvement of derivative of a function can be done using this method. This method reduces the number of function evaluation to achieve the higher order accuracy. The formula to find the first order derivative using two points is

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + E_{\text{trunc}} = g(h) + E_{\text{trunc}}$$

where E_{trunc} is the truncation error and $g(h)$ is the approximate first order derivative of $f(x)$.

Using Taylor's series expansion, it can be shown that, E_{trunc} is of the following form.

$$E_{\text{trunc}} = c_1h^2 + c_2h^4 + c_3h^6 + \cdots .$$

The **Richardson's extrapolation** method combines two values of $f'(x)$ obtained by a certain method with two different step sizes, say, h_1 and h_2. Generally, h_1 and h_2 are taken as h and $h/2$. Thus

$$f'(x) = g(h) + E_{\text{trunc}} = g(h) + c_1h^2 + c_2h^4 + c_3h^6 + \cdots \tag{7.52}$$

$$\text{and } f'(x) = g(h/2) + c_1\frac{h^2}{4} + c_2\frac{h^4}{16} + c_3\frac{h^6}{32} + \cdots . \tag{7.53}$$

The constants $c_1, c_2, \ldots$ are independent of h. Eliminating c_1 between (7.52) and (7.53), and we get

$$\begin{aligned} f'(x) &= \frac{4g(h/2) - g(h)}{3} - \frac{3}{4}c_2h^4 - \frac{7}{8}c_3h^6 + \cdots \\ &= \frac{4g(h/2) - g(h)}{3} + d_1h^4 + d_2h^6 + \cdots \end{aligned} \tag{7.54}$$

Denoting

$$\frac{4g(h/2) - g(h)}{3} \quad \text{by} \quad g_1(h/2), \tag{7.55}$$

equation (7.54) becomes

$$f'(x) = g_1(h/2) + d_1h^4 + O(h^6). \tag{7.56}$$

This equation shows that $g_1(h/2)$ is an approximate value of $f'(x)$ with fourth-order accuracy. Thus a result accurate up to fourth order is obtained by combining two results accurate up to second order.

Now, by repeating the above result one can obtain

$$\begin{aligned} f'(x) &= g_1(h/2) + d_1 h^4 + O(h^6) \\ f'(x) &= g_1(h/2^2) + d_1 \frac{h^4}{16} + O(h^6) \end{aligned} \tag{7.57}$$

Eliminating d_1 from (7.57) to find $O(h^6)$ order formula, as

$$f'(x) = g_2(h/2^2) + O(h^6), \tag{7.58}$$

where

$$g_2(h/2^2) = \frac{4^2 g_1(h/2^2) - g_1(h/2)}{4^2 - 1}. \tag{7.59}$$

Thus $g_2(h/2^2)$ is the sixth-order accurate result of $f'(x)$.

Hence the successive higher order results can be obtained from the following formula

$$\begin{aligned} g_k\left(\frac{h}{2^m}\right) &= \frac{4^k g_{k-1}\left(\frac{h}{2^m}\right) - g_{k-1}\left(\frac{h}{2^{m-1}}\right)}{4^k - 1}, \\ & k = 1, 2, 3, \ldots;\ m = k, k+1, \ldots \\ & \text{where } g_0(h) = g(h). \end{aligned} \tag{7.60}$$

This process is called repeated extrapolation to the limit. The values of $g_k(h/2^m)$ for different values of k and m are tabulated as shown in Table 7.1.

Table 7.1: Richardson's extrapolation table

h	second order	fourth order	sixth order	eight order
h	$g(h)$			
		$g_1(h/2)$		
$h/2$	$g(h/2)$		$g_2(h/2^2)$	
		$g_1(h/2^2)$		$g_3(h/2^3)$
$h/2^2$	$g(h/2^2)$		$g_2(h/2^3)$	
		$g_1(h/2^3)$		
$h/2^3$	$g(h/2^3)$			

It may be noted that the successive values in a particular column give better approximations of the derivative than the preceding columns. This process will terminate when

$$|g_m(h/2) - g_{m-1}(h)| \leq \varepsilon$$

for a given error tolerance ε.

We have seen that one approximate value of $f'(x)$ is $g_1(h/2)$ where

$$g_1(h/2) = \frac{4g(h/2) - g(h)}{3} = g(h/2) + \frac{g(h/2) - g(h)}{3}.$$

Here $g(h/2)$ is more accurate than $g(h)$ and then $g_1(h/2)$ gives an improved approximation over $g(h/2)$. If $g(h) < g(h/2), g_1(h/2) > g(h/2)$ and if $g(h/2) < g(h)$, $g_1(h/2) < g(h/2)$. Thus the value of $g_1(h/2)$ lies outside the interval $[g(h), g(h/2)]$ or $[g(h/2), g(h)]$ as the case may be. Thus $g_1(h/2)$ is obtained from $g(h)$ and $g(h/2)$ by means of an extrapolation operation. So, this process is called (Richardson) extrapolation.

Example 7.7.1 Use Richardson's extrapolation method to find $f'(0.5)$ where $f(x) = 1/x$ starting with $h = 0.2$.

Solution. Here $h = 0.2$ and $x = 0.5$.
Then

$$g(h) = \frac{f(x+h) - f(x-h)}{2h} = \frac{\dfrac{1}{0.5+0.2} - \dfrac{1}{0.5-0.2}}{2 \times 0.2} = -4.76190,$$

$$g(h/2) = \frac{f(x+h/2) - f(x-h/2)}{2(h/2)} = \frac{\dfrac{1}{0.5+0.1} - \dfrac{1}{0.5-0.1}}{0.2} = -4.16667.$$

Then $g_1(h/2) = \dfrac{4g(h/2) - g(h)}{4-1} = \dfrac{4 \cdot (-4.16667) - (-4.76190)}{3} = -3.96826.$

Halving the step size further, we compute

$$g(h/2^2) = \frac{\dfrac{1}{0.5+0.05} - \dfrac{1}{0.5-0.05}}{2 \times 0.05} = -4.04040,$$

$$g_1(h/2^2) = \frac{4g(h/2^2) - g(h/2)}{4-1} = \frac{4 \times (-4.04040) - (-4.16667)}{3} = -3.99831.$$

Now, $$g_2(h/2^2) = \frac{4^2 g_1(h/2^2) - g_1(h/2)}{4^2 - 1} = \frac{16 \times (-3.99831) - (-3.96826)}{15}$$
$$= -4.00031.$$

The above calculations are tabulated as follows:

h	g	g_1	g_2
0.2	–4.76190		
		–3.96826	
0.1	–4.16667		–4.00031
		–3.99831	
0.05	–4.04040		

Thus, after two steps we found that $f'(0.5) = -4.00031$ while the exact value is $f'(0.5) = \left[-\dfrac{1}{x^2}\right]_{x=0.5} = -4.0$.

Example 7.7.2 For the following table

x	:	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
$f(x)$	:	1	2/3	1/2	2/5	1/3	2/7	1/4	2/9	1/5	2/11	1/6

find the value of $f'(3)$ using Richardson's extrapolation.

Solution. Let $x = 3$ and $h = 2$.

$$g(h) = \frac{f(x+h) - f(x-h)}{2h} = \frac{f(5) - f(1)}{2 \times 2} = \frac{\frac{1}{6} - \frac{1}{2}}{4} = -0.083333.$$

$$g(h/2) = \frac{f(x+h/2) - f(x-h/2)}{2(h/2)} = \frac{f(4) - f(2)}{2} = \frac{\frac{1}{5} - \frac{1}{3}}{2} = -0.066666.$$

$$g_1(h/2) = \frac{4g(h/2) - g(h)}{4-1} = \frac{4 \times (-0.06666) - (-0.083333)}{3} = -0.061110.$$

$$g(h/2^2) = g(0.5) = \frac{f(3.5) - f(2.5)}{2 \times 0.5} = \frac{\frac{2}{9} - \frac{2}{7}}{1} = -0.063492.$$

$$g_1(h/2^2) = \frac{4g(h/2^2) - g(h/2)}{4-1} = \frac{4 \times (-0.063492) - (-0.066666)}{3} = -0.062434.$$

Thus

$$\begin{aligned} g_2(h/2^2) &= \frac{4^2 g_1(h/2^2) - g_1(h/2)}{4^2 - 1} \\ &= \frac{16 \times (-0.062434) - (-0.061110)}{15} = -0.062522. \end{aligned}$$

Hence $f'(3) = -0.062522$.

Algorithm 7.2 (Richardson's extrapolation). This algorithm is used to find the derivative using Richardson's extrapolation formula.

The formula (7.60) can be written as

$$g_k(h) = \frac{4^k g_{k-1}(h) - g_{k-1}(2h)}{4^k - 1}$$

at each iteration h becomes half of its previous value.
We denote $g_{k-1}(2h)$ (calculated as previous iteration) by $g_0(k-1)$ (old value) and $g_{k-1}(h)$ by $g_n(k-1)$ (new value) to remove h from the formula.
Then the above formula reduces to

$$g_n(k) = \frac{4^k g_n(k-1) - g_o(k-1)}{4^k - 1}, \quad k = 1, 2, \ldots$$
$$\text{where } g_n(0) = \frac{f(x+h) - f(x-h)}{2h} = g_0(0).$$

```
Algorithm Richardson_extrapolation
        Input function $f(x)$;
        Read $x, h, \varepsilon$; //error tolerance//
        Compute $g_0(0) = \frac{f(x+h) - f(x-h)}{2h}$;
        Set $j = 1$;
10: Set $h = h/2$;
        Compute $g_n(0) = \frac{f(x+h) - f(x-h)}{2h}$;
        for $k = 1$ to $j$ do
                $g_n(k) = \frac{4^k g_n(k-1) - g_o(k-1)}{4^k - 1}$;
        if $|g_n(j-1) - g_n(j)| < \varepsilon$ then
                Print $g_n(j)$ as the value of derivative;
                Stop;
        else
                for $k = 0$ to $j$ do
                        $g_0(k) = g_n(k)$; //set new values as old values//
                $j = j + 1$;
                goto 10;
        endif;
end Richardson_extrapolation
```

Program 7.2

```
/* Program Richardson Extrapolation
   This program finds the first order derivative of a function
   f(x) at a given value of x by Richardson Extrapolation.
   Here we assume that f(x)=1/(x*x).
*/
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
void main()
 {
 int j=1,k;
 float x,h,eps=1e-5,g0[100],gn[100];
 float f(float x);
 printf("Enter the value of x ");
 scanf("%f",&x);
 printf("Enter the value of h ");
 scanf("%f",&h);
 g0[0]=(f(x+h)-f(x-h))/(h+h);
start: h=h/2;
         gn[0]=(f(x+h)-f(x-h))/(h+h);
         for(k=1;k<=j;k++)
             gn[k]=(pow(4,k)*gn[k-1]-g0[k-1])/(pow(4,k)-1);
         if(fabs(gn[j-1]-gn[j])<eps)
           {
            printf("The derivative is %8.5f at %8.5f ",gn[j],x);
            exit(0);
           }
         else
           {
            for(k=0;k<=j;k++) g0[k]=gn[k];
            j++;
            goto start;
           }
 } /* main */
/* definition of f(x) */
float f(float x)
 {
    return(1/(x*x));
 }
```

A sample of input/output:

```
Enter the value of x 1.5
Enter the value of h 0.5
The derivative is -0.59259 at  1.50000
```

7.8 Cubic Spline Method

The cubic spline may be used to determine the first and second derivatives of a function. This method works into two stages. In first stage the cubic splines will be constructed with suitable intervals and in second stage the first and second derivatives are to be determined from the appropriate cubic spline. This method is labourious than the other methods, but, once a cubic spline is constructed then the method becomes very efficient. The process of finding derivative is illustrated by as example in the following.

Example 7.8.1 Let $y = f(x) = \cos x, 0 \le x \le \pi/2$ be the function. Find the natural cubic spline in the intervals $0 \le x \le \pi/4$ and $\pi/4 \le x \le \pi/2$ and hence determine the approximate values of $f'(\pi/8)$ and $f''(\pi/8)$. Also, use two-point formula to find the value of $f'(\pi/8)$. Find the error in each case.

Solution. Here $n = 2$. Therefore, $h = \pi/4, y_0 = \cos 0 = 1, y_1 = \cos \pi/4 = 1/\sqrt{2}$ and $y_2 = \cos \pi/2 = 0$.
Also, $M_0 = M_2 = 0$.
Then by the formula (3.99)

$$M_0 + 4M_1 + M_2 = \frac{6}{h^2}[y_0 - 2y_1 + y_2]$$

That is, $4M_1 = \frac{96}{\pi^2}(1 - \sqrt{2})$ or, $M_1 = \frac{24}{\pi^2}(1 - \sqrt{2}) = -1.007246602$.
Hence the cubic spline is

$$S(x) = \begin{cases} s_1(x), \; 0 \le x \le \pi/4 \\ s_2(x), \; \pi/4 \le x \le \pi/2, \end{cases}$$

where

$$s_1(x) = \frac{4}{\pi}\left[\frac{x^3}{6}M_1 - \left(1 - \frac{1}{\sqrt{2}} + \frac{\pi^2}{96}M_1\right)x + \frac{\pi}{4}\right],$$

and

$$s_2(x) = \frac{4}{\pi}\left[\frac{(\pi/2 - x)^3}{6}M_1 - \left(\frac{1}{\sqrt{2}} - \frac{\pi^2}{96}M_1\right)(\pi/2 - x)\right].$$

Now, $f'(\pi/8) \simeq s_1'(\pi/8) = -0.33996116$, $f''(\pi/8) \simeq s_1''(\pi/8) = -0.503623301$.

Two-point formula.
Let $h = \pi/30$.
Then $f'(\pi/8) \simeq \dfrac{f(\pi/8 + \pi/30) - f(\pi/8 - \pi/30)}{2.\pi/30} = -0.381984382.$
The actual value of $f'(\pi/8)$ is $-\sin\pi/8 = -0.382683432$.
Therefore, error in cubic spline method is 0.042722272 while in two-point formula that is 0.000699050365.

7.9 Determination of Extremum of a Tabulated Function

It is known that, if a function is differentiable then the maximum and minimum value of that function can be determined by equating the first derivative to zero and solving for the variable. The same method is applicable for the tabulated function.

Now, consider the Newton's forward difference interpolation formula.

$$y = f(x) = y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 + \cdots, \tag{7.61}$$

where $u = \dfrac{x - x_0}{h}$.

Then

$$\frac{dy}{dx} = \Delta y_0 + \frac{2u-1}{2}\Delta^2 y_0 + \frac{3u^2 - 6u + 2}{6}\Delta^3 y_0 + \cdots.$$

For maxima and minima $\dfrac{dy}{dx} = 0$. Then

$$\Delta y_0 + \frac{2u-1}{2}\Delta^2 y_0 + \frac{3u^2 - 6u + 2}{6}\Delta^3 y_0 + \cdots = 0 \tag{7.62}$$

For simplicity, the third and higher differences are neglected and obtain the quadratic equation for u as

$$au^2 + bu + c = 0, \tag{7.63}$$

where $a = \frac{1}{2}\Delta^3 y_0, b = \Delta^2 y_0 - \Delta^3 y_0, c = \Delta y_0 - \frac{1}{2}\Delta^2 y_0 + \frac{1}{3}\Delta^3 y_0$.

The values of u can be determined by solving this equation. Then the values of x are determined from the relation $x = x_0 + uh$. Finally, the maximum value of y can be obtained from the equation (7.61).

Example 7.9.1 Find x for which y is maximum and also find the corresponding value of y, from the following table.

x :	1.5	2.0	2.5	3.0	3.5
y :	0.40547	0.69315	0.91629	1.09861	1.25276

Solution. The forward difference table is

x	y	Δy	$\Delta^2 y$
1.5	0.40547		
		0.28768	
2.0	0.69315		–0.06454
		0.22314	
2.5	0.91629		–0.04082
		0.18232	
3.0	1.09861		–0.02817
		0.15415	
3.5	1.25276		

Let $x_0 = 1.5$. Using formula (7.62) we have
$\Delta y_0 + \frac{2u-1}{2}\Delta^2 y_0 = 0$ or, $0.28768 + \frac{2u-1}{2} \times (-0.06454) = 0$ or, $u = 4.95739$.
Therefore, $x = x_0 + uh = 1.5 + 0.5 \times 4.95739 = 3.97870$.
For this x, the value of y is obtained by Newton's backward formula as

$$\begin{aligned} y(3.97870) &= 1.25276 + 0.47870 \times (0.15415) \\ &\quad + \frac{0.47870(0.47870+1)}{2} \times (-0.02817) = 1.31658. \end{aligned}$$

This is the approximate maximum value of y when $x = 3.97870$.

7.10 Integration

It is well known that, if a function $f(x)$ is known completely, even then it is not always possible to evaluate the definite integral of it using analytic method. Again, in many real life problems, we are required to integrate a function between two given limits, but the function is not known explicitly, but, it is known in a tabular form (equally or unequally spaced). Then a method, known as **numerical integration or quadrature** can be used to solve all such problems.

The problem of numerical integration is stated below:
Given a set of data points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ of a function $y = f(x)$, it is required to find the value of the definite integral $\int_a^b f(x)\,dx$. The function $f(x)$ is replaced by a suitable interpolating polynomial $\phi(x)$.

Then the approximate value of the definite integral is calculated using the following formula

$$\int_a^b f(x)\,dx \simeq \int_a^b \phi(x)\,dx. \qquad (7.64)$$

Thus, different integration formulae can be derived depending on the type of the interpolation formulae used.

A numerical integration formula is said to be of **closed type**, if the limits of integration a and b are taken as interpolating points. If a and b are not taken as interpolating points then the formula is known as **open type** formula.

7.11 General Quadrature Formula Based on Newton's Forward Interpolation

The Newton's forward interpolation formula for the equispaced points $x_i, i = 0, 1, \dots, n$, $x_i = x_0 + ih$ is

$$\phi(x) = y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 + \cdots, \tag{7.65}$$

where $u = \dfrac{x - x_0}{h}$, h is the spacing.

Let the interval $[a, b]$ be divided into n equal subintervals such that $a = x_0 < x_1 < x_2 < \cdots < x_n = b$. Then

$$\begin{aligned} I &= \int_a^b f(x)\,dx \simeq \int_{x_0}^{x_n} \phi(x)\,dx \\ &= \int_{x_0}^{x_n} \Big[y_0 + u\Delta y_0 + \frac{u^2 - u}{2!}\Delta^2 y_0 + \frac{u^3 - 3u^2 + 2u}{3!}\Delta^3 y_0 + \cdots\Big]\,dx. \end{aligned}$$

Since $x = x_0 + uh$, $dx = h\,du$, when $x = x_0$ then $u = 0$ and when $x = x_n$ then $u = n$. Thus,

$$\begin{aligned} I &= \int_0^n \Big[y_0 + u\Delta y_0 + \frac{u^2 - u}{2!}\Delta^2 y_0 + \frac{u^3 - 3u^2 + 2u}{3!}\Delta^3 y_0 + \cdots\Big]\,hdu \\ &= h\Big[y_0[u]_0^n + \Delta y_0\Big[\frac{u^2}{2}\Big]_0^n + \frac{\Delta^2 y_0}{2!}\Big[\frac{u^3}{3} - \frac{u^2}{2}\Big]_0^n + \frac{\Delta^3 y_0}{3!}\Big[\frac{u^4}{4} - u^3 + u^2\Big]_0^n + \cdots\Big] \\ &= nh\Big[y_0 + \frac{n}{2}\Delta y_0 + \frac{2n^2 - 3n}{12}\Delta^2 y_0 + \frac{n^3 - 4n^2 + 4n}{24}\Delta^3 y_0 + \cdots\Big]. \end{aligned} \tag{7.66}$$

From this formula, one can generate different integration formulae by substituting $n = 1, 2, 3, \dots$.

7.11.1 Trapezoidal Rule

Substituting $n = 1$ in the equation (7.66). In this case all differences higher than the first difference become zero. Then

$$\int_{x_0}^{x_n} f(x)\,dx = h\Big[y_0 + \frac{1}{2}\Delta y_0\Big] = h\Big[y_0 + \frac{1}{2}(y_1 - y_0)\Big] = \frac{h}{2}(y_0 + y_1). \tag{7.67}$$

The formula (7.67) is known as the **trapezoidal rule**.

In this formula, the interval $[a, b]$ is considered as a single interval, and it gives a very rough answer. But, if the interval $[a, b]$ is divided into several subintervals and this formula is applied to each of these subintervals then a better approximate result may be obtained. This formula is known as composite formula, deduced below.

Composite trapezoidal rule

Let the interval $[a, b]$ be divided into n equal subintervals by the points $a = x_0$, x_1, x_2, …, $x_n = b$, where $x_i = x_0 + ih$, $i = 1, 2, \ldots, n$.

Applying the trapezoidal rule to each of the subintervals, one can find the composite formula as

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_{x_0}^{x_1} f(x)\,dx + \int_{x_1}^{x_2} f(x)\,dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)\,dx \\ &\simeq \frac{h}{2}[y_0 + y_1] + \frac{h}{2}[y_1 + y_2] + \frac{h}{2}[y_2 + y_3] + \cdots + \frac{h}{2}[y_{n-1} + y_n] \\ &= \frac{h}{2}[y_0 + 2(y_1 + y_2 + \cdots + y_{n-1}) + y_n]. \qquad (7.68)\end{aligned}$$

Error in trapezoidal rule

The error of trapezoidal rule is

$$E = \int_a^b f(x)\,dx - \frac{h}{2}(y_0 + y_1). \qquad (7.69)$$

Let $y = f(x)$ be continuous and possesses continuous derivatives of all orders. Also, it is assumed that there exists a function $F(x)$ such that $F'(x) = f(x)$ in $[x_0, x_1]$.

Then

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_{x_0}^{x_1} F'(x)\,dx = F(x_1) - F(x_0) \\ &= F(x_0 + h) - F(x_0) = F(x_0) + hF'(x_0) + \frac{h^2}{2!}F''(x_0) \\ &\quad + \frac{h^3}{3!}F'''(x_0) + \cdots - F(x_0) \\ &= hf(x_0) + \frac{h^2}{2!}f'(x_0) + \frac{h^3}{3!}f''(x_0) + \cdots \\ &= hy_0 + \frac{h^2}{2}y_0' + \frac{h^3}{6}y_0'' + \cdots \qquad (7.70)\end{aligned}$$

Again,

$$\begin{aligned}\frac{h}{2}(y_0+y_1) &= \frac{h}{2}[y_0+y(x_0+h)]\\ &= \frac{h}{2}[y_0+y(x_0)+hy'(x_0)+\frac{h^2}{2!}y''(x_0)+\cdots]\\ &= \frac{h}{2}[y_0+y_0+hy_0'+\frac{h^2}{2!}y_0''+\cdots].\end{aligned} \tag{7.71}$$

Using (7.70) and (7.71), equation (7.69) becomes

$$\begin{aligned}E &= h\Big[y_0+\frac{h}{2}y_0'+\frac{h^2}{6}y_0''+\cdots\Big]-\frac{h}{2}\Big[2y_0+hy_0'+\frac{h^2}{2!}y_0''+\cdots\Big]\\ &= -\frac{h^3}{12}y_0''+\cdots\\ &= -\frac{h^3}{12}f''(x_0)+\cdots\simeq -\frac{h^3}{12}f''(\xi),\end{aligned} \tag{7.72}$$

where $a=x_0<\xi<x_1=b$.

Equation (7.72) gives the error in the interval $[x_0,x_1]$.

The total error in the composite rule is

$$E=-\frac{h^3}{12}(y_0''+y_1''+\cdots+y_{n-1}'').$$

If $y''(\xi)$ is the largest among the n quantities $y_0'',y_1'',\ldots,y_{n-1}''$ then

$$E\le -\frac{1}{12}h^3ny''(\xi)=-\frac{(b-a)}{12}h^2y''(\xi),\ \text{as } nh=b-a.$$

Note 7.11.1 The error term shows that if the second and higher order derivatives of $f(x)$ vanish then the trapezoidal rule gives exact result of the integral. This means, the method gives exact result when $f(x)$ is linear.

Geometrical interpretation of trapezoidal rule

In this rule, the curve $y=f(x)$ is replaced by the line joining the points $A(x_0,y_0)$ and $B(x_1,y_1)$ (Figure 7.1). Thus the area bounded by the curve $y=f(x)$, the ordinates $x=x_0$, $x=x_1$ and the x-axis is then approximately equivalent to the area of the trapezium (ABCD) bounded by the line AB, $x=x_0$, $x=x_1$ and x-axis.

The geometrical significance of composite trapezoidal rule is that the curve $y=f(x)$ is replaced 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). Then the area bounded by the curve $y=f(x)$, the lines $x=x_0,x=x_n$ and the x-axis is then approximately equivalent to the sum of the area of n trapeziums (Figures 7.2).

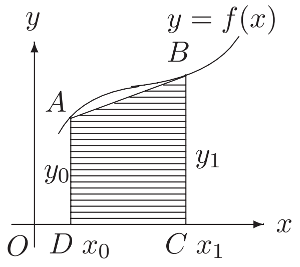


Figure 7.1: Geometrical interpretation of trapezoidal rule.

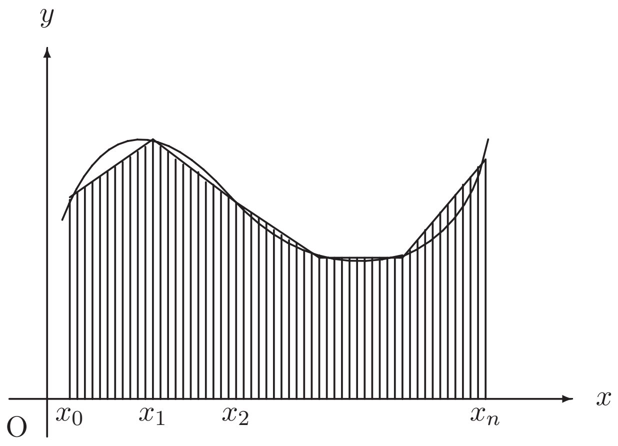


Figure 7.2: Composite trapezoidal rule.

Alternative deduction of trapezoidal rule

Let $f \in C^2[a,b]$, where $[a,b]$ is a finite interval. Now, transfer the interval $[a,b]$ to $[-1,1]$ using the relation $x = \dfrac{a+b}{2} + \dfrac{b-a}{2}t = p + qt$ (say).

Let $f(x) = f(p+qt) = g(t)$. When $x = a, b$ then $t = -1, 1$, i.e., $g(1) = f(b), g(-1) = f(a)$.

Thus

$$\begin{aligned}
I &= \int_a^b f(x)dx = \int_{-1}^1 g(t)\ q\ dt = q\Big[\int_{-1}^0 g(t)dt + \int_0^1 g(t)dt\Big] \\
&= q\int_0^1 [g(t) + g(-t)]dt.
\end{aligned}$$

Now, applying integration by parts.

$$\begin{aligned}
I &= q\Big[\{g(t)+g(-t)\}t\Big]_0^1 - q\int_0^1 t[g'(t)-g'(-t)]dt \\
&= q[g(1)+g(-1)] - q\int_0^1 t.2tg''(c)dt,\ \text{where } 0<c<1 \\
&\quad \text{[by Lagrange's MVT]} \\
&= q[f(a)+f(b)] - 2\,q\,g''(d)\int_0^1 t^2dt, 0<d<1, \\
&\quad \text{[by MVT of integral calculus]} \\
&= q[f(a)+f(b)] - \frac{2}{3}qg''(d) \\
&= q[f(a)+f(b)] - \frac{2}{3}q^3f''(p+qd) \\
&= q[f(a)+f(b)] - \frac{2}{3}q^3f''(\xi),\ \text{where } a<\xi<b \\
&= \frac{b-a}{2}[f(a)+f(b)] - \frac{2}{3}\Big(\frac{b-a}{2}\Big)^3 f''(\xi) \\
&= \frac{h}{2}[f(a)+f(b)] - \frac{1}{12}h^3f''(\xi),\ \text{as } h=b-a.
\end{aligned}$$

In this expression, the first term is the approximate integration obtained by trapezoidal rule and the second term represents the error.

Algorithm 7.3 (Trapezoidal). This algorithm finds the value of $\int_a^b f(x)dx$ based on the tabulated values $(x_i, y_i), y_i = f(x_i), i = 0, 1, 2, \ldots, n$, using trapezoidal rule.

```
Algorithm Trapezoidal
Input function $f(x)$;
Read $a, b, n$; //the lower and upper limits and number of subintervals.//
Compute $h = (b - a)/n$;

Set $sum = \frac{1}{2}[f(a) + f(a + nh)]$;

for $i = 1$ to $n - 1$ do
    Compute $sum = sum + f(a + ih)$;
endfor;
Compute $result = sum * h$;
Print $result$;
end Trapezoidal
```

Program 7.3

```
/* Program Trapezoidal
   This program finds the value of integration of a function
   by trapezoidal rule.
   Here we assume that f(x)=x^3. */
#include<stdio.h>
void main()
{
 float a,b,h,sum; int n,i;
 float f(float);
 printf("Enter the values of a, b ");
 scanf("%f %f",&a,&b);
 printf("Enter the value of n ");
 scanf("%d",&n);
 h=(b-a)/n;
 sum=(f(a)+f(a+n*h))/2.;
 for(i=1;i<=n-1;i++) sum+=f(a+i*h);
 sum=sum*h;
 printf("The value of the integration is %8.5f ",sum);
}

/* definition of the function f(x) */
float f(float x)
 {
   return(x*x*x);
 }
```

A sample of input/output:

```
Enter the values of a, b 0 1
Enter the value of n 100
The value of the integration is  0.25002
```

7.11.2 Simpson's 1/3 rule

In this formula the interval $[a, b]$ is divided into two equal subintervals by the points x_0, x_1, x_2, where $h = (b - a)/2$, $x_1 = x_0 + h$ and $x_2 = x_1 + h$.

This rule is obtained by putting $n = 2$ in (7.66). In this case, the third and higher order differences do not exist.

The equation (7.66) is simplified as

$$\begin{aligned}\int_{x_0}^{x_n} f(x)\,dx &\simeq 2h\left[y_0+\Delta y_0+\frac{1}{6}\Delta^2 y_0\right] = 2h[y_0+(y_1-y_0)+\frac{1}{6}(y_2-2y_1+y_0)] \\ &= \frac{h}{3}[y_0+4y_1+y_2]. \qquad (7.73)\end{aligned}$$

The above rule is known as **Simpson's 1/3 rule** or simply **Simpson's rule**.

Composite Simpson's 1/3 rule

Let the interval $[a,b]$ be divided into n (an *even number*) equal subintervals by the points $x_0, x_1, x_2, \ldots, x_n$, where $x_i = x_0 + ih$, $i = 1, 2, \ldots, n$. Then

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_{x_0}^{x_2} f(x)\,dx + \int_{x_2}^{x_4} f(x)\,dx + \cdots + \int_{x_{n-2}}^{x_n} f(x)\,dx \\ &= \frac{h}{3}[y_0+4y_1+y_2] + \frac{h}{3}[y_2+4y_3+y_4] + \cdots + \frac{h}{3}[y_{n-2}+4y_{n-1}+y_n] \\ &= \frac{h}{3}[y_0+4(y_1+y_3+\cdots+y_{n-1})+2(y_2+y_4+\cdots+y_{n-2})+y_n]. \qquad (7.74)\end{aligned}$$

This formula is known as **Simpson's 1/3 composite rule** for numerical integration.

Error in Simpson's 1/3 rule

The error in this formula is

$$E = \int_{x_0}^{x_n} f(x)\,dx - \frac{h}{3}[y_0+4y_1+y_2]. \qquad (7.75)$$

Let the function $f(x)$ be continuous in $[x_0, x_2]$ and possesses continuous derivatives of all order. Also, let there exists a function $F(x)$ in $[x_0, x_2]$, such that $F'(x) = f(x)$. Then

$$\begin{aligned}\int_{x_0}^{x_2} f(x)\,dx &= \int_{x_0}^{x_2} F'(x)\,dx = F(x_2) - F(x_0) \\ &= F(x_0+2h) - F(x_0) = F(x_0) + 2hF'(x_0) + \frac{(2h)^2}{2!}F''(x_0) \\ &\quad + \frac{(2h)^3}{3!}F'''(x_0) + \frac{(2h)^4}{4!}F^{iv}(x_0) + \frac{(2h)^5}{5!}F^{v}(x_0) + \cdots - F(x_0) \\ &= 2hf(x_0) + 2h^2 f'(x_0) + \frac{4}{3}h^3 f''(x_0) + \frac{2}{3}h^4 f'''(x_0) \\ &\quad + \frac{4}{15}h^5 f^{iv}(x_0) + \cdots. \qquad (7.76)\end{aligned}$$

Again,

$$\begin{aligned}
\frac{h}{3}[y_0+4y_1+y_2] &= \frac{h}{3}[f(x_0)+4f(x_1)+f(x_2)]\\
&= \frac{h}{3}[f(x_0)+4f(x_0+h)+f(x_0+2h)]\\
&= \frac{h}{3}\Big[f(x_0)+4\Big\{f(x_0)+hf'(x_0)+\frac{h^2}{2!}f''(x_0)+\frac{h^3}{3!}f'''(x_0)\\
&\quad +\frac{h^4}{4!}f^{iv}(x_0)+\cdots\Big\}+\Big\{f(x_0)+2hf'(x_0)+\frac{(2h)^2}{2!}f''(x_0)\\
&\quad +\frac{(2h)^3}{3!}f'''(x_0)+\frac{(2h)^4}{4!}f^{iv}(x_0)+\cdots\Big\}\Big]\\
&= 2hf(x_0)+2h^2f'(x_0)+\frac{4}{3}h^3f''(x_0)+\frac{2}{3}h^4f'''(x_0)\\
&\quad +\frac{5}{18}h^5f^{iv}(x_0)+\cdots . \qquad (7.77)
\end{aligned}$$

Using (7.76) and (7.77), equation (7.75) becomes,

$$E=\Big(\frac{4}{15}-\frac{5}{18}\Big)h^5f^{iv}(x_0)+\cdots \simeq -\frac{h^5}{90}f^{iv}(\xi), \qquad (7.78)$$

where $x_0<\xi<x_2$.

This is the error in the interval $[x_0,x_2]$.

The total error in composite formula is

$$\begin{aligned}
E &= -\frac{h^5}{90}\{f^{iv}(x_0)+f^{iv}(x_2)+\cdots+f^{iv}(x_{n-2})\}\\
&= -\frac{h^5}{90}\frac{n}{2}f^{iv}(\xi)\\
&= -\frac{nh^5}{180}f^{iv}(\xi),\\
&\qquad \text{(where } f^{iv}(\xi) \text{ is the maximum among } f^{iv}(x_0),f^{iv}(x_2),\ldots,f^{iv}(x_{n-2}))\\
&= -\frac{(b-a)}{180}h^4f^{iv}(\xi). \qquad (7.79)
\end{aligned}$$

Geometrical interpretation of Simpson's 1/3 rule

In Simpson's 1/3 rule, the curve $y=f(x)$ is replaced by the second degree parabola passing through the points $A(x_0,y_0),B(x_1,y_1)$ and $C(x_2,y_2)$. Therefore, the area bounded by the curve $y=f(x)$, the ordinates $x=x_0,x=x_2$ and the x-axis is approximated to the area bounded by the parabola ABC, the straight lines $x=x_0,x=x_2$ and x-axis, i.e., the area of the shaded region ABCDEA.

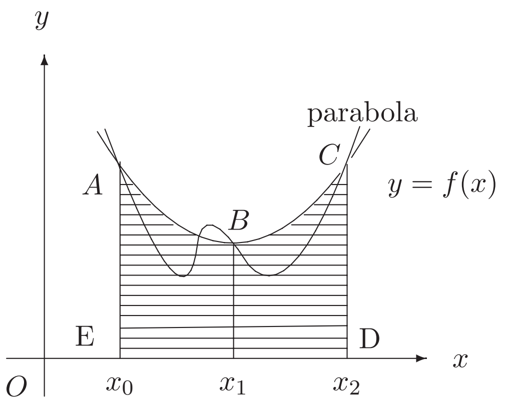


Figure 7.3: Geometrical interpretation of Simpson's 1/3 rule.

Example 7.11.1 Evaluate $\int_0^3 (2x - x^2)\,dx$, taking 6 intervals, by (i) Trapezoidal rule, and (ii) Simpson's 1/3 rule.

Solution. Here $n = 6, a = 0, b = 3, y = f(x) = 2x - x^2$.
So, $h = \dfrac{b-a}{n} = \dfrac{3-0}{6} = 0.5$.
The tabulated values of x and y are shown below.

	x_0	x_1	x_2	x_3	x_4	x_5	x_6
x_i :	0.0	0.5	1.0	1.5	2.0	2.5	3.0
y_i :	0.0	0.75	1.0	0.75	0.0	-1.25	-3.0
	y_0	y_1	y_2	y_3	y_4	y_5	y_6

(i) By Trapezoidal rule:

$$\int_0^3 (2x - x^2)\,dx = \frac{h}{2}[y_0 + 2(y_1 + y_2 + y_3 + y_4 + y_5) + y_6]$$
$$= \frac{0.5}{2}[0 + 2(0.75 + 1.0 + 0.75 + 0 - 1.25) - 3.0] = -0.125.$$

(ii) By Simpson's rule:

$$\int_0^3 (2x - x^2)\,dx = \frac{h}{3}[y_0 + 4(y_1 + y_3 + y_5) + 2(y_2 + y_4) + y_6]$$
$$= \frac{0.5}{3}[0 + 4(0.75 + 0.75 - 1.25) + 2(1.0 + 0.0) - 3.0]$$
$$= \frac{0.5}{3}[0 + 1 + 2 - 3] = 0.$$

Alternative deduction of Simpson's 1/3 rule

This rule can also be deduced by applying MVT of differential and of integral calculus. Let $f \in C^4[a,b]$ and $x = \dfrac{a+b}{2} + \dfrac{b-a}{2}z = p + qz, p = \dfrac{a+b}{2}, q = \dfrac{b-a}{2}$.

Then when $x = a, b$ then $z = -1, 1$.

Therefore,

$$\begin{aligned} I &= \int_a^b f(x)dx = q\int_{-1}^1 f(p+qz)dz \\ &= q\int_{-1}^1 g(z)dz, \text{ where } g(z) = f(p+qz) \\ &= q\left[\int_{-1}^0 g(z)dz + \int_0^1 g(z)dz\right] = q\int_0^1 [g(z)+g(-z)]dz \\ &= q\int_0^1 \phi(z)dz, \end{aligned} \tag{7.80}$$

where $\phi(z) = g(z) + g(-z)$.

Note that $\phi(0) = 2g(0) = 2f(p) = 2f(\frac{a+b}{2}), \phi(1) = g(1)+g(-1) = f(a)+f(b), \phi'(0) = 0$.

To prove $\displaystyle\int_0^1 \phi(z)dz = (1+c)\phi(1) - c\phi(0) - \int_0^1 (z+c)\phi'(z)dz$, for arbitrary constant c.

$$\begin{aligned} \int_0^1 \phi(z)dz &= \int_0^1 \phi(z)d(z+c) = \int_c^{1+c} \phi(y-c)dy \qquad [\text{where } z+c=y] \\ &= \Big[y\phi(y-c)\Big]_c^{1+c} - \int_c^{1+c} y\phi'(y-c)dy \\ &= (1+c)\phi(1) - c\phi(0) - \int_0^1 (z+c)\phi'(z)d(z+c) \\ &= (1+c)\phi(1) - c\phi(0) - \int_0^1 (z+c)\phi'(z)dz. \end{aligned} \tag{7.81}$$

Now, integrating (7.80) thrice

$$\int_0^1 \phi(z)dz = (1+c)\phi(1) - c\phi(0) - \int_0^1 (z+c)\phi'(z)dz$$
$$= (1+c)\phi(1) - c\phi(0) - \left[\Big(\frac{z^2}{2} + cz + c_1\Big)\phi'(z)\right]_0^1 + \int_0^1 \Big(\frac{z^2}{2} + cz + c_1\Big)\phi''(z)dz$$

$$
\begin{aligned}
&= (1+c)\phi(1) - c\phi(0) - \Big(\frac{1}{2}+c+c_1\Big)\phi'(1) + c_1\phi'(0)\\
&\quad + \Big[\Big(\frac{z^3}{6}+c\frac{z^2}{2}+c_1z+c_2\Big)\phi''(z)\Big]_0^1 - \int_0^1 \Big(\frac{z^3}{6}+c\frac{z^2}{2}+c_1z+c_2\Big)\phi'''(z)dz\\
&= (1+c)\phi(1) - c\phi(0) - \Big(\frac{1}{2}+c+c_1\Big)\phi'(1) + \Big(\frac{1}{6}+\frac{c}{2}+c_1+c_2\Big)\phi''(1)\\
&\quad - c_2\phi''(0) - \int_0^1 \Big(\frac{z^3}{6}+c\frac{z^2}{2}+c_1z+c_2\Big)\phi'''(z)dz, \qquad (7.82)
\end{aligned}
$$

where c_1, c_2, c_3 are arbitrary constants and they are chosen in such a way that $\phi'(1), \phi''(1)$ and $\phi''(0)$ vanish. Thus

$$\frac{1}{2}+c+c_1 = 0, \qquad \frac{1}{6}+\frac{c}{2}+c_1+c_2 = 0, \qquad \text{and} \qquad c_2 = 0.$$

The solution of these equations is $c_2 = 0, c_1 = \dfrac{1}{6}, c = -\dfrac{2}{3}$.
Hence

$$
\begin{aligned}
I &= q\Big[\frac{1}{3}\phi(1) + \frac{2}{3}\phi(0) - \int_0^1 \Big(\frac{z^3}{6} - \frac{z^2}{3} + \frac{z}{6}\Big)\phi'''(z)dz\Big]\\
&= h\Big[\frac{1}{3}\Big\{f(a)+f(b)\Big\} + \frac{4}{3}f\Big(\frac{a+b}{2}\Big)\Big] - \frac{h}{6}\int_0^1 (z^3-2z^2+z)\phi'''(z)dz\Big]\\
&\quad \Big[\text{as } q = \frac{b-a}{2} = h\Big]\\
&= \frac{h}{3}\Big[f(a) + 4f\Big(\frac{a+b}{2}\Big) + f(b)\Big] + E
\end{aligned}
$$

where

$$
\begin{aligned}
E &= -\frac{h}{6}\int_0^1 z(z-1)^2\phi'''(z)dz = -\frac{h}{6}\int_0^1 z(z-1)^2[g'''(z) - g'''(-z)]dz\\
&= -\frac{h}{6}\int_0^1 z(z-1)^2.[2zg^{iv}(\xi)]dz, \qquad -z < \xi < z\\
&\quad \text{[by Lagrange's MVT]}\\
&= -\frac{h}{3}g^{iv}(\xi_1)\int_0^1 z^2(z-1)^2dz \qquad \text{[by MVT of integral calculus]}\\
&= -\frac{h}{3}g^{iv}(\xi_1).\frac{1}{30} = -\frac{h}{90}g^{iv}(\xi_1), \qquad 0 < \xi_1 < 1.
\end{aligned}
$$

Again, $g(z) = f(p+qz), g^{iv}(z) = q^4 f^{iv}(p+qt) = h^4 f^{iv}(\xi_2),\ a < \xi_2 < b$.
Therefore,

$$E = -\frac{h^5}{90}f^{iv}(\xi_2).$$

Hence,

$$\int_a^b f(x)dx = \frac{h}{3}\Big[f(a) + 4f\Big(\frac{a+b}{2}\Big) + f(b)\Big] - \frac{h^5}{90}f^{iv}(\xi_2).$$

Here, the first term is the value of the integration obtained from the Simpson's 1/3 rule and the second term is its error.

Algorithm 7.4 (Simpson's 1/3). This algorithm determines the value of $\int_a^b f(x)\,dx$ using Simpson's 1/3 rule.

```
Algorithm Simpson_One_Third
Input function $f(x)$;
Read $a, b, n$; //the lower and upper limits and number of subintervals.//
Compute $h = (b - a)/n$;
Set $sum = [f(a) - f(a + nh)]$;
for $i = 1$ to $n - 1$ step 2 do
    Compute $sum = sum + 4 * f(a + ih) + 2 * f(a + (i + 1)h)$;
endfor;
Compute $result = sum * h/3$;
Print $result$;
end Simpson_One_Third.
```

Program 7.4

```
/* Program Simpson's 1/3
   Program to find the value of integration of a function
   f(x) using Simpson's 1/3 rule. Here we assume that f(x)=x^3.*/
#include<stdio.h>
void main()
{
 float f(float);
 float a,b,h,sum;
 int i,n;
 printf("\nEnter the values of a, b ");
 scanf("%f %f",&a,&b);
 printf("Enter the value of subintervals n ");
 scanf("%d",&n);
 if(n%2!=0) {
     printf("Number of subdivision should be even");
     exit(0);
   }
 h=(b-a)/n;
 sum=f(a)-f(a+n*h);
 for(i=1;i<=n-1;i+=2)
    sum+=4*f(a+i*h)+2*f(a+(i+1)*h);
 sum*=h/3.;
 printf("Value of the integration is %f ",sum);
} /* main */

/* definition of the function f(x) */
float f(float x)
  {
   return(x*x*x);
  }
```

A sample of input/output:

```
Enter the values of a, b 0 1
Enter the value of subintervals n 100
Value of the integration is 0.250000
```

7.11.3 Simpson's 3/8 rule

Simpson's 3/8 rule can be obtained by substituting $n = 3$ in (7.66). Note that the differences higher than the third order do not exist here.

$$\begin{aligned}\int_a^b f(x)dx = \int_{x_0}^{x_3} f(x)dx &= 3h\Big[y_0 + \frac{3}{2}\Delta y_0 + \frac{3}{4}\Delta^2 y_0 + \frac{1}{8}\Delta^3 y_0\Big]\\ &= 3h\Big[y_0 + \frac{3}{2}(y_1 - y_0) + \frac{3}{4}(y_2 - 2y_1 + y_0) + \frac{1}{8}(y_3 - 3y_2 + 3y_1 - y_0)\Big]\\ &= \frac{3h}{8}[y_0 + 3y_1 + 3y_2 + y_3]. \end{aligned} \tag{7.83}$$

This formula is known as **Simpson's 3/8 rule**.

Now, the interval $[a, b]$ is divided into n (divisible by 3) equal subintervals by the points $x_0, x_1, \ldots, x_n$ and the formula is applied to each of the intervals.

Then

$$\begin{aligned}\int_{x_0}^{x_n} f(x)dx &= \int_{x_0}^{x_3} f(x)dx + \int_{x_3}^{x_6} f(x)dx + \cdots + \int_{x_{n-3}}^{x_n} f(x)dx\\ &= \frac{3h}{8}[(y_0 + 3y_1 + 3y_2 + y_3) + (y_3 + 3y_4 + 3y_5 + y_6)\\ &\quad + \cdots + (y_{n-3} + 3y_{n-2} + 3y_{n-1} + y_n)]\\ &= \frac{3h}{8}[y_0 + 3(y_1 + y_2 + y_4 + y_5 + y_7 + y_8 + \cdots + y_{n-2} + y_{n-1})\\ &\quad + 2(y_3 + y_6 + y_9 + \cdots + y_{n-3}) + y_n]. \end{aligned} \tag{7.84}$$

This formula is known as **Simpson's 3/8 composite rule**.

Note 7.11.2 This method is not so accurate as Simpson's 1/3 rule. The error in this formula is $-\frac{3}{80}h^5 f^{iv}(\xi), x_0 < \xi < x_3$.

7.11.4 Boole's rule

Substituting $n = 4$ in (7.66). The equation (7.66) reduces to

$$\begin{aligned}\int_a^b f(x)dx &= 4h\left[y_0 + 2\Delta y_0 + \frac{5}{3}\Delta^2 y_0 + \frac{2}{3}\Delta^3 y_0 + \frac{7}{90}\Delta^4 y_0\right]\\ &= 4h[y_0 + 2(y_1 - y_0) + \frac{5}{3}(y_2 - 2y_1 + y_0) + \frac{2}{3}(y_3 - 3y_2 + 3y_1 - y_0)\\ &\quad + \frac{7}{90}(y_4 - 4y_3 + 6y_2 - 4y_1 + y_0)]\\ &= \frac{2h}{45}[7y_4 + 32y_3 + 12y_2 + 32y_1 + 7y_0]. \qquad (7.85)\end{aligned}$$

This rule is known as **Boole's rule**.

It can be shown that the error of this formula is $-\frac{8h^7}{945}f^{vi}(\xi), a < \xi < b$.

7.11.5 Weddle's rule

To find Weddle's rule, substituting $n = 6$ in (7.66). Then

$$\begin{aligned}&\int_a^b f(x)dx\\ &= 6h\left[y_0 + 3\Delta y_0 + \frac{9}{2}\Delta^2 y_0 + 4\Delta^3 y_0 + \frac{41}{20}\Delta^4 y_0 + \frac{11}{20}\Delta^5 y_0 + \frac{41}{840}\Delta^6 y_0\right]\\ &= 6h\left[y_0 + 3\Delta y_0 + \frac{9}{2}\Delta^2 y_0 + 4\Delta^3 y_0 + \frac{41}{20}\Delta^4 y_0 + \frac{11}{20}\Delta^5 y_0 + \frac{1}{20}\Delta^6 y_0\right] - \frac{h}{140}\Delta^6 y_0.\end{aligned}$$

If the sixth order difference is very small, then we may neglect the last term $\frac{h}{140}\Delta^6 y_0$. But, this rejection increases a negligible amount of error, though, it simplifies the integration formula. Then the above equation becomes

$$\begin{aligned}&\int_{x_0}^{x_6} f(x)dx\\ &= \frac{3h}{10}[20y_0 + 60\Delta y_0 + 90\Delta^2 y_0 + 80\Delta^3 y_0 + 41\Delta^4 y_0 + 11\Delta^5 y_0 + \Delta^6 y_0]\\ &= \frac{3h}{10}[y_0 + 5y_1 + y_2 + 6y_3 + y_4 + 5y_5 + y_6]. \qquad (7.86)\end{aligned}$$

This formula is known as **Weddle's rule** for numerical integration.

Composite Weddle's rule

In this rule, interval $[a, b]$ is divided into n (divisible by 6) subintervals by the points $x_0, x_1, \ldots, x_n$. Then

$$\begin{aligned}\int_{x_0}^{x_n} f(x)dx &= \int_{x_0}^{x_6} f(x)dx + \int_{x_6}^{x_{12}} f(x)dx + \cdots + \int_{x_{n-6}}^{x_n} f(x)dx \\ &= \frac{3h}{10}[y_0 + 5y_1 + y_2 + 6y_3 + y_4 + 5y_5 + y_6] \\ &\quad + \frac{3h}{10}[y_6 + 5y_7 + y_8 + 6y_9 + y_{10} + 5y_{11} + y_{12}] + \cdots \\ &\quad + \frac{3h}{10}[y_{n-6} + 5y_{n-5} + y_{n-4} + 6y_{n-3} + y_{n-2} + 5y_{n-1} + y_n] \\ &= \frac{3h}{10}[y_0 + 5(y_1 + y_5 + y_7 + y_{11} + \cdots + y_{n-5} + y_{n-1}) \\ &\quad + (y_2 + y_4 + y_8 + y_{10} + \cdots + y_{n-4} + y_{n-2}) \\ &\quad + 6(y_3 + y_9 + y_{15} + \cdots + y_{n-3}) + 2(y_6 + y_{12} + \cdots + y_{n-6})].\end{aligned} \tag{7.87}$$

The above formula is known as **Weddle's composite rule**.

By the technique used in trapezoidal and Simpson's 1/3 rules one can prove that the error in Weddle's rule is $-\frac{h^7}{140} f^{vi}(\xi), x_0 < \xi < x_6$.

Degree of Precision

The degree of precision of a quadrature formula is a positive integer n such that the error is zero for all polynomials of degree $i \le n$, but it is non-zero for some polynomials of degree $n + 1$.

The degree of precision of some quadrature formulae are given in Table 7.2.

Table 7.2: Degree of precision of some quadrature formulae.

Method	Degree of precision
Trapezoidal	1
Simpson's 1/3	3
Simpson's 3/8	3
Boole's	5
Weddle's	5

Comparison of Simpson's 1/3 and Weddle's rules

The Weddle's rule gives more accurate result than Simpson's 1/3 rule. But, Weddle's rule has a major disadvantage that it requires the number of subdivisions (n) as a multiple of six. In many cases, the value of $h = \frac{b-a}{n}$ (n is multiple of six) is not finite in decimal representation. For these reasons, the values of $x_0, x_1, \ldots, x_n$ can not be determined accurately and hence the values of y i.e., $y_0, y_1, \ldots, y_n$ become inaccurate. In Simpson's 1/3 rule, n, the number of subdivisions is even, so one can take n as 10, 20 etc. and hence h is finite in decimal representation. Thus the values of $x_0, x_1, \ldots, x_n$ and $y_0, y_1, \ldots, y_n$ can be computed correctly.

However, Weddle's rule should be used when Simpson's 1/3 rule does not give the desired accuracy.

7.12 Integration Based on Lagrange's Interpolation

Let the function $y = f(x)$ be known at the $(n + 1)$ points $x_0, x_1, \ldots, x_n$ of $[a, b]$, these points need not be equispaced.

The Lagrange's interpolation polynomial is

$$\phi(x) = \sum_{i=0}^{n} \frac{w(x)}{(x - x_i)w'(x_i)} y_i \tag{7.88}$$
$$\text{where } w(x) = (x - x_0) \cdots (x - x_n)$$

and $\phi(x_i) = y_i, i = 0, 1, 2, \ldots, n$.

If the function $f(x)$ is replaced by the polynomial $\phi(x)$ then

$$\int_a^b f(x)dx \simeq \int_a^b \phi(x)dx = \sum_{i=0}^{n} \int_a^b \frac{w(x)}{(x - x_i)w'(x_i)} y_i dx. \tag{7.89}$$

The above equation can be written as

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{n} C_i y_i, \tag{7.90}$$

$$\text{where } \quad C_i = \int_a^b \frac{w(x)}{(x - x_i)w'(x_i)} dx, \qquad i = 0, 1, 2, \ldots, n. \tag{7.91}$$

It may be noted that the coefficients C_i are independent of the choice of the function $f(x)$ for a given set of points.

7.13 Newton-Cotes Integration Formulae (Closed type)

Let the interpolation points $x_0, x_1, \ldots, x_n$ be equispaced, i.e., $x_i = x_0 + ih$, $i = 1, 2, \ldots, n$. Also, let $x_0 = a, x_n = b$, $h = (b-a)/n$ and $y_i = f(x_i), i = 0, 1, 2, \ldots, n$. Then the definite integral $\int_a^b f(x)dx$ can be determined on replacing $f(x)$ by Lagrange's interpolation polynomial $\phi(x)$ and then the approximate integration formula is given by

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{n} C_i y_i, \tag{7.92}$$

where C_i are some constant coefficients.

Now, the explicit expressions for C_i's are evaluated in the following.

The Lagrange's interpolation polynomial is

$$\phi(x) = \sum_{i=0}^{n} L_i(x) y_i, \tag{7.93}$$

where

$$L_i(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}. \tag{7.94}$$

Introducing $x = x_0 + sh$. Then $x - x_i = (s-i)h$ and $x_i - x_j = (i-j)h$. Therefore,

$$\begin{aligned} L_i(x) &= \frac{sh(s-1)h\cdots(s-\overline{i-1})h(s-\overline{i+1})h\cdots(s-n)h}{ih(i-1)h\cdots(i-\overline{i-1})h(i-\overline{i+1})h\cdots(i-n)h} \\ &= \frac{(-1)^{n-i}}{i!(n-i)!}\frac{s(s-1)(s-2)\cdots(s-n)}{(s-i)}. \end{aligned} \tag{7.95}$$

Then (7.92) becomes

$$\int_{x_0}^{x_n} f(x)dx \simeq \sum_{i=0}^{n} C_i y_i$$

$$\text{or,} \quad \int_{x_0}^{x_n} \sum_{i=0}^{n} \frac{(-1)^{n-i}}{i!(n-i)!}\frac{s(s-1)(s-2)\cdots(s-n)}{(s-i)} y_i dx = \sum_{i=0}^{n} C_i y_i$$

$$\text{or,} \quad \sum_{i=0}^{n} \left\{ \int_{x_0}^{x_n} \frac{(-1)^{n-i}}{i!(n-i)!}\frac{s(s-1)(s-2)\cdots(s-n)}{(s-i)} dx \right\} y_i = \sum_{i=0}^{n} C_i y_i. \tag{7.96}$$

Now, comparing both sides to find the expression for C_i in the form

$$\begin{aligned} C_i &= \int_{x_0}^{x_n} \frac{(-1)^{n-i}}{i!(n-i)!}\frac{s(s-1)(s-2)\cdots(s-n)}{(s-i)} dx \\ &= \frac{(-1)^{n-i}h}{i!(n-i)!}\int_0^n \frac{s(s-1)(s-2)\cdots(s-n)}{(s-i)} ds, \end{aligned} \tag{7.97}$$

$i = 0, 1, 2, \ldots, n$ and $x = x_0 + sh$.

Since $h = \dfrac{b-a}{n}$, substituting

$$C_i = (b-a)H_i, \tag{7.98}$$

where

$$H_i = \frac{1}{n}\frac{(-1)^{n-i}}{i!(n-i)!}\int_0^n \frac{s(s-1)(s-2)\cdots(s-n)}{(s-i)}ds, i = 0, 1, 2, \ldots, n. \tag{7.99}$$

These coefficients H_i are called **Cotes coefficients**.

Then the integration formula (7.92) becomes

$$\int_a^b f(x)dx \simeq (b-a)\sum_{i=0}^n H_i y_i, \tag{7.100}$$

where H_i's are given by (7.99).

Note 7.13.1 The cotes coefficients H_i's do not depend on the function $f(x)$.

7.13.1 Some results on Cotes coefficients

(i) $\sum_{i=0}^n C_i = (b-a)$.

By the property of Lagrangian functions, $\sum_{i=0}^n \dfrac{w(x)}{(x-x_i)w'(x_i)} = 1$

$$\text{That is, } \int_a^b \sum_{i=0}^n \frac{w(x)}{(x-x_i)w'(x_i)}dx = \int_a^b dx = (b-a). \tag{7.101}$$

Again,

$$\begin{aligned}\int_a^b \sum_{i=0}^n \frac{w(x)}{(x-x_i)w'(x_i)}dx &= \sum_{i=0}^n \int_0^n h(-1)^{n-i}\frac{s(s-1)(s-2)\cdots(s-n)}{i!(n-i)!(s-i)}ds \\ &= \sum_{i=0}^n C_i. \end{aligned} \tag{7.102}$$

Hence from (7.101) and (7.102),

$$\sum_{i=0}^n C_i = b-a. \tag{7.103}$$

(ii) $\sum_{i=0}^{n} H_i = 1.$
From the relation (7.98),
$C_i = (b-a)H_i$

or, $\sum_{i=0}^{n} C_i = (b-a)\sum_{i=0}^{n} H_i$

or, $(b-a) = (b-a)\sum_{i=0}^{n} H_i$. [using (7.103)]

Hence,

$$\sum_{i=0}^{n} H_i = 1. \tag{7.104}$$

That is, sum of cotes coefficients is one.

(iii) $C_i = C_{n-i}$.
From the definition of C_i, one can find

$$C_{n-i} = \frac{(-1)^i h}{(n-i)!i!}\int_0^n \frac{s(s-1)(s-2)\cdots(s-n)}{s-(n-i)}ds.$$

Substituting $t = n-s$, we obtain

$$\begin{aligned} C_{n-i} &= -\frac{(-1)^i h(-1)^n}{i!(n-i)!}\int_n^0 \frac{t(t-1)(t-2)\cdots(t-n)}{t-i}dt \\ &= \frac{(-1)^{n-i}h}{i!(n-i)!}\int_0^n \frac{s(s-1)(s-2)\cdots(s-n)}{s-i}dt = C_i. \end{aligned}$$

Hence,

$$C_i = C_{n-i}. \tag{7.105}$$

(iv) $H_i = H_{n-i}$.
Multiplying (7.105) by $(b-a)$ and hence obtain

$$H_i = H_{n-i}. \tag{7.106}$$

7.13.2 Deduction of quadrature formulae

Trapezoidal rule

Substituting $n = 1$ in (7.100), we get

$$\int_a^b f(x)dx = (b-a)\sum_{i=0}^{1} H_i y_i = (b-a)(H_0y_0 + H_1y_1).$$

Now H_0 and H_1 are obtained from (7.99) by substituting $i = 0$ and 1. Therefore,

$$H_0 = -\int_0^1 \frac{s(s-1)}{s}ds = \frac{1}{2} \text{ and } H_1 = \int_0^1 sds = \frac{1}{2}.$$

Here, $h = (b-a)/n = b-a$ for $n = 1$.

Hence, $\displaystyle\int_a^b f(x)dx = \frac{(b-a)}{2}(y_0 + y_1) = \frac{h}{2}(y_0 + y_1)$.

Simpson's 1/3 rule

For $n = 2$, $H_0 = \displaystyle\frac{1}{2}.\frac{1}{2}\int_0^2 (s-1)(s-2)ds = \frac{1}{6}$

$H_1 = \displaystyle -\frac{1}{2}\int_0^2 s(s-2)ds = \frac{2}{3}, \qquad H_2 = \frac{1}{2}.\frac{1}{2}\int_0^2 s(s-1)ds = \frac{1}{6}.$

In this case $h = (b-a)/2$.

Hence equation (7.100) gives the following formula.

$$\begin{aligned}\int_a^b f(x)dx &= (b-a)\sum_{i=0}^{2} H_i y_i = (b-a)(H_0y_0 + H_1y_1 + H_2y_2)\\ &= \frac{h}{3}(y_0 + 4y_1 + y_2).\end{aligned}$$

Weddle's rule

To deduce the Weddle's rule, $n = 6$ is substituted in (7.100).

$$\begin{aligned}\int_a^b f(x)dx &= (b-a)\sum_{i=0}^{6} H_i y_i\\ &= 6h(H_0y_0 + H_1y_1 + H_2y_2 + H_3y_3 + H_4y_4 + H_5y_5 + H_6y_6)\\ &= 6h[H_0(y_0 + y_6) + H_1(y_1 + y_5) + H_2(y_2 + y_4) + H_3y_3].\end{aligned}$$

To find the values of H_i's one may use the result $H_i = H_{n-i}$. Also the value of H_3 can be obtained by the formula

$H_3 = 1 - (H_0 + H_1 + H_2 + H_4 + H_5 + H_6) = 1 - 2(H_0 + H_1 + H_2)$.

Now, $H_0 = \frac{1}{6} \cdot \frac{1}{6!} \int_0^6 \frac{s(s-1)(s-2)\cdots(s-6)}{s} ds = \frac{41}{840}$.

Similarly, $H_1 = \frac{216}{840}, H_2 = \frac{27}{840}, H_3 = \frac{272}{840}$.

Hence,

$$\int_a^b f(x)dx = \frac{h}{140}[41y_0 + 216y_1 + 27y_2 + 272y_3 + 27y_4 + 216y_5 + 41y_6]. \qquad (7.107)$$

Again, we know that $\Delta^6 y_0 = y_0 - 6y_1 + 15y_2 - 20y_3 + 15y_4 - 6y_5 + y_6$,
i.e., $\frac{h}{140}[y_0 - 6y_1 + 15y_2 - 20y_3 + 15y_4 - 6y_5 + y_6] - \frac{h}{140}\Delta^6 y_0 = 0$.

Adding left hand side of above identity (as it is zero) to the right hand side of (7.107). After simplification the equation (7.107) finally reduces to

$$\int_a^b f(x)dx = \frac{3h}{10}[y_0 + 5y_1 + y_2 + 6y_3 + y_4 + 5y_5 + y_6] - \frac{h}{140}\Delta^6 y_0.$$

The first term is the well known Weddle's rule and the last term is the error in addition to the truncation error.

Table 7.3: Weights of Newton-Cotes integration rule for different n.

n							
1	$\frac{1}{2}$	$\frac{1}{2}$					
2	$\frac{1}{3}$	$\frac{4}{3}$	$\frac{1}{3}$				
3	$\frac{3}{8}$	$\frac{9}{8}$	$\frac{9}{8}$	$\frac{3}{8}$			
4	$\frac{14}{45}$	$\frac{64}{45}$	$\frac{24}{45}$	$\frac{64}{45}$	$\frac{14}{45}$		
5	$\frac{95}{288}$	$\frac{375}{288}$	$\frac{250}{288}$	$\frac{250}{288}$	$\frac{375}{288}$	$\frac{95}{288}$	
6	$\frac{41}{140}$	$\frac{216}{140}$	$\frac{27}{140}$	$\frac{272}{140}$	$\frac{27}{140}$	$\frac{216}{140}$	$\frac{41}{140}$

7.14 Newton-Cotes Formulae (Open Type)

All the formulae based on Newton-Cotes formula developed in Section 7.13 are of closed type, i.e., they use the function values at the end points a, b of the interval $[a, b]$ of integration. Here, some formulae are introduced those take the function values at equispaced intermediate points, but, not at the end points. These formulae may be used when the function has singularity(s) at the end points or the values of the function are unknown at the endpoints. Also, these methods are useful to solve ordinary differential equations numerically when the function values at the end points are not available. These formulae are sometimes known as the **Steffensen formulae**.

(i) **Mid-point** formula

$$\int_{x_0}^{x_1} f(x)dx = hf(x_0 + h/2) + \frac{1}{24}h^3 f''(\xi), x_0 \le \xi \le x_1. \tag{7.108}$$

(ii) **Two-point** formula

$$\int_{x_0}^{x_3} f(x)dx = \frac{3h}{2}[f(x_1) + f(x_2)] + \frac{3h^3}{4} f''(\xi), x_0 \le \xi \le x_3. \tag{7.109}$$

(iii) **Three-point** formula

$$\int_{x_0}^{x_4} f(x)dx = \frac{4h}{3}[2f(x_1) - f(x_2) + 2f(x_3)] + \frac{14h^5}{45} f^{iv}(\xi),$$
$$x_0 \le \xi \le x_4. \tag{7.110}$$

(iv) **Four-point** formula

$$\int_{x_0}^{x_5} f(x)dx = \frac{5h}{24}[11f(x_1) + f(x_2) + f(x_3) + 11f(x_4)]$$
$$+ \frac{95h^5}{144} f^{iv}(\xi), x_0 \le \xi \le x_5. \tag{7.111}$$

These formulae may be obtained by integrating Lagrange's interpolating polynomial for the data points (x_i, y_i), $i = 1, 2, \ldots, (n-1)$ between the given limits.

Methods Based on Undetermined Coefficients

In the Newton-Cotes method all the nodes $x_i, i = 0, 1, 2, \ldots, n$ are known and equispaced. Also, the formulae obtained from Newton-Cotes method are exact for the polynomials of degree up to n. When the nodes $x_i, i = 0, 1, 2, \ldots, n$ are unknown then one can devise some methods which give exact result for the polynomials of degree up to $2n - 1$. These methods are called **Gaussian quadrature methods**.

7.15 Gaussian Quadrature

The Gaussian quadrature is of the form

$$\int_a^b \psi(x)f(x)dx = \sum_{i=1}^{n} w_i f(x_i), \tag{7.112}$$

where x_i and w_i are respectively called nodes and weights and $\psi(x)$ is called the weight function. Depending on the weight function different quadrature formula can be obtained.

The fundamental theorem of Gaussian quadrature states that *the optimal nodes of the m-point Gaussian quadrature formula are precisely the zeros of the orthogonal polynomial for the same interval and weight function.* Gaussian quadrature is optimal because it fits all polynomial up to degree $2m$ exactly.

To determine the weights corresponding to the Gaussian nodes x_i, compute a Lagrange's interpolating polynomial for $f(x)$ by assuming

$$\pi(x) = \prod_{j=1}^{m}(x - x_j). \tag{7.113}$$

Then

$$\pi'(x_j) = \prod_{\substack{i=1\\ i\neq j}}^{m}(x_j - x_i). \tag{7.114}$$

Then Lagrange's interpolating polynomial through m points is

$$\phi(x) = \sum_{j=1}^{m} \frac{\pi(x)}{(x - x_j)\pi'(x_j)} f(x_j) \tag{7.115}$$

for arbitrary points x. Now, determine a set of points x_j and w_j such that for a weight function $\psi(x)$ the following relation is valid.

$$\begin{aligned}\int_a^b \phi(x)\psi(x)dx &= \int_a^b \sum_{j=1}^{m} \frac{\pi(x)\psi(x)}{(x - x_j)\pi'(x_j)} f(x_j)dx \\ &= \sum_{j=1}^{m} w_j f(x_j),\end{aligned} \tag{7.116}$$

where weights w_j are obtained from

$$w_j = \frac{1}{\pi'(x_j)} \int_a^b \frac{\pi(x)\psi(x)}{x - x_j} dx. \tag{7.117}$$

The weights w_j are sometimes called the **Christofell numbers**.

It can be shown that the error is given by

$$E = \frac{f^{(2n)}(\xi)}{(2n)!} \int_a^b \psi(x)[\pi(x)]^2 dx. \tag{7.118}$$

Any finite interval $[a, b]$ can be transferred to the interval $[-1, 1]$ using linear transformation

$$x = \frac{b-a}{2}t + \frac{b+a}{2} = qt + p. \tag{7.119}$$

Then,

$$\int_a^b f(x)\, dx = \int_{-1}^1 f(qt+p)\, q\, dt. \tag{7.120}$$

Thus to study the Gaussian quadrature, we consider the integral in the form

$$\int_{-1}^1 \psi(x) f(x) dx = \sum_{i=1}^n w_i f(x_i) + E. \tag{7.121}$$

7.15.1 Gauss-Legendre integration methods

Here $\psi(x)$ is taken as 1 and so the formula (7.121) reduces to

$$\int_{-1}^1 f(x) dx = \sum_{i=1}^n w_i f(x_i). \tag{7.122}$$

It may be noted that w_i and x_i are $2n$ parameters and therefore the weights and nodes can be determined such that the formula is exact when $f(x)$ is a polynomial of degree not exceeding $2n-1$.

Let

$$f(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{2n-1} x^{2n-1}. \tag{7.123}$$

Therefore,

$$\begin{aligned} \int_{-1}^1 f(x) dx &= \int_{-1}^1 [c_0 + c_1 x + c_2 x^2 + \cdots + c_{2n-1} x^{2n-1}] dx \\ &= 2c_0 + \frac{2}{3}c_2 + \frac{2}{5}c_4 + \cdots . \end{aligned} \tag{7.124}$$

When $x = x_i$, equation (7.123) becomes

$$f(x_i) = c_0 + c_1 x_i + c_2 x_i^2 + c_3 x_i^3 + \cdots + c_{2n-1} x_i^{2n-1}.$$

Substituting these values to the right hand side of (7.122) to get

$$\begin{aligned}\int_{-1}^{1} f(x)dx &= w_1[c_0 + c_1x_1 + c_2x_1^2 + \cdots + c_{2n-1}x_1^{2n-1}]\\ &\quad +w_2[c_0 + c_1x_2 + c_2x_2^2 + \cdots + c_{2n-1}x_2^{2n-1}]\\ &\quad +w_3[c_0 + c_1x_3 + c_2x_3^2 + \cdots + c_{2n-1}x_3^{2n-1}]\\ &\quad + \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\ &\quad +w_n[c_0 + c_1x_n + c_2x_n^2 + \cdots + c_{2n-1}x_n^{2n-1}]\\ &= c_0(w_1 + w_2 + \cdots + w_n) + c_1(w_1x_1 + w_2x_2 + \cdots + w_nx_n)\\ &\quad +c_2(w_1x_1^2 + w_2x_2^2 + \cdots + w_nx_n^2) + \cdots\\ &\quad +c_{2n-1}(w_1x_1^{2n-1} + w_2x_2^{2n-1} + \cdots + w_nx_n^{2n-1}). \qquad (7.125)\end{aligned}$$

Since (7.124) and (7.125) are identical, compare the coefficients of c_i, and find $2n$ equations as follows:

$$\begin{aligned} w_1 + w_2 + \cdots + w_n &= 2\\ w_1x_1 + w_2x_2 + \cdots + w_nx_n &= 0\\ w_1x_1^2 + w_2x_2^2 + \cdots + w_nx_n^2 &= \tfrac{2}{3}\\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots &\quad \cdots\\ w_1x_1^{2n-1} + w_2x_2^{2n-1} + \cdots + w_nx_n^{2n-1} &= 0. \end{aligned} \qquad (7.126)$$

Now, equation (7.126) is a set of $2n$ non-linear equations consisting $2n$ unknowns w_i and x_i, $i = 1, 2, \ldots, n$. Solution of these equations gives the values of w_i and x_i. Let $w_i = w_i^*$ and $x_i = x_i^*, i = 1, 2, \ldots, n$ be the solution of (7.126). Then the Gauss-Legendre formula is finally given by

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} w_i^* f(x_i^*). \qquad (7.127)$$

Unfortunately, determination of general solution of the system (7.126) is very complicated. Thus we concentrate for its particular cases.
Case I. When $n = 1$, the formula is

$$\int_{-1}^{1} f(x)dx = w_1f(x_1), \text{ where } w_1 = 2 \text{ and } w_1x_1 = 0, \text{ i.e., } x_1 = 0.$$

Thus for $n = 1$,

$$\int_{-1}^{1} f(x)dx = 2f(0). \qquad (7.128)$$

Case II. When $n = 2$ then the integral is

$$\int_{-1}^{1} f(x)dx = w_1 f(x_1) + w_2 f(x_2). \tag{7.129}$$

and the system (7.126) reduces to

$$\begin{aligned} w_1 + w_2 \quad &= 2 \\ w_1x_1 + w_2x_2 &= 0 \\ w_1x_1^2 + w_2x_2^2 &= \tfrac{2}{3} \\ w_1x_1^3 + w_2x_2^3 &= 0. \end{aligned} \tag{7.130}$$

The above equations can also be obtained by the following way:
The formula (7.129) is exact when $f(x)$ is a polynomial of degree ≤ 3. Substituting successively $f(x) = 1, x, x^2$ and x^3 in (7.129) and obtain the following system of equations

$$\begin{aligned} w_1 + w_2 \quad &= 2 && (f(x) = 1) \\ w_1x_1 + w_2x_2 &= 0 && (f(x) = x) \\ w_1x_1^2 + w_2x_2^2 &= \tfrac{2}{3} && (f(x) = x^2) \\ w_1x_1^3 + w_2x_2^3 &= 0 && (f(x) = x^3). \end{aligned} \tag{7.131}$$

The solution of these equations is $w_1 = w_2 = 1, x_1 = -1/\sqrt{3}, x_2 = 1/\sqrt{3}$. Hence, the equation (7.129) becomes

$$\int_{-1}^{1} f(x)dx = f(-1/\sqrt{3}) + f(1/\sqrt{3}).$$

Case III. When $n = 3$ then the integral becomes

$$\int_{-1}^{1} f(x)dx = w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3). \tag{7.132}$$

The six unknowns x_1, x_2, x_3 and w_1, w_2, w_3 are related as

$$\begin{aligned} w_1 + w_2 + w_3 \quad &= 2 \\ w_1x_1 + w_2x_2 + w_3x_3 &= 0 \\ w_1x_1^2 + w_2x_2^2 + w_3x_2^3 &= \tfrac{2}{3} \\ w_1x_1^3 + w_2x_2^3 + w_3x_2^3 &= 0 \\ w_1x_1^4 + w_2x_2^4 + w_3x_2^4 &= \tfrac{2}{5} \\ w_1x_1^5 + w_2x_2^5 + w_3x_2^5 &= 0. \end{aligned}$$

These equations may also be obtained by substituting $f(x) = 1, x, x^2, x^3, x^4, x^5$ to the equation (7.132).

Solution of this system of equations is

$$x_1 = -\sqrt{3/5}, x_2 = 0, x_3 = \sqrt{3/5}, w_1 = 5/9, w_2 = 8/9, w_3 = 5/9.$$

For these values, equation (7.132) becomes

$$\int_{-1}^{1} f(x)dx = \frac{1}{9}[5f(-\sqrt{3/5}) + 8f(0) + 5f(\sqrt{3/5})]. \tag{7.133}$$

In this way one can determine the formulae for higher values of n. The solution of the equations (7.126) for higher values of n is very complicated as they are non-linear with respect to the nodes $x_1, x_2, \ldots, x_n$. But, they are linear with respect to weights.

If can be shown that $x_i, i = 1, 2, \ldots, n$ are the zeros of the nth degree Legendre's polynomial $P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}[(x^2-1)^n]$, which can be generated from the recurrence relation

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x) \tag{7.134}$$

where $P_0(x) = 1$ and $P_1(x) = x$. Some lower order Legendre polynomials are

$$\begin{aligned} &P_0(x) = 1 \\ &P_1(x) = x \\ &P_2(x) = \tfrac{1}{2}(3x^2 - 1) \\ &P_3(x) = \tfrac{1}{2}(5x^3 - 3x) \\ &P_4(x) = \tfrac{1}{8}(35x^4 - 30x^2 + 3). \end{aligned} \tag{7.135}$$

The weights w_i are given by

$$w_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{x - x_i}{x_i - x_j}\right) dx \tag{7.136}$$

where x_i are known nodes.

Also, the weights w_i may be obtained from the relation

$$w_i = \frac{2}{(1 - x_i^2)[P_n'(x_i)]^2}. \tag{7.137}$$

It can be shown that the error of this formula is

$$E = \frac{2^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi), -1 < \xi < 1. \tag{7.138}$$

The nodes and weights for different values of n are listed in Table 7.4.

Table 7.4: Values of x_i and w_i for Gauss-Legendre quadrature.

n	node x_i	weight w_i	order of truncation error
2	± 0.57735027	1.00000000	$f^{(4)}(\xi)$
3	0.00000000	0.88888889	$f^{(6)}(\xi)$
	± 0.77459667	0.55555556	
4	± 0.33998104	0.65214515	$f^{(8)}(\xi)$
	± 0.86113631	0.34785485	
5	0.00000000	0.56888889	
	± 0.53846931	0.47862867	$f^{(10)}(\xi)$
	± 0.90617985	0.23692689	
6	± 0.23861919	0.46791393	
	± 0.66120939	0.36076157	$f^{(12)}(\xi)$
	± 0.93246951	0.17132449	

Example 7.15.1 Find the value of $\int_0^1 \frac{1}{1+x^2}dx$ by Gauss's formula for $n = 2, 4, 6$. Also, calculate the absolute errors.

Solution. To apply the Gauss's formula, the limits are transferred to $-1, 1$ by substituting $x = \frac{1}{2}u(1-0) + \frac{1}{2}(1+0) = \frac{1}{2}(u+1)$.

Then $I = \int_0^1 \frac{1}{1+x^2}dx = \int_{-1}^1 \frac{2du}{(u+1)^2+4} = 2\sum_{i=1}^n w_i f(u_i)$,

where $f(x_i) = \frac{1}{(x_i+1)^2+4}$.

<u>For the two-point formula ($n = 2$)</u>

$x_1 = -0.57735027, x_2 = 0.57735027, w_1 = w_2 = 1.$
Then $I = 2[1 \times 0.23931272 + 1 \times 0.15412990] = 0.78688524.$

<u>For the four-point formula ($n = 4$)</u>

$x_1 = -0.33998104, x_2 = -0.86113631, x_3 = -x_1, x_4 = -x_2,$
$w_1 = w_3 = 0.65214515, w_2 = w_4 = 0.34785485.$
Then $I = 2[w_1 f(x_1) + w_3 f(x_3) + w_2 f(x_2) + w_4 f(x_4)]$
$= 2[w_1\{f(x_1) + f(-x_1)\} + w_2\{f(x_2) + f(-x_2)\}]$
$= 2[0.65214515 \times (0.22544737 + 0.17254620) + 0.34785485 \times (0.24880059 + 0.13397950)]$
$= 0.78540297.$

For the six-point formula ($n = 6$)

$x_1 = -0.23861919, x_2 = -0.66120939, x_3 = -0.93246951, x_4 = -x_1,$
$x_5 = -x_2, x_6 = -x_3, w_1 = w_4 = 0.46791393, w_2 = w_5 = 0.36076157,$
$w_3 = w_6 = 0.17132449.$
Then $I = 2[w_1\{f(x_1) + f(-x_1)\} + w_2\{f(x_2) + f(-x_2) + w_3\{f(x_3) + f(-x_3)\}]$
$= 2[0.46791393 \times (0.21835488 + 0.18069532) + 0.36076157 \times (0.24302641 + 0.14793738)$
$+ 0.17132449 \times (0.24971530 + 0.12929187)]$
$= 0.78539814.$
The exact value is $\pi/4 = 0.78539816$.
The following table gives a comparison among the different Gauss's formulae.

n	Exact value	Gauss formula	Error
2	0.78539816	0.78688524	1.49×10^{-3}
4	0.78539816	0.78540297	4.81×10^{-6}
6	0.78539816	0.78539814	2.00×10^{-8}

Algorithm 7.5 (Gauss-Legendre's quadrature). This algorithm finds the value of $\int_a^b f(x)\,dx$ using 6-point Gauss-Legendre's quadrature method.

```
Algorithm Gauss_Legendre
Input function $f(x)$;
Read $a, b$. //the lower and upper limits of the integral.//
Assign
$x_1 = 0.23861919$,    $x_2 = -x_1$,    $w_1 = w_2 = 0.46791393$,
$x_3 = 0.66120939$,    $x_4 = -x_3$,    $w_3 = w_4 = 0.36076157$,
$x_5 = 0.93246951$,    $x_6 = -x_5$,    $w_5 = w_6 = 0.17132449$,
Set $p = (a + b)/2, q = (b - a)/2$ //limits changed to –1, 1//
Initialize $Result = 0$;
for $i = 1$ to 6 do
    Compute $result = result + w_i * f(p + qx_i)$;
endfor;
Compute $result = result * q$;
Print $result$;
end Gauss_Legendre
```

Program 7.5
```
/* Program Gauss-Legendre
   Program to find the integration of a function by 6-point
   Gauss-Legendre method. Here f(x)=x^3. */
#include<stdio.h>
```

```
void main()
{
 float a,b,p,q,result=0,x[7],w[7]; int i;
 float f(float);
 printf("Enter the values of a, b ");
 scanf("%f %f",&a,&b);
 x[1]=0.23861919; x[2]=-x[1];
 x[3]=0.66120939; x[4]=-x[3];
 x[5]=0.93246951; x[6]=-x[5];
 w[1]=w[2]=0.46791393;
 w[3]=w[4]=0.36076157;
 w[5]=w[6]=0.17132449;
 p=(a+b)/2; q=(b-a)/2;
 for(i=1;i<=6;i++)
    result+=w[i]*f(p+q*x[i]);
 result*=q;
 printf("The value of the integration is %f ",result);
}

/* definition of the function f(x) */
float f(float x)
 {
   return(x*x*x);
 }
```

A sample of input/output:

```
Enter the values of a, b 0 1
The value of the integration is 0.250000
```

7.15.2 Lobatto integration methods

Lobatto integration is a Gaussian quadrature with weight function $\psi(x) = 1$ in which the endpoints of the interval $[-1, 1]$ are included in n nodes. The remaining $n-2$ nodes are free to choose. Like Gauss-Legendre methods, the nodes are symmetrical about the origin and the general formula is

$$\int_{-1}^{1} f(x)dx = w_1 f(-1) + w_n f(1) + \sum_{i=2}^{n-1} w_i f(x_i). \tag{7.139}$$

Here, the total number of unknown is $2n-2$ (n weights and $n-2$ nodes), so the formula is exact for polynomial of degree up to $2n-3$.

Table 7.5: Nodes and weights for Lobatto quadrature.

n	node x_i	weight w_i	order of truncation error
3	0.00000000	1.33333333	$f^{(4)}(\xi)$
	$\pm$1.00000000	0.33333333	
4	$\pm$0.44721360	0.83333333	$f^{(6)}(\xi)$
	$\pm$1.00000000	0.16666667	
5	0.00000000	0.71111111	
	$\pm$0.65465367	0.54444444	$f^{(8)}(\xi)$
	$\pm$1.00000000	0.10000000	
6	$\pm$0.28523152	0.55485837	
	$\pm$0.76505532	0.37847496	$f^{(10)}(\xi)$
	$\pm$1.00000000	0.06666667	

For $n = 3$, the formula (7.139) becomes

$$\int_{-1}^{1} f(x)dx = w_1 f(-1) + w_3 f(1) + w_2 f(x_2). \tag{7.140}$$

The formula (7.140) is exact for polynomials of degree up to three. Substituting $f(x) = 1, x, x^2, x^3$ in (7.140) to generate the following system of equations.

$$\begin{aligned} w_1 + w_2 + w_3 &= 2 \\ -w_1 + w_2 x_2 + w_3 &= 0 \\ w_1 + w_2 x_2^2 + w_3 &= \tfrac{2}{3} \\ -w_1 + w_2 x_2^3 + w_3 &= 0. \end{aligned}$$

Solution of this system of equations is

$$x_2 = 0, w_1 = w_3 = \frac{1}{3}, w_2 = \frac{4}{3}.$$

Hence (7.140) becomes

$$\int_{-1}^{1} f(x)dx = \frac{1}{3}[f(-1) + 4f(0) + f(1)]. \tag{7.141}$$

In general the nodes $x_i, i = 2, 3, \ldots, n-1$ are the zeros of the polynomial $P'_{n-1}(x)$, where $P_n(x)$ is a Legendre polynomial of degree n.

The weights are given by

$$\begin{aligned} w_i &= -\frac{2n}{n(n-1)[P_{n-1}(x_i)]^2}, \quad n = 2, 3, \ldots, n-1 \\ w_1 = w_n &= \frac{2}{n(n-1)}. \end{aligned} \tag{7.142}$$

The error term is

$$E = -\frac{n(n-1)^3 2^{2n-1}[(n-2)!]^4}{(2n-1)[(2n-2)!]^3} f^{(2n-2)}(\xi), -1 < \xi < 1. \tag{7.143}$$

The nodes and the corresponding weights for Lobatto integration for $n = 3, 4, 5, 6$ are given in Table 7.5.

7.15.3 Radau integration methods

In Radau method, $(n+1)$ points are needed to fit all polynomials of degree $2n$, so it effectively fits exactly all polynomials of degree $(2n-1)$. It uses the weight function $\psi(x) = 1$ in which the endpoint -1 in the interval $[-1, 1]$ is included out of n nodes and remaining $(n-1)$ nodes are free. The general formula is

$$\int_{-1}^{1} f(x)dx = w_1 f(-1) + \sum_{i=2}^{n} w_i f(x_i). \tag{7.144}$$

Since the formula (7.144) contains $2n-1$ (n weights and $n-1$ nodes) unknowns, this method gives exact values for polynomials of degree up to $2n-2$. For $n = 3$ (7.144) becomes

$$\int_{-1}^{1} f(x)dx = w_1 f(-1) + w_2 f(x_2) + w_3 f(x_3). \tag{7.145}$$

This formula gives exact result for polynomials of degree up to 4. For $f(x) = 1, x, x^2, x^3, x^4$, equation (7.145) generates the following system of equations.

$$\begin{aligned} w_1 + w_2 + w_3 &= 2 \\ -w_1 + w_2 x_2 + w_3 x_3 &= 0 \\ w_1 + w_2 x_2^2 + w_3 x_3^2 &= \tfrac{2}{3} \\ -w_1 + w_2 x_2^3 + w_3 x_3^3 &= 0 \\ w_1 + w_2 x_2^4 + w_3 x_3^4 &= \tfrac{2}{5}. \end{aligned}$$

Solution of these equations is

$$x_2 = \frac{1-\sqrt{6}}{5}, x_3 = \frac{1+\sqrt{6}}{5}, w_1 = \frac{2}{9}, w_2 = \frac{16+\sqrt{6}}{18}, w_3 = \frac{16-\sqrt{6}}{18}.$$

Table 7.6: Nodes and weights for Radau quadrature.

n	node x_i	weight w_i	order of truncation error
2	–1.0000000	0.5000000	$f^{(3)}(\xi)$
	1.3333333	1.5000000	
3	–1.0000000	0.2222222	
	–0.2898979	1.0249717	$f^{(5)}(\xi)$
	0.6898979	0.7528061	
4	–1.0000000	0.1250000	
	–0.5753189	0.6576886	$f^{(7)}(\xi)$
	0.1810663	0.7763870	
	0.8228241	0.4409244	
5	–1.0000000	0.0800000	
	–0.7204803	0.4462078	
	0.1671809	0.6236530	$f^{(9)}(\xi)$
	0.4463140	0.5627120	
	0.8857916	0.2874271	

The formula (7.145) becomes

$$\int_{-1}^{1} f(x)dx = \frac{2}{9}f(-1)+\frac{16+\sqrt{6}}{18}f\left(\frac{1-\sqrt{6}}{5}\right)+\frac{16-\sqrt{6}}{18}f\left(\frac{1+\sqrt{6}}{5}\right). \tag{7.146}$$

In general, the nodes $x_i, i = 2, 3, \ldots, n$ are the zeros of the polynomial

$$\frac{P_{n-1}(x) + P_n(x)}{1+x}, \tag{7.147}$$

where $P_n(x)$ is the Legendre's polynomial.

The weights $w_i, i = 2, 3, \ldots, n$ are given by

$$w_i = \frac{1-x_i}{n^2[P_{n-1}(x_i)]^2} = \frac{1}{(1-x_i)[P'_{n-1}(x_i)]^2} \text{ and}$$
$$w_1 = \frac{2}{n^2}. \tag{7.148}$$

The error term is given by

$$E = \frac{2^{2n-1}n[(n-1)!]^4}{[(2n-1)!]^3}f^{(2n-1)}(\xi), -1 < \xi < 1. \tag{7.149}$$

The nodes and the corresponding weights for the Radau integration methods for $n = 2, 3, 4, 5$ are given in Table 7.6.

7.15.4 Gauss-Chebyshev integration methods

Gauss-Chebyshev quadrature is also known as **Chebyshev quadrature**. Its weight function is taken as $\psi(x) = (1-x^2)^{-1/2}$. The general form of this method is

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)dx = \sum_{i=1}^{n} w_i f(x_i). \tag{7.150}$$

These methods give exact result for polynomials of degree up to $2n-1$. The nodes $x_i, i = 1, 2, \ldots, n$ are the zeros of the Chebyshev polynomials

$$T_n(x) = \cos(n \cos^{-1} x). \tag{7.151}$$

That is,

$$x_i = \cos\left(\frac{(2i-1)\pi}{2n}\right), i = 1, 2, \ldots, n. \tag{7.152}$$

For $n = 3$ the equation (7.150) becomes

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)dx = w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3). \tag{7.153}$$

Since the method gives exact value for the polynomials of degree up to $2n-1$. i.e., up to 5. Therefore, for $f(x) = 1, x, x^2, x^3, x^4, x^5$ the following equations are obtained from (7.153).

$$\begin{aligned}
w_1 + w_2 + w_3 &= \pi \\
w_1 x_1 + w_2 x_2 + w_3 x_3 &= 0 \\
w_1 x_1^2 + w_2 x_2^2 + w_3 x_3^2 &= \tfrac{\pi}{2} \\
w_1 x_1^3 + w_2 x_2^3 + w_3 x_3^3 &= 0 \\
w_1 x_1^4 + w_2 x_2^4 + w_3 x_3^4 &= \tfrac{3\pi}{8} \\
w_1 x_1^5 + w_2 x_2^5 + w_3 x_3^5 &= 0.
\end{aligned}$$

The nodes $x_i, i = 1, 2, 3$ can easily be obtained from the relation

$$x_i = \cos(2i-1)\frac{\pi}{6}, \qquad i = 1, 2, 3.$$

That is, $x_1 = \sqrt{3}/2, x_2 = 0, x_3 = -\sqrt{3}/2$. Then the solution of the above equations is $w_1 = w_2 = w_3 = \pi/3$.

Thus formula (7.153) finally becomes

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)dx = \frac{\pi}{3}\Big[f(\sqrt{3}/2) + f(0) + f(-\sqrt{3}/2)\Big]. \tag{7.154}$$

Table 7.7: Nodes and weights for Gauss-Chebyshev quadrature.

n	node x_i	weight w_i	order of truncation error
2	±0.7071068	1.5707963	$f^{(4)}(\xi)$
3	0.0000000	1.0471976	
	±0.8660254	1.0471976	$f^{(6)}(\xi)$
4	±0.3826834	0.7853982	
	±0.9238795	0.7853982	$f^{(8)}(\xi)$
5	0.0000000	0.6283185	
	±0.5877853	0.6283185	$f^{(10)}(\xi)$
	±0.9510565	0.6283185	

In general, the nodes are given by

$$x_i = \cos\left(\frac{(2i-1)\pi}{2n}\right), \qquad i = 1, 2, \ldots, n$$

and the weights are

$$w_i = -\frac{\pi}{T_{n+1}(x_i)T_n'(x_i)} = \frac{\pi}{n}, \qquad i = 1, 2, \ldots, n. \tag{7.155}$$

The error term is

$$E = \frac{2\pi}{2^{2n}(2n)!}f^{(2n)}(\xi), \qquad -1 < \xi < 1. \tag{7.156}$$

The explicit formula is then

$$\int_{-1}^{1} \frac{f(x)dx}{\sqrt{1-x^2}} = \frac{\pi}{n}\sum_{i=1}^{n} f\left[\cos\left\{\frac{(2i-1)}{2n}\pi\right\}\right] + \frac{2n}{2^{2n}(2n)!}f^{(2n)}(\xi). \tag{7.157}$$

Table 7.7 gives the values for the first few points and weights for Gauss-Chebyshev quadrature.

Example 7.15.2 Find the value of $\int_0^1 \frac{1}{1+x^2}\,dx$ using Gauss-Chebyshev four-point formula.

Solution. Let $f(x) = \frac{\sqrt{1-x^2}}{1+x^2}$. Here $x_1 = -0.3826834 = x_2$, $x_3 = -0.9238795 = -x_4$ and $w_1 = w_2 = w_3 = w_4 = 0.7853982$.

Then

$$\begin{aligned}
I &= \int_0^1 \frac{1}{1+x^2}dx = \frac{1}{2}\int_{-1}^1 \frac{1}{1+x^2}dx = \frac{1}{2}\int_{-1}^1 \frac{f(x)}{\sqrt{1-x^2}}dx \\
&= \frac{1}{2}[w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3) + w_4 f(x_4)] \\
&= \frac{1}{2} \times 0.7853982[f(x_1) + f(x_2) + f(x_3) + f(x_4)] \\
&= \frac{1}{2} \times 0.7853982[2 \times 0.8058636 + 2 \times 0.2064594] = 0.7950767,
\end{aligned}$$

while the exact value is $\pi/4 = 0.7853982$. The absolute error is 0.0096785.

Remark 7.15.1 *It may be noted that the Gauss-Legendre four-point formula gives better result for this problem.*

7.15.5 Gauss-Hermite integration methods

Gauss-Hermite quadrature, also known as **Hermite quadrature** is a Gaussian quadrature over the interval $(-\infty, \infty)$ with weight function $\psi(x) = e^{-x^2}$. The nodes x_i are the zeros of the Hermite polynomial $H_n(x)$, where

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}(e^{-x^2}). \tag{7.158}$$

The zeros are symmetrical about the origin.

These polynomials satisfy the recurrence relation

$$H_n'(x) = 2nH_{n-1}(x) = 2xH_n(x) - H_{n+1}(x).$$

The first few Hermite polynomials are

$$H_0(x) = 1, H_1(x) = 2x, H_2(x) = 2(2x^2 - 1), H_3(x) = 4(2x^3 - 3x).$$

The weights w_i are given by

$$w_i = \frac{2^{n+1} n! \sqrt{\pi}}{[H_{n+1}'(x_i)]^2} = \frac{2^{n-1} n! \sqrt{\pi}}{n^2 [H_{n-1}(x_i)]^2}. \tag{7.159}$$

The Gauss-Hermite formula is

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx = \sum_{i=1}^{n} w_i f(x_i). \tag{7.160}$$

The error term is

$$E = \frac{n! \sqrt{\pi}}{2^n (2n)!} f^{(2n)}(\xi). \tag{7.161}$$

Table 7.8 gives the nodes and weights for Gauss-Hermite integration methods.

Table 7.8: Nodes and weights for Gauss-Hermite quadrature.

n	node x_i	weight w_i	order of truncation error
2	±0.7071068	0.8862269	$f^{(4)}(\xi)$
3	$+0.0000000$	1.1816359	
	±1.2247449	0.2954090	$f^{(6)}(\xi)$
4	±0.5246476	0.8049141	
	±1.6506801	0.0813128	$f^{(8)}(\xi)$
5	$+0.0000000$	0.945309	
	±0.958572	0.393619	$f^{(10)}(\xi)$
	±2.02018	0.0199532	

7.15.6 Gauss-Laguerre integration methods

Gauss-Laguerre quadrature is also known as **Laguerre quadrature** and it is a Gaussian quadrature over the interval $[0, \infty)$ with weight function $\psi(x) = e^{-x}$. The general form is

$$\int_0^\infty e^{-x} f(x)dx = \sum_{i=1}^{n} w_i f(x_i). \tag{7.162}$$

The nodes x_i are the zeros of the Laguerre polynomial

$$L_n(x) = (-1)^n e^x \frac{d^n}{dx^n}(e^{-x}x^n), \tag{7.163}$$

which satisfies the recurrence relation

$$xL'_n(x) = nL_n(x) - nL_{n-1}(x). \tag{7.164}$$

Few lower order Laguerre polynomials are

$$L_0(x) = 1, L_1(x) = x - 1, L_2(x) = x^2 - 4x + 2, L_3(x) = x^3 - 9x^2 + 18x - 6.$$

The weights w_i are given by

$$w_i = \frac{1}{x_i[L'_n(x_i)]^2} = \frac{x_i}{(n+1)^2[L_{n+1}(x_i)]^2}. \tag{7.165}$$

The error term is

$$E = \frac{(n!)^2}{(2n)!} f^{(2n)}(\xi), 0 < \xi < \infty. \tag{7.166}$$

Table 7.9 gives the nodes and corresponding weights of Gauss-Laguerre integration.

Table 7.9: Nodes and weights for Gauss-Laguerre quadrature.

n	node x_i	weight w_i	order of truncation error
2	0.5857864376	0.8535533906	$f^{(4)}(\xi)$
	3.4142135624	0.1464466094	
3	0.4157745568	0.7110930099	
	2.2942803603	0.2785177336	$f^{(6)}(\xi)$
	6.2899450829	0.0103892565	
4	0.3225476896	0.6031541043	
	1.7457611012	0.3574186924	$f^{(8)}(\xi)$
	4.5366202969	0.0388879085	
	9.3950709123	0.0005392947	
5	0.2635603197	0.5217556106	
	1.4134030591	0.3986668111	
	3.5964257710	0.0759424497	$f^{(10)}(\xi)$
	7.0858100059	0.0036117587	
	12.6408008443	0.0000233700	

7.15.7 Gauss-Jacobi integration methods

This quadrature is also called **Jacobi quadrature** or **Mehler quadrature**. It is a Gaussian quadrature over $[-1, 1]$ with weight function

$$\psi(x) = (1-x)^{\alpha}(1+x)^{\beta}. \tag{7.167}$$

The general form of the formula is

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} w_i f(x_i), \tag{7.168}$$

where the nodes x_i's are the zeros of the Jacobi polynomial

$$P_n^{(\alpha,\beta)}(x) = \frac{(-1)^n}{2^n n!}(1-x)^{-\alpha}(1+x)^{-\beta}\frac{d^n}{dx^n}[(1-x)^{\alpha+n}(1+x)^{\beta+n}],$$
$$\alpha, \beta > -1. \tag{7.169}$$

$P_n^{(0,0)}(x)$ is the Legendre polynomial.

The weights are

$$w_i = \frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{\Gamma(n+\alpha+\beta+1)}\frac{2^{2n+\alpha+\beta+1}n!}{(1-x_i^2)[V_n'(x_i)]^2}, i = 1, 2, \ldots, n, \tag{7.170}$$

where $V_n = (-1)^n P_n^{(\alpha,\beta)}(x).2^n n!$.

The error term is

$$E = \frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)\Gamma(n+\alpha+\beta+1)}{(2n+\alpha+\beta+1)[\Gamma(2n+\alpha+\beta+1)]^2}.\frac{2^{2n+\alpha+\beta+1}}{(2n)!}f^{(2n)}(\xi),$$
$$-1 < \xi < 1. \tag{7.171}$$

Table 7.10 is a list of some Gaussian quadrature along with weights, nodes and the corresponding intervals.

Table 7.10: Summary of Gaussian quadrature.

	$\psi(x)$	interval	x_i are roots of
Gauss-Legendre	1	$(-1,1)$	$P_n(x)$
Lobatto	1	$(-1,1)$	$P'_{n-1}(x)$
Radau	1	$(-1,1)$	$\frac{P_{n-1}(x)+P_n(x)}{1+x}$
Gauss-Chebyshev	$(1-x^2)^{-1/2}$	$(-1,1)$	$T_n(x)$
Gauss-Hermite	e^{-x^2}	$(-\infty,\infty)$	$H_n(x)$
Gauss-Laguerre	e^{-x}	$(0,\infty)$	$L_n(x)$
Gauss-Jacobi	$(1-x)^\alpha(1+x)^\beta$	$(-1,1)$	$P_n^{(\alpha,\beta)}(x)$

Theorem 7.1 *If $x_i, i = 1, 2, \ldots, n$ are the zeros of an orthogonal polynomial, orthogonal with respect to the weight function $\psi(x)$ over the interval (a, b), then the degree of precision of the formula*

$$\int_a^b f(x)\psi(x)dx = \sum_{i=1}^n w_i f(x_i) \tag{7.172}$$

is $(2n-1)$ and each of the weights w_i is positive.

Proof. We define an inner product with respect to the weight function $\psi(x)$ as

$$\langle f, g\rangle = \int_a^b f(x)g(x)\psi(x)dx. \tag{7.173}$$

A set of orthogonal polynomials $p_n(x)$ satisfies the result

$$\langle p_m, p_n\rangle = \begin{cases} 1, m = n \\ 0, m \neq n. \end{cases} \tag{7.174}$$

Our aim is to find the nodes and the weights such that

$$\int_a^b f(x)\psi(x)dx = \sum_{i=1}^{n} w_i f(x_i) + E \tag{7.175}$$

is exact ($E = 0$) if $f(x)$ is a polynomial of degree $2n - 1$ or less.

Let $f(x)$ be a polynomial of degree $2n - 1$. If $f(x)$ is divided by the orthogonal polynomial $p_n(x)$ then $f(x)$ can be expressed as

$$f(x) = p_n(x)Q_{n-1}(x) + R(x) \tag{7.176}$$

where $Q_{n-1}(x)$ is the quotient and $R(x)$ is the remainder polynomial of degree $n - 1$. Now, multiplying the equation (7.176) by the weight function $\psi(x)$ and integrating it over $[a, b]$, we get

$$\int_a^b f(x)\psi(x)dx = \int_a^b p_n(x)Q_{n-1}(x)\psi(x)dx + \int_a^b R(x)\psi(x)dx. \tag{7.177}$$

The first term of right hand side is zero, since $Q_{n-1}(x)$ can be expressed as a linear combination of the orthogonal polynomial $p_0, p_1, \ldots, p_{n-1}$ and so it must be orthogonal to p_n. Then from (7.177),

$$\int_a^b f(x)\psi(x)dx = \int_a^b R(x)\psi(x)dx. \tag{7.178}$$

Now, let the nodes x_i be the n zeros of the polynomial $p_n(x)$. Then from (7.176) $f(x_i) = R(x_i)$. We now introduce another special set of polynomials, the Lagrange's polynomials

$$L_i(x) = \prod_{\substack{k=1\\k\neq i}}^{n} \frac{x - x_k}{x_i - x_k}, \tag{7.179}$$

where $L_i(x)$ satisfies the following result.

$$L_i(x_k) = \begin{cases} 1, \, i = k \\ 0, \, i \neq k. \end{cases}$$

Since $R(x)$ is of degree $(n - 1)$ it can be written as a sum of Lagrange's polynomials, i.e.,

$$R(x) = \sum_{i=1}^{n} f(x_i)L_i(x). \tag{7.180}$$

Then from (7.178),

$$\int_a^b f(x)\psi(x)dx = \int_a^b \sum_{i=1}^n f(x_i)L_i(x)\psi(x)dx$$
$$= \sum_{i=1}^n f(x_i)\int_a^b L_i(x)\psi(x)dx = \sum_{i=1}^n f(x_i)w_i, \tag{7.181}$$

where $w_i = \displaystyle\int_a^b \psi(x)L_i(x)dx$.

This proves that the formula (7.172) has precision $2n-1$.

Note that $L_i^2(x)$ is a polynomial of degree less than or equal to $2n$. Let $f(x) = L_i^2(x)$. Then (7.181) reduces to

$$\int_a^b \psi(x)L_j^2(x)dx = \sum_{i=1}^n w_i L_j^2(x_i).$$

Therefore,

$$w_i = \int_a^b \psi(x)L_i^2(x)dx = \langle L_i, L_i\rangle > 0$$

using (7.173) and (7.174).

7.16 Euler-Maclaurin's Sum Formula

Euler-Maclaurin's sum formula is used to determine the sum of finite and infinite series of numbers and is also used to numerical quadrature, if the values of derivatives are known at the end points of the interval. The well-known quadrature formulae such as trapezoidal rule, Simpson's rule including error terms can be deduced from this formula.

This formula is also known as **Maclaurin's sum formula**. To deduce this formula, let us consider the expansion of $\dfrac{x}{e^x-1}$ in ascending powers of x.

Let $\dfrac{x}{e^x-1} = b_0 + b_1x + b_2x^2 + b_3x^3 + b_4x^4 + b_5x^5 + \cdots$.

That is,

$$x = (b_0 + b_1x + b_2x^2 + b_3x^3 + b_4x^4 + b_5x^5 + \cdots)\Big(x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots\Big)$$
$$= b_0x + x^2\Big(\frac{b_0}{2!} + b_1\Big) + x^3\Big(\frac{b_0}{3!} + \frac{b_1}{2!} + b_2\Big) + x^4\Big(\frac{b_0}{4!} + \frac{b_1}{3!} + \frac{b_2}{2!} + b_3\Big)$$
$$+x^5\Big(\frac{b_0}{5!} + \frac{b_1}{4!} + \frac{b_2}{3!} + \frac{b_3}{2!} + b_4\Big) + x^6\Big(\frac{b_0}{6!} + \frac{b_1}{5!} + \frac{b_2}{4!} + \frac{b_3}{3!} + \frac{b_4}{2!} + b_5\Big) + \cdots .$$

Equating the like powers of x on both sides and we find the following relations.

$$\begin{aligned}
&b_0 = 1\\
&b_1 = -\frac{b_0}{2!} = -\frac{1}{2}\\
&b_2 = -\frac{b_0}{3!} - \frac{b_1}{2!} = \frac{1}{12}\\
&b_3 = -\frac{b_0}{4!} - \frac{b_1}{3!} - \frac{b_2}{2!} = 0\\
&b_4 = -\frac{b_0}{5!} - \frac{b_1}{4!} - \frac{b_2}{3!} - \frac{b_3}{2!} = -\frac{1}{720}\\
&b_5 = -\frac{b_0}{6!} - \frac{b_1}{5!} - \frac{b_2}{4!} - \frac{b_3}{3!} - \frac{b_4}{2!} = 0\\
&b_6 = -\frac{b_0}{7!} - \frac{b_1}{6!} - \frac{b_2}{5!} - \frac{b_3}{4!} - \frac{b_4}{3!} - \frac{b_5}{2!} = \frac{1}{30240}\\
&\text{etc.}
\end{aligned}$$

Hence,

$$\begin{aligned}
\frac{x}{e^x - 1} &= 1 - \frac{1}{2}x + \frac{1}{12}x^2 - \frac{1}{720}x^4 + \frac{1}{30240}x^6 - \cdots\\
\frac{1}{e^x - 1} &= \frac{1}{x} - \frac{1}{2} + \frac{1}{12}x - \frac{1}{720}x^3 + \frac{1}{30240}x^5 - \cdots . \qquad (7.182)
\end{aligned}$$

From the above expansion one can write

$$\begin{aligned}
\frac{1}{E-1} &= \frac{1}{e^{hD} - 1} \qquad (\text{since } E = e^{hD})\\
&= \frac{1}{hD} - \frac{1}{2} + \frac{1}{12}(hD) - \frac{1}{720}(hD)^3 + \frac{1}{30240}(hD)^5 - \cdots .
\end{aligned}$$

Note that

$$(E^n - 1)f(x_0) = f(x_0 + nh) - f(x_0) = f(x_n) - f(x_0).$$

Now,

$$\begin{aligned}
&\frac{E^n - 1}{E - 1} f(x_0)\\
&= \left[\frac{1}{hD} - \frac{1}{2} + \frac{1}{12}(hD) - \frac{1}{720}(hD)^3 + \frac{1}{30240}(hD)^5 - \cdots\right](E^n - 1)f(x_0).
\end{aligned}$$

That is,

$$\begin{aligned}
&(E^{n-1} + E^{n-2} + \cdots + E + 1)f(x_0)\\
&= \frac{1}{hD}[f(x_n) - f(x_0)] - \frac{1}{2}[f(x_n) - f(x_0)] + \frac{h}{12}[f'(x_n) - f'(x_0)]\\
&\quad - \frac{h^3}{720}[f'''(x_n) - f'''(x_0)] + \frac{h^5}{30240}[f^v(x_n) - f^v(x_0)] + \cdots . \qquad (7.183)
\end{aligned}$$

The left hand side of (7.183) is

$$\sum_{r=0}^{n-1} E^r f(x_0) = \sum_{r=0}^{n-1} f(x_0+rh) = \sum_{r=0}^{n-1} f(x_r) = \sum_{r=0}^{n} f(x_r) - f(x_n).$$

Also, $\dfrac{1}{hD}[f(x_n)-f(x_0)] = \dfrac{1}{h}\displaystyle\int_{x_0}^{x_n} f(x)dx.$
Hence (7.183) becomes

$$\begin{aligned}&\sum_{r=0}^{n} f(x_r) - f(x_n)\\ &= \frac{1}{h}\int_{x_0}^{x_n} f(x)dx - \frac{1}{2}[f(x_n)-f(x_0)] + \frac{h}{12}[f'(x_n)-f'(x_0)]\\ &\quad -\frac{h^3}{720}[f'''(x_n)-f'''(x_0)] + \frac{h^5}{30240}[f^v(x_n)-f^v(x_0)] + \cdots\end{aligned}$$

That is,

$$\begin{aligned}\sum_{r=0}^{n} f(x_r) = \frac{1}{h}\int_{x_0}^{x_n} f(x)dx &+ \frac{1}{2}[f(x_n)+f(x_0)]\\ &+\frac{h}{12}[f'(x_n)-f'(x_0)] - \frac{h^3}{720}[f'''(x_n)-f'''(x_0)]\\ &+\frac{h^5}{30240}[f^v(x_n)-f^v(x_0)] + \cdots .\end{aligned}\tag{7.184}$$

This formula may also be used as a quadrature formula when it is written as

$$\begin{aligned}\int_{x_0}^{x_n} f(x)dx = h\Big[\frac{1}{2}f(x_0) + f(x_1) + \cdots + f(x_{n-1}) + \frac{1}{2}f(x_n)\Big]&\\ -\frac{h^2}{12}[f'(x_n)-f'(x_0)] + \frac{h^4}{720}[f'''(x_n)-f'''(x_0)]&\\ -\frac{h^6}{30240}[f^v(x_n)-f^v(x_0)] + \cdots .&\end{aligned}\tag{7.185}$$

Example 7.16.1 Find the sum of the series

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{100}$$

correct to six decimal places.

Solution. We have to determine the value of $\sum\limits_{x=1}^{100}\frac{1}{x}$.

Let $f(x)=\frac{1}{x}$ and $h=1, x_0=1, x_n=100$.

Then $f'(x)=-\frac{1}{x^2}, f''(x)=\frac{2}{x^3}, f'''(x)=-\frac{6}{x^4}, f^{iv}(x)=\frac{24}{x^5}, f^{v}(x)=-\frac{120}{x^6}$.

Now, $\int_{x_0}^{x_n} f(x)dx=\int_1^{100}\frac{1}{x}dx=\log 100=4.6051702$.

$f(x_n)+f(x_0)=\frac{1}{100}+1=1.01$,

$f'(x_n)-f'(x_0)=-\frac{1}{100^2}+1=0.9999$,

$f'''(x_n)-f'''(x_0)=-\frac{6}{100^4}+6=5.999999$,

$f^v(x_n)-f^v(x_0)=-\frac{120}{100^6}+120=120$.

Hence

$$\begin{aligned}\sum_{x=1}^{100} f(x) &= \frac{1}{h}\int_1^{100} f(x)dx+\frac{1}{2}[f(x_n)+f(x_0)]\\ &\quad+\frac{h}{12}[f'(x_n)-f'(x_0)]-\frac{h^3}{720}[f'''(x_n)-f'''(x_0)]\\ &\quad+\frac{h^5}{30240}[f^v(x_n)-f^v(x_0)]+\cdots\\ &=4.6051702+\frac{1}{2}\times 1.01+\frac{1}{12}\times 0.9999-\frac{1}{720}\times 5.999999+\frac{1}{30240}\times 120\\ &=5.1891301.\end{aligned}$$

Hence, the approximate value of the given sum is 5.189130 correct up to six decimal places.

Example 7.16.2 Using the relation $\frac{\pi^2}{6}=\sum\limits_{x=1}^{\infty}\frac{1}{x^2}$, compute the value of π^2 correct to six decimals.

Solution. Let $f(x)=\frac{1}{x^2}$. Here $h=1, x_0=1, x_n=\infty$.

Then $f'(x)=-\frac{2}{x^3}, f''(x)=\frac{6}{x^4}, f'''(x)=-\frac{24}{x^5}, f^{iv}(x)=\frac{120}{x^6}, f^v(x)=-\frac{720}{x^7}$.

Now, $\int_{x_0}^{x_n} f(x)dx=\int_1^{\infty}\frac{1}{x^2}dx=1$.

$f(x_n)+f(x_0)=f(\infty)+f(1)=1$. $f'(x_n)-f'(x_0)=f'(\infty)-f'(1)=2$,

$f'''(x_n) - f'''(x_0) = 24$,
$f^v(x_n) - f^v(x_0) = 720$.
Hence

$$\begin{aligned}\sum_{x=1}^{\infty} f(x) &= \frac{1}{h}\int_1^{\infty} f(x)dx + \frac{1}{2}[f(x_n) + f(x_0)]\\ &\quad + \frac{h}{12}[f'(x_n) - f'(x_0)] - \frac{h^3}{720}[f'''(x_n) - f'''(x_0)]\\ &\quad + \frac{h^5}{30240}[f^v(x_n) - f^v(x_0)] + \cdots\\ &= 1 + \frac{1}{2}\times 1 + \frac{1}{12}\times 2 - \frac{1}{720}\times 24 + \frac{1}{30240}\times 720\\ &= 1.6571429.\end{aligned}$$

Thus $\pi^2/6 = 1.6571429$ or, $\pi^2 = 9.9428574$. Hence the approximate value of π^2 is 9.942857 correct up to six decimal places.

Example 7.16.3 Find the sum of the series

$$\frac{1}{51^2} + \frac{1}{53^2} + \frac{1}{55^2} + \cdots + \frac{1}{99^2} + \frac{1}{100^2}.$$

Solution. Firstly, we find the sum of $\frac{1}{51^2} + \frac{1}{53^2} + \frac{1}{55^2} + \cdots + \frac{1}{99^2}$.
Let $f(x) = \frac{1}{x^2}$. Here $h = 2, x_0 = 51, x_n = 99$.
$f'(x) = -\frac{2}{x^3}, f'''(x) = -\frac{24}{x^5}$.
$\int_{x_0}^{x_n} f(x)dx = \int_{51}^{99} \frac{1}{x^2}dx = -\frac{1}{99} + \frac{1}{51} = 0.0095068$.
$f(x_n) + f(x_0) = \frac{1}{99^2} + \frac{1}{51^2} = 0.0004865$,
$f'(x_n) - f'(x_0) = -\frac{2}{99^3} + \frac{2}{51^3} = 0.000013$,
$f'''(x_n) - f'''(x_0) = -\frac{24}{99^5} + \frac{24}{51^5} = 0$.
Now,

$$\begin{aligned}\sum_{x=51}^{99} f(x) &= \frac{1}{h}\int_{x_0}^{x_n} f(x)dx + \frac{1}{2}[f(x_n) + f(x_0)] + \frac{h}{12}[f'(x_n) - f'(x_0)] - \cdots\\ &= \frac{1}{2}\times 0.0095068 + \frac{1}{2}\times 0.0004865 + \frac{2}{12}\times 0.0000130 = 0.0049988.\end{aligned}$$

Hence the required sum is $0.0049988 + \frac{1}{100^2} = 0.0050988$.

Example 7.16.4 Find the value of $\int_1^5 \frac{dx}{1+x}$ using Maclaurin's sum formula.

Solution. The Maclaurin's sum formula is

$$\int_{x_0}^{x_n} f(x)dx = h\sum_{x=x_0}^{x_n} f(x) - \frac{h}{2}[f(x_n)+f(x_0)] - \frac{h^2}{12}[f'(x_n)-f'(x_0)]$$
$$+\frac{h^4}{720}[f'''(x_n)-f'''(x_0)] - \cdots .$$

Let $f(x) = \frac{1}{1+x}, x_0 = 1, x_n = 5$ and $h = 1$.
$f'(x) = -\frac{1}{(1+x)^2}, f''(x) = \frac{2}{(1+x)^3}, f'''(x) = -\frac{6}{(1+x)^4}$.
$f(x_n)+f(x_0) = \frac{1}{6}+\frac{1}{2} = 0.6666667,\quad f'(x_n)-f'(x_0) = -\frac{1}{6^2}+\frac{1}{2^2} = 0.2222222,$
$f'''(x_n)-f'''(x_0) = -\frac{6}{6^4}+\frac{6}{2^4} = 0.3703704.$

Also, $\sum_{x=x_0}^{x_n} f(x) = f(1.0)+f(2.0)+f(3.0)+f(4.0)+f(5.0)$
$$= \frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6} = 1.45.$$

Hence

$$\int_1^5 \frac{dx}{1+x} = 1\times 1.45 - \frac{1}{2}\times 0.6666667 - \frac{1}{12}\times 0.2222222 + \frac{1}{720}\times 0.3703704$$
$$= 1.0986625.$$

The exact value is $\log 6 - \log 2 = 1.0986123$. Thus the absolute error is 0.00005020.

Example 7.16.5 Deduce trapezoidal and Simpson's 1/3 rules from Maclaurin's sum formula.

Solution. From (7.185),

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{2}[f(x_0)+2\{f(x_1)+f(x_2)+\cdots+f(x_{n-1})\}+f(x_n)]$$
$$-\frac{h^2}{12}[f'(x_n)-f'(x_0)] + \frac{h^4}{720}[f'''(x_n)-f'''(x_0)] - \cdots$$
$$= I_T + E_T, \tag{7.186}$$

where $I_T = \frac{h}{2}[f(x_0) + 2\{f(x_1) + f(x_2) + \cdots + f(x_{n-1})\} + f(x_n)]$ is the composite trapezoidal rule and

$$\begin{aligned}E_T &\simeq -\frac{h^2}{12}[f'(x_n) - f'(x_0)] \simeq -\frac{h^2}{12}(x_n - x_0)f''(\xi_1), x_0 < \xi_1 < x_n\\ &\quad \text{[by MVT of differential calculus]}\\ &\simeq -\frac{nh^3}{12}f''(\xi_1).\end{aligned}$$

Thus $\int_{x_0}^{x_n} f(x)dx = I_T + E_T$ is the composite trapezoidal rule.
To deduce Simpson's 1/3 rule, taking n as even and replacing h by $2h$ in (7.185), we have

$$\begin{aligned}\int_{x_0}^{x_n} f(x)dx &= 2h\Big[\frac{1}{2}f(x_0) + f(x_2) + f(x_4) + \cdots + f(x_{n-2}) + \frac{1}{2}f(x_n)\Big]\\ &\quad -\frac{2^2h^2}{12}[f'(x_n) - f'(x_0)] + \frac{2^4h^4}{720}[f'''(x_n) - f'''(x_0)] - \cdots .\end{aligned} \tag{7.187}$$

Using the expression $[(7.186) \times 4 - (7.187)]/3$, we obtain

$$\begin{aligned}\int_{x_0}^{x_n} f(x)dx &= \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + f(x_n)]\\ &\quad +\frac{1}{3}(4 - 2^4)\frac{h^4}{720}[f'''(x_n) - f'''(x_0)] - \cdots\\ &= I_S + E_S,\end{aligned}$$

where $I_S = \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + f(x_n)]$
is the composite Simpson's 1/3 rule and

$$\begin{aligned}E_S &= -\frac{h^4}{180}[f'''(x_n) - f'''(x_0)] - \cdots\\ &\simeq -\frac{h^4}{180}(x_n - x_0)f^{iv}(\xi_2) \qquad \text{[by MVT of differential calculus]}\\ &= -\frac{nh^5}{90}f^{iv}(\xi_2), [\text{ since } (x_n - x_0)/n = 2h], x_0 < \xi_2 < x_n\end{aligned}$$

is the error term.

7.17 Romberg's Integration

The value of the integration obtained from trapezoidal or Simpson's rules or other rules may be improved by repeated use of Richardson's extrapolation procedure as described in Section 7.7. This method is known as **Romberg's integration** method as he was the first one to describe the algorithm in recursive form.

We assume that $f \in C^n[a,b]$ for all n, then the error term for the trapezoidal rule can be represented in a series involving only even powers of h, i.e.,

$$I = \int_a^b f(x)dx = I_0(h) + c_1h^2 + c_2h^4 + c_3h^6 + \cdots . \tag{7.188}$$

The Richardson's extrapolation method is used successively to eliminate c_1, c_2, c_3 and so on. This process generates integration formulae whose error terms are of the even orders $O(h^4), O(h^6), O(h^8)$, etc.

The step length h is replaced by $h/2$ in (7.188) then

$$I = I_0(h/2) + c_1\frac{h^2}{4} + c_2\frac{h^4}{16} + c_3\frac{h^6}{64} + \cdots \tag{7.189}$$

To eliminate c_1, equation (7.189) is multiplied by 4 and this equation is subtracted from (7.188). Thus

$$3I = 4I_0(h/2) - I_0(h) - \frac{3}{4}c_2h^4 - \frac{15}{16}c_4h^6 - \cdots .$$

This equation is divided by 3 and the coefficients of h^4, h^6 etc. are denoted by d_1, d_2 etc. Thus

$$I = \frac{4I_0(h/2) - I_0(h)}{3} + d_1h^4 + d_2h^6 + \cdots \tag{7.190}$$

Denoting $\frac{4I_0(h/2) - I_0(h)}{3}$ by $I_1(h/2)$ and this is the first Romberg's improvement. Thus by this notation the equation (7.190) can be written as

$$I = I_1(h/2) + d_1h^4 + d_2h^6 + \cdots . \tag{7.191}$$

Again, h is replaced by $h/2$ in (7.191), therefore,

$$I = I_1(h/2^2) + d_1\frac{h^4}{16} + d_2\frac{h^6}{64} + \cdots . \tag{7.192}$$

Now, eliminating d_1, we obtain

$$15I = 16I_1(h/2^2) - I_1(h/2) - \frac{3}{4}d_2h^6 + \cdots$$

That is,

$$\begin{aligned} I &= \frac{16I_1(h/2^2) - I_1(h/2)}{15} + e_1h^6 + \cdots \\ &= I_2(h/2^2) + e_1h^6 + \cdots, \end{aligned} \tag{7.193}$$

where

$$I_2(h/2^2) = \frac{16I_1(h/2^2) - I_1(h/2)}{15} = \frac{4^2I_1(h/2^2) - I_1(h/2)}{4^2 - 1} \tag{7.194}$$

is the second Romberg's improvement.

In general,

$$I_m(h/2^k) = \frac{4^kI_{m-1}(h/2^k) - I_{m-1}(h/2^{k-1})}{4^k - 1}, \tag{7.195}$$

where $m = 1, 2, \ldots; k = m, m+1, \ldots$ and $I_0(h) = I(h)$.

Thus

$$I = I_m(h/2^m) + O(h^{2m+2}). \tag{7.196}$$

Now,

$$\begin{aligned} I_1(h/2) &= \frac{4I(h/2) - I(h)}{3} = \frac{1}{3}\Big[4.\frac{h}{4}(y_0 + 2y_1 + y_2) - \frac{h}{2}(y_0 + y_2)\Big] \\ &= \frac{h/2}{3}[y_0 + 4y_1 + y_2]. \end{aligned}$$

This is the Simpson's 1/3 rule with step size $h/2$. That is, the first improved value is equal to the value obtained by Simpson's 1/3 rule.

Now,

$$\begin{aligned} I_2(h/2^2) &= \frac{16I_1(h/2^2) - I_1(h/2)}{15} \\ &= \frac{1}{15}\Big[16.\frac{h/4}{3}\Big\{y_0 + 4(y_1 + y_3) + 2y_2 + y_4\Big\} - \frac{h/2}{3}(y_0 + 4y_2 + y_4)\Big] \\ &\quad \text{[Simpson's rule for 4 and 2 intervals]} \\ &= \frac{2(h/4)}{45}[7y_0 + 32y_1 + 12y_2 + 32y_3 + 7y_4]. \end{aligned}$$

This is the Boole's rule for step length $h/4$.

The Romberg's integration can be carried out using the triangular array of successive approximations to the integral as shown in Table 7.11.

Table 7.11: Romberg's integration table.

$$\begin{tikzcd}
n & h & \text{0th order trapezoidal} & \text{1st order Simpson} & \text{2nd order Boole} & \text{3rd order} & \text{4th order} \\
1 & h & I_0(h) \arrow[dr] & & & & \\
2 & h/2 & I_0(h/2) \arrow[r] \arrow[dr] & I_1(h/2) \arrow[dr] & & & \\
4 & h/2^2 & I_0(h/2^2) \arrow[r] \arrow[dr] & I_1(h/2^2) \arrow[r] \arrow[dr] & I_2(h/2^2) \arrow[dr] & & \\
8 & h/2^3 & I_0(h/2^3) \arrow[r] \arrow[dr] & I_1(h/2^3) \arrow[r] \arrow[dr] & I_2(h/2^3) \arrow[r] \arrow[dr] & I_3(h/2^3) \arrow[dr] & \\
16 & h/2^4 & I_0(h/2^4) \arrow[r] & I_1(h/2^4) \arrow[r] & I_2(h/2^4) \arrow[r] & I_3(h/2^4) \arrow[r] & I_4(h/2^4)
\end{tikzcd}$$

The entries in the 0th order are computed directly and the other entries are calculated by using the formula (7.195). The values along the diagonal would converge to the integral.

The advantage of Romberg's method is that the method gives much more accurate result than the usual composite trapezoidal rule. A computational weakness of this method is that twice as many function evaluations are needed to decrease the error from $O(h^{2n})$ to $O(h^{2n+2})$. Practically, the computations are carried out rowwise until the desired accuracy is achieved.

Note 7.17.1 If the 0th order (starting) approximation is calculated using Simpson's rule then first order approximation I_1 gives the Boole's approximation and so on.

Example 7.17.1 Find the value of $\displaystyle\int_0^1 \frac{dx}{1+x^2}$ using Romberg's method starting with trapezoidal rule.

Solution. Let $x_0 = 0, x_1 = 1, h = \frac{1-0}{n}, x_i = x_0 + ih$.
The initial approximations are computed by trapezoidal rule.
$n = 1, h = 1$. Then $I_0 = \dfrac{h}{2}(y_0 + y_1) = 0.5(1 + 0.5) = 0.75000$.

$n = 2, h = 0.5$. $I_0 = \dfrac{h}{2}(y_0 + 2y_1 + y_2) = 0.25(1 + 2 \times 0.8 + 0.5) = 0.775$.

$n = 4, h = 0.25$. $I_0 = \dfrac{h}{2}[y_0 + 2(y_1 + y_2 + y_3) + y_4]$
$= 0.125(1 + 2(0.94118 + 0.8 + 0.64) + 0.5) = 0.7828$.

$n = 8, h = 0.125$. $I_0 = \dfrac{h}{2}[y_0 + 2(y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7) + y_8]$
$= 0.0625[1 + 2(0.98462 + 0.94118 + 0.87671 + 0.8 + 0.7191$

$+0.64 + 0.56637) + 0.5] = 0.78475.$

$$I_1(h/2) = \frac{4I_0(0.5) - I_0(1.0)}{3} = \frac{4 \times 0.77500 - 0.75000}{3} = 0.78333$$
$$I_1(h/2^2) = \frac{4I_0(0.25) - I_0(0.5)}{3} = \frac{4 \times 0.78280 - 0.77500}{3} = 0.78540$$
$$I_1(h/2^3) = \frac{4I_0(0.125) - I_0(0.25)}{3} = \frac{4 \times 0.78475 - 0.78280}{3} = 0.78540.$$

All the calculations are shown in the following table.

n	h	I_0	I_1	I_2	I_3
1	1.000	0.75000			
2	0.500	0.77500	0.78333		
4	0.250	0.78280	0.78540	0.78554	
8	0.125	0.78475	0.78540	0.78540	0.78540

The exact integral is $\pi/4 = 0.78540$. In each column the numbers are converging to the value 0.78540. The values in Simpson's rule column (I_1) converge faster than the values in the trapezoidal rule column (I_0). Ultimately, the value of the integral is 0.78540 correct up to five decimal places.

Algorithm 7.6 (Romberg's integration). This algorithm implements Romberg's integration starting with trapezoidal rule.

From Table 7.11 it is observed that the elements of the row j are

$$I(j,k) = \frac{4^k I(j,k-1) - I(j-1,k-1)}{4^k - 1} = I(j,k-1) + \frac{I(j,k-1) - I(j-1,k-1)}{4^k - 1},$$

$1 \le k \le j$.
$I(j,0)$ are obtained from trapezoidal rule. The algorithm will terminate in the jth row when $|I(i,j) - I(i-1,j-1)| < \varepsilon$.

```
Algorithm Romberg_Integration
Input function $F(x)$;
Read $a, b, eps, maxrow$;
// $a, b$ are lower and upper limits of the integral, $eps$ is the error tolerance and $maxrow$ is maximum number of rows to be evaluated if error tolerance is not archived.//
Real $I(0 : maxrow, 0 : maxrow)$; //value of the integral.//
Set $n = 1$; //initialize the number of subintervals//
Set $h = b - a$; // step size and current row.//
Set $j = 0$;
Set $error = 1$; //initialize error to 1.//
Compute $I(0,0) = h * [F(a) + F(b)]/2$;
while ($error > eps$) and ($j < maxrow$) do
    $j = j + 1, h = h/2, n = 2 * n$;
    //compute initial value using trapezoidal rule.//
    Compute $I(j,0)$=TRAP$(a,h,n)$
    for $k = 1$ to $j$ do
        $I(j,k) = I(j,k-1) + [I(j,k-1) - I(j-1,k-1)]/(4^k - 1)$;
    endfor;
    Print $I(j,l), l = 0,1,\ldots,j$, the entries of $j$th row;
    Compute $error = |I(j-1,j-1) - I(j,j)|$;
endwhile;
if ($error < eps$) then
    Print 'The value of integration', $I(j,j)$;
else
    Print 'The result does not achieve the desired accuracy,
    the best approximation is', $I(j,j)$.
end Romberg_Integration

Function Trap$(a,h,n)$
Input function $f(x)$;
Set $sum = 0$;
for $i = 1$ to $n-1$ do
    $sum = sum + f(a+ih)$;
endfor;
Compute Trap=$[f(a) + f(a+nh) + 2 * sum] * h/2$;
return;
end TRAP
```

Program 7.6

```
/* Program Romberg
   Integration by Romberg method. The initial integration is
   computed by trapezoidal rule.
   Here we assume that f(x)=x*x*exp(x). */
#include<stdio.h>
#include<math.h>
void main()
{
 float a,b,h,error=1,eps=1e-5,I[10][10];
 int n=1,j=0,k,l,maxrow;
 float trap(float a, float h, int n);
 float f(float x);
 printf("Enter the limits a and b ");
 scanf("%f %f",&a,&b);
 printf("Enter max. number of rows to be computed ");
 scanf("%d",&maxrow);
 h=b-a;
 I[0][0]=h*(f(a)+f(b))/2;
 printf("Romberg integration table\n");
 while((error>eps) && (j<maxrow))
   {
     j++; h/=2; n*=2;
     I[j][0]=trap(a,h,n);
     for(k=1;k<=j;k++)
        I[j][k]=I[j][k-1]+(I[j][k-1]-I[j-1][k-1])/(pow(4,k)-1);
     for(l=0;l<=j;l++) printf("%f ",I[j][l]); printf("\n");
     error=fabs(I[j-1][j-1]-I[j][j]);
   }
 if(error<eps)
   printf("The value of the integration is %f ",I[j][j]);
 else
  {
   printf("The result does not achieve the desired accuracy");
   printf("\nthe best approximation is %f ",I[j][j]);
 }
} /* main */

/* definition of the function f(x) */
float f(float x)
 {
  return(x*x*exp(x));
 }

/* function for the trapezoidal rule */
float trap(float a, float h, int n)
 {
   float sum=0;
   int i;
   for(i=1;i<=n-1;i++) sum+=f(a+i*h);
   sum=(f(a)+f(a+n*h)+2*sum)*h/2;
   return(sum);
 }
```

A sample of input/output:

```
Enter the limits a and b 0 1
Enter max. number of rows to be computed 10
Romberg integration table
0.885661 0.727834
0.760596 0.718908 0.718313
0.728890 0.718321 0.718282 0.718282
0.720936 0.718284 0.718282 0.718282 0.718282
The value of the integration is 0.718282
```

7.18 Double Integration

The double integration

$$I = \int_c^d \int_a^b f(x,y)\,dx\,dy \tag{7.197}$$

can also be evaluated using the methods discussed earlier. Generally, trapezoidal or Simpson's methods are used. The integral can be evaluated numerically by two successive integrations in x and y directions respectively taking one variable fixed at a time.

7.18.1 Trapezoidal method

Taking y is fixed. The inner integral is integrated using trapezoidal method. Then

$$\begin{aligned} I &= \frac{b-a}{2}\int_c^d [f(a,y)+f(b,y)]dy \\ &= \frac{b-a}{2}\left[\int_c^d f(a,y)dy + \int_c^d f(b,y)dy\right]. \end{aligned} \tag{7.198}$$

Again, applying trapezoidal rule on two integrals of right hand side and obtain the trapezoidal formula for double integral as

$$\begin{aligned} I &= \frac{b-a}{2}\left[\frac{d-c}{2}\{f(a,c)+f(a,d)\} + \frac{d-c}{2}\{f(b,c)+f(b,d)\}\right] \\ &= \frac{(b-a)(d-c)}{4}[f(a,c)+f(a,d)+f(b,c)+f(b,d)]. \end{aligned} \tag{7.199}$$

This expression shows that the integration can be done only if the values of the function $f(x,y)$ is available at the corner points $(a,c),(a,d),(b,c)$ and (b,d) of the rectangular region $[a,b;c,d]$.

The composite rule may also be used to determine the integral (7.197). To do this the interval $[a,b]$ is divided into n equal subintervals each of length h and the interval $[c,d]$ into m equal subintervals each of length k. That is,

$$x_i = x_0 + ih, \qquad x_0 = a, \qquad x_n = b, \qquad h = \frac{b-a}{n}$$

$$y_j = y_0 + jk, \qquad y_0 = c, \qquad y_m = d, \qquad k = \frac{d-c}{m}.$$

Now,

$$\begin{aligned}\int_a^b f(x,y) = \frac{h}{2}[&f(x_0,y) + 2\{f(x_1,y) + f(x_2,y) + \cdots \\ &+ f(x_{n-1},y)\} + f(x_n,y)]. \end{aligned} \tag{7.200}$$

The equation (7.200) is integrated between c and d, term by term, using trapezoidal rule. Therefore,

$$\begin{aligned} I &= \int_c^d \int_a^b f(x,y)dxdy \\ &= \frac{h}{2}\left[\frac{k}{2}\left\{f(x_0,y_0) + 2(f(x_0,y_1) + f(x_0,y_2) + \cdots + f(x_0,y_{m-1})) + f(x_0,y_m)\right\}\right. \\ &\quad +2.\frac{k}{2}\left\{f(x_1,y_0) + 2(f(x_1,y_1) + f(x_1,y_2) + \cdots + f(x_1,y_{m-1})) + f(x_1,y_m)\right\} \\ &\quad +2.\frac{k}{2}\left\{f(x_2,y_0) + 2(f(x_2,y_1) + f(x_2,y_2) + \cdots + f(x_2,y_{m-1})) + f(x_2,y_m)\right\} \\ &\quad + \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &\quad +2.\frac{k}{2}\left\{f(x_{n-1},y_0) + 2(f(x_{n-1},y_1) + \cdots + f(x_{n-1},y_{m-1})) + f(x_{n-1},y_m)\right\} \\ &\quad +\frac{k}{2}\left\{f(x_n,y_0) + 2(f(x_n,y_1) + f(x_n,y_2) + \cdots + f(x_n,y_{m-1})) + f(x_n,y_m)\right\} \\ &= \frac{hk}{4}\Bigg[f_{00} + 2\{f_{01} + f_{02} + \cdots + f_{0,m-1}\} + f_{0m} \\ &\quad +2\sum_{i=1}^{n-1}\{f_{i0} + 2(f_{i1} + f_{i2} + \cdots + f_{i,m-1}) + f_{im}\} \\ &\quad +f_{n0} + 2(f_{n1} + f_{n2} + \cdots + f_{n,m-1}) + f_{nm}\Bigg], \end{aligned} \tag{7.201}$$

where $f_{ij} = f(x_i,y_j), i = 0,1,\ldots,n; j = 0,1,2,\ldots,m$.

The method is of second order in both h and k.

Table 7.12: Tabular form of trapezoidal rule for double integration.

	y_0 y_1 y_2 y_{m-1} y_m
x_0	$[f_{00}+2(f_{01}+f_{02}+\cdots+f_{0,m-1})+f_{0,m}]\frac{k}{2}=I_0$
x_1	$[f_{10}+2(f_{11}+f_{12}+\cdots+f_{1,m-1})+f_{1,m}]\frac{k}{2}=I_1$
$\cdots$	$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$
x_n	$[f_{n0}+2(f_{n1}+f_{n2}+\cdots+f_{n,m-1})+f_{n,m}]\frac{k}{2}=I_n$
	$I=\frac{h}{2}[I_0+2(I_1+I_2+\cdots+I_{n-1})+I_n]$

Algorithm 7.7 (Double integration using trapezoidal rule). This algorithm determines the double integration of $f(x,y)$ within the region $[a,b;c,d]$ by trapezoidal rule.

The formula (7.201) can be written as

$$I=\frac{hk}{4}\left[\left\{f_{00}+f_{0m}+2\sum_{j=1}^{m-1}f_{0j}\right\}+2\sum_{i=1}^{n-1}\left\{f_{i0}+f_{im}+2\sum_{j=1}^{m-1}f_{ij}\right\}\right.$$
$$\left.+f_{n0}+f_{nm}+2\sum_{j=1}^{m-1}f_{nj}\right]=\frac{hk}{4}\left[sum(0)+2\sum_{i=1}^{n-1}sum(i)+sum(n)\right],$$

where $sum(i)=f_{i0}+f_{im}+2\sum_{j=1}^{m-1}f_{ij}$.

```
Algorithm Double_Trapezoidal
Input function $f(x, y)$;
Read $a, b, c, d, n, m$; //limits of $x$ and $y$ and number of subintervals.//
Compute $h = (b - a)/n$ and $k = (d - c)/m$;
Compute $f_{ij} = f(x_i, y_j), x_i = a + ih, y_j = c + jk$, $i = 0, 1, \ldots, n, j = 0, 1, \ldots, m$;
Set $result = 0$;
for $i = 0$ to $n - 1$ do
    $result = result + sum(i)$;
endfor;
Compute $result = (sum(0) + 2 * result + sum(n)) * h * k/4$;
Print $result$;

function $sum(i)$
Input array $f_{ij}$;
Set $t = 0$;
```

```
for $j = 1$ to $m - 1$ do
    $t = t + f_{ij}$;
endfor;
return($f_{i0} + f_{im} + 2 * t$);
end sum
end Double_Trapezoidal
```

Program 7.7

```
/* Program Trapezoidal for Two Variables
   The program to find the double integration of the function
   F(x,y)=1/{(1+x*x)(1+y*y)} defined over a rectangular region
   [a,b;c,d] by trapezoidal rule. */
#include<stdio.h>
int n,m; float f[15][15];
float sum(int);
float F(float,float);
void main()
 {
  int i,j; float a,b,c,d,h,k,x,y,result=0;
  printf("Enter the limits of x and y; [a,b;c,d] ");
  scanf("%f %f %f %f",&a,&b,&c,&d);
  printf("Enter the number of subdivisions n,m of x,y ");
  scanf("%d %d",&n,&m);
  h=(b-a)/n;
  k=(d-c)/m;
  x=a;
  for(i=0;i<=n;i++) /* computation of the function */
   {
    y=c;
    for(j=0;j<=m;j++)
      {
       f[i][j]=F(x,y);
       y+=k;
      }
      x+=h;
   }
  for(i=0;i<n;i++) result+=sum(i);
  result=(sum(0)+2*result+sum(n))*h*k/4;
  printf("The value of the integration is %8.5f",result);
}
```

```
float sum(int i)
 {
   float t=0; int j;
   for(j=1;j<m;j++) t+=f[i][j];
   return(f[i][0]+f[i][m]+2*t);
 }
/* definition of the function f(x,y) */
float F(float x,float y)
 {
  return( (1/(1+x*x))*(1/(1+y*y)));
 }
```

A sample of input/output:

```
Enter the limits of x and y; [a,b;c,d]
0 1 0 1
Enter the number of subdivisions n,m of x,y
10 10
The value of the integration is  0.69469
```

7.18.2 Simpson's 1/3 method

Let $x_0 = a, x_1 = x_0 + h, x_2 = b$ be the three points on the interval $[a, b]$ and $y_0 = c, y_1 = y_0 + k, y_2 = d$ be that on $[c, d]$, where $h = \dfrac{b-a}{2}, k = \dfrac{d-c}{2}$.

Then by Simpson's 1/3 rule on $\int_a^b f(x,y)dx$ one can write

$$\int_a^b f(x)\,dx = \frac{h}{3}[f(x_0,y) + 4f(x_1,y) + f(x_2,y)].$$

Again, by same rule on each term of the above expression, the final formula is given by

$$\begin{aligned} I &= \frac{h}{3}\Big[\frac{k}{3}\{f(x_0,y_0) + 4f(x_0,y_1) + f(x_0,y_2)\} \\ &\quad +4.\frac{k}{3}\{f(x_1,y_0) + 4f(x_1,y_1) + f(x_1,y_2)\} \\ &\quad +\frac{k}{3}\{f(x_2,y_0) + 4f(x_2,y_1) + f(x_2,y_2)\}\Big] \\ &= \frac{hk}{9}[f_{00}+f_{02}+f_{20}+f_{22}+4(f_{01}+f_{10}+f_{12}+f_{21})+16f_{11}], \end{aligned} \tag{7.202}$$

where $f_{ij} = f(x_i, y_j), i = 0, 1, 2; j = 0, 1, 2.$

Table 7.13: Tabular form of Simpson's rule for double integration.

	y_0 y_1 y_2 y_3 y_4 y_{m-1} y_m
x_0	$[f_{00}+4f_{01}+2f_{02}+4f_{03}+2f_{04}+\cdots+4f_{0,m-1}+f_{0,m}]\frac{k}{3}=I_0$
x_1	$[f_{10}+4f_{11}+2f_{12}+4f_{13}+2f_{14}+\cdots+4f_{1,m-1}+f_{1,m}]\frac{k}{3}=I_1$
$\cdots$	$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$
x_n	$[f_{n0}+4f_{n1}+2f_{n2}+4f_{n3}+2f_{n4}+\cdots+4f_{n,m-1}+f_{n,m}]\frac{k}{3}=I_n$
	$I=\frac{h}{3}[I_0+4(I_1+I_3+\cdots+I_{n-1})+2(I_2+I_4+\cdots+I_{n-2})+I_n]$

In general,

$$\int_{y_{j-1}}^{y_{j+1}}\int_{x_{i-1}}^{x_{i+1}} f(x,y)\,dx\,dy = \frac{hk}{9}\Big[f_{i-1,j-1}+f_{i-1,j+1}+f_{i+1,j-1}+f_{i+1,j+1}$$
$$+4(f_{i-1,j}+f_{i,j-1}+f_{i,j+1}+f_{i+1,j})+16f_{ij}\Big]. \qquad (7.203)$$

This formula is known as Simpson's 1/3 rule for double integration.

Example 7.18.1 Find the value of $\displaystyle\int_1^2\int_1^2 \frac{dx\,dy}{x^2+y^2}$ using trapezoidal and Simpson's 1/3 rules taking $h=k=0.25$.

Solution. Since $h=k=0.25$, let $x=1,1.25,1.50,1.75,2.0$ and $y=1,1.25,1.50,1.75,2.0$. The following table is constructed for the integrand $f(x,y)=\dfrac{1}{x^2+y^2}$.

	y				
x	1.00	1.25	1.50	1.75	2.00
1.00	0.50000	0.39024	0.30769	0.24615	0.20000
1.25	0.39024	0.32000	0.26230	0.21622	0.17978
1.50	0.30769	0.26230	0.22222	0.18824	0.16000
1.75	0.24615	0.21622	0.18824	0.16327	0.14159
2.00	0.20000	0.17978	0.16000	0.14159	0.12500

Let x be fixed and y be varying variable. Then by the **trapezoidal rule** on each row of the above table, we obtain

$$I_0=\int_1^2 f(1,y)dy=\frac{0.25}{2}[0.50000+2(0.39024+0.30769+0.24615)+0.20000]$$
$$=0.32352.$$

$$I_1 = \int_1^2 f(1.25, y)dy = \frac{0.25}{2}[0.39024 + 2(0.32000 + 0.26230 + 0.21622) + 0.17978]$$
$$= 0.27088.$$
$$I_2 = \int_1^2 f(1.5, y)dy = \frac{0.25}{2}[0.30769 + 2(0.26230 + 0.22222 + 0.18824) + 0.16000]$$
$$= 0.22665.$$
$$I_3 = \int_1^2 f(1.75, y)dy = \frac{0.25}{2}[0.24615 + 2(0.21622 + 0.18824 + 0.16327) + 0.14159]$$
$$= 0.19040.$$
$$I_4 = \int_1^2 f(2.0, y)dy = \frac{0.25}{2}[0.20000 + 2(0.17978 + 0.16000 + 0.14159) + 0.12500]$$
$$= 0.16097.$$

Hence finally

$$\begin{aligned}\int_1^2 \int_1^2 \frac{dx\, dy}{x^2 + y^2} &= \frac{h}{2}[f(1, y) + 2\{f(1.25, y) + f(1.5, y) + f(1.75, y)\} + f(2, y)] \\ &= \frac{h}{2}[I_0 + 2(I_1 + I_2 + I_3) + I_4] \\ &= \frac{0.25}{2}[0.32352 + 2(0.27088 + 0.22665 + 0.1904) + 0.16097] = 0.23254.\end{aligned}$$

Again, applying **Simpson's 1/3 rule** to each row of the above table, we have

$$I_0 = \int_1^2 f(1, y)dy = \frac{0.25}{3}[0.50000 + 4(0.39024 + 0.24615) + 2(0.30769) + 0.20000]$$
$$= 0.32175.$$

Similarly, $I_1 = 0.26996, I_2 = 0.22619, I_3 = 0.19018, I_4 = 0.16087$.
Hence the Simpson's 1/3 rule gives

$$\begin{aligned} I &= \frac{h}{3}[f(1, y) + 4\{f(1.25, y) + f(1.75, y)\} + 2f(1.5, y) + f(2, y)] \\ &= \frac{h}{3}[I_0 + 4(I_1 + I_3) + 2I_2 + I_4] \\ &= 0.23129.\end{aligned}$$

7.19 Monte Carlo Method

The Monte Carlo method is used to solve a large number of problems. The name of the method is derived from the name of the city Monte Carlo, the city of Monaco famous for its casino. Credit for inventing the Monte Carlo method often goes to Stanislaw

Ulam, a Polish mathematician who worked for John von Neumann in the United States Manhattan Project during World War II. He invented this method in 1946 and the first paper on it was published in 1949.

This method depends on a random sample and it is not suitable for hand calculation, because it depends on a large number of random numbers. The algorithm of Monte Carlo method is prepared to perform only one random trial. The trial is repeated for N times and the trials are independent. The final result is the average of all the results obtained in different trials.

Now, the Monte Carlo method is introduced to find numerical integration of a single valued function. Suppose the definite integral be

$$I = \int_a^b g(x)\, dx, \tag{7.204}$$

where $g(x)$ is a real valued function defined on $[a, b]$. The idea is to manipulate the definite integral into a form that can be solved by Monte Carlo method. To do this, we define the uniform probability density function (pdf) is defined on $[a, b]$ in the following.

$$f(x) = \begin{cases} \frac{1}{b-a}, & a < x < b \\ 0, & \text{otherwise.} \end{cases}$$

This this function is inserted to the equation (7.204) to obtain the following expression for I.

$$I = (b-a) \int_a^b g(x) f(x)\, dx. \tag{7.205}$$

It is observed that the integral on the right hand side of the equation (7.205) is simply the expectation of $g(x)$ under uniform pdf. Thus,

$$I = (b-a) \int_a^b g(x) f(x)\, dx = (b-a)\overline{g}. \tag{7.206}$$

Now, a sample x_i is drawn from the pdf $f(x)$, and for each x_i the value of $g(x_i)$ is calculated. To get a good approximation, a large number of sample is to be chosen and let G be the average of all the values of $g(x_i), i = 1, 2, \ldots, N$ (the sample size). Then

$$G = \frac{1}{N} \sum_{i=1}^{N} g(x_i). \tag{7.207}$$

It is easy to prove that the expectation of the average of N samples is the expectation of $g(x)$, i.e., $\overline{G} = \overline{g}$. Hence

$$I = (b-a)\overline{G} \simeq (b-a)\left(\frac{1}{N} \sum_{i=1}^{N} g(x_i)\right). \tag{7.208}$$

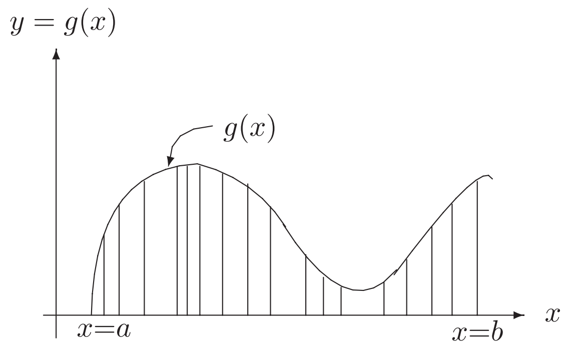


Figure 7.4: Random points are chosen in $[a, b]$. The value of $g(x)$ is evaluated at each random point (marked by straight lines in the figure).

Figure 7.4 illustrates the Monte Carlo method.

Thus the approximate value of the integral I on $[a, b]$ can be evaluated by taking the average of N observations of the integrand with the random variable x sampled uniformly over the interval $[a, b]$. This implies that the interval $[a, b]$ is finite, since an infinite interval cannot have a uniform pdf. The infinite limits of integration can be accommodated with more sophisticated techniques.

The true variance in the average G is equal to the true variance in g, i.e.,

$$var(G) = \frac{1}{N} var(g).$$

Although $var(G)$ is unknown, since it is a property of the pdf $f(x)$ and the real function $g(x)$, it is a constant. Furthermore, if an error is committed in the estimate of the integral I with the standard deviation, then one may expect the error in the estimate of I to decrease by the factor $1/\sqrt{N}$.

Algorithm 7.8 (Monte Carlo integration). This algorithm finds the value of the integral $\int_a^b f(x)\,dx$ using Monte Carlo method based on a sample of size N.

```
Algorithm Monte_Carlo
Input function $g(x)$;
Step 1: Read sample size $N$ and the lower and upper limits $a, b$, of the integral.
Step 2: For $i = 1$ to $N$ do steps 2.1–2.3
    Step 2.1: Generate a random number $y_i$ between 0 and 1.
    Step 2.2: Let $x_i = a + (b - a)y_i$. //$a \le x_i \le b$.//
    Step 2.3: Calculate $g(x_i)$ and compute $sum = sum + g(x_i)$. //Initially $sum = 0$
Step 3: Calculate $I = (sum/N) * (b - a)$.
Step 4: Print $I$.
end Monte_Carlo.
```

The function to generate the random numbers `rand()` is available in C programming language though it is not available in FORTRAN. Several methods are available to generate random numbers. A method is described in the following which will generate pseudo-random numbers.

7.19.1 Generation of random numbers

The random numbers are generated by a random process and these numbers are the values of a random variable. Several methods are available to generate random numbers. Practically, these methods do not generate ideal random numbers, because these methods follow some algorithms. So the available methods generate **pseudo random numbers**.

The commonly used simplest method to generate random numbers is the **power residue method**. A sequence of non-negative integers $x_1, x_2, \ldots$ is generated from the following relation

$$x_{n+1} = (a\,x_n)\,(\text{mod } m)$$

where x_0 is a starting value called the seed, a and m are two positive integers ($a < m$). The expression (ax_n) (mod m) gives the remainder when ax_n is divided by m. The possible values of x_{n+1} are $0, 1, 2, \ldots, m-1$, i.e., the number of different random numbers is m. The period of random number depends on the values of a, x_0 and m. Appropriate choice of a, x_0 and m gives a long period random numbers.

Suppose the computer which will generate the random numbers have a word-length of b bits. Let $m = 2^{b-1} - 1$, $a =$ an odd integer of the form $8k \pm 3$ and closed to $2^{b/2}$, $x_0 =$ an odd integer between 0 and m. Now ax_0 is a $2b$ bits integer. The least b significant bits form the random number x_1. The process is repeated for a desired number of times. For a 32-bit computer, $m = 2^{31} - 1 = 2147483647, a = 2^{16} + 3 = 65539, x_0 = 1267835015$, $(0 < x_0 < m)$.

To obtain the random numbers between $[0, 1]$, all numbers are divided by $m - 1$. These numbers are uniformly distributed over $[0, 1]$. The following FORTRAN function `RAND01()` generates random numbers between 0 and 1.

```
C   FORTRAN FUNCTION TO GENERATE A RANDOM NUMBER BETWEEN 0 AND 1
       FUNCTION RAND01()
       INTEGER*4 A,X0
       REAL M
       PARAMETER(M=2147483647.0)
       DATA A,X0/65539,1267835015/
       X0=IABS(A*X0)
       RAND01=X0/M
       RETURN
       END
```

Program 7.8

```
/* Program Monte Carlo
   Program to find the value of the integration of 1/(1+x^2)
   between 0 and 1 by Monte Carlo method for different values of N. */
#include<stdio.h>
#include<stdlib.h>
void main()
{
 float g(float x); /* g(x) may be changed accordingly */
 float x,y,I,sum=0.0,a=0.0,b=1.0;
 int i, N;
 srand(100); /* seed for random number */
 printf("Enter the sample size ");
 scanf("%d",&N);
 for(i=0;i<N;i++)
    {   /* rand() generates a random number between 0 and RAND_MAX */
      y=(float)rand()/RAND_MAX;
           /* generates a random number between 0 and 1*/
      x=a+(b-a)*y;
      sum+=g(x);
   }
I=sum*(b-a)/N;
printf("%f",I);
}
/* definition of function */
float g(float x)
{
  return(1/(1+x*x));
}
```

The results obtained for different values of N are tabulated in the following.

N	Integration
500	0.790020
1000	0.789627
1500	0.786979
3000	0.786553
4000	0.784793
10000	0.784094
15000	0.782420

Choice of method

In this chapter, three types of integration techniques viz., Newton-Cotes, Gaussian and Monte Carlo are discussed. These methods are computationally different from each other.

No definite rule can be given to choose integration method. But, the following points may be kept in mind while choosing an integration method.

(i) The simplest but crudest integration formula is the trapezoidal rule. This method gives a rough value of the integral. When a rough value is required then this method may be used.

(ii) Simpson's 1/3 rule gives more accurate result and it is also simple. Practically, this is the most widely used formula. Thus, if $f(x)$ does not fluctuate rapidly and is explicitly known then this method with a suitable subinterval can be used. If high accuracy is required then the Gaussian quadrature may be used.

(iii) Double integration may be done by using Simpson's rule with suitable subintervals.

(iv) If the integrand is known at some unequally spaced points, then the trapezoidal rule along with Romberg's integration is useful.

(v) If the integrand is violently oscillating or fluctuating then the Monte Carlo method can be used.

7.20 Worked out Examples

Example 7.20.1 The arc length of the curve $y = f(x)$ over the interval $a \le x \le b$ is $\displaystyle\int_a^b \sqrt{1+[f'(x)]^2}\,dx$. For the function $f(x) = x^3, 0 \le x \le 1$ find the approximate arc length using the composite trapezoidal and Simpson's 1/3 rules with $n = 10$.

Solution. The arc length $I = \int_0^1 \sqrt{1+[f'(x)]^2}\,dx = \int_0^1 \sqrt{1+9x^4}\,dx$.
Since $n = 10, h = (1-0)/10 = 0.1$.

x :	0.0	0.1	0.2	0.3	0.4	0.5
y :	1.00000	1.00045	1.00717	1.03581	1.10923	1.25000
x :	0.6	0.7	0.8	0.9	1.0	
y :	1.47187	1.77789	2.16481	2.62772	3.16228	

Trapezoidal rule gives

$$\begin{aligned}I &= \frac{h}{2}[y_0 + 2(y_1 + y_2 + \cdots + y_9) + y_{10}]\\ &= \frac{0.1}{2}[1.00000 + 2(1.00045 + 1.00717 + 1.03581 + 1.10923 + 1.25000 + 1.47187\\ &\quad +1.77789 + 2.16481 + 2.62772) + 3.16228]\\ &= 1.55261.\end{aligned}$$

Simpson's 1/3 rule gives

$$\begin{aligned}I &= \frac{h}{3}[y_0 + 4(y_1 + y_3 + \cdots + y_9) + 2(y_2 + y_4 + \cdots + y_8) + y_{10}]\\ &= \frac{0.1}{3}[1.00000 + 4(1.00045 + 1.03581 + 1.25000 + 1.77789 + 2.62772)\\ &\quad +2(1.00717 + 1.10923 + 1.47187 + 2.16481) + 3.16228]\\ &= 1.54786.\end{aligned}$$

The exact value is 1.547866.

Example 7.20.2 Find the number n and the step size h such that the error for the composite Simpson's 1/3 rule is less than 5×10^{-7} when evaluating $\int_2^5 \log x \, dx$.

Solution. Let $f(x) = \log x$. $f''(x) = -\frac{1}{x^2}, f'''(x) = \frac{2}{x^3}, f^{iv}(x) = -\frac{6}{x^4}$.
The maximum value of $|f^{iv}(x)| = \frac{6}{x^4}$ over [2,5] occurs at the end point $x = 2$.
Thus $|f^{iv}(\xi)| \le |f^{iv}(2)| = \frac{3}{8}, 2 \le \xi \le 5$.
The error E of Simpson's 1/3 rule is $E = -\frac{(b-a)f^{iv}(\xi)h^4}{180}$.
Therefore,

$$|E| = \left|\frac{(b-a)f^{iv}(\xi)h^4}{180}\right| \le \frac{(5-2)h^4}{180}.\frac{3}{8} = \frac{h^4}{160}.$$

Also $h = \frac{b-a}{n} = \frac{3}{n}$. Thus, $|E| \le \frac{h^4}{160} = \frac{81}{n^4 \times 160} \le 5 \times 10^{-7}$.

That is, $n^4 \ge \frac{81}{160} \cdot \frac{1}{5 \times 10^{-7}}$ or, $n \ge 31.72114$.

Since n is integer, we choose $n = 32$ and the corresponding step size is

$$h = \frac{3}{n} = \frac{3}{32} = 0.09375.$$

Example 7.20.3 Obtain the approximate quadrature formula

$$\int_{-1}^{1} f(x)dx = \frac{5}{9}f(-\sqrt{0.6}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{0.6}).$$

Solution. Let $\displaystyle\int_{-1}^{1} f(x)dx = w_1 f(-\sqrt{0.6}) + w_2 f(0) + w_3 f(\sqrt{0.6})$,
where w_1, w_2, w_3 are to be determined. To find them, substituting $f(x) = 1, x, x^2$ successively to the above equation, we obtain the following system of equations
$2 = w_1 + w_2 + w_3$ (when $f(x) = 1$)
$0 = \sqrt{-0.6}\, w_1 + \sqrt{0.6}\, w_3$ (when $f(x) = x$)
$\frac{2}{3} = 0.6\, w_1 + 0.6\, w_3$ (when $f(x) = x^2$).
Solution of this system is $w_1 = w_3 = \dfrac{5}{9}$ and $w_2 = \dfrac{8}{9}$.
Hence the result follows.

Example 7.20.4 Deduce the following quadrature formula

$$\int_{0}^{n} f(x)dx = n\left[\frac{3}{8}f(0) + \frac{1}{24}\left\{19f(n) - 5f(2n) + f(3n)\right\}\right].$$

Solution. If the above formula gives exact result for the polynomial of degree up to 4 then let $f(x) = 1, x, x^2, x^3$. Substituting $f(x) = 1, x, x^2, x^3$ successively to the equation

$$\int_{0}^{n} f(x)dx = w_1 f(0) + w_2 f(n) + w_3 f(2n) + w_4 f(3n),$$

we get
$n = w_1 + w_2 + w_3 + w_4$, $n^2/2 = nw_2 + 2nw_3 + 3nw_4$,
$n^3/3 = n^2 w_2 + 4n^2 w_3 + 9n^2 w_4$, $n^4/4 = n^3 w_2 + 8n^3 w_3 + 27n^3 w_4$.
Solution is $w_1 = \dfrac{3n}{8}, w_2 = \dfrac{19n}{24}, w_3 = -\dfrac{5n}{24}, w_4 = \dfrac{n}{24}$.
Hence $\displaystyle\int_{0}^{n} f(x)dx = n\left[\frac{3}{8}f(0) + \frac{1}{24}\left\{19f(n) - 5f(2n) + f(3n)\right\}\right]$.

Example 7.20.5 If $f(x)$ is a polynomial of degree 2, prove that

$$\int_{0}^{1} f(x)dx = \frac{1}{12}[5f(0) + 8f(1) - f(2)].$$

Solution. Let the formula be
$\displaystyle\int_{0}^{1} f(x)dx = w_1 f(0) + w_2 f(1) + w_3 f(2)$. The formula is exact for $f(x) = 1, x, x^2$.

Substituting $f(x) = 1, x, x^2$ successively to the above formula and find the following set of equations.

$$1 = w_1 + w_2 + w_3,\ 1/2 = w_2 + 2w_3,\ 1/3 = w_2 + 4w_3.$$

Solution of these equations is $w_1 = \dfrac{5}{12}, w_2 = \dfrac{2}{3}, w_3 = -\dfrac{1}{12}$.

Hence the formula becomes $\displaystyle\int_0^1 f(x)dx = \frac{1}{12}[5f(0) + 8f(1) - f(2)]$.

Example 7.20.6 Deduce the following quadrature formula

$$\int_{-1}^1 f(x)dx = \frac{2}{3}[f(0) + f(1/\sqrt{2}) + f(-1/\sqrt{2})].$$

Solution. Let $\displaystyle\int_{-1}^1 f(x)dx = w_1 f(0) + w_2 f(1/\sqrt{2}) + w_3 f(-1/\sqrt{2})$.

As in previous examples, the system of equations for $f(x) = 1, x, x^2$ are

$$2 = w_1 + w_2 + w_3,\ 0 = \frac{1}{\sqrt{2}}w_2 - \frac{1}{\sqrt{2}}w_3,\ \frac{2}{3} = \frac{1}{2}w_2 + \frac{1}{2}w_3.$$

Solution of these equations is $w_1 = w_2 = w_3 = 2/3$. Hence the result follows.

Example 7.20.7 Write down the quadrature polynomial which takes the same values as $f(x)$ at $x = -1, 0, 1$ and integrate it to obtain the integration formula

$$\int_{-1}^1 f(x)dx = \frac{1}{3}[f(-1) + 4f(0) + f(1)].$$

Assuming the error to have the form $Af^{iv}(\xi), -1 < \xi < 1$, find the value of A.

Solution. To find the quadratic polynomial, the Lagrange's interpolation formula is used. The Lagrange's interpolation for the points $-1, 0, 1$ is

$$\begin{aligned}
f(x) &= \frac{(x-0)(x-1)}{(-1-0)(-1-1)}f(-1) + \frac{(x+1)(x-1)}{(0+1)(0-1)}f(0) + \frac{(x+1)(x-0)}{(1+1)(1-0)}f(1)\\
&= \frac{1}{2}(x^2 - x)f(-1) - (x^2 - 1)f(0) + \frac{1}{2}(x^2 + x)f(1)\\
&= \Big[\frac{1}{2}f(-1) - f(0) + \frac{1}{2}f(1)\Big]x^2 + \Big[\frac{1}{2}f(1) - \frac{1}{2}f(-1)\Big]x + f(0).
\end{aligned}$$

This is the required quadratic polynomial. Integrating it between -1 and 1.

$$\int_{-1}^1 f(x)dx = \frac{1}{2}f(-1)\int_{-1}^1 (x^2 - x)dx - f(0)\int_{-1}^1 (x^2 - 1)dx + \frac{1}{2}f(1)\int_{-1}^1 (x^2 + x)dx$$

$$= \frac{1}{2}f(-1).\frac{2}{3} - f(0)\Big(-\frac{4}{3}\Big) + \frac{1}{2}f(1).\frac{2}{3}$$
$$= \frac{1}{3}[f(-1) + 4f(0) + f(1)].$$

Here the error is of the form $Af^{iv}(\xi)$. Let the error be $E = Af^{iv}(\xi) = \dfrac{C}{4!}f^{iv}(\xi)$, $-1 < \xi < 1$. This indicates that the above formula gives exact result for the polynomials of degree up to 3 and has error for the polynomial of degree 4. Let $f(x) = x^4$. Then the value of $\displaystyle\int_{-1}^{1} x^4 dx$ obtain from the above formula is $\frac{1}{3}[(-1)^4 + 4.0^4 + 1^4] = \frac{2}{3}$ and the exact value is $\displaystyle\int_{-1}^{1} x^4 dx = \frac{2}{5}$.
Therefore, $C = \displaystyle\int_{-1}^{1} x^4 dx - \frac{2}{3} = -\frac{4}{15}$.
Hence $A = \dfrac{C}{4!} = -\dfrac{1}{90}$.

Example 7.20.8 Derive Simpson's 1/3 rule using the method of undetermined coefficients.

Solution. Let

$$I = \int_{x_0}^{x_2} f(x)dx = \int_{-h}^{h} f(z + x_1)dz \quad \text{where } z = x - x_1$$
$$= \int_{-h}^{h} F(z)dz = w_1F(-h) + w_2F(0) + w_3F(h), \qquad F(z) = f(z + x_1).$$

The coefficients w_1, w_2, w_3 are to be determined. To determine these numbers assume that the formula is exact for $F(z) = 1, z, z^2$. Substituting $F(z) = 1, z, z^2$ successively to the above formula and find the following equations

$$w_1 + w_2 + w_3 = 2h, \qquad -hw_1 + hw_3 = 0, \qquad h^2w_1 + h^2w_3 = \frac{2}{3}h^3.$$

Solution of these equations is $w_1 = w_3 = \dfrac{h}{3}$ and $w_3 = \dfrac{4h}{3}$.
Therefore,

$$I = \int_{x_0}^{x_2} f(x)dx = \int_{-h}^{h} F(z)dz = \frac{h}{3}F(-h) + \frac{4h}{3}F(0) + \frac{h}{3}F(h)$$
$$= \frac{h}{3}[F(-h) + 4F(0) + F(h)]$$

This can be written as

$$\begin{aligned} I &= \frac{h}{3}[f(x_1 - h) + 4f(x_1) + f(x_1 + h)] \\ &= \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]. \end{aligned}$$

This is the required Simpson's 1/3 rule.

7.21 Exercise

1. From the following table of values, estimate $y'(1.05)$ and $y''(1.05)$:

x :	1.00	1.05	1.10	1.15	1.20	1.25
y :	1.1000	1.1347	1.1688	1.1564	1.2344	1.2345

2. A slider in a machine moves along a fixed straight rod. Its distance x cm along the rod is given below for various values of time t (second). Find the velocity of the slider and its acceleration when $t = 0.3$ sec.

t :	0.3	0.4	0.5	0.6	0.7	0.8
x :	3.364	3.395	3.381	3.324	3.321	3.312

   Use the formula based on Newton's forward difference interpolation to find the velocity and acceleration.

3. Use approximate formula to find the values of $y'(2)$ and $y''(2)$ from the following table:

x :	2.0	3.0	4.0	5.0	6.0	7.0
y :	–0.15917	1.09861	1.38629	1.60944	1.79176	1.94591

4. Find the values of $f'(5)$ and $f''(5)$ from the following table:

x :	1	2	3	4	5
$f(x)$:	10	26	50	82	122

5. Find the values of $y'(1), y'(1.2), y'(4), y'(3.9)$ from the following values

x :	1	2	3	4
y :	0.54030	–0.41615	–0.98999	–0.65364

6. Use two-point and three-point formulae to find the values of $f'(2.0)$ and $f''(2.0)$.

x :	0.5	1.0	1.5	2.0	2.5	3.0	3.5
$f(x)$:	–0.30103	0.00000	0.17609	0.30103	0.39794	0.47712	0.54407

7. Deduce the following relation between the differential and finite difference operators

   (a) $D \equiv \frac{1}{h}\Big[\Delta - \frac{\Delta^2}{2} + \frac{\Delta^3}{3} - \frac{\Delta^4}{4} + \cdots\Big]$

   (b) $D \equiv \frac{1}{h}\Big[\nabla + \frac{\nabla^2}{2} + \frac{\nabla^3}{3} + \frac{\nabla^4}{4} + \cdots\Big]$

   (c) $D^2 \equiv \frac{1}{h^2}\Big[\Delta^2 - \Delta^3 + \frac{11}{12}\Delta^4 - \frac{5}{6}\Delta^5 + \frac{137}{180}\Delta^6 - \cdots\Big]$

   (d) $D^2 \equiv \frac{1}{h^2}\Big[\nabla^2 + \nabla^3 + \frac{11}{12}\nabla^4 + \frac{5}{6}\nabla^5 + \frac{137}{180}\nabla^6 + \cdots\Big]$

8. Use Taylor's series to deduce the following formula

$$f'(x_0) = \frac{-3f(x_0) + 4f(x_1) - f(x_2)}{2h} + \frac{h^2}{3}f'''(\xi), \qquad x_0 < \xi < x_2.$$

   Also, determine the optimum value of h which minimizes the total error (sum of truncation and round-off errors).

9. Deduce the formula

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2}f''(\xi), x_0 < \xi < x_1.$$

   Also, find the value of h such that the sum of truncation and round-off errors is minimum.

10. Determine the value of k in the formula

$$f''(x_0) = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} + O(h^k).$$

11. Use Lagrange interpolation formula to deduce the following formulae

   (a) $f''(x_0) \simeq \dfrac{2f(x_0) - 5f(x_1) + 4f(x_2) - f(x_3)}{h^2}$

   (b) $f'''(x_0) \simeq \dfrac{-5f(x_0) + 18f(x_1) - 24f(x_2) + 14f(x_3) - 3f(x_4)}{2h^3}$,

   (c) $f^{iv}(x_0) \simeq \dfrac{3f(x_0) - 14f(x_1) + 26f(x_2) - 24f(x_3) + 11f(x_4) - 2f(x_5)}{h^4}$

12. Use Taylor's expansion to derive the formula

$$f'''(x_0) \simeq \frac{f(x_0+2h) - 2f(x_0+h) + 2f(x_0-h) - f(x_0-2h)}{2h^3}.$$

13. Use Richardson's extrapolation to find $f'(1)$ from the following:

x :	0.6	0.8	0.9	1.0	1.1	1.2	1.4
$f(x)$:	1.70718	1.85979	1.092586	1.98412	2.03384	2.07350	2.12899

Apply the approximate formula

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h}$$

with $h = 04, 0.2$ and 0.1, to find initial values.

14. Find the value of $\int_0^2 (1 + e^{-x} \sin 4x)\, dx$ using basic (non-composite) rules of (i) trapezoidal, (ii) Simpson's 1/3, (iii) Simpson's 3/8, (iv) Boole's, and (v) Weddle's.

15. Consider $f(x) = 3 + \sin(2\sqrt{x})$. Use the composite trapezoidal and Simpson's 1/3 rules with 11 points to compute an approximation to the integral of $f(x)$ taken over $[0, 5]$.

16. Consider the integrals

(a) $\int_0^5 e^{-x^2}\, dx$ (b) $\int_0^1 x^4 e^{x-1}\, dx.$

Evaluate these by (i) Trapezoidal rule with 11 points
(ii) Simpson's 1/3 rule with 11 points.

17. Evaluate the following integral using Simpson's 1/3 rule.

$$\int_0^{\pi/2} \frac{dx}{\sin^2 x + 2\cos^2 x}.$$

18. Evaluate the integral

$$\int_0^1 e^{x+1}\, dx$$

using Simpson's 1/3 rule, by dividing the interval of integration into eight equal parts.

19. Evaluate the integral
$$\int_{1.0}^{1.8} \frac{e^x + e^{-x}}{2}\, dx$$
using Simpson's 1/3 rule and trapezoidal rule, by taking $h = 0.2$.

20. A curve is drawn to pass through the points 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	3.6	2.4

Use Simpson's 1/3 rule to estimate the area bounded by the curve and the lines $x = 1$ and $x = 4$ and x-axis.

21. Find the value of
$$\int_0^1 \frac{dx}{1 + x^2}$$
taking 5 sub-intervals, by trapezoidal rule, correct to five significant figures. Also find the error by comparing with the exact value.

22. Evaluate the integral
$$\int_4^{5.2} ln\; x\; dx,$$
using Simpson's one-third rule with $h = 0.1$ and compare the result with the exact value.

23. Find the number of subintervals n and the step size h so that the error for the composite trapezoidal rule is less than 5×10^{-4} for the approximation $\displaystyle\int_1^5 \sin x\; dx$.

24. Verify that the Simpson's 1/3 rule is exact for polynomial of degree less than or equal to 3 of the form $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ over [0,2].

25. Integrate the Lagrange's interpolation polynomial
$$\phi(x) = \frac{x - x_1}{x_0 - x_1} f_0 + \frac{x - x_0}{x_1 - x_0} f_1$$
over the interval $[x_0, x_1]$ and establish the trapezoidal rule.

26. The solid of revolution obtained by rotating the region under the curve $y = f(x)$, $a \leq x \leq b$, about the x-axis has surface area given by
$$\text{area } = 2\pi \int_a^b f(x)\sqrt{1 + [f'(x)]^2}\; dx.$$

Find the surface area for the functions (a) $f(x) = \cos x$, $0 \le x \le \pi/4$, (b) $f(x) = \log x$, $1 \le x \le 5$, using trapezoidal and Simpson's 1/3 rules with 10 subintervals.

27. Show that the Simpson's 1/3 rule produces exact result for the function $f(x) = x^2$ and $f(x) = x^3$, that is, (a) $\int_a^b x^2\,dx = \frac{b^2}{3} - \frac{a^2}{3}$ and (b) $\int_a^b x^3\,dx = \frac{b^3}{4} - \frac{a^3}{4}$, by taking four subintervals.

28. A rocket is lunched from the ground. Its acceleration $a(t)$ measured in every 5 second is tabulated below. Find the velocity and the position of the rocket at $t = 30$ second. Use trapezoidal as well as Simpson's rules. Compare the answers.

t	:	0	5	10	15	20	25	30
$a(t)$	:	40.0	46.50	49.25	52.25	55.75	58.25	60.50

29. The natural logarithm of a positive number x is defined by

$$\log_e x = -\int_x^1 \frac{dt}{t}.$$

Find the maximum step size h to get the truncation error bound 0.5×10^{-3} when finding the value of $\log_2(0.75)$ by trapezoidal rule.

30. Expand $F(x)$ where $F'(x) = f(x)$, by Taylor series about $x_0 + h/2$ and establish the midpoint rule

$$\int_{x_0}^{x_1} f(x)\,dx = hf(x_0 + h/2) + \frac{h^2}{24}f''(\xi), \qquad x_0 < \xi < x_1, h = \frac{x_1 - x_0}{2}.$$

31. Complete the following table to compute the value of the integral $\int_0^3 \frac{\sin 2x}{1+x^5}dx$ using Romberg integration.

I_0 Trapezoidal rule	I_1 Simpson's rule	I_2 Boole's rule	I_3 Third improvement
–0.00171772			
0.02377300	$\cdots$		
0.60402717	$\cdots$	$\cdots$	
0.64844713	$\cdots$	$\cdots$	$\cdots$
0.66591329	$\cdots$	$\cdots$	$\cdots$

32. Find the values of the following integrals using Romberg integration starting from trapezoidal rule, correct up to five decimal points.
(a) $\int_1^2 \sqrt{4x - x^2}\, dx$, (b) $\int_{1/(2\pi)}^2 \sin(1/x)\, dx$.

33. Use three-point Gauss-Legendre formula to evaluate the integrals
(a) $\int_{-1}^1 \frac{1}{1+x^2} dx$, (b) $\int_0^{\pi/2} \sin x\, dx$.

34. The three-point Gauss-Legendre formula is

$$\int_{-1}^1 f(x)dx = \frac{5f(-\sqrt{0.6}) + 8f(0) + 5f(\sqrt{0.6})}{9}.$$

Show that the formula is exact for $f(x) = 1, x, x^2, x^3, x^4, x^5$.

35. Find the value of $\int_0^2 \frac{x}{1+x^3}\, dx$ using four-point and six-point Gauss-Legendre quadrature formulae.

36. Find the value of $\int_{-1}^1 e^{5x} \cos x\, dx$ using
(a) Gauss-Legendre quadrature for $n = 3, 4$,
(b) Lobatto quadrature for $n = 3, 4$.

37. Find the value of $\int_{-1}^1 (1 - x^2) \sin x\, dx$ using
(a) Six-point Gauss-Legendre quadrature,
(b) Three-point Gauss-Chebyshev quadrature.

38. Find the value of $\int_0^\infty e^{-x} \cos x\, dx$ using
(a) Three-point Gauss-Laguerre quadrature,
(b) Three-point Gauss-Hermite quadrature.

39. Evaluate $\int_0^1 \cos 2x\, (1 - x^2)^{-1/2}\, dx$ using any suitable method.

40. Using the method of undetermined coefficients, derive the following formulae.
(a) $\int_0^{2\pi} f(x) \sin x\, dx = f(0) - f(2\pi)$
(b) $\int_0^h y\, dx = \frac{h}{2}(y_0 + y_1)$.

41. Find the weights w_1, w_2, w_3 so that the relation

$$\int_{-1}^{1} f(x)\,dx = w_1 f(-\sqrt{0.6}) + w_2 f(0) + w_3 f(\sqrt{0.6})$$

is exact for the functions $f(x) = 1, x, x^2$.

42. Find the values of a, b, c such that the truncation error in the formula

$$\int_{-h}^{h} f(x)\,dx = h[af(-h) + bf(0) + cf(h)]$$

is minimized.

43. Determine the weights and the nodes in the formula

$$\int_{-1}^{1} f(x)dx = \sum_{i=0}^{3} w_i f(x_i)$$

with $x_0 = -1$ and $x_3 = 1$ so that the formula becomes exact for polynomials of highest degree.

44. Find the values of a, b, c such that the formula

$$\int_{0}^{h} f(x)dx = h[af(0) + hf(h/3) + cf(h)]$$

is exact for polynomial of as high order as possible.

45. Compare Newton-Cotes and Gaussian quadrature methods.

46. Use Euler-Maclaurin formula to find the value of π from the relation

$$\frac{\pi}{4} = \int_0^1 \frac{dx}{1+x^2}.$$

47. Use Euler-Maclaurin formula to find the values of the following series:
(a) $\dfrac{1}{51^2} + \dfrac{1}{53^2} + \dfrac{1}{55^2} + \cdots + \dfrac{1}{99^2}$

(b) $\dfrac{1}{11^2} + \dfrac{1}{12^2} + \dfrac{1}{13^2} + \cdots + \dfrac{1}{99^2}$

(c) $\dfrac{1}{1} + \dfrac{1}{2} + \dfrac{1}{3} + \cdots + \dfrac{1}{20}$.

48. Use Euler-Maclaurin formula to prove the following results:
    (a) $\displaystyle\sum_{x=1}^{n} x = \frac{n(n+1)}{2}$
    (b) $\displaystyle\sum_{x=1}^{n} x^2 = \frac{n(n+1)(2n+1)}{6}$
    (c) $\displaystyle\sum_{x=1}^{n} x^4 = \frac{n(6n^4+15n^3+10n^2-1)}{30}$.

49. Use the identity $\displaystyle\frac{\pi^2}{6} = \sum_{n=1}^{\infty} \frac{1}{n^2}$ to compute π^2.

50. Use Euler-Maclaurin formula to find the value of $\int_0^1 x^3\, dx$.

51. Use Euler-Maclaurin formula to deduce trapezoidal and Simpson's 1/3 rules.

52. Evaluate the double integral
$$I = \int_0^1 \int_0^2 \frac{2xy}{\sqrt{(1+x^2)(1+y^2)}}\, dy\, dx$$
using Simpson's 1/3 rule with step size $h = k = 0.25$.

53. Use Simpson's 1/3 rule to compute the integral
$$I = \int\int_R \frac{dx\, dy}{x^2+y^2}$$
where R is the square region with corners $(1,1), (2,1), (2,2), (1,2)$.

54. Use Simpson's 1/3 rule to compute the integral $\displaystyle\int_0^1 \int_0^1 \frac{\sin xy}{1+xy}\, dx\, dy$ with $h = k = 0.25$.

55. Use Monte Carlo method to find the value of $\displaystyle\int_1^5 \frac{x}{x+\cos x}\, dx$, taking sample size $N = 10$.

Chapter 8

Ordinary Differential Equations

Many problems in science and engineering can be represented in terms of differential equations satisfying certain given conditions. The analytic methods to solve differential equations are used for a limited classes of differential equations. Most of the differential equations governed by physical problems do not possess closed form solutions. For these types of problems the numerical methods are used. A number of good numerical methods are available to find numerical solution of differential equations.

Let us consider the general first order differential equation

$$\frac{dy}{dx} = f(x, y) \tag{8.1}$$

with initial condition

$$y(x_0) = y_0. \tag{8.2}$$

The solution of a differential equation can be done in one of the following two forms:

(i) A series solution for y in terms of powers of x. Then the values of y can be determined by substituting $x = x_0, x_1, \ldots, x_n$.

(ii) A set of tabulated values of y for $x = x_0, x_1, \ldots, x_n$ with spacing h.

In case (i), the solution of the differential equation is computed in terms of x, and the values of y are calculated from this solution, where as, in case (ii), the solution of the differential equation is obtained by applying the method repeatedly for each value of x $(= x_1, x_2, \ldots, x_n)$.

If the differential equation is of nth order then its general solution contains n arbitrary constants. To find the values of these constants, n conditions are needed. The problems in which all the conditions are specified at the **initial** point only, are called **initial value**

problems (IVPs). The problems of order two or more and for which the conditions are given at two or more points are called **boundary value problems (BVPs)**.

There may not exist a solution of an ordinary differential equation always. The sufficient condition for existence of unique solution is stated below.

Existence and Uniqueness

Theorem 8.1 (Lipschitz conditions). *Let $f(x,y)$ be a real valued function and*

(i) $f(x,y)$ is defined and continuous in the strip $x_0 \leq x \leq b$, $-\infty < y < \infty$,

(ii) there exists a constant L such that for any $x \in [x_0, b]$ and for any two numbers y_1 and y_2

$$|f(x,y_1) - f(x,y_2)| \leq L|y_1 - y_2|,$$

where L is called the Lipschitz constant.

Then for any y_0, the initial value problem

$$\frac{dy}{dx} = f(x,y), \quad y(x_0) = y_0$$

has a unique solution $y(x)$ for $x \in [x_0, b]$.

The methods to find approximate solution of an initial value problem are referred as **difference methods** or **discrete variable methods**. The solutions are determined at a set of discrete points called a **grid** or **mesh** of points.

The errors committed in solving an initial value problem are of two types– discretization and round-off. The discretization error, again, are of two types – **global discretization** error and **local discretization error**. The **global discretization error** or **global truncation error** E_i^{global} is defined as

$$E_i^{\text{global}} = y(x_i) - Y(x_i), i = 1, 2, \ldots. \tag{8.3}$$

that is, it is the difference between the exact solution $Y(x_i)$ and the solution $y(x_i)$ obtained by discrete variable method.

The **local discretization error** or **local truncation error** E_{i+1}^{local} is defined by

$$E_{i+1}^{\text{local}} = y(x_{i+1}) - y(x_i), i = 0, 1, 2, \ldots. \tag{8.4}$$

This error generated at each step from x_i to x_{i+1}. The error at the end of the interval is called the final global error (F.G.E.)

$$E_n^{\text{FGE}} = |y(x_n) - Y(x_n)|. \tag{8.5}$$

In analyzing different procedures used to solve ordinary as well as partial differential equations, we use the following numerical concepts.

The order of a finite difference approximation of a differential equation is the rate at which the global error of the finite difference solution approaches to zero as the size of the grid spacing (h) approaches zero. When applied to a differential equation with a bounded solution, a finite difference equation is **stable** if it produces a bounded solution and is **unstable** if it produces an unbounded solution. It is quite possible that the numerical solution to a differential equation grows unbounded even though its exact solution is well behaved. Of course, there are cases for which the exact solution may be unbounded, but, for our discussion of stability we concentrate only on the cases in which the exact solution is bounded. In stability analysis, conditions are deduced in terms of the step size h for which the numerical solution remains bounded. In this connection, the numerical methods are of three classes.

(i) **Stable numerical scheme:** The numerical solution does not blow up with choice of step size.

(ii) **Unstable numerical scheme:** The numerical solution blows up with any choice of step size.

(iii) **Conditionally stable numerical scheme:** Numerical solution remains bounded with certain choices of step size.

A finite difference method is **convergent** if the solution of the finite difference equation approaches to a limit as the size of the grid spacing tends to zero. But, there is no guarantee, in general, that this limit corresponds to the exact solution of the differential equation.

A finite difference equation is consistent with a differential equation if the difference between the solution of the finite difference equation and those of the differential equation tends to zero as the size of the grid spacing tends to zero independently.

If the value of y_{i+1} depends only on the value of y_i then the method is called **single-step method** and if two or more values are required to evaluate the value of y_{i+1} then the method is known as **two-step** or **multistep method**.

Again, if the value of y_{i+1} depends only on the values of y_i, h and $f(x_i, y_i)$ then the method used to determine y_{i+1} is called **explicit method**, otherwise the method is called **implicit method**.

8.1 Taylor's Series Method

The Taylor's series method is the most fundamental method and it is the standard to which we compare the accuracy of the various other numerical methods for solving an initial value problem.

Let us consider the first order differential equation

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0. \tag{8.6}$$

The exact solution $y(x)$ of (8.6) around the point $x = x_0$ is given by

$$y(x) = y_0 + (x - x_0)y_0' + \frac{(x - x_0)^2}{2!}y_0'' + \cdots \tag{8.7}$$

If the values of y_0, y_0', y_0'', etc. are known then (8.7) becomes a power series of x.

The derivatives can be obtained by taking the total derivatives of $f(x, y)$.

Given that $y'(x) = f(x, y)$.

From the definition of total derivatives, one can write

$$\begin{aligned}
y''(x) &= f_x + y'(x)f_y = f_x + f_y f \\
y'''(x) &= (f_{xx} + f_{xy}f) + (f_{yx} + f_{yy}f)f + f_y(f_x + f_y f) \\
&= f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y f_x + f_y^2 f \\
y^{iv}(x) &= (f_{xxx} + 3f_{xxy}f + 3f_{xyy}f^2 + f_{yy}f^3) + f_x(f_{xx} + 2f_{xy}f + f_{yy}f^2) \\
&\quad +3(f_x + f_y f)(f_{xy} + f_{yy}f) + f_y^2(f_x + f_x f)
\end{aligned}$$

and in general

$$y^{(n)} = \left(\frac{\partial}{\partial x} + f\frac{\partial}{\partial y}\right)^{(n-1)} f(x, y).$$

The general expression for **Taylor's series method of order** n is

$$y_{i+1} = y_i + hy'(x_i) + \frac{h^2}{2!}y''(x_i) + \cdots + \frac{h^n}{n!}y^{(n)}(x_i). \tag{8.8}$$

From above it is clear that if $f(x, y)$ is not easily differentiable, then the computation of y is very laborious. Practically, the number of terms in the expansion must be restricted.

Error

The final global error of Taylor's series method is of the order of $O(h^{n+1})$. Thus, for large value of n the error becomes small. If n is fixed then the step size h is chosen in such a way that the global error becomes as small as desired.

Example 8.1.1 Use Taylor's series method to solve the equation

$$\frac{dy}{dx} = x - y, \quad y(0) = 1$$

at x and at $x = 0.1$.

Solution. The Taylor's series around $x = 0$ for $y(x)$ is

$$y(x) = y_0 + xy_0' + \frac{x^2}{2!}y_0'' + \frac{x^3}{3!}y_0''' + \frac{x^4}{4!}y_0^{iv} + \cdots .$$

Differentiating y' repeatedly with respect to x and substituting $x = 0$, we obtain

$$\begin{aligned}
y'(x) &= x - y, \; y_0' = -1 \\
y''(x) &= 1 - y', \; y_0'' = 1 - y_0' = 2 \\
y'''(x) &= -y'', \; y_0''' = -2 \\
y^{iv}(x) &= -y''', \; y_0^{iv} = 2
\end{aligned}$$

and so on.
The Taylor's series becomes

$$\begin{aligned}
y(x) &= 1 - x + \frac{x^2}{2!} \cdot 2 + \frac{x^3}{3!} \cdot (-2) + \frac{x^4}{4!} \cdot 2 + \cdots \\
&= 1 - x + x^2 - \frac{x^3}{3} + \frac{x^4}{12} - \cdots .
\end{aligned}$$

This is the Taylor's series solution of the given differential equation at any point x.
Now,

$$y(0.1) = 1 - (0.1) + (0.1)^2 - \frac{(0.1)^3}{3} + \frac{(0.1)^4}{12} - \cdots = 0.909675.$$

8.2 Picard's Method of Successive Approximations

In this method, the value of dependent variable y is expressed as a function of x.

Let us consider the differential equation

$$\frac{dy}{dx} = f(x, y) \tag{8.9}$$

with initial conditions

$$x = x_0, \qquad y(x_0) = y_0. \tag{8.10}$$

Now, integration of (8.9) between x_0 and x gives

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

Thus

$$y(x) = y_0 + \int_{x_0}^{x} f(x, y)\, dx. \tag{8.11}$$

This equation satisfies the initial condition (8.10), as

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

The value of y is replaced by y_0 in the right hand side of (8.11) and let this solution be $y^{(1)}(x)$, the first approximation of y, i.e.,

$$y^{(1)}(x) = y_0 + \int_{x_0}^{x} f(x, y_0)\, dx. \tag{8.12}$$

Again, the value of $y^{(1)}(x)$ is replaced in (8.11) from (8.12) and the second approximation $y^{(2)}(x)$ is obtained as

$$y^{(2)}(x) = y_0 + \int_{x_0}^{x} f(x, y^{(1)})\, dx. \tag{8.13}$$

In this way, the following approximations of y are generated.

$$\begin{aligned}
y^{(3)}(x) &= y_0 + \int_{x_0}^{x} f(x, y^{(2)})\, dx \\
y^{(4)}(x) &= y_0 + \int_{x_0}^{x} f(x, y^{(3)})\, dx \\
&\cdots\cdots\cdots\cdots\cdots\cdots \\
y^{(n)}(x) &= y_0 + \int_{x_0}^{x} f(x, y^{(n-1)})\, dx.
\end{aligned}$$

Thus a sequence $y^{(1)}, y^{(2)}, \ldots, y^{(n)}$ of y is generated in terms of x.

Note 8.2.1 If $f(x, y)$ is continuous and has a bounded partial derivative $f_y(x, y)$ in the neighbourhood of (x_0, y_0), then in a certain interval containing the point x_0, the sequence $\{y^{(i)}\}$ converges to the function $y(x)$ which is the solution of the differential equation (8.9) with initial condition (8.10).

Example 8.2.1 Use Picard's method to solve the differential equation $y' = x^2 + y$ with initial condition $y(0) = 0$. Also, find the values of $y(0.1)$ and $y(0.2)$.

Solution. Let $f(x, y) = x^2 + y, x_0 = 0, y_0 = 0$.
Then

$$\begin{aligned}
y^{(1)}(x) &= y_0 + \int_{x_0}^{x} f(x, y_0)\, dx = 0 + \int_0^x (x^2 + 0)\, dx = \frac{x^3}{3} \\
y^{(2)}(x) &= y_0 + \int_{x_0}^{x} f(x, y^{(1)}(x))\, dx = \int_0^x \left(x^2 + \frac{x^3}{3}\right) dx = \frac{x^3}{3} + \frac{x^4}{3.4}
\end{aligned}$$

$$\begin{aligned} y^{(3)}(x) &= y_0 + \int_{x_0}^{x} f(x, y^{(2)}(x))\, dx = \int_0^x \left(x^2 + \frac{x^3}{3} + \frac{x^4}{3.4}\right) dx \\ &= \frac{x^3}{3} + \frac{x^4}{3.4} + \frac{x^5}{3.4.5}. \end{aligned}$$

Similarly,

$$y^{(4)}(x) = \frac{x^3}{3} + \frac{x^4}{3.4} + \frac{x^5}{3.4.5} + \frac{x^6}{3.4.5.6}.$$

Now,

$$\begin{aligned} y(0.1) &= \frac{(0.1)^3}{3} + \frac{(0.1)^4}{3 \times 4} + \frac{(0.1)^5}{3 \times 4 \times 5} + \frac{(0.1)^6}{3 \times 4 \times 5 \times 6} \\ &= 3.41836 \times 10^{-4}. \\ y(0.2) &= \frac{(0.2)^3}{3} + \frac{(0.2)^4}{3 \times 4} + \frac{(0.2)^5}{3 \times 4 \times 5} + \frac{(0.2)^6}{3 \times 4 \times 5 \times 6} \\ &= 2.80551 \times 10^{-3}. \end{aligned}$$

8.3 Euler's Method

This is the most simple but crude method to solve differential equation of the form

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0. \tag{8.14}$$

Let $x_1 = x_0 + h$, where h is small. Then by Taylor's series

$$\begin{aligned} y_1 = y(x_0 + h) &= y_0 + h\left(\frac{dy}{dx}\right)_{x_0} + \frac{h^2}{2}\left(\frac{d^2y}{dx^2}\right)_{c_1}, \\ &\text{where } c_1 \text{ lies between } x_0 \text{ and } x \\ &= y_0 + hf(x_0, y_0) + \frac{h^2}{2} y''(c_1) \end{aligned} \tag{8.15}$$

If the step size h is chosen small enough, then the second-order term may be neglected and hence y_1 is given by

$$y_1 = y_0 + hf(x_0, y_0). \tag{8.16}$$

Similarly,

$$y_2 = y_1 + hf(x_1, y_1) \tag{8.17}$$

$$y_3 = y_2 + hf(x_2, y_2) \tag{8.18}$$

and so on.

In general,

$$y_{n+1} = y_n + hf(x_n, y_n), \quad n = 0, 1, 2, \ldots \tag{8.19}$$

This method is very slow. To get a reasonable accuracy with Euler's methods, the value of h should be taken as small.

It may be noted that the Euler's method is a single-step explicit method.

Example 8.3.1 Find the values of $y(0.1)$ and $y(0.2)$ from the following differential equation

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

Solution. Let $h = 0.05$, $x_0 = 0, y_0 = 1$.
Then
$x_1 = x_0 + h = 0.05$
$y_1 = y(0.05) = y_0 + hf(x_0, y_0) = 1 + 0.05 \times (0 + 1) = 1.05$
$x_2 = x_1 + h = 0.1$
$y_2 = y(0.1) = y_1 + hf(x_1, y_1) = 1.05 + 0.05 \times (0.1^2 + 1.05^2) = 1.105625$
$x_3 = x_2 + h = 0.15$
$y_3 = y(0.15) = y_2 + hf(x_2, y_2) = 1.105625 + 0.05 \times (0.1^2 + 1.105625^2)$
$= 1.167245$
$x_4 = x_3 + h = 0.2$
$y_4 = y(0.2) = y_3 + hf(x_3, y_3) = 1.167245 + 0.05 \times (0.15^2 + 1.167245^2)$
$= 1.236493.$
Hence $y(0.1) = 1.105625, y(0.2) = 1.236493.$

8.3.1 Geometrical interpretation of Euler's method

The graphical representation of Euler's method is shown in Figure 8.1.

Geometrically, the desired function curve (solution curve) is approximated by a polygon train, where the direction of each part is determined by the value of the function $f(x, y)$ at its starting point.

Error

The local truncation error of Euler's method is $O(h^2)$, it follows obviously from (8.15). The neglected term at each step is $y''(c_i)\frac{h^2}{2}$. Then at the end of the interval $[x_0, x_n]$, after n steps, the global error is

$$\sum_{n=1}^{n} y''(c_i)\frac{h^2}{2} = ny''(c)\frac{h^2}{2} = \frac{hn}{2}y''(c)h = \frac{(x_n - x_0)y''(c)}{2}h = O(h).$$

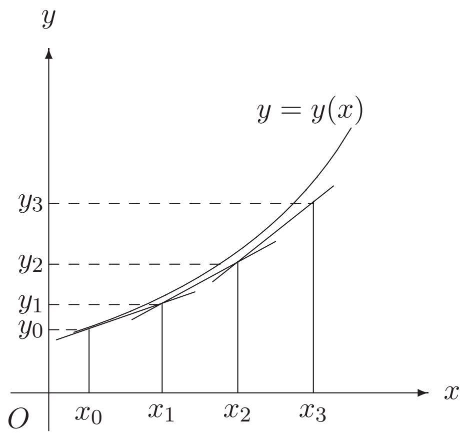


Figure 8.1: Geometrical meaning of Euler's method.

Algorithm 8.1 (Euler's method). This algorithm finds the solution of the equation $y' = f(x, y)$ with $y(x_0) = y_0$ over the interval $[x_0, x_n]$, by Euler's method

$$y_{i+1} = y_i + hf(x_i, y_i), \qquad i = 0, 1, 2, \ldots, n-1.$$

```
Algorithm Euler
Input function $f(x, y)$
Read $x_0, y_0, x_n, h$  //$x_0, y_0$ are the initial values and $x_n$ is the last value of $x$//
                         //where the process will terminate; $h$ is the step size//
for $x = x_0$ to $x_n$ step $h$ do
    $y = y_0 + h * f(x, y_0)$;
    Print $x, y$;
    $y_0 = y$;
endfor;
end Euler
```

Program 8.1

```
/* Program Euler
   Solution of a differential equation of the form y'=f(x,y),
   y(x0)=y0 by Euler's method. */
#include<stdio.h>
#include<math.h>
void main()
{
 float x0,y0,xn,h,x,y;
```

```
  float f(float x, float y);
  printf("Enter the initial (x0) and final (xn) values of x ");
  scanf("%f %f",&x0,&xn);
  printf("Enter initial value of y ");
  scanf("%f",&y0);
  printf("Enter step length h ");
  scanf("%f",&h);
  printf(" x-value   y-value\n");
  for(x=x0;x<xn;x+=h)
     {
       y=y0+h*f(x,y0);
       printf("%f  %f \n",x+h,y);
       y0=y;
     }
} /* main */
/* definition of the function f(x,y) */
float f(float x, float y)
{
   return(x*x+x*y+2);
}
```

A sample of input/output:

```
Enter the initial (x0) and final (xn) values of x
0 .2
Enter initial value of y 1
Enter step length h  .05
 x-value   y-value
0.050000  1.100000
0.100000  1.202875
0.150000  1.309389
0.200000  1.420335
```

8.4 Modified Euler's Method

In Euler's method there is no scope to improve the value of y. The improvement can be done using modified Euler's method.

Now, consider the differential equation

$$\frac{dy}{dx} = f(x, y) \text{ with } y(x_0) = y_0. \tag{8.20}$$

To obtain the solution at x_1, integrating (8.20) over $[x_0, x_1]$. That is

$$\int_{x_0}^{x_1} dy = \int_{x_0}^{x_1} f(x,y)\,dx$$

which gives

$$y_1 = y_0 + \int_{x_0}^{x_1} f(x,y)\,dx. \tag{8.21}$$

The integration of right hand side can be done using any numerical method. If the trapezoidal rule is used with step size $h(= x_1 - x_0)$ then above integration become

$$y(x_1) = y(x_0) + \frac{h}{2}[f(x_0, y(x_0)) + f(x_1, y(x_1))]. \tag{8.22}$$

Note that the right hand side of (8.22) involves an unknown quantity $y(x_1)$. This value can be determined by the Euler's method. Let us denote this value by $y^{(0)}(x_1)$ and the value obtained from (8.22) by $y^{(1)}(x_1)$. Then the resulting formula for finding y_1 is

$$y^{(1)}(x_1) = y(x_0) + \frac{h}{2}[f(x_0, y(x_0)) + f(x_1, y^{(0)}(x_1))].$$

That is,

$$\begin{aligned} y_1^{(0)} &= y_0 + hf(x_0, y_0) \\ y_1^{(1)} &= y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1^{(0)})]. \end{aligned} \tag{8.23}$$

This is the first approximation of y_1.
The second approximation is

$$y_1^{(2)} = y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1^{(1)})]. \tag{8.24}$$

The $(k+1)$th approximate value of y_1 is

$$y_1^{(k+1)} = y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1^{(k)})], k = 0, 1, 2, \ldots. \tag{8.25}$$

In general,

$$\begin{aligned} y_{i+1}^{(0)} &= y_i + hf(x_i, y_i) \\ y_{i+1}^{(k+1)} &= y_i + \frac{h}{2}[f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(k)})], \\ & k = 0, 1, 2, \ldots;\ \ i = 0, 1, 2, \ldots. \end{aligned} \tag{8.26}$$

The iterations are continued until two successive approximations $y_{i+1}^{(k)}$ and $y_{i+1}^{(k+1)}$ coincide to the desired accuracy. The iterations converge rapidly for sufficiently small spacing h.

Example 8.4.1 Determine the value of y when $x = 0.1$ and 0.2 given that

$$y(0) = 1 \text{ and } y' = x^2 - y.$$

Solution. Let $h = 0.1, x_0 = 0, y_0 = 1, x_1 = 0.1, x_2 = 0.2$ and $f(x, y) = x^2 - y$.

$$\begin{aligned}
y_1^{(0)} &= y_0 + hf(x_0, y_0) = 1 + 0.1f(0, 1) = 1 + 0.1 \times (0 - 1) = 0.9000.\\
y_1^{(1)} &= y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1^{(0)})]\\
&= 1 + \frac{0.1}{2}[(0^2 - 1) + (0.1^2 - 0.9)] = 0.9055.\\
y_1^{(2)} &= y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1^{(1)})]\\
&= 1 + \frac{0.1}{2}[(0^2 - 1) + (0.1^2 - 0.9055)] = 0.9052.\\
y_1^{(3)} &= y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1^{(2)})]\\
&= 1 + \frac{0.1}{2}[(0^2 - 1) + (0.1^2 - 0.9052)] = 0.9052.
\end{aligned}$$

Therefore, $y_1 = y(0.1) = 0.9052$.

$$\begin{aligned}
y_2^{(0)} &= y_1 + hf(x_1, y_1) = 0.9052 + 0.1f(0.1, 0.9052)\\
&= 0.9052 + 0.1 \times (0.1^2 - 0.9052) = 0.8157.\\
y_2^{(1)} &= y_1 + \frac{h}{2}[f(x_1, y_1) + f(x_2, y_2^{(0)})]\\
&= 0.9052 + \frac{0.1}{2}[(0.1^2 - 0.9052) + (0.2^2 - 0.8157)] = 0.8217.\\
y_2^{(2)} &= y_1 + \frac{h}{2}[f(x_1, y_1) + f(x_2, y_2^{(1)})]\\
&= 0.9052 + \frac{0.1}{2}[(0.1^2 - 0.9052) + (0.2^2 - 0.8217)] = 0.8214.\\
y_2^{(3)} &= y_1 + \frac{h}{2}[f(x_1, y_1) + f(x_2, y_2^{(2)})]\\
&= 0.9052 + \frac{0.1}{2}[(0.1^2 - 0.9052) + (0.2^2 - 0.8214)] = 0.8214.
\end{aligned}$$

Hence, $y_2 = y(0.2) = 0.8214$.

8.4.1 Geometrical interpretation of modified Euler's method

Let T_0 be the tangent at (x_0, y_0) on the solution curve $y = y(x)$, L_1 is the line passing through $(x_1, y_1^{(0)})$ of slope $f(x_1, y_1^{(0)})$ shown in Figure 8.2. Then $\overline{L}$ is the line passes through $C(x_1, y_1^{(0)})$ but with a slope equal to the average of $f(x_0, y_0)$ and $f(x_1, y_1^{(0)})$. The line L through (x_0, y_0) and parallel to $\overline{L}$ is the approximate curve to find the improved value $y_1^{(1)}$. The ordinate of the point B is the approximate value $y_1^{(1)}$.

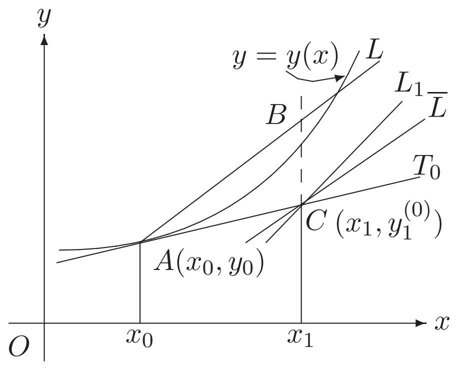


Figure 8.2: Geometrical meaning of modified Euler's method.

Again, the formula (8.23) can be interpreted as follows, by writing it in the form

$$y_1^{(1)} - y_0 = \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1^{(0)})]. \tag{8.27}$$

The first improvement, i.e., the right hand of (8.27) is the area of the trapezoid (see Figure 8.3) with the vertices $(x_0, 0), (x_0, f(x_0, y_0)), (x_1, f(x_1, y_1^{(0)}))$ and $(x_1, 0)$.

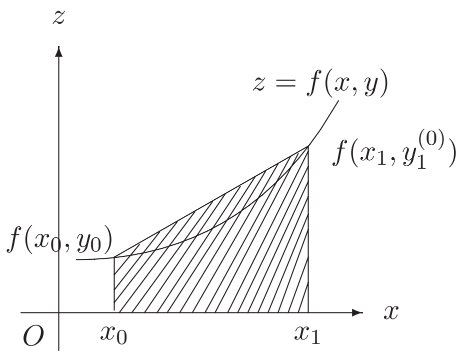


Figure 8.3: Geometrical interpretation of incremented value of Euler's method.

Error

The error of the trapezoidal rule is $-\frac{h^3}{12}y''(c_i)$. So the local truncation error of modified Euler's method is $O(h^3)$.

After n steps, the local accumulated error of this method is

$$-\sum_{i=1}^{n} \frac{h^3}{12}y''(c_i) \simeq -\frac{x_n - x_0}{12}y''(c)h^2 = O(h^2).$$

Thus the global truncation error is $O(h^2)$.

Algorithm 8.2 (Modified Euler's method). This algorithm solves the initial value problem $y' = f(x, y)$ with $y(x_0) = y_0$ over the interval $[x_0, x_n]$ with step size h. The formulae are given by

$$y_{i+1}^{(0)} = y_i + hf(x_i, y_i)$$
$$y_{i+1}^{(k)} = y_i + \frac{h}{2}[f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(k-1)})], \text{ for } k = 1, 2, \ldots$$

```
**Algorithm Modified_Euler**
Input function $f(x, y)$;
Read $x_0, x_n, y_0, h$;   //initial and final values of $x$, initial value of $y$ and step size $h$.//
Read $\varepsilon$;        //$\varepsilon$ is the error tolerance.//
Set $y = y_0$;
for $x = x_0$ to $x_n$ step $h$ do
    Compute $f_1 = f(x, y)$;
    Compute $y_c = y + h * f_1$;      //evaluated from Euler's method//
    do
        Set $y_p = y_c$;
        Compute $y_c = y + \frac{h}{2}[f_1 + f(x + h, y_p)]$      //modified Euler's method//
    while ($|y_p - y_c| > \varepsilon$)       //check for accuracy//
    Reset $y = y_c$;
    Print $x, y$;
endfor;
**end Modified_Euler**
```

Program 8.2
```
/* Program Modified Euler
   Solution of a differential equation of the form y'=f(x,y),
   y(x0)=y0 by Modified Euler's method. */
#include<stdio.h>
#include<math.h>
```

```
void main()
{
 float x0,y0,xn,h,x,y;/*x0, xn the initial and final values of x*/
   /* y0 initial value of y, h is the step length */
 float eps=1e-5; /* the error tolerance */
 float yc,yp,f1;
 float f(float x, float y);
 printf("Enter the initial (x0) and final (xn) values of x ");
 scanf("%f %f",&x0,&xn);
 printf("Enter initial value of y ");
 scanf("%f",&y0);
 printf("Enter step length h ");
 scanf("%f",&h);
 printf(" x-value    y-value\n");
 y=y0;
 for(x=x0;x<xn;x+=h)
    {
      f1=f(x,y);
      yc=y+h*f1; /* evaluated by Euler's method */
      do
      {
        yp=yc;
        yc=y+h*(f1+f(x+h,yp))/2; /*modified Euler's method*/
      }while(fabs(yp-yc)>eps);
      y=yc;
      printf("%f   %f\n",x+h,y);
    }
} /* main */
/* definition of the function f(x,y) */
float f(float x, float y)
 {
   return(x*x-2*y+1);
 }
```

A sample of input/output:

```
Enter the initial (x0) and final (xn) values of x
0 .5
Enter initial value of y 1
Enter step length h  .1
```

```
 x-value    y-value
0.100000   0.909546
0.200000   0.837355
0.300000   0.781926
0.400000   0.742029
0.500000   0.716660
```

8.5 Runge-Kutta Methods

The Euler's method is less efficient in practical problems because if h is not sufficiently small then this method gives inaccurate result.

The Runge-Kutta methods give more accurate result. One advantage of this method is it requires only the value of the function at some selected points on the subinterval and it is stable, and easy to program.

The Runge-Kutta methods perform several function evaluations at each step and avoid the computation of higher order derivatives. These methods can be constructed for any order, i.e., second, third, fourth, fifth, etc. The fourth-order Runge-Kutta method is more popular. These methods are single-step explicit methods.

8.5.1 Second-order Runge-Kutta method

The modified Euler's method to compute y_1 is

$$y_1 = y_0 + \frac{h}{2}[f(x_0, y_0) + f(x_1, y_1^{(0)})]. \tag{8.28}$$

If $y_1^{(0)} = y_0 + hf(x_0, y_0)$ is substituted in (8.28) then

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

Setting,

$$\begin{aligned} &k_1 = hf(x_0, y_0) \text{ and} \\ &k_2 = hf(x_0 + h, y_0 + hf(x_0, y_0)) = hf(x_0 + h, y_0 + k_1). \end{aligned} \tag{8.29}$$

Then equation (8.28) becomes

$$y_1 = y_0 + \frac{1}{2}(k_1 + k_2). \tag{8.30}$$

This is known as second-order Runge-Kutta formula. The local truncation error of this formula is of $O(h^3)$.

General derivation

Assume that the solution is of the form

$$y_1 = y_0 + ak_1 + bk_2, \tag{8.31}$$

where

$$k_1 = hf(x_0, y_0) \text{ and}$$
$$k_2 = hf(x_0 + \alpha h, y_0 + \beta k_1), a, b, \alpha \text{ and } \beta \text{ are constants.}$$

By Taylor's series,

$$\begin{aligned} y_1 &= y(x_0 + h) = y_0 + hy_0' + \frac{h^2}{2}y_0'' + \frac{h^3}{6}y_0''' + \cdots \\ &= y_0 + hf(x_0, y_0) + \frac{h^2}{2}\Big[\Big(\frac{\partial f}{\partial x}\Big)_{(x_0,y_0)} + f(x_0,y_0)\Big(\frac{\partial f}{\partial y}\Big)_{(x_0,y_0)}\Big] + O(h^3) \\ &\Big[\text{As } \frac{df}{dx} = \frac{\partial f}{\partial x} + f(x,y)\frac{\partial f}{\partial y}\Big] \\ k_2 &= hf(x_0 + \alpha h, y_0 + \beta k_1) \\ &= h\Big[f(x_0,y_0) + \alpha h\Big(\frac{\partial f}{\partial x}\Big)_{(x_0,y_0)} + \beta k_1\Big(\frac{\partial f}{\partial y}\Big)_{(x_0,y_0)} + O(h^2)\Big] \\ &= hf(x_0,y_0) + \alpha h^2\Big(\frac{\partial f}{\partial x}\Big)_{(x_0,y_0)} + \beta h^2 f(x_0,y_0)\Big(\frac{\partial f}{\partial y}\Big)_{(x_0,y_0)} + O(h^3). \end{aligned}$$

Then the equation (8.31) becomes

$$\begin{aligned} &y_0 + hf(x_0,y_0) + \frac{h^2}{2}[f_x(x_0,y_0) + f(x_0,y_0)f_y(x_0,y_0)] + O(h^3) \\ &\quad = y_0 + (a+b)hf(x_0,y_0) + bh^2[\alpha f_x(x_0,y_0) + \beta f(x_0,y_0)f_y(x_0,y_0)] + O(h^3). \end{aligned}$$

The coefficients of f, f_x and f_y are compared and the following equations are obtained.

$$a + b = 1, \quad b\alpha = \frac{1}{2} \text{ and } b\beta = \frac{1}{2}. \tag{8.32}$$

Obviously, $\alpha = \beta$ and if α is assigned any value arbitrarily, then the remaining parameters can be determined uniquely. However, usually the parameters are chosen as $\alpha = \beta = 1$, then $a = b = \frac{1}{2}$.

Thus the formula is

$$y_1 = y_0 + \frac{1}{2}(k_1 + k_2) + O(h^3), \tag{8.33}$$

where $k_1 = hf(x_0, y_0)$ and $k_2 = hf(x_0 + h, y_0 + k_1)$.

It follows that there are several second-order Runge-Kutta formulae and (8.33) is just one among such formulae.

8.5.2 Fourth-order Runge-Kutta Method

The fourth-order Runge-Kutta formula is

$$y_1 = y_0 + ak_1 + bk_2 + ck_3 + dk_4, \tag{8.34}$$

where

$$\begin{aligned} k_1 &= hf(x_0, y_0) \\ k_2 &= hf(x_0 + \alpha_0 h, y_0 + \beta_0 k_1) \\ k_3 &= hf(x_0 + \alpha_1 h, y_0 + \beta_1 k_1 + \gamma_1 k_2) \\ \text{and} \quad k_4 &= hf(x_0 + \alpha_2 h, y_0 + \beta_2 k_1 + \gamma_2 k_2 + \delta_1 k_3). \end{aligned} \tag{8.35}$$

The parameters $a, b, c, d, \alpha_0, \beta_0, \alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2, \gamma_2, \delta_1$ are to be determined by expanding both sides of (8.34) by Taylor's series retaining the terms up to and including those containing h^4. The choice of the parameters is arbitrary and depending on the choice of parameters, several fourth-order Runge-Kutta formulae can be generated.

The above functions are expanded by using Taylor's series method retaining terms up to fourth order. The local truncation error is $O(h^5)$. Runge and Kutta obtained the following system of equations.

$$\left.\begin{aligned} &\beta_0 &&= \alpha_0 \\ &\beta_1 + \gamma_1 &&= \alpha_1 \\ &\beta_2 + \gamma_2 + \delta_1 &&= \alpha_2 \\ &a + b + c + d &&= 1 \\ &b\alpha_0 + c\alpha_1 + d\alpha_2 &&= 1/2 \\ &b\alpha_0^2 + c\alpha_1^2 + d\alpha_2^2 &&= 1/3 \\ &b\alpha_0^3 + c\alpha_1^3 + d\alpha_2^3 &&= 1/4 \\ &c\alpha_0\gamma_1 + d(\alpha_0\gamma_2 + \alpha_1\delta_1) &&= 1/6 \\ &c\alpha_0\alpha_1\gamma_1 + d\alpha_2(\alpha_0\gamma_2 + \alpha_1\delta_1) &&= 1/8 \\ &c\alpha_0^2\gamma_1 + d(\alpha_0^2\gamma_2 + \alpha_1^2\delta_1) &&= 1/12 \\ &d\alpha_0\gamma_1\delta_1 &&= 1/24. \end{aligned}\right\} \tag{8.36}$$

The above system of equations contain 11 equations and 13 unknowns. Two more conditions are required to solve the system. But, these two conditions are taken arbitrarily. The most common choice is

$$\alpha_0 = \frac{1}{2}, \qquad \beta_1 = 0.$$

With these choice, the solution of the system (8.36) is

$$\begin{aligned} &\alpha_1 = \frac{1}{2}, \alpha_2 = 1, \beta_0 = \frac{1}{2}, \gamma_1 = \frac{1}{2}, \beta_2 = 0, \gamma_2 = 0, \delta_1 = 1, \\ &a = \frac{1}{6}, b = c = \frac{1}{3}, d = \frac{1}{6}. \end{aligned} \tag{8.37}$$

The values of these variables are substituted in (8.34) and (8.35) and the fourth-order Runge-Kutta method is obtained as

$$y_1 = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{8.38}$$

where

$$\begin{aligned} k_1 &= hf(x_0, y_0) \\ k_2 &= hf(x_0 + h/2, y_0 + k_1/2) \\ k_3 &= hf(x_0 + h/2, y_0 + k_2/2) \\ k_4 &= hf(x_0 + h, y_0 + k_3). \end{aligned}$$

Starting with the initial point (x_0, y_0), one can generate the sequence of solutions at $x_1, x_2, \ldots$ using the formula

$$y_{i+1} = y_i + \frac{1}{6}(k_1^{(i)} + 2k_2^{(i)} + 2k_3^{(i)} + k_4^{(i)}) \tag{8.39}$$

where

$$\begin{aligned} k_1^{(i)} &= hf(x_i, y_i) \\ k_2^{(i)} &= hf(x_i + h/2, y_i + k_1^{(i)}/2) \\ k_3^{(i)} &= hf(x_i + h/2, y_i + k_2^{(i)}/2) \\ k_4^{(i)} &= hf(x_i + h, y_i + k_3^{(i)}). \end{aligned}$$

Example 8.5.1 Given $y' = y^2 - x^2$, where $y(0) = 2$. Find $y(0.1)$ and $y(0.2)$ by second-order Runge-Kutta method.

Solution. Here $h = 0.1, x_0 = 0, y_0 = 2, f(x, y) = y^2 - x^2$.
Then

$$\begin{aligned} k_1 &= hf(x_0, y_0) = 0.1(2^2 - 0^2) = 0.4000. \\ k_2 &= hf(x_0 + h, y_0 + k_1) = 0.1 \times f(0 + 0.1, 2 + 0.4000) \\ &= 0.1 \times (2.4^2 - 0.1^2) = 0.5750. \end{aligned}$$

Therefore, $y_1 = y_0 + \frac{1}{2}(k_1 + k_2) = 2 + \frac{1}{2}(0.4000 + 0.5750) = 2.4875$, *i.e.*, $y(0.1) = 2.4875$.
To determine $y_2 = y(0.2)$, let $x_1 = 0.1$ and $y_1 = 2.4875$.

$$\begin{aligned} k_1 &= hf(x_1, y_1) = 0.1 \times f(0.1, 2.4875) = 0.1 \times (2.4875^2 - 0.1^2) \\ &= 0.6178. \\ k_2 &= hf(x_1 + h, y_1 + k_1) = 0.1 \times f(0.2, 2.4875 + 0.6178) \\ &= 0.1 \times f(0.2, 3.1053) = 0.1 \times (3.1053^2 - 0.2^2) = 0.9603. \end{aligned}$$

Therefore, $y_2 = y_1 + \frac{1}{2}(k_1 + k_2) = 2.4875 + \frac{1}{2}(0.6178 + 0.9603) = 3.2766$.
Hence, $y(0.2) = 3.2766$.

Example 8.5.2 Given $y' = x^2 + y^2$ with $x = 0, y = 1$. Find $y(0.1)$ by fourth-order Runge-Kutta method.

Solution. Here $h = 0.1, x_0 = 0, y_0 = 1, f(x, y) = x^2 + y^2$.

$$\begin{aligned}
k_1 &= hf(x_0, y_0) = 0.1 \times (0^2 + 1^2) = 0.1000.\\
k_2 &= hf(x_0 + h/2, y_0 + k_1/2) = 0.1 \times f(0.05, 1.05)\\
&= 0.1 \times (0.05^2 + 1.05^2) = 0.1105.\\
k_3 &= hf(x_0 + h/2, y_0 + k_2/2) = 0.1 \times f(0.05, 1.0553)\\
&= 0.1 \times (0.05^2 + 1.0553^2) = 0.1116.\\
k_4 &= hf(x_0 + h, y_0 + k_3) = 0.1 \times f(0.1, 1.1116)\\
&= 0.1 \times (0.1^2 + 1.1116^2) = 0.1246.
\end{aligned}$$

Therefore,

$$\begin{aligned}
y_1 &= y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\\
&= 1 + \frac{1}{6}(0.1000 + 2 \times 0.1105 + 2 \times 0.1116 + 0.1246)\\
&= 1.1115.
\end{aligned}$$

Note 8.5.1 The Runge-Kutta method gives better result, though, it has some disadvantages. This method uses numerous calculations of function to find y_{i+1}. When the function $f(x, y)$ has a complicated analytic form then the Runge-Kutta method is very laborious.

Geometrical interpretation of $\mathbf{k_1, k_2, k_3, k_4}$

Let ABC be the solution curve (Figure 8.4) and B is the point on the curve at the ordinate $x_i + h/2$ which is the middle point of the interval $[x_i, x_i + h]$. Let AL_1T_1 be the tangent drawn at A makes an angle θ_1 with the horizontal line AT_0. L_1 and T_1 are the points of intersection with the ordinates BD and CC_0. Then the number k_1 is the approximate slope (within the factor h) of the tangent at A of the solution curve ABC, i.e., $k_1 = hy_1' = hf(x_i, y_i)$. The coordinates of L_1 is $(x_i + h/2, y_i + k_1/2)$. The number k_2 is the approximate slope (within the factor h) of the tangent drawn to the curve ABC at L_1. A straight line AL_2 is drawn parallel to the line segment L_1T_2. Then the coordinates of L_2 is $(x_i + h/2, y_i + k_2/2)$. The number k_3 is the approximate slope (within factor h) of the tangent to the curve ABC at the point L_2. Finally, a straight line is drawn through A and parallel to L_2T_3, which cuts the extension of ordinates C_0T_4 at T_4. The coordinates of T_4 are $(x_i + h, y_i + k_3)$. Then k_4 is the approximate slope (within the factor h) of the tangent drawn to the curve ABC at T_4.

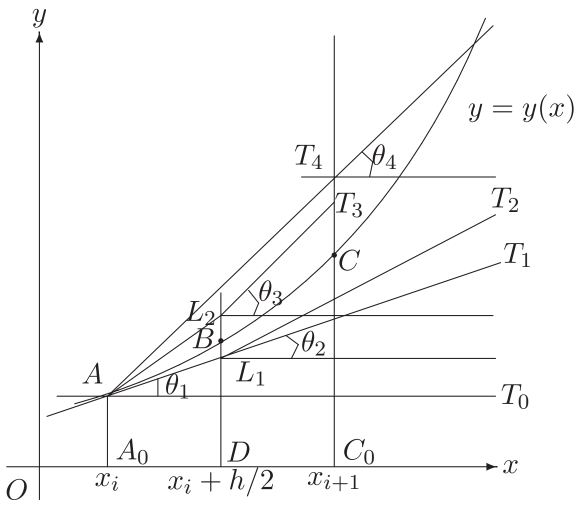


Figure 8.4: Interpretation of k_1, k_2, k_3, k_4.

Error

The fourth-order Runge-Kutta formula is

$$y_1 = y_0 + \frac{(h/2)}{3}\Big[k_1 + \frac{4(k_2 + k_3)}{2} + k_4\Big].$$

This is similar to the Simpson's formula with step size $h/2$, so the local truncation error of this formula is $-\frac{h^5}{2880}y^{iv}(c_1)$, i.e., of $O(h^5)$ and after n steps, the accumulated error is

$$-\sum_{i=1}^{n} \frac{h^5}{2880}y^{iv}(c_i) = -\frac{x_n - x_0}{5760}y^{iv}(c)h^4 = O(h^4).$$

Algorithm 8.3 (Fourth-order Runge-Kutta method). This algorithm finds the solution of the differential equation $y' = f(x, y)$ with $y(x_0) = y_0$ using fourth-order Runge-Kutta method, i.e., using the formula

$$y_{i+1} = y_i + \frac{1}{6}[k_1 + 2(k_2 + k_3) + k_4]$$

within the interval $[x_1, x_n]$ at step h.

Algorithm RK4

```
Input function $f(x, y)$;
Read $x_0, x_n, y_0, h$;        //initial and final value of $x$, initial value of $y$ and step size.//
Set $y = y_0$;
for $x = x_0$ to $x_n$ step $h$ do
    Compute $k_1 = h * f(x, y)$;
    Compute $k_2 = h * f(x + h/2, y + k_1/2)$;
    Compute $k_3 = h * f(x + h/2, y + k_2/2)$;
    Compute $k_4 = h * f(x + h, y + k_3)$;
    Compute $y = y + [k_1 + 2(k_2 + k_3) + k_4]/6$;
    Print $x, y$;
endfor;
end RK4
```

Program 8.3

```c
/* Program Fourth Order Runge-Kutta
   Solution of a differential equation of the form y'=f(x,y),
   y(x0)=y0 by fourth order Runge-Kutta method. */
#include<stdio.h>
#include<math.h>
void main()
{
 float x0,y0,xn,h,x,y,k1,k2,k3,k4;
 float f(float x, float y);
 printf("Enter the initial values of x and y ");
 scanf("%f %f",&x0,&y0);
 printf("Enter last value of x ");
 scanf("%f",&xn);
 printf("Enter step length h ");
 scanf("%f",&h);
 y=y0;
 printf(" x-value    y-value\n");

 for(x=x0;x<xn;x+=h)
    {
      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+k3)+k4)/6;
      printf("%f   %f\n",x+h,y);
    }
} /* main */
/* definition of the function f(x,y) */
float f(float x, float y)
 {
   return(x*x-y*y+y);
 }
```

A sample of input/output:

```
Enter the initial values of x and y 0 2
Enter last value of x 0.5
Enter step length h 0.1
 x-value    y-value
0.100000   1.826528
0.200000   1.695464
0.300000   1.595978
0.400000   1.521567
0.500000   1.468221
```

8.5.3 Runge-Kutta method for a pair of equations

The Runge-Kutta methods may also be used to solve a pair of first order differential equations.

Consider a pair of first-order differential equations

$$\begin{aligned}\frac{dy}{dx} &= f(x,y,z)\\ \frac{dz}{dx} &= g(x,y,z)\end{aligned} \tag{8.40}$$

with initial conditions

$$x = x_0,\ \ y(x_0) = y_0,\ \ z(x_0) = z_0. \tag{8.41}$$

Then the values of y_i and z_i at x_i are obtained by the formulae

$$\begin{aligned}y_{i+1} &= y_i + \frac{1}{6}[k_1^{(i)} + 2k_2^{(i)} + 2k_3^{(i)} + k_4^{(i)}],\\ z_{i+1} &= z_i + \frac{1}{6}[l_1^{(i)} + 2l_2^{(i)} + 2l_3^{(i)} + l_4^{(i)}],\end{aligned} \tag{8.42}$$

$$\text{where}\quad \begin{aligned}
k_1^{(i)} &= hf(x_i, y_i, z_i)\\
l_1^{(i)} &= hg(x_i, y_i, z_i)\\
k_2^{(i)} &= hf(x_i + h/2, y_i + k_1^{(i)}/2, z_i + l_1^{(i)}/2)\\
l_2^{(i)} &= hg(x_i + h/2, y_i + k_1^{(i)}/2, z_i + l_1^{(i)}/2)\\
k_3^{(i)} &= hf(x_i + h/2, y_i + k_2^{(i)}/2, z_i + l_2^{(i)}/2)\\
l_3^{(i)} &= hg(x_i + h/2, y_i + k_2^{(i)}/2, z_i + l_2^{(i)}/2)\\
k_4^{(i)} &= hf(x_i + h, y_i + k_3^{(i)}, z_i + l_3^{(i)})\\
l_4^{(i)} &= hg(x_i + h, y_i + k_3^{(i)}, z_i + l_3^{(i)}).
\end{aligned} \tag{8.43}$$

Example 8.5.3 Solve the following pair of differential equations $\dfrac{dy}{dx} = \dfrac{x+y}{z}$ and $\dfrac{dz}{dx} = xy + z$ with initial conditions $x_0 = 0.5, y_0 = 1.5, z_0 = 1$ for $x = 0.6$.

Solution. Let $h = 0.1$ and calculate the values of k_1, k_2, k_3, k_4; l_1, l_2, l_3, l_4.
Here $f(x, y) = \dfrac{x+y}{z}$, $g(x, y) = xy + z$.

$k_1 = hf(x_0, y_0, z_0) = 0.1 \times \dfrac{0.5 + 1.5}{1} = 0.2$

$l_1 = hg(x_0, y_0, z_0) = 0.1 \times (0.5 \times 1.5 + 1) = 0.175$

$k_2 = hf(x_0 + h/2, y_0 + k_1/2, z_0 + l_1/2) = 0.1 \times \dfrac{0.55 + 1.6}{1.0875} = 0.197701$

$l_2 = hg(x_0 + h/2, y_0 + k_1/2, z_0 + l_1/2) = 0.1 \times (0.55 \times 1.6 + 1.0875)$
$= 0.19675$

$k_3 = hf(x_0 + h/2, y_0 + k_2/2, z_0 + l_2/2) = 0.1 \times \dfrac{0.55 + 1.59885}{1.098375} = 0.195639$

$l_3 = hg(x_0 + h/2, y_0 + k_2/2, z_0 + l_2/2) = 0.1 \times (0.55 \times 1.59885 + 1.098375)$
$= 0.197774$

$k_4 = hf(x_0 + h, y_0 + k_3, z_0 + l_3) = 0.1 \times \dfrac{0.6 + 1.695639}{1.197774} = 0.191659$

$l_4 = hg(x_0 + h, y_0 + k_3, z_0 + l_3) = 0.1 \times (0.6 \times 1.695639 + 1.197774)$
$= 0.221516.$

Hence,

$$\begin{aligned}
y(0.6) = y_1 &= y_0 + \frac{1}{6}[k_1 + 2(k_2 + k_3) + k_4]\\
&= 1.5 + \frac{1}{6}[0.2 + 2(0.197701 + 0.195639) + 0.191659] = 1.696390.\\
z(0.6) = z_1 &= z_0 + \frac{1}{6}[l_1 + 2(l_2 + l_3) + l_4]\\
&= 1.0 + \frac{1}{6}[0.175 + 2(0.19675 + 0.197774) + 0.221516] = 1.197594.
\end{aligned}$$

Algorithm 8.4 (Runge-Kutta method for a pair of equations). A pair of first order differential equation of the form $y' = f(x, y, z), z' = g(x, y, z)$ with initial conditions $x = x_0, y(x_0) = y_0$ and $z(x_0) = z_0$ can be solved by this algorithm using fourth-order Runge-Kutta method. The formulae are

$$y_{i+1} = y_i + \frac{1}{6}[k_1 + 2(k_2 + k_3) + k_4],$$
$$z_{i+1} = z_i + \frac{1}{6}[l_1 + 2(l_2 + l_3) + l_4]$$

where

$k_1 = hf(x_i, y_i, z_i)$
$l_1 = hg(x_i, y_i, z_i)$
$k_2 = hf(x_i + h/2, y_i + k_1/2, z_i + l_1/2)$
$l_2 = hg(x_i + h/2, y_i + k_1/2, z_i + l_1/2)$
$k_3 = hf(x_i + h/2, y_i + k_2/2, z_i + l_2/2)$
$l_3 = hg(x_i + h/2, y_i + k_2/2, z_i + l_2/2)$
$k_4 = hf(x_i + h, y_i + k_3, z_i + l_3)$
$l_4 = hg(x_i + h, y_i + k_3, z_i + l_3)$.

```
Algorithm RK4_Pair
Input functions $f(x, y)$ and $g(x, y)$;
Read $x_0, y_0, z_0, h, x_n$;   //initial values of $x, y, z$; step size and final value of $x$.//
Set $y = y_0$;
Set $z = z_0$;
for $x = x_0$ to $x_n$ step $h$ do
    Compute the following
    $k_1 = hf(x, y, z)$;
    $l_1 = hg(x, y, z)$;
    $k_2 = hf(x + h/2, y + k_1/2, z + l_1/2)$;
    $l_2 = hg(x + h/2, y + k_1/2, z + l_1/2)$;
    $k_3 = hf(x + h/2, y + k_2/2, z + l_2/2)$;
    $l_3 = hg(x + h/2, y + k_2/2, z + l_2/2)$;
    $k_4 = hf(x + h, y + k_3, z + l_3)$;
    $l_4 = hg(x + h, y + k_3, z + l_3)$;

    $y = y + [k_1 + 2(k_2 + k_3) + k_4]/6$;
    $z = z + [l_1 + 2(l_2 + l_3) + l_4]/6$;
    Print $x, y, z$;
endfor;
end RK4_Pair
```

Program 8.4

```
/* Program Runge-Kutta (for Pair of Equations)
   Solution of a differential equation of the form y'=f(x,y,z),
   z'=g(x,y,z) with x=x0, y(x0)=y0 and z(x0)=z0 by fourth order
   Runge-Kutta method.
   Here the equations are taken as y'=y+2z, z'=3y+2z with
   y(0)=6, z(0)=4. */
#include<stdio.h>
#include<math.h>
void main()
{
 float x0,y0,z0,xn,h,x,y,z,k1,k2,k3,k4,l1,l2,l3,l4;
 float f(float x, float y, float z);
 float g(float x, float y, float z);
 printf("Enter the initial values of x, y and z ");
 scanf("%f %f %f",&x0,&y0,&z0);
 printf("Enter last value of x ");
 scanf("%f",&xn);
 printf("Enter step length h ");
 scanf("%f",&h);
 y=y0;
 z=z0;
 printf("x-value    y-value    z-value\n");
 for(x=x0;x<xn;x+=h)
    {
      k1=h*f(x,y,z);
      l1=h*g(x,y,z);
      k2=h*f(x+h/2,y+k1/2,z+l1/2);
      l2=h*g(x+h/2,y+k1/2,z+l1/2);
      k3=h*f(x+h/2,y+k2/2,z+l2/2);
      l3=h*g(x+h/2,y+k2/2,z+l2/2);
      k4=h*f(x+h,y+k3,z+l3);
      l4=h*g(x+h,y+k3,z+l3);

      y=y+(k1+2*(k2+k3)+k4)/6;
      z=z+(l1+2*(l2+l3)+l4)/6;
      printf("%f   %f   %f\n",x+h,y,z);
    }
} /* main */
```

```
/* definition of the function f(x,y,z) */
 float f(float x, float y, float z)
  {
   return(y+2*z);
  }
/* definition of the function g(x,y,z) */
 float g(float x, float y, float z)
  {
   return(3*y+2*z);
  }
```

A sample of input/output:

```
Enter the initial values of x, y and z
0 6 4
Enter last value of x 0.4
Enter step length h 0.1
x-value    y-value    z-value
0.100000   7.776608   7.140725
0.200000   10.538535   11.714149
0.300000   14.759665   18.435406
0.400000   21.147917   28.370275
```

8.5.4 Runge-Kutta method for a system of equations

Let

$$\begin{aligned}\frac{dy_1}{dx} &= f_1(x, y_1, y_2, \ldots, y_n)\\ \frac{dy_2}{dx} &= f_2(x, y_1, y_2, \ldots, y_n)\\ \cdots\;\; & \;\;\cdots\cdots\cdots\cdots\cdots\\ \frac{dy_n}{dx} &= f_n(x, y_1, y_2, \ldots, y_n)\end{aligned} \tag{8.44}$$

with initial conditions

$$y_1(x_0) = y_{10},\;\; y_2(x_0) = y_{20},\;\; \ldots,\;\; y_n(x_0) = y_{n0} \tag{8.45}$$

be a system of first order differential equations, where $y_1, y_2, \ldots, y_n$ are n dependent variables and x is the only independent variable.

The above system can be written as

$$\frac{d\mathbf{y}}{dx} = \mathbf{f}(x, y_1, y_2, \ldots, y_n) \text{ with } \mathbf{y}(x_0) = \mathbf{y_0}. \tag{8.46}$$

The fourth-order Runge-Kutta method for the above system is

$$\mathbf{y_{j+1}} = \mathbf{y_j} + \frac{1}{6}[\mathbf{k_1} + 2\mathbf{k_2} + 2\mathbf{k_3} + \mathbf{k_4}] \tag{8.47}$$

where

$$\mathbf{k_1} = \begin{bmatrix} k_{11} \\ k_{21} \\ \vdots \\ k_{n1} \end{bmatrix}, \mathbf{k_2} = \begin{bmatrix} k_{12} \\ k_{22} \\ \vdots \\ k_{n2} \end{bmatrix}, \mathbf{k_3} = \begin{bmatrix} k_{13} \\ k_{23} \\ \vdots \\ k_{n3} \end{bmatrix}, \mathbf{k_4} = \begin{bmatrix} k_{14} \\ k_{24} \\ \vdots \\ k_{n4} \end{bmatrix} \tag{8.48}$$

and

$$\begin{aligned}
k_{i1} &= hf_i(x_j, y_{1j}, y_{2j}, \ldots, y_{nj}) \\
k_{i2} &= hf_i(x_j + h/2, y_{1j} + k_{11}/2, y_{2j} + k_{21}/2, \ldots, y_{nj} + k_{n1}/2) \\
k_{i3} &= hf_i(x_j + h/2, y_{1j} + k_{12}/2, y_{2j} + k_{22}/2, \ldots, y_{nj} + k_{n2}/2) \\
k_{i4} &= hf_i(x_j + h, y_{1j} + k_{13}, y_{2j} + k_{23}, \ldots, y_{nj} + k_{n3})
\end{aligned}$$

for $i = 1, 2, \ldots, n$.

It may be noted that y_{kj} is the value of the kth dependent variable y_k evaluated at x_j. The above formula in explicit vector notation is

$$\begin{bmatrix} y_{1\,j+1} \\ y_{2\,j+1} \\ \vdots \\ y_{n\,j+1} \end{bmatrix} = \begin{bmatrix} y_{1\,j} \\ y_{2\,j} \\ \vdots \\ y_{n\,j} \end{bmatrix} + \frac{1}{6}\left(\begin{bmatrix} k_{11} \\ k_{21} \\ \vdots \\ k_{n1} \end{bmatrix} + 2\begin{bmatrix} k_{12} \\ k_{22} \\ \vdots \\ k_{n2} \end{bmatrix} + 2\begin{bmatrix} k_{13} \\ k_{23} \\ \vdots \\ k_{n3} \end{bmatrix} + \begin{bmatrix} k_{14} \\ k_{24} \\ \vdots \\ k_{n4} \end{bmatrix} \right). \tag{8.49}$$

8.5.5 Runge-Kutta method for second order differential equation

Let the second order differential equation be

$$a(x)y'' + b(x)y' + c(x)y = f(x). \tag{8.50}$$

This equation can be written as

$$y''(x) = g(x, y(x), y'(x)) \tag{8.51}$$

with initial conditions

$$x = x_0, \;\; y(x_0) = y_0, \;\; y'(x_0) = z_0. \tag{8.52}$$

This second order differential equation can be converted as a pair of first order differential equations by substituting

$$y'(x) = z(x).$$

Then $y''(x) = z'(x)$. The equation (8.51) becomes

$$\frac{dy}{dx} = z$$
$$\frac{dz}{dx} = g(x, y, z) \tag{8.53}$$

with initial conditions

$$x = x_0, \quad y(x_0) = y_0, \quad z(x_0) = z_0. \tag{8.54}$$

Now, Runge-Kutta methods may be used to solve the equations (8.53) with initial conditions (8.54). This system of equations generate two sequences $\{y_i\}$ and $\{z_i\}$. The first sequence is the solution of (8.51). It may be noted that the value of z represents the derivative at the same point.

Example 8.5.4 Find the value of $y(0.1)$ for the following second order differential equation by fourth-order Runge-Kutta method.

$$2y''(x) - 5y'(x) - 3y(x) = 45e^{2x} \text{ with } y(0) = 2, y'(0) = 1.$$

Solution. Substituting $y' = z$. Then the given equation reduces to $2z' - 5z - 3y = 45e^{2x}$ or, $z' = \frac{1}{2}(5z + 3y + 45e^{2x}) = g(x, y, z)$, (say).
$x_0 = 0, y_0 = 2, z_0 = 1,\ h = 0.1.$
$k_1 = h \times z_0 = 0.1$
$l_1 = hg(x_0, y_0, z_0) = 0.1 \times g(0, 2, 1) = 2.80$
$k_2 = h \times (z_0 + l_1/2) = 0.1 \times 2.4 = 0.24$
$l_2 = hg(x_0 + h/2, y_0 + k_1/2, z_0 + l_1/2) = 0.1 \times g(0.05, 2.05, 2.4) = 3.394135$
$k_3 = h \times (z_0 + l_2/2) = 0.1 \times 2.697067 = 0.269707$
$l_3 = hg(x_0 + h/2, y_0 + k_2/2, z_0 + l_2/2) = 0.1 \times g(0.05, 2.12, 2.697067)$
$\quad = 3.478901$
$k_4 = h \times (z_0 + l_3) = 0.1 \times 4.478901 = 0.447890$
$l_4 = hg(x_0 + h, y_0 + k_3, z_0 + l_3) = 0.1 \times g(0.1, 2.269707, 4.478901) = 4.208338$
Therefore,
$y_1 = y_0 + \frac{1}{6}[k_1 + 2(k_2 + k_3) + k_4]$
$\quad = 2 + \frac{1}{6}[0.1 + 2(0.24 + 0.269707) + 0.447890] = 2.261217$
$z_1 = z_0 + \frac{1}{6}[l_1 + 2(l_2 + l_3) + l_4]$
$\quad = 1 + \frac{1}{6}[2.80 + 2(3.394135 + 3.478901) + 4.208338] = 4.459068.$
The required value of $y(0.1)$ is 2.261217. In addition, $y'(0.1)$ is 4.459068.

8.5.6 Runge-Kutta-Fehlberg method

Runge-Kutta methods have become very popular both as computational techniques and as a topic for research. Many variant of Runge-Kutta methods are available. Some of them are RK-Butcher, RK-Fehlberg, RK-Merson, RK-Centroidal mean, RK-arithmetic mean etc. The Runge-Kutta-Fehlberg method gives better accuracy by solving the problem twice using step sizes h and $h/2$. In this method, at each step two different approximations for the solution are calculated. If the two approximations are closed then the solution is obtained. If the approximation is not acceptable then the step size is reduced.

In each step the following six values are required.

$$\begin{aligned}
k_1 &= hf(x_i, y_i) \\
k_2 &= hf\left(x_i + \frac{h}{4}, y_i + \frac{k_1}{4}\right) \\
k_3 &= hf\left(x_i + \frac{3h}{8}, y_i + \frac{3}{32}k_1 + \frac{9}{32}k_2\right) \\
k_4 &= hf\left(x_i + \frac{12}{13}h, y_i + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right) \\
k_5 &= hf\left(x_i + h, y_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right) \\
k_6 &= hf\left(x_i + \frac{h}{2}, y_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right).
\end{aligned} \tag{8.55}$$

Then an approximation using fourth-order Runge-Kutta method is

$$y_{i+1} = y_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5. \tag{8.56}$$

It may be noted that the value of k_2 is not used in the above formula. The other value of y is determined by fifth order Runge-Kutta method as follows:

$$y^*_{i+1} = y_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6. \tag{8.57}$$

If $|y_{i+1} - y^*_{i+1}|$ is small enough then the method is terminated; otherwise, the computation is repeated by reducing the step size h. The local truncation error of this method is $y_{i+1} - y^*_{i+1}$.

8.5.7 Runge-Kutta-Butcher method[1]

RK-Butcher method is normally considered to be sixth order since it requires six function evaluations (it looks like a sixth-order method, but it is a fifth-order method only), but

[1]J.C.Butcher, The Numerical Analysis of Ordinary Differential Equation: Runge-Kutta and General Linear Methods, (Chichester, John Wiley), 1987.

in practice the working order is closer to five. In this method, a pair of expressions are developed to determine y as follows:

$$\begin{aligned} y_{i+1} &= y_i + \frac{1}{90}(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6) \\ \text{and}\quad y^*_{i+1} &= y_i + \frac{1}{6}(k_1 + 4k_4 + k_6), \end{aligned} \tag{8.58}$$

where

$$\begin{aligned} k_1 &= hf(x_i, y_i) \\ k_2 &= hf(x_i + h/4, y_i + k_1/4) \\ k_3 &= hf(x_i + h/4, y_i + k_1/8 + k_2/8) \\ k_4 &= hf(x_i + h/2, y_i - k_1/2 + k_3) \\ k_5 &= hf(x_i + 3h/4, y_i + 3k_1/16 + 9k_4/16) \\ k_6 &= hf(x_i + h, y_i - 3k_1/7 + 2k_2/7 + 12k_3/7 - 12k_4/7 + 8k_5/7). \end{aligned} \tag{8.59}$$

The method will terminate when $|y_{i+1} - y^*_{i+1}|$ is small enough.

The local truncation error of this method is $y_{i+1} - y^*_{i+1}$.

8.6 Predictor-Corrector Methods

The methods, viz., Taylor's series, Picard, Euler's and Runge-Kutta are **single-step methods** as these methods use only one previous values to compute the successive values, i.e., y_i is used to compute y_{i+1}. Certain efficient methods are available which need some more values to compute the successive values. These methods are called **multistep methods**. General k-step method needs $y_{i-\overline{k-1}}, y_{i-\overline{k-2}}, \ldots, y_{i-1}$ and y_i to compute y_{i+1}. The predictor-corrector method is a combination of two formulae. The first formula (called predictor) finds an approximate value of y_i and the second formula (called corrector) improves this value. The commonly used predictor-corrector multistep methods are due to Adams-Bashforth-Moulton and Milne-Simpson. These two methods are discussed below.

8.6.1 Adams-Bashforth-Moulton methods

This is a fourth-order multistep method and it needs four values (x_{i-3}, y_{i-3}), (x_{i-2}, y_{i-2}), (x_{i-1}, y_{i-1}) and (x_i, y_i) to compute y_{i+1}. These values are called starting values of this method and they may be determined by using any single step method such as Euler, Runge-Kutta, etc.

Let us consider a differential equation

$$\frac{dy}{dx} = f(x, y) \text{ with initial conditions } x = x_0, y(x_0) = y_0. \tag{8.60}$$

This differential equation is integrated between x_i and x_{i+1} and obtained the following equation

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x,y)\,dx. \tag{8.61}$$

Now, Newton's backward interpolation formula is used for y' as

$$y' = y'_i + v\nabla y'_i + \frac{v(v+1)}{2!}\nabla^2 y'_i + \frac{v(v+1)(v+2)}{3!}\nabla^3 y'_i$$

where $v = \dfrac{x - x_i}{h}$. After simplification, it reduces to

$$y' = y'_i + v\nabla y'_i + \frac{v^2+v}{2}\nabla^2 y'_i + \frac{v^3+3v^2+2v}{6}\nabla^3 y'_i.$$

Since $y' = f(x,y)$, this value is substituted in (8.61) for $f(x,y)$.
Then

$$\begin{aligned}
y_{i+1} &= y_i + h\int_0^1 \left[y'_i + v\nabla y'_i + \frac{v^2+v}{2}\nabla^2 y'_i + \frac{v^3+3v^2+2v}{6}\nabla^3 y'_i\right] dv\\
&= y_i + hy'_i + \frac{1}{2}h\nabla y'_i + \frac{5}{12}h\nabla^2 y'_i + \frac{3}{8}h\nabla^3 y'_i\\
&= y_i + hf_i + \frac{1}{2}h\nabla f_i + \frac{5}{12}h\nabla^2 f_i + \frac{3}{8}\nabla^3 f_i\\
&\quad \text{where } f_j = f(x_j, y_j) \text{ for all } j = i, i-1, i-2, i-3.\\
&= y_i + \frac{h}{24}(-9f_{i-3} + 37f_{i-2} - 59f_{i-1} + 55f_i).
\end{aligned} \tag{8.62}$$

This formula is known as **Adams-Bashforth** predictor formula and it is denoted by y^p_{i+1}, i.e.,

$$y^p_{i+1} = y_i + \frac{h}{24}(-9f_{i-3} + 37f_{i-2} - 59f_{i-1} + 55f_i). \tag{8.63}$$

The corrector formula can also be developed in the same way. In corrector formula, the value of y^p_{i+1} is used. Again, Newton's backward formula is employed on the points $(x_{i-2}, y_{i-2}), (x_{i-1}, y_{i-1}), (x_i, y_i)$ and (x_{i+1}, y^p_{i+1}). The polynomial is

$$y' = y'_{i+1} + v\nabla y'_{i+1} + \frac{v(v+1)}{2!}\nabla^2 y'_{i+1} + \frac{v(v+1)(v+2)}{3!}\nabla^3 y'_{i+1}, \text{ where } v = \frac{x - x_{i+1}}{h}.$$

This value is substituted in (8.61). Therefore,

$$y_{i+1} = y_i + h\int_{-1}^{0}\left[f_{i+1} + v\nabla f_{i+1} + \frac{v^2+v}{2}\nabla^2 f_{i+1} + \frac{v^3+3v^2+2v}{6}\nabla^3 f_{i+1}\right]dv$$

[since $dx = hdv$ and $y_i' = f_i = f(x_i, y_i)$]

$$= y_i + h\left[f_{i+1} - \frac{1}{2}\nabla f_{i+1} - \frac{1}{12}\nabla^2 f_{i+1} - \frac{1}{24}\nabla^3 f_{i+1}\right]$$

$$= y_i + \frac{h}{24}[f_{i-2} - 5f_{i-1} + 19f_i + 9f_{i+1}]. \tag{8.64}$$

This formula is known as **Adams-Moulton** corrector formula and y_{i+1} is denoted by y^c_{i+1}. Thus

$$y^c_{i+1} = y_i + \frac{h}{24}[f_{i-2} - 5f_{i-1} + 19f_i + 9f_{i+1}], \tag{8.65}$$

where $f_{i+1} = f(x_{i+1}, y^p_{i+1})$.

The value of y^p_{i+1} is computed using (8.63). The formula (8.65) can be used repeatedly to obtain the value of y_{i+1} to the desired accuracy.

Example 8.6.1 Find the value of $y(0.20)$ and $y(0.25)$ from the differential equation $y' = 2y - y^2$ with $y(0) = 1$ taking step size $h = 0.05$ using Adams-Bashforth-Moulton predictor-corrector method.

Solution. The Runge-Kutta method is used to find the starting values at $x = 0.05$, 0.10, 0.15. Here $f(x, y) = 2y - y^2, h = 0.05$. The values are shown below:

i	x_i	y_i	k_1	k_2	k_3	k_4	y_{i+1}
0	0.00	1.000000	0.050000	0.049969	0.049969	0.049875	1.049959
1	0.05	1.049959	0.049875	0.049720	0.049720	0.049503	1.099669
2	0.10	1.099669	0.049503	0.049226	0.049228	0.048891	1.148886

Thus the starting values are
$y_0 = y(0) = 1, y_1 = y(0.05) = 1.049959, y_2 = y(0.10) = 1.099669,$
$y_3 = y(0.15) = 1.148886.$
Now,

$$\begin{aligned}y^p(0.20) &= y^p_4 = y_3 + \tfrac{h}{24}[-9f(x_0,y_0) + 37f(x_1,y_1) - 59f(x_2,y_2) + 55f(x_3,y_3)]\\ &= 1.148886 + \tfrac{0.05}{24}[-9\times 1 + 37\times 0.997504 - 59\times 0.990066\\ &\quad +55\times 0.977833]\\ &= 1.197375.\end{aligned}$$

$$\begin{aligned}y^c(0.20) &= y^c_4 = y_3 + \tfrac{h}{24}[f(x_1,y_1) - 5f(x_2,y_2) + 19f(x_3,y_3) + 9f(x_4,y^p_4)]\\ &= 1.148886 + \tfrac{0.05}{24}[0.997504 - 5\times 0.990066 + 19\times 0.977833\\ &\quad +9\times 0.961043]\\ &= 1.197376.\end{aligned}$$

Thus, $y_4 = y(0.20) = 1.197376$.

$y^p(0.25) = y_5^p = y_4 + \frac{h}{24}[-9f(x_1,y_1) + 37f(x_2,y_2) - 59f(x_3,y_3) + 55f(x_4,y_4)]$
$= 1.197376 + \frac{0.05}{24}[-9 \times 0.997504 + 37 \times 0.990066 - 59 \times 0.977833$
$+55 \times 0.961043]$
$= 1.244918.$

$y^c(0.25) = y_5^c = y_4 + \frac{h}{24}[f(x_2,y_2) - 5f(x_3,y_3) + 19f(x_4,y_4) + 9f(x_5,y_5^p)]$
$= 1.197376 + \frac{0.05}{24}[0.990066 - 5 \times 0.977833 + 19 \times 0.961043$
$+9 \times 0.940015]$
$= 1.244919.$

Hence $y(0.25) = 1.244919$.

It may be noted that the predicted and corrected values are equal up to five decimal places.

Error

The local truncation error for predictor formula (8.63) is

$$h\int_0^1 \frac{v(v+1)(v+2)(v+3)}{4!}\nabla^4 f_i \simeq \frac{251}{720}y^v(c_{i+1})h^5 \quad (= Y_{i+1} - y_{i+1}^p)$$

and that of corrector formula (8.65) is

$$h\int_{-1}^0 \frac{v(v+1)(v+2)(v+3)}{4!}\nabla^4 f_{i+1} \simeq -\frac{19}{720}y^v(d_{i+1})h^5 \quad (= Y_{i+1} - y_{i+1}^c),$$

where Y_{i+1} is the true value of y at x_{i+1}.

When h is small then $y^v(x)$ is almost constant over the interval. Then the local truncation error for Adams-Bashforth-Moulton method is

$$Y_{i+1} - y_{i+1}^c \simeq -\frac{19}{270}(y_{i+1}^c - y_{i+1}^p). \tag{8.66}$$

Algorithm 8.5 (Adams-Bashforth-Moulton method). This algorithm is used to solve the differential equation $y' = f(x,y)$ with $y(x_0) = y_0$ at $x = x_1, x_2, \ldots, x_n$ with step length h. The predictor and corrector formulae are respectively

$$y_{i+1}^p = y_i + \frac{h}{24}[-9f_{i-3} + 37f_{i-2} - 59f_{i-1} + 55f_i]$$

$$\text{and} \qquad y_{i+1}^c = y_i + \frac{h}{24}[f_{i-2} - 5f_{i-1} + 19f_i + 9f_{i+1}],$$

where $f_i = f(x_i, y_i); j = i-3, i-2, i-1, i$ and $f_{i+1} = f(x_{i+1}, y_{i+1}^p)$.

Algorithm Adams-Bashforth-Moulton

```
Input function $f(x, y)$;
Read $x_0, x_n, h, y_0, y_1, y_2, y_3, \varepsilon$;
    //$x_0, x_n$ are the initial and final values of $x$; $h$ is step length;
    $y_0, y_1, y_2, y_3$ are the starting values of $y$ obtained from
    any single-step method; $\varepsilon$ is the error tolerance.//
Compute the following
$x_1 = x_0 + h$;        $x_2 = x_1 + h$;     $x_3 = x_2 + h$;
$f_0 = f(x_0, y_0)$;     $f_1 = f(x_1, y_1)$;
$f_2 = f(x_2, y_2)$;     $f_3 = f(x_3, y_3)$;
for $x_4 = x_3 + h$ to $x_n$ step $h$ do
    Compute $y^p = y_3 + \frac{h}{24}[-9f_0 + 37f_1 - 59f_2 + 55f_3]$;
    Set $y_{\text{old}} = y^p$;
    Compute $y^c = y_3 + \frac{h}{24}[f_1 - 5f_2 + 19f_3 + 9f(x_4, y_{\text{old}})]$;
    if ($|y^c - y_{\text{old}}| > \varepsilon$) then
        Set $y_{\text{old}} = y^c$;
        Calculate $y^c$ from above relation;
    else
        Reset $y_0 = y_1$;     $y_1 = y_2$;
        Reset $y_2 = y_3$;     $y_3 = y^c$;
        Reset $f_1 = f_2$;     $f_2 = f_3$;
        Reset $f_3 = f(x_4, y^c)$;
        Print $x_4, y^c$;
    endif;
endfor;
end Adams-Bashforth-Moulton
```

Program 8.5

```
/* Program Adams-Bashforth-Moutlon
   Solution of a differential equation of the form y'=f(x,y),
   y(x0)=y0 by Adams-Bashforth-Moutlon method.
   Here the equation is taken as y'=x*x*y+y*y with y(0)=1. */
#include<stdio.h>
#include<math.h>
void main()
{
 float x0,y0,xn,h,y1,y2,y3,yc,yp;
   /* x0, xn are the initial and final values of x */
   /* y0,y1,y2,y3 are starting values of y,
      h is the step length */
 float eps=1e-5; /* the error tolerance */
 float x1,x2,x3,x4,f0,f1,f2,f3,yold;
 float f(float x, float y);
 float rk4(float x,float y,float h);
 printf("Enter the initial values of x and y ");
 scanf("%f %f",&x0,&y0);
 printf("Enter last value of x ");
 scanf("%f",&xn);
 printf("Enter step length h ");
 scanf("%f",&h);
 printf(" x-value    y-value\n");
/* initial values of y are computed using Runge-Kutta method */
 x1=x0+h;     x2=x1+h;     x3=x2+h;
 y1=rk4(x0,y0,h);
 y2=rk4(x1,y1,h);
 y3=rk4(x2,y2,h);
 f0=f(x0,y0); f1=f(x1,y1); f2=f(x2,y2); f3=f(x3,y3);
 for(x4=x3+h;x4<=xn;x4+=h)
    {
      yp=y3+h*(-9*f0+37*f1-59*f2+55*f3)/24;
      yold=yp;
      yc=yp;

      do
      {
        yold=yc;
        yc=y3+h*(f1-5*f2+19*f3+9*f(x4,yold))/24;
      }while((yc-yold)>eps);

      printf("%8.5f  %8.5f\n",x4,yc);
      y0=y1;  y1=y2;  y2=y3;  y3=yc;
      f1=f2;  f2=f3;  f3=f(x4,yc);
    }
} /* main */
/* definition of the function f(x,y) */
 float f(float x, float y)
  {
   return(x*x*y+y*y);
  }
/* the fourth order Runge-Kutta method */
```

```
  float rk4(float x,float y,float h)
   {
    float k1,k2,k3,k4;
    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+k3)+k4)/6;
    return (y);
   }
```

A sample of input/output:

```
Enter the initial values of x and y 0 1
Enter last value of x 0.4
Enter step length h  0.05
 x-value    y-value
 0.20000   1.25355
 0.25000   1.34099
 0.30000   1.44328
 0.35000   1.56458
```

8.6.2 Milne's method

Another popular predictor-corrector formula is **Milne's method** which is also known as **Milne-Simpson method**. The differential equation $y' = f(x, y)$ with $y(x_0) = y_0$ is integrated between x_{i-3} and x_{i+1} and find

$$y_{i+1} = y_{i-3} + \int_{x_{i-3}}^{x_{i+1}} f(x, y)\, dx. \tag{8.67}$$

Now, the function $f(x, y)$ is replaced by Newton's forward difference formula in the form

$$f(x, y) = f_{i-3} + u\Delta f_{i-3} + \frac{u(u-1)}{2!}\Delta^2 f_{i-3} + \frac{u(u-1)(u-2)}{3!}\Delta^3 f_{i-3}, \tag{8.68}$$

where $u = \dfrac{x - x_{i-3}}{h}$.

The value of $f(x, y)$ is substituted from (8.68) to (8.67) and find

$$\begin{aligned} y_{i+1} &= y_{i-3} + h\int_0^4 \left[f_{i-3} + u\Delta f_{i-3} + \frac{u^2-u}{2}\Delta^2 f_{i-3} + \frac{u^3-3u^2+2u}{6}\Delta^3 f_{i-3}\right] du \\ &= y_{i-3} + h\Big[4f_{i-3} + 8\Delta f_{i-3} + \frac{20}{3}\Delta^2 f_{i-3} + \frac{8}{3}\Delta^3 f_{i-3}\Big] \\ &= y_{i-3} + \frac{4h}{3}[2f_{i-2} - f_{i-1} + 2f_i]. \end{aligned}$$

Thus the Milne's predictor formula is

$$y_{i+1}^p = y_{i-3} + \frac{4h}{3}[2f_{i-2} - f_{i-1} + 2f_i]. \tag{8.69}$$

The corrector formula is developed in a similar way. The value of y_{i+1}^p will now be used. Again, the given differential equation is integrated between x_{i-1} and x_{i+1} and the function $f(x, y)$ is replaced by the Newton's formula (8.68). Then

$$\begin{aligned}
y_{i+1} &= y_{i-1} + \int_{x_{i-1}}^{x_{i+1}} \left[f_{i-1} + u\Delta f_{i-1} + \frac{u(u-1)}{2}\Delta^2 f_{i-1}\right] dx \\
&= y_{i-1} + h\int_0^2 \left[f_{i-1} + u\Delta f_{i-1} + \frac{u^2-u}{2}\Delta^2 f_{i-1}\right] du \\
&= y_{i-1} + h\left[2f_{i-1} + 2\Delta f_{i-1} + \frac{1}{3}\Delta^2 f_{i-1}\right] \\
&= y_{i-1} + \frac{h}{3}[f_{i-1} + 4f_i + f_{i+1}].
\end{aligned}$$

This formula is known as corrector formula and it is denoted by y_{i+1}^c. That is,

$$y_{i+1}^c = y_{i-1} + \frac{h}{3}[f(x_{i-1}, y_{i-1}) + 4f(x_i, y_i) + f(x_{i+1}, y_{i+1}^p)]. \tag{8.70}$$

When y_{i+1}^p is computed using the formula (8.69), formula (8.70) can be used iteratively to obtain the value of y_{i+1} to the desired accuracy.

Example 8.6.2 Find the value of $y(0.20)$ for the initial value problem

$$\frac{dy}{dx} = y^2 \sin x \text{ with } y(0) = 1$$

using Milne's predictor-corrector method, taking $h = 0.05$.

Solution. Let $f(x, y) = y^2 \sin x$, $x_0 = 0, y_0 = 1, h = 0.05$.
Fourth-order Runge-Kutta method is used to compute the starting values y_1, y_2 and y_3.

i	x_i	y_i	k_1	k_2	k_3	k_4	y_{i+1}
0	0.00	1.000000	0.000000	0.001250	0.001251	0.002505	1.001251
1	0.05	1.001251	0.002505	0.003765	0.003770	0.005042	1.005021
2	0.10	1.005021	0.005042	0.006328	0.006336	0.007643	1.011356

The predictor value

$$\begin{aligned}
y_4^p &= y_0 + \frac{4h}{3}[2f(x_1, y_1) - f(x_2, y_2) + 2f(x_3, y_3)] \\
&= 1 + \frac{4 \times 0.05}{3}[2f(0.05, 1.001251) - f(0.10, 1.005021) + 2f(0.15, 1.011356)] \\
&= 1 + \frac{4 \times 0.05}{3}[2 \times 0.0501043 - 0.1008385 + 2 \times 0.1528516] \\
&= 1.0203380
\end{aligned}$$

The corrector value is

$$\begin{aligned}
y_4^c &= y_2 + \frac{h}{2}[f(x_2, y_2) + 4f(x_3, y_3) + f(x_4, y_4^p)] \\
&= 1,005021 + \frac{0.05}{2}[0.1008385 + 4 \times 0.1528516 + 0.2068326] \\
&= 1.0203390.
\end{aligned}$$

Again, the corrector value y_4^c is calculated by using the formula

$$\begin{aligned}
y_4^c &= y_2 + \frac{h}{2}[f(x_2, y_2) + 4f(x_3, y_3) + f(x_4, y_4^c)] \\
&= 1.005021 + \frac{0.05}{2}[0.1008385 + 4 \times 0.1528516 + 0.2068392] \\
&= 1.0203390.
\end{aligned}$$

Since these two values are same, the required solution is
$y_4 = y(0.20) = 1.0203390$ correct up to seven decimal places.

Error

The local truncation error for prediction formula is

$$\frac{28}{90} y^v(c_{i+1})h^5 = O(h^5)$$

and that of the corrector formula is $-\frac{1}{90} y^v(d_{i+1})h^5 = O(h^5)$.

Note 8.6.1 Milne's predictor-corrector method is widely used formula. In this method there is a scope to improve the value of y by repeated use of corrector formula. So it gives more accurate result. But, this method needs the starting values y_1, y_2, y_3 to obtain y_4. These values may be obtained from any single-step method, such as Taylor's series, Euler's, Runge-Kutta or any similar method.

Algorithm 8.6 (Milne's Predictor-Corrector method). This algorithm is used to solve the initial value problem $y' = f(x, y)$ with $y(x_0) = y_0$ at the points $x_4, x_5, \ldots, x_n$ by Milne's method. The starting values y_1, y_2, y_3 may be obtained from any self starting method, viz., Taylor's series, Euler's or Runge-Kutta methods.

```
Algorithm Milne
Input function $f(x, y)$;
Read $x_0, x_n, h, y_0, y_1, y_2, y_3, \varepsilon$;
    //$x_0, x_n$ are the initial and final values of $x$; $h$ is step length;
    $y_0, y_1, y_2, y_3$ are the starting values of $y$ obtained from
    any single-step method; $\varepsilon$ is the error tolerance.//
Compute the following
$x_1 = x_0 + h$;      $x_2 = x_1 + h$;      $x_3 = x_2 + h$;
$f_1 = f(x_1, y_1)$;
$f_2 = f(x_2, y_2)$;      $f_3 = f(x_3, y_3)$;
for $x_4 = x_3 + h$ to $x_n$ step $h$ do
    Compute $y^p = y_0 + \frac{4h}{3}[2f_1 - f_2 + 2f_3]$;
    Set $y_{\text{old}} = y^p$;
    Compute $y^c = y_2 + \frac{h}{3}[f_2 + 4f_3 + f(x_4, y_{\text{old}})]$;
    if ($|y^c - y_{\text{old}}| > \varepsilon$) then
        Reset $y_{\text{old}} = y^c$;
        Calculate $y^c$ from above relation;
    else
        Reset $y_0 = y_1$;      $y_1 = y_2$;
        Reset $y_2 = y_3$;      $y_3 = y^c$;
        Reset $f_1 = f_2$;      $f_2 = f_3$;
        Compute $f_3 = f(x_4, y^c)$;
        Print $x_4, y^c$;
    endif;
endfor;
end Milne
```

Program 8.6

```
/* Program Milne Predictor-Corrector
   Solution of a differential equation of the form y'=f(x,y),
   y(x0)=y0 by Milne Predictor-Corrector method.
   Here the equation is taken as y'=x*y+y*y with y(0)=1.
*/
#include<stdio.h>
#include<math.h>
```

```
void main()
{
 float x0,y0,xn,h,y1,y2,y3,yc,yp;
   /* x0, xn the intial and final value of x */
   /* y0,y1,y2,y3 are starting values of y,
      h is the step length */
 float eps=1e-5; /* the error tolerance */
 float x1,x2,x3,x4,f0,f1,f2,f3,yold;
 float f(float x, float y);
 float rk4(float x,float y,float h);
 printf("Enter the initial values of x and y ");
 scanf("%f %f",&x0,&y0);
 printf("Enter last value of x ");
 scanf("%f",&xn);
 printf("Enter step length h ");
 scanf("%f",&h);
 printf(" x-value   y-value\n");

/* initial values of y are computed using Runge-Kutta method */
 x1=x0+h;     x2=x1+h;     x3=x2+h;
 y1=rk4(x0,y0,h);
 y2=rk4(x1,y1,h);
 y3=rk4(x2,y2,h);
 f1=f(x1,y1); f2=f(x2,y2); f3=f(x3,y3);
 for(x4=x3+h;x4<=xn;x4+=h)
    {
      yp=y0+4*h*(2*f1-f2+2*f3)/3;
      yold=yp;
      yc=yp;
      do
      {
        yold=yc;
        yc=y2+h*(f2+4*f3+f(x4,yold))/3;
      }while((yc-yold)>eps);
      printf("%8.5f  %8.5f\n",x4,yc);
      y0=y1;  y1=y2;  y2=y3;  y3=yc;
      f1=f2;  f2=f3;  f3=f(x4,yc);
    }
} /* main */
```

```
/* definition of the function f(x,y) */
 float f(float x, float y)
  {
    return(x*y+y*y);
  }
/* the fourth order Runge-Kutta method */
float rk4(float x,float y,float h)
 {
   float k1,k2,k3,k4;
   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+k3)+k4)/6;
   return (y);
 }
```

A sample of input/output:

```
Enter the initial values of x and y 0 1
Enter last value of x 0.4
Enter step length h  0.5
 x-value   y-value
 0.20000   1.27740
 0.25000   1.38050
 0.30000   1.50414
 0.35000   1.65418
```

8.7 Finite Difference Method

In this method, the derivatives y' and y'' are replaced by finite differences (either by forward or central) and generates a system of linear algebraic equations. The answer of this system is the solution of the differential equation at different mesh.

The central difference formula discussed in Chapter 7 are used to replace derivatives

$$\begin{aligned} y'(x_i) &= \frac{y_{i+1} - y_{i-1}}{2h} + O(h^2) \\ \text{and}\quad y''(x_i) &= \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + O(h^2). \end{aligned} \tag{8.71}$$

The method to solve first order differential equation using finite difference method is nothing but the Euler's method. The finite difference method is commonly used method to solve second order initial value problem and boundary value problem.

8.7.1 Second order initial value problem (IVP)

Let us consider the second order linear IVP of the form

$$y'' + p(x)y' + q(x)y = r(x) \tag{8.72}$$

with the initial conditions

$$x = x_0, \qquad y(x_0) = y_0, \qquad y'(x_0) = y'_0. \tag{8.73}$$

The values of $y'(x_i)$ and $y''(x_i)$ are substituted from (8.71) to (8.72) and (8.73), to find the equation

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p(x_i)\frac{y_{i+1} - y_{i-1}}{2h} + q(x_i)y_i = r(x_i).$$

After simplification, the above equation reduces to

$$[2 - hp(x_i)]y_{i-1} + [2h^2q(x_i) - 4]y_i + [2 + hp(x_i)]y_{i+1} = 2h^2r(x_i). \tag{8.74}$$

Let us consider

$$\begin{aligned} C_i &= 2 - hp(x_i) \\ A_i &= 2h^2q(x_i) - 4 \\ B_i &= 2 + hp(x_i) \text{ and} \\ D_i &= 2h^2r(x_i). \end{aligned} \tag{8.75}$$

Then equation (8.74) is simplified to

$$C_iy_{i-1} + A_iy_i + B_iy_{i+1} = D_i. \tag{8.76}$$

The initial condition $y'(x_0) = y'_0$ reduces to

$$y'_0 = \frac{y_1 - y_{-1}}{2h} \qquad \text{or,} \qquad y_{-1} = y_1 - 2hy'_0. \tag{8.77}$$

Again, from (8.76),

$$C_0y_{-1} + A_0y_0 + B_0y_1 = D_0. \tag{8.78}$$

The quantity y_{-1} is eliminated between (8.77) and (8.78), and the value of y_1 is obtained as

$$y_1 = \frac{D_0 - A_0y_0 + 2hC_0y'_0}{C_0 + B_0}. \tag{8.79}$$

Since p, q, r are known functions, A, B, C, D can easily be determined at $x = x_0, x_1, \ldots$. That is, right hand side of (8.79) is a known quantity. Thus (8.79) gives the value of y_1. Also, y_0 is known. Hence the value of y_{i+1} can be obtained from (8.76) as

$$y_{i+1} = \frac{D_i - C_i y_{i-1} - A_i y_i}{B_i}, \qquad x_{i+1} = x_i + h \tag{8.80}$$

for $i = 1, 2, \ldots$.

Thus, the values of $y_1, y_2, \ldots$ are determined recursively from (8.80).

Example 8.7.1 Solve the following IVP $y'' - y = x$ with $y(0) = 0$ and $y'(0) = 1$ using finite difference method for $x = 0.01, 0.02, \ldots, 0.10$.

Solution. The second order derivative y'' is replaced by $\dfrac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$ in the given differential equation and obtained the following system of equations.

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - y_i = x_i \qquad \text{or} \qquad y_{i+1} - (2 + h^2)y_i + y_{i-1} = h^2 x_i$$

and

$$y'_0 = \frac{y_1 - y_{-1}}{2h} \qquad \text{or} \qquad y_{-1} = y_1 - 2hy'_0.$$

Again from above equation
$y_1 - (2 + h^2)y_0 + y_{-1} = h^2 x_0$ or, $y_1 = (2 + h^2)y_0 - (y_1 - 2hy'_0) + h^2 x_0$.
That is,

$$\begin{aligned} y_1 &= \frac{(2 + h^2)y_0 + 2hy'_0 + h^2 x_0}{2} \\ &= \frac{[2 + (0.01)^2] \times 0 + 2 \times 0.01 \times 1.0 + (0.01)^2 \times 0}{2} = 0.01. \end{aligned}$$

The values of $y_i, i = 2, 3, \ldots$ are obtained from the relation
$y_{i+1} = (2 + h^2)y_i - y_{i-1} + h^2 x_i$.

i	y_{i-1}	x_i	y_i	y_{i+1}
1	0.000000	0.01	0.010000	0.020002
2	0.010000	0.02	0.020002	0.030008
3	0.020002	0.03	0.030008	0.040020
4	0.030008	0.04	0.040020	0.050040
5	0.040020	0.05	0.050040	0.060070
6	0.050040	0.06	0.600070	0.070112
7	0.060070	0.07	0.070112	0.080168
8	0.070112	0.08	0.080168	0.090240
9	0.080168	0.09	0.090240	0.100330
10	0.090240	0.10	0.100330	0.110440

Error

The local truncation error is

$$E_{\text{LTE}} = \left(\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} - y_i''\right) + p_i\left(\frac{y_{i+1} - y_{i-1}}{2h} - y_i'\right).$$

Expanding the terms y_{i-1} and y_{i+1} by Taylor's series and simplifying, the above expression gives

$$E_{\text{LTE}} = \frac{h^2}{12}(y^{iv} + 2p_i y_i''') + O(h^4).$$

Thus, the finite difference approximation has second-order accuracy for the functions with continuous fourth derivatives.

8.7.2 Second order boundary value problem (BVP)

Let us consider the linear second order differential equation

$$y'' + p(x)y' + q(x)y = r(x), \qquad a < x < b \tag{8.81}$$

with boundary conditions $y(a) = \gamma_1$ and $y(b) = \gamma_2$.

Let the interval $[a, b]$ be divided into n subintervals with spacing h. That is, $x_i = x_{i-1} + h, i = 1, 2, \ldots, n-1$.

The equation (8.81) is satisfied by $x = x_i$. Then

$$y_i'' + p(x_i)y_i' + q(x_i)y_i = r(x_i). \tag{8.82}$$

Now, y_i'' and y_i' are replaced by finite difference expressions

$$y_i'' = \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} + O(h^2), \qquad y_i' = \frac{y_{i+1} - y_{i-1}}{2h} + O(h^2).$$

Using these substitutions and drooping $O(h^2)$, equation (8.82) becomes

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} + p(x_i)\frac{y_{i+1} - y_{i-1}}{2h} + q(x_i)y_i = r(x_i).$$

That is,

$$y_{i-1}[2 - hp(x_i)] + y_i[2h^2q(x_i) - 4] + y_{i+1}[2 + hp(x_i)] = 2h^2r(x_i). \tag{8.83}$$

Let us consider

$$\begin{aligned} C_i &= 2 - hp(x_i) \\ A_i &= 2h^2q(x_i) - 4 \\ B_i &= 2 + hp(x_i) \\ \text{and } D_i &= 2h^2r(x_i). \end{aligned} \tag{8.84}$$

With these notations, the equation (8.83) is simplified to

$$C_i y_{i-1} + A_i y_i + B_i y_{i+1} = D_i, \tag{8.85}$$

for $i = 1, 2, \ldots, n-1$.

The boundary conditions then are $y_0 = \gamma_1, y_n = \gamma_2$.

For $i = 1, n-1$ equation (8.85) reduces to
$C_1 y_0 + A_1 y_1 + B_1 y_2 = D_1$ or, $A_1 y_1 + B_1 y_2 = D_1 - C_1\gamma_1$ (as $y_0 = \gamma_1$)
and $C_{n-1} y_{n-2} + A_{n-1} y_{n-1} + B_{n-1} y_n = D_{n-1}$
or, $C_{n-1} y_{n-2} + A_{n-1} y_{n-1} = D_{n-1} - B_{n-1}\gamma_2$ (as $y_n = \gamma_2$).

The equation (8.85) in matrix notation is

$$\mathbf{Ay} = \mathbf{b} \tag{8.86}$$

where

$$\mathbf{y} = [y_1, y_2, \ldots, y_{n-1}]^t$$
$$\mathbf{b} = 2h^2[r(x_1) - \{C_1\gamma_1\}/(2h^2), r(x_2), \ldots, r(x_{n-2}), r(x_{n-1}) - \{B_{n-1}\gamma_2\}/(2h^2)]^t$$

and $$\mathbf{A} = \begin{bmatrix} A_1 & B_1 & 0 & 0 & \cdots & 0 & 0 \\ C_2 & A_2 & B_2 & 0 & \cdots & 0 & 0 \\ 0 & C_3 & A_3 & B_3 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & C_{n-1} & A_{n-1} \end{bmatrix}.$$

Equation (8.86) is a tri-diagonal system which can be solved by the method discussed in Chapter 5. The solution of this system i.e., the values of $y_1, y_2, \ldots, y_{n-1}$ constitutes the approximate solution of the BVP.

Example 8.7.2 Solve the following boundary value problem $y'' + xy' + 1 = 0$ with boundary conditions $y(0) = 0, y(1) = 0$.

Solution. Here $nh = 1$. The difference scheme is

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} + x_i \frac{y_{i+1} - y_{i-1}}{2h} + 1 = 0.$$

That is,

$$y_{i-1}(2 - x_i h) - 4y_i + y_{i+1}(2 + x_i h) + 2h^2 = 0, i = 1, 2, \ldots, n-1 \tag{8.87}$$

together with boundary condition $y_0 = 0, y_n = 0$.
Let $n = 2$. Then $h = 1/2$, $x_0 = 0, x_1 = 1/2, x_2 = 1, y_0 = 0, y_2 = 0$.
The difference scheme is
$y_0(2 - x_1 h) - 4y_1 + y_2(2 + x_1 h) + 2h^2 = 0$ or, $-4y_1 + 2(1/4) = 0$ or, $y_1 = 0.125$.

That is, $y(0.5) = 0.125$.

Let $n = 4$. Then $h = 0.25$, $x_0 = 0, x_1 = 0.25, x_2 = 0.50, x_3 = 0.75$, $x_4 = 1.0, y_0 = 0, y_4 = 0$.

The system of equations (8.87) becomes

$$y_0(2 - x_1h) - 4y_1 + y_2(2 + x_1h) + 2h^2 = 0$$
$$y_1(2 - x_2h) - 4y_2 + y_3(2 + x_2h) + 2h^2 = 0$$
$$y_2(2 - x_3h) - 4y_3 + y_4(2 + x_3h) + 2h^2 = 0.$$

This system is finally simplified to

$$-4y_1 + 2.06250y_2 + 0.125 = 0$$
$$1.875y_1 - 4y_2 + 2.125y_3 + 0.125 = 0$$
$$1.8125y_2 - 4y_3 + 0.125 = 0.$$

The solution of this system is

$y_1 = y(0.25) = 0.09351$, $y_2 = y(0.50) = 0.12075$, $y_3 = y(0.75) = 0.08597$.

This is also the solution of the given differential equation.

Algorithm 8.7 (Finite Difference Method for BVP). Using this algorithm, the BVP $y'' + p(x)y' + q(x)y = r(x)$ with $y(x_0) = \gamma_1, y(x_n) = \gamma_2$ is solved by finite difference method.

```
Algorithm BVP_FD
Input functions $q(x), q(x), r(x)$;
//The functions $p, q, r$ to be changed accordingly.//
Read $x_0, x_n, h, \gamma_1, \gamma_2$; //The boundary values of $x$; step size $h$; and
                                    the boundary value of $y$.//
for $i = 1, 2, \ldots, n-1$ do
    Compute $A_i = 2h^2q(x_i) - 4$;
    Compute $C_i = 2 - hp(x_i)$;
    Compute $B_i = 2 + hp(x_i)$;
    Compute $D_i = 2h^2r(x_i)$;
    Reset $D_1 = D_1 - C_1\gamma_1$;
    Reset $D_{n-1} = D_{n-1} - B_{n-1}\gamma_2$;
Solve the tri-diagonal system of equations $\mathbf{Ay} = \mathbf{b}$,
where $\mathbf{A}, \mathbf{b}, \mathbf{y}$ are given by the equation (8.86);
Print $y_1, y_2, \ldots, y_{n-1}$ as solution;
end BVP_FD
```

Program 8.7

```
/* Program BVP Finite Difference
   This program solves the second order boundary value
   y''+p(x)y'+q(x)y=r(x) with y(x0)=y0, y(xn)=yn by finite difference
   method. Here we consider the equation y''+2y'+y=10x with
   boundary conditions y(0)=0 and y(1)=0. */
```

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
float y[10];
void main()
 {
   int i,n;
   float a[10],b[10],c[10],d[10],x0,xn,y0,yn,temp,h,x;
   float p(float x); float q(float x); float r(float x);
   float TriDiag(float a[],float b[],float c[],float d[],int n);
   printf("Enter the initial and final values of x ");
   scanf("%f %f",&x0,&xn);
   printf("Enter the initial and final values of y ");
   scanf("%f %f",&y0,&yn);
   printf("Enter number of subintervals ");
   scanf("%d",&n);
   h=(xn-x0)/n;
   x=x0;
   for(i=1;i<=n-1;i++)
      {
        x+=h;
        a[i]=2*h*h*q(x)-4;
        b[i]=2+h*p(x);
        c[i]=2-h*p(x);
        d[i]=2*h*h*r(x);
      }  /* end of loop i */

   d[1]-=c[1]*y0;
   d[n-1]-=b[n-1]*yn;
   temp=TriDiag(c,a,b,d,n-1);
   y[0]=y0; y[n]=yn;
   printf("The solution is\n x-value    y-value\n");
   for(i=0;i<=n;i++) printf("%8.5f   %8.5f \n",x0+i*h,y[i]);
} /* main */

/* definition of the functions p(x), q(x) and r(x) */
float p(float x)
 {
   return(2);
 }
```

```
float q(float x)
 {
   return(1);
 }
float r(float x)
 {
   return(10*x);
 }
float TriDiag(float a[10],float b[10],float c[10],float d[10],int n)
 {
 /* output y[i], i=1, 2,..., n, is a global variable.*/
 int i; float gamma[10],z[10];
 gamma[1]=b[1];
 for(i=2;i<=n;i++){
     if(gamma[i-1]==0.0){
           printf("A minor is zero: Method fails ");
           exit(0);
        }
     gamma[i]=b[i]-a[i]*c[i-1]/gamma[i-1];
    }
 z[1]=d[1]/gamma[1];
 for(i=2;i<=n;i++)
     z[i]=(d[i]-a[i]*z[i-1])/gamma[i];
 y[n]=z[n];
 for(i=n-1;i>=1;i--)
     y[i]=z[i]-c[i]*y[i+1]/gamma[i];
 return(y[0]);
} /*end of TriDiag */
```

A sample of input/output:

```
Enter the initial and final values of x 0 1
Enter the initial and final values of y 0 0
Enter number of subintervals 4
The solution is
 x-value    y-value
 0.00000    0.00000
 0.25000   -0.52998
 0.50000   -0.69647
 0.75000   -0.51154
 1.00000    0.00000
```

Error

The local truncation error of this method is similar to the IVP, i.e., this method is also of second-order accuracy for functions with continuous fourth order derivatives on $[a, b]$. It may be noted that when $h \to 0$ then the local truncation error tends to zero, i.e., greater accuracy in the result can be achieved by using small h. But, small h produces a large number of equations and take more computational effort.

The accuracy can be improved to employ Richardson's deferred approach to the limit.

The error of this method has the form

$$y(x_i) - y_i = h^2 E(x_i) + O(h^4). \tag{8.88}$$

For extrapolation to the limit, we find the value of (8.88) at two intervals h and $h/2$. The values of y_i are denoted by $y_i(h)$ and $y_i(h/2)$. Thus for $x = x_i$ for different step sizes

$$\begin{aligned} & y(x_i) - y_i(h) = h^2 E(x_i) + O(h^4) \\ \text{and} \quad & y(x_i) - y_i(h/2) = \frac{h^2}{4} E(x_i) + O(h^4) \end{aligned} \tag{8.89}$$

Now, by eliminating $E(x_i)$ we find the expression for $y(x_i)$, in the form

$$y(x_i) = \frac{4y_i(h/2) - y_i(h)}{3}. \tag{8.90}$$

The existence and uniqueness conditions of BVP are stated below.

Theorem 8.2 *Assume that $f(x, y, y')$ is continuous on the region $R = \{(x, y, y') : a \le x \le b, -\infty < y < \infty, -\infty < y' < \infty\}$ and that f_y and $f_{y'}$ are continuous on R. If there exists a constant $M > 0$ for which f_y and $f_{y'}$ satisfy $f_y > 0$ for all $(x, y, y') \in R$ and $|f_{y'}(x, y, y')| < M$ for all $(x, y, y') \in R$, then the BVP $y'' = f(x, y, y')$ with $y(a) = \gamma_1$ and $y(b) = \gamma_2$ has a unique solution $y = y(x)$ for $a \le x \le b$.*

8.8 Shooting Method for Boundary Value Problem

This method works in three stages

(i) The given BVP is transferred into two IVPs,

(ii) Solutions of these two IVPs can be determined by Taylor's series or Runge-Kutta or any other method,

(iii) Combination of these two solutions is the required solution of the given BVP.

Reduction to two IVPs

Let the BVP be

$$y'' - p(x)y' - q(x)y = r(x) \text{ with } y(a) = \alpha \text{ and } y(b) = \beta. \tag{8.91}$$

Suppose that $u(x)$ is the unique solution to the IVP

$$u''(x) - p(x)u'(x) - q(x)u(x) = r(x) \text{ with } u(a) = \alpha \text{ and } u'(a) = 0. \tag{8.92}$$

Furthermore, suppose that $v(x)$ is the unique solution to the IVP,

$$v''(x) - p(x)v'(x) - q(x)v(x) = 0 \text{ with } v(a) = 0 \text{ and } v'(a) = 1. \tag{8.93}$$

Then the linear combination

$$y(x) = c_1u(x) + c_2v(x) \tag{8.94}$$

is the solution of (8.91) for some constants c_1, c_2.

As $y(a) = \alpha$ and $y(b) = \beta$,
$\alpha = c_1u(a) + c_2v(a)$ and $\beta = c_1u(b) + c_2v(b)$.

From the first equation, $c_1 = 1$ and from second equation, $c_2 = \dfrac{\beta - u(b)}{v(b)}$.

Hence (8.94) reduces to

$$y(x) = u(x) + \frac{\beta - u(b)}{v(b)}v(x), \tag{8.95}$$

this is a solution of the BVP (8.91) and it also satisfies the boundary conditions $y(a) = \alpha, y(b) = \beta$.

The method involves to solve two systems of equations over $[a, b]$. The first system is

$$\begin{aligned} u' &= w(x) \text{ with } u(a) = \alpha \\ \text{and } w'(x) &= p(x)w(x) + q(x)u(x) + r(x) \text{ with } w(a) = u'(a) = 0, \end{aligned} \tag{8.96}$$

and the second system is

$$\begin{aligned} v' &= z(x) \text{ with } v(a) = 0 \\ z'(x) &= p(x)z(x) + q(x)v(x) \text{ with } z(a) = v'(a) = 1. \end{aligned} \tag{8.97}$$

Finally, the desired solution $y(x)$ is obtained from (8.95).

Unfortunately, when the expression $p(x)y'(x) + q(x)y(x) + r(x)$ is non-linear then several iterations are needed to obtain the solution at b to a prescribed accuracy. One may encounter difficulty to obtain a convergent solution if $y(b)$ is a very sensitive function of $y'(a)$. In this case it may be suggested to integrate from the opposite direction by guessing a value of $y'(b)$ and iterating until $y(a)$ is sufficiently close to y_0. The speed of convergence depends on how good the initial guess was chosen.

Example 8.8.1 Find the solution of the boundary value problem $y'' = y - x$ with $y(0) = 0, y(1) = 0$ using the shooting method.

Solution. The second-order Runge-Kutta method is used to solve the initial value problem with $h = 0.25$.
Here $p(x) = 0, q(x) = 1, r(x) = -x$, $a = 0, b = 1, \alpha = 0, \beta = 0$.
Now, two IVPs are

$$\begin{aligned} u' &= w \text{ with } u(0) = 0 \\ w' &= u - x \text{ and } w(0) = u'(0) = 0 \end{aligned} \tag{8.98}$$

and

$$\begin{aligned} v' &= z \text{ with } v(0) = 0 \\ z' &= v \text{ and } z(0) = v'(0) = 1. \end{aligned} \tag{8.99}$$

Solution of the system (8.98) is shown below.

i	x_i	u_i	w_i	k_1	k_2	l_1	l_2	u_{i+1}	w_{i+1}
0	0.0	0.00000	0.00000	0.00000	0.00000	0.00000	-0.04000	0.00000	-0.02000
1	0.2	0.00000	-0.02000	-0.00400	-0.01200	-0.04000	-0.08080	-0.00800	-0.08040
2	0.4	-0.00800	-0.08040	-0.01608	-0.03240	-0.08160	-0.12482	-0.03224	-0.18361
3	0.6	-0.03224	-0.18361	-0.03672	-0.06201	-0.12645	-0.17379	-0.08161	-0.33373
4	0.8	-0.08161	-0.33373	-0.06675	-0.10201	-0.17632	-0.22967	**-0.16598**	-0.53672

Solution of the system (8.99) is shown in the following table.

i	x_i	v_i	z_i	k_1	k_2	l_1	l_2	v_{i+1}	z_{i+1}
0	0.0	0.00000	1.00000	0.20000	0.20000	0.00000	0.04000	0.20000	1.02000
1	0.2	0.20000	1.02000	0.20400	0.21200	0.04000	0.08080	0.40800	1.08040
2	0.4	0.40800	1.08040	0.21608	0.23240	0.08160	0.12482	0.63224	1.18361
3	0.6	0.63224	1.18361	0.23672	0.26201	0.12645	0.17379	0.88161	1.33373
4	0.8	0.88161	1.33373	0.26675	0.30201	0.17632	0.22967	**1.16598**	1.53672

Now,
$$c = \frac{\beta - u(b)}{v(b)} = \frac{0 - u(1)}{v(1)} = \frac{0.16598}{1.16598} = 0.142352.$$

The values of $y(x)$ given by $y(x) = u(x) + cv(x) = u(x) + 0.142352\ v(x)$ are listed below:

x	$u(x)$	$v(x)$	$y(x)$	y_{exact}
0.2	+0.00000	0.20000	0.02847	0.02868
0.4	-0.00800	0.40800	0.05008	0.05048
0.6	-0.03224	0.63224	0.05776	0.05826
0.8	-0.08161	0.88161	0.04389	0.04429
1.0	-0.16598	1.16598	0.00000	0.00000

8.9 Finite Element Method

Finite element method (FEM) is widely used technique to solve many engineering problems. Here, a very brief introduction is presented to solve a BVP by using this method. The detailed discussion of FEM is beyond the scope of this book.

The main idea of this method is – the whole interval (of integration) is divided into a finite number of subintervals called **element** and over each element the continuous function is approximated by a suitable piecewise polynomial. This approximated problem is solved by Rayleigh-Ritz or Galerkin methods.

Let us consider the functional

$$J[y(x)] = \int_a^b F(x, y(x), y'(x))\, dx \tag{8.100}$$

subject to the boundary conditions

$$y(a) = y_a, \quad y(b) = y_b. \tag{8.101}$$

We assume that F is differentiable. The curve $y = y(x)$ which extremizing J under the boundary condition (8.101) is a solution of the **Euler equation**

$$\frac{\partial F}{\partial y} - \frac{d}{dx}\left(\frac{\partial F}{\partial y'}\right) = 0. \tag{8.102}$$

The Euler's equation (8.102) has many solutions but, for a given boundary condition, it gives a unique solution.

Let us consider the boundary value problem

$$-\frac{d}{dx}\big[p(x)y'(x)\big] + q(x)y(x) = r(x) \tag{8.103}$$

with boundary condition (8.101). It can easily be verified that the variational form of (8.103) is given by

$$J[y(x)] = \frac{1}{2}\int_a^b \big[p(x)\ \{y'(x)\}^2 + q(x)\ \{y(x)\}^2 - 2r(x)\ y(x)\big]\, dx. \tag{8.104}$$

The main steps to solve a BVP using FEM are given below.

Step 1. Discretization of the interval

The interval $[a, b]$ is divided into a finite number of subintervals, called elements, of unequal length. Let $x_0, x_1, \ldots, x_n$, $a = x_0 < x_1 < \cdots < x_n = b$, be the division points, called the **nodes**. Let $e_i = [x_i, x_{i+1}]$ be the ith element of length $h_i = x_{i+1} - x_i$.

Step 2. Variational formulation of BVP over the element e_i

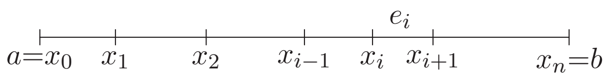


Figure 8.5: Division of interval into elements.

The variational form of (8.103) is given by

$$J[y(x)] = \frac{1}{2}\int_a^b [p(y')^2 + qy^2 - 2ry]\,dx. \tag{8.105}$$

Let J_i be the functional (called **element functional**) over the element $e_i = [x_i, x_{i+1}]$, $i = 0, 1, 2, \ldots, n-1$. Then

$$J_i = \frac{1}{2}\int_{x_i}^{x_{i+1}} \left[p\Big(\frac{dy^{(i)}}{dx}\Big)^2 + q\Big(y^{(i)}\Big)^2 - 2ry^{(i)}\right] dx, \tag{8.106}$$

where $y^{(i)}$ is the value of y over the element e_i and it is zero outside the element e_i. x_i, x_{i+1} are the end nodes of the element e_i and let $\phi^{(i)} = [y(x_i)\;\; y(x_{i+1})]^T$.

Thus the functional J over the whole interval $[a, b]$ is the sum of the n functionals J_i, that is,

$$J[y] = \sum_{i=0}^{n-1} J_i. \tag{8.107}$$

The element functional J_i will be extremum with respect to $\phi^{(\mathbf{i})}$ if

$$\frac{\partial J_i}{\partial \phi^{(i)}} = 0. \tag{8.108}$$

This gives the required finite element equation for the element e_i.

Step 3. Rayleigh-Ritz finite element approximation over e_i

The function $y(x)$ may be approximated over the element e_i by linear Lagrange's interpolation polynomial as

$$y^{(i)}(x) = L_i(x)y_i + L_{i+1}(x)y_{i+1} \tag{8.109}$$

where

$$y_i = y(x_i) \text{ and } L_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i},\; L_{i+1}(x) = \frac{x - x_i}{x_{i+1} - x_i}. \tag{8.110}$$

The function $L_i(x)$ and $L_{i+1}(x)$ are called **shape functions**.

The equation (8.109) can be written as

$$y^{(i)}(x) = \begin{bmatrix} L_i(x) & L_{i+1} \end{bmatrix} \begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix} = \mathbf{L^{(i)}}\phi^{(\mathbf{i})} \tag{8.111}$$

where $\mathbf{L^{(i)}} = \begin{bmatrix} L_i & L_{i+1} \end{bmatrix}$ and $\phi^{(\mathbf{i})} = \begin{bmatrix} y_i & y_{i+1} \end{bmatrix}^T$.

It may be noted that

$$L_i(x_j) = \begin{cases} 0, & i \neq j \\ 1, & i = j. \end{cases}$$

From (8.110), it is easy to obtain

$$\frac{\partial L_i}{\partial x} = -\frac{1}{x_{i+1} - x_i} = -\frac{1}{h_i} \text{ and } \frac{\partial L_{i+1}}{\partial x} = \frac{1}{x_{i+1} - x_i} = \frac{1}{h_i}. \tag{8.112}$$

The value of $y^{(i)}(x)$ is substituted from (8.111) to (8.106) and obtain

$$\begin{aligned} J_i &= \frac{1}{2}\int_{x_i}^{x_{i+1}} \Bigg[p(x)\left\{\begin{bmatrix} L_i' & L_{i+1}' \end{bmatrix}\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix}\right\}^2 + q(x)\left\{\begin{bmatrix} L_i & L_{i+1} \end{bmatrix}\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix}\right\}^2 \\ &\quad -r(x)\left\{\begin{bmatrix} L_i & L_{i+1} \end{bmatrix}\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix}\right\}^2\Bigg] dx, \end{aligned}$$

where prime denotes differentiation with respect to x.

To extremize J_i, differentiating it with respect to y_i and y_{i+1} as

$$\begin{aligned} \frac{\partial J_i}{\partial y_i} &= \int_{x_i}^{x_{i+1}} \Bigg[p(x)L_i'\left\{\begin{bmatrix} L_i' & L_{i+1}' \end{bmatrix}\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix}\right\} \\ &\quad +q(x)L_i\left\{\begin{bmatrix} L_i & L_{i+1} \end{bmatrix}\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix}\right\} - r(x)L_i\Bigg] dx = 0. \end{aligned} \tag{8.113}$$

$$\begin{aligned} \text{and } \frac{\partial J_i}{\partial y_{i+1}} &= \int_{x_i}^{x_{i+1}} \Bigg[p(x)L_{i+1}'\left\{\begin{bmatrix} L_i' & L_{i+1}' \end{bmatrix}\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix}\right\} \\ &\quad +q(x)L_{i+1}\left\{\begin{bmatrix} L_i & L_{i+1} \end{bmatrix}\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix}\right\} - r(x)L_{i+1}\Bigg] dx = 0. \end{aligned} \tag{8.114}$$

These two equations can be written in matrix form as

$$\begin{aligned} &\int_{x_i}^{x_{i+1}} \left\{p(x)\begin{bmatrix} L_i'L_i' & L_i'L_{i+1}' \\ L_{i+1}'L_i' & L_{i+1}'L_{i+1}' \end{bmatrix} + q(x)\begin{bmatrix} L_iL_i & L_iL_{i+1} \\ L_{i+1}L_i & L_{i+1}L_{i+1} \end{bmatrix}\right\}\begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix} dx \\ &\qquad - \int_{x_i}^{x_{i+1}} r(x)\begin{bmatrix} L_i \\ L_{i+1} \end{bmatrix} dx = 0. \end{aligned}$$

This gives

$$\mathbf{A^{(i)}}\phi^{(\mathbf{i})} - \mathbf{b^{(i)}} = \mathbf{0}, \tag{8.115}$$

where

$$\begin{aligned}
\mathbf{A^{(i)}} &= \int_{x_i}^{x_{i+1}} \left\{ p(x) \begin{bmatrix} L_i'L_i' & L_i'L_{i+1}' \\ L_{i+1}'L_i' & L_{i+1}'L_{i+1}' \end{bmatrix} + q(x) \begin{bmatrix} L_iL_i & L_iL_{i+1} \\ L_{i+1}L_i & L_{i+1}L_{i+1} \end{bmatrix} \right\} dx, \\
\mathbf{b^{(i)}} &= \int_{x_i}^{x_{i+1}} r(x) \begin{bmatrix} L_i \\ L_{i+1} \end{bmatrix} dx, \qquad \text{and} \qquad \phi^{(\mathbf{i})} = \begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix}.
\end{aligned} \tag{8.116}$$

Step 4. Assembly of element equations

Let the elements of the matrix $\mathbf{A^{(i)}}$ be taken as

$$\mathbf{A^{(i)}} = \begin{bmatrix} a_{i,i}^{(i)} & a_{i,i+1}^{(i)} \\ a_{i+1,i}^{(i)} & a_{i+1,i+1}^{(i)} \end{bmatrix} \text{ and those of } \mathbf{b^{(i)}} \text{ as } \mathbf{b^{(i)}} = \begin{bmatrix} b_i^{(i)} \\ b_{i+1}^{(i)} \end{bmatrix}.$$

Substituting $i = 0, 1, 2, \ldots, n-1$ in (8.115) and taking summation over all the elements as

$$\begin{aligned}
&\begin{bmatrix} a_{0,0}^{(0)} & a_{0,1}^{(0)} \\ a_{1,0}^{(0)} & a_{1,1}^{(0)} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} + \begin{bmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} \\ a_{2,1}^{(1)} & a_{2,2}^{(1)} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \cdots + \begin{bmatrix} a_{i,i}^{(i)} & a_{i,i+1}^{(i)} \\ a_{i+1,i}^{(i)} & a_{i+1,i+1}^{(i)} \end{bmatrix} \begin{bmatrix} y_i \\ y_{i+1} \end{bmatrix} \\
&+ \cdots + \begin{bmatrix} a_{n-1,n-1}^{(n-1)} & a_{n-1,n}^{(n-1)} \\ a_{n,n-1}^{(n-1)} & a_{n,n}^{(n-1)} \end{bmatrix} \begin{bmatrix} y_{n-1} \\ y_n \end{bmatrix} \\
= &\begin{bmatrix} b_0^{(0)} \\ b_1^{(0)} \end{bmatrix} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \end{bmatrix} + \cdots + \begin{bmatrix} b_i^{(i)} \\ b_{i+1}^{(i)} \end{bmatrix} + \cdots + \begin{bmatrix} b_{n-1}^{(n-1)} \\ b_n^{(n-1)} \end{bmatrix}.
\end{aligned}$$

After simplification, the assembled matrix equation becomes

$$\begin{bmatrix}
a_{0,0}^{(0)} & a_{0,1}^{(0)} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\
a_{1,0}^{(0)} & a_{1,1}^{(0)} + a_{1,1}^{(1)} & a_{1,2}^{(1)} & 0 & 0 & \cdots & 0 & 0 & 0 \\
0 & a_{2,1}^{(1)} & a_{2,2}^{(1)} + a_{2,2}^{(2)} & a_{2,2}^{(2)} & 0 & \cdots & 0 & 0 & 0 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & a_{n,n-1}^{(n-1)} & a_{n,n}^{(n-1)}
\end{bmatrix}
\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}
=
\begin{bmatrix} b_0^{(0)} \\ b_1^{(0)} + b_1^{(1)} \\ b_2^{(1)} + b_2^{(2)} \\ \vdots \\ b_n^{(n-1)} \end{bmatrix}.$$

This equation can be written as

$$\mathbf{A}\phi = \mathbf{b}. \tag{8.117}$$

It may be noted that $\mathbf{A}$ is a tri-diagonal matrix. The solution of this system can be determined by the method discussed in Chapter 5.

Step 5. Incorporation of boundary conditions

For simplicity, let the system (8.117) be of the following form.

$$\begin{bmatrix} a_{0,0} & a_{0,1} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_{1,0} & a_{1,1} & a_{12} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_{2,1} & a_{2,2} & a_{2,3} & 0 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & a_{n,n-1} & a_{n,n} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}. \tag{8.118}$$

The second and nth equations are $a_{1,0}y_0 + a_{1,1}y_1 + a_{1,2}y_2 = b_1$ and $a_{n-1,n-2}y_{n-2} + a_{n-1,n-1}y_{n-1} + a_{n-1,n}y_n = b_{n-1}$. Now, the boundary conditions $y(a) = y_a$ and $y(b) = y_b$ are introduced to the above equations and they become

$$\begin{aligned} & a_{1,1}y_1 + a_{1,2}y_2 = b_1 - a_{1,0}y_a \\ \text{and} \quad & a_{n-1,n-2}y_{n-2} + a_{n-1,n-1}y_{n-1} = b_{n-1} - a_{n-1,n}y_n. \end{aligned} \tag{8.119}$$

Now, first and last rows and columns are removed from $\mathbf{A}$ also first and last elements are removed from $\mathbf{b}$ and the equations of (8.119) are incorporated.

The equation (8.118) finally reduces to

$$\begin{bmatrix} a_{1,1} & a_{12} & 0 & 0 & \cdots & 0 & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & a_{n-1,n-2} & a_{n-1,n-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix}. \tag{8.120}$$

The above equation is a tri-diagonal system of equations containing $(n-1)$ unknowns $y_1, y_2, \ldots, y_{n-1}$.

Example 8.9.1 Solve the boundary value problem

$$\frac{d^2y}{dx^2} + 2y = x, \qquad y(0) = 1, \qquad y(1) = 2$$

using finite element method for two and four elements of equal lengths.

Solution. Here $p(x) = -1, q(x) = 2, r(x) = x$. If the lengths of elements are equal then $h_i = h$ (say) for all i.

Now, $L'_i = -\frac{1}{h}, L'_{i+1} = \frac{1}{h}$.

Therefore,

$$\mathbf{A}^{(\mathbf{i})} = \int_{x_i}^{x_{i+1}} \left\{ -\begin{bmatrix} L_i'L_i' & L_i'L_{i+1}' \\ L_{i+1}'L_i' & L_{i+1}'L_{i+1}' \end{bmatrix} + 2\begin{bmatrix} L_iL_i & L_iL_{i+1} \\ L_{i+1}L_i & L_{i+1}L_{i+1} \end{bmatrix} \right\} dx$$
$$= -\frac{1}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \frac{h}{3}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

$$\int_{x_i}^{x_{i+1}} x\, L_i\, dx = \int_{x_i}^{x_{i+1}} \frac{x(x_{i+1}-x)}{h}\, dx = \frac{h}{6}(3x_i+h)$$
and
$$\int_{x_i}^{x_{i+1}} x\, L_{i+1}\, dx = \int_{x_i}^{x_{i+1}} \frac{x(x-x_i)}{h}\, dx = \frac{h}{6}(3x_i+2h).$$

Then

$$\mathbf{b}^{(\mathbf{i})} = \int_{x_i}^{x_{i+1}} x \begin{bmatrix} L_i \\ L_{i+1} \end{bmatrix} dx = \frac{h}{6}\begin{bmatrix} 3x_i+h \\ 3x_i+2h \end{bmatrix}.$$

For two elements of equal length.

In this case $h = 1/2$.

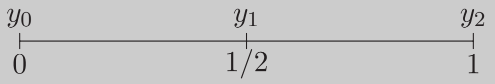


Figure 8.6: Two elements.

For element e_0: $x_0 = 0, x_1 = 1/2$.

$$\mathbf{A}^{(\mathbf{0})} = -2\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = \frac{1}{6}\begin{bmatrix} -10 & 13 \\ 13 & -10 \end{bmatrix}$$
$$\mathbf{b}^{(\mathbf{0})} = \frac{1}{12}\begin{bmatrix} 1/2 \\ 1 \end{bmatrix} = \frac{1}{24}\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \qquad \phi^{(0)} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}.$$

For element e_1: $x_1 = 1/2, x_2 = 1$.

$$\mathbf{A}^{(\mathbf{1})} = \frac{1}{6}\begin{bmatrix} -10 & 13 \\ 13 & -10 \end{bmatrix}, \quad \mathbf{b}^{(\mathbf{1})} = \frac{1}{24}\begin{bmatrix} 4 \\ 5 \end{bmatrix}, \qquad \phi^{(1)} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.$$

Combination of these two element equations gives

$$\frac{1}{6}\begin{bmatrix} -10 & 13 & 0 \\ 13 & -10 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} + \frac{1}{6}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -10 & 13 \\ 0 & 13 & -10 \end{bmatrix}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \frac{1}{24}\begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} + \frac{1}{24}\begin{bmatrix} 0 \\ 4 \\ 5 \end{bmatrix}$$

$$\text{or, } 4\begin{bmatrix} -10 & 13 & 0 \\ 13 & -20 & 13 \\ 0 & 13 & -10 \end{bmatrix}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 6 \\ 5 \end{bmatrix}.$$

Incorporating boundary conditions $y_0 = 1, y_2 = 2$, the second equation is written as $4(13y_0 - 20y_1 + 13y_2) = 6$.
This gives $80y_1 = 52(y_0 + y_2) - 6$, that is, $y_1 = 15/8 = 1.875$.
The solution of the given equation for two elements can also be written as

$$y(x) = \begin{cases} \dfrac{x_1 - x}{x_1 - x_0}y_0 + \dfrac{x - x_0}{x_1 - x_0}y_1 = 1 + 1.75x, & 0 \le x \le 1/2 \\ \dfrac{x_2 - x}{x_2 - x_1}y_1 + \dfrac{x - x_1}{x_2 - x_1}y_2 = 0.25x + 1.75, & 1/2 \le x \le 1 \end{cases}$$

For four elements of equal length.

In this case $h = 1/4$.

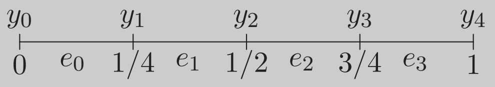


Figure 8.7: Four elements.

For element e_0: $x_0 = 0, x_1 = 1/4$.

$$\mathbf{A}^{(0)} = \frac{1}{12}\begin{bmatrix} -46 & 49 \\ 49 & -46 \end{bmatrix}, \quad \mathbf{b}^{(0)} = \frac{1}{96}\begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \phi^{(0)} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}.$$

For element e_1: $x_1 = 1/4, x_2 = 1/2$.

$$\mathbf{A}^{(1)} = \frac{1}{12}\begin{bmatrix} -46 & 49 \\ 49 & -46 \end{bmatrix}, \quad \mathbf{b}^{(1)} = \frac{1}{96}\begin{bmatrix} 4 \\ 5 \end{bmatrix}, \quad \phi^{(1)} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.$$

For element e_2: $x_2 = 1/2, x_3 = 3/4$.

$$\mathbf{A}^{(2)} = \frac{1}{12}\begin{bmatrix} -46 & 49 \\ 49 & -46 \end{bmatrix}, \quad \mathbf{b}^{(2)} = \frac{1}{96}\begin{bmatrix} 7 \\ 8 \end{bmatrix}, \quad \phi^{(2)} = \begin{bmatrix} y_2 \\ y_3 \end{bmatrix}.$$

For element e_3: $x_3 = 3/4, x_4 = 1$.

$$\mathbf{A^{(3)}} = \frac{1}{12}\begin{bmatrix} -46 & 49 \\ 49 & -46 \end{bmatrix}, \qquad \mathbf{b^{(3)}} = \frac{1}{96}\begin{bmatrix} 10 \\ 11 \end{bmatrix}, \qquad \phi^{(3)} = \begin{bmatrix} y_3 \\ y_4 \end{bmatrix}.$$

Now, combining four element equations as

$$\frac{1}{12}\begin{bmatrix} -46 & 49 & 0 & 0 & 0 \\ 49 & -46 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} + \frac{1}{12}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -46 & 49 & 0 & 0 \\ 0 & 49 & -46 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

$$+\frac{1}{12}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -46 & 49 & 0 \\ 0 & 0 & 49 & -46 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} + \frac{1}{12}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -46 & 49 \\ 0 & 0 & 0 & 49 & -46 \end{bmatrix}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

$$= \frac{1}{96}\begin{bmatrix} 1 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{96}\begin{bmatrix} 0 \\ 4 \\ 5 \\ 0 \\ 0 \end{bmatrix} + \frac{1}{96}\begin{bmatrix} 0 \\ 0 \\ 7 \\ 8 \\ 0 \end{bmatrix} + \frac{1}{96}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 10 \\ 11 \end{bmatrix}.$$

After simplification, the above equation reduces to

$$\frac{1}{12}\begin{bmatrix} -46 & 49 & 0 & 0 & 0 \\ 49 & -92 & 49 & 0 & 0 \\ 0 & 49 & -92 & 49 & 0 \\ 0 & 0 & 49 & -92 & 49 \\ 0 & 0 & 0 & 49 & -46 \end{bmatrix}\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \frac{1}{96}\begin{bmatrix} 1 \\ 6 \\ 12 \\ 18 \\ 11 \end{bmatrix}.$$

Incorporating boundary conditions $y_0 = 1, y_4 = 2$ the above system reduces to [using (8.120)]

$$\begin{bmatrix} -92 & 49 & 0 \\ 49 & -92 & 49 \\ 0 & 49 & -92 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} -193 \\ 6 \\ -383 \end{bmatrix}.$$

Solution of this system is

$$y_1 = 1.53062, y_2 = 1.88913, y_3 = 2.04693.$$

That is, $y(0) = 1, y(1/4) = 1.53062, y(1/2) = 1.88913, y(3/4) = 2.04693, y(1) = 2$.

The solution at each element is

$$y(x) = \begin{cases} \dfrac{x_1 - x}{x_1 - x_0} y_0 + \dfrac{x - x_0}{x_1 - x_0} y_1 = 1 + 2.12248x, & 0 \le x \le 1/4 \\[2ex] \dfrac{x_2 - x}{x_2 - x_1} y_1 + \dfrac{x - x_1}{x_2 - x_1} y_2 = 1.17211 + 1.43404x, & 1/4 \le x \le 1/2 \\[2ex] \dfrac{x_3 - x}{x_3 - x_2} y_2 + \dfrac{x - x_2}{x_3 - x_2} y_3 = 1.57353 + 0.63120x, & 1/2 \le x \le 3/4 \\[2ex] \dfrac{x_4 - x}{x_4 - x_3} y_3 + \dfrac{x - x_3}{x_4 - x_3} y_4 = 2.18772 - 0.18772x, & 3/4 \le x \le 1. \end{cases}$$

The exact solution of the equation is

$$y = \cos\sqrt{2}x + 1.3607\ \sin\sqrt{2}x + \frac{x}{2}.$$

8.10 Discussion About the Methods

In this section, the advantages and disadvantages of the numerical methods to solve ordinary differential equations presented in this chapter are placed.

The Taylor's series method has a serious disadvantage that higher order derivatives of $f(x, y)$ are required while computing y at a given value of x. Since this method involves several computations of higher order derivatives, it is a labourious method, but, once the series is available then one can compute the values of y at different values of x, provided the step size is small enough.

The Picard's method involves successive integrations and it is difficult for a complicated function.

The Euler's method is the simplest and the most crude of all single-step methods and gives a rough idea about the solution.

The Runge-Kutta methods are most widely used method to solve a single or a system of IVP, though, it is laborious. These methods can also be used to solve higher order equations. To get the starting values of some predictor-corrector methods the Runge-Kutta methods are used. After the starting values are found, the remaining values can be determined by using predictor-corrector methods. An important drawback of Runge-Kutta method is that there is no technique to check or estimate the error occurs at any step. If an error generates at any step, then it is propagated through the subsequent steps without detection.

The multistep method is, in general, more efficient than a single-step method. When starting values are available then at each step only one (for explicit method) or a few (for implicit method) functions evaluations are required. In this contrast, a single-step method requires multiple functions evaluations, but, single-step method is self starting.

The finite difference method is useful to solve second order IVP and BVP. To solve BVP, an algebraic tri-diagonal system of equations is generated. When the step size h is small then this method gives better result, but, for small h, the number of equations becomes very large and then it is very complicated to solve such system.

8.11 Stability Analysis

Stability analysis is an important part in the study of the numerical methods to solve differential equations. Most of the methods used to solve differential equation are based on difference equation. To study the stability, the model differential and difference equations are defined in the following.

8.11.1 Model differential problem

For convenience and feasibility of analytical treatment and without loss of generality, stability analysis will be performed on the **model initial value differential equation**

$$y' = \lambda y, \qquad y(0) = y_0, \tag{8.121}$$

where λ is a constant and it may be a real or a complex number. The solution of this problem is

$$y = e^{\lambda t} y_0. \tag{8.122}$$

In our treatment, let us consider $\lambda = \lambda_R + i\lambda_I$, where λ_R and λ_I represent respectively the real and imaginary parts of λ, with $\lambda_R \leq 0$.

8.11.2 Model difference problem

Similar to the differential problem, let us consider the single first order linear model initial value difference problem

$$y_{n+1} = \sigma y_n, \quad n = 0, 1, 2, \ldots \tag{8.123}$$

where y_0 is given and σ is, in general, a complex number. The solution of this problem is

$$y_n = \sigma^n y_0. \tag{8.124}$$

It may be noted that the solution remains bounded only if $|\sigma| \leq 1$.

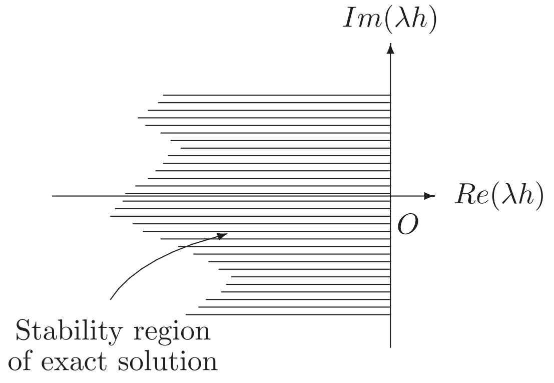


Figure 8.8: Stability region of exact solution.

The connection between the exact solution and the difference solution is evident if we evaluate the exact solution at $t_n = nh$, for $n = 0, 1, \ldots$ where $h > 0$ and

$$y_n = e^{\lambda t_n} y_0 = e^{\lambda nh} y_0 = \sigma^n y_0 \tag{8.125}$$

where $\sigma = e^{\lambda h}$.

If the exact solution is bounded then $|\sigma| = |e^{\lambda h}| \leq 1$. This is possible if $Re(\lambda h) = \lambda_R h \leq 0$.

That is, in the $Re(\lambda h)$-$Im(\lambda h)$ plane, the region of stability of the exact solution is the left half-plane as shown in Figure 8.8.

The single-step method is called **absolutely stable** if $|\sigma| \leq 1$ and **relatively stable** if $|\sigma| \leq e^{\lambda h}$. If λ is pure imaginary and $|\sigma| = 1$, then the absolute stability is called the **periodic stability (P-stability)**.

When the region of stability of a difference equation is identical to the region of stability of the differential equation, the finite difference scheme is sometimes referred to as ***A*-stable**.

8.11.3 Stability of Euler's method

The solution scheme of (8.121) by Euler's method is

$$y_{n+1} = y_n + hf(x_n, y_n) = y_n + \lambda h y_n = (1 + \lambda h)y_n. \tag{8.126}$$

The solution of this difference equation is

$$y_n = (1 + \lambda h)^n y_0 = \sigma^n y_0, \tag{8.127}$$

where $\sigma = 1 + \lambda h$.

The numerical method is stable if $|\sigma| \leq 1$.

Now, the different cases are discussed for different nature of λ.

(i) Real λ: $|1+\lambda h| < 1$ or, $-2 < \lambda h < 0$,

(ii) Pure imaginary λ: Let $\lambda = iw$, w is real.
Then $|1+iwh| = \sqrt{1+w^2h^2} > 1$. That is, the method is not stable when λ is pure imaginary.

(iii) Complex λ: Let $\lambda = \lambda_R + i\lambda_I$. Then
$|\sigma| = |1+\lambda h| = |1+\lambda_R h + i\lambda_I h| = \sqrt{(1+\lambda_R h)^2 + (\lambda_I h)^2} \le 1$.
It means λh lies inside the unit circle.

That is, only a small portion of the left half-plane is the region of stability for the Euler's method. This region is inside the circle $(1+\lambda_R h)^2 + (\lambda_I h)^2 = 1$, which is shown in Figure 8.9.

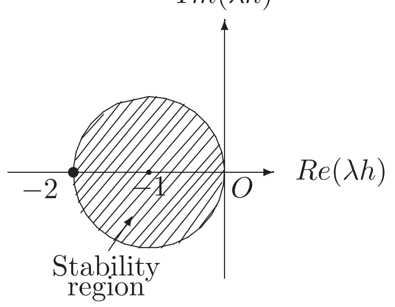


Figure 8.9: Stability region of Euler's method.

For any value of λh in the left half-plane and outside this circle, the numerical solution blows-up while the exact solution decays. Thus the numerical method is **conditionally** stable.

To get a stable numerical solution, the step size h must be reduced so that λh falls within the circle. If λ is real and negative then the maximum step size for stability is $0 \le h \le 2/|\lambda|$. The circle is only tangent to the imaginary axis. If λ is real and the numerical solution is unstable, then $|1+\lambda h| > 1$, which means that $(1+\lambda h)$ is negative with magnitude greater than 1. Since $y_n = (1+\lambda h)^n y_0$, the numerical solutions exhibits oscillations with changes of sign at every step. This behavior of the numerical solutions is a good indicator of instability.

Note 8.11.1 The numerical stability does not imply accuracy. A method can be stable even if it gives inaccurate result. From the stability point of view, our objective is to use the maximum step size h to reach the final destination at $x = x_n$. If h is large then number of function evaluations is low and needs low computational cost. This may not be the optimum h for acceptable accuracy, but, it is optimum for stability.

8.11.4 Stability of Runge-Kutta methods

Let us consider the second-order Runge-Kutta method for the model equation $y' = \lambda y$. Then

$$\begin{aligned} k_1 &= hf(x_n, y_n) = \lambda h y_n \\ k_2 &= hf(x_n + h, y_n + k_1) = \lambda h(y_n + k_1) = \lambda h(y_n + \lambda h y_n) \\ &= \lambda h(1 + \lambda h) y_n \end{aligned}$$

and

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{2}(k_1 + k_2) = y_n + \left(\lambda h + \frac{(\lambda h)^2}{2}\right) y_n \\ &= \left(1 + \lambda h + \frac{\lambda^2 h^2}{2}\right) y_n. \end{aligned} \tag{8.128}$$

This expression confirms that the method is of second order accuracy. For stability, $|\sigma| \leq 1$ where

$$\sigma = 1 + \lambda h + \frac{\lambda^2 h^2}{2}. \tag{8.129}$$

Now, the stability is discussed for different cases of λ.

(i) Real λ: $1 + \lambda h + \frac{\lambda^2 h^2}{2} \leq 1$ or, $-2 \leq \lambda h \leq 0$.

(ii) Pure imaginary λ: Let $\lambda = iw$. Then $|\sigma| = \sqrt{1 + \frac{1}{4} w^4 h^4} > 1$. That is, the method is unstable.

(iii) Complex λ: Let $1 + \lambda h + \frac{\lambda^2 h^2}{2} = e^{i\theta}$ and find the complex roots, λh, of this polynomial for different values of θ. Note that $|\sigma| = 1$ for all values of θ.

The resulting stability region is shown in Figure 8.10.

When fourth-order Runge-Kutta method is applied to the model equation $y' = \lambda y$ then

$$\begin{aligned} k_1 &= \lambda h y_n \\ k_2 &= \lambda h(y_n + k_1/2) = \lambda h(1 + \lambda h/2) y_n \\ k_3 &= \lambda h(y_n + k_2/2) = \lambda h\Big(1 + \frac{1}{2}\lambda h + \frac{1}{4}\lambda^2 h^2\Big) y_n \\ k_4 &= \lambda h(y_n + k_3) = \lambda h\Big(1 + \lambda h + \frac{1}{2}\lambda^2 h^2 + \frac{1}{4}\lambda^3 h^3\Big) y_n. \end{aligned}$$

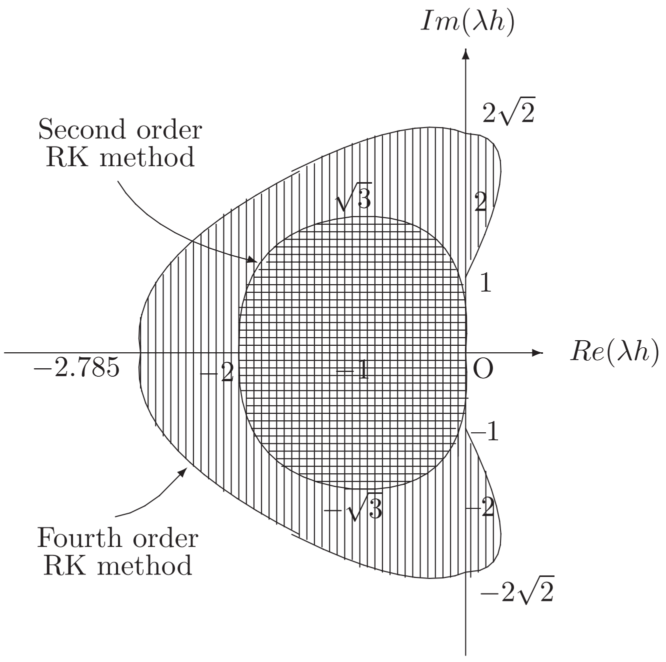


Figure 8.10: Stability regions of Runge-Kutta methods.

Therefore,

$$\begin{aligned} y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ &= \left\{1 + \lambda h + \frac{1}{2!}(\lambda h)^2 + \frac{1}{3!}(\lambda h)^3 + \frac{1}{4!}(\lambda h)^4\right\} y_n, \end{aligned} \tag{8.130}$$

which confirms the fourth-order accuracy of the method.

For different λ, the stability of fourth-order Runge-Kutta method is discussed in the following:

(i) Real λ: $-2.785 \le \lambda h \le 0$.

(ii) Pure imaginary λ: $0 \le |\lambda h| \le 2\sqrt{2}$.

(iii) Complex λ: In this case, the stability region is obtained by finding the roots of the fourth-order polynomial with complex coefficients:
$1 + \lambda h + \frac{1}{2!}(\lambda h)^2 + \frac{1}{3!}(\lambda h)^3 + \frac{1}{4!}(\lambda h)^4 = e^{i\theta}$.

The region of stability is shown in Figure 8.10. It shows a significant improvement over the second-order Runge-Kutta scheme. In particular, it has a large stability region on the imaginary axis.

8.11.5 Stability of Finite difference method

To test the stability of finite difference method, let us consider a simple second order differential equation

$$y'' + ky' = 0, \tag{8.131}$$

where k is a constant and very large in comparison to 1.

The central difference approximation for (8.131) is

$$\frac{1}{h^2}(y_{i+1} - 2y_i + y_{i+1}) + \frac{k}{2h}(y_{i+1} - y_{i-1}) = 0. \tag{8.132}$$

The solution of this difference equation is

$$y_i = A + B\left(\frac{2 - kh}{2 + kh}\right)^i, \tag{8.133}$$

where A and B are arbitrary constants to be determined by imposing boundary conditions.

The analytical solution of (8.131) is

$$y(x) = A + Be^{-kx}. \tag{8.134}$$

If $k > 0$ and $x \to \infty$ then the solution becomes bounded. The term e^{-kx} is monotonic for $k > 0$ and $k < 0$. Thus we expect the finite difference is also monotonic for $k > 0$ and $k < 0$.

If the behavior of the exponential term of (8.133) is analyzed, it is observed that it behaves monotonic for $k > 0$ and $k < 0$ if $h < |2/k|$. This is the condition for stability of the difference scheme (8.132).

8.12 Exercise

1. Discuss Taylor's series method to solve an initial value problem of the form

$$\frac{dy}{dx} = f(x, y), \qquad y(x_0) = y_0.$$

2. For the differential equation $\dfrac{dy}{dx} - 1 = x^2y$ with $y(0) = 1$, obtain Taylor's series for $y(x)$ and compute the values of $y(0.1), y(0.2)$ correct up to four decimal places.

3. Use Taylor's series method to find the solution of
$$\frac{dy}{dt} = t + y^2 \text{ with } y(1) = 0$$
at $t = 1.5$.

4. Use Taylor's series method to solve $y' = y \sin x + \cos x$ at $x = 0.5, 0.6$ with initial condition $x = 0, y = 0$.

5. Use Picard's method to solve the following differential equation
$$\frac{dy}{dx} + 2y = 0, \qquad y(0) = 1$$
at $x = 0.1$.

6. Using the Picard's method, find three successive approximations to the solutions of the following differential equations:
(a) $y' = 2y(1 + x), \qquad y(0) = 1,$
(b) $y' = x - y, \qquad y(0) = 1.$

7. Deduce Euler's method to solve the initial value problem $y' = f(x, y), y(x_0) = y_0$ using (i) Taylor's series, and (ii) forward difference formula.

8. Discuss Euler's method to solve the differential equation of the form $y' = f(x, y)$, $y(x_0) = y_0$.

9. Solve the following differential equation
$$\frac{dy}{dx} = 3x^2 + y, \qquad y(0) = 4$$
for the range $0.1 \le x \le 0.5$, using Euler's method by taking $h = 0.1$.

10. Obtain numerically the solution of
$$y' = x^2 + y^2, \qquad y(0) = 0.5$$
using Euler's method to find the value of y at $x = 0.1$ and 0.2.

11. Deduce modified Euler's method and compare it with Euler's method.

12. Give geometrical interpretations of Euler's and modified Euler's methods.

13. Using modified Euler's method, evaluate $y(0.1)$ correct to two significant figures from the differential equation
$$\frac{dy}{dx} = y + x, \qquad y = 1 \text{ when } x = 0, \text{ taking } h = 0.05.$$

14. Deduce second and fourth order Runge-Kutta methods to solve the initial value problem $y' = f(x, y)$ with $y(x_0) = y_0$.

15. Explain the significance of the numbers k_1, k_2, k_3, k_4 used in fourth order Runge-Kutta methods.

16. Analyze the stability of second and fourth order Runge-Kutta methods.

17. Use second order Runge-Kutta method to solve the initial value problem
$$5\frac{dy}{dx} = x^2 + y^2, \qquad y(0) = 1$$
and find y in the interval $0 \leq x \leq 0.4$, taking $h = 0.1$.

18. Using Runge-Kutta method of fourth order, solve
$$\frac{dy}{dx} = xy + y^2,$$
given that $y(0) = 1$. Take $h = 0.2$ and find y at $x = 0.2, 0.4, 0.6$.

19. Use Runge-Kutta method of fourth order to compute the solution of the following problem in the interval [0,0.1] with $h = 0.02$.
$$y' = x + y, \qquad y(0) = 1.$$

20. Consider the following system of first order differential equation
$$\frac{dy}{dx} = y + 2z, \qquad \frac{dz}{dx} = 3y + 2z$$
with $y(0) = 6, z(0) = 4$. Use second and fourth order Runge-Kutta methods to find the values of y and z at $x = 0.1, 0.2$.

21. Solve the system $\dot{x} = x + 2y, \dot{y} = 2x + y$ with the initial condition $x(0) = -2$ and $y(0) = 2$ over the interval $0 \leq t \leq 1$ using the step size $h = 0.1$. Fourth order Runge-Kutta method may be used.

22. Use fourth order Runge-Kutta method to solve the second order initial value problem $2y''(x) - 6y'(x) + 2y(x) = 4e^x$ with $y(0) = 1$ and $y'(0) = 1$, at $x = 0.1, 0.2$.

23. Use fourth order Runge-Kutta method to solve $y'' = xy' + y$ with initial conditions $y(0) = 1, y'(0) = 2$. Take $h = 0.2$ and find y and y' at $x = 0.2$.

24. Solve the following initial value problem $y' = -xy$ over [0, 0.2] with $y(0) = 1$ using
(a) fourth order Runge-Kutta method,
(b) Runge-Kutta-Butcher method, and compare them.

25. Solve the following initial value problem $y' + xy = 1$ at $x = 0.1$ with initial condition $y(0) = 1$ using Runge-Kutta-Fehlberg method.

26. Discuss Milne's predictor-corrector formula to find the solution of $y' = f(x, y)$, $y(x_0) = y_0$.

27. Use Milne's predictor-corrector formula to find the solutions at $x = 0.4, 0.5, 0.6$ of the differential equation

$$\frac{dy}{dx} = x^3 + y^2, \qquad y(0) = 1.$$

28. Deduce Adams-Bashforth-Moulton predictor-corrector formula to find the solution of $y' = f(x, y), y(x_0) = y_0$.

29. Consider the initial value problem

$$\frac{dy}{dx} = x^2y + y^2, y(0) = 1.$$

Find the solution of this equation at $x = 0.4, 0.5, 0.6$ using Adams-Bashforth-Moulton predictor-corrector method.

30. Use Adams-Bashforth-Moulton and Milne's predictor-corrector methods with $h = 0.5$ to compute the approximate solution of the IVP $y' = (x - y)/3$ with $y(0) = 1$ over $[0, 2]$. The Runge-Kutta method may be used to find the starting solution.

31. Using Milne's predictor-corrector method, find the solution of $y' = xy + y^2$, $y(0) = 1$ at $x = 0.4$ given that $y(0.1) = 1.11689, y(0.2) = 1.27739$ and $y(0.3) = 1.50412$.

32. Find the solution of the IVP $y' = y^2 \sin x$ with $y(0) = 1$ using Adams-Bashforth-Moulton predictor-corrector method, in the interval [0.2,0.3], given that $y(0.05) = 1.00125, y(0.10) = 1.00502$ and $y(0.15) = 1.01136$.

33. Use Adams-Bashforth method to solve the differential equation $y' = 2x - 3y$ for the initial condition $y(0) = 1$ on the interval [0,1]. The starting solutions may be obtained by Euler's method.

34. Consider the second order initial value problem

$$\frac{d^2y}{dx^2} + xy = 0$$

with initial conditions $y(0) = 1, y'(0) = 0$. Compute $y(0.2)$ and $y(0.4)$ using
(a) Runge-Kutta methods,
(b) Finite difference method, and compare them.

35. Convert the following second order differential equation $y''+p(x)y'+q(x)y+r(x)=0$ with $y(x_0)=y_0, y(x_n)=y_n$ into a system of algebraic equations.

36. Solve the boundary value problem

$$y''=\frac{2x}{1+x^2}y'-\frac{2}{1+x^2}y+1$$

with $y(0)=1.25$ and $y(2)=-0.95$ for $x=0.5, 1.0, 1.5$, using finite difference method.

37. Use finite difference method to solve the BVP $y''+2y'+y=10x$, $y(0)=0$ and $y(1)=0$ at $x=0.25, 0.50, 0.75$, and compare these results with exact solutions.

38. Use the finite difference method to find the solution of the boundary value problem $y''-xy'+2y=x+1$ with boundary conditions $y(0.9)=0.5$, $y'(0.9)=2$, $y(1.2)=1$, taking $h=0.1$.

39. Convert the following BVP $y''-12xy+1=0$ with boundary conditions $y(0)=y(1)=0$, into two initial value problems.

40. Use shooting method to solve the boundary value problem $y''+xy=0$ with boundary conditions $y(0)=0, y(1)=1$.

41. Consider the boundary value problem $y''-y=1,\ 0<x<1$ with the boundary conditions $y(0)=0, y(1)=e-1$.

    Use finite element method to find the solution of this problem.

42. Solve the boundary value problem $y''+y=x^2,\ 0<x<1$ with boundary condition $y(0)=3, y(1)=1$ using finite element method for two and three elements of equal lengths.

43. Use finite element method to obtain the difference scheme for the boundary value problem

$$\frac{d}{dx}\left[(1+x^2)\frac{dy}{dx}\right]-y=1+x^2, y(-1)=y(1)=0$$

with linear shape functions and element length $h=0.25$.

Chapter 9

Partial Differential Equations

The partial differential equation (PDE) is an important subject in applied mathematics, physics and some branches of engineering. Several analytical methods are available to solve PDEs, but, these methods are based on advanced mathematical techniques. Also, several numerical methods are available to solve PDEs. The numerical methods are, in general, simple but generate erroneous result. Among several numerical methods, the finite-difference method is widely used for its simplicity. In this chapter, only finite-difference method is discussed to solve PDEs.

The general second order PDE is of the form

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu = G \tag{9.1}$$

$$\text{i.e.,} \qquad Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G, \tag{9.2}$$

where A, B, C, D, E, F, G are functions of x and y.

The PDEs can be classified into three different types – elliptic, hyperbolic and parabolic. These classifications are done by computing the discriminant

$$\Delta = B^2 - 4AC.$$

The equation (9.2) is called elliptic, parabolic and hyperbolic according as the value of Δ at any point (x, y) are $<, =$ or > 0.

Elliptic equations

The simplest examples of this type of PDEs are Poisson's equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = g(x, y) \tag{9.3}$$

and Laplace equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad \text{or} \qquad \nabla^2 u = 0. \tag{9.4}$$

These equations are generally associated with equilibrium or steady-state problems.

For example, the velocity potential for the steady flow of incompressible non-viscous fluid satisfies Laplace's equation. The electric potential V associated with a two-dimensional electron distribution of charge density ρ satisfies Poisson's equation $\dfrac{\partial^2 V}{\partial x^2} + \dfrac{\partial^2 V}{\partial y^2} = -\dfrac{\rho}{\varepsilon}$, where ε is the dielectric constant.

The analytic solution of an elliptic equation is a function of x and y which satisfies the PDE at every point of the region S which is bounded by a plane closed curve C and satisfies some conditions at every point on C. The condition that the dependent variable satisfies along the boundary curve C is known as **boundary condition**.

Parabolic equation

In parabolic equation, time t is involved as an independent variable.

The simplest example of parabolic equation is the heat conduction equation

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}. \tag{9.5}$$

The solution u is the temperature at a distance x units of length from one end of a thermally insulated bar after t seconds of heat conduction. In this problem, the temperatures at the ends of a bar are known for all time, i.e., the boundary conditions are known.

Hyperbolic equation

In this equation also, time t appears as an independent variable.

The simplest example of hyperbolic equation is the one-dimensional wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}. \tag{9.6}$$

Here u is the transverse displacement at a distance x from one end of a vibrating string of length l after a time t.

Hyperbolic equations generally originate from vibration problems or from problems where discontinuities can persist in time.

The values of u at the ends of the string are usually known for all time (i.e., boundary conditions are known) and the shape and velocity of the string are given at initial time (i.e., initial conditions are known).

Generally, three types of problems occur in PDEs.

(i) **Dirichlet's problem**

Let R be a region bounded by a closed curve C and f be a continuous function on C.

Now the problem is to find u satisfying the Laplace's equation

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} &= 0 \text{ in } R \\ \text{and } u &= f(x, y) \text{ on } C. \end{aligned} \tag{9.7}$$

(ii) **Cauchy's problem**

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= \frac{\partial^2 u}{\partial x^2} \text{ for } t > 0 \\ u(x, 0) &= f(x) \\ \text{and } u_t(x, 0) &= g(x) \\ &\text{where } f(x) \text{and } g(x) \text{ are arbitrary.} \end{aligned} \tag{9.8}$$

(iii)

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} \text{ for } t > 0 \\ \text{and } u(x, 0) &= f(x). \end{aligned} \tag{9.9}$$

The above cited problems are all well-defined (i.e., well-posed) and it can be shown that each of the above problems has unique solution.

But, the problem of Laplace's equation with Cauchy boundary conditions, i.e., the problem

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} &= 0 \\ u(x, 0) &= f(x) \\ \text{and } u_y(x, 0) &= g(x) \end{aligned} \tag{9.10}$$

is an ill-posed problem.

9.1 Finite-Difference Approximations to Partial Derivatives

Let the xy plane be divided into sets of equal rectangles of sides $\Delta x = h$ and $\Delta y = k$ by drawing the equally spaced grid lines parallel to the coordinates axes, defined by

$$\begin{aligned} x_i &= ih, \qquad i = 0, \pm 1, \pm 2, \ldots \\ y_j &= jk, \qquad j = 0, \pm 1, \pm 2, \ldots. \end{aligned}$$

The value of u at a mesh point (intersection of horizontal and vertical lines) $P(x_i, y_j)$ or, at $P(ih, jk)$ is denoted by $u_{i,j}$, i.e., $u_{i,j} = u(x_i, y_j) = u(ih, jk)$.

Now,

$$u_x(ih, jk) = \frac{u_{i+1,j} - u_{i,j}}{h} + O(h) \tag{9.11}$$

(forward difference approximation)

$$= \frac{u_{i,j} - u_{i-1,j}}{h} + O(h) \tag{9.12}$$

(backward difference approximation)

$$= \frac{u_{i+1,j} - u_{i-1,j}}{2h} + O(h^2) \tag{9.13}$$

(central difference approximation)

Similarly,

$$u_y(ih, jk) = \frac{u_{i,j+1} - u_{i,j}}{k} + O(k) \tag{9.14}$$

(forward difference approximation)

$$= \frac{u_{i,j} - u_{i,j-1}}{k} + O(k) \tag{9.15}$$

(backward difference approximation)

$$= \frac{u_{i,j+1} - u_{i,j-1}}{2k} + O(k^2) \tag{9.16}$$

(central difference approximation)

$$u_{xx}(ih, jk) = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} + O(h^2). \tag{9.17}$$

$$u_{yy}(ih, jk) = \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{k^2} + O(k^2). \tag{9.18}$$

9.2 Parabolic Equations

9.2.1 An explicit method

Let us consider the heat conduction equation

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}. \tag{9.19}$$

Using the finite-difference approximation for u_t and u_{xx}, equation (9.19) reduces to

$$\frac{u_{i,j+1} - u_{i,j}}{k} \simeq \alpha \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2}, \tag{9.20}$$

where $x_i = ih$ and $t_j = jk$; $i, j = 0, 1, 2, \ldots$.

This can be written as

$$u_{i,j+1} = ru_{i-1,j} + (1-2r)u_{i,j} + ru_{i+1,j}, \tag{9.21}$$

where $r = \alpha k/h^2$.

This formula gives the unknown 'temperature' $u_{i,j+1}$ at the mesh point $(i, j+1)$ when the values of $u_{i-1,j}, u_{i,j}$ and $u_{i+1,j}$ are known and hence the method is called the explicit method. It can be shown that (by stability analysis) the formula is valid for $0 < r \leq 1/2$.

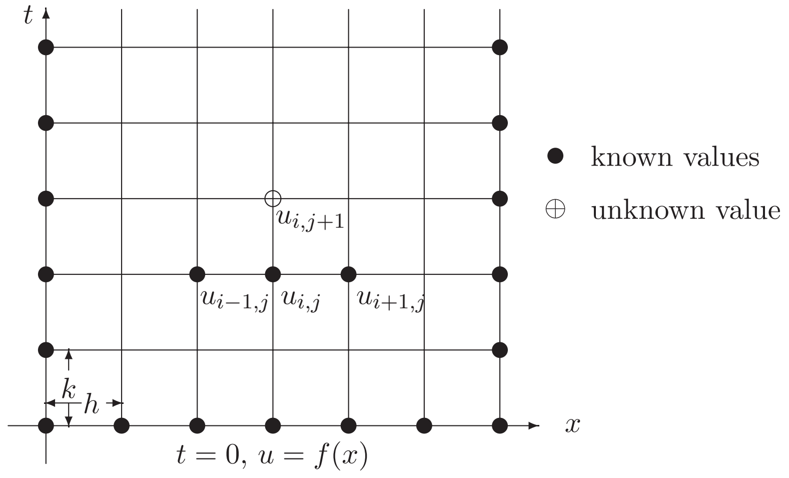


Figure 9.1: Known and unknown meshes of heat equation.

Example 9.2.1 Solve the heat equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

subject to the boundary conditions $u(0,t) = 0, u(1,t) = 2t$ and initial condition $u(x,0) = \frac{1}{2}x$.

Solution. Let $h = 0.2$ and $k = 0.01$, so $r = k/h^2 = 0.25 < 1/2$. The initial and boundary values are shown in the following table.

	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
	$x=0$	$x=0.2$	$x=0.4$	$x=0.6$	$x=0.8$	$x=1.0$
$j=0, t=0.00$	0.0	0.1	0.2	0.3	0.4	0.00
$j=1, t=0.01$	0.0					0.02
$j=2, t=0.02$	0.0					0.04
$j=3, t=0.03$	0.0					0.06
$j=4, t=0.04$	0.0					0.08
$j=5, t=0.05$	0.0					0.10

For this problem the equation (9.21) reduces to

$$u_{i,j+1} = \frac{1}{4}(u_{i-1,j} + 2u_{i,j} + u_{i+1,j}).$$

Then

$$u_{1,1} = \frac{1}{4}(u_{0,0} + 2u_{1,0} + u_{2,0}) = 0.1$$
$$u_{2,1} = \frac{1}{4}(u_{1,0} + 2u_{2,0} + u_{3,0}) = 0.2$$

and so on.
The complete values are shown in the following table.

	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
	$x=0$	$x=0.2$	$x=0.4$	$x=0.6$	$x=0.8$	$x=1.0$
$j=0, t=0.00$	0.0	0.1	0.2	0.3	0.4	0.00
$j=1, t=0.01$	0.0	0.10000	0.20000	0.30000	0.27500	0.02
$j=2, t=0.02$	0.0	0.10000	0.20000	0.26875	0.21750	0.04
$j=3, t=0.03$	0.0	0.10000	0.19219	0.23875	0.18594	0.06
$j=4, t=0.04$	0.0	0.09805	0.18078	0.21391	0.16766	0.08
$j=5, t=0.05$	0.0	0.09422	0.16838	0.19406	0.15730	0.10

9.2.2 Crank-Nicolson implicit method

The above explicit method is simple but it has one serious drawback. This method gives a meaningfull result if $0 < r \le 1/2$, i.e., $0 < \alpha k/h^2 \le 1/2$ or, $\alpha k \le h^2/2$. This means, the time step k is necessarily small. Crank and Nicolson (1947) have developed a method that reduces the total computation time and is valid for all finite values of r. In this method, the equation is approximated by replacing both space and time derivatives by their central difference approximations at a point $(ih, (j+1/2)k)$, which is the midpoint of the points (ih, jk) and $(ih, (j+1)k)$. Thus the equation (9.19) can be written as

$$\left(\frac{\partial u}{\partial t}\right)_{i,j+1/2} = \alpha\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j+1/2} = \frac{\alpha}{2}\left[\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} + \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j+1}\right]. \tag{9.22}$$

Then by using central difference approximation the above equation reduces to

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{\alpha}{2}\left[\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{k^2} + \frac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{k^2}\right]$$

and after simplification this equation gives

$$-ru_{i-1,j+1} + (2+2r)u_{i,j+1} - ru_{i+1,j+1} = ru_{i-1,j} + (2-2r)u_{i,j} + ru_{i+1,j}, \tag{9.23}$$

where $r = \alpha k/h^2$.

In general, the left hand side of (9.23) contains three unknowns and the right hand side has three known values of u.

For $j = 0$ and $i = 1, 2, \ldots, N-1$, equation (9.23) generates N simultaneous equations for $N-1$ unknown $u_{1,1}, u_{2,1}, \ldots, u_{N-1,1}$ (of first row) in terms of known initial and boundary values $u_{0,0}, u_{1,0}, u_{2,0}, \ldots, u_{N,0}$; $u_{0,0}$ and $u_{N,0}$ are the boundary values and $u_{1,0}, u_{2,0}, \ldots, u_{N-1,0}$ are the initial values.

Similarly, for $j = 1$ and $i = 1, 2, \ldots, N-1$ we obtain another set of unknown values $u_{1,2}, u_{2,2}, \ldots, u_{N-1,2}$ in terms of calculated values for $j = 0$, and so on.

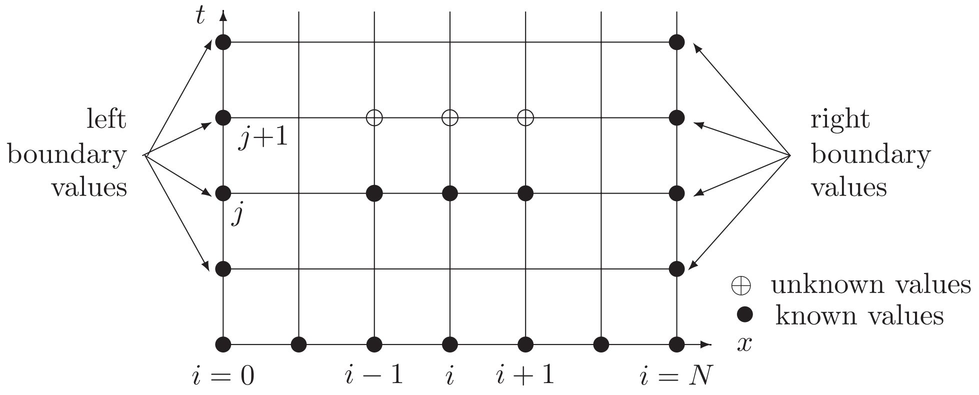


Figure 9.2: Meshes of Crank-Nicolson implicit method.

In this method, the value of $u_{i,j+1}$ is not given directly in terms of known $u_{i,j}$ at one time step earlier but is also a function of unknown values at previous time step, and hence the method is called an implicit method.

The system of equation (9.23) can be viewed in the following matrix notation.

$$\begin{bmatrix} 2+2r & -r & & & & & \\ & -r & 2+2r & -r & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & & -r & 2+2r & -r \\ & & & & & -r & 2+2r \end{bmatrix} \begin{bmatrix} u_{1,j+1} \\ u_{2,j+1} \\ u_{3,j+1} \\ \vdots \\ u_{N-1,j+1} \end{bmatrix} = \begin{bmatrix} d_{1,j} \\ d_{2,j} \\ d_{3,j} \\ \vdots \\ d_{N-1,j} \end{bmatrix} \tag{9.24}$$

where

$$\begin{aligned} d_{1,j} &= ru_{0,j} + (2-2r)u_{1,j} + ru_{2,j} + ru_{0,j+1} \\ d_{i,j} &= ru_{i-1,j} + (2-2r)u_{i,j} + ru_{i+1,j}; \qquad i = 2, 3, \ldots, N-2 \\ d_{N-1,j} &= ru_{N-2,j} + (2-2r)u_{N-1,j} + ru_{N,j} + ru_{N,j+1}. \end{aligned}$$

The right hand side of (9.24) is known.

This resulting tri-diagonal system can be solved by the method discussed in Chapter 5.

Example 9.2.2 Use the Crank-Nicolson method to calculate a numerical solution of the problem

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \qquad 0 < x < 1, t > 0$$

where $u(0,t) = u(1,t) = 0, t > 0, u(x,0) = 2x, t = 0$. Mention the value of $u\left(\frac{1}{2}, \frac{1}{8}\right)$ by taking $h = \frac{1}{2}, \frac{1}{4}$ and $k = \frac{1}{8}, \frac{1}{16}$.

Solution. Case I.
Let $h = \frac{1}{2}$ and $k = \frac{1}{8}$.

In this case $r = \frac{k}{h^2} = \frac{1}{2}$.
The Crank-Nicolson scheme (9.23) now becomes

$$-u_{i-1,j+1} + 6u_{i,j+1} - u_{i+1,j+1} = u_{i-1,j} + 2u_{i,j} + u_{i+1,j}.$$

The boundary and initial conditions are shown in Figure 9.3.

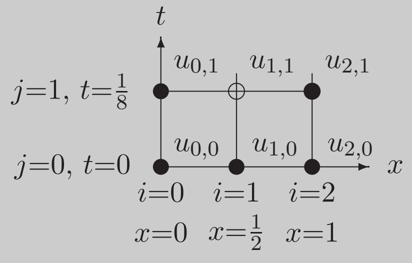


Figure 9.3: Boundary and initial values when $h = 1/2, k = 1/8$.

The initial and boundary values are

$$u_{0,0} = 0, u_{1,0} = 1, u_{2,0} = 2; u_{0,1} = 0, u_{2,1} = 0.$$

Substituting $i = 0$ and $j = 0$ to the Crank-Nicolson scheme, we obtain

$$-u_{0,1} + 6u_{1,1} - u_{2,1} = u_{0,0} + 2u_{1,0} + u_{2,0}.$$

Using initial and boundary conditions the above equation reduces to

$$6u_{1,1} = 2 + 2 = 4, \text{ i.e., } u_{1,1} = u(1/2, 1/8) = 2/3.$$

Case II.
Let $h = \dfrac{1}{4}$ and $k = \dfrac{1}{8}$. In this case $r = \dfrac{k}{h^2} = 2$.
The Crank-Nicolson scheme is

$$-u_{i-1,j+1} + 3u_{i,j+1} - u_{i+1,j+1} = u_{i-1,j} - u_{i,j} + u_{i+1,j}.$$

The initial and boundary conditions are shown in Figure 9.4.

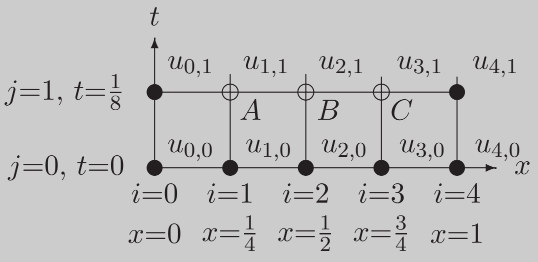

t
$u_{0,1}$
$u_{1,1}$
$u_{2,1}$
$u_{3,1}$
$u_{4,1}$
$j{=}1,\ t{=}\frac{1}{8}$
A
B
C
$u_{0,0}$
$u_{1,0}$
$u_{2,0}$
$u_{3,0}$
$u_{4,0}$
$j{=}0,\ t{=}0$
x
$i{=}0$ $i{=}1$ $i{=}2$ $i{=}3$ $i{=}4$
$x{=}0$ $x{=}\frac{1}{4}$ $x{=}\frac{1}{2}$ $x{=}\frac{3}{4}$ $x{=}1$

Figure 9.4: Boundary and initial values when $h = 1/4, k = 1/8$.

That is, $u_{0,0} = 0, u_{1,0} = \dfrac{1}{2}, u_{2,0} = 1, u_{3,0} = \dfrac{3}{2}, u_{4,0} = 2; u_{0,1} = 0, u_{4,1} = 0$.
The Crank-Nicolson equations for the mesh points $A(i = 1), B(i = 2)$ and $C(i = 3)$ are respectively.

$$\begin{aligned}
-u_{0,1} + 3u_{1,1} - u_{2,1} &= u_{0,0} - u_{1,0} + u_{2,0}\\
-u_{1,1} + 3u_{2,1} - u_{3,1} &= u_{1,0} - u_{2,0} + u_{3,0}\\
-u_{2,1} + 3u_{3,1} - u_{4,1} &= u_{2,0} - u_{3,0} + u_{4,0}.
\end{aligned}$$

Using initial and boundary conditions the above system becomes

$$\begin{aligned}
0 + 3u_{1,1} - u_{2,1} &= 0 - \frac{1}{2} + 1 = \frac{1}{2}\\
-u_{1,1} + 3u_{2,1} - u_{3,1} &= \frac{1}{2} - 1 + \frac{3}{2} = 1\\
-u_{2,1} + 3u_{3,1} + 0 &= 1 - \frac{3}{2} + 2 = \frac{3}{2}.
\end{aligned}$$

The above system has three equations and three unknowns and the solution is
$u_{1,1} = u\left(\dfrac{1}{4}, \dfrac{1}{8}\right) = \dfrac{17}{42}, \qquad u_{2,1} = u\left(\dfrac{1}{2}, \dfrac{1}{8}\right) = \dfrac{5}{7}, \qquad u_{3,1} = u\left(\dfrac{3}{4}, \dfrac{1}{8}\right) = \dfrac{31}{42}.$

Case III.
Let $h = \dfrac{1}{4}, k = \dfrac{1}{16}$. Then $r = 1$. To find the value of u at $t = \dfrac{1}{8}$ we have to apply two steps instead of one step as in Case I and Case II.

The Crank-Nicolson scheme for this case is

$$-u_{i-1,j+1} + 4u_{i,j+1} - u_{i+1,j+1} = u_{i-1,j} + u_{i+1,j}.$$

The initial and boundary conditions are shown in Figure 9.5.

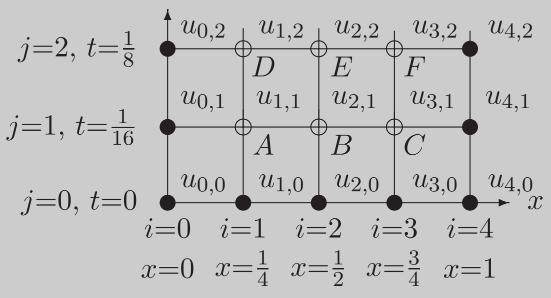


Figure 9.5: Boundary and initial values when $h = 1/4, k = 1/16$.

Also, $u_{0,0} = 0, u_{1,0} = \frac{1}{2}, u_{2,0} = 1, u_{3,0} = \frac{3}{2}, u_{4,0} = 2; u_{0,1} = 0, u_{4,1} = 0; u_{0,2} = 0,$ $u_{4,2} = 0$
The Crank-Nicolson equations for the mesh points $A(i = 1, j = 1)$, $B(i = 2, j = 1)$ and $C(i = 3, j = 1)$ are respectively

$$\begin{aligned}
-u_{0,1} + 4u_{1,1} - u_{2,1} &= u_{0,0} + u_{2,0} \\
-u_{1,1} + 4u_{2,1} - u_{3,1} &= u_{1,0} + u_{3,0} \\
-u_{2,1} + 4u_{3,1} - u_{4,1} &= u_{2,0} + u_{4,0}.
\end{aligned}$$

That is,

$$\begin{aligned}
4u_{1,1} - u_{2,1} &= 1 \\
-u_{1,1} + 4u_{2,1} - u_{3,1} &= \frac{1}{2} + \frac{3}{2} = 2 \\
-u_{2,1} + 4u_{3,1} &= 3.
\end{aligned}$$

The solution of this system is

$$u_{1,1} = u\left(\frac{1}{4}, \frac{1}{16}\right) = \frac{13}{28}, u_{2,1} = u\left(\frac{1}{2}, \frac{1}{16}\right) = \frac{6}{7}, u_{3,1} = u\left(\frac{3}{4}, \frac{1}{16}\right) = \frac{27}{28}.$$

Again, the Crank-Nicolson equations for the mesh points $D(i = 1, j = 2), E(i = 2, j = 2)$ and $F(i = 3, j = 2)$ are respectively,

$$\begin{aligned}
-u_{0,2} + 4u_{1,2} - u_{2,2} &= u_{0,1} + u_{2,1} \\
-u_{1,2} + 4u_{2,2} - u_{3,2} &= u_{1,1} + u_{3,1} \\
-u_{2,2} + 4u_{3,2} - u_{4,2} &= u_{2,1} + u_{4,1}.
\end{aligned}$$

Using boundary conditions and values of right hand side obtained in first step, the above system becomes

$$\begin{aligned}
4u_{1,2} - u_{2,2} &= 0 + \frac{6}{7} = \frac{6}{7}\\
-u_{1,2} + 4u_{2,2} - u_{3,2} &= \frac{13}{28} + \frac{27}{28} = \frac{10}{7}\\
-u_{2,2} + 4u_{3,2} \qquad &= \frac{6}{7} + 0 = \frac{6}{7}.
\end{aligned}$$

The solution of this system is

$$u_{1,2} = u\Big(\frac{1}{4}, \frac{1}{8}\Big) = \frac{17}{49}, u_{2,2} = u\Big(\frac{1}{2}, \frac{1}{8}\Big) = \frac{26}{49}, u_{3,2} = u\Big(\frac{3}{4}, \frac{1}{8}\Big) = \frac{17}{49}.$$

Algorithm 9.1 (Crank-Nicolson). This algorithm solves the heat equation $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$ subject to the boundary conditions $u(0,t) = f_0(t), u(l,t) = f_l(t)$ and initial condition $u(x,0) = g(x)$. The interval $[0,l]$ is divided into N subintervals each of length h. k is the step length of t.

Algorithm Crank_Nicolson
Input functions $f_0(t), f_l(t), g(x)$.
Read N, M;//number of subintervals of $[0,l]$ and $[0,t]$ respectively.//
Read l; //upper limit of x.//
Read k; //step length of t.//
Read α;
Compute $h = l/N, \qquad r = \alpha k/h^2$;
//Construction of the tri-diagonal matrix.//

Let $T = \begin{bmatrix} b_1 & c_1 & 0 & \cdots & 0 & 0 & 0\\ a_2 & b_2 & c_2 & \cdots & 0 & 0 & 0\\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & 0 & \cdots & a_{n-1} & b_{n-1} & c_{n-1}\\ 0 & 0 & 0 & \cdots & \cdots & a_n & b_n \end{bmatrix}$ be the tri-diagonal matrix,

where $b_i = 2 + 2r, i = 1, 2, \ldots, n$; $a_i = -r, i = 2, 3, \ldots, n$;
$c_i = -r, i = 1, 2, \ldots, n-1$.

```
for $i = 1$ to $N - 1$ do
    $u_i = g(i * h)$;
endfor;
for $j = 0$ to $M$ do
    //construction of right hand vector//
    Compute $d_1 = rf_0(j * k) + (2 - 2r)u_1 + ru_2 + rf_0((j+1) * k)$;
    Compute $d_{N-1} = ru_{N-2} + (2 - 2r)u_{N-1} + rf_l(j * k) + rf_l((j+1) * k)$;
endfor;
```

```
    for $i = 2$ to $N - 2$ do
        Compute $d_i = ru_{i-1} + (2 - 2r)u_i + ru_{i+1}$;
    endfor;
    Solve the tri-diagonal system $\mathbf{TU} = \mathbf{D}$, where $\mathbf{U} = (u_1, u_2, \ldots, u_{N-1})^t$
    and $\mathbf{D} = (d_1, d_2, \ldots, d_{N-1})^t$.
    Print 'The values of $u$', $u_1, u_2, \ldots, u_{N-1}$, 'when $t =$', $(j + 1) * k$;
endfor;
end Crank_Nicolson
```

Program 9.1

```
/* Program Crank-Nicolson
   Program to solve the heat equation by Crank-Nicolson
   method. This program solves the problem of the form
   Ut=alpha Uxx, with boundary conditions
   U(0,t)=f0(t); U(l,t)=fl(t); and initial condition
   U(x,0)=g(x). The interval for x [0,l] is divided into
   N subintervals of length h. k is the step length of t.
   Here g(x)=cos(pi*x/2) with u(0,t)=1, u(1,t)=0, h=k=1/3.*/
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
float u[10];
void main()
 {
   int i,j,N,M;
   float h,k,l,alpha,r,a[10],b[10],c[10],d[10],y,temp1,temp2;
   float TriDiag(float a[],float b[],float c[],float d[],int n);
   float g(float x);
   float f0(float t);
   float fl(float t);
   printf("Enter the number subintervals of x and t ");
   scanf("%d %d",&N,&M);
   printf("Enter the step size (k) of t ");
   scanf("%f",&k);
   printf("Enter the upper limit of x ");
   scanf("%f",&l);
   printf("Enter the value of alpha ");
   scanf("%f",&alpha);
   h=l/N;  r=alpha*k/(h*h);
   for(i=1;i<=N;i++) b[i]=2+2*r;
   for(i=2;i<=N;i++) a[i]=-r;
   for(i=1;i<N;i++) c[i]=-r;
   for(i=0;i<=N;i++) u[i]=g(i*h);
   printf("h=%8.5f, k=%8.5f, r=%8.5f\n",h,k,r);
   temp1=f0(0);
   if(fabs(temp1)!=fabs(u[0])){
     printf("u[0]=%f and f0(0)=%f are different!\n",u[0],temp1);
     printf("Enter correct value ");
     scanf("%f",&temp1);}
   temp2=fl(0);
   if(fabs(temp2)!=fabs(u[N]))
     {
       printf("u[N]=%f and fl(0)=%f are different!\n",u[N],temp2);
       printf("Enter correct value ");
       scanf("%f",&temp2);
     }
   printf("Solution is\nx->       ");
   for(i=1;i<N;i++) printf("%8.5f ",i*h); printf("\n");
   for(i=0;i<N;i++) printf("---------");printf("\n");
   for(j=0;j<=M;j++)
      {
       if(j!=0) {temp1=f0(j*k);  temp2=fl(j*k); }
       /* construction of right hand vector */
       d[1]=r*temp1+(2-2*r)*u[1]+r*u[2]+r*f0((j+1)*k);
       d[N-1]=r*u[N-2]+(2-2*r)*u[N-1]+r*temp2+r*fl((j+1)*k);
       for(i=2;i<=N-2;i++)
           d[i]=r*u[i-1]+(2-2*r)*u[i]+r*u[i+1];
       y=TriDiag(a,b,c,d,N-1); /*solution of tri-diagonal system*/
       printf("%8.5f| ",(j+1)*k);
       for(i=1;i<=N-1;i++) printf("%8.5f ",u[i]);printf("\n");
     } /* end of j loop */
} /* main */
/* definitions of the initial and boundary functions*/
float g(float x)
 {
    return(cos(3.141592*x/2));
 }
float f0(float t)
 {
    return(1);
 }
```

```
float fl(float t)
 {
    return(0);
 }
float TriDiag(float a[10],float b[10],float c[10],float d[10],int n)
 {
 /* output u[i], i=1, 2,..., n, is a global variable.*/
 int i; float gamma[10],z[10];
 gamma[1]=b[1];
 for(i=2;i<=n;i++)
    {
     if(gamma[i-1]==0.0)
        {
           printf("A minor is zero: Method fails ");
           exit(0);
        }
     gamma[i]=b[i]-a[i]*c[i-1]/gamma[i-1];
    }
 z[1]=d[1]/gamma[1];
 for(i=2;i<=n;i++)
     z[i]=(d[i]-a[i]*z[i-1])/gamma[i];
 u[n]=z[n];
 for(i=n-1;i>=1;i--)
     u[i]=z[i]-c[i]*u[i+1]/gamma[i];
 /* for(i=1;i<=n;i++) printf("%f ",u[i]); */
 return(u[0]);
} /*end of TriDiag */
```

A sample of input/output:

```
Enter the number subintervals of x and t 4 5
Enter the step size (k) of t 0.03125
Enter the upper limit of x 1
Enter the value of alpha 1
h= 0.25000, k= 0.03125, r= 0.50000
Solution is
x->        0.25000  0.50000  0.75000
------------------------------------
 0.03125|  0.86871  0.65741  0.35498
 0.06250|  0.83538  0.61745  0.33080
 0.09375|  0.81261  0.58746  0.31109
```

```
0.12500|  0.79630  0.56512  0.29579
0.15625|  0.78437  0.54848  0.28420
0.18750|  0.77555  0.53610  0.27550
```

9.3 Hyperbolic Equations

Let us consider the wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, t > 0, 0 < x < 1 \tag{9.25}$$

where initial conditions $u(x, 0) = f(x)$ and

$$\left.\frac{\partial u}{\partial t}\right)_{(x,0)} = g(x), 0 < x < 1 \tag{9.26}$$

and boundary conditions

$$u(0, t) = \phi(t) \text{ and } u(1, t) = \psi(t), t \geq 0. \tag{9.27}$$

This problem may occur in the transverse vibration of a stretched string. As in the previous cases, the central-difference approximation for u_{xx} and u_{tt} at the mesh points $(x_i, t_j) = (ih, jk)$ are

$$\begin{aligned} u_{xx} &= \frac{1}{h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + O(h^2) \\ \text{and } u_{tt} &= \frac{1}{k^2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}) + O(k^2), \end{aligned}$$

where $i, j = 0, 1, 2, \ldots$.

Using the value of u_{xx} and u_{tt}, the equation (9.25) becomes

$$\frac{1}{k^2}(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}) = \frac{c^2}{h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j})$$

i.e.,

$$u_{i,j+1} = r^2 u_{i-1,j} + 2(1 - r^2)u_{i,j} + r^2 u_{i+1,j} - u_{i,j-1}, \tag{9.28}$$

where $r = ck/h$.

The value of $u_{i,j+1}$ depends on the values of u at three time-levels $(j - 1), j$ and $(j + 1)$. The known and unknown values of u are shown in Figure 9.6.

On substituting $j = 0$, the equation (9.28) yields

$$\begin{aligned} u_{i,1} &= r^2 u_{i-1,0} + 2(1 - r^2)u_{i,0} + r^2 u_{i+1,0} - u_{i,-1} \\ &= r^2 f_{i-1} + 2(1 - r^2)f_i + r^2 f_{i+1} - u_{i,-1}, \text{ where } f_i = f(x_i). \end{aligned}$$

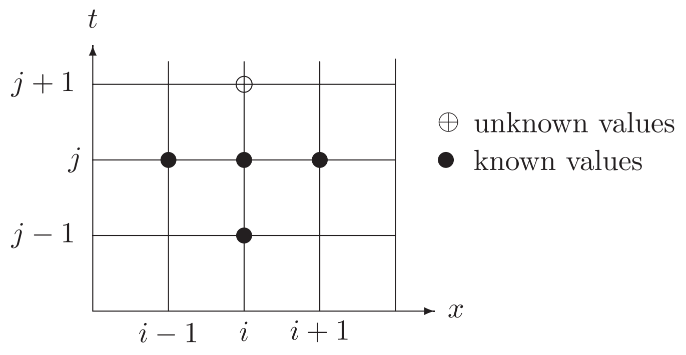


Figure 9.6: Known and unknown meshes for hyperbolic equations.

Again, the central difference approximation to the initial derivative condition gives

$$\frac{1}{2k}(u_{i,1} - u_{i,-1}) = g_i.$$

Eliminating $u_{i,-1}$ between above two relations and we obtain the expression for u along $t = k$, i.e., for $j = 1$ as

$$u_{i,1} = \frac{1}{2}\big[r^2 f_{i-1} + 2(1 - r^2)f_i + r^2 f_{i+1} + 2kg_i\big]. \tag{9.29}$$

The truncation error of this method is $O(h^2+k^2)$ and the formula (9.28) is convergent for $0 < r \leq 1$.

Example 9.3.1 Solve the second-order wave equation

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}$$

with boundary conditions $u = 0$ at $x = 0$ and 1, $t > 0$ and the initial conditions $u = \frac{1}{2}\sin \pi x, \frac{\partial u}{\partial t} = 0$, when $t = 0, 0 \leq x \leq 1$, for $x = 0, 0.2, 0.4, \ldots, 1.0$ and $t = 0, 0.1, 0.2, \ldots, 0.5$.

Solution. The explicit formula is

$$u_{i,j+1} = r^2 u_{i-1,j} + 2(1 - r^2)u_{i,j} + r^2 u_{i+1,j} - u_{i,j-1}.$$

Let $h = 0.2$ and $k = 0.1$, so $r = k/h = 0.5 < 1$.
The boundary conditions become $u_{0,j} = 0, u_{5,j} = 0$ and initial conditions reduce to $u_{i,0} = \frac{1}{2}\sin \pi(ih), i = 1, 2, 3, 4, 5$ and $\frac{u_{i,1} - u_{i,-1}}{2k} = 0$, so that $u_{i,-1} = u_{i,1}$.

For $r = 0.5$, the difference scheme is then

$$u_{i,j+1} = 0.25u_{i-1,j} + 1.5u_{i,j} + 0.25u_{i+1,j} - u_{i,j-1}. \tag{9.30}$$

For $j = 0$, this relation becomes

$$\begin{aligned} u_{i,1} &= 0.25u_{i-1,0} + 1.5u_{i,0} + 0.25u_{i+1,0} - u_{i,-1} \\ \text{i.e., } u_{i,1} &= 0.125u_{i-1,0} + 0.75u_{i,0} + 0.125u_{i+1,0}, [\text{using } u_{i,-1} = u_{i,1}] \\ &= 0.125(u_{i-1,0} + u_{i+1,0}) + 0.75u_{i,0}. \end{aligned}$$

The above formula gives the values of u for $j = 1$. For $j = 2, 3, \ldots$ the values are obtained from the formula (9.30).
Hence,

$$\begin{aligned} u_{1,1} &= 0.125(u_{0,0} + u_{2,0}) + 0.75u_{1,0} = 0.27986 \\ u_{2,1} &= 0.125(u_{1,0} + u_{3,0}) + 0.75u_{2,0} = 0.45282 \\ u_{3,1} &= 0.125(u_{2,0} + u_{4,0}) + 0.75u_{3,0} = 0.45282 \\ u_{4,1} &= 0.125(u_{3,0} + u_{5,0}) + 0.75u_{4,0} = 0.27986. \end{aligned}$$

All values are shown in the following table.

	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$	$i = 5$
	$x = 0$	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1.0$
$j = 0, t = 0.0$	0	0.29389	0.47553	0.47553	0.29389	0
$j = 1, t = 0.1$	0	0.27986	0.45282	0.45282	0.27986	0
$j = 2, t = 0.2$	0	0.23910	0.38688	0.38688	0.23910	0
$j = 3, t = 0.3$	0	0.17552	0.28399	0.28399	0.17552	0
$j = 4, t = 0.4$	0	0.09517	0.15398	0.15398	0.09517	0
$j = 5, t = 0.5$	0	0.00573	0.00927	0.00927	0.00573	0

The exact solution of this equation is

$$u(x, t) = \frac{1}{2} \sin \pi x \, \cos \pi t.$$

9.3.1 Implicit difference methods

The implicit methods generate a tri-diagonal system of algebraic equations. So, it is suggested that the implicit methods should not be used without simplifying assumption to solve pure IVPs because they generate an infinite number of system of equations. But, these methods may be used for initial-boundary value problems. Two such implicit methods are presented below.

(i) At the mesh point (ih, jk) the scheme is

$$\left(\frac{\partial^2 u}{\partial t^2}\right)_{i,j} = \frac{c^2}{2}\left[\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j+1} + \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j-1}\right].$$

That is,

$$\begin{aligned}&\frac{1}{k^2}[u_{i,j+1} - 2u_{i,j} + u_{i,j-1}] \qquad (9.31)\\ &= \frac{c^2}{2h^2}[(u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}) + (u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1})].\end{aligned}$$

(ii) $$\left(\frac{\partial^2 u}{\partial t^2}\right)_{i,j} = \frac{c^2}{4}\left[\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j+1} + 2\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} + \left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j-1}\right]$$

Using the finite difference scheme this equation reduces to

$$\begin{aligned}&\frac{1}{k^2}[u_{i,j+1} - 2u_{i,j} + u_{i,j-1}]\\ &= \frac{c^2}{4h^2}[(u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}) \qquad (9.32)\\ &\quad +2(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + (u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1})].\end{aligned}$$

Both the formulae hold good for all values of $r = ck/h > 0$.

9.4 Elliptic Equations

Let us consider the two-dimensional Laplace equation

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} &= 0 \text{ within } R \qquad (9.33)\\ \text{and} \quad u &= f(x, y) \text{ on the boundary } C.\end{aligned}$$

Using the central difference approximation to both the space derivatives, the finite difference approximation of above equation is given by

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{k^2} = 0.$$

If the mesh points are uniform in both x and y directions then $h = k$ and the above equation becomes a simple one. That is,

$$u_{i,j} = \frac{1}{4}[u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}]. \qquad (9.34)$$

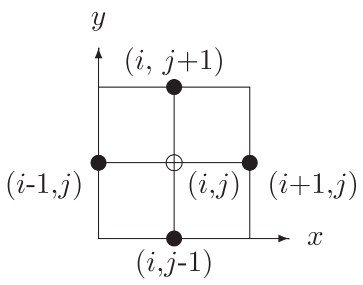


Figure 9.7: Standard five-point formula.

This shows that the value of u at the point (i, j) is the average of its values at the four neighbours – north $(i, j+1)$, east $(i+1, j)$, south $(i, j-1)$ and west $(i-1, j)$. This is shown in Figure 9.7, and the formula is known as **standard five-point formula**.

It is also well known that the Laplace equation remains invariant when the coordinates axes are rotated at an angle 45^0.

Hence equation (9.34) can also be expressed in the form

$$u_{i,j} = \frac{1}{4}[u_{i-1,j-1} + u_{i+1,j-1} + u_{i+1,j+1} + u_{i-1,j+1}]. \tag{9.35}$$

This formula is known as **diagonal five-point formula** and the mesh points are shown in Figure 9.8.

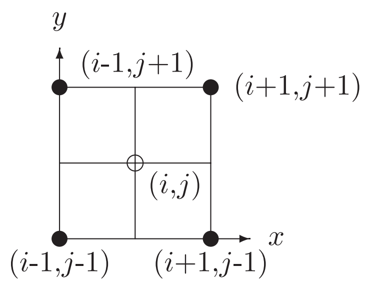


Figure 9.8: Diagonal five-point formula.

Let us consider the Poisson's equation in two-dimension in the form

$$u_{xx} + u_{yy} = g(x, y) \tag{9.36}$$

with the boundary condition $u = f(x, y)$ along C.

Using central difference approximation with uniform mesh points in both x and y directions, the equation (9.36) is given by

$$u_{i,j} = \frac{1}{4}[u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h^2 g_{i,j}] \text{ where } g_{i,j} = g(x_i, y_j). \tag{9.37}$$

Let $u = 0$ be the boundary and $x_i = ih, y_j = jh$ where $i, j = 0, 1, 2, 3, 4$. Then $i, j = 0, 4$ represent the boundary shown in Figure 9.9.

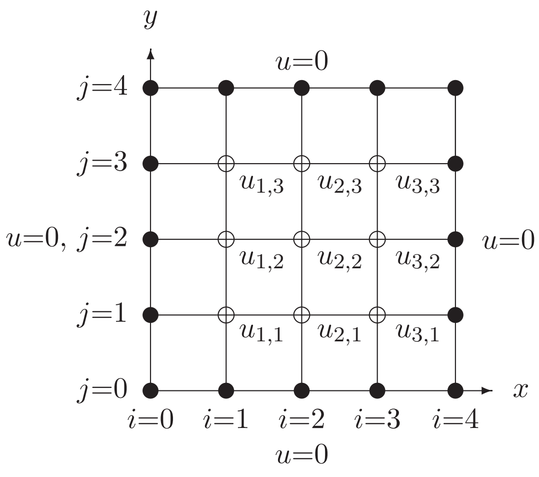


Figure 9.9: The meshes for elliptic equation.

For $i, j = 1, 2$ and 3, the equation (9.37) is a system of nine equations with nine unknowns, which is written in matrix notation as

$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_{1,1} \\ u_{1,2} \\ u_{1,3} \\ u_{2,1} \\ u_{2,2} \\ u_{2,3} \\ u_{3,1} \\ u_{3,2} \\ u_{3,3} \end{bmatrix} = \begin{bmatrix} -h^2 g_{1,1} \\ -h^2 g_{1,2} \\ -h^2 g_{1,3} \\ -h^2 g_{2,1} \\ -h^2 g_{2,2} \\ -h^2 g_{2,3} \\ -h^2 g_{3,1} \\ -h^2 g_{3,2} \\ -h^2 g_{3,3} \end{bmatrix} \tag{9.38}$$

It may be noted that the coefficient matrix is symmetric, positive definite and sparse. Thus the solution of an elliptic PDE depends on a sparse system of equations. To solve this system an iterative method is suggested rather a direct method. Three iterative methods are commonly used to solve such system, viz., (i) Jacobi's method, (ii) Gauss-Seidel's method and (iii) successive overrelaxation method.

The another iterative method known as **alternate direction implicit (ADI)** method is also used.

Method to obtain first approximate value of Laplace's equation

Let us consider the Laplace's equation

$$u_{xx} + u_{yy} = 0.$$

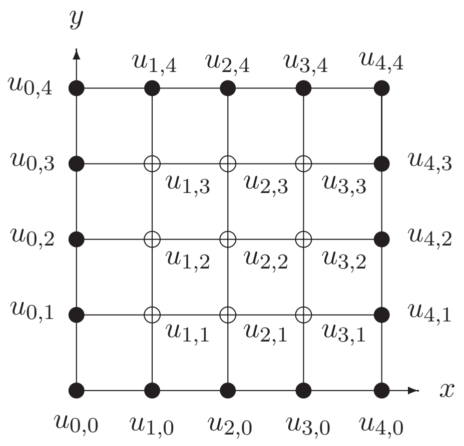


Figure 9.10: Boundary values for Laplace equation.

Let R be a square region divided into $N \times N$ small squares of side h. The boundary values are $u_{o,j}, u_{N,j}, u_{i,0}, u_{i,N}$ where $i, j = 0, 1, 2, \ldots, N$, shown in Figure 9.10.

Initially, diagonal five-point formula is used to compute the values of $u_{2,2}, u_{1,3}$, $u_{3,3}$, $u_{1,1}$ and $u_{3,1}$ in this sequence. The values are

$$\begin{aligned}
u_{2,2} &= \frac{1}{4}(u_{0,0} + u_{4,4} + u_{0,4} + u_{4,0})\\
u_{1,3} &= \frac{1}{4}(u_{0,2} + u_{2,4} + u_{0,4} + u_{2,2})\\
u_{3,3} &= \frac{1}{4}(u_{2,2} + u_{4,4} + u_{2,4} + u_{4,2})\\
u_{1,1} &= \frac{1}{4}(u_{0,0} + u_{2,2} + u_{0,2} + u_{2,0})\\
u_{3,1} &= \frac{1}{4}(u_{2,0} + u_{4,2} + u_{2,2} + u_{4,0}).
\end{aligned}$$

In the second step, the remaining values, viz., $u_{2,3}, u_{1,2}, u_{3,2}$ and $u_{2,1}$ are computed using standard five-point formula as

$$\begin{aligned}
u_{2,3} &= \frac{1}{4}(u_{1,3} + u_{3,3} + u_{2,2} + u_{2,4})\\
u_{1,2} &= \frac{1}{4}(u_{0,2} + u_{2,2} + u_{1,1} + u_{1,3})
\end{aligned}$$

$$u_{3,2} = \frac{1}{4}(u_{2,2} + u_{4,2} + u_{3,1} + u_{3,3})$$
$$u_{2,1} = \frac{1}{4}(u_{1,1} + u_{3,1} + u_{2,0} + u_{2,2}).$$

These are the initial values of u at the nine internal mesh points, their accuracy can be improved by any iterative method.

Note 9.4.1 The above scheme gives a better first approximate values, but, $u_{i,j} = 0$ for $i, j = 1, 2, \ldots, N-1$ may also be taken as first approximate values of u. Sometimes for these initial values the iteration converges slowly.

Example 9.4.1 Find the first approximate values at the interior mesh points of the following Dirichlet's problem

$$\begin{aligned} &u_{xx} + u_{yy} = 0, \\ &u(x,0) = 0, \quad u(0,y) = 0, \\ &u(x,1) = 10x, \quad u(1,y) = 20y. \end{aligned}$$

Solution. Let the region $0 \le x, y \le 1$ be divided into 4×4 squares of side $h = 0.25$. Let $x_i = ih, y_j = jk$, $i, j = 0, 1, 2, 3, 4$. The mesh points are shown in Figure 9.11.

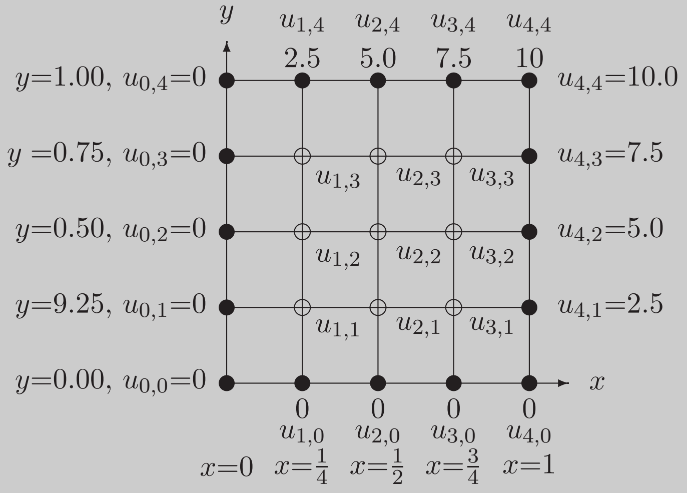

y
$u_{1,4}$ $u_{2,4}$ $u_{3,4}$ $u_{4,4}$
2.5 5.0 7.5 10
y=1.00, $u_{0,4}$=0
$u_{4,4}$=10.0
y =0.75, $u_{0,3}$=0
$u_{4,3}$=7.5
$u_{1,3}$ $u_{2,3}$ $u_{3,3}$
y=0.50, $u_{0,2}$=0
$u_{4,2}$=5.0
$u_{1,2}$ $u_{2,2}$ $u_{3,2}$
y=9.25, $u_{0,1}$=0
$u_{4,1}$=2.5
$u_{1,1}$ $u_{2,1}$ $u_{3,1}$
y=0.00, $u_{0,0}$=0
x
0 0 0 0
$u_{1,0}$ $u_{2,0}$ $u_{3,0}$ $u_{4,0}$
x=0 $x=\frac{1}{4}$ $x=\frac{1}{2}$ $x=\frac{3}{4}$ x=1

Figure 9.11: Mesh points for Dirichlet's problem.

The diagonal five-point formula is used to find the values of $u_{2,2}, u_{1,3}, u_{3,3}, u_{1,1}$ and $u_{3,1}$. Thus

$$u_{2,2} = \frac{1}{4}(u_{0,0} + u_{4,4} + u_{0,4} + u_{4,0}) = \frac{1}{4}(0 + 10 + 0 + 0) = 2.5$$

$$u_{1,3} = \frac{1}{4}(u_{0,2} + u_{2,4} + u_{0,4} + u_{2,2}) = \frac{1}{4}(0 + 5 + 0 + 2.5) = 1.875$$
$$u_{3,3} = \frac{1}{4}(u_{2,2} + u_{4,4} + u_{2,4} + u_{4,2}) = \frac{1}{4}(2.5 + 10 + 5 + 5) = 5.625$$
$$u_{1,1} = \frac{1}{4}(u_{0,0} + u_{2,2} + u_{0,2} + u_{2,0}) = \frac{1}{4}(0 + 2.5 + 0 + 0) = 0.625$$
$$u_{3,1} = \frac{1}{4}(u_{2,0} + u_{4,2} + u_{2,2} + u_{4,0}) = \frac{1}{4}(0 + 5 + 2.5 + 0) = 1.875.$$

The values of $u_{2,3}, u_{1,2}, u_{3,2}$ and $u_{2,1}$ are obtained by using standard five-point formula.

$$u_{2,3} = \frac{1}{4}(u_{1,3} + u_{3,3} + u_{2,2} + u_{2,4}) = \frac{1}{4}(1.875 + 5.625 + 2.5 + 5) = 3.75$$
$$u_{1,2} = \frac{1}{4}(u_{0,2} + u_{2,2} + u_{1,1} + u_{1,3}) = \frac{1}{4}(0 + 2.5 + 0.625 + 1.875) = 1.25$$
$$u_{3,2} = \frac{1}{4}(u_{2,2} + u_{4,2} + u_{3,1} + u_{3,3}) = \frac{1}{4}(2.5 + 5 + 1.875 + 5.625) = 3.75$$
$$u_{2,1} = \frac{1}{4}(u_{1,1} + u_{3,1} + u_{2,0} + u_{2,2}) = \frac{1}{4}(0.625 + 1.875 + 0 + 2.5) = 1.25.$$

Thus we obtain the first approximate values of u at the interior mesh points.

9.4.1 Iterative methods

When first approximate values of u are available then those values can be improved by applying any iterative method. There are a large number of iterative methods available with different rates of convergence. Some of the classical methods are discussed below.

The finite-difference scheme for the Poison's equation (9.36) when discretized using standard five-point formula is

$$u_{i,j} = \frac{1}{4}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h^2 g_{i,j}). \tag{9.39}$$

Let $u_{i,j}^{(r)}$ denote the rth iterative value of $u_{i,j}$.

Jacobi's method

The iterative scheme to solve (9.39) for the interior mesh points is

$$u_{i,j}^{(r+1)} = \frac{1}{4}[u_{i-1,j}^{(r)} + u_{i+1,j}^{(r)} + u_{i,j-1}^{(r)} + u_{i,j+1}^{(r)} - h^2 g_{i,j}]. \tag{9.40}$$

This is called the **point Jacobi's method**.

Gauss-Seidel's method

In this method, the most recently computed values as soon as they are available are used and the values of u along each row are computed systematically from left to right. The iterative formula is

$$u_{i,j}^{(r+1)} = \frac{1}{4}[u_{i-1,j}^{(r+1)} + u_{i+1,j}^{(r)} + u_{i,j-1}^{(r+1)} + u_{i,j+1}^{(r)} - h^2 g_{i,j}]. \tag{9.41}$$

The rate of convergence of this method is twice as fast as the Jacobi's method.

Successive Over-Relaxation or S.O.R. method

In this method, the rate of convergence of an iterative method is accelerated by making corrections on $[u_{i,j}^{(r+1)} - u_{i,j}^{(r)}]$. If $\overline{u_{i,j}^{(r+1)}}$ is the value obtained from a basic iterative (such as Jacobi's or Gauss-Seidel's) method then the value at the next iteration is given by

$$u_{i,j}^{(r+1)} = w\overline{u_{i,j}^{(r+1)}} + (1-w)u_{i,j}^{(r)}, \tag{9.42}$$

where w is the over-relaxation factor.

Thus, the Jacobi's over-relaxation method for the Poisson's equation is

$$u_{i,j}^{(r+1)} = \frac{1}{4}w\left[u_{i-1,j}^{(r)} + u_{i+1,j}^{(r)} + u_{i,j-1}^{(r)} + u_{i,j+1}^{(r)} - h^2 g_{i,j}\right] + (1-w)u_{i,j}^{(r)} \tag{9.43}$$

and the Gauss-Seidel's over-relaxation method for that is

$$u_{i,j}^{(r+1)} = \frac{1}{4}w\left[u_{i-1,j}^{(r+1)} + u_{i+1,j}^{(r)} + u_{i,j-1}^{(r+1)} + u_{i,j+1}^{(r)} - h^2 g_{i,j}\right] + (1-w)u_{i,j}^{(r)}. \tag{9.44}$$

The rate of convergence of (9.43) and (9.44) depends on the value of w and its value lies between 1 and 2. But, the choice of w is a difficult task.

It may be noted that $w = 1$ gives the corresponding basic iteration formula.

Example 9.4.2 Solve the Laplace's equation $\nabla^2 u = 0$ with boundary conditions shown in Figure 9.12.

Solution. The quantities $u_{2,2}, u_{1,3}, u_{3,3}, u_{1,1}$ and $u_{3,1}$ are computed using diagonal five-point formula and let those be the first approximate values. That is,

$$u_{2,2}^{(1)} = \frac{1}{4}(u_{0,0} + u_{4,4} + u_{0,4} + u_{4,0}) = 45.0$$

$$u_{1,3}^{(1)} = \frac{1}{4}(u_{0,2} + u_{2,4} + u_{0,4} + u_{2,2}) = 23.75$$

$$u_{3,3}^{(1)} = \frac{1}{4}(u_{2,2} + u_{4,4} + u_{2,4} + u_{4,2}) = 41.25$$

$$u_{1,1}^{(1)} = \frac{1}{4}(u_{0,0} + u_{2,2} + u_{0,2} + u_{2,0}) = 48.75$$

$$u_{3,1}^{(1)} = \frac{1}{4}(u_{2,0} + u_{4,2} + u_{2,2} + u_{4,0}) = 66.25.$$

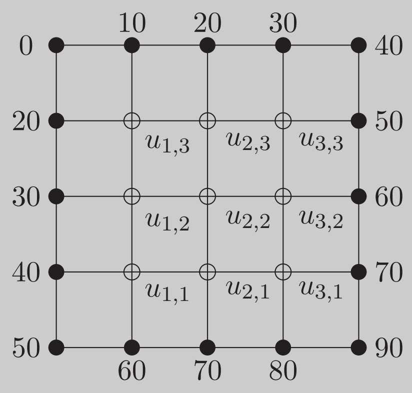


Figure 9.12: Boundary conditions of Laplace equation.

The values of $u_{2,3}, u_{1,2}, u_{3,2}$ and $u_{2,1}$ are computed using standard five-point formula.

$$u_{2,3}^{(1)} = \frac{1}{4}(u_{1,3} + u_{3,3} + u_{2,2} + u_{2,4}) = 32.5$$
$$u_{1,2}^{(1)} = \frac{1}{4}(u_{0,2} + u_{2,2} + u_{1,1} + u_{1,3}) = 36.875$$
$$u_{3,2}^{(1)} = \frac{1}{4}(u_{0,2} + u_{4,2} + u_{3,1} + u_{3,3}) = 53.125$$
$$u_{2,1}^{(1)} = \frac{1}{4}(u_{1,1} + u_{3,1} + u_{2,0} + u_{2,2}) = 57.5.$$

Thus the first approximate values are obtained for nine internal mesh points. These values can be improved by using any iterative method. Here Gauss-Seidel's iterative method is used to obtain better approximate values.

$u_{1,1}$	$u_{2,1}$	$u_{3,1}$	$u_{1,2}$	$u_{2,2}$	$u_{3,2}$	$u_{1,3}$	$u_{2,3}$	$u_{3,3}$
48.5938	57.4609	65.1465	36.8359	44.9805	52.8442	24.8340	32.7661	41.4026
48.5742	57.1753	65.0049	37.0972	44.9707	52.8445	24.9658	32.8348	41.4198
48.5681	57.1359	64.9951	37.1262	44.9854	52.8501	24.9902	32.8489	41.4247
48.5655	57.1365	64.9966	37.1353	44.9927	52.8535	24.9960	32.8534	41.4267
48.5679	57.1393	64.9982	37.1392	44.9963	52.8553	24.9981	32.8553	41.4277
48.5696	57.1410	64.9991	37.1410	44.9982	52.8562	24.9991	32.8562	41.4281
48.5705	57.1419	64.9995	37.1419	44.9991	52.8567	24.9995	32.8567	41.4283
48.5710	57.1424	64.9998	37.1424	44.9995	52.8569	24.9998	32.8569	41.4285
48.5712	57.1426	64.9999	37.1426	44.9998	52.8570	24.9999	32.8570	41.4285
48.5713	57.1427	64.9999	37.1427	44.9999	52.8571	24.9999	32.8571	41.4285

Example 9.4.3 Solve the Laplace's equation $u_{xx} + u_{yy} = 0$ in the domain shown in Figure 9.13, by (a) Gauss-Seidel's method, (b) Jacobi's method, and (c) Gauss-Seidel S.O.R. method.

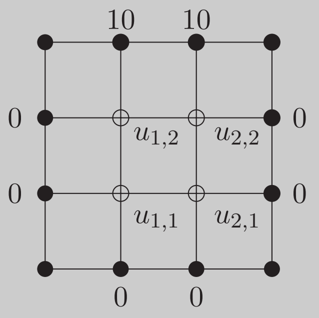


Figure 9.13: Boundary conditions of Laplace equation.

Solution.
(a) Gauss-Seidel's method
Let $u_{2,1} = u_{1,2} = u_{2,2} = u_{1,1} = 0$ at the beginning.
The Gauss-Seidel's iteration scheme is

$$\begin{aligned}
u_{1,1}^{(r+1)} &= \frac{1}{4}\big[u_{2,1}^{(r)} + u_{1,2}^{(r)}\big]\\
u_{2,1}^{(r+1)} &= \frac{1}{4}\big[u_{1,1}^{(r+1)} + u_{2,2}^{(r)}\big]\\
u_{1,2}^{(r+1)} &= \frac{1}{4}\big[u_{1,1}^{(r+1)} + u_{2,2}^{(r)} + 10\big]\\
u_{2,2}^{(r+1)} &= \frac{1}{4}\big[u_{1,2}^{(r+1)} + u_{2,1}^{(r+1)} + 10\big].
\end{aligned}$$

For $r = 0$,

$$\begin{aligned}
u_{1,1}^{(1)} &= \frac{1}{4}\big[0 + 0\big] = 0\\
u_{2,1}^{(1)} &= \frac{1}{4}\big[0 + 0\big] = 0\\
u_{1,2}^{(1)} &= \frac{1}{4}\big[0 + 0 + 10\big] = 2.5\\
u_{2,2}^{(1)} &= \frac{1}{4}\big[2.5 + 0 + 10\big] = 3.125.
\end{aligned}$$

The subsequent iterations are shown below.

$u_{1,1}$	$u_{2,1}$	$u_{1,2}$	$u_{2,2}$
0.00000	0.00000	2.50000	3.12500
0.62500	0.93750	3.43750	3.59375
1.09375	1.17188	3.67188	3.71094
1.21094	1.23047	3.73047	3.74023
1.24023	1.24512	3.74512	3.74756
1.24756	1.24878	3.74878	3.74939
1.24939	1.24969	3.74969	3.74985
1.24985	1.24992	3.74992	3.74996
1.24996	1.24998	3.74998	3.74999
1.24999	1.25000	3.75000	3.75000

(b) Jacobi's method

Initially we take $u_{2,1} = u_{1,2} = u_{2,2} = u_{1,1} = 0$.

The Jacobi's iteration scheme is

$$u_{1,1}^{(r+1)} = \frac{1}{4}\big[u_{2,1}^{(r)} + u_{1,2}^{(r)} + 0 + 0\big] = \frac{1}{4}\big[u_{2,1}^{(r)} + u_{1,2}^{(r)}\big]$$

$$u_{2,1}^{(r+1)} = \frac{1}{4}\big[u_{1,1}^{(r)} + u_{2,2}^{(r)} + 0 + 0\big] = \frac{1}{4}\big[u_{1,1}^{(r)} + u_{2,2}^{(r)}\big]$$

$$u_{1,2}^{(r+1)} = \frac{1}{4}\big[u_{1,1}^{(r)} + u_{2,2}^{(r)} + 0 + 10\big] = \frac{1}{4}\big[u_{1,1}^{(r)} + u_{2,2}^{(r)} + 10\big]$$

$$u_{2,2}^{(r+1)} = \frac{1}{4}\big[u_{1,2}^{(r)} + u_{2,1}^{(r)} + 10 + 0\big] = \frac{1}{4}\big[u_{1,2}^{(r)} + u_{2,1}^{(r)} + 10\big].$$

For $r = 0$, $u_{1,1}^{(1)} = 0, u_{2,1}^{(1)} = 0, u_{1,2}^{(1)} = 2.5, u_{2,2}^{(1)} = 2.5$.

The successive iterations are given below.

$u_{1,1}$	$u_{2,1}$	$u_{1,2}$	$u_{2,2}$
0.00000	0.00000	2.50000	2.50000
0.62500	0.62500	3.12500	3.12500
0.93750	0.93750	3.43750	3.43750
1.09375	1.09375	3.59375	3.59375
1.17188	1.17188	3.67188	3.67188
1.21094	1.21094	3.71094	3.71094
1.23047	1.23047	3.73047	3.73047
1.24023	1.24023	3.74023	3.74023
1.24512	1.24512	3.74512	3.74512
1.24756	1.24756	3.74756	3.74756
1.24878	1.24878	3.74878	3.74878
1.24939	1.24939	3.74939	3.74939
1.24969	1.24969	3.74969	3.74969
1.24985	1.24985	3.74985	3.74985
1.24992	1.24992	3.74992	3.74992

(c) Gauss-Seidel's S.O.R. method
Let $u_{2,1} = u_{1,2} = u_{2,2} = u_{1,1} = 0$ at the beginning.
The Gauss-Seidel's S.O.R. scheme for interior mesh points are

$$u_{i,j}^{(r+1)} = \frac{w}{4}\left[u_{i-1,j}^{(r+1)} + u_{i+1,j}^{(r)} + u_{i,j-1}^{(r+1)} + u_{i,j+1}^{(r)}\right] + (1-w)u_{i,j}^{(r)}.$$

For $j = 1, 2$ and $i = 1, 2$, the formulae are

$$\begin{aligned}
u_{1,1}^{(r+1)} &= \frac{w}{4}[u_{2,1}^{(r)} + u_{1,2}^{(r)}] + (1-w)u_{1,1}^{(r)} \\
u_{2,1}^{(r+1)} &= \frac{w}{4}[u_{1,1}^{(r+1)} + u_{2,2}^{(r)}] + (1-w)u_{2,1}^{(r)} \\
u_{1,2}^{(r+1)} &= \frac{w}{4}[u_{2,2}^{(r)} + u_{1,1}^{(r+1)} + 10] + (1-w)u_{1,2}^{(r)} \\
u_{2,2}^{(r+1)} &= \frac{w}{4}[u_{1,2}^{(r+1)} + u_{3,2}^{(r)} + u_{2,1}^{(r+1)} + 10] + (1-w)u_{2,2}^{(r)}.
\end{aligned}$$

For $w = 1.1$, the values are listed below.

r	$u_{1,1}$	$u_{2,1}$	$u_{1,2}$	$u_{2,2}$
1	0.00000	0.00000	2.75000	3.50625
2	0.75625	1.17219	3.64719	3.72470
3	1.24970	1.25074	3.75324	3.75363
4	1.25113	1.25123	3.75098	3.75025
5	1.25050	1.25008	3.75011	3.75003
6	1.25000	1.25000	3.75000	3.75000

This result shows that the method converges at 6th step. But, to get the same result using Gauss-Seidel's method (in this case $w = 1$) 11 iterations are needed . Also, the rate of convergence depends on the value of w. For some specific values of w, the value of r, at which the solution converges, are tabulated below.

w	1	1.05	1.07	1.08	1.09	1.1	1.2	1.3	1.4	1.009	1.1009	1.109
r	11	9	7	**6**	7	**6**	9	10	16	11	**6**	7

Example 9.4.4 Solve the Poisson's equation $u_{xx}+u_{yy} = 8x^2y^2$ for the square region $0 \le x \le 1, 0 \le y \le 1$ with $h = 1/3$ and the values of u on the boundary are every where zero. Use (a) Gauss-Seidel's method, and (b) Gauss-Seidel's S.O.R. method.

Solution. In this problem, $g(x, y) = 8x^2y^2$, $h = 1/3$ and the boundary conditions are $u_{0,0} = u_{1,0} = u_{2,0} = u_{3,0} = 0$, $u_{0,1} = u_{0,2} = u_{0,3} = 0$,
$u_{1,3} = u_{2,3} = u_{3,3} = 0, u_{3,1} = u_{3,2} = 0$.
(a) The Gauss-Seidel's iteration scheme is

$$u_{i,j}^{(r+1)} = \frac{1}{4}\left[u_{i-1,j}^{(r+1)} + u_{i+1,j}^{(r)} + u_{i,j-1}^{(r+1)} + u_{i,j+1}^{(r)} - h^2 g(ih, jk)\right].$$

Now, $g(ih, jk) = 8h^4 i^2 j^2 = \frac{8}{81} i^2 j^2$. Thus

$$\begin{aligned}
u_{1,1}^{(r+1)} &= \frac{1}{4}\left[u_{0,1}^{(r+1)} + u_{2,1}^{(r)} + u_{1,0}^{(r+1)} + u_{1,2}^{(r)} - \frac{1}{9}.\frac{8}{81}.1.1\right] \\
&= \frac{1}{4}\left[0 + u_{2,1}^{(r)} + 0 + u_{1,2}^{(r)} - \frac{8}{729}\right] = \frac{1}{4}\left[u_{2,1}^{(r)} + u_{1,2}^{(r)} - \frac{8}{729}\right] \\
u_{2,1}^{(r+1)} &= \frac{1}{4}\left[u_{1,1}^{(r+1)} + u_{3,1}^{(r)} + u_{2,0}^{(r+1)} + u_{2,2}^{(r)} - \frac{1}{9}.\frac{8}{81}.2^2.1^2\right] \\
&= \frac{1}{4}\left[u_{1,1}^{(r+1)} + u_{2,2}^{(r)} - \frac{32}{729}\right] \\
u_{1,2}^{(r+1)} &= \frac{1}{4}\left[u_{0,2}^{(r+1)} + u_{2,2}^{(r)} + u_{1,1}^{(r+1)} + u_{1,3}^{(r)} - \frac{1}{9}.\frac{8}{81}.1^2.2^2\right] \\
&= \frac{1}{4}\left[u_{2,2}^{(r)} + u_{1,1}^{(r+1)} - \frac{32}{729}\right] \\
u_{2,2}^{(r+1)} &= \frac{1}{4}\left[u_{1,2}^{(r+1)} + u_{3,2}^{(r)} + u_{2,1}^{(r+1)} + u_{2,3}^{(r)} - \frac{1}{9}.\frac{8}{81}.2^2.2^2\right] \\
&= \frac{1}{4}\left[u_{1,2}^{(r+1)} + u_{2,1}^{(r+1)} - \frac{128}{729}\right].
\end{aligned}$$

Let $u_{2,1}^{(0)} = u_{2,2}^{(0)} = u_{1,2}^{(0)} = 0$.
All the values are shown in the following table.

r	$u_{1,1}$	$u_{2,1}$	$u_{1,2}$	$u_{2,2}$
1	-0.00274	-0.01166	-0.01166	-0.04973
2	-0.00857	-0.02555	-0.02555	-0.05667
3	-0.01552	-0.02902	-0.02902	-0.05841
4	-0.01725	-0.02989	-0.02989	-0.05884
5	-0.01769	-0.03011	-0.03011	-0.05895
6	-0.01780	-0.03016	-0.03016	-0.05898
7	-0.01782	-0.03017	-0.03017	-0.05898
8	-0.01783	-0.03018	-0.03018	-0.05898

(b) The Gauss-Seidel's S.O.R. scheme is

$$\begin{aligned}
u_{i,j}^{(r+1)} &= \frac{w}{4}[u_{i-1,j}^{(r+1)} + u_{i+1,j}^{(r)} + u_{i,j-1}^{(r+1)} + u_{i,j+1}^{(r)} - h^2 g(ih, jh)] + (1-w)u_{i,j}^{(r)} \\
&= \frac{w}{4}\left[u_{i-1,j}^{(r+1)} + u_{i+1,j}^{(r)} + u_{i,j-1}^{(r+1)} + u_{i,j+1}^{(r)} - \frac{8}{729} i^2 j^2\right] + (1-w)u_{i,j}^{(r)}.
\end{aligned}$$

Let the initial values be $u_{2,1}^{(0)} = u_{1,2}^{(0)} = u_{2,2}^{(0)} = 0$.

For $w = 1.1$, the values of $u_{1,1}, u_{2,1}, u_{1,2}$ and $u_{2,2}$ are shown below.

r	$u_{1,1}$	$u_{2,1}$	$u_{1,2}$	$u_{2,2}$
1	-0.00302	-0.01290	-0.01290	-0.05538
2	-0.00981	-0.02871	-0.02871	-0.05854
3	-0.01783	-0.03020	-0.03020	-0.05904
4	-0.01785	-0.03020	-0.03020	-0.05899
5	-0.01784	-0.03018	-0.03018	-0.05899
6	-0.01783	-0.03018	-0.03018	-0.05898

Algorithm 9.2 (Gauss-Seidel's S.O.R. method to solve elliptic equation). This algorithm solves an elliptic (Poisson's) equation of the form $u_{xx} + u_{yy} = g(x, y)$ with given boundary conditions $u_{i,0}, u_{i,N}, u_{0,j}, u_{N,j}, i, j = 0, 1, 2, \dots, N$.

```
Algorithm Elliptic_SOR
Input function $g(x, y)$; //$g(x, y) = 0$ reduces the problem to Laplace equation//
Read boundary conditions
    $u_{i,0}, u_{i,N}, u_{0,j}, u_{N,j}, i, j = 0, 1, 2, \dots, N$;
Read $h, k, \varepsilon$; //$\varepsilon$ is the error tolerance//
Read $w$; //$w = 1$ gives Gauss-Seidel's method//
Step 1. Set initial values for internal mesh points $u_{i,j} = 0$
        for $i, j = 1, 2, \dots, N-1$.
Step 2. for $j = 1, 2, \dots, N-1$ do
        for $i = 1, 2, \dots, N-1$ do
            $u^n_{i,j} = \frac{w}{4}\Big[u^n_{i-1,j} + u_{i+1,j} + u^n_{i,j-1} + u_{i,j+1} - h^2 g(ih, jk)\Big] + (1-w)u_{i,j}$
            //$u^n_{i,j}$ denotes the new value of $u$ at $(i, j)$.//
Step 3. Print $u^n_{i,j}$ for $i, j = 1, 2, \dots, N-1$.
Step 4. If $|u^n_{i,j} - u_{i,j}| < \varepsilon$ for all $i, j = 1, 2, \dots, N-1$ then Stop.
Step 4. Set $u_{i,j} = u^n_{i,j}$ for $i, j = 1, 2, \dots, N-1$
        goto Step 2.
end Elliptic_SOR
```

Program 9.2

```
/* Program Elliptic PDE
   Program to solve the Poisson PDE by Gauss-Seidal S.O.R.
   method. This program solves the problem of the form
   Uxx+Utt=g(x,y). Boundary conditions are given by
   U(x,0)=f1(x); U(0,y)=f2(y); U(x,1)=f3(x); U(1,y)=f4(y).
   g(x,y)=0 gives the Laplace equation.
   Here g(x,y)= -2*x*x+y*y; u(x,y)=0 at boundary.
*/
```

```
#include<stdio.h>
#include<math.h>
void main()
 {
   int i,j,N,flag;
   float u[6][6],un[6][6],h,k,w,eps=1e-5,temp;
   float g(float x,float y);
   float f1(float x);
   float f2(float y);
   float f3(float x);
   float f4(float y);
   printf("Enter the number of mesh points N ");
   scanf("%d",&N);
   printf("Enter the step sizes of x (h) and y (k) ");
   scanf("%f %f",&h,&k);
   printf("Enter the relaxation factor w ");
   scanf("%f",&w);
   /* set boundary conditions */
   for(i=0;i<=N;i++) for(j=0;j<=N;j++) u[i][j]=0;
   for(i=0;i<=N;i++){
        u[i][0]=f1(i*h); u[i][N]=f3(i*h);
      }
   for(j=0;j<=N;j++){
        u[0][j]=f2(j*k); u[N][j]=f4(j*k);
      }
   for(i=0;i<=N;i++)
       for(j=0;j<=N;j++) un[i][j]=u[i][j];
   printf("The values of N, h, k and w are respectively\n");
   printf("N=%3d, h=%6.4f, k=%6.4f, w=%5.3f\n",N,h,k,w);
   do
   {
    for(i=1;i<=N-1;i++)
       for(j=1;j<=N-1;j++) u[i][j]=un[i][j];
    for(i=1;i<=N-1;i++)
       for(j=1;j<=N-1;j++)
          un[i][j]=0.25*w*(un[i-1][j]+u[i+1][j]+un[i][j-1]
                 +u[i][j+1]-h*h*g(i*h,j*k))+(1-w)*u[i][j];
    flag=0;
    for(i=1;i<=N-1;i++)
      for(j=1;j<=N-1;j++) if(fabs(un[i][j]-u[i][j])>eps) flag=1;
   }while(flag==1);
   /* printing of the boundary and internal values */
   printf("The interior and boundary values are shown below\n");
   printf("          ");
   for(i=0;i<=N;i++) printf("%8.5f ",i*h);printf("\n");
   printf("---------");
   for(i=0;i<=N;i++) printf("---------");printf("\n");
   for(i=0;i<=N;i++){
        printf("%8.5f| ",i*k);
        for(j=0;j<=N;j++) printf("%8.5f ",un[i][j]);printf("\n");
     }
 }
/* definitions of the functions */
float g(float x,float y)
 {
   return(-2*x*x+y*y);
 }
float f1(float x)
 {
   return 0;
 }
float f2(float y)
 {
   return 0;
 }
float f3(float x)
 {
   return 0;
 }
float f4(float y)
 {
   return 0;
 }
```

A sample of input/output:

```
Enter the number of mesh points N 5
Enter the step sizes of x (h) and y (k) 0.5 0.1
Enter the relaxation factor w 1.1
The values of N, h, k and w are respectively
N=  5, h=0.5000, k=0.1000, w=1.100
```

```
The interior and boundary values are shown below
          0.00000  0.50000  1.00000  1.50000  2.00000  2.50000
---------------------------------------------------------------
 0.00000|  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
 0.10000|  0.00000  0.37538  0.55816  0.55020  0.35862  0.00000
 0.20000|  0.00000  0.82088  1.19205  1.18154  0.79930  0.00000
 0.30000|  0.00000  1.21861  1.71762  1.70711  1.19702  0.00000
 0.40000|  0.00000  1.21345  1.63770  1.62975  1.19669  0.00000
 0.50000|  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
```

9.5 Stability

To investigate the stability of a finite difference method to solve a PDE, we consider

$$\begin{aligned} u_{p,q} &= e^{i\beta x}e^{\alpha t} = e^{i\beta ph}e^{\alpha qk} \qquad (\text{since } x = ph, y = qk) \\ &= e^{i\beta ph}\xi^q, \end{aligned} \tag{9.45}$$

where $\xi = e^{\alpha k}$ and α is in general, a complex constant.

The finite difference equation will be stable if $|u_{p,q}|$ remain bounded for $qk \le T, 0 \le t \le T$, T is finite as $h \to 0, k \to 0$, and for all values of β, those satisfy the initial conditions.

If the exact solution of the finite difference equation does not increase exponentially with time, then a necessary and sufficient condition for stability is that

$$|\xi| \le 1, \text{ i.e., } -1 \le \xi \le 1. \tag{9.46}$$

But, if $u_{p,q}$ increases with time, then the necessary and sufficient condition for stability is

$$|\xi| \le 1 + O(k). \tag{9.47}$$

Example 9.5.1 Investigate the stability of the parabolic equation $\dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2}$ which is approximated by finite difference scheme

$$\frac{1}{k}(u_{p,q+1} - u_{p,q}) = \frac{1}{h^2}(u_{p-1,q+1} - 2u_{p,q+1} + u_{p+1,q+1})$$

at (ph, qk).

Solution. Substituting $u_{p,q} = e^{i\beta ph}\xi^q$ to the difference equation and we obtain

$$e^{i\beta ph}\xi^{q+1} - e^{i\beta ph}\xi^q = r\{e^{i\beta(p-1)h}\xi^{q+1} - 2e^{i\beta ph}\xi^{q+1} + e^{i\beta(p+1)h}\xi^{q+1}\}$$

where $r = k/h^2$.

Dividing both sides by $e^{i\beta ph}\xi^q$,

$$\begin{aligned}\xi - 1 &= r\xi\{e^{-i\beta h} - 2 + e^{i\beta h}\} = r\xi(2\cos\beta h - 2)\\ &= -4r\xi\sin^2(\beta h/2).\end{aligned}$$

That is,

$$\xi = \frac{1}{1 + 4r\sin^2(\beta h/2)}. \tag{9.48}$$

Clearly $0 < \xi \le 1$ for all $r > 0$ and all β. Therefore, the difference scheme is unconditionally stable.

Example 9.5.2 Investigate the stability of the hyperbolic equation $\dfrac{\partial^2 u}{\partial t^2} = \dfrac{\partial^2 u}{\partial x^2}$, which is approximated by the difference equation

$$\frac{1}{k^2}(u_{p,q+1} - 2u_{p,q} + u_{p,q-1}) = \frac{1}{h^2}(u_{p+1,q} - 2u_{p,q} + u_{p-1,q})$$

at (ph, qk).

Solution. Substituting $u_{p,q} = e^{i\beta ph}\xi^q$ to the above equation, we obtain the relation

$$e^{i\beta ph}\xi^{q+1} - 2e^{i\beta ph}\xi^q + e^{i\beta ph}\xi^{q-1} = r^2\{e^{i\beta(p+1)h}\xi^q - 2e^{i\beta ph}\xi^q + e^{i\beta(p-1)h}\xi^q\}.$$

Now, dividing both sides by $e^{i\beta ph}\xi^q$. Thus

$$\begin{aligned}\xi - 2 + \xi^{-1} &= r^2\{e^{i\beta h} - 2 + e^{-i\beta h}\}\\ &= r^2\{2\cos\beta h - 2\} = -4r^2\sin^2(\beta h/2)\end{aligned}$$

i.e., $\quad \xi^2 - 2\{1 - 2r^2\sin^2(\beta h/2)\}\xi + 1 = 0 \quad$ or, $\quad \xi^2 - 2a\xi + 1 = 0,$

where

$$a = 1 - 2r^2\sin^2(\beta h/2), \qquad r = k/h. \tag{9.49}$$

Let ξ_1 and ξ_2 be two values of ξ and they are given by

$$\xi_1 = a + \sqrt{(a^2 - 1)} \text{ and } \xi_2 = a - \sqrt{(a^2 - 1)}.$$

Here, u does not increase exponentially with t as the difference equation is a three time-level. Thus, a necessary condition for stability is that $|\xi| \le 1$.
It is clear from (9.49) that $a \le 1$ because r, k and β are real.

If $a < -1$, then $|\xi_2| > 1$ and leads to instability, when $-1 \leq a \leq 1$, $a^2 \leq 1$ then ξ_1 and ξ_2 are imaginary and they are given by

$$\xi_1 = a + i\sqrt{(1-a^2)}, \xi_2 = a - i\sqrt{(1-a^2)}.$$

Hence $|\xi_1| = |\xi_2| = \sqrt{a^2 + (1-a^2)} = 1$ and this shows that a necessary condition for stability is $-1 \leq a \leq 1$. That is, $-1 \leq 1 - 2r^2 \sin^2(\beta h/2) \leq 1$. The right hand side inequality is obvious. The left hand side inequality is useful.
Then

$$-1 \leq 1 - 2r^2 \sin^2(\beta h/2) \qquad \text{or,} \qquad r^2 \sin^2(\beta h/2) \leq 1.$$

This gives $r = k/h \leq 1$.
This condition is also a sufficient condition.

9.6 Exercise

1. Solve the heat equation
$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$
subject to the conditions $u(x,0) = 0, u(0,t) = 0$ and $u(1,t) = 2t$, taking $h = 1/2, k = 1/16$.

2. Given the differential equation $\dfrac{\partial^2 u}{\partial x^2} = \dfrac{\partial u}{\partial t}$ and the boundary condition $u(0,t) = u(5,t) = 0$ and $u(x,0) = x^2(30 - x^2)$. Use the explicit method to obtain the solution for $x_i = ih, y_j = jk$; $i = 0,1,\ldots,5$ and $j = 0,1,2,\ldots,6$.

3. Solve the differential equation $\dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2}$, $0 \leq x \leq 1/2$, given that $u = 0$ when $t = 0, 0 \leq x \leq 1/2$ and with boundary conditions $\dfrac{\partial u}{\partial x} = 0$ at $x = 0$ and $\dfrac{\partial u}{\partial x} = 1$ at $x = 1/2$ for $t > 0$, taking $h = 0.1, k = 0.001$.

4. Solve the following initial value problem $f_t = f_{xx}$, $0 \leq x \leq 1$ subject to the initial condition $f(x,0) = \cos\dfrac{\pi x}{2}$ and the boundary conditions $f(0,t) = 1, f(1,t) = 0$ for $t > 0$, taking $h = 1/3, k = 1/3$.

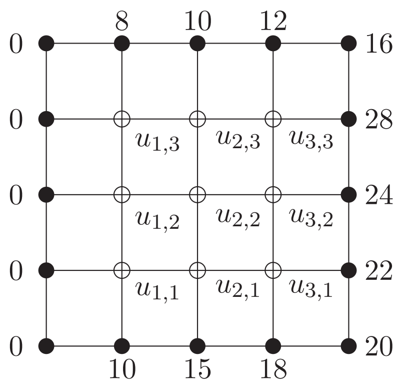
5. Solve the parabolic differential equation $u_t = u_{xx}, 0 \leq x \leq 1$ subject to the boundary conditions $u = 0$ at $x = 0$ and $x = 1$ for $t > 0$ and the initial conditions
$$u(x,0) = \begin{cases} 2x & \text{for } 0 \leq x \leq 1/2 \\ 2(1-x) & \text{for } 1/2 \leq x \leq 1, \end{cases}$$
using explicit method.

6. The differential equation $u_{tt} = u_{xx}, 0 \le x \le 1$ satisfies the boundary conditions $u = 0$ at $x = 0$ and $x = 1$ for $t > 0$, and the initial conditions $u(x,0) = \sin \dfrac{\pi x}{4}$, $\left(\dfrac{\partial u}{\partial t}\right)_{(x,0)} = 0$. Compute the values of u for $x = 0, 0.1, 0.2, \ldots, 0.5$ and $t = 0, 0.1, 0.2, \ldots, 0.5$.

7. Solve the hyperbolic partial differential equation $u_{tt} = u_{xx}, 0 \le x \le 2, t \ge 0$, subject to the boundary conditions $u(0,t) = u(2,t) = 0, t \ge 0$ and the initial conditions $u(x,0) = 5 \sin \dfrac{\pi x}{2}, 0 \le x \le 2, u_t(x,0) = 0$, taking $h = 1/8$ and $k = 1/8$.

8. Solve the Poisson's equation $u_{xx} + u_{yy} = -2x^2 + y^2$ over the region $0 \le x \le 2, 0 \le y \le 2$ taking the boundary condition $u = 0$ on all the boundary sides with $h = 0.5$. Use Gauss-Seidel's method to improve the solution.

9. Solve the Laplace equation $u_{xx} + u_{yy} = 0$ taking $h = 1$, with boundary values as shown below.



10. Solve the elliptic differential equation $u_{xx} + u_{yy} = 0$ and for the region bounded by $0 \le x \le 5$, $0 \le y \le 5$, the boundary conditions being
$u = 0$ at $x = 0$ and $u = 2 + y$ at $x = 5$,
$u = x^2$ at $y = 0$ and $u = 2x$ at $y = 5$.

Take $h = k = 1$. Use
(a) Jacobi's method, (b) Gauss-Seidel's method, and (c) Gauss-Seidel's S.O.R. method.

Chapter 10

Least Squares Approximation

In science and engineering, the experimental data are usually viewed by plotting as graphs on plane paper. But, the problem is: *what is the 'best curve' for a given set of data ?* If n data points $(x_i, y_i), i = 1, 2, \ldots, n$ are given, then an nth degree polynomial can be constructed using interpolation methods, such as Lagrange's, Newton's etc. But, the handling of higher degree polynomial is practically difficult, though it gives exact values at the given nodes $x_0, x_1, \ldots, x_n$, and errors at the other points. For the given data points we can construct lower degree polynomials such as linear, quadratic etc. and other types of curve, viz., geometric, exponential etc., using least squares method, which minimizes the sum of squares of the absolute errors. The curve fitted by this method not always give exact values at the given nodes $x_0, x_1, \ldots, x_n$ and other points.

The least squares method is used to fit polynomials of different degrees, other special curves and also to fit orthogonal polynomials, etc.

10.1 General Least Squares Method

Let $(x_i, y_i), i = 1, 2, \ldots, n$ be a bivariate sample of size n. The problem is to fit the curve

$$y = g(x; a_0, a_1, \ldots, a_k), \tag{10.1}$$

where $a_0, a_1, \ldots, a_k$ are the unknown parameters to be determined based on the given sample values such that the error is minimum.

When $x = x_i$, the value of y obtained from (10.1) is denoted by Y_i and it is given by

$$Y_i = g(x_i; a_0, a_1, \ldots, a_k). \tag{10.2}$$

The quantity Y_i is called the **expected** or **predicted** value of y corresponding to $x = x_i$ and y_i is called the **observed value** of y. These two values, in general, are different as the observed values not necessarily lie on the curve (10.1).

The difference $(y_i - Y_i)$ is called the **residual** corresponding to $x = x_i$. The parameters $a_0, a_1, \ldots, a_k$ are chosen by least squares method in such a way that the sum of the squares of residuals, S, is minimum.

Now,

$$S = \sum_{i=1}^{n}(y_i - Y_i)^2 = \sum_{i=1}^{n}[y_i - g(x_i; a_0, a_1, \ldots, a_k)]^2. \tag{10.3}$$

The expression for S contains $(k+1)$ unknowns $a_0, a_1, \ldots, a_k$.

The value of S will be minimum if

$$\frac{\partial S}{\partial a_0} = 0, \quad \frac{\partial S}{\partial a_1} = 0, \quad \ldots, \quad \frac{\partial S}{\partial a_k} = 0. \tag{10.4}$$

These equations are called **normal equations**. Solution of these $(k+1)$ equations give the values of the parameters $a_0, a_1, \ldots, a_k$. Let $a_0 = a_0^*, a_1 = a_1^*, \ldots, a_k = a_k^*$ be the solution of the system of equations (10.4).

Then the fitted curve is

$$y = g(x; a_0^*, a_1^*, \ldots, a_k^*). \tag{10.5}$$

The sum of the squares of residuals is obtained from the equation

$$S = \sum_{i=1}^{n}(y_i - Y_i)^2 = \sum_{i=1}^{n}[y_i - g(x_i; a_0^*, a_1^*, \ldots, a_k^*)]^2. \tag{10.6}$$

10.2 Fitting of a Straight Line

Let

$$y = a + bx \tag{10.7}$$

be the equation of a straight line, where a and b are two parameters whose values are to be determined. Let (x_i, y_i), $i = 1, 2, \ldots, n$, be a given sample of size n.

Here S is given by

$$S = \sum_{i=1}^{n}(y_i - Y_i)^2 = \sum_{i=1}^{n}(y_i - a - bx_i)^2.$$

The normal equations are

$$\frac{\partial S}{\partial a} = -2\sum(y_i - a - bx_i) = 0$$
$$\frac{\partial S}{\partial b} = -2\sum(y_i - a - bx_i)x_i = 0$$

which give on simplification,

$$\sum y_i = na + b\sum x_i$$
$$\sum x_i y_i = a\sum x_i + b\sum x_i^2. \tag{10.8}$$

The solution of these equations is

$$b = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \qquad \text{and} \qquad a = \frac{1}{n}\Big[\sum y_i - b\sum x_i\Big]. \tag{10.9}$$

But, when the sample size is large or the data are large then there is a chance for data overflow while computing $\sum x_i y_i$, $\sum x_i^2$ and $(\sum x_i)^2$. Then the suggested expression for b is

$$b = \frac{\sum(x-\overline{x})(y-\overline{y})}{\sum(x_i-\overline{x})^2}, \text{ where } \overline{x} = \frac{1}{n}\sum x_i, \quad \overline{y} = \frac{1}{n}\sum y_i. \tag{10.10}$$

Let this solution be denoted by $a = a^*, b = b^*$. Then the fitted straight line is

$$y = a^* + b^* x. \tag{10.11}$$

Example 10.2.1 Use least squares method to fit the line $y = a + bx$ based on the sample $(2, 1)$, $(\frac{1}{6}, -\frac{5}{6})$, $(-\frac{3}{2}, -2)$ and $(-\frac{1}{3}, -\frac{2}{3})$. Estimate the total error.

Solution. Here $n = 4$. The normal equations are

$$\sum y_i = na + b\sum x_i$$
$$\sum x_i y_i = a\sum x_i + b\sum x_i^2$$

The values of $\sum x_i$, $\sum y_i$ and $\sum x_i y_i$ are calculated in the following table.

	x_i	y_i	x_i^2	$x_i y_i$
	2	1	4	2
	1/6	–5/6	1/36	–5/36
	–3/2	–2	9/4	3
	–1/3	–2/3	1/9	2/9
Total	1/3	–5/2	115/18	61/12

The normal equations are then reduced to

$$-\frac{5}{2} = 4a + \frac{1}{3}b \text{ and } \frac{61}{12} = \frac{1}{3}a + \frac{115}{18}b.$$

Solution of these equations is

$$a = -0.6943, b = 0.8319.$$

Therefore, the fitted straight line is

$$y = -0.6943 + 0.8319x.$$

<u>Estimation of error.</u>

x	Given y (y_i)	y, obtained from the curve (Y_i)	$(y_i - Y_i)^2$
2	1	0.9695	0.0009
1/6	–5/6	–0.5557	0.0771
–3/2	–2	–1.9422	0.0033
–1/3	–2/3	–0.9716	0.0930
Total			0.1743

Thus the sum of the squares of error is 0.1743.

Algorithm 10.1 (Straight line fit). This algorithm fits a straight line for the given data points $(x_i, y_i), i = 1, 2, \ldots, n$, by least squares method.

```
Algorithm Straight_Line
 Step 1.  Read $(x_i, y_i), i = 1, 2, \ldots, n$.
 Step 2.  //Computation of $\overline{x}$ and $\overline{y}$//
          Set $sx = 0, sy = 0$.
          for $i = 1$ to $n$ do
               $sx = sx + x_i$ and $sy = sy + y_i$
          endfor;
          Compute $\overline{x} = sx/n$ and $\overline{y} = sy/n$;
 Step 3.  Set $sxy = 0, sx2 = 0$;
          for $i = 1$ to $n$ do
               $sxy = sxy + (x_i - \overline{x})(y_i - \overline{y})$;
               $sx2 = sx2 + (x_i - \overline{x})^2$;
          endfor;
 Step 4.  Compute $b = sxy/sx2$; and $a = \overline{y} - b\overline{x}$;
 Step 5.  Print ‘The fitted line is $y$=’,$a$, ’+’, $b$, ‘$x$’.
end Straight_Line
```

Program 10.1

```
/* Program Straight Line Fit
   Program to fit a straight line for the given data points
   (x[i],y[i]), i=1, 2, . . ., n, by least squares method.*/
#include<stdio.h>
#include<math.h>
void main()
{
 int n,i; float x[50], y[50], sx=0,sy=0,sxy=0,sx2=0,xb,yb,a,b;
 char sign;
 printf("Enter the sample size and the sample (x[i],y[i]) ");
 scanf("%d",&n);
 for(i=1;i<=n;i++) scanf("%f %f",&x[i],&y[i]);
 for(i=1;i<=n;i++){
     sx+=x[i]; sy+=y[i];
   }
 xb=sx/n; yb=sy/n;
 for(i=1;i<=n;i++){
     sxy+=(x[i]-xb)*(y[i]-yb);
     sx2+=(x[i]-xb)*(x[i]-xb);
   }
 b=sxy/sx2; a=yb-b*xb;
 sign=(b<0)? '-':'+';
 printf("\nThe fitted line is y = %f %c %f x",a,sign,fabs(b));
 } /* main */
```

A sample of input/output:

```
Enter the sample size and the sample (x[i],y[i]) 5 1 12 4
13 6 10 7 8 10 3 The fitted line is y = 15.097345 - 1.053097 x
```

10.3 Fitting of a Parabolic Curve

Let

$$y = a + bx + cx^2 \tag{10.12}$$

be the second degree parabolic curve, where a, b, c are unknown parameters and the values of them are to be determined based on the sample values $(x_i, y_i), i = 1, 2, \ldots, n$.

Assume that Y_i is the predicted value of y obtained from the curve (10.12) at $x = x_i$, i.e.,

$$Y_i = a + bx_i + cx_i^2.$$

The sum of squares of residuals

$$S = \sum_{i=1}^{n}(y_i - Y_i)^2 = \sum_{i=1}^{n}(y_i - a - bx_i - cx_i^2)^2.$$

The normal equations are given by

$$\frac{\partial S}{\partial a} = 0, \qquad \frac{\partial S}{\partial b} = 0, \qquad \text{and} \qquad \frac{\partial S}{\partial c} = 0.$$

That is,

$$-2\sum_{i=1}^{n}(y_i - a - bx_i - cx_i^2) = 0 \text{ or } \sum y_i = na + b\sum x_i + c\sum x_i^2$$
$$-2\sum_{i=1}^{n}(y_i - a - bx_i - cx_i^2)x_i = 0 \text{ or } \sum x_iy_i = a\sum x_i + b\sum x_i^2 + c\sum x_i^3$$
$$-2\sum_{i=1}^{n}(y_i - a - bx_i - cx_i^2)x_i^2 = 0 \text{ or } \sum x_i^2y_i = a\sum x_i^2 + b\sum x_i^3 + c\sum x_i^4.$$

Let $a = a^*, b = b^*$ and $c = c^*$ be the solution of the above equations.
Then the fitted parabolic curve is

$$y = a^* + b^*x + c^*x^2 \tag{10.13}$$

Example 10.3.1 Fit a parabola to the following data by taking x as the independent variable.

x	:	1	2	3	4	5	6	7	8	9
y	:	2	6	7	8	10	11	11	10	9

Solution. Let the equation of the parabola be

$$y = a + bu + cu^2$$

where $u = 5 - x$, and u is taking as the independent variable.
Therefore, the normal equations are

$$\begin{aligned}\sum y &= na + b\sum u + c\sum u^2\\ \sum uy &= a\sum u + b\sum u^2 + c\sum u^3\\ \sum u^2y &= a\sum u^2 + b\sum u^3 + c\sum u^4.\end{aligned}$$

The calculations are shown in the following table.

x	u	y	u^2	uy	u^3	u^2y	u^4
1	4	2	16	8	64	32	256
2	3	6	9	18	27	54	81
3	2	7	4	14	8	28	16
4	1	8	1	8	1	8	1
5	0	10	0	0	0	0	0
6	–1	11	1	–11	–1	11	1
7	–2	11	4	–22	–8	44	16
8	–3	10	9	–30	–27	90	81
9	–4	9	16	–36	–64	144	256
Total	0	74	60	–51	0	411	708

Then the above normal equations reduce to

$$\begin{aligned} 74 &= 9a + 0.b + 60.c \\ -51 &= 0.a + 60.b + 0.c \\ 411 &= 60.a + 0.b + 708.c. \end{aligned}$$

Solution of these equations is $b = -0.85, a = 10.0043, c = -0.2673$.
Thus the fitted parabola is

$$\begin{aligned} y &= 10.0043 - 0.85u - 0.2673u^2 = 10.0043 - 0.85(5-x) - 0.2673(5-x)^2 \\ &= -0.9282 + 3.5230x - 0.2673x^3. \end{aligned}$$

10.4 Fitting of a Polynomial of Degree k

Let

$$y = a_0 + a_1x + a_2x^2 + \cdots + a_kx^k, \tag{10.14}$$

be a polynomial or a parabola of degree k and (x_i, y_i), $i = 1, 2, \ldots, n$ be the given sample.

The sum of squares of residuals is

$$S = \sum_{i=0}^{n} \Big[y_i - (a_0 + a_1x_i + a_2x_i^2 + \cdots + a_kx_i^k)\Big]^2. \tag{10.15}$$

As in previous cases, the normal equations are given by

$$\frac{\partial S}{\partial a_0} = 0, \frac{\partial S}{\partial a_1} = 0, \ldots, \frac{\partial S}{\partial a_k} = 0.$$

That is,

$$\begin{aligned}
na_0 + a_1 \sum x_i + a_2 \sum x_i^2 + \cdots + a_k \sum x_i^k &= \sum y_i \\
a_0 \sum x_i + a_1 \sum x_i^2 + a_2 \sum x_i^3 + \cdots + a_k \sum x_i^{k+1} &= \sum x_i y_i \qquad (10.16)\\
a_0 \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4 + \cdots + a_k \sum x_i^{k+2} &= \sum x_i^2 y_i \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
a_0 \sum x_i^k + a_1 \sum x_i^{k+1} + a_2 \sum x_i^{k+2} + \cdots + a_k \sum x_i^{2k} &= \sum x_i^k y_i.
\end{aligned}$$

These $(k+1)$ equations contain $(k+1)$ unknowns $a_0, a_1, \ldots, a_k$. Let $a_0 = a_0^*, a_1 = a_1^*, \ldots, a_k = a_k^*$ be the solution of the above system of linear equations. Then the fitted polynomial is

$$y = a_0^* + a_1^* x + a_2^* x_2 + \cdots + a_k^* x^k. \qquad (10.17)$$

10.5 Fitting of Other Curves

10.5.1 Geometric curve

Let

$$y = ax^b \qquad (10.18)$$

be the geometric curve to be fitted to the given data $(x_i, y_i), i = 1, 2, \ldots, n$, where a, b are unknown parameters.

Taking logarithm on both sides, we obtain the equation

$$\log y = \log a + b \log x.$$

This can be written as

$$Y = A + bX,$$

where

$$Y = \log y, X = \log x, A = \log a.$$

The original points (x_i, y_i) in the xy-plane are transferred to (X_i, Y_i), $i = 1, 2, \ldots, n$ in the XY-plane. This process is called **data linearization**.

Now, using the method described in Section 10.2, one can determine the values of A and b, and a can be obtained from $a = e^A$.

Example 10.5.1 Fit a curve of the type $y = ax^b$ to the following points.

x	:	1	2	3	4	5
y	:	3.5	6.2	9.5	15.3	20.4

Solution. Let $y = ax^b$. Then its corresponding linear curve is $Y = A + bX$, where $\log y = Y, \log x = X, \log a = A$.
The normal equations are

$$\sum Y_i = nA + b\sum X_i$$
$$\sum X_iY_i = A\sum X_i + b\sum X_i^2.$$

The values of $\sum X_i, \sum X_i^2, \sum Y_i, \sum X_iY_i$ are calculated in the following table.

x	y	X	Y	X^2	XY
1	3.5	0	1.25276	0	0
2	6.2	0.69315	1.82455	0.48046	1.26469
3	9.5	1.09861	2.25129	1.20694	2.47329
4	15.3	1.38629	2.72785	1.92180	3.78159
5	20.4	1.60944	3.01553	2.59030	4.85331
Total		4.78749	11.07198	6.19950	12.37288

Then the normal equations are

$$5A + 4.78749b = 11.07198,$$
$$4.78749A + 6.19950b = 12.37288.$$

The solution of these equations is $A = 1.16445, b = 1.09655$ and hence $a = e^A = 3.20416$.
Thus the fitted curve is

$$y = 3.20416\ x^{1.09655}.$$

10.5.2 Rectangular hyperbola

Let the equation of rectangular hyperbola be

$$y = \frac{1}{a + bx}. \tag{10.19}$$

This non-linear equation can be converted to linear form by substituting $Y = 1/y$. The corresponding linear equation then becomes $Y = a + bx$.

Now, using the method discussed in Section 10.2 one can fit the above curve.

10.5.3 Exponential curve

Let the exponential curve be

$$y = ae^{bx}. \tag{10.20}$$

Taking logarithm on both sides, we get

$$\log y = \log a + bx$$

which can be written as

$$Y = A + bx$$

where $Y = \log y, A = \log a$, i.e., $a = e^A$.

Now the parameters A and b can be determined by using the technique adopted in Section 10.2.

10.6 Weighted Least Squares Method

Sometimes it may happen that while fitting a curve based on a given sample $(x_i, y_i), i = 1, 2, \ldots, n$, some data points may be more significant than the other. In this situation, we find a curve such that the curve either pass through that important points or pass very near to that points. The amount of 'importance' can be introduced by assigning weights to that data points. If all the data have same importance, then the weights are set to 1.

10.6.1 Fitting of a weighted straight line

Let $y = a + bx$ be the straight line to be fitted to the given data points $(x_i, y_i), i = 1, 2, \ldots, n$ with weights $w_i, i = 1, 2, \ldots, n$. In this case, the sum of squares of residuals is defined as

$$S = \sum_{i=1}^{n} w_i \Big[y_i - (a + bx_i)\Big]^2. \tag{10.21}$$

For the minimum S,

$$\frac{\partial S}{\partial a} = 0, \qquad \frac{\partial S}{\partial b} = 0.$$

These give

$$-2\sum_{i=1}^{n} w_i[y_i - (a + bx_i)] = 0$$

$$\text{and} \qquad -2\sum_{i=1}^{n} w_i[y_i - (a + bx_i)]x_i = 0$$

After simplification these give a system of linear equations for a and b as

$$a\sum_{i=1}^{n} w_i + b\sum_{i=1}^{n} w_i x_i = \sum_{i=1}^{n} w_i y_i \quad \text{and} \quad a\sum_{i=1}^{n} w_i x_i + b\sum_{i=1}^{n} w_i x_i^2 = \sum_{i=1}^{n} w_i x_i y_i.$$

These are the normal equations and give the values of a and b.

Example 10.6.1 Fit the following data

x	:	0	2	4	6
y	:	10	15	18	25

to a straight line by considering that the data (2,15) and (4,18) are more *significant* or *reliable* with weights 5 and 10 respectively.

Solution. Let the straight line be $y = a + bx$. The normal equations are

$$a\sum w_i + b\sum w_i x_i = \sum w_i y_i$$
$$\text{and} \quad a\sum w_i x_i + b\sum w_i x_i^2 = \sum w_i x_i y_i.$$

The related values are calculated in the following table.

x	y	w	wx	wx^2	wy	wxy
0	10	1	0	0	10	0
2	15	5	10	20	75	150
4	18	10	40	160	180	720
6	25	1	6	36	25	150
Total		17	56	216	290	1020

The normal equations are then $17a + 56b = 290$ and $56a + 216b = 1020$.
The solution of these equations is $a = 10.29851$, $b = 2.05224$.
Thus the fitted line is

$$y = 10.29851 + 2.05224x.$$

Estimation of error.

x	y	w	Predicted y	Absolute error
0	10	1	10.29851	0.29851
2	15	5	14.40299	0.59701
4	18	10	18.50747	0.50747
6	25	1	22.61195	2.38805
Sum of squares of errors				6.40584

Example 10.6.2 Consider the above example again with the modified weights 25 and 40 instead of 5 and 10.

Solution. The modified calculations are shown below.

x	y	w	wx	wx^2	wy	wxy
0	10	1	0	0	10	0
2	15	25	50	100	375	750
4	18	40	160	640	720	2880
6	25	1	6	36	25	150
Total		67	216	776	1130	3780

The normal equations are
$67a + 216b = 1130$ and $216a + 776b = 3780$.
These give $a = 11.31934, b = 1.72039$.
The fitted straight line is

$$y = 11.31934 + 1.72039x.$$

Estimation of error.

x	y	w	Predicted y	Absolute error
0	10	1	11.31934	1.31934
2	15	25	14.76012	0.23988
4	18	40	18.20090	0.20090
6	25	1	21.64168	3.35832
Sum of squares of errors				13.11687

It is observed that when the weights on $x = 2$ and $x = 4$ are increased then the absolute errors in y are reduced at these points, but, the sum of squares of errors is increased due to the less importance of the data $(0, 10)$ and $(6, 25)$.

10.7 Least Squares Method for Continuous Data

In the previous sections, the least squares method is considered for the discrete data. This method is also applicable for continuous data.

Let $y = f(x)$ be a continuous function on $[a, b]$ and it is to be approximated by the kth degree polynomial

$$y = a_0 + a_1x + a_2x^2 + \cdots + a_kx^k. \tag{10.22}$$

In this case, the sum of the squares of residuals S is defined as

$$S = \int_a^b w(x)[y - (a_0 + a_1x + a_2x^2 + \cdots + a_kx^k)]^2 \, dx \tag{10.23}$$

where $w(x)$ is a suitable weight function.

The necessary conditions for minimum S are

$$\frac{\partial S}{\partial a_0} = \frac{\partial S}{\partial a_1} = \cdots = \frac{\partial S}{\partial a_k} = 0. \tag{10.24}$$

These give the normal equations as

$$\begin{aligned}
-2\int_a^b w(x)[y-(a_0+a_1x+a_2x^2+\cdots+a_kx^k)]\,dx &= 0\\
-2\int_a^b w(x)[y-(a_0+a_1x+a_2x^2+\cdots+a_kx^k)]x\,dx &= 0\\
-2\int_a^b w(x)[y-(a_0+a_1x+a_2x^2+\cdots+a_kx^k)]x^2\,dx &= 0\\
\vdots \qquad\qquad & \quad \vdots\\
-2\int_a^b w(x)[y-(a_0+a_1x+a_2x^2+\cdots+a_kx^k)]x^k\,dx &= 0.
\end{aligned}$$

After simplification these equations reduce to

$$\begin{aligned}
a_0\int_a^b w(x)dx + a_1\int_a^b xw(x)\,dx + \cdots + a_k\int_a^b x^kw(x)\,dx &= \int_a^b w(x)y\,dx\\
a_0\int_a^b xw(x)dx + a_1\int_a^b x^2w(x)\,dx + \cdots + a_k\int_a^b x^{k+1}w(x)\,dx &= \int_a^b w(x)xy\,dx\\
a_0\int_a^b x^2w(x)dx + a_1\int_a^b x^3w(x)\,dx + \cdots + a_k\int_a^b x^{k+2}w(x)\,dx &= \int_a^b w(x)x^2y\,dx\\
\vdots \qquad\qquad & \quad \vdots\\
a_0\int_a^b x^kw(x)dx + a_1\int_a^b x^{k+1}w(x)\,dx + \cdots + a_k\int_a^b x^{2k}w(x)\,dx &= \int_a^b w(x)x^ky\,dx
\end{aligned} \tag{10.25}$$

Since $w(x)$ and $y = f(x)$ are known, the above equations form a system of linear equations with $(k+1)$ unknowns $a_0, a_1, \ldots, a_k$. This system of equations possesses a unique solution. If

$$a_0 = a_0^*, a_1 = a_1^*, \ldots, a_k = a_k^*$$

is the solution for $a_0, a_1, \ldots, a_k$ then the approximate polynomial is

$$y = a_0^* + a_1^*x + a_2^*x^2 + \cdots + a_k^*x^k.$$

Example 10.7.1 Construct a least squares quadratic approximation to the function

$$y = e^x$$

on $[0, 1]$.

Solution. Let the weight function be $w(x) = 1$ and $y = a_0 + a_1x + a_2x^2$ be the required quadratic approximation. Then the normal equations are

$$\begin{aligned}
a_0 \int_0^1 dx + a_1 \int_0^1 x\,dx + a_2 \int_0^1 x^2\,dx &= \int_0^1 e^x\,dx\\
a_0 \int_0^1 x\,dx + a_1 \int_0^1 x^2\,dx + a_2 \int_0^1 x^3\,dx &= \int_0^1 xe^x\,dx\\
a_0 \int_0^1 x^2\,dx + a_1 \int_0^1 x^3\,dx + a_2 \int_0^1 x^4\,dx &= \int_0^1 x^2e^x\,dx.
\end{aligned}$$

After simplification these equations reduce to

$$\begin{aligned}
a_0 + \frac{1}{2}a_1 + \frac{1}{3}a_2 &= e - 1,\\
\frac{1}{2}a_0 + \frac{1}{3}a_1 + \frac{1}{4}a_2 &= 1,\\
\frac{1}{3}a_0 + \frac{1}{4}a_1 + \frac{1}{5}a_2 &= e - 2.
\end{aligned}$$

The solution of these system of equations is

$$a_0 = 1.0129913, a_1 = 0.8511251, a_2 = 0.8391840.$$

Thus the least squares approximation to $y = e^x$ is

$$y_l = 1.0129913 + 0.8511251x + 0.8391840x^2. \tag{10.26}$$

It is well known that the Taylor's series expansion of $y = e^x$ up to second and third degree terms are

$$y_2 = 1 + x + \frac{x^2}{2} \tag{10.27}$$

$$y_3 = 1 + x + \frac{x^2}{2} + \frac{x^3}{6}. \tag{10.28}$$

The values of y obtained from (10.26), (10.27) and (10.28) are listed below.

x	y_l	y_2	y_3	Exact
0.0	1.01299	1.00000	1.00000	1.00000
0.1	1.10650	1.10500	1.10517	1.10517
0.2	1.21678	1.22000	1.22133	1.22140
0.3	1.34386	1.34500	1.34950	1.34986
0.4	1.48771	1.48000	1.49067	1.49182
0.5	1.64835	1.62500	1.64583	1.64872
0.6	1.82577	1.78000	1.81600	1.82212
0.7	2.01998	1.94500	2.00217	2.01375
0.8	2.23097	2.12000	2.20533	2.22554
0.9	2.45874	2.30500	2.42650	2.45960
1.0	2.70330	2.50000	2.66667	2.71828

This table shows that the least squares quadratic approximation gives more better result as compared to second degree Taylor's series approximation, even to third degree approximation.

10.8 Approximation Using Orthogonal Polynomials

The least squares methods discussed in previous sections, generate a system of $(k+1)$ linear equations. This system can be solved by using any method presented in Chapter 5. But, there is a problem to solve a large system of equations, specially when the system is ill-conditioned. These drawbacks can be removed by using the orthogonal polynomials.

In the previous section, a function is approximated as a polynomial containing the terms 1, $x, x^2, \dots, x^k$. These terms are called **base functions**, because, any function or even discrete data are approximated based on these functions.

Now, we assume that the base functions are some orthogonal polynomials $f_0(x)$, $f_1(x)$, …, $f_k(x)$. Let the given function be approximated as

$$y = a_0 f_0(x) + a_1 f_1(x) + \cdots + a_k f_k(x), \tag{10.29}$$

where $f_i(x)$ is a polynomial in x of degree i. Then as in the previous cases, the residue is given by

$$S = \int_a^b w(x)[y - \{a_0 f_0(x) + a_1 f_1(x) + \cdots + a_k f_k(x)\}]^2 \, dx. \tag{10.30}$$

For minimum S,

$$\frac{\partial S}{\partial a_0} = 0, \frac{\partial S}{\partial a_1} = 0, \dots, \frac{\partial S}{\partial a_k} = 0.$$

These equations give the following normal equations.

$$
\begin{aligned}
-2\int_a^b w(x)[y-\{a_0f_0(x)+a_1f_1(x)+\cdots+a_kf_k(x)\}]f_0(x)\,dx &= 0\\
-2\int_a^b w(x)[y-\{a_0f_0(x)+a_1f_1(x)+\cdots+a_kf_k(x)\}]f_1(x)\,dx &= 0\\
\vdots\qquad\qquad\qquad & \vdots\\
-2\int_a^b w(x)[y-\{a_0f_0(x)+a_1f_1(x)+\cdots+a_kf_k(x)\}]f_k(x)\,dx &= 0.
\end{aligned}
$$

After simplification, the ith equation becomes

$$
\begin{aligned}
& a_0\int_a^b w(x)f_0(x)f_i(x)\,dx + a_1\int_a^b w(x)f_1(x)f_i(x)\,dx+\cdots\\
& +a_i\int_a^b w(x)f_i^2(x)\,dx+\cdots+a_k\int_a^b w(x)f_k(x)f_i(x)\,dx \qquad (10.31)\\
= & \int_a^b w(x)\,y\,f_i(x)\,dx,
\end{aligned}
$$

$i = 0, 1, 2, \dots, n$.

A set of polynomial $\{f_0(x), f_1(x), \dots, f_k(x)\}$ is said to be **orthogonal** with respect to the weight function $w(x)$ if

$$
\int_a^b f_i(x)f_j(x)w(x)\,dx = \begin{cases} 0, & \text{if } i\neq j\\ \displaystyle\int_a^b f_i^2(x)w(x)\,dx, & \text{if } i=j. \end{cases} \qquad (10.32)
$$

Incorporating this property, equation (10.31) becomes

$$
a_i\int_a^b w(x)f_i^2(x)\,dx = \int_a^b w(x)yf_i(x)\,dx, \qquad i=0,1,2,\dots,n.
$$

That is,

$$
a_i = \frac{\displaystyle\int_a^b w(x)\,y\,f_i(x)\,dx}{\displaystyle\int_a^b w(x)f_i^2(x)\,dx}, \qquad i=0,1,2,\dots,n. \qquad (10.33)
$$

From this relation one can determine the values of $a_0, a_1, \dots, a_k$ and the least squares approximation is obtained by substituting these values in (10.29). But, the functions $f_0(x), f_1(x), \dots, f_k(x)$ are unknown. Several orthogonal functions are available in literature, some of them are listed in Table 10.1.

Based on the problem, any one of them can be selected to fit a function.

Table 10.1: Some standard orthogonal polynomials.

Name	$f_i(x)$	Interval	$w(x)$
Legendre	$P_n(x)$	$[-1,1]$	1
Leguerre	$L_n(x)$	$[0,\infty)$	e^{-x}
Hermite	$H_n(x)$	$(-\infty,\infty)$	e^{-x^2}
Chebyshev	$T_n(x)$	$[-1,1]$	$(1-x^2)^{-1/2}$

Gram-Schmidt orthogonalization process

Let $f_i(x)$ be a polynomial in x of degree i and $\{f_i(x)\}$ be a given sequence of polynomials. Then the sequence of orthogonal polynomials $\{f_i^*(x)\}$ over the interval $[a,b]$ with respect to the weight function $w(x)$ can be generated by the relation

$$f_i^*(x) = x^i - \sum_{r=0}^{i-1} a_{ir} f_r^*(x), \qquad i = 1, 2, \dots, n, \tag{10.34}$$

for suitable constants a_{ir}, and $f_0^*(x) = 1$.

To find a_{ir}, multiplying (10.34) by $w(x)f_k^*(x), 0 \le k \le i-1$ and integrating over $[a,b]$. Then

$$\int_a^b f_i^*(x)f_k^*(x)w(x)\,dx = \int_a^b x^i f_k^*(x)w(x)\,dx - \int_a^b \sum_{r=0}^{i-1} a_{ir} f_r^*(x)f_k^*(x)w(x)\,dx.$$

Using the property of orthogonal polynomial, this equation reduces to

$$\int_a^b x^i f_k^*(x)w(x)\,dx - \int_a^b a_{ik} f^{*2}_k(x)w(x)\,dx = 0$$

$$\text{or,}\quad a_{ik} = \frac{\int_a^b x^i \;\; f_k^*(x) \;\; w(x)\,dx}{\int_a^b f_k^{*2}(x) \;\; w(x)\,dx}, \qquad 0 \le k \le i-1.$$

Thus the set of orthogonal polynomials $\{f_i^*(x)\}$ are given by

$$\begin{aligned} f_0^*(x) &= 1 \\ f_i^*(x) &= x^i - \sum_{r=0}^{i-1} a_{ir} f_r^*(x), \qquad i = 1, 2, \dots, n \\ \text{where } a_{ir} &= \frac{\int_a^b x^i \;\; f_k^*(x) \;\; w(x)\,dx}{\int_a^b f_k^{*2}(x) \;\; w(x)\,dx}. \end{aligned} \tag{10.35}$$

For the discrete data, the integral is replaced by summation.

Note 10.8.1 It may be noted that Gram-Schmidt process generates a sequence of monic (leading coefficient unity) orthogonal polynomials.

Example 10.8.1 Use Gram-Schmidt orthogonalization process to determine the first four orthogonal polynomials on $[-1, 1]$ with respect to the weight function $w(x) = 1$.

Solution. Let $f_0^*(x) = 1$.

Then $f_1^*(x) = x - a_{10}f_0^*(x)$, where $a_{10} = \dfrac{\int_{-1}^{1} x\, dx}{\int_{-1}^{1} dx} = 0$.

Thus $f_1^*(x) = x$.
The second orthogonal polynomial is
$f_2^*(x) = x^2 - a_{20}f_0^*(x) - a_{21}f_1^*(x)$

$$\text{where} \qquad a_{20} = \frac{\int_{-1}^{1} x^2\, dx}{\int_{-1}^{1} dx} = \frac{1}{3}, \qquad a_{21} = \frac{\int_{-1}^{1} x^2.x\, dx}{\int_{-1}^{1} x^2\, dx} = 0.$$

Thus $f_2^*(x) = x^2 - \dfrac{1}{3} = \dfrac{1}{3}(3x^2 - 1)$.
Again, $f_3^*(x) = x^3 - a_{30}f_0^*(x) - a_{31}f_1^*(x) - a_{32}f_2^*(x)$
where

$$a_{30} = \frac{\int_{-1}^{1} x^3\, dx}{\int_{-1}^{1} dx} = 0, \quad a_{31} = \frac{\int_{-1}^{1} x^3.x\, dx}{\int_{-1}^{1} x^2\, dx} = \frac{3}{5}, \quad a_{32} = \frac{\int_{-1}^{1} x^3.\frac{1}{3}(3x^2 - 1)\, dx}{\int_{-1}^{1} \frac{1}{9}(3x^2 - 1)^2\, dx} = 0.$$

Hence $f_3^*(x) = x^3 - \frac{3}{5}x = \frac{1}{5}(5x^3 - 3x)$.
Thus the first three orthogonal polynomials are $f_0^*(x) = 1, f_1^*(x) = x$,
$f_2^*(x) = \frac{1}{3}(3x^2 - 1)$ and $f_3^*(x) = \frac{1}{5}(5x^3 - 3x)$.
These polynomials are called (monic) Legendre polynomials.

10.9 Approximation of Functions

The problem of approximation of functions is an important problem in numerical analysis, due to its wide applications in science and engineering, specially in computer graphics, to plot two- and three-dimensional figures. Under some restriction, almost all functions can be approximated using Taylor's series with base functions $1, x, x^2, \ldots$. It is mentioned earlier that the method of approximation of a function using Taylor's series is not economic. But, the approximation using orthogonal polynomials is economic. Among several orthogonal polynomials, Chebyshev polynomials seem to be more economic.

10.9.1 Chebyshev polynomials

The Chebyshev polynomial, $T_n(x)$, of degree n over the interval $[-1,1]$ is defined by

$$T_n(x) = \cos(n\cos^{-1} x), \qquad n = 0, 1, 2, \ldots. \tag{10.36}$$

This expression can also be written as

$$T_n(x) = \cos n\theta, \qquad \text{where} \qquad x = \cos\theta. \tag{10.37}$$

Thus $T_0(x) = 1$ and $T_1(x) = x$.

From equation (10.36), it is easy to observe that $T_n(x) = T_{-n}(x)$.

Also, $T_{2n}(x) = T_{2n}(-x)$ and $T_{2n+1}(-x) = -T_{2n+1}(x)$, i.e., $T_n(x)$ is even or odd functions according as n is even or odd.

Now, from the trigonometric formula

$$\cos(n-1)\theta + \cos(n+1)\theta = 2\cos n\theta . \cos\theta,$$

the recurrence relation for Chebyshev polynomial is obtained as

$$\begin{aligned} &T_{n-1}(x) + T_{n+1}(x) = 2xT_n(x), \\ \text{i.e.,}\quad &T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \qquad n = 1, 2, 3, \ldots. \end{aligned} \tag{10.38}$$

The zeros of Chebyshev polynomial $T_n(x)$ lie between $[-1,1]$, being all distinct and given by

$$x_i = \cos\left(\frac{(2i+1)\pi}{2n}\right), \qquad i = 0, 1, 2, \ldots, n-1. \tag{10.39}$$

These values are called the **Chebyshev abscissas** or **nodes**.

From the equation (10.37), it is obvious that $|T_n(x)| \le 1$ for $-1 \le x \le 1$. Thus extreme values of $T_n(x)$ are -1 and 1.

From the recurrence relation (10.38), it is observed that the relation doubles the leading coefficient of $T_n(x)$ to get the leading coefficient of $T_{n+1}(x)$. Also $T_1(x) = x$. Thus the coefficient of x^n in $T_n(x)$ is $2^{n-1}, n \ge 1$.

This polynomial satisfies a second order differential equation

$$(1-x^2)\frac{d^2y}{dx^2} - x\frac{dy}{dx} + n^2 y = 0. \tag{10.40}$$

To justify it, let $y = T_n(x) = \cos n\theta, x = \cos\theta$.

$$\text{Then } \frac{dy}{dx} = \frac{n\sin n\theta}{\sin\theta} \text{ and } \frac{d^2y}{dx^2} = \frac{-n^2\cos n\theta + n\sin n\theta\cot\theta}{\sin^2\theta} = \frac{-n^2y + x\dfrac{dy}{dx}}{1-x^2}.$$

This gives $(1-x^2)\dfrac{d^2y}{dx^2} - x\dfrac{dy}{dx} + n^2y = 0$.

Orthogonal property

We mentioned earlier that the Chebyshev polynomials are orthogonal and they are orthogonal with respect to the weight function $(1-x^2)^{-1/2}$, i.e.,

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}\,dx = 0 \qquad \text{for } m \neq n.$$

To prove it, let $x = \cos\theta$. Then

$$\begin{aligned}
I &= \int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}\,dx = \int_{0}^{\pi} T_n(\cos\theta)T_m(\cos\theta)\,d\theta \\
&= \int_{0}^{\pi} \cos n\theta\,\cos m\theta\,d\theta = \frac{1}{2}\int_{0}^{\pi}[\cos(m+n)\theta + \cos(m-n)\theta]\,d\theta \\
&= \frac{1}{2}\left[\frac{\sin(m+n)\theta}{m+n} + \frac{\sin(m-n)\theta}{m-n}\right]_0^{\pi}.
\end{aligned}$$

Now, when $m \neq n$ it gives $I = 0$, when $m = n = 0$ then $I = \pi$ and when $m = n \neq 0$ then $I = \frac{\pi}{2}$.

Thus

$$\int_{-1}^{1} \frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}\,dx = \begin{cases} 0, & \text{if } m \neq n \\ \pi, & \text{if } m = n = 0 \\ \pi/2, & \text{if } m = n \neq 0 \end{cases} \tag{10.41}$$

Example 10.9.1 Use Chebyshev polynomials to find least squares approximation of second degree for $f(x) = \sqrt{1-x^2}$ on $[-1,1]$.

Solution. Let $f(x) \simeq a_0T_0(x) + a_1T_1(x) + a_2T_2(x)$. Then the residue is

$$S = \int_{-1}^{1} w(x)[f(x) - \{a_0T_0(x) + a_1T_1(x) + a_2T_2(x)\}]^2\,dx,$$

where $w(x) = (1-x^2)^{-1/2}$.
For minimum S,

$$\frac{\partial S}{\partial a_0} = 0, \qquad \frac{\partial S}{\partial a_1} = 0, \qquad \frac{\partial S}{\partial a_2} = 0.$$

Now,

$$\frac{\partial S}{\partial a_0} = -2\int_{-1}^{1} w(x)[f(x) - \{a_0T_0(x) + a_1T_1(x) + a_2T_2(x)\}]T_0(x)\,dx = 0.$$

Using the property of orthogonal polynomials, this equation is simplified as

$$\int_{-1}^{1} w(x)f(x)T_0(x)\,dx - \int_{-1}^{1} w(x)a_0T_0^2(x)\,dx = 0.$$

This equation gives

$$a_0 = \frac{\int_{-1}^{1} w(x)f(x)T_0(x)\,dx}{\int_{-1}^{1} w(x)T_0^2(x)\,dx} = \frac{1}{\pi}\int_{-1}^{1} \frac{\sqrt{1-x^2}}{\sqrt{1-x^2}}\,dx = \frac{2}{\pi}$$

[Using (10.41) and $T_0(x) = 1$].

Similarly, from the relations $\dfrac{\partial S}{\partial a_1} = 0$ and $\dfrac{\partial S}{\partial a_2} = 0$, we obtain

$$a_1 = \frac{\int_{-1}^{1} w(x)f(x)T_1(x)\,dx}{\int_{-1}^{1} w(x)T_1^2(x)\,dx} = \frac{2}{\pi}\int_{-1}^{1} x\,dx = 0$$

$$a_2 = \frac{\int_{-1}^{1} w(x)f(x)T_2(x)\,dx}{\int_{-1}^{1} w(x)T_2^2(x)\,dx} = \frac{2}{\pi}\int_{-1}^{1} (2x^2-1)\,dx = -\frac{4}{3\pi}.$$

Thus the Chebyshev approximation is

$$f(x) = \frac{2}{\pi}T_0(x) - \frac{4}{3\pi}T_2(x) = \frac{2}{\pi} - \frac{4}{3\pi}(2x^2-1) = 1.061033 - 0.8488264x^2.$$

Using the recurrence relation (10.38) and $T_0(x) = 1, T_1(x) = x$, one can generate Chebyshev polynomials. The first seven Chebyshev polynomials are

$$\begin{aligned}
T_0(x) &= 1\\
T_1(x) &= x\\
T_2(x) &= 2x^2 - 1\\
T_3(x) &= 4x^3 - 3x\\
T_4(x) &= 8x^4 - 8x^2 + 1\\
T_5(x) &= 16x^5 - 20x^3 + 5x\\
T_6(x) &= 32x^6 - 48x^4 + 18x^2 - 1\\
T_7(x) &= 64x^7 - 112x^5 + 56x^3 - 7x.
\end{aligned} \tag{10.42}$$

The graph of first four Chebyshev polynomials are shown in Figure 10.1.

The equation (10.42) suggests that all powers of x can be expressed in terms of Chebyshev polynomials, as

$$\begin{aligned}
1 &= T_0(x)\\
x &= T_1(x)\\
x^2 &= \tfrac{1}{2}[T_0(x) + T_2(x)]\\
x^3 &= \tfrac{1}{4}[3T_1(x) + T_3(x)]\\
x^4 &= \tfrac{1}{8}[3T_0(x) + 4T_2(x) + T_4(x)]\\
x^5 &= \tfrac{1}{16}[10T_1(x) + 5T_3(x) + T_5(x)]\\
x^6 &= \tfrac{1}{32}[10T_0(x) + 15T_2(x) + 6T_4(x) + T_6(x)]\\
x^7 &= \tfrac{1}{64}[35T_1(x) + 21T_3(x) + 7T_5(x) + T_7(x)].
\end{aligned} \tag{10.43}$$

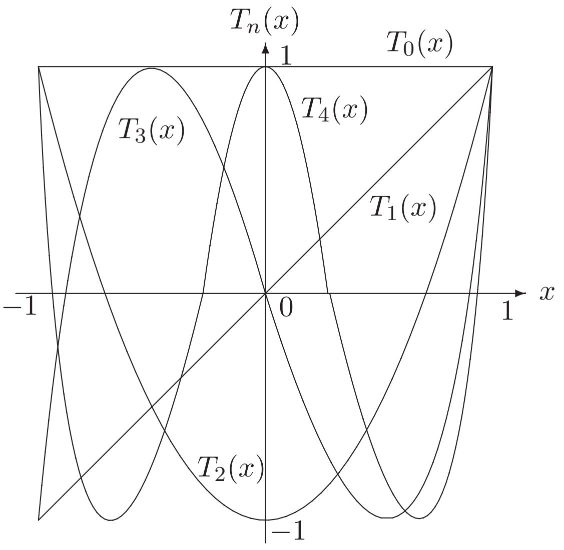


Figure 10.1: Chebyshev polynomials $T_n(x), n = 0, 1, 2, 3, 4$.

Thus every polynomial can be approximated using Chebyshev polynomials.

Example 10.9.2 Express the polynomial $x^3 + 2x^2 - 7$ in terms of Chebyshev polynomials.

Solution.
$$\begin{aligned} x^3 + 2x^2 - 7 &= \tfrac{1}{4}[3T_1(x) + T_3(x)] + 2.\tfrac{1}{2}[T_0(x) + T_2(x)] - 7T_0(x) \\ &= \tfrac{1}{4}T_3(x) + T_2(x) + \tfrac{3}{4}T_1(x) - 6T_0(x). \end{aligned}$$

10.9.2 Expansion of function using Chebyshev polynomials

Let $y = f(x)$ be a function to be approximated by Chebyshev polynomials. This can be done as $f(x) = a_0T_0(x) + a_1T_1(x) + a_2T_2(x) + \cdots + a_kT_k(x)$, where the coefficients a_i are given by

$$a_i = \frac{\displaystyle\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}}\, yT_i(x)\, dx}{\displaystyle\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}}\, T_i^2(x)\, dx}, \qquad i = 0, 1, 2, \ldots, n.$$

But, the integral in the denominator of a_i is improper and it is not easy to evaluate. So, a discretization technique is adopted here to approximate a function using Chebyshev polynomials. The orthogonal property for discrete case is stated below.

$$\sum_{k=0}^{n} T_i(x_k)T_j(x_k) = \begin{cases} 0, & \text{if } i \neq j \\ \dfrac{n+1}{2}, & \text{if } i = j \neq 0 \\ n+1, & \text{if } i = j = 0 \end{cases} \tag{10.44}$$

where $x_k = \cos\left(\dfrac{(2k+1)\pi}{2n+2}\right), \qquad k = 0, 1, 2, \ldots, n.$

This result is used to establish the following theorem.

Theorem 10.1 (Chebyshev approximation). *The function $f(x)$ can be approximated over $[-1, 1]$ by Chebyshev polynomials as*

$$f(x) \simeq \sum_{i=0}^{n} a_i T_i(x). \tag{10.45}$$

The coefficients a_i are given by

$$a_0 = \frac{1}{n+1}\sum_{j=0}^{n} f(x_j)T_0(x_j) = \frac{1}{n+1}\sum_{j=0}^{n} f(x_j), \tag{10.46}$$

$$x_j = \cos\left(\frac{(2j+1)\pi}{2n+2}\right)$$

$$\textit{and } a_i = \frac{2}{n+1}\sum_{j=0}^{n} f(x_j)T_i(x_j)$$

$$= \frac{2}{n+1}\sum_{j=0}^{n} f(x_j)\cos\left(\frac{(2j+1)i\pi}{2n+2}\right) \tag{10.47}$$

for $i = 1, 2, \ldots, n$.

Example 10.9.3 Approximate $f(x) = e^x$ to second order Chebyshev approximation over the interval [0, 1].

Solution. The second order Chebyshev approximation is

$$f(x) \simeq \sum_{i=0}^{2} a_i T_i(x).$$

The Chebyshev nodes are given by

$$x_j = \cos\left(\frac{(2j+1)\pi}{6}\right), j = 0, 1, 2.$$

$x_0 = 0.86660254, x_1 = 0, x_2 = -0.8660254.$

$$a_0 = \frac{1}{3}[f(x_0) + f(x_1) + f(x_2)] = 1.2660209$$

$$a_1 = \frac{2}{3}\sum_{j=0}^{2} f(x_j)\cos\left(\frac{(2j+1)\pi}{6}\right) = 1.1297721$$

$$a_2 = \frac{2}{3}\sum_{j=0}^{2} f(x_j)\cos\left(\frac{2(2j+1)\pi}{6}\right) = 0.26602093.$$

Therefore, $f(x) \simeq 1.2660209\, T_0(x) + 1.1297721\, T_1(x) + 0.2660209\, T_2(x)$.

Algorithm 10.2 (Chebyshev approximation). This algorithm approximates and evaluates a function $f(x)$ by Chebyshev polynomials as

$f(x) \simeq \sum_{i=0}^{n} a_i T_i(x)$ where $a_0 = \frac{1}{n+1}\sum_{j=0}^{n} f(x_j)$ and

$a_i = \frac{2}{n+1}\sum_{j=0}^{n} f(x_j)\cos\left(\frac{(2j+1)i\pi}{2n+2}\right), i = 1, 2, \dots, n.$

Algorithm Chebyshev

```
Input function $F(x)$
 Step 1.  Read $n$; //degree of the polynomial//
 Step 2.  Set $Pi = 3.1415926535$ and $A = Pi/(2n+2)$.
 Step 3.  //Calculation of nodes and function values//
          for $i = 0$ to $n$ do
              $x_i = \cos((2i+1)A)$ and $y_i = F(x_i)$;
          endfor;
 Step 4.  //Calculation of the coefficients $a_i$.//
          for $i = 0$ to $n$ do
              Set $a_i = 0$;
              for $j = 0$ to $n$ do
                  $a_i = a_i + \cos((2j+1) * i * A) * y_j$;
              endfor;
          endfor;
          $a_0 = a_0/(n+1)$;
          for $i = 1$ to $n$ do
              $a_i = 2a_i/(n+1)$;
          endfor;
          //Evaluation of Chebyshev polynomial approximation//
 Step 5.  Read $x$; //The point at which the function is to be evaluated//
 Step 6.  Set $T_0 = 1, T_1 = x$;
 Step 7.  //Evaluation of Chebyshev polynomial//
          if $n > 1$ then
              for $i = 1$ to $n - 1$ do
                  $T_{i+1} = 2xT_i - T_{i-1}$;
              endfor;
          endif;
 Step 8.  $sum = a_0 * T_0$;
          for $i = 1$ to $n$ do
              $sum = sum + a_i * T_i$;
          endfor;
 Step 9.  Print 'The value of Chebyshev polynomial approximation is', $sum$;
end Chebyshev
```

Program 10.2

```
/* Program Chebyshev Approximation
   This program approximates and evaluates a function f(x)
   by Chebyshev polynomials. */
#include <stdio.h>
#include <math.h>
#define f(x) exp(x) /* definition of function f(x) */
#define pi 3.1415926535
void main()
{
 int n,i,j; float a,xi,x[40],y[40],c[40];
 float xg,T[40],sum;
 printf("Enter the degree of the polynomial ");
 scanf("%d",&n);
 a=pi/(2.0*n+2.0);
 for(i=0;i<=n;i++)
   {
     x[i]=cos((2*i+1.)*a);
     xi=x[i];
     y[i]=f(xi);
   }
 for(i=0;i<=n;i++)
   {
     c[i]=0;
     for(j=0;j<=n;j++)
         c[i]=c[i]+cos((2*j+1)*i*a)*y[j];
   }
 c[0]=c[0]/(n+1.);
 for(i=1;i<=n;i++)
    c[i]=2*c[i]/(n+1.);
/* Printing of Chebyshev coefficients*/
 printf("The Chebyshev coefficients are\n ");
 for(i=0;i<=n;i++) printf("%f  ",c[i]);
/* Evaluation of the polynomial */
 printf("\nEnter the value of x ");
 scanf("%f",&xg);
 T[0]=1.0; T[1]=xg;
 /*Computation of Chebyshev polynomial at x*/
 if(n>1)
    for(i=1;i<=n-1;i++) T[i+1]=2*xg*T[i]-T[i-1];
 sum=c[0]*T[0];
 for(i=1;i<=n;i++)
   sum+=c[i]*T[i];
 printf("\nThe value of the Chebyshev polynomial approximation
          at %6.3f is %9.8f",xg,sum);
} /* main */
```

A sample of input/output:

```
Enter the degree of the polynomial 4
The Chebyshev coefficients are
 1.266066  1.130318  0.271495  0.044334  0.005429
Enter the value of x 0.5
The value of the Chebyshev polynomial approximation at 0.500
 is 1.64842904
```

Minimax principle

It is well known that the error in polynomial interpolation is

$$E_n(x) = w_{n+1}(x)\frac{f^{(n+1)}(\xi)}{(n+1)!},$$

where $w_{n+1}(x) = (x-x_0)(x-x_1)\cdots(x-x_n)$ is a polynomial of degree $(n+1)$.

Now,

$$|E_n(x)| \leq \max_{-1\leq x\leq 1}|w_{n+1}(x)|\frac{\max\limits_{-1\leq x\leq 1}|f^{(n+1)}(\xi)|}{(n+1)!}. \tag{10.48}$$

The expression $\max\limits_{-1\le x\le 1} |f^{(n+1)}(\xi)|$ is fixed for a given function $f(x)$. Thus the error bound depends on the value of $|w_{n+1}(x)|$ and the value of $|w_{n+1}(x)|$ depends on the choice of the nodes $x_0, x_1, \ldots, x_n$. The maximum error depends on the product of $\max\limits_{-1\le x\le 1} |w_{n+1}(x)|$ and $\max\limits_{-1\le x\le 1} |f^{(n+1)}(\xi)|$. Russian mathematician Chebyshev invented that $x_0, x_1, \ldots, x_n$ should be chosen such that $w_{n+1}(x) = 2^{-n}T_{n+1}(x)$. The polynomial $2^{-n}T_{n+1}(x)$ is the monic Chebyshev polynomial.

If n is fixed, then all possible choices for $w_{n+1}(x)$ and thus among all possible choices for the nodes $x_0, x_1, \ldots, x_n$ on $[-1, 1]$, the polynomial $\tilde{T}_{n+1}(x) = 2^{-n}T_{n+1}(x)$ is the unique choice that satisfies the relation

$$\max_{-1\le x\le 1}\{|\tilde{T}_{n+1}(x)|\} \le \max_{-1\le x\le 1}\{|w_{n+1}(x)|\}.$$

Moreover, $\max\limits_{-1\le x\le 1}\{|\tilde{T}_{n+1}(x)|\} = 2^{-n}$ as $|T_{n+1}(x)| \le 1$.

This property is called **minimax principle** and the polynomial

$$\tilde{T}_{n+1}(x) = 2^{-n}T_{n+1}(x) \qquad \text{or,} \qquad \tilde{T}_n(x) = 2^{1-n}T_n(x)$$

is called **minimax polynomial**.

10.9.3 Economization of power series

From the equation (10.43), it is observed that every polynomial of degree n can be expressed in terms of Chebyshev polynomials of same degree. If

$$f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \tag{10.49}$$

be the given polynomial, then it can be expressed in the form

$$f(x) = b_0 + b_1T_1(x) + b_2T_2(x) + \cdots + b_nT_n(x). \tag{10.50}$$

For a large number of functions, it is computationally observed that the expansion of the form (10.50) converges more rapidly than the form (10.49). Thus representation of the form (10.50) is computationally better and the process is called **economization of the power series**. This is illustrated in the following example.

Example 10.9.4 Economize the power series

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$$

correct to four significant digits.

Solution. Since the result is required to four significant digits and the coefficients of the term x^8, $1/8! = 0.0000248$ can affect the fifth decimal place only, thus the terms after fourth term may be truncated.

Hence

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!}. \tag{10.51}$$

Now, express the above series using Chebyshev polynomials, as

$$\begin{aligned}\cos x &= T_0(x) - \frac{1}{2!}\cdot\frac{1}{2}[T_0(x)+T_2(x)] + \frac{1}{4!}\cdot\frac{1}{8}[3T_0(x)+4T_2(x)+T_4(x)]\\ &\quad -\frac{1}{6!}\cdot\frac{1}{32}[10T_0(x)+15T_2(x)+6T_4(x)+T_6(x)]\\ &= 0.7651910 - 0.2298177\,T_2(x) + 0.0049479\,T_4(x) - 0.0000434\,T_6(x).\end{aligned}$$

Again, the term $0.0000434\ T_6(x)$ does not affect the fourth decimal place, so it is discarded and the economized series is

$$\cos x = 0.7651910 - 0.2298177\,T_2(x) + 0.0049479\,T_4(x).$$

In terms of x, the series is

$$\begin{aligned}\cos x &= 0.7651910 - 0.2298177\,(2x^2-1) + 0.0049479\,(8x^4-8x^2+1)\\ &= 0.9999566 - 0.4992186\,x^2 + 0.0395832\,x^4.\end{aligned}$$

This expression gives the values of $\cos x$ correct up to four decimal places.

10.10 Exercise

1. Fit a straight line for the following data.

x	:	1	3	5	7	9
y	:	10	13	25	23	33

   Also calculate the sum of squares of residuals.

2. Fit a straight line of the form $y = a + bx$ for the following data.

x	:	1951	1961	1971	1981	1991
y	:	33	43	60	78	96

   Find y when $x = 2001$.

3. If the straight line $y = a + bx$ is the best fit curve for the data points $(x_i, y_i), i = 1, 2, \ldots, n$, then show that

$$\begin{vmatrix} x & y & 1\\ \sum x_i & \sum y_i & n\\ \sum x_i^2 & \sum y_i^2 & \sum x_i \end{vmatrix} = 0.$$

4. Use weighted least squares method to fit the straight line $y = a + bx$ to the following data.

x	:	2	4	6	7	9
y	:	6	10	8	15	20
w	:	1	1	5	10	1

5. Fit a parabola of the form $y = a + bx + cx^2$ for the following data.

x	:	0	1	2	3	4	5
y	:	0	4	16	36	64	100

Estimate the residue and conclude about the data. Find the predicted value of y when $x = -1$.

6. Fit a straight line and a second degree parabola for the following data and explain which curve is better fitted for the given data by estimating the residue.

x	:	–1	0	3	4	6
y	:	0	11	22	25	34

7. Fit a curve of the form ae^{-bx} for the data given below.

x	:	0.5	1.0	1.5	1.6	1.8	1.9	2.0	2.25
y	:	2.1	3.4	3.9	4.8	5.1	5.8	6.0	8.9

8. Fit a curve of the type $y = cx^a$ for the following data.

x	:	1	2	3	4	5
y	:	0.6	1.9	4.4	7.8	11.9

9. Fit the curves $y_1 = ce^{ax}$ and $y_2 = 1/(ax + b)$ by using the following data.

x	:	-1	0	1	2	3
y	:	6.62	3.98	2.76	1.25	0.5

Also find the sum of squares of errors in both the cases and conclude which curve is better for this data points.

10. Find the set of equations to find the values of a, b, c when the curve $y = a + \frac{b}{x} + \frac{c}{x^2}$ is to be fitted for the data points $(x_i, y_i), i = 1, 2, \ldots, n$.

11. Find the appropriate transformations to linearize the following curves
(a) $y = \dfrac{a}{x+b}$, (b) $x = \dfrac{b}{y+a}$, (c) $y = \dfrac{1}{a+bx}$,
(d) $y = a \log x + b$, (e) $y = \dfrac{x}{a+bx}$, (f) $y = ce^{-bx}$.

12. Find the equations for the coefficients a and b if the curve $y = a \sinh bx$ is to be fitted to the data points $(x_i, y_i), i = 1, 2, \ldots, n$.

13. By expanding the expression $(\cos\theta + i \sin\theta)^n$ show that the coefficient of x^n in $T_n(x)$ is 2^{n-1}.

14. Show that $T_n(x)$ is a polynomial in x of degree n. Also show that $T_n(x)$ is even or odd functions according as n is even or odd.

15. Show that the Chebyshev polynomials are orthogonal with respect to the weight function $w(x) = (1 - x^2)^{-1/2}$.

16. Use Gram-Schmidt orthogonalization process to find first three orthogonal polynomials whose weight function is $w(x) = e^{-x^2}$.

17. Express the following in terms of Chebyshev polynomials.
(a) $x^4 - 3x^3 + 2x^2 + 10x - 5$, and (b) $x^5 - 6x^3 + 11x^2 - 10x + 8$.

18. Express the following as a polynomials in x.
(a) $T_5(x) + 5T_3(x) - 11T_2(x) + 3T_1(x)$, and
(b) $2T_4(x) - 7T_3(x) + 2T_2(x) + 5T_0(x)$.

19. Approximate the following functions using Chebyshev polynomials.
(a) $\log(1 + x)$, $-1 < x \le 1$, (b) $\sin x$, $-1 \le x \le 1$, (c) e^x.

20. Suppose $y = 1 - \frac{x}{2!} + \frac{x^2}{4!} - \frac{x^3}{6!} + \frac{x^4}{8!} - \cdots$. Economize this series if the fourth decimal place is not to be affected, near $x = 1$.

21. Economize the power series $\sin x = x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \cdots$
on the interval $[-1, 1]$ correct up to four decimal places.

Bibliography

[1] A. Ben Israel and T.N.E. Greville, Generalized Inverses: Theory and Applications, Wiley, New York, 1974.

[2] J.C. Butcher, The Numerical Analysis of Ordinary Differential Equation: Runge-Kutta and General Linear Methods, (Chichester, John Wiley), 1987.

[3] N.I. Danilina, S.N. Dubrovskaya, O.P. Kvasha and G.L. Smirnov, Computational Mathematics, Mir Publishers, Moscow, 1998.

[4] M.E.A. El-Mikkawy, A fast algorithm for evaluating nth order tri-diagonal determinants, J. Computational and Applied Mathematics, 166 (2004) 581-584.

[5] L. Fox, Numerical Solution of Ordinary and Partial Differential Equations, Pergamon, London, 1962.

[6] A. Gupta and S.C. Bose, Introduction to Numerical Analysis, Academic Publishers, Calcutta, 1989.

[7] T.N.E. Greville, The pseudo-inverse of a rectangular or singular matrix and its application to the solution of systems of linear equations, SIAM Review, 1 (1959) 38-43.

[8] F.B. Hildebrand, Introduction of Numerical Analysis, McGraw-Hill, New York, London, 1956.

[9] H.H.H. Homeier, A modified Newton method for root finding with cubic convergence, J. Computational and Applied Mathematics, 157 (2003) 227-230.

[10] A.S. Householder, The Theory of Matrices in Numerical Analysis, Blaisdell, New York, 1964.

[11] M.K. Jain, S.R.K. Iyengar and R.K. Jain, Numerical Methods for Scientific and Engineering Computation, New Age International (P) Limited, New Delhi, 1984.

[12] E.V. Krishnamurthy and S.K. Sen, Numerical Algorithms, Affiliated East-West Press Pvt. Ltd., New Delhi, 1986.

[13] J.H. Mathews, Numeical Methods for Mathematics, Science, and Engineering, 2nd ed., Prentice-Hall, Inc., N.J., U.S.A., 1992.

[14] J.Y. Park, D.J. Evans, K. Murugesan, S. Sekar and V. Murugesh, Optimal control of singular systems using the RK-Butcher algorithm, Inter. J. Computer Mathematics, 81 (2004) 239-249.

[15] G.D. Smith, Numerical Solution of Partial Differential Equations: Finite Difference Methods, Clarendon Press, Oxford, 1985.

[16] E.A. Volkov, Numerical Methods, Mir Publishers, Moscow, 1986.

[17] E.W. Weisstein, Numerical quadrature, From MathWorld – A Wolfram Web Resource.
http://mathworld.wolfram.com

[18] http://www.riskglossary.com/link/monte-carlo-method.htm

[19] http://www.damtp.cam.ac.uk