DON BOSCO COLLEGE

DEPARTMENT OF PHYSICS

STUDY MATERIAL

SUBJECT NAME	**: COMPUTATIONAL PHYSICS & C++ PROGRAMMING**
PAPER CODE	**: 21PPH06**
CLASS	**: I -M.Sc., PHYSICS**
SEMESTER	**: II**

UNIT-I:

SOLUTIONS OF LINEAR AND NONLINEAR EQUATIONS

Simultaneous Linear Equations: Gauss elimination method - Jordan's modification - Gauss-Seidel method. Curve fitting - Method of least squares - Normal equations - Straight line fit - Interpolation - Least squares Approximation - Newton Interpolation polynomials - Linear Interpolation - Gregory-Newton Interpolation polynomials.

Roots of Non-linear Equations: Bisection method - Iteration method - Newton-Raphson method - Termination criteria – Pitfalls - Order of convergence.

UNIT-II:

NUMERICAL INTEGRATION AND DIFFERENTIATION

Numerical Differentiation - Numerical Integration - Trapezoidal rule - Simpson's 1/3 and 3/8 rules - Random number generation - Park and Miller method - Newton-Cotes formulas - Gaussian quadrature formula - Estimation of errors in evaluating the integrals - Roots of Equation.

UNIT-III:

NUMERICAL SOLUTIONS OF ORDINARY DIFFERENTIAL EQUATIONS:

Ordinary Differential equation: Taylor's series method - Euler and Picard methods - Predictor - corrector methods - Chaotic dynamics of a driven pendulum - Boundary-value and eigenvalue problems - The Shooting Method - Linear equations and the Sturm - Liouville problem.

First order equations: Euler and improved Euler methods - Formulas - Second order equation -Euler methods - Solution of Ordinary differential equation by Euler, Runge-Kutta Fourth Order method for solving first order ordinary differential equations.

UNIT- IV

FUNDAMENTALS OF C++ PROGRAMMING

Basic structure of C++ programs - Creating the Source File - compiling and Linking. Tokens, Keywords – Identifiers - Basic Data Types - Symbolic Constants - Type Compatibility - Declarations of Variables - Dynamic Initialization of Variables - Reference Variables - Reading and writing a character - formatted inputs and outputs.

Operators in C++: Arithmetic, relational, logical, assignment, increment, decrement, and conditional, bitwise special operators - Operator Precedence - Type Cast Operator – Expressions and Implicit Conversions - Operator Overloading - C++ math library functions- C++ standard library header files.

UNIT V:

DECISION MAKING, ARRAYS, STRUCTURES, FUNCTIONS & POINTERS

Decision Making Statements: If-else statement - nested if-else, else-if ladder - switch case statement - conditional statement - go to statement - break and continue statement - Nested control statements.

Loops: While loop - do-while loop - For loop - Nested For loop.

Arrays: Defining, initializing arrays - accessing array elements - One/Two dimensional arrays.

Structures: Specifying the structure - accessing structure members. Functions: Function declaration and definition - Calling the Function. Pointers: Address and pointers - Address of operator & pointer variables.

UNIT – I

Solution of Simultaneous Linear Algebraic Equations

Gauss elimination Method (Direct Method)

This is a direct method based on the elimination of the unknowns by combining equations such that the n equations in n unknowns are reduced to an equivalent upper triangular system which could be solved by back substitution.

Consider the n linear equations in n unknowns.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots\ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots\ldots + a_{2n}x_n &= b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots\ldots + a_{nn}x_n &= b_n \end{aligned} \qquad \ldots\ldots (1)$$

where a_{ij} and b_i are known constants and x_i's are unknowns.

The system of equations given in (1) is equivalent to

$$AX = B \qquad \ldots\ldots(2)$$

where

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots\ldots & a_{1n} \\ a_{21} & a_{22} & \ldots\ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots\ldots & a_{nn} \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ .. \\ x_n \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{pmatrix}$$

The augmented coefficient matrix is given by (A,B).

$$(A, B) = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \ldots\ldots & a_{1n} & b_1 \\ a_{21} & a_{22} & \ldots\ldots & a_{2n} & b_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots\ldots & a_{nn} & b_n \end{array}\right) \qquad \ldots\ldots(3)$$

One has to reduce the augmented matrix (A, B) given in (3) in to an upper triangular matrix Considering a_{11} as the pivot element multiply the first row of (3) by $(-a_{i1}/a_{11})$ and add to the i^{th} row of (A,B), where $i = 2,3, \dots n$ so that all elements in the first column of (A, B) except a_{11} are made zero.

Now matrix (A,B) will be of the form

$$\left(\begin{array}{l|c} a_{11} \; a_{12} \; \dots\dots \; a_{1n} & b_1 \\ 0 \quad b_{22} \; \dots\dots \; b_{2n} & c_2 \\ \dots\dots\dots\dots\dots\dots & \dots \\ 0 \quad b_{n2} \; \dots\dots \; b_{nn} & c_n \end{array}\right) \qquad \dots\dots(4)$$

Considering b_{22} as the pivot, we have to make all elements below b_{22} in the second column of (4) as zero. This is achieved by multiplying second row of (4) by $-b_{i2}/b_{22}$ and add to the corresponding elements of the i^{th} row ($i= 3, 4, \dots n$). Now all the elements below b_{22} are reduced to zero. Now the augmented matrix in (4) has the elements as given below.

$$\left(\begin{array}{ccccc|c} a_{11} & a_{12} & a_{13} & \dots\dots & a_{1n} & b_1 \\ 0 & b_{22} & b_{23} & \dots\dots & b_{2n} & c_2 \\ 0 & 0 & c_{33} & \dots\dots & c_{3n} & d_3 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & c_{n3} & \dots\dots & c_{nn} & k_n \end{array}\right) \qquad \dots\dots(5)$$

Continuing the process, all elements below the leading diagonal elements of A are made to zero. Repeating the procedure of making the lower diagonal elements to zero for all the columns, we find that the augmented coefficient matrix (A,B) is converted into an upper triangular matrix as shown below.

$$\left(\begin{array}{cccccc|c} a_{11} & a_{12} & a_{13} & a_{14} & \dots\dots & a_{1n} & b_1 \\ 0 & b_{22} & b_{23} & b_{24} & \dots\dots & b_{2n} & c_2 \\ 0 & 0 & c_{33} & c_{34} & \dots\dots & c_{3n} & d_3 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots\dots & \alpha_{nn} & k_n \end{array}\right) \qquad \dots\dots(6)$$

From (6), the given system of linear equations is equivalent to

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots\ldots + a_{1n}x_n &= b_1 \\
b_{22}x_2 + b_{23}x_3 + \ldots\ldots + b_{2n}x_n &= c_2 \\
c_{33}x_3 + \ldots\ldots + c_{3n}x_n &= d_3 \\
\ldots\ldots\ldots\ldots\ldots\ldots & \\
\alpha_{nn}\, x_n &= k_n
\end{aligned}$$

Going from the bottom of these equations, we solve for $x_n = k_n / \alpha_{nn}$. Using this in the penultimate equation, we get x_{n-1} and so on. By this back substitution method, we solve for

$x_n, x_{n-1}, x_{n-2}, \ldots\ldots\ldots\ldots x_2, x_1$

Gauss-Jordan elimination method (Direct Method)

In this method, the coefficient matrix A of the system AX= B is converted into a diagonal matrix by making all the off diagonal elements to zero by similarity transformations. Now the system (A,B) will be reduced to the form

$$\left(\begin{array}{ccccc|c}
a_{11} & 0 & 0 & \ldots & 0 & b_1 \\
0 & b_{22} & 0 & \ldots & 0 & c_2 \\
0 & 0 & c_{33} & \ldots & 0 & d_3 \\
\ldots & \ldots & \ldots & \ldots & \ldots & . \\
0 & 0 & 0 & \ldots & \alpha_{nn} & k_n
\end{array}\right) \qquad \ldots\ldots(7)$$

From (7) we get

$x_n = k_n / \alpha_{nn}, \ldots\ldots\ x_2 = c_2 / b_{22},\ x_1 = b_1 / a_{11}$

Problems

1. **Solve the system of equations by (i) Gauss elimination method and by (ii) Gauss-Jordan method.**

   X+2y+z=3, 2x+3y+3z=10, 3x-y+2z=13

Gauss elimination method:

The set of equations are given in matrix form as

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 3 & 3 \\ 3 & -1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ Z \end{pmatrix} = \begin{pmatrix} 3 \\ 10 \\ 13 \end{pmatrix}$$

$$A \quad X = B$$

The augmented coefficient matrix (A,B) for the given system of equations is given as

$$(A,B) = \left(\begin{array}{ccc|c} 1 & 2 & 1 & 3 \\ 2 & 3 & 3 & 10 \\ 3 & -1 & 3 & 13 \end{array}\right)$$

By making the transformations given by the side of the corresponding row of the matrix (A,B), we get

$$(A,B) = \left(\begin{array}{ccc|c} 1 & 2 & 1 & 3 \\ 0 & -1 & 1 & 4 \\ 0 & -7 & -1 & 4 \end{array}\right) \begin{array}{l} \\ R_2=R_2 + (-2)R_1 \\ R_3=R_3 + (-3)R_1 \end{array}$$

Now take $b_{22} = -1$ as the pivot and make b_{32} as zero

$$(A,B) = \left(\begin{array}{ccc|c} 1 & 2 & 1 & 3 \\ 0 & -1 & 1 & 4 \\ 0 & 0 & -8 & -24 \end{array}\right) \begin{array}{l} \\ \\ R_3 = R_3 + (-7)\, R_2 \end{array}$$

From this, we get

$$\begin{aligned} x+ 2y+ z &= 3 \\ -y + z &= 4 \\ -8z &= -24 \end{aligned}$$

Solving the above equations by back substitution we

get $z = 3$, $y = -1$ and $x = 2$

Gauss – Jordan method:

The upper triangular matrix (A,B) is

$$(A,B) = \left(\begin{array}{ccc|c} 1 & 2 & 1 & 3 \\ 0 & -1 & 1 & 4 \\ 0 & 0 & -8 & -24 \end{array}\right)$$

The off diagonal elements of (A,B) are made to zero by the following transformations given below.

$$(A,B) = \left(\begin{array}{ccc|c} 1 & 0 & 3 & 11 \\ 0 & -1 & 1 & 4 \\ 0 & 0 & -1 & -3 \end{array}\right) \qquad \begin{array}{l} R_1 = R_1 + 2\,R_2 \\ \\ R_3=R_3*(1/8) \end{array}$$

$$= \left(\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & -3 \end{array}\right) \qquad \begin{array}{l} R_1 = R_1 + 3\,R_3 \\ R_2 = R_2 + R_3 \\ \end{array}$$

Therefore we get , x = 2, y = **-**I, and z= 3. By substituting these values in the given equations, we find that the values satisfy the equations.

2. Solve the system of equations given below by Gauss elimination method.
$2x + 3y - z = 5$, $4x + 4y - 3z = 3$ and $2x - 3y + 2z = 2$

The set of equations are given in matrix form as

$$\begin{pmatrix} 2 & 3 & -1 \\ 4 & 4 & -3 \\ 2 & -3 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ Z \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ 2 \end{pmatrix}$$

A X = B

The augmented coefficient matrix (A,B) is

$$(A,B) = \left(\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 4 & 4 & -3 & 3 \\ 2 & -3 & 2 & 2 \end{array}\right)$$

Taking $a_{11} = 2$ as the pivot, reduce all elements below in the first column to zero

$$(A,B) = \left(\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & -6 & 3 & -3 \end{array}\right) \quad \begin{array}{l} \\ R_2=R_2+(-2)*R_1 \\ R_3=R_3+(-1)*R_1 \end{array}$$

Taking the element -2 in the position (2,2) as pivot, reduce the element below that to zero, we get

$$(A,B) = \left(\begin{array}{ccc|c} 2 & 3 & -1 & 5 \\ 0 & -2 & -1 & -7 \\ 0 & 0 & 6 & 18 \end{array}\right) \quad \begin{array}{l} \\ \\ R_3=R_3+(-3)*R_2 \end{array}$$

Hence $2x + 3y - z = 5$

$-2y - z = -7$

$6z = 18$

Then by back substitution we get $z = 3$, $y = 2$ and $x = 1$.

These values are substituted in the given equations and are found to satisfy them.

3. Solve the system of equations by Gauss – elimination method.

$10x - 2y + 3z = 23$

$2x + 10y - 5z = -33$

$3x - 4y + 10z = 41$

The given system of equations are written in augmented matrix form as

$$\begin{pmatrix} 10 & -2 & 3 & 23 \\ 2 & 10 & -5 & -33 \\ 3 & -4 & 10 & 41 \end{pmatrix}$$

Using the transformations given by the side, we can write

$$\begin{pmatrix} 1 & -1/5 & 3/10 & 23/10 \\ 2 & 10 & -5 & -33 \\ 3 & -4 & 10 & 41 \end{pmatrix} \quad R_1=R_1 \div 10$$

$$\begin{pmatrix} 1 & -1/5 & 3/10 & 23/10 \\ 0 & 52/5 & -28/5 & -188/5 \\ 0 & -17/5 & 91/10 & 341/10 \end{pmatrix} \quad \begin{matrix} R_2=R_2 - 2\,R_1, \\ R_3 = R_3 - 3R_1 \end{matrix}$$

$$\begin{pmatrix} 1 & -1/5 & 3/10 & 23/10 \\ 0 & 1 & -7/13 & -47/13 \\ 0 & -17/5 & 91/10 & 341/10 \end{pmatrix} \quad R_2 = R_2 \div 52/5$$

$$\begin{pmatrix} 1 & -1/5 & 3/10 & 23/10 \\ 0 & 1 & -7/13 & -47/13 \\ 0 & 0 & 189/26 & 567/26 \end{pmatrix} \quad R_3 = R_3 + 17/5\ R_2$$

$$\begin{pmatrix} 1 & -1/5 & 3/10 & 23/10 \\ 0 & 1 & -7/13 & -47/13 \\ 0 & 0 & 1 & 3 \end{pmatrix} \quad R_3 = R_3 \div 189/26$$

Now the coefficient matrix is upper diagonal and using back substitution we get

$Z=3$

$y - 7/13\,(z) = -47/13$

$13\,y - 7\,(3) = -47$

$13\,y = -47 + 21$

$13\,y = -26$

$Y=2$

$x - 1/5\ y + 3/10\ z = 23/10$

$10x - 2y + 3z = 23$

$10x - 2(-2) + 3(3) = 23$

$X=1$

The solution is $x=1$, $y=2$, $z=3$

4. Solve the given system of equations

$x + 3y + 3z = 16$, $x + 4y + 3z = 18$, $x + 3y + 4z = 19$

by Gauss – Jordan method.

$x + 3y + 3z = 16$

$x + 4y + 3z = 18$

$x + 3y + 4z = 19$

Write the given system of equations in augmented matrix form as

$$\begin{pmatrix} 1 & 3 & 3 & 16 \\ 1 & 4 & 3 & 18 \\ 1 & 3 & 4 & 19 \end{pmatrix}$$

Add multiples of the first row to the other rows to make all the other components in the first column equal to zero.

$$\begin{pmatrix} 1 & 3 & 3 & 16 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{pmatrix} \begin{array}{l} \\ R_2 = R_2 - R_1 \\ R_3 = R_3 - R_1 \end{array}$$

$$\begin{pmatrix} 1 & 0 & 3 & 10 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{pmatrix} \begin{array}{l} R_1 = R_1 - 3R_2, \\ \\ \end{array}$$

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{pmatrix} \quad R_1 = \quad R_1 - 3\,R_3$$

The coefficient matrix finally reduces to the diagonal form and the matrix equation is given by

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

We get $x = 1,\ y = 2,\ z = 3$

5. Solve the given system of equations

$10\,x + y + z = 12$

$2\,x + 10\,y + z = 13$

$x + y + 5\,z = 7$

by Gauss – Jordan method.

The given system of equations are written in augmented matrix form as

$$\begin{pmatrix} 10 & 1 & 1 & 12 \\ 2 & 10 & 1 & 13 \\ 1 & 1 & 5 & 7 \end{pmatrix}$$

Make the element in the first row and first column as 1by

$$\begin{pmatrix} 1 & 1/10 & 1/10 & 12/10 \\ 2 & 10 & 1 & 13 \\ 1 & 1 & 5 & 7 \end{pmatrix} \quad R_1 = R_1 \div 10$$

Add multiples of the first row to the other rows and make all the other components in the first column equal to zero

$$\begin{pmatrix} 1 & 1/10 & 1/10 & 12/10 \\ 0 & 49/5 & 4/5 & 106/10 \\ 0 & 9/10 & 49/10 & 58/10 \end{pmatrix} \quad \begin{array}{l} R_2 = R_2 - 2R_1, \\ R_3 = R_3 - R_1 \end{array}$$

Make the element in the second row and second column as 1

$$\begin{pmatrix} 1 & 1/10 & 1/10 & 12/10 \\ 0 & 1 & 4/49 & 53/49 \\ 0 & 9/10 & 49/10 & 58/10 \end{pmatrix} \quad R_2 = R_2 \div 49/5$$

Add multiples of the second row to the other rows to make all the other components in the second column equal to zero.

$$\begin{pmatrix} 1 & 0 & 0.0918 & 1.0918 \\ 0 & 1 & 4/49 & 53/49 \\ 0 & 0 & 4.8265 & 4.8265 \end{pmatrix} \quad \begin{array}{l} R_1 = R_1 - (1/10)R_2 \\ \\ R_3 = R_3 - (9/10)R_2, \end{array}$$

Make the element in the third row and third column as 1

$$\begin{pmatrix} 1 & 0 & 0.0918 & 1.0918 \\ 0 & 1 & 4/49 & 53/49 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad R_3 = R_3 \div 4.8265$$

Add multiples of the third row to the other rows to make the components in the third column equal to zero

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad \begin{array}{lll} R_1 & \rightarrow & R_1 - 0.0918\,R_3, \\ R_2 & \rightarrow & R_2 - 4/49\,R_3 \end{array}$$

The matrix finally reduces to the form given by

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

Therefore $x = 1, \; y = 1, \; z = 1$

Gauss – Seidel Iterative Method

Let the given system of equations be

$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots\ldots + a_{1n}x_n = C_1$

$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots\ldots + a_{2n}x_n = C_2$

$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots\ldots + a_{3n}x_n = C_3$

…………………………………………………

…………………………………………………

$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots\ldots + a_{nn}x_n = C_n$

The system of equations is first rewritten in the form

$x_1 = (1/a_{11})(C_1 - a_{12}x_2 - a_{13}x_3 - \ldots\ldots - a_{1n}x_n)$ …. (1)

$x_2 = (1/a_{22})(C_2 - a_{21}x_1 - a_{23}x_3 - \ldots\ldots - a_{2n}x_n)$ …. (2)

$x_3 = (1/a_{33})(C_3 - a_{31}x_1 - a_{32}x_2 - \ldots\ldots - a_{3n}x_n)$ …. (3)

………………………………………………………………

$X_n = (1/a_{nn})(C_n - a_{n1}x_1 - a_{n2}x_2 - \ldots\ldots - a_{n,n-1}x_{n-1})$ …. (4)

First let us assume that $x_2 = x_3 = x_4 = \ldots\ldots = x_n = 0$ in (1) and find x_2. Let it be x_1^*. Putting x_1^* for x_1 and $x_3 = x_4 = \ldots\ldots = x_n = 0$ in (2) we get the value for x_2 and let it be x_2^*. Putting x_1^* for x_1 and x_2^* for x_2 and $x_3 = x_4 = \ldots\ldots = x_n = 0$ in (3) we get the value for x_3 and let it be x_3^*. In this way we can find the first approximate values for $x_1, x_2, \ldots\ldots x_n$ by using the relation

$x_1^* = (1/a_{11})(C_1)$

$x_2^* = (1/a_{22})(C_2 - a_{21}x_1^*)$

$x_3^* = (1/a_{33})(C_3 - a_{31}x_1^* - a_{32}x_2^*)$

……………………………………………………………

$X_n^* = (1/a_{nn})(C_n - a_{n1}x_1^* - a_{n2}x_2^* - \ldots\ldots\ldots - a_{n,n-1}x_{n-1}^*)$

Substituting these values of X* in equations (1),(2),(3),…….(4) we get

$x_1^{**} = (1/a_{11})(C_1 - a_{12}x_2^* - a_{13}x_3^* - \ldots\ldots\ldots - a_{1n}x_n^*)$

$x_2^{**} = (1/a_{22})(C_2 - a_{21}x_1^{**} - a_{23}x_3^* - \ldots\ldots\ldots - a_{2n}x_n^*)$

$x_3^{**} = (1/a_{33})(C_3 - a_{31}x_1^{**} - a_{32}x_2^{**} - \ldots\ldots\ldots - a_{3n}x_n^*)$

………………………………………………………………………..

$X_n^{**} = (1/a_{nn})(C_n - a_{n1}x_1^{**} - a_{n2}x_2^{**} - \ldots\ldots\ldots - a_{n,n-1}x_{n-1}^{**})$

Repeating this iteration procedure until two successive iterations give same value, one can get the solution for the given set of equations. This method is efficient for diagonally dominant equations.

Diagonally dominant matrix:

We say a matrix is diagonally dominant if the numerical value of the leading diagonal element (a_{ii}) in each row is greater than or equal to the sum of the numerical values of the other elements in that row.

For example the matrix $\begin{pmatrix} 5 & 1 & -1 \\ 1 & 4 & 2 \\ 1 & -2 & 5 \end{pmatrix}$ is diagonally dominant.

But the matrix $\begin{pmatrix} 5 & 1 & -1 \\ 5 & 2 & 3 \\ 1 & -2 & 5 \end{pmatrix}$ is not, since in the

Second row, the leading diagonal element 2 is less than the sum of the other two elements viz., 5 and 3 in that row.

For the Gauss – Seidel method to converge quickly, the coefficient matrix must be diagonally dominant. If it is not so, we have to rearrange the equations in such a way that the coefficient matrix is diagonally dominant and then only we can apply Gauss – Seidel method.

Problem

1. **Solve the system of equations $4x + 2y + z = 14$, $x + 5y - z = 10$, $x + y + 8z = 20$ using Gauss – Seidel iteration method**.

The given system of equations is

$4x + 2y + z = 14$ …………..(1)

$x + 5y - z = 10$ …………...(2)

$x + y + 8z = 20$ …………...(3)

The coefficient matrix
$$\begin{pmatrix} 4 & 2 & 1 \\ 1 & 5 & -1 \\ 1 & 1 & 8 \end{pmatrix}$$

is diagonally dominant. Hence we can apply Gauss- Seidel method.

From (1), (2) and (3) we get

$x = 1/4\ (14 - 2y - z)$ ……………….....(4)

$y = 1/5\ (10 - x + z)$ ……………….(5)

$z = 1/8\ (20 - x - y)$ ……………….....(6)

First Iteration

Let $y = 0$, $z = 0$ in (4) we get $x = 14/4 = 3.5$

Putting $x = 3.5$, $z = 0$ in (5) we get

$y = 1/5\ [10 - 3.5 + 0] = 1.3$

Putting $x = 3.5$, $y = 1.3$ in (6) we get

$z = 1/8\ [20 - 3.5 - 1.3] = 15.2/8 = 1.9$

Therefore in the first iteration we get $x=3.5$, $y=1.3$ and $z=1.9$

Second Iteration

Putting $y = 1.3$, $z = 1.9$ in (4) we get

$x = 1/4\ [14 - 2\ (1.3) - 1.9] = 2.375$

Putting $x = 2.375$, $z = 1.9$ in (5) we get

$y = 1/5\ [10 - 2.375 + 1.9] = 1.905$

Putting $x = 2.375$, $y = 1.905$ in (6) we get

$z = 1/8\ [20 - 2.375 - 1.905] = 1.965$

In the second iteration we get $x=2.375$, $y=1.905$ and $z=1.965$

Third Iteration

Putting $y = 1.905$, $z = 1.965$ in (4) we get

$x = 1/4\ [14 – 2\ (1.905) – 1.965] = 2.056$

Putting $x = 2.056$, $z = 1.965$ in (5) we get

$y = 1/5\ [10 – 2.056 + 1.965] = 1.982$

Putting $x = 2.056$, $y = 1.982$ in (6) we get

$z = 1/8\ [20 – 2.0565 – 1.982] = 1.995$

In the third iteration we get $x=2.056$, $y= 1.982$, $z=1.995$

Fourth Iteration

Putting $y = 1.982$, $z = 1.995$ in (4) we get

$x = 1/4\ [14 – 2\ (1.982) – 1.995] = 2.010$

Putting $x = 2.010$ $z = 1.995$ in (5) we get

$y = 1/5\ [10 – 2.010 + 1.995] = 1.997$

Putting $x = 2.010$, $y = 1.997$ in (6) we get

$z = 1/8\ [20 – 2.010 – 1.997] = 1.999$

In the fourth iteration we get $x=2.01$, $y=1.997$, $z=1.999$

Fifth Iteration

Putting $y = 1.997$, $z = 1.999$ in (4) we get

$x = 1/4\ [14 – 2\ (1.997) – 1.999] = 2.001$

Putting $x = 2.001$ $z = 1.999$ in (5) we get

$y = 1/5\ [10 – 2.001 + 1.999] = 1.999$

Putting $x = 2.010$, $y = 1.999$ in (6) we get

$z = 1/8\ [20 – 2.001 – 1.999] = 2$

In the fifth iteration we get $x=2$, $y=2$ and $z=2$ and the values satisfy the equations.

2. Solve the system of equations $x + y + 54z = 110$, $27x + 6y - z = 85$, $6x + 15y + 2z = 72$ using Gauss – Seidel iteration method.

The given system is

$$x + y + 54z = 110,$$
$$27x + 6y - z = 85,$$
$$6x + 15y + 2z = 72$$

Interchanging the equations

$$27x + 6y - z = 85, \qquad \text{…. (1)}$$
$$6x + 15y + 2z = 72 \qquad \text{…. (2)}$$
$$x + y + 54z = 110, \qquad \text{…. (3)}$$

we get a diagonally dominant system of equations.

From (1), (2) and (3) we get

$x = 1/27\ (85 - 6y + z)$ ………………....(4)

$y = 1/15\ (2 - 6x - 2z)$ ……………….(5)

$z = 1/54\ (110 - x - y)$ ……………....(6)

First Iteration

Putting $y = 0$, $z = 0$ in (4) we get $x = 85/27 = 3.148$

Putting $x = 3.148$, $z = 0$ in (5) we get

$$y = 1/15\ [72 - 3.148 - 2(0)] = 3.5408$$

Putting $x = 3.148$, $y = 3.5408$ in (6) we get

$$z = 1/54\ [110 - 3.148 - 3.5408] = 1.913$$

In the first iteration we get $x=3.148$, $y=3.5408$, $z=1.913$

Second Iteration

Putting $y = 3.5408$, $z = 1.913$ in (4) we get

$$x = 1/27\ [85 - 6(3.5408) - 1.913] = 2.4322$$

Putting $x = 2.4322$, $z = 1.913$ in (5) we get

$y = 1/15\,[72 - 2.4322 - 2\,(1.913)] = 3.572$

Putting $x = 2.4322$, $y = 3.572$ in (6) we get

$z = 1/54\,[110 - 2.4322 - 3.572] = 1.92585$

In the second iteration we get $x=2.4322$, $y=3.572$, $z=1.92585$

Third Iteration

Putting $y = 3.572$, $z = 1.92585$ in (4) we get

$x = 1/27\,[85 - 6\,(3.572) - 1.92585] = 2.42569$

Putting $x = 2.42569$, $z = 1.92585$ in (5) we get

$y = 1/15\,[72 - 2.42569 - 2\,(1.92585)] = 3.5729$

Putting $x = 2.42569$, $y = 3.5729$ in (6) we get

$z = 1/54\,[110 - 2.42569 - 3.5729] = 1.92595$

In the third iteration we get $x=2.42569$, $y=3.5729$, $z=1.92595$

Fourth Iteration

Putting $y = 3.5729$, $z = 1.92595$ in (4) we get

$x = 1/27\,[85 - 6\,(3.5729) - 1.92595] = 2.42550$

Putting $x = 2.42550$, $z = 1.92595$ in (5) we get

$y = 1/15\,[72 - 2.42550 - 2\,(1.92595)] = 3.5730$

Putting $x = 2.42550$, $y = 3.573$ in (6) we get

$z = 1/54\,[110 - 2.42550 - 3.573] = 1.92595$

In the fourth iteration we get $x=2.425$, $y=3.573$, $z= 1.92595$

Since the values of successive iterations are same, the answer is

$x=2.425$, $y=3.573$, $z= 1.92595$

INTERPOLATION

INTERPOLATION WITH UNEQUAL INTERVALS

Lagrange's Interpolation Formula for unequal intervals

Let $f(x_0), f(x_1), \ldots\ldots f(x_n)$ be the values of the function $y = f(x)$ corresponding to the arguments $x_0. x_1, x_2, \ldots\ldots.. x_n$. The arguments are not equally spaced. Let $f(x)$ be a polynomial in x of degree n. $f(x)$ can be written as

$$\begin{aligned} f(x) = {} & a_0 (x - x_1)(x - x_2) \ldots\ldots\ldots (x - x_n) \\ & + a_1 (x - x_0)(x - x_2) \ldots. (x - x_n) + \ldots\ldots \\ & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ & + a_n (x - x_0)(x - x_1) \ldots.. (x - x_{n-1}) \qquad \ldots. (1) \end{aligned}$$

Where $a_0, a_1, \ldots.. a_n$ are constants. Their values can be obtained by

replacing $x = x_0$ in (1) we get

$$f(x_0) = a_0 (x_0 - x_1)(x_0 - x_2) \ldots\ldots\ldots (x_0 - x_n)$$

$$\text{i.e.} \quad a_0 = \left[\frac{f(x_0)}{(x_0 - x_1)(x_0 - x_2) \ldots\ldots\ldots (x_0 - x_n)} \right] \quad \ldots. (2)$$

Putting $x = x_1$ in (1) we get

$$f(x_1) = a_1 (x_1 - x_0)(x_1 - x_2) \ldots\ldots\ldots (x_1 - x_n)$$

$$\text{i.e.} \quad a_1 = \left[\frac{f(x_1)}{(x_1 - x_0)(x_1 - x_2) \ldots\ldots\ldots (x_1 - x_n)} \right] \quad \ldots. (3)$$

Similarly

$$. \quad a_2 = \left[\frac{f(x_2)}{(x_2 - x_0)(x_2 - x_1) \ldots\ldots\ldots (x_2 - x_n)} \right] \quad \ldots. (4)$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$a_n = \left[\frac{f(x_n)}{(x_n - x_0)(x_n - x_1) \ldots\ldots\ldots (x_0 - x_{n-1})} \right] \quad \ldots. (5)$$

Substituting (2), (3), (4), (5) in (1) we get

$$f(x) = \left[\frac{(x-x_1)(x-x_2)\ldots\ldots.\ (x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots\ldots\ldots(x_0-x_n)}\right] f(x_0)$$

$$+ \left[\frac{(x-x_0)(x-x_2)\ldots\ldots.\ (x-x_n)}{(x_1-x_0)(x_1-x_2)\ldots\ldots\ldots(x_1-x_n)}\right] f(x_1)$$

$$+ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..$$

$$+ \left[\frac{(x-x_0)(x-x_1)\ldots\ldots.\ (x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\ldots\ (x_n-x_{n-1})}\right] f(x_n)$$

If we denote f (x$_0$), f (x$_1$), ….. f (x$_n$) by y$_0$, y$_1$, …… y$_n$, we get

$$f(x) = \left[\frac{(x-x_1)(x-x_2)\ldots\ldots.\ (x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots\ldots\ldots(x_0-x_n)}\right] y_0$$

$$+ \left[\frac{(x-x_0)(x-x_2)\ldots\ldots.\ (x-x_n)}{(x_1-x_0)(x_1-x_2)\ldots\ldots\ldots(x_1-x_n)}\right] y_1$$

$$+ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..$$

$$+ \left[\frac{(x-x_0)(x-x_1)\ldots\ldots.\ (x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\ldots\ldots\ldots(x_n-x_{n-1})}\right] y_n \qquad \ldots\ldots.(6)$$

and this is known as **Lagrange's Interpolation formula**.

Problem:

1. Using Lagrange's interpolation formula calculate y(3) from the data given below.

x	0	1	2	4	5	6
y (x)	1	14	15	5	6	19

Here $x_0 = 0$ $x_1 = 1$ $x_2 = 2$ $x_3 = 4$ $x_4 = 5$ $x_5 = 6$

$Y_0 = 1$ $y_1 = 14$ $y_2 = 15$ $y_3 = 5$ $y_4 = 9$ $y_5 = 19$

Lagrange's interpolation formula is

$$y(x) = \left[\frac{(x-x_1)(x-x_2)(x-x_3)(x-x_4)(x-x_5)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)(x_0-x_4)(x_0-x_5)}\right] y_0$$

$$+ \left[\frac{(x-x_0)(x-x_2)(x-x_3)(x-x_4)(x-x_5)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)(x_1-x_4)(x_1-x_5)}\right] y_1$$

$$+ \left[\frac{(x-x_0)(x-x_1)(x-x_3)(x-x_4)(x-x_5)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)(x_2-x_4)(x_2-x_5)}\right] y_2$$

$$+ \left[\frac{(x-x_0)(x-x_1)(x-x_2)(x-x_4)(x-x_5)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)(x_3-x_4)(x_3-x_5)}\right] y_3$$

$$+ \left[\frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)(x-x_5)}{(x_4-x_0)(x_4-x_1)(x_4-x_2)(x_4-x_3)(x_4-x_5)}\right] y_4$$

$$+ \left[\frac{(x-x_0)(x-x_1)(x-x_2)(x-x_3)(x-x_4)}{(x_5-x_0)(x_5-x_1)(x_5-x_2)(x_5-x_3)(x_5-x_4)}\right] y_5$$

Substituting the values, we can write

$$y(x) = \left[\frac{(x-1)(x-2)(x-4)(x-5)(x-6)}{(0-1)(0-2)(0-4)(0-5)(0-6)}\right] X\ 1$$

$$+ \left[\frac{(x-0)(x-2)(x-4)(x-5)(x-6)}{(1-0)(1-2)(1-4)(1-5)(1-6)}\right] X14$$

$$+ \left[\frac{(x-0)(x-1)(x-4)(x-5)(x-6)}{(2-0)(2-1)(2-4)(2-5)(2-6)}\right] X15$$

$$+ \left[\frac{(x-0)(x-1)(x-2)(x-5)(x-6)}{(4-0)(4-1)(4-2)(4-5)(4-6)}\right] X5$$

$$+ \left[\frac{(x-0)(x-1)(x-2)(x-4)(x-6)}{(5-0)(5-1)(5-2)(5-4)(5-6)}\right] X6$$

$$+ \left[\frac{(x-0)(x-1)(x-2)(x-4)(x-5)}{(6-0)(6-1)(6-2)(6-4)(6-5)}\right] X19$$

Substituting $x = 3$, we get $y(3) = 10$.

2. using Lagrange's interpolation formula for unequal intervals find the value of y when x = 10 using the values of x and y given below.

x	5	6	9	11
y	12	13	14	16

Given: $x_0 = 5$, $x_1 = 6$, $x_2 = 9$, $x_3 = 11$, $y_0 = 12$ $y_1 = 12$ $y_2 = 14$ $y_3 = 16$

Lagrange's interpolation formula is

$$y(x) = \left[\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}\right] y_0$$

$$+ \left[\frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}\right] y_1$$

$$+ \left[\frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}\right] y_2$$

$$+ \left[\frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}\right] y_3$$

Substituting the values with x=10, we get

$$y(10) = \left[\frac{(10-6)(10-9)(10-11)}{(5-6)(5-9)(5-11)}\right] x\,(12)$$

$$+ \left[\frac{(10-5)(10-9)(10-11)}{(6-5)(6-9)(6-11)}\right] x\,(13)$$

$$+ \left[\frac{(10-5)(10-6)(10-11)}{(9-5)(9-6)(9-11)}\right] x\,(14)$$

$$+ \left[\frac{(10-5)(10-6)(10-9)}{(11-5)(11-6)(11-9)}\right] x\,(16)$$

$= \quad 2 \quad - \quad 4.33 \quad + \quad 11.66 \quad + \quad 5.3$

$Y(10) = 14.63$

3. Use Lagrange's interpolation formula to find the form of the function for the Data given below.

x	3	2	1	-1
f (x)	3	12	15	-21

Solution:

Here $x_0 = 3, x_1 = 2, x_2 = 1, x_3 = -1, y_0=3, y_1=12, y_2 = 15, y_3 = -21$

$$f(x) = \left[\frac{(x-2)(x-1)(x+1)}{(3-2)(3-1)(3+1)}\right] \times 3$$

$$+ \left[\frac{(x-3)(x-1)(x+1)}{(2-3)(2-1)(2+1)}\right] \times 12$$

$$+ \left[\frac{(x-3)(x-1)(x+1)}{(1-3)(1-2)(1+1)}\right] \times 15$$

$$+ \left[\frac{(x-3)(x-1)(x+1)}{(-1-3)(-1-3)(-1-1)}\right] \times (-21)$$

Simplifying, we get $f(x) = x^3 - 9x^2 + 17x + 6$.

f(x) is the required form of function for the given data.

Newton's Forward Interpolation Formula

We know that

$$\Delta y_0 = y_1 - y_0 \quad \text{i.e.} \quad y_1 = y_0 + \Delta y_0 = (1+\Delta)\, y_0$$

$$\Delta y_1 = y_2 - y_1 \quad \text{i.e.} \quad y_2 = y_1 + \Delta y_1 = (1+\Delta)\, y_1 = (1+\Delta)^2 y_0$$

$$\Delta y_2 = y_3 - y_1 \quad \text{i.e.} \quad y_3 = y_2 + \Delta y_2 = (1+\Delta)\, y_2 = (1+\Delta)^3 y_0$$

In general $y_n = (1+\Delta)^n y_0$

Expanding $(1+\Delta)^n$ by using Binomial theorem we have

$$y_n = \left[1 + n\Delta + \frac{n(n-1)}{2!}\Delta^2 + \frac{n(n-1)(n-2)}{3!}\Delta^3 + \ldots\ldots\ldots \right] y_0$$

$$y_n = \left[y_0 + n\Delta y_0 + \frac{n(n-1)}{2!}\Delta^2 y_0 + \frac{n(n-1)(n-2)}{3!}\Delta^3 y_0 + \ldots\ldots\ldots \right]$$

This result is known as Gregory-Newton forward interpolation formula(or) Newton's formula for equal intervals.

Problem:

1. **The following table gives the population of a town taken during six censuses. Estimate the increase in the population during the period 1946 to 1948**.

Year	1911	1921	1931	1941	1951	1961
Population (in thousands)	12	13	20	27	39	52

The difference table is given below.

x	y = f(x)	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$
1911 (x_o)	12 (y_0)					
		1 (Δy_0)				
1921	13		6 ($\Delta^2 y_0$)			
		7		-6 ($\Delta^3 y_0$)		
1931	20		0		11 ($\Delta^4 y_0$)	
		7		5		-20 ($\Delta^5 y_0$)
1941	27		5		-9	
		12		-4		
1951	39		1			
		13				
1961	52					

Here $x_0 = 1911$ $h = 10$ $y_0 = 12$

Newton's forward interpolation formula is

$$y(x_0 + nh) = y_0 + n\Delta y_0 + \frac{n(n-1)}{2!}\Delta^2 y_0 + \frac{n(n-1)(n-2)}{3!}\Delta^3 y_0 + \dots\dots\dots$$

The population in the year 1946 ie y(1946) is to be calculated.

Here $x_0 + nh = 1946$

i.e., $1911 + n \times 10 = 1946$ i.e., $n = 3.5$

Therefore

$$y(1946) = 12 + (3.5)(1) + \frac{(3.5)(3.5-1)}{2} \times 6$$

$$+ \frac{(3.5)(3.5-1)(3.5-2)}{6} \times (-6)$$

$$+ \frac{(3.5)(3.5-1)(3.5-2)(3.5-3)}{24} \times (11)$$

$$+ \frac{(3.5)(3.5-1)(3.5-2)(3.5-3)(3.5-4)}{120} \times (-20)$$

$= 12 + 3.5 + 26.25 - 13.125 + 3.0078 + 0.5469$

$= 32.18$ Thousands

The population in the year 1948 ie y (1948) is calculated as below.

Here $x_0 + nh = 1948$

i.e. $1911 + n.10 = 1948$, i.e., $n = 3.7$

Therefore $y(1946) = 12 + (3.7)(1) + \frac{(3.7)(3.7-1)}{2} \times 6$

$$+ \frac{(3.7)(3.7-1)(3.7-2)}{6} \times (-6)$$

$$+ \frac{(3.7)(3.7-1)(3.7-2)(3.7-3)}{24} \times (11)$$

$$+ \frac{(3.7)(3.7-1)(3.7-2)(3.7-3)(3.7-4)}{120} \times (-20)$$

$= 12 + 3.7 + 29.97 - 16.983 + 5.4487 + 0.5944$

$= 34.73$ thousands

Increase in the population during the period 1946 to 1948 is

= Population in 1948 - Population in 1946

= 34.73 - 32.18 = **2.55** thousands.

2. From the following data, find θ at x = 43.

x	40	50	60	70	80	90
θ	184	204	226	250	276	304

Since the vale to be interpolated is at the beginning of the table, we can use Newton’s forward interpolation formula.

The difference table is given below.

x	y = f(x)	Δy	$\Delta^2 y$	$\Delta^3 y$
40 $(=x_o)$	184 $(=y_0)$			
		20 $(=\Delta y_0)$		
50	204		2$(=\Delta^2 y_0)$	
		22		0 $(=\Delta^3 y_0)$
60	226		2	
		24		0
70	250		2	
		26		0
80	276		2	
		28		
90	304			

Here $x_0 = 40$ $h = 10$ $y_0 = 184$

By Newton's formula we have

$$y(x_0 + nh) = y_0 + n\Delta y_0 + \frac{n(n-1)}{2!}\Delta^2 y_0 + \frac{n(n-1)(n-2)}{3!}\Delta^3 y_0 + \ldots\ldots\ldots$$

We have to find the value of y at x=43 ie. y(43)

i.e., $x_0 + nh = 43$

i.e., $40 + n.10 = 43$ i.e., $n = (43 - 40)/10 = 0.3$

Therefore $y(43) = 184 + (0.3)(20) + \frac{(0.3)(0.3 - 1)}{2} \times (2)$

$= 184 + 6 - 0.21 =$ **189.79**

Newton – Gregory Formula for Backward Interpolation

The backward difference formula for any function f (x) is given as

$\nabla f(x) = f(x) - f(x - h)$, h being interval of data x. If f (a), f(a+h), f(a+2h) ……….. f(a+nh) are the (n+1) values for the function f (x) corresponding to an independent values of x at equal intervals x = a, a+h, a+2h, …. (a+nh). We can write f(x) in a polynomial of the form

$$f(x) = a_0 + a_1\{x - (a+nh)\} + a_2\{x - (a+nh)\}\{x - a + (n-1)h\}$$

$$+ \dots\dots + a_n\{x - (a+nh)\}\{x + a + (n-1)h\} \dots \{x - (a+h)\} \quad \dots(1)$$

Where the constant a_0, a_1, a_2, …… a_n, are to be determined.

Putting x = a + nh, $a + \overline{n-1}\,h$, ….. a in succession in (1) we get

$f(a + nh) = a_0$ ie. $a_0 = f(a + nh)$

$f(a + (n-1)h) = a_0 + a_1(-h)$

$$\text{ie. } a_1 = \frac{f(a+nh) - f(a+(n-1)h)}{h} = \frac{\nabla f(a+nh)}{h}$$

Similarly, $a_2 = \dfrac{\nabla^2 f(a+nh)}{2!\,h^2}$ $\qquad a_n = \dfrac{\nabla^n f(a+nh)}{n!\,h^n}$

Substituting these values of a_0, a_1, a_2, ……. a_n in (1) we get

$$f(x) = f(a+nh) + \{x - (a-nh)\}\frac{\nabla f(a+nh)}{h} +$$

$$+ \{x - (a+nh)\}\{x - (a + \overline{n-1}\,h)\} * \frac{\nabla^2 f(a+nh)}{2!\,h^2}$$

$$+ \{x - (a+nh)\}\{x - (a + \overline{n-1}\,h)\} \dots \{x - (a+h)\} * \frac{\nabla^n f(a+nh)}{n!\,h^n} \quad \dots (2)$$

This is Newton – Gregory formula for backward interpolation.

Putting $u = x - (a + nh) / h$ or $x = a + nh + hu$ in (2) we get

$$f(x) = f[a + h(u+n)]$$

$$= f(a + nh) + u\nabla f(a + nh) + \frac{u(u+1)}{2!}\nabla^2 f(a + nh) + \ldots\ldots$$

$$+ \frac{u(u+1)\ldots\ldots(u+n-1)\nabla^n f(a + nh)}{n!}$$

This is the usual form of Newton-Gregory interpolation formula.

Problems:

1. Using Newton's backward interpolation method, fit a polynomial of degree three for the given data.

x	3	4	5	6
y	6	24	60	120

Solution

The Newton-Gregory backward interpolation formula is

$$y(x_0 + nh) = y_0 + n\nabla y_0 + \frac{n(n-1)}{2!}\nabla^2 y_0 + \frac{n(n-1)(n-2)}{3!}\nabla^3 y_0 + \ldots\ldots\ldots$$

Here $x_0 + n h = x$ $x_0 = 6$, $n=?$, $h=1$

$n = x-6$

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$
3	6	18		
4	24	36	18	$6=\nabla^3 y_0$
5	60	$60=\nabla y_0$	$24=\nabla^2 y_0$	
$6=x_0$	$120=y_0$			

$Y(x) = 120 + (x-6)\,60 + (1/2)(x-6)(x-5)\,24 + (1/6)(x-6)(x-5)(x-4)\,6$

Therefore $Y(x)= x^3-3x^2+2x$

2. Find the value of y(63) from the data given below.

x	45	50	55	60	65
y	114.84	96.16	83.32	74.48	68.48

Solution

The Newton-Gregory backward interpolation formula is

$$y(x_0 + nh) = y_0 + n\nabla y_0 + \frac{n(n-1)}{2!}\nabla^2 y_0 + \frac{n(n-1)(n-2)}{3!}\nabla^3 y_0 + \ldots\ldots\ldots$$

Here, $x_0 + nh = x$ $x_0 = 65$, $h=5$, $x= 63$, $n=?$

$n = (63 - 65)/5 = -2/5$

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$
45	114.84	-18.68			
50	96.16	-12.84	5.84	-1.84	
55	83.32	-8.84	4	$-1.16=\nabla^3 y_0$	$0.68=\nabla^4 y$
60	74.48	$-6=\nabla y_0$	$2.84= \nabla^2 y_0$		
$65= x_0$	$68.48=y_0$				

$$Y(63) = 68.48 + (-\tfrac{2}{5})(-6) + (\tfrac{1}{2!})(-\tfrac{2}{5})(-\tfrac{2}{5}+1)(2.84) + (\tfrac{1}{3!})(-\tfrac{2}{5})(-\tfrac{2}{5}+1)(-\tfrac{2}{5}+2)(-1.16)$$

$$+ (\tfrac{1}{4!})(-\tfrac{2}{5})(-\tfrac{2}{5}+1)(-\tfrac{2}{5}+2)(-\tfrac{2}{5}+3)\ (0.68)$$

Y(63) = 68.48+2.4 – 0.3408 + 0.07424 – 0.028288

Y(63) = 70.585

3. Using Newton's backward interpolation formula evaluate y(9.5) from the data given below.

x	6	7	8	9	10
y	46	66	81	93	101

Solution

$$y(x_0 + nh) = y_0 + n\nabla y_0 + \frac{n(n-1)}{2!}\nabla^2 y_0 + \frac{n(n-1)(n-2)}{3!}\nabla^3 y_0 + \ldots\ldots\ldots$$

Here, $x_0 + nh = x$ $x_0 = 10$, h=10, x= 9.5, n=?

n = (9.5 – 10)/10 = - 0.5

x	y	∇y	$\nabla^2 y$	$\nabla^3 y$	$\nabla^4 y$
6	46	20			
7	66	15	-5	2	
8	81	12	-3	$-1=\nabla^3 y_0$	$-3=\nabla^4 y$
9	93	$8=\nabla y_0$	$-4=\nabla^2 y_0$		
$10= x_0$	$101=y_0$				

$$Y(9.6) = 101 + (-0.5)(8) + \left(\frac{1}{2!}\right)(-0.5)(-0.5+1)(-4) + \left(\frac{1}{3!}\right)(-0.5)(-0.5+1)(-0.5+2)(-1)$$

$$+ \left(\frac{1}{4!}\right)(-0.5)(-0.5+1)(-0.5+2)(-0.5+3)\ (-3)$$

Y(9.6) = 101-4 + 0.5 + 0.0625+0.1172

Y(9.6) = 97.68

The Method of Least Squares:

Given n-paired observations (x_1, y_1), (x_2, y_2), ... (x_n, y_n) of two variables x and y and we want to determine a function f (x) such that

$$f(x_i) = y_i, \qquad i = 1, 2, 3, \ldots\ldots\ n$$

we have to fit a straight line of the form $y = a + bx$ and choose the parameters a and b such that the sum of squares of deviations of the fitted value with the given data is least or minimum. This method of fitting a curve is called **least squares fitting method.** The sum of squares of deviations of the fitted value with the given value is given by

$$s = \sum_1^n [f(x_i) - y_i]^2 \quad \text{is a function of } a_1, a_2, a_3, \ldots\ a_n$$

According to the principle of least squares a's may be determined by the requirement that $\sum_i s$ is least i.e.

$$\partial s / \partial a_1 = 0,\ \ \partial s / \partial a_2 = 0,\ \ \ldots\ldots\ \ \partial s / \partial a_i = 0,$$

A set of these equations is called the normal equations. The unknowns in y = f (x) are determined using these normal equations. For the linear equation y=a+bx

the residuals are given by $v_i = a + bx_i - y_i$, and the sum of residues is given by

$s = \sum_1^n [a + bx_i - y_i]^2$ Differentiating with respect to a and b, we get

$$\partial s / \partial a = 2. \sum_1^n [a + bx_i - y_i] = 0$$

and $$\partial s / \partial b = 2. \sum_1^n x_i[a + bx_i - y_i] = 0$$

collecting the coefficients of a and b in the above equations, we get

$$\mathbf{na + b} \sum_{i=1}^{n} x_i \;\; \mathbf{=} \;\; \sum_{i=1}^{n} y_i \qquad \text{and}$$

$$\mathbf{a} \sum_{i=1}^{n} x_i \; \mathbf{+} \; \mathbf{b} \sum_{i=1}^{n} x_i^2 \; \mathbf{=} \; \sum_{i=1}^{n} x_i y_i$$

These equations are called the normal equations. Solving these two equations simultaneously we can get the values of a and b..

In the case of a quadratic polynomial the normal equations are given by

$$na_0 + a_1 \sum x_i + a_2 \sum x_i^2 = \sum y_i$$

$$a_0 \sum x_i + a_1 \sum x_i^2 + a_2 \sum x_i^3 = \sum x_i y_i$$

$$a_0 \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4 = \sum x_i^2 y_i$$

Solving these three equations simultaneously one get the values of a_0,a_1 and a_2 and the quadratic equation is fitted as $y= a_0+a_1x+a_2x^2$.

Problems:

1. **Using the method of least squares, fit a straight line of the form y=a+bx to the given data**

x	1	2	3	4
y	1.7	1.8	2.3	3.2

Solution: In this case n = 4

$\sum x_i$ = 1 + 2 + 3 + 4 = 10

$\sum y_i$ = 1.7 + 1.8 + 2.3 + 3.2 = 9

$\sum x_i^2$= 1 + 4 + 9 + 16 = 30

$\sum x_i y_i =$ (1x1.7) + (2x1.8) + (3x2.3) + (4x32) = 25

The normal equations are given by

$$4a + 10b = 9$$

$$10a + 30b = 25$$

Solving for a and b, we get

$$a = 1, \qquad b = ½$$

The linear equation fitted to the given data is

$$y = 1 + (½)\, x$$

2**. Fit a quadratic curve from the following data using the principle of least squares.**

x	0	1	2	3	4
y	1	1.8	1.3	2.5	6.3

The normal equations are

$$na_0 + a_1 \sum x_i + a_2 \sum x_i^2 = \sum y_i$$

$$a_0 \sum x_i + a_1 \sum x_i^2 + a_2 \sum x_i^3 = \sum x_i y_i$$

$$a_0 \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4 = \sum x_i^2 yi$$

Here number of data n=5

$$\sum x_i = 0 + 1 + 2 + 3 + 4 = 10$$

$$\sum x_i y_i = (0x1) + (1x1.8) + (2x1.3) + (3x2.5) + (4x6.3) = 37.1$$

$$\sum x_i^2 = 0 + 1 + 4 + 9 + 16 = 30$$

$$\sum x_i^3 = 0 + 1 + 8 + 27 + 64 = 100$$

$\sum x_i^4 = \quad 0 + 1 + 16 + 81 + 256 = 354$

$\sum x_i^2 y_i = \quad 0 + (1x1.8) + (4x1.3) + (9x2.5) + (16x6.3) = 103.3$

With these substitutions equation (2) becomes

$5a_0 + 10a_1 + 30a_2 = 12.9$

$10a_0 + 30a_1 + 100a_2 = 37.1$

$30a_0 + 100a_1 + 354a_2 = 103.3$

Solving these we get

$a_0 = 1.42, \quad a_1 = -1.07, \quad a_2 = 0.55$

The fitted quadratic curve becomes

$$Y = 1.42 - 1.07 x + 0.55 x^2$$

Unit I - Solution of algebraic and transcendental equations

An equation f(x)=0 which is only a polynomial in x is known as an algebraic equation.

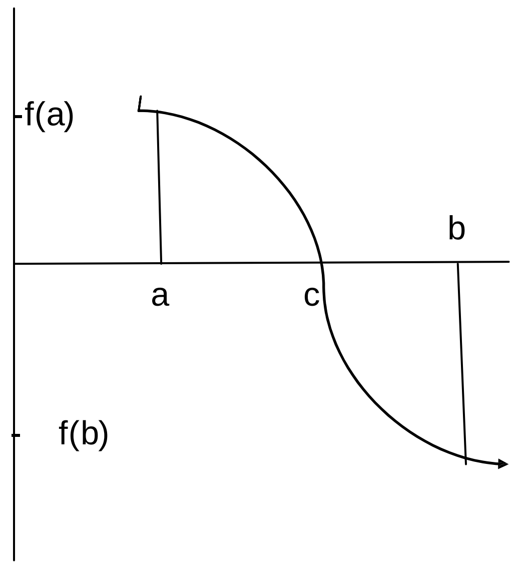
eg. $X^5 - x^4 + x - 26 = 0$.

Whereas, an equation containing transcendental terms, such as exponential terms, trigonometric terms, logarithmic terms etc, are known as transcendental equations.

eg. $e^x - x + 5 = o$; $\quad x + 5 \cos x + 2 = 0$

The point at which the function y = f (x) cuts in the x – axis is known as the root of the equation (or) zero of the equation.

y = f (x)



y = f (a) is positive

y = f (b) is neative y

= f (c) is zero

Therefore c is the root

of the equation

Two points a and b are to be identified for the given equation, such that for one of the points a, y=f(a) is positive and for the point b, f(b) is negative. Then the first approximate root is given be $x_0 = (a + b) /2$.

Iteration Method :

The root of the given equation f(x) = 0 can be determined by iteration method. Let x_0 be the approximate initial root of the given equation calculated as mentioned above. We have to rewire the given equation in the form of

$$x = \Phi (x)$$

The first approximate root is given by $x_1 = \Phi(x_0)$

The successive approximations are given by

$X2 = \Phi (x_1)$

$x_3 = \Phi (x_2)$

…………..

$x_n = \Phi (x_{n-1})$

The process of finding root is continued until the successive roots are same to the required accuracy.

Note

The iteration method will give converging results if

$\Phi' (x) < 1.$

e.g. Consider the equation $f (x) = x^2 + x – 3 = 0$

f (0) =-3, negative; f (1) = -1, negative; f (2) = 3, positive

Hence the root lies between 1 and 2. The initial root is calculated as

$x_0 = (1 + 2) / 2 = 1.5$

From the given equation, we can write $x = 3 – x^2 = \Phi (x)$.

Differentiating, we get $\Phi' (x) = - 2 (x)$

$|\Phi'(1.5)| = 3 > 1$

Therefore this form of Φ(x) will not give converging solutions.

If we rewrite the given equation as $x(x + 1) = 3$; then we can write

$x = 3 / (x + 1) = \Phi(x)$

$\Phi'(x) = -3 / (1 + x)^2$ and $\Phi'(1.5) < 1$

Therefore this form of Φ (x) will give converging solutions.

Problems:

1. **Find the root of the equation $x^{3+} x^{2-} 1 = 0$ by iteration method correct to two decimal places.**

Given $f(x) = x^3 + x^{2-} 1 = 0$ f

(0) = -1 negative

f (1) = 1, positive

Therefore the initial root is $x_0 = (0 + 1) / 2 = 0.5$

We can rewrite the given equation $x^3 + x^2 - 1 = 0$ as

$x^2 (x + 1) = 1$

$x = 1/ \sqrt{(1 + x)}$

Therefore $\Phi(x) = 1/ \sqrt{(1 + x)}$ and $\Phi'(x) = -1/ 2(x + 1)^{3/2}$

$|\Phi'(x)|_{x = 0.5} < 1$

Therefore this form of Φ(x) will give converging solutions.

The first approximate solution is

$x_1 = \Phi(x_0) = 1/ \sqrt{(1+ x_0)} = 1 / \sqrt{(1+ 0.5)} = 0.81649$

$x_2 = \Phi(x_1) = 1/ \sqrt{(1 + 0.81649)} = 0.74196$

$x_3 = \Phi (x_2) = 1/\sqrt{(1 + 0.74196)} = 0.75767$

$x_4 = \Phi (x3) = 1/ (\sqrt{1+ 0.75767) =} 0.75427$

Since $x_3 \approx x_4$ up to two decimal places, the root of the given equationis 0.7543.

2. Find the root of the equation $3x - \log_{10 x} - 6 = 0$ by iteration method.

Given $f(x) = 3x - \log_{10 x} - 6$

$f(1) = -3$ (negative) $f(2) = -0.3010$ (negative)

$f(3) = 2.5229$ (positive)

The root lies between 2 and 3.

Let the initial approximate root be, $x_0 = 2$

We can rewrite the equation $3x - \log_{10} x - 6 = 0$ as

$X = 1/3(6 + \log_{10} x)$

Therefore $\Phi (x) = 1/3(6 + \log_{10} x)$

The first approximate solution is

$x_1 = \Phi(x_0) = 1/3(6 + \log_{10} x_0) = 1/3 (6 + \log_{10} 2) = 2.1003$

$x_2 = \Phi(x_1) = 1/3 (6 + \log_{10} 2.1003) = 2.1074$

$x_3 = \Phi(x_2) = 1/3 (6 + \log_{10} 2.1074) = 2.1079$

Since $x_2 \approx x_3$ up to three decimal places, the root of the given equation is 2.1079

3. Find the root of the equation $f(x) = 3x - \cos x - 1 = 0$ by iteraion method.

Given $f(x) = 3x - \cos x - 1 = 0$ Now $f(0)$ = negative and $f(1)$ = positive

Let us take initial root as $x_0 = 0.6$

We can write $x = \frac{1}{3}(1 + \cos x) = \Phi(x)$

$x_1 = \Phi(x_0) = 1/3\,(1 + \cos 0.6) = 0.60845$

$x_2 = \Phi(x_1) = 1/3\,(1 + \cos 0.60845) = 0.60684$

$x_3 = \Phi(x_2) = 1/3\,(1 + \cos 0.60684) = 0.60715$

$x_4 = \Phi(x_3) = 1/3\,(1 + \cos 0.60715) = 0.6071$

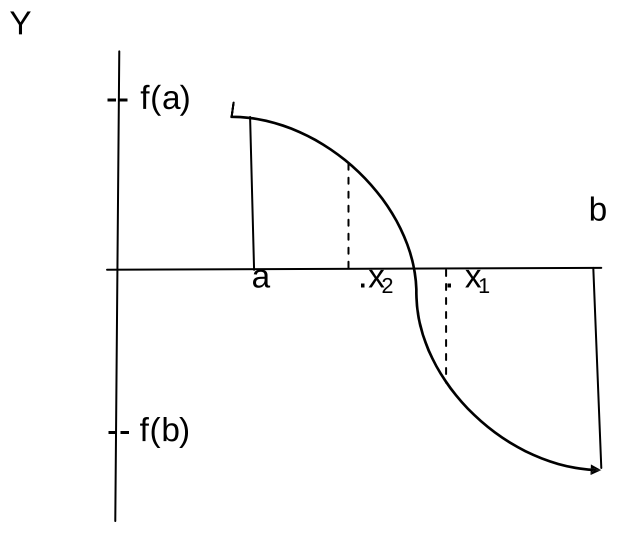
Since $x_3 \approx x_4$ correct to three decimal places, the root of the given equation is 0.6071.

Bisection method:

If f (a) and f (b) are of opposite signs, then the equation f (x) = 0 will have at least one real root between a and b. The bisection method is useful to find the root between a and b. The first approximate root is taken as the midpoint of the range a and b. i.e. $x_0 = (a + b) / 2$

Then we have to find $f(x_0)$. Let us assume f(a) as positive and f(b) as negative. Let $f(x_0)$ be negative. Then the root lies between a and x_0.

If $f(x_0)$ is positive, then the root lies between x_0 and b. We have to bisect the interval in which the root lies and the process is to be repeated.



In the above diagram, for the given curve $y = f(x)$, $f(a)$ is positive and $f(b)$ is negative. The first approximate root is the midpoint of a and b

i.e. $x_1 = (a + b) / 2$

$f(x_1)$ is negative. Therefore the next approximation is the midpoint of x_1 and a

$x_2 = (a + x_1) / 2$

Similarly $x_3 = (x_1 + x_2)/ 2$

The process is repeated until two successive roots are equal to the required degree of approximation.

Problems :

1. Find the root of the equation $x^3 – x – 1 = 0$ correct to two decimal places by bisection method.

Let $f(x) = x^3 – x – 1$

$f(0) = -1$ ie negative

$f(1) = -1$ negative

$f(1.5) = 0.875$ is positive

Therefore the initial root is $x_0 = (1 + 1.5) / 2 = 1.25$

$f(x_0) = f(1.25) = (1.25)^3 – 1.25 – 1 = - 0.29688$

Since $f(x_0)$ is negative, the root lies between 1.25 and 1.5. The next approximate root is

$x_1 = (1.25 + 1.5)/ 2 = 1.375$

$f(1.375) = (1.375)^3 – 1.375 – 1 = 0.22461$

Since f (1.375) is positive, the next root lies between 1.25 and 1.375

Ie. $x_2 = (1.25 + 1.375) / 2 = 1.3125$

$f(1.3125) = (1.3125)^3 - 1.3125 - 1 = -0.051514$ (negative)

Therefore the root lies between 1.3125 and 1.375

$x_3 = (1.3125 + 1.375) / 2 = 1.3438$

$f(x_3) = f(1.3438) = 0.0824$ (Positive)

Therefore the root lies between 1.3438 and 1.3125

Ie. $x_4 = (1.3125 + 1.3438) / 2 = 1.3282$

$f(x_4) = 0.014898$ (Positive)

Therefore the root lies between 1.3282 and 1.3125

Therefore $x_5 = (1.3125 + 1.3282) / 2 = 1.3204$

Since $x_4 \approx x_5$ correct to two decimal places, the root is 1.3204

Note: If $f(x) \approx 0$, then x is the root. Here $f(x_5) = 0.0183$ which is very small and near to zero. Therefore the root is 1.3204.

2. Find the root of the equation $x \log_{10} x - 1.2 = 0$ by bisection method.

Let $f(x) = x \log_{10} x - 1.2$

$f(2) = 2 \log_{10} 2 - 1.2 = -0.598$ (negative)

$f(3) = 3 \log_{10} 3 - 1.2 = 0.2313$ (positive)

Therefore the root lies between 2 and 3

$x_0 = (2 + 3) / 2 = 2.5$

$f(x_0) = f(2.5) = 2.5 \log_{10} 2.5 - 1.2 = -0.2053$

Since f (2.5) is negative and f (3) is positive the root lies between 2.5 and 3

Therefore the next approximate root is

$x_1 = (2.5 + 3) / 2 = 2.75$

$f(2.75) = 2.75 \log_{10} 2.75 - 1.2 = 0.008$

Since f (2.75) is > 0 and f (2.5) is < o the next root lies between 2.75 and 2.5

Therefore $x_2 = (2.75 + 2.5) / 2 = 2.625$

$f(x_2) = f(2.625) = -0.10$

Therefore the next approximation is

$x_3 = (2.625 + 2.75) / 2 = 2.6875$

$f(x_3) = f(2.6875) \approx 0$

Therefore the root of the equation is 2.6875

Newton- Raphson Method:

Let x_0 be the approximate root of the equation $f(x) = 0$

Let $x_1 = x_0 + h$ be the exact root of the equation

Therefore $f(x_1) = 0$

By Taylor's series expansion, we can write

$f(x_1) = f(x_0 + h) = f(x_0) + h/1!\, f'(x_0) + (h^2/2!)\, f''(x_0) + \ldots..$

Since h is very small, h^2 is negligibly small and we can ignore the higher order terms. Therefore we can write

$f(x_1) = f(x_0) + h f'(x_0) = 0$

i.e. $h = -f(x_0) / f'(x_0)$

Therefore $x_1 = x_0 + h = x_0 - f(x_0) / f'(x_0)$

The next approximation is

$x_2 = x_1 - f(x_1) / f'(x_1)$

In general, $x_{n+1} = x_n - f(x_n) / f'(x_n)$

This is known as Newton – Raphson iteration formula.

Problems:

1. Find the root of the equation $x^3 - 3x + 1 = 0$ by Newton – Raphson method.

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

$f'(x) = 3x^2 - 3$

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

$f(2) = 2^3 - 6 + 1 = 3$ (Positive)

Therefore $x_0 = (1 + 2) / 2 = 1.5$

$f(x_0) = 1.5^3 - 3 \times 1.5 + 1 = -0.125$

$f'(x_0) = 3(1.5)^2 - 3 = 3.75$

$x_1 = 1.5 - (-0.125/3.75) = 1.5333$

$f(x_1) = (1.5333)^3 - 3 \times 1.5333 + 1 = 0.0049$

$f'(x_1) = 3(1.5333)^2 - 3 = 4.053$

$x_2 = 1.5333 - (0.0049/4.053) = 1.5321$

Since $x_2 \approx x_3$ the root of the given equation is 1.5321

2. Find the root of the equation $\cos x - xe^x = 0$ by Newton – Raphson method.

$f(x) = \cos x - xe^x$

$f'(x) = -\sin x - xe^x - e^x$

Let $x_0 = 0.5$

$x_1 = x_0 - f(x_0) / f^!(x_0) = 0.5 – f(0.5)/ f^!(0.5)$

$= 0.5 – (0.533/-2.9525)$

$= 0.5 + 0.0181 = 0.5181$

$x_2 = x_1 - f(x_1) / f^!(x_1) = 0.5181 – f(0.5181)/ f^!(0.5181)$

$= 0.5181 – (- 0.00104 /-3.0438)$

$= 0.5181 - 0.00034 = 0.5178$

$x_3 = x_2 - f(x_2) / f^!(x_2) = 0.5178 – f(0.5178)/ f^!(0.5178)$

$= 0.5178 – (- 0.00012 /-3.0422)$

$= 0.5178$

Since $x_2 \approx x_3$ the root of the given equation is 0.5178.

Rate of convergence of Newton-Raphson method

Let α be the root of the equation f(x) = 0

Let x_n be the approximate root of the equation and e_n be the small error by which x_n and α differs

Therefore $x_n = \alpha + e_n$

Similarly $x_{n+1} = \alpha + e_{n+1}$

Newton- Raphson formula is

$$x_{n+1} = x_n – f(x_n) / f'(x_n)$$

$$\alpha + e_{n+1} = \alpha + e_n - f(\alpha + e_n) / f'(\alpha + e_n)$$

$$e_{n+1} = e_n – [f(\alpha + e_n) / f '(\alpha + e_n)]$$

using Taylor’s series expansion we can write

$$e_{n+1} = e_n - \left[\frac{f(\alpha) + e_n f'(\alpha) + \frac{e_n^2}{2!} f''(\alpha) + \dots\dots\dots}{f'(\alpha) + e_n f''(\alpha) + \dots\dots\dots} \right]$$

$$= e_n - \left[\frac{f(\alpha) + e_n f'(\alpha) + \frac{e_n^2}{2!} f''(\alpha)}{f'(\alpha) + e_n f''(\alpha)} \right]$$

$$= \frac{e_n f'(\alpha) + e_n^2 f''(\alpha) - f(\alpha) - e_n f'(\alpha) - e_n^2 f''(\alpha)/2}{f'(\alpha) + e_n f''(\alpha)}$$

$$= \frac{\frac{e_n^2 f''(\alpha)}{2!}}{f'(\alpha)\,[1 + e_n f''(\alpha)/ f'(\alpha)]}$$

$$= K e_n^2$$

$$e_{n+1} \propto e_n^2$$

i.e. the error in successive steps decreases as the square of error in the previous step. Thus the order of convergence is two.

Birge – Vieta method :

The real root of a polynomial equation $f(x) = 0$ can be obtained by this method.

Let $f(x) = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \dots\dots\dots + a_n$

Let $(x - r)$ be a factor of $f(x)$.

$f(x) = (x - r)\, q(x) + R$

where $q(x) = b_0 x^{n-1} + b_1 x^{n-2} + b_2 x^{n-3} + \dots\dots\dots + b_{n-1}$

and R is the remainder

Let r_0 be the initial approximation to r. Using Newton – Raphson method, the close approximation to the root r is given by $r_1 = r_0 - f(r_0)/ f'(r_0)$.

The values of f (r_0) and $f^!$ (r_0) are calculated by synthetic division formula in Brge – Vieta method.

Substituting f (x) and q (x) in the equation f (x) = (x - r) q (x) + R and comparing the coefficients of like power of x on both sides we get

$$b_0 = a_0$$

$$b_1 = a_1 + r_0 a_0$$

$$b_2 = a_2 + r_0 b_1$$

$$R = b_n = a_n + r_0 b_{n-1}$$

In the synthetic division method,

$f(r_0) = R = b_n$ and $C_i = b_i + r_0 C_{i-1}$ where $C_0 = b_0$ and

$f^!(r_0) = dR / dr_0 = C_{n-1}$

Substituting in the formula $r_1 = r_0 - f(r_0)/ f^!(r_0)$

we get $r_1 = r_0 - b_n / C_{n-1}$

Synthetic division

$$\begin{array}{lccccccc}
 & a_0 & a_1 & a_2 & \ldots\ldots\ldots.. & a_{n-2} & a_{n-1} & an \\
r_0 & & r_0b_0 & r_0b_1 & \ldots\ldots\ldots.. & r_0b_{n-3} & r_0b_{n-2} & r_0b_{n-1} \\
\hline
 & b_0 & b_1 & b_2 & \ldots\ldots\ldots.. & b_{n-2} & b_{n-1} & \boxed{bn} \\
r_0 & & r_0c_0 & r_0c_1 & \ldots\ldots\ldots.. & r_0c_{n-3} & r_0c_{n-2} & \ldots\ldots \\
\hline
 & c_0 & c_1 & c_2 & \ldots\ldots\ldots.. & c_{n-2} & \boxed{c_{n-1}} & \ldots\ldots..
\end{array}$$

Using the formula $r_1 = r_0 - b_n / C_{n-1}$ the approximate root r_1 is calculated.

Replacing the value of r_1 in r_0 in the above synthetic division method the next approximated value $r_2 = r_1 - b_n / C_{n-1}$ can be obtained.

Problem:

1**. Find the root of the equation $x^4 + 2x^3 - 21x^2 - 22x + 40 = 0$ using Birge – Vieta method. Perform two iterative. Take initial root as 3.5.**

Solution:

Here	$a_0 = 1$	$a_1 = 2$	$a_2 = -21$	$a_3 = -22$	$a_4 = 40$
	$r_o = 3.5$				

	1	2	-21	-22	40
3.5		3.5	19.25	-6.125	-98.4375
	1	5.5	-1.75	-28.125	-58.4375 (= b_n)
3.5		3.5	31.5	104.125	
	1	9	29.75	76 (=c_{n-1})	

$r_1 = r_0 - b_n / C_{n-1}$ = 3.5 + (58.4375 / 76) =4.2689

Second iteration

	1	2	-21	-22	40
4.2689		4.2689	26.7613	24.5944	11.0752
	1	6.2689	5.7613	2.5944	51.0752 (= b_n)
3.5		4.2689	44.9848	216.63	
	1	10.5378	50.7461	219.224 (= c_{n-1})	

$r_2 = r_1 - b_n / C_{n-1}$ = 4.2689 - (51.0752 / 219.224) = 4.0359

The root of the given equation is 4.0359

2. **Find the root of the equation $x^3 + 2x^2 + 10x - 20 = 0$ using Birge – Vieta method. Take initial root as 1. Perform two iteations.**

Solution:

Here	$a_0 = 1$	$a_1 = 2$	$a_2 = 10$	$a_3 = -20$
	$r_o = 1$			
	1	2	10	-20
1		1	3	13
	------------	------------	------------	------------
	1	3	13	-7 (= b_n)
1		1	4	17
	------------	------------	------------	------------
	1	4	17 (= c_{n-1})	

$r_1 = r_0 - b_n / C_{n-1} =$ 1 + (7 / 17) = 1.4118

The Second iteration is

	1	2	10	-20
1.4118		1.4118	4.8168	20.9184
	------------	------------	------------	------------
	1	3.4118	14.8168	0.9184 (= b_n)
1.4118		1.4118	6.8100	
	------------	------------	------------	------------
	1	4.8236	21.6268 (=	c_{n-1})

$r_2 = r_1 - (b_n / C_{n-1}) = 1.4118 – (0.9184 / 21.6268) = 1.3693$

The root of the given equation is 1.3693.

UNIT II

Numerical Differentiation and Integration

Numerical differentiation is a process by which one can evaluate the approximate numerical value of the derivative of a function at some assigned value of the independent variable, by using the set of given values of that function. To solve the problem of differentiation, we first approximate the function by an interpolation formula and then differentiate it as many times as .....

In case the given argument values are equally spaced, we represent the function by Newton-Gregory formula. If the derivative of the function is to be evaluated at a point near the beginning of the tabular values, we use Newton-Gregory forward formula. If the derivative of the function is to be evaluated at a point near the end of the tabular values, we use Newton- Gregory backward formula.

Newton's forward difference formula to find numerical differentiation

We are given with (n+1) ordered pairs (x_i, $\mathbf{y_i}$) $i = 0, 1, 2, \ldots n$. We want to find the derivative of $y = f(x)$ passing through the (n+1) points, at a point nearer to the starting value $x = x_0$.

Newton's forward difference interpolation formula is

$$y(x_0 + uh) = y_u = y_0 + u\,\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 + \ldots \quad (1)$$

Where y (x) is a polynomial of degree n in x and $u = (x - x_0)/h$

Differentiating y (x) w.r.t. x,

$$\begin{aligned} dy/dx &= (dy/du)(du/dx) = 1/h\ (dy/du) \\ &= 1/h\{ [\,\Delta y_0 + [(2u-1)/2]\,\Delta^2 y_0 + [(3u^2 - 6u + 2)/6]\,\Delta^3 y_0 + \\ &\qquad [(4u^3 - 18u^2 + 22u - 6)/24]\,\Delta^4 y_0 + \ldots\} \quad (2) \end{aligned}$$

when $x = x_0$, i.e., $u = 0$, equation (2) can be written as

$(dy/dx)_{x=x0}$ =, $(dy/dx)_{u=0}$

$$= \frac{1}{h}\left[\Delta y_0 - \frac{1}{2}\Delta^2 y_0 + \frac{1}{3}\Delta^3 y_0 - \frac{1}{4}\Delta^4 y_0 + \dots\right] \quad \dots (3)$$

Differentiating (2) again w.r.t. x

$$d^2y/dx^2 = d/du\,(dy/dx) .(du/dx)$$

$$= d/du\,(dy/dx) . 1/h$$

$$= 1/h^2 \left[\Delta^2 y_0 + (u-1)\, \Delta^3 y_0 + (6u^2 - 18u + 11)/12\, \Delta^4 y_0 + \dots \quad (4)\right.$$

Equation (4) gives the second derivative value of y with respect to x.

Setting $x = x_0$ i.e., $u = 0$ in (4)

$$(d^2y/dx^2)_{x=x0} = 1/h^2 \left[\Delta^2 y_0 - \Delta^3 y_0 + (11/12)\, \Delta^4 y_0 + \dots \right] \quad \dots (5)$$

Therefore, the Newton's forward difference formula to evaluate numerical differentiation is

$$(dy/dx)_{x=x0} = 1/h \left[\Delta y_0 - 1/2\, \Delta^2 y_0 + 1/3\, \Delta^3 y_0 - \dots\right]$$

$$(d^2y/dx^2)_{x=x0} = 1/h^2 \left[\Delta^2 y_0 - \Delta^3 y_0 + 11/12\, \Delta^4 y_0 + \dots \right]$$

Newton's backward difference formula for numerical differentiation

Newton's backward difference interpolation formula is

$$y (x) = y (x_n + vh)$$

$$= y_n + v \nabla y_n + \frac{v (v+1)}{2!} \nabla^2 y_n + \frac{v (v+1) (v+2)}{3!} \nabla^3 y_n + \dots \quad (6)$$

Here $v = x - x_n / h$

Differentiate (6) w.r.t. x

$$dy/dx = (dy/dv) (dv/dx) = (dy/dv) (1/h)$$

$$dy/dx = 1/h \left[\nabla y_n + \frac{2v+1}{2} \nabla^2 y_n + \frac{3 v^2 + 6 v + 2}{6} \nabla^3 y_n + \frac{4 v^3 + 18 v^2 + 22 v + 6}{24} \nabla^4 y_n + \dots \right] \quad \dots (7)$$

$$d^2y/dx^2 = 1/h^2 \left[\nabla^2 y_n + (v+1)\, \nabla^3 y_n + \frac{6 v^2 + 18 v + 11v}{12} \nabla^4 y_n + \dots \right] \quad \dots (8)$$

Equations (7) and (8) give the first second derivative at general x.

Setting x = x_n or v = 0 in (7) and (8) we get

$$dy/dx = 1/h\,[\nabla y_n + 1/2\ \nabla^2 y_n + 1/3\,\nabla^3 y_n + 1/4\ \nabla^4 y_n + \ldots] \ \ldots (9)$$

$$d^2y/dx^2 = 1/h^2\,[\nabla^2 y_n + \nabla^3 y_n + 11/12\ \nabla^4 y_n + \ldots\ ] \ \ldots (10)$$

Equations (9) and (10) are the Newton's backward difference formulae to evaluate numerical differentiation.

Problems

1. Find the first two derivatives of $(x)^{1/3}$ at x = 50 and y = 56 given in the table below

x	50	51	52	53	54	55	56
$y = x^{1/3}$	3.6840	3.7084	3.7325	3.7563	3.7798	3.8030	3.8259

Solution:

Since we require $f^!(x)$ at x = 50, we use Newton's forward difference formula and to get $f^!(x)$ at x= 56 we have to use Newton's backward difference formula

Difference Table

x	y	Δ y	Δ^2 y	Δ^3 y
50	3.6840			
		0.0244		
51	3.7084		- 0.0003	
		0.0241		0
52	3.7325		- 0.0003	
		0.0238		0
53	3.7563		- 0.0003	
		0.0235		0
54	3.7798		- 0.0003	
		0.0232		0
55	3.8030		- 0.0003	
		0.0229		
56	3.8259			

By Newton's forward difference formula,

$$\left(\frac{dy}{dx}\right)_{x=x_0} = \left(\frac{dy}{dx}\right)_{u=0}$$

$= 1/h\,[\,\Delta y_0 - 1/2\,\Delta^2 y_0 + 1/3\,\Delta^3 y_0 \;\ldots\,]$

$= 1/1\,[\,0.0244 - 1/2\,(-0.0003) + 1/3\,(0)\,]$

$= \mathbf{0.02455}$

$$\left(\frac{d^2y}{d^2x}\right)_{x=50} = 1/h^2\,[\,\Delta^2 y_0 - \Delta^3 y_0 + \ldots\,]$$

$= 1\,[\,-0.0003\,]$

$= \mathbf{-0.0003}$

By Newton's backward difference formula,

$$\left(\frac{dy}{dx}\right)_{x=x_n} = \left(\frac{dy}{dx}\right)_{v=v_0}$$

$= 1/h\,[\,\nabla y_n - 1/2\,\nabla^2 y_n + 1/3\,\nabla^3 y_n \;\ldots\,]$

$= 1/1\,[\,0.0229 + 1/2\,(-0.0003) + 1/3\,(0)\,]$

$= \mathbf{0.02275}$

$$\left(\frac{d^2y}{d^2x}\right)_{x=56} = 1/h^2\,[\,\nabla^2 y_n + \nabla^3 y_n + \ldots\,]$$

$= 1\,[\,-0.0003\,]$

$= \mathbf{-0.0003}$

2. The population of a town is given below. Find the rate of growth of the population in the years 1931, 1941, 1961 and1971.

Year x	1931	1941	1951	1961	1971
Population in thousands y	40.62	60.80	79.95	103.56	132.65

Solution

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$
1931	40.62				
		20.18			
1941	60.80		-1.03		
		19.15		5.49	
1951	79.95		4.46		-4.47
		23.61		1.02	
1961	103.56		5.48		
		29.09			
1971	132.65				

(i) To get f^1 (1931) and f^1(1941) we use forward formula.

$x_0 = 1931, \quad x_1 = 1941, \ \ldots.$

$u = x - x_0 / h$ Here $x_0 = 1931$ Corresponding to $u = 0$

By Newton's forward difference formula,

$$\left(\frac{dy}{dx}\right)_{x=1931} = \left(\frac{dy}{dx}\right)_{u=0}$$

$$= 1/h \,[\, \Delta y_0 - 1/2\, \Delta^2 y_0 + 1/3\, \Delta^3 y_0 - 1/4\, \Delta^4 y_0 + \ldots\,]$$

$$= 1/10 \,[\, (20.18) - 1/2(-1.03) + 1/3(5.49) - 1/4(-4.47)\,]$$

$$= 1/10 \,[\, 20.18 + 0.515 + 1.83 + 1.1175\,]$$

$$= \mathbf{2.36425}$$

(ii) when $x = 1941$, we get $u = x - x_0 / h = (1941 - 1931)/10 = 1$

Putting $u = 1$ in the formula given below

$$\frac{dy}{dx} = 1/h \,[\, \Delta y_0 + [(2u-1)/2]\, \Delta^2 y_0 + [(3u^2 - 6u + 2)/6]\, \Delta^3 y_0 +$$
$$(4u^3 - 18u^2 + 22u - 6)/24]\, \Delta^4 y_0 + \ldots\ldots$$

We get

$$\left(\frac{dy}{dx}\right)_{u=1} = 1/10\,[\,(20.18) + 1/2(-1.03) - 1/6(5.49) + 1/12(-4.47)]$$

$$= 1/10\,[\,20.18 - 0.515 - 0.915 - 0.3725]$$

$$= \mathbf{1.83775}$$

(iii) To get f! (1971), we use the formula

$$\left(\frac{dy}{dx}\right)_{x=x0} = 1/h\,[\nabla y_n + 1/2\ \nabla^2 y_n + 1/3\ \nabla^3 y_n + 1/4\ \nabla^4 y_n + \ldots]$$

$$\left(\frac{dy}{dx}\right)_{1971} = 1/10\,[\,29.09 + 1/2(\,5.48) + 1/3(1.02) + 1/4\,(-4.47)\,]$$

$$= 1/10\,[31.10525]$$

$$= \mathbf{3.10525}$$

(iv) To get f! (1961), we use

$$v = x - x_n /h = (1961 - 1971)/10 = -1$$

$$\left(\frac{dy}{dx}\right)_{x=1961} = 1/h\,[\nabla y_n + \frac{2v+1}{2}\ \nabla^2 y_n + \frac{3\,v^2 + 6\,v + 2}{6}\ \nabla^3 y_n + \ldots]$$

$$= 1/10\,[\,29.09 - 1/2(\,5.48) - 1/6(1.02) - 1/4\,(-4.47)\,]$$

$$= 1/10\,[29.09 - 2.74 - 0.17 + 0.3725\,]$$

$$= \mathbf{2.65525}$$

3. Find the first and second derivatives of the function tabulated below at the point x = 1.5

x	1.5	2.0	2.5	3.0	3.5	4.0
f (x)	3.375	7.0	13.625	24.0	38.875	59.0

Solution

The difference table is as follows:

x	y = f(x)	Δ y	Δ^2 y	Δ^3 y	Δ^4 y
1.5 (x_0)	3.375 (y_0)				
		3.625 =Δ y_0			
2.0	7.0		3.0 =Δ^2 y_0		
		6.625		0.75 Δ^3 y_0	
2.5	13.625		3.75		0 =Δ^4 y_0
		10.375		0.75	
3.0	24.0		4.5		0
		14.875		0.75	
3.5	38.875		5.25		
		20.125			
4.0	59.0				

Here we have to find the derivative at the point x = 1.5 which is the initial value of the table. Therefore by Newton's forward difference formula for derivatives at

$x = x_0$, we have

$f^{!}(x_0) \quad = \quad 1/h\,[\Delta y_0 - 1/2\,\Delta^2 y_0 + 1/3\,\Delta^3 y_0 - \ \ldots]$

Here $x_0 = 1.5$, $h = 0.5$

$f^{!}(1.5) \quad = \quad (1/0.5)\,[\,(3.625) - 1/2\,(3.0) + 1/3\,(0.75)\,]$

$f^{!}(1.5) \quad = \quad 4.75$

At the point $x = x_0$,

$f^{!!}(x_0) \quad = \quad 1/h^2\,[\,\Delta^2 y_0 - \Delta^3 y_0 + 11/12\,\Delta^4 y_0 + \ldots\,]$

Here $x_0 \quad = \quad 1.5, \quad h \quad = \quad 0.5$

$f^{!!}(1.5) \quad = \quad [1/(0.5)^2]\,[\,(3.0) - (0.75)\,]$

$f^{!}(1.5) \quad = \quad 9.0$

Numerical Integration

Introduction:

We know that $\int_a^b f(x)dx$ represents the area between y = f (x), x-axis and the ordinates x = a and x = b. This integration is possible only if the function is explicitly given and if it is integrable. We can replace f (x) by an interpolating polynomial P_n (x) and obtain $\int Pn(x)$ dx which is approximately taken as the value n

for $\int_{x0}^{xn} f(x)dx$.

Newton-cote's formula for Numerical Integration

For equally spaced intervals, we have Newton's forward difference formula as

$$f(x) = y(x_0 + uh) = y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 + \ldots\ldots \qquad (1)$$

f (x) can be replaced by Newton's iinterpolating formula in integration.

Here, $u = (x - x_0)/h$ where h is iinterval of differencing

Since $x_n = x_0 + nh$ and $u = (x - x_0)/h$, we have $(x - x_0)/h = n = u$

$$\int_{x0}^{xn} f(x)dx = \int_{x0}^{x0+nh} f(x)dx$$

$$= \int_{x0}^{x0+nh} P_n(x)\,dx,$$ where P_n (x) is the iinterpolating polynomial of degree n

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

Since dx = hdu, and when x = x_0, the lower limit u = 0 and when x = x_0 + nh, the upper limit u = n

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

$$= h\left\{ y_0(u) + (u^2/2)\Delta y_0 + 1/2\left[(u^3/3) - (u^2/2)\right]\Delta^2 y_0 + 1/6\left[(u^4/4) - u^3 + u^2\right]\Delta^3 y_0 + \ldots \right\}$$

$$\int_{x0}^{xn} f(x)dx = h\left\{[\ n y_0 + (n^2/2)\Delta y_0 + 1/2\left[(n^3/3) - (n^2/2)\right]\Delta^2 y_0 + 1/6\left[(n^4/4) - n^3 + n^2\right]\Delta^3 y_0 + \ldots \right\} \qquad \ldots (2)$$

Equation (2) is called as Newton-Cote's quadrature formula or in general quadrature formula. Various values for n yield number of special formulae.

Trapezoidal rule

When n = 1, in the quadrature formula (i.e. there are only two paired values and the interpolating polynomial is linear) we get

$\int_{x0}^{x0+h} f(x)dx = h[\,y_0 + 1/2\ \Delta\, y_0\,]$ since other differences do not exist when n=1

$= h[\,y_0 + 1/2\,(y_1 - y_0)]$

$= h/2\,(\,y_0 + y_1\,)$

$$\int_{x0}^{xn} f(x)dx = \int_{x0}^{x0+nh} f(x)dx$$

$$= \int_{x0}^{x0+h} f(x)dx + \int_{x0}^{x0+2h} f(x)dx + \ldots + \int_{x0}^{x0+nh} f(x)dx$$

$$= h/2\,(\,y_0 + y_1\,) + h/2\,(\,y_1 + y_2\,) + \ldots + h/2\,(\,y_{n-1} + y_n\,)$$

$$= h/2\,[\,(\,y_0 + y_n\,) + 2\,(\,y_1 + y_2 + y_3 + \ldots + y_{n-1}\,)]$$

= h/2 [(sum of the first and last ordinates) + 2(sum of the remaining ordinates)]

This is known as Trapezoidal Rule.

This method is very simple for calculation purposes of numerical integration. The error in this case is significant. The accuracy of the result can be improved by increasing the number of intervals and decreasing the value of h.

Truncation error in Trapezoidal rule

In the neighborhood of $x = x_0$, we can expand $y = f(x)$ by Taylor series in powers of $x - x_0$. That is,

$$y(x) = y_0 + \frac{(x - x_0)}{1!} y_0' + \frac{(x - x_0)^2}{2!} y_0'' + \ldots \qquad \ldots \qquad (1)$$

Where $y_0' \quad = \quad [\,y'(x)\,]_{x=x0}$

$$\int_{x0}^{x1} y\,dx \quad = \quad \int_{x0}^{x1} \left[\, y_0 + \frac{(x - x_0)}{1!} y_0' + \frac{(x - x_0)^2}{2!} y_0'' + \ldots\ldots \right] dx$$

$$= \left[\, y_0 x + (\overline{x - x}_0)\, y_0' + (x - x_0)^2\, y_0'' + \ldots\ldots \right]$$ with upper limit x_1 and lower limit x_0

$$= y_0\,(x_1 - x_0) + \frac{(x_1 - x_0)^2}{2!} y_0' + \frac{(x_1 - x_0)^3}{3!} y_0'' + \ldots.$$

$$= h\,y_0 + (h^2/2!)\, y_0' + (h^3/3!)\, y_0'' + \ldots \qquad \ldots \qquad (2)$$

where h is the equal interval length.

Also $\int_{x0}^{x1} y\,dx \quad = \quad h/2\,(y_0 + y_1) =$ area of the first trapezium $= A_0 \quad \ldots$ (3)

Putting $x = x_1$ in (1)

$$y(x_1) = y_1 = y_0 + \frac{(x_1 - x_0)}{1!} y_0' + \frac{(x_1 - x_0)^2}{2!} y_0'' + \ldots\ldots$$

i.e. $\quad y_1 \quad = \quad y_0 + (h/1!)\, y_0' + (h^2/2!)\, y_0'' + \ldots \qquad \ldots \qquad (4)$

$A_0 \quad = \quad h/2\,[\, y_0 + y_0 + (h/1!)\, y_0' + (h^2/2!)\, y_0'' + \ldots\,]$ using (4) in (3)

$\quad = \quad h\,y_0 + (h^2/2)\, y_0' + (h^3/\,2 \times 2!)\, y_0'' + \ldots$

Subtracting A_0 value from (2)

$$\int_{x0}^{x1} y\,dx - A_0 = h^3 y_0''\,[\,(1/3!) - (1/2 \times 2!)\,] + \ldots.$$

$$= -(1/12)\, h^3 y_0'' + \ldots.$$

Therefore the error in the first interval (x_0, x_1) is $-(1/12)\,h^3 y_0''$ (neglecting other terms)

Similarly the error in the ith interval $= -(1/12)\,h^3 y_{i-1}''$

Therefore, the total error is $E = -1/12\, h^3\,(y_0'' + y_1'' + y_2'' + \ldots. + y_{n-1}'')$

$|E| < nh^3/12 \,.\, M$, where M is the maximum value of

$|y_0''| \quad |y_1''| \quad |y_2''| \quad \ldots..$

$|E| < (b - a)\, h^2/\,12 \,.\, M$

If the iinterval is (a, b) and h = (b – a) /n

Hence, the error in the trapezoidal rule is of the order h^2

Simpson's one-third rule

Setting n = 2 in Newton-cote's quadrature formula, we have

$\int_{x0}^{x0+h} f(x)dx$ $= h [2 y_0 + 4/2 \Delta y_0 + 1/2(8/3 – 4/2) \Delta^2 y_0]$

(since other terms vanish)

$= h [2y_0 + 2 (y_1 – y_0) + 1/3 (E – 1)^2 y_0]$

$= h [2y_0 + 2y_1 – 2y_0 + 1/3 (y_2 – 2y_1 + y_0)]$

$= h [1/3 y_2 + 4/3 y_1 + 1/3 y_0]$

$= h/3 [y_2 + 4y_1 + y_0]$

Similarly, $\int_{x2}^{x4} f(x)dx = h/3 [y_2 + 4y_3+ y_4]$

$\int_{xi}^{xi+2} f(x)dx = h/3 [y_i + 4y_{i+1}+ y_{i+2}]$

If n is an even integer, last integral will be

$\int_{xn}^{xn+2} f(x)dx = h/3 [y_{n-2} + 4y_{n-1}+ y_n]$

Adding all these integrals, if n is an even positive integer, that is, the number of ordinates $y_0, y_1, \ldots. Y_n$ is odd, we have

$\int_{x0}^{xn} f(x)dx = \int_{x0}^{x2} f(x)dx + \int_{x2}^{x4} f(x)dx + \ldots. + \int_{xn-2}^{xn} f(x)dx$

$= h/3 [(y_0 + 4y_1 + y_2) + (y_2 + 4y_3 + y_4) + \ldots + (y_{n-2} + 4y_{n-1} + y_n)]$

$= h/3 [(y_0 + y_n) + 2 (y_2 + y_4 + \ldots) + 4 (y_1 + y_3 + \ldots)]$

= h/3 [sum of the first and last ordinates

+ 2 (sum of remaining odd ordinates)

+ 4 (sum of the even ordinates)]

Note: Though y_2 has suffix even, it is the third ordinate (odd)

Simpson's three-eighths rule

Putting $n = 3$ in Newton-cote's formula we get

$$\int_{x0}^{x3} f(x)dx = h\,[\,3\,y_0 + 9/2\,\Delta\,y_0 + 1/2(\,9/2)\,\Delta^2\,y_0 + 1/6(\,81/4 - 27 + 9)\,\Delta^3 y_0\,]$$

$$= h\,[\,3\,y_0 + 9/2\,(\,y_1 - y_0\,) + 9/4(\,E - 1\,)^2\,y_0 + 3/8(\,E - 1)^3\,y_0\,]$$

$$= h\,[\,3\,y_0 + (9/2)\,y_1 - (9/2)\,y_0 + 9/4\,(\,y_2 - 2y_1 + y_0) +$$
$$3/8\,(\,y_3 - 3y_2 + 3y_1 - y_0\,)]$$

$$= \quad 3h/8\,[\,y_3 + 3y_2 + 3y_1 + y_0\,]$$

If n is a multiple of 3,

$$\int_{x0}^{x0+nh} f(x)dx = \int_{x0}^{x0+3h} f(x)dx + \int_{x0+3h}^{x0+6h} f(x)dx + \ldots. + \int_{x0+(n-3)h}^{x0+nh} f(x)dx$$

$$= 3h/8\,[\,(y_0 + 3y_1 + 3y_2 + y_3) + (y_3 + 3y_4 + 3y_5 + y_6) + \ldots..$$
$$+ (y_{n-3} + 3y_{n-2} + 3y_{n-1} + y_n)\,]$$

$$= 3h/8\,[\,(\,y_0 + y_n\,) + 3\,(\,y_1 + y_2 + y_4 + y_5 + \ldots. + y_{n-1}\,)$$
$$+ 2\,(\,y_3 + y_6 + y_9 + \ldots + y_n\,)\,]$$

The above equation is called Simpson's three-eighths rule which is applicable only when n is a multiple of 3.

Problems

1. Evaluate $\int_{-3}^{+3} x^4\,dx$ by using (1) Trapezoidal rule (2) Simpson's rule. Verify your results by actual integration.

Solution

Here $y(x) = x^4$. Interval length $(\,b - a\,) = 6$, so, we divide equal intervals with $h = 6/6 = 1$, we form below the table

x	-3	-2	-1	0	1	2	3
y	81	16	1	0	1	16	81

By Trapezoidal rule,

$\int_{-3}^{3} y\,dx$ = h/2 [(sum of the first and last ordinates) + 2(sum of the remaining ordinates)]

= 1/2 [(81 + 81) + 2 (16 + 1 + 0 + 1 + 16)]

= 115

By Simpson's one – third rule (since number of ordinate is odd)

$\int_{-3}^{3} y\,dx$ = 1/3 [(81 + 81) + 2 (1 + 1) + 4 (16 + 0 + 16)]

= 98

Since n = 6, (multiple of three), we can use Simpson's three-eighths rule. By this rule

$\int_{-3}^{3} y\,dx$ = 3/8 [(81 + 81) + 3 (16 + 1 + 1 + 16) + 2 (0)]

= 99

By actual integration

$\int_{-3}^{3}$ x^4 dx = 2 x $\left[x^5/5\right]$ with upper limit 3 and lower limit 0.

= 2 x 243 / 5

= 97.2

From the results obtained by various methods, we see that Simpson's rule gives Better result than Trapezoidal rule (It is true in general)

2. Evaluate $\int_{0}^{1} dx$ / 1+x^2, using Trapezoidal rule and h = 0.2. Hence obtain an approximate value of π.

Solution

Let y (x) = 1/(1+x^2)

Interval is (1 – 0) = 1 Since the value of y are calculated as points using h = 0.2

x	0	0.2	0.4	0.6	0.8	1.0
y = 1/(1+x^2)	1	0.96154	0.86207	0.73529	0.60976	0.50000

(1) By Trapezoidal rule

$\int_0^1 dx \,/\, 1+x^2 \quad = \quad h/2\,[\,(\,y_0 + y_n\,) + 2\,(\,y_1 + y_2 + \ldots + y_{n-1})\,]$

$= \quad 0.2/2\,[\,(\,1 + 0.5\,) + 2\,(0.96154 + 0.086207 + 0.73529 + 0.60976\,)$

$= \quad (\,0.1\,)\,[\,1.5 + 6.33732\,]$

$= \quad 0.783732$

By actual integration,

$\int_0^1 dx \,/\, 1+x^2 \quad = \quad (\,\tan^{-1} x\,)^1{}_0 \quad = \quad \pi/4$

$\pi/4 \quad = \quad 0.783732.$

Hence π = (4X0.783732) = 3.13493

3. From the following table, find the area bounded by the f (x) and the x-axis from x = 7.47 to x = 7.52

x	7.47	7.48	7.49	7.50	7.51	7.52
Y= f (x)	1.93	1.95	1.98	2.01	2.03	2.06

Solution

Using Trapezoidal rule we can write the area as the integral of f(x).

$\int_{7.47}^{7.52} f\,(x)dx \quad = \quad 0.01/2\;[(1.93 + 2.06)+2(1.95 + 1.98 + 2.01 + 2.03)\;]$

$= 0.09965$

4. Evaluate $I = \int_0^6 dx \,/\, (1 + x\,)$ using (1) Trapezoidal rule (2) Simpson's rule. Verify your results by actual integration.

Solution

Take the number of intervals as 6. Therefore h = 6 – 0 / 6 = 1

x	0	1	2	3	4	5	6
y=1/(1+x)	1	1/2	1/3	1/4	1/5	1/6	1/7

(1) By Trapezoidal rule,

$\int_0^6 dx/1+x$ = 1/2 [(1 + 1/7) + 2 (1/2 + 1/3 + 1/4 + 1/5 + 1/6)]

= **2.02142857**

(2) By Simpson's one – third rule

I = 1/3 [(1 + 1/7) + 2 (1/3 + 1/5) + 4 (1/2 + 1/4 + 1/6)]

= 1/3 [1 + 1/7 + 16/15 + 22/6

= **1.95873016**

(3) By Simpson's three-eighths rule.

I = 3/8 [(1 + 1/7) + 3 (1/2 + 1/3 + 1/5 + 1/6) + 2 (1/4)]

= **1.96607143**

(4) By actual integration

$\int_0^6 dx/1+x$ = $[\log (1 + x)]_0^6$

= $\log_e 7$

= **1.94591015**

5. Evaluate $\int_0^1 e^{-x2} dx$ by dividing the range of integration into equal parts using Simpson's one-third rule.

Solution

Here the length of the interval is h = 1- 0/4 = 0.25. The values of the function y = e^{-x2} for each point of subdivision are given below

x	0	0.25	0.5	0.75	1
e^{-x2}	1 y_0	0.9394 y_1	0.7788 y_2	0.5698 y_3	0.3678 y_4

By Simpson's rule we have

$$\int_0^1 e^{-x^2}\,dx = h/3\,[\,(y_0 + y_4) + 2\,y_2 + 4\,(y_1 + y_3)\,]$$

$$= 0.25/3\,[\,1.3678 + 1.5576 + 6.0368\,]$$

$$\boldsymbol{\int_0^1 e^{-x^2}\,dx = 0.7468}$$

Gauss Quadrature formula

Carl Frederich Gauss approached the problem of numerical integration in a different way. Instead of finding the area under the given curve, he tried to evaluate the function at some points along with the abscissa. Here the values of abscissa are not equal. Then apply certain weight to the evaluated function.

Thus for Gauss two point formula

$$\int_a^b f(x)dx = \int_{-1}^1 f(t)dt$$

$$= \omega_1 f(t_1) + \omega_2 f(t_2) \qquad \dots \qquad (1)$$

The function f (t) is evaluated at t1 and t_2. ω_1 and ω_2 are the weights given to the two functions.

The basic methodology is explained as given below for **Gauss Two Point Formula.**

Gauss Two Point Formula

First one has to change the interval (a,b) to (-1, 1) by using the following transformation equation

$X = [(a + b)/2] + [(b - a)/2]\,t$

Thus the independent variable 'x' is changed to 't'.

Then we use an interpolation formula which will give the true value of the integral at certain points. Here the interpolation points are t_1 and t_2.

In equation (1), we want to find the four unknown quantities ω_1, ω_2 and t_1 t_2. So we need four algebraic equations to solve it. Let the equation (1) be exact for f (t) = 1

f (t) = t

f (t) = t^2 and

f (t) = t^3

when f (1) = 1 we get

$\int_{-1}^{1} 1dt$ = 2 = $\omega_1 + \omega_2$ … (2)

[since f (t_1) = f (t_2) = 1]

When f (t) = t

$\int_{-1}^{1} tdt$ = $\left[t^2/2 \right]_{-1}^{1}$ = 0 = $\omega_1 t_1 + \omega_2 t_2$ … (3)

When f (t) = t^2 we get

$\int_{-1}^{1} t^2\, dt$ = $\left[t^3/3 \right]_{-1}^{1}$ = 2/3 = $\omega_1 t_1^2 + \omega_2 t_2^2$ … (4)

when f (t) = t^3

$\int_{-1}^{1} t^3\, dt$ = $\left[t^4/4 \right]_{-1}^{1}$ = 0 = $\omega_1 t_1^3 + \omega_2 t_2^3$ … (5)

This set of equations (2), (3), (4) and (5) can be solved as follows

From (3) we get

$\omega_1 t_1$ = $-\omega_2 t_2$ … (6)

From (5) we get

$\omega_1 t_1^3$ = $-\omega_2 t_2^3$ … (7)

From (6) and (7) we get

t_1 = $-t_2$

ω_1 = ω_2 = 1

From (4) we get

$t_1^2 + t_2^2$ = 2/3

t_1 = $1/\sqrt{3}$

$t_2 = -1/\sqrt{3}$

From equation (1) we get

$$I = \int_{-1}^{1} f(t)\,dt = \omega_1 f(t_1) + \omega_2 f(t_2)$$

$$I = f(1/\sqrt{3}) + f(-1/\sqrt{3}) \qquad \ldots \qquad (A)$$

$[\text{since } \omega_1 = \omega_2 = 1]$

Problems

1. Evaluate $\int_1^2 dx/x$ using Gauss two point formula.

Solution

Transform the variable x to t by the transformation

$X = [(a + b)/2] + [(b - a)/2]\, t$

$= [(1 + 2)/2] + [(2 - 1)/2)]\, t$

$X = 3/2 + t/2 = (3 + t)/2$

i.e. $dx = dt/2$

Therefore $I = \int_1^2 dx/x = \int_{-1}^{1} \frac{2}{3+t}\frac{dt}{2}$

$$= \int_{-1}^{1} \frac{dt}{3+t}$$

Here $f(t) = \dfrac{1}{3+t}$

$f(1/\sqrt{3}) = 1/(3 + \sqrt{3}) = 0.2795$

$f(-1/\sqrt{3}) = 1/(3 - \sqrt{3}) = 0.41288$

$I = f(1/\sqrt{3}) + f(-1/\sqrt{3})$

$\mathbf{I = 0.6923}$

2. Evaluate $\int_1^2 dx/$ $(1+x^3)$ using Gauss 2 point formula.

Solution

Transform the variable x to t by the transformation equation

X = [(a + b)/2] + [(b – a)/2] t

X = 3/2 + t/2 = (3 + t) / 2

and dx = dt/2

Therefore I = $\int_1^2 dx/$ $(1+x^3)$

$$= \int_{-1}^{1} \frac{1}{1+((3+t)/2)3} \frac{dt}{2}$$

= 4 $\int_{-1}^{1}$ dt/ 8 + (3+t $)^3$

= 4 [f (1/√3) + f (- 1/√3)]

Here f (t) = 1 / [8 + (3 +t$)^3$]

f (1/√3) = 1/[8 + (3 + 1/√3$)^3$ = 0.0185

f (-1/√3) = 1/[8 + (3 - 1/√3$)^3$ = 0.045

I = 4 [f (1/√3) + f (- 1/√3)]

= 4 [0.0185 + 0.045]

I = 0.254

Gauss three point formula

$\int_a^b f(x)dx$ = $\int_{-1}^{1} f(t)dt$

Where the interval (a, b) is changed into (-1, 1) by the transformation

X = [(b + a)/2] + [(b – a)/2] t

Then

$\int_{-1}^{1} f(t)dt$ = A_1 f (t_1) + A_2 f (t_2) + A_3 f (t_3)

Where

A_1	**=**	**A_3**	**=**	**0.5555**
A_2	**=**	**0.8888**		
t_1	**=**	**- 0.7745**		
t_2	**=**	**0**		
t_3	**=**	**0.7745**		

Problem

1. Evaluate $\int_1^2 dx/x$ using Gauss three point formula.

Solution

Transform the variable x to t by the transformation

$X = [(b + a)/2] + [(b - a)/2]\, t$

$= [(1 + 2)/2] + [(2 - 1)/2]\, t$

$x = 3/2 + t/2 = (3 + t)/2$ and $dx = dt/2$

Therefore $I = \int_1^2 dx/x = \int_{-1}^{1} f(t)\, dt$

$= A_1 f(t_1) + A_2 f(t_2) + A_3 f(t_3)$

$A_1 = A_3 = 0.5555$

$A_2 = 0.8888$

In this problem $f(x) = \frac{1}{x}$ and $f(t) = 1/(3+t)$

$f(t_1) = f(-0.7745)$

$= 1/(3 - 0.7745) = 0.4493$

$f(t_2) = f(0)$

$= 1/3 = 0.3333$

$f(t_3) = f(0.7745)$

$= 1/(3 + 0.7745) = 0.2649$

Substituting the values of A_1, A_2, A_3 and f (t_1), f (t_2), f (t_3) in the formula, we get

$$I = 0.5555 (0.4493) + 0.8888 (0.3333) + 0.5555 (0.2649)$$

$$\mathbf{I = 0.6929}$$

2. Evaluate $\int_{0.2}^{1.5} e^{-x2}\, dx$ by using the three point Gaussian Quadrature formula.

Solution

Transform the variable from x to t by the transformation

$$X = [(b + a)/2] + [(b – a)/2] t$$

where $a = 0.2$ $b = 1.5$

$$= (1.7/2) + (1.3/2) t$$

i.e. $x = (1.7 + 1.3t)/2$ and $dx = 1.3\, t/2 = 0.65\, t$

$$I = \int_{0.2}^{1.5} e^{-x2}\, dx$$

$$= \int_{-1}^{1} e^{-[(1.7 +1.3t)/2]2} (0.65)dt$$

$$= 0.65 \int_{-1}^{1} e^{- [(1.7 +1.3t)/2]2}\, dt$$

Using Gauss three point formula we can write

$$I = 0.65 [A_1 f (t_1) + A_2 f (t_2) + A_3 f (t_3)]$$

Where $f(t) = e^{- [(1.7 +1.3t)/2]2}$

$A_1 = A_3 = 0.5555$

$A_2 = 0.8888$

$f (t_1) = f (- 0.7745) = e^{- [(1.7 +1.3 (-0.7745)/2]2}$

$= 0.8868$

$f (t_2) = f (0) = e^{- [(1.7 +1.3 (0)/2]2}$

$= 0.48555$

$f (t_3) = f (0.7745) = e^{- [(1.7 +1.3 (0.7745)/2]2}$

$= 0.16013$

Substituting the values of A_1 , A_2, A_3 and f (t_1), f (t_2), f (t_3) in the formula, we get

I = 0.5555 (0.8868) + 0.8888 (0.4855) + 0.5555 (0.16013)

= 0.4926 + 0.4315 + 0.08895

I = 1.01307

3. Evaluate $\int_0^1$ dx/ 1+x^2, using Gauss 3 point formula

Solution

Transform the variable from x to t by the transformation

X = [(b + a)/2] + [(b – a)/2] t

where a = 0 b = 1

x = (1/2) + (t/2)

i.e. x = (t + 1)/2 when x = 0, t = -1

x = 1, t = 1

dx = dt/2 = 0. 5t

$$I = \int_0^1 \frac{dx}{(1+x2)}$$

$$= \int_{-1}^{1} [\, 1/ \ (1 + (\,(\,t+1)/2)^{2)}\ ]^* \, dt/2$$

= 2 $\int_{-1}^{1}$ dt/ 4 + (t + 1)^2 using Gauss three point formula we get

I = 2 [A_1 f (t_1) + A_2 f (t_2) + A_3 f (t_3)]

Where f (t) = 1/ 4 + $(t+1)^2$

A_1 = A_3 = 0.5555

A_2 = 0.8888

f (t_1) = f (- 0.7745) = 1/ 4 + (- 0.7745 + 1)^2

= 0.2468

f (t_2) = f (0) = 1/ 4 + 1 = 0.2

$f(t_3) = f(0.7745) = 1/4 + (0.7745 + 1)^2$

$= 0.13988$

Substituting the values of A_1 , A_2, A_3 and $f(t_1)$, $f(t_2)$, $f(t_3)$ in the formula, we get

$I = 2 [0.5555 (0.2468) + 0.8888 (0.2) + 0.5555 (0.13988)]$

$= 2 [0.39256]$

$\mathbf{I = 0.78512}$

Unit III

Solution of first order differential equations – Taylor series method

Initial value problem is an ordinary differential equation given along with a specified initial value of the unknown function at a given point in the domain of the solution. In other words, initial value problem is defined as the problem of finding a Function y of x when we know its derivative and its value y_0 at a particular point x_0.

A first order differential equation is given by $\frac{dy}{dx} = y' = f(x,y)$ with the condition that $y(x_0)=y_0$. The approximate solution of a first order differential equation is given by

$(y_m - y_{m-1}) = f(x_m,y_m)$. (or) $y_m = y_{m-1} + f(x_m,y_m)$ and this method is called as single step method. Taylor series method is one such single step method.

Let us consider a first order differential equation $\frac{dy}{dx} = f(x,y)$ with the initial condition $y(x_0)=y_0$. We can expand $y(x)$ about a point x_0 in Taylor series as

$y(x) = y(x_0) + (x-x_0)/1!\ [y'(x)]_{x0} + (x-x_0)^2/2!\ [y''(x)]_{x0} + (x-x_0)^3/3!\ [y'''(x)]_{x0} + \ldots\ldots$

$= y_0 + (x-x_0)/1!\ y'_0 + (x-x_0)^2/2!\ y''_0 + (x-x_0)^3/3!\ y'''x_0 + \ldots\ldots$

When $x=x_0+h=x_1$ we can write,

$y(x_1) = y_0 + h/1!\ y'_0 + h^2/2!\ y''_0 + h^3/3!\ y'''_0 + \ldots..$

In general we can write $\mathbf{y_{m+1} = y_m + h/1!\ y'_m + h^2/2!\ y''_m + h^3/3!\ y'''_m + \ldots..}$

Problems

1. Solve the initial value problem $\frac{dy}{dx} = x^2 - y$ with the initial condition y(0)=1 by Taylor series method and find y(0.1) and y(0.2).

The formula we have to use is $y_1 = y_0 + h/1!\ y'_0 + h^2/2!\ y''_0 + h^3/3!\ y'''_0 + \ldots$

Here $\frac{dy}{dx} = y' = x^2 - y$ and it is given that y=1 when x=0.

Differentiating $y' = x^2 - y$ wrt x we get $y'' = -y'$

Ie. $y'' = -(x^2 - y) = y - x^2$

Here
$x_0=0$, $y_0=1$ and $h = (x - x_0) = 0.1$

$y'_0 = x_0^2 - y_0 = 0-1 = -1$

$y''_0 = y_0 - x_0^2 = 1-0=1$

$y_1 = y_0 + h/1!\ y'_0 + h^2/2!\ y''_0$

$y(0.1) = y_1 = 1 + (0.1)\,[(-1) + (0.1)^2/2]\,(1) = 1 - 0.1 + \frac{0.01}{2} = 0.905$

Similarly $y_2 = y_1 + h/1!\ y'_1 + h^2/2!\ y''_1$

Here $y_1 = 0.905$, $y'_1 = x_1^2 - y_1 = (0.1)^2 - 0.905 = -0.895$

$y''_1 = y_1 - x_1^2 = 0.905 - (0.1)^2 = 0.895$

Therefore $y2 = 0.905 + (0.1)(-0.895) + \frac{0.001}{2} 0.895 = 0.82$

Answer: $y_1 = 0.905$ and $y_2 = 0.82$

2. Using Taylor Series method solve the initial value problem $y' = 1 + y^2$ with the initial condition y(0) =0 and find the value y(0.2) and y(0.4).

The formula we have to use is $y_1 = y_0 + h/1!\ y'_0 + h^2/2!\ y''_0$

Here $x_0=0$, $y_0=0$, h=0.2 and $x_1=.2$

$y'_0 = 1 + y_0^2 = 1+0 = 1$

$y''_0 = 2\ y_0\ y'_0 = 0$

$y_1 = y(0.2) = 0 + [\ 0.2\ /1!]\ 1 + [\ (0.2)^2/2!\ ]\ 0 = 0.2$

$y_1 = 0.2$

To find y(0.4) let us use the formula $y_2 = y_1 + h\ /1!\ y'_1 + h^2/2!\ y''_1$

Here $y'_1 = 1 + y_1^2 = 1+(0.2)^2 = 1.04$and $y''_1 = 2\ y_1\ y'_1 = 2\ (0.2)\ (1.04) = 0.416$

$y_2 = 0.2+0.2\ (1.04)+[\ (0.2)^2/2]0.416 = 0.4163$

Therefore $y(0.4) = 0.4163$

Euler's Method

Consider the first order differential equation $\frac{dy}{dx} = y' = f(x,y)$ with the initial condition $y(x_0)=y_0$. Let us assume that the intervals x_0, x_1, x_2, x ,.. are equi distant and at each interval the given function is nearly linear. The problem is to find y_1 at the point $x=x_1$ having known the value of y_0 at $x=x_0$. The equation of tangent for the given function

$\frac{dy}{dx} = y' = f(x,y)$ is written as

$$\frac{y - y0}{x - x0} = (dy/dx)_{x=x0} = f(x_0,y_0)$$

Ie. $y-y_0 = (x-x_0)\ f(x_0,y_0)$

$y = y_0 + (x-x_0)\ f(x_0,y_0)$

When $x_1=x_0+h$ and $y=y(x_1)=y_1$ we can write

$y_1 = y_0 + h\ f(x_0,y_0)$ In a similar manner

$y_2 = y_1 + h\ f(x_1,y_1)$ OR in general

$y_{n+1} = y_n + h\ f(x_n,y_n)$ This is the Euler's formula to solve an initial value problem.

Improved Euler's formula

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

Modified Euler's formula

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

Problems

1.Solve the initial value problem $y' = - y$ with the initial condition $y(0)=1$ by Euler's method , improved Euler's method and modified Euler's method. And find the value of y(0.01)

Solution

Given $\frac{dy}{dx} = y' = -y$, $x_0=0$, $h=0.01$, $y_0=1$ and $f(x_0,y_0) = -y_0 = -1$

Euler's formula is $y_1 = y_0 + h\ f(x_0,y_0)$

$y(0.01) =1 +0.01 (-1) =0.99$

Euler's Improved formula is **$y_1 = y_0 + \frac{h}{2}\{ f(x_0,y_0) + f(x_{0+h} , y_0+h\ f(x_0,y_0)) \}$**

$y(0.1)=y_1 = 1+ \frac{0.01}{2}[(-1)+(-0.99)] =1-0.00995 = 0.9901$

Euler's modified formula is $y_1 = y_0 + h \{ f(x_0+ h/2 , y_0+h/2\ f(x_0,y_0)) \}$

$y(0.1) = y_1 = 1+0.01[-1+\frac{0.01}{2}(-1)] = 0.9901$

2 . Solve the differential equation $\frac{dy}{dx}$ = y' = y-x with the initial condition y(0)=2 evaluate y (0.1) by Euler's methods.

Given: $\frac{dy}{dx} = y' = f(x,y) = y - x$

$x_0 = 0$, $y_0 = 2$, $h = 0.1$, $f(x_0,y_0) = y_0 - x_0 = 2 - 0 = 2$

Euler's formula is $y_1 = y_0 + h\ f(x_0,y_0)$

$y(0.1) = 2 + 0.1\ (2) = 2.2$

Euler's Improved formula is $y_1 = y_0 + \frac{h}{2}\{ f(x_0,y_0) + f(x_{0+h}\ , y_0 + h\ f(x_0,y_0)) \}$

$y(0.1) = y_1 = 2 + \frac{0.1}{2}[(2-0)+(2.2 - 0.1)] = 2.21$

Euler's modified formula is $y_1 = y_0 + h \{ f(x_0 + h/2\ , y_0 + h/2\ f(x_0,y_0)) \}$

$y(0.1) = y_1 = 2+0.1[(2+\frac{0.1}{2}(2-0) – (0+\frac{0.1}{2})] = 2.205$

3. Using Euler's method evaluate y(0.2),y(0.4), y(0.6) by solving the equation y' = x + y with the initial condition y(0)=1.

Solution:

Given: $\frac{dy}{dx} = y' = f(x,y) = x + y$

$x_0 = 0$, $y_0 = 1$ $h = 0.2$, $f(x_0,y_0) = y_0 + x_0 = 1 + 0 = 1$

Euler's formula is $y_1 = y_0 + h\ f(x_0,y_0)$

$Y(0.2) = 1+(0.2)(0+1) = 1.2$ Now

$Y_2 = y_1 + h\ f(x_1,y_1)$ ie. $Y(0.4) = 1.2 + (0.2)\ (0.2+1.2) = 1.48$

$Y_3 = y_2 + h\ f(x_2,y_2) = 1.48 + 0.2(0.4+1.48) = 1.856$

FOURTH ORDER RUNGE-KUTTA METHOD

This method is most commonly used in practice. Let $dy/dx = f(x,y)$ be a given differential equation to be solved under the condition $y(x_0) = y_0$. Let h be the length of the interval between equidistant values. The first increment in y is computed using the formulae given below.

$k_1 = h f(x_0, y_0)$
$k_2 = h f(x_0 + h/2, y_0 + k_1/2)$
$k_3 = h f(x_0 + h/2, y_0 + k_2/2)$
$k_4 = h f(x_0 + h, y_0 + k_3)$
$\Delta y = 1/6(k_1 + 2k_2 + 2k_3 + k_4)$

Now $x_1 = x_0 + h$, $y_1 = y_0 + \Delta y$

The increment in y for the second interval is computed in a similar manner by using the formulae given above.

Problem:

1. Find the values of y (1.1) using fourth order Runge-Kutta method, given that $dy/dx = y^2 + xy$ and $y(1) = 1$

Solution:

Given $y' = f(x, y) = y^2 + xy$

Here it is given that $x_0 = 1$, $y_0 = 1$ and let $h = 0.1$

Now $k_1 = h f(x_0, y_0)$

$= h(y_0^2 + x_0 y_0)$

$= (0.1)[1^2 + (1)(1)] \quad = (0.1)(2)$

Therefore $\mathbf{k_1 = 0.2}$

$k_2 = h f(x_0 + h/2, y_0 + k_1/2)$

$k_2 = h[(y_0 + k_1/2)^2 + (x_0 + h/2)(y_0 + k_1/2)]$

$= (0.1)[(1 + 0.2/2)^2 + (1 + 0.1/2)(1 + 0.2/2)]$

$= (0.1)[(1.1)^2 + (1.05)(1.1)]$

$\mathbf{= 0.2365}$

$k_3 = h f(x_0 + h/2, y_0 + k_2/2)$

$k_3 = h[(y_0 + k_2/2)^2 + (x_0 + h/2)(y_0 + k_2/2)]$

$= (0.1)[(1 + 0.2365/2)^2 + (1 + 0.1/2)(1 + 0.2365/2)]$

$= (0.1)[(1.11825)^2 + (1.05)(1.11825)]$

$= (0.1)[2.4246]$

$\mathbf{= 0.24246}$

$k_4 = h f(x_0 + h, y_0 + k_3)$

$k_4 = h[(y_0 + k_3)^2 + (x_0 + h)(y_0 + k_3)]$

$= (0.1)[(1 + 0.24246)^2 + (1 + 0.1)(1 + 0.24246)]$

$= (0.1)[(1.24246)^2 + (1.01)(1.24246)]$

$= (0.1)[1.5437 + 1.366706]$

$= (0.1)[2.9104]$

$\mathbf{= 0.29104}$

$\Delta y = 1/6(k_1 + 2k_2 + 2k_3 + k_4)$

$\Delta y = 1/6[0.2 + 2(0.2365) + 2(0.24246) + 0.29104]$

$= 1/6[1.44896]$

$= 0.24149$

Therefore $y_1 = y_0 + \Delta y$

$= 1 + 0.24149$

$= 1.24149$

$\mathbf{y(1.1) = 1.24149}$

2. Solve the differential equation $y' = y - x$ with the initial condition $y(0.1) = 2.20517$ and find $y(0.2)$ by fourth order Runge-Kutta method.

Solution:

Given: $f(x,y) = y - x$ and $x_0 = 0.1$, $y_0 = 2.20517$ and let $h = 0.1$

Now $k_1 = h f(x_0, y_0) = 0.1(2.20517-0.1) = 0.210517$

$k_2 = h f(x_0 + h/2, y_0 + k_1/2) = h[\ y_0 + k_1/2 - (x_0 + h/2)]$

$= 0.1[\ (92.20517 + 0.2105/2) - (0.1 + 0.1/2)]$

$= 0.1[2.31042-0.15] = 0.21604$

$k_3 = h f(x_0 + h/2, y_0 + k_2/2)$

$= h[\ y_0 + k_2/2 - (x_0 + h/2)]$

$= 0.1[(2.20517 + 0.21604/2) - (0.1 + 0.1/2)]$

$= 0.1[2.31319- 0.15) = 0.21632$

$k_4 = h f(x_0 + h, y_0 + k_3)$

$k_4 = h[\ (y_0 + k_3) - (x_0 + h)]$

$= 0.1[(2.20517+ 0.21632) - (0.1 + 0.1)]$

$= 0.22214$

$\Delta y = 1/6(k_1 + 2k_2 + 2k_3 + k_4)$

$\Delta y = 1/6[0.2105 + 2(0.21604) + 2(0.21632) + 0.22214]$

$= 0.21622$

Therefore $y_1 = y_0 + \Delta y$

$= 2.20517 + 0.21622 = 2.42139$

$Y(0.2) = 2.42139$

****

MILNE'S PREDICTOR CORRECTOR METHOD

The function $y(x_0 + rh)$ can be expanded as

$$y(x_0 + rh) = y_0 + r\Delta y_0 + \frac{r(r-1)}{2!}\Delta^2 y_0 + \frac{r(r-1)(r-2)}{3!}\Delta^3 y_0 + \frac{r(r-1)(r-2)(r-3)}{4!}\Delta^4 y_0 + \ldots$$

Differentiating w.r.t. 'r' we get

$$h y'(x_0 + rh) = \Delta y_0 + [(2r-1)/2]\ \Delta^2 y_0 + [(3r^2 - 6r + 2)/6]\ \Delta^3 y_0 + [(2r^3 - 9r^2 + 11r - 3)/12]\ \Delta^4 y_0 + \ldots \quad (1)$$

Putting $r = 1, 2, 3$ and 4 in (1) we get

$$hy'(x_0 + h) = \Delta y_0 + 1/2\ \Delta^2 y_0 - 1/6\ \Delta^3 y_0 + 1/12\ \Delta^4 y_0 + \ldots \quad (2)$$

$$hy'(x_0 + 2h) = \Delta y_0 + 3/2\ \Delta^2 y_0 + 1/3\ \Delta^3 y_0 - 1/12\ \Delta^4 y_0 + \ldots \quad (3)$$

$$hy'(x_0 + 3h) = \Delta y_0 + 5/2\ \Delta^2 y_0 + 11/6\ \Delta^3 y_0 + 1/4\ \Delta^4 y_0 + \ldots \quad (4)$$

$$hy'(x_0 + 4h) = \Delta y_0 + 7/2\ \Delta^2 y_0 + 13/3\ \Delta^3 y_0 + 25/12\ \Delta^4 y_0 + \ldots \quad (5)$$

Now

$$\left.\begin{aligned} y(x_0 + h) &= y_1 \Rightarrow y'(x_0 + h) = y_1' \\ y(x_0 + 2h) &= y_2 \Rightarrow y'(x_0 + 2h) = y_2' \\ y(x_0 + 3h) &= y_3 \Rightarrow y'(x_0 + 3h) = y_3' \\ y(x_0 + 4h) &= y_4 \Rightarrow y'(x_0 + 4h) = y_4' \end{aligned}\right\} \ldots (6)$$

Substituting (6) in (2), (3), (4) and (5)

$$y_1' = 1/h\ [\Delta y_0 + 1/2\ \Delta^2 y_0 - 1/6\ \Delta^3 y_0 + 1/12\ \Delta^4 y_0 + \ldots] \ldots (7)$$

$$y_2' = 1/h\ [\Delta y_0 + 3/2\ \Delta^2 y_0 + 1/3\ \Delta^3 y_0 - 1/12\ \Delta^4 y_0 + \ldots] \ldots (8)$$

$$y_3' = 1/h\ [\Delta y_0 + 5/2\ \Delta^2 y_0 + 11/6\ \Delta^3 y_0 + 1/4\ \Delta^4 y_0 + \ldots] \ldots (9)$$

$$y_4' = 1/h\ [\Delta y_0 + 7/2\ \Delta^2 y_0 + 13/3\ \Delta^3 y_0 + 25/12\ \Delta^4 y_0 + \ldots] \ldots (10)$$

We know that

$$\left.\begin{aligned} \Delta y_0 &= y_1 - y_0 \\ \Delta^2 y_0 &= y_2 - 2y_1 + y_0 \\ \Delta^3 y_0 &= y_3 - 3y_2 + 3y_1 - y_0 \end{aligned}\right\} \ldots (11)$$

$\Delta^4 y_0 = y_4 - 4y_3 + 6y_2 - 4y_1 + y_0$

Neglecting differences beyond the fourth power in (7) , (8), (9) and (10) and replacing the remaining differences by (11) we get

$y_1{}' = 1/h [(y_1\ y_o) + 1/2 (y_2\ -2y_1 + y_0) - 1/6 (y_3 - 3y_2 + 3y_1 - y_0$

$+ 1/12 (y_4 - 4y_3 + 6y_2 - 4y_1 + y_0)]$

$y_1{}' = 1/12\ h [-3y_0 - 10y_1 + 18y_2 - 6y_3 + y_4]$ … (12)

Similarly we get

$y_2{}' = 1/12\ h [y_0 - 8y_1 + 8y_3 - y_4]$ … (13)

$y_3{}' = 1/12\ h [-y_0 + 6y_1 - 18y_2 + 10y_3 + 3y_4]$ … (14)

$y_4{}' = 1/12\ h [3y_0 - 16y_1 + 36y_2 - 48y_3 + 25y_4]$ … (15)

Now our aim is to find y_4

$2 (y_1{}' + y_3{}') = 2/12h [-4 y_0 - 4 y_1 + 4 y_3 + 4 y_4]$ … (16)

$y_2{}' = 1/12h [y_0 - 8 y_1 + 8 y_3 - y_4]$ … (17)

(16) – (17) => $= 2 y_1{}' + 2 y_3{}' - y_2{}'$

$= 1/12h [- 9 y_0 + 9 y_4]$

$= 9/12h [y_4 - y_0]$

$y_4 - y_0 = 4h/3 [2y_1{}' - y_2{}' + 2 y_3{}']$

(or) $y_4 = y_0 + 4h/3 [2y_1{}' - y_2{}' + 2 y_3{}']$ … (18)

In general we can write (18) as

$y_{n+1} = y_{n-3} + 4h/3 [2y_{n-2}{}' - y_{n-1}{}' + 2 y_n{}']$

This is **Milne's Predictor formula** and is denoted by

$\mathbf{y_{n+1,\ p} = y_{n-3} + 4h/3 [2y_{n-2}{}' - y_{n-1}{}' + 2 y_n{}']}$ **… (19)**

This formula is in general compatible for the step-by-step solution of $y' = f (x, y)$. But, as a precaution against errors of various kinds, it is desirable to have a second, independent formula into which y_{n+1} can be substituted as a check. This formula is called **Milne's corrector formula** and can be obtained as follows.

$y_2{}' + y_4{}' = 1/12h\,[\,4\,y_0 - 24\,y_1 + 36\,y_2 - 40\,y_3 + 24\,y_4\,]$ … (20)

$4\,y_3{}' = 4/12h\,[\,-\,y_0 + 6\,y_1 - 18\,y_2 + 10\,y_3 + 3\,y_4\,]$ … (21)

(20) + (21) =>

$y_2{}' + 4\,y_3{}' + y_4{}' = 1/12h\,[\,36\,y_4 - 36\,y_2\,]$

$= 3h\,[\,y_4 - y_2\,]$

i.e., $y_4 - y_2 = h/3\,[y_2{}' + 4\,y_3{}' + y_4{}'\,]$

$\mathbf{y_4 = y_2 + h/3\,[y_2{}' + 4\,y_3{}' + y_4{}']}$

This formula is called **Milne's corrector formula** and can be written in general as

$\mathbf{y_{n+1,\,c} = y_{n-1} + h/3\,[y_{n-1}{}' + 4\,y_n{}' + y_{n+1}{}']}$

If the initial four values are not given, we can obtain these values either by using Taylor series method or by Runge-Kutta method.

Problem:

1. The differential equation $dy/dx = y - x^2$ is satisfied by $y\,(0) = 1$, $y\,(0.2) = 1.12186$, $y\,(0.4) = 1.46820$, $y\,(0.6) = 1.7379$. Compute the value of $y\,(0.8)$ by Milne's Predictor – Corrector Method.

Solution:

Given $dy/dx = y' = y - x^2$ and $h = 0.2$

x_0	= 0		y_0	= 1
x_1	= 0.2		y_1	= 1.12186
x_2	= 0.4		y_2	= 1.46820
x_3	= 0.6		y_3	= 1.7379
x_4	= 0.8		y_4	= ?

By **Milne"s Predictor formula** we have

$y_{n+1},\,p = y_{n-3} + 4h/3\,[\,2y_{n-2}{}' - y_{n-1}{}' + 2\,y_n{}'\,]$ … (1)

To get y_4, put $n = 3$ in (1) we get

$y_4, p \quad = \quad y_0 + 4h/3\ [\ 2y_1' - y_2' + 2\ y_3'\ ]$ … (2)

Now $y_1 \quad = \quad (\ y - {}'x^2)_1 \quad = \quad y_1 - x_1^{\ 2}$

$= \quad 1.12186\ -\ (0.2)^2 \quad = \quad 1.08186$ … (3)

$Y_2 \quad = \quad (\ y - {}'x^2)_2 \quad = \quad y_2 - x_2^{\ 2}$

$= \quad 1.46820\ -\ (0.4)^2 \quad = \quad 1.3082$ … (4)

$y_3 \quad = \quad (\ y - {}'x^2)_3 \quad = \quad y_3 - x_3^{\ 2}$

$= \quad 1.7379\ -\ (0.6)^2 \quad = \quad 1.3779$ … (5)

Substituting (3) , (4) and (5) in (2) we get

$y_4, p \quad = \quad 1 + 4\ (0.2)/3\ [\ 2\ (1.08186)\ - 1.3082 + 2\ (1.3779)\ ]$

$= \quad 1 + 0.266\ [\ 2.1637 - 1.3082 + 2.7558\ ]$

$= \quad 1.9630187$

Therefore y (0.8) = 1.9630187 (By Milne's Predictor Formula)

By Milne's Corrector Formula we have

$y_{n+1}, c \quad = \quad y_{n-1} + h/3\ [y_{n-1}' + 4\ y_n' + y_{n+1}\ ']$

To get y_4 , put $n = 3$, we get

$y_4, c \quad = \quad y_2 + h/3\ [y_2' + 4\ y_3' + y_4\ ']$ … (6)

Now $y_4' \quad = \quad (\ y - x^2\ )_4 \quad = \quad y_4 - x_4^{\ 2}$

$= \quad 1.96301 - (0.8)^2$

$= \quad 1.3230187$ … (7)

Substituting (4) , (5) and (7) in (6) we get

$y_4, c \quad = \quad 1.46820 + (0.2)/3[\ 1.3082 + 4\ (1.3779) + 1.3230187\ ]$

$= \quad 2.0110546$

y (0.8) = 2.0110546 (By Milne's Corrector Formula)

Problem

2. Using Taylors series method, solve $dy/dx = xy + y^2$, $y(0) = 1$ at $x = 0.1$, 0.2 and 0.3 continue the solution at $x = 0.4$ by Milne's predictor corrector method.

Solution:

Given $y' = xy + y^2$ and $x_0 = 0$ $y_0 = 1$ and $h = 0.1$

Now $y' = xy + y^2$

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

$y''' = xy'' + 2y' + 2yy'' + 2y'^2$

Since the values of y's are not given directly we can find them by using Taylors method as given below

To find y (0.1)

By Taylors series we have

$y(0.1) = y_1 = y_0 + hy_0 + (h^2/2!)\, y_0'' + (h^3/3!)\, y_0''' + \dots$ … (1)

$y_0' = (xy + y^2)_0 = (x_0y_0 + y_0^2) = 1$ … (2)

$y_0'' = (xy' + y + 2yy')_0$

$= (x_0y_0' + y_0 + 2y_0y_0') = 3$ … (3)

$y_0''' = (xy'' + 2y' + 2yy'' + 2y'^2)_0 = 10$ … (4)

Substituting (2) , (3) and (4) in (1) we get

$y(0.1) = 1 + (0.1) + [(0.1)^2/2]\, 3 + [(0.1)^3/6]\, 10$

$= 1 + 0.1 + 0.015 + 0.001666$

Therefore y (0.1) = 1.11666

To find y (0.2)

By Taylors series we have

$y_2 = y_1 + hy_1 + (h^2/2!)\, y_1'' + (h^3/3!)\, y_1''' + \dots$ … (5)

Now $y_1' = (xy + y^2) = x_1y_1 + y_1^2$

$= (0.1)(1.11666) + (1.11666)^2$

= 0.111666 + 1.2469

= 1.3585 … (6)

y_1^{I} = ($xy^{I} + y + 2\,yy^{I}$)

= $x_1y_1^{I} + y_1 + 2\,y_1y_1^{I}$

= (0.1) (1.3585) + 1.11666 + 2 (1.11666)(1.3585)

= 0.13585 + 1.11666 + 3.0339

= 4.2865 … (7)

y_1^{III} = ($xy^{I} + 2\,y^{I\,I} + 2\,yy^{I} + 2\,y^{2I}$)

= ($x_1y_1^{II} + 2\,y_1^{I} + 2\,y_1y_1^{II} + 2\,y_1^{2I}$)

= (0.1)(4.2865) + 2 (1.3585) + 2 (1.1167)(4.2865) + 2 (1.3585)^2

= 0.4287 + 2.717 + 9.5735 + 3.6910

= 16.4102 … (8)

Substituting (6) , (7) and (8) in (5) we get

y (0.2) = 1.1167 + (0.1)(1.3585) + [$(0.1)^2/2$)(4.2865)] + [$(0.1)^3/6$)(16.4102)]

= 1.1167 + 0.13585 + 0.0214 + 0.002735

= 1.27668

Therefore y (0.2) = 1.27668

To find y (0.3)

By Taylors series we have

$Y_3 = y_2 + hy_2 + (h^2/2!)\, y_2^{II} + (h_3/3!)\, y_2^{III} + \ldots$ … (9)

Now y_2^{I} = ($xy + y^2$)$_2$ = $x_2y_2 + y_2^2$

= (0.2) (1.2767) + (1.2767)^2

= 0.2553 + 1.6299

= 1.8852 … (10)

$$\begin{aligned}
y_2' &= (xy' + y + 2\, yy')_2 \\
&= x_2y_2' + y_2 + 2\, y_2y_2' \\
&= (0.2)\,(1.8852) + 1.2767 + 2\,(1.2767)(1.8852) \\
&= 0.3770 + 1.2767 + 4.8136 \\
&= 6.4674 && \dots (11) \\
y_1''' &= (xy'' + 2\, y' + \; yy'' + 2\, y^{2'})_2 \\
&= (x_2y_2'' + 2\, y_2' + 2\, y_2y_2'' + 2\, y_2^{2'}) \\
&= (0.2)(6.4674) + 2\,(1.8852) + 2\,(1.2767)(6.4674) + \\
&\qquad 2\,(1.8852)^2 \\
&= 1.2934 + 3.7704 + 16.5139 + 7.1079 \\
&= 28.6855 && \dots (12)
\end{aligned}$$

Substituting (10) , (11) and (12) in (9) we get

$$\begin{aligned}
y\,(0.3) &= 1.2767 + 2\,(1.8852) + [\,(0.1)^2/2)(6.4674)] + \\
&\qquad [\,(0.1)^3/6)(28.6855)] \\
&= 1.2767 + 0.18852 + 0.0323 + 0.004780 \\
&= 1.5023
\end{aligned}$$

Therefore $y\,(0.3) = 1.5023$

Therefore we have the following values

$$\begin{aligned}
x_0 &= 0 & y_0 &= 1 \\
x_1 &= 0.1 & y_1 &= 1.11666 \\
x_2 &= 0.2 & y_2 &= 1.27668 \\
x_3 &= 0.3 & y_3 &= 1.5023
\end{aligned}$$

To find y (0.4) by **Milne"s Predictor formula**

$$\begin{aligned}
y_{n+1},\, p &= y_{n-3} + 4h/3\,[\,2y_{n-2}' - y_{n-1}' + 2\, y_n'\,] && \dots (13) \\
y_3' &= (xy + y^2)_3 \\
&= x_3y_3 + y_3^2
\end{aligned}$$

$= [(0.3)(1.5023) + (1.5023)^2]$

$= 0.45069 + 2.2569$

$= 2.7076$

Putting n = 3 in (13) we get

$y_4, p = y_0 + 4h/3 [2y_1' - y_2' + 2y_3']$

$y_4, p = 1 + 4 (0.1)/3 [2 (1.3585) - 1.8852 + 2 (2.7076)]$

$= 1 + 0.1333 [2.717 - 1.8852 + 5.4152]$

$= 1.8329$

Therefore y_4, p = 1.8329 (By Milne's Predictor Formula)

To find y = (0.4) by Milne's Corrector Formula:

By Milne's Corrector Formula we have

$y_{n+1}, c = y_{n-1} + h/3 [y_{n-1}' + 4 y_n' + y_{n+1}']$ … (14)

Now $y_4' = (xy + y^2)_4 = x_4y_4 + y_4^2$

$= [(0.4)(1.8329) + 1.8329)^2]$

$= 0.7332 + 3.3595$

$= 4.0927$

,Putting n = 3 , in (14) we get

$y_4, c = y_2 + h/3 [y_2' + 4 y_3' + y_4']$

$y_4, c = 1.27668 + (0.1)/3 [1.8852 + 4 (2.7076) + 4.0927]$

$= 1.27668 + 0.0333 [1.8852 + 10.8304 + 4.0927]$

$= 1.8369$

The value of y(0.4) calculated by Milne's Corrector Formula is

y (0.4) = 1.8369

ADAM'S MOULTON METHOD:

Let $dy/dx = f(x,y)$ be the differential equation to be solved with the i9nitial condition $y_0 = y(x_0)$. we have to compute

$y_{-1} = y(x_0 - h)$

$y_{-2} = y(x_0 - 2h)$ and

$y_{-3} = y(x_0 - 3h)$ by Taylor series or Runge-Kutta Method. Then calculate

$f_{-1} = f(x_0 - h, y_{-1})$

$f_{-2} = f(x_0 - 2h, y_{-2})$

$f_{-3} = f(x_0 - 3h, y_{-3})$

$y_1 = y_0 + \oint_{x0}^{x1} f(x,y)dx$

Using Newton's backward interpolation formula

$$f(x, y) = f_0 + n\nabla f_0 + \frac{n(n+1)}{2}\nabla^2 f_0 + \frac{n(n+1)(n+2)}{6}\nabla^3 f_0 + \ldots.$$

We have

$$y_1 = y_0 + \oint_{x0}^{x1} f0 + n\nabla f0 + \frac{n(n+1)}{2}\nabla 2 f0 + \ldots.)dx$$

Since $x = x_0 + nh$ $dx = hdn$

$$y_1 = y_0 + \oint_0^1 f0 + n\nabla f0 + \frac{n(n+1)}{2}\nabla 2 f0 + \ldots.)dx$$

$$y_1 = y_0 + h[f_0 + 1/2\,\nabla f_0 + 5/12\,\nabla^2 f_0 + 3/8\,\nabla^3 f_0 + \ldots.]$$

Substituting for $\nabla f_0, \nabla^2 f_0 \ldots..$ we get

$$\mathbf{y_1 = y_1^{(P)} = y_0 + h/24\,[55 f_0 - 59 f_{-1} + 37 f_{-2} - 9 f_{-3}]}$$

Here y_1^P is the Predictor formula.

To derive the Adam's Moulton Corrector formula, we have to use Newton's backward formula at f_1 i.e.,

$$f(x, y) = f_1 + n\nabla f_1 + \frac{n(n+1)}{2}\nabla^2 f_1 + \frac{n(n+1)(n+2)}{6}\nabla^3 f_1 + \dots$$

$$y_1 = y_0 + \oint_{x0}^{x1} f1 + n\nabla f1 + \frac{n(n+1)}{2}\nabla 1\, f0 + \dots)dx$$

Since x = x_1+ nh dx = hdn

$$y_1 = y_0 + \oint_0^1 f1 + n\nabla f1 + \frac{n(n+1)}{2}\nabla 2\, f1 + \dots)dx$$

$$y_1 = y_0 + h[f_1 - 1/2\nabla f_1 - 1/12\nabla^2 f_1 - 1/24\nabla^3 f_1 + \dots]$$

Substituting for $\nabla f_1, \nabla^2 f_1 \dots$. we get

$$y_1 = \mathbf{y_1^{(C)}} = \mathbf{y_0 + h/24\,[9 f_1 + 19 f_0 - 5 f_1 + f_{-2}]}$$

This is **Adam's – Moulton Corrector Formula**

Problem:

1. Solve the equation dy/dx = $x^2(1+y)$ with $y(1) = 1$ $y(1.1) = 1.233$ $y(1.2) = 1.548$ and $y(1.3) = 1.979$ evaluate $y(1.4)$ by Adam's Moulton Method.

Solution:

Given $f(x,y) = x^2(1 + y)$

$h = 0.1$

When $x = 1$ $y = 1$

Therefore $f_{-3} = f(x,y)$

$= (1)^2(1 + 1)$

$= 2$

When $x = 1.1$ $y = 1.233$

Therefore $f_{-2} = f(x,y)$

$= (1.1)^2(1.1 + 1.233)$

		=	2.702
When	x	=	1.2 $\quad y = 1.548$
Therefore	f_{-1}	=	$f(x,y)$
		=	$(1.2)^2 (1.2 + 1.548)$
		=	3.669
When	x	=	1.3 $\quad y = 1.979$
Therefore	f_0	=	$f(x,y)$
		=	$(1.3)^2 (1.3 + 1.979)$
		=	5.035

The Predictor formula is

$$y_1^{(P)} = y_0 + h/24 [55 f_0 - 59 f_{-1} + 37 f_{-2} - 9 f_{-3}]$$

$$= 2.4011 + (0.1/24) [(55 \times 5.035) - (59 \times 3.669) + (37 \times 2.702) - (9 \times 2)]$$

$$= 2.573$$

The Corrector formula is

$$y_1^{(c)} = y_0 + h/24 [9 f_1 + 19 f_0 - 5 f_{-1} + f_{-2}]$$

$$f_1 = f (x_1,y_1)$$

$$= x^2 (1 + y)) \quad \text{when} \quad x = 1.4, \quad y = 2.573$$

$$= (1.4)^2 (1.4 \times 2.573)$$

$$= 7.004$$

Therefore $y_1^{(c)} = 1.979 + (0.1/24) [(9 \times 7.004) + (19 \times 5.035) - (5 \times 3.609) + (2.702)]$

$$= 2.575$$

Therefore $y(1.4) = 2.575$

2. Find $y(0.4)$ given that $y' = 1 + xy$ and $y(0) = 2$, $y(0.1) = 2.1103$, $y(0.2) = 2.243$ and $y(0.3) = 2.4011$ by Adam's Moulton Predictor-Corrector method.

Solution:

Given $f(x,y) = 1 + xy$

$h = 0.1$

$f_{-3} = f(x,y)$ when $x = 0$, $y = 2$

$= 1 + (0 \times 2)$

$= 1$

$f_{-2} = f(x_{-2}, y_{-2})$ when $x = 0.1$ $y = 2.1103$

$= 1 + (0.1 \times 2.1103)$

$= 1.21103$

$f_{-1} = f(x_{-1}, y_{-1})$ when $x = 0.2$ $y = 2.243$

$= 1 + (0.2 \times 2.243)$

$= 1.4486$

$f_0 = f(x_0, y_0)$ when $x = 0.3$ $y = 2.4011$

$= 1 + (0.3 \times 2.4011)$

$= 1.7203$

The Predictor formula is

$$y_1^{(P)} = y_0 + h/24\,[\,55 f_0 - 59 f_{-1} + 37 f_{-2} - 9 f_{-3}\,]$$

$$= 2.4011 + (0.1/24)\,[\,(55 \times 1.72033) - (59 \times 1.4486) + (37 \times 1.21103) - (9 \times 1)\,]$$

$$= 2.5884$$

The Corrector formula is

$$y_1^{(c)} = y_0 + h/24\,[\,9 f_1 + 19 f_0 - 5 f_{-1} + f_{-2}\,]$$

$$f_1 = f(x_1, y_1)$$

$= (1 + xy)$ when $x = x_1, y = y_1$

$= (1 + xy)$ when $x = 0.4, \quad y = 2.5884$

$= 1 + (0.4 \times 2.5884)$

$= 2.0354$

Therefore $y_1^{(c)} = 2.4011 + (0.1/24) [(9 \times 2.0354) + (19 \times 1.72033)$

$- (5 \times 1.4486) + (1.21103)]$

$= 2.5885$

....

UNIT - 4

Structure of a C++ program

A **C++ program** is structured in a specific and particular manner. In C++, a program is divided into the following three sections:

1. Standard Libraries Section
2. Main Function Section
3. Function Body Section

For example, let's look at the implementation of the Hello World program:

Run

Standard libraries section

- *#include* is a specific preprocessor command that effectively copies and pastes the entire text of the file, specified between the angle brackets, into the source code.
- The file <iostream>, which is a standard file that should come with the C++ compiler, is short for **input-output streams**. This command contains code for displaying and getting an input from the user.
- namespace is a prefix that is applied to all the names in a certain set. iostream file defines two *names* used in this program - **cout** and **endl**.
- This code is saying: Use the cout and endl tools from the std toolbox.

Main function section

- The starting point of all C++ programs is the main function.
- This function is called by the operating system when your program is executed by the computer.
- { signifies the start of a block of code, and } signifies the end.

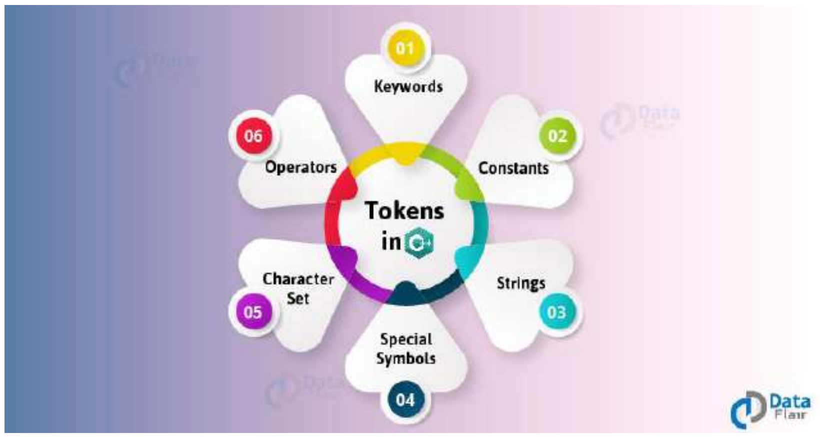
Function body section

- The name cout is short for **character output** and displays whatever is between the << brackets.

- Symbols such as << can also behave like functions and are used with the keyword cout.
- The return keyword tells the program to return a value to the function int main
- After the return statement, execution control returns to the operating system component that launched this program.
- Execution of the code terminates here.

Tokens in C++

Tokens act as building blocks of a program. Just like a living cell is the smallest possible unit of life, *tokens in C++ are referred to as the smallest individual units in a program*. *Keywords in C++ help the user in framing statements and commands in a language*. Each keyword conveys a unique connotation to the compiler to perform a specific task. Just like the combination of words helps us in framing sentences, the combination of keywords helps us in framing statements to perform logical operations in a programming language. Simply combining keywords wouldn't help to serve the purpose.
As we need to use proper grammar to form a meaningful sentence, we need to be well-acquainted with the *syntax of C++* to instruct the compiler what to do. If these statements are not formed in a logical manner, they would sound gibberish and you would get a compilation error. Let's discuss the concept of Tokens with Character set in C++ in detail.



1. C++ Character Set

Before we begin with C++ tokens, let us understand what Character set has to offer.

C++ Character set is basically a set of valid characters that convey a specific connotation to the compiler. We use characters to represent letters, digits, special symbols, white spaces, and other characters.
The C++ character set consists of 3 main elements. They are:

1. **Letters:** These are alphabets ranging from A-Z and a-z (both uppercase and lowercase characters convey different meanings)
2. **Digits:** All the digits from 0 – 9 are valid in C++.
3. **Special symbols:** There are a variety of special symbols available in C++ like mathematical, logical and relational operators like **+,-, *, /, \, ^, %, !, @, #, ^, &, (,), [,], ;** and many more.

2. Tokens in C++

As discussed earlier, ***tokens in C++ are the smallest individual unit of a program***.
The following tokens are available in C++ which are similar to that seen in C with the addition of certain exclusive keywords, strings, and operators:

- Keywords
- Identifiers
- Constants
- Strings
- Special symbols
- Operators

3. C++ Keywords

Keywords in C++ refer to the pre-existing, reserved words, each holding its own position and power and has a specific function associated with it.
It is important to note that we cannot use C++ keywords for assigning variable names as it would suggest a totally different meaning entirely and would be incorrect.

Here is a list of keywords available in C++ according to the latest standards:

Align as	Align of	asm	auto	bool	break
case	catch	char	char16_t	char32_t	class
const	constexpr	const_cast	continue	decltype	default
delete	double	do	dynamic_cast	else	enum
explicit	export	extern	FALSE	float	for
friend	goto	if	inline	int	long
mutable	namespace	new	noexcept	nullptr	operator
private	protected	public	register	reinterpret_cast	return

short	signed	sizeof	static	static_assert	static_cast
struct	switch	template	this	thread_local	throw
TRUE	try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile	wchar_t
while	–	–	–	–	–

4. C++ Identifiers

C++ allows the programmer to assign names of his own choice to variables, arrays, functions, structures, classes, and various other data structures called identifiers. The programmer may use the mixture of different types of character sets available in C++ to name an identifier.

Rules for C++ Identifiers

There are certain rules to be followed by the user while naming identifiers, otherwise, you would get a compilation error. These rules are:

1. **First character:** The first character of the identifier in C++ should positively begin with either an alphabet or an underscore. It means that it strictly cannot begin with a number.
2. **No special characters:** C++ does not encourage the use of special characters while naming an identifier. It is evident that we cannot use special characters like the e*xclamatory mark* or the *“@”* symbol.
3. **No keywords:** Using keywords as identifiers in C++ is strictly forbidden, as they are reserved words that hold a special meaning to the C++ compiler. If used purposely, you would get a compilation error.
4. **No white spaces:** Leaving a gap between identifiers is discouraged. White spaces incorporate blank spaces, newline, carriage return, and horizontal tab.
5. **Word limit:** The use of an arbitrarily long sequence of identifier names is restrained. The name of the identifier must not exceed 31 characters, otherwise, it would be insignificant.
6. **Case sensitive:** In C++, uppercase and lowercase characters connote different meanings.

Here is a table which illustrates the valid use of Identifiers:

Identifier Name	Valid or Invalid	Correction or alternative, if invalid	Elucidation if invalid

5th_element	Invalid	element_5	It violates Rule 1 as it begins with a digit
_delete	Valid	–	–
school.fee	Invalid	school_fee	It violates Rule 2 as it contains a special character '.'
register[5]	Invalid	Register[5]	It violates Rule 3 as it contains a keyword
Student[10]	Valid	–	–
employee name	Invalid	employee _name	It violates Rule 4 as it contains a blank space
perimeter()	Valid	–	–

5. C++ Constants

Before we begin our discussion on constants in C++, it is important to note that we can use the terms "constants" and "literals" interchangeably.

As the name itself suggests, constants are referred to as fixed values that cannot change their value during the entire program run as soon as we define them.

Syntax:
const data_type variable_name = value;
Types of Constants in C++

The different types of constants are:

- **Integer constants –** These constants store values of the int data type.
- For instance:
- *const int data = 5;*
- **Floating constants –** These constants store values of the float data type.
- For instance:
- *const float e = 2.71;*
- **.**
- **Character constants –** These constants store values of the character data type.
- For instance:
- *const char answer = 'y';*
- **String constants –** These constants are also of the character data type but differ in the declaration part.
- For instance:

- *const char title[] = ‘‘DataFlair’’;*
- **Octal constants –** The number system which consists of only 8 digits, from 0 to 7 is called the octal number system. The constant octal values can be declared as:
- *const int oct = 034;*
- It is the octal equivalent of the digit 28 in the decimal number system.

- **Hexadecimal constants –** The number system which consists of 16 digits, from 0 to 9 and alphabets ‘a’ to ‘f’ is called hexadecimal number system. The constant hexadecimal values can be declared as:
- *const int hex = 0x40;*
- It is the hexadecimal equivalent of the digit 64 in the decimal number system.

6. C++ Strings

Just like characters, ***strings in C++*** are used to store letters and digits. Strings can be referred to as an array of characters as well as an individual data type.
It is enclosed within double quotes, unlike characters which are stored within single quotes. The termination of a string in C++ is represented by the null character, that is, ***‘\0’.*** The size of a string is the number of individual characters it has.
In C++, a string can be declared in the following ways:

char name[30] = ‘’Hello!”; // The compiler reserves 30 bytes of memory for the string.
char name[] = “Hello!”; // The compiler reserves the required amount of memory for the string.
char name[30] = { ‘H’ , ’e’ , ’l’ , ’l’ , ’o’};; // This is how a string is represented as a set of characters.
string name = “Hello” // The compiler reserves 32 bytes of memory.
7. Special Symbols

Apart from letters and digits, there are some special characters in C++ which help you manipulate or perform data operations. Each special symbol has a specific meaning to the C++ compiler.
Here is a table which illustrates some of the special characters in C:

Special Character	**Trivial Name**	**Function**
[ ]	Square brackets	The opening and closing brackets of an array symbolize single and multidimensional subscripts.
()	Simple brackets	The opening and closing brackets represent function declaration and calls, used in print statements.
{ }	Curly braces	The opening and closing curly brackets to denote the start and end of a particular fragment of code which may be functions, loops or

		conditional statements
,	Comma	We use commas to separate more than one statements, like in the declaration of different variable names
#	Hash / Pound / Preprocessor	The hash symbol represents a preprocessor directive used for denoting the use of a header file
*	Asterisk	We use the asterisk symbol in various respects such as to declare pointers, used as an operand for multiplication
~	Tilde	We use the tilde symbol as a destructor to free memory
.	Period / dot	The use the dot operator to access a member of a structure

C++ Operators

In this tutorial, we will learn about the different types of operators in C++ with the help of examples. In programming, an operator is a symbol that operates on a value or a variable.

Operators in C++ can be classified into 6 types:

1.Arithmetic Operators
2.Assignment Operators
3.Relational Operators
4.Logical Operators
5.Bitwise Operators
6.Other Operators

1. C++ Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on variables and data. For example,

```
a + b;
```

Here, the `+` operator is used to add two variables `a` and `b`. Similarly there are various other arithmetic operators in C++.

Operator	Operation
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulo Operation (Remainder after division)

Example 1: Arithmetic Operators

```
#include <iostream>
using namespace std;

int main() {
    int a, b;
    a = 7;
    b = 2;
    // printing the sum of a and b
    cout << "a + b = " << (a + b) << endl;

    // printing the difference of a and b
    cout << "a - b = " << (a - b) << endl;

    // printing the product of a and b
    cout << "a * b = " << (a * b) << endl;

    // printing the division of a by b
    cout << "a / b = " << (a / b) << endl;

    // printing the modulo of a by b
    cout << "a % b = " << (a % b) << endl;

    return 0;
}
```

Output

```
a + b = 9
a - b = 5
a * b = 14
a / b = 3
a % b = 1
```

Here, the operators `+`, `-` and `*` compute addition, subtraction, and multiplication respectively as we might have expected.

Division Operator

Note the operation `(a / b)` in our program. The `/` operator is the division operator.

As we can see from the above example, if an integer is divided by another integer, we will get the quotient. However, if either divisor or dividend is a floating-point number, we will get the result in decimals.

```
In C++,
7/2 is 3

7.0 / 2 is 3.5

7 / 2.0 is 3.5

7.0 / 2.0 is 3.5
```

% Modulo Operator
The modulo operator `%` computes the remainder. When `a = 9` is divided by `b = 4`, the remainder is **1**.

Increment and Decrement Operators

C++ also provides increment and decrement operators: `++` and `--` respectively.

- `++` increases the value of the operand by **1**
- `--` decreases it by **1**

For example,

```
int num = 5;

// increment operator
++num;  // 6
```

Here, the code `++num;` increases the value of `num` by **1**.

Example 2: Increment and Decrement Operators

```
// Working of increment and decrement operators

#include <iostream>
using namespace std;
```

```
int main() {
    int a = 10, b = 100, result_a, result_b;

    // incrementing a by 1 and storing the result in result_a
    result_a = ++a;
    cout << "result_a = " << result_a << endl;


    // decrementing b by 1 and storing the result in result_b
    result_b = --b;
    cout << "result_b = " << result_b << endl;

    return 0;
}
```

Output

```
result_a = 11
result_b = 99
```

In the above program, we have used the `++` and `--` operators as **prefixes (++a and --b)**. However, we can also use these operators as **postfix (a++ and b--)**.

2. C++ Assignment Operators

In C++, assignment operators are used to assign values to variables. For example,

```
// assign 5 to a
a = 5;
```

Here, we have assigned a value of `5` to the variable `a`.

Operator	Example	Equivalent to
`=`	`a = b;`	`a = b;`
`+=`	`a += b;`	`a = a + b;`

-=	a -= b;	a = a - b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a % b;

Example 3: Assignment Operators

```
#include <iostream>
using namespace std;

int main() {
    int a, b;

    // 2 is assigned to a
    a = 2;

    // 7 is assigned to b
    b = 7;

    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << "\nAfter a += b;" << endl;

    // assigning the sum of a and b to a
    a += b;  // a = a +b
    cout << "a = " << a << endl;

    return 0;
}
```

Output

```
a = 2
b = 7

After a += b;
a = 9
```

---

3. C++ Relational Operators

A relational operator is used to check the relationship between two operands. For example,

```
// checks if a is greater than b
a > b;
```

Here, `>` is a relational operator. It checks if `a` is greater than `b` or not.
If the relation is **true**, it returns **1** whereas if the relation is **false**, it returns **0**.

Operator	Meaning	Example
`==`	Is Equal To	`3 == 5` gives us **false**
`!=`	Not Equal To	`3 != 5` gives us **true**
`>`	Greater Than	`3 > 5` gives us **false**
`<`	Less Than	`3 < 5` gives us **true**

>=	Greater Than or Equal To	3 >= 5 give us **false**
<=	Less Than or Equal To	3 <= 5 gives us **true**

Example 4: Relational Operators

```
#include <iostream>
using namespace std;

int main() {
    int a, b;
    a = 3;
    b = 5;
    bool result;

    result = (a == b);   // false
    cout << "3 == 5 is " << result << endl;

    result = (a != b);  // true
    cout << "3 != 5 is " << result << endl;

    result = a > b;   // false
    cout << "3 > 5 is " << result << endl;

    result = a < b;   // true
    cout << "3 < 5 is " << result << endl;

    result = a >= b;  // false
    cout << "3 >= 5 is " << result << endl;

    result = a <= b;  // true
    cout << "3 <= 5 is " << result << endl;

    return 0;
}
```

Output

```
3 == 5 is 0
3 != 5 is 1
3 > 5 is 0
3 < 5 is 1
3 >= 5 is 0
3 <= 5 is 1
```

4. C++ Logical Operators

Logical operators are used to check whether an expression is **true** or **false**. If the expression is **true**, it returns **1** whereas if the expression is **false**, it returns **0**.

Operator	Example	Meaning
&&	expression1 **&&** expression2	Logical AND. True only if all the operands are true.
||	expression1 **||** expression2	Logical OR. True if at least one of the operands is true.
!	**!**expression	Logical NOT. True only if the operand is false.

In C++, logical operators are commonly used in decision making. To further understand the logical operators, let's see the following examples,

```
Suppose,
a = 5
b = 8
```

```
Then,

(a > 3) && (b > 5) evaluates to true
(a > 3)  && (b < 5) evaluates to false

(a > 3) || (b > 5) evaluates to true
(a > 3) || (b < 5) evaluates to true
(a < 3) || (b < 5) evaluates to false

!(a < 3) evaluates to true
!(a > 3) evaluates to false
```

Example 5: Logical Operators

```
#include <iostream>
using namespace std;

int main() {
    bool result;

    result = (3 != 5) && (3 < 5);     // true
    cout << "(3 != 5) && (3 < 5) is " << result << endl;

    result = (3 == 5) && (3 < 5);    // false
    cout << "(3 == 5) && (3 < 5) is " << result << endl;

    result = (3 == 5) && (3 > 5);    // false
    cout << "(3 == 5) && (3 > 5) is " << result << endl;

    result = (3 != 5) || (3 < 5);    // true
    cout << "(3 != 5) || (3 < 5) is " << result << endl;

    result = (3 != 5) || (3 > 5);    // true
    cout << "(3 != 5) || (3 > 5) is " << result << endl;
    result = (3 == 5) || (3 > 5);    // false
    cout << "(3 == 5) || (3 > 5) is " << result << endl;

    result = !(5 == 2);    // true
    cout << "!(5 == 2) is " << result << endl;

    result = !(5 == 5);    // false
    cout << "!(5 == 5) is " << result << endl;

    return 0;
}
```

Output

```
(3 != 5) && (3 < 5) is 1
(3 == 5) && (3 < 5) is 0
(3 == 5) && (3 > 5) is 0
(3 != 5) || (3 < 5) is 1
(3 != 5) || (3 > 5) is 1
(3 == 5) || (3 > 5) is 0
!(5 == 2) is 1
!(5 == 5) is 0
```

Explanation of logical operator program

- (3 != 5) && (3 < 5) evaluates to **1** because both operands (3 != 5) and (3 < 5) are **1** (true).
- (3 == 5) && (3 < 5) evaluates to **0** because the operand (3 == 5) is **0** (false).
- (3 == 5) && (3 > 5) evaluates to **0** because both operands (3 == 5) and (3 > 5) are **0** (false).
- (3 != 5) || (3 < 5) evaluates to **1** because both operands (3 != 5) and (3 < 5) are **1** (true).
- (3 != 5) || (3 > 5) evaluates to **1** because the operand (3 != 5) is **1** (true).
- (3 == 5) || (3 > 5) evaluates to **0** because both operands (3 == 5) and (3 > 5) are **0** (false).
- !(5 == 2) evaluates to **1** because the operand (5 == 2) is **0** (false).
- !(5 == 5) evaluates to **0** because the operand (5 == 5) is **1** (true).

5. C++ Bitwise Operators

In C++, bitwise operators are used to perform operations on individual bits. They can only be used alongside `char` and `int` data types.

Operator	Description
`&`	Binary AND
`|`	Binary OR
`^`	Binary XOR
`~`	Binary One's Complement
`<<`	Binary Shift Left
`>>`	Binary Shift Right

6. Other C++ Operators

Here's a list of some other common operators available in C++. We will learn about them in later tutorials.

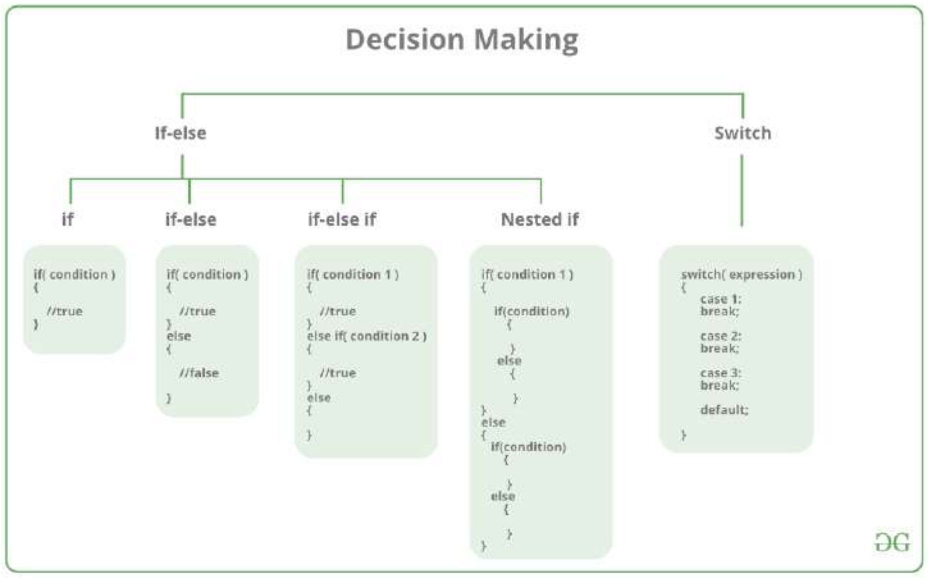
Operator	Description	Example
`sizeof`	returns the size of data type	`sizeof(int); // 4`
`?:`	returns value based on the condition	`string result = (5 > 0) ? "even" : "odd"; // "even"`
`&`	represents memory address of	`&num; // address of num`

	the operand	
`.`	accesses members of struct variables or class objects	`s1.marks = 92;`
`->`	used with pointers to access the class or struct variables	`ptr->marks = 92;`
`<<`	prints the output value	`cout << 5;`
`>>`	gets the input value	`cin >> num;`

UNIT 5

Decision Making in C / C++ (if , if..else, Nested if, if-else-if)

There come situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions we will execute the next block of code. For example, in C if x occurs then execute y else execute z. There can also be multiple conditions like in C if x occurs then execute p, else if condition y occurs execute q, else execute r. This condition of C else-if is one of the many ways of importing multiple conditions.



```
if( condition )
{
   //true
}
```

```
if( condition )
{
   //true
}
else
{
   //false
}
```

```
if( condition 1 )
{
   //true
}
else if( condition 2 )
{
   //true
}
else
{
}
```

```
if( condition 1 )
{
  if(condition)
    {
    }
   else
    {
    }
}
else
{
 if(condition)
   {
   }
 else
   {
   }
}
```

```
switch( expression )
{
    case 1:
    break;

    case 2:
    break;

    case 3:
    break;

    default;
}
```

Decision-making statements in programming languages decide the direction of the flow of program execution. Decision-making statements available in C or C++ are:

1. if statement
2. if..else statements
3. nested if statements
4. if-else-if ladder
5. switch statements

6. Jump Statements:
   1. break
   2. continue
   3. goto

4. return

if statement in C/C++

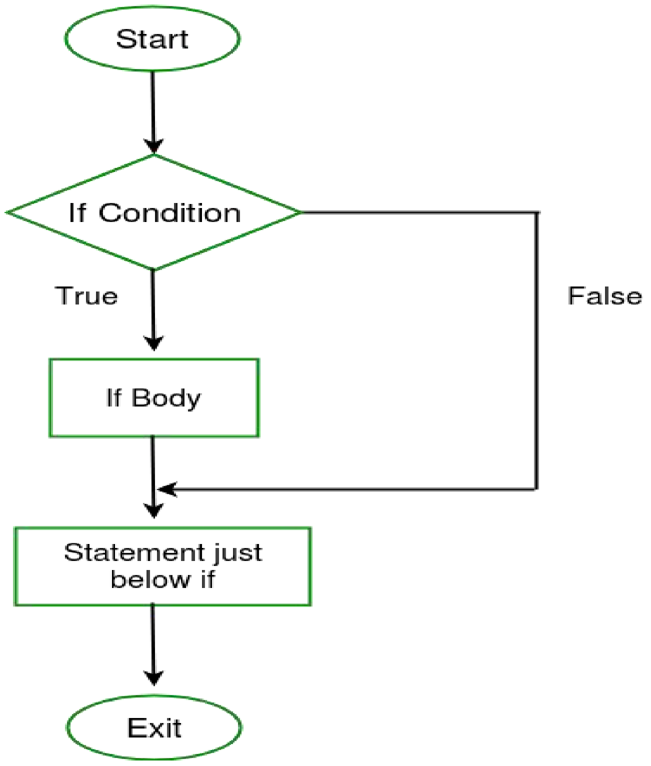
if statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.
Syntax:

```
if(condition)
{
   // Statements to execute if
   // condition is true
}
```

Here, the **condition** after evaluation will be either true or false. C if statement accepts boolean values – if the value is true then it will execute the block of statements below it otherwise not. If we do not provide the curly braces '{' and '}' after if(condition) then by default if statement will consider the first immediately below statement to be inside its block.
Example:

```
if(condition)
   statement1;
   statement2;


// Here if the condition is true, if block
// will consider only statement1 to be inside
// its block.
```

Flowchart

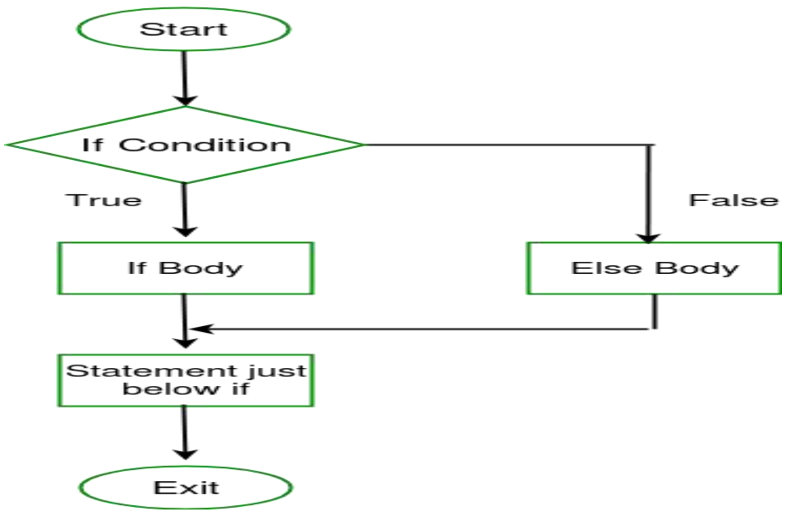


if-else in C/C++

The *if* statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else if the condition is false. Here comes the C *else* statement. We can use the *else* statement with *if* statement to execute a block of code when the condition is false.
Syntax:

```
if (condition)

{

    // Executes this block if

    // condition is true

}

else

{

    // Executes this block if
    // condition is false
}
```

Flowchart:

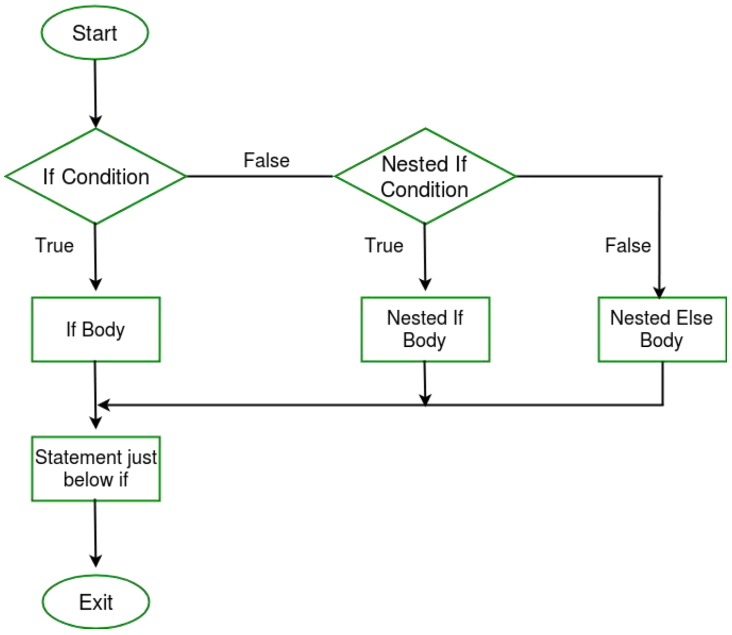


nested-if in C/C++

A nested if in C is an if statement that is the target of another if statement. Nested if statements mean an if statement inside another if statement. Yes, both C and C++ allow us to nested if statements within if statements, i.e, we can place an if statement inside another if statement.
Syntax:

```
if (condition1)
{
   // Executes when condition1 is true
   if (condition2)
   {
      // Executes when condition2 is true
   }
}
```

Flowchart

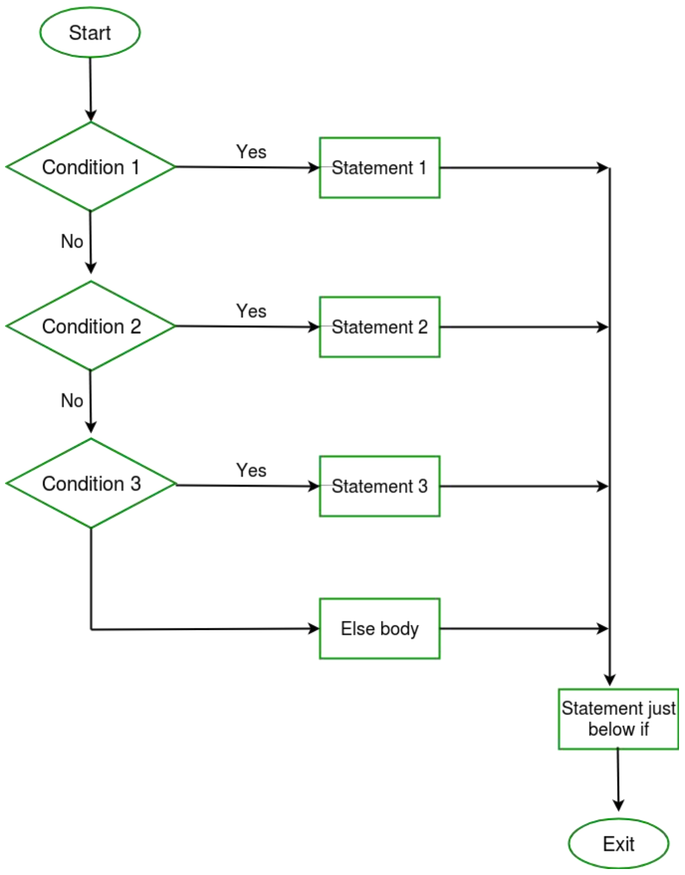


if-else-if ladder in C/C++

Here, a user can decide among multiple options. The C if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the C else-if ladder is bypassed. If none of the conditions are true, then the final else statement will be executed.

Syntax:

```
if (condition)

    statement;

else if (condition)

    statement;

.

.

else

    statement;
```



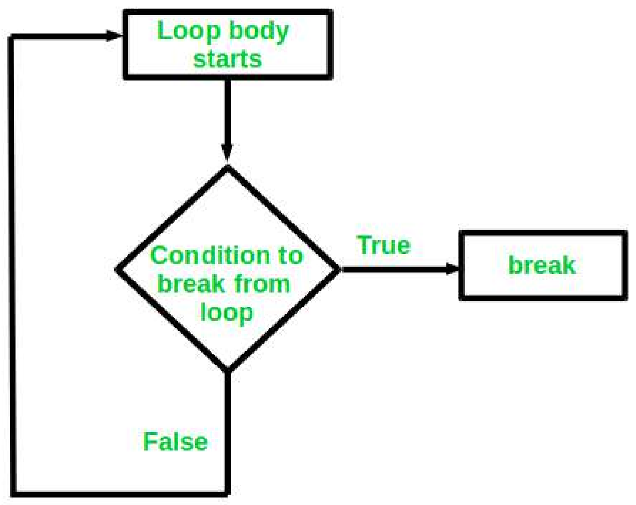
Jump Statements in C/C++

These statements are used in C orC++ for the unconditional flow of control throughout the functions in a program. They support four types of jump statements:

1. **C break:** This loop control statement is used to terminate the loop. As soon as the break statement is encountered from within a loop, the loop iterations stop there, and control returns from the loop immediately to the first statement after the loop.
   Syntax:

```
break;
```

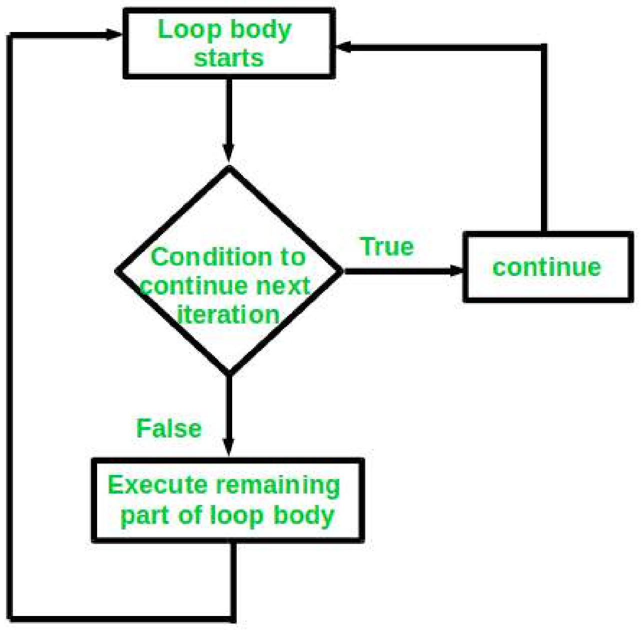
1. Basically, break statements are used in situations when we are not sure about the actual number of iterations for the loop or we want to terminate the loop based on some condition.



1. **C** continues**:** This loop control statement is just like the break statement.
   The *continue* statement is opposite to that of the break *statement*, instead of terminating the loop, it forces to execute the next iteration of the loop.
   As the name suggests the continue statement forces the loop to continue or execute the next iteration. When the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped and the next iteration of the loop will begin.
   Syntax:

```
continue;
```

1.


Loop body starts
Condition to continue next iteration
True
continue
False
Execute remaining part of loop body

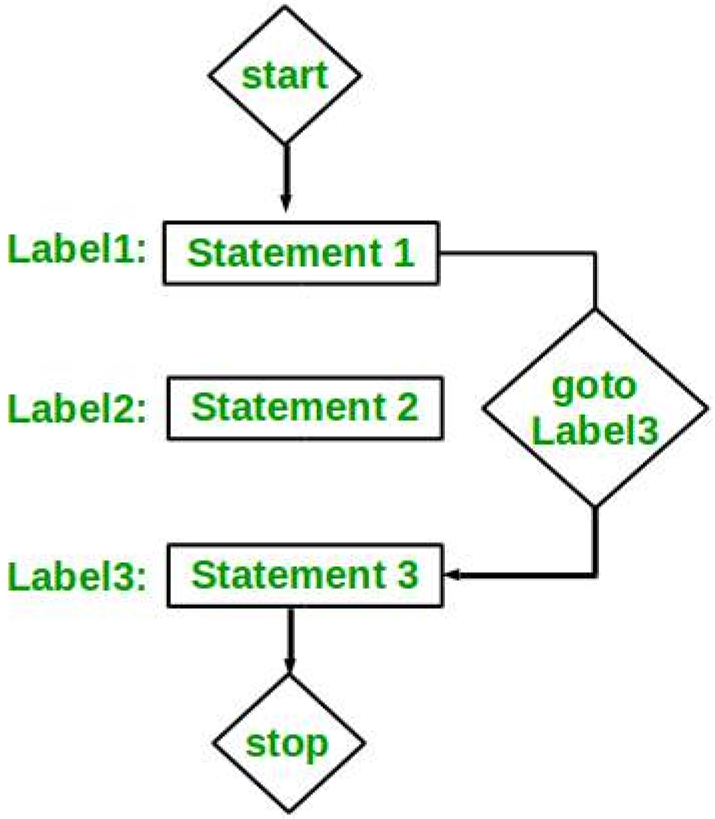
C goto:

The goto statement in C/C++ also referred to as unconditional jump statement can be used to jump from one point to another within a function.

Syntax:

```
Syntax1      |   Syntax2
----------------------------
goto label;  |    label:
.            |    .
.            |    .
.            |    .
label:       |    goto label;
```

1. In the above syntax, the first line tells the compiler to go to or jump to the statement marked as a label. Here label is a user-defined identifier that indicates the target statement. The statement immediately followed after ‘label:’ is the destination statement. The ‘label:’ can also appear before the ‘goto label;’ statement in the above syntax.



C return:

The return in C or C++ returns the flow of the execution to the function from where it is called. This statement does not mandatorily need any conditional statements. As soon as the statement is executed, the flow of the program stops immediately and return the control from where it was called. The return statement may or may not return anything for a void function, but for a non-void function, a return value is must be returned.

Syntax:

```
return[expression];
```

while loop in c++

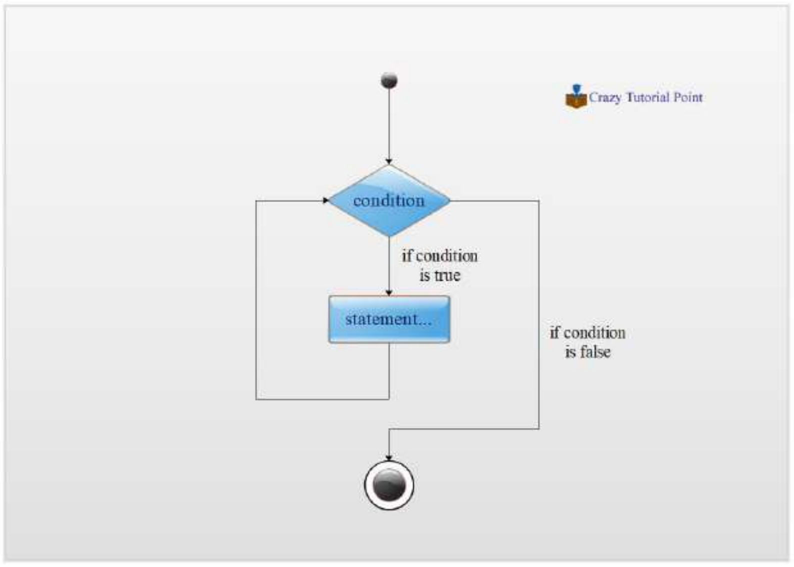
- A while loop in C++ programming repeatedly executes a target statement as long as a given condition is true.

- Its called a loop because control keeps looping back to the start of the statement until the test becomes false.
- The loop iterates as long as the defined condition is true. When it ceases to be true and becomes false, control passes to the first line after the loop.
- The reserved word while begins the while statement.
- The Boolean expression condition determines whether the body will be (or will continue to be) executed. The expression must be enclosed within parentheses as shown.
- The statement is the statement to be executed while the Boolean expression is true. The statement makes up the body of the while statement.

Syntax

```
while(condition)
  {
     statements(s);
  }
```

Flow Diagram



C++ while and do...while Loop

In this tutorial, we will learn the use of while and do...while loops in C++ programming with the help of some examples.

In computer programming, loops are used to repeat a block of code.

For example, let's say we want to show a message 100 times. Then instead of writing the print statement 100 times, we can use a loop.

That was just a simple example; we can achieve much more efficiency and sophistication in our programs by making effective use of loops.

There are **3** types of loops in C++.

1. for loop
2. while loop
3. do...while loop

In the previous tutorial, we learned about the C++ for loop. Here, we are going to learn about while and do...while loops.

C++ while Loop

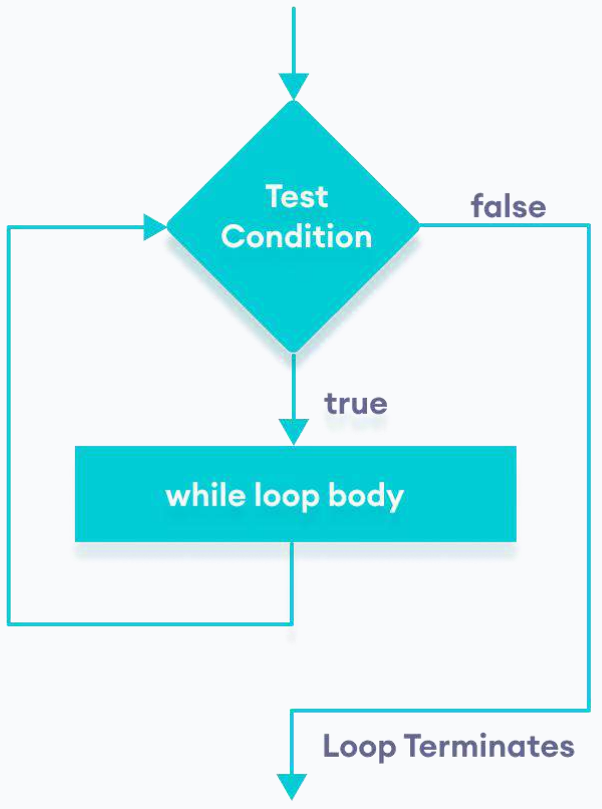
The syntax of the while loop is:

```
while (condition) {
   // body of the loop
}
```

Here,

- A while loop evaluates the condition
- If the condition evaluates to true, the code inside the while loop is executed.
- The condition is evaluated again.
- This process continues until the condition is false.
- When the condition evaluates to false, the loop terminates.
- To learn more about the conditions, visit C++ Relational and Logical Operators.

Flowchart of while Loop



Flowchart of C++ while loop

Iteration	Variable	i <= 5	Action
1st	i = 1	true	1 is printed and i is increased to 2.
2nd	i = 2	true	2 is printed and i is increased to 3.
3rd	i = 3	true	3 is printed and i is increased to 4

4th	i = 4	true	4 is printed and i is increased to 5.
5th	i = 5	true	5 is printed and i is increased to 6.
6th	i = 6	false	The loop is terminated

C++ do...while Loop

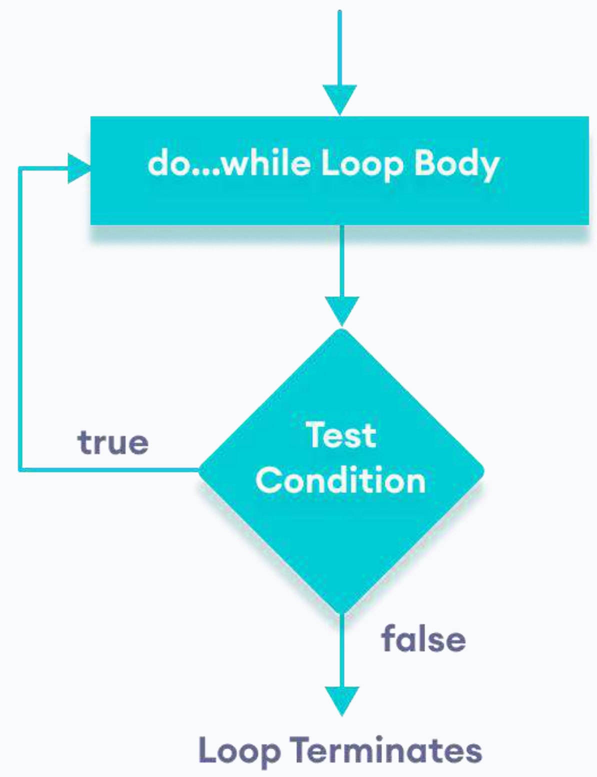
The do...while loop is a variant of the while loop with one important difference: the body of do...while loop is executed once before the condition is checked.
Its syntax is:

```
do {
   // body of loop;
}
while (condition);
```

Here,

- ➢ The body of the loop is executed at first. Then the condition is evaluated.
- ➢ If the condition evaluates to true, the body of the loop inside the do statement is executed again.
- ➢ The condition is evaluated once again.
- ➢ If the condition evaluates to true, the body of the loop inside the do statement is executed again.
- ➢ This process continues until the condition evaluates to false. Then the loop stops.

Flowchart of do...while Loop



Flowchart of C++ do...while loop

Example 3: Display Numbers from 1 to 5

```
// C++ Program to print numbers from 1 to 5

#include <iostream>

using namespace std;

int main() {
    int i = 1;

    // do...while loop from 1 to 5
    do {
        cout << i << " ";
        ++i;
    }
    while (i <= 5);
    return 0;
}
```

Output

```
1 2 3 4 5
```

Here is how the program works.

Iteration	Variable	i <= 5	Action
	i = 1	not checked	1 is printed and i is increased to 2
1st	i = 2	true	2 is printed and i is increased to 3
2nd	i = 3	true	3 is printed and i is increased to 4
3rd	i = 4	true	4 is printed and i is increased to 5
4th	i = 5	true	5 is printed and i is increased to **6**
5th	i = 6	false	The loop is terminated

Example 4: Sum of Positive Numbers Only

```
// program to find the sum of positive numbers
// If the user enters a negative number, the loop ends
// the negative number entered is not added to the sum
```

```
#include <iostream>
using namespace std;

int main() {
    int number = 0;
    int sum = 0;

    do {
        sum += number;

        // take input from the user
        cout << "Enter a number: ";
        cin >> number;
    }
    while (number >= 0);

    // display the sum
    cout << "\nThe sum is " << sum << endl;

    return 0;
}
```

Output 1

```
Enter a number: 6
Enter a number: 12
Enter a number: 7
Enter a number: 0
Enter a number: -2

The sum is 25
```

Here, the `do...while` loop continues until the user enters a negative number. When the number is negative, the loop terminates; the negative number is not added to the `sum` variable.

Output 2

```
Enter a number: -6
The sum is 0.
```

The body of the do...while loop runs only once if the user enters a negative number.

Infinite while loop

If the condition of a loop is always true, the loop runs for infinite times (until the memory is full). For example,

```
// infinite while loop
while(true) {
    // body of the loop
}
```

Here is an example of an infinite do...while loop.

```
// infinite do...while loop

int count = 1;

do {
   // body of loop
}
while(count == 1);
```

In the above programs, the condition is always true. Hence, the loop body will run for infinite times.

for vs while loops

A for loop is usually used when the number of iterations is known. For example,

```
// This loop is iterated 5 times
for (int i = 1; i <=5; ++i) {
   // body of the loop
}
```

Here, we know that the for-loop will be executed 5 times.

However, while and do...while loops are usually used when the number of iterations is unknown. For example,

```
while (condition) {
    // body of the loop
}
```

Declaring, Defining and Calling a Function

Function declaration, is done to tell the compiler about the existence of the function. Function's return type, its name & parameter list is mentioned. Function body is written in its definition. Lets understand this with help of an example.

```
#include < iostream>

using namespace std;



//declaring the function

int sum (int x, int y);



int main()

{

    int a = 10;

    int b = 20;

    int c = sum (a, b);    //calling the function

    cout << c;

}
```

```
//defining the function

int sum (int x, int y)

{

    return (x + y);

}
```

Copy

Here, initially the function is **declared**, without body. Then inside main() function it is **called**, as the function returns sumation of two values, and variable c is there to store the result. Then, at last, function is **defined**, where the body of function is specified. We can also, declare & define the function together, but then it should be done before it is called.

Calling a Function

Functions are called by their names. If the function is without argument, it can be called directly using its name. But for functions with arguments, we have two ways to call them,

1. Call by Value
2. Call by Reference

Call by Value

In this calling technique we pass the values of arguments which are stored or copied into the formal parameters of functions. Hence, the original values are unchanged only the parameters inside function changes.

```
void calc(int x);



int main()
```

```
{

    int x = 10;

    calc(x);

    printf("%d", x);

}



void calc(int x)

{

    x = x + 10 ;

}
```

In this case the actual variable x is not changed, because we pass argument by value, hence a copy of x is passed, which is changed, and that copied value is destroyed as the function ends(goes out of scope). So the variable **x** inside main() still has a value 10.

But we can change this program to modify the original **x**, by making the function **calc()** return a value, and storing that value in x.

```
int calc(int x);



int main()

{

    int x = 10;
    x = calc(x);

    printf("%d", x);

}



int calc(int x)

{

    x = x + 10 ;

    return x;

}
```
Copy

Call by Reference

In this we pass the address of the variable as arguments. In this case the formal parameter can be taken as a reference or a pointer, in both the case they will change the values of the original variable.

```
void calc(int *p);



int main()

{

    int x = 10;

    calc(&x);     // passing address of x as argument
    printf("%d", x);

}



void calc(int *p)

{

    *p = *p + 10;

}
```
Copy

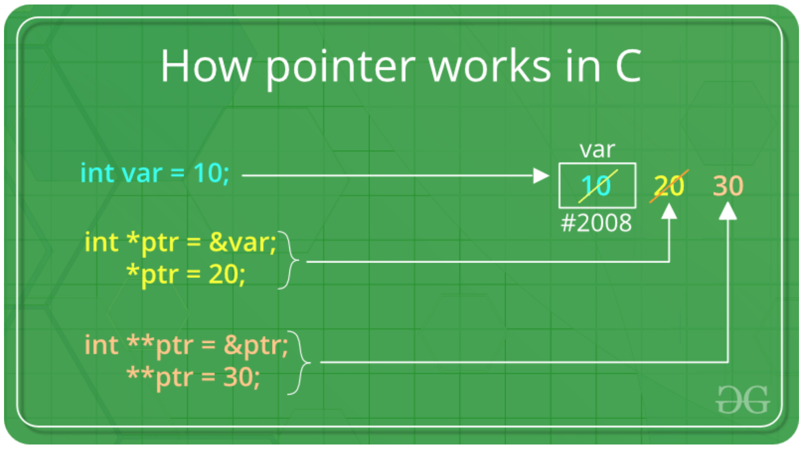
.

Pointers in C/C++

Pointers are symbolic representation of addresses. They enable programs to simulate call-by-reference as well as to create and manipulate dynamic data structures. It's general declaration in C/C++ has the format:

Syntax:
datatype *var_name;

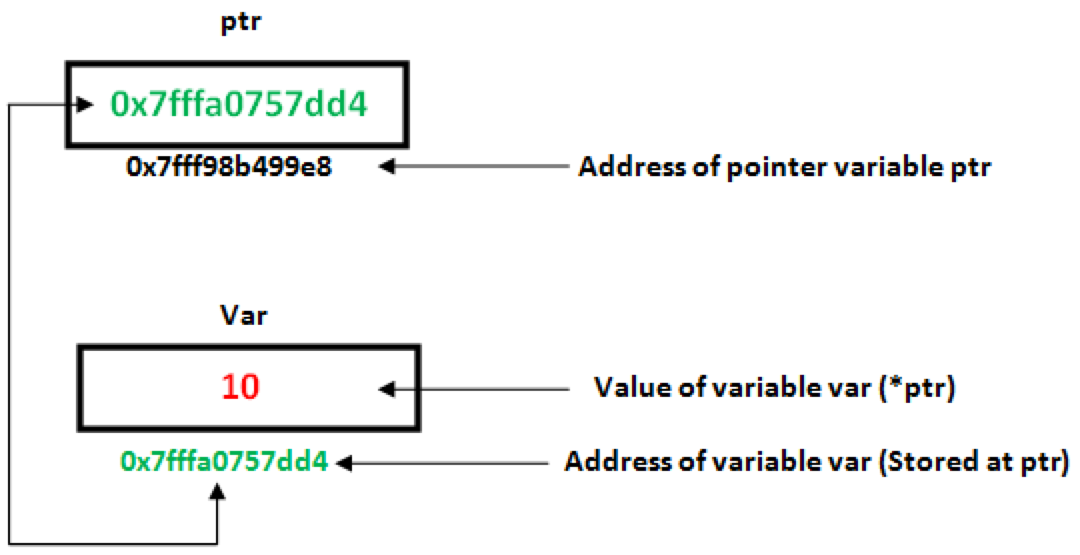
int *ptr;   //ptr can point to an address which holds int data



How to use a pointer?

- Define a pointer variable
- Assigning the address of a variable to a pointer using unary operator (&) which returns the address of that variable.
- Accessing the value stored in the address using unary operator (*) which returns the value of the variable located at the address specified by its operand.

The reason we associate data type to a pointer is **that it knows how many bytes the data is stored in**. When we increment a pointer, we increase the pointer by the size of data type to which it points.


ptr
0x7fffa0757dd4
0x7fff98b499e8
Address of pointer variable ptr
Var
10
Value of variable var (*ptr)
0x7fffa0757dd4
Address of variable var (Stored at ptr)

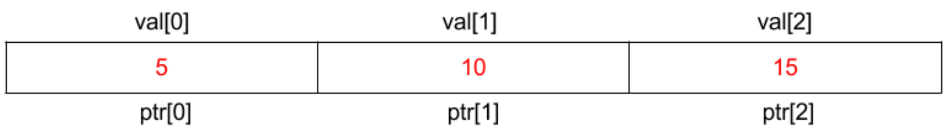
References and Pointers

There are 3 ways to pass C++ arguments to a function:

- call-by-value
- call-by-reference with pointer argument
- call-by-reference with reference argument

Array Name as Pointers

An array name contains the address of first element of the array which acts like constant pointer. It means, the address stored in array name can't be changed.
For example, if we have an array named val then **val** and **&val[0]** can be used interchangeably.



val[0]	val[1]	val[2]
5	10	15
ptr[0]	ptr[1]	ptr[2]

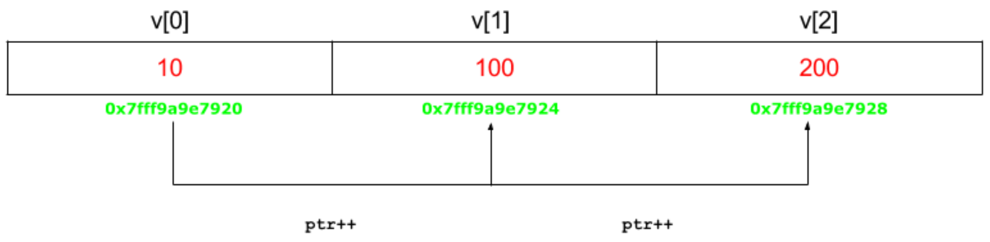
If pointer ptr is sent to a function as an argument, the array val can be accessed in a similar fashion.

Pointer vs Array

Pointer Expressions and Pointer Arithmetic

A limited set of arithmetic operations can be performed on pointers which are:

- incremented (++)
- decremented (—)
- an integer may be added to a pointer (+ or +=)
- an integer may be subtracted from a pointer (– or -=)
- difference between two pointers (p1-p2)



Advanced Pointer Notation

Consider pointer notation for the two-dimensional numeric arrays. consider the following declaration

int nums[2][3] = { { 16, 18, 20 }, { 25, 26, 27 } };

In general, nums[i][j] is equivalent to *(*(nums+i)+j)

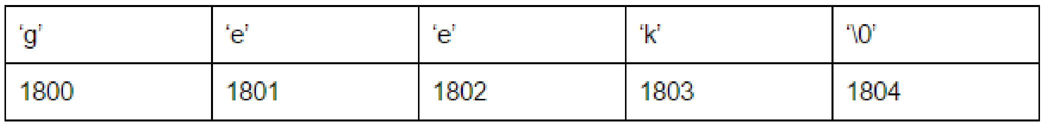
Pointer Notation	Array Notation	Value
*(*nums)	nums[0] [0]	16
*(*nums+1)	nums[0] [1]	18
*(*nums+2)	nums[0] [2]	20
((nums + 1))	nums[1] [0]	25
((nums + 1)+1)	nums[1] [1]	26
((nums + 1)+2)	nums[1] [2]	27

Pointers and String literals

String literals are arrays containing null-terminated character sequences. String literals are arrays of type character plus terminating null-character, with each of the elements being of type const char (as characters of string can’t be modified).

```
const char * ptr = "geek";
```

This declares an array with the literal representation for “geek”, and then a pointer to its first element is assigned to ptr. If we imagine that “geek” is stored at the memory locations that start at address 1800, we can represent the previous declaration as:



‘g’	‘e’	‘e’	‘k’	‘\0’
1800	1801	1802	1803	1804

As pointers and arrays behave in the same way in expressions, ptr can be used to access the characters of string literal. For example:

```
char x = *(ptr+3);
char y = ptr[3];
```

Here, both x and y contain k stored at 1803 (1800+3).

Pointers to pointers

In C++, we can create a pointer to a pointer that in turn may point to data or other pointer. The syntax simply requires the unary operator (*) for each level of indirection while declaring the pointer.

```
char a;
char *b;
char ** c;
a = ’g’;
b = &a;
c = &b;
```

Here b points to a char that stores ‘g’ and c points to the pointer b.

Void Pointers

This is a special type of pointer available in C++ which represents absence of type. void pointers are pointers that point to a value that has no type (and thus also an undetermined length and undetermined dereferencing properties).

This means that void pointers have great flexibility as it can point to any data type. There is payoff for this flexibility. These pointers cannot be directly dereferenced. They have to be first transformed into some other pointer type that points to a concrete data type before being dereferenced.

Invalid pointers

A pointer should point to a valid address but not necessarily to valid elements (like for arrays). These are called invalid pointers. Uninitialized pointers are also invalid pointers.

```
int *ptr1;
int arr[10];
int *ptr2 = arr+20;
```

Here, ptr1 is uninitialized so it becomes an invalid pointer and ptr2 is out of bounds of arr so it also becomes an invalid pointer.
(Note: invalid pointers do not necessarily raise compile errors)

NULL Pointers

Null pointer is a pointer which point nowhere and not just an invalid address.
Following are 2 methods to assign a pointer as NULL;

```
int *ptr1 = 0;
int *ptr2 = NULL;
```

Nested Loop

In this tutorial, we will learn about nested loops in C++ with the help of examples. We will also learn about break and continue in Nested Loop.

Introduction of Nested Loop in C++

A loop within another loop is called a nested loop. Nested loop means a loop statement inside another loop statement. That's why nested loop are also called as **loop inside loop**.

Working of Nested Loop

- Execution of statement within the loop flows in a way that the inner loop of the nested loop gets declared, initialized and then incremented.
- Once all the condition within the inner loop gets satisfied and becomes true it moves for the search of the outer loop. It is often called a loop within a loop.

Let's take an example:-
Suppose we want to loop through each day of a week for 3 weeks. To achieve this, we can create a loop to iterate three times (3 weeks). And inside the loop, we can create another loop to iterate 7 times (7 days). This is how we can use nested loops.

Nested for Loop

A for loop within another for loop is called Nested For loop

The syntax of nested for loop is:

```
for (initialization; condition; update) {
    for (initialization; condition; update) {
        // body of inner for-loop
    }
    // body of outer for-loop
}
```

Nested while Loop

A while loop within another while loop is called Nested while loop.

The syntax of nested while loop is:

```
while (condition) {
    while (condition) {
        // body of inner while-loop
    }
    // body of outer while-loop
}
```

Nested do-while Loop

A do-while loop within another do-while loop is called Nested do-while loop.

The syntax of nested do-while loop is:

```
do {
   do{
   // body of inner do-while-loop
   }while (condition);
   // body of outer do-while-loop
}while (condition);
```

break and continue Inside Nested Loops

When we use a break statement inside the inner loop, it terminates the inner loop but not the outer loop. For example,