# Radhai Mahavidyalaya Aurangabad College of Computer science& management science

Approved by Govt. of Maharashtra & Affilate to Dr. Babasaheb Ambedkar university Aurangabad.

Recognized Under Section 2(f) & 12 (B) of the U.G.C Act.

An ISO 9001-2015 Certified institution

## LABORATORY MANUAL

Course Title: Numerical Methods Lab

**Student Name:……………………………………………………**

**RollNo :…………………………………………………………**

**Branch:……………………………Section……………………**

**Year ………………………Semester………………………..**

# TABLE OF CONTENTS

## COURSE OBJECTIVES

1. Implement appropriate numerical methods to solve algebraic and transcendental equations.
2. Implement appropriate numerical methods to approximate a function.
3. Implement appropriate numerical methods to solve a differential equation.
4. Implement appropriate numerical methods to evaluate a derivative at a value.
5. Implement appropriate numerical methods to solve a linear system of equations.
6. Implement various numerical methods for finding root(s).
7. Implement appropriate numerical methods to calculate a definite integral.

# Session 1

## Problem 1

**OBJECTIVES:** To find the roots of non linear equations using Bisection method.

**ALGORITHM:**

1. Decide initial values for $x_1$ and $x_2$ and stopping criterion E.
2. Compute $f_1 = f(x_1)$ and $f2 = f(x_2)$.
3. If $f_1 * f_2 > 0$, $x_1$ and $x_2$ do not bracket any root and go to step 1.
4. Compute $x_0 = (x_1 + x_2) / 2$ and compute $f_0 = f(x_0)$.
5. If $f_0 = 0$ then $x_0$ is the root of the equation, print the root
6. If $f_1 * f_0 < 0$ then set $x_2 = x_0$ else set $x_1 = x_0$.
7. If $|(x_2-x_1)/x_2| < E$ then root $= (x_1 + x_2) / 2$, print the root and go to step 8
   Else go to step 4
8. Stop.

**Sample Input/output:**

```
Enter the value of x0: -1

Enter the value of x1: -2
______________________________________________________________________

 Iteration   x0         x1        x2         f0        f1       f2


  1       -1.000000 -2.000000 -1.500000 -5.000000 2.000000 -1.750000
  2       -1.500000 -2.000000 -1.750000 -1.750000 2.000000  0.062500
  3       -1.500000 -1.750000 -1.625000 -1.750000 0.062500 -0.859375
  4       -1.625000 -1.750000 -1.687500 -0.859375 0.062500 -0.402344
  5       -1.687500 -1.750000 -1.718750 -0.402344 0.062500 -0.170898
  6       -1.718750 -1.750000 -1.734375 -0.170898 0.062500 -0.054443
  7       -1.734375 -1.750000 -1.742188 -0.054443 0.062500  0.003967
  8       -1.734375 -1.742188 -1.738281 -0.054443 0.003967 -0.025253
  9       -1.738281 -1.742188 -1.740234 -0.025253 0.003967 -0.010647
  10      -1.740234 -1.742188 -1.741211 -0.010647 0.003967 -0.003341
  11      -1.741211 -1.742188 -1.741699 -0.003341 0.003967  0.000313
______________________________________________________________________

Approximate root = -1.741699
```

**Tasks:**

Write a program on the function $f(x) = 2x^3 + 3x - 1$ with starting interval [0, 1] and a tolerance of $10^{-8}$. Show the steps, the program uses to achieve this tolerance (You can count the step by adding 1 to a counting variable i in the loop of the program).

# Problem 2

**OBJECTIVES**: To find the roots of non linear equations using False Position method.

**ALGORITHM:**

1. Decide initial values for $x_1$ and $x_2$ and stopping criterion E.
2. Compute $f_1 = f(x_1)$ and $f2 = f(x_2)$.
3. If $f_1 * f_2 > 0$, $x_1$ and $x_2$ do not bracket any root and go to step 1.
4. Compute $x_0 = x_1 - (f(x_1)(x_2 - x_1)) / (f(x_2) - f(x_1))$ and compute $f_0 = f(x_0)$.
5. If $f_0 = 0$ then $x_0$ is the root of the equation, print the root
6. If $f_1 * f_0 < 0$ then set $x_2 = x_0$ else set $x_1 = x_0$.
7. If $|(x_2 - x_1)/x_2| < E$ then root = $(x_1 + x_2) / 2$, print the root and go to step 8
   Else go to step 4
8. Stop.

**Sample Input/output:**

```
Enter the value of x0: -1

Enter the value of x1: 1
________________________________________________________________________

   Iteration  x0          x1         x2         f0        f1        f2
________________________________________________________________________

   1      -1.000000 1.000000 0.513434 -4.540302 1.459698 -0.330761
   2      0.513434  1.000000 0.603320 -0.330761 1.459698 -0.013497
   3      0.603320  1.000000 0.606954 -0.013497 1.459698 -0.000527
   4      0.606954  1.000000 0.607096 -0.000527 1.459698 -0.000021
________________________________________________________________________

Approximate root = 0.607096
```

**Tasks:**

Write a program to perform 3 iterations of the false position method on the function $f(x) = x^3 - 4$, with starting interval [1, 3]. Calculate and show the errors and percentage errors of x0, x1, x2, and x3.

# Session 2

## Problem 1

**OBJECTIVES**: To find the roots of non linear equations using Newton-Raphson method.

**ALGORITHM:**

1. Assign an initial value for x, say $x_0$ and stopping criterion E.
2. Compute $f(x_0)$ and $f'(x_0)$.
3. Find the improved estimate of $x_0$
   $x_1 = x_0 - f(x_0) / f'(x_0)$
4. Check for accuracy of the latest estimate.
   If $|(x_1 - x_0)/x_1| < E$ then stop; otherwise continue.
5. Replace $x_0$ by $x_1$ and repeat steps 3 and 4.

**Sample Input/output:**

```
 ENTER THE TOTAL NO. OF POWER:::: 3
  x^0::-3
  x^1::-1
  x^2::0
  x^3::1
  THE POLYNOMIAL IS ::: 1x^3 0x^2 -1x^1 -3x^0
  INTIAL X1---->3

**************************************
   ITERATION    X1     FX1     F'X1
**************************************
   1           2.192  21.000 26.000
   2           1.794  5.344  13.419
   3           1.681  0.980  8.656
   4           1.672  0.068  7.475
   5           1.672  0.000  7.384
**************************************

    THE ROOT OF EQUATION IS 1.671700
```

**Tasks:**

Write a program to perform all iterations of the Newton-Raphson method using Horner's rule for any function. Show the table with iterations, values, errors and percentage errors of all variables.

## Problem 2

**OBJECTIVES:** To find the roots of non linear equations using Secant method.

**ALGORITHM:**

1. Decide two initial points $x_1$ and $x_2$ and required accuracy level E.
2. Compute $f_1 = f(x_1)$ and $f_2 = f(x_2)$
3. Compute $x_3 = (f_2 x_1 - f_1 x_2) / (f_2 - f_1)$
4. If $|(x_3 - x_2) / x_3| > E$, then
   set $x_1 = x_2$ and $f_1 = f_2$
   set $x_2 = x_3$ and $f_2 = f(x_3)$
   go to step 3
   Else
   set root = $x_3$
   print results
5. Stop.

**Sample Input/output:**

```
Enter the value of x1: 4

Enter the value of x2: 2
____________________________________________________________________

  Iteration    x1          x2          x3          f(x1)        f(x2)


  1         4.000000   2.000000   9.000000   -10.000000   -14.000000
  2         2.000000   9.000000   4.000000   -14.000000    35.000000
  3         9.000000   4.000000   5.111111    35.000000   -10.000000
  4         4.000000   5.111111   5.956522   -10.000000   -4.320987
  5         5.111111   5.956522   5.722488   -4.320987     1.654063
  6         5.956522   5.722488   5.741121    1.654063    -0.143084
  7         5.722488   5.741121   5.741659   -0.143084    -0.004015
  8         5.741121   5.741659   5.741657   -0.004015     0.000010
____________________________________________________________________

    Approximate root = 5.741657
```

**Tasks:**

Modify the above program using Horner's rule to iterate until the absolute value of the residual is less than a given tolerance (Let tolerance be an input instead of E).

# Session 3

## Problem 1

**OBJECTIVES**: To find the roots of non linear equations using Newton's method.

**ALGORITHM:**

1. Obtain degree and co-efficient of polynomial (n and $a_i$).
2. Decide an initial estimate for the first root ($x_0$) and error criterion, E.

   *Do while n >1*

3. Find the root using Newton-Raphson algorithm

   $x_r = x_0 - f(x_0) / f'(x_0)$

4. Root (n) = $x_r$
5. Deflate the polynomial using synthetic division algorithm and make the factor polynomial as the new polynomial of order n-1.
6. Set $x_0 = x_r$ [Initial value of the new root]

   *End of Do*

7. Root (1) = $- a_0 / a_1$
8. Stop

**Sample Input/output:**

```
Enter the degree of the equation: 2

Enter the coefficients of the equation: -10    -4    1

Root No. 1   -1.74166
Root No. 2   5
```

**Tasks:**

Modify the above program to show the table with all iterations and values of all variables.
Test the program for $x^3 - 6x^2 + 11x - 6 = 0$

# Problem 2

**OBJECTIVES**: To find the roots of non linear equations using Modified Bisection method.

**ALGORITHM:**

1. Choose lower limit **a** and upper limit **b** of the interval covering all the roots.
2. Decide the size of the increment interval $\Delta x$
3. set $x_1 = a$ and $x_2 = x_1 + \Delta x$
4. Compute $f_1 = f(x_1)$ and $f_2 = f(x_2)$
5. If $(f_1 * f_2)>0$, then the interval does not bracket any root and go to step 9
6. Compute $x_0 = (x_1 + x_2)/2$ and $f_0 = f(x_0)$
7. If $(f_1 * f_2)<0$, then set $x_2 = x_0$
   Else set $x_1 = x_0$ and $f_1 = f_0$

8. If $|(x_2 - x_1)/ x_2| < E$, then

        root = $(x_1+x_2) / 2$
        write the value of root
        go to step 9
   Else
        go to step 6

9. If $x_2 < b$, then set $a = x_2$ and go to step 3
10. Stop.

**Sample Input/output:**

```
Enter the maximum power: 2
Enter the coefficients (from maximum power): 1 -4 -10
Enter the lower and upper limit: -2 6

Between -1.74375 and -1.7375 there is a root -1.74375
Between 5.7375 and 5.74375 there is a root 5.74375
```

**Tasks:**

Modify the above program to show the table with all iterations and values of all variables.
Test the program for $x^3 - 7x^2 + 15x - 9 = 0$.

# Session 4

## Problem 1

**OBJECTIVES:** To solve the system of linear equations using Basic Gauss Elimination method.

**ALGORITHM:**

1. Arrange equations such that $a_{11} \neq 0$
2. Eliminate $x_1$ from all but the first equation. This is done as follows:
   i. Normalize the first equation by dividing it by $a_{11}$.
   ii. Subtract from the second equation $a_{21}$ times the normalized first equation.
   iii. Similarly, subtract from the third equation $a_{31}$ times the normalized first equation.
3. Eliminate $x_2$ from the third to the last equation in the new set. We assume that $a'_{22} \neq 0$.
   i. Subtract from the third equation $a'_{32}$ times the normalized first equation.
   ii. Subtract from the fourth equation $a'_{42}$ times the normalized first equation and so on.
4. Obtain solution by back substitution.

**Sample Input/output:**

```
ENTER THE NUMBER OF EQUATIONS = 3
ENTER THE COEFFICENTS OF EQUATIONS  = 2    4   -6   -8
                                      1    3    1    10
                                      2   -4   -2   -12
 Step 1:
                                      2   4   -6   -8
                                      0   1    4   14
                                      0  -8    4   -4

Step 2:
                                      2   4   -6    -8
                                      0   1    4    14
                                      0   0    36   108
SOLUTION OF GIVEN SYSTEM: x1 = 1
                          X2 = 2
                          X3 = 3
```

**Tasks:**

1. Write a code to solve the system of linear equations using Gauss Jacobi method.

# Problem 2

**OBJECTIVES**: To solve the system of linear equations Using Gauss - Jordan Method.

**ALGORITHM:**

1. Normalize the first equation by dividing it by its pivot element.
2. Eliminate $x_1$term from all the other equations.
3. Now, normalize the second equation by dividing it by its pivot element.
4. Eliminate $x_2$term from all the equations, above and below the normalized pivotal equation.
5. Repeat the process until $x_n$is eliminated from all but the last equation.
6. The resultant $b$ vector is the solution vector.

**Sample Input/output:**

```
ENTER THE NUMBER OF EQUATIONS = 3
ENTER THE COEFFICENTS OF EQUATIONS  = 2    4   -6   -8
                                      1    3    1    10
                                      2   -4   -2   -12
 Step 1:

                                      1   2   -3   -4
                                      0   1    4   14
                                      0  -8    4   -4

Step 2:
                                      1   0   -11  -32
                                      0   1    4    14
                                      0   0    1     3

Step 3:
                                      1   0   0    1
                                      0   1   0    2
                                      0   0   1    3
SOLUTION OF GIVEN SYSTEM: x1 = 1
                          X2 = 2
                          X3 = 3
```

**Tasks:**

1. Write a code to solve the system of linear equations using Gauss Seidel method.

# Session 5

**OBJECTIVES**: To fit a straight line and a polynomial using curve fitting regression method.

**ALGORITHMS:**

*Linear Regression:*

1. Read the data values
2. Compute sum of powers and products
   $\sum x_i, \sum y_i, \sum x_i^2, \sum x_i y_i$
3. Check whether the denominator of the equation for b is zero
4. Compute *b and a*
5. Print out the equation
6. Interpolate data, if required

*Polynomial Regression:*

1. Read number of data points *n* and order of polynomial *mp*
2. Read the data values
3. If $n \le mp$, print out "regression is not possible" and stop; else continue
4. Set $m = mp+1$
5. Compute coefficients of **C** matrix
6. Compute coefficients of **B** matrix
7. Solve for the coefficients $a_1, a_2 \ldots a_m$
8. Write the coefficients
9. Estimate the function value at the given value of independent variable
10. Stop

**Sample Input/output:**

```
---------Fitting a Straight line----------
Enter how many values you want for (x,y) : 5
Enter value for x:  1 2 3 4 5
Enter value for y: 3 4 5 6 8

xi     yi     xi*xi     xi*yi

1      3        1         3
2      4        4         8
3      5        9         15
4      6        16        24
5      8        25        40

Sum = 15   26     55     90

 The equation is y = 1.6 + 1.2 x
```

**Sample Input/output:**

```
---------Fitting a polynomial----------
Enter how many values you want for (x,y) : 4
Enter value for x: 1  2    3  4
Enter value for y: 6  11  18  27

The equation is y = 3 + 2x + x²
```

**Tasks:**

1. Write a code to fit a power function using curve fitting regression method.
2. Write a code to fit a hyperbola using curve fitting regression method.

# Session 6

## Problem 1

**OBJECTIVES**: To find the value of *f(x)* for *x* using Lagrange interpolation method.

**ALGORITHM:**

1. Read *x*, *n*
2. *for i = 1* to (n+1) in steps of 1 do read $x_i$, $f_i$ *end for*
3. sum ← 0
4. *for i = 1* to (n+1) in steps of 1 do
5. prodfunc ← 1
6. *for j = 1* to (n+1) in steps of 1 do
7. if (j≠i) then prodfunc ← prodfunc $\times (x - x_j)/(x_i - x_j)$
   *end for*
8. sum ← sum + $f_i$ ×prodfunc
   *end for*
9. Write *x*, sum
10. Stop

**Sample Input/output:**

```
How many record you will be enter: 4
Enter the value of x0: 0
Enter the value of f(x0): 0
Enter the value of x1: 1
Enter the value of f(x1): 2
Enter the value of x2: 2
Enter the value of f(x2): 8
Enter the value of x3: 3
Enter the value of f(x3): 27
Enter X for finding f(x): 2.5
 f(2.5) = 15.312500
```

## Problem 2

**OBJECTIVES**: To find *x* using Newton's divided difference interpolation method.

**ALGORITHM:**

Input: $x_0$, $(x_0)$, $x_1$, $(x_1)$, … , $x_n$, $f(x_n)$
Output: Divided differences $F_{0,0}$, … , $F_{n,n}$
//comment: $(x) = F_{0,0} + \sum^{n}_{i=1}[F_{i,i}(x - x_0) \ldots (x - x_{i-1})]$
Step 1: For $i$ = 0, … , $n$
set $F_{i,0} = f(x_i)$
Step 2: For $i$ = 1, … , $n$
For $j$ = 1, … , $i$
set $F_{i,j}= F_{i,j-1}-F_{i-1,j-1} / x_i-x_{i-j}$
End
End
Output($F_{0,0}$,…$F_{i,i}$,…,$F_{n,n}$)
STOP

**Sample Input/output:**

```
How many record you will be enter: 5

Enter the value of x0: 5

Enter the value of f(x0): 150

Enter the value of x1: 7

Enter the value of f(x1): 392

Enter the value of x2: 11

Enter the value of f(x2): 1452

Enter the value of x3: 13

Enter the value of f(x3): 2366

Enter the value of x4: 21

Enter the value of f(x4): 9702

Enter x for finding f(x): 6


    x        f(x)

    5        150

                    121

    7       392             24

                    265             1

   11       1452            32           0

                    457              1

   13       2366            46

                    917

   21      9702
___________________________________________________________

* * * x = 252 * * *
```

.