Gokhale Education Society's

# R. H. SAPAT COLLEGE OF ENGINEERING, MANAGEMENT STUDIES, AND RESEARCH, NASHIK

## DEPARTMENT OF ELECTRICAL ENGINEERING

## INSTRUCTOR'S LAB MANUAL

Second Year Electrical Engineering 2019 pattern

## SUBJECT CODE: 203148

## SUBJECT NAME: NUMERICAL METHODS AND COMPUTER PROGRAMMING

**List of Experiments:**

**Develop computer program using Python language**

Compulsory Experiments-1,2,3,4,7,10

Any one from 5 or 6 and any one from 8 or 9

**1**. Develop algorithm, draw flow chart and write a program to implement following:

(a) for loop and while loop-- application in Descarte's rule of sign.

(b) if-else and functions-- application in Intermediate value theorem.

(c) 2DArray formation-- application in matrix data entry, transposition and printing matrix.

**2.** Develop algorithm, draw flow chart and write a program to implement Birge-Vieta method.

**3.** Develop algorithm, draw flow chart and write a program to implement Bisection/Regula falsi /Newton-Raphson method (single variable) in following applications (formulate problem statement in anyone of following area(but not limited to))

(a) Finding critical clearing angle in power system stability (give equation directly)

(b) Relation between voltage and current in solar PV.

**4.** Develop algorithm, draw flow chart and write a program to implement curve fitting using least square approximation in following applications (formulate problem statement in any one of following area(but not limited to))

(a) Voltage across capacitor during charging.

(b) Relate temperature and resistance in thermocouple.

(c) Current through inductor during excitation.

**5.** Develop algorithm, draw flow chart and write a program to apply Newton's forward/backward interpolation method in following applications (formulate problem statement in any one of following area(but not limited to))

(a) Voltage across capacitor during charging

(b) Relation of speed and armature voltage in DC motor.

(c) Relation of breakdown voltage and thickness of insulation

**6.** Develop algorithm, draw flow chart and write a program to apply Newton's divided difference/Lagrange's interpolation method in following applications (formulate problem statement in anyone of following area(but not limited to))

(a) Power transfer equation to find power at particular angle

(b) Transformer efficiency at particular loading (data of % loading and efficiency in known at a particular power factor)

(c) Growth of electricity consumption in India (year Vs. Per capita electrical consumption).

**7.** Develop algorithm, draw flow chart and write a program to implement trapezoidal/ Simpson (1/3)rd rule in following applications (formulate problem statement in any one of following area(but not limited to))

(a) RMS/Average value of given waveform.

(b) Finding current through first order circuit (RL series)

(c) kWh consumption from load curve

(d) Magnetic field intensity in overhead transmission line

**8.** Develop algorithm, draw flow chart and write a program to implement Gauss elimination/Jordan in following applications (formulate problem statement in any one of following area(but not limited to))

(a) Electrical network using KVL

(b) Electrical Network using KCL

**9.** Develop algorithm, draw flow chart and write a program to implement Gauss Jacobi/Seidel in following applications (formulate problem statement in any one of following area (but not limited to))

(a) Electrical network using KVL

(b) Electrical Network using KCL

**10.** Develop algorithm, draw flow chart and write a program to implement Modified Euler's/4 th order RK method in following applications (formulate problem statement in any one of following area (but not limited to)

(a) Response of RC series circuit with DC

(b) Response of RL circuit with DC

(c) Deflection angle in MI type instrument

**Guidelines for Student's Lab Journal**

The student's Lab Journal should contain following related to every experiment:

- Theory related to the method
- Algorithm and Flowchart of the method
- Problem statement for numerical method
- Solve numerical using appropriate method
- Program printout with output
- Conclusion
- Ten questions based on method and related Python commands

# EXPERIMENT NO. – 1

**Title:** Descartes' Rule of Sign, Intermediate Value Theorem and 2D array formation

**Aim**: Develop algorithm, draw flowchart, and write a program using python language to implement following

a) For loop and while loop – application in Descartes sign rule of sign
b) If-else – application in Intermediate value theorem
c) 2D array formation – application in matrix data entry, transposition and printing matrix.

**Prerequisites:** 1. Basic Programming knowledge
2. Descartes sign rule of sign and Intermediate value theorem concept

**Objectives:** 1. To understand application of Descartes sign rule of sign
2. To understand application of Intermediate value theorem

**Theory:**

Descartes Sign Rule:

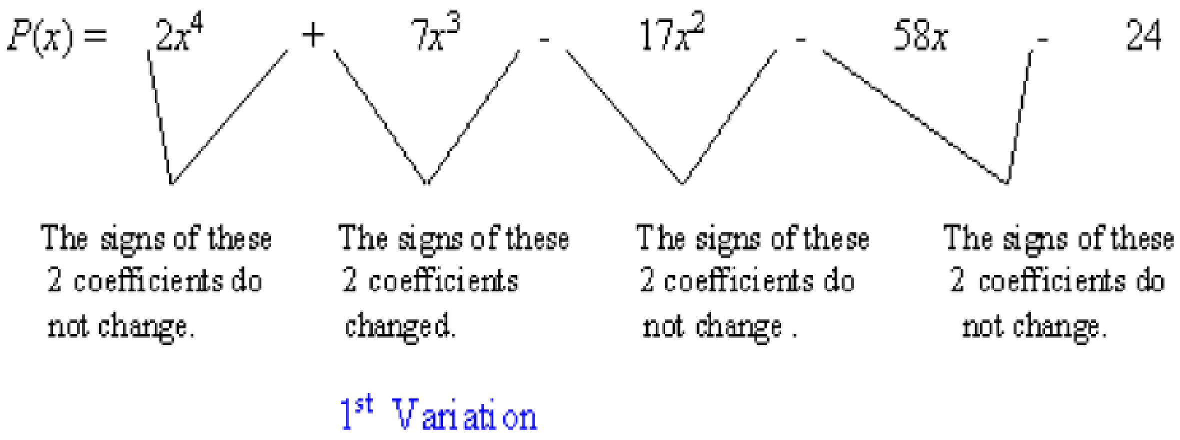
The rule gives us an upper bound number of positive or negative roots of a polynomial. It is not a complete criterion, i.e. it does not tell the exact number of positive or negative roots and their respective intervals.

***"The maximum number of real positive roots equal to number of sign changes in f(x) and the maximum number of real negative roots equal to number of sign changes in f(-x)."***

***In other words***

The number of +ve real roots of $\mathbf{P_n(x) = 0}$ cannot exceed the number of sign changes in $\mathbf{P_n(x)}$ and the number of -ve real roots of $\mathbf{P_n(x) = 0}$ cann't exceed the number of sign changes in $\mathbf{P_n(-x)}$.

$$P(x) = 2x^4 + 7x^3 - 17x^2 - 58x - 24$$

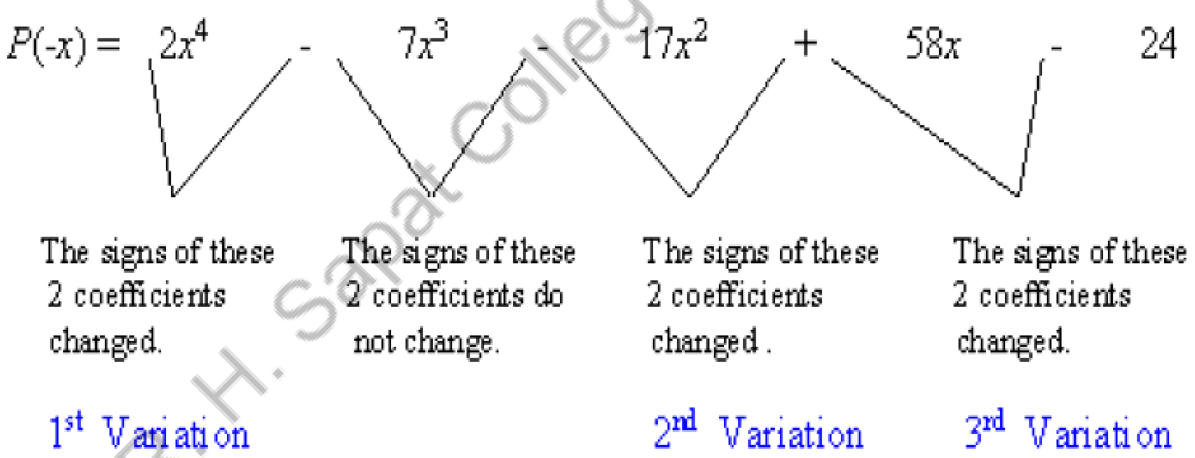



Because there is only 1 variation for the polynomial equation when x is positive, there is only 1 real positive root

Now, let's look at how the signs of consecutive integers vary for $P(-x)$.

Substitute all $x$'s with $(-x)$ and simplify the polynomial.

$P(-x) = 2(-x)^4 + 7(-x)^3 - 17(-x)^2 - 58(-x) -24$
$P(-x) = 2x^4 - 7x^3 - 17x^2 + 58x -24$






Because there are 3 variations for the polynomial equation when x is negative, there 3 real negative roots.

1.Use Descartes's Rule of Signs to determine the possible number of positive and negative real zeros for the given function. f(x) = x4- 3x2 + 2x -1

1) 3 positive roots, 1 negative roots

2) 1 positive roots, 3 negative roots

3) 1 positive roots, 2 negative roots

4) 3 positive roots, 2 negative roots

2.Use Descartes's Rule of Signs to determine the possible number of positive and negative real zeros for the given function.

$$f(x) = x^7 + x^4 + x^2 + x + 9$$

a) 0 positive roots, 3 or 1 negative roots

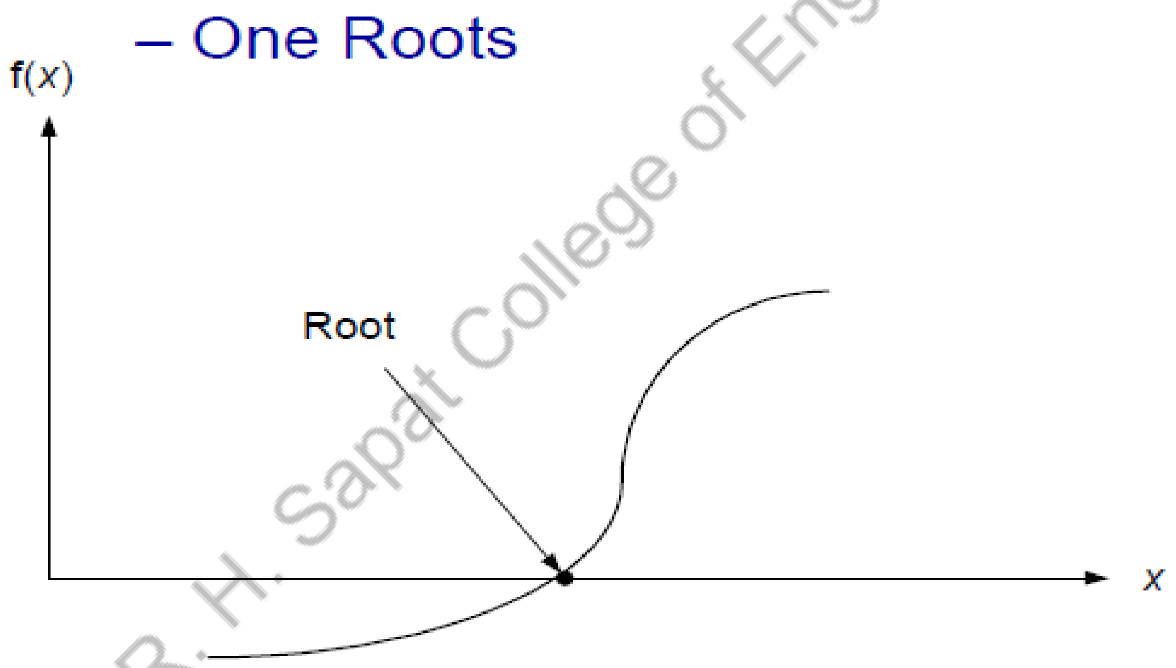
b) 0 positive roots, 1 negative roots

c) 0 positive roots, 2 or 0 negative roots

d) 0 positive roots, 0 negative roots

***The Intermediate Value Theorem***

- Statement: If $f(x)$ is a continuous function on $[a, b]$, and $f(a)$ and $f(b)$ have opposite signs ( $f(a).f(b) < 0$ ) then, there exists at least one root between a and b.






**Example: is there a solution to $x^5 - 2x^3 - 2 = 0$ between x=0 and x=2?**

- **At x=0:**

$0^5 - 2 \times 0^3 - 2 = \mathbf{-2}$

- **At x=2:**

$2^5 - 2 \times 2^3 - 2 = \mathbf{14}$

- Now we know:

at x=0, the curve is below zero

at x=2, the curve is above zero

- And, being a polynomial, the curve will be continuous,
- so somewhere in between, the curve must cross through 0

**Yes, there is a solution to $x^5 - 2x^3 - 2 = 0$ in the interval [0,2]**

**// PROGRAM 1//**

**# Program for Descartes Rule of Signs**

```
a=[]
b=[]
proots=0
nroots=0
coeff=input('Enter polynomial coefficients separated by space ')
a=list(map(float,coeff.split()))
for j in range(0,len(a)-1):
 if a[j]*a[j+1]<0:
    proots=proots+1
for j in range(0,len(a)):
    b.append(a[j]*((-1)**j))
for j in range(0,len(a)-1):
 if b[j]*b[j+1]<0:
    nroots=nroots+1
    croots=len(a)-1-proots-nroots
print("\nApplying Descartes Rule of Signs, we get")
print("No of +ve real roots: atmost",proots)
print("No of -ve real roots: atmost",nroots)
print("No of complex roots: atleast",croots)
```

**Result:**

```
Enter polynomial coefficients separated by space 2 7 -17 -58 -24
Applying Descartes Rule of Signs, we get
No of +ve real roots: atmost 1
No of -ve real roots: atmost 3
No of complex roots: atleast 0
```

**# Program for Intermediate Value Theorem**

#Function Definition

```
import math
func=input('Enter given function:  ')
a=float(input('Enter lower range value a:  '))
b=float(input('Enter upper range value b:  '))
def f(x):
    y=eval(func)
    return(y)
# Process and output section
print('As per Intermediate Value Theorem')
if f(a)*f(b)<0:
 print('Atleast one root lies in the interval [a, b]=',a,b)
elif f(a)*f(b)==0:
 print('any one initial value may the root')
else:
 print('No root lies in the interval [a, b]=',a,  b)
```

**Result:**

```
Enter given function:  math.cos(x)-x*math.exp(x)
Enter lower range value a:  0
Enter upper range value b:  1
As per Intermediate Value Theorem
Atleast one root lies in the interval [a, b]= 0.0 1.0
```

```
#2D array formation
import numpy as np
# Input section matrix
M1 = np.array([[1,4],[5,6]])
M2 = np.array([[1,-4],[3,-2]])
# Output section matrix
# Matrix Addition
print("[M1]+[M2]=",M1+M2)
# Matrix Subtraction
print("[M1]-[M2]=",M1-M2)
# Matrix Multiplication
print("[M1][M2]=",M1.dot(M2))
# Matrix Transpose
print("Transpose of [M1]=",M1.transpose())
```

Result:

```
[M1]+[M2]=    [[2 0]
               [8 4]]
[M1]-[M2]=    [[0 8]
               [2 8]]
[M1][M2]=     [[ 13 -12]
               [ 23 -32]]
Transpose of [M1]=   [[1 5]
                      [4 6]]
```

Questions:

1) Explain Python codes append, len, eval, list
2) Explain program code orally during submission.
3) State and explain Descartes sign rule
4) State and explain intermediate value theorem

| | |
|---|---|
| 01 | In following equation find number of positive real roots, negative real roots and complex roots using Descartes' rule of sign.<br>$x^6+x^5-x^4+x^2+8=0$ |
| 02 | Apply intermediate value theorem to determine existence of root in given interval.<br>$f(x)=x^2+5x+6=0$<br>In interval (a) (-1.5,-2.5) (b) (-2.5, -3.5) (c) (-3.5, -4.5) (d) (-4.5, -5.5) |
| 03 | Following matrix equations are given:<br>$M_1=\begin{bmatrix}1 & 4 & 5\\ 2 & 3 & 6\\ 7 & 8 & 9\end{bmatrix}$ $M_2=\begin{bmatrix}-1 & -4 & -5\\ -2 & -3 & -6\\ -7 & -8 & -9\end{bmatrix}$<br>Perform following operations:<br>(a)M1+M2 (b) M1-M2 (c) M1xM2 (d) transpose of M2 (e) Eigen values of M1 |
| **Group 02** | |
| 01 | In following equation find number of positive real roots, negative real roots and complex roots using Descartes' rule of sign.<br>$x^5+3x^4+2x^2+9=0$ |
| 02 | Apply intermediate value theorem to determine existence of root in given interval.<br>$xe^x=1$<br>In interval (a) (-1,0) (b) (0, 1) (c) (1, 2) |
| 03 | Following matrix equations are given:<br>$M_1=\begin{bmatrix}1 & 4 & 5\\ 6 & 3 & 2\\ 7 & 9 & 8\end{bmatrix}$ $M_2=\begin{bmatrix}-1 & -4 & -5\\ -6 & 3 & -2\\ -7 & -9 & 8\end{bmatrix}$<br>Perform following operations:<br>(a)M1+M2 (b) M1-M2 (c) M1xM2 (d) transpose of M2 (e) Eigen values of M1 |

| Group 03 | |
|---|---|
| 01 | In following equation find number of positive real roots, negative real roots and complex roots using Descartes' rule of sign. $-6x^5 - 4x^4 - 3x^2 - 8 = 0$ |
| 02 | Apply intermediate value theorem to determine existence of root in given interval. $\cos x - xe^x = 0$ In interval (a) (-5,-4) (b) (-3, -2) (c) (0, 1) (d) (2, 3) |
| 03 | Following matrix equations are given: $M_1 = \begin{bmatrix} -1 & 2 \\ 5 & -8 \end{bmatrix}$ $M_2 = \begin{bmatrix} -1 & 1 \\ 1 & -2 \end{bmatrix}$ Perform following operations: (a)M1+M2 (b) M1-M2 (c) M1xM2 (d) transpose of M2 (e) Eigen values of M1 |
| **Group 04** | |
| 01 | In following equation find number of positive real roots, negative real roots and complex roots using Descartes' rule of sign. $8x^6 + 6x^5 + 4x^4 + 3x^3 - x^2 - 8 = 0$ |
| 02 | Apply intermediate value theorem to determine existence of root in given interval. $2x - \log_{10} x = 7$ In interval (a) (1,2) (b) (3, 4) (c) (5, 6) |
| 03 | Following matrix equations are given: $M_1 = \begin{bmatrix} 1 & 2 \\ 5 & 8 \end{bmatrix}$ $M_2 = \begin{bmatrix} -1 & 1 \\ 1 & 2 \end{bmatrix}$ Perform following operations: (a)M1+M2 (b) M1-M2 (c) M1xM2 (d) transpose of M2 (e) Eigen values of M1 |

# EXPERIMENT NO. – 2

**Title:** Solution of polynomial equation using Birge vieta Method.

**Aim**: To write Program in C language to find the root of polynomial equation using Birge Vieta Method.

**Prerequisites:** 1. Concept of root

2. Synthetic division

**Objectives:** 1. To understand solution of polynomial equation using Birge Vieta Method.

2. Use of python coding

**Theory: Birge -Vieta Method**

This is an iterative method to find a real root of the **n**th degree polynomial equation **f(x) = 0** of the form

$$f(x) = a_0x^n + a_1 x^{n-1} + a_2 x^{n-2} + \ . \ . \ . + a_{n-1}x + a_n = 0$$

The real root of equation by Birge Vieta Method is given by

$$p1 = p0 \ - \ \frac{bn}{cn-1}$$

Let 'p' be an approximate estimate of the root of f(x), then synthetic division is performed to obtain the deflected polynomial which is given as $f_{n-1}(x)=b_0x^{n-1}+b_1x^{n-2}+----+b_{n-2}x+b_{n-1}$
Polynomial of n degree can be expressed as
$f(x)=(x-p)\ f_{n-1}(x) +R$ then
$a_0x^n+a_1x^{n-1}+----+a_{n-1}x+a_n= (x-p)\ [b_0x^{n-1}+b_1x^{n-2}+----+b_{n-2}x+b_{n-1}]$
$a_0x^n+a_1x^{n-1}+----+a_{n-1}x+a_n = x\ [b_0x^{n-1}+b_1x^{n-2}+---+b_{n-2}x+b_{n-1}]- p\ [b_0x^{n-1}+b_1x^{n-2}+---+b_{n-1}]$
By comparing the coefficients of like powers of x on both the sides of above equation we get following relation between them
$b_i = a_i + pb_{i-1} \qquad i=1,2,...n$
$R = b_n=f_n(p)$
*To find* $f'_n(p)$*, let us differentiate the equation*
$b_i = a_i + pb_{i-1}$ *with respect to* **p**
$db_i / dp = b_{i-1}\ (dp / dp) + p\ (db_{i-1} / dp)$
*if we substitute* $(db_i / dp) = c_{i-1}$ *then*
$c_{i-1} = b_{i-1} + pc_{i-2}$ ***or***
$c_i = b_i + pc_{i-1} \qquad i=1, 2, . . ., n-1$
Then the $c_{n-1}$ obtained from the last equation is nothing but
$c_{n-1} = db_n / dp = dR / dp = f'_n(p)$
$p_{i+1} = p_i - f_n(p)/ f'_n(p)$ --------- *for solving* ***MCQ (Formula of Newton Raphson's method)***
$p_{i+1} = p_i - \frac{bn}{cn-1}$ --------- *for solving* ***Theory problem***
On convergence this iterative process will give one root **p** of the polynomial equation $f_n(x) = 0$. Now the deflated polynomial equation $Q_{n-1}(x) = 0$ can be used to find the other real roots. This method is often called as **Birge-Vieta method.**

Synthetic division- p is initial approximation given.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| p | $a_0$ | $a_1$ | $a_2$ | -------- | $a_{n-1}$ | $a_n$ | |
| | | $pb_0$ | $pb_1$ | | $pb_{n-2}$ | $pb_{n-1}$ | |
| p | $b_0=a_0$ | $b_1=a_1+pb_0$ | b2 | -------- | $b_{n-1}$ | | $b_n=f(p)$ |
| | | $pc_0$ | $pc_1$ | | $pc_{n-2}$ | | |
| | $c_0=b_0$ | $c_1=b_1+pc_0$ | c2 | -------- | $c_{n-1}=f'(p)$ | | |

**Example :**

Find the real root of **$x^3 - x^2 - x + 1 = 0$ with $p_0 = 0.5$** using birge vieta method In this problem the coefficients are **$a_0 = 1, a_1 = -1, a_2 = -1, a_3 = 1$** Let the initial approximation to **p** be **$p_0 = 0.5$**

**iteration 1**

| | | | | | |
|---|---|---|---|---|---|
| | $a_0$ | $a_1$ | $a_2$ | $a_3$ | |
| $p_0 = 0.5$ | +1 | -1 | -1 | +1 | |
| | | +0.5 | -0.25 | -0.625 | |
| $p_0 = 0.5$ | +1 | -0.5 | -1.25 | +0.375 | ($b_3$ = R) |
| | | +0.5 | 0 | -0.625 | |
| | +1 | 0 | -1.25 | | ($c_2$ = R') |

$$p_1 = p_0 - b_4 / c_3 \quad = 0.5 - \frac{0.375}{-1.25} \quad = 0.5 + \frac{0.375}{1.25} \quad = 0.5 + 0.3 = 0.8$$

**iteration 2**

| | | | | | |
|---|---|---|---|---|---|
| | $a_0$ | $a_1$ | $a_2$ | $a_3$ | |
| $p_1 = 0.8$ | +1 | -1 | -1 | +1 | |
| | | +0.8 | -0.16 | -0.928 | |
| $p_1 = 0.8$ | +1 | -0.2 | -1.16 | +0.072 | ($b_3$ = R) |
| | | +0.8 | +0.48 | | |
| | +1 | +0.6 | -0.68 | | ($c_2$ = R') |

$$p_2 = 0.8 - \frac{0.072}{-0.68} = 0.8 + \frac{0.072}{0.68} = 0.8 + 0.1059 = 0.9059$$

**iteration 3**

| | $a_0$ | $a_1$ | $a_2$ | $a_3$ | |
|---|---|---|---|---|---|
| $p_2 = 0.9059$ | +1 | -1 | -1 | +1 | |
| | | +0.9059 | -0.0852 | -0.9831 | |
| $p_2 = 0.9059$ | +1 | -0.0941 | -1.0852 | +0.0169 | ($b_3$ = R) |
| | | +0.9059 | +0.7354 | | |
| | +1 | +0.8118 | -0.3498 | | ($c_2$ = R') |

$$p_3 = 0.905 - \frac{0.0169}{-0.3498} = 0.905 + \frac{0.0169}{0.3498} = 0.905 + 0.0483 = 0.9533$$

**The exact root after 3rd iteration is 0.9533**



Questions:

1. At end of one iteration the smallest positive root of equation

$$f = x^4 - 3x^3 + 3x^2 - 3x + 2 = 0$$

With initial approximation $p0 = 0$ By Birge Vieta method is

a)1.0356 b) 1.0456 c)1.765 d) 1.0008

2. Birge vieta method is suitable only for solution of **polynomial** type equation.

**Write Algorithm and draw flow chart:**

**# Program for Birge-Vieta Method**

```
a=[]; p=[];
n=int(input('Enter order of equation:'))
a=list(map(float,(input("Enter coefficients:").split())))
p=list(map(float,(input('Enter initial guess:').split())))
it=int(input('Enter number of iterations required:'))
for k in range(0,it):
    b=[];
    c=[];
    b.append(a[0])
    c.append(b[0])
    j=1
    for j in range(1,n+1):
        b.append(a[j]+p[k]*b[j-1])
        c.append(b[j]+p[k]*c[j-1])
    p.append(p[k]-b[n]/c[n-1])
    print("Root of equation at iteration",k+1,"is%.4f"%p[k+1])
```

Result:

```
Enter order of equation:3
Enter coefficients:1 -1 -1 1
Enter initial guess:0.5
Enter number of iterations required:4
Root of equation at iteration 1 is  0.8000
Root of equation at iteration 2 is  0.9059
Root of equation at iteration 3 is  0.9541
Root of equation at iteration 4 is  0.9773
```

| Group 01 | |
|---|---|
| 01 | Find the root of following polynomial equation using Birge-Vieta method correct to three decimal places. Take initial value of root is $p_0 = -0.5$ <br> $f(x) = x^4 + x^3 - 8x^2 - 11x - 3 = 0.$ |
| **Group 02** | |
| 01 | Find the root of following polynomial equation using Birge-Vieta method correct to three decimal places. Take initial value of root is $p_0 = -0.5$ <br> $f(x) = x^4 + 2x^3 + 3x^2 + 4x + 1 = 0.$ |
| **Group 03** | |
| 01 | Find the root of following polynomial equation using Birge-Vieta method correct to three decimal places. Take initial value of root is $p_0 = 0$ <br> $f(x) = x^3 - 3x + 1 = 0.$ |
| **Group 04** | |
| 01 | Find the root of following polynomial equation using Birge-Vieta method correct to three decimal places. Take initial value of root is $p_0 = -1.5$ <br> $f(x) = x^4 + 2x^3 + 3x^2 + 4x + 1 = 0.$ |

# EXPERIMENT NO. – 3

**TITLE :** Solution of Transcendental equation using Bisection Method

**Aim :** Write a program to find the roots of algebraic and transcendental equation using Bisection method.

**Prerequisites:** 1. Concept of root

2. Definition of algebraic and transcendental equation

3. Intermediate value theorem

**Objectives:** 1. To understand solution of transcendental equation using Bisection Method.

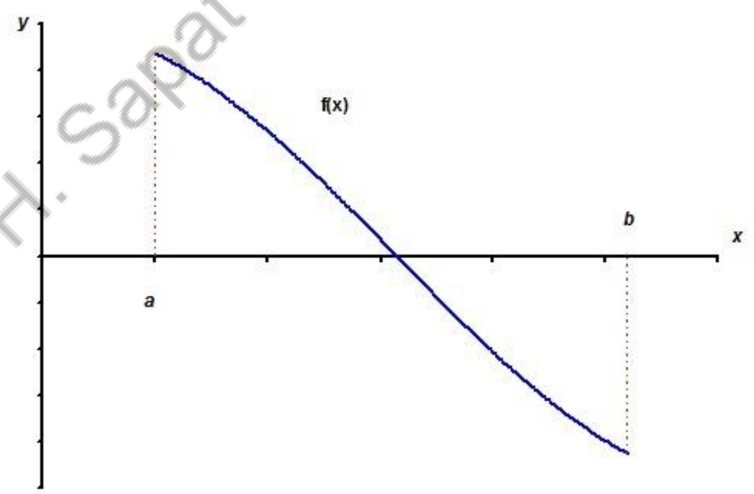
2. To apply intermediate value theorem

3. To use for and do –while loop in c programming

4. To write function c programming.

**Theory: Bisection Method**

This method is based on intermediate value theorem which states that , "if *f(x)* = 0 is a continuous function in the interval [a,b] and f(a) & f(b) having opposite sign ,then the equation *f(x)* = 0 has atleast one root lies between interval [a,b] ,which is given by x0 = (a+b) /2 in bisection method."




If $f(x_0)=0$ , it can conclude that $x_0$ is a root of the equation f(x)=0. Otherwise the root lies either between ($x_0$ & b) or ($x_0$ & a) depending on whether $f(x_0)$ is positive or negative.

When f(a)*f(b) < 0 then root of f(x) lies [a,b] , let f(a) positive and f(b) negative

If $f(x_0)*f(a) < 0$ then root of f(x) lies $[a, x_0]$ ,replace

If $f(x_0)*f(b) < 0$ then root of f(x) lies $[x_0,b]$

The new interval as $[a, x_0]$ or $[b,x_0]$ whose length is (b-a)/2 . Next iteration this interval is bisected at $x_1$ and new interval will be exactly half the length of previous one ie. $(b-a)/2^2$

Repeating this process until the latest interval (which contains the root) is as small as desired.

If repeating this process n times $(b-a)/2^n$ .

If permissible error is E then approximate number of iteration required given by

$(b-a)/2^n \leq E$

$(b-a)/E \leq 2^n$

Taking logs on both sides & simplify it.

$\log((b-a)/E) \leq \log(2^n)$

$(\log(b-a) - \log E) \leq n \log(2)$

$((\log(b-a) - \log E) / \log 2) \leq n$

$$n \geq \frac{log(b-a) - log\ \varepsilon}{log 2}$$

If number of iterations are not given then above formula is used to find number of iterations. Bisection method is called as **bracketing** method because it need two initial value within root lies. This method is always **convergent** means finding value of root which is closer value to exact value of root. The criteria for deciding when to terminate the computation A convergent criteria is to compute the percentage error Er defined by %Er = |(x'r –xr)/ x'r | ×100 where x'r is new value of xr. The computation can be terminated when Er becomes less than a prescribed tolerance say Ep =0.0005

Steps:

1. Find a and b in which f(a) and f(b) are opposite signs for the given equation using intermediate value theorem
2. Initial approximation of root xo = (a+b)/2.
3. If $f(x_0)$ is negative , the root lies between a and $x_0$ and replace b by x0 as goto step 2
4. If $f(x_0)$ is positive , then the root lies between xo and b and replace a by x0 as goto step 2.
5. Repeat the process until any two consecutive values are equal and hence the root.

***Find the positive root of x – cos x = 0 by using bisection method.***

**Solution :**

**Note during trigonometric function x acts as radian angle.**

*Let $f(x) = x - \cos x$ use intermediate value theorem*

*$f(0) = 0 - \cos(0) = 0 - 1 = -1 = -ve$*

*$f(0.5) = 0.5 - \cos(0.5) = -0.37758 = -ve$*

*$f(1) = 1 - \cos(1) = 0.42970 = +ve$*

$f(0.5) \times f(1) < 0$ So root lies between 0.5 and 1

Let $xo = (0.5 + 1)/2 = 0.75$ *as initial root and proceed.*

$f(0.75) = 0.75 - \cos(0.75) = 0.018311 = +ve$

iteration 1 So root lies between 0.5 and 0.75

$x1 = (0.5 + 0.75)/2 = 0.625$

$f(0.625) = 0.625 - \cos(0.625) = -0.18596$

So root lies between 0.625 and 0.750

| *Itr no.* | *a* | *b* | *c=(a+b)/2* | *f(a)* | *f(b)* | *f(c)* | *Interval (b-a)/2* |
|---|---|---|---|---|---|---|---|
| 0 | 0.5 | 1 | 0.75 | -0.37758 | 0.42970 | 0.018311 | 0.25 |
| 1 | 0.5 | 0.75 | 0.625 | -0.37758 | 0.018311 | - 0.18596 | 0.125 |
| 2 | 0.625 | 0.75 | 0.6875 | - 0.18596 | 0.018311 | - 0.085335 | 0.0625 |
| 3 | 0.6875 | 0.75 | 0.71875 | - 0.085335 | 0.018311 | - 0.033879 | 0.03125 |
| 4 | 0.71875 | 0.75 | 0.73438 | - 0.033879 | 0.018311 | -0.0078664 | 0.015625 |
| 5 | 0.73438 | 0.75 | 0.742190 | -0.0078664 | 0.018311 | 0.0051999 | 7.8E-3 |
| 6 | 0.73438 | 0.742190 | 0.73829 | -0.0078664 | 0.0051999 | -0.0013305 | 3.9E-3 |
| 7 | 0.73829 | 0.742190 | 0.7402 | -0.0013305 | 0.0051999 | 0.0018663 | 1.9E-3 |
| 8 | 0.73829 | 0.7402 | 0.73925 | -0.0013305 | 0.0018663 | 0.00027593 | 9.7E-4 |
| 9 | 0.73829 | 0.73925 | 0.73877 | -0.0013305 | 0.00027593 | N.A. | 4.8E-4 |

**The root after 9$^{th}$ iteration is 0.7388.**

```
/*Python program for bisection method to find the root of equation */
#Function Definition
import math
func=input('Enter given function:  ')
def f(x):
    y=eval(func)
    return(y)
# Main Program
# Input Section
a=float(input('Enter initial value of a:'))
```

```
b=float(input('Enter initial value of b:'))
n=int(input('Enter no of iterations n:'))
# Process and output section
if f(a)*f(b)<0:
     print('root lies in the interval [a, b]=',a,b)
     print('The iterative values of root c is')
     for k in range(0,n):
        c=(a+b)/2;
        if f(a)*f(c)<0:
         b=c;
        else:
        a=c;
        print("c= %.4f"%c)
elif f(a)*f(b)= =0:
    print('root is anyone outof initial guess')
else:
     print('No lies in the interval [a, b]=',a,b)
```

Result:

```
Enter given function  math.cos(x)-x*math.exp(x)
Enter initial value of a:0
Enter initial value of b:1
Enter no of iterations n:10
root lies in the interval [a, b]= 0.0 1.0
The iterative values of root c is
c=0.5000
c=0.7500
c=0.6250
c=0.5625
c=0.5312
c=0.5156
c=0.5234
c=0.5195
c=0.5176
c=0.5186
```

# EXPERIMENT NO. – 4

**Title:** Curve fitting of given data

**Aim**: To develop algorithm, draw flow chart and write for curve fitting using least square error approximation

**Prerequisites:** 1. Straight line equation and curve equation solution

2. Derivative of equation

**Objectives:** 1. To understand how to fit value in equation form

2. To understand how to find best fitted value for given data to fit in curve

**Theory: Curve fitting**

When we need to fit given data in the curve with the help of some law. And that law is established such a that squares of errors between actual and approximated law is minimized, it is called least square approximation.

1. Least Square Approximation – First Order

(linear regression)

Fitting a straight line

2. Least Square Approximation –Second Order

(polynomial regression)

Fitting quadratic equation or parabola

Least Square Approximation – First Order

Lets consider the mathematical equation for straight line

$y = a+bx$ actual line

& curve line is given by $y=f(x)$

$f(x)= a+bx$ approximated line

Error $e_i$ for various data points given by

$e_i = y_i - f(x_i)$

$e_i = y_i - a - bx_i$

Total Error which is to be minimize is given by

$$S= \sum_{i=1}^{n} [y_i - a - bx_i]^2 \quad \text{------------- (1)}$$

n is total data points.

- Then the method of least squares consists of minimizing S means minimizing sum of squares of error.
- Hence it is required to choose a & b such that S minimum.
  $\partial S/ \partial a = 0$ & $\partial S/ \partial b = 0$ --------(2)

$$\partial S/ \partial a = -2 \sum_{i=1}^{n} [y_i - a - bx_i] = 0$$

$$\sum_{i=1}^{n} y_i - \sum_{i=1}^{n} a - \sum_{i=1}^{n} bx_i = 0$$

n n

$\sum_{i=1} y_i = n\,a + b\sum_{i=1} x_i$ --------------(3)

$\partial S/\partial b = -2\sum_{i=1}^{n} x_i\,[y_i - a - bx_i] = 0$

$\sum_{i=1}^{n} [x_i\,y_i - a\,x_i - bx_i^2\,] = 0$

$\sum_{i=1}^{n} x_i\,y_i - a\sum_{i=1}^{n} x_i - b\sum_{i=1}^{n} x_i^2 = 0$

$\sum_{i=1}^{n} x_i\,y_i = a\sum_{i=1}^{n} x_i + b\sum_{i=0}^{n} x_i^2$ --------------(4)

$\sum_{i=1}^{n} y_i = n\,a + b\sum_{i=0}^{n} x_i$ --------------(3)

$\sum_{i=1}^{n} x_i\,y_i = a\sum_{i=1}^{n} x_i + b\sum_{i=0}^{n} x_i^2$ --------------(4)

- Equation (3) & (4) is called as normal equations and can be easily solved to get a & b for getting straight line eq.

Questions:

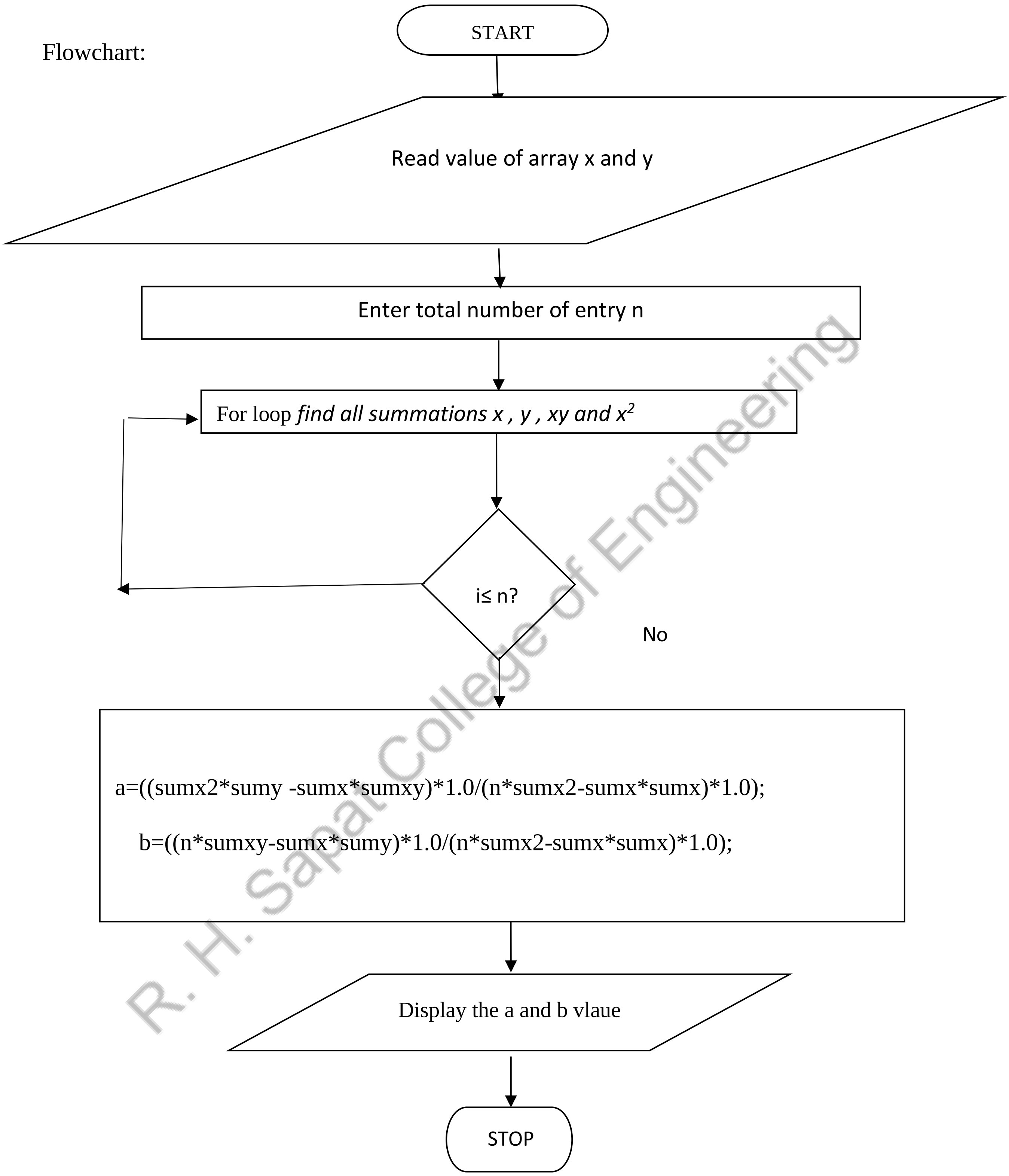
1. A procedure used for finding the equation of a straight line which provides the best approximation for the relationship between the independent and dependent variables is the

a. correlation analysis
b. mean squares method
c. least squares method
d. most squares method

Algorithm:

Step1: Start
Step2: Read values of x and y
Step3: read total entry n
Step4: calculate all summations of x, of y , $x^2$ and xy
Step5: find best suitable value of a and b with formula
a=((sumx2*sumy -sumx*sumxy)*1.0/(n*sumx2-sumx*sumx)*1.0);
b=((n*sumxy-sumx*sumy)*1.0/(n*sumx2-sumx*sumx)*1.0);
Step6: display result
Step7 : Stop.

Flowchart:


START
Read value of array x and y
Enter total number of entry n
For loop find all summations x , y , xy and x²
i≤ n?
No
a=((sumx2*sumy -sumx*sumxy)*1.0/(n*sumx2-sumx*sumx)*1.0);
b=((n*sumxy-sumx*sumy)*1.0/(n*sumx2-sumx*sumx)*1.0);
Display the a and b vlaue
STOP

**// PROGRAM FOR CURVE FITTING //**

```
# Program for curve fitting using LSE approximation in python.
x=[]
y=[]
xy=[]
x2=[]

coeff=input('Enter x values separated by space  ')
x=list(map(float,coeff.split()))
coeff=input('Enter y values separated by space  ')
y=list (map(float,coeff.split()))
n=len(x)
for i in range(0,n):
   xy.append(x[i]*y[i])
   x2.append(x[i]*x[i])
sum_x=sum(x)
sum_y=sum(y)
sum_xy=sum(xy)
sum_x2=sum(x2)
b = (n*sum_xy - sum_x*sum_y)/(n*sum_x2 - (sum_x*sum_x));
a = (sum_y - b*sum_x)/n;
print("Required Equation is")
print("y = %0.4f + %0.4fx"%(a,b))
```

Result:

```
Enter x values separated by space  20 30 40 50 60 70
Enter y values separated by space  800.3 800.4 800.6 800.7 800.9 801
Required Equation is
y = 799.9943 + 0.0146x
```

| Group 01 | |
|---|---|
| 01 | In an experiment capacitor of 10µF is connected to 100V DC supply through a resistance of 1kΩ. The reading of voltmeter connected across capacitor is given in following table. Find the best fit equation for voltage across capacitor using least square error method |

| t(msec) | 1 | 5 | 10 | 15 | 20 | 30 |
|---|---|---|---|---|---|---|
| Vc(volts) | 10 | 40 | 65 | 75 | 85 | 95 |

| Group 02 | |
|---|---|
| 01 | In speed control of DC shunt motor using armature voltage control method, following readings are recorded. Find the best fit equation for speed using least square error method |

| Va(Volts) | 10 | 50 | 100 | 150 | 200 | 220 |
|---|---|---|---|---|---|---|
| N(rpm) | 70 | 345 | 680 | 1025 | 1360 | 1495 |

| Group 03 | |
|---|---|
| 01 | In a 100Ω, platinum RTD is connected to measure temperature of resistance of a liquid. During experiment following readings are obtained. Find the best fit equation for resistance using least square error method |

| t(°C) | 0 | 200 | 300 | 600 | 900 |
|---|---|---|---|---|---|
| R(Ω) | 100 | 185 | 215 | 345 | 450 |

| Group 04 | |
|---|---|
| 01 | In an experiment capacitor of 10µF is connected to 100V DC supply through a resistance of 1kΩ. The reading of ammeter connected in series with resistance is given in following table. Find the best fit equation for voltage across capacitor using least square error method |

| t(msec) | 1 | 5 | 10 | 15 | 20 | 30 |
|---|---|---|---|---|---|---|
| i(mA) | 90 | 60 | 35 | 25 | 15 | 5 |

# EXPERIMENT NO. – 5

**TITLE :**Interpolation at given point by using Newton's Forward Difference formula.

**Aim :** Write a program to find the value at given point by using Newton's Forward Difference Interpolation method.

**Prerequisites:** 1. Interpolation and extrapolation

2. Single dimensional array in C programming.

**Objectives:** 1. To understand Interpolation using Newton's Forward Difference formula.

2. To apply two dimensional array in C programming.

3. To use for loop in c programming.

**Theory: Newton's forward interpolation formula**

Let *y = f(x) denote a function which takes the values y0, y1, y2 ………., yn corresponding* to the values *x0, x1, x2 ………., xn.*

Let suppose that the values of x i.e., x0, x1, x2 ………., xn. are equidistant/uniform .

$x_1 = x_0 + h$ ; $x_2 = x_1 + h$ ; *and so on* $x_n = x_{n-1} + h$ ;

Therefore $x_i = x_0 + i\,h$, *where* $i = 1,2, \dots , n$

Let Pn(x) be a polynomial of the $n^{th}$ degree in which *x is such that*

$y_i = f(x_i) = P_n(x_i),\ i = 0,1,2, \dots n$

Let us assume $P_n(x)$ in the form given below

$P_n(x) = a_0 + a_1 (x - x_0) + a_2 (x - x_0)(x - x_1) + \dots + a_n (x - x_0)(x - x_1) \dots (x - x_{n-1})$ ---------- (1)

This polynomial contains the n + 1 constants $a_1, a_2, a_3, \dots a_n$ can be found as follows :

$Pn(x_0) = y_0 = a_0$ *(setting* $x = x_0$, *in eq.(1) )*

$\boldsymbol{a_0 = y_0}$

Similarly $y_1 = a_0 + a_1 (x_1 - x_0)$ *(setting* $x = x_1$, *in eq.(1) )*

$y_1 - y_0 = a_1 (x_1 - x_0)$

$\Delta y_0 = y_1 - y_0 = a_1 (x_1 - x_0) = a_1 h$

$\boldsymbol{a_1 = \Delta y_0 / h}$

Similarly *by setting* $x = x_2$ *then* $x_3, x_4$ & $x_n$, *in eq.(1)*

$\boldsymbol{a_2 = (\Delta y_1 - \Delta y_0) / 2! h^2 = \Delta^2 y_0 / 2!\, h^2}$

$\boldsymbol{a_3 = \Delta^3 y_0 / 3!\, h^3}$

$\boldsymbol{a_n = \Delta^n y_0 / n!\, h^n}$

Putting these values in (1), we get

$P_n(x) = y_0 + (\Delta y_0 / h)(x - x_0) + (\Delta^2 y0 / 2!\ h^2)(x - x_0)(x - x_1) + \ldots\ldots\ldots$

$+ \ldots\ldots\ldots + (\Delta^n y0 / n!\ h^n)(x - x_0)(x - x_1) \ldots\ldots (x - x_{n-1})$ ---------(2)

*We have* $y_i = f(x_i) = P_n(x_i),\ i = 0,1,2, \ldots. n$

Eq.(2) is called Newton Difference interpolation polynomial

*By substituting* $(x - x_0)/h = u$ *, the above equation becomes*

$$f(x) = y_0 + u\Delta y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \frac{u(u-1)(u-2)}{3!}\Delta^3 y_0 + \frac{u(u-1)(u-2)(u-3)}{4!}\Delta^4 y_0 + \cdots \ldots\ldots$$

$$where\ u = \frac{(x - x_0\ )}{h}$$

The above equation is known as **Gregory-Newton forward formula or Newton's forward interpolation formula.**

**Note :**

1. This formula is applicable **only** when the interval of difference is uniform.

2. This formula apply forward differences of y0, hence this is used to interpolate the values of y nearer to beginning value of the table ( i.e., x lies between x0 to x1 or x1 to x2 )

**FORWARD DIFFERENCE TABLE**

| x | y | $\Delta y$ | $\Delta^2 y$ | $\Delta^3 y$ | $\Delta^4 y$ |
|---|---|---|---|---|---|
| $x_0$ | $y_0$ | | | | |
| | | $\Delta y_0$ | | | |
| $x_1$ | $y_1$ | | $\Delta^2 y_0$ | | |
| | | $\Delta y_1$ | | $\Delta^3 y_0$ | |
| $x_2$ | $y_2$ | | $\Delta^2 y_1$ | | $\Delta^4 y_0$ |
| | | $\Delta y_2$ | | $\Delta^3 y_1$ | |
| $x_3$ | $y_3$ | | $\Delta^2 y_2$ | | |
| | | $\Delta y_3$ | | | |
| $x_4$ | $y_4$ | | | | |

The formula can be written in form given below

$$y_r = y_0 + r \cdot \Delta y_0 + \frac{r(r-1)}{2!}\Delta^2 y_0 + \frac{r(r-1)(r-2)}{3!}\Delta^3 y_0 + \cdots$$

$$r = \frac{x_r - x_0}{h}$$

**Find the values of y at x = 21 from the following data using appropriate interpolation technique**

| x | 20 | 23 | 26 | 29 |
|---|---|---|---|---|
| y | 0.3420 | 0.3907 | 0.4384 | 0.4848 |

Solution :

Step 1.Since x = 21 is nearer to beginning of the table. Hence we apply Newton's forward formula.

Step 2. Construct the difference table

| x | y | $\Delta y$ | $\Delta^2 y$ | $\Delta^3 y$ |
|---|---|---|---|---|
| 20 | 0.3420 | | | |
| | | 0.0487 | | |
| 23 | 0.3907 | | -0.001 | |
| | | 0.0477 | | -0.0003 |
| 26 | 0.4384 | | -0.0013 | |
| | | 0.0464 | | |
| 29 | 0.4384 | | | |

Step 3. Write down the formula and put the various values :

$$y_r = y_0 + r \cdot \Delta y_0 + \frac{r(r-1)}{2!}\Delta^2 y_0 + \frac{r(r-1)(r-2)}{3!}\Delta^3 y_0 + \cdots$$

$$r = \frac{x_r - x_0}{h}$$

*h=23-20 =3*

*r= (21 – 20) / 3 = 0.3333*

*y ($x_r$=21)= y(21) = 0.3420 + (0.3333)( 0.0487)+ (0.3333) (-0.6666) ( -0.001)*
*+ (0.3333) (-0.6666)(-1.6666) ( -0.0003)*

***y(21) = 0.3583***

Questions

1. What are different types of difference operator?
2. What is interpolation?

**<u>Algorithm</u> :**

Step - 1 : Read number of total data points and values of those data points. i.e. *x* and *y = f(x)*.

Step - 2 : Read value of $x = x_r$ at which *y* is to be interpolated.

Step - 3 : Calculate forward differences.

$$\Delta y_0 = y_1 - y_0$$
$$\Delta^2 y_0 = \Delta y_1 - \Delta y_0$$
$$\Delta^3 y_0 = \Delta^2 y_1 - \Delta^2 y_0 \text{ and so on.}$$

Step - 4 : Calculate,

$$h = x_1 - x_0 \text{ and}$$
$$r = \frac{x_r - x_0}{h}$$

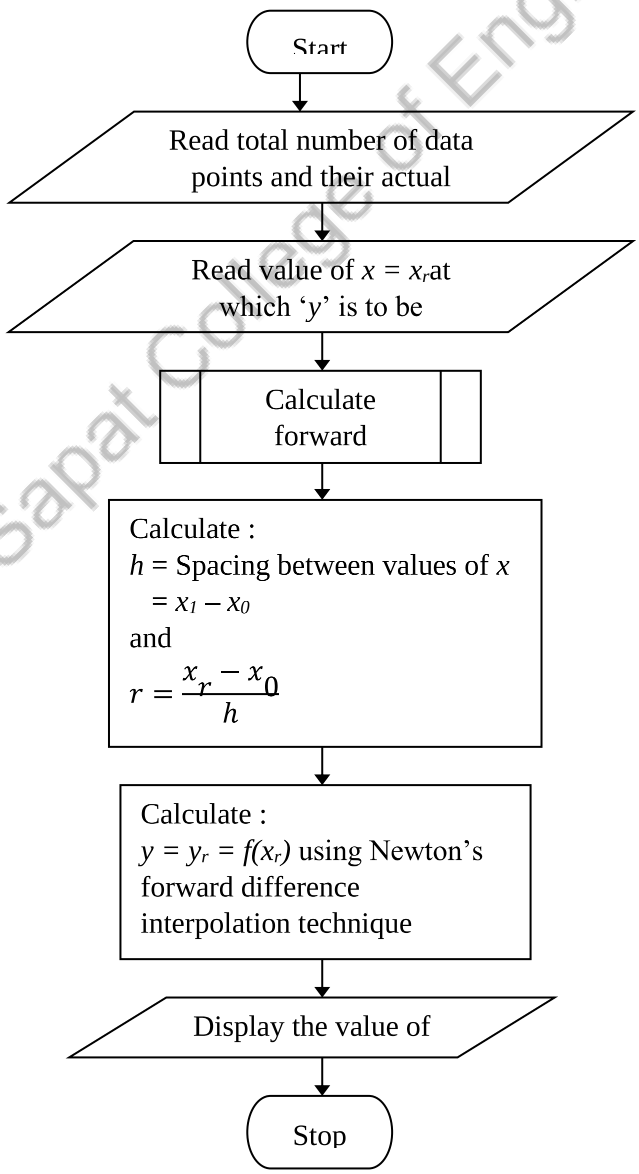
Step - 5 : Calculate,

$$y_r = y_0 + r \cdot \Delta y_0 + \frac{r(r-1)}{2!}\Delta^2 y_0 + \frac{r(r-1)(r-2)}{3!}\Delta^3 y_0 + \cdots$$

Step - 6 : The interpolated value of $y$ at $x = x_r$ is equal to $y_0$ i.e. $f(x_r) = y_r$.

Step - 7 : Display $x_r$ and $y_r$ on the screen and stop.

**Flowchart :**



**GROUP-A**

1) In an experiment efficiency of transformer is recorded at 0.8 pf lagging at various loading conditions which is given in following table

| n(% loading) | 20 | 40 | 60 | 80 | 100 |
|---|---|---|---|---|---|
| % efficiency | 92 | 94 | 96 | 98 | 96 |

Calculate efficiency of transformer at (i) 25% of loading and (ii) 85% of loading.

**GROUP-B**

1) The power handling capacity of transmission line with respect to load angle delta is given in following table

| $\delta$(deg) | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|
| Power Transfer (MW) | 54 | 80 | 103 | 123 | 139 |

Determine power handling capacity of transmission line at (i) $\delta = 25^0$ (ii) $\delta = 55^0$

**GROUP-C**

1) The per capita energy consumption in India with year is given in following table. (source: https://cea.nic.in/wp-content/uploads/pdm/2020/12/growth_2020.pdf)

| $Year$ | 1997 | 2002 | 2007 | 2012 | 2017 |
|---|---|---|---|---|---|
| Per capita consumption (kWh) | 465 | 559 | 672 | 884 | 1122 |

Determine Per capita consumption (kWh) in (i) 2000 (ii) 2015

**GROUP-D**

1) A capacitor of 10uf is connected to DC supply of 50V through 1kΩ resistor. The following are the observations of voltage across capacitor with time

| t(msec) | 1 | 6 | 11 | 16 | 21 |
|---|---|---|---|---|---|
| Vc(Volts) | 5 | 23 | 34 | 40 | 44 |

Calculate voltage across capacitor at (i) 3msec and (ii) 18mesc using appropriate interpolation formula.

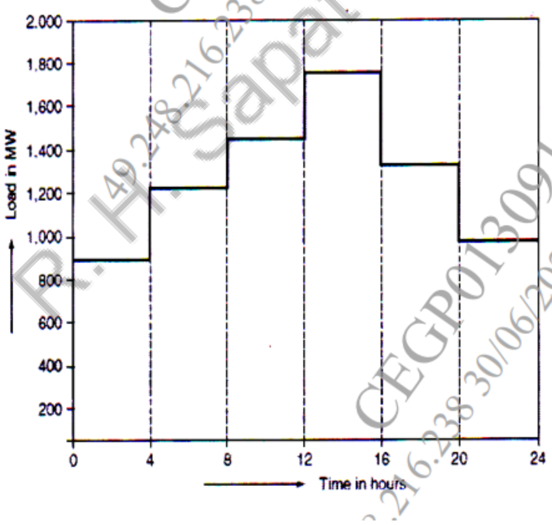
```
# Progarm for Newton Forward Interpolation
import math
# Input section
x=[];y=[]; d1y=[];d2y=[];d3y=[];d4y=[];d5y=[];
n=int(input("Enter number of entries: " ))
x=list(map(float,input("Enter x data points: ").split()))
y=list(map(float,input("Enter y data points: ").split()))
xr=float(input("Enter xr required: " ))
h=x[1]-x[0]
u=(xr-x[0])/h;
# Difference table calculation
for i in range(1,n):
    d1y.append(y[i]-y[i-1])
for i in range(1,n-1):
    d2y.append(d1y[i]-d1y[i-1])
for i in range(1,n-2):
    d3y.append(d2y[i]-d2y[i-1])
for i in range(1,n-3):
```

```
    d4y.append(d3y[i]-d3y[i-1])
for i in range(1,n-4):
    d5y.append(d4y[i]-d4y[i-1])
print(y[0], d1y[0] ,d2y[0], d3y[0], d4y[0] , d5y[0])
# Implementation of Newton Forward Formula
yr=(y[0])+(u*d1y[0])+((1/2)*u*(u-1)*d2y[0])+((1/6)*(u)*(u-1)*(u-
2)*d3y[0])+((1/24)*(u)*(u-1)*(u-2)*(u-3)*d4y[0])+((1/120)*(u)*(u-1)*(u-2)*(u-3)*(u-
4)*d5y[0])
print("\nNewton Forward Interpolation Result")
print("Required Solution is")
print("At x = %.3f, y = %.3f"%(xr,yr))
```

**Result:**

```
Enter number of entries: 6
Enter x data points: 0 4 8 12 16 20
Enter y data points: 900 1200 1450 1700 1300 1000
Enter xr required: 3
900.0  300.0  -50.0  50.0  -700.0  2100.0
Newton Forward Interpolation Result
Required Solution is
At x = 3.000, y = 1177.014
```

The total load of the power system is not constant but varies thoughout the day and reaches a different peak value from one day to another. It follows a particular hourly load cycle over a day. There will be different discreate load levels at each period as shown in the following figure. From the same diagram, the time in hours versus active power consumption by load (MW) data is tabulated. What will be the power consumed by the load when the time in hours will be 3 hrs? Use Newton's forward interpolation method. **[6]**




| Time in hours | 0 | 4 | 8 | 12 | 16 | 20 |
|---|---|---|---|---|---|---|
| active power consumption by load (MW) | 900 | 1200 | 1450 | 1700 | 1300 | 1000 |

# EXPERIMENT NO. – 6

**TITLE :** Program on Lagrange's Interpolation method for unequally spaced data.

**Aim :** Write a program to find the value at given point by using Lagrange's interpolation method.

**Prerequisites:** 1. Interpolation and extrapolation

2. Single dimensional array in C programming.

**Objectives:** 1. To understand Interpolation using Lagrange's formula.

2. To apply single dimensional array in C programming.

3. To use for loop & if statement in c programming.

**Theory:** Lagrange's interpolation

Lagrange's interpolation method applicable for uniform as well as non-uniform spaced data types. Let interpolating polynomial for fitting data (x0,y0),(x1,y1)….. (xn ,yn)

$f(x)= A_1 (x-x_1)( x-x_2)\ldots\ldots (x-x_n) + A_2(x-x_0)( x-x_2)\ldots\ldots (x-x_n) +\ldots\ldots + A_n(x-x_0)( x-x_1)\ldots\ldots (x-x_{n-1})$

*it is required to find values of A1,A2,….An setting*

*first* $x=x_0$ , $f(x)=y_0$

$$A_1 = \frac{y0}{(x0-x1)(x0-x2)\ldots\ldots(x0-xn)}$$

*Then* $x=x_1$ , $f(x)=y_1$

$$A_2 = \frac{y1}{(x1-x0)(x1-x2)\ldots\ldots(x1-xn)}$$

*Then* $x=x_2$ , $f(x)=y_2$

$$A_3 = \frac{y2}{(x2-x0)(x2-x1)\ldots\ldots(x2-xn)}$$

*Hence in general*

$x=x_2$ , $f(x)=y_2$

$$A_n = \frac{yn}{(xn-x0)(xn-x2)\ldots\ldots(xn-xn-1)}$$

*Then interpolating equation becomes*

$$f(x)=\frac{y0}{(x0-x1)(x0-x2)\ldots\ldots(x0-xn)}(x-x_1)(x-x_2)\ldots\ldots(x-x_n) + \frac{y1}{(x1-x0)(x1-x2)\ldots\ldots(x1-xn)}(x-x_0)(x-x_2)\ldots\ldots(x-x_n) + \ldots\ldots + \frac{yn}{(xn-x0)(xn-x2)\ldots\ldots(xn-xn-1)}(x-x_0)(x-x_1)\ldots\ldots(x-x_{n-1})$$

$$f(x)=\frac{(x-x1)(\,x-x2)\ldots\ldots(x-xn)}{(x0-x1)(\,x0-x2)\ldots\ldots(x0-xn)}\,y0+\frac{(x-x0)(\,x-x2)\ldots\ldots(x-xn)}{(x1-x0)(\,x1-x2)\ldots\ldots(x1-xn)}\,y1+\ldots\ldots$$

$$\ldots\ldots\ldots.+\frac{(x-x0)(\,x-x1)\ldots\ldots(x-xn-1)}{(xn-x0)(\,xn-x2)\ldots\ldots(xn-xn-1)}\,yn$$

**Find y(2) from following data using Lagrange's interpolation**

| X | 0 | 1 | 4 | 6 |
|---|---|---|---|---|
| Y | 1 | -1 | 1 | -1 |

Solution :

$$y(x)=\frac{(x-x1)(\,x-x2)\ldots\ldots(x-xn)}{(x0-x1)(\,x0-x2)\ldots\ldots(x0-xn)}\,y0+\frac{(x-x0)(\,x-x2)\ldots\ldots(x-xn)}{(x1-x0)(\,x1-x2)\ldots\ldots(x1-xn)}\,y1+\ldots\ldots$$

$$\ldots\ldots\ldots.+\frac{(x-x0)(\,x-x1)\ldots\ldots(x-xn-1)}{(xn-x0)(\,xn-x2)\ldots\ldots(xn-xn-1)}\,yn$$

$$y(x)=\frac{(x-x1)(\,x-x2)\,(x-x3)}{(x0-x1)(\,x0-x2)(x0-x3)}\,y0+\frac{(x-x0)(\,x-x2)\,(x-x3)}{(x1-x0)(\,x1-x2)\,(x1-x3)}\,y1+$$

$$\frac{(x-x0)(\,x-x1)(x-x3)}{(x2-x0)(\,x2-x1)\,(x2-x3)}\,y2+\frac{(x-x0)(\,x-x1)(x-x2)}{(x3-x0)(\,x3-x1)\,(x3-x2)}\,y3$$

$$y(2)=\frac{(2-1)(\,2-4)\,(2-6)}{(0-1)(\,0-4)(0-6)}\,(1)+\frac{(2-0)(\,2-4)\,(2-6)}{(1-0)(\,1-4)\,(1-6)}\,(-1)+\frac{(2-0)(\,2-1)(2-6)}{(4-0)(\,4-1)\,(4-6)}\,(1)+$$

$$\frac{(2-0)(\,2-1)(2-4)}{(6-0)(\,6-1)\,(6-4)}\,(-1)$$

$y(2)= -\ 0.3333\ -\ 1.0666\ +0.3333\ +0.0666$

$y(2)= -\ 1$

Questions :

1. If the number of entries are n the interpolating polynomial is of ______ order.
2. Lagrange interpolation method is mainly suitable for ________ spaced data.
3. Enlist the method used for equally spaced data only
4. Enlist method used for both equally and unequally spaced data.

| GROUP-A | |
|---|---|
| 1) | In an experiment efficiency of transformer is recorded at 0.8 pf lagging at various loading conditions which is given in following table |

| n(% loading) | 20 | 30 | 50 | 75 | 100 |
|---|---|---|---|---|---|
| % efficiency | 92 | 94 | 96 | 98 | 96 |

Calculate efficiency of transformer at (i) 25% of loading and (ii) 85% of loading.

| GROUP-B | |
|---|---|
| 1) | The power handling capacity of transmission line with respect to load angle delta is given in following table |

| $\delta$(deg) | 20 | 35 | 45 | 60 | 80 |
|---|---|---|---|---|---|
| Power Transfer (MW) | 55 | 92 | 113 | 139 | 158 |

Determine power handling capacity of transmission line at (i) $\delta = 30^0$

| GROUP-C | |
|---|---|
| 1) | A capacitor of 10uf is connected to DC supply of 50V through 1kΩ resistor. The following are the observations of current though capacitor with time |

| t(msec) | 0 | 10 | 15 | 25 | 30 |
|---|---|---|---|---|---|
| ic(mA) | 50 | 20 | 10 | 5 | 3 |

Calculate current though capacitor at (i) 5msec

| GROUP-D | |
|---|---|
| 1) | A capacitor of 10uf is connected to DC supply of 50V through 1kΩ resistor. The following are the observations of voltage across capacitor with time |

| t(msec) | 0 | 10 | 15 | 25 | 30 |
|---|---|---|---|---|---|
| Vc(Volts) | 0 | 32 | 38 | 46 | 48 |

Calculate voltage across capacitor at (i) 5msec



**Algorithm :**

Step - 1 : Read total number of data points and actual values of these data points. i.e. $x$ and $y = f(x)$.

Step - 2 : Read the value of $x = x_r$ at which $y$ is to be calculated.

Step - 3 : If there are (n+1) data points, then calculate,

$$L_k(x) = \frac{\prod_{i=0,i\neq k}^{n} (x_r - x_i)}{\prod_{i=0,i\neq k}^{n} (x_k - x_i)}$$

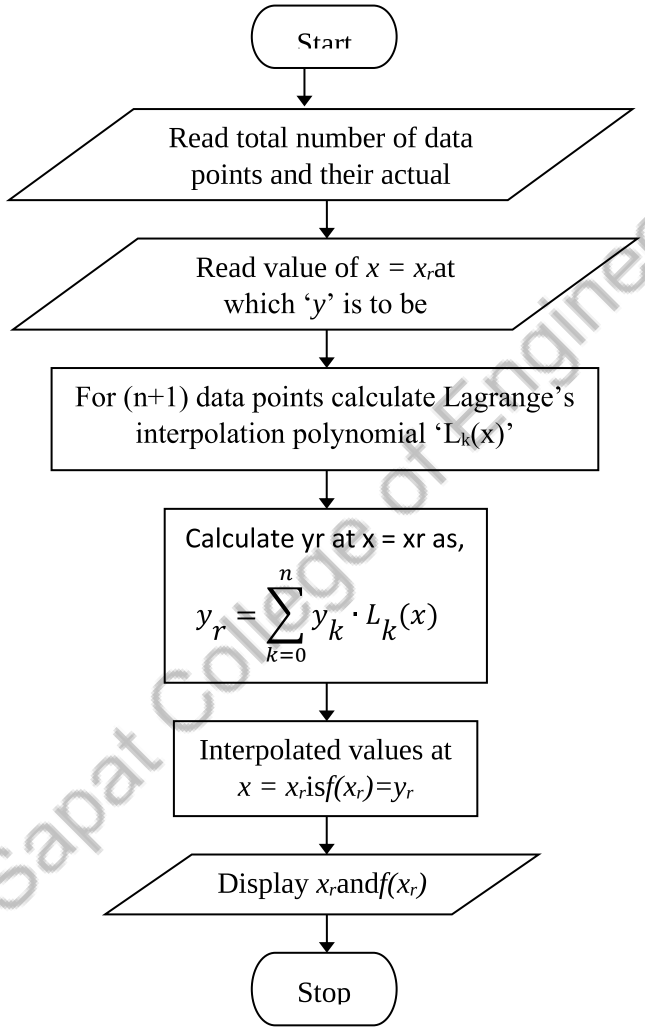
Then calculate,

$$y_r = \sum_{k=0}^{n} y_k \cdot L_k(x)$$

Step - 4 : The interpolated value at $x = x_r$ is equal to $y_r$ i.e. $f(x_r) = y_r$.

Step - 5 : Display values of $x_r$ and $f(x_r)$ on the screen and stop.

**Flowchart :**



**/* LAGRANGES INTERPOLATION*/**

```
# Progarm for Lagrange Interpolation in python
import math
# Input section
x=[];y=[];
n=int(input("Enter number of entries: " ))
x=list(map(float,input("Enter x values: ").split()))
y=list(map(float,input("Enter y values: ").split()))
xr=float(input("Enter xr at yr required: " ))
sum=0
for i in range(0, n):
    prod=1.0
    for j in range(0, n):
        if j!=i:
            prod=prod*(xr-x[j])/(x[i]-x[j])
    sum=sum+prod*y[i]
print("At x = %.3f, y = %.3f"%(xr,yr))
```

**Result:**
**Enter number of entries: 4**
**Enter x values: 0 1 4 6**
**Enter y values: 1 -1 1 -1**
**Enter xr at yr required: 2**
**At xr=  2.0 yr= -1.000**

# EXPERIMENT NO. – 7

**TITLE :**Solution of Numerical integration using Simpson's 1/3rd rule .

**Aim :** Write a program for Solution of Numerical integration using Simpson's 1/3rd rule .

**Prerequisites:** 1. Concept of integration

2. To write function c programming.

**Objectives:** 1. To understand Solution of Numerical integration using Simpson's 1/3rd rule .

2. Different method for Numerical integration with formula and graph

3. Graphical representation of all methods solution of Numerical integration.

**Theory :Simpson's 1/3rd rule** .

Newtons Quadrature cote formula is used for replacing any complicated integrand or tabulated data by an appropriatate function.

$Let\ \ I = \int_a^b y\, dx$ where *y takes values y0,y1,y2,……yn for x= x0,x1,x2,……xn*

Let the interval integration (a,b) br divided into n equal sub intervals ,each width *h= (b-a)/n*

*x0=a ,x1=x0+h, x2=x0+2h ……. xn=x0+nh =b*

$$I = \int_{x0}^{x0+nh} y\, dx$$

We have Newtons Quadrature cote formula

$$I = \int_{x0}^{x0+nh} y\, dx \ = nh\left[y0 + \frac{n}{2}\,\Delta y0 + \frac{n(2n-3)}{12}\,\Delta^2 y0 + \frac{n(n-2)^2)}{24}\,\Delta^3 y0 + \cdots \ldots \ldots \ldots\right]$$

By putting n=2 in Newtons Quadrature cote formula and taking the curve through (x0,y0),(x1,y1)&(x2,y2) as polynimial of degree 2 so as higher order differences are vanish

$$I1 = \int_{x0}^{x0+2h} y\, dx \ = 2h\left[y0 + \frac{2}{2}\,\Delta y0 + \frac{2(2*2-3)}{12}\,\Delta^2 y0\right]$$

$$= 2h\left[y0 + \Delta y0 + \tfrac{1}{6}\,\Delta^2 y0\right]$$

$$= 2h\left[y0 + (y1 - y0) + \tfrac{1}{6}\,(\Delta y1 - \Delta y0)\right]$$

$$= \tfrac{2h}{6}\,[6y0 + 6(y1 - y0) + (y2 - 2y1 + y0)]$$

$$I1 = \int_{x0}^{x0+2h} y\, dx \;\; = \frac{h}{3}\, [y0 + 4y1 + y2]$$

Similarly taking the curve through (x2,y2),(x3,y3)&(x4,y4)

$$I2 = \int_{x0+2h}^{x0+4h} y\, dx \;\; = \frac{h}{3}\, [y2 + 4y3 + y4]$$

Similarly taking the curve through $(x_{n-2}, y_{n-2})$,( $x_{n-1}, y_{n-1}$)&( $x_n, y_n$)

$$In = \int_{x0+(n-2)h}^{x0+nh} y\, dx \;\; = \frac{h}{3}\, [y_{n-2} + 4y_{n-1} + y_n]$$

Adding All integrals ,we get

$$I = \int_{x0}^{x0+nh} y\, dx \;\; = \frac{h}{3}\left[(y_0 + y_n) + 2(y_2 + y_4 + \cdots + y_{n-2)}\;) + 4(\; y_1 + y_3 + \cdots + y_{n-1})\right]$$

Which is known as simpsons one-third rule .

**Evaluate** $\boldsymbol{I = \int_0^6 \frac{1}{(1+x)}\, dx}$ **using simpsons 1/3rd rule of integration taking 8 intervals**

Solution : *x0=0, xn=6 and n=8 hence* $h = \frac{6-0}{8} = 0.75$

$x_i = x_0 + nh$ *i= 1,2,3,4,5,6,7,8.*

Calculate all y = $\frac{1}{(1+x)}$ at x0, x1,x2,….x8

| *x* | $y = \frac{1}{(1+x)}$ |
|---|---|
| *x0 = 0* | *y0=1* |
| *x1=0.75* | *y1=0.5714* |
| *x2=1.5* | *y2=0.4* |
| *x3=2.25* | *y3=0.3076* |
| *x4=3* | *y4=0.25* |
| *x5=3.75* | *y5=0.2105* |
| *x6=4.5* | *y6=0.1818* |
| *x7=5.25* | *y7=0.16* |
| *x8=6* | *y8=0.1428* |

$$I = \int_0^6 \frac{1}{(1+x)}\,dx \quad = \frac{h}{3}\,[(y_0 + y_8) + 2(y_2 + y_4 + y_6\,) + 4(\;y_1 + y_3 + y_5 + y_7)]$$

$$I = \int_0^6 \frac{1}{(1+x)}\,dx$$

$$= \frac{0.75}{3}\,[(1 + 0.1428) + 2(0.4 + 0.25 + 0.1818) + 4(0.5714 + 0.3076 + 0.2015 + 0.16)]$$

$I = \int_0^6 \frac{1}{(1+x)}\,dx \;= 1.9421$

Questions :

S1 batch : Evaluate $I = \int_{0.5}^{0.9} x^{\frac{1}{2}}\; e^{-x}\,dx$ using simpsons 1/3rd rule of integration taking 8 intervals

S2 batch: Evaluate $I = \int_0^{\pi/2} \sqrt{sin\,\theta}\,d\theta$ using simpsons 1/3rd rule of integration taking 6 intervals

S3 batch: Evaluate $I = \int_0^6 \frac{1}{(1+x)}\,dx$ using simpsons 1/3rd rule of integration taking 8 intervals

1. What function in C
2. What is integration
3. Graphically show representation of simpsons rule
4. Different types of difference operator
5. Different loops in c

**Algorithm :**

Function to be integrated is predefined.

Step - 1 : Read the lower and upper limits of integration.
Step - 2 : Read value of ‘h’ or read number of intervals ‘n’.

$$n = \frac{x_n - x_0}{h}$$

Step - 3 : Calculate :

$$y_0 = f(x)|_{x=x0}$$
$$y_1 = f(x)|_{x=x1}$$
$$y_2 = f(x)|_{x=x2} \qquad \text{and so on.}$$

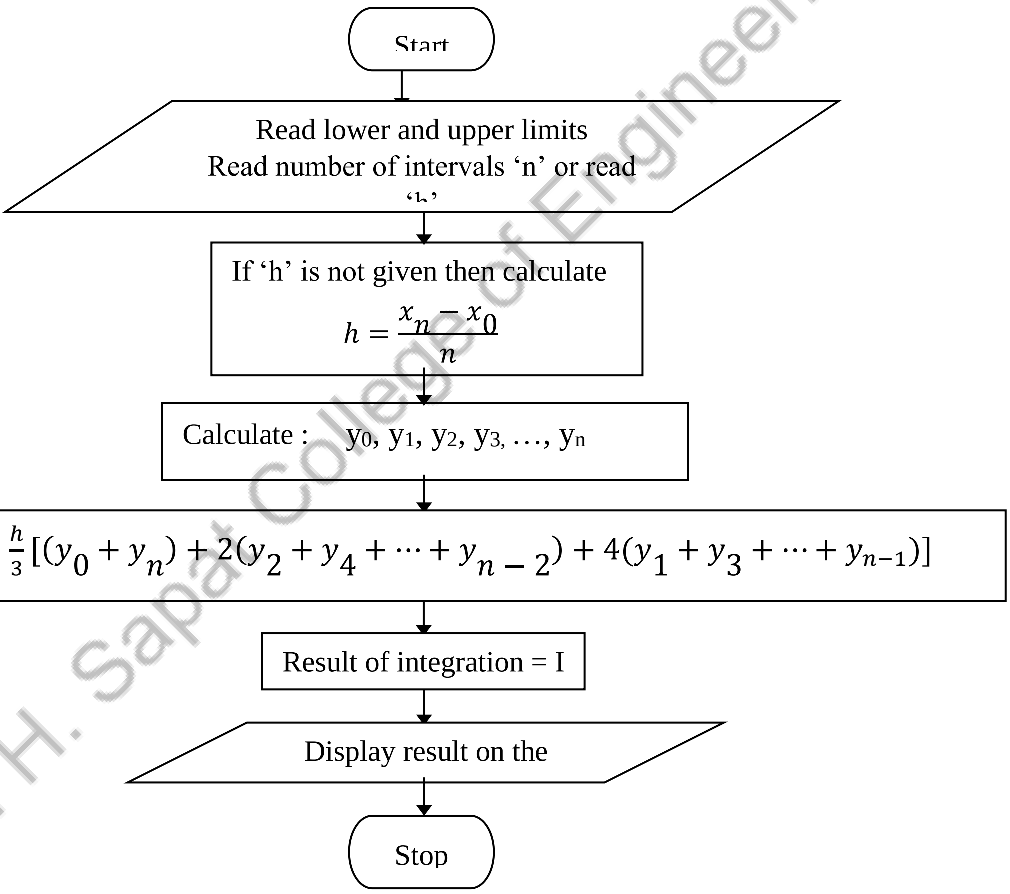
Step - 4 : Calculate,

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

Step - 5 : Display the value of result of integration 'I' on the screen.

Step - 6 : Stop.

**<u>Flowchart</u> :**



**/*SIMSONS 1/3 RULE OF INTEGRATION */**

```
#Program in python for Simpson's 1/3 rd rule of integration
#Function Definition
def y(x):
    z=1/(1+x)
    return z
# Main Program
# Input Section
x0=float(input('Enter the value of x0:'))
xn=float(input('Enter the value of xn:'))
n=int(input('Enter the value of subintervals n:'))
#process section
h=(xn-x0)/n ;
print('step size h=',h)
print('Formula = (h/3)*[(y0+yn)+2*(y2+y4+...) +4*(y1+y3+y5..)] ')
sum=y(x0)+y(xn)+4*y(x0+h);
for k in range(3,n+1,2):
   sum=sum+4*y(x0+k*h)+2*y(x0+(k-1)*h);
result=(h/3)*(sum);
print("The result of integration is=%.3f"%result);
```

**Enter the value of x0:0**
**Enter the value of xn:6**
**Enter the value of subintervals n:8**
**step size h= 0.75**
**Formula = (h/3)*[(y0+yn)+2*(y2+y4+...) +4*(y1+y3+y5..)]**
**The result of integration is= 1.9512**

# EXPERIMENT NO. – 9

**TITLE :**Solution of Linear simultaneous algebraic equation using iterative Gauss Seidel Method

**Aim :** Write a program to find the solution of given equations using Gauss Seidel Method.

**Prerequisites:** 1. Linear simultaneous equations

2. Direct methods and iterative method

**Objectives:** 1. 1. To understand Solution of Linear simultaneous algebraic equation

2. Python programming

**Theory : Gauss Seidel Method.**

Gauss seidel method is an iterative method for solution of linear simultaneous equations.

Let the system of equation is given by

$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 = b_1$

$a_{21} x_1 + a_{22} x_2 + a_{23} x_3 = b_2$

$a_{31} x_1 + a_{32} x_2 + a_{33} x_3 = b_3$

*check condition if true continue process else change equation sequence*

$|a11| \geq |a12| + |a13|$

$|a22| \geq |a21| + |a23|$

$|a33| \geq |a31| + |a32|$

*Then Rearranging system of equations*

$$x1 = \frac{1}{a11} \ ( b1 - a12\, x2 \ - \ a13\, x3)$$

$$x2 = \frac{1}{a22} \ ( b2 - a21\, x1 \ - \ a23\, x3)$$

$$x3 = \frac{1}{a33} \ ( b3 - a31\, x1 \ - \ a32\, x2)$$

*Assume initial values as* $x_1^{(0)}$ , $x_2^{(0)}$ , $x_3^{(0)}$ *then value to updated in next iteration*

$$x1^{(1)} = \frac{1}{a11} \ ( b1 - a12\, x2^{(0)} - \ a13\, x3^{(0)})$$

$$x2^{(1)} = \frac{1}{a22} \ ( b2 - a21\, x1^{(1)} - \ a23\, x3^{(0)})$$

$$x3^{(1)} = \frac{1}{a33}\ (b3 - a31\ x1^{(1)} - a32\ x2^{(1)})$$

*To perform next iteration take $x_1^{(1)}$, $x_2^{(1)}$, $x_3^{(1)}$ as initial value and find next approximate values*

$$x1^{(2)} = \frac{1}{a11}\ (b1 - a12\ x2^{(1)} - a13\ x3^{(1)})$$

$$x2^{(2)} = \frac{1}{a22}\ (b2 - a21\ x1^{(2)} - a23\ x3^{(1)})$$

$$x3^{(2)} = \frac{1}{a33}\ (b3 - a31\ x1^{(2)} - a32\ x2^{(2)})$$

*In general k+1 th iteration*

$$x1^{(k+1)} = \frac{1}{a11}\ (b1 - a12\ x2^{(k)} - a13\ x3^{(k)})$$

$$x2^{(k+1)} = \frac{1}{a22}\ (b2 - a21\ x1^{(k+1)} - a23\ x3^{(k)})$$

$$x3^{(k+1)} = \frac{1}{a33}\ (b3 - a31\ x1^{(k+1)} - a32\ x2^{(k+1)})$$

*The process will continue till desires accuracy.*

**Solve System of equation using Gauss seidel method**

$8x1 - 3x2 + 2x3 = 20$

$4x1 + 11x2 - x3 = 33$

$6x1 + 3x2 + 12x3 = 35$

*With initial values* $x1 = 3,\ x2 = 2\ \&\ x3 = 1$

*In general k+1 th iteration*

$$x1^{(k+1)} = \frac{1}{8}\ (20 + 3\ x2^{(k)} - 2x3^{(k)})$$

$$x2^{(k+1)} = \frac{1}{11}\ (33 - 4\ x1^{(k+1)} + x3^{(k)})$$

$$x3^{(k+1)} = \frac{1}{12}\ (35 - 6\ x1^{(k+1)} - 3\ x2^{(k+1)})$$

*Iteration 1*

$$x1^{(1)} = \frac{1}{8}\ (20 + 3\ x2^{(0)} - 2x3^{(0)})$$

$$x2^{(1)} = \frac{1}{11}\ (33 - 4\ x1^{(1)} + x3^{(0)})$$

$x3^{(1)} = \frac{1}{12}\ (35 - 6\,x1^{(1)} - 3\,x2^{(1)})$

*hence*

$x1^{(1)} = \frac{1}{8}\ (20 + 3 * 2 - 2 * 1) = 3$

$x2^{(1)} = \frac{1}{11}\ (33 - 4 * 3 + 1) = 2$

$x3^{(1)} = \frac{1}{12}\ (35 - 6 * 3 - 3 * 2) = 0.91667$

New approximate values are $x1^{(1)} = 3$ , $x2^{(1)} = 2$ , $x3^{(1)} = 0.91667$

Iteration 2

$x1^{(2)} = \frac{1}{8}\ (20 + 3\,x2^{(1)} - 2x3^{(1)})=3.0208$

$x2^{(2)} = \frac{1}{11}\ (33 - 4\,x1^{(2)} + x3^{(1)})=1.9848$

$x3^{(2)} = \frac{1}{12}\ (35 - 6\,x1^{(2)} - 3\,x2^{(2)})=0.9100$

| | | |
|---|---|---|
| $x1^{(1)} = 3$ | $x2^{(1)} = 2$ | $x3^{(1)} = 0.9166$ |
| $x1^{(2)} = 3.0208$ | $x2^{(2)} = 1.9848$ | $x3^{(2)} = 0.9100$ |
| $x1^{(3)} = 3.0168$ | $x2^{(3)} = 1.9857$ | $x3^{(3)} = 0.9118$ |
| $x1^{(4)} = 3.0166$ | $x2^{(4)} = 1.9859$ | $x3^{(4)} = 0.9118$ |
| $x1^{(5)} = 3.0166$ | $x2^{(5)} = 1.9858$ | $x3^{(5)} = 0.9118$ |

S1batch : Solve System of equation using Gauss seidel method

$2x1 + x2 + x3 = 5$

$5x1 + 5x2 + 2x3 = 15$

$2x1 + x2 + 4x3 = 8$

*With initial values* $x1 = 0,\ x2 = 0\ \&\ x3 = 0$

S2 batch :Solve System of equation using Gauss seidel method

$8x1 - 3x2 + 2x3 = 20$

$4x1 + 11x2 - x3 = 33$

$6x1 + 3x2 + 12x3 = 35$

*With initial values* $x1 = 3,\ x2 = 2\ \&\ x3 = 1$

S3 batch : Solve System of equation using Gauss seidel method

$20x1 + x2 - 2x3 = 17$

$3x1 + 20x2 - x3 = -18$

$2x1 - 3x2 + 20x3 = 25$

$\textit{With initial values } x1 = 1,\ x2 = 0\ \& x3 = 0$

**Questions :**

1) What are direct and iterative methods for solution of simultaneous equation?
2) Graphically show representation of linear simultaneous equation

**Algorithm :**

Step - 1 : Define all equations using preprocessor directives

#define X1(x2,x3) $(\frac{1}{a11}(\ b1 - a12\ x2\ -\ a13\ x3))$

#define X1(x2,x3) $(\frac{1}{a22}\ \ (\ b2 - a21\ x1\ -\ a23\ x3)\ )$

#define X1(x2,x3) $(\frac{1}{a33}\ \ (\ b3 - a31\ x2\ -\ a32\ x2)\ )$

Step - 2 : Read the initial approximation given x1,x2,x3

Step - 3 : Read number of iteration n

Step - 4 : Use for loop i=0 ,i<=n ? yes then continue ,no then stop

Step - 5 :Calculate new approximations

y1=X1(x2,x3)
y2=X2(y1,x3)
y3=X3(y1,y2)
x1=y1 x2=y2 x3=y3 updated values

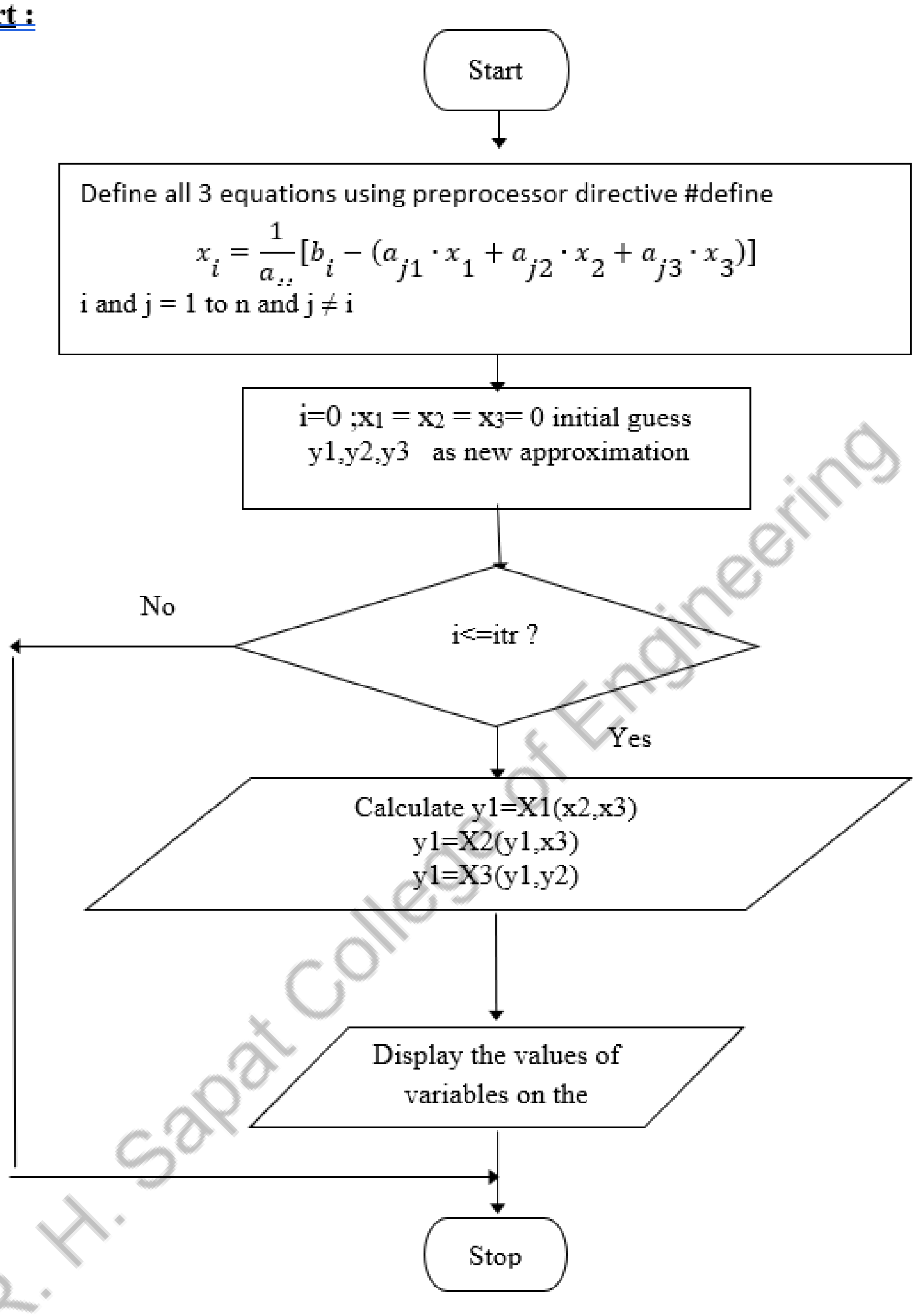
if(fabs(y1-x1)<EP && fabs(y2-x2)<EP && fabs(y3-x3)<EP ) got desired accurcy

break for loop

Step - 6 : Display the values of variables $x_i$ ;i = 1 to n on the screen

Step - 7 : Stop.

**Flowchart :**


Start
Define all 3 equations using preprocessor directive #define
$x_i = \frac{1}{a_{ii}}[b_i - (a_{j1} \cdot x_1 + a_{j2} \cdot x_2 + a_{j3} \cdot x_3)]$
i and j = 1 to n and j ≠ i
i=0 ;x1 = x2 = x3= 0 initial guess
y1,y2,y3 as new approximation
No
i<=itr ?
Yes
Calculate y1=X1(x2,x3)
y1=X2(y1,x3)
y1=X3(y1,y2)
Display the values of
variables on the
Stop

```
/*PROGRAM FOR GAUSS SEIDAL ITERATIVE METHOD*/
#Program for Gauss seidel method
#Function Definition
import math
def X1(x2,x3):
   return((20 +3*(x2) - 2*(x3))/8)
def X2(x1,x3):
   return((33 -4*(x1) + (x3))/11)
def X3(x1,x2):
   return((35 - 6*(x1) - 3*(x2))/12)
x1=float(input('Enter the value of x1:'))
x2=float(input('Enter the value of x2:'))
x3=float(input('Enter the value of x3:'))
print('display all values x1,    x2,    x3 ')
print('                    %0.3f'%x1,  '%0.3f'%x2,  '%0.3f'%x3,);
n=5
for k in range(0,n):
   y1=X1(x2,x3);
   y2=X2(y1,x3);
   y3=X3(y1,y2);
   x1 = y1;
   x2 = y2;
   x3 = y3;
   print('                   %0.3f'%x1,     '%0.3f'%x2,     '%0.3f'%x3,);
```

Result:

```
Enter the value of x1:3
Enter the value of x2:2
Enter the value of x3:1
display all values x1,    x2,    x3
                    3.0000 2.0000 1.0000
                   3.0000 2.0000 0.9167
                   3.0208 1.9848 0.9100
                   3.0168 1.9857 0.9118
                   3.0167 1.9859 0.9118
                   3.0168 1.9859 0.9118
```

Prepared by Checked by Approved by