Session:2024-25

Babasaheb Bhimrao Ambedkar University

Lab assignment

Course: B.Tech. (Computer Engineering)
SECOND Year (IIIrd Semester)
Subject: NUMERICAL TECHNIQUES LAB
Code: ECS-353
Name: Avnish Srivastava
Roll Number: 238209

S No	**Problem STATEMENT**	**Page No.**
1	ALGORITHM & PROGRAM FOR IMPLIMANTATION OF **BISECTION METHOD**	2-3
2	ALGORITHM & PROGRAM IMPLIMANTATION OF **REGULAR-FALSI METHOD**	4-5
3	ALGORITHM & PROGRAM IMPLIMANTATION OF **NEWTON REPHSON METHOD**	6-7
4	ALGORITHM & PROGRAM IMPLIMANTATION OF **NEWTONS FORWORD METHOD OF INTERPOLUTION**	8-9
5	ALGORITHM & **PROGRAM IMPLIMANTATION OF NEWOTN'S BACKWORD METHOD OF INTERPOLATION**	10-11
6	ALGORITHM & PROGRAM IMPLIMANTATION OF **GAUSS'S FORWORD METHOD OF INTERPOLATION**	12-13
7	ALGORITHM & **GAUSS'S BACKWARD METHOD OF INTERPOLATION**	15-17
8	ALGORITHM & **Algorithm of LAGRANGE'S INTERPOLATION FORMULA**	18-19
9	ALGORITHM & PROGRAM IMPLIMANTATION OF **EULER'S METHOD**	20-20
10	ALGORITHM & PROGRAM IMPLIMANTATION OF **MODIFIED EULER'S METHOD**	21-22
11	ALGORITHM & PROGRAM TO IMPLEMENT **STIRLING'S METHOD**	23-24
12	ALGORITHM & PROGRAM IMPLIMANTATION OF **RUNGA KUTTA METHOD**	25-25
13	ALGORITHM & PROGRAM TO IMPLEMENT **TRAPEZOIDAL METHOD**	26-27
14	ALGORITHM & PROGRAM TO IMPLEMENT **SIMPSON'S 1/3rd METHOD OF NUMERICAL INTEGRATION**	28-29
15	ALGORITHM & PROGRAM TO IMPLEMENT **SIMPSON'S 3/8TH METHOD OF NUMERICAL INTEGRATION**	30-31
16	ALGORITHM & PROGRAM TO **Draw frequency chart like histogram**	32-33
17	ALGORITHM & PROGRAM TO **Draw frequency chart like Pi-Chart**	34-34

1 Algorithm of BISECTION METHOD.

Step 01. Start of the program.

Step 02. Input the variable x1, x2 for the task.

Step 03. Check f(x1)*f(x2)<0

Step 04. If yes proceed

Step 05. If no exit and print error message

Step 06. Repeat 7-11 if condition not satisfied

Step 07. X0=(x1+x2)/2

Step 08. If f(x0)*f(x1)<0

Step 09. X2=x0

Step 10. Else

Step 11. X1=x0

Step 12. Condition:

Step 13. | (x1-x2)x1) | < maximum possible error or f(x0)=0

Step 14. Print output

Step 15. End of program.

PROGRAM IMPLIMANTATION OF **BISECTION METHOD.**

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
#include<process.h>
#include<string.h>
#define EPS 0.00000005
#define F(x) (x)*log10(x)-1.2
```

```
void Bisect();
int count=1,n;
float root=1;
void main()
     {
 clrscr();
printf("\n Solution by BISECTION method \n");
printf("\n Equation is ");
printf("\n\t\t\t x*log(x) - 1.2 = 0\n\n");
printf("Enter the number of iterations:");
scanf("%d",&n);
Bisect();
getch();
            }
void Bisect()
           {
                float x0,x1,x2;
                float f0,f1,f2;
                int i=0;
for(x2=1;;x2++)
{
    f2=F(x2);
    if (f2>0)
{
break;
}
  }
for(x1=x2-1;;x2--)
{
  f1=F(x1);
      if(f1<0)
                 {
                   break;
                              }
                                     }
printf("\t\t----------------------------------------");
printf("\n\t\t ITERATIONS\t\t ROOTS\n");
printf("\t\t----------------------------------------");
for(;count<=n;count++)
{
      x0=((x1+x2)/2.0);
       f0=F(x0);
if(f0==0)
{
       root=x0;
}
if(f0*f1<0)
  {
    x2=x0;
       }
 else
      {
     x1=x0;
     f1=f0;
            }
printf("\n\t\t ITERATION %d", count);
printf("\t :\t %f",x0);
if(fabs((x1-x2)/x1) < EPS)
{
printf("\n\t\t---------------------------------");
printf("\n\t\t Root = %f",x0);
printf("\n\t\t Iterations = %d\n", count);
printf("\t\t-----------------------------------");
getch();
exit(0);
         }
           }
printf("\n\t\t---------------------------------------");
printf("\n\t\t\t Root = %7.4f",x0);
printf("\n\t\t\t Iterations = %d\n", count-1);
printf("\t\t------------------------------------------");
getch(); }
```


2. Algorithm of REGULAR-FALSI METHOD.

Step 01. Start of the program .

Step 02 . Input the variable x0, x1,e, n for the task.

Step 03 . f0=f(x0)

Step 03.   f2=f(x2)

Step 04 .   for i=1 and repeat if i<=n

Step 05 .   x2 = (x0f1-x1f0)/(f1-f0)

Step 06 .   f2 = x2

Step 07 .   if | f2 | <=e

Step 08 . print “convergent “, x2, f2

Step 09 . if sign (f2)!=sign(f0)

Step 10 .   x1=x2   &   f1 = f2

Step 11 .   else

Step 12.   X0 = x2   & f0 = f2

Step 13.   End loop

Step 14.   Print output

Step 15. End the program.

PROGRAM IMPLIMANTATION OF **REGULAR-FALSI METHOD.**

```
#include<stdio.h>
#include<math.h>
#include<conio.h>
#include<string.h>
#include<process.h>
#define EPS 0.00005
#define f(x) 3*x+sin(x)-exp(x)
void FAL_POS();
void main()
{
    clrscr();
```

```
printf("\n Solution by FALSE POSITION method\n");
printf("\n Equation is ");
printf("\n\t\t\t 3*x + sin(x)-exp(x)=0\n\n");
FAL_POS();
               }
void FAL_POS()
      {
            float f0,f1,f2;
            float x0,x1,x2;
            int itr;
             int i;
printf("Enter the number of iteration:");
scanf("%d",&itr);
for(x1=0.0;;)
     {
        f1=f(x1);
if(f1>0)
{
   break;
}
else
{
x1=x1+0.1;
}
    }
x0=x1-0.1;
f0=f(x0);
printf("\n\t\t-----------------------------------------");
printf("\n\t\t ITERATION\t x2\t\t F(x)\n");
printf("\t\t--------------------------------------------");
for(i=0;i<itr;i++)
{
    x2=x0-((x1-x0)/(f1-f0))*f0;
     f2=f(x2);
if(f0*f2>0)
{
     x1=x2;
    f1=f2;
}
else
{
     x0=x2;
    f0=f2;
}
if(fabs(f(2))>EPS)
  {
     printf("\n\t\t%d\t%f\t%f\n",i+1,x2,f2);
}
    }
printf("\t\t--------------------------------------------");
printf("\n\t\t\t\tRoot=%f\n",x2);
printf("\t\t-------------------------------------------");
getch(); }
```


3. Algorithm of NEWTON-REPHSON METHOD.

Step 01 . start of the program.

Step 02 . input the variables x0, n for the task.

Step 03 . input Epsilon & delta

Step 04 . for i= 1 and repeat if i <= n

Step 05 . f0 = f(x0)

Step 06 . dfo = df(x1)

Step 07 . if | dfo | <= delta

a. Print slope too small
b. Print x0, f0, df0, i
c. End of program

Step 08 . x1 = x0 –(f0/df0)

Step 09 . if | (x1-x0/x1) | < epsilon

a. Print convergent
b. Print x1, f(x1), i
c. End of program

Step 10 . x0 = x1
Step 11 . End loop.

PROGRAM IMPLIMANTATION OF **NEWTON REPHSON METHOD.**

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <process.h>
# include <string.h>
# define f(x) 3*x -cos(x)-1
# define df(x) 3+sin(x)
void NEW_RAP();
void main()
 {
   clrscr();
printf ("\n Solution by NEWTON RAPHSON method \n");
printf ("\n Equation is: ");
printf ("\n\t\t\t 3*X - COS X - 1=0 \n\n ");
NEW_RAP();
   getch();
        }
```

```
void NEW_RAP()
   {
     long float x1,x0;
     long float f0,f1;
     long float df0;
        int i=1;
        int itr;
float EPS;
float error;
for(x1=0;;x1 +=0.01)
    {
     f1=f(x1);
if (f1 > 0)
  {
    break;
}
   }
    x0=x1-0.01;
    f0=f(x0);
printf(" Enter the number of iterations: ");
scanf(" %d",&itr);
printf(" Enter the maximum possible error: ");
scanf("%f",&EPS);
   if (fabs(f0) > f1)
      {
printf("\n\t\t The root is near to %.4f\n",x1);
   }
if(f1 > fabs(f(x0)))
  {
printf("\n\t\t The root is near to %.4f\n",x0);
  }
    x0=(x0+x1)/2;
for(;i<=itr;i++)
  {
    f0=f(x0);
   df0=df(x0);
   x1=x0 - (f0/df0);
printf("\n\t\t The %d approximation to the root is:%f",i,x1);
error=fabs(x1-x0);
if(error<EPS)
   {
    break;
}
  x0 = x1;
   }
if(error>EPS)
   {
printf("\n\n\t NOTE:- ");
printf("The number of iterations are not sufficient.");
}
printf("\n\n\n\t\t\t ------------------------------");
printf("\n\t\t\t The root is %.4f ",x1);
printf("\n\t\t\t ------------------------------");
}
```


4. Algorithm for Newton's Forward Formula

Step 01. Start of the program

Step 02. Input number of terms n

Step 03. Input the array ax

Step 04. Input the array ay

Step 05. h=ax[1] – ax[0]

Step 06. for i=0; i<n-1; i++

Step 07. diff[i] [1]=ay[i + 1] – ay[i]

Step 08. End Loop i

Step 09. for j=2; j<=4; j++

Step 10. for i = 0; i <n – j; i++

Step 11. diff[i][j]=diff [i + 1] [j – 1]-diff [i][j – 1]

Step 12. End Loop i

Step 13. End Loop j

Step 14. i=0

Step 15. Repeat Step 16 until ax[i]<x

Step 16. i=i + 1

Step 17. i=i – 1;

Step 18. p=(x – ax [i])/h

Step 19. y1=p*diff[i – 1][1]

Step 20. y2=p*(p+1)*diff [i – 1][2]/2

Step 21. y3=(p+1)*p*(p-1)*diff[i –2][3]/6

Step 22. y4=(p+2)*(p+1)*p*(p – 1)*diff[i – 3][4]/24

Step 23. y=ay[i]+y1+y2+y3+y4

Step 24. Print output x, y

Step 25. End of program.

PROGRAM IMPLIMANTATION OF **NEWTONS FORWORD METHOD OF INTERPOLUTION**

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <process.h>
# include <string.h>
void main()
{
    int n;
    int i,j;
    float ax[10];
    float ay[10];
    float x;
    float y = 0;
    float h;
   float p;
   float diff[20][20];
   float y1,y2,y3,y4;
clrscr();
printf("\n Enter the number of terms - ");
```

```
scanf("%d",&n);
printf("Enter the value in the form of x - ");
for (i=0;i<n;i++)
{
  printf("Enter the value of x%d - ",i+1);
  scanf("%f",&ax[i]);
}
printf("\n Enter the value in the form of y - ");
for (i=0;i<n;i++)
   {
    printf ("Enter the value of y%d - ", i+1);
    scanf ("%f",&ay [i]);
}
printf("\nEnter the value of x for");
printf("\nwhich you want the value of y - ");
scanf("%f",&x);
h=ax[1]-ax[0];
for(i=0;i<n-1;i++)
{
   diff[i][1]=ay[i+1]-ay[i];
      }
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
  {
     diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
       }
    }
     i=0;
do
    {
     i++;
       }
while(ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*diff[i-1][1];
y2=p*(p+1)*diff[i-1][2]/2;
y3=(p+1)*p*(p-1)*diff[i-2][3]/6;
y4=(p+2)*(p+1)*p*(p-1)*diff[i-3][4]/24;
y=ay[i]+y1+y2+y3+y4;

printf("\nwhen x=%6.4f, y=%6.8f ",x,y);
getch();
}
```


5. Algorithm for Newton’s Backward Difference formula

Step 01. Start of the program.

Step 02. Input number of terms n

Step 03. Input the array ax

Step 04. Input the array ay

Step 05. h=ax[1]-ax[0]

Step 06. for i=0; i<n–1; i++

Step 07. diff[i][1]=ay[i+1]–ay[i]

Step 08. End Loop i

Step 09. for j = 2; j < = 4; j + +

Step 10. for i=0; i<n–j; i++

Step 11. diff[i][j]=diff[i+1][j–1]–diff [i][j–1]

Step 12. End Loop i

Step 13. End Loop j

Step 14. i=0

Step 15. Repeat Step 16 until (!ax[i]<x)

Step 16. i=i+1

Step 17. x0=mx[i]

Step 18. sum=0

Step 19. y0=my[i]

Step 20. fun=1

Step 21. p=(x–x0)/h

Step 22. sum=y0
Step 23. for k=1; k<=4; k++

Step 24. fun=(fun*(p–(k–1)))/k

Step 25. sum=sum+fun*diff[i][k]

Step 26. End loop k

Step 27. Print Output x,sum

Step 28. End of Program

PROGRAM IMPLIMANTATION OF NEWOTN’S BACKWORD METHOD OF INTERPOLATION

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
#include<string.h>
void main()
{
int n,i,j,k;
float mx[10],my[10],x,x0=0,y0,sum,h,fun,p,diff[20][20],y1,y2,y3,y4;
clrscr();
printf("\n enter the no. of terms -     ");
scanf("%d",&n);
printf("\n enter the value in the form of x    -  ");
for(i=0;i<n;i++)
{
printf("\n enter the value of x%d-   ",i+1);
scanf("%f",&mx[i]);
}
printf("\n enter the value in the form of y -    ");
for(i=0;i<n;i++)
{
```

```
printf("\n\n enter the value of y%d-    ",i+1);
scanf("%f",&my[i]);
}
printf("\n enter the value of x for");
printf("\nwhich you want the value of of y -");

scanf("%f",&x);h=mx[1]-mx[0];
for(i=0;i<n-1;i++)
{
diff[i][1]=my[i+1]-my[i];
}
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
{
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
}
}
i=0;
while(!mx[i]>x)
{
 i++;
 }
 x0=mx[i];
 sum=0;
 y0=my[i];
 fun=1;
 p=(x-x0)/h;
 sum=y0;
 for(k=1;k<=4;k++)
 {
 fun=(fun*(p-(k-1))/k);
 sum=sum+fun*diff[i][k];}
 printf("\n when x=%6.4f,y=%6.8f",x,sum);
 printf("\n press enter to exit");
 getch(); }
```


6. Algorithm of GAUSS'S FORWORD METHOD OF INTERPOLATION

Step 01. Start of the program.

Step 02. Input number of terms n

Step 03. Input the array ax

Step 04. Input the array ay

Step 05. h=ax[1]-ax[0]

Step 06. for i=0;i<n–1;i++

Step 07. diff[i][1]=ay[i+1]-ay[i]

Step 08. End Loop i

Step 09. for j=2;j<=4;j++

Step 10. for i=0;i<n–j;i++

Step 11. diff[i][j]=diff[i+1][j–1]–diff[i][j–1]

Step 12. End Loop i

Step 13. End Loop j

Step 14. i=0

Step 15. Repeat Step 16 until ax[i]<x

Step 16. i=i+1

Step 17. i=i–1;

Step 18. p=(x–ax[i])/h

Step 19. y1=p*diff[i][1]

Step 20. y2=p*(p–1)*diff[i–1][2]/2

Step 21. y3=(p+1)*p*(p-1)*diff[i–2][3]/6

Step 22. y4=(p+1)*p*(p–1)*(p–2)*diff[i–3][4]/24

Step 23. y=ay[i]+y1+y2+y3+y4

Step 24. Print Output x,y

Step 25. End of Program

PROGRAM IMPLIMANTATION OF **GAUSS’S FORWORD METHOD OF INTERPOLATION**

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <process.h>
# include <string.h>
void main()
{
int n;
int i,j;
float ax[10];
float ay[10];
float x;
float nr,dr;
float y=0; float h;
float p;
float diff[20][20];
float y1,y2,y3,y4;
clrscr();
printf(" Enter the number of terms -   ");
scanf("%d",&n);
printf("\n Enter the value in the form of x -  ");
for (i=0;i<n;i++)
{
printf(" Enter the value of x%d -  ",i+1);
scanf("%f",&ax[i]);
}
printf(" Enter the value in the form of y -   ");
```

```
for(i=0;i<n;i++)
{
printf("Enter the value of y%d  -  ",i+1);
scanf("%f",&ay[i]);
}
printf("\nEnter the value of x for  -  ");
printf("\nwhich you want the value of y  -  ");
scanf ("%f",&x);
h=ax[1]-ax[0];
for(i=0;i<n-1;i++)
{
diff[i][1]=ay[i+1]-ay[i];
}
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
{
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
}
}
i=0;
do {
i++;
}
while(ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*diff[i][1];
y2=p*(p-1)*diff[i-1][2]/2;
y3=(p+1)*p*(p-1)*diff[i-2][3]/6;
y4=(p+1)*p*(p-1)*(p-2)*diff[i-3][4]/24;
y=ay[i]+y1+y2+y3+y4;
printf("\nwhen x=%6.4f,y=%6.8f ",x,y);
getch();
}
```


7. Algorithm of Gauss's Backward Formula

Step 01. Start of the program.

Step 02. Input number of terms n

Step 03. Input the array ax

Step 04. Input the array ay

Step 05. h=ax[1]-ax[0]

Step 06. for i=0;i<n-l;i++

Step 07. diff[i][1]=ay[i+1]-ay[i]

Step 08. End Loop i

Step 09. for j=2;j<=4;j++

Step 10. for i=0;i<n–j;i++

Step 11. diff[i][j]=diff[i+1][j–1]–diff[i][j–1]

Step 12. End Loop i

Step 13. End Loop j

Step 14. i=0

Step 15. Repeat Step 16 until ax[i]<x

Step 16. i=i+1

Step 17. i=i–1;

Step 18. p=(x–ax[i])/h

Step 19. y1=p*diff[i-1][1]

Step 20. y2=p*(p+1)*diff[i–1][2]/2

Step 21. y3=(p+1)*p*(p-1)*diff[i–2][3]/6

Step 22. y4=(p+2)*(p+1)*p*(p–1)*diff[i–3][4]/24
Step 23. y=ay[i]+y1+y2+y3+y4

Step 24. Print Output x,y

Step 25. End of Program

PROGRAM TO IMPLIMENT **GAUSS'S BACKWORD METHOD OF INTERPOLATION.**

```
# include <stdio.h>
# include <conio.h>
# include <math.h>
# include <process.h>
# include <string.h>
void main()
{
int n;
int i,j; float ax[10];
 float ay[10];
float x;
float y=0;
float h;
float p;
float diff[20][20];
float y1,y2,y3,y4;
clrscr();
printf("\n Enter the number of terms -    ");
scanf("%d",&n);
printf("\n Enter the value in the form of x   -   ");
for (i=0;i<n;i++)
{
printf("\n\n Enter the value of x%d  -  ",i+1);
scanf("%f",&ax[i]);
}
printf("\n\n Enter the value in the form of y  -  ");
for(i=0;i<n;i++)
{
printf("\n Enter the value of y%d   -   ",i+1);

scanf("%f",&ay[i]);
```

```
}
printf("\nEnter the value of x for  -   ");
printf("\nwhich you want the value of y   -    ");
scanf("%f",&x);
h=ax[1]-ax[0];
for(i=0;i<n-1;i++)
{
diff[i][1]=ay[i+1]-ay[i];
}
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
{
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
}
}
i=0;
do {
i++;
}
while (ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*diff[i-1][1];
y2=p*(p+1)*diff[i-1][2]/2;
y3=(p+1)*p*(p-1)*diff[i-2][3]/6;
y4=(p+2)*(p+1)*p*(p-1)*diff[i-3][4]/24;
y=ay[i]+y1+y2+y3+y4;
printf("\nwhen x=%6.1f,y=%6.4f ",x,y);

getch();
}
```


8. Algorithm of LAGRANGE'S INTERPOLATION FORMULA.

Step 01. Start of the program

Step 02. Input number of terms n

Step 03. Input the array ax

Step 04. Input the array ay

Step 05. for i=0; i<n; i++

Step 06. nr=1

Step 07. dr=1

Step 08. for j=0; j<n; j++

Step 09. if j !=i

a. nr=nr*(x-ax[j])

b.dr*(ax[i]-ax[j])

Step 10. End Loop j

Step 11. y+=(nr/dr)∗ay[i]

Step 12. End Loop i

Step 13. Print Output x, y

Step 14. End of Program

PROGRAM IMPLIMANTATION OF **LAGRANGE'S INTERPOLATION FORMULA.**

```
#include<stdio.h>
#include<conio.h>
#define MAX 10
void main()
{
float x[MAX],y[MAX],k=0,z,nr,dr;
int i,j,m;
//clrscr();
printf("\n enter the range ");
scanf("%d",&m);
printf("\n enter the x value ");
 for(i=0;i<m;i++)
scanf("%f",&x[i]);
printf("\n enter the y value ");
for(i=0;i<m;i++)
scanf("%f",&y[i]);
printf("\n enter  value OF Z to be calculated ");
scanf("%f",&z);

for(i=0;i<m;i++)
{  nr=1;dr=1;
for(j=0;j<m;j++)
{
if (j!=i)
{
nr=nr*(z-x[j]);
dr=dr*(x[i]-x[j]);
}
}
k=k+((nr/dr)*y[i]);
}
printf("\n final result=%f\n",k);
getch();
```

}

To implement Numerical Differentiations

9 ALGORITHM OF EULER’S METHOD

1. Function F(x,y)=(x–y)/(x+y)
2. Input x0,y0,h,xn
3. n=((xn–x0)/h)+1
4. For i=1,n
5. y=y0+h*F(x0,y0)
6. x=x+h
7. Print x0,y0
8. If x<xn then

x0=x

y0=y

ELSE

9. Next i
10. Stop

PROGRAM IMPLIMANTATION OF **EULER’S METHOD**

```
#include<stdio.h>
#include<conio.h>
#define F(x,y) (x-y)/(x+y)
main ( )
{
int i,n;
float x0,y0,h,xn,x,y;
clrscr();
printf("\n   Enter the values: x0,y0,h,xn: ");
scanf ("%f%f%f%f",&x0,&y0,&h,&xn);
n=(xn-x0)/h+1;
for (i=1;i<=n;i++)
{
y=y0+h*F(x0,y0);
x=x0+h;
printf("\n      X=%f Y=%f",x0,y0);
if(x<xn)
{
x0=x;
y0=y;
}
}
getch();}
```

10 ALGORITHM OF MODIFIED EULER'S METHOD.

1. Function F(x)=(x–y)/(x+y)

2. Input x(1),y(1),h,xn

3. yp=y(1)+h*F(x(1),y(1))

4. itr=(xn–x(1))/h

5. Print x(1),y(1)

6. For i=1,itr

7. x(i+1)=x(i)+h

8. For n=1,50

9. yc(n+1)=y(i)+(h/2*(F(x(i),y(i))+F(x(i+1),yp))

10. Print n,yc(n+1)

11. p=yc (n+1)-yp

12. If abs(p)<.0001 then

goto Step 14

ELSE

yp=yc(n+1)

13. Next n

14. y(i+1)=yc(n+1)

15. print x(i+1),yp

16. Next i

17. Stop.

PROGRAM IMPLIMANTATION OF **MODIFIED EULER'S METHOD**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define F(x,y) (x-y)/(x+y)
main ()
{
int i,n,itr ;
float x[5],y[50],yc[50],h,yp,p,xn;
clrscr();
printf("\n Enter the values: x[1],y[1],h,xn:-  ");
scanf("%f%f%f%f",&x[1],&y[1],&h,&xn);
yp=y[1]+h*F(x[1],y[1]);
itr=(xn-x[1])/h;
printf("\n\n X=%2f Y=%f\n",x[1],y[1]);
for (i=1;i<=itr;i++)
{
x[i+1]=x[i]+h;
for (n=1;n<=50;n++)
{
yc[n+1]=y[i]+(h/2.0)*(F(x[i],y[i])+F(x[i+1],yp));
printf("\nN=%2d Y=%f",n,yc[n+1]);
p=yc[n+1]-yp;
if(fabs (p)<0.0001)
goto next;
else
yp=yc[n+1];
}
next:
y[i+1]=yc[n+1];
printf("\n\n X=%2f Y=%f\n",x[i+1], yp);
}
getch();
}
```

11 Algorithm of Stirling’s Formula

Step 01. Start of the program.

Step 02. Input number of terms n

Step 03. Input the array ax

Step 04. Input the array ay

Step 05. h = ax[1]-ax[0]

Step 06. for i = 1;i < n-1; i++

Step 07. diff [i][1] = ay[i + 1]-ay[i]

Step 08. End loop i

Step 09. for j = 2; j < = 4; j++

Step 10. for i = 0; i < n-j; i++

Step 11. diff[i][j] = diff[i + 1][j-1]-diff[i][j-1]

Step 12. End loop i

Step 13. End loop j

Step 14. i = 0

Step 15. Repeat step 16 until ax[i] < x

Step 16. i = i + 1

Step 17. i = i-1;

Step 18. p = (x-ax[i])/h

Step 19. y1= p*(diff[i][1] + diff[i-1][1])/2

Step 20. y2 = p*p*diff[i-1][2]/2

Step 21. y3 = p*(p*p-1)*(diff[i-1][3]+diff[i-2][3])/6

Step 22. y4 = p*p*(p*p-1)*diff[i-2][4]/24
Step 23. y = ay[i]+y1 + y2 + y3 + y4

Step 24. Print output

Step 25. End of program

PROGRAM TO IMPLEMENT **STIRLING’S METHOD.**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
void main()
{
int n;
int i,j;
float ax[10];
float ax[10];
float h;
float p;
float diff[20][20];
float x,y;
float y1,y2,y3,y4;
clrscr();
 printf("\n Enter the value of terms");
scanf("%d",%n);
printf(”\n Enter the values for x \n”);
for(i=0;i<n;i++)
{
printf("\n Enter the value for x%d-",i+1);
scanf("%f”,&ax[i]);
}
printf("\n Enter the values for y \n");
for(i=0;i<n;i++)
{
printf("\n Enter the value for y%d-",i+1);
scanf("%f",&ay[i]);
}
printf("\n Enter the value of x for");
printf("\n which you want the value of y");
scanf("%f",&x);
h=ax[1]-ax[0];
for(i=0;i<n-1;i++)
{
diff[i][1]=ay[i+1]-ay[i];
}
for(j=2;j<=4;j++)
{
for(i=0;i<n-j;i++)
{
diff[i][j]=diff[i+1][j-1]-diff[i][j-1];
}
}
i=0;
```

```
do {
i++;
}
while(ax[i]<x);
i--;
p=(x-ax[i])/h;
y1=p*(diff[i][1]+diff[i-1][1])/2;
y2=p*p*diff[i-1][2]/2;
y3=p*(p*p-1)*(diff[i-1][3]+diff[i-2][3])/6;
y4=p*p*(p*p-1)*diff[i-2][4]/24;
y=ay[i]+y1+y2+y3+y4;
printf("\n\n When x=%6.2f, y=%6.8f",x,y);
getch(); }
```


12 Algorithm of Runge-Kutta Method.

Steps

1. Function F(x)=(x-y)/(x+y)

2. Input x0,y0,h,xn

3. n=(xn-x0)/h

4. x=x0

5. y=y0

6. For i=0, n

7. k1=h*F(x,y)

8. k2=h*F(x+h/2,y+k1/2)

9. k3=h*F(x+h/2,y+k2/2)

10. k4=h*F(x+h,y+k3)

11. k=(k1+(k2+k3)2+k4)/6

12. Print x,y

13. x=x+h

14. y=y+k

15. Next i

16. Stop

PROGRAM IMPLIMANTATION OF **RUNGA KUTTA METHOD.**

```
#include<stdio.h>
#include<conio.h>
#define F(x,y) (x-y)/(x+y)
main()
{
int i,n;
float x0,y0,h,xn,k1,k2,k3,k4,x,y,k;
clrscr();
printf("\n\n\t Enter the values: x0,y0,h,xn:- ");
scanf("%f%f%f%f", &x0,&y0,&h,&xn);
n=(xn-x0)/h;
x=x0;
y=y0;
for(i=0;i<=n;i++)
{
k1=h*F(x,y);
k2=h*F(x+h/2.0,y+k1/2.0);
k3=h*F(x+h/2.0,y+k2/2.0);
k4=h*F(x+h,y+k3);
k=(k1+(k2+k3)*2.0+k4)/6.0;
printf("\n\t X=%f Y=%f", x, y);
x=x+h;
y=y+k;
}getch();}
```

13. ALGORITHM OF TRAPEZOIDAL RULE

Step 01. Start of the program.

Step 02. Input Lower limit a

Step 03. Input Upper Limit b

Step 04. Input number of sub intervals n

Step 05. h=(b-a)/n

Step 06. sum=0

Step 07. sum=fun(a)+fun(b)

Step 08. for i=1; i<n; i++

Step 09. sum +=2*fun(a+i)

Step 10. End Loop i

Step 11. result =sum*h/2;

Step 12. Print Output result

Step 13. End of Program

Step 14. Start of Section fun

Step 15. temp = 1/(1+(x*x))

Step 16. Return temp

Step 17. End of Section fun.

PROGRAM TO IMPLEMENT **TRAPEZOIDAL METHOD.**

```
 # include <stdio.h>
# include <conio.h>
# include <math.h>
# include <process.h>
# include <string.h>
float fun(float);
void main()
{
float result=1;
float a,b;
float h,sum;
int i,j;
int n;
clrscr();
printf("\n\n Enter the range - ");
printf("\n\n Lower Limit a - ");
scanf("%f" ,&a);
printf("\n\n Upper Limit b - ");
scanf("%f" ,&b);
printf("\n\n Enter number of subintervals - ");
scanf("%d" ,&n);
h=(b-a)/n;
sum=0;
sum=fun(a)+fun(b);
for(i=1;i<n;i++)
{
sum+=2*fun(a+i);
}
result=sum*h/2;
printf("n\n\n\n Value of the integral is %6.4f\t",result);
printf("\n\n\n Press Enter to Exit");
getch();
}
float fun(float x)
{
float temp;
temp = 1/(1+(x*x));
return temp;
}
```

14 ALGORITHM OF SIMPSON'S 1/3rd RULE

Step 01. Start of the program.

Step 02. Input Lower limit a

Step 03. Input Upper limit b

Step 04. Input number of subintervals n

Step 05. h=(b–a)/n

Step 06. sum=0

Step 07. sum=fun(a)+4*fun(a+h)+fun(b)

Step 08. for i=3; i<n; i + = 2

Step 09. sum + = 2*fun(a+(i – 1)*h) + 4*fun(a+i*h)

Step 10. End of loop i

Step 11. result=sum*h/3

Step 12. Print Output result

Step 13. End of Program

Step 14. Start of Section fun

Step 15. temp = 1/(1+(x*x))

Step 16. Return temp

Step 17. End of Section fun

PROGRAM TO IMPLEMENT **SIMPSON'S 1/3rd METHOD OF NUMERICAL INTEGRATION**

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<process.h>
#include<string.h>
float fun(float);
void main()
{
float result=1;
float a,b;
float sum,h;
int i,j,n;
clrscr();
printf("\n Enter the range - ");
printf("\n Lower Limit a - ");
scanf("%f",&a)
;printf("\n Upper limit b - ");
scanf("%f",&b);
printf("\n\n Enter number of subintervals - ");
scanf("%d",&n);
h=(b-a)/n;
sum=0;
sum=fun(a)+4*fun(a+h)+fun(b);
for(i=3;i<n;i+=2)
{
sum+=2*fun(a+(i-1)*h)+4*fun(a+i*h);
}
result=sum*h/3;
printf("\n\nValue of integral is %6.4f\t",result);
getch();}
float fun(float x)
{
float temp;
temp=1/(1+(x*x));
return temp;
}
```


15. ALGORITHM OF SIMPSON'S 3/8th RULE

Step 01. Start of the program.

Step 02. Input Lower limit a

Step 03. Input Upper limit b

Step 04. Input number of sub itervals n

Step 05. h = (b – a)/n

Step 06. sum = 0

Step 07. sum = fun(a) + fun (b)

Step 08. for i = 1; i < n; i++

Step 09. if i%3=0:

Step 10. sum + = 2*fun(a + i*h)

Step 11. else:

Step 12. sum + = 3*fun(a+(i)*h)

Step 13. End of loop i

Step 14. result = sum*3*h/8

Step 15. Print Output result

Step 16. End of Program

Step 17. Start of Section fun

Step 18. temp = 1/(1+(x*x))

Step 19. Return temp

Step 20. End of section fun

PROGRAM TO IMPLEMENT **SIMPSON'S 3/8th METHOD OF NUMERICAL INTEGRATION**

```
#include<stdio.h>
```

```
#include<conio.h>
float fun(int);
void main()
{
int n,a,b,i;
float h, sum=0, result;
//clrscr();

  printf("enter range");
  scanf("%d",&n);
  printf("enter lower limit");
  scanf("%d",&a);
  printf("enter upper limit");
  scanf("%d",&b);
  h=(b-a)/n;
  sum=fun(a)+fun(b);
   for(i=0;i<n;i++)
   {
   if (i%2==0)
    sum+=2*fun(a+i*h);
   else
     sum+=3*fun(a+i*h);
   }
   result=sum*3/8*h;
  printf("%f", result);
  getch();
}

float fun(int x)
{
float val;
val=1/(1+(x*x));
return(val);
}
```


16. Draw frequency chart like histogram

```
#include<stdio.h>
#include<conio.h>

#define    N    5
```

```
main()
{
   intvalue[N];
   int i, j, n, x;

   for (n=0; n < N; ++n)
   {
     printf("Enter employees in Group - %d : ",n+1);
     scanf("%d", &x);
     value[n] = x;
     printf("%d\n", value[n]);
   }
   printf("\n");

   printf("          |\n");
   for (n = 0 ; n < N ; ++n)
   {
     for (i = 1 ; i <= 3 ; i++)
     {
        if ( i == 2)
          printf("Group-%1d   |",n+1);
        else
          printf("          |");




        for (j = 1 ; j <= value[n]; ++j)
          printf("*");
        if (i == 2)
          printf("(%d)\n", value[n]);
        else
          printf("\n");
     }
     printf("          |\n");
   }
}
```


Output

```
Enter employees in Group - 1 : 12
12
Enter employees in Group - 2 : 23
23
Enter employees in Group - 3 : 35
35
Enter employees in Group - 4 : 20
```

```
20
Enter Employees in Group - 5 : 11
11

          |
          |************
Group-1   |************(12)
          |************
          |
          |***********************
Group-2   |***********************(23)
          |***********************
          |
          |***********************************
Group-3   |***********************************(35)
          |***********************************
          |
          |********************
Group-4   |********************(20)
          |********************
          |
          |***********
Group-5   |***********(11)
          |***********
            |
```


17. Draw frequency chart Pie-chart

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
main()
{
  int gd, gm, x, y;
  gd=DETECT;
  initgraph(&gd, &gm, "C:\\TC\\BGI");
  settextstyle(SANS_SERIF_FONT,HORIZ_DIR,2);
```

```
    setcolor(WHITE);
    outtextxy(275,10,"Pie CHART");
    x = getmaxx()/2;
    y = getmaxy()/2;
    setfillstyle(LINE_FILL,CYAN);
    pieslice(x, y, 0, 75, 100);
    outtextxy(x+100, y - 75, "25");
    setfillstyle(HATCH_FILL,GREEN);
    pieslice(x, y, 75, 225, 100);
    outtextxy(x-175, y - 75, "40");
    setfillstyle(INTERLEAVE_FILL,WHITE);
    pieslice(x, y, 225, 360, 100);
    outtextxy(x+75, y + 75, "35");
    getch();
    return 0;
}
```

Output:-