**KARPAGAM ACADEMY OF HIGHER EDUCATION**
**(Deemed University Established Under Section 3 of UGC Act 1956)**
**Pollachi Main Road, Eachanari (Po),**
**Coimbatore –641 021**
**SYLLABUS(Practical)**

**Semester – V**

| | | L | T | P | C |
|---|---|---|---|---|---|
| **15MMU511** | **NUMERICAL METHODS -PRACTICAL** | **0** | **0** | **5** | **3** |

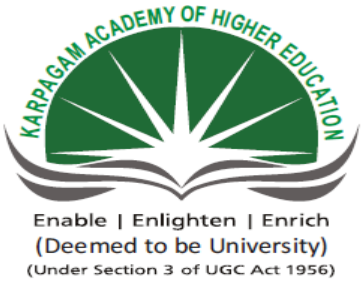
1. Roots of a given Algebraic Equation using Bisection method.
2. Roots of a given Algebraic Equation using Newton Raphson method.
3. Solution of linear algebraic equations using Gauss Elimination method.
4. Solution of linear algebraic equations using Gauss Jordan method.
5. Solution of linear algebraic equations using Gauss Seidal method.
6. Interpolation using Newton‟s forward .
7. Interpolation using Newton‟s backward .
8. Interpolation using Lagrange‟s method.
9. Derivative using Newton‟s forward difference formula.
10. Derivative using Newton‟s backward difference formula.
11. The value of the given integral using Simpson‟s one third rule.
12. The value of the given integral using Trapezoidal rule.

# KARPAGAM ACADEMY OF HIGHER EDUCATION

***(Deemed to be University Established Under Section 3 of UGC Act 1956)***

POLLACHI MAIN ROAD,EACHANARRI(PO),

COIMBATORE – 641021.




## DEPARTMENT OF MATHEMATICS

# NUMERICAL METHODS-PRACTICAL

## (15MMU511)

# INDEX


| NO. | CONTENT |
|---|---|
| 1. | BISECTION METHOD |
| 2. | NEWTON RAPHSON METHOD |
| 3. | GAUSS ELIMINATION METHOD |
| 4. | GAUSS JORDAN METHOD |
| 5. | GAUSS SEIDAL METHOD |
| 6. | INTERPOLATION USING NEWTON'S FORWARD METHOD |
| 7. | INTERPOLATION USING NEWTON'S BACKWARD METHOD |
| 8. | INERPOLATION USING LAGRANGE'S METHOD |
| 9. | DERIVATIVE USING NEWTON'S FORWARD DIFFERENCE FORMULA |
| 10. | DERIVATIVE USING NEWTON'S BACKWARD DIFFERENCE FORMULA |
| 11. | SIMPSON'S ONE THIRD RULE |
| 12. | TRAPEZOIDAL RULE |

| EX.NO:1 | BISECTION METHOD |
|---|---|

## AIM:

To calculate the roots of given algebraic equation by bisection method using c program.

## ALGORITHM:

STEP 1: Start the process

STEP 2: Define the given equation as in header file.

STEP 3: Declare the variables as interger and float data type.

STEP 4: By using scanf statement get the values of c[i].

STEP 5: Print the result

STEP 6: Stop the process.

## PROGRAM CODING:

```
#include<stdio.h>
#include<coni
#include<math.h>
# define f(x) (c[0] *pow(x,3)+c[i]*pow(x,2)+c[2]*x+c[3])
void main ()
{
int i,j;
double a,b,m, f[5],c[9],r;
clrscr();
for(j=0;j<2;j++)
{
if(j==0)
printf("Enter the coefficient of positive root:\n");
else
printf("Enter the coefficient of negative root :\n");
for(i=0 ; i<4; i++)
scanf("%lf",c[i]);
a=0;
f[1]=f[a];
read:
```

```
if(j==0)
b=a+1;
else
b=a-1;
f[2]=f[b];
if(f[1]*f[2]>0)
{
f[1]=f[2];
a=b;
goto read;
}
else
for (i=1; i<21; i++)
{
m=((a+b)/2);
f[3]=f(m);
if(f(1)*f(3)<0)
b=m;
else
a=m;
f[1]=f[a];
}
if(fabs(a-b)<pow(10,-6))
```

```
{
r=a+b/2;
goto out;
}
out:
printf("required root of f(x)=0 is : %f \n",r);
}
getch();
}
```

## OUTPUT:

Enter the coefficient for positive root:

2

4

6

8

Required root of f(x)=p is : 1.499999

Enter the coefficient for negative root:

2

4

6

8

Required root of f(x)=p is : -1.499999

## RESULT:

The above program is executed successfully by bisection method and the output is verified.

| EX.NO : 2 | NEWTON RAPHSON METHOD |
|---|---|

## AIM:

To calculate the positive root of the given algebraic equation by Newton Raphson method by using c program.

## ALGORITHM:

STEP 1: Start the process

STEP 2: Define the given equation as f(x) in the header file.

STEP 3: Define the derivative function as g(x) in the header file.

STEP 4: Declare the variables as float and integer data type.

STEP 5: By using the formula

X[i+1] = X[i] – ( f[i] / g[i] ) , and the program is executed.

STEP 6: Print the result.

STEP 7: Stop the process.

## PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x) (pow(x,x)-1000)
#define g(x) (pow(x,x)*(1+log10(x)))
 void main()
  {
   int i;
   float a,b,f[10],g[10],x[10];
   clrscr();
   a=0.1;
   read:
   b=a+1;
   if(f(a)*f(b)>0)
   {
   a=b;
   goto read;
   }
   else
   x[0]=a;
```

```
for(i=0; i<10; i++)
{
f[i]=f(x[i]);
g[i]=g(x[i]);
x[i+1]=x[i]-(f[i]/g[i]);
if(fabs  (x[i+1] - x[i] < pow(10,-6)))
goto write;
}
write:
printf("The positive root of x^x-1000=0 is: %f \n",x[i]);
getch();
}
```

## OUTPUT:

The Positive root of x^x-1000 is : 5.385459.

## RESULT:

The above program is executed successfully by newton raphson method and the output is verified .

| EX.NO:3 | GAUSS ELIMINATION METHOD |
|---|---|

## AIM:

To solve the system of linear algebraic equation by Gauss Elimination method by using C program.

## ALGORITHM:

STEP 1: Start the process.

STEP 2: Declare the variable in as float data type.

STEP 3: Using scanf statement obtain the value of n.

STEP 4: Read the RHS constant.

STEP 5: Read the coefficient row wise.

STEP 6: Calculate the value of unknown variable using

x[i]=c[i]/a[i][i];

STEP 7: Print the result.

STEP 8: Stop the process.

# PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int i,j,k,n;
float a[10][10],x[10],c[10];
clrscr();
printf("Enter the value of n:\n");
scanf("%d", &n);
printf("Enter the right hand side constants:\n");
for(i=0; i<n ; i++)
scanf("%f", &c[i]);
printf("Enter the coefficients in row wise:\n");
for(i=0;i<n; i++)
{
for(j=0;j<n;j++)
scanf("%f", &a[i][j]);
}
for(k=0; k<n-1; k++)
{
```

```
for(i=k+1; i<n; i++)
{
for(j=k+1; j<n; j++)
a[i][j]=a[i][j]-(a[i][k]/a[k][k]*a[k][j]);
c[i]=c[i]-(a[i][k]/a[k][k]*c[k]);
}
}
x[n-1]=c[n-1]/a[n-1][n-1];
printf (“\n The solution is:\n”);
printf (“\n x[%d]= %f \n”,n-1,x[n-1]);
for(k=0; k<n-1; k++)
{
i=n-k-2;
for(j=i+1; j<n; j++)
c[i]=c[i]-a[i][j]*x[j];
x[i]=c[i]/a[i][i];
printf(“\n x[%d]= %f \n”,i,x[i]);
}
getch();
}
```

## OUTPUT:

Enter the value of n:

3

Enter the right hand side constants:

3 10 13

Enter the coefficients in row wise:

1 2 1

2 3 3

3 -1 2

The solution is:

x[2] = 3.000

x[1] = -1.000

x[0] = 2.000

## RESULT:

The above program has been executed successfully by Gauss Elimination method and the output is verified.

| EX.NO:4 | GAUSS JORDAN METHOD |
|---|---|

## AIM:

To solve the system of linear algebraic equation by Gauss Jordan method using C program.

## ALGORITHM:

STEP1: Start the process.

STEP2:Declare the variable i,j,k,n as integer data type and x[10],a[10][10] as float data type.

STEP3: Get the 'n' value using scanf statement.

STEP4: Get the right hand side constant in array in for loop using scanf statement.

STEP5: Get the coefficient row wise value using for loop (i=0; i<n; i++)(j=0; j<n; j++) and in array a[i][j].

STEP6: Calculate a[i][j]=a [i][j]-(a[i][k]/a[k][k]*a[k][j]);

STEP7: Get the result using for loop and by using

x[i] =a[i][n]/a[i][i]

and print the result.

STEP8: Stop the process

## PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
inti,j,k,n;
float x[10],a[10][10];
clrscr();
printf("\n Enter the values for n : \n");
scanf("%d",&n);
printf("\n Enter the right hand side constants: \n");
for(i=0; i<n; i++)
scanf("%f", &a[i][n]);
printf("\n Enter the coefficients in row wise: \n");
for(i=0; i<n; i++)
{
for(j=0;j<n;j++)
scanf("%f",&a[i][j]);
printf("\n");
}
for(k=0; k<n; k++)
```

```
{
for(i=0; i<n; i++)
{
if(i!=k)
{
for(j=k+1; j<n+1; j++)
a[i][j]=a[i][j]-(a[i][k]/a[k][k]*a[k][j]);
}
}
}
 printf("\n The solution is : \n");
for(i=0; i<n; i++)
{
x[i]=(a[i][n]/a[i][i]);
printf("x[%d]= %f \n",i,x[i]);
}
getch();
}
```

## OUTPUT:

Enter the values for n:

3

Enter the right hand side constants:

4 9 2

Enter the coefficients in row wise:

1 1 2

2 -1 3

3 -1 -1

The solution is:

x[0] = 1.000000

x[1] = -1.000000

x[2] = 2.000000

## RESULT:

The above program has been executed successfully by Gauss Jordan method and the output is verified.

| EX.NO:5 | GAUSS SEIDAL METHOD |
|---|---|

## AIM:

To solve the system of linear algebraic equation by Gauss Seidal method using C program.

## ALGORITHM:

STEP1: Start the process.

STEP2:Declare the variable i,j,m,n,l as integer data type and

x[10],a[10][10] ,b[10],c as float data type.

STEP3: Get the ‘n’ value and ‘l’ value using scanf statement.

STEP4: Get the right hand side constant in array in the for loop using scanf

statement.

STEP5: Get the coefficients in row wise value using for loop

(i=0;i<n;i++),(j=0;j<n;j++)and in array a[i][j].

STEP6: Calculate c=c-(a[i][j]*x[j]);

STEP7: Get the result using for loop and by using

x[i] =c/a[i][j] * x[j] and print the result.

STEP8: Stop the process.

## PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int i,j,m,n,l;
float x[10],a[10][10],b[10],c;
clrscr();
printf("\n Enter the value of n:\n");
scanf("%d",&n);
printf("\n Enter the no.of iterations:\n");
scanf("%d",&l);
printf("Enter the right hand side constants:\n");
for(i=0; i<n; i++)
scanf("%f", &b[i]);
printf("Enter the coefficients in row wise:\n");
for(i=0; i<n; i++)
{
x[i]=0;
for(j=0; j<n; j++)
scanf("%f", &a[i][j]);
}
m=1;
line:
for(i=0; i<n; i++)
{
c=b[i];
for(j=0; j<n; j++)
{
if(i!=j)
c=c-(a[i][j]*x[j]);
}
x[i]=c/a[i][i];
```

```
}
m=m+1;
if(m<=l)
goto line;
else
{
printf("The solution is:\n");
for(i=0; i<n; i++)
printf("x[%d]=%f\n",i,x[i]);
}
getch();
}
```

## OUTPUT:

Enter the values of n:

3

Enter the no. of iterations:

10

Enter the right hand side constants:

-6 -7 -8

Enter the coefficients in row wise:

10 -2 -2

-1 10 -1

-1 -1 10

The solution is:

x[0] = -0.976744

x[1] = -0.896406

x[2] = -0.987315

## RESULT:

The above program has been executed successfully by Gauss Seidal method and the output is verified.

| EX.NO:6 | INTERPOLATION USING NEWTON FORWARD FORMULA |
|---|---|

## AIM:

To calculate the values of x and y by Newton's Forward Interpolation method using c program.

## ALGORITHM:

STEP 1: Strat the process.

STEP 2: Declare the variables as i,n,j,k,l as integer data type and

x0,y[20],f[10][10],x[10] ,z[10],h,u,p as float data type.

STEP 3: Get the 'n' values using scanf statement.

STEP 4: Get the values for x for which values of y are required.

STEP 5: By using the formula

z[k] = z[k] + p*f[i][0] and initialize the value for p=1,

the program is executed.

STEP 6: Print the result

STEP 7: Stop the process.

## PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
Void main ( )
{
int  i,j,k,l,n;
float  x0,y[20], f[10][10] , x[10] , z[10] , h, u ,p ;
clrscr ( );
printf ( " Enter the values of n ( no. of pairs -1) :\n");
scanf(" %d ",  &n);
printf (" Enter the initial values of x : \n" );
scanf(" %f ", &x0 );
printf (" Enter the step size h: \n" );
scanf(" %f ",&h );
printf(" Enter the values of y \n" );
for(i=0; i< n+1 ; i++ )
scanf(" %f ", &y[i] );
printf( "Enter the required no. of  interpolated  values of y :\n" );
scanf(" %d ", &l);
printf(" Enter the %d values of x for which values of y are
        required :\n" ,l);
for(k=0; k<l ; k++)
scanf ("%f ", &x[k]);
```

```
for(j=0; j<n+1 ; j++)
f[0][j]=y[j];
for(i=1; i<n+1; i++)
for(j=0; j<n-1; j++)
f[i][j]=f[i-1][j+1] – f[i-1][j];
for(k=0 ; k<l ;k++)
{
u=[x[k] –x0 ] /h;
z[k] =y[0];
p=1;
for (i=1; i<n+1;i++)
{
p = p* (u – i+1)/ i;
z[k] = z[k] +p*f[i][0];
}
printf(“ The values of x and y are : f\t %f \”n,x[k],z[k]);
}
getch( );
}
```

# OUTPUT:

Enter the value of n(No of data pairs - 1) :

7

Enter the initial value of x:

0

Enter the step size h:

1

Enter the values of y:

1 2 4 7 11 16 22 29

Enter the required no.of interpolated values of y:

6

Enter the 6 values of X for which values of y are required:

-0.4 0.8 3.5 5 6.3 7.7

The values of X and Y are :

-0.400000 0.880000

The values of X and Y are :

0.800000 1.720000

The values of X and Y are :

3.500000 8.875000

The values of X and Y are :

5.000000 16.000000

The values of X and Y are :

6.300000 23.995003

The values of X and Y are :

7.700000 34.494999

## RESULT:

The above program is executed successfully by newton forward interpolation formula and the output is verified.

| EX.NO:7 | INTERPOLATION USING NEWTON BACKWARD FORMULA |
|---|---|

## AIM:

To calculate the values of x and y by newton backward interpolation method using c program**.**

## **ALGORITHM**:

STEP 1: Strat the process.

STEP 2: Declare the variables as i,n,j,k,l as integer data type and

x0,y[20],f[10][10],x[10],z[10],h,u,p as float data type.

STEP 3: Get the values of 'n' using scanf statement

STEP 4: Get the required values of y

STEP 5: By using the formula

y[k] = y[k] + p*f[i][h] and initialize the value for p=1,

the program is executed.

STEP 6: Print the result

STEP 7: Stop the process**.**

## PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
void main( )
{
inti,j,k,n,l;
 float xn,y[20],f[10][10],X[10],Y[10],h,u,p;
clrscr();
printf("Enter the value of n(No of data pairs - 1) :\n");
scanf("%d",&n);
printf("Enter the last value of x:\n");
scanf("%f",&xn);
printf("Enter the step size h:\n");
scanf("%f",&h);
printf("Enter the values of y:\n");
for(i=0;i<n+1;i++)
scanf("%f",&y[i]);
printf("Enter the required no.of interpolated values of y:\n");
scanf("%d",&l);
printf("Enter the %d values of X for which values of y are required:\n",l);
for(k=0;k<l;k++)
scanf("%f",&X[k]);
```

```
for(j=0;j<n+1;j++)
f[0][j]=y[j];
for(i=1;i<n+1;i++)
for(j=i;j<n+1;j++)
f[i][j]=f[i-1][j]-f[i-1][j-1];
for(k=0;k<l;k++)
 {
  u=(X[k]-xn)/h;
  Y[k]=y[n];
  p=1;
for(i=1;i<n+1;i++)
 {
  p=p*(u+i-1)/i;
  Y[k]=Y[k]+p*f[i][n];
 }
printf("The values of X and Y are : %f\t%f\n",X[k],Y[k]);
 }
getch();
 }
```

## OUTPUT:

Enter the value of n(No of data pairs - 1) :

7

Enter the initial value of x:

7

Enter the step size h:

1

Enter the values of y:

1 2 4 7 11 16 22 29

Enter the required no.of interpolated values of y:

6

Enter the 6 values of X for which values of y are required:

-0.4 0.8 3.5 5 6.3 7.7

The values of X and Y are :

-0.400000 0.879999

The values of X and Y are :

0.800000 1.720000

The values of X and Y are :

3.500000 8.875000

The values of X and Y are :

5.000000 16.000000

The values of X and Y are :

6.300000 23.995003

The values of X and Y are :

7.700000 34.494999

RESULT:

The above program is executed successfully by newton backward interpolation formula and tha output is verified.

| EX.NO:8 | INTERPOLATION USING LAGRANGE'S METHOD |
|---|---|

## AIM:

To calculate the values of x and y by lagrange's interpolation method using c program.

## **ALGORITHM**:

STEP 1: Strat the process.

STEP 2: Declare the variables as i,n,j,k,l as integer data type and

x[20],y[20],a[10] , X[10],Y[10] as float data type .

STEP 3: Get the values of 'n' using scanf statement

STEP 4: Get the values for X for which values of y are required.

STEP 5: By using the formula

y[k] = Y[k] + a[i] * y[i] and the program is executed.

STEP 6: Print the result

STEP 7: Stop the process.

## PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
void main()
{
inti,j,n,k,l;
 float x[20],y[20],a[20],X[20],Y[20];
clrscr();
printf("Enter the value of n(No. of pairs - 1)\n");
scanf("%d", &n);
printf("Enter the values of x and y:\n");
for(i=0;i<n+1;i++)
scanf("%f  %f", &x[i] ,&y[i]);
printf("Enter the required no.of interpolated values of y:\n");
scanf("%d", &l);
printf("Enter the %d values of X for which values of y are required:\n",l);
for(k=0; k< l ;k++)
scanf("%f",& X[k]);
for(k=0; k< l; k++)
 {
for(i=0; i<n+1;i++)
  {
```

```
a[i]=1;
for(j=0;j<n+1;j++)
   {
if (i!=j)
a[i]=a[i]*(X[k]-x[j])/(x[i]-x[j]);
   }
  }
  Y[k]=0;
for(i=0;i<n+1;i++)
  Y[k]=Y[k]+a[i]*y[i];
printf("Enter the values of X and Y are: %f \t \t %f\n" ,X[k],Y[k]);
 }
getch();
}
```

## OUTPUT:

Enter the value of n(No of data pairs - 1) :

6

Enter the values of x and y:

-0.2 2.60

0.0 3.00

0.2 3.40

0.4 4.28

0.6 7.08

0.8 14.20

1.0 29.00

Enter the required no.of interpolated values of y:

5

Enter the 5 values of X for which values of y are required:

-0.1 0.08 0.43 0.84 0.97

The values of X and Y are :

-0.100000 2.792500

The values of X and Y are :

0.800000 3.154624

The values of X and Y are :

0.430000 4.529800

The values of X and Y are :

0.840000 16.421181

The values of X and Y are :

0.970000 26.131424

## RESULT:

The above program is executed successfully by lagrange's method and the output is verified.

| EX.NO:9 | DERIVATIVE USING NEWTON'S FORWARD DIFFERENCE FORMULA |
|---|---|

## AIM:

To calculate the derivative of y at initial point by newton's forward difference formula using c program.

## ALGORITHM:

STEP 1: Start the process

STEP 2: Declare the variables i,j,n as integer data type and

h,y[20],f[20][20],x[20],s,Dy as float data type.

STEP 3: Get the values of 'n' using scanf statement

STEP 4: Initialize the values for

h = x[i] – x[0] and

s = s + pow(-1, i+1) * f[i][0] / i

STEP 5: By using the formula the derivative Dy = s/ h , the program is executed.

STEP 6: Print the result

STEP 7: Stop the process.

## PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
Void  main ( )
{
int  i , j , n;
float  h, y[20] ,f[20][20], x[20], s , Dy ;
clrscr( );
printf("Enter the value of n ( no. of pairs of data values -1) :\n");
scanf(" %d ", &n);
printf("Enter  the values of  x and y :\n');
for(i=0; i< n+1 ; i++)
scanf("%f %f ", &x[i] , &y[i] );
h = x[1] – x[0] ;
printf("\n");
printf(" Derivative of y at initial point %f   is  = " , x[0] );
for(j=0; j< n+1 ; j++)
f[0][j] = y[j] ;
for(i=0; i<n+1 ; i++)
for( j=0; j<n-i+1; j++)
f [i][j] = f [i-1][j+1] – f[i-1][j] ;
```

```
s=0;
for(i=1 ; i<n+1 ; i++)
s = s + pow(-1, i+1) * f[i][0] / i ;
Dy = s / h ;
printf(“ %f \n”, Dy);
getch( );
}
```

## OUTPUT:

Enter the value of n (no. of pairs of data values -1):

4

Enter the values of x and y:

1931 40.62

1941 60.80

1951 79.95

1961 103.56

1971 132.65

Derivative of y at initial point 1931.000000 is

= 2.364251

## RESULT:

The above program is executed successfully by Newton's forward difference formula and the output is verified.

| EX.NO:10 | DERIVATIVE USING NEWTON'S BACKWARD DIFFERENCE FORMULA |
|---|---|

## AIM:

To calculate the derivative of y at final point by newton's backward difference formula using c program.

## ALGORITHM:

STEP 1: Start the process

STEP 2: Declare the variables i,j,n as integer data type and

h,y[20],f[20][20],x[20],s,Dy as float data type.

STEP 3: Get the values of 'n' using scanf statement

STEP 4: Initialize the values for

h = x[i] – x[0] and

s = s + f[i][k] / i

STEP 5: By using the formula the derivative Dy = s/ h , the program is executed.

STEP 6: Print the result

STEP 7: Stop the process.

## PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
Void  main ( )
{
int  i , j , n;
float  h, y[20] ,f[20][20], x[20], s , Dy ;
clrscr( );
printf("Enter the value of n ( no. of pairs of data values -1) :\n");
scanf(" %d ", &n);
printf("Enter  the values of  x and y :\n');
for(i=0; i< n+1 ; i++)
scanf("%f %f ", &x[i] , &y[i] );
h = x[1] – x[0] ;
printf("\n");
printf(" Derivative of y at final point %f   is = " , x[n] );
for(j=0; j< n+1 ; j++)
f[0][j] = y[j] ;
for(i=0; i<n+1 ; i++)
for( j=0; j<n+1; j++)
f [i][j] = f [i-1][j] – f[i-1][j-1] ;
s=0;
```

```
for(i=1 ; i<n+1 ; i++)
s = s+ f [i][n] / i ;
Dy = s / h ;
printf(“ %f  ”, Dy);
getch( );
}
```

## OUTPUT:

Enter the value of n (no. of pairs of data values -1):

6

Enter the values of x and y :

50 3.6840

51 3.7084

52 3.7325

53 3.7563

54 3.7798

55 3.8030

56 3.8259

Derivative of y at final point 56.000000 is

= 0.022750

## RESULT:

The above program is executed successfully by Newton's backward difference formula and the output is verified.

| EX.NO:11 | **SIMPSON'S ONE-THIRD RULE** |
|---|---|

## AIM:

To using evaluate the integral by Simpson's 1/3 rule using C program.

## ALGORITHM:

STEP1: Start the process.

STEP2: Declare the variables i,n as integer data type.

STEP3: Declare the variables a,b,x[30],y[30] in float data type and double as h,i1.

STEP4: Get the upper and lower limits using scanf statement.

STEP5: Make sure that the multiple of sub intervals should not exceed 30.

STEP6: Calculate h value

h=(b-a)/n;

STEP7: Calculate i value by using for loop and initialize i1=0.

STEP8: Get the result by using the formula.

i1=i1+(h/3)*(y[i]+4*y[i+1]+y[i+2]);

STEP9: Stop the process.

# PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x) ( log(x))
void main()
{
int i,n;
float a,b,x[30],y[30];
double h,il;
clrscr();
printf("Enter the lower and upper limits:\n");
scanf("%f %f" ,&a, &b);
printf("Enter the no. of subintervals (multiple of 6 not exceeding 30):\n");
scanf("%d" ,&n);
h=(b-a)/n;
x[0]=a;
for(i=0;i<n+1;i++)
{
y[i]=f(x[i]);
if(i==n)
goto line;
x[i+1]=x[i]+h;
}
line:
il=0;
for(i=0;i<n-1;i++)
il=il+(h/3)*(y[i]+4*y[i+1]+y[i+2]);
printf("By simpsons 1/3 rule the solution is :\t%f",il);
getch();
}
```

# OUTPUT:

Enter the lower and upper limits:

4 5.2

Enter the no.of sub intervals(multiple of 6 notexceeding 30):

6

By Simpsons1/3 Rule the solution is: 1.827647

# RESULT:

The above program has been executed successfully by simpson's one third rule and the output is verified.

| EX.NO:12 | TRAPEZOIDAL RULE |
|---|---|

## AIM:

To evaluate the integral by Trapezoidal rule using C program.

## ALGORITHM:

STEP1: Start the process.

STEP2: Declare the variables in integer data type .

STEP3: Declare the variables a,b,x [30],y[30],n in float data type and double as h, i1.

STEP4: Get the upper and lower limits using scanf statement .

STEP5: Make sure that the multiple of sub intervals should not exceed 30.

STEP6: Calculate h value

h=(b-a)/n;

STEP7: Calculate i value by using for loop and initialize i1=0.

STEP8: Get the result by using the formula

i1=i1+(h/2)*(x[i]+y[i+1]);

STEP9: Stop the process

.

# PROGRAM CODING:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define f(x)(log(x))
void main()
{
int i,n;
float a,b,x[30],y[30];
double h,i1;
clrscr();
printf("Enter the lower and upper limits:\n");
scanf("%f %f", &a, &b);
printf("Enter the no. of subintervals (multiple of 6 not exceeding 30):\n");
scanf("%d", &n);
h=(b-a)/n;
x[0]=a;
for(i=0;i<n+1;i++)
{
y[i]=f(x[i]);
if(i==n)
goto line;
x[i+1]=x[i]+h;
}
line:
i1=0;
for(i=0;i<n;i++)
i1=i1+(h/2)*(y[i]+y[i+1]);
printf("By Trapezoidal Rule the solution is:%f",i1);
getch();
}
```

## OUTPUT:

Enter the lower and upper limits:

4 5.2

Enter the no.of sub intervals(multiple of 6 notexceeding 30):

6

By Trapezoidal Rule the solution is: 1.827655

## RESULT:

The above program is executed successfully by trapezoidal rule and the output is verified.

**KARPAGAM ACADEMY OF HIGHER EDUCATION**
**Coimbatore – 21**
**DEPARTMENT OF MATHEMATICS**

**MODEL PRACTICAL EXAMINATION – Sep'17**

**Date : 23.09.17 (AN)**
**Class : III B.Sc., (Mathematics)**
**Subject : Numerical Methods - Practical** **Subject Code: 15MMU511**
**Maximum Marks : 25**

---

1.a) Write a C – program to find Roots of a given Algebraic Equation using Bisection method.

$2x^3+4x^2+6x+8=0$

**(OR)**

b) Write a C – program to find Roots of a given Algebraic Equation using Newton Raphson method.

$x^x- 1000=0$

2.a) Write a C – program to find Solution of linear algebraic equations using Gauss Elimination method

$x_1+2x_2+x_3=3$
$2x_1+3x_2+3x_3=10$
$3x_1-x_2+2x_3=13$

**(OR)**

b) Write a C – program to find The value of the given integral using Simpson‟s one third rule

$f(x)=\log x$

3.a) Write a C – program to find Solution of linear algebraic equations using Gauss Jordan method

$x_1+x_2+2x_3=4$
$2x_1-1x_2+3x_3=9$
$3x_1-x_2-x_3=2$

**(OR)**

b) Evaluate the integral value of log x limit tends to (4,5.2) when the number of subintervals is 30 by Trapezoidal rule using C program.

4.a) Write a C – program to find Solution of linear algebraic equations using Gauss Seidal method

$10x_1-2x_2-2x_3=-6$
$-x_1+10x_2-3x_3=-7$
$-x_1-x_2+10x_3=-8$

**(OR)**

b) Write a C – program to find Roots of a given Algebraic Equation using Newton Raphson method.

$x^x- 1000=0$

6.a) Write a C – program to find the values of x and y by using Newton‟s forward Interpolation

| X | -0.4 | 0.8 | 3.5 | 5 | 6.3 | 7.7 |
|---|---|---|---|---|---|---|
| Y | 1 | 2 | 4 | 7 | 11 | 16 |

**(OR)**

b) Evaluate the integral value of log x limit tends to (0,1) when the number of subintervals is 30 by Simpson's 1/3 Rule using C program.

7. a) Write a C – program to find the values of x and y by using Newton‟s backward Interpolation

| X | -2 | 3 | 11 | 14 | 23 | 27 |
|---|---|---|---|---|---|---|
| Y | 0 | 0.0699 | 0.1405 | 0.2126 | 0.2867 | 0.3640 |

**(OR)**

b) Evaluate the integral value of log x limit tends to (0,1) when the number of subintervals is 30 by Trapezoidal rule using C program.

8.a) Write a C – program to find solution Interpolation using Lagrange‟s method

| X | -0.2 | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 |
|---|---|---|---|---|---|---|---|
| Y | 2.60 | 3.00 | 3.40 | 4.28 | 7.08 | 14.20 | 29.00 |

**(OR)**

b) Evaluate the integral value of log x limit tends to (4,5.2) when the number of subintervals is 30 by Simpson's 1/3 Rule using C program.

9. a) Write a C – program to find derivative at 1931 using Newton‟s forward difference formula

| X | 1931 | 1942 | 1951 | 1961 | 1971 |
|---|---|---|---|---|---|
| Y | 40.62 | 60.80 | 79.95 | 103.56 | 132.56 |

**(OR)**

b) Write a C – program to find Roots of a given Algebraic Equation using Newton Raphson method.

$x^x - 1000 = 0$

10.a) Write a C – program to find Derivative at 56 using Newton‟s backward difference formula

| X | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
|---|---|---|---|---|---|---|---|
| Y | 3.6840 | 3.7084 | 3.7325 | 3.7563 | 3.7798 | 3.8030 | 3.8259 |

**(OR)**

b) Evaluate the integral value of log x limit tends to (0,1) when the number of subintervals is 30 by Simpson's 1/3 Rule using C program.