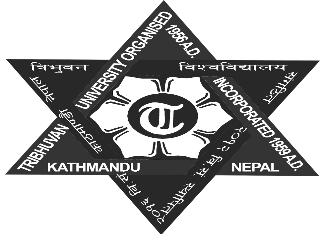
**TRIBHUVAN UNIVERSITY**

**INSTITUTE OF ENGINEERING**

**Lab Sheet**

**PURWANCHAL CAMPUS**

DHARAN-8

**Submitted by:** **Submitted to:**

Name: **Arbind Kumar Mehta** Department of

Roll No: **PUR075BCT017** Electronics & Computer

Faculty: BCT Engineering

Group: II/II ‘A’

Date: 2077/11/03 Checked by: ……………………….

**Title:**

Bisection Method

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include<math.h>

int main()

{

int n=0;

float x,y;

float lr,ur,cr,precission;

precission=0.001;

lr=3;

ur=10;

//y=(pow(x,3)-(4\*x)-9);

//printf("%f",y);

while(1){

cr=(ur+lr)/2;

x=cr;

y=(pow(x,2)-(4\*x)-5);

printf("\nIteration: %d, \nCurrent root %f",n,cr);

if(y<precission&&y>-precission){

printf("\n\nRoot is x = %f",cr);

printf("\n\n\tF(%f) = %f\n",cr,y);

break;

}

if(y>0){

ur=cr;

}

if(y<0){

lr=cr;

}

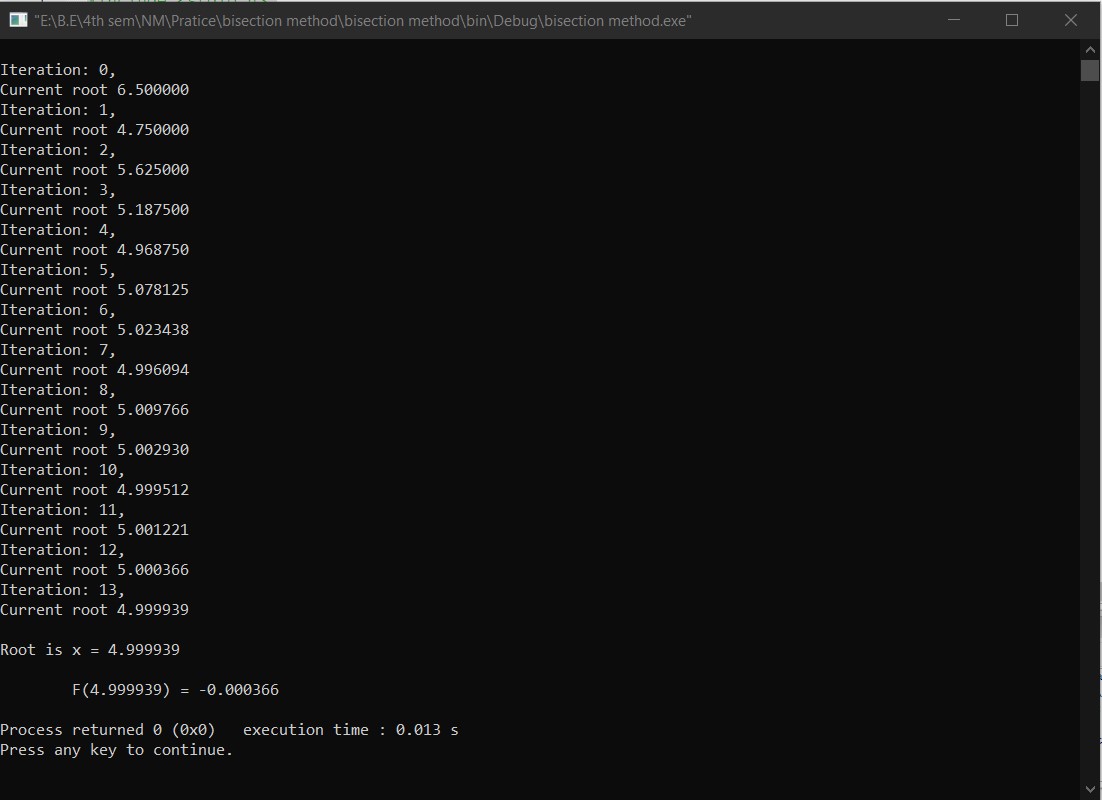
n++;

}

return 0;

}

**Output:**



**Title:**

Secant Method

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define F(x) (cos(x)-(x\*exp(x)))

int main()

{

float xnm1=0,xn=10,xnp1,precission=0.0001;

int n=0;

while(1){

xnp1=(xn-(((xn-xnm1)/(F(xn)-F(xnm1)))\*F(xn)));

if(F(xnp1)<=precission&&F(xnp1)>=(-precission)){

printf("\nRoot is x= %f",xnp1);

printf("\n\tF(%f)= %f",xnp1,F(xnp1));

break;

}

printf("%d, Current x= %f\n",n++,xnp1);

xnm1=xn;

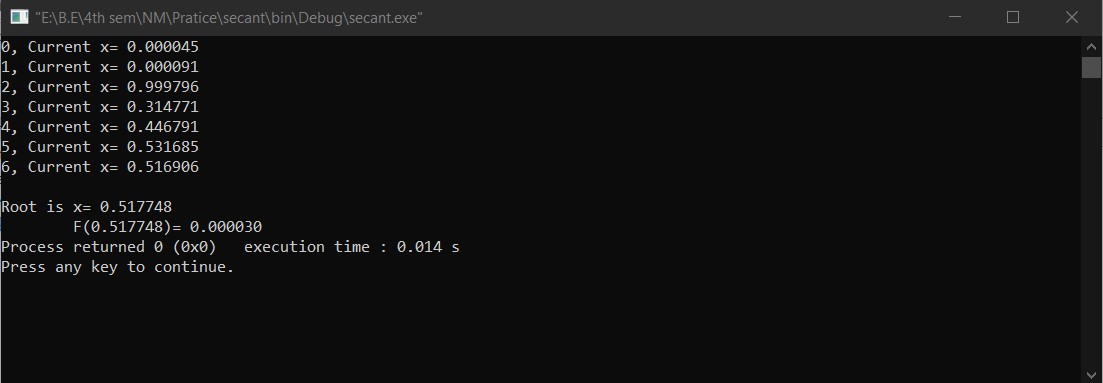
xn=xnp1;

}

return 0;

}

**Output:**

****

**Title:**

Newton Raphson Method

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define F(x) ((3\*x)-cos(x)-1)

float FirstDerivative(float);

int main()

{

float x=100,precission=0.0001;

int n=0;

while(1){

printf("%d, Current x= %f\n",n++,x);

x=(x-(F(x)/FirstDerivative(x)));

if(F(x)<=precission&&F(x)>=(-precission)){

printf("\nRoot is x= %f",x);

printf("\n\tF(%f) = %f",x,F(x));

break;

}

}

return 0;

}

float FirstDerivative(float x)

{

float derv\_precision=0.0001;

float y;

float x1=x-derv\_precision;

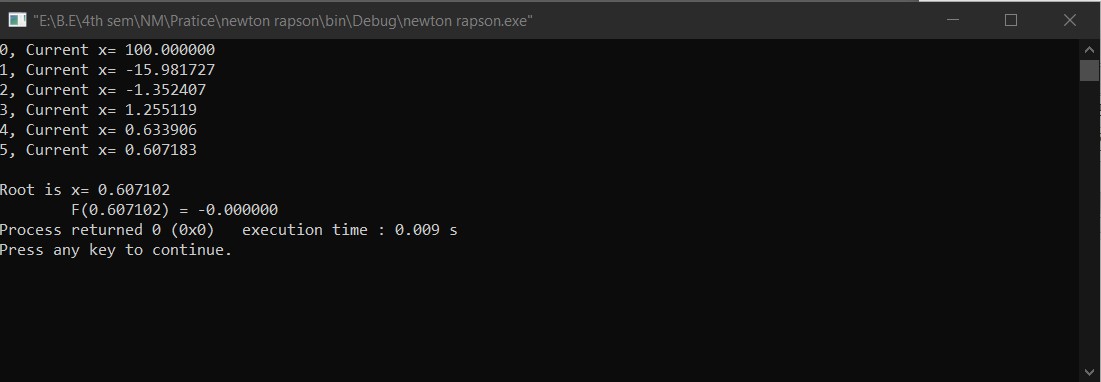
float x2=x+derv\_precision;

y=((F(x2)-F(x1))/(x2-x1));

return y;

}

**Output:**

****

**Title:**

Guss Elimination Method

**Program:**

#include<stdio.h>

#define row 4

#define col 5

void display(float A[row][col]){

int i,j;

for(i=0;i<row;i++){

for(j=0;j<col;j++){

printf("%0.2f\t",A[i][j]);

}

printf("\n");

}

}

int main()

{

int i,j,k,n=0,l,f=0;

float A[row][col]={

{10,-7,3,5,6},

{-6,8,-1,-4,5},

{3,1,4,11,2},

{5,-9,-2,4,7}

};

/\*float A[row][col]={

{1,4,-1,-5},

{1,1,-6,-12},

{3,-1,-1,4},

};\*/

float temp,Factor,Sum,X[row];

printf("Original Equations:\n\n");

display(A);

printf("\n\n");

for(k=0;k<row;k++){

for(f=(k+1);f<row;f++){

//printf("\n(%f/%f)\n",A[f][k],A[k][k]);

Factor=(A[f][k]/A[k][k]);

for(i=k;i<col;i++){

temp=(A[f][i]-(A[k][i]\*Factor));

A[f][i]=temp;

}

//printf("\n");

//printf("\n%f\n",A[k][k]);

}

}

printf("Equations after elimination:\n\n");

display(A);

printf("\n");

X[row-1]=A[row-1][col-1]/A[row-1][col-2];

//printf("T %f",X[row-1]);

for(i=(row-2);i>=0;i--){

Sum=0;

for(j=(col-2);j>=(i+1);j--){

//printf("X%f\t",X[i+1]);

//printf("y%f\t%f",X[j],A[i][j]);

Sum=Sum+(A[i][j]\*X[j]);

}

//printf("ksd\n");

X[i]=(A[i][col-1]-Sum)/A[i][i];

//printf("l%f",Sum);

//printf("\n");

}

printf("\nSolution is:\n\n");

for(i=0;i<row;i++){

printf("%0.2f\t",X[i]);

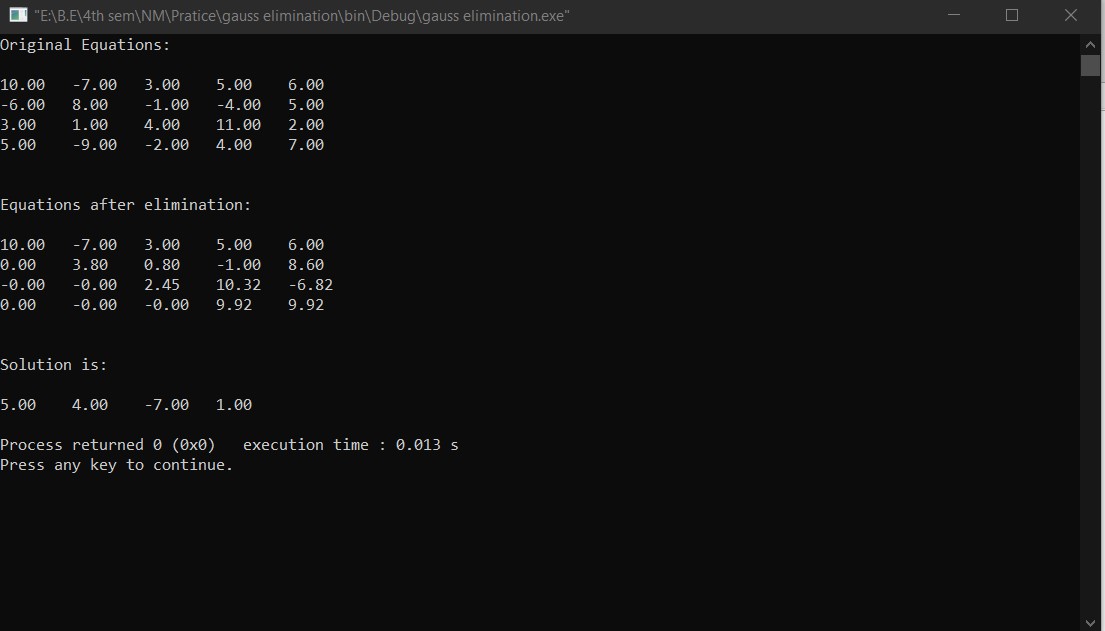
}

printf("\n");

return(0);

}

**Output:**



**Title:**

Guss Jordan Method

**Program:**

#include<stdio.h>

#define row 4

#define col 5

void display(float A[row][col]){

int i,j;

for(i=0;i<row;i++){

for(j=0;j<col;j++){

printf("%0.2f\t",A[i][j]);

}

printf("\n");

}

}

int main()

{

int i,j,k,n=0,l,f=0;

float A[row][col]={

{10,-7,3,5,6},

{-6,8,-1,-4,5},

{3,1,4,11,2},

{5,-9,-2,4,7}

};

/\*float A[row][col]={

{1,4,-1,-5},

{1,1,-6,-12},

{3,-1,-1,4},

};\*/

float temp,Factor,Sum,X[row];

printf("Original Equations:\n\n");

display(A);

printf("\n\n");

for(k=0;k<row;k++){

for(f=(k+1);f<row;f++){

//printf("\n(%f/%f)\n",A[f][k],A[k][k]);

Factor=(A[f][k]/A[k][k]);

for(i=k;i<col;i++){

temp=(A[f][i]-(A[k][i]\*Factor));

A[f][i]=temp;

}

//printf("\n");

//printf("\n%f\n",A[k][k]);

}

}

printf("Equations after elimination:\n\n");

display(A);

printf("\n");

for(k=(row-2);k>=0;k--){

for(f=(k);f>=0;f--){

//printf("%f\t",A[k][f]);

//printf("(%f/%f)\t",A[f][k+1],A[k+1][k+1]);

Factor=(A[f][k+1]/A[k+1][k+1]);

for(i=f;i<col;i++){

//printf("%f\t",A[f][i]);

//printf("xx%f\t",A[k+1][i]);

temp=(A[f][i]-(A[k+1][i]\*Factor));

A[f][i]=temp;

}

//printf("\n");

}

}

printf("Equations after elimination:\n\n");

display(A);

printf("\n");

printf("\nSolution is:\n\n");

for(i=0;i<row;i++){

printf("%0.2f\t",(A[i][col-1]/A[i][i]));

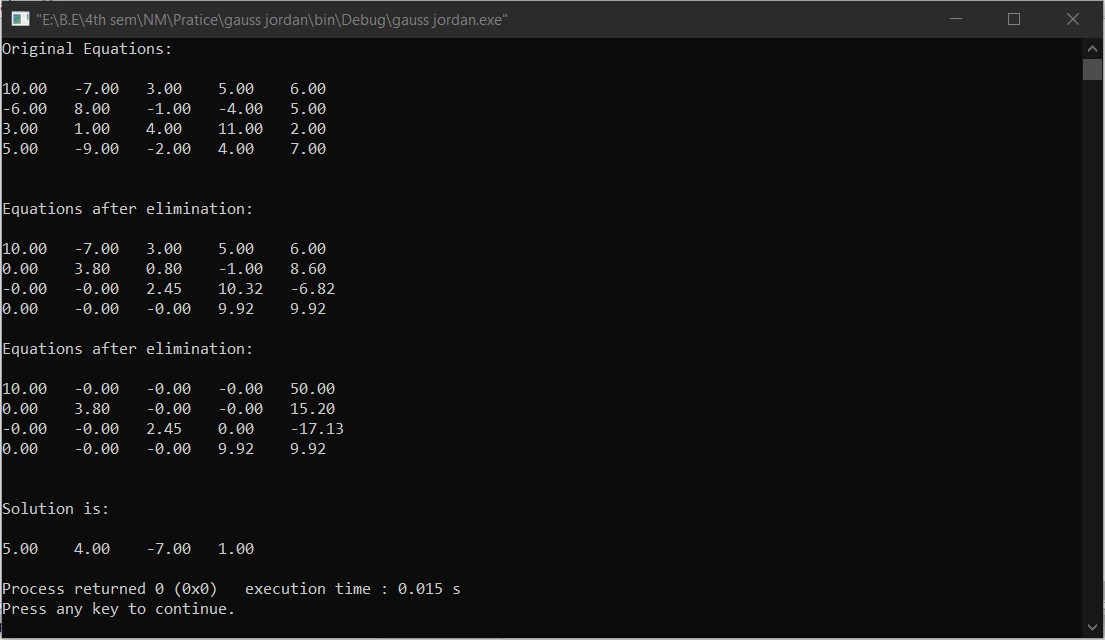
}

printf("\n");

return(0);

}

**Output:**

****

**Title:**

Guss Sidel Method

**Program:**

#include<stdio.h>

#define row 4

#define col 5

void display(float A[row][row]){

int i,j;

for(i=0;i<row;i++){

for(j=0;j<row;j++){

printf("%0.2f\t",A[i][j]);

}

printf("\n");

}

}

int main()

{

int i,j,k,n=0,l,f=0;

/\*float A[row][col]={

{20,1,-2,17},

{3,20,-1,-18},

{2,-3,20,25}

};\*/

float A[row][col]={

{10,-2,-1,-1,3},

{-2,10,-1,-1,15},

{-1,-1,10,-2,27},

{-1,-1,-2,10,-9}

};

float temp,Factor,Sum,InitialGuess[row]={0,0,0};

printf("Original Equations:\n\n");

display(A);

printf("\n\n");

for(n=0;n<10;n++){ //precision

for(i=0;i<row;i++){

Sum=A[i][col-1];

//printf("X%0.2f\n",Sum);

for(j=0;j<(row);j++){

if(i!=j){

Sum=(Sum-(InitialGuess[j]\*A[i][j]));

}

}

InitialGuess[i]=(Sum/A[i][i]);

//printf("X%0.2f",A[i][i]);

}

}

printf("Solution are:\n\n");

for(i=0;i<row;i++){

printf("%f\t",InitialGuess[i]);

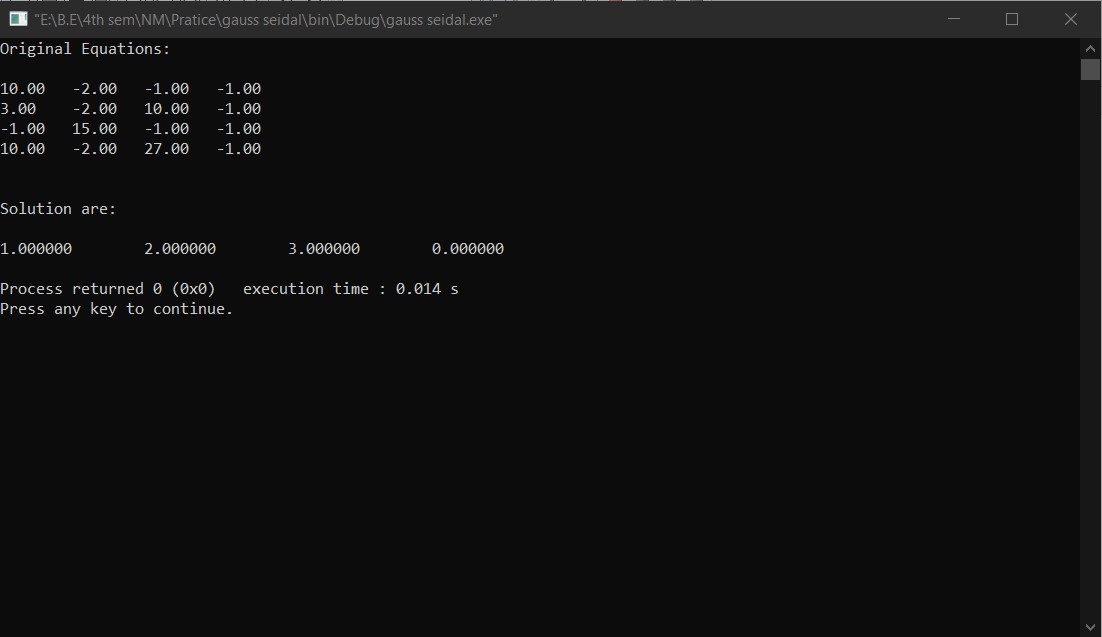
}

printf("\n");

return(0);

}

**Output:**



**Title:**

Lagrange Interpolation

**Program:**

#include <stdio.h>

#include <stdlib.h>

#define Size 5

int main()

{

float X[Size]={5,7,11,13,17};

float Y[Size]={150,392,1452,2366,5202};

/\*float X[Size]={1,3,4,6,7};

float Y[Size]={1,53,127,531,687};\*/

int i,j;

float x=9,Sum=1,Px=0,numtr,dnumtr;

for(i=0;i<Size;i++){

//printf("%f\t",Y[i]);

numtr=1;

dnumtr=1;

for(j=0;j<Size;j++){

if(i!=j){

numtr=numtr\*(x-X[j]);

dnumtr=dnumtr\*(X[i]-X[j]);

//printf("Xj= %f\t",X[j]);

}

}

//printf("\n");

Sum=numtr/dnumtr;

Px=Px+(Y[i]\*Sum);

printf("L%d= %f\n",i,Sum);

Sum=1;

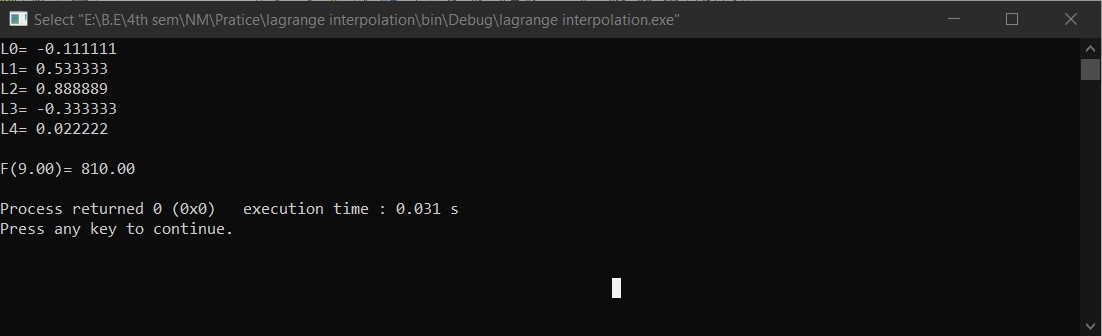
}

printf("\nF(%0.2f)= %0.2f\n",x,Px);

return 0;

}

**Output:**



**Title:**

Newton Divided Difference Interpolation

**Program:**

#include <stdio.h>

#include <stdlib.h>

#define Size 5

/\*

float X[Size]={5,7,11,13,17};

float Y[Size]={150,392,1452,2366,5202};

\*/

/\*float X[Size]={-4,-1,0,2,5};

float Y[Size]={1245,33,5,9,1335};\*/

float X[Size]={1,3,4,6,7};

float Y[Size]={1,53,127,531,687};

float table[Size][Size];

void DividedDifference(int n){

float diff;

int i,j;

if(n==0){

for(i=0;i<(Size-(n+1));i++){

diff=((Y[i+1]-Y[i])/(X[i+1]-X[i]));

table[i][n]=diff;

}

}

else{

for(i=0;i<(Size-(n+1));i++){

//printf("T= %f\n",table[n-1][i]);

diff=((table[i+1][n-1]-table[i][n-1])/(X[i+n+1]-X[i]));

table[i][n]=diff;

}

}

}

int main()

{

int i,j;

float x=5.5,Sum=1,Px=0,numtr,dnumtr;

for(i=0;i<(Size-1);i++){

DividedDifference(i);

}

printf("\n Divided Difference table is:\n\n");

for(i=0;i<(Size-1);i++){

for(j=0;j<(Size-(i+1));j++){

printf(" %7.2f\t",table[i][j]);

}

printf("\n");

}

printf("\n");

Px=Y[0];

//printf("%f",Px);

for(i=0;i<(Size-1);i++){

for(j=0;j<(i+1);j++){

Sum=Sum\*(x-X[j]);

//printf("l%f\t",X[j]);

}

//printf("%f",table[0][i]\*Sum);

//printf("\n");

Px=Px+(Sum\*table[0][i]);

Sum=1;

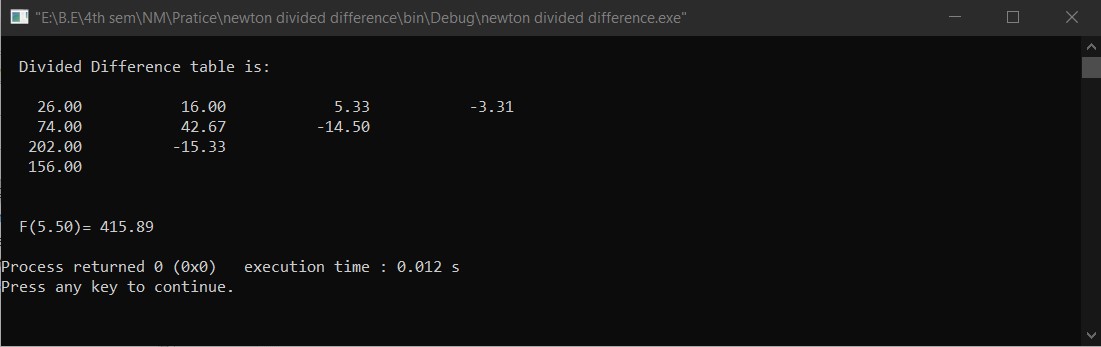
}

printf("\n F(%0.2f)= %0.2f\n",x,Px);

return 0;

}

**Output:**



**Title:**

Newton Forward Interpolation

**Program:**

#include <stdio.h>

#include <stdlib.h>

#define Size 6

/\*

float X[Size]={5,7,11,13,17};

float Y[Size]={150,392,1452,2366,5202};

\*/

/\*float X[Size]={-4,-1,0,2,5};

float Y[Size]={1245,33,5,9,1335};\*/

/\*float X[Size]={0,0.001,0.002,0.003,0.004,0.005};

float Y[Size]={1.121,1.123,1.1255,1.127,1.128,1.1285};\*/

/\*float X[Size]={0,0.2,0.4,0.6,0.8,1.0};

float Y[Size]={1,0.808,0.664,0.616,0.712,1};\*/

float X[Size]={0,0.04,0.08,0.12,0.16,0.20};

float Y[Size]={0,3,26,90,214,419};

float table[Size][Size];

float Factroial(int num){

int i;

float fact=1;

for(i=1;i<=num;i++){

fact=fact\*i;

}

return fact;

}

void DividedDifference(int n){

float diff;

int i,j;

if(n==0){

for(i=0;i<(Size-(n+1));i++){

diff=(Y[i+1]-Y[i]);

table[i][n]=diff;

}

}

else{

for(i=0;i<(Size-(n+1));i++){

//printf("T= %f\n",table[n-1][i]);

diff=(table[i+1][n-1]-table[i][n-1]);

table[i][n]=diff;

}

}

}

int main()

{

int i,j;

float x=0.00070,Sum=1,Px=0,h,u;

for(i=0;i<(Size-1);i++){

DividedDifference(i);

}

h=X[1]-X[0];

u=((x-X[0])/h);

printf("\n\tp= %0.2f\n\th= %0.2f\n",u,h);

printf("\n Forward Difference table is:\n\n");

for(i=0;i<(Size-1);i++){

for(j=0;j<(Size-(i+1));j++){

printf(" %8f\t",table[i][j]);

}

printf("\n");

}

printf("\n");

Px=Y[0];

//printf("l%f",Px);

for(i=0;i<(Size-1);i++){

for(j=0;j<(i+1);j++){

Sum=Sum\*(u-j);

}

Px=Px+((Sum\*table[0][i])/Factroial(i+1));

Sum=1;

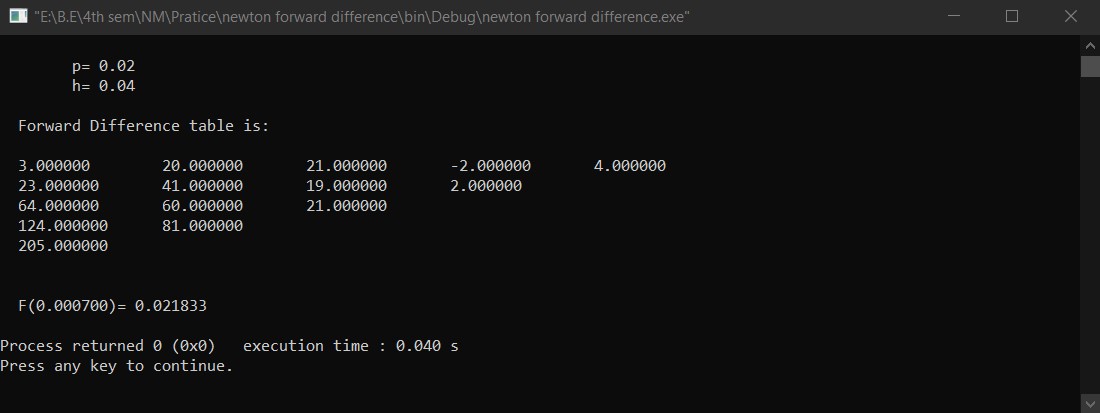
}

printf("\n F(%f)= %f\n",x,Px);

return 0;

}

**Output:**

****

**Title:**

Numerical differentiation (central formulafor f’(x)and f’’(x))

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include<math.h>

#define Size 7

float X[Size]={0,0.1,0.2,0.3,0.4,0.5,0.6}; //central

float Y[Size]={30.13,31.62,32.87,33.64,33.95,33.81,33.24};

/\*float X[Size]={1,1.05,1.10,1.15,1.20,1.25,1.30}; //central

float Y[Size]={1,1.0247,1.0488,1.0723,1.0954,1.1180,1.1401};\*/

float table[Size][Size]={0}; //initilize array with 0

float Factroial(int num){

int i;

float fact=1;

for(i=1;i<=num;i++){

fact=fact\*i;

}

return fact;

}

void CentralDifference(){

float diff;

int i,j;

for(int n=0;n<(Size-1);n++){

if(n==0){

for(i=0;i<(Size-(n+1));i++){

diff=(Y[i+1]-Y[i]);

table[i][n]=diff;

}

}

else{

for(i=0;i<(Size-(n+1));i++){

//printf("T= %f\n",table[n-1][i]);

diff=(table[i+1][n-1]-table[i][n-1]);

table[i][n]=diff;

}

}

}

}

float FirstDiff(float h, int X0){

float sum=0;

int cur\_indx=X0,term\_count=0;

for(int i=0;i<Size;i+=2){

if(i%2==0){

sum=sum+((pow(Factroial(term\_count),2)/Factroial(i+1))\*((table[cur\_indx][i]+table[cur\_indx-1][i])/2));

}

else{

sum=sum-(((pow(Factroial(term\_count),2)/Factroial(i+1)))\*((table[cur\_indx][i]+table[cur\_indx-1][i])/2));

}

//printf("%d %f ",i,(pow(Factroial(term\_count),2)/Factroial(i+1)));

cur\_indx--;

term\_count++;

}

sum=sum/h;

//printf("\n%d",X0);

return sum;

}

float SecondDiff(float h, int X0){

float sum=0;

int cur\_indx=X0,term\_count=0;

for(int i=0;i<Size;i++){

if(i%2!=0){

cur\_indx--;

if(term\_count%2==0){

sum=sum+((pow(Factroial(term\_count),2)/(Factroial(i)\*(term\_count+1)))\*(table[cur\_indx][i]));

}

else{

sum=sum-((pow(Factroial(term\_count),2)/(Factroial(i)\*(term\_count+1)))\*(table[cur\_indx][i]));

}

//printf("%d \n",i+1);

//printf(" %f ",(pow(Factroial(term\_count),2)/(Factroial(i)\*(term\_count+1))));

term\_count++;

}

}

sum=sum/pow(h,2);

//printf("\n%d",X0);

return sum;

}

int main()

{

int i,j,X0;

float x=0.3,Px=0,h,u,temp;

//finding nearer value index

X0=0;

temp=X[X0];

for(i=0;i<Size;i++){

for(j=1;j<Size;j++){

if(fabs(X[j]-x)<=fabs(temp-x)){

X0=j;

temp=X[j];

}

}

}

h=X[1]-X[0];

u=((x-X[X0])/h);

Px=Y[X0];

printf("\n X0= %f\n",X[X0]);

printf(" h = %f\n",h);

printf(" p = %f\n",u);

CentralDifference();

printf("\n Central Difference table is:\n\n");

for(i=0;i<(Size-1);i++){

for(j=0;j<(Size-(i+1));j++){

printf(" %f\t",table[i][j]);

}

printf("\n");

}

printf("\n");

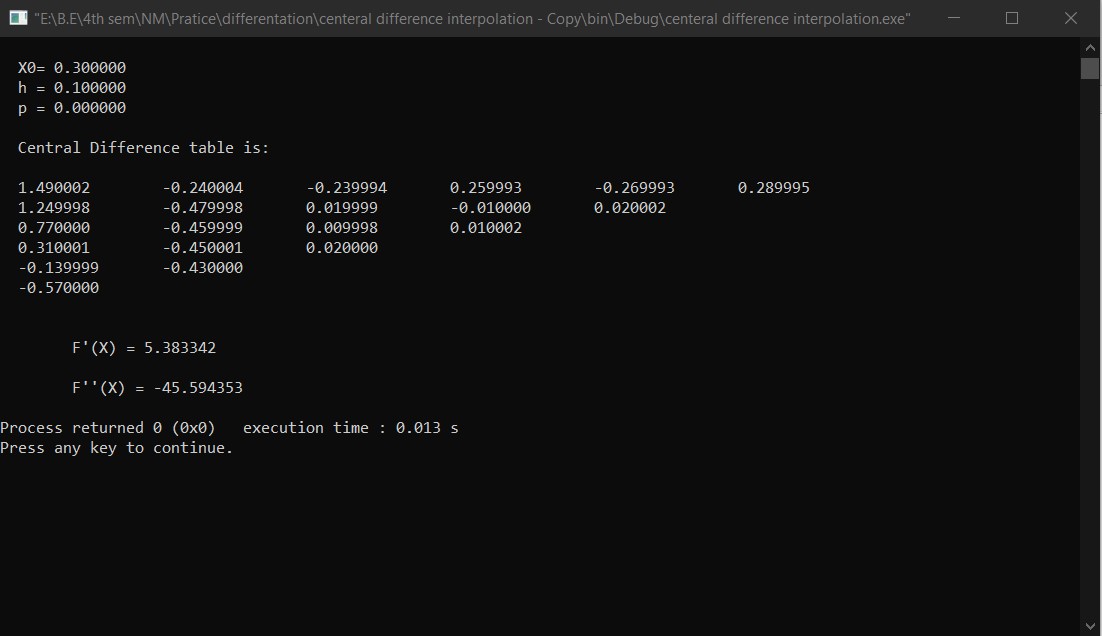
printf("\n\tF'(X) = %f\n",FirstDiff(h,X0));

printf("\n\tF''(X) = %f\n",SecondDiff(h,X0));

return 0;

}

**Output:**

****

**Title:**

Numerical Integration (trapezoidal,Simpson’s 1/3 and simpson’s 3/8)

**Program:**

#include <iostream>

#include<math.h>

#define Size 10

//#define F(X) (1/(1+pow(X,2))

using namespace std;

float F(float X){

return (1/(1+pow(X,2)));

}

int ul=6,ll=0;

float interval=1;

int N=(ul-ll)/interval; //for number of iteration

float X[Size],Y[Size];

/\*float X[]={};

float Y[]={};\*/

void AssignXY(){

float tmp1;

tmp1=ll;

for(int i=0;i<=N;i++){

//cout<<F(i)<<"\n";

X[i]=i;

Y[i]=F(tmp1);

tmp1+=interval;

}

}

void DisplayXY(){

cout<<" "<<"X"<<" "<<"Y"<<endl;

for(int i=0;i<=N;i++){

cout<<" "<<X[i]<<" "<<Y[i]<<endl;

}

}

float Trapezoidal(){

float sum=0;

for(int i=1;i<=(N-1);i++){

sum=sum+Y[i];

}

sum=interval\*(((Y[0]+Y[N])+(2\*sum))/2);

return sum;

}

float Simpson13(){

float sum=0;

for(int i=1;i<=(N-1);i++){

if(i%2==0){

sum=sum+(2\*Y[i]);

}

else{

sum=sum+(4\*Y[i]);

}

}

sum=interval\*(((Y[0]+Y[N])+sum)/3);

return sum;

}

float Simpson38(){

float sum=0;

for(int i=1;i<=(N-1);i++){

if(i%3==0){

sum=sum+(2\*Y[i]);

}

else{

sum=sum+(3\*Y[i]);

}

}

sum=3\*interval\*(((Y[0]+Y[N])+sum)/8);

return sum;

}

int main()

{

AssignXY();

DisplayXY();

cout<<"\nBy Trapezoidal rule,\n\t\t= "<<Trapezoidal();

cout<<"\nBy Simpson's 1/3 rule,\n\t\t= "<<Simpson13();

cout<<"\nBy Simpson's 3/8 rule,\n\t\t= "<<Simpson38();

return 0;

}

**Output:**

