

Term-Work Of

CBNST Lab(PMA-502)

Submitted in partial fulfillment of the requirement for the V semester

# Bachelor of Technology

**By**

**Karan Chauhan**

**2261308**

**5thSem**

**Under the Guidance of Assistant Professor Mr. Ansh Dhingra**

**Dept.of CSE**

# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS SATTAL ROAD, P.O. BHOWALI

**2024-2025**



STUDENT’SDECLARATION

I Dev Joshi here by declare the work which is being presented in the term-work, entitled

“CBNST Lab” in partial fulfillment of the requirement for the award of the degree **B.Tech** in the session **2024-25**, is an authentic record of my own work carried out under the supervision of Mr. Parthak Mehra. The matter embodied in this term-work has not been submitted by me for the award of any other degree.

Date:

Karan Chuahan

**ACKNOWLEDGEMENT**

We take immense pleasure in thanking Honorable **“Mr. Ansh Dhingra”** (**Assistant Professor, CSE, GEHU Bhimtal Campus**) to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me everything that we need.We again want to extend thanks to our President **“Prof. (Dr.) Kamal Ghanshala”** for providing us all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to Professor **“Col. Anil Kumar”**(Director GEHU Bhimtal),

other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this thesis.

Finally,yetimportantly,wewouldliketoexpressmyheartiestthankstoourbelovedparents,

for their moral support, affection and blessings. We would also like to pay our sincere thanks to all ourfriends and well-wishers fortheir help and wishes forthe successful completion of this research.

# Karan Chauhan

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **SRNo.** | **Objective** | **Date** | **Sign** |
| **1.** | WAP to find the roots of non-linear equation using Bisection method. |  |  |
| **2.** | WAP to find the roots of non-linear equation using False position method. |  |  |
| **3.** | WAP to find the roots of non-linear equation using Newton’s Raphson method. |  |  |
| **4.** | WAP to find the roots of non-linear equation using Iteration method. |  |  |
| **5.** | WAP to interpolate numerically using Newton’s forward difference method. |  |  |
| **6.** | WAP to interpolate numerically using Newton’s backward difference method. |  |  |
| **7.** | WAP to interpolate numerically using Lagrange’s method. |  |  |
| **8.** | WAP to Integrate numerically using Trapezoidal rule. |  |  |
| **9.** | WAP to Integrate numerically using Simpson’s 1/3 rules. |  |  |
| **10.** | WAP to Integrate numerically using Simpson’s 3/8 rules. |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. WAPtofindtherootsofnon-linearequationusingBisectionmethod.

#include<bits/stdc++.h>using namespace std;

doublefun(doublex)

{

returnx\*x\*x-4\*x-9;

}

intmain()

{

doublex0,x1,x2;

cout<<"Enter the interval for your function:"; cin>> x0 >> x1;

if(fun(x0)\*fun(x1)>0){

cout<<"No root in this interval."<<endl; return 0;

}

inti;

for(i=0;i<10;i++)

{

x2=(x0+x1)/2;

if(fun(x2)==0)

{

cout<<"Exact root:"<<x2<<endl; return 0;

}

elseif(fun(x0)\*fun(x2)< 0)

{

x1=x2;//Rootliesinleftsubinterval

}

else

{

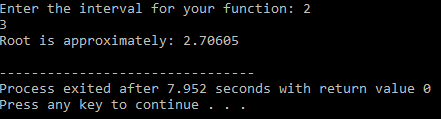
x0=x2;//Rootliesinrightsubinterval

}

}

cout<<"Rootisapproximately:"<<x2<<endl; return 0;

}



1. WAPtofindtherootsofnon-linearequationusingFalsepositionmethod.

#include<bits/stdc++.h>using namespace std;

doublefun(doublex)

{

returnx\*x\*x-x-1;

}

intmain()

{

doublex0,x1,x2;

cout<<"enter the interval for your function"<<endl; cin>>x0>>x1;

int i; for(i=0;i<10;i++)

{

x2=(x0\*fun(x1)-x1\*fun(x0))/(fun(x1)-fun(x0));

if(fun(x2)==0)

{

cout<<x2; return 0;

}

elseif(fun(x0)\*fun(x2)<0)

{

x1=x2;

}

elseif(fun(x1)\*fun(x2)<0)

{

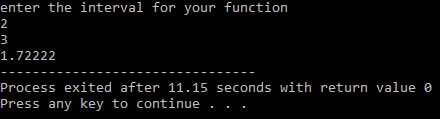
x0=x2;

}

}

cout<<x2; return 0;

}



1. WAPtofindtherootsofnon-linearequationusingNewton’sRaphsonmethod.

#include<bits/stdc++.h>using namespace std;

doublefun(doublex)

{

returnx\*x\*x-4\*x-9;

}

doublederivative(doublex)

{

return3\*x\*x- 1;

}

intmain()

{

doublex0,x1;

cout<<"Enter the initial guess:"; cin>> x0;

inti;

for(i=0;i<10;i++)

{

x1=x0-fun(x0)/derivative(x0);

if(fun(x1)==0)//Ifrootisfound

{

cout<<"Exact root:"<<x1<<endl; return 0;

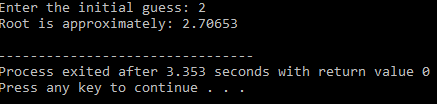
}

x0=x1;//Updatex0fornextiteration

}

cout<<"Root is approximately:"<<x1<<endl; return 0;

}



1. WAPtofindtherootsofnon-linearequationusingIterationmethod.

#include<bits/stdc++.h>using namespace std;

doubleg(doublex)

{

returncbrt(x+1);//Rearrangedfunctionx= g(x)

}

intmain()

{

doublex0,x1;

cout<<"Entertheinitialguess:"; cin>> x0;

inti;

for(i=0;i<10;i++)//Adjustthenumberofiterationsorusetolerance

{

x1=g(x0);//Fixedpointiterationformula:x1=g(x0)

if(fabs(x1-x0)<1e-6)//Ifthedifferenceissmallenough,stop

{

cout<<"Rootisapproximately:"<<x1<<endl; return 0;

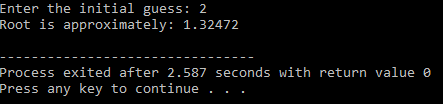
}

x0=x1;//Updatex0forthenextiteration

}

cout<<"Rootafteriterationsisapproximately:"<<x1<<endl; return 0;

}



1. WAPtointerpolatenumericallyusingNewton’sforwarddifferencemethod.

#include <iostream> #include <iomanip> using namespace std;

//Functiontocalculatefactorial int factorial(int n) {

intfact=1;

for(inti=2;i<=n;i++){ fact \*= i;

}

returnfact;

}

intmain(){ int n;

cout<<"Enter the number of data points:"; cin>> n;

doublex[n],y[n][n];//yi s the forward difference table

//Inputdatapoints

cout<<"Enterthexandyvalues:\n"; for (int i = 0; i< n; i++) {

cout<<"x["<<i<<"]:"; cin>> x[i];

cout<<"y["<<i<<"]:"; cin>> y[i][0];

}

//Calculatetheforwarddifferencetable for (int j = 1; j < n; j++) {

for(inti=0;i<n-j;i++){

y[i][j]= y[i+1][j-1]-y[i][j-1];

}

}

// Display the forward difference table cout<<"\nForwardDifferenceTable:\n"; for (int i = 0; i< n; i++) {

cout<<setw(10) << x[i]; for(intj=0;j<n-i;j++){

cout<<setw(10)<<y[i][j];

}

cout<<endl;

}

doublexp;

cout<<"\nEnterthexvaluetointerpolate:"; cin>>xp;

//NewtonForwardInterpolation

doubleh=x[1]-x[0];//Assumingequalspacing double u = (xp - x[0]) / h;

doubleyp=y[0][0];//Initialvalueofinterpolatedresult

for (int i = 1; i< n; i++) { double term = y[0][i]; for(intj=0;j<i;j++){

term\*=(u-j);

}

term/=factorial(i); yp += term;

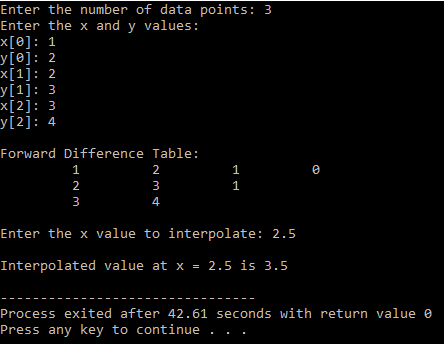
}

//Displaythe result

cout<<"\nInterpolatedvalueatx="<<xp<<"is"<<yp<<endl;

return0;

}



1. WAPtointerpolatenumericallyusingNewton’sbackwarddifferencemethod.

#include <iostream> #include <iomanip> usingnamespacestd;

//Functiontocalculatefactorial int factorial(int n) {

intfact=1;

for(inti=2;i<=n;i++){ fact \*= i;

}

returnfact;

}

intmain(){ int n;

cout<<"Enterthenumberofdatapoints:"; cin>> n;

doublex[n],y[n][n];//yisthebackwarddifferencetable

//Inputdatapoints

cout<<"Enterthexandyvalues:\n"; for (int i = 0; i< n; i++) {

cout<<"x["<<i<<"]:"; cin>> x[i];

cout<<"y["<<i<<"]:"; cin>> y[i][0];

}

//Calculatethebackwarddifferencetable for (int j = 1; j < n; j++) {

for(inti=n-1;i>= j;i--){

y[i][j]= y[i][j-1]- y[i-1][j-1];

}

}

// Display the backward difference table cout<<"\nBackwardDifferenceTable:\n"; for (int i = 0; i< n; i++) {

cout<<setw(10)<<x[i]; for(intj=0;j<= i;j++){

cout<<setw(10)<<y[i][j];

}

cout<<endl;

}

doublexp;

cout<<"\nEnterthexvaluetointerpolate:"; cin>>xp;

//NewtonBackwardInterpolation

doubleh=x[1]-x[0];//Assumingequalspacing double u = (xp - x[n - 1]) / h;

doubleyp=y[n-1][0];//Initialvalueofinterpolatedresult

for (int i = 1; i< n; i++) { doubleterm =y[n-1][i]; for(intj=1;j<= i;j++){

term\*=(u+j-1);

}

term/=factorial(i); yp += term;

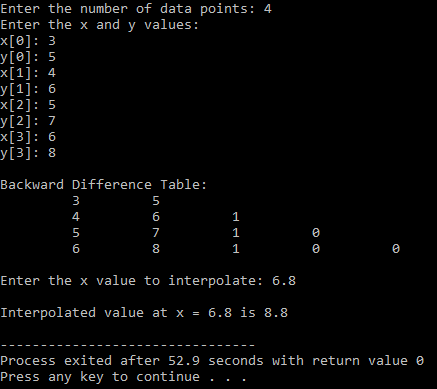
}

//Displaythe result

cout<<"\nInterpolatedvalueatx="<<xp<<"is"<<yp<<endl;

return0;

}



1. WAPtointerpolatenumericallyusingLagrange’smethod.

#include <iostream> usingnamespacestd;

doublelagrangeInterpolation(doublex[],doubley[],intn,doublexp){ double yp = 0;// Initial value of interpolated result

for(inti=0;i<n;i++){ double term = y[i];

for(intj=0;j<n;j++){ if (j != i) {

term\*=(xp-x[j])/(x[i]-x[j]);

}

}

yp+=term;//Addtermtofinalresult

}

returnyp;

}

intmain(){ int n;

cout<<"Enterthenumberofdatapoints:"; cin>> n;

doublex[n],y[n];

cout<<"Enterthexandyvalues:\n"; for (int i = 0; i< n; i++) {

cout<<"x["<<i<<"]:"; cin>> x[i];

cout<<"y["<<i<<"]:"; cin>> y[i];

}

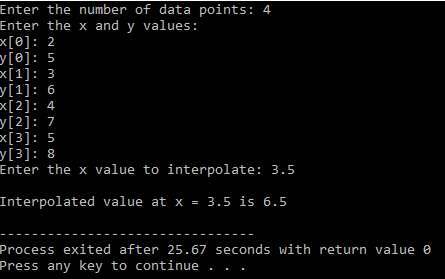
doublexp;

cout<<"Enterthexvaluetointerpolate:"; cin>>xp;

doubleyp=lagrangeInterpolation(x,y,n,xp);

cout<<"\nInterpolatedvalueatx="<<xp<<" is" <<yp<<endl; return 0;

}



1. WAPtoIntegratenumericallyusingTrapezoidalrule.

#include <iostream> #include <cmath> usingnamespacestd;

//Definethefunctiontointegrate double f(double x) {

returnx\*x;//Example:f(x)=x^2

}

//FunctiontocalculatetheintegralusingtheTrapezoidalRule double trapezoidalRule(double a, double b, int n) {

doubleh=(b-a)/n;//Calculatethewidthofeachsubinterval double sum = f(a) + f(b); // Add the first and last terms

//Calculatethesum ofthemiddleterms for (int i = 1; i< n; i++) {

doublex=a+i\*h; sum += 2 \* f(x);

}

//Applythetrapezoidalruleformula return (h / 2) \* sum;

}

int main() { doublea,b; int n;

//Inputthelimitsofintegrationandnumberofsubintervals cout<< "Enter the lower limit (a): ";

cin>>a;

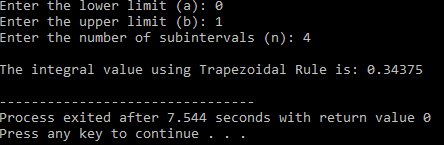
cout<<"Entertheupperlimit(b):"; cin>> b;

cout<<"Enterthenumberofsubintervals(n):"; cin>> n;

doubleresult=trapezoidalRule(a,b,n);

cout<<"\nTheintegralvalueusingTrapezoidalRuleis:" <<result<<endl; return 0;

}



1. WAPtoIntegratenumericallyusingSimpson’s1/3rules.

#include <iostream> #include <cmath> usingnamespacestd;

//Definethefunctiontointegrate double f(double x) {

returnx\*x;//Example:f(x)=x^2

}

//FunctiontocalculatetheintegralusingSimpson's1/3Rule double simpsonsRule(double a, double b, int n) {

//Checkifniseven if (n % 2 != 0) {

cout<<"Error:Numberofsubintervals(n)mustbeeven.\n"; return -1;

}

doubleh=(b-a)/n;//Calculatethewidthofeachsubinterval double sum = f(a) + f(b); // Add the first and last terms

//Calculatethesum ofoddterms(4\*f(x\_i)) for (int i = 1; i< n; i += 2) {

doublex=a+i\*h; sum += 4 \* f(x);

}

//Calculatethesumofeventerms(2\*f(x\_i)) for (int i = 2; i< n; i += 2) {

doublex=a+i\*h; sum += 2 \* f(x);

}

//ApplytheSimpson's1/3Ruleformula return (h / 3) \* sum;

}

int main() { doublea,b; int n;

//Inputthelimitsofintegrationandnumberofsubintervals cout<< "Enter the lower limit (a): ";

cin>>a;

cout<<"Entertheupperlimit(b):"; cin>> b;

cout<<"Enterthenumberofsubintervals(n):"; cin>> n;

//Calculatetheintegral

doubleresult=simpsonsRule(a,b, n);

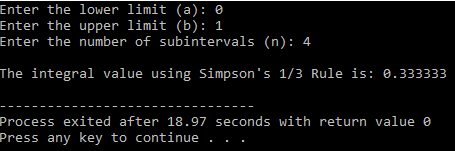
//Displaytheresultifnisvalid if (result != -1) {

cout<<"\nTheintegralvalueusingSimpson's1/3Ruleis:"<<result<<endl;

}

return0;

}



1. WAPtoIntegratenumericallyusingSimpson’s3/8rules.

#include <iostream> #include <cmath> usingnamespacestd;

//Definethefunctiontointegrate double f(double x) {

returnx\*x;//Example:f(x)=x^2

}

//FunctiontocalculatetheintegralusingSimpson's3/8Rule double simpsons38Rule(double a, double b, int n) {

//Check ifnisamultipleof3 if (n % 3 != 0) {

cout<<"Error:Numberofsubintervals(n)mustbeamultipleof3.\n"; return -1;

}

doubleh=(b-a)/n;//Calculatethewidthofeachsubinterval double sum = f(a) + f(b); // Add the first and last terms

//Calculatethesum oftermsmultipliedby3(oddandmostmiddlepoints) for (int i = 1; i< n; i++) {

doublex=a+i\*h; if (i % 3 == 0) {

sum+=2\*f(x);//Every3rdtermgetsmultipliedby2

}else{

sum+=3\*f(x);//Allothertermsgetmultipliedby3

}

}

//ApplytheSimpson's3/8Ruleformula return (3 \* h / 8) \* sum;

}

int main() { doublea,b; int n;

//Inputthelimitsofintegrationandnumberofsubintervals cout<< "Enter the lower limit (a): ";

cin>>a;

cout<<"Entertheupperlimit(b):"; cin>> b;

cout<<"Enterthenumberofsubintervals(n):"; cin>> n;

//Calculatetheintegral

doubleresult=simpsons38Rule(a,b,n);

//Displaytheresultifnisvalid

if(result!= -1){

cout<<"\nTheintegralvalueusingSimpson's3/8Ruleis:"<<result<<endl;

}

return0;

}

