1.

// Bisection-Method

#include<iostream>

#include<cmath>

#include<iomanip>

using namespace std;

double f(double x)

{

double a = pow(x,3) – x - 1.0;

return a;

}

int main()

{

cout.precision(5);

cout.setf(ios::fixed);

double a,b,c,e,fa,fb,fc,ss;

a:cout<<"Enter the initial guesses:\na=";

cin>>a;

cout<<"\nb=";

cin>>b;

cout<<"\nEnter the degree of accuracy desired"<<endl;

cin>>e;

if (f(a)\*f(b)>=0)

{

cout<<"Please enter a different intial guess"<<endl;

goto a;

}

else

{

while (fabs(a-b)>=e)

{

c=(a+b)/2.0;

fa=f(a);

fb=f(b);

fc=f(c);

cout<<"a="<<a<<" "<<"b="<<b<<" "<<"c="<<c<<" f(c)="<<fc<<endl;

if (fc==0)

{

cout<<"The root of the equation is= "<<c;

break;

}

else if (fa\*fc>0)

{

a=c;

}

else

{

b=c;

}

}

}

cout<<"The root of the equation is= "<<c;

cout<<"\n\n To calculate other, press 1 : ";

cin>>ss;

if (ss==1)

goto a;

return 0;

}

// Newton-Raphson

#include<iostream>

#include<cmath>

#include<iomanip>

using namespace std;

double f(double x)

{

double a;

a=(x\*exp(x)-1);

return a;

}

double fprime(double x)

{

double b;

b=(x\*exp(x)+exp(x));

return b;

}

int main()

{

double x0,x1,e,ss;

int i=0;

cout.precision(9);

cout.setf(ios::fixed);

a: cout<<"\nEnter the initial guess of the root\n";

cin>>x1;

cout<<"\nEnter the desired degree of accuracy\n";

cin>>e;

while( fabs(x1-x0)>=e)

{

x0=x1;

x1=x0-f(x0)/fprime(x0);

i++;

cout<<"\nx"<<i<<" = "<<x1<<endl;

}

cout<<"\nThe required root of thr function is:"<<x1<<endl;

cout<<"\n\n To calculate other, press 1 : ";

cin>>ss;

if (ss==1)

goto a;

return 0;

}

// Iterative Method

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

#define f(x) ((cos(x)+3)/2)

#define df(x) ((-sin(x))/2)

int main()

{

cout.precision(6);

cout.setf(ios::fixed);

double x0, x, ep;

cout<<"Enter initial approximation:\n";

cin>>x0;

cout<<"Enter desired accuracy:\n";

cin>>ep;

if (fabs(df(x0))<1)

{

do

{

x = x0;

x0 = f(x);

}

while(fabs(x-x0)>ep);

cout<< endl<<"Root is: "<<x0<<endl;

}

else

{

cout<<"Initial approximation isn't convergent! Please choose another approximation."<<endl;

}

return 0;

// Newton-Raphson for multi-variable

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

#define f(x,y) (pow(x,2)+pow(y,2)-1)

#define g(x,y) (y-pow(x,2))

#define dfx(x,y) (2\*x)

#define dfy(x,y) (2\*y)

#define dgx(x,y) (-2\*x)

#define dgy(x,y) (1)

int main()

{

cout.precision(6);

cout.setf(ios::fixed);

double x, x0, y, y0, fx, fx0, d, h, k, ep, ss;

cout<<"Enter the initial guesses x0 and y0 respectively:\n";

m:

cin>>x0>>y0;

cout<<"Enter the desired accuracy:\n"<<endl;

cin>>ep;

do

{

x=x0,y=y0;

d=dfx(x,y)\*dgy(x,y)-dgx(x,y)\*dfy(x,y);

if (d==0)

{

cout<<"The system doesn't converge for this initial guess. Please, give new guess:"<<endl;

goto m;

}

else

{

h=(g(x,y)\*dfy(x,y)-f(x,y)\*dgy(x,y))/d;

k=(f(x,y)\*dgx(x,y)-g(x,y)\*dfx(x,y))/d;

x0=x+h, y0=y+k;

}

}

while(fabs(x-x0)>ep || fabs(y-y0)>ep);

cout<<"Root of the equation: x= "<<x0<<" y= "<<y0<<endl;

return 0;

}

**Output:**

Enter the initial guesses:

1

Enter the desired accuracy:

0.001

The root of the equation is: 0.567143

**Problem-4, 1(d): Program to solve a system of nonlinear equations (two variables only) with example by Newton-Raphson method.**

**Program:**

//Problem-4, 1(d) Newton Raphson (two variables)

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

#define f(x,y) (pow(x,2)+pow(y,2)-1)

#define g(x,y) (y-pow(x,2))

#define dfx(x,y) (2\*x)

#define dfy(x,y) (2\*y)

#define dgx(x,y) (-2\*x)

#define dgy(x,y) (1)

int main()

{

cout.precision(6);

cout.setf(ios::fixed);

double x, x0, y, y0, fx, fx0, d, h, k, ep;

cout<<"Enter the initial guesses x0 and y0 respectively:\n";

m:

cin>>x0>>y0;

cout<<"Enter the desired accuracy:\n"<<endl;

cin>>ep;

do

{

x=x0,y=y0;

d=dfx(x,y)\*dgy(x,y)-dgx(x,y)\*dfy(x,y);

if (d==0)

{

cout<<"The system doesn't converge for this initial guess. Please, give new guess:"<<endl;

goto m;

}

else

{

h=(g(x,y)\*dfy(x,y)-f(x,y)\*dgy(x,y))/d;

k=(f(x,y)\*dgx(x,y)-g(x,y)\*dfx(x,y))/d;

x0=x+h, y0=y+k;

}

}

while(fabs(x-x0)>ep || fabs(y-y0)>ep);

cout<<"Root of the equation: x= "<<x0<<" y= "<<y0<<endl;

return 0;

}

**// Newton- Backward table**

**#include<iostream>**

**#include<iomanip>**

**#include<cmath>**

**using namespace std;**

**int main()**

**{**

**cout.precision(5);**

**cout.setf(ios::fixed);**

**int i,j,n;**

**cout<<"enter the number of values to be entered\n";**

**cin>>n;**

**double x[n],y[n][n];**

**cout<<"enter the values of x\n";**

**for(i=0;i<n;i++)**

**cin>>x[i];**

**cout<<"enter the values of y\n";**

**for(i=0;i<n;i++)**

**cin>>y[i][0];**

**//calculated the difference table**

**for(j=1;j<n;j++)**

**{**

**for(i=j;i<n;i++)**

**{**

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

**}**

**}**

**//print differece table**

**cout<<"\nthe backward difference table is as follows:\n\n";**

**cout<<"x"<<setw(10)<<"y"<<setw(10);**

**for(i=1;i<n;i++)**

**cout<<"d"<<i<<"y"<<setw(10);**

**cout<<"\n-----------------\n";**

**//k=0;**

**for(i=0;i<n;i++)**

**{**

**cout<<x[i]<<setw(10);**

**// for(j=0;j<k;j++)**

**for(j=0;j<=i;j++)**

**{**

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

**}**

**cout<<"\n";**

**// k++;**

**}**

**//code of interpolation**

**double xn,h,u,sum=y[n-1][0],temp=1;**

**h=x[1]-x[0];**

**cout<<"Enter the value of x at which y to be calculated\n";**

**cin>>xn;**

**u=(xn-x[n-1])/h;**

**for(j=1;j<n;j++)**

**{**

**temp=temp\*(u+j-1)/j;**

**sum=sum+temp\*y[n-1][j];**

**}**

**cout<<"the values of y at x= "<<xn<<" is : "<<sum;**

**double ac;**

**cout<<"\nenter the actual value of y\n";**

**cin>>ac;**

**double p\_e=fabs((ac-sum)/ac)\*100;**

**cout<<"\nthe percentage error is ="<<p\_e<<"%";**

**return 0;**

**}**

**2. forward table**

**#include<iostream>**

**#include<iomanip>**

**#include<cmath>**

**using namespace std;**

**int main()**

**{**

**cout.precision(5);**

**cout.setf(ios::fixed);**

**int i,j,n;**

**cout<<"enter the number of values to be entered\n";**

**cin>>n;**

**double x[n],y[n][n];**

**cout<<"enter the values of x\n";**

**for(i=0;i<n;i++)**

**cin>>x[i];**

**cout<<"enter the values of y\n";**

**for(i=0;i<n;i++)**

**cin>>y[i][0];**

**// Calculate the difference table**

**for(j=1;j<n;j++)**

**{**

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

**{**

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

**}**

**}**

**// Print difference table**

**cout<<"\nthe forward difference table is as follows:\n\n";**

**cout<<"x"<<setw(10)<<"y"<<setw(10);**

**for(i=1;i<n;i++)**

**cout<<"d"<<i<<"y"<<setw(10);**

**cout<<"\n-----------------\n";**

**for(i=0;i<n;i++)**

**{**

**cout<<x[i]<<setw(10);**

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

**{**

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

**}**

**cout<<"\n";**

**}**

**// Interpolation**

**double xn,h,u,sum=y[0][0],temp=1;**

**h=x[1]-x[0];**

**cout<<"Enter the value of x at which y to be calculated\n";**

**cin>>xn;**

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

**for(j=1;j<n;j++)**

**{**

**temp=temp\*(u-j+1)/j;**

**sum=sum+temp\*y[0][j];**

**}**

**cout<<"the values of y at x= "<<xn<<" is : "<<sum;**

**double ac;**

**cout<<"\nenter the actual value of y\n";**

**cin>>ac;**

**double p\_e=fabs((ac-sum)/ac)\*100;**

**cout<<"\nthe percentage error is ="<<p\_e<<"%";**

**return 0;**

**}**

*Program to calculate Interpolation and extrapolation by Newton’s divided difference formula with example.*

***Code:***

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

int main()

{

cout.precision(5);

cout.setf(ios::fixed);

int i, j, k, n;

cout<<"Enter the number of values to be entered\n";

cin>>n;

double x[n], y[n][n];

cout<<"Enter the of values of x\n";

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

cin>>x[i];

cout<<"Enter the of values of y\n";

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

cin>>y[i][0];

//Calculate difference table

for(j=1; j<n; j++)

{

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

{

y[i][j]=(y[i+1][j-1]-y[i][j-1])/(x[i+j]-x[i]);

}

}

//Print difference table

cout<<"\nThe Divided difference table is as follows:\n\n";

cout<<"x"<<setw(20)<<"y"<<setw(20);

for(i=1; i<n; i++)

cout<<"d"<<i<<"y"<<setw(20);

cout<<"\n-------------------------------------------------------\n";

k=n;

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

{

cout<<x[i]<<setw(20);

for(j=0; j<k; j++)

{

cout<<y[i][j]<<setw(20);

}

cout<<"\n";

k--;

}

//Code of interpolation

double xn, sum=y[0][0], temp=1.0;

cout<<"Enter the values of x at which y to be calculated\n";

cin>>xn;

for(j=1; j<n; j++)

{

temp=temp\*(xn-x[j-1]);

sum=sum+temp\*y[0][j];

}

cout<<"The value of y at x="<<xn<<" is: "<<sum<<endl;

return 0;

}

*Program to calculate interpolation and extrapolation by Lagrange’s formula with example.*

***Code:***

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

int main()

{

cout.precision(7);

cout.setf(ios::fixed);

int i, j, k, n;

cout<<"Enter the number of values to be entered\n";

cin>>n;

double x[n], y[n];

cout<<"Enter the of values of x\n";

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

cin>>x[i];

cout<<"Enter the of values of y\n";

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

cin>>y[i];

//code of interpolation starts here

double xn, sum=0, temp;

cout<<"Enter the values of x at which y to be calculated\n";

cin>>xn;

for(j=0; j<n; j++)

{

temp=1;

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

{

if(i!=j)

temp=temp\*(xn-x[i])/(x[j]-x[i]);

}

sum=sum+temp\*y[j];

}

cout<<"The value of y at x="<<xn<<" is: "<<sum<<endl;

return 0;

}

**3(a) Gauss elimination method**

**#include<iostream>**

**#include<iomanip>**

**#include<cmath>**

**using namespace std;**

**int main()**

**{**

**cout.precision(6);**

**cout.setf(ios::fixed);**

**int i, j, n, k;**

**n=3; //set number of equations**

**double a[n][n+1], x[n], temp, t;**

**cout<<"Enter equations in the form ax+by+cz=d:"<<endl;**

**for (i = 0; i < n; i++)**

**{**

**for (j = 0; j <= n; j++)**

**{**

**cout<<"a["<<i<<","<<j<<"]: ";**

**cin>>a[i][j];**

**}**

**}**

**cout<<"\nThe matrix you have entered is \n";**

**for(i=0; i<n; i++)**

**{**

**for(j=0; j<=n; j++)**

**{**

**cout<<a[i][j]<<setw(10);**

**}**

**cout<<"\n";**

**}**

**//Do pivoting**

**for(k=0; k<n-1; k++)**

**{**

**for(i=k+1; i<n; i++)**

**{**

**if(fabs(a[k][k])<fabs(a[i][k]))**

**{**

**for(j=0; j<=n; j++)**

**{**

**temp=a[k][j];**

**a[k][j]= a[i][j];**

**a[i][j]=temp;**

**}**

**}**

**}**

**}**

**cout<<"\nThe matrix after pivoting is \n";**

**for(i=0; i<n; i++)**

**{**

**for(j=0; j<=n; j++)**

**{**

**cout<<a[i][j]<<setw(10);**

**}**

**cout<<"\n";**

**}**

**//Do the elementary row operation**

**for(k=0; k<n-1; k++)**

**{**

**for(i=k+1; i<n; i++)**

**{**

**t=a[i][k]/a[k][k];**

**for(j=0; j<=n; j++)**

**{**

**a[i][j]=a[i][j]-t\*a[k][j];**

**}**

**}**

**}**

**cout<<"\nThe matrix after elementary row operation is \n";**

**for(i=0; i<n; i++)**

**{**

**for(j=0; j<=n; j++)**

**{**

**cout<<a[i][j]<<setw(10);**

**}**

**cout<<"\n";**

**}**

**//Now Let's do the back substitution**

**for(i=n-1; i>=0; i--)**

**{**

**x[i]=a[i][n];**

**for(j=i+1; j<n; j++)**

**{**

**x[i]=x[i]-a[i][j]\*x[j];**

**}**

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

**}**

**cout<<"\nThe values of the variables are \n";**

**for (i = 0; i < n; i++)**

**{**

**cout << "\nx[" << i<<"]= "<<x[i];**

**}**

**return 0;**

**}**

*3(b):*

*Program to solve a system of linear equations (3 variables) with example by Jacobi iterative method.*

***Code:***

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

int main()

{

cout.precision(4);

cout.setf(ios::fixed);

int k,i,j,n;

cout<<"\nEnter the no. of equations\n";

cin>>n; //Input no. of equations

double a[n][n+1]; //declare a 2d array for storing the elements of the augmented matrix

double x[n];

double x1[n]; //declare an array to store the values of variables

double r[n];

double eps,temp;

cout<<"\n Enter the elements of the augmented matrix row-wise:\n";

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

for (j=0;j<=n;j++)

cin>>a[i][j];

cout<<"\n The matrix you have entered is\n";

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

{

for (j=0;j<=n;j++)

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

cout<<"\n";

}

cout<<"\n";

cout<<"\n Enter the initial values of the variables:\n";

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

cin>>x[i];

cout<<"\n Enter the accuracy upto which you want the solution:\n";

cin>>eps;

for (i=0;i<n-1;i++) //Pivotisation(partial) to make the equations diagonally dominant

for (k=i+1;k<n;k++)

if (abs(a[i][i])<abs(a[k][i]))

for (j=0;j<=n;j++)

{

temp=a[i][j];

a[i][j]=a[k][j];

a[k][j]=temp;

}

cout<<"\n The matrix after pivoting\n";

cout<<"\n----------------------------------------------------------------------\n";

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

{

for(j=0;j<=n;j++)

{

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

}

cout<<"\n";

}

cout<<"\n";

cout<<"\n----------------------------------------------------------------------";

do //Perform iterations to calculate x1,x2,...xn

{

for (i=0;i<n;i++) //Loop that calculates x1,x2,...xn

{

x1[i]=x[i];

r[i]=(a[i][n]/a[i][i]);

for (j=0;j<n;j++)

{

if (j!=i)

// continue;

r[i]=r[i]-(a[i][j]/a[i][i]\*x[j]);

}

}

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

{

x[i]=r[i];

}

} while (fabs(x1[0]-x[0])>eps||fabs(x1[1]-x[1])>eps||fabs(x1[2]-x[2])>eps);

cout<<"\n The solution is as follows:\n";

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

cout<<"\n x["<<i<<"]="<<x[i]; //Print the contents of x[]

return 0;

}

*3(c):*

*Program to solve a system of linear equations (3 variables) with example by Gauss-Seidal method.*

***Code:***

//Gaus-seidel (Written by: Manas Sharma - University of Delhi)

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

int main()

{

cout.precision(7);

cout.setf(ios::fixed);

int n,i,j,k,flag=0,count=0;

cout<<"\nEnter the no. of equations\n";

cin>>n; //Input no. of equations

double a[n][n+1]; //declare a 2d array for storing the elements of the augmented matrix

double x[n]; //declare an array to store the values of variables

double eps,y;

cout<<"\nEnter the elements of the augmented matrix row-wise:\n";

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

for (j=0;j<=n;j++)

cin>>a[i][j];

cout<<"\nEnter the initial values of the variables:\n";

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

cin>>x[i];

cout<<"\nEnter the accuracy upto which you want the solution:\n";

cin>>eps;

for (i=0;i<n;i++) //Pivotisation(partial) to make the equations diagonally dominant

for (k=i+1;k<n;k++)

if (abs(a[i][i])<abs(a[k][i]))

for (j=0;j<=n;j++)

{

double temp=a[i][j];

a[i][j]=a[k][j];

a[k][j]=temp;

}

cout<<"Iter"<<setw(15);

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

cout<<"x"<<i<<setw(15);

cout<<"\n----------------------------------------------------------------------";

do //Perform iterations to calculate x1,x2,...xn

{

cout<<"\n"<<count+1<<"."<<setw(15);

for (i=0;i<n;i++) //Loop that calculates x1,x2,...xn

{

y=x[i];

x[i]=a[i][n];

for (j=0;j<n;j++)

{

if (j!=i)

x[i]=x[i]-a[i][j]\*x[j];

}

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

if (abs(x[i]-y)<=eps) //Compare the ne value with the last value

flag++;

cout<<x[i]<<setw(15);

}

cout<<"\n";

count++;

}while(flag<n); //If the values of all the variables don't differ from their previious values with error more than eps then flag must be n and hence stop the loop

cout<<"\n The solution is as follows:\n";

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

cout<<"x"<<i<<" = "<<x[i]<<endl; //Print the contents of x[]

return 0;

}

**//trapezoidal rule**

**#include <iostream>**

**#include <cmath>**

**// Function to calculate the value of the function f(x) = x^2**

**double f(double x) {**

**return (pow(x,2));**

**}**

**// Trapezoidal rule for numerical integration**

**double trapezoidalRule(double a, double b, int n) {**

**double h = (b - a) / n;**

**double result = (f(a) + f(b)) / 2.0;**

**for (int i = 1; i < n; i++) {**

**double x = a + i \* h;**

**result += f(x);**

**}**

**result \*= h;**

**return result;**

**}**

**int main() {**

**double a, b;**

**int n;**

**// Take user input for lower limit, upper limit, and subintervals**

**std::cout << "Enter lower limit (a): ";**

**std::cin >> a;**

**std::cout << "Enter upper limit (b): ";**

**std::cin >> b;**

**std::cout << "Enter the number of subintervals (n): ";**

**std::cin >> n;**

**// Calculate the integral using the Trapezoidal Rule**

**double integral = trapezoidalRule(a, b, n);**

**// The exact value of the integral**

**double exactIntegral = (b \* b \* b / 3.0) - (a \* a \* a / 3.0);**

**// Calculate the percentage error**

**double error = std::abs((integral - exactIntegral) / exactIntegral) \* 100;**

**std::cout << "Estimated Integral: " << integral << std::endl;**

**std::cout << "Exact Integral: " << exactIntegral << std::endl;**

**std::cout << "Percentage Error: " << error << "%" << std::endl;**

**return 0;**

**}**

**//simpson 1/3**

**#include<iostream>**

**#include<iomanip>**

**#include<cmath>**

**using namespace std;**

**//#define f(x) (log(x))**

**//#define f(x) (pow(x,4))**

**#define f(x) (1/(1+x))**

**int main()**

**{**

**cout.precision(6);**

**cout.setf(ios::fixed);**

**int i,n;**

**double a,b,h,sum=0.0,sum1=0.0,integral;**

**cout<<"\nEnter the lower limit of integration\n";**

**cin>>a;**

**cout<<"\nEnter the upper limit of integration\n";**

**cin>>b;**

**m:cout<<"\nEnter the number of division(even number)\n";**

**cin>>n;**

**/\* double x[n+1],y[n+1];**

**h=(b-a)/n;**

**if(n%2==0)**

**{\*/**

**if(n%2==0)**

**{**

**double x[n+1],y[n+1];**

**h=(b-a)/n;**

**for (i=0;i<=n;i++)**

**{**

**x[i]=a+i\*h;**

**y[i]=f(x[i]);**

**}**

**for (i=1;i<=n-1;i=i+2)**

**sum=sum+4\*y[i];**

**for (i=2;i<=n-2;i=i+2)**

**sum1=sum1+2\*y[i];**

**integral=h/3.0\*(y[0]+y[n]+sum+sum1);**

**cout<<"the value of the integral is :"<<integral<<endl;**

**double ac=0.693147180;**

**double p\_e= fabs((ac-integral)/ac)\*100;**

**cout<<"\nThe percentage error= "<<p\_e<<"%"<<endl;**

**}**

**else**

**{**

**goto m;**

**}**

**return 0;**

**}**

**//simpson 3/8**

**#include<iostream>**

**#include<iomanip>**

**#include<cmath>**

**using namespace std;**

**#define f(x) (1/(1+x\*x))**

**//#define f(x) (pow(x,4))**

**int main()**

**{**

**cout.precision(6);**

**cout.setf(ios::fixed);**

**int i,n;**

**double a,b,h,sum=0.0,integral;**

**cout<<"\nEnter the left limit of integration\n";**

**cin>>a;**

**cout<<"\nEnter the right limit of integration\n";**

**cin>>b;**

**m:cout<<"\nEnter the number of division(even number)\n";**

**cin>>n;**

**// if(n%3==0)**

**{**

**double x[n+1],y[n+1];**

**h=(b-a)/n;**

**for (i=0;i<=n;i++)**

**{**

**x[i]=a+i\*h;**

**y[i]=f(x[i]);**

**}**

**for (i=1;i<=n-1;i++)**

**{**

**if(i%3==0)**

**sum=sum+2\*y[i];**

**else**

**sum=sum+3\*y[i];**

**}**

**integral=3\*h/8.00\*(y[0]+y[n]+sum);**

**cout<<"the value of the integral is :"<<integral<<endl;**

**double ac=0.7854;**

**double p\_e= fabs((ac-integral)/ac)\*100;**

**cout<<"\nThe percentage error= "<<p\_e<<"%"<<endl;**

**}**

**// else**

**{**

**goto m;**

**}**

**return 0;**

**}**

**7. curve fitting**

**#include<iostream>**

**#include<iomanip>**

**#include<cmath>**

**using namespace std;**

**int main()**

**{**

**cout.precision(6);**

**cout.setf(ios::fixed);**

**int i,j,k,n,m;**

**cout<<"\nEnter the number of points to be entered\n";**

**cin>>m;**

**double p[m], q[m], temp, t;**

**cout<<"\nEnter the x values\n";**

**for(i=0; i<m; i++)**

**{**

**cin>>p[i];**

**}**

**cout<<"\nEnter the y values\n";**

**for(i=0; i<m; i++)**

**{**

**cin>>q[i];**

**}**

**n=3; //Set degree of polynomial**

**double a[n+1][n+2],x[n+1];**

**for(i=0; i<=n; i++)**

**{**

**//Calculate different element of the augmented matrix in a row**

**for(j=0; j<=n; j++)**

**{**

**a[i][j]=0;**

**for(k=0;k<m;k++)**

**{**

**a[i][j]=a[i][j]+pow(p[k],(i+j));**

**}**

**}**

**//Calculate last element of the augmented matrix in a row**

**a[i][n+1]=0;**

**for(k=0;k<m;k++)**

**{**

**a[i][n+1]=a[i][n+1]+pow(p[k],i)\*q[k];**

**}**

**}**

**n=n+1;**

**//Do pivoting**

**for(k=0; k<n-1; k++)**

**{**

**for(i=k+1; i<n; i++)**

**{**

**if(fabs(a[k][k])<fabs(a[i][k]))**

**{**

**for(j=0; j<=n; j++)**

**{**

**temp=a[k][j];**

**a[k][j]= a[i][j];**

**a[i][j]=temp;**

**}**

**}**

**}**

**}**

**//Do the elementary row operation**

**for(k=0; k<n-1; k++)**

**{**

**for(i=k+1; i<n; i++)**

**{**

**t=a[i][k]/a[k][k];**

**for(j=0; j<=n; j++)**

**{**

**a[i][j]=a[i][j]-t\*a[k][j];**

**}**

**}**

**}**

**//Back substitution**

**for(i=n-1; i>=0; i--)**

**{**

**x[i]=a[i][n];**

**for(j=i+1; j<n; j++)**

**{**

**if(j!=i)**

**x[i]=x[i]-a[i][j]\*x[j];**

**}**

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

**}**

**cout<<"\nThe values of the coefficients are\n";**

**for(i=0; i<n; i++)**

**{**

**cout<<x[i]<<endl;**

**}**

**cout<<"\nThe required polynomial is\n";**

**cout<<"y="<<x[0]<<"+"<<x[1]<<"x+"<<x[2]<<"x^2+"<<x[3]<<"x^3"<<endl;**

**double xm, xn;**

**cout<<"\nEnter the value of x at which you want to check the value: "; cin>>xm;**

**xn= x[0] + x[1]\*xm + x[2]\*xm\*xm + x[3]\*xm\*xm\*xm;**

**cout<<"\nValue of y at x= "<<xm<<" is: "<<xn<<endl;**

**return 0;**

**}**

**Extra**

***Problem – 15:***

*5(a):*

*Numerical solution of first order DE of the type by Euler formula at a point and for particular range with example.*

***Code:***

//Numerical solution of first order ODE using Euler Method

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

double df(double x, double y)

{

double a=y+pow(x,4);

return a;

}

int main()

{

int n,i;

double x0,y0,h,X;

cout.precision(5);

cout.setf(ios::fixed);

cout<<"\nEnter the initial condition\n";

cout<<"\nEnter the initial value of x\n";

cin>>x0;

cout<<"\nEnter the initial value of y\n";

cin>>y0;

cout<<"\nFor what value of x do you want to find the value of y\n";

cin>>X;

cout<<"\nEnter the number of subdivisions\n";

cout<<"\nPlease enter a large value of \n";

cin>>n;

h=(X-x0)/n;

double x[n],y[n];

x[0]=x0; y[0]=y0;

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

{

y[i+1]=y[i]+h\*df(x[i],y[i]);

x[i+1]=x[i]+h;

}

cout<<"\n\nThe approximate value of y at x="<<x[n]<<" is "<<y[n]<<endl;

#define f(x) (exp(x)-pow(x,4)-4\*pow(x,3)-12\*pow(x,2)-24\*x-24)

double per\_err;

per\_err=(abs(f(x[n])-y[n]))/(f(x[n]))\*100;

cout<<"Percent Error = "<<per\_err<<" % "<<endl;

return 0;

}

***Output:***

Enter the initial condition

Enter the initial value of x

0

Enter the initial value of y

-23

For what value of x do you want to find the value of y

1

Enter the number of subdivisions

Please enter a large value of

5

The approximate value of y at x=1.00000 is -57.11041

The approximate value of y at x=1.0000000 is=-57.9963837

Percent Error = 8.3435646 %

--------------------------------

Process exited after 4.563 seconds with return value 0

Press any key to continue . . .

***Problem – 16:***

*5(b):*

*Numerical solution of first order DE of the type by Modified Euler formula at a point and for particular range with example.*

***Code:***

//Modified Euler's Method for differential equations

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

double df(double x, double y)

{

double a=y+pow(x,4);

return a;

}

int main()

{

int n,i;

double x0,y0,h,X;

cout.precision(7);

cout.setf(ios::fixed);

cout<<"\nEnter the initial condition\n";

cout<<"\nEnter the initial value of x\n";

cin>>x0;

cout<<"\nEnter the initial value of y\n";

cin>>y0;

cout<<"\nFor what value of x do you want to find the value of y\n";

cin>>X;

cout<<"\nEnter the number of subdivisions\n";

cout<<"\nPlease enter a large value of \n";

cin>>n;

h=(X-x0)/n;

double x[n],y1[n],y[n];

x[0]=x0; y[0]=y0,y1[0]=0;

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

{

y[i+1]=y[i]+h\*df(x[i],y[i]);

x[i+1]=x[i]+h;

y1[i+1]=y[i]+h/2.0\*(df(x[i],y[i])+df(x[i+1],y[i+1]));

}

cout<<"\n\nThe approximate value of y at x="<<x[n]<<" is="<<y1[n]<<endl;

#define f(x) (exp(x)-pow(x,4)-4\*pow(x,3)-12\*pow(x,2)-24\*x-24)

double per\_err;

per\_err=(abs(f(x[n])-y[n]))/(f(x[n]))\*100;

cout<<"Percent Error = "<<per\_err<<" % "<<endl;

return 0;

}

***Output:***

Enter the initial condition

Enter the initial value of x

0

Enter the initial value of y

-23

For what value of x do you want to find the value of y

1

Enter the number of subdivisions

Please enter a large value of

5

The approximate value of y at x=1.0000000 is=-57.9963837

Percent Error = -6.8876436 %

--------------------------------

Process exited after 5.131 seconds with return value 0

Press any key to continue . . .

***Problem – 17:***

*5(c):*

*Numerical solution of first order DE of the type by Runge-Kutta 4th order formula at a point and for particular range with example.*

***Code:***

//RUNGE-KUTTA 4th Order Method for solving ODE

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

double df(double x, double y)

{

double a=x\*y+y\*y;

return a;

}

int main()

{

int n,i;

double x0,y0,h,X,k1,k2,k3,k4;

cout.precision(7);

cout.setf(ios::fixed);

cout<<"\nEnter the initial condition\n";

cout<<"\nEnter the initial value of x\n";

cin>>x0;

cout<<"\nEnter the initial value of y\n";

cin>>y0;

cout<<"\nFor what value of x do you want to find the value of y\n";

cin>>X;

cout<<"\nEnter the number of subdivisions\n";

//cout<<"\nPlease enter a large value of \n";

cin>>n;

h=(X-x0)/n;

double x[n],y[n];

x[0]=x0; y[0]=y0;

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

{

k1=h\*df(x[i],y[i]);

k2=h\*df(x[i]+h/2.0, y[i]+k1/2.0);

k3=h\*df(x[i]+h/2.0, y[i]+k2/2.0);

k4=h\*df(x[i]+h, y[i]+k3);

y[i+1]=y[i]+1/6.0\*(k1+2\*k2+2\*k3+k4);

x[i+1]=x[i]+h;

}

cout<<"\n\nThe approximate value of y at x="<<x[n]<<" is="<<y[n]<<endl;

return 0;

}

***Output:***

Enter the initial condition

Enter the initial value of x

0

Enter the initial value of y

1

For what value of x do you want to find the value of y

2

Enter the number of subdivisions

4

The approximate value of y at x=2.0000000 is=1.5209311

--------------------------------

Process exited after 7.611 seconds with return value 0

Press any key to continue . .

//Problem-18, 5(d) Runge-Kutta 4th order formula for system of 1st order DE

#include<iostream>

#include<cmath>

#include<iomanip>

using namespace std;

#define dy\_dx(x,y,z) (y\*z+x)

#define dz\_dx(x,y,z) (x\*z+y)

int main()

{

cout.precision(6);

cout.setf(ios::fixed);

int i,n;

double x0,y0,z0,h,xn;

cout<<"Enter the initial value of x:\n";

cin>>x0;

cout<<"Enter the initial value of y corresponding to x:\n";

cin>>y0;

cout<<"Enter the initial value of z corresponding to x:\n";

cin>>z0;

cout<<"Enter the value of x up to which you want to find the value of y & z:\n";

cin>>xn;

cout<<"Enter the value of h\n";

cin>>h;

n=(xn-x0)/h;

double x[n+1],y[n+1],z[n+1],k1,k2,k3,k4,l1,l2,l3,l4;

x[0]=x0, y[0]=y0, z[0]=z0;

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

{

k1=h\*dy\_dx(x[i],y[i],z[i]);

l1=h\*dz\_dx(x[i],y[i],z[i]);

k2=h\*dy\_dx((x[i]+h/2.0),(y[i]+k1/2.0),(z[i]+l1/2.0));

l2=h\*dz\_dx((x[i]+h/2.0),(y[i]+k1/2.0),(z[i]+l1/2.0));

k3=h\*dy\_dx((x[i]+h/2.0),(y[i]+k2/2.0),(z[i]+l2/2.0));

l3=h\*dz\_dx((x[i]+h/2.0),(y[i]+k2/2.0),(z[i]+l2/2.0));

k4=h\*dy\_dx((x[i]+h),(y[i]+k3),(z[i]+l3));

l4=h\*dz\_dx((x[i]+h),(y[i]+k3),(z[i]+l3));

y[i+1]=y[i]+(1/6.0)\*(k1+2\*k2+2\*k3+k4);

z[i+1]=z[i]+(1/6.0)\*(l1+2\*l2+2\*l3+l4);

x[i+1]=x[i]+h;

}

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

{

cout<<"y("<<x[i]<<"): "<<y[i]<<endl;

cout<<"z("<<x[i]<<"): "<<z[i]<<endl;

}

return 0;

}

**Output:**

Enter the initial value of x:

0

Enter the initial value of y corresponding to x:

1

Enter the initial value of z corresponding to x:

-1

Enter the value of x up to which you want to find the value of y & z:

0.1

Enter the value of h

0.1

y(0.000000): 1.000000

z(0.000000): -1.000000

y(0.100000): 0.913936

z(0.100000): -0.909218