Department of Computer Science and Engineering

.

Course Code: CSE- 4746

Course Title: Numerical Methods Lab

Submitted By:

Name: BIJOY KUMAR DAS JOY

Semester: 7Th

Matric ID: C201016

Section: 7AM

Date of Submission: 6 october,2023.

Submitted To:

Prof. Mohammed Shamsul Alam

Professor

Dept. Of CSE, IIUC

- **The following values of f (x) are given.**

x = 1 2 3 4 5

y = f(x) 1 8 27 64 125

Write a program to find the first derivative and the second derivative of the function tabulated above at the point x = 1.

Source Code:

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


int main()
{
    float x[5]={1,2,3,4,5};
    float y[5][5];

y[0][0]=1;
y[1][0]=8;
y[2][0]=27;
y[3][0]=64;
y[4][0]=125;

float x0=1;

float h=1;
float u=(x0-x[0])/h;



for (int i = 1; i < 5; i++) {
        for (int j = 0; j < 5 - i; j++)
            y[j][i] = y[j + 1][i - 1] - y[j][i - 1];
    }


    for (int i = 0; i < 5; i++) {
        cout << x[i]
```

```
            << "\t";
        for (int j = 0; j < 5 - i; j++)
            cout << setw(4) << y[i][j] << "\t";

        cout << endl;
    }

//first derivation

float f_d=y[1][0];
f_d+=y[2][0]*((2*u-1)/2.0);
f_d += y[3][0]*((3*u*u-6*u+2)/6.0);
f_d += y[4][0]*((4*u*u*u*u-18*u*u+22*u-6)/24.0);
f_d /= h;

//second derivation

double s_d = y[2][0];
    s_d += y[3][0] * (u - 1);
    s_d += y[4][0] * ((6 * u * u - 18 * u + 11) / 12.0);
    s_d /= (h * h);


    cout << "First Derivative: " << f_d << endl;
    cout << "Second Derivative: " << s_d<< endl;


}
```

- **The following values of f (x) are given.**

  X= 1 2 3 4 5

  y = f(x) 1 8 27 64 125

Write a program to find the first derivative and the second derivative of the function tabulated above at the point x = 1.5.

Source Code:

```
 #include<bits/stdc++.h>

using namespace std;


int main()
{
    float x[5]={1,2,3,4,5};
    float y[5][5];

y[0][0]=1;
y[1][0]=8;
y[2][0]=27;
y[3][0]=64;
y[4][0]=125;

float x0=1.5;

float h=1;
float u=(x0-x[0])/h;



for (int i = 1; i < 5; i++) {
        for (int j = 0; j < 5 - i; j++)
            y[j][i] = y[j + 1][i - 1] - y[j][i - 1];
    }


for (int i = 0; i < 5; i++) {
        cout << x[i]
            << "\t";
        for (int j = 0; j < 5 - i; j++)
            cout << setw(4) << y[i][j] << "\t";

        cout << endl;
    }
//first derivation

float f_d=y[1][0];
```

```
f_d+=y[2][0]*((2*u-1)/2.0);
f_d += y[3][0]*((3*u*u-6*u+2)/6.0);
f_d += y[4][0]*((4*u*u*u*u-18*u*u+22*u-6)/24.0);
f_d /= h;

//second derivation

double s_d = y[2][0];
    s_d += y[3][0] * (u - 1);
    s_d += y[4][0] * ((6 * u * u - 18 * u + 11) / 12.0);
    s_d /= (h * h);


    cout << "First Derivative: " << f_d << endl;
    cout << "Second Derivative: " << s_d<< endl;


}
```

- **Write a program to calculate the approximate area under the curve y = $\int^{5}$ log 10x dx**
  by using trapezoidal rule.

  <u>Source code</u>:

```
#include <bits/stdc++.h>

using namespace std;

    double f(double x) {
        return  log10(x);
    }

double trapezoidal(double a, double b, int n) {


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

```
    double sum =0.5* (f(a) + f(b));
    for (int i = 1; i < n; i++) {
    sum += f(a + i * h);
    }
  return sum * h;
}



int main() {

double a = 1;
double b = 5;
int n = 10;


double area = trapezoidal(a, b, n);


cout << "The approximate area under the curve is: " << area << endl;

return 0;
}
```

- **Write a program to calculate the approximate area under the curve y = $\int_{pi/2} e^{sinx}$ dx by using Simpson's 1/3 rule.**
  Source Code:

```
#include<bits/stdc++.h>

using namespace std;
double f(double x)
{
    return exp(sin(x));
}
```

```
double simponse_1_3(double a, double b, int n)
{
    double h = (b - a) / n;
    double sum = f(a) + f(b);
    for (int i = 1; i < n; i++)
{
    if (i % 2 == 0) {
      sum += 2 * f(a + i * h);
    } else {
      sum += 4 * f(a + i * h);
    }
}
  return sum * h / 6;
}
int main()
{
    double a=0;
    double b=90;
    int n=10;
    double area= simponse_1_3(a,b,n);

    cout << "The approximate area under the curve is: " << area << endl;
}
```

- **Write a program to calculate the approximate area under the curve y = $\int^2 x / (1+x^2)$ by using Simpson's 3/8 rule.**
  <u>Source Code:</u>

```
#include<bits/stdc++.h>
using namespace std;
double f(double x){
    return x / (1 + x * x);
}
double Simpson3_8(double a, double b, int n)
{
    double h = (b - a) / n;
    double sum = f(a) + f(b);
    for (int i = 1; i < n; i += 3) {
```

```
      sum += 3 * f(a + i * h);
    }
    for (int i = 2; i < n; i += 3) {
      sum += 2 * f(a + i * h);
    }
  return sum * h * 3 / 8;
}
int main()
{
    double a=1;
    double b=2;
    int n=10;
    double area = Simpson3_8(a,b,n);

    cout << "The approximate area under the curve is: " << area << endl;

}
```

- **Write a program to find the determinant of a 3X3 matrix.**

  Source Code:

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

int main()
{
    double matrix[3][3];
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<3;j++)
        {
            cin>>matrix[i][j];
        }
    }
```

```
    double d= matrix[0][0] * (matrix[1][1] * matrix[2][2] - matrix[1][2] * matrix[2][1])
                    - matrix[0][1] * (matrix[1][0] * matrix[2][2] - matrix[1][2] * matrix[2][0])
                    + matrix[0][2] * (matrix[1][0] * matrix[2][1] - matrix[1][1] * matrix[2][0]);
        cout<<"Answer:"<<d<<endl;
}
```

- **Write a program to solve the following system of linear equations by using Matrix inversion method.**

  x + y + z = 1
  x + 2y + 3z = 6
  x + 3y + 4z = 6

  Source Code:

```
#include<bits/stdc++.h>

using namespace std;
double a[3][3]={
        {1,1,1},
        {1,2,3},
        {1,3,4}
    };
    double b[3]={1,6,6};
    double x[3]={0};
    double a_inv[3][3];
double determinant()
{
    return a[0][0]*(a[1][1]*a[2][2]-a[1][2]*a[2][1])-
    a[0][1]*(a[1][0]*a[2][2]-a[1][2]*a[2][0])+
    a[0][2]*(a[1][0]*a[2][1]-a[1][1]*a[2][0]);
}
double inversematrix()
{
    double d = determinant();
    if(d==0)
    cout<<"it is not possible "<<endl;

    //inverse function value
    a_inv[0][0] = (a[1][1] * a[2][2] - a[1][2] * a[2][1]) / d;
    a_inv[0][1] = (a[0][2] * a[2][1] - a[0][1] * a[2][2]) / d;
```

```
    a_inv[0][2] = (a[0][1] * a[1][2] - a[0][2] * a[1][1]) / d;
    a_inv[1][0] = (a[1][2] * a[2][0] - a[1][0] * a[2][2]) / d;
    a_inv[1][1] = (a[0][0] * a[2][2] - a[0][2] * a[2][0]) / d;
    a_inv[1][2] = (a[0][1] * a[1][0] - a[0][0] * a[1][1]) / d;
    a_inv[2][0] = (a[1][0] * a[2][1] - a[1][1] * a[2][0]) / d;
    a_inv[2][1] = (a[0][1] * a[2][0] - a[0][0] * a[2][1]) / d;
    a_inv[2][2] = (a[0][0] * a[1][1] - a[0][1] * a[1][0]) / d;

}
int main()
{

    inversematrix();

    for(int i=0;i<3;i++)
    {
        for(int j=0;j<3;j++)
        {
            x[i]+=a_inv[i][j]*b[j];
        }
    }
    //solution

    cout << "Solution:" << endl;
    cout << "x = " << x[0] << endl;
    cout << "y = " << x[1] << endl;
    cout << "z = " << x[2] << endl;


}
```

- **Write a program to solve the following system of linear equations by using Cramer's Rule:**

  27x + 6y – z = 85

  6x + 15y + 2z = 72

  x + y + 54z = 110

Source Code:

```
#include<bits/stdc++.h>

using namespace std;

double determinant(double a[3][3])
{
    return a[0][0]*(a[1][1]*a[2][2]-a[1][2]*a[2][1])-
    a[0][1]*(a[1][0]*a[2][2]-a[1][2]*a[2][0])+
    a[0][2]*(a[1][0]*a[2][1]-a[1][1]*a[2][0]);
}

int main()
{
    double a[3][3]={
        {27,6,-1},
        {6,15,2},
        {1,1,54}
    };
    double b[3]={85,72,110};
    double x[3]={0};

    double d= determinant(a);

    if(d==0)
    {
        cout<<"it is not possible "<<endl;
        return 1;
    }

    double ax[3][3]={
        {b[0],6,-1},
```

```
        {b[1],15,2},
        {b[2],1,54}
    };
    double ay[3][3]={
        {27,b[0],-1},
        {6,b[1],2},
        {1,b[2],54}
    };
    double az[3][3]={
        {27,6,b[0]},
        {6,15,b[1]},
        {1,1,b[2]}
    };



    x[0]= determinant(ax)/d;
    x[1]= determinant(ay)/d;
    x[2]= determinant(az)/d;



    cout << "Solution:" << endl;
    cout << "x = " << x[0] << endl;
    cout << "y = " << x[1] << endl;
    cout << "z = " << x[2] << endl;

}
```

- **Write a program to solve the following system of linear equations by using Jacobi's method.**

  83x + 11y - 4z = 95

  3x + 8y + 29z = 71

  7x + 52y + 13z = 104

Source Code:

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

int main()
{
    double x =0.0;
    double y=0.0;
    double z=0.0;


    double e =0.0001;
    int i=1;
    while(i)
    {
        double x_n=(95.0 - 11.0 * y + 4.0 * z) / 83.0;
        double y_n=(71.0 - 3.0 * x - 29.0 * z) / 8.0;
        double z_n= (104.0 - 7.0 * x - 52.0 * y) / 13.0;
        double x_d= fabs(x_n-x);
        double y_d= fabs(y_n-y);
        double z_d= fabs(z_n-z);

        x=x_n;
        y=y_n;
        z=z_n;
        i++;
        if(x_d<e && y_d<e && z_d<e)
        {
            break;
        }
        else
        cout<<"iteration: "<<i<<" : no result"<<endl;


    }
    cout << "Solution:" << endl;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "z = " << z << endl;
```

```
}
```

- **Write a program to solve the following system of linear equations by using Gauss-Seidel method.**

  $$10x_1 + x_2 + x_3 = 12$$
  $$2x_1 + 10x_2 + x_3 = 13$$
  $$2x_1 + 2x_2 + 10x_3 = 14$$

  Source Code:

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

int main()
{
    double x =0.0;
    double y=0.0;
    double z=0.0;


    double e =0.0001;
    int i=1;
    while(i)
    {
        double x_n=(12.0 - y - z) / 10.0;
        double y_n=(13.0 - 2.0 * x_n - z) / 10.0;
        double z_n= (14.0 - 2.0 * x_n - 2.0 * y_n) / 10.0;
        double x_d= fabs(x_n-x);
        double y_d= fabs(y_n-y);
        double z_d= fabs(z_n-z);

        x=x_n;
        y=y_n;
        z=z_n;
        i++;
        if(x_d<e && y_d<e && z_d<e)
        {
            break;
```

```
        }
        else
        cout<<"iteration: "<<i<<" : no result"<<endl;


    }
    cout << "Solution:" << endl;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "z = " << z << endl;
}
```