# International Islamic University Chittagong (IIUC)

# Department of Computer Science Engineering (CSE)

## LAB - 3

| | |
|---|---|
| **Course title** | **: Numerical Methods Lab** |
| **Course code** | **: CSE-4746** |
| **Session** | **: Spring-2024** |

**Submitted To:**

Mohammed Shamsul Alam

Professor, Dept. of CSE

International Islamic University Chittagong

**Submitted By:**

| | |
|---|---|
| **Name** | : Afif Hossain Irfan |
| **Matric ID** | : C211005 |
| **Semester** | : 7th |
| **Section** | : 7AM |
| **Mobile no.** | : 01521536082 |

# Numerical Differentiation - 1

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

int main()
{
    ///Peace be with you.


    vector<double> x = {1, 2, 3, 4, 5};
    vector<double> y = {1, 8, 27, 64, 125};
    double GivenX = 1;
    int n = y.size();

    vector<vector<double>> table(n, vector<double>(n));

    double h = x[1] - x[0];
    double u = (GivenX - x[0])/h;

    for (int i = 0; i < n; ++i)
    {
        table[i][0] = y[i];
    }

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

    cout << "Difference Table:" << endl;
    cout << "0Y0" << "\t";
    for (int i = 1; i < n; i++)
    {
        cout << i << "Y0" << "\t";
    }
    cout << endl;

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - i; j++)
        {
            //cout << i << j << " = " ;
            cout << table[i][j] << "\t";
        }
        cout << endl;
    }



    double first_derivative = (1.00/h) * ( table[0][1] + (2*u-1)/2 * table[0][2] +
(3*u*u-6*u+2)/(2*3) * table[0][3]);
    double second_derivative = (1.00/h*h) * ( table[0][2] + (u-1) * table[0][3] +
(6*u*u-18*u+11)/12 * table[0][4]);


    cout << endl;
    cout << "First Derivative: " << first_derivative << endl;
    cout << "Second Derivative: " << second_derivative << endl;

    return 0;
}
```

# Numerical Differentiation - 2

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

int main()
{
    ///Peace be with you.


    vector<double> x = {1, 2, 3, 4, 5};
    vector<double> y = {1, 8, 27, 64, 125};
    double GivenX = 1.5;
    int n = y.size();

    vector<vector<double>> table(n, vector<double>(n));

    double h = x[1] - x[0];
    double u = (GivenX - x[0])/h;

    for (int i = 0; i < n; ++i)
    {
        table[i][0] = y[i];
    }

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

    cout << "Difference Table:" << endl;
    cout << "0Y0" << "\t";
    for (int i = 1; i < n; i++)
    {
        cout << i << "Y0" << "\t";
    }
    cout << endl;

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - i; j++)
        {
            //cout << i << j << " = " ;
            cout << table[i][j] << "\t";
        }
        cout << endl;
    }



    double first_derivative = (1.00/h) * ( table[0][1] + (2*u-1)/2 * table[0][2] +
(3*u*u-6*u+2)/(2*3) * table[0][3]);
    double second_derivative = (1.00/h*h) * ( table[0][2] + (u-1) * table[0][3] +
(6*u*u-18*u+11)/12 * table[0][4]);


    cout << endl;
    cout << "First Derivative: " << first_derivative << endl;
    cout << "Second Derivative: " << second_derivative << endl;

    return 0;
}
```

# Trapezoidal.

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

#define f(x) log10(x)

int main()
{
    double a = 1, b = 5;
    double n = 8;
    double h = (b - a)/n;

    vector<double> y;

    for(double i = a; i <= b; i = i+h)
    {
        double ans = f(i);
        y.push_back(ans);
    }

    double sum1 = y[0] + y[n];
    double sum2 = 0.0;

    for(int i = 1; i < n; i++)
    {
        sum2 = sum2 + (2 * y[i]);
    }

    double area = (h/2) * (sum1 + sum2);

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

    return 0;
}
```

# Simpson’s 1/3.

```
// Simpson’s 1/3
#include<bits/stdc++.h>
using namespace std;

#define f(x) exp(sin(x))
#define PI 3.1416

int main()
{
    double a = 0, b = PI/2;
    double n = 6;
    double h = (b-a)/n;

    vector<double> y;

    for(double i = a; i <= b; i = i+h)
    {
        double ans = f(i);
        y.push_back(ans);
    }

    double sum1 = y[0] + y[n];
    double sum2 = 0.0;
    double sum3 = 0.0;

    for(int i = 1; i < n; i = i+2)
    {
        sum2 = sum2 + (4 * y[i]);
    }

    for(int i = 2; i < n; i = i+2)
    {
        sum3 = sum3 + (2 * y[i]);
    }

    double area = (h/3) * (sum1 + sum2 + sum3);

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

    return 0;
}
```

# Simpson’s 3/8.

```
// Simpson’s 3/8
#include<bits/stdc++.h>
using namespace std;

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

int main()
{
    double a = 0, b = 1;
    double n = 6;
    double h = (b-a)/n;

    vector<double> y;

    for(double i = a; i <= b; i = i+h)
    {
        double ans = f(i);
        y.push_back(ans);
    }

    double sum1 = y[0] + y[n];
    double sum2 = 0.0;
    double sum3 = 0.0;

    for(int i = 1; i < n; i++)
    {
        if(i%3 != 0)
        {
            sum2 = sum2 + (3 * y[i]);
        }
        else
        {
            sum3 = sum3 + (2 * y[i]);
        }
    }

    double area = ((3*h)/8) * (sum1 + sum2 + sum3);

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

    return 0;
}
```

# Determinant.

```
// Determinant
#include <iostream>
using namespace std;


int main()
{
    double matrix[3][3];

    cout << "Enter the elements of the 3x3 matrix:" << endl;
    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            cin >> matrix[i][j];
        }
    }

    double det = 0.0;

    det = 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 << "Determinant of the matrix is: " << det << endl;

    return 0;
}
```

# Matrix Inversion.

```
// Matrix Inversion
#include <iostream>
using namespace std;

int main()
{
    double a[4][4], b[4][1], x[4][1];

    cout << "Enter the elements of the A matrix:" << endl;
    for (int i = 1; i <= 3; i++)
    {
        for (int j = 1; j <= 3; j++)
        {
            cin >> a[i][j];
        }
    }

    cout << "Enter the elements of the B matrix:" << endl;
    for (int i = 1; i <= 3; i++)
    {
        cin >> b[i][1];
    }

    double det_a = 0.0;

    det_a = a[1][1] * (a[2][2] * a[3][3] - a[2][3] * a[3][2])
            - a[1][2] * (a[2][1] * a[3][3] - a[2][3] * a[3][1])
            + a[1][3] * (a[2][1] * a[3][2] - a[2][2] * a[3][1]);

    cout << "Determinant of the matrix is: " << det_a << endl;

    double d[4][4]; // Cofactor
    d[1][1] = +(a[2][2] * a[3][3] - a[2][3] * a[3][2]);
    d[1][2] = -(a[2][1] * a[3][3] - a[2][3] * a[3][1]);
    d[1][3] = +(a[2][1] * a[3][2] - a[2][2] * a[3][1]);
    d[2][1] = -(a[1][2] * a[3][3] - a[1][3] * a[3][2]);
    d[2][2] = +(a[1][1] * a[3][3] - a[1][3] * a[3][1]);
    d[2][3] = -(a[1][1] * a[3][2] - a[1][2] * a[3][1]);
    d[3][1] = +(a[1][2] * a[2][3] - a[1][3] * a[2][2]);
    d[3][2] = -(a[1][1] * a[2][3] - a[1][3] * a[2][1]);
    d[3][3] = +(a[1][1] * a[2][2] - a[1][2] * a[2][1]);

    double adj_a[4][4]; // adjoint matrix
    adj_a[1][1] = d[1][1];
    adj_a[1][2] = d[2][1];
    adj_a[1][3] = d[3][1];
    adj_a[2][1] = d[1][2];
    adj_a[2][2] = d[2][2];
    adj_a[2][3] = d[3][2];
    adj_a[3][1] = d[1][3];
    adj_a[3][2] = d[2][3];
    adj_a[3][3] = d[3][3];

    double a_Inv[4][4]; // Inverse matrix
    for (int i = 1; i <= 3; i++)
    {
        for (int j = 1; j <= 3; j++)
        {
            a_Inv[i][j] = adj_a[i][j] / det_a;
        }
    }
    for (int i = 1; i <= 3; i++)
    {
        for (int j = 1; j <= 1; j++)
        {
            for (int k = 1; k <= 3; k++)
            {
                x[i][j] += a_Inv[i][k] * b[k][j];
            }
        }
    }

    cout << "Solution:" << endl;
    for(int i=1; i<=3; i++)
    {
        cout << "x[" << i << "] = " << x[i][1] << endl;
    }

    return 0;
}
```

# Cramer’s Rule.

```
// Cramer’s Rule
#include <iostream>
using namespace std;

const int MAX_SIZE = 100;

double determinant(double mat[MAX_SIZE][MAX_SIZE], int n)
{
    double det = 0;

    det = mat[1][1] * (mat[2][2] * mat[3][3] - mat[2][3] * mat[3][2])
          - mat[1][2] * (mat[2][1] * mat[3][3] - mat[2][3] * mat[3][1])
          + mat[1][3] * (mat[2][1] * mat[3][2] - mat[2][2] * mat[3][1]);

    return det;
}

double Cramer_Determinant(int row, double A[MAX_SIZE][MAX_SIZE], double B[MAX_SIZE][1],
int n)
{
    double original_A[n][n];

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            original_A[i][j] = A[i][j];
        }
    }

    // Replace the specified row of A with B
    for (int i = 1; i <= n; i++)
    {
        A[i][row] = B[i][1];
    }

    // Calculate the determinant of the modified A matrix
    double det = determinant(A, n);


    // Restore the original A matrix
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            A[i][j] = original_A[i][j];
        }
    }

    return det;
}

int main()
{
    int n;
    cout << "Order of the matrix : ";
    cin >> n;
    double a[MAX_SIZE][MAX_SIZE], b[MAX_SIZE][1], x[MAX_SIZE][1];

    cout << "Enter the elements of the A matrix:" << endl;
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            cin >> a[i][j];
        }
    }

    cout << "Enter the elements of the B matrix:" << endl;
    for (int i = 1; i <= n; i++)
    {
        cin >> b[i][1];
    }

    //cout << "Determinant of original A matrix: " << determinant(a, n) << endl;

    for (int row = 1; row <= n; row++)
    {
        double ans = Cramer_Determinant(row, a, b, n)/determinant(a, n);
        cout << "x[" << row << "]: " << ans << endl;
    }

    return 0;
}
```

# Jacobi,s Method.

```
// Jacobi,s Method
#include <iostream>
using namespace std;

int main()
{
    double a[5][5];
    double b[5];
    double x, y, z;

    cout << "Enter the elements of the A matrix:" << endl;
    for (int i = 1; i <= 3; i++)
    {
        for (int j = 1; j <= 3; j++)
        {
            cin >> a[i][j];
        }
    }

    cout << "Enter the elements of the B matrix:" << endl;
    for (int i = 1; i <= 3; i++)
    {
        cin >> b[i];
    }

    x = 0;
    y = 0;
    z = 0;

    // Jacobi method
    double x1, y1, z1;
    do
    {
        x1 = x;
        y1 = y;
        z1 = z;

        x = (b[1] - a[1][3] * z - a[1][2] * y) / a[1][1];
        y = (b[2] - a[2][3] * z - a[2][1] * x) / a[2][2];
        z = (b[3] - a[3][1] * x - a[3][2] * y) / a[3][3];
    }
    while (abs(x1 - x) > 0.001 || abs(y1 - y) > 0.001 || abs(z1 - z) > 0.001);


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

    return 0;
}
```

# Gauss-Seidel Method.

```cpp
// Gauss-Seidel Method
#include <iostream>
using namespace std;

int main()
{
    float a[10][10], b[10], x[10], y[10];
    int n = 0, m = 0, i = 0, j = 0;

    cout << "Enter the order of the matrix: ";
    cin >> n;


    cout << "Enter the elements of the A matrix:" << endl;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            cin >> a[i][j];
        }
    }


    cout << "Enter the elements of the B matrix:" << endl;
    for (i = 0; i < n; i++)
    {
        cin >> b[i];
    }


    cout << "\nEnter the number of iterations: ";
    cin >> m;


    while (m > 0)
    {
        for (i = 0; i < n; i++)
        {
            y[i] = (b[i] / a[i][i]);
            for (j = 0; j < n; j++)
            {
                if (j == i)
                {
                    continue;
                }
                y[i] = y[i] - ((a[i][j] / a[i][i]) * x[j]);
                x[i] = y[i];
            }
            cout << "x[" << i << "] = " << y[i] << "    ";
        }
        cout << "\n";
        m--;
    }


    return 0;
}
```