**International Islamic University Chittagong (IIUC)**

**Department of Computer Science Engineering (CSE)**

**LAB – 4**

**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**

**Least Square Line**

#include <bits/stdc++.h>

using namespace std;

int main()

{

///Peace be with you.

vector<int> x = {-2, -1, 0, 1, 2};

vector<int> y = {1, 2, 3, 4, 5};

int n = y.size();

double sum\_x = 0, sum\_y = 0, sum\_xy = 0, sum\_xx = 0;

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

{

sum\_x = sum\_x + x[i];

sum\_y = sum\_y + y[i];

sum\_xy = sum\_xy + (x[i] \* y[i]);

sum\_xx = sum\_xx + (x[i] \* x[i]);

}

double b = ((n\*sum\_xy) - (sum\_x\*sum\_y)) / ((n\*sum\_xx) - (sum\_x\*sum\_x));

double a = (sum\_y/n) - ((b\*sum\_x)/n);

cout << "y = " << a << " + " << b << "x" << endl;

return 0;

}

**Least Square Parabola**

#include <bits/stdc++.h>

using namespace std;

double determinant(double mat[3][3])

{

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[3][3], double B[3][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];

}

}

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

{

A[i][row] = B[i][1];

}

double det = determinant(A);

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()

{

///Peace be with you.

vector<double> x = {2, 4, 6, 8};

vector<double> y = {1.4, 2.0, 2.4, 2.8};

int n = y.size();

double sum\_x = 0, sum\_y = 0, sum\_xy = 0, sum\_xx = 0, sum\_xxx = 0, sum\_xxxx = 0, sum\_xxy = 0;

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

{

sum\_x = sum\_x + x[i];

sum\_y = sum\_y + y[i];

sum\_xy = sum\_xy + (x[i] \* y[i]);

sum\_xx = sum\_xx + (x[i] \* x[i]);

sum\_xxx = sum\_xxx + (x[i] \* x[i] \* x[i]);

sum\_xxxx = sum\_xxxx + (x[i] \* x[i] \* x[i] \* x[i]);

sum\_xxy = sum\_xxy + (x[i] \* x[i] \* y[i]);

}

/\*

cout << sum\_x << " " << sum\_y << endl;

cout << sum\_xx << " " << sum\_xxx << " " << sum\_xxxx << endl;

cout << sum\_xy << " " << sum\_xxy << endl;

\*/

double a[3][3], b[3][1];

a[1][1] = n;

a[1][2] = sum\_x;

a[1][3] = sum\_xx;

b[1][1] = sum\_y;

a[2][1] = sum\_x;

a[2][2] = sum\_xx;

a[2][3] = sum\_xxx;

b[2][1] = sum\_xy;

a[3][1] = sum\_xx;

a[3][2] = sum\_xxx;

a[3][3] = sum\_xxxx;

b[3][1] = sum\_xxy;

double order = 3, ans\_a, ans\_b, ans\_c;

/\*

cout << determinant(a) << endl;

cout << Cramer\_Determinant(1, a, b, order) << endl;

cout << Cramer\_Determinant(2, a, b, order) << endl;

cout << Cramer\_Determinant(3, a, b, order) << endl;

\*/

ans\_a = Cramer\_Determinant(1, a, b, order)/determinant(a);

ans\_b = Cramer\_Determinant(2, a, b, order)/determinant(a);

ans\_c = Cramer\_Determinant(3, a, b, order)/determinant(a);

cout << "y = " << ans\_a << " + " << ans\_b << "x + " << ans\_c << "x^2" << endl;

return 0;

}

**Best Possible Values of a and b**

#include <bits/stdc++.h>

using namespace std;

int main()

{

///Peace be with you.

vector<double> x = {2, 10, 26, 61};

vector<double> y = {600, 500, 400, 350};

int n = y.size();

double sum\_x = 0, sum\_y = 0, sum\_Y = 0, sum\_xY = 0, sum\_xx = 0;

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

{

sum\_x = sum\_x + x[i];

sum\_y = sum\_y + y[i];

sum\_Y = sum\_Y + log(y[i]);

sum\_xY = sum\_xY + (x[i] \* log(y[i]));

sum\_xx = sum\_xx + (x[i] \* x[i]);

}

double b = ((n\*sum\_xY) - (sum\_x\*sum\_Y)) / ((n\*sum\_xx) - (sum\_x\*sum\_x));

double a = (sum\_Y/n) - ((b\*sum\_x)/n);

cout << "y = " << exp(a) << " e^ " << b << "x" << endl;

return 0;

}

**Euler’s Method**

#include<bits/stdc++.h>

using namespace std;

int main()

{

///Peace be with you.

// y(0) = 1

// y(x0) = y0

double x0 = 0.0, y0 = 1.0;

double h = 0.01, Given\_x = 0.02;

for(double i = h; i <= Given\_x; i = i + h)

{

x0 = x0 + h;

y0 = y0 + h\*((x0\*x0\*x0) + y0);

}

cout<<"y("<< Given\_x << ") = " << y0 << endl;

return 0;

}

**Runge – Kutta Method.**

#include<bits/stdc++.h>

using namespace std;

int main()

{

///Peace be with you.

// y(0) = 1

// y(x0) = y0

double x0 = 0.0, y0 = 1.0;

double h = 0.1, Given\_x = 0.2;

double m1, m2, m3, m4;

for(double i = h; i <= Given\_x; i = i + h)

{

m1 = h \* (x0 + y0);

m2 = h \* ((x0 + h / 2) + (y0 + m1 / 2));

m3 = h \* ((x0 + h / 2) + (y0 + m2 / 2));

m4 = h \* ((x0 + h) + (y0 + m3));

y0 = y0 + (m1 + 2 \* m2 + 2 \* m3 + m4) / 6;

}

cout<<"y("<< Given\_x << ") = " << y0 << endl;

return 0;

}