RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Lab report: 06

Date of Experiment: 24.03.2018
Date of Submission: 01.04.2018

Submitted to:
Shyla Afroge
Assistant Professor,
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Submitted by:
Riyad Morshed Shoeb
Roll No: 1603013
Section: A
Department of Computer Science and Engineering
Rajshahi University of Engineering and Technology

Name of the experiment: Implementation of Least Square Curve Fitting Procedures

Theory:

Fitting a Straight Line

$Let, Y = a_0 + a_1 x \text{ be the stright line to be fitted to the given data. Then we have,}$

$$S = [y_1 - (a_0 + a_1 x_1)]^2 + [y_2 - (a_0 + a_1 x_2)]^2 + \cdots + [y_m - (a_0 + a_1 x_m)]^2$$

for S to be minimum, we have,

$$\frac{\partial S}{\partial a_0} = 0 \qquad \frac{\partial S}{\partial a_1} = 0$$

$$\frac{\partial S}{\partial a_0} = -2[y_1 - (a_0 + a_1 x_1)] - 2[y_2 - (a_0 + a_1 x_2)] - \cdots - 2[y_m - (a_0 + a_1 x_m)]$$

$$\therefore [y_1 - (a_0 + a_1 x_1)] + [y_2 - (a_0 + a_1 x_2)] + \cdots + [y_m - (a_0 + a_1 x_m)] = 0$$

$$\Rightarrow y_1 + y_2 + \cdots + y_m = m a_0 + a_1 (x_1 + x_2 + \cdots + x_m)$$

$$\Rightarrow \sum_{i=1}^{m} y_i = m a_0 + a_1 \sum_{i=1}^{m} x_i$$

$$\frac{\partial S}{\partial a_1} = -2x_1[y_1 - (a_0 + a_1 x_1)] - 2x_2[y_2 - (a_0 + a_1 x_2)] - \cdots - 2x_m[y_m - (a_0 + a_1 x_m)]$$

$$\therefore x_1[y_1 - (a_0 + a_1 x_1)] + x_2[y_2 - (a_0 + a_1 x_2)] + \cdots + x_m[y_m - (a_0 + a_1 x_m)] = 0$$

$$\Rightarrow x_1 y_1 + x_2 y_2 + \cdots + x_m y_m = a_0 (x_1 + x_2 + \cdots + x_m) + a_1 (x_1^2 + x_2^2 + \cdots + x_m^2)$$

$$\Rightarrow \sum_{i=1}^{m} x_i y_i = a_0 \sum_{i=1}^{m} x_i + a_1 \sum_{i=1}^{m} x_i^2$$

Polynomial of nth Degree

Let the polynomial of nth degree,

$$Y = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

$be\ fitted\ to\ the\ data\ points\ (x_i, y_i), i = 1,2, \ldots, m. Then, we\ have,$

$$S = [y_1 - (a_0 + a_1 x_1 + \cdots + a_n x_1^n)]^2 + [y_2 - (a_0 + a_1 x_2 + \cdots + a_n x_2^n)]^2 + \cdots + [y_m - (a_0 + a_1 x_m + \cdots + a_n x_m^n)]^2$$

Equating the first partial derivatives,

$$m a_0 + a_1 \sum_{i=1}^{m} x_i + a_2 \sum_{i=1}^{m} x_i^2 + \cdots + a_n \sum_{i=1}^{m} x_i^n = \sum_{i=1}^{m} y_i$$

$$a_0 \sum_{i=1}^{m} x_i + a_1 \sum_{i=1}^{m} x_i^2 + \cdots + a_n \sum_{i=1}^{m} a_i^{n+1} = \sum_{i=1}^{m} x_i y_i$$

$$\vdots$$

$$a_0 \sum_{i=1}^{m} x_i^n + a_1 \sum_{i=1}^{m} x_i^{n+1} + \cdots + a_n \sum_{i=1}^{m} x_i^{2n} = \sum_{i=1}^{n} x_i^n y_i$$

Exponential Function

Let the curve

$$y = a_0 e^{a_1 x}$$

be fitted to the given data. Taking logarithms of both sides, we get,

$$\log_e y = \log_e a_0 + a_1 x \log_e e$$

$$\Rightarrow \ln y = \ln a_0 + a_1 x$$

which can be written in the form,

$$Z = A + Bx$$

$$where, \qquad Z = \ln y, \qquad A = \ln a_0, \qquad B = a_1$$

The problem therefore reduces to a least squares straight line through the given data.

Code:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cmath>
using namespace std;

void linear_curve(void)
{
    int i,n;
    double s_x=0;
    double s_y=0;
    double s_x2=0;
    double s_xy=0;
    double a0,a1;
    double x_check;
    double y_check;

    printf("Enter the number of inputs: ");
    cin>>n;
    double x[n];
    double y[n];
    printf("Enter the values:\n  x  |  y\n");
    for(i=0;i<n;i++)
        cin>>x[i]>>y[i];
    for(i=0;i<n;i++)
    {
        s_x+=x[i];
        s_y+=y[i];
        s_x2+=(x[i]*x[i]);
        s_xy+=(x[i]*y[i]);
    }
    a0=((s_x2*s_y)-(s_x*s_xy))/((n*s_x2)-(s_x*s_x));
    a1=((n*s_xy)-(s_x*s_y))/((n*s_x2)-(s_x*s_x));
    printf("a0= ");
    cout<<a0<<endl;
    printf("a1= ");
    cout<<a1<<endl;
    printf("Enter a value of x: ");
    cin>>x_check;
    y_check=a0+(a1*x_check);
    printf("Y= ");
    cout<<y_check<<endl;
    for(i=0;i<n;i++)
    {
        if(x_check==x[i])
        {
            printf("Absolute error: ");
            cout<<abs(y_check-y[i]);
            break;
        }
    }
    printf("\n\n");
}

void non_linear_curve(void)
{
    int i,n;
```

```
    double s_x=0;
    double s_y=0;
    double s_x2=0;
    double s_x3=0;
    double s_x4=0;
    double s_x2y=0;
    double s_xy=0;
    double d,dC,dA,dB,a0,a1,a2;
    double x_check,y_check;

    printf("Enter the number of inputs: ");
    cin>>n;
    double x[n];
    double y[n];
    printf("Enter the values:\n  x  |  y\n");
    for(i=0;i<n;i++)
        cin>>x[i]>>y[i];
    for(i=0;i<n;i++)
    {
        s_x=s_x+x[i];
        s_y=s_y+y[i];
        s_x2=s_x2+(x[i]*x[i]);
        s_xy=s_xy+(x[i]*y[i]);
        s_x3=s_x3+(x[i]*x[i]*x[i]);
        s_x4=s_x4+(x[i]*x[i]*x[i]*x[i]);
        s_x2y=s_x2y+(x[i]*x[i]*y[i]);
    }
    d=n*(s_x2*s_x4-s_x3*s_x3)-s_x*(s_x*s_x4-s_x2*s_x3)+s_x2*(s_x*s_x3-
s_x2*s_x2);
    dA=s_y*(s_x2*s_x4-s_x3*s_x3)-s_x*(s_xy*s_x4-s_x2y*s_x3)+s_x2*(s_xy*s_x3
-s_x2*s_x2y);
    dB=n*(s_xy*s_x4-s_x2y*s_x3)-s_y*(s_x*s_x4-s_x2*s_x3)+s_x2*(s_x*s_x2y-
s_x2*s_xy);
    dC=n*(s_x2*s_x2y-s_x3*s_xy)-s_x*(s_x*s_x2y-s_x2*s_xy)+s_y*(s_x*s_x3-
s_x2*s_x2);
    a0=dA/d;
    a1=dB/d;
    a2=dC/d;
    cout<<"a0="<<a0<<"\na1="<<a1<<"\na2="<<a2<<endl;
    printf("Enter a value of x: ");
    cin>>x_check;
    y_check=a0+(a1*x_check)+(a2*x_check*x_check);
    printf("Y= ");
    cout<<y_check<<endl;
    for(i=0;i<n;i++)
    {
        if(x_check==x[i])
        {
            printf("Absolute error: ");
            cout<<abs(y[i]-y_check);
            break;
        }
    }
    printf("\n\n");
}

void exponential_curve(void)
{
    int n,i;
    double s_x=0;
    double s_y=0;
```

```
    double s_x2=0;
    double s_xy=0;
    double a0,a1;
    double a,b;
    double x_check,y_check;

    printf("Enter the number of inputs: ");
    cin>>n;
    double x[n];
    double y[n];
    double Y[n];
    printf("Enter the values:\n  x  |  y\n");
    for(i=0;i<n;i++)
        cin>>x[i]>>y[i];
    for(i=0;i<n;i++)
        Y[i]=log(y[i]);
    for(i=0;i<n;i++)
    {
        s_x=s_x+x[i];
        s_y=s_y+Y[i];
        s_x2=s_x2+(x[i]*x[i]);
        s_xy=s_xy+(x[i]*Y[i]);
    }
    a0=((s_x2*s_y)-(s_x*s_xy))/((n*s_x2)-(s_x*s_x));
    b=((n*s_xy)-(s_x*s_y))/((n*s_x2)-(s_x*s_x));
    a=exp(a0);
    cout<<"a= "<<a<<endl;
    cout<<"b= "<<b<<endl;
    printf("Enter a value of x: ");
    cin>>x_check;
    y_check=a*exp(b*x_check);
    printf("Y= ");
    cout<<y_check<<endl;
    for(i=0;i<n;i++)
    {
        if(x_check==x[i])
        {
            printf("Absolute error: ");
            cout<<abs(y_check-y[i]);
            break;
        }
    }
    printf("\n\n");
}

int main(void)
{
    int checker;

    while(1)
    {
        printf("1. Linear curve fitting\n2. Non-linear curve fitting\n3.
Exponential curve fitting\n0. Exit\n   Enter your choice: ");
        cin>>checker;
        switch(checker)
        {
        case 0:
            return 0;
        case 1:
            linear_curve();
            break;
```

```
        case 2:
            non_linear_curve();
            break;
        case 3:
            exponential_curve();
            break;
        default:
            printf("Wrong Choice...\n\n");
        }
    }
}
```

Output:

```
"D:\2nd year odd sem\CSE 2104\Lab_6\Curve_fitting_menu.exe"
1. Linear curve fitting
2. Non-linear curve fitting
3. Exponential curve fitting
0. Exit
   Enter your choice: 1
Enter the number of inputs: 6
Enter the values:
  x  |  y
20 800.3
30 800.4
40 800.6
50 800.7
60 800.9
70 801.0
a0= 799.994
a1= 0.0145714
Enter a value of x: 50
Y= 800.723
Absolute error: 0.0228571

1. Linear curve fitting
2. Non-linear curve fitting
3. Exponential curve fitting
0. Exit
   Enter your choice: 2
Enter the number of inputs: 3
Enter the values:
  x  |  y
0 1
1 6
2 17
a0=1
a1=2
a2=3
Enter a value of x: 1
Y= 6
Absolute error: 0

1. Linear curve fitting
2. Non-linear curve fitting
3. Exponential curve fitting
0. Exit
   Enter your choice: 3
Enter the number of inputs: 5
Enter the values:
  x  |  y
2 4.077
4 11.084
6 30.128
8 81.897
10 222.62
a= 1.4999
b= 0.500008
Enter a value of x: 6
Y= 30.1278
Absolute error: 0.000163863

1. Linear curve fitting
2. Non-linear curve fitting
3. Exponential curve fitting
0. Exit
   Enter your choice: 4
Wrong Choice...

1. Linear curve fitting
2. Non-linear curve fitting
3. Exponential curve fitting
0. Exit
   Enter your choice: 0

Process returned 0 (0x0)   execution time : 176.703 s
Press any key to continue.
```