**Least Square Curve Fitting(Nonlinear):**

The given data may not always follow a linear relationship. We consider a polynomial of nth degree and an exponential function to fit the data points (Xi,Yi).

Where i=1, 2, 3 …m.

**Linearization of Nonlinear Laws :**

**Y=aebx**

In this case ,we write

lny=lna+bx

=>Y=A0+A1X

Where Y=lny , A0=lna , A1=b ,X=x

**Source Code:**

#include<stdio.h>

int main()

{

int i,m;

float x[10],y[10],sum1,sum2,sum3,sum4,sum5,sum6,sum7;

float avr1, avr2,a1,a0,p,q;

printf("Enter the number of records : ");

scanf("%d",&m);

printf("Enter the value of x & y :\n");

sum1=0,sum2=0,sum3=0,sum4=0,sum5=0,sum6=0,sum7=0;

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

{

scanf("%f%f",&x[i],&y[i]);

sum1+=x[i];

sum2+=log(y[i]);

sum3+=x[i]\*x[i];

sum4+=x[i]\*log(y[i]);

}

avr1=sum1/m;

avr2=sum2/m;

printf("\n\t Xi Yi X\*X Xi\*Yi\n");

printf("-----------------------------------------");

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

printf("\n\t %.3f %.3f %.3f %.3f ",x[i],y[i],x[i]\*x[i],x[i]\*y[i]);

}

printf("\n-----------------------------------------");

printf("\n\t%.3f %.3f %.3f %.3f \n",sum1,sum2,sum3,sum4);

a1=(m\*sum4-sum1\*sum2)/(m\*sum3-sum1\*sum1);

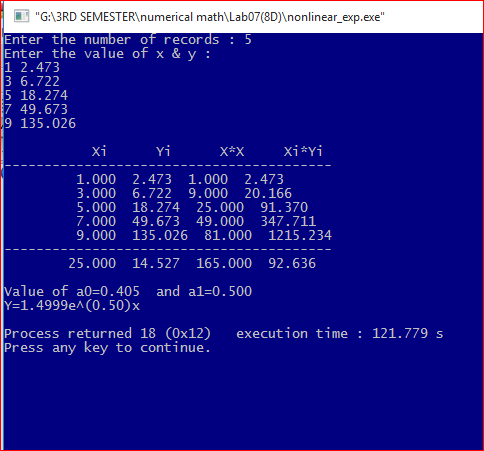
a0=avr2-a1\*avr1;

printf("\nValue of a0=%.3f and a1=%.3f\n",a0,a1);

printf("Y=%.4fe^(%.2f)x\n",pow(exp(1),a0),a1);

}

**Input/Output:**



**Curve fitting by Polynomial:**

Let the polynomial of the nth degree ,

Y=a0+a1x+a2x2+…….+anxn

Be fitted to the data points (xi,yi), i=1,2,3,……..m.We then have

S=[y1-(a0+a1x1+a2x1+…….anx1n)]2+[y2-(a0+a1x2+a2x22+…….anxnn)]2+…..+[ym- (a0+a1xm+a2xm2+…….anxmn)]2

Equating to zero the first partial derivatives and simplifying ,we obtain the normal equations:

ma0+a1∑xi+a2 an ∑yi

a0∑xi+a1 …….an ∑xiyi

.

a0a1 …….an ∑xniyi

where the summations are performed from i=1 to i=m

**Source Code:**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int i,n;

double sumx=0,sumy=0,sumx2=0,sumx3=0,sumx4=0,sumx2y=0,sumxy=0,D,dC,dA,dB,a,b,c;

cout<<"Enter the number of record";

cin>>n;

double x[n],y[n];

cout<<"X Y"<<endl;

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

cin>>x[i]>>y[i];

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

{

sumx=sumx+x[i];

sumy=sumy+y[i];

sumx2=sumx2+(x[i]\*x[i]);

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

sumx3=sumx3+(x[i]\*x[i]\*x[i]);

sumx4=sumx4+(x[i]\*x[i]\*x[i]\*x[i]);

sumx2y=sumx2y+(x[i]\*x[i]\*y[i]);

}

D=n\*(sumx2\*sumx4-sumx3\*sumx3)-sumx\*(sumx\*sumx4-sumx2\*sumx3)+sumx2\*(sumx\*sumx3-sumx2\*sumx2);

dA=sumy\*(sumx2\*sumx4-sumx3\*sumx3)-sumx\*(sumxy\*sumx4-sumx2y\*sumx3)+sumx2\*(sumxy\*sumx3-sumx2\*sumx2y);

dB=n\*(sumxy\*sumx4-sumx2y\*sumx3)-sumy\*(sumx\*sumx4-sumx2\*sumx3)+sumx2\*(sumx\*sumx2y-sumx2\*sumxy);

dC=n\*(sumx2\*sumx2y-sumx3\*sumxy)-sumx\*(sumx\*sumx2y-sumx2\*sumxy)+sumy\*(sumx\*sumx3-sumx2\*sumx2);

a=dA/D;

b=dB/D;

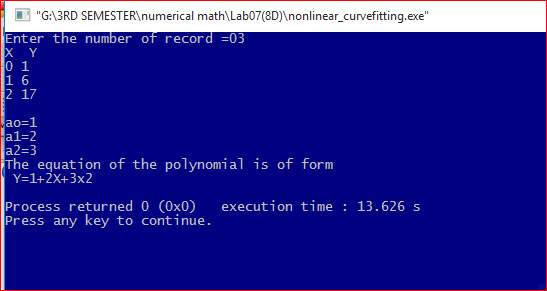
c=dC/D;

cout<<"\nao="<<a<<"\na1="<<b<<"\na2="<<c<<"\n";

cout<<"The equation of the polynomial is of form\n Y="<<a<<"+"<<b<<"X"<<"+"<<c<<"x2\n";

return 0;}

**Input/Output:**



**Discussion:**

The least square curve fitting procedure is a mathematical procedure for finding the best fitting curve to a given set of points by minimizing the sum of the squares of the offsets of the points from the curve. The given data may not always follow a linear relationship. The non-linear form is used for the sum of exponential function and polynomial functions.