In this post I will be showing you how to write a code that fits the data-points to an exponential function, like:

$$y = Ae^{Bx} \qquad eq(i)$$

where, A & B are some constants that we will determine.

We will be using the Least Squares Method (also known as Chi square minimization) to achieve this.

Let's say you have *n* data points: x_i and y_i .
Then the fitted function can be calculated by minimizing the error(difference between the actual and fitted point):

minimize: $\boxed{err = \Sigma_{i=1}^{n}(Y_i - Ae^{Bx_i})^2}$

where $Y_i = Ae^{Bx_i}$

But this will give us alot of problem as doing that is not easy and a topic for another post, and very mathematical.
To cut the long story short, what we do instead is, we apply a trick, that is, we take the logarithm of eq(1) to get rid of the exponential

$$\ln(Y_i) = \ln(A) + Bx_i$$

and applying a quick change of variables as :

$$Yn_i = \ln(Y_i)$$
$$c = \ln(A)$$

we get,

$$Yn_i = Bx_i + c$$

which is exactly the equation of a straight line, and therefore, it becomes a problem of linear fitting. And we have already seen how to write a Linear Fitting program. We will use the following formulae from there:

$$\boxed{B = \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}}$$

$$\boxed{c = \frac{\Sigma x_i^2 \Sigma y_i - \Sigma x_i \Sigma x_i y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}}$$

You can refer to this link for a detailed proof.

From 'c' we calculate the value of A using:

$$A = e^c$$

So you will need to have some code for the user two enter the data-points or you could add them manually by initializing the arrays.

Once you have the data-points stored in the x and y arrays,
you can use the following code to find out the value of 'A' and 'B', which are the coefficients of exponential fitting funtion.

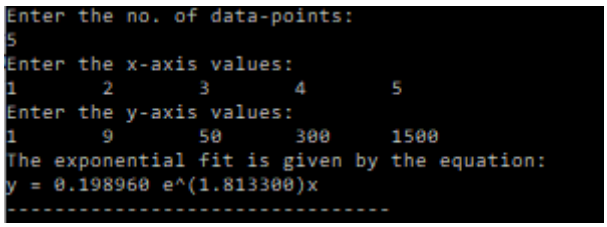
CODE:

```
/*******************************************************
*************Chi-square fitting**************
Exponential Fitting: y=Ae^bx
*******************************************************/
#include<stdio.h>
#include<math.h>
/*****
```

```
Function that calculates and returns the slope of the best fit line
Parameters:
N: no. of data-points
x[N]: array containing the x-axis points
y[N]: array containing the corresponding y-axis points
*****/
double slope(int N, double x[N], double y[N]){
        double m;
        int i;
        double sumXY=0;
        double sumX=0;
        double sumX2=0;
        double sumY=0;
        for(i=0;i<N;i++){
                sumXY=sumXY+x[i]*y[i];
                sumX=sumX+x[i];
                sumY=sumY+y[i];
                sumX2=sumX2+x[i]*x[i];
        }
        sumXY=sumXY/N;
        sumX=sumX/N;
        sumY=sumY/N;
        sumX2=sumX2/N;
        m=(sumXY-sumX*sumY)/(sumX2-sumX*sumX);
        return m;
}
/*****
Function that calculates and returns the intercept of the best fit line
Parameters:
N: no. of data-points
x[N]: array containing the x-axis points
y[N]: array containing the corresponding y-axis points
*****/
double intercept(int N, double x[N], double y[N]){
        double c;
        int i;
        double sumXY=0;
        double sumX=0;
        double sumX2=0;
        double sumY=0;
        for(i=0;i<N;i++){
                sumXY=sumXY+x[i]*y[i];
                sumX=sumX+x[i];
                sumY=sumY+y[i];
                sumX2=sumX2+x[i]*x[i];
        }
        sumXY=sumXY/N;
        sumX=sumX/N;
        sumY=sumY/N;
        sumX2=sumX2/N;
        c=(sumX2*sumY-sumXY*sumX)/(sumX2-sumX*sumX);
        return c;
```

```
}
main(){
        int N;
        printf("Enter the no. of data-points:\n");
        scanf("%d",&N);
        double x[N], y[N], Y[N];
        printf("Enter the x-axis values:\n");
        int i;
        for(i=0;i<N;i++){
                scanf("%lf",&x[i]);
        }
        printf("Enter the y-axis values:\n");
        for(i=0;i<N;i++){
                scanf("%lf",&y[i]);
        }
        for(i=0;i<N;i++){
                Y[i]=log(y[i]);
        }
        printf("The exponential fit is given by the equation:\n");
        double m=slope(N,x,Y);
        double c=intercept(N,x,Y);
        double A, b; //y=Ae^bx
        A=exp(c);
        b=m;
        printf("y = %lf e^(%lf)x",A,b);
}
```

OUTPUT:



Sample output 1

So that's it.
You now have the value of 'A' and 'B' and thus the exponential fit:

$$y = Ae^{Bx}$$

You can refer to the following links for more info:
Exponential Fitting – Lab Write-Up
Exponential Fitting – C++ Program
Exponential Fitting – Scilab Code
Curve Fit Tools – Android App (using the above code)
Curve Fit Tools – Documentation
Curve Fit Tools – Play Store
Curve Fit Tools – GitHub Repository
Curve Fitters – Scilab Toolbox

Hope you found this post useful.
If you have any questions/doubts drop them in the comments section down below.

Manas Sharma

PhD researcher at Friedrich-Schiller University Jena, Germany. I'm a physicist specializing in theoretical, computational and experimental condensed matter physics. I like to develop Physics related apps and softwares from time to time. Can code in most of the popular languages. Like to share my knowledge in Physics and applications using this Blog and a YouTube channel.