In this post I will be showing you how to write a code that fits the data-points to an exponential function, like:
$y = ce^{ax}$ eq(1)
where, c & a are some constants that we will determine.

We will be using the Least Squares Method 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: $err = \Sigma_{i=1}^{n}(Y_i - ce^{aX_i})^2$
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 apply a trick, that is, we take the logarithm of eq(1) to get rid of the exponential $ln(y) = ln(c) + ax$
and applying a quick change of variables as :

it becomes a problem of linear fit. We already know the formula for calculating the coefficients.
Skipping all the math, we get the following formulas for c and a:
You can refer to this link for a detailed proof.

The following JAVA Code for exponential fitting works for android devices too.
Let's say you have declared two arrayLists to store the x-axis and y-axis values.

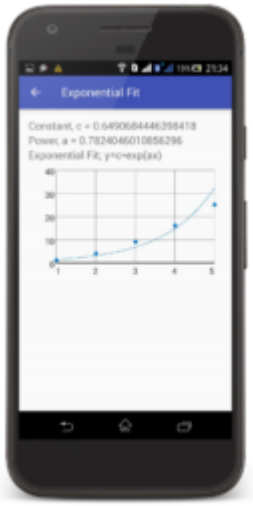
```
public static ArrayList<String> x_axis=new ArrayList<String>();
public static ArrayList<String> y_axis=new ArrayList<String>();
```

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 arrayLists. If you are using this for Android App, then you would get the input of the user inside an EditText and then keep appending the values to the arrayList.
The advantage of using arrayList over an array is that you don't need to know it's size before declaring it. And it can keep getting bigger.

Once you have the data-points stored in the x_axis and y_axis arraylists,
you can use the following code to find out the value of 'c' and 'a'.

```
        double xsum=0;
        double ysum=0;
        double xysum=0;                                     //variables for sums/sigma of
xi,yi,xi^2,xiyi etc
        double x2sum=0;
        int n=x_axis.size();    //find out the size of the arraylist(to know the no. of
data points entered)
        double y[]=new double[n];     //create an array where we will store ln(yi)
        for (int i=0;i<n;i++){
            y[i]=Math.log(Double.parseDouble(y_axis.get(i)));    //Calculate the values
of ln(yi)
        }
        for(int i=0;i<n;i++){
            xsum=Double.parseDouble(x_axis.get(i))+xsum;     //calculate sigma(xi)
            ysum=y[i]+ysum;                                  //calculate sigma(yi)
            xysum=Double.parseDouble(x_axis.get(i))*y[i]+xysum;            //calculate
sigma(xi*yi)
x2sum=Double.parseDouble(x_axis.get(i))*Double.parseDouble(x_axis.get(i))+x2sum;
//calculate sigma(x^2i)

        }
        a=(n*xysum-xsum*ysum)/(n*x2sum-xsum*xsum);              //calculate slope(or
the the power of exp)
        b=(x2sum*ysum-xsum*xysum)/(x2sum*n-xsum*xsum);          //calculate intercept
        c=Math.exp(b);                                          //since b=ln(c)
```



Exponential Fit Code being used in an app

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

$$y = ce^{ax}$$

To make your code even cleaner, you could wrap all the above code into a method/function and then pass the 'x' and 'y' datapoints as arraylists and get back the values of 'c' and 'a'.
The reason I didn't do it is because Java sucks and doesn't even support multiple return variables.
So I would have had to return the values in an array, and it felt unnecessary.

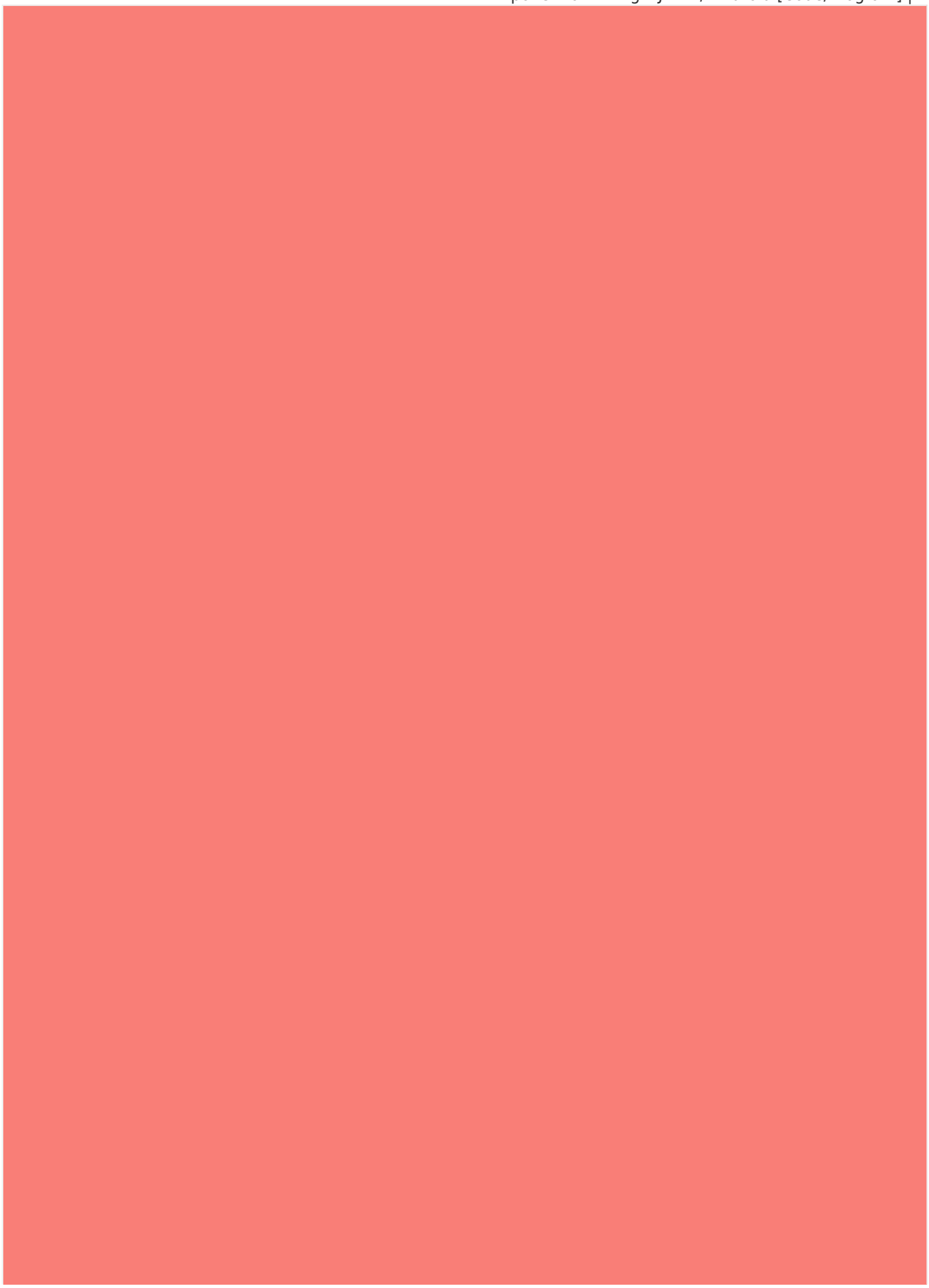
You can refer to the following links for more info:
Exponential Fitting – Lab Write-Up
Exponential Fitting – C++ Program
Exponential Fitting – Scilab Code
Curve Fit Tools – Android App (using the above code)
Curve Fit Tools – Documentation
Curve Fit Tools – Play Store
Curve Fit Tools – GitHub Repository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

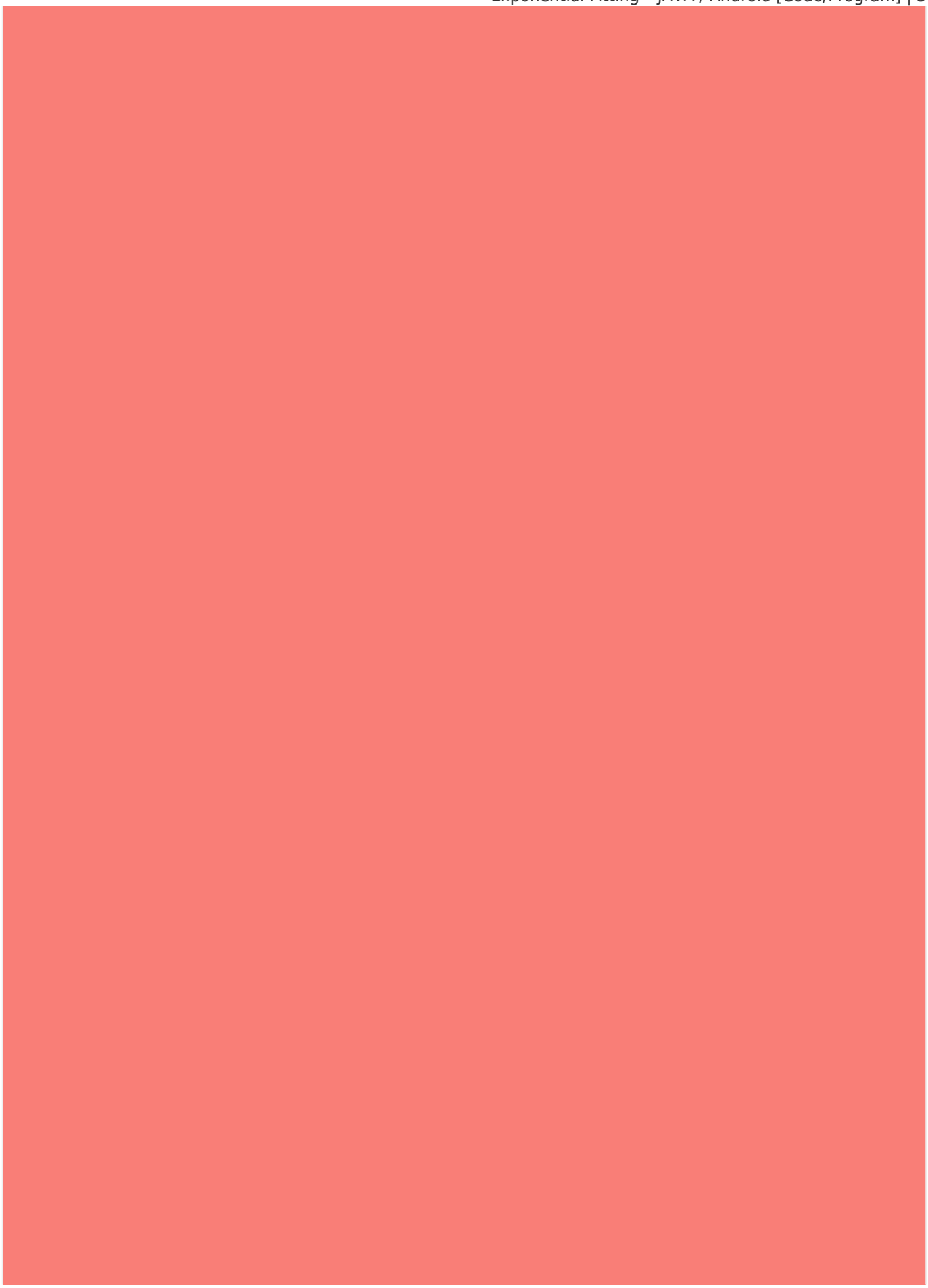
Hope you found this post useful.
If you have any questions/doubts drop them in the comments section down below.

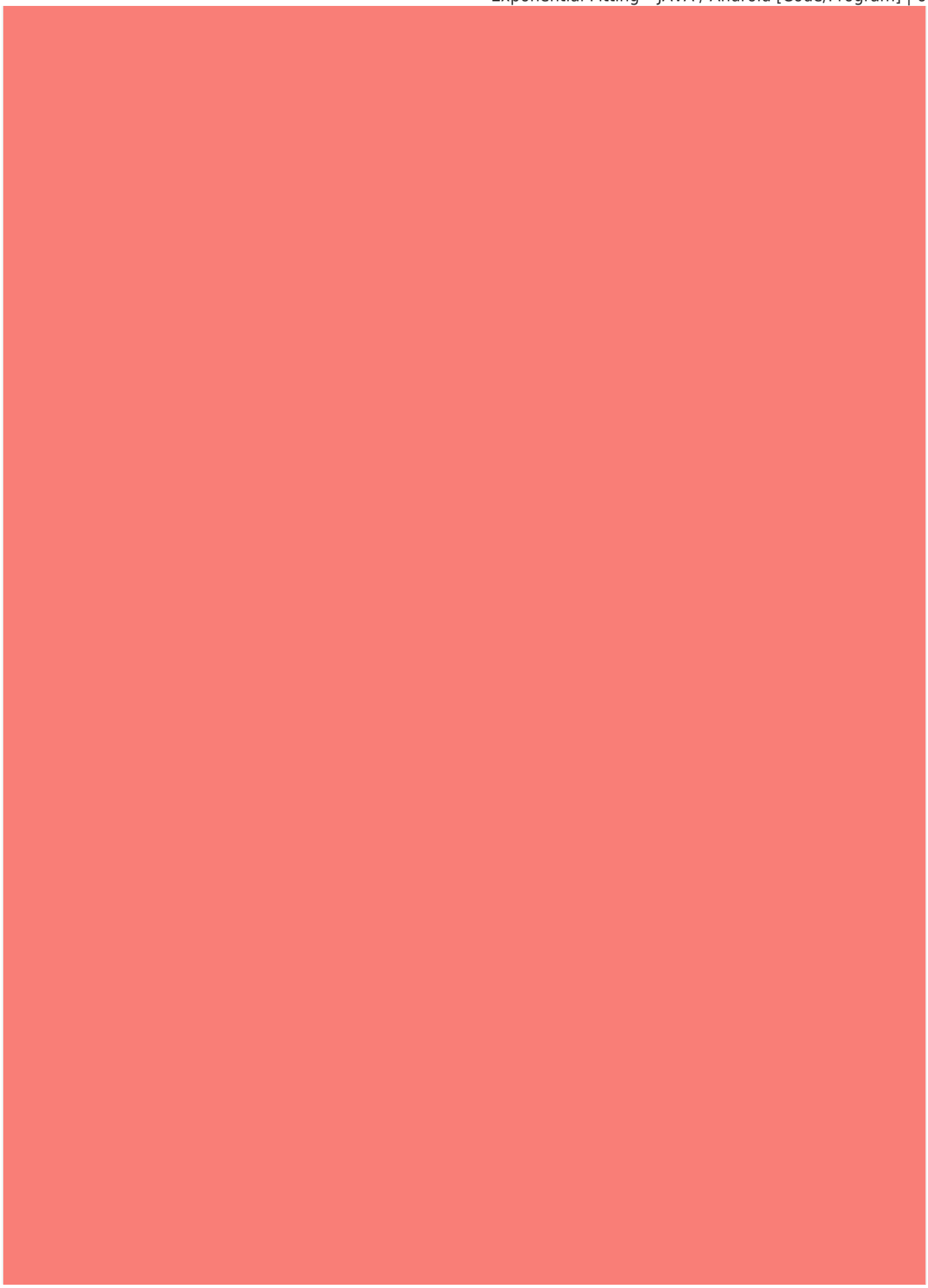
Manas Sharma

I'm a physicist specializing in computational material science with a PhD in Physics from Friedrich-Schiller University Jena, Germany. I write efficient codes for simulating light-matter interactions at atomic scales. I like to develop Physics, DFT, and Machine Learning related apps and software from time to time. Can code in most of the popular languages. I like to share my knowledge in Physics and applications using this Blog and a YouTube channel.

manas.bragitoff.com/







Share this:

Click to share on Facebook (Opens in new window)

Click to share on Twitter (Opens in new window)

Click to share on WhatsApp (Opens in new window)

Click to share on Pinterest (Opens in new window)

Click to share on Reddit (Opens in new window)

Click to share on LinkedIn (Opens in new window)

Click to email a link to a friend (Opens in new window)

Click to print (Opens in new window)

Click to share on Tumblr (Opens in new window)

Click to share on Pocket (Opens in new window)

Click to share on Telegram (Opens in new window)

[wpedon id="7041" align="center"]