In this post I would be explaining the code to find the best linear fit for a given set of data points.
Or in other words, the equation of a line that best fits a given set of data.

The equation of a line is given by:
$$y = mx + c$$
where 'm' is the slope and 'c' is the intercept.

So we will need to determine these constants in the above equation.

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:
$$err = \Sigma_{i=1}^{n}(Y_i - (mx_i + c))^2$$
where, Y_i are the fitted points.
Skipping all the math, we get the following formulae for m and c:
$$m = \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}$$
$$c = \frac{\Sigma x_i^2 \Sigma y_i - \Sigma x_i \Sigma x_i y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}$$

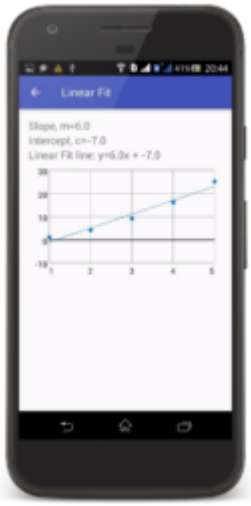
You can refer to this link for a detailed proof.

The following JAVA Code for linear 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>();
```

```
int n=x_axis.size();
for(int i=0;i<n;i++){
        xsum=Double.parseDouble(x_axis.get(i))+xsum;
        ysum=Double.parseDouble(y_axis.get(i))+ysum;
xysum=Double.parseDouble(x_axis.get(i))*Double.parseDouble(y_axis.get(i))+xysum;
x2sum=Double.parseDouble(x_axis.get(i))*Double.parseDouble(x_axis.get(i))+x2sum;

        }
m=(n*xysum-xsum*ysum)/(n*x2sum-xsum*xsum);
c=(x2sum*ysum-xsum*xysum)/(x2sum*n-xsum*xsum);
```



Linear Fit code being used in an app

So that's it.
You now have the value of 'm'(slope) and 'c'(intercept) and thus the linear fit:

$$y = mx + c$$

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 'm' and 'c'.
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.

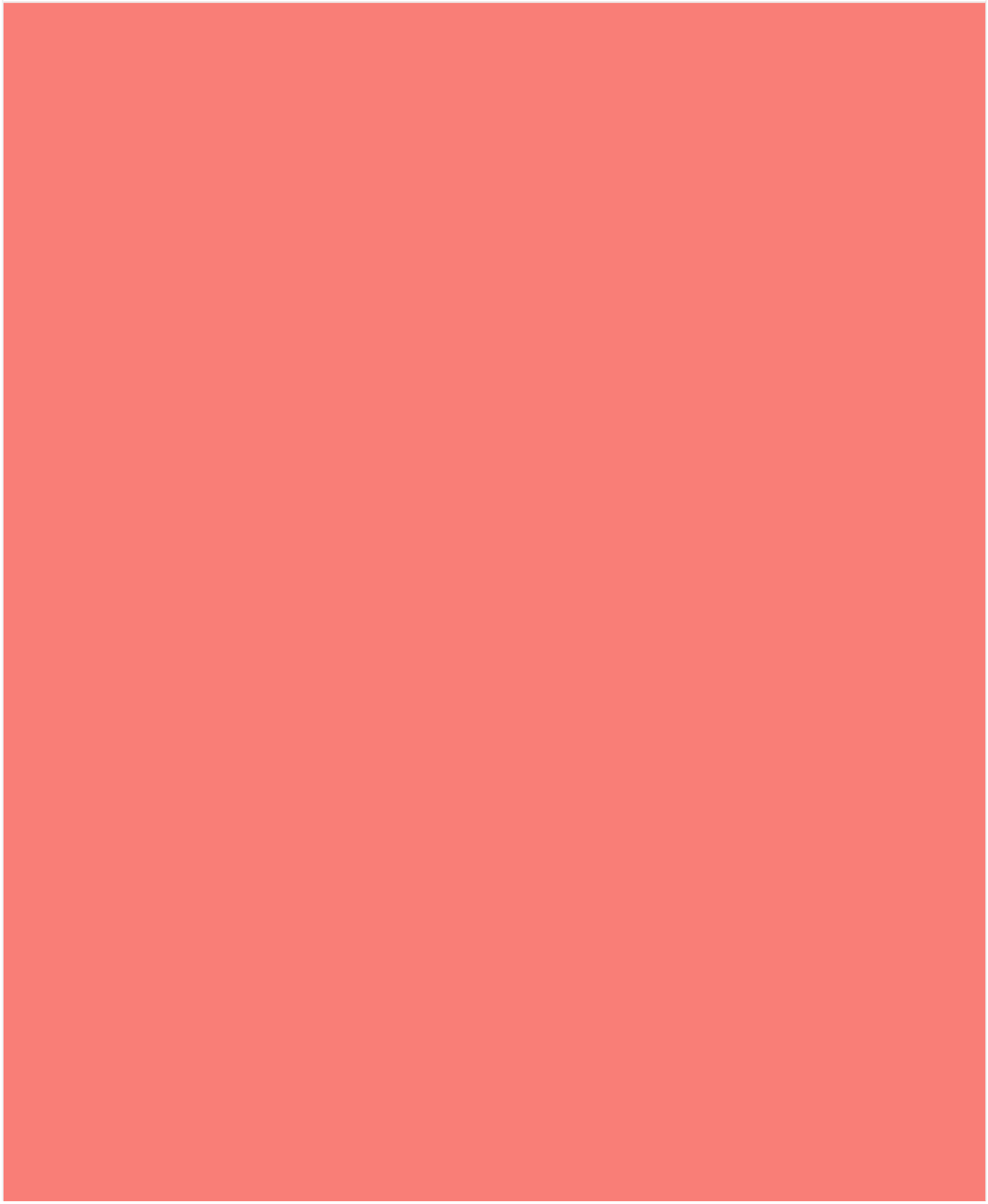
You can refer to the following links for more info:
Linear Fitting – Lab Write-Up
Linear Fitting – C++ Program
Linear Fitting – Scilab Code
Curve Fit Tools – Android App (using the above code)
Curve Fit Tools – Documentation
Curve Fit Tools – Play Store
Curve Fit Tools – GitHub Repository
Curve Fitters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

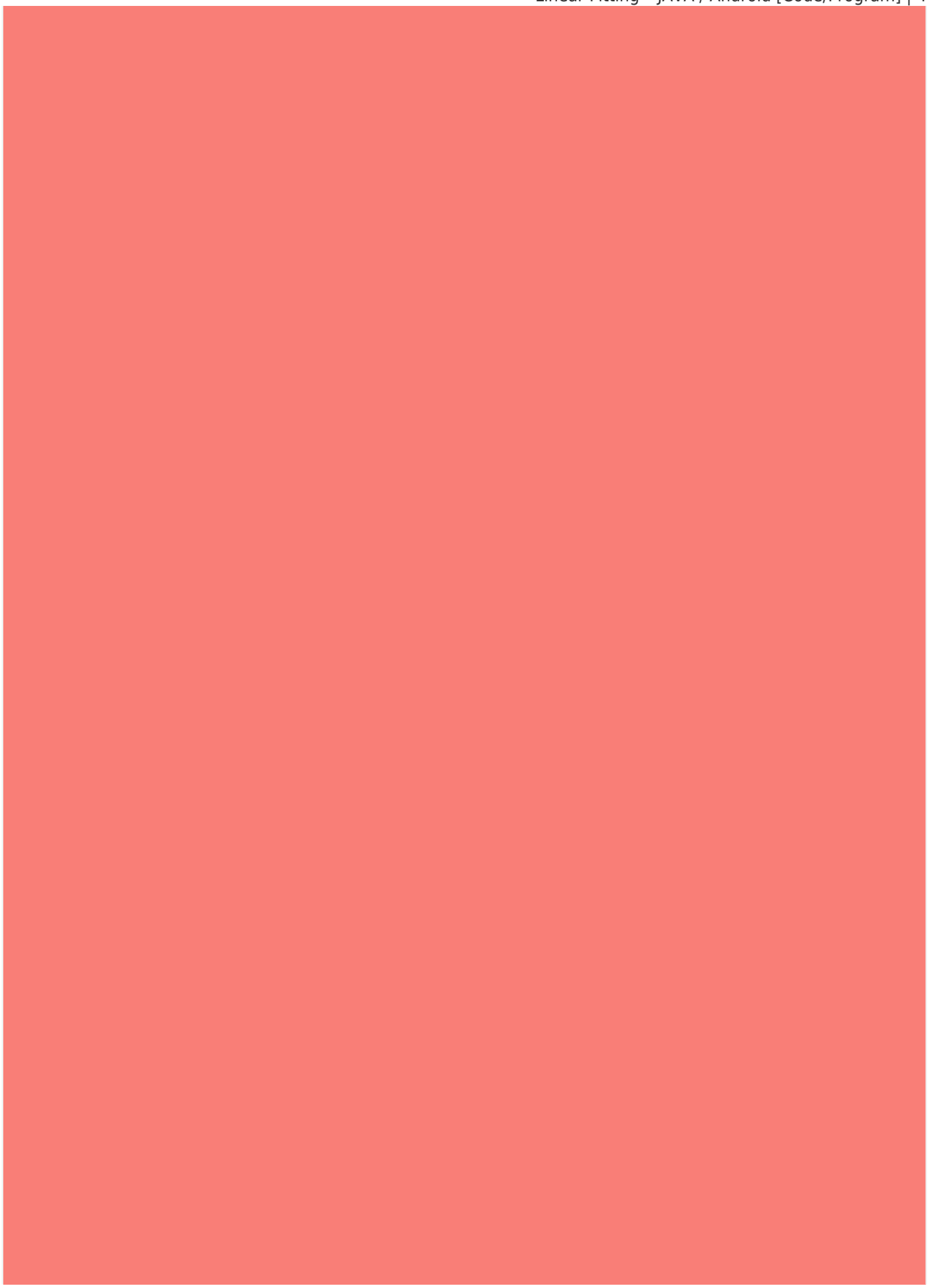
Manas Sharma

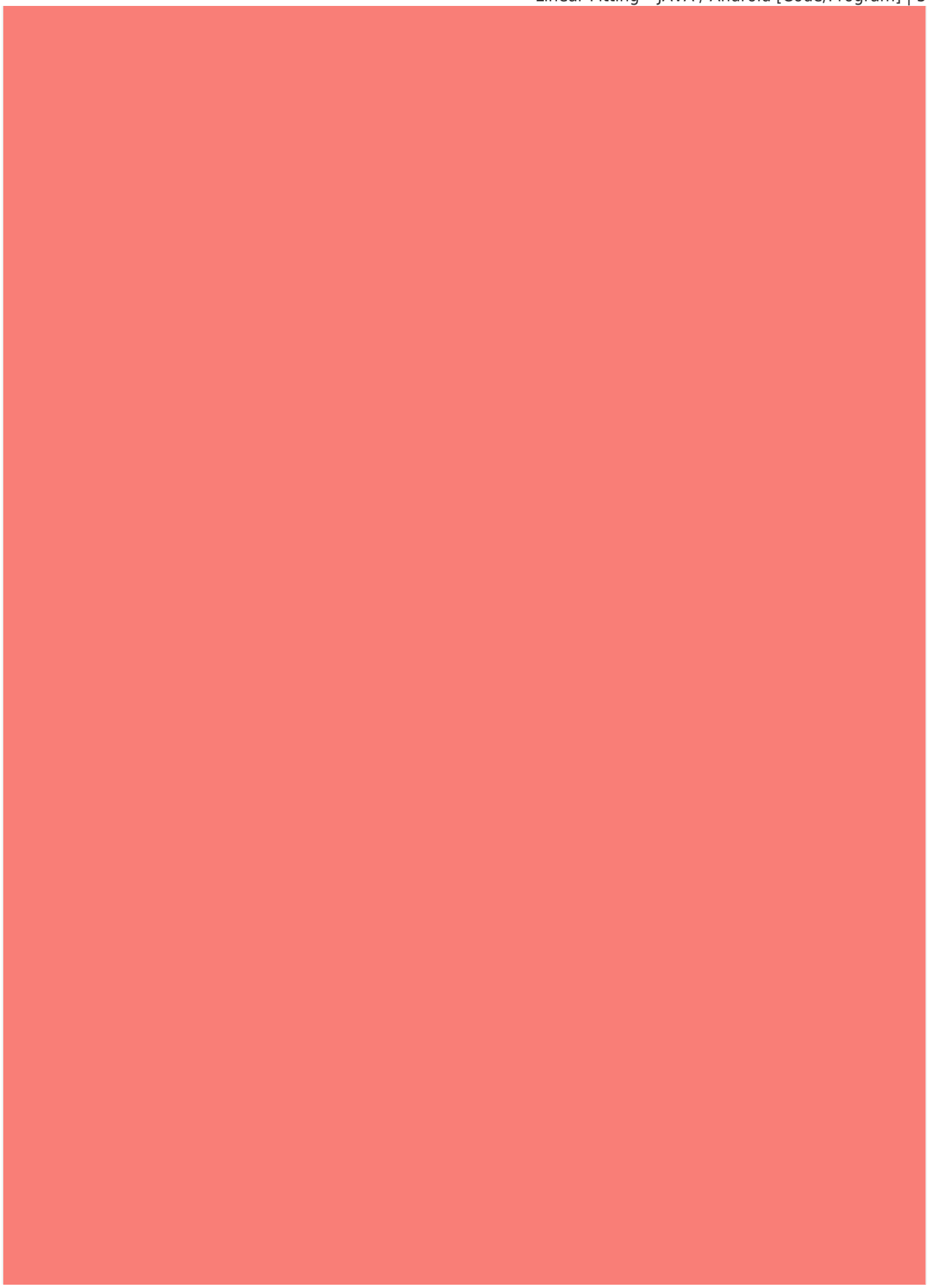
I'm a physicist specializing in computational material science with a PhD in Physics from Friedrich-Schiller University Jena, Germany. I write efficient codes for simulating light-matter interactions at

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"]