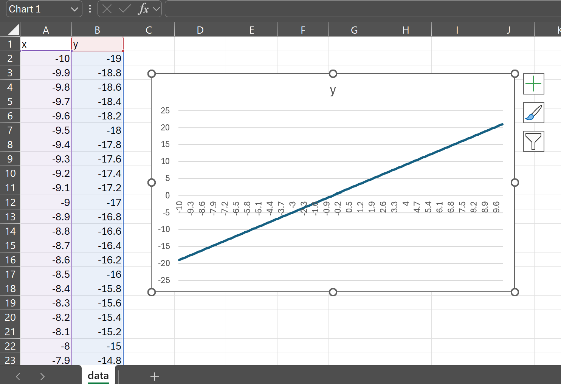
Java Plotting, Salting, Smoothing

Elijah White

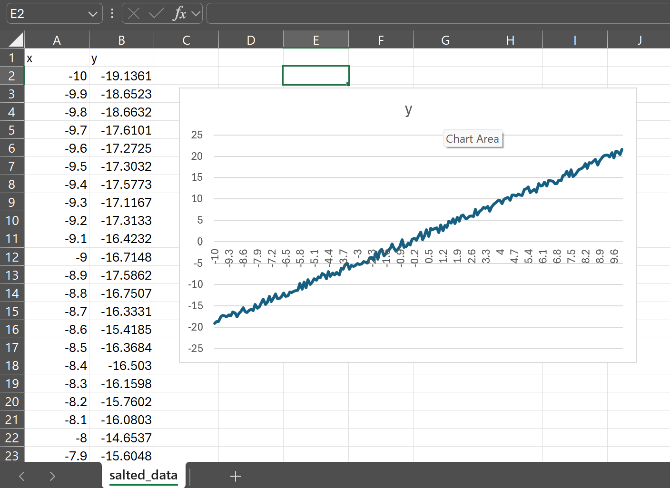
I was asked to create a program in Java to plot any mathematical function we want and play around with its parameters as the first phase of this project. I decided to keep it simple and begin with the linear function y = 2 x + 1 since it is simple and an effective method of testing the program's fundamental operation. The application creates x-values between -10 and 10, determines the appropriate y-values, and exports all the data to a CSV file. I then proceeded to plot the data in Excel after it was exported so I could look at the data outcomes. I ran into some minor problems trying to get the data into Excel. Still, I soon realized that I needed to download an extension called Rainbow CSV, which VS Code kindly suggested and helped make the process easier.

After going into Excel with the newly acquired CSV file, you can see that the plotted graph reveals a clean, upward-sloping straight line, exactly as expected for a linear function with a slope of 2. This confirmed that the logic behind the function and data export process was solid.



Moving forward and learning from my previous mistakes in the past, I’ve realized that I had to plot, salt, and smooth for all three aspects of the project in Java, and now it’s time for me to add on to that. So, we were tasked with creating a salting algorithm, and I wrote a new function called SaltingFunction.java whose job is to salt the previous data that we plotted.

After running the salting program, it was completed, and a new CSV file was generated named salted\_data.csv. I loaded the data into Excel and plotted it just like I did with the plotting part. This time, the graph looked noticeably different. At one point, the original smooth and predictable straight line was now slightly distorted; there were small, random spikes along the line, causing it to appear bumpy or scattered in some areas. However, the general direction and trend of the line were still very visible, indicating that the base function was still intact beneath the noise.



After salting the data, the final steps we were tasked with in the project were to apply a smoothing algorithm to reduce the effects of the introduced noise and restore a clearer pattern. To accomplish this, I created a new Java class called SmootherFunction.java. This function loads the salted data and applies a smoothing technique by averaging the y-values within a configurable window range set to +- 5 for my case. The goal was to replace each y-value with the average of its neighboring values, effectively softening the random fluctuations. Once the smoothing was complete and the new data was exported to a CSV file, I plotted it in Excel to observe the changes. The results were solid, as the jagged, noisy line from the salted data was now much more refined and closely followed the original linear trend.

