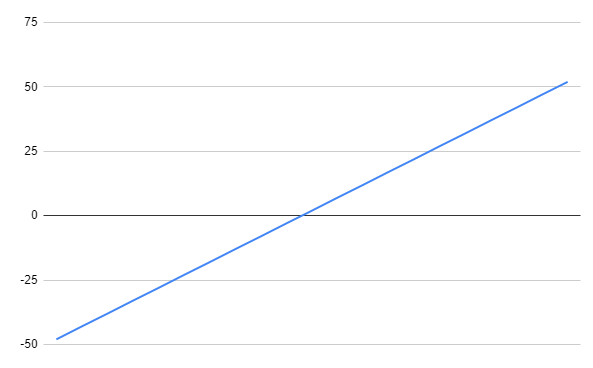
Smoothing, Salting, and Plotting Code

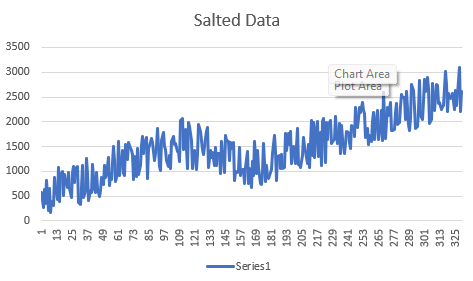
By Nicole Wiechmann

Part One: Original Programs

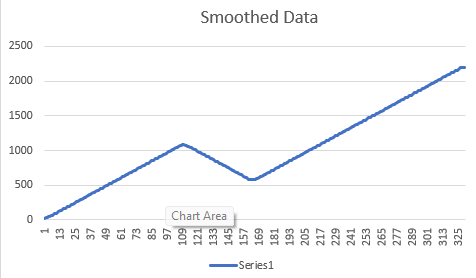
When learning to code a Salter, Smoother, and a Plotter in java, I found the Salter and Plotter to be much easier than the Smoother. Plotter was by far the easiest for me to code and had little-to-no problems when I got to the debugging stage. When working on the plotter, I decided to use the Y = mx + b example from the powerpoint to see if I could get it to work. From there, I chose to let the user input an integer from m and an integer for b. Next, I had to worry about the ranges, so I changed the parameters to let the user input the minimum range and the total range. I went for total range instead of a maximum range, as if the minimum range was a negative number it worked better with the for loop if the program just already knew the full range needed. Lastly, I included an integer interval for how far apart the points would be from each other.



After working on the plotter I started the salter and had a little bit more trouble with it than the plotter. Before this class, I only worked with csv files with python code, as python is my stronger programming language. I did, however, take how you read in a csv file in python and change it to work with java code. Once I figured out how to use BufferedReader and FileReader, I found salter to be extremely easy. I decided on making the parameters be a File to import the csv file, and an int called “range”. With “range”, the method would take whatever the integer was and create a Random object with that range. So if the user imputed a range of 10, the object would print out a random integer within the range 0 - 9.

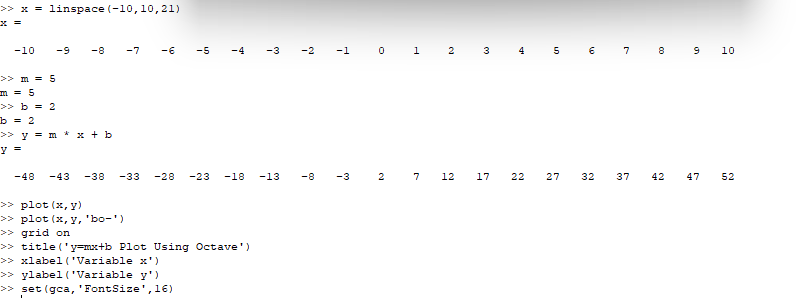


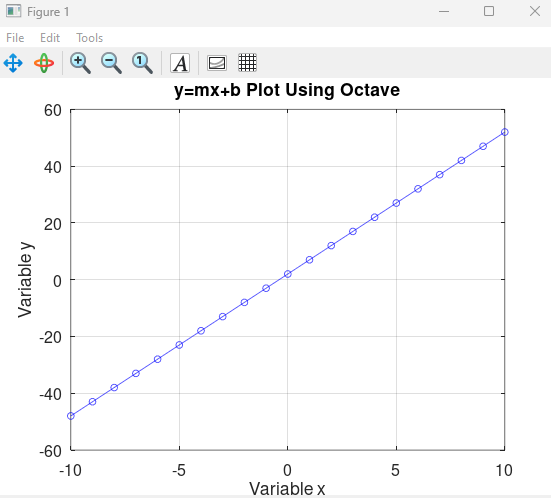
Once I finished up with the salter method, I moved onto the smoother. When it came to the smoother, I had a very hard time trying to get it to work the way described in class. It felt like everything I was trying was coming up with a new error, so I decided to start looking if there was a better way to accomplish making a smoother method. I then realized that a double for loop would probably be the easiest way to make the smoother work. I then changed my parameters to just include the File object csv and an integer called windowValue. WindowValue was used in order to have a range of integers to go before or after a value and add them all together to get the mean. Once it found the mean, it would replace that value in the yValues arraylist.



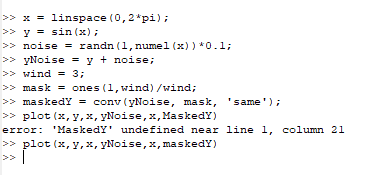
Part Two: Learning Octave and Matlab

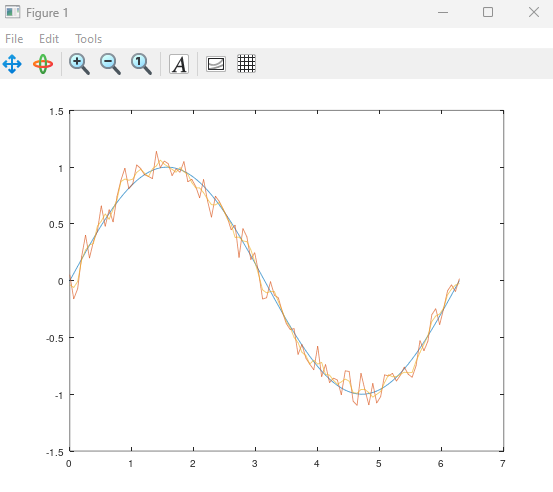
When doing my research into coding with Octave and Matlab, I had the exact opposite problem from before. Smoothing and Plotting were the easiest to find information on, with multiple different youtube videos that went through every step. Salting on the other hand, I could not find any information on. I understand that the assignment was to just learn Octave and Matlab, but I thought what is a better way to learn a new program than to relearn how to program something I'm familiar with. I once again started by learning how to plot data. I mostly used a Youtube tutorial that was roughly 2 hours long and it taught me a lot about plotting, looping, and setting values. I did not have much issues with plotting, the only real difference being the command linspace, which is to create what is similar to an array or matrix of a set of numbers, so for the example below I chose to still use Y = mx + b from -10 to 10.



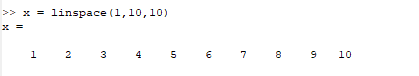


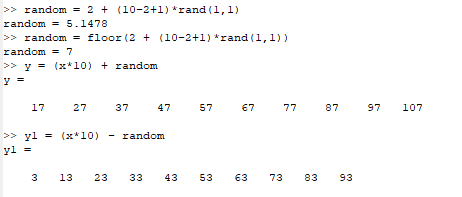
After plotting, I started working on smoothing. I did have a couple more problems with smoothing than I did with plotting, but nothing major. I decided to use data from a different equation this time to better show the actual smoothed data line. In the graph, the orange line is the actual “noise” data created, and the light blue line is the “noise” data smoothed. In order to smooth the data, I mostly went along with the tutorial, as I was confused about what ones and conv were at first. In a way, this graph does show salted data, as “noise” was inserted into the data from the start in order to see the smoothed data much easier. By using ones and conv, the program took the average of 3 points around the data and inserted it into mask.

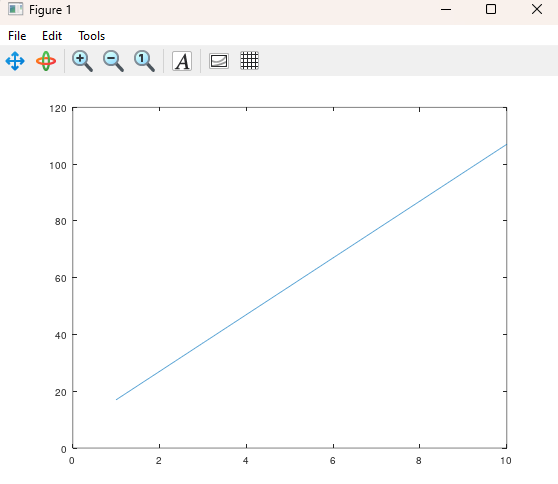


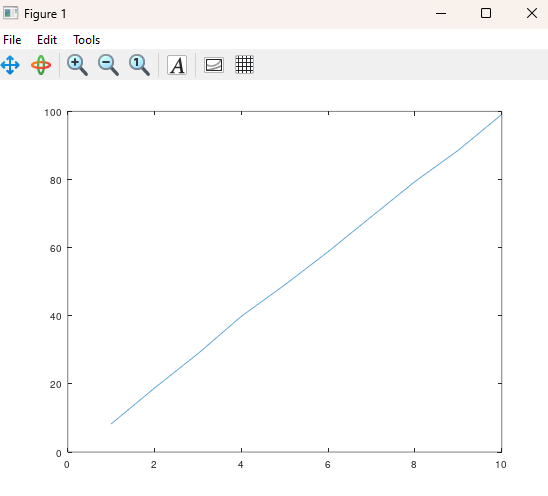


When it came to salting data, I was extremely confused, as I was unable to find any type of tutorial or anything that would help me with that. By this point, I did not realize what I had done in the previous section was the closest I’d get to salting data, so I tried to find other ways of doing it. I also had a lot of trouble with Octave crashing on me for any matrix over 50, so I had to work with smaller numbers. I decided to try and work with the rand function, which is a lot different from its java counterpart. The rand function in Octave takes two parameters, and depending on what numbers you put into it it would create different lines of numbers. For example, if 2 was inputted instead of 1, I got two different lines of “random” numbers.What I ended up doing was multiplying the numbers in the matrix by the random number, the random number that came out was random but it would multiply the same number to all the values. This didn’t necessarily salt the data, but it did add a random element into the equation.



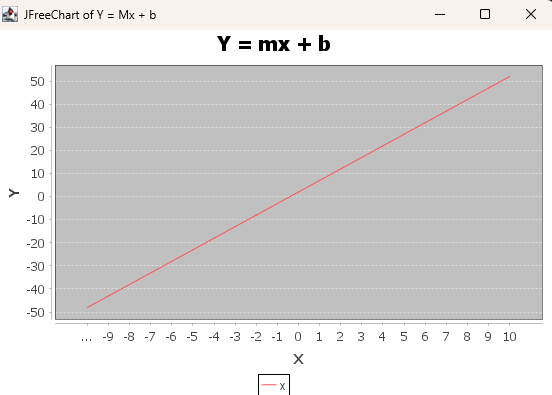




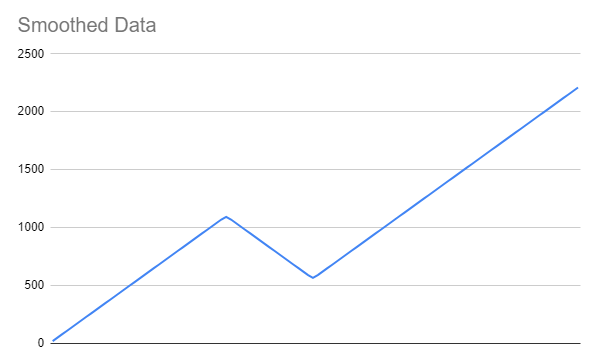


Part Three: Learning Apache and Jars

For me, this was actually the hardest part of the three parts. I started once again with plotting and trying to get JFreeChart into Visual Studio Code. By this point, I had worked with Jar files before, but never in VS Code, and no tutorial was really helping since most of them were from over 5-10 years ago. I should also mention that the JFreeChart website does not take you directly to the actual JFreeChart download so I had to search for that as well. Finally I was able to find one tutorial about adding any Jar files to VS Code and imported all the JFreeChart Jar files separately. From there, I mostly followed a YouTube tutorial on how to create a line chart using the Y = mx + b equation from the other two programs. There was definitely a learning curve with JFreeChart, as I had trouble getting the arraylist of values to print onto the graph in a clean way. For a while, I could not get dataset to accept anything from the xValues arraylist and struggled for a bit on it. Eventually, I got it to work and print out the graph just under the paragraph.



Next I worked on smoother, as I had heard from others that they were having trouble finding a salter in Apache. Downloading Apache was much easier than trying to locate the right download for JFreeChart, and by this point I knew how to import the Jar files. I mostly used the Apache website for help with the coding and found how to code the “rolling mean” for a smoother method. This code, however, only showed how to do it directly from a csv file, and I wanted to keep it simple by taking it from my already created arraylist. I mostly had trouble with this because DescriptiveStatistics would only take a double, and not an integer. What I ended up doing was adding the value into stats as a double, and then changing it back to an int when getting the mean of stats and adding it into yValues. Finally, I had the method write the X and Y values into a separate CSV file.



Lastly, I had the salter to do. I was looking through Apache’s website for about an hour trying to figure out what I could use to salt the data. Absolutely nothing was coming up and whenever I tried to look up a tutorial I came up empty as well. Eventually, I tried to find anything in Apache I could use to try and recreate my code, even if it was just a small part of the code. I once again came up empty, and I figured the reason for this could be because Apache might not see a reason someone might want to salt their data, but once again I’m not sure. I ended up just using my salting method once again, as I felt it did a great job portraying how “noisy” or “salted” data may look.

References:

<https://www.mathworks.com/videos/using-convolution-to-smooth-data-with-a-moving-average-in-matlab-97193.html>

<https://www.youtube.com/watch?v=TqwSlEsbObg&t=4692s>

<https://commons.apache.org/math/download_math.cgi>

<https://www.tutorialspoint.com/jfreechart/jfreechart_line_chart.htm>

<https://commons.apache.org/proper/commons-math/userguide/stat.html>