Plotter, Salter, and Smoother Across Different Programming Languages

Kaden Fuller-Aujla

**1.0** Plotter, Salter, and Smoother using Java

**Background**

The basis of the project was to be able to create a plotter program inside of java and have it write a set of points to a graph.csv file for a given function. Once this was done, we would have to salt the data by reading from the existing .csv file and modifying the y values with random numbers between a specified range and add it to a salted.csv file. Finally, we would smooth the salted data by reading from the salted.csv file and continuously loop through the points and average them to get back to the original graph.csv.

The concept leverages CSV reading and writing combined with being able to create lists of points to make graphs. Then manipulating the data however, we want.

**Programming**

The first part of the project was to lay out a plan to be able to do all these functionalities inside of java, so I broke it up into smaller problems.

*Plotter*

The methodology behind this was to first be able to write data to the csv file once I figured this out by using ‘File’ (to create a new .csv) and ‘PrintWriter’ (to be able to output the data). All I had to do was be able to format the data.

For formatting the data, I chose to make an ArrayList of type Plotter and add the x, y values into it, once this was done all I had to do was iterate through the ArrayList and input the data into the csv file.

*Salter*

The salter was a bit more challenging because it required us to read from a file first to take all the existing data we had plotted. To do this we had to utilize a new tool, and this was the ‘BufferReader’ (reads text from an input stream).

Once this was done salting the data was not too difficult, we had to iterate through the file splitting the points into x. y coordinates. Then store the y value inside of a temporary integer to add the salted random value to it. After this the new value is then re paired with its x coordinate inside of a temporary array and added to the new Salted.csv.

*Smoother*

The smoother was the most challenging piece to code. Since we already know how to read and write from a file, we just need to worry about the logic behind the smoother for this one. We start by reading in Salted.csv to be able to have data that can be smoother (you cannot do this with an already smoothed graph its pointless).

There is one main component to a smoother which is the Window Value, there is another component but that is just how many times we are going to repeat this process.

Let’s take an example to understand what a Window Value is. Pretend you have an array of [ 1, 4, 6, 7, 3, 8, 9] and a Window Value of 2, and you are sitting at index 3 right now. This will take the average of values 1-5 and replace it at index 3. Instead of 7 it will become a value of 5.6 and continuously loop through until you tell it to stop (therefore it will smooth the data back to its original line or curve)

**Results**

The results for this were just as expected. The first program was able to output a list of x and y values so I can graph a function in excel by just highlighting them. Additionally with the salter, it made the graph rougher and had more sharp edges within reason. You were still able to make out which function it was just by looking at it. Finally, the smoother ended up getting back to the original graph with about 95% accuracy. This is a great way to visualize data and be able to manipulate it yourself utilizing the basics of java.

***Figure 1.1(plot)***

Chart, line chart

Description automatically generated

***Figure 1.2(salted)***

Chart, line chart

Description automatically generated

***Figure 1.3(smooth)***

Chart, line chart

Description automatically generated

**2.0** Plotter, Salter, and Smoother using Octave

**Background**

Building on the previous piece of the project we are now creating a plotter, salter, and smoother inside of octave (high level language for number computation). This was much easier than doing it inside of java because the functions for plotting are already built in, so instead of writing to a csv then plotting from there. It would spit out a plot instantly and you can still read/write to and from the csv.

Since I did not know the language, I had to do some research to figure out how to make basic plots, then read and write from a csv. Learning this was much easier than in java because it required anywhere from 1-5 lines, and the way it is written is more straightforward.

**Programming**

*Plotter*

The Octave plotter had about 7 lines of code with about 10 characters per line, it was extremely fast to code. The first step is to set a range of x values, then set a function equal to y. From there you can literally just plot x, y values and then you have the graph done. It is one additional line of code to be able to write to a csv which is just csvwwrite then you specify the file name and range of values.

*Salter*

The salter was about 12 lines of code, and this is because you needed to reshape the data into a matrix to look nice when writing to the Salter.csv file. First you read from the Plotter.csv file then salt the y values and write it back into the new csv. This can be done extremely quickly in Octave.

*Smoother*

The smoother was about 25 lines of code, and this is because of the triple for loop to be able to smooth each y value and repeat a set number of times. Even though this is an inefficient method for smoothing data because of a triple for loop, it still was able to suit my use case. There was nothing new code wise that had to be done here, we had already learned about csvread/write in the plotter and salter, additionally we had utilized the reshape method already. Therefore, we just had to add the logic and then read from Salter.csv and then write to a new Smoother.csv.

**Results**

These results were very consistent with the ones in java. Although this method was much faster in doing so and has a lot of built-in technologies for being able to graph, and output and input to and from files. Overall, I am satisfied with the graphs made and below will include images of them.

***Figure 2.1 (plot)***

Chart, line chart

Description automatically generated

***Figure 2.2 (salted)***

Chart, line chart

Description automatically generated

***Figure 2.3 (smoothed)***

Chart, line chart

Description automatically generated

**3.0** Plotter, Salter, Smoother using JFree Charts and Apache Libraries

**Background**

Essentially is the same as the previous 2 except we are importing JAR files and using them to create charts. These APIs have built in functionality to be able to create graphs much easier and faster inside of java itself. Without having to export to a csv and graph there it is all done at the same time inside of java.

**Programming**

*Plotter*

This was by far the biggest challenge in the project for me personally. It took a long time to be able to figure out how to correctly input the JAR file to the project. Once this was done, I followed a tutorial online which I linked inside of all my files to be able to just plot my points. Using the existing code, I had I was able to take some of that and combine it with the new XYSeries and XYDataset. By doing this I was able to get a function of points graphed with these API.

*Salter*

Once I was able to do the plotter this was almost identical except for when I was touching the y values I could just salt them with an additional 1 line of code. From there I was able to export the same and get a good, salted graph which matched my previous 2.

*Smoother*

The smoother I was not able to do with the existing method I had devised which was a pitfall in the code I created. It was not optimal and very bad, but it was a solution that would at least run. I could not even get the values to smooth here, and I was able to find another way to do it in APACHE but could not use that either.

**Results**

Overall, this was the most useful piece of the project from a real-world application standpoint in java but I could not successfully finish it all which is something I have to go back and continue once I finish the exam. The results were very consistent with the other two methods and look just as good.

***Figure 3.1.2(plot, salted)***

Graphical user interface, chart

Description automatically generated