**Self-Programmed Plotter Report**

Reece Cohen

CSCI 3327, Probability and Statistics

December 11, 2024

**Section 1: The Equation**

A graph of a function

Description automatically generated Before I even start, I would have to find out how to make a CSV and after some searching and testing, I concluded using the File object and making CSV files that ended with “.csv”. After that I then dwelled on what kind of equation I wanted to work with. I originally wanted to use something like a bell curve but after seeing the formula decided to go with something simpler. I ended up with going as it was a simple.

The division of 10 was more for small scale since I believe it makes it look better. This equation will also help with testing bigger numbers since its exponential allowing me to work with larger numbers in a more easily accessible way.

For ease of use I made the Equation as its own class since it is its own niche function and would not fit in with the other methods later. For the method and its functionality, it has configurable bounds and how many data points to be provided. The Class is mainly used to populate an array list with arrays of size two that have the x and y coordinate. When you give it a number of points the method provided with the class would return the previously mentioned array list with the x values being equidistant from each other.

**Section 2: The Plotter**

I wasn’t too worried about plotting since it should have been simple coding practices and for the most part it was, I originally wanted to guarantee that the endpoint no matter the parameters gave you a data point with an x value containing the upper bound but complications came into that where I would get two nearly identical data points at the end of the csv so after some tweaking I decided to rule against having that as a strict requirement as it was causing problems and data isn’t always as pretty as we would like it to be. The main method of making the data points was using an array list of arrays that stored two values x and y, this became confusing sometimes when working with it due to thinking I was working with a 2d array from time to time. Everything considering plotting after this was smooth when it came to loading a tester with parameters and taking the resulting csv into excel and graphing the results

*The graph shown above is a plotted example of from the bounds 84 to 524 (EX1.1)*

**Section 3: The Salter**

When deciding on how to salt given data I wanted to go with the most formulaic way to get the best results for how messed up my data could get so I decided to go with variance as it was as skewed the data could get realistically. How it affected my data set is that I would take the variance of the all the points and make a randomly generated number from -1 \* variance to variance and add the number to the y value at that point. however, the results this provided me with were quite unexpected. I will go into more detail about my expectations on the next section but simply put I think it was too chaotic.

*The graph above is a plotted example of the data that was salted from EX1.1 (EX1.2)*

**Section 4: Smoothing the Data**

This part of the project threw me for a curve for a time. My first iteration of a smoother was taking one point at a time and replacing the y value of the average of the points 3 units to the left and 3 units to the right, if there were not enough units on one side the method would take what it could, initially the outcome of this would leave me with a graph similar to the original plotted one but with the end dangling downwards, my initial reactions to these were that it seemed unlikely that the smoother would have worked so well on the first try and also that it was weird that near the end of the graph it would suddenly start declining. The solutions I came to fix these problems were that for the decline near the end of the graph I had to divide the sum of the numbers by a variable equal to the number of elements taken, instead of dividing by a constant 7. For the issue of why the graph looked identical to the originally plotted graph is that I was passing the originally plotted graph instead of the salted one. The finalized result was more of what I was expecting but I was still surprised at the result

*First iteration of salted graph being smoothed (EX1.3)*

*Second iteration of salted graph being smoothed (EX1.4)*

*Third iteration of salted graph being smoothed (EX1.5)*

The conclusions I have found is that even if I smoothed this graph through 100 iterations the overall shape and form of the graph is irreparably damaged when trying to repair it to its original form and that variance is evil.

**Section 5: Salting and Smoothing (again)**

Since the first salter did irreparable damage to the graph I had to find a different way to salt my data and instead of using variance I used standard deviation since its more controlled considering standard deviations is how far out your data will be spread out on average.

*Salted data of EX1.1 using standard deviations (EX1.6)*

Already from the chart I was more confident as the chart had the salted look but still followed the same general slope of the original graph of EX1.1. After smoothing the graph through 15 iterations this is what it came out as

*15th iteration of smoothing the graph with Standard deviation salting*

While it the concavity of the graph is down instead of up the graph still follows generally the same curve. These results are closer to the original graph than the results shown in EX1.5.

**Section 6: Conclusions**

The experiment here was to create a plotter, salter, and smoother and test whether my expectations were correctly met or not. I was wildly surprised when I first saw the salter (EX1.2) I had made although I knew how far out the bounds were initially, I was still expecting the graph to be repairable. After changing the salter to use the standard deviation for the bounds (EX1.6) the results of the graph were more what I had in mind. Overall, the graph had been smoothed the results were closer to the original which is what I was expecting to have, although there was the unexpected flip of concavity.