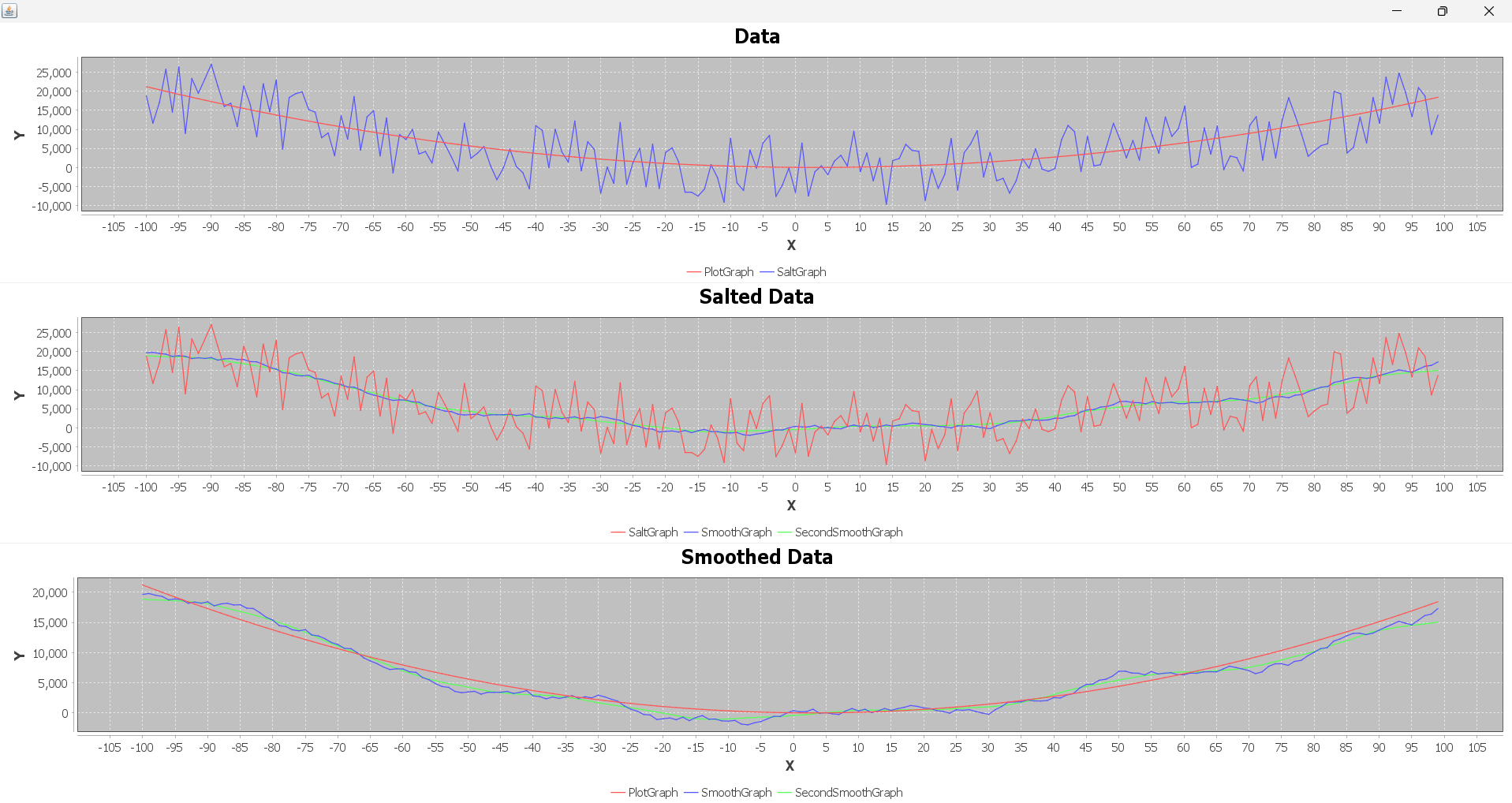
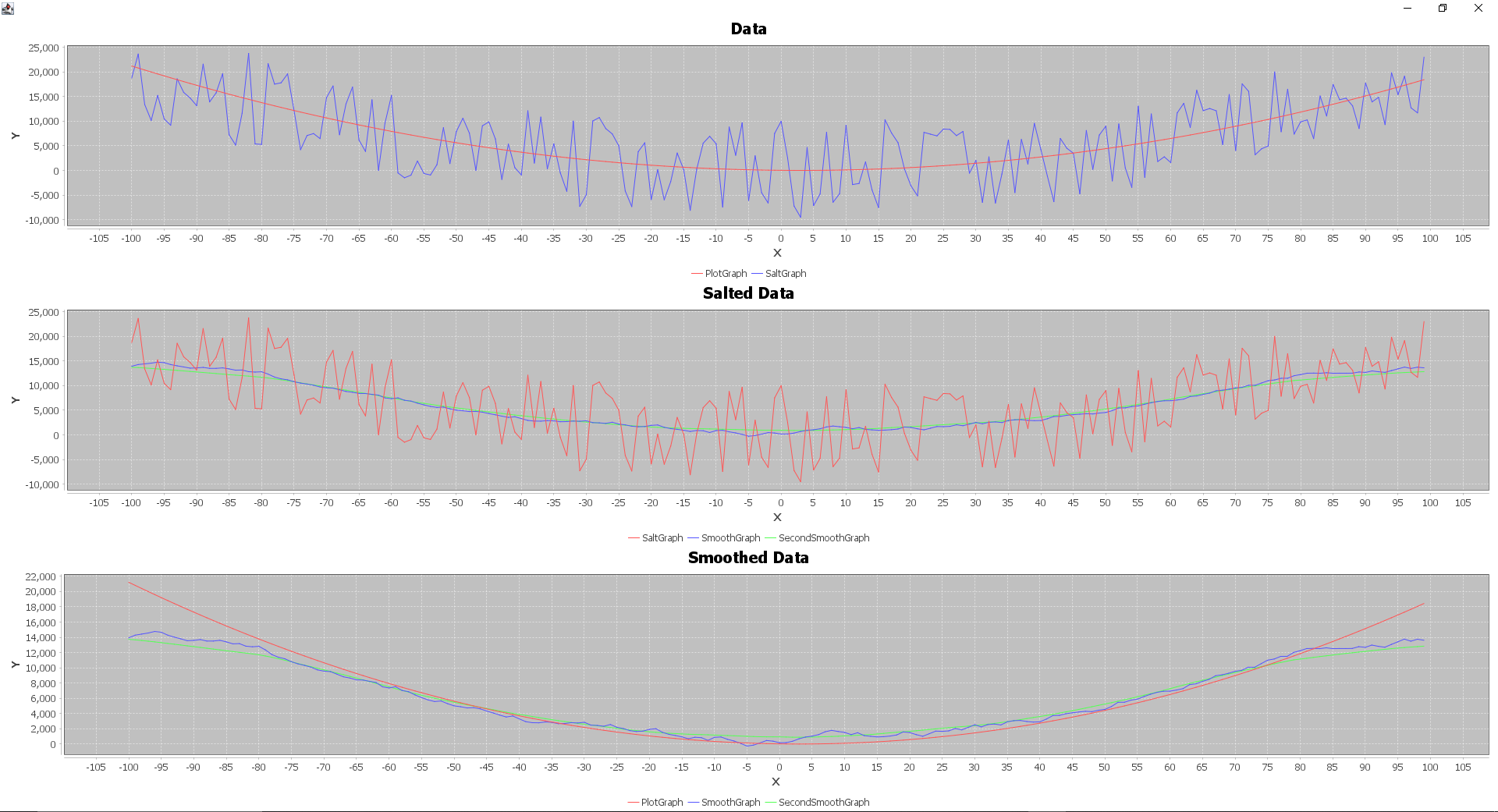
Jaylen Knight  
CSCI-3327-001  
Plotter, Salter, and Smoother report

The java program I have created pseudo mimics how programs like Microsoft Word and Excel Takes the data that is inputted in these programs and encrypts them through salting. A salter’s purpose is to feed the data already present with additional “dummy” data to prevent the file from being properly utilized by other programs. This salted data by itself can’t really be read and needs to be converted back to its original form through a method called smoothing. This generally undoes any salting done to the data to have it brought back to a readable format for an end user.

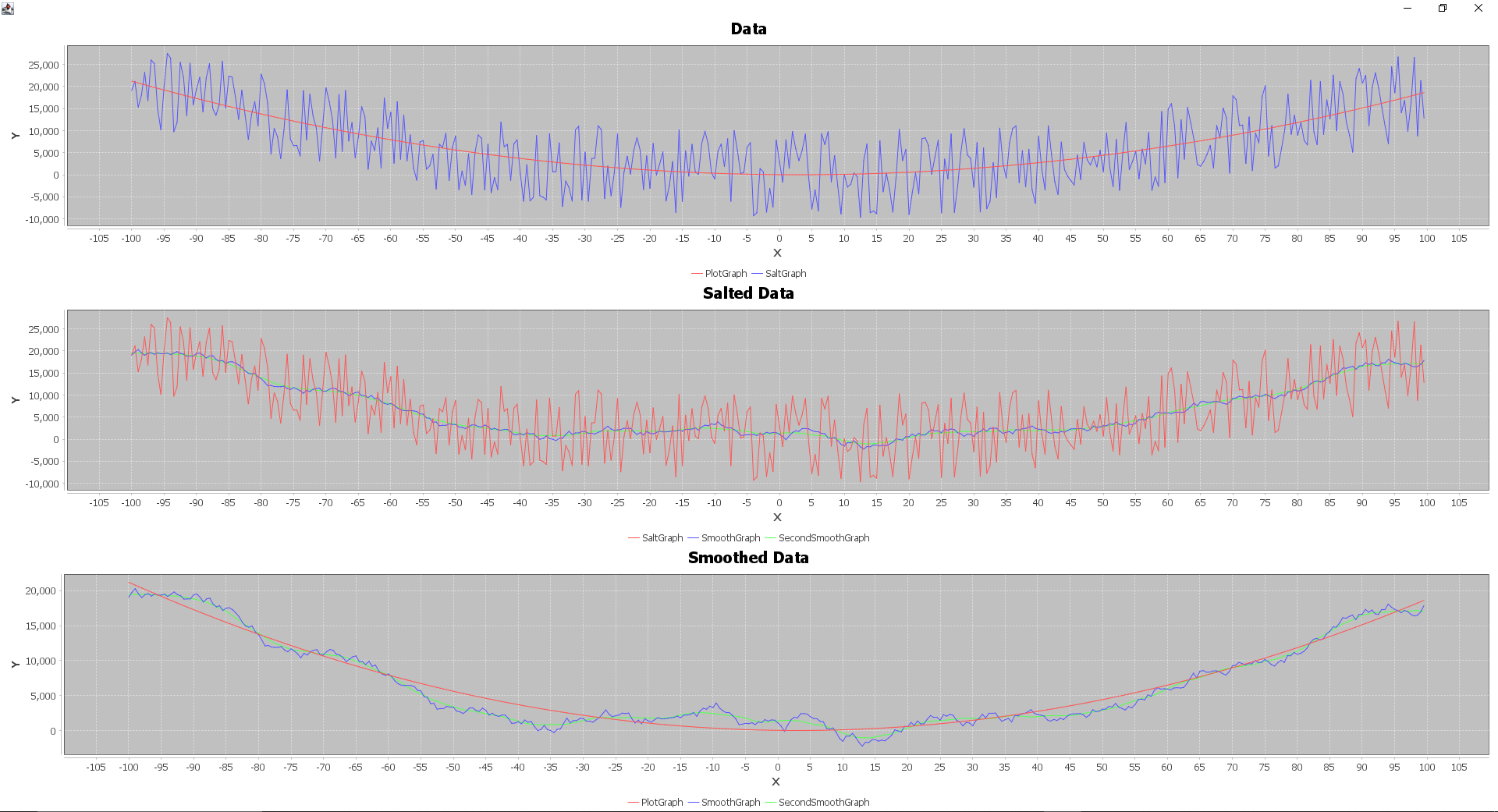
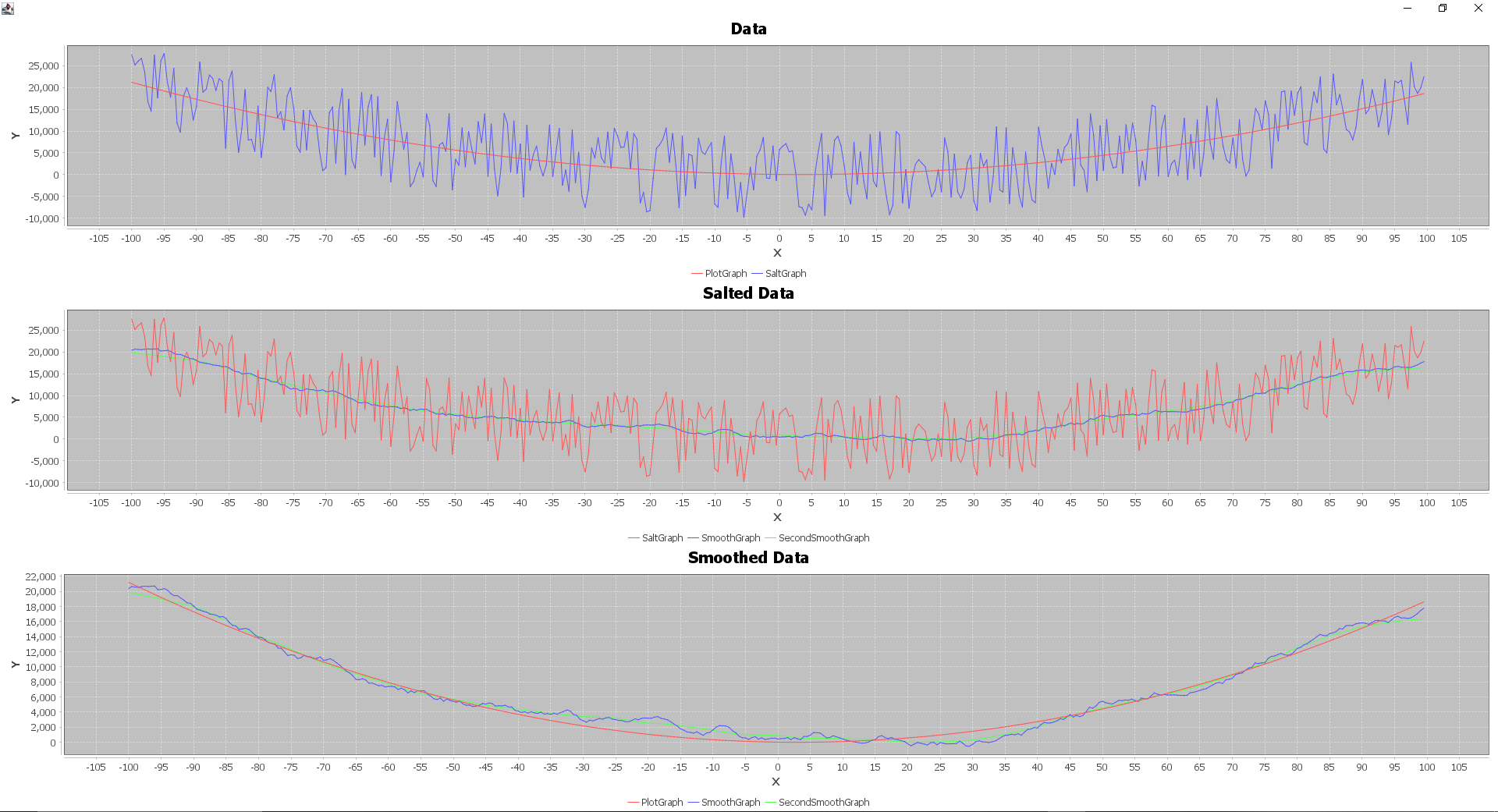
For my program I have simulated this process by graphing a given function with an amount of data points determined by the user. Then randomizing each data point on the graph by adding a number between -10,000 and 10,000 generated randomly for each data point then plotting said data. Lastly the smoothing part of this process which involves employing a moving average in order to average out all the data points by using the points around each point currently being smoothed. The accuracy of the smoother is heavily dependent on not only the range of the moving average but also the volatility of the salter. The following are 3 graphs 1st being the data from the function y=2\*(x-3)^2+4 with 200 points of x ranging from -100 to 100 while also containing the function data salted –10,000 to 10,000, the second containing the aforementioned salted data and said data being smoothed twice over denoted by the blue and green lines respectively using a range of 10 and the third being the 2 smoothing results compared to the original plot.



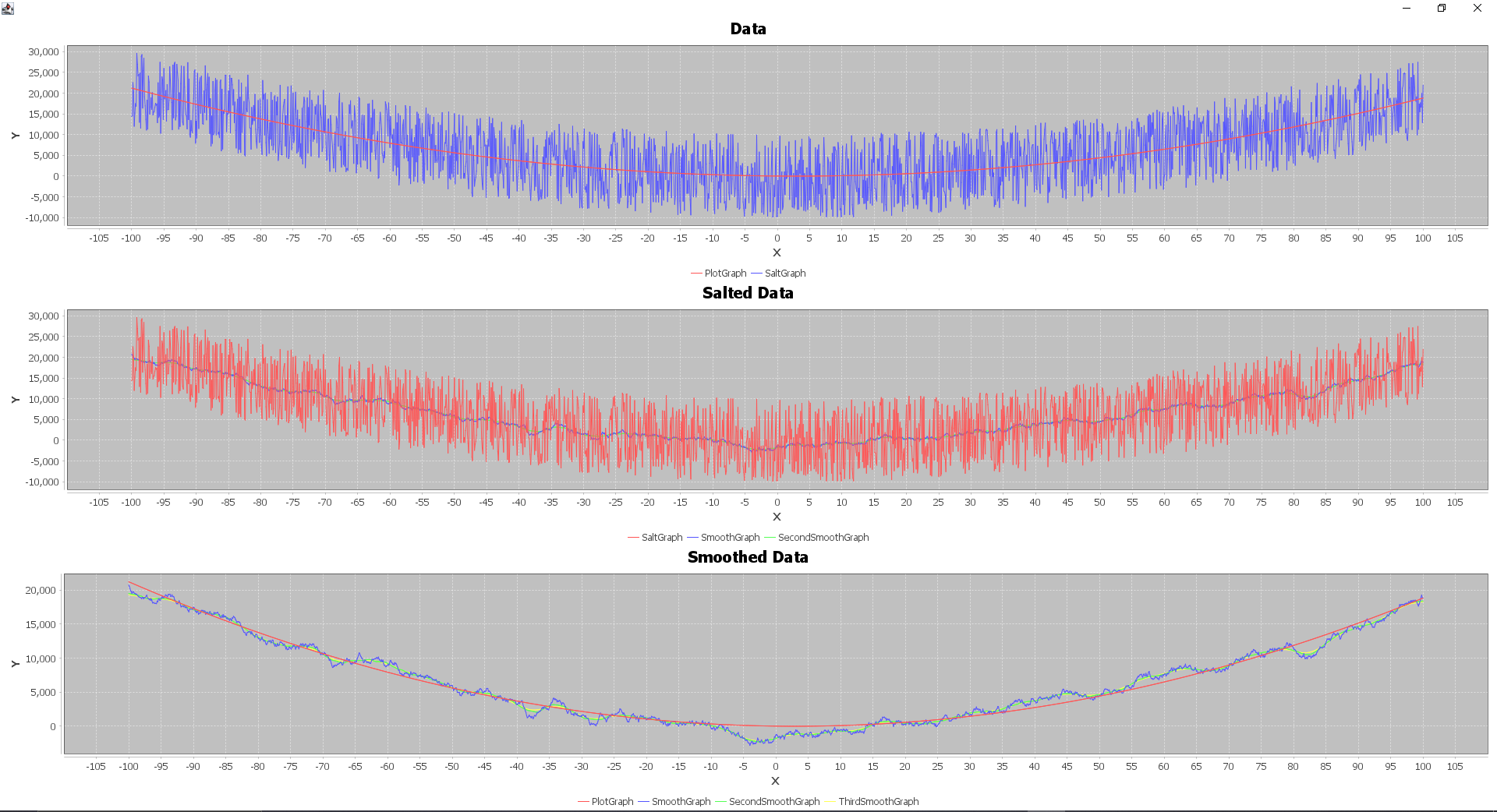
The smoothing of the salted function isn’t perfect but comes significantly closer to the original plot than the salt. Running the 1st smoothed function through another smoothing process results in the green line and while the change between the first smooth and second is not as drastic it still visibly smooths out the rigidness of the previous smooth and gets that much closer to the original equation. Although smoother it still isn't replicating the original equation closely enough which prompted me to attempt and run through the process above once more but this time using a moving average range of 20 and not 10, The results are below



We achieved what we set out to do with a second smooth of almost fully eliminating any rigidness within the graph but now a different problem arises of the ends of the smoothed graphs. For roughly the first and last 20 points on the graph namely when the moving average is at its weakest, the data is largely off form the original function and only corrects itself in the middle of those 2 ranges or in percentage the first and last 10% of the graph is inaccurate while the middle 80% very closely resembles the original function. The reason why the start and end of our graphs are at its most inaccurate has to do with how the moving average works. The moving average takes our range (being 20 in this instance) before and after the point currently being smoothed and averages them out to add to the smoothing point. Our problem arises when there isnt enough points before or after due to being at the start or end of our plot respectively hence why we can visibly see the first 20 and last 20 points on the graph start to trail off. There are 2 solutions I can see for this problem, 1) shrinking the range of our smoother which can lead to other unintended consequences or 2) which I will be doing is increasing the amount of data points present to mitigate the percentage of the graph being off. Now I want to compare these graphs which incorporate 200 total points against using the same algorithm however instead of iterating between –100 and 100 by 1 we will now iterate at a rate of 0.5, doubling our total amount of data points to be salted and smoothed to 400 instead of 200. First with a smoothing range of 10 then 20 respectively.

As we add more data points of data for the smoother to work with the overall curve of the data gets smoother and smoother however that doesn't necessarily mean more accurate to the original function, especially with the start and end of each graph. Although the smoother is given more points to work with, introducing more points overall leads to the original plot becoming more distorted in the salting process as seen by how much more noise or frequency of ups and downs are present for the salter compared to when we only had 200 points in the first graphs. This additional frequency makes the smoothing process that much more inaccurate in the end as evidenced by the middle of our new graphs consistently being above or below our original plot. A more extreme version of this problem will be shown below by making our increments 0.1 instead of 0.5 bringing our total amount of points from 200 originally to 2,000 at a range of 20.



Even running the smoothing algorithm, a third time shown by the yellow line doesn’t give us ideal results as the original plotted function is too varied at this point for this algorithm to correct. Some improvements I can think of for the program would be reducing the values of the bounds on the salter even though it would compromise some of the pseudo encryption would result in the smoothing process being much more accurate in replicating the original plot. Another idea would involve leaving the first 5% or the first half of whatever your moving average range is alone in the slating process to give the smoothing process a better reference of the original data to more closely replicate it. Each subsequent point past the initial 5% would become a more accurate replication giving the moving average more accurate points to work with to produce even more accurate points in relation to our original plot. Another method of improvement would be adding an upper and lower limit for the graph at each point. For example, if your smoothed data is outside of a certain percentage range of the original data then it would round to the nearest range. This would be most effective at the start of the smoothed graph and would massively benefit the rest of the smoothing process as mentioned above by giving the smoother a better reference.