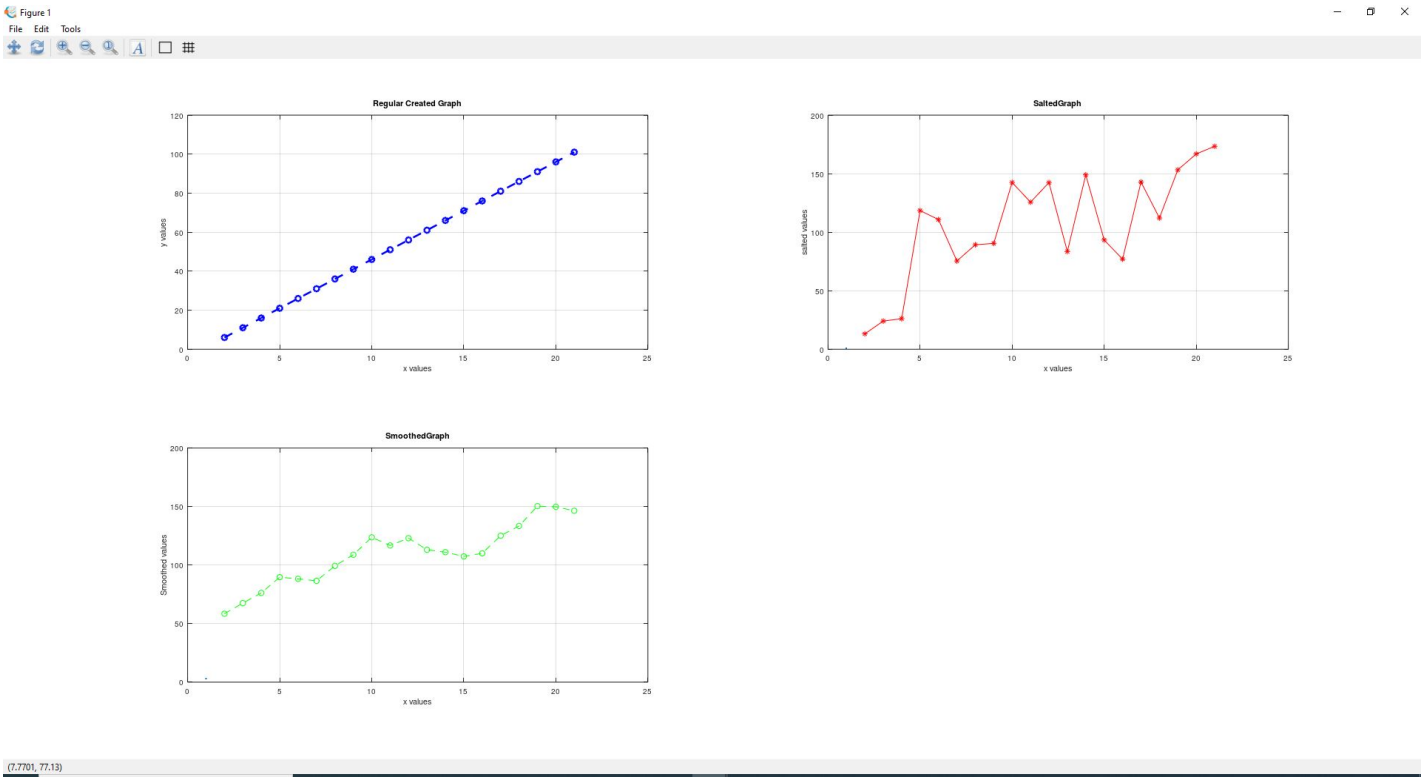
Patrick Niederhauser
Project 2 report

Project 2 consisted of many unique projects that challenged me over the course of 6 weeks. I really enjoyed working on this project and project two as well as project one has helped me improve my coding skills tremendously in a short amount of time.

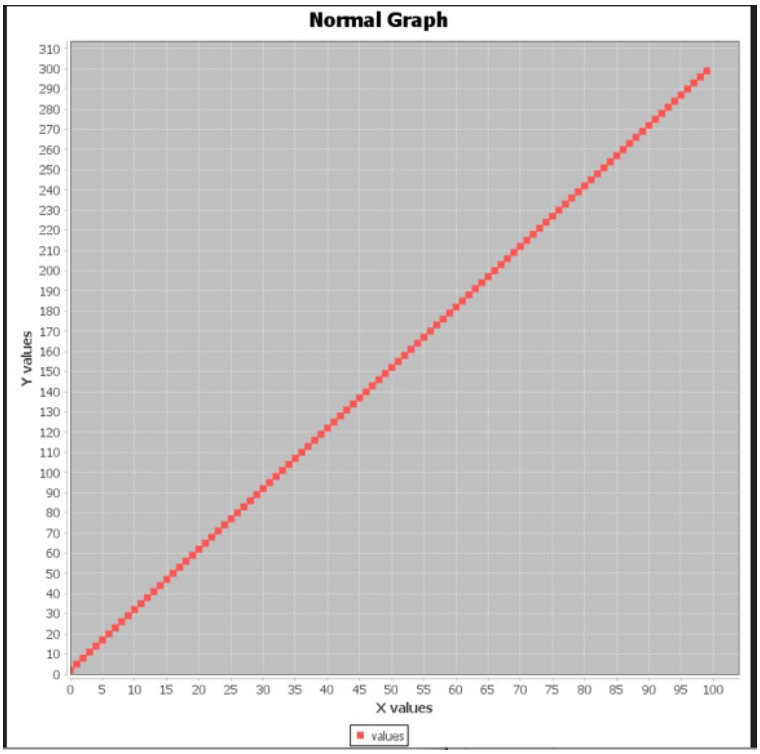
The first assignment that I worked on in this project was the matlab tutorial. This assignment wanted us to follow a matlab tutorial and then create our salter smoother program inside of matlab. I first followed a youtube tutorial which was informative, but lacked concrete examples on how to use matlab properly. Once I finished the youtube video I felt that I needed further tutorials on matlab to increase my overall skill. I was able to locate a pdf file that included a short tutorial as well as two exercises that I can try on my own. The tutorial on this pdf was short, but focused on plotting in matlab which is going to be useful for the salter-smoother program. After completing this tutorial, I began on the exercise which wanted me to use my newly gained knowledge to graph a function. The first exercise gave us the function e^x^2 on the intervals (0,1). To start, I first set my intervals and then solved for my y values. I then just used the plot function to plot my answer. It's very interesting to me how easy it is to plot stuff in matlab. If I were to use java for this project it would be about 150 lines of code, but in matlab it's only 5. The next exercise repeated the above step with a new function, but this time it gave us two functions, which means I had to learn to use the subplot function. The subplot function works well, but can be confusing. It is called by just saying subplot(xxx). The first two x's are the size of the matrix you want and the last x is the spot in the matrix you want this graph to be in. After I figured out the subplot function the rest of the exercise was simple, and I felt confident enough to begin on the salter-smoother program.

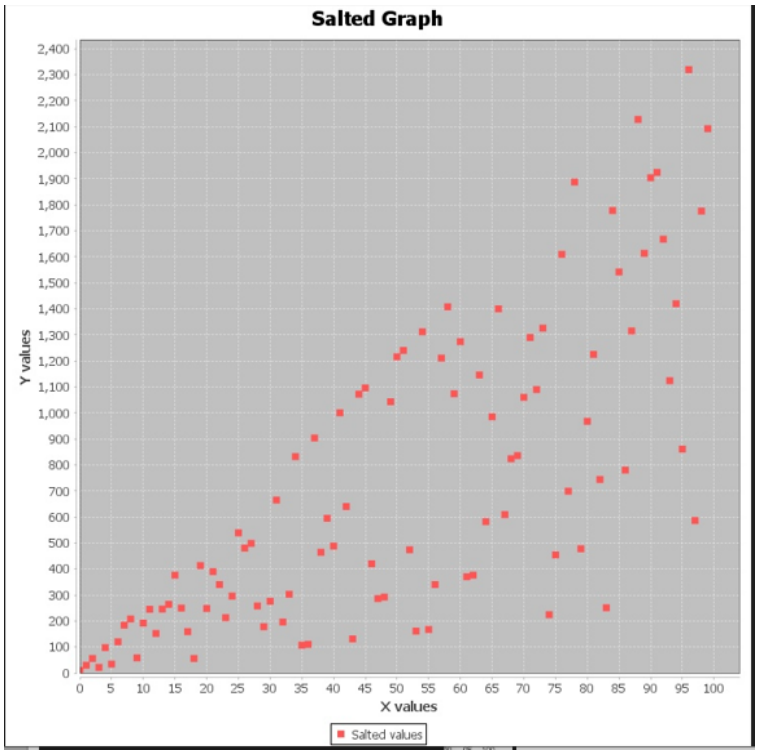
To begin on the salter smoother I first needed to create a regular function and graph it. I first created an array of x values from 1-20, and using the function y=5x+1 I plugged in one value at a time and solved for my y values. Once I had my x and y values I exported to csv and began to plot my values. To plot the function I first set up my subplot, I then used the plot function to graph my x and y values. I added x and y labels to my graph as well as a title, and then called my salter function passing my x and y values through. The salter function took my y values and multiplied each one by a random value from 0-100. I then followed the same steps above, first printing the values to a csv file, and then plotting our x and y values. Everytime I used the plot function in this project I tried to change the layout, so the color, line spacing, and line design is different for each plot. Once I graphed my salted data I called my smoother, passing my x and y values to my new function. My smoother then took each y value one at a time and smoothed the data out. It used the 5 closest points to each y value to find the running mean and smooth our data. Once we had our smoothed data I again called our plot function and graphed our data. I included a screenshot of what my graphs looked like below:

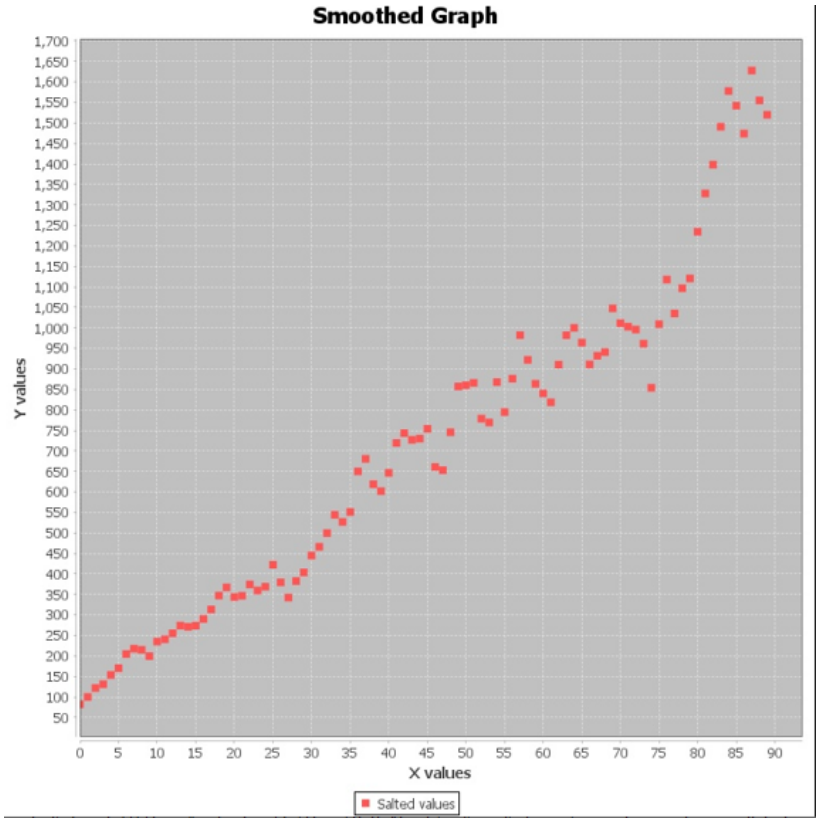


The next program I worked on was the java salter smoother program that used JFreeChart and apache. To start this project I first had to research how to add libraries to eclipse. Once I figured out how to add these libraries I then started on the actual project itself. To start I wanted to just graph a regular problem using the formula y=mx+b on a scatter plot. To do this problem I needed to understand JfreeChart a little so I started my research off by looking up scatterplots. Scatter plots use data sets, so I started researching datasets, which led me to series. Series work in tandem with data sets to store data and are the JfreeChart version of an array list. To start this project I declared a multiple series and datasets to store different data points in my class. The first method is called graph, this method is going to just plot a normal graph. I first took user input that would determine how many points this program is going to plot. I then ran a for loop that calculated values using the formula above and populated my first series called values. I then passed this series to a data set, which then allowed me to call the chart factory method from jfreechart to graph my data. I next wanted to salt my data which can be done with a simple for loop that just multiples each y value by a random integer. During this for loop I added each salted value to my series, which I then passed to my data set. Similar to the above method I just then passed these values off to the chart factor, which then graphed the data. The final method was called smoother, this method is going to use an external apache library to smooth my salted data. I started this method by declaring a double array, although I don't like using arrays of type double; this is what apache likes to use. I next had to pass my salted y values from my above dataset to my new double array, i did this by using a simple if statement. I next ran a for loop that calculated the mean of each point within a 10 point window. By researching on apache website i found a method that will find the mean of a window of x

amount of points. This method takes a double array and finds the mean from an index to an index, by using this I was able to smooth my data. I had a little issue on this method that causes my data to miss 10 x values, because of bounds issues. I could have lowered my mean function from calculating 10 values to calculating a lower number which would have given me more x values, but would have produced a worse graph. After this for loop i then just again called the JFreeChart chart factory again. My graphs are displayed below.







The final project I worked on was the poker monte carlo simulation. This program runs a poker simulation 10000 times and displays the probability for multiple outcomes. To start this program I first created a deck class that would create a card object and add 52 of them to a deck. These cards must consist of 13 different numbers with 4 different suits. By using a collection of if statements and for loops I was able to create a createDeck() method that would create a deck array of 52 card objects. From here I then created a souffle deck method that just shuffled the deck 1000 times. After I had my deck created, I made a class called poker that would simulate 100000 games of poker and evaluate each hand of 5 cards. To do this I first created a method called river that just took 5 cards from our shuffled deck. After I had our river I then created various methods to determine if our river included: a pair, three of a kind, four of a kind, straight, flush,straight flush, or royal flush. By using a collection of if statements and for loops I was able to pass each array list to each method and return true or false if the method decided we had one of the situations listed above. I then ran this 100000 times and counted each time something appeared. I then took these counters, divided them by the amount of runs and displayed my analytics in the consol. The below image is the answer I got:

```
Our pair odds are: 47.179%
Our three of a kind odds are: 2.157%
Our four of a kind odds are: 0.026%
Our flush odds are: 0.211%
Our full house odds are: 0.145%
Our mulitple pairs odds are: 4.875%
Our straight odds are: 0.404%
Our straight flush odds are: 0.002%
Our Royal flush odds are: 0.001%
Our nothing probabilty is: 49.854%
```

The next class I worked on was the game simulation class. This class would simulate a game of poker, where each hand had 5 cards and we are comparing each hand. To do this method I created a shuffled deck and then created two array lists, one called hand one and the other called hand two. I then took 5 cards from our deck and passed them to our river. From here I gave the option to the user to dump or keep a certain amount of cards. Depending on the user's input the hand can change for the better or for the worse. After this I then ran a similar evaluation as above, but this time I need to add a value to each possibility. For example a pair was one point, two unique pairs is 2 points, three of a kind is 3 points, a straight is 4 points, flush is 5 points, full house is 6 points, four of a kind is 7 points, a straight flush is 8 points, and a royal flush is 9 points. This means that each possibility has a ranking system, and determining hands would become as easy as just comparing two values. Once I ran the evaluation I determined what hand had the higher value and then printed that to our console. My answer is displayed below.

```
hand 1 is :
[5, spades, King, spades, King, clubs, Queen, spades, 7, hearts]
hand 2 is :
[5, hearts, 6, spades, Ace, spades, 2, hearts, 4, clubs]
Enter the amount of cards you want to replace for hand 1:
0
Enter the amount of cards you want to replace for hand 2:
3
our new hand one is:
[5, spades, King, spades, King, clubs, Queen, spades, 7, hearts]
our new hand two is:
[6, spades, 2, hearts, 6, hearts, 5, diamonds, 6, clubs]
Hand 2 wins
```

```
hand 1 is :
[7, spades, 2, spades, 4, hearts, Jack, spades, 9, spades]
hand 2 is :
[Jack, diamonds, 4, clubs, 3, hearts, 10, clubs, 8, diamonds]
Enter the amount of cards you want to replace for hand 1:
2
Enter the amount of cards you want to replace for hand 2:
3
our new hand one is:
[2, spades, Jack, spades, 9, spades, King, clubs, 8, clubs]
our new hand two is:
[4, clubs, 10, clubs, 3, clubs, Jack, clubs, 7, clubs]
Hand 2 wins
```