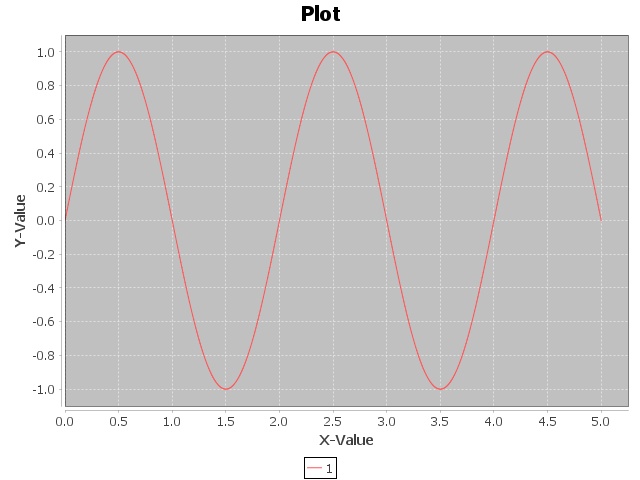
**Third Party Plotting**

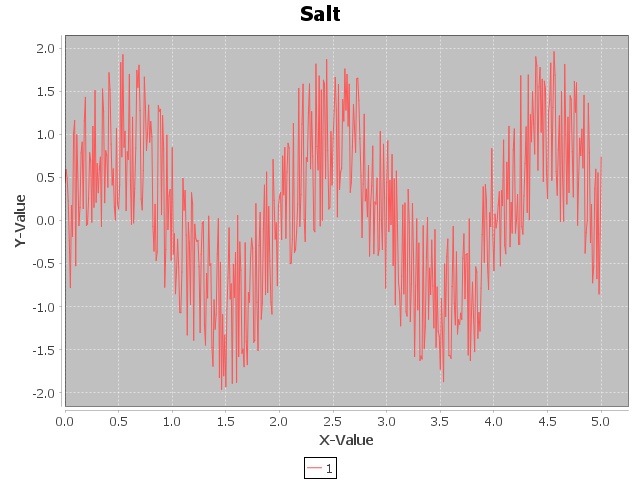
**Nicholas Wilkinson**

Recreating the plotter, salter and smoother using JFreeChart and Apache Common Math was harder than implementing it into Octave, as I had to shift through multiple resources to try and understand what I had to do. The salting took the least amount of time, as all I had to do was slightly modify the original code. The plotter took a bit longer, but the smoother took a long time to find what I had to do. However, when I did find out what to do, I got it done quickly. The programs are stored in the ‘Graphing2’ under ‘src’.

For the plotter, I started by creating a sine wave that uses user input to determine the start and end time. I created a data point for every hundredth x-value. I stored the data in an XYSeries, then added the XYSeries into an XYSeriesCollection. Lastly, I made a JFreeChart to create a .jpeg file to contain the graph. All of this is done in the constructor. I made methods to create the jpeg file and another to return the XYSeries. For the graph below, I started at ‘0’ and ended at ‘5’.



The salter was programmed the same as I had programmed it before, but instead it took a XYSeries and salted the data from that instead. Like the plotter, the salting is done in the constructor, and it has an XYSeries and a salting value as parameters. Additionally, I made the process of printing the graph into a jpeg and returning the XYSeries as their own methods. The process of salting was the same as what I had programmed before without the use of third party libraries. For the graph below, I had ‘1’ as the salting parameter.



The smoother, however, took much longer. I tried many different Internet searches, but nothing gave me a clear answer on how to create a moving/rolling average with Apache Common Maths. Initially, I tried using the HarmonicFitter. Even after reading and implementing it, I am still unsure of what it does, but clearly it wasn’t smoothing the data. Afterwards I looked at the Kalman Filter, but it quickly became apparent that what it was doing was either far beyond or nowhere near what I was looking for. I finally found an article asking the same question I was, and one of the replies used the DescriptiveStatistics class. After a quick look, I was unsure if it was exactly what was expected of us to do, but seeing no alternatives I went with it. In the constructor, with an XYSeries, window size (if the window size is even, the program increments it by one), and number of smoothing runs as parameters, I utilized a DescriptiveStatistics to aid the smoothing process. The DescriptiveStatistics acts like a queue with a limited size, so I simply added data values from the XYSeries input to the DescriptiveStatistics and used the mean function to get the moving/rolling average. When I reached the end of the data and needed to shrink the window, I had to flip the data, since only the most recently added element can be removed without adding a new element, and then removed elements for the last few data points instead of adding new ones. The inability to replace data in the middle of the DescriptiveStatistics meant I could not replace values as I smoothed them, so I opted to instead run the smoothing a number of times as specified by the parameter. Like the plotter and salter, I added methods for printing the graph into a jpeg and returning the XYSeries. For the graph below, I had the window set to ‘25’ and the number of smoothing set to ‘4’.

