Using jar files for Plotting

Written By: George Camac

Import Libraries

The first portion of this project consisted of downloading jar files and using the imported libraries to create a graph within the java code. The imported libraries at hand were Apache Math Commons, as well as JFreeChart. Most of what I struggled with was the importing of the jars without the use of maven or other builders. They way I decided to take it was manually adding the path’s to json files as well as adding a lib folder to house my jar files. Once imported and the paths were added and configured to allowing me to locally import the needed imports.

A screenshot of a computer

Description automatically generated

**Here is where I imported the jar files into a lib folder, as well as adding the paths to the jars in my json file.**

Plotting with jarPlotter

When originally going to plot with the jarPlotter is was quite similar exporting to csv files. The main difference was figuring out how to properly use the syntax and usage of jFreeChart’s and Apache’s libraries. However, in all reality it wasn’t that difficult to follow the code using apache commons since it was so like other Math functions it was quick and easy to grasp. However, when it came to using jFreeChart that took a little longer to figure out the Frames and sizing in which I mainly referred to the jFreeChart tutorial found online giving syntax and code. The formula I decided to go with since provided by Apache was cosine and sine. Here are examples of the outputs I got from my Plotter.

**Figure 1-1 plotting sine and cosine from -20 to 20 with 100 points**

**A graph of a wave

Description automatically generated**

**Figure 1-2 plotting sine and cosine from with 75 points**

**A graph of a wave

Description automatically generated**

**Salting**

The salter I created is meant to alter the code in which is received from the graph. This is meant to obscure the data to make it ineligible to those without a smoother to view the data. One issue the salter has is that it initializes the graph entirely within the same method rather than calling a graph builder which could definitely be improved for future code. My code refers to calling a random function in which will choose and value from the salt value that you provide within the parameters to construct a proper salt value for each point on the graph. It will go through the for loop to properly salt every value of the array that is then added to a series to be graphed. Here are some examples of those graphs.

**Figure 2-1 salting plot from Figure 1-1 with a salt value of 20**

**A graph showing a wave

Description automatically generated with medium confidence**

**Figure 2-2 salting plot from Figure 1-2 with a salt value of 13**

**A graph showing a graph of waves

Description automatically generated with medium confidence**

**Smoothing**

My smoothing method is meant to be used to try and “decode” or “smooth” the data that really is just trying to find the average step in between the values to guess the expected output. In my jar program I didn’t use any new type of smoothing besides using a sliding window pane which will guess the step bases on the surrounding values of those on the graph. Similar to the salter an improvement that can be made on this is by separating methods and possibly classes for the different method making it easier to call one another for graphing or salting to a specific value rather than the given local values that can’t be altered unless it is from the original graph itself.

**Figure 3-1 Smoothing salted plot from Figure 2-1**

**A graph with red and blue lines

Description automatically generated**

**Figure 3-2 Smoothing smooth plot from Figure 3-1**

**A graph of a graph

Description automatically generated**

**Figure 3-3 Smoothing salted plot from Figure 2-2**

**A graph with red and blue lines

Description automatically generated**

**Figure 3-4 Smoothing smooth plot from Figure 3-3**

**A graph with red and blue lines

Description automatically generated**

**Figure 3-5 smoothing smooth plot from Figure 3-4**

A graph showing a graph of waves

Description automatically generated with medium confidence

As seen throughout running the Plotter, Salter, and Smoother we find that this isn’t a perfect program that will return the values back to its original state, but rather tries to mimic the curve of the series as well as it can from its surrounding values. Even though a lot of functions were used from preexisting libraries more could be added to try and improve all functions of the program with making it more precise, smoother, and or faster. Things like a different more accurate way of assuring the smoothed graph is able to return to its original states. However, using these jars definitely made it easier to view graphs and see the differences with less effort than individually adding graphs from csv files.

**Citations**

YouTube. (n.d.-a). *JFree Chart Graphing in Java with Arduino*. YouTube. https://www.youtube.com/watch?v=dPJndZjzyBY&t=984s

Tutorialspoint. (n.d.). *JFreeChart tutorial*. Tutorialspoint. https://www.tutorialspoint.com/jfreechart/index.htm

BoostMyTool. (n.d.). *Add JAR files to Java Project using Visual Studio Code*. YouTube. https://www.youtube.com/watch?v=3Qm54znQX2E&t=41s