Mia Watts April 25, 2024

**Probability and Applied Statistics Final Project**

**JAR/API Plotter, Salter, and Smoother**

After programming a Java plotter, salter, and smoother from scratch, another part of the project required a plotter, salter, and smoother to be made from external APIs. The external APIs and .jar files used for this section of the project came from JFreeChart and Apache Math Commons. These downloads can be found at the following links:

* <https://commons.apache.org/math/download_math.cgi>
* <https://sourceforge.net/projects/jfreechart/files/>

When downloading the Apache Commons Math files, select the binary files rather than the source files. The binary .zip download was where the .jar files were placed. No .jar files were found after downloading the source files for this project.

**JFreeChart**

JFreeChart is a site / collection of programs and .jar files that allow users who download the files to graph results in GUI formatting. Using numerous class imports, like XYSeries, JFreeChart, and ChartFactory, programmers can create their own graphs by adding data to a series, creating a graphable object, and initializing it with x-axis labels, y-axis labels, and a title.

In creating the program using JFreeChart, previous methods were modified to make the plotter, salter, and smoother work with the new imports. Further details on these methods can be seen below the graphed results. The plotter method for all three formulas was changed to support an x-value and a y-value, iterating from -50 to 50 and calculating the result for each x-value. The salter method for all three formulas was changed to support the same thing with an added salting section that would either add or subtract values to or from the original values and save them into an array. The array of salted values is then used for the smoother, where the array is traversed and smoothed, each value being added to the corresponding series. The results for each formula can be found below. *Note: There is an issue with the smoother function specifically for this part of the project. The smoother is fine normally but cannot print the proper line for this part.*

**Polynomial JFreeChart Plotter, Salter, and Smoother Results**

The first formula that was selected was 0.1x2 – 0.5x – 2. The other two equations were done similarly, minus the equation that was used for calculated. After plotting, salting, and smoothing with the help of external .jar files that were added to the IntelliJ project structure libraries, the graph looked like this:

A graph with blue lines

Description automatically generated

**Sine JFreeChart Plotter, Salter, and Smoother Results**

The second formula was sin(2x), and the results for this plotter, salter, and smoother are:

A graph with a line graph

Description automatically generated

**Cosine JFreeChart Plotter, Salter, and Smoother Results**

The third formula was 3cos(x) – 5cos(2x) – 2cos(3x) – cos(4x), and the results for this plotter, salter, and smoother are:

A graph with a graph of data

Description automatically generated with medium confidence

**Method Descriptions**

*For more information on how the original plotter, salter, and smoother work, please see the Java program documentation file.*

* jPolynomialPlotter() – Plots, salts, and smooths data using the JFreeChart graphs. Creates a graph that has three series, or three lines: one for the salted data, one for the smoothed data, and one for the original data. The plotter is a smaller version of the original, as it does not include file writing. The salter is similar to the original but once again has no file writing. These methods, including the smoother, were all changed to work with the JFreeChart version of the program. This works with the polynomial formula.
* jSinePlotter()– Plots, salts, and smooths data using the JFreeChart graphs. Creates a graph that has three series, or three lines: one for the salted data, one for the smoothed data, and one for the original data. The plotter is a smaller version of the original, as it does not include file writing. The salter is similar to the original but once again has no file writing. These methods, including the smoother, were all changed to work with the JFreeChart version of the program. This works with the sine formula.
* jCosinePlotter()– Plots, salts, and smooths data using the JFreeChart graphs. Creates a graph that has three series, or three lines: one for the salted data, one for the smoothed data, and one for the original data. The plotter is a smaller version of the original, as it does not include file writing. The salter is similar to the original but once again has no file writing. These methods, including the smoother, were all changed to work with the JFreeChart version of the program. This works with the cosine formula.

**Apache Math Commons**

Apache Math Commons is a website that contains downloadable files that can be used for statistical purposes. Specifically, for the case of this project, the mean and sine methods/imports are both used. After adding the .jar files to the IntelliJ project structure libraries, the methods and classes associated with Apache Math Commons can be imported. These make the calculations for plotting, salting, and smoothing easier in some cases. The formula used for this part of the project is sin(2x + 1).

For the plotter method, Apache Math Commons Sin is used. The formula is needed to plot the results, so the formula includes the new version of Sin instead of Java’s Math.sin. For the salter method, Apache Math Commons RNG is used. A random number is generated using the Apache Commons imports rather than java.util.Random. Lastly, for the smoother method, Apache Math Commons Mean is used instead of manually calculating the mean. The left sum, right sum, and current value are added to an array which is then passed to the mean method. Together, these do the same work as the Java implementation of the plotter, salter, and smoother, just with Apache Math Commons imports instead.

**Apache Math Commons Plotter, Salter, and Smoother Results**

**Plotter**

**Salter**

**Smoother**

*Note: This is with a window of eight. As shown towards the end of the graph, the line looks similar to the original. The first part is quite erratic, but this may be due to the nature of the salting method.*

**Master Graph (All Lines Together)**