Introduction to MATLAB

Although I had previously used other computer algebra software, such as Maple, I had never used MATLAB prior to this assignment. To familiarize myself with the program, I first followed the tutorial provided by MATLAB itself, which is easily accessible via a link that appears on the homepage when MATLAB is first started. The tutorial was fairly short, but it covered the most important and commonly-used features, and also provided many links along the way where they have more detailed tutorials about particular topics. For my purposes, it was important that the tutorial covered the syntax related to plotting graphs and visualizing data, which it did. After completing the tutorial and playing around with the various features I had learned about, I still felt like I didn't have even the minimum amount of requisite knowledge of MATLAB to even attempt to do anything serious (the feeling could be compared to walking on ice with ordinary shoes), so I followed a more in-depth MATLAB tutorial on LinkedIn Learning. [LinkedIn Learning is an educational site that features many different topics and courses. It is similar to YouTube but is much better organized and has none of the annoyances of YouTube, such as ads. Also the people teaching the material are usually fairly prominent and experienced in their respective fields.]

To demonstrate the tiny bit of knowledge I gained from the tutorials I followed, I will reproduce the plotting, salting, and smoothing of data that I did in project 1 using Java.

We begin by defining the the interval over which we will plot all the graphs. We will use the interval [0, 1] with 1000 equally-spaced points:

```
x = linspace(0,1,1000);
```

We then define the original function, y = |sin(5x)|:

```
y = abs(sin(5*x));
```

We next add noise to the original graph by the following procedure: for each value y in the original graph, we generate a random real number in the interval (0,1), multiply it by 2, and subtract 1. This results in each y value being shifted into the interval [-1,1].

```
salted = y + (2*rand(1,1000) -1);
```

To smooth the data, we use the convenient and appropriately-named inbuilt MATLAB function SMOOTHDATA:

```
smoothed = smoothdata(salted, 'gaussian', 50);
```

The parameter 'gaussian' is one of the types of smoothing methods provided by MATLAB, and '50' is the window size. [To be perfectly honest, I am not quite sure of the differences between the various smoothing methods, and I would probably need to invest a significant amount of time to familiarize myself with the nuances of each. Since this one seems to accomplish the task I was striving to achieve, I decided to use it.]

We then plot the three graphs in one window to be able to compare them at a glance.

```
% set the layout to 3 rows and 1 column
tiledlayout(3,1);

% place the first graph
nexttile

% plot the first graph using dots in a blue color
scatter(x,y,"b.")

% give it a title
title('Original data')

%{
The first two arguments set the range for the x-axis
and the second two arguments set the range for the y-axis.
The arguments for the y-axis ensure that all the data points
appear on the graph, because the default range for the axes
could cause the window to cut out some of the data points.
}%
axis([0,1,min(y), max(y)])

% label the axes
xlabel('x')
ylabel('|sin(5x)|')

% repeat for the other two graphs
nexttile

% use a red color for the second graph
scatter(x,salted, "r.")
title('Salting the original data')
axis([0,1, min(salted), max(salted)])
```

```
% use a magenta color for the third graph
nexttile
scatter(x, smoothed, "m.")
title('Smoothing the salted data')
axis([0,1, min(smoothed), max(smoothed)])
```

The resulting graphs appear as follows: