Tutorial

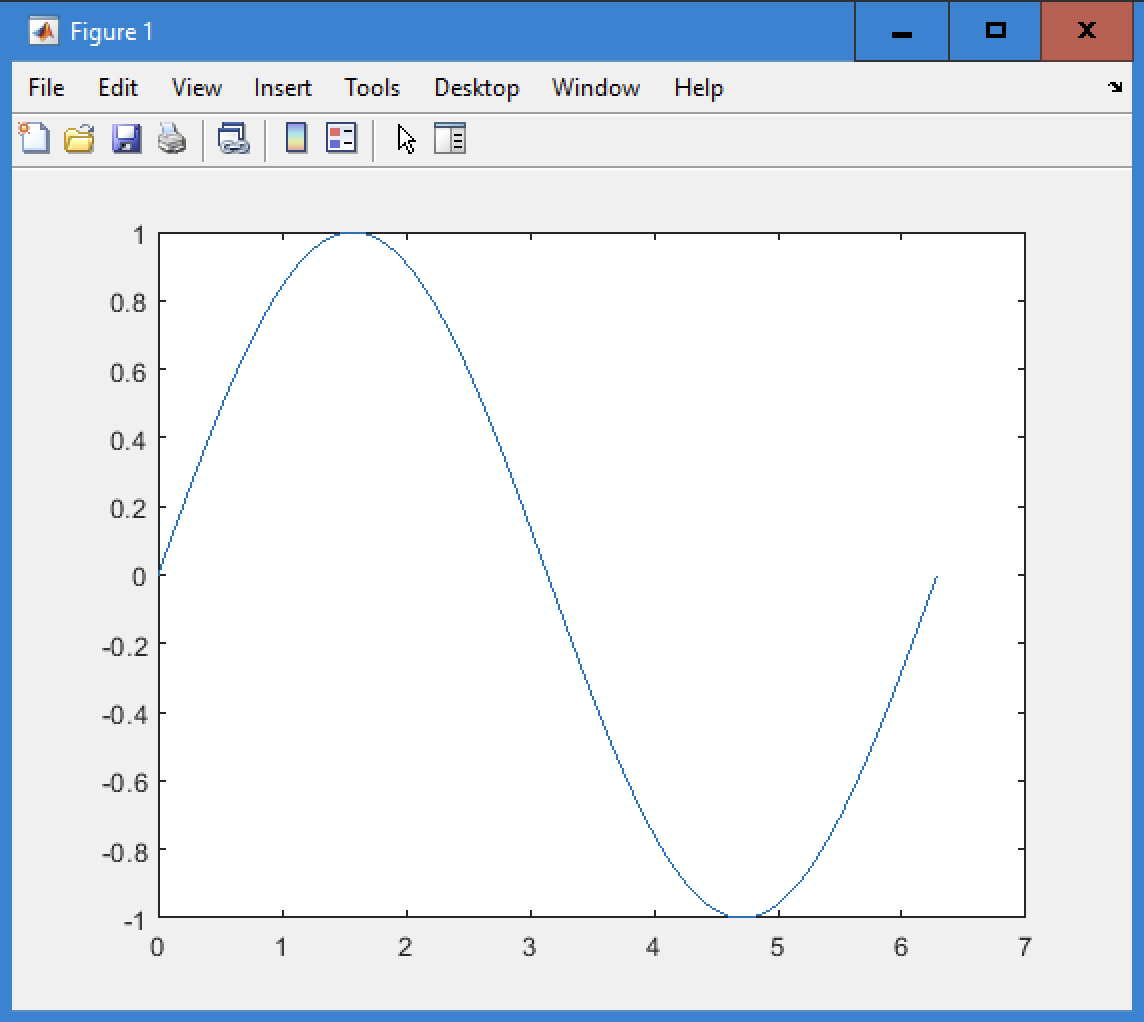
Step 1: Plotting

For this section of plotting, salting, and smoothing, I will be using MATLAB to demonstrate these three graphing methods. For the first chart, we will be assembling a normal graph to depict a sin wave function. By using the commands below, we will be able to construct our initial graph. The first command creates a vector that starts a 0 and increments by 0.01 and ends at 2pi. The second command then makes x equal to sin(t) and generates a sine wave for our graph. Finally, the last command plot(x) allows you to see a graph of the sine wave that you established. Ensure that you put all the commands in the image below into your MATLAB command window.

Graphical user interface, text

Description automatically generated

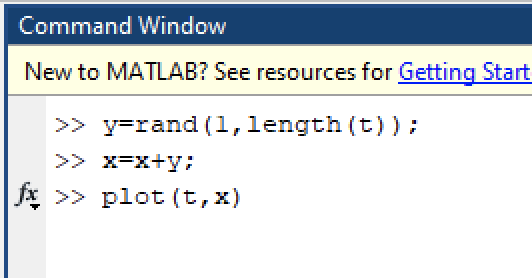
Once the commands above are inputted into the MATLAB command window, then you should get a chart the depicts the image below.



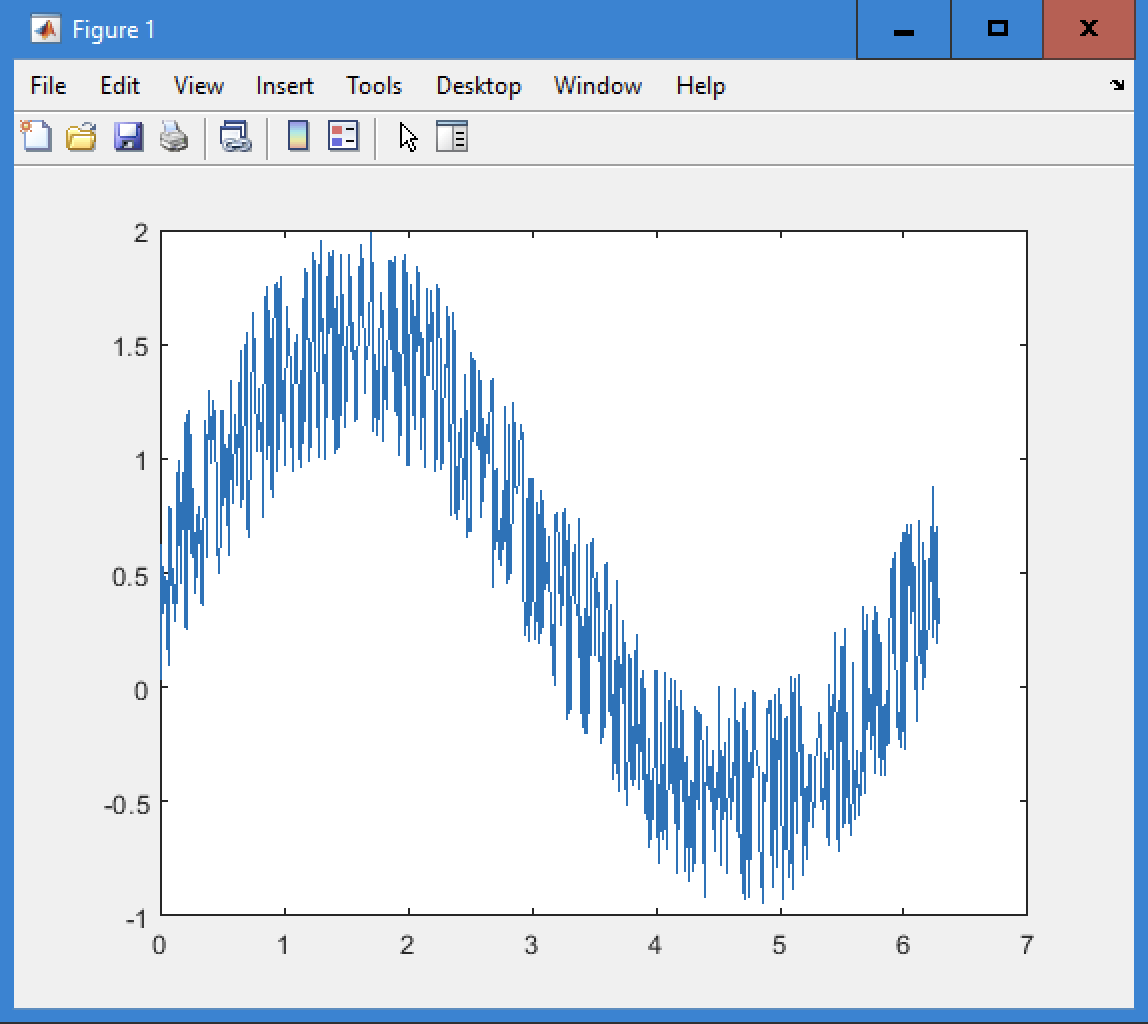
Once this has been done, you can then move onto salting the data.

Step 2: Salting the Data

Moving onto the next step, it is now time to salt the data. Salting the data basically means to add noise to the y values and make the data look sporadic in its y point placements. This can be done by inputting the below commands into the command window. The first command generates random y values for the length of t. The second command x=x+y adds the random y values to x. Finally, we are able to plot our newfound graph by using plot(t,x) as seen in the commands below.



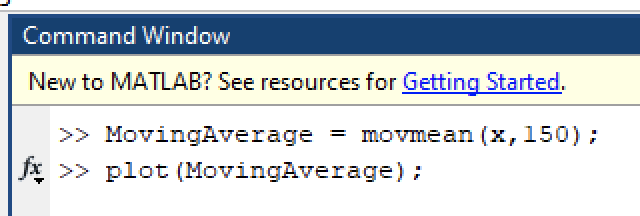
Once the commands above are inputted into the MATLAB command window, then you should get a chart the depicts the image below.



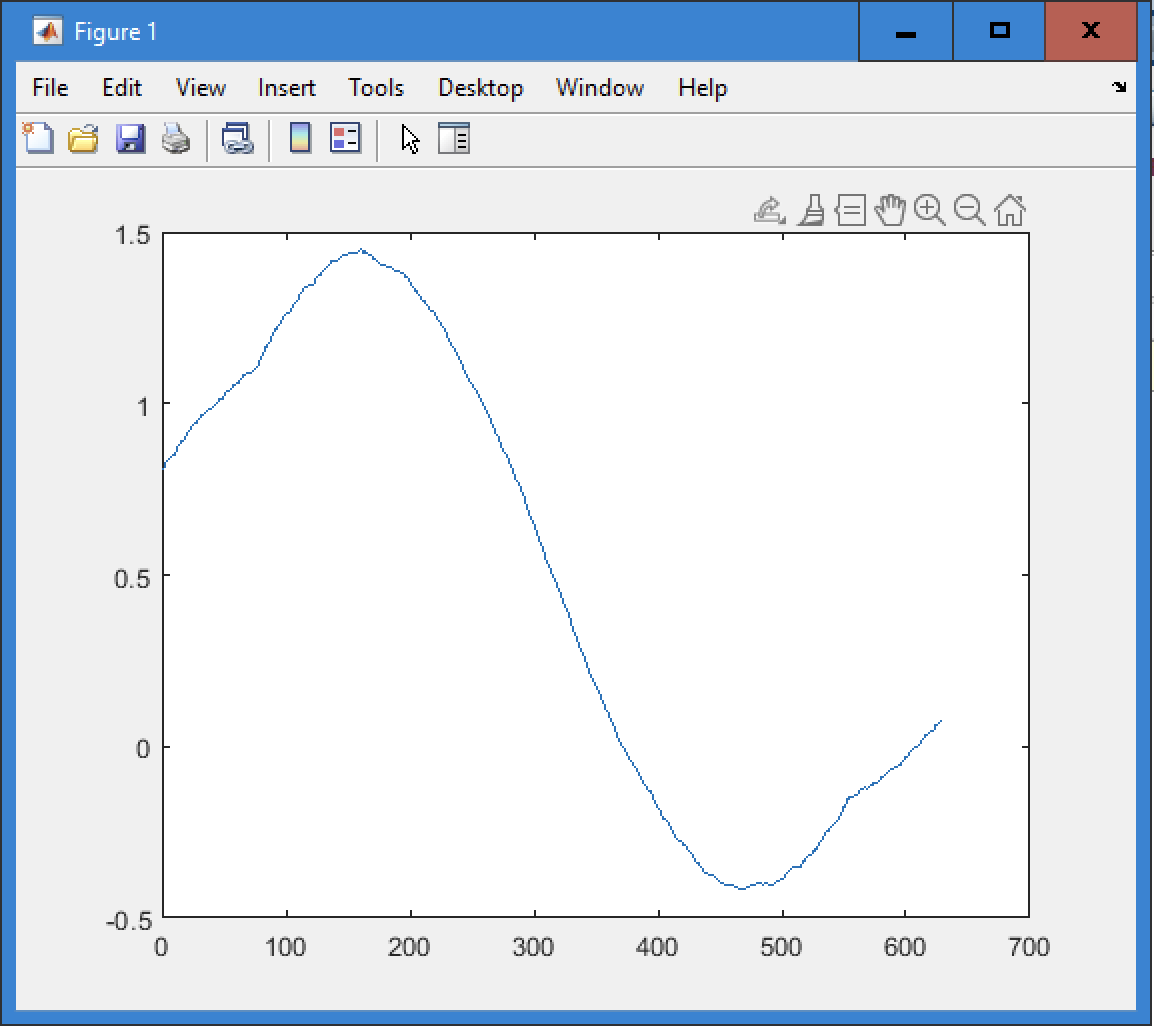
If your graph looks like this, congratulations! You have successfully added noise to your graph. You may now go onto the next step.

Step 3: Smoothing

Now it is time to smooth out our graph. The goal here is to be able to make the salted graph look like the one we originally created. To do this, we can use the movmean function in order to compare each y point with 150 neighboring points to get an accurate average to smooth out the graph. You can do this by entering the arguments below. The first command is what allows us to compare an x value with its 150 neighboring points and calculate their average. The second command actually allows us to plot the information we just calculated in the command before.



Once the commands above are inputted into the MATLAB command window, then you should get a chart the depicts the image below.



If your graph looks like this, then you successfully completed smoothing the data and removing the noise from it. Congratulations, you are now finished with the tutorial!

Learning Process and Results

The tutorial was relatively easy since I was able to find information about what I was trying to do on YouTube. The [YouTube video](https://www.youtube.com/watch?v=SP3zTandUjA) was able to help me for the first two parts of the tutorial since it detailed setting up the graph and then adding noise to it. The specific commands to plot the original sin graph were t=0:0.01:2\*pi, x=sin(t), and plot(x);. The t variable indicates a vector that starts at 0 and increments by 0.01 all the way until 2pi. Then x=sin(t) calculates the sin of each element of t. Then plot(x) is how you can depict the image of the graph that we just put into the command window. After doing all this, I was able to have MATLAB show me the image of the sine graph. The learning process was pretty much being able to find videos that were beneficial to my understanding of MATLAB and accurately follow them.

Then moving on to the salter, this methodology was also supplemented in the YouTube video I used previously and the most important command was y=rand(1, length(t)). This command generates random values for each element in t. The next command x=x+y adds the random value for each x value, and ultimately adds noise to the graph. Then the next command actually plots the function with plot(t,x). This creates the ability to depict the function in a graph with t representing the horizontal axis and x now representing the vertical axis.

Finally, the smoothing function required more research to better understand MATLAB and its possible uses. I searched “moving average MATLAB” on Google and found the first [result](https://www.mathworks.com/help/matlab/ref/movmean.html) to be extremely helpful. It was a link to the MATLAB website and presented me with the “movmean” function. This function was extremely helpful since it was able to find neighboring points for a single x index and then calculate the average. I was then able to utilize this function by making a variable called MovingAverage and making it equal to movmean(x,150), where x is a single point and then adding the 150 preceding values and calculating the average. An important fact to note was that if there weren’t enough points to calculate the average of 150 separate points, then the number of points were shortened to avoid an error. Once this was done, I was able to enter plot(MovingAverage) and plot the smoothed data that you see above.