Juan Villaman

05/02/2024

**Plotter, Salter, and Smoother in MATLAB**

# Learning MATLAB and Making the Plotter

Taking on this project, I have practically never worked with MATLAB, but had seen some it around the internet as the years went by and I was learning how to code. I had a rough understanding that it was an application that was able to easily run complicated and lengthy math operations that may be difficult to do by pen and paper.

Since my initial goal is to make these 3 programs, I began on the Plotter. I wanted user input in MATLAB, and I did not know if it was possible. After a quick google search, I learned two things: that MATLAB syntax reads like English, and how to take user input ([Official MATLAB Documentation](https://www.mathworks.com/help/matlab/ref/input.html)). Taking user input take a simple line of code:



Next was figuring out how to generate the X values. This led to me looking for some sort of for-loop within MATLAB. While there are for loops, I found an operator which works way better for this case, which is the [colon (:)](https://www.mathworks.com/help/matlab/ref/double.colon.html) operator. Below is an example of what it looks like:

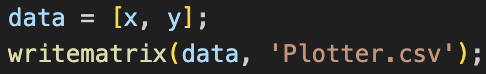


This generates a list of X-values from -20 to 20, incrementing by 1. The number in the middle determines how much x is incremented between -20 and 20. For the purpose of the plotter, we need x to increment by 1. Next, we need to calculate the Y-value for each X. This is pretty simple, as MATLAB writes like English, it is easy to write the quadratic function:

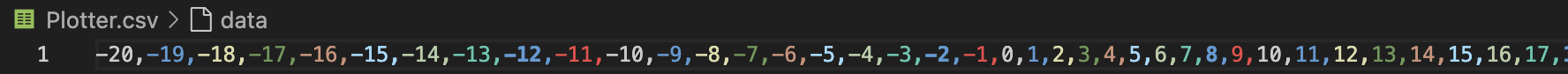
Like this: **y = a \* x \* x + b \* x + c**. However, we need to calculate the Y-value for EACH X-value that is in the range we have designated. There is an operator that allows us to do that for every single X-value. Since x is a vector in this case in MATLAB, we can use element-wise power to calculate each Y-value of the X values ([Source](https://www.mathworks.com/help/matlab/ref/double.power.html)/[Source #2](https://www.mathworks.com/help/matlab/matlab_prog/array-vs-matrix-operations.html)). It looks like this in our program:



The little bit allows us to calculate every Y-value of the X-values for our plotter program. The next step is getting to store those values into a CSV file. Using the writematrix() method, we can write matrices directly into CSV files. At first, I ran into an issue when the first CSV file was created. I created a matrix called data that store [x, y]. I then put that data into the CSV file. Implemented below:



The output in the CSV file was not exactly what I expected, however.



The CSV file contained all X-values, followed by all the Y-values. This is not what I wanted as I couldn’t properly put this in Excel and plot it easily. Since it looked like the matrix data makes X and Y rows, I needed to make them columns. Learned from Linear Algebra taken in a past semester, I need that the taking the transpose of a row vector turns it into a column vector. MATLAB conveniently can transpose rather easy by simply using ‘ after the row you want to apply it to. Implemented below:

A close up of a black background

AI-generated content may be incorrect.

After that quick fix, our output in our CSV file is now fixed. We can clearly see which X-value corresponds with its Y-value. Here is a snippet of our CSV file:

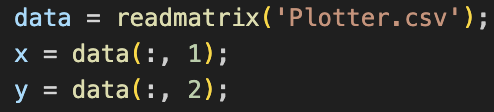
A screenshot of a computer

AI-generated content may be incorrect.

**This is the function graphed below:**

## Making The Salter

Now we need a way to salt the Y values of our function. First, we must read the data our Plotter program made. We used writematrix to write the data to our CSV, and now we are using readmatrix to read the CSV file. MATLAB has convenient methods of reading and writing files, as I learned when reading documentation. I also need to store the X and Y values into their own separate vectors. Using the colon operator that I used in my Plotter, I was able to successfully extract the data. Implemented below:



The colon inside the parentheses tells MATLAB to grab that column of our CSV file that data holds. X is stored in the first column, and Y is stored in the second column.