MATLAB Assignment 3

Spring 2019, Section B

This problem set's goal is to familiarize you with two useful concepts in MATLAB, functions and plotting, as well as continue getting you comfortable with indexing. It should also add to lesson 3's emphasis that ***for*** loops are often practically inefficient, despite the fact that they are often the first way we may imagine to tackle a problem. Note that at a minimum, all plots should have a title, x-axis and y-axis labels, and if there is more than one function in the same figure, a legend as well. Additionally, make sure your axis bounds are adequate.

Please submit this homework as *.m* files (note the plural), with suppressed output (obviously, the plots will still be displayed). Remember that all lectures and homeworks may be found at *github.com/guybaryosef/ECE210-materials*. Homework is due on ——— to guybymatlab@gmail.com.

1. Fun with find Write a function to return the value and index of a number in a vector / matrix that is closest to a desired value. The function should be called as $[val, ind] = findClosest(x, desiredValue)$. This function can be accomplished in less than five lines. Show that the function works by finding the value closest to 3/2 (and index of said value) in ***sin(linspace(0,5,100)) + 1***. You will find ***abs***, ***min*** and/or ***find*** useful. **Hint**: You may have some trouble using ***min*** when x is a matrix. To convert the matrix to a vector, you can use $y = x(:)$.

2. Sincing Ship Here we will look at a function near and dear to the signal processing community's heart - the sinc function. You are going to implement your own functions to find the local extrema and roots (remember to label your plots)!

- Sample a sinc with 10001 linearly spaced points on $[-2\pi, 2\pi]$ using the ***sinc*** function. Plot your results.
- Create a function (either anonymously or in another file) which locates the indices at which the input vector transitions from one sign to another. **Note:** This can be done in one line of code but it is $trecherous^{FF}$. For one scenario the vector has a positive value and then a negative value, i.e. $v(n) > 0$ and $v(n+1) < 0$. The root occurs somewhere in between, you can pick either n or $n+1$. We could loop through and check this condition at every point - don't do that. Instead think of a way to use logical indexing: You will want to write conditions on the vector and some kind of shifted version of itself. Beware however, when you do this you will have non-overlapping points. It is up to you to figure out what to with them.
- Apply your function to the sinc you created. Find the roots (x and y coordinates) and plot them as black circles on top of the sinc using ***plot(xRoots,yRoots,'ko')***. (make sure your axis is tight.)

- Now we are interested in finding the extrema (local minimums and maximums). Firstly, approximate the derivative by taking the difference between all adjacent elements and dividing by their time spacing. Then apply your function to the approximate derivative of your sinc to obtain the extrama. Finally, plot them as red stars on top of the sinc using ***plot(xMinMax,yMinMax,'r*')***.

3. Gotta Go Fast Generate a 300×500 matrix with entries $a_{i,j} = \frac{i^2+j^2}{i+j+3}$ using the following methods and use ***tic toc*** to time the speed of each and report the times in a table (using the ***table*** function).

(a) Using for loops and no pre-allocation.

(b) Using for loops and pre-allocating memory with ***zeros***.

(c) Using only element-wise matrix operations. **Note**: ***repmat*** and ***meshgrid*** will be useful here.