MATLAB Assignment 3

Spring 2017, Section B

This homework delves deeper into relational and logical indexing of matrices and plotting in MATLAB. We will walk you through three problems that will serve as useful examples of logical indexing. Assignment is due on February 15th, 2017 to nezin@cooper.edu.

1. Lunar Eclipse This question guides you through some basic image processing techniques in MATLAB. You would create interesting images with relational and logical indexing, as well as the *imshow* function to visualize what you have created.

(a) Create a 100×100 A where its contents are all ones.

(b) Create a 100×100 B where its contents are all zeros.

(c) In matrix A, set the values of entry $a_{i,j}$ equal to 0 if $\sqrt{(i-50)^2 + (j-50)^2} < 20$. **Hint** : *meshgrid* would be useful in creating the indices.

(d) In matrix B, set the values of entry $a_{i,j}$ equal to 1 if $\sqrt{(i-40)^2 + (j-40)^2} < 20$.

(e) Visualize the following results with *figure* and *imshow*. Describe each of the results with one sentence each.

   (a) A

   (b) B

   (c) Intersection between A and B

   (d) Union between A and B

   (e) Complement of intersection between A and B

   (f) Complement of union between A and B

2. 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. 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 $y = x(:)$. Show that it works by finding the value closest to 3/2 (and index of said value) in *sin(linspace(0,5,100)) + 1.*

3. Sincing Ship Here we will look at a function near and dear to the signal processing community's heart - the cardinal sine function, or sinc. You are going to implement your own functions to find the local extremum and roots.

(a) Sample a sinc with 10001 linearly spaced points on $[-2\pi, 2\pi]$ using the *sinc* function. Create a plot using *plot(x,y)*

(b) 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.

(c) 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.)

(d) Now we are interested in finding the extremum. (local minimums and maximums.) Apply your function to the approximate derivative of your sinc to obtain them, then plot them as red stars on top of the sinc using *plot(xMinMax,yMinMax,'r*')*.