MATLAB Assignment 3

Spring 2022, Section B

February 2nd, 2022

This homework explores some of the techniques we talked about in class regarding indexing. We will walk through some useful and interesting applications, more so than the examples shown in class! This assignment is due by 11:59pm on February 9th, 2022 .

1. Fly Me too the Moon :) This question guides you through some basic image processing techniques in MATLAB. You will create some interesting images using logical indexing as well as the *imshow* function to visualize what you've created

(a) Create a 100x100 matrix A whose contents are all ones

(b) Create a 100x100 matrix B whose contents are all zeros

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

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

(e) Visualize the following results using *figure* and *imshow*.

  (a) A

  (b) B

  (c) Intersection between A and B

  (d) Union between A and B

  (e) Complement of the intersection of A and B

  (f) Complement of the Union of 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. Sinc Exploration! We're going to take a look at a function that is super important in the signal processing world - the cardinal sine function or *sinc*. You're going to implement your own functions to find the local extremum and roots of a sinc function! You may need to install the signal processing toolbox to use sinc in matlab, but if you have trouble with that you can use the normalized sinc function definition: $sinc(x) = \frac{sin(\pi * x)}{\pi * x}$ if you use enough points in linspace.

(a) Sample sinc with 10001 linearly spaced points on the interval $[-2\pi, 2\pi]$ using the *sinc* function. Create a plot using *plot(x,y)*. You may have seen this last semester in Physics!

(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 gets pretty nasty. 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').

(d) Now we want to find the extremum. Apply your function to the approximate derivative (which we learned how to do last week) of your sinc to obtain them. Plot them as red stars on top of the sinc using plot(xMinMax, yMinMax, 'r*').