iPBR Assignment 2

Introduction to MATLAB part I

Due June 28th, 2017 @ 7pm

(Problem sets turned in after this deadline will not receive feedback)

Email finished problem sets and questions to: matthewsmith01@g.harvard.edu

Problem 1

Create a new script and save it as "yourName_P1". Execute the commands listed below, and use comments to explain what each command is doing.

```
1)   M1 = [ 3 5 1; 2 8 4];
      Creates a 2x3 matrix consisting
      of 3 5 1 in the top row and
      2 8 4 in the bottom row, and
      saves it under the variable name M1.
      The semi-colon after the 1 tells
      MATLAB to make a new row in
      the matrix.

2)   m1 = [ 80 75 6; 100 34 8];
     Creates a 2x3 matrix consisting
     of 3 5 1 in the top row and
     2 8 4 in the bottom row, and
     saves it under the variable name m1.

3)   m1 == M1
     Using the double equals sign "==" commands MATLAB to perform
     a logical comparison of these two variables.
4)   matrix3 = [ 89 4 9; 1 -80 503];
     Creates a 2x3 matrix consisting of 3 5 1 in the top row and
     2 8 4 in the bottom row, and saves it under the variable
     name matrix3.

5)   matrix3(1,:) = m1;
     This command attempts to assign m1 to the first row of the
     variable matrix3. This causes MATLAB to throw a dimension
     mismatch error because the user is attempting to assign 6
     elements to 3 positions.
6)   store{2,1} = matrix3
     Create a cell array named "store", and save the variable
     "matrix3" in the 2nd row 1st column.
7)   store{2,2} = M1
     In the variable named store, save the variable M1 in row 2,
     column 2.
```

8) `store{1,1} = 'Data of matrix3';`
In the variable named store, save the string 'Data of matrix3' in the cell at row 1, column 1.

9) `store{1,2} = 'Data of M1'`
In the variable named store, save the string 'Data of M1' in the cell at row 1, column 2.

10) `store{2,2}(1,:)`
Tells MATLAB to return the contents in cell in row 2, and column 2 (which is where we store the variable M1). The (1,:) to the left of the curly braces tells MATLAB to give the first row of the contents stored in position {2,2}.

11) `for i = 1 : 0.75 : 75`
The above line tells MATLAB to start a for-loop. The word "for" signifies the start of the loop. We then define the variable "i", which is the variable that MATLAB will be changing the value of in every loop.
```
        save1(cnt) = i;
```
The above line creates the variable called "save1" and goes to position "cnt" and stores the value of "i". HOWEVER, we haven't defined what the value of "cnt" is, so MATLAB will throw an error and this loop will not work.
```
    End
```
This ends the for-loop

12) **Challenge:**
```
n = 1;
for i = 1:800
    for k = 800:-1:1
        embedded(i,k) = n;
        n = n+1;
    end
end
```
Line1: defines variable n to be 1.
Line2: starts the for-loop and defines "i" as the looping-variable. The value of I will change within each loop, going from 1, increasing by one, until 800.
Line3: starts a second for-loop within the first for-loop (this is called nested/embedded for-looping). This for-loop uses the variable k as the looping-variable and starts at 800, decreases by 1, until reaching 1.
Line4: We're creating the variable, "embedded" and storing the value of "n" at position row i and column k. Remember, because the for-loop (k) is within for-loop (i). The for-loop (i) cannot move to the next loop until the nested for-loop (k) has finished all 800 loops. This effectively creates an 800x800 matrix.
Line 5: add 1 to n
Line6: end the for-loop (k)
Line7: end the for-loop (i)

Download the pSet2data.m file located on the course website under the link to this problem set. Double-click on the downloaded file to open and import the files to MATLAB.

Question 2

Suppose you are a young and talented scientist who is interested in behavioral neurobiology of humans. It's your first day in a new lab and your PI throws some fMRI data at you. They had recorded activity data for 300 seconds at 10Hz (10frames/sec). Your PI suspects that there is a single spike in neural activity somewhere in the data set, and needs you to tell them when this spike occurs. They tell you that spike resulted in a peak at least 10x that of baseline, so simply finding the position of the maximum value in this array will correspond to the time of the spike. Your PI doesn't know how to code, so they have been looking at each time-point in this dataset manually. They wrote a basic algorithm for finding the spike, but need your expertise in coding to implement it in MATLAB.

PI Algorithm:

```
i)     Set LargestValue equal to 0
ii)    Move to element 1 in the dataset
iii)   Is element 1 greater than LargestValue
iv)    If so, then store position of element 1 as LargestValue
v)     Else, do not change LargestValue
vi)    Move to next element.
vii)   Is element 2 greater than LargestValue
viii)  If so, then store position of element 2 as LargestValue
ix)    Else, do not change LargestValue
x)     Repeat steps ii-v for every element in the array
```

Your PI seems to think their algorithm is correct, however, it would be much too time consuming to do this manually. They come to you and beg for you to execute this algorithm (or something like it) in MATLAB.

Please refrain from using MATLAB's internal functions for this exercise

1) What are the dimensions of p2_vector?

    - This variable should have been imported into MATLAB's workspace after you opened the p2_data.mat file.

   P1_vector is a 1x3000 element array of numerical data.

2) Write a for-loop to index through every position in the p2_vector.

```
for i = 1:3000
    P1_vector(i)
End
```

   The first line defines the For-loop, and as so, it starts with for. Next we define our variable whose value will change as we loop, i. We set what are variable, I, will be each time in the loop, with 1:3000 (1 through 3000, counting by 1). The second line calls the value at each position in P1_vector.

3) Inside your for-loop write a conditional statement to find the maximum value of the p2_vector.

    *hint: one way of accomplishing this is to use your for-loop to search through every position in your vector. As you're proceeding through your for-loop use a conditional statement to compare each value to a variable called LargestValue. If the current value is greater than the LargestValue, then replace the value of LargestValue with the element at the current position.

```
largestValue = [ 0 ];
for i = 1:3000
    if p1_vector( i ) > largestValue
        largestValue = p1_vector( i );
    end
end
```

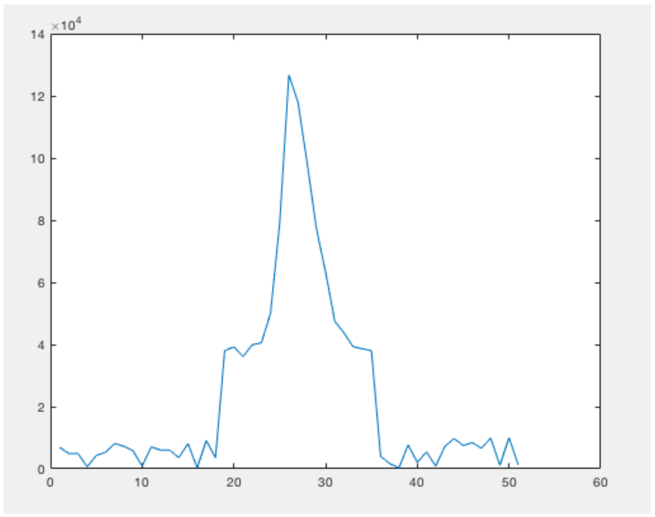
Line 1: we define the variable largestValue, which will be our placeholder for the largest value in the vector p1_vector.
Line 2: We begin our for-loop. We are using "i" as our looping-variable. The value of i will start at 1 and after each loop will increase by 1, until reaching 3000.
Line 3: Here we have the most important line, the conditional statement. If the value of p1_vectort at position (i) is greater than largestValue, then proceed to the next step.
Line 4: If the conditional evaluated true, then redefine largestValue to be the value at position (i) within the p1_vector.
Line 5: ends the conditional statement
Line 6: ends the for-loop

4) Write a conditional statement to find the time-point (index) for when the LargestValue occurs in p2_vector, and save this position as, spikeTime. This will be the time at which the spike occurs.

```
spikeTime = find(p1_vector == largestValue)
```

This is logical comparison of p1_vector to the largestValue. MATLAB will return a logical of 1s and 0s for each element in p1_vector.
I use the find function within MATLAB, which returns the positions that have 1s or true values. I then save this position as spikeTime

5) Execute the code and take a screenshot of the output:
figure; plot(p1_vector(spikeTime-25:spikeTime+25))



Question 3 (Challenge)

This question is a continuation of Question 3 from problem set 1. The details are listed below:

> Suppose you are interested in mouse behavior, and you want to know what odors scare mice. When mice are afraid, they stop moving and freeze. To test which odors are fear inducing, you record a movie of the mouse walking around its cage, and randomly release one of ten odors. You record the mouse for 2 hours, and expose the mouse to a randomly selected odor once every 2 minutes (for 30 seconds).
>
> Somebody else's program automatically records the results in two variables. The first variable contains the coordinates of the mouse, recorded every second. The second variable records which odor is being released (stored as the numbers 1-10, with a number 0 if no odor is being released), also recorded every second. Your goal is to create a bar graph that plots the average speed of the mouse when each odor is exposed.

Create a new script titled "yourName_P3". Write a script to convert the two variables you've received into a bar graph.

a. What type of variable is odorExp?
   OdorExp is a 2x2 cell array.
b. What data is stored in odorExp?
   - In position 1,1: Is string data, which reads 'Odor data'
   - In position 1,2: Is string data, which reads 'Mouse speed data'
   - In position 2,1: Numerical data with 1x7200 elements
   - In position 2,2: Numerical data with 1x7200 elements

c. Identify all time-points that have odor 1 presented.

   *hint: one method is to write a for-loop to search through each position of the odor array. In the for-loop you'll provide a conditional statement: if the odor presented == odor 1, then store that time point in an array. Your output should be an array containing all the time points in which odor 1 was presented.

```
odor1_presented = find ( odorExp{2 , 1} == 1 );
```

   Here I've used logical indexing again to do this complex task in one line of code. Of course, there are many other ways of accomplishing this task with for-loops. In my command, the line: odorExp{2 , 1} == 1, will create a logical index of all elements within the odorExp{2 , 1} variable that are equal to 1. I then use the MATLAB internal function, find, that returns the position of 1s within an array.

d. Find the speeds of the mouse when odor 1 is presented. To do this: use the array created in part c to index the speed array (in odorExp) and return only speeds when odor 1 is presented.

```
odor1_speed = odorExp{2,2}(odor1_presented);
```

   I've defined a variable odor1_speed, which equals odorExp{2,2} and indexed by the output of part C (which is the times at which odor 1 is presented).

e. Use the mean function to find the average value of the array created in part d. This single value will be the mean speed of the mouse when odor 1 is presented.
   The mean function in MATLAB works by typing: mean(yourArray). You're telling MATLAB to use the mean function on the variable, yourArray.

```
mean(odor1_speed)
>> 4.8267
```

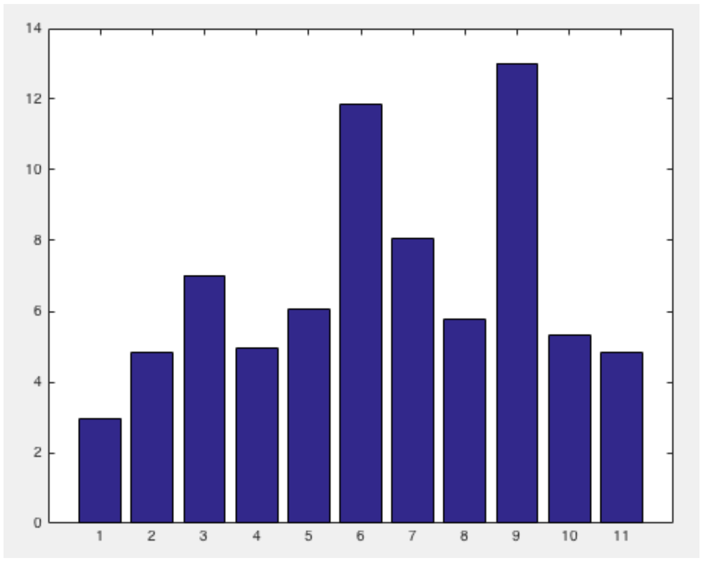
f. Perform steps C-E for all odor values (0-10) and produce an array that contains the mean speed for each odor, and title it AvgSpeed. This array should have 11 elements, one value for each odor.

```
cnt = 1;
odor1_presented = {};
odor1_speed = {};
for i = 0:10
    odor1_presented{cnt} = find(odorExp{2,1} == i);
    odor1_speed{cnt} = odorExp{2,2}(odor1_presented{cnt});
    AvgSpeed(cnt) = mean(odor1_speed{cnt});
    cnt = cnt+1;
end
```

   1st line: define the variable "cnt", which will be our placeholder later in the for-loop.
   2nd line: define the variable "odor1_presented" as an empty cell-array
   3rd line: define the variable "odor1_speed" as an empty cell-array
   4th line: start defining the for-loop. The variable that we'll loop through is "I". The values of i will start at 0, increase by 1, until 10.

5th line: This is the same as part C, except I've changed odor1_presented to be a cell-array, AND most importantly, added in the looping variable I, at the end of the command.
6th line: Find all the speeds at which odor I is presented. This is smilar to part D, with some key differences. Again, I've changed odor1_speed to a cell array, and I've indexed the cell array of odor1_presented with cnt.
7th line: I've now defined the variable AvgSpeed at position cnt, store the mean of odor1_speed{cnt}.
8th line: add 1 to the variable cnt
9th line: end the for-loop

g. Execute the command:
   figure;bar(AvgSpeed)



14
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
11