CS195H — Homework 0 — Matlab Warmup

Due: January 27th, 2015, before class

This first homework aims to get you warmed up with Matlab. Please work on this problem set in pairs. You can talk with your partner or the prof, and look things up on the internet, but please talk to no other students about the problems.

Organizational Stuff

Problem 1

[5 pts] Join the CS195h Google group here and use it to announce who your partner for this assignment is, or to offer to partner with someone.

General Information

For the remainder of the homework, you should hand in a single matlab file, called hw0.m. The webpage will tell you exactly how to hand this in.

In that file, you should write your answers to these problems, writing each textual answer as a comment. For problem 2, for instance, your answer might begin

\% 2a. x(:) produces a vertical list of the numbers 1 through 10 ...
\%
\% 2b. min(y) produces the list [1 5 ...]

It’s OK to have code scattered in with your answers; just make the answers easy to find.

Matlab functions follow a few basic rules (although the rules are sometimes broken as well). Most of them operate on matrices, i.e., rectangular arrays of numbers. In what follows, \( m \) denotes a matrix.

To show the “rules” by example, \( \text{size}(m) \) returns [3 5] meaning that \( m \) has 3 rows and 5 columns. If you want to know only how many rows or columns \( m \) has, you can use \( \text{size}(m, 1) \) or \( \text{size}(m, 2) \), respectively. Similarly, \( \text{sum}(m) \) finds the column sums of \( m \), but \( \text{sum}(m, 2) \) finds the row-sums. (Usually, when there is some direction needed for an operator like \( \text{sum} \), the default choice is “columns”.) Many procedures have multiple arguments; typically many of the extra arguments are “properties”; for the axes in a figure, for instance, one property is \( XLim \), the lower and upper limit for the extent of the \( x \)-axis in the plot. For “plot”, one of the extra arguments is \( \text{LineWidth} \), so you might write \( \text{plot}(x, y, \text{’LineWidth’}, 1.8) \) to get slightly thicker lines than usual. The convention on these properties is that each word gets an initial capital, and most words are spelled out, as in \( \text{MarkerFaceColor} \). Some of the older properties that have been around since the early days of Matlab have shorter names, line \( XLim \) instead of \( XLimits \).

Here are some Matlab functions/operators that you might find useful: \( \text{help help} \) gives a command line help for help function.
doc help: shows up the reference page for help function in Help browser.
size(x): returns the sizes of each dimension of matrix x.
umel(x): returns the number elements in matrix x.
length(x): returns the length of the largest matrix dimension. Equivalent to max(size(x)).
max(x): returns the maximum entries for each column of x.
min(x(:)): returns the minimum entry for the whole matrix x.
x .* y: multiplies two matrices element-wise.
x ./ y: divides two matrices element-wise.
x': transpose of matrix x.
x(x == 1) = []: deletes all the 1s in x.
plot(x(:, 1), x(:, 2), go-): draws an n x 2 array of points in green, with points marked by circles, and connected by lines.
plot(x(:, 1), x(:, 2), Color, [1 1 0]): draws an n x 2 array of points in yellow (100% red + 100% green + 0% blue), connected by lines.
gcf: returns the current figures handle.
figure(gcf): brings the current figure to the foreground so you can see it.
gca: returns the current axes handle.
set(gca, XLim, [-10 10], DataAspectRatio, [1 1 1]): make the current figure axes go from -10 to 10 in x (you can also set YLim and ZLim...), and make sure that one unit of x, y, or z shows up as the same thing in screen cords.
hold on: retains the current plot so that subsequent graphing commands add to it, instead of replacing it. Useful for displaying multiple plots together.
isequal(x, y): tests if the matrices x and y are equal. This works even if the matrices have different sizes, or one is empty.

There are a few different ways to write Matlab code in files to handin for homework. One is to write a script (e.g. a series of command line inputs stored in a separate file) which you run each time. Another is to write a series of functions which can print to the command window using the disp function call. Scripts can also call user-defined functions, as long as they are in the current Matlab folder. Therefore, it is also possible to use some combination of functions and scripts. A good approach is to write one script, which when run, prints output and displays figures for the entire homework assignment. It is also highly recommended that you look at using cells inside of your scripts or functions. This allows you to test chunks of code without having to copy and paste them into the command window. There is a good video tutorial for these, here.

Matlab Problems

Problem 1

[10 pts] Start Matlab on your computer. Assuming you’re using release 2011b or newer, the following should get you to the tutorial introduction to matlab:

1. Click the Help menu, and select Product Help; alternatively, press F1.
2. In the left column of the help browser that appears, click on the "+" box next to MATLAB
3. Click on Getting Started.
Now read three sections of the Getting Started tutorial: Introduction, Matrices and Arrays, and Graphics. Then read the first three parts of the fourth section, Programming: Flow Control, Other Data Structures (the part on multidimensional arrays only), and Scripts and Functions (up through ”Global Variables”; then skip to Vectorization and Preallocation).

Having read these, write, in the top of your solutions file, %Problem 1: I’ve read the tutorial. (The “percent sign” makes this a comment in the matlab file, so that it doesn’t affect execution of your later solutions.)

Problem 2

[10 pts] (a) Start up Matlab; set up a few variables by typing
> x = 1:10
> y = reshape(1:12, 4, 3)
> z = randi([1 4], 2, 3)

Now type x(:), y(:), and z(:) and see what each produces. Explain what you see.

(b) Type min(y) and min(y, [], 2) and min(min(y)). Explain what you see. Hint: type doc min.

(c) Can you think of a simpler way to get the minimum entry of a rectangular matrix than min(min(y))? Hint: use part a.

Problem 3

[8 pts]

1. Use randi to build an 8 by 7 matrix x of integers between 0 and 10, inclusive. Then do each of the following without using loops.

2. Write an expression that produces all the even-numbered columns, and another that produces all the odd numbered rows, and a third that produces a vector that contains the main diagonal of your matrix (i.e., the elements x(1,1), x(2, 2), ..., x(7, 7).

3. Set the third row of x to be all zeros.

4. Set the last column of x to be all ones.

   (You can do both of these without using the ”zeros” or ”ones” function.)

5. Append a copy of the first row as a new row of the matrix. Check the size of the new matrix.

6. Delete the first row.

7. Produce the averages of the rows of x.

8. Produce the column-sums of x.

9. Find the maximum and minimum of each row and column. Find the maximum entry of the whole matrix.

10. Reshape x into a 28 × 2 matrix called y and display the result (hint: reshape)

11. Add [1 0 1 0 1 0 1] to each row of the matrix. Hint: use repmat.

12. Swap the order of the rows in x so that the last row comes first, then the second-to-last, etc.
Problem 4

[20 pts] Here is a graph of the scaled arctangent function on the interval $[-10, 10]$.

To produce this, I computed $\arctan(x)$ at the points $x = -10.0, -9.9, \ldots, 9.9, 10.0$ and plotted the blue line. I then re-computed the arctan at integer points, and drew the boxes with the red outline and black fill, and added a label to the graph.

Your job is to reproduce this graph, or something like it. You'll need to read about `plot`, `hold on`, `title`, and you'll probably need to look at “Plot properties” to figure out how to set the `MarkerFaceColor`, etc. The one tricky thing is the label. I’ve used LaTeX (a mathematical word processing program) to write a pretty-looking formula:

```
title('Plot of $y = \frac{2}{\pi}\arctan\,(x)$','Interpreter','latex')
```

You may feel free to plot some other graph than arctan, on some other domain, but your graph should have both a curvy part and a discrete marker plot, and should have a nice LaTeX label like mine.

Problem 5

[20 pts] Recall that in class we discussed loops in a circle, where a loop always started at the bottom of the circle (which we’ll call angle zero), and tooks steps of 90 degrees clockwise (CW) or counterclockwise (CCW), which we indicated with the values $-1$ and $+1$, respectively. So labelling the circle with the points of a compass, $+1 + 1 + 1 + 1$ indicates a loop that starts at S and goes S-E-N-W-S. The initial ”S” is the starting point; the first $+1$ adds the $-E$ to the sequence; the second adds the $-N$, and so on.

(a) Given a $1 \times n$ array of $+1$ and $-1$ values, how can you tell whether this represents a closed loop (as in the example above) or an unclosed path (as in $+1 + 1 - 1$, which corresponds to $S - E - N - W$)? Write a procedure

```
function res = isLoop(seq)
    % Takes a +1/-1 sequence defining a path in the circle and tells whether it’s a loop;
    % if so, return true; else false.
    ...
```

There’s code, `randomLoop.m`, on the website to produce a random loop sequence (a vector of $+1$ and $-1$ values). You’re going to show that any such loop sequence is “grid homotopic” to a
sequence that’s either empty or wraps some number of times clockwise or counterclockwise around the circle, without hairpin turns (i.e., a sequence of all +1s or all −1s. You’re going to do so by writing a program. We’ll call a sequence that’s all +1s or all −1s, or empty, a “simple sequence.”

(b) Write a procedure `loopPlot` that takes a loop-sequence and produces an ”angle plot” of the loop. The angles should be in degrees, and each successive angle should differ from the previous one by either +90 or −90. The first angle should be zero, and the last angle should be zero (mod 360). (If the input describes a path that’s not a loop, the code is allowed to fail, by the way!)

(c) Write a procedure

```
function res = isSimple(seq)
    % Takes a +1/-1 sequence defining a LOOP in the circle and tells whether it's simple
    % (i.e. all +1 or all -1 or empty.
    ...
```

(c) Build a function `res = simplify(loop)` whose input is a non-simple loop-sequence of length \( n \), and whose output is a “simpler” loop sequence that’s grid-homotopic to the first.

(d) Put this all together by writing a program that resembles the following:

```
function showLoopSimplification()

    figure(1)
    clear
    myLoop = randomLoop();
    n = numel(myLoop);
    max = n*90;  % largest angle that could possibly occur.

    hold on;
    if (~isSimple(myLoop))
        error('randomLoop returned a non-loop')
    end

    while ~isLoop(myLoop)
        loopPlot(myLoop);
        pause(0.1); % short wait to allow plot time to show itself
        figure(gcf); % Bring plot to the front so we can see it
        myLoop = simplify(myLoop);
    end
    loopPlot(myLoop);
    hold off
```

and test the program by running it multiple times. You might want to remove the `hold on` and `hold off` parts of the program, so that it’s easier to see the most recent plot, but if you do so, you’ll see some jerkiness as the loop resizes during the course of the grid homotopy.

**Problem 6**

[5 pts] Go to the CS195h Google group here and post a useful Matlab expression or technique that is not listed here and has not been posted by any other student. (This might include hints about how to use the Matlab IDE effectively, for instance).
Problem 7

[0 pts] Tell us how long you spent on this problem set, and who you worked with. (Please hand in only one copy for the two of you.)