Homework 8

Due 4:00PM, Wednesday, April 10, 2019

Introduction

The babies and puppies now take their battles to a new language.

The primary goal of this homework is to familiarize you with MATLAB. Most of the questions are straightforward, and hints are included. The challenge here is converting your understanding of data manipulations from Python to MATLAB.

Please feel free to post to Piazza for additional support.

Installation and Handin

Homework Installation. To copy support files to your to your home directory for this homework type the following in a Brown CS terminal window:

```
cs4_install hw08
```

There should now be a `hw08` folder within your homeworks directory. Using Terminal, you can move into the `hw08` folder with the `cd` command:

```
cd ~/course/cs004/homeworks/hw08
```

Note: You should make a separate script for each problem that requires you to write a function. Name the script with the same name as the function. But make sure to place all code including test cases and script calls to files containing your functions of the same name in **`cs4hw08.m.`**

The stencil code is organized in the following publish-friendly format in **`cs4hw08.m`**. You should place test cases and function listings in the corresponding sections like so. If you don't have to create an entirely new function that goes in a separate file, you can just place the code under "function listing(s)".

```
%% Question 8.1
% Put comments for any written answers to non-coding questions here,
% as well as any calculations and script calls. Use the following
% template for functions that you are asked to write.
%% *A) Test cases*
% (Put your assert statements here.)
assert(isequal(yourFunction(a,b,c), [1 2 2]));
...
%% *B) Function Listing(s)*
dbtype yourFunction.m
% Where yourFunction.m is the file in which your function of the same name
is written.
...
%% Question 8.2
...
```

* *if required*

Homework hand-in. Be sure to turn in all the files requested and that they are named exactly as specified, including spelling and case. This includes **`cs4hw08.m`** and all the scripts of the different functions that you create in this homework. When you're ready to submit the files, run:

```
cs4_handin hw08
```

from a Brown CS Terminal window from your `~/course/cs0040/homeworks/hw08` directory. The entire contents of `~/course/cs0040/homeworks/hw08` will be handed in. Check for a confirmation email to ensure that your assignment was correctly submitted using the `cs4_handin` command. You can re-submit this assignment using the `cs4_handin` command at any time, but be careful, only your most recent submission with be graded.

Assignment

In this assignment, you will develop several functions designed to increase your understanding of how matrices and conditions work in MATLAB.

Do not use `for` and `while` loops to solve these problems. Babies and puppies have signed a treaty to not use these weapons for the sake of the world. Instead, use array indexing and MATLAB's element-wise operations, and the other array support commands that are built into MATLAB.

Problem 8.1) Make a Matrix (2)

Create the following matrix in a single assignment. Try to use as few numbers as possible. Enter your code directly into the **`cs4hw08.m`** script.

```
M =
0 0 0 0 0 1
0 0 0 0 0 1
1 2 3 4 5 1
0 2 4 6 8 1
8 7 2 5 9 1
```

Hint: Use `zeros`, `ones`, [,], ',', :, and ';'.

Problem 8.2) Array Clipping (4)

Create an anonymous function `myLimit` that changes all values in an array that are greater than 1 to 1. Include tests of your function.

Hint: Use logical arrays, element-wise array multiplication and addition.

Problem 8.3) 2D Plotting (4)

Plot the functions

$$f(x) = 3 + e^{-x} sin(6x)$$

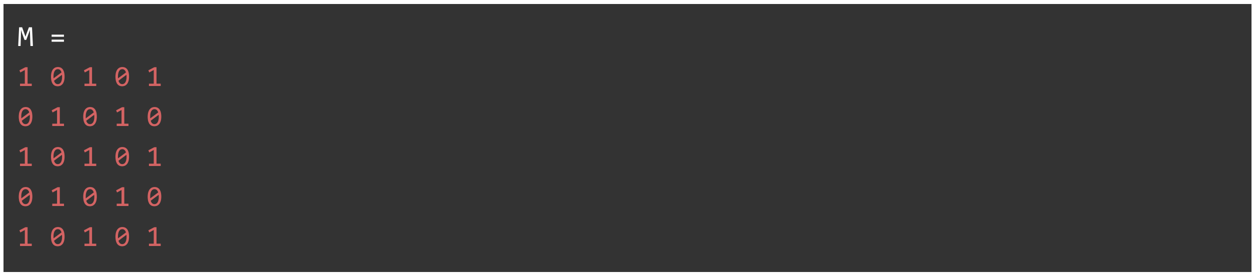
and:

$$g(x) = 4 + e^{-x} cos(6x)$$

for `0 <= x <= 5` on a single axis using the `plot` and `hold` command. Give the plot axes labels, a title, and a legend. Put your solution directly in your hand-in script.

Problem 8.4) Regular Patterns (10)

Write a function with header `[M] = myCheckerBoard(n)`, where `M` is an `n` x `n` matrix consisting of zeros and ones laid out in a checkerboard pattern. Assume `n` is odd. For `n` = 5, `M` would have the following form:



```
M =
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
```

Hint: Use linear indexing and `reshape`.

Additional Challenge (no extra credit): Do you see a one line solution for n even? How about both cases?

Problem 8.5) Splitting (10)

Write a function `[M1, M2] = mySplitMatrix(M)` where `M` is a matrix, `M1` is the left half of `M`, and `M2` is the right half of `M`. In the case where `M` has an odd number of columns, the middle column should go to `M1`. Assume that `M` has at least two columns.

Hint: Use `size` and `ceil`.

Problem 8.6) Linear Indices (6)

Create a function with header `ind = myLinearIndex(i, j, m, n)` which computes the linear index of the element `(i,j)` in an `m` by `n` matrix. Include test cases.

Hint: Try using a matrix generated by `reshape(1:m*n, m, n)’` as one of your test cases.

Problem 8.7) Multiple Dispatch (6)

Write a function with the header `[f] = myMultOperation(a, b, operation)` that computes multiple operations on a pair of arrays. The input argument operation is a string that is either ‘plus’, ‘minus’, ‘mult’, ‘div’, or ‘pow’, and `f` should be computed as `a+b`, `a-b`, `a.*b`, `a./b`, and `a.^b` for the respective values for operation. If an operation that is not recognized is used, report this fact using the `error` function. Make sure your function is vectorized.

Below are some examples:

```
x = [1 2 3 4];
y = [2 3 4 5];
myMultOperation(x,y,”plus”)
ans =
     3 5 7 9
myMultOperation(x,y,”minus”)
ans =
     -1 -1 -1 -1
myMultOperation(x,y,”exp”)
Error using myMultOperation exp isn't an allowed operation.
```

Problem 8.8) Triangle Test (10)

Consider a triangle with vertices at (0,0), (1,0), and (0,1). Write a function with signature `[S] = myInsideTriangle(x, y)` where `S` is the string ‘outside’ if the point `(x, y)` is outside of the triangle, ‘border’ if the point is exactly on the border of the triangle, and ‘inside’ if the point is on the inside of the triangle.

Hint: Test if the point is outside the unit square, then check if the point is on or below the diagonal, etc.

```
%% Example
myInsideTriangle(.1,.1)
ans =
     inside
```

Problem 8.9) Vector Padding (6)

Write a function with signature `[out] = myMakeSize10(x)`, where `x` is a vector, and out is the first 10 elements of `x` if `x` has more than 10 elements, or `out` is the array `x` padded with enough zeros to make it length 10 if `x` has fewer than 10 elements. Do not change the shape of `x` (i.e. if `x` is a column vector, `out` should also be a column vector; similarly, if `x` is a row vector, `out` should also be row vector). If `x` is empty or 1 by 1, `out` should be a row vector.

Use `if` statements (e.g., `if`, `elseif`, and `else`) to implement your solution.

```
%% Examples
myMakeSize10(1:2)
ans =
     1 2 0 0 0 0 0 0 0 0
myMakeSize10([3 ; 4])
ans =
     3
     4
     0
     0
     0
     0
     0
     0
     0
     0
myMakeSize10(1:20)
ans =
     1 2 3 4 5 6 7 8 9 10
```

Problem 8.10) Roots (20)

Let $Q(x)$ be the the quadratic equation $Q(x) = ax^2 + bx + c$ for some scalar values `a`, `b`, `c`. The root of a function $Q(x)$ is a value of its arguments `r`, such that the function's value is zero, $Q(r) = 0$. The two roots of a quadratic equation can be described by the quadratic formula:

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

A quadratic equation has either two real roots (i.e., $b^2 > 4ac$), or one root ($r = -b/2a$).

Write a function with header `[nRoots, r] = myNRoots(a,b,c)` where `a`, `b`, and `c` are the coefficients of the quadratic $Q(x)$, `nroots` is 2 if $Q(x)$ has 2 real roots, 1 if $Q(x)$ has one root, -2 if $Q(x)$ has two imaginary roots, and r is an array containing the roots of $Q(x)$.

Please let us know if you find any mistakes, inconsistencies, or confusing language in this document or have any concerns about this and any other CS4 document by posting on Piazza or filling out our anonymous feedback form.