COE 301 Introduction to Computer Programming Fall, 2024

Homework 12 – Due: 12/06/2024 11:59 pm

Problem 1. [30 points] *Data Fitting*. We have 500 pairs of samples numbered x= [x1; x2; …; x500] and y= [y1; y2; …; y500] in "hw12prob1.mat" file under "Files" tab "lab12_data". Write a MATLAB script to find the least squares solution for a = [a_0; a_1; a_2] so that the function $y_m(x) = a_0+a_1x+ a_2\sin(x)$ best fits the data using \ similar to Lecture 17 page 19-22. Plot data and the model fitting curve on the same plot.

Note: you may import .mat format data into your workspace as:

```
load hw12prob1.mat
```

After the loading, you will see two vectors x and y in your workspace.

Report your figures and comments in the write-up (Please do not submit the .fig file, instead, please insert your figure in the write-up).
Please submit your .m file as "yourLastName_hw12_prob2.m" with all the MATLAB commands you used.

Problem 2. [30 points] Consider a mass-spring-damper system as we described in Lecture 17. The initial displacement of the mass is r(0) = 0.5 m and its initial velocity is v(0) = 1.0 m/s. A force of 1 N is applied for the first 4 seconds. The system has the following parameters:

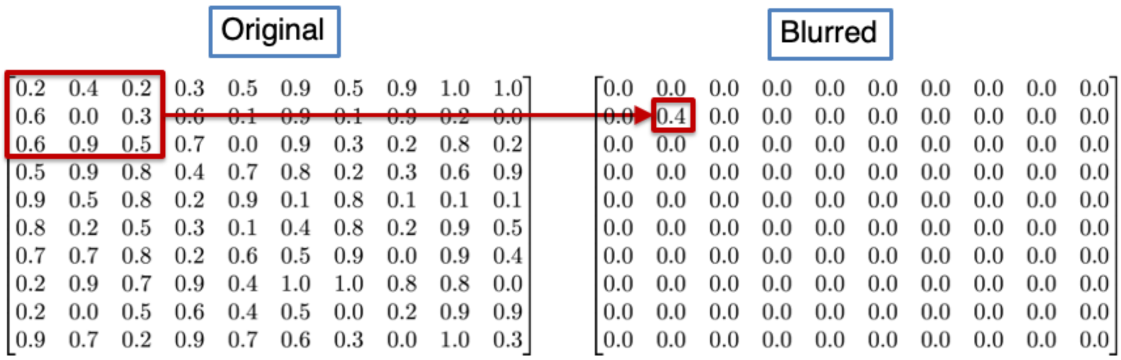
```
k  =  4;  % spring coefficient N/m
b  =  1;  % damper coefficient N.s/m
m =  1;   % mass in kg
```

Plot r(t) vs. t and v(t) vs. t on the same plot for a period of 30 seconds with 5001 samples.

Report your plot in the write-up.
Please submit your .m file as "yourLastName_hw12_prob3.m" with all the MATLAB commands you used.

Problem 3. [40 points] *Basic Image processing II*. (1) Download the file 'butterfly.gif' under "Files" tab "Lab12_data", we will use it as the input image and generate a blurred version using a n-by-n blurring window. For example, if n equals 3, imgOut(2,2) on the right can be calculated by averaging the 9 numbers in the red box from the original image. If any terms in the 3-by-3 box are out of the image, you can ignore them. One solution is to use the MATLAB built-in function filter2 (see the extra optional practices as an example). However, here

we would like you to get practices on writing for loops in MATLAB. You must **use a nested for loop to solve this problem, make the code work for any n (a positive odd number).**



(2) Design an image warping and shift operator so that the image rotates 90-degree clock-wise (Hint: Note that the rotation is around (0,0), you may try a few output pixels and see where they are mapped with respect to your source image. This allows you to determine if any shift is necessary). **You are not allowed to use any MATLAB images rotation functions to solve this problem**.

<u>Submit your .m file as “yourLastName_hw12_prob3.m”.</u>
<u>Report your blurred images for the case n equals 5, 9, and 15 and the rotated image in the write up.</u>

Bonus (+30 points) *The Mars Orbiter Laser Altimeter (MOLA)*. The file ‘hw12mola.mat’ under “lab12_data” contains a stream of data acquired by a laser altimeter orbiting Mars and measuring its topography.

Each entry is a Mars surface elevation measurement (in meters) referenced to a nominal reference Mars sphere of radius 3396200 m. Therefore, the positive data represent “mountain” and negative values represent “valley”.

The first entry was acquired at (lat = 0.0 and lon = -180.0). Because the satellite is in a circular polar orbit and Mars is rotating from the west to the east, the next entry was acquired at (lat+dlat, lon+dlon) where:

dlat =1.013895028871900;
dlon =-0.081103695129408;

The third entry was acquired at at (lat+2*dlat, lon+2*dlon) and so on. When satellite passes the north/south pole or longitude 180/-180 degree, you have to adjust the jump in the lat/lon accordingly. Note that the satellite is initially traveling northwards (dlat is positive). After it passes the north pole, it travels southward and dlat becomes negative.

Given the above information, create a topographic map of Mars on a latitude/longitude grid. If multiple measurements fall into one pixel, we can simply take an average of these measurements. Display your result using imagesc with proper colorbar and axis labels. Try different color maps and pick one you like. Submit your code and your final topography map image.

Submission Instructions:

There should be 4 files in your submission:

1. A write up (any type- .txt, .docx, .pdf are all fine) that contains your answers to all questions in problem 1-3.
2. The .m file for problem 1.
3. The .m file for problem 2.
4. The .m file for problem 3.

Please make sure your last name is included in the filename.

Optional Short answers questions. The following questions will not be graded. You may use them for preparing your next quiz.

(1) What is the output of the following block of code?

```
x = [-4:9];
y = [-10:3];
a = x(x>0 & y>0);
disp(a)
```

(2) you are given a vector:

```
x = [0.1  0.4  0.5  0.05  0.9  0.8  0.5];
```

a) Write a snippet of code to count the number of elements in x greater than 0.25 but less than 0.75. **Do not use any loops.**
For the given vector x, this would produce 3

b) Now, write a snippet to compute the sum of all elements in x that are greater than 0.7. **Do not use any loops.**
For the given x, this would produce 1.7.

(3) What is the output of the following block of code? Use the MATLAB function diag to generate the same matrix A without using loops.

```
nrows = 5;
ncols = 5;
% IMPORTANT initialize A before the loop
A = zeros(nrows,ncols);
for c = 1:ncols
    for r = 1:nrows

        if r == c
            A(r,c) = 2;
        elseif abs(r-c) == 1
            A(r,c) = -1;
        end

    end
end
```

(4) Explain what this MATLAB snippet computes.

```
n = 1;
nFactorial = 1;
while nFactorial < 10^4
    n = n + 1;
    nFactorial = nFactorial * n;
end
```

*****Extra optional practices*****

Short problems to practice MATLAB syntax. **We will be giving most of the answers here simply as a way for you to practice using these new commands.**

% Conway's Game of Life in 5 steps.
Logical Array operations! We're going to walk through remaking the game of life with MATLAB. You'll see how the matrix-centered approach to MATLAB makes many of the functions much simpler to compute.

1. Make a random board, size 800 x 600

We use the "randi" function to make random integers, which accepts a range (which we want to be from 0 to 1)

```
board = randi([0 1], 800, 600);
```

2. Make a for loop to run through 100 iterations

```
for ii = 1:100
     ...we'll fill this in ...
end
```

3. Within the for loop, display the board.
We can do this using the function "imshow", which shows a matrix as an image.
There are a few similar functions we could also use: "imagesc" and "image" would both work.

```
imshow(board); drawnow;
```

The "drawnow" command is only needed because we are running this in a for loop and want MATLAB to immediately plot the result.

4. Now we will sum the surrounding pixels for each spot on the board.
There is a built-in function called "filter2" which will make this very easy.
You'll learn more about how filters work in another class I teach, so we won't explain it here, but the quickest description is that it takes a moving weighted-sum at each pixel.
For now, just use the following command, and you'll get back a matrix the same size as the board with the number of live neighbors:

```
liveNeighbors = filter2([1 1 1; 1 0 1; 1 1 1], board);
```

5. We'll now "advance" the board by using logical matrix operations.
There are two cases where a pixel stays alive:
(1) If the number of liveNeighbors is equal to 3, OR
(2) If the number of liveNeighbors is equal to 2, AND the current `board` pixel is a 1

Thus, we can make two temporary matrices and "OR" them together with '|'
The 'AND' we will perform using '&'
Note that you'll often see "&&" as a logical AND, and '||' as a logical OR, but those can only operate on scalars.

```
n3 = (liveNeighbors == 3);
n2 = ((liveNeighbors == 2) & board);
board = (n2 | n3);
```

We broke that down into several pieces, but if we wanted, we could have just done:

```
board = (liveNeighbors==3) | ((liveNeighbors==2)&board) ;
```

but our first way is probably clearer to the reader.

We're done!

The for loop should look like:

```
for ii = 1:1000
    imshow(board); drawnow;
    liveNeighbors = filter2([1 1 1; 1 0 1; 1 1 1], board);

    n3 = liveNeighbors == 3;
    n2 = (liveNeighbors == 2) & board;
    board = (n2 | n3);
end
```

If you run this code, you'll get an evolving random board. You can watch it all the way, or break out with "ctrl-c". There are plenty of board patterns for game of life (often as a text file). You may load an input file as a matrix and watch it run.