CSE 402: Biometrics and Pattern Recognition
An Introduction to Matlab
Exercise#1

Due by 11:00 PM on October 11, 2021

The purpose of this exercise is to introduce you to arrays, matrices and some basic image processing operations in MATLAB. A few basic MATLAB commands are presented below. However, you are welcome to use any programming language. Assignment will NOT be graded; but you must nevertheless submit it.

%% Arrays and Matrices

1. Create an **array**. The semicolon suppresses the display on screen. Try executing the same statement without inserting the semicolon

```
>> A = [1 2 3 4 6 4 3 4 5];
```

2. **Add** 3 to all elements of A

```
>> B = A + 3
```

3. **Plot** the 1D points in A and B in the same graph

```
>> plot(A, 'r*-') % r denotes the color red; single quotes have been used
>> hold on
>> plot(B, 'ko-') % k denotes the color black; single quotes have been used
```

4. **2D plot** after "combining" points in A and B

```
>> hold off
>> plot(A, B, 'r*-')
>> xlabel('A') % note: that A has been placed between two single quotes
>> ylabel('B')
```

5. Create a simple 3x4 **matrix** (3 rows, 4 columns). A semi-colon is used to separate the rows of the matrix

```
>> A = [1 2 0 7; 2 5 -1 6; 4 10 -1 8]
```

6. Compute the **transpose** of the matrix

```
>> B = A'
```

7. Create a simple 4x2 matrix

```
>> C = [3 1; 5 2; 6 8; 1 7]
```

8. **Multiply** A with C. Multiplying a 3x4 matrix with a 4x2 matrix results in a 3x2 matrix.

```
>> D = A*C
```

9. Perform an **element-by-element** multiplication of two equally-sized matrices. Note that this is different from "matrix multiplication".

```
>> E = A.*B'
```

10. Square **all the elements** in matrix A. Note that all the elements in the matrix can be changed by a single command.

```
>> F = A.^2
```

%% Image Processing

11. Input an **image** file

```
>> fpimg = imread('fingerprint1.tif');
%You must copy this image from the zip file to the working directory in Matlab
% note that single quotes are used around Prewitt
```

12. The variable "fpimg" is a matrix. You can determine the **size of the image** (matrix) by using the size command. The size command outputs the width and height of the matrix (i.e., the image)

```
>> size(fpimg)
```

13. **Visualize** the image

```
>> close all %this command closes all the figures previously opened in Matlab
>> imshow(fpimg, [ ]);
```

14. Modify the image by applying a "**filter**" to it (we will be learning more about filters later in the course). A filter can be created using the "fspecial" command in Matlab. For example, a "log" filter can be created as follows:

```
>> log_filter = fspecial('log')
```

15. Apply the **log filter** to the input image:

```
>> log_output = imfilter(fpimg, log_filter);
```

16. **View** the filtered image

```
>> figure %this command is used to ensure that the filtered image is displayed in a new window,
rather than the window that was used to display the original image.
>> imshow(log_output, [ ]);
```

17. Apply an **edge detector** to the input image. An edge detector uses a "filter" such as Prewitt, Roberts, Sobel, etc. We will discuss these filters in Chapter 2. Use the Prewitt filter to detect edges (note that the "Prewitt" filter was created using a different function called "fspecial" in the previous question)

```
>> fpimg_prewitt = edge(fpimg, 'Prewitt');
% note that single quotes are used around Prewitt
```

18. When using the Sobel or Prewitt filters, the "**direction**" of the edges can be specified. The direction can be "horizontal", "vertical" or "both". Try the "horizontal" and "vertical" options

```
>> fpimg_prewitt_horz = edge(fpimg, 'Prewitt', 'horizontal');
>> fpimg_prewitt_vert = edge(fpimg, 'Prewitt', 'vertical);
```

19. View the input image along with each of the 2 edge images as a **montage**. Based on your observation, what is the difference between the "horizontal" and "vertical" edge detectors?

```
>> close all; % dismiss previous figures
>> figure; % new figure
>> imshowpair(fpimg, fpimg_prewitt_horz, 'montage'); title('Horizontal Edge Image');
>> figure;
>> imshowpair(fpimg, fpimg_prewitt_vert, 'montage'); title('Vertical Edge Image');
```

20. Combine the "horizontal" and "vertical" edges using **imfuse.**

```
>> fused_edges = imfuse(fpimg_prewitt_horz, fpimg_prewitt_vert);
>> close all;
>> figure;
>> imshowpair(fpimg, fused_edges, 'montage'); title('Fused Edge Image');
```

Exercise#1

I. Write a sequence of Matlab (or any other language) commands to do the following in the context of vector and matrix operations:

- Create a column vector "a" with 10 **random** elements; this will be a 10x1 matrix
- Create a column vector "b" such that each element in "a" is related to the corresponding element in "b" as follows:
  - $b = a^3 + a^2 + a + 5$
- Plot the 2D points (a,b)
- Multiply "a" with the transpose of "b"
- Multiply each element in "a" with the corresponding element in "b"
- Find out commands that can perform the following:
  - Compute the inverse of a matrix
  - Compute the eigen vectors and eigen values of a matrix
  - Given a matrix, determine the number of times a certain value occurs in it

You can place the commands that you used along with the corresponding outputs in a word document for submission.

II. Create each of the following filters using the fspecial command:

a. ‘average’
b. ‘laplacian’
c. ‘motion’
d. ‘prewitt’
e. ‘sobel’
f. ‘unsharp’
g. ‘gaussian’.

Apply each filter to an input image (e.g., fingerprint1.tif) and visualize each of the filtered outputs separately. Experiment with 3 different input images (e.g., Face1.jpg, fingerprint2.tif, Iris1_gray.bmp). Your solution should consist of the list of commands used for this exercise along with the 8 output images (original image + 7 filtered images). You can place the commands and images in a single word document for submission.

Upload your solutions in D2L by 11:00pm on October 11, 2021. Note that your solution should include commands and the corresponding outputs. Assignment will not be graded, but you must submit it.