Comp 470
Spring 2020
Lab 2
Assigned: 2/06/2020
Due: 2/13/2020

Name (Print): ____________________

This is a group lab assignment, so you should work with your lab partner and each submit the same solutions.
You are required to show your work and include your source code and MATLAB output on each problem in this assignment.

1. Systems of equations involving only two unknowns can be analyzed graphically. For example, consider the system

$$2x + 3y = 1 \tag{1}$$
$$x - y = 2 \tag{2}$$

(since we only have two unknowns I have switched from using the unknowns x_1 and x_2 to x and y). The first equation, $2x + 3y = 1$ is the equation of a straight line. To see this, note that we can solve for y to get

$$y = \frac{1}{3} - \frac{2}{3}x \tag{3}$$

In the same way $x - y = 2$ is also the equation of a straight line. We shall use MATLAB to draw the graphs.

If you have not started MATLAB, do so now. At the MATLAB prompt type (carefully!)

```
ezplot('1/3 - 2x/2', [-10,10])
grid
```

The first line uses a simple plotting tool in MATLAB to plot the equation in quotes. You should be able to see how the equation $y = 1/3 - 2x/3$ was translated into MATLAB ($*$ is the MATLAB multiplication operator and we only need enter the right hand side). The entry $[-10, 10]$ tells MATLAB to have the x-axis run from -10 to 10. Type `help ezplot` for more details and to see the MATLAB help system. The command `grid` puts up some grid lines on the graph.

Now we shall plot the graphs of the first and second equations together and add a title by typing (replace `My name` by your own name):

```
ezplot('1/3-2*x/3',[-10,10])
hold on
ezplot('x-2',[-10,10])
title('My name')
grid hold off
```

The command `hold on` tells MATLAB to hold the graph and not erase it when we ez-plot a second time (otherwise the first graph would be lost). The `hold off` tells MATLAB to turn off the hold, so that any new graph will overwrite the old graphs.

You can see from the screen that this pair of equations is consistent and has a unique solution.

Now you are to draw three graphs using MATLAB commands like the ones you just used. In each case you should solve each equation for y in terms of x and plot all the graphs on one axis. For each linear system

- Is there a solution? Many solutions? No solution?
- If there is a unique solution, you can give an estimate of it's value by reading off the coordinates of the point where the lines cross.
- If there is a unique solution, use `A\b` to find the unique solution x. Does this agree with your estimate from above?

- On a separate figure you should plot the linear combination of the columns of A that you determined above with `A\b`. Does this point to b?

1. Analyze the following system of two equations in two unknowns (2 × 2 system)

$$\begin{aligned} -x + 2y &= 3 && (4) \\ 2x - 4y &= -6 && (5) \end{aligned}$$

2. Analyze the following system of two equations in two unknowns (2 × 2 system)

$$\begin{aligned} x + y &= 1 && (6) \\ 2x + 2y &= 3 && (7) \end{aligned}$$

3. Analyze the following system of three equations in two unknowns (3 × 2 system)

$$\begin{aligned} x + y &= 1 && (8) \\ 2x - y &= 0 && (9) \\ -x - y &= -1 && (10) \end{aligned}$$

4. Analyze the following system of three equations in two unknowns (3 × 2 system)

$$\begin{aligned} x + y &= 1 && (11) \\ -x - y &= -1 && (12) \\ 2x + 2y &= 2 && (13) \end{aligned}$$

2. Enter the following matrices into MATLAB.

$$A = \begin{bmatrix} 1 & -1 & 2 \\ -3 & 1 & 1 \\ 1 & 4 & -6 \end{bmatrix} \quad (14)$$

$$B = \begin{bmatrix} 0.5 & 0.35 & 0.15 \\ 0.35 & 0.6 & 0.05 \\ 0.15 & 0.05 & 0.8 \end{bmatrix} \quad (15)$$

$$C = \begin{bmatrix} 1 & -1 \\ 1 & 2 \\ -3 & 2 \end{bmatrix} \quad (16)$$

Then perform the following operations and display the output:

1. `2*3`
2. `A*A`
3. `A*B`
4. `B*A`
5. `A*C`
6. `C*A`
7. `2*B`

- Did MATLAB refuse to do any of the requested calculations? Why?
- Does `A*B = B*A` in general?
- What did `2*B` produce? Was that what you expected?

3. For A, B, and C from above perform the following operations and display the output:

   1. `A'`
   2. `B'`
   3. `C'`
   4. `C'*A`
   5. `A*C'`
   6. `(A')'`
   7. `(A' + A)/2`

   - Did MATLAB refuse to do any of the requested calculations? Why?
   - Does `B = B'`? Is B symmetric?
   - What is the relationship between `(A')'` and `A`?

4. Any text message, such as, for example, "I LOVE MY TA," can be translated into a string of numbers by agreeing that each letter will correspond to its position in the alphabet, and a blank space will correspond to the number 27. Thus A corresponds to 1,B corresponds to 2, and M corresponds to 13. Then the message above can be written as "9 27 12 15 22 5 27 13 25 27 20 1" If we did this alone, our message would be quite easy to break. Let us use our sophistication in linear algebra to try to come up with a better encoding procedure.

   Let us pick some invertible 4×4 matrix C with integer coefficients. Enter the following matrix into MATLAB.

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 1 & 1 & 0 \\ 1 & 4 & 2 & 3 \end{bmatrix} \tag{17}$$

   Now to encode our numerical message we simply take the first four numbers in our string and multiply C by it.

```
>> C*[9;27;12;15]
```

   This gives us the first four numbers of the encoded string:
   63 105 48 186

   Thus to encode the rest of the message we keep multiplying C by the next four numbers in the original array.

   Using the matrix above, encode the word "NOON". Note that even though in the pre-encoded numerical string the numbers representing the letter "O" are the same, once we encode the word, the numbers in the encoded message are all distinct. Thus the naive approach of decoding the message under the assumption that each letter is represented by a unique number will fail. Include your input and output in the final report.

   Now we know how to encode a text message, but it is of little use to us if the party to whom we are sending it cannot decode it. Thus we must devise a way of decoding an already-encoded message. That is, given an encoded message we must work in reverse to find the original message. This is done by applying the inverse of the encoding matrix to the encoded string.

Let us decode the encrypted message ”42 51 34 81,” knowing that it was encrypted using the matrix C.

We must first find the decoding matrix. This involves finding the inverse of C. Type in:

```
>> D = inv(C)
```

Then simply multiply D by the column vector representing the encoded message.

```
>> D*[42;51;34;81]
```

Explain why you think we must have an invertible matrix to do the encoding?

The string ”56 83 37 127 42 56 40 83 82 118 55 182 77 119 50 191 48 70 41 121” was obtained by encoding a message with the matrix C. What does the original message say? Include all relevant MATLAB commands with the output in your write up.