OBJECTIVES

The main purpose of this in-class exercise is to write simple MATLAB scripts to perform flow control, specifically, conditional statements and loops, and solve a system of linear equations, i.e., solving `Ax=b`.

NOTES

MATLAB COMMANDS

Throughout this document, the MATLAB commands that you must use in your scripts will appear in **`bold`** face.

COMMENTING

Comment your codes extensively. You can use the percent symbol `%` to enter comments in your scripts. Comments allow the user to understand and follow code easily; it is therefore highly recommended to develop a habit to extensively provide commentary in your codes.

A nice feature you may want to use in your scripts is code sectioning. Code sections allow you to organize, add comments, and execute portions of your code. Code sections begin with double percent signs (`%%`), e.g.,

```
%% Vector Operations
% You can perform a number of binary operations on vectors.
%%
A = 1:3;
B = 4:6;

%% Dot Product
% A dot product of two vectors yields a scalar.
% MATLAB has a simple command for dot products.
s = dot(A,B);

%% Cross Product
% A cross product of two vectors yields a third
% vector perpendicular to both original vectors.
% Again, MATLAB has a simple command for cross products.
v = cross(A,B);
```

CODE COPY-AND-PASTE

If you decide to copy and paste example command(s) presented in lecture slides and in-class exercises in MATLAB, be wary of single quotation marks—you may need to delete and re-enter single quotation marks after pasting the command(s) in MATLAB.

EXERCISE 3: SOLVING LINEAR EQUATIONS

PART A

The purpose of this exercise is to use MATLAB to solve linear equations. Specifically, we would like to solve the equation `Ax=b`, with `A`

```
A =
     3     4     2
     7     0     4
     4     6     0
```

and `b`

```
b =
    2.1500
    2.9500
    2.2000
```

Create a script and name it `matlabExercise3a.m`. In your script,

- Create a matrix called `A` and a vector called `b` that contain the numbers shown above
- Devise a simple conditional statement (a simple **`if-else`** statement) to check if the determinant (**`det`**) of matrix `A` exists and if true, solve for the unknowns, i.e., `x`. If the determinant does not exist, then display the following message to screen: Determinant of A equals 0!

After saving the script as `matlabExercise3a.m`, run the script by typing `matlabExercise3a` at the MATLAB prompt to see the output.

PART B

Let's make the previous script more general. Specifically, let's create a new script that will allow the user to load saved data containing the matrix `A` and the vector `b` to solve for the unknown values `x`.

Before creating the script, and in MATLAB's Command Window, combine matrix `A` and vector `b` data into a new matrix called `D` so that it will look like:

```
D =
    3.0000    4.0000    2.0000    2.1500
    7.0000         0    4.0000    2.9500
    4.0000    6.0000         0    2.2000
```

where the values of vector `b` have been added to the last column of matrix `A`. Do not enter these values manually; take advantage of each variable's dimension (i.e. size of `A` and `b`) and concatenate them to create the new matrix `D`. After the new matrix is created, save matrix `D` into a `.dat` file. Specifically,

```
save data.dat -ascii D
```

Create a new script and name it `matlabExercise3b.m`. In your script,

- Ask the name of the file and store it in a variable called `filename`. To allow your script to interact with the user, i.e., to ask for the name of the file, use MATLAB's **`input`** command (type `help input` or `doc input` at the MATLAB prompt for more information). For example, to have the user enter a filename, you can use:

```
%example using MATLAB's input command
%note the additional 's' used in the input command
%'s' is used if you want to enter string
filename=input('Please enter filename: ', 's');
```

- Check if the filename entered is correct, i.e., check if the filename does exist. Devise a simple **`while`** loop to check this. If the name has been entered incorrectly, then give the user the change to input it again (this must take place within the loop). To check if a filename exists in the current folder the script is being executed from, use MATLAB's **`exist`** command (type `help exist` or `doc exist` at the MATLAB prompt for more information). If the file does not exist within the folder, this command will return a `0`, therefore your **`while`** loop variable should check if the returned value from **`exist`** is equal to zero or not. If it is, the loop continues until the filename is entered correctly. Specifically:

```
%check if entered filename (this is a variable
%storing the name of the file entered by the
%user) is correct
while exist(filename)==0
        %ask the user to re-enter the name
        filename=input('Please re-enter filename: ','s');
end
```

- Load (**`load`**) the data into MATLAB. Note that MATLAB will automatically assign the loaded data to a variable called `data`
- Extract matrix `A` and vector `b` from the variable `data`
- Now solve for the unknowns (simply use previous script from PART A to do this)

EXTRA EXERCISES

Additional exercises are included for extra practice purposes. I encourage everyone to work on them, but first complete the regular exercises—the ones without an X next to their number—then work on the extra exercises.

EXERCISE 1X: CONDITIONAL STATEMENTS

The purpose of this exercise is to ask the user to choose a condition (e.g., 1, 2, etc.) and have that condition (e.g., Condition 1, Condition 2, etc.) displayed to the screen. For this, you need to write a simple conditional statement, using **`if-elseif-else`** structure, that takes the following tasks into account:

- If the user enters 1, i.e., if the input (see below) equals (**`==`**) 1, then the following text is displayed (**`disp`**) on the screen: `You have selected condition 1`

- If the user enters 2, i.e., if the input equals (**==**) 2, then the following text is displayed on the screen: `You have selected condition 2`
- Otherwise, if none of the above numbers are entered, then the following text should be displayed on the screen: `Not a valid condition, sorry!`

To allow your script to interact with the user, i.e., to allow the user to enter a number, use MATLAB's **`input`** command (type `help input` or `doc input` at the MATLAB prompt for more information). For example, to have the user enter a condition, you can use:

```
%example using MATLAB's input command
n=input('Enter condition = ');
```

where `n` is a variable that will store the data entered by the user.

Create a script and name it `matlabExercise1x.m`, then write the necessary commands to implement this conditional statement. After saving the script as `matlabExercise1x`, run the script by typing `matlabExercise1x` at the MATLAB prompt to see the output.

EXERCISE 2X: FOR LOOP

The purpose of this exercise is to ask the user to choose a condition (e.g., 1, 2, etc.) and have that condition (e.g., Condition 1, Condition 2, etc.) displayed to the screen. However, unlike the previous example where the user had to enter his/her choice a single time, this script should allow the user to enter his/her choice three times.

Use the script developed in the previous exercise and wrap the conditional statement portion of the script in a **`for`** loop that runs three times. The following MATLAB snippet should give you an idea as to how your code should be structured:

```
for (ii=1:3)

        %in the lines following this comment, insert conditional
        %statement, where the user is asked to choose a
        %condition and then have that condition displayed to screen

end
```

Create a script and name it `matlabExercise2x.m`, then write the necessary commands to implement this loop structure. After saving the script as `matlabExercise2x.m`, run the script by typing `matlabExercise2x` at the MATLAB prompt to see the output.

EXERCISE 3X: WHILE LOOP

The purpose of this exercise is to ask the user to choose a condition (e.g., 1, 2, etc.) and have that condition (e.g., Condition 1, Condition 2, etc.) displayed to the screen. However, unlike the previous example where the user had entered his/her choice three times, here the user has the option to continue with entering his/her choice as long as they like.

Use the script developed in `matlabExercise1x` and wrap the conditional statement portion of the script in a **`while`** loop. The **`while`** loop should test the condition: `lower(answer) == 'y'`, where **`lower`** is a MATLAB command that converts strings to lowercase (this means that the user can enter either Y or y), and `answer` is a user-defined variable that is initialized before the **`while`** loop with the string 'y', i.e., `answer = 'y'`.

Within the **`while`** loop, you need to first ask the user to enter his/her choice (use **`input`** command), then use the conditional statement to display the appropriate condition to screen, and follow this by asking the user (again, using the **`input`** command but with additional arguments to make sure that a string is entered—read the help documentation for more information) if they wish to continue with entering another choice (note that the last line of your **`while`** loop should be the **`while`** loop control variable). Your code should have the following structure:

```
answer='y';%while-loop control variable initialized

%start while-loop
while (lower(answer)=='y')

    %in the line following this comment, ask user to enter
    %desired condition (use an input command)

    %in the lines following this comment, include conditional statement
    %to check user entry and, depending on the condition, display
    %appropriate string to screen

    %in the line following this comment, ask user if the
    %process should be repeated (use an input command)

end
```

Create a script and name it `matlabExercise3x.m`, then write the necessary commands to implement this loop structure. After saving the script as `matlabExercise3x.m`, run the script by typing `matlabExercise3x` at the MATLAB prompt to see the output.