Engineering 14100
MATLAB 1 ACT: Introduction to MATLAB

Recall the guidelines for activities:

1. You should work as a team; **all** team members will be held responsible for all material.
2. You should work on this Task using one computer for the entire team unless otherwise directed.
3. Also, make sure everybody created header files for MATLAB as part of the pre-activity.

Task 1 (of 5)

Learning Objectives: Perform arithmetic operations in MATLAB (i.e. addition, subtraction, multiplication, division, and exponentiation), accounting for order of operations; Create valid identifiers, accounting for relevant MATLAB rules (e.g. keywords) and code standard.
Computer Operator: Team member with the longest pointer finger

Background: Each programming language has its specific strengths for accomplishing various computational goals by their users. For example, Python's general collection type makes it possible to mix data types in a single collection (lists, tuples, and dictionaries). This makes processing database records easier, such as a personal contact directory containing names, addresses, phone numbers and email. MATLAB, on the other hand, is designed to operate on arrays of numbers and perform numerical processes. This makes it an excellent environment for numerical modeling and analysis. As a result, each environment will be very efficient with some programming actions and very inefficient with others. These exercises explore some of these strengths and weaknesses of various programming languages.

The command window in the MATLAB IDE (Integrated Development Environment) is similar to the Python 3 interpreter environment. They are both interpreters that process lines of code as they are entered. Therefore, you can use the command window to test out specific lines of code. In this exercise, we will go straight to building scripts using the built-in editor in MATLAB, similar to how you generated code with Notepad++ for Python 3 scripts.

Type **`edit`** in the command window. This launches the MATLAB editor/debugger and creates a new .m file. You can type the same commands in the MATLAB editor as you can in the command window. The only difference is that using the editor, you can run a series of commands as often as you like. For this exercise and all future exercises, unless otherwise indicated, you should use the editor and script files to construct and comment your MATLAB code.

Assuming that equations (1) through (9), shown below, are computed sequentially in the order shown, calculate the result of each expression using:

- Python (save your file as `ML1_ACT_Task1_login.py`)
- MATLAB (save your file as `ML1_ACT_Task1_login.m`)

```
1. A = 10
2. B = A ^ 2
3. C = B - A * 2 / 5
4. D = 13 // 3 + 4 / 3
5. E = 4 ^ (3 * 2)
6. F = -4 ^ (1 / 2)
7. M = 4 ^ (i / 2)
8. Z = 133 % 20
9. P = exp(2 * pi)
```

Answer the following questions:

1) What occurs when entering line 4? What are the differences between Python and MATLAB?
2) Why does line 6 cause a problem? What happens in MATLAB?
3) In line 7, how does "i" function? Is it possible to use the letter "i" as a variable?
4) Why does line 8 cause a problem in MATLAB? How would you find a command or a function in MATLAB to compute the modulus?
5) How do you calculate the natural logarithm of a value in MATLAB? How do you call the base of the natural logarithm, e, in MATLAB?

Save your response to these questions to `ML1_ACT_login.pdf`.

Task 1 Files:

1) `ML1_ACT_login.pdf`
2) `ML1_ACT_Task1_login.py`
3) `ML1_Act_Task1_login.m`

Task 2 (of 5)

Learning Objectives: Create and execute simple script comprised of basic MATLAB commands; Output data from a script to the screen in MATLAB; Apply course code standard in development of MATLAB scripts; Comment your code using the percent operator (%) and modularize your code using the (%%) operator in a useful and efficient fashion in MATLAB.
Computer Operator: Team member with the shortest last name

Background: A fuel tank is constructed in the shape of a cylinder. The design engineer that designed the tank provided the dimensions in meters. The construction engineer that is building the tank needs to know the capacity in U.S. gallons, as well as the dimensions in feet.

Part A:
Write a MATLAB script to find the volume of the cylinder in U.S. gallons, as well as the tank dimensions in feet. Assume that the initial measurements are 5 meters in diameter and 10 meters tall.

Comment your code adequately. Start your script by removing any data and output from previous scripts by using `clc` and `clear` prior to performing anything else. Display your final answers to the screen without using `disp` or `fprintf`, e.g. write the following into your script

```
disp('The capacity in U.S. gallons is:')
capacity
```

where `capacity` is a variable that you defined in preceding calculations.

Note: The given values should be assigned at the top of your code and referenced as variables in the body of the code.

Part B:
In the same MATLAB script, perform the same calculations for a second tank that is 6 meters in diameter and 9 meters tall. This time, use `fprintf` to achieve the following output (no decimals for the capacity and one decimal for the dimensions):

```
The capacity is X U.S. gallons.
The tank has a diameter of X.X ft and is X.X ft tall.
```

Save your script as: `ML1_ACT_Task2_`*`login`*`.m`

Note: At this point, creating a flowchart should be done without instructions explicitly asking for it. Append your flowchart to the previously created PDF.

Task 2 Files:
1) `ML1_ACT_Task2_`*`login`*`.m`

Task 3 (of 5)

Learning Objectives Explain the advantages of implementing user-defined functions in MATLAB; Create and implement user-defined functions in MATLAB.
Computer Operator: The person taking the most credit hours next semester.

Background: Functions are a critical component of any language. They can perform a specific set of operations based on values that are passed to it. Any arguments that are defined in the function are identified and used during computation. Once the function performs its actions and possibly returns a value, all of its local variables are dereferenced (i.e. removed) from memory.

Part A:
Answer the following questions:

1. How are functions useful?
2. Describe the process of defining a function in MATLAB. What is the appropriate syntax for calling a function in MATLAB?
3. How are parameters passed between the function and the main program?
4. Is there any fundamental difference between MATLAB and Python declaration of a function?
5. What is the significance of a header in a MATLAB function?
6. What effect have `clc` and `clear` in a function? Should you use these commands there?

Save your answers in a PDF file named:

`ML1_ACT_`*`login`*`.pdf`

Part B:
Write a MATLAB function that will calculate the area of the space between two concentric circles. The function should take as input `ROuter` and `RInner`, corresponding to the outer and inner radius, respectively.

Save your function as:

`ML1_ACT_Task3b_`*`login`*`.m`

Task 1 Files:

1) `ML1_ACT_`*`login`*`.pdf`
2) `ML1_ACT_Task3b_`*`login`*`.m`

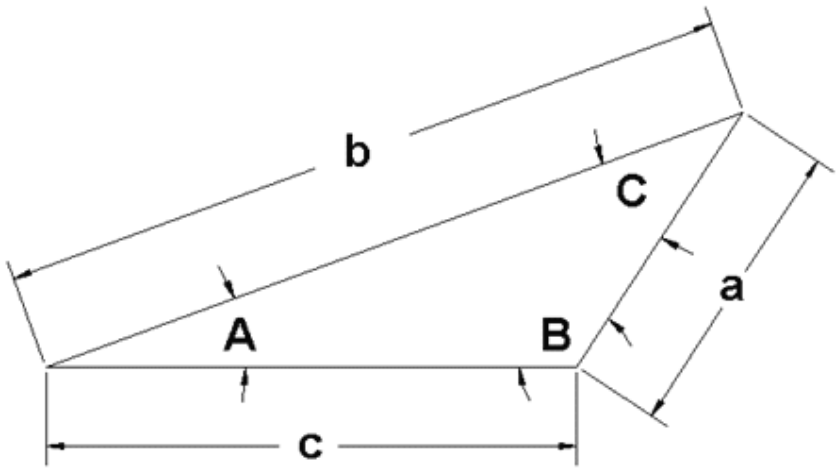
Task 4 (of 5)

Learning Objectives: Create and execute simple functions to perform arithmetic operations in MATLAB (i.e. addition, subtraction, multiplication, division, and exponentiation), accounting for order of operations;

Computer Operator: Team member who slept the least last night

Given 1 side and 2 angles of an oblique triangle (see figure), write a user-defined function in MATLAB that takes these values as inputs and determines the unknown distances, a and c, and the unknown angle, B. The function will print all values to the screen. Further, the function returns a, c and B to the user (see example below). Make sure to add sufficient comments to clarify the order of inputs/outputs and units expected/returned.

Distance b: 1300 meters
Angle A: 25 degrees
Angle C: 37 degrees



Note: Triangle shown is not to scale.

The program should output the lengths a and c to the nearest meter and the angle B to the nearest tenth of a degree using `fprintf`.

Example output (command line input in bold):

>> [a, c, B] = ML1_ACT_Task4_*login*(b, A, C);

```
Distance b: 1300
Angle A: 25
Angle C: 37
Distance a: 622 meters
Distance c: 886 meters
Angle B: 118.0 degrees
>>
```

Task 4 Files:

1) `ML1_ACT_Task4_login.m`

Task 5 (of 5)

Learning Objectives: Identify which resources are available to you to aid in learning MATLAB; Access MATLAB "help" in MATLAB and through online MATLAB documentation.
Computer Operator: Team member with the longest hair

Background: MATLAB has a very powerful set of help tools. By going to Help and then going to MATLAB Help, you will open the interactive help files. They operate in a similar fashion to most Windows help files. By clicking on Contents, you will have access to the entire directory of MATLAB help, all commands, and some examples. Clicking on the index tab allows you to search for details on a known function. The Search tab will allow you to search for a function to do a specific task. These two tools can help you find information on any function.

Use the interactive help files to look for more information on the following functions. Test them out in the command window.

- sin
- exp
- mod
- sqrt
- abs
- pi
- semicolons (in relation to output)

MATLAB also has a command line help utility. In the MATLAB command window, you can type `help` *`command`* where command is the item that you want help with. For instance, typing `help sin` will give you information on the sin function. Use the help command to find out more information on the following commands:

- clc
- clear
- whos
- what
- why
- ans

For **all 13** of the above commands, answer the following:

1) What are the arguments required by the function?
2) What is the appropriate syntax of the argument?
3) Do the arguments have any special units?
4) What is the purpose of the function?

Once you have tested all of the commands, answer the following:

1) Describe what happens when you type "`help ML1_ACT_Task4_`*`login`*".
2) Describe what happens when you type "`help ML1_ACT_Task2_`*`login`*".
3) Describe the difference between the `fprintf` and `disp` commands

Save your answers in your previously created PDF, `ML1_ACT_`*`login`*`.pdf`.

Bonus Activity Submissions

Instructions: Complete and submit **ALL** Task materials associated with this Activity (see 'Submit Files' below). You are allowed to combine the work you and your team completed during the Activity with materials you individually (or as a team) complete outside of class. The Bonus Activity Submission will not be graded and returned to you like a typical assignment. Instead, it will be reviewed, and the bonus point awarded, for its completeness, i.e., for completing ALL the Tasks associated with the Activity. Submitting an incomplete Bonus Activity (something less than all of the Tasks) will be considered an act of **Academic Dishonesty** for which the penalty will be forfeiture of the opportunity to earn future Bonus Activity Submission points.

There are two options for completing the materials for the Bonus Activity Submission:

As an Individual: Combine the work you and your team completed in class with materials you have individually completed outside of class. When submitting an individual Bonus Activity Submission you will append your electric signature (i.e., your typed name) at the top of the file that represents your individual work. Your electronic signature indicates that this is your individual work and you have not collaborated with other individuals (other than the teaching team) to obtain the final materials being submitted – working with other individuals/groups (e.g., discussing ideas and concepts, helping find errors, talking about potential solutions to errors) is permissible up to the point where the work represents a coloration (i.e., working with another person or group to achieve an answer). Any work previously completed by your team should include each team member's electronic signature. The significance of an electronic signature by an individual for team work is stated below.

As a Team or Ad Hoc Group: Combine the work you and your team completed in class with materials your team (or ad hoc group) completed outside of class **(For the Bonus Activity Submission ONLY**: you are allowed to work with any other members of the class to complete the assignment). However, you should exercise care when appending your electronic signature to ensure you are in full compliance). When submitting a Bonus Activity that has been worked on as a team (or ad hoc group) each person will append his/her electronic signature (i.e., his/her typed name) at the top of the file that represents the collaborative work. The electronic signature of each individual implies he/she was an active participant in the preparation of the materials; and has a general understanding of **ALL** the materials being submitted. Even for work submitted as a team, each individual who wishes to receive credit must submit the team's file (with all appropriate signatures) to their own individual assignment drop box.

Submit Files: Submit *all* files electronically via Blackboard to the appropriate box on time.

1) `ML1_ACT_`*`login`*`.pdf`
2) `ML1_ACT_Task1b_`*`login`*`.py`
3) `ML1_ACT_Task1b_`*`login`*`.m`
4) `ML1_ACT_Task2_`*`login`*`.m`
5) `ML1_ACT_Task3b_`*`login`*`.m`
6) `ML1_ACT_Task4_`*`login`*`.m`