CMPE 8 Lab 1: Basic MATLAB Use (Part 1)

Lab Objectives

By the end of this lab you should be able to:

1. **Use** the ‘help’ command in MATLAB to identify how various functions work.
2. **Identify** what a variable is and how to use one in MATLAB.
3. **Use** the ‘plot’ command in MATLAB to create figures.
4. **Use** loops to write more efficient code.

Create Your Working Directory

Creating your own working directory lets you keep all your work in one spot and enables you to easily backup your work (See Robot Automation handbook Section 2.2.1). It is recommended that you create your working directory on a removable flash drive, so if/when you change work stations in the lab you can take your working directory with you. MATLAB can only execute programs that are in its current working directory, so if you write a program but save it somewhere besides your working directory, MATLAB will not be able to find it and issue an error message.

The ‘help’ Command

The ‘help’ command is very useful for learning how various functions and commands work in MATLAB. While many functions in MATLAB work as you would expect them to, there are many subtleties that can trip you up. For example, ‘cos(x)’ is universally used to calculate the cosine of an angle, however it is important to know if the angle should be given in radians or degrees. You can use the command ‘help cos’ to learn how the ‘cos’ function works in MATLAB.

Variables

A variable in MATLAB is essentially a tag that you assign to a value while that value remains in memory. The tag gives you a way to reference the value in memory so that your programs can read it, operate on it, and save it back to memory. When creating variables it is important to remember that the expression on the right hand side of the equal sign is evaluated first and then assigned to the variable on the left hand side. For example, suppose you want to create the variable xyz and assign it the value 5. You would simply type:

\[
xyz = 5
\]

The expression on the right hand side of the equal sign can range from a single value (e.g. \( x = 5 \)) to a complex equation (e.g. \( z = 5x + \sin(y) \)). You can also use the variable on the left hand side of the equal sign in the expression on the right hand side. For example:

\[
x = 5+3
\]

\[
x = x*2
\]

The first statement adds 5 plus 3 and assigns the value 8 to the variable x. The second statement multiplies the value of x (which is 8) by 2 and assigns the new value 16 to x **overwriting the previous value**.
Vectors

Variables in MATLAB can also be assigned an array of values instead of just a single value. When this is the case, the variable is referred to as a vector. In order to create a vector, an array of values enclosed by square brackets is listed on the right hand side of the equal sign. For example, you can assign the values 1, 2, 3 and 4 to the variable y by typing:

\[ y = [1 \ 2 \ 3 \ 4] \]

Another way to create a vector is to use vector notation to create an array of values. In vector notation, an array of values is defined by a starting point, an increment size, and an ending point separated by colons. For example, to assign the values 1, 2, 3 and 4 to the variable y using vector notation you would type:

\[ y = 1:1:4 \]

The first 1 indicates the start of the vector, the second 1 indicates the increment size, and the 4 indicates the end of the vector. So in the example above, the variable y is assigned the value 1 along with every integer up to the value 4 in order.

The ‘plot’ Command

Plotting figures in MATLAB is easy using the ‘plot’ command. This command allows you to plot everything from very simple equations (e.g. \( x = 2*y \)) to more complex equations (e.g. \( z = 5*x^2 + 12*\sin(y) \)) and also allows you to plot multiple curves on the same figure. Once you are familiar with how to use some elementary functions, plotting them will help you visualize your answer and better understand the effects of different variables on those functions.

Loops

Sometimes MATLAB coding becomes tedious and inefficient. If the program you are creating performs the same action over and over again, you can use loops to simplify your code, make it easier to debug and save time. For example, suppose you want to calculate 100 data points for one equation. Instead of writing 100 lines of code that all perform the same action, you can use a for-loop with only a few lines of code and get the same result with less effort. Below is an example of a working for-loop.

\[ x = 0; \]
\[ \text{for } i = 1:2:10 \]
\[ x = x+2; \]
\[ \text{end} \]

In the example above, the variable i is called the loop counter. The loop counter determines how many times the actions inside the for-loop are repeated. The syntax \( i = 1:2:10 \) is used to set the starting point (1), increment size (2), and ending point (10) of the loop counter. In this example, i will take on the values \( [1 \ 3 \ 5 \ 7 \ 9] \) one after another and the for-loop will repeat the action \( x = x+2 \) five times. What is the value of x after the loop is complete? What is the value of i?
Exercises

1. Plot the following:
   a. \( y = 0.1x \) where \( x = 0:2:100 \)
   b. \( y = \sin(x) \) where \( x = 0:0.1:2\pi \)
   c. \( y = \cos(x) \) where \( x = 0:0.1:2\pi \)
   d. Plot all three of the above on one plot.
   *Please submit all four plots (a, b, c and d) for the lab report.

2. Using a \textit{for-loop}, calculate the summation of all the integers from 1 to 10.

3. Create a vector \( x = 0:0.1:2\pi \). Using a \textit{for-loop}, plot ten different curves on the same graph where the first curve is \( y_1 = \sin(x) \), the second curve is \( y_2 = \sin(y_1) \), the third curve is \( y_3 = \sin(y_2) \) and so on. Hint: by using a \textit{for-loop} it is not necessary to create ten separate variables to solve this problem.

\textbf{Note:} It's very \textbf{IMPORTANT} to comment your code. Please make sure to include a brief explanation of the code used to complete each of the exercises.