1. **Introduction**

MATLAB can be used to carryout simple calculations at the command line or in an m-file (e.g. a script). The variables and equations used in these calculations are slightly different from their algebraic counterparts. Some key characteristics in MATLAB (and other programming environments):

* **Names:** While variable names are generally a single character in Algebra they are most often multiple characters in programming. In MATLAB names must start with a letter, be made up of letters, numbers and underscores only (no spaces). On ETCS’s version of MATLAB names of up to 63 characters are allowed.
* **The assignment operator (=) & “equation” arrangement:** The “=” assigns the value resulting from any expression on its ***right-hand side*** (RHS) to the variable named on the ***left-hand side*** (LHS). The LHS must be a storage location that can fit the answer of the expression.
* **Statement Order:** The order of steps must be organized carefully. Before a variable is used on the RHS of an equation it must have received a value sometime earlier in the code.

In addition to scalars, variables can be used to store a range of data types that can be helpful in programming. This includes storing lists of numbers (vectors), tables of numbers (Matrices), complex numbers and text. This tutorial focuses on vector variables. Other types will be considered over the next few weeks.

**Instructions:**

Run all statements that follow the >> prompt, in Command Window without semicolons, will allow you to see the result of each command immediately (do not use a script for the executables that follow the >> prompt). Take your time and look carefully at each example. Try variations on the example given to make sure you understand what each example does. Understanding these techniques is critical to future labs. This should help your understanding.

The tutorial examples on the white part of the page are there to show you how these commands functions. You should run all of these but you do not need to turn them in. The exercises and problems in the shaded boxes must be copied to a word document and turned in. In order to keep your output compact, before starting type:

>> format compact

**Vector variables Tutorial**

1. **Creating Vector Variables**

In programing, a ***vector*** variable is simply a list of numbers contained in one variable. Try each of the following examples at the MATLAB command window.

Creating Row Vectors: There are several ways to create vector variables. Try these examples.

* Making a List >> x = [ 4, 5, 6, 7]
* Setting up regular spacing (Start:Interval:Stop) >> x = 4:1:7
* Creating a fixed number of elements
linspace(start, stop, number of elements) >> x = linspace(4, 7, 4)
* Concatenating Vectors (combining them together) >> a = [4 5]; b =[6 7];
 >> x = [a, b]

Notice all of these examples will result in a ***row vector*** (all numbers lined up in a row) consisting of the four ***elements*** 4, 5, 6 & 7. Vary the numbers in the above lines to see what each form is doing.

Creating Column Vectors: Can also create vectors that are a column:

* by ***transposing*** a row vector: Create the row vector >> x = [ 1, 3, 4] Transpose it to get a column >> y1 = x’
* by using semicolons between numbers to create the column >> y2 = [1; 3; 4]

In a program, it is often necessary to determine the number of elements in a vector. The length and size functions in MATLAB can accomplish this task. Run the following statements and see how the report the size or length of the vector. Compare the results for the row vector x and the column vector y1 (created by the statements above). Describe the difference for the two different inputs for each function.

Row vector input Column vector input

>> size(x) >> size(y1)

>> length(x) >> length(y1)

What is the difference between these two commands (what does each do)? Write a brief answer here.

**I. Vector Creation Exercise:**

Start a word document to copy exercise results into. In the Command Window create the following vectors in MATLAB (do not use semicolons at the end of the lines) :

* A row vector that contains the values 5, 2, 8, and 9
* A row vector contains the integers from 1 to 10.
* A column vector that lists odd integers from 1 to 11.

Copy these exercises to the word document and label as “I. Vector Creation Exercise”.

1. **Addressing Vectors**

Knowing how to address vectors is also critical as more advanced programs are attempted. In MATLAB addresses are integers counting up from one. Vectors are addressed with a single integer or vector of integers in parenthesis after the variable name. For example: for the vector, V1 = [4, 5, 6, 7], the expression V1(3) will return the value 6 (the third element in vector V1).

Vector Addressing Tutorial Exercises: Type the following executables and observe what happens.
Vary the numbers and note the change in results.

>> V = 1:2:9

>> V(4)

>> V(3:5)

>> V(2:end)

>> V(1: end-1)

**II.** **Vector Addressing Exercises**

Carry out the following in the Command Window (without semicolns)

* Create a row vector with the statement y = 6: -2: -6
* Use vector addressing to display the 5th element of this y vector
* Use vector addressing, including the use of end for the last address to display the last three elements in y.

Copy the command window executable and echo print to the word document and label as “II. Vector Addressing Exercises”.

1. **Element-by-Element Calculations with Vectors**

A key advantage of using vectors is they allow us to carryout multiple similar calculations with one expression. For example, in the studio first studio we came up with Equation 1 for the combination of a resistor and a power source that would ensure that the current 20 mA through a 100 Ω resistor.

VS = 0.02 RS + 2 (1)

Where: Vs = supply voltage [volts]
 Rs = the second resistor (Ohms)

It would be handy to calculate several possible combinations at once. This is easy in MATLAB. Try the following:

>> Rs = [100:50:400]
>> Vs = 0.02\*Rs + 2

Notice that multiple answers are easily calculated. In order to be able to fully use vector calculations it is necessary to understand the details of how MATLAB handles them.

Only one vector: In most cases if only one vector is being used (as in the example above) no change is needed to the equations used for scalar calculations. Try the following calculations to see how this works:

>> Rs+2

>> Rs/2

>> Rs\*2

Many of MATLAB’s built in functions handle single vectors without much complication. The sine function is a good example of this. Try the following:

>> Theta = 0:pi/2:2\*pi
>> sin(Theta)

Working with two vectors is equally straight forward for addition and subtraction. Try these:

>> x = [ 3 8 2];

>> y = [1 2 3];

>> z = x + y

>> z – y

These above calculations are done “***element-by-element***”. This means the first element of the first vector is combined with the first element of the second vector, the second element of the first vector is combined with the second element of the second vector, and so on. This element-by-element approach is not automatic for multiplication, division or powers. Notice what happens when the above two vectors are multiplied:

>> x\*y

It does not work so well. In order to get an element-by-element calculation for multiplication, division or powers the “***dot operators***” are used (i.e., ***.\****, ***./*** and ***.^***). Try the following:

>> x.\*y

>> x./y

>> x.^y

Do take note that the period proceeds the standard operator, hence “dot-operators”.

**III.** **Vector Calculations Exercise:**

Carry out the following in the Command Window (without semicolons) using appropriate operators:

* Make sure the vector V is still in your workspace.
* Use a dot operator to create a new vector Vs that contains the square of each element in V.
* Divide Vs by V using a dot operator. Notice the resulting vector.
* Use single expression in MATLAB to calculate, element-by-element, V times Vs with that quantity divided by 12.

 **Key Application – Vector capable scripts**: Element-by-Element calculations are particularly useful for allowing scripts to do multiple calculations through the use of vector inputs. Figure 1 shows an example of a vector capable version of the cylinder script example on the summary sheet from the first week. Notice the use of dot operators in the two equations. Dot operators are not needed when a vector is multiplied by a scalar (e.g., pi\*r) but are used for two vectors (e.g., r.\*(r + h) ). This function is still capable of scalar input. It can also handle input of two equal length vectors, or the input of a vector and a scalar.

**IV. Vector Script Exercise:**

In the word document add the label “IV. Vector Script Exercises”. Download CylScript2.m from the lab website and save to your working directory. Carry out the following in the Command Window:

1. Create three vectors: 1) a vector of radii, r1 =[2, 2, 6, 6],
 2) a vector of heights, h1 = [1, 4, 1, 4]
 3) another vector of radii, r2 that is the integers from 1 to 4.
2. Run CylScript2 (available online and shown in Figure 1) when it requests input use r1 as the radius input and h1 as the height input briefly describe what the program has done in this case in your work document
3. Run CylScript2 using r2 as the radius input and the scalar value 4 as the height input.

Copy the results to the word document.

In your document briefly describe what the program has done in each case.

**Figure 1:** This script illustrates the use of dot operators for element-by-element calculations. This allows the script to handle vector input and to do multiple cases at once. The script also includes an example of interactive input and output titling.

% Program CylScript2.m

% S. Scott Moor January 2017

% This program will calculate the Volume and Surface Area of a right circular cylinder
% based on radius and height. It is capable of handling vectors of dimensions as input.

%

% Variables used:

% Inputs:

% r = radius (length units)

% h = height (same length units)

%

% Outputs

% V = volume of cylinder (cubic length units)

% SA = surface Area of cylinder (squared length units)

% input section - interactive input from command window

r = input(‘Enter the radius of the cylinder: ‘)

h = input(‘Enter the height of the cylinder: ‘);

% Calculation & display of volume

disp(‘Volume of Cylinder (cubic meters):’)

V = pi\*r.^2.\*h 🡸 notice the dots in front of the **^** and the second **\***

 The first **\*** does not require a dot because pi is a scalar

% Calculation & display of surface area

disp(‘Surface Area of Cylinder (square meters):’)

SA = 2\*pi\*r.\*(r + h) 🡸 notice the dot in front of the last **\*.** An \* is not

 needed for first cases because 2 & pi are scalars

**Script I/O Tutorial: Interactive Input and Labeling Output**

1. **Adding labeled output and interactive input**

In simple scripts written so far in this course the inputs have been “hard coded”, i.e., written right into the code. It is helpful to set up the script so the user can supply the input values with out opening the script and editing the code. Notice the CylScript2 program in figure 1 uses a different approach. It uses MATLAB’s ***input*** function. The syntax for this function is:

 x = input('question program will ask user ');

When a script is run, this function will display the text that is in quotes in the Command Window. It will wait for the user to provide a response. When the user types the answer and presses the enter key, the user response will be transferred to the variable on the LHS of the equal sign (i.e., into the x variable in the example above). N.B., notice that the last character in the example above is a space. Including this separates the users answer from the programs question or prompt. It is a good practice to include this.

The CylScript2 also titles its output using MATLAB’s ***disp*** function (short for display). The format for the disp function is:

disp(‘type the desired text here’)

Whatever is enclosed in the single quotes will be displayed in the Command Window before the commands that follow it are executed. For both functions it is important to notice the exact syntax (i.e., punctuation). The desired display text must be inside of single quotes which are in the parentheses following the function name.

**V. Problem:** **Vector Capable Script with Interactive input**

For lab 1 you developed a script to calculate the area and perimeter of a shape consisting of a rectangle and a semicircle. Start with the script you developed and convert it so:

* + It has interactive input (using the input function) for L & W, and
	+ It will allow vector input (by adding dot operators where needed). Be sure to note vector capability in the intro comments and/or input prompts. See example in Figure 1.

Include execution for three input cases:

1. L and W are both scalars
2. L and W are equal length vectors
3. L is a vector and W is a scalar

Add the code and the three execution cases to the word document.

When complete print out the word document and turn it in at the beginning of your lab period during the week of January 27th

**List of Required Deliverables**  **Due:** Before the start of Lab, the week of January 27th

1. Vector Creation Exercise:
2. Vector Addressing Exercises
3. Vector Calculations Exercise:
4. Vector Script Exercise:
5. Problem: Vector Capable Script with Interactive input