# MATLAB Tutorial

## Chapter 1. Basic MATLAB commands
## 1.1 Basic scalar operations

First, let's talk about how we add comments (such as this line) to a program. Comments are lines of text that we want to add to explain what we are doing, so that if we or others read this code later, it will be easier to figure out what the code is doing. In a MATLAB file, if a percentage sign, , appears in a row of text, all of the text following the sign is a comment that MATLAB does not try to interpret as a command. First, let us write a message to the screen to say that we are beginning to run section 1.1.

The command disp('string') displays the text string to the screen.
**disp('Beginning section 1.1 ...')**

Next, we set a variable equal to one.
**x=1**

This command both allocates a space in memory for the variable x, if x has not already been declared, and then stores the value of 1 in the memory location associated with this variable. It also writes to the screen "x = 1". Usually, we do not want to clutter the screen with output such as this, so we can make the command "invisible" by ending it with a semi-colon. As an example, let us use the following commands to "invisibly" change the value of x to 2 and then to write out the results to the screen. **x=2;** this changes the value of x but does not write to the screen disp('We have changed the value of x.');

Then, we display the value of x by typing "x" without a semi-colon.
**x**

Now, let's see how to declare other variables.
**y = 2*x**; This initializes the value of y to twice that of x
**x = x + 1**; This increases the value of x by 1.
**z = 2*x**; This declares another variable z.

z does not equal y because the value of x changed between the
times when we declared each variable.
**difference = z - y**

Next, we want to see the list of variables that are stored in memory. To do this, we use the command "who".
**who;**

We can get more information by using "whos".
**whos;**

These commands can be used also to get information about only certain variables.
**whos z difference;**

Let us say we want to get rid of the variable "difference".
We do this using the command "clear".
**clear difference;**
**who;**

Next, we want to get rid of the variables x and y.
Again, we use the command "clear".

**clear x y;**
**who;**

It is generally good programming style to write only one command per line; however, MATLAB does let you put multiple commands on a line.
**x = 5; y = 13; w = 2*x + y; who;**

More commonly one wishes to continue a single command across multiple lines due to the length of the syntax. This can be accomplished by using three dots.
**z = 2*x + ...**
**y**

Finally, when using clear we can get rid of all of the variables at once with the command "clear all".
**clear all;**
**who;** It does not print out anything because there are no variables.

### 1.2. Basic vector operations

The simplest, but NOT RECOMMENDED, way to declare a variable is by entering the components one-by-one.
**x(1) = 1;**
**x(2) = 4;**
**x(3) = 6;**
**x** display contents of x

It is generally better to declare a vector all at once, because then MATLAB knows how much memory it needs to allocate from the start. For large vectors, this is much more efficient.
**y = [1 4 6]** does same job as code above

Note that this declares a row vector. To get a column vector, we can either use the transpose (adjoint for complex x) operator **xT = x';** takes the transpose of the real row vector x or, we can make it a column vector right from the beginning
**yT = [1; 4; 6];**

To see the difference in the dimensions of a row vs. a column vector, use the command "size" that returns the dimensions of a vector or matrix.
**size(xT)**
**size(y)**
**size(yT)**
The command length works on both row and column vectors.
**length(x), length(xT)**

Adding or subtracting two vectors is similar to scalars.
**z = x + y**
**w = xT - yT**

Multiplying a vector by a scalar is equally straight-forward.
**v = 2*x**
**c = 4;**
**v2 = c*x**

We can also use the . operator to tell MATLAB to perform a given operation on an element-by-element basis. Let us say we want to set each value of y such that y(i) = 2*x(i) + z(i)^2 + 1. We can do this using the code
**y = 2.*x + z.^2 + 1**