# Introduction to MATLAB and Octave

*The goal of this section is to help you gain familiarity with the fundamental tasks that can be accomplished with MATLAB and Octave: defining vectors, computing functions, and plotting.*

*You'll find many good MATLAB/Octave references online. I particularly recommend:*

- Cleve Moler, *Numerical Computing with MATLAB*, which you can access at `http://uk.mathworks.com/moler/chapters`
- Tobin Driscoll, *Learning MATLAB*, which you can access through the NUI Galway library portal.

**MATLAB** is an interactive environment for mathematical and scientific computing. It the standard tool for numerical computing in industry and research. **MATLAB** is not free, however **Octave**, which is free, is largely compatible. In this document any reference to "MATLAB" can be interpreted as "MATLAB/Octave"

MATLAB stands for <u>Mat</u>rix <u>Lab</u>oratory. It specialises in matrix and vector computations, but includes functions for graphics, numerical integration and differentiation, solving differential equations, etc.

MATLAB differs from most significantly from, say, Maple, by not having a facility for abstract computation.

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

## The Basics

MATLAB an *interpretive* environment – you type a command and it will execute it immediately.

1. In MATLAB, everything is a *matrix*. A scalar variable is just a $1 \times 1$ matrix. To check this set, say, $t = 10$, and use the `size()` command to find the numbers of rows and columns of $t$.

2. To declare a row-vector array, try:
   `x = [-4, -3, -2, -1, 0, 1, 2, 3, 4]`

   Or, more simply,
   `x = -4:4`

   To access, say, the 3rd entry
   `x(3)`

3. If we want to define a vector $x = (a, a+h, a+2h, \ldots, b)$, we could use `x = a:h:b;`
   For example
   `>> x=10:-2:0` gives $x = (10, 8, 6, 4, 2, 0)$.

4. We usually like to think of vectors as *column* vectors. To define one, try
   `x = [1;2;3]`
   Or you can take the transpose of a row vector: `x = [1,2,3]';`

5. If you put a semicolon at the end of a line of MATLAB, the line is executed, but the output is not shown. (This is useful if you are dealing with large vectors). If no semicolon is used, the output is shown in the command window.

6. We'll often want to run a collection of commands repeatedly. So, rather than type them individually, create a file with the following code

   ```
   x=-4:4
   for i=1:9
     y(i) = cos( x(i) );
   end
   plot(x,y);
   ```

   Save this as, say `class1.m`. To execute it, just type `class1` in the MATLAB command window.

   Your file is an example of a MATLAB *script file.*

7. If the picture isn't particularly impressive, then this might be because MATLAB is actually only printing the 9 points that you defined. To make this more clear, use
   `>> plot(x, y, '-o')`
   This means to plot the vector $y$ as a function of the vector $x$, placing a circle at each point, and joining adjacent points with a straight line.

8. The plot generated is not particularly good one. The points plotted are a unit apart. To get a better picture, try "easy plot" `ezplot(@cos,[-4,4])`

9. *Use the* `>> help` *menu to find out more about the* `ezplot` *function, and how to use it.*

10. The script file from Part (5) is a little redundant. In MATLAB, most functions can take a vector or matrix as an argument. So, in fact, we can just use `y = cos( x )`
    which sets $y$ to be a vector such that $y_i = \cos(x_i)$.

11. MATLAB has most of the standard mathematical functions: `sin`, `cos`, `exp`, `log`, `sqrt`, etc.
    In each case, write the function name followed by the argument in round brackets, e.g.,
    `>> exp(x)` for $e^x$.

12. The `*` operator performs matrix-matrix multiplication. For element-by-element multiplication use `.*` For example, `y = x.*x` sets $y_i = (x_i)^2$. So does `y = x.^2`.
    Similarly, `y=1./x` sets $y_i = 1/x_i$.

13. To define function in terms of *any* variable, type:
    ```
     >> F = @(x)(x.^2 -2);
    ```
    Now you can use this *function* as follows:
    ```
    >> plot(x, F(x));
    ```
    Take care to note that MATLAB is *case sensitive.*

    In this last case, it might be helpful to also observe where the function cuts the $x$-axis. That can be done by also plotting the line joining, for example, the points $(0, 0)$, and $(3, 0)$:
    ```
    >> plot(x,F(x), [0,3], [0,0]);
    ```