Fundamentals of MATLAB

Goals

1. Learning how real and complex numbers are assigned to variables.
2. Learning how vectors and matrices are assigned values using simple assignment, the colon operator, and the `linspace` and `logspace` functions.
3. Understanding the priority rules for constructing mathematical expressions.
4. Gaining a general understanding of built-in functions and how you can learn more about them with MATLAB's Help facilities.
5. Learning how to use vectors to create a simple line plot based on an equation

MATLAB Environment

MATLAB is a computer program that provides the user with a convenient environment for performing many types of calculations. In particular, it provides a very nice tool to implement numerical methods.

This is hands-on exercise, and you need to sit in front of your computer and follow the commands.

MATLAB uses three primary windows:

- Command window. Used to enter commands and data.
- Graphics window. Used to display plots and graphs.
- Edit window. Used to create and edit m-files.

After starting MATLAB, the command window will open with the command prompt being displayed

```
>>
```

MATLAB operates in a sequential fashion as you type in commands line by line: Try the following commands

```
>> 36 + 72
```
(the output will be `ans = 108`)

```
>> 85 - 47

>> 85*47

>> 141/47

>> 13^15

>> a = 4
```
(the output will be `a = 4` - number 4 is assigned to variable `a`)

```
>> a = 4;
```
(the output will be nothing but number 4 is assigned to variable `a`)

```
>> a = 4,A = 6;x = 1;
```
(several commands on the same line separated by commas or semicolons)

^	Exponentiation	
–	Negation	
* /	Multiplication and division	
\	Left division[2]	Applies to matrix algenra
+ –	Addition and subtraction	

Try the following and see the output:

```
>> y = -4^2

>> y =(-4)^2

>> 4^2^3

>> 4^(2^3)

>> (4^2)^3

>> 2*-4

>> 2*(-4)
```

```
>> 2^-4

>> 2^(-4)
```

MATLAB is case-sensitive!

A complex value can be assigned simply in either following ways:

```
>> x = 2 +i*4        (the output will be  x = 2.0000 +4.0000i)

>> x = 2 +j*4        (the output will be  x = 2.0000 +4.0000i)
```

There are several predefined variables in MATAB, such as `pi` for the number π, that cannot be used as a variable name.

By default, MATLAB displays four decimal places. If you desire additional precision, use `format long` and `format short`. Enter the following and see the outputs:

```
>> pi

>> format long

>> pi

>> format short

>> pi
```

Table 1. Summary of the format commands

type	Result	Example
short	Scaled fixed-point format with 5 digits	3.1416
long	Scaled fixed-point format with 15 digits for double and 7 digits for single	3.14159265358979
short e	Floating-point format with 5 digits	3.1416e+000
long e	Floating-point format with 15 digits for double and 7 digits for single	3.141592653589793e+000
short g	Best of fixed- or floating-point format with 5 digits	3.1416
long g	Best of fixed- or floating-point format with 15 digits for double and 7 digits for single	3.14159265358979
short eng	Engineering format with at least 5 digits and a power that is a multiple of 3	3.1416e+000
long eng	Engineering format with exactly 16 significant digits and a power that is a multiple of 3	3.14159265358979e+000
bank	Fixed dollars and cents	3.14

Arrays, Vectors, and Matrices

A row vector can be assigned as follows:

```
>> a = [1 2 3 4 5]
```

A column vector can be entered in several ways. Try them and see the output.

```
>> b = [2;4;6;8;10]
>> b = [2
        4
        6
        8
        10]
```

`>> b = [2 4 6 8 10]'` (by transposing a row vector with the ' operator)

A matrix of values can be assigned in several ways. Try them and see the outputs:

```
>> A = [1 2 3; 4 5 6; 7 8 9]
>> A = [1 2 3
            4 5 6
            7 8 9]
```

`>> A = [[1 4 7]' [2 5 8]' [3 6 9]']` (by concatenating the vectors representing each column)

```
>> b(4), A(2,3)
>> E = zeros(3,5), U=ones(3,5)
```

The colone operator: If a colon is used to separate two numbers, MATLAB generates the numbers between them using an increment of one:

`>> t = 1:5` (the output is `t = 1 2 3 4 5`)

If colons are used to separate three numbers, MATLAB generates the numbers between the first and third numbers using an increment equal to the second number:

`>> t = 1:0.5:2.5` (the output is `t = 1.0000 1.5000 2.0000 2.5000`)

The `linspace` and `logspace` Functions: The `linspace(x1,x2,n)` function generates `n` equally spaced points between `x1` and `x2`.

```
>> linespace(0,1,3)          (output: ans=0.0000   0.5000 1.0000)
```

The `logspace(x1, x2, n)` function generates n logarithmically equally spaced points between decades 10^{x1} and 10^{x2}.

```
>> logspace(-1,2,4)  (output: ans= 0.1000 1.0000 10.0000 100.0000)
```

Very long lines can be continued by placing an ellipsis (three consecutive periods) at the end of the line to be continued.

```
>> a = [1 2 3 4 5 ...

        6 7 8]
```

The strings can be represented within single quotation marks. Try the following:

```
>> f = 'Miles';

>> s = 'Davis';

>> x = [f s]
```

Some useful string functions.

Function	Description
`n=length(s)`	Number of characters, n, in a string, s.
`b=strcmp(s1,s2)`	Compares two strings, s1 and s2; if equal returns true (`b = 1`). If not equal, returns false (`b = 0`).
`n=str2num(s)`	Converts a string, s, to a number, n.
`s=num2str(n)`	Converts a number, n, to a string, s.
`s2=strrep(s1,c1,c2)`	Replaces characters in a string with different characters.
`i=strfind(s1,s2)`	Returns the starting indices of any occurrences of the string s2 in the string s1.
`S=upper(s)`	Converts a string to upper case.
`s=lower(S)`	Converts a string to lower case.

Try the following and see the outputs:

```
>> x1='Canada'; x2='Mexico'; x3 ='USA'; x4='2010'; x5=810;
```

```
>> strcmp(a1,a2)

>> strcmp(x2,'Mexico')

>> str2num(x4)

>> num2str(x5)

>> strrep

>> lower

>> upper

>> disp(sprintf('Yo\nAdrian!'))
```
(display strings in multiple lines)

Matrix operations: If the dimensions match, the inner and cross (outer) products of vectors will be performed by the following commands:

```
>> a=[1 2 3]; b=[4;5;9];

>> a*b
```
(inner product)

```
>> cross(a,b)
```
(cross product)

Note that if the multiplication cannot be performed, MATLAB returns an error as follows:

```
??? Error using ==> mtimes

Inner matrix dimensions must agree.
```

For the matrices, try the following commands and see the outputs:

```
>> a = [1 2 3]; A=[2 -1 0;3 5 1;-4 5 3];

>> A*a

>> a*A

>> A*A
```
(regular matrix multiplication)

```
>> A^3
```
(returns A*A*A)

```
>> A.^3
```
(returns the cube of each element of A – an element-wise operation)

MATLAB and its Toolboxes have a rich collection of built-in functions. Type `help`

function name in the command line to learn more about that function. Try the following:

```
>> help log

>> help elfun

>> log(A)

>> sqrtm(A)                    (returns the m-th square root of
the matrix A)

>> B = [-2.35 4.86 1.55 0.28];

>> round(B), ceil(B), floor(B), sum(B), max(B), min(B)

>> mean(B), prod(B), sort(B), length(B)
```

Graphics in MATLAB: MATLAB allows graphs to be created quickly and conveniently. Try the following:

```
>> t = [0:2:20]'; g = 9.81; m = 68.1; cd = 0.25;

>> v = sqrt(g*m/cd)*tanh(sqrt(g*cd/m)*t);

>> plot(t, v)                  (the output is a 2D solid thin blue graph)
```

You can customize the graph a bit with commands such as the following:

```
>> title('Plot of v versus t')

>> xlabel('Values of t')

>> ylabel('Values of v')

>> grid
```

To plot each point with a symbol, you can include a specifier enclosed in single quotes:

```
>> plot(t, v, 'o')
```

Try the specifiers or their combinations in the following table, for example:

```
>> plot(t, v, 's--r')
```

Specifiers for colors, symbols, and line types.

Colors		Symbols		Line Types	
Blue	b	Point	.	Solid	-
Green	g	Circle	o	Dotted	:
Red	r	X-mark	x	Dashdot	-.
Cyan	c	Plus	+	Dashed	--
Magenta	m	Star	*		
Yellow	y	Square	s		
Black	k	Diamond	d		
White	w	Triangle(down)	v		
		Triangle(up)	^		
		Triangle(left)	<		
		Triangle(right)	>		
		Pentagram	p		
		Hexagram	h		

You can also control the line width as well as the marker's size and its edge and face

```
>> plot(x,y,'--dc','LineWidth', 2,'MarkerSize',10,...
            'MarkerEdgeColor','k','MarkerFaceColor','m')
```

Multiple plots on the same plot:

```
>> w = sqrt(8*m/(3*cd))*tanh(nthroot(g*cd/m,3)*t);
>> plot(t, v)
>> hold on
>> plot(t, w)
```

In addition to hold, another handy function is subplot, which allows you to split the graph window into subwindows or panes. It has the syntax

```
>> subplot(m, n, p)
```

For example:

```
>> subplot(1,2,1); plot(t,v)
>> subplot(1,2,2); plot(t,w)
```

The simplest manifestation to generate three-dimensional plots is to use `plot3(x,y,z)` where x, y, and z are three vectors of the same length. Try the following and see the outputs:

```
>> t = 0:pi/50:10*pi;
>> subplot(1,2,1); plot(sin(t),cos(t))      (2D circle centered at origin)
>> axis square
>> subplot(1,2,2); plot3(sin(t),cos(t),t)      (graph of helix curve)
```

Practice problems:

1. What is displayed when the following MATLAB statements are executed?

   a) `A = [1 2; 3 4; 5 6]; A(2,:)'`

   b) `y = [0:1.5:7]'`

   c) `a = 2; b = 8; c = 4; a + b / c`

2. The MATLAB humps function defines a curve that has 2 maxima (peaks) of unequal height over the interval $0 \leq x \leq 2$,

$$f(x) = \frac{1}{(x-0.3)^2 + 0.01} + \frac{1}{(x-0.9)^2 + 0.04} - 6$$

   Use MATLAB to generate a plot of $f(x)$ versus x with $x = [0{:}\,1/256{:}\,2]$.

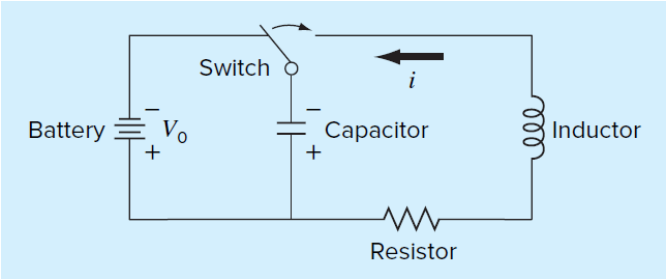
3. Use colon notation to create vectors identical to the following created with the `linspace` function:

   a) `v = linspace(-2,1.5,8)`

   b) `r = linspace(8,4.5,8)`

4. A simple electric circuit consisting of a resistor, a capacitor, and an inductor is depicted in the following figure. The charge on the capacitor $q(t)$ as a function of time can be computed as

$$q(t) = q_0 e^{-Rt/(2L)} \cos\left(\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}\, t\right)$$



where t = time, q_0 the initial charge, R = the resistance, L = inductance, and C = capacitance. Use MATLAB to generate a plot of this function from $t = 0$ to 0.8, given that q_0 = 10, R = 60, L = 9, and C = 0.00005.

5. It is general practice in engineering and science that equations be plotted as lines and discrete data as symbols. Here are some data for concentration (c) versus time (t) for the photodegradation of aqueous bromine:

***t*, min**	10	20	30	40	50	60
***c*, ppm**	3.4	2.6	1.6	1.3	1.0	0.5

These data can be described by the following function:

$$c = 4.84e^{-0.034t}$$

Use MATLAB to create a plot displaying both the data (using diamond-shaped, filled-red symbols) and the function (using a green, dashed line). Plot the function for t = 0 to 70 min.

6. The following figure shows a uniform beam subject to a linearly increasing distributed load. The deflection y (m) can be computed with

$$y = \frac{w_0}{120EIL}(-x^5 + 2L^2x^3 - L^4x)$$

where E = the modulus of elasticity and I = the moment of inertia (m^4). Employ this equation and calculus to generate MATLAB plots of the following quantities versus distance along the beam:

a) displacement (y),

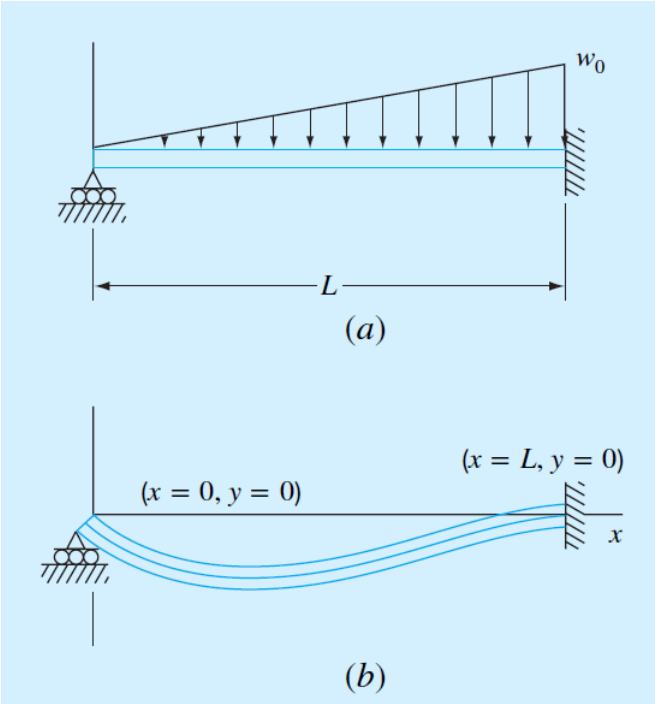
b) slope [$\theta(x) = dy/dx$],

c) moment [$M(x) = EI\ d^2y/dx^2$],

d) shear $[V(x) = EI\ d^3y/dx^3]$, and

e) loading $[w(x) = -EI\ d^4y/dx^4]$.

Use the following parameters for your computation: L = 600 cm, E = 50,000 kN/cm^2, I = 30,000 cm^4, wo = 2.5 kN/cm, and Δx = 10 cm. Employ the subplot function to display all the plots vertically on the same page in the order **(a)** to **(e)**. Include labels and use consistent MKS units when developing the plots.



Resources:

Chapra, Steven C. (2017). Numerical Methods with MATLAB for Engineers and Scientists, 4th Ed. McGraw Hill.