Lecture one
Multimedia Systems

Introduction to MATLAB

Part One

By: Eng. Haydara Mahmoud.

Damascus University – Faculty of Informatics - 2019

Using Matlab as a calculator:

```
>> 1+2*3
ans =
     7

>> x = 1+2*3
x   =
    7

>> 4*x
ans   =
      28.0000
```

Table 1.1: Basic arithmetic operators

Symbol	Operation	Example
+	Addition	$2+3$
−	Subtraction	$2-3$
*	Multiplication	$2*3$
/	Division	$2/3$

```
variable name = a value (or an expression)
```

Overwriting variable:

```
>> t = 5;
>> t = t+1
t    =
     6
```

you can suppress the numerical output by putting a semicolon

Error messages:

```
>> x = 10;
>> 5x
??? 5x
      |
Error: Unexpected MATLAB expression.
```

Formula presentation in MATLAB:

$$\frac{1}{2+3^2}+\frac{4}{5}\times\frac{6}{7}$$

```
>> 1/(2+3^2)+4/5*6/7
ans   =
      0.7766
```

```
>> 1/2+3^2+4/5*6/7
ans   =
      10.1857
```

Miscellaneous commands:

- To clear the Command Window, type `clc`
- To abort a MATLAB computation, type `ctrl-c`

Mathematical Functions:

`cos(x)`	Cosine	`abs(x)`	Absolute value
`sin(x)`	Sine	`sign(x)`	Signum function
`tan(x)`	Tangent	`max(x)`	Maximum value
`acos(x)`	Arc cosine	`min(x)`	Minimum value
`asin(x)`	Arc sine	`ceil(x)`	Round towards $+\infty$
`atan(x)`	Arc tangent	`floor(x)`	Round towards $-\infty$
`exp(x)`	Exponential	`round(x)`	Round to nearest integer
`sqrt(x)`	Square root	`rem(x)`	Remainder after division
`log(x)`	Natural logarithm	`angle(x)`	Phase angle
`log10(x)`	Common logarithm	`conj(x)`	Complex conjugate

Predefined constant values:

`pi`	The π number, $\pi = 3.14159\ldots$
`i,j`	The imaginary unit i, $\sqrt{-1}$
`Inf`	The infinity, ∞
`NaN`	Not a number

Vector presentation:

The vectors $\mathbf{x} = (1, 2, 3, 4, 5, 6)$

```
>> x = [1 2 3 4 5 6];
```

```
>> w = [1;4;7;10;13]
w  =
     1
     4
     7
     10
     13
```

```
>> v = [1 4 7 10 13]
v  =
     1     4     7     10     13
```

```
>> w = v'
w  =
     1
     4
     7
     10
     13
```

```
>> v(1:3)
ans =
     1     4     7
```

```
>> v(3,end)
ans =
     7     10     13
```

`>> v(:)` produces a column vector

Entering Matrix:

A matrix is an array of numbers. To type a matrix into MATLAB you must

- begin with a square bracket, [
- separate elements in a row with spaces or commas (,)
- use a semicolon (;) to separate rows
- end the matrix with another square bracket,].

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

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

Elements in Matrix:

```
>> A(2,1)
ans   =
      4
```

```
A   =
     1     2     3
     4     5     6
     7     8     9
```

Changing an element in matrix:

```
>> A(3,3) = 0
A   =
     1     2     3
     4     5     6
     7     8     0
```

Colon operator:

Often we must deal with matrices or vectors that are too large to enter one element at a time. For example, suppose we want to enter a vector x consisting of points $(0, 0.1, 0.2, 0.3, \cdots, 5)$. We can use the command

```
>> x = 0:0.1:5;
```

The row vector has 51 elements.

Linear spacing:

```
y = linspace(a,b)
```

generates a row vector `y` of 100 points linearly spaced between and including `a` and `b`.

```
y = linspace(a,b,n)
```

generates a row vector `y` of `n` points linearly spaced between and including `a` and `b`. This is useful when we want to divide an interval into a number of subintervals of the same length. For example,

```
>> theta = linspace(0,2*pi,101)
```

divides the interval $[0, 2\pi]$ into 100 equal subintervals, then creating a vector of 101 elements.

Colon operator in matrix:

```
A =
     1     2     3
     4     5     6
     7     8     0
```

```
>> A(2,:)
ans =
     4     5     6
```

```
>> A(:,2:3)
ans =
     2     3
     5     6
     8     0
```

Deleting a row or column in matrix:

A row or a column of a matrix can be deleted by setting it to a *null* vector, [ ].

```
>> A(:,2)=[]
ans   =
      1    3
      4    6
      7    0
```

Putting all elements in one column:

```
A    =
     1     2     3
     4     5     6
     7     8     0
```

```
>> A(:)
ans =
     1
     2
     3
     4
     5
     6
     7
     8
     0
```

Creating a sub-matrix:

To extract a *submatrix* B consisting of rows 2 and 3 and columns 1 and 2 of the matrix A, do the following

```
>> B = A([2 3],[1 2])
B    =
     4    5
     7    8
```

To interchange rows 1 and 2 of A, use the vector of row indices together with the colon operator.

```
>> C = A([2 1 3],:)
C    =
     4    5    6
     1    2    3
     7    8    0
```

- `A(:,j)` is the `j`th column of `A`, while
- `A(i,:)` is the `i`th row, and
- `A(end,:)` picks out the last row of `A`.

The keyword `end`, used in `A(end,:)`, denotes the last index in the specified dimension.

```
A  =
     1     2     3
     4     5     6
     7     8     9
```

```
>> A(end:-1:1,end)
ans =
     9
     6
     3
```

Adding a row to a matrix:

```
A  =
     1     2     3
     4     5     6
```

```
>> A = [A(1,:);A(2,:);[7 8 0]]
A  =
     1     2     3
     4     5     6
     7     8     0
```

Calculate dimension of a matrix:

To determine the *dimensions* of a matrix or vector, use the command `size`.

```
>> size(A)
ans    =
       3     3
```

means 3 rows and 3 columns.

```
>> [m,n]=size(A)
```

Transposing a matrix:

```
A  =
     1     2     3
     4     5     6
     7     8     0
```

```
>> A'
ans =
     1     4     7
     2     5     8
     3     6     0
```

A^T

Concatenating matrices:

```
A    =
     1     2     3
     4     5     6
     7     8     9

>> B = [A 10*A; -A [1 0 0; 0 1 0; 0 0 1]]
B    =
     1     2     3    10    20    30
     4     5     6    40    50    60
     7     8     9    70    80    90
    -1    -2    -3     1     0     0
    -4    -5    -6     0     1     0
    -7    -8    -9     0     0     1
```

Matrix generators:

`eye(m,n)`	Returns an m-by-n matrix with 1 on the main diagonal
`eye(n)`	Returns an n-by-n square identity matrix
`zeros(m,n)`	Returns an m-by-n matrix of zeros
`ones(m,n)`	Returns an m-by-n matrix of ones
`diag(A)`	Extracts the diagonal of matrix A
`rand(m,n)`	Returns an m-by-n matrix of random numbers

```
>> b=ones(3,1)
b    =
     1
     1
     1
```

```
>> eye(3)
ans    =
     1     0     0
     0     1     0
     0     0     1
```

Matrix operations:

`A+B` or `B+A`	is valid if `A` and `B` are of the same size
`A*B`	is valid if `A`'s number of column equals `B`'s number of rows
`A^2`	is valid if `A` is square and equals `A*A`
α`*A` or `A*`α	multiplies each element of `A` by α

Array operations:

`.*`	Element-by-element multiplication
`./`	Element-by-element division
`.^`	Element-by-element exponentiation

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix}$$

```
>> C = A.*B
C    =
     10     40     90
    160    250    360
    490    640    810

>> A.^2
ans    =
      1      4      9
     16     25     36
     49     64     81
```

Matrix Inverse:

Let's consider the same matrix A.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$$

Calculating the inverse of A manually is probably not a pleasant work. Here the hand-calculation of A^{-1} gives as a final result:

$$A^{-1} = \frac{1}{9} \begin{bmatrix} -16 & 8 & -1 \\ 14 & -7 & 2 \\ -1 & 2 & -1 \end{bmatrix}$$

```
>> A = [1 2 3; 4 5 6; 7 8 0];
>> inv(A)
ans =
   -1.7778    0.8889   -0.1111
    1.5556   -0.7778    0.2222
   -0.1111    0.2222   -0.1111
```

Determinant of a matrix:

```
>> A = [1 2 3; 4 5 6; 7 8 0];
```

```
>> det(A)
ans   =
      27
```

Solving linear equations easily:

$$\begin{cases} x + 2y + 3z &= 1 \\ 4x + 5y + 6z &= 1 \\ 7x + 8y &= 1 \end{cases}$$

$$Ax = b$$

The result is $x = A^{-1}b$.

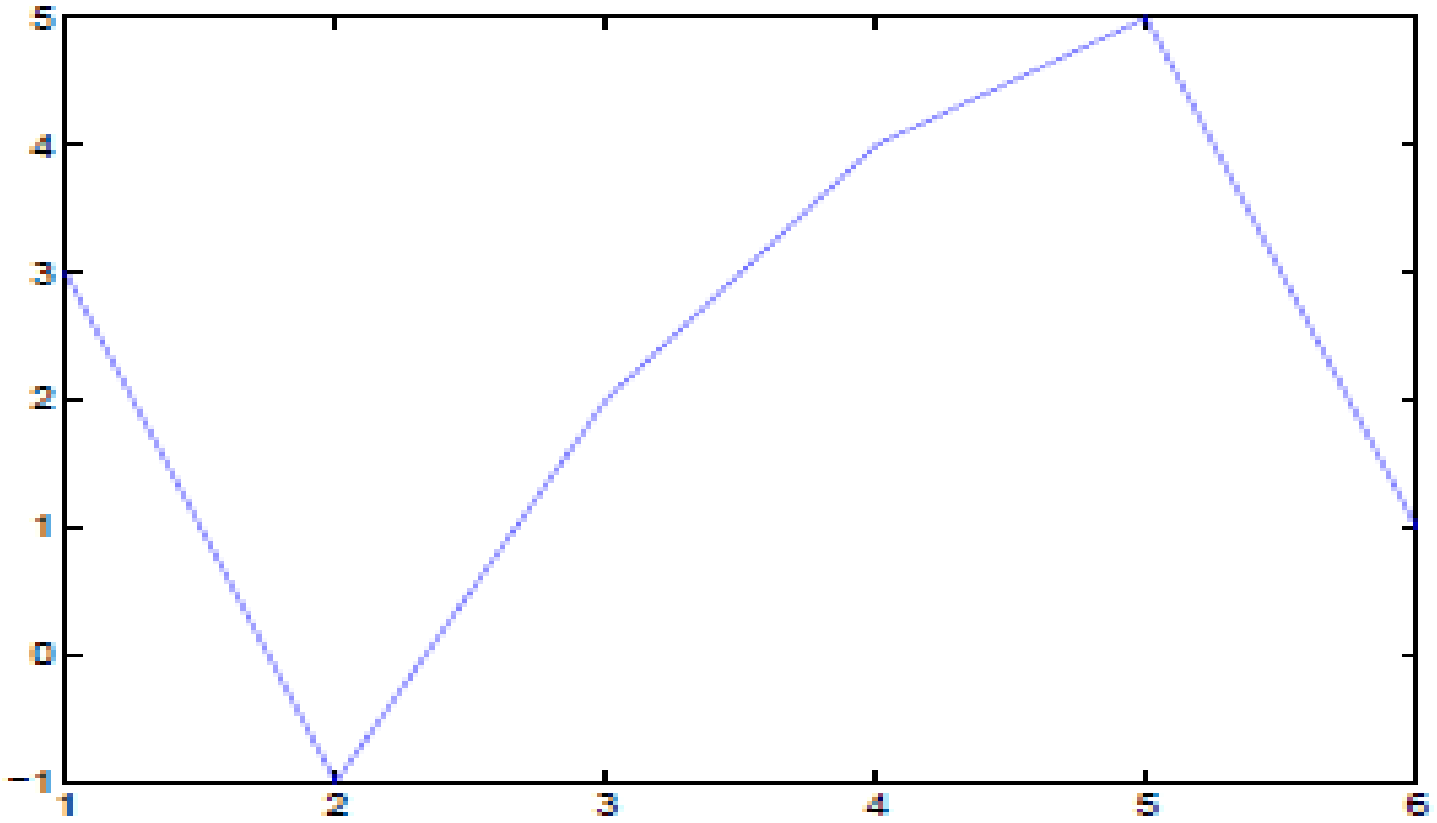
Result Must Be

```
x =
   -1.0000
    1.0000
   -0.0000
```

USING MATLAB, Write a code to solve the linear equation?

Drawing function in MATLAB:

```
>> x = [1 2 3 4 5 6];
>> y = [3 -1 2 4 5 1];
>> plot(x,y)
```



Adding titles, axis labels and annotations:

```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> plot(x,y)
>> xlabel('x = 0:2\pi')
>> ylabel('Sine of x')
>> title('Plot of the Sine function')
```

The color of a single curve is, by default, blue, but other colors are possible. The desired color is indicated by a third argument. For example, red is selected by `plot(x,y,'r')`. Note the single quotes, ' ', around `r`.

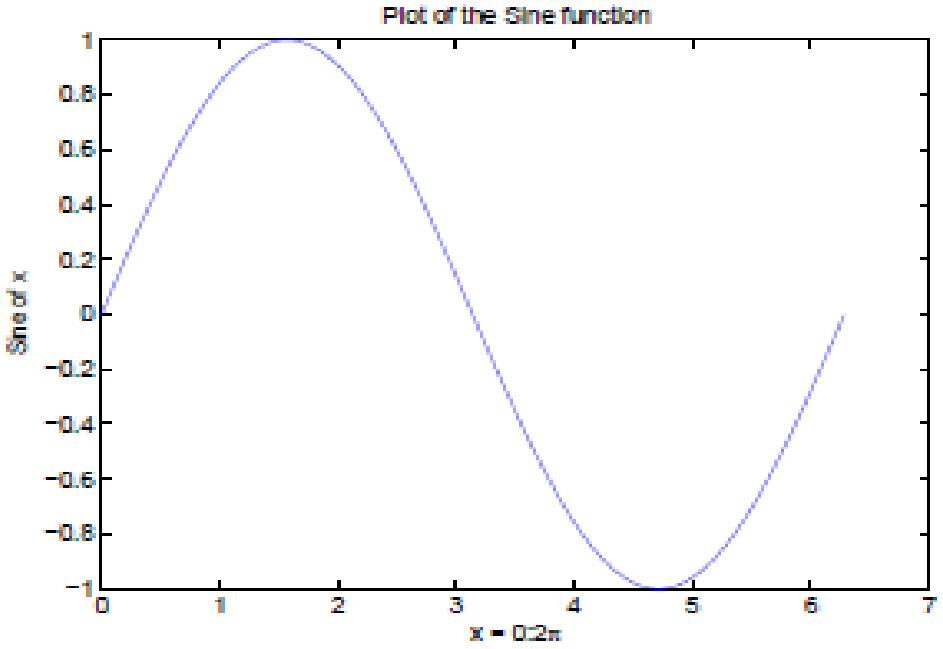
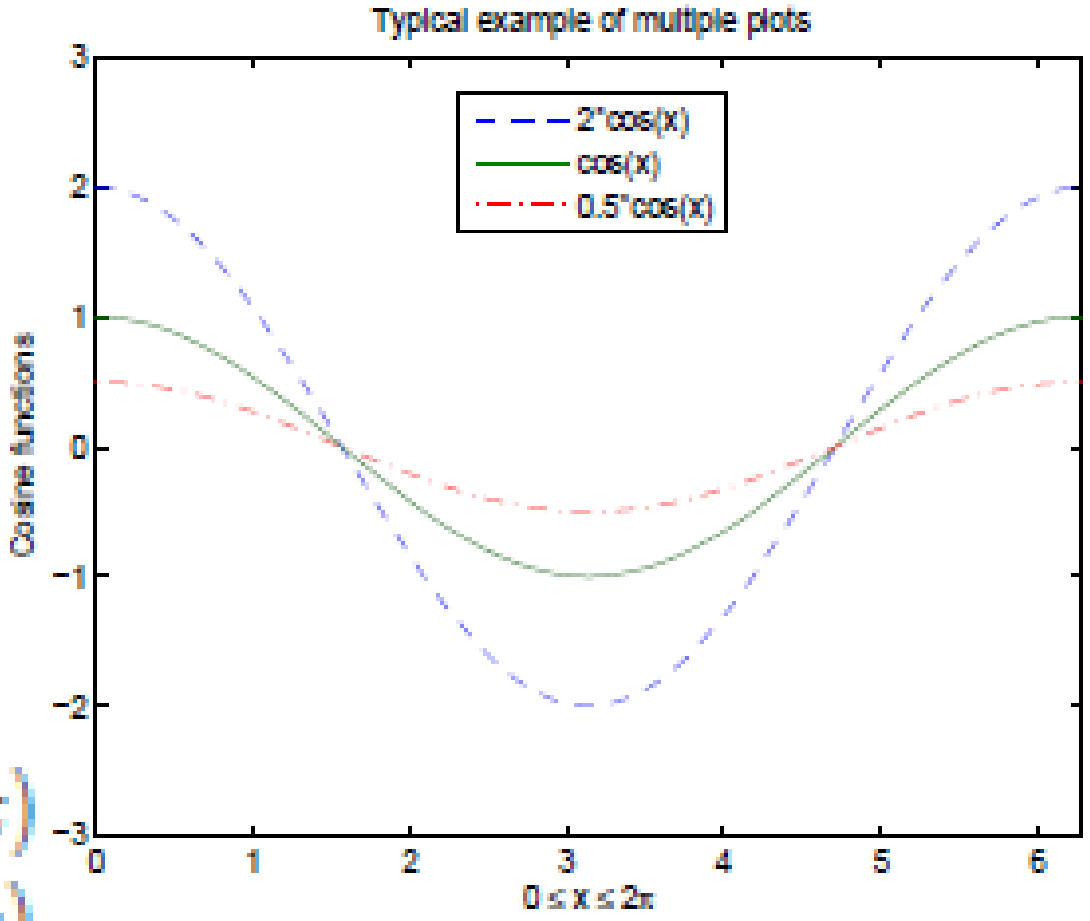


Table 2.3: Attributes for `plot`

Symbol	Color	Symbol	Line Style	Symbol	Marker
k	Black	–	Solid	+	Plus sign
r	Red	– –	Dashed	o	Circle
b	Blue	:	Dotted	*	Asterisk
g	Green	–.	Dash-dot	.	Point
c	Cyan	none	No line	×	Cross
m	Magenta			*s*	Square
y	Yellow			*d*	Diamond

Exercise:

Draw the three functions below:



$$y_1 = 2\cos(x)$$
$$y_2 = \cos(x)$$
$$y_3 = 0.5 * \cos(x), \text{ in the interval } 0 \leq x \leq 2\pi.$$

Where, the first function's curve is dashed, and the second function's curve is solid and the third one is dotted ? -- *Hint:* `0 \leq x \leq 2\pi` ***will give you*** $0 \leq x \leq 2\pi$