SYSTEMS AND CONTROL THEORY

A GENERAL INTRODUCTION TO MATLAB

Luigi Biagiotti

E-mail: luigi.biagiotti@unimore.it
http://www.dii.unimore.it/~lbiagiotti

Outline

- The objectives of this lecture are
  - To become familiar with the MATLAB environment
  - To enable you to use some simple MATLAB commands from the *Command Window*

- **Bibliography**:
  - Brian Hahn and Daniel T. Valentine, *Essential MATLAB for Engineers and Scientists*, Academic Press.

Matlab

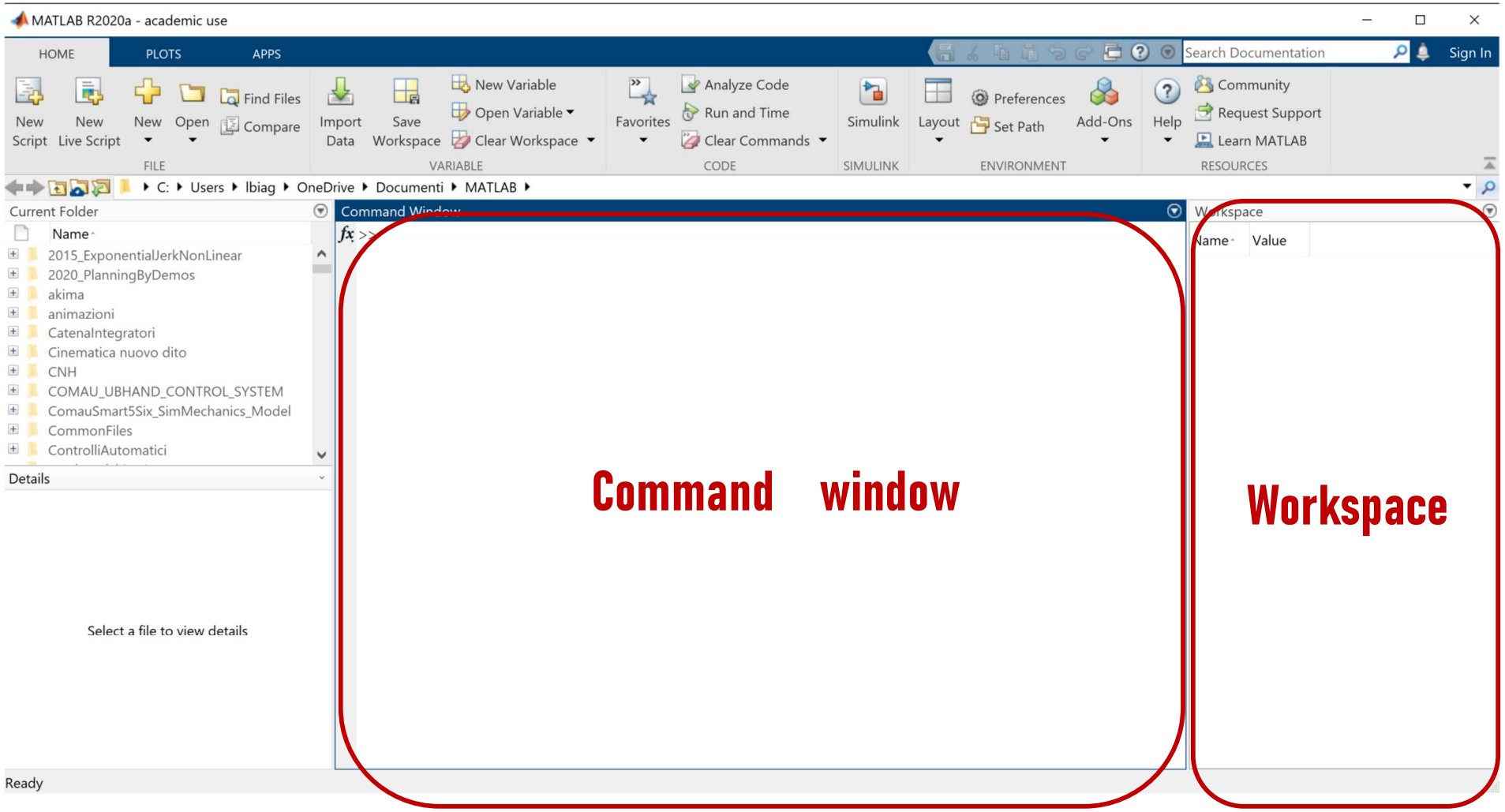
- MATLAB is a powerful computing system for handling scientific and engineering calculations.

- The name MATLAB stands for **Matrix Laboratory**, because the system was designed to make matrix computations particularly easy.

- Matlab is based on a kernel of **general purpose functions** enhanced with additional tools, the so-called **Toolboxes**, that help users to solve specific problems, e.g. the Control System Toolbox. A toolbox is a simple collection of matlab functions.

A toolbox is a simple collection of matlab functions.

Matlab Desktop



Command Window

- **MATLAB commands must be inserted by means of the Command Window.**

- Some commands of general use are
  - `pwd` provides the current directory
  - `dir` lists the files of current directory
  - `clc` clears the command window

- Variables defined in the MATLAB environment are collected in the **WorkSpace**. `who` lists the variables in the current workspace. Command `clear` removes all variables from the workspace.

- Command `help` provides the list of all the toolboxes which are installed in the system. By typing in

  `>> help <toolbox name>`

  one obtains the list of the functions composing the toolbox (e.g. **help control**). The command

  `>> help <command name>`

  provides a description of this command

Variables

- MATLAB variables are created with an assignment statement

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

where ***expression*** is a combination of numerical values, mathematical operators, variables, and function calls

- For example

```
>> x=12;<ENTER>
```

By omitting the semicolon (`;`) the name and the value of the variable are printed in the screen. Conversely, the *echo* of the command is not provided

- Once a variable has been created, it can be reassigned

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

The intermediate result is not shown

Error messages

- If we enter an expression incorrectly, MATLAB will return an error message.

- For example

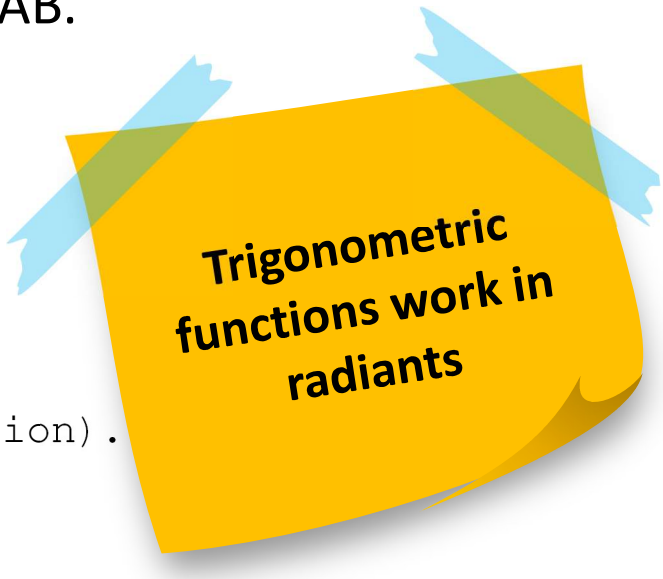
```
>> x = 10;
>> 5x
??? 5x
      |
Error: Unexpected MATLAB expression.
```

Basic mathematical functions

```
Trigonometric.
   sin           - Sine
   sinh          - Hyperbolic sine.
   asin          - Inverse sine.
   cos           - Cosine.
   cosh          - Hyperbolic cosine.
   acos          - Inverse cosine.
   tan           - Tangent.
   tanh          - Hyperbolic tangent.
   atan          - Inverse tangent.
   atan2         - Four quadrant inverse tangent.
Exponential.
   exp           - Exponential.
   log           - Natural logarithm.
   log10         - Common (base 10) logarithm.
   sqrt          - Square root.
Complex.
   abs           - Absolute value.
   angle         - Phase angle.
Rounding and remainder.
   floor         - Round towards minus infinity.
   ceil          - Round towards plus infinity.
   round         - Round towards nearest integer.
   mod           - Modulus (signed remainder after division).
   rem           - Remainder after division.
   sign          - Signum.
```

MATLAB offers many predefined mathematical functions for technical computing.

Typing **`help elfun`** and **`help specfun`** calls up full lists of elementary and special functions that are built into MATLAB.



Predefined constant values

- In addition to the elementary functions, MATLAB includes a number of predefined constant values. The most common values are:
  - `pi` ← The π number, π = 3.14159…
  - `i,j` ← The immaginary unit i
  - `Inf` ← The infinity, ∞
  - `NaN` ← Not a number

Matrices and vectors

- To type a matrix into MATLAB it is necessary to
  - 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, `]` .

- Example

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

A =

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

Matrices and vectors

- A vector is a special case of a matrix

```
>> c=[4;5;6]

c =

     4
     5
     6
```

column vector

```
>> r=[4,5,6]

r =

     4     5     6
```

row vector

- A row vector can be converted to a column vector using the transpose operator, and vice-versa. The transpose operation is denoted by an apostrophe or a single quote (').

- The easiest way of defining a vector where the elements (components) increase by the same amount is

```
>> t=[0:0.1:10]
```

row vector with elements from 0 to 10 with step 0.1

.

Matrices and vectors indexing

- Once we have entered a matrix, we can refer to it simply as matrix **A**. We can then view a particular element in a matrix by specifying its location

```
>> Element = A(2,1)

Element =

     4
```

column index

row index

The indices start from 1

- Correcting any entry is easy through indexing

```
>> A(3,3) = 0

A =

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

Here we substitute **A(3,3)=9** by **A(3,3)=0**

The colon operator (:)

- The colon operator can be used to pick out a certain row or column. For example, the statement `A(m:n,k:l)` specifies rows `m` to `n` and columns `k` to `l`.
- Example 1

```
>> A(1,:)
ans =
     1   2   3
```

First row of A

- Example 2

```
>> A(:,1)
ans =
     1
     4
     7
```

First column of A

- Example 3

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

Sub-matrix of A

Matrix generators

- MATLAB provides functions that generate elementary matrices:

```
A=eye(n);
```
← n-by-n identity matrix

```
A=eye(m,n);
```
← m-by-n matrix with 1 on the main diagonal

```
A=zeros(m,n);
```
← m-by-n matrix of zeros

```
A=ones(m,n);
```
← m-by-n matrix of ones

```
A=rand(m,n);
```
← m-by-n matrix of random numbers

```
A=diag(V);
```
← n-by-n matrix with the element of vector **v** on the main diagonal

- For a complete list of *elementary matrices* and *matrix manipulations*, type `help elmat` or `doc elmat`

Concatenating matrices

- Matrices can be made up of sub-matrices
- **Problem**: make up the 6-by-6 matrix

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

where

$$A_{11} = [v_1, v_2, v_3]$$
$$A_{12} = 0_3$$
$$A_{21} = I_3$$
$$A_{22} = [v_3, v_2, v_1]$$

and v_1, v_2 e v_3 are column vetors defined by the user.

- **Solution**:

```
>> v1 = [1 2 3]';
>> v2 = rand(3,1);
>> v3 = [3; 2; 1];
>> A11=[v1 v2 v3];
>> A22=[v3 v2 v1];
>> A = [A11, eye(3); zeros(3), A22]
```

Array operations

- MATLAB allows the following arithmetic operations on matrices:
  - addition **+**

```
>> A+B
```

  - subtraction **–**

```
>> A-B
```

valid if **A** and **B** are of the same size

  - multiplication *****

```
>> A*B
```

valid if **A**'s number of column equals **B**'s number of rows

  - (right and left) divisions **/** ****

```
>> A/B
```

Equivalent to **A*inv(B)**

```
>> A\B
```

Equivalent to inv(**A**)***B**

  - Exponentiation **^**

```
>> A^2
```

valid if **A** is square and equals **A*A**

Operations element-by-element

- Arithmetic operations can be done *element-by-element*. The period character (**.**) distinguishes these operations from standard matrix operations.
  - multiplication **.***

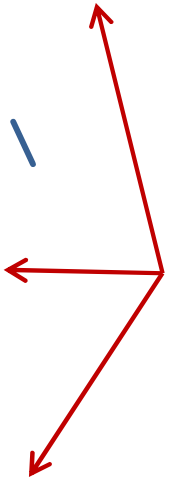
```
>> A.*B
```

  - (right and left) divisions **./** **.**

```
>> A./B   >> A.\B
```

  - Exponentiation **.^**

```
>> A.^B
```



A and **B** must have the same size

- Example:

```
>> v = [1 2 3].*[1 2 3]
v =
   1     4     9
```

Matrix functions

MATLAB provides many matrix functions for various matrix/vector manipulations

- Dimensions

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

number of column

number of row

for vectors, see command `length`

- Transpose

`>>B=A'` (alternatively `>>B=transpose(A)`)

- Determinant

```
>> d=det(A)
```

← **A** must be square

Matrix functions

- Inverse

```
>>I=inv(A)
```

← **A** must be square

For rectangular matrices, see command `pinv`

- Rank, i.e. number of linearly independent rows or columns

```
>> r=rank(A)
```

- Eigenvalues

```
>> e=eig(A)
```

↑

Vector containing the eigenvalues

Solution of a linear system

- Problem: solve the system

$$\begin{cases} x_1 + x_2 + x_3 - x_4 = 1 \\ x_1 + x_2 - x_3 = 2 \\ x_1 - x_2 + x_3 = 0 \\ x_1 + 2x_2 - 3x_3 = 2 \end{cases}$$

- Solution:

```
>> A = [1, 1, 1, -1; 1, 1, -1, 0; 1, -1, 1, 0; 1, 2, -3, 0];
>> b = [1, 2, 0, 2]';
>> x = inv(A)*b;
```

or

```
>> x = A\b
```

Vectors role

- The vectors have in MATLAB two fundamental functions:
  - polynomials representation, a polynomial is represented by the vector of its coefficients
  - signals representation, a signal is represented by the sequence of values that it takes during time, therefore by a vector

Operations on polynomials

- Polynomial "pol" (= 3 s^2+ 2 s+ 1) can be defined with the statement:

```
>> pol= [3 2 1]
 pol =
 3 2 1
```

- **roots**: roots computation (pol=0):

```
>> roots(pol)
ans =
-0.3333 + 0.4714i
-0.3333 - 0.4714i
```

- **polyval**: pointwise evaluation of a polynomial:

```
>> polyval(pol,1)
ans =
6
```

Operations on polynomials

- **Computation of the residues, poles and direct term of the partial fraction expansion of the ratio of two polynomials**:

  es.

$$\frac{2s^3+5s^2+3s+6}{s^3+6s^2+11s+6} = \frac{-6}{s+3} + \frac{-4}{s+2} + \frac{3}{s+1} + 2$$

```
>> num = [2 5 3 6]; den = [1 6 11 6];
>> [r,p,k] = residue(num,den)
r =
   -6.0000
   -4.0000
    3.0000
p =
   -3.0000
   -2.0000
   -1.0000
k =
     2
```

Operations on polynomials

- **Polynomial multiplication** (pol3=(s+1)(s+1)):

```
>> pol1=[1 1]; pol2=[1 1];
>> pol3=conv(pol1,pol2)
pol3 =
1   2   1
```

- **Polynomial division** ($(s^2+2s+2)=q(s)(s+1)+r(s)$):

```
>> pol1=[1 2 2]; pol2=[1 1];
>> [q,r]=deconv(pol1,pol2)
q =
1     1
r =
0     0     1
```

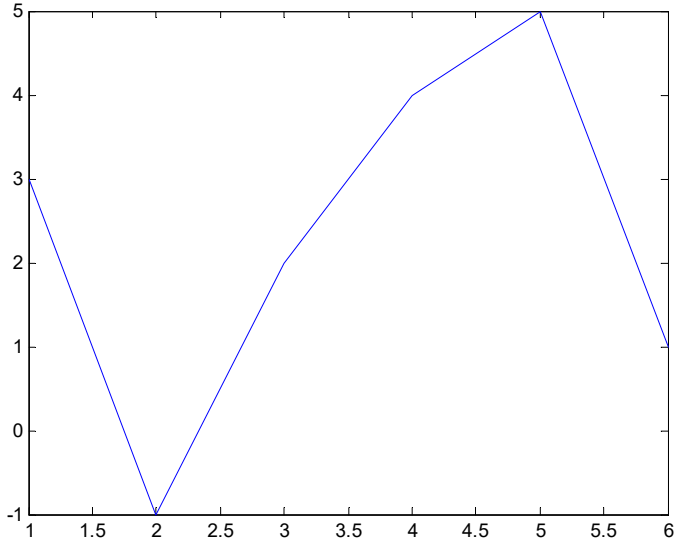
Basic plotting

- The basic MATLAB graphing procedure, for example in 2D, is to take a vector of *x*-coordinates, $x = (x_1, \ldots, x_n)$ and a vector of *y*-coordinates, $y = (y_1, \ldots, y_n)$, locate the points (x_i, y_i), $i = 1, \ldots, n$ and then join them by straight lines.

- This procedure is made by the command **`plot`**.

- Example:

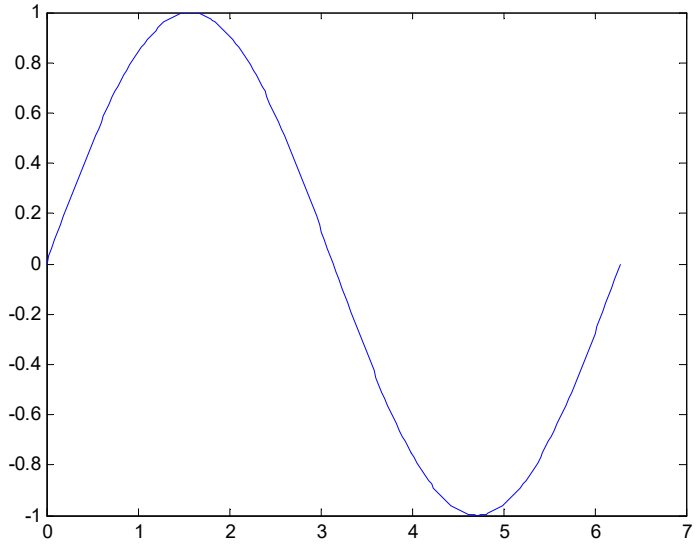
```
>> x = [1 2 3 4 5 6];
>> y = [3 -1 2 4 5 1];
>> plot(x,y)
```



x and **y** must have the same length

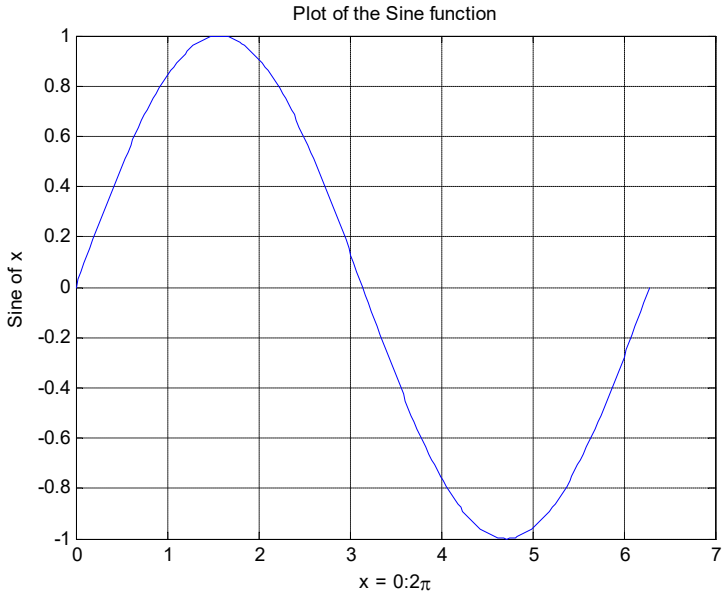
- Problem: plot the function **`sin(x)`** for **`x`** from 0 to 2π

Adding titles, axis labels, and annotations



```
>> x=0:0.1:2*pi;
>> plot(x,sin(x))
```

- MATLAB enables you to add axis labels, titles, and a grid.




```
>> grid on
>> xlabel('x = 0:2\pi')
>> ylabel('Sine of x')
>> title('Plot of the Sine function')
```

Specifying line styles and colors

- It is possible to specify line styles, colors, and markers (e.g., circles, plus signs, . . .) using the plot command:

```
plot(x,y,'style_color_marker')
```

where `style_color_marker` is a triplet of values from the following table

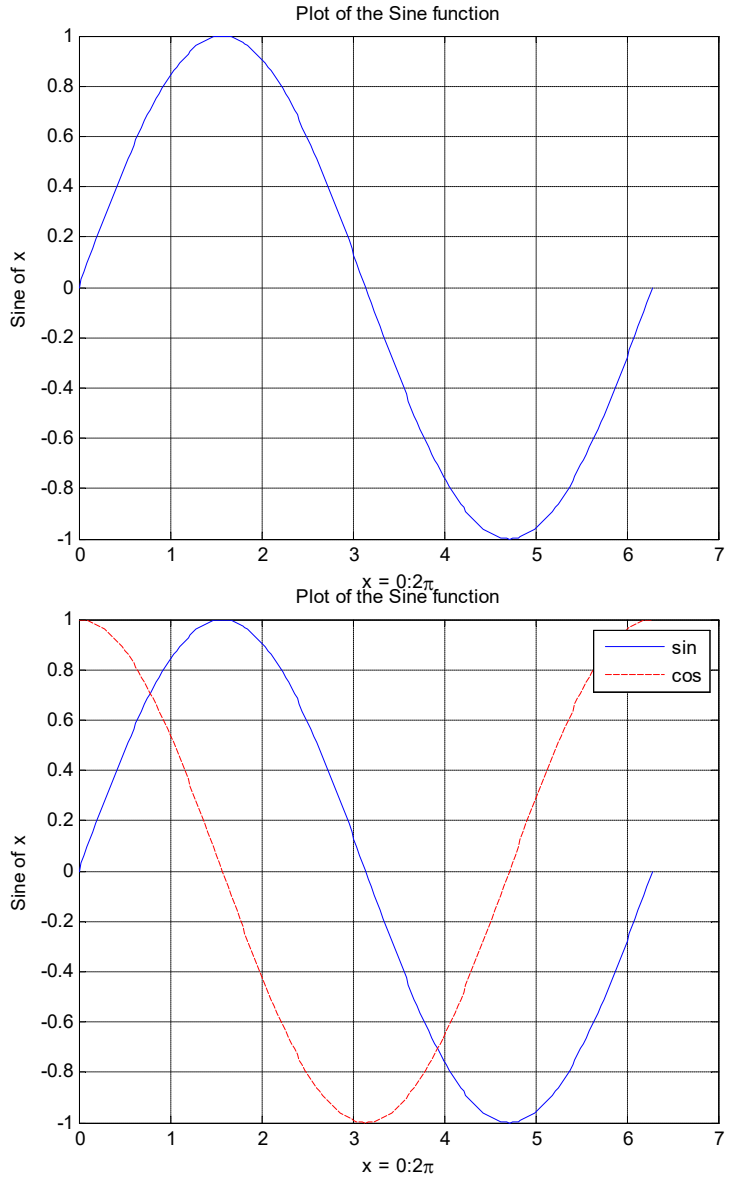
Symbol	Color
k	Black
r	Red
b	Blue
g	Green
c	Cyan
m	Magenta
y	Yellow

Symbol	Line Style
-	Solid
--	Dashed
:	Dotted
-.	Dash-dot

Symbol	Color
+	Plus sign
o	Circle
*****	Asterisk
.	Point
x	Cross
s	Square
d	Diamond

Multiple data sets in one plot

- The command **`hold on`** holds the current plot and all axis properties so that subsequent graphing commands add to the existing graph.

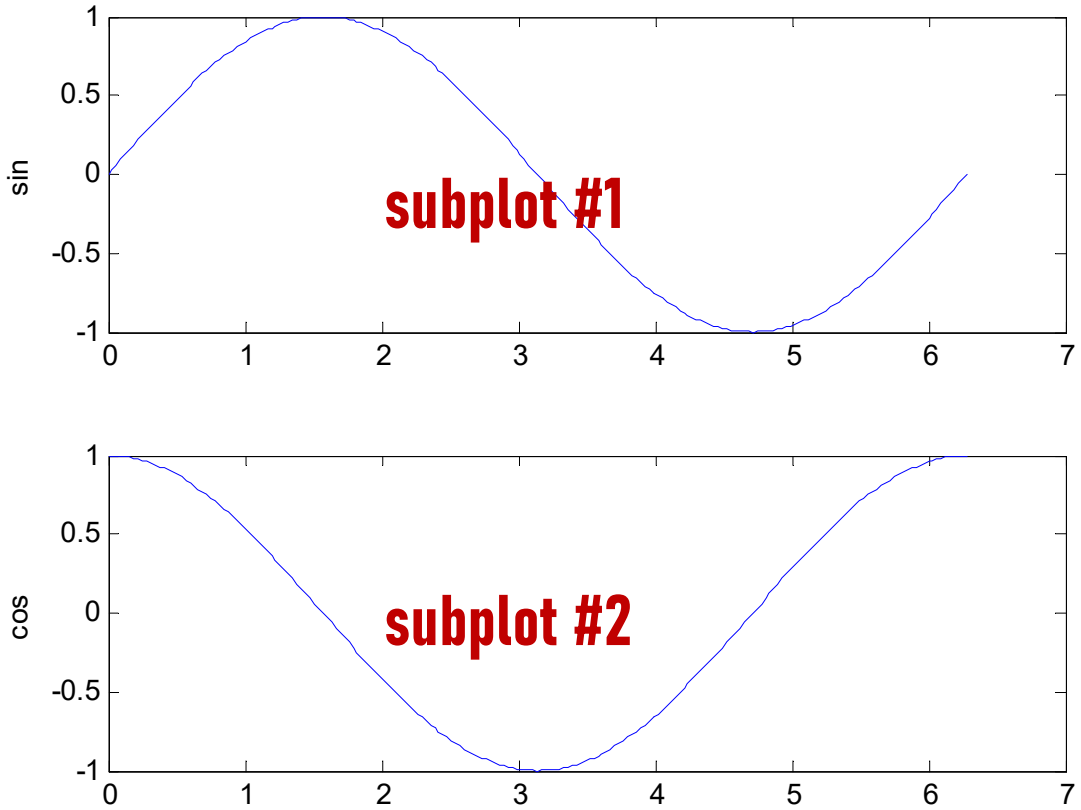



```
>> hold on
>> plot(x, cos(x),'r--')
>> legend('sin', 'cos')
```

Multiple data sets in one plot

- The command `subplot(m,n,p)` breaks the Figure window into an *m*-by-*n* matrix of small axes, and selects the *p*-th axes for the current plot

```
>> subplot(2,1,1)
>> plot(x,sin(x))
>> ylabel('sin')
>> subplot(2,1,2)
>> plot(x,cos(x))
>> ylabel('cos')
```



Miscellaneous on plot

- The command `figure` opens a new figure or can be used to select a figure previously defined (`figure(FigNum)`)
- The command `close(FigNum)` closes a specific figure. The command `close all` closes all the figure defined in the MATLAB session
- The command `print` can be use to produce jpeg or eps images from the current figure

```
>> print -depsc FileName
```
⟵ It produces the file FileName.eps

```
>> print -djpeg FileName
```
⟵ It produces the file FileName.jpg