

MATLAB® & Simulink®

For

**Technical
Engineering**

Students!

**MATLAB®
& SIMULINK®**

Prepared and edited by Assist. Professor / Emad Jihad

 **MathWorks®**

MATLAB® & Simulink®

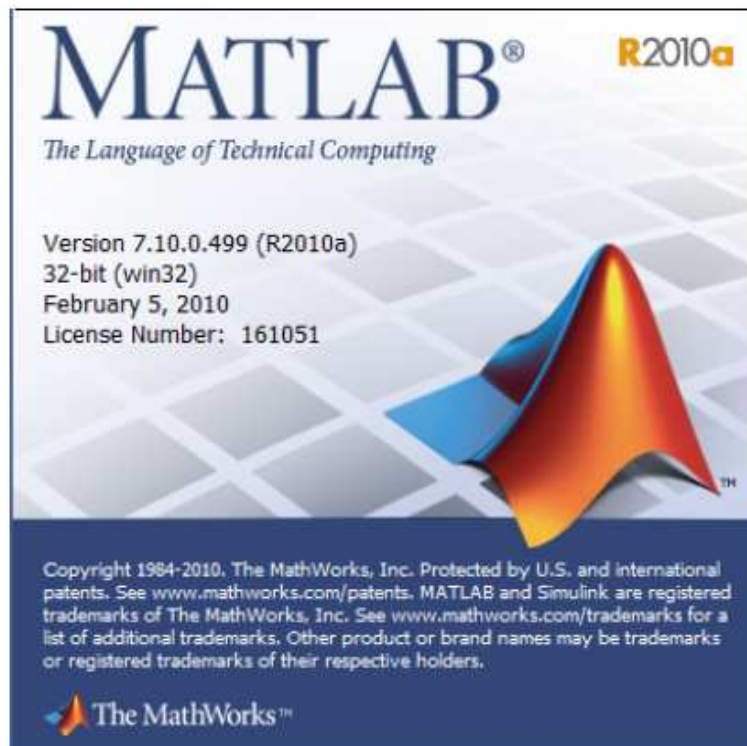
Chapter 1

Introduction to Matlab

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**



Introduction to MATLAB

For

Technical Engineering

Students!

Lectures Prepared and Edited by
Assistant Professor/ Emad Jihad

Introduction to Matlab

The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.

MATLAB is a high-performance language for technical computing.

It integrates *computation*, *visualization*, and *programming* environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated *data structures*, contains built-in editing and *debugging tools*, and supports *object-oriented programming*. These factors make MATLAB an excellent tool for teaching and research.

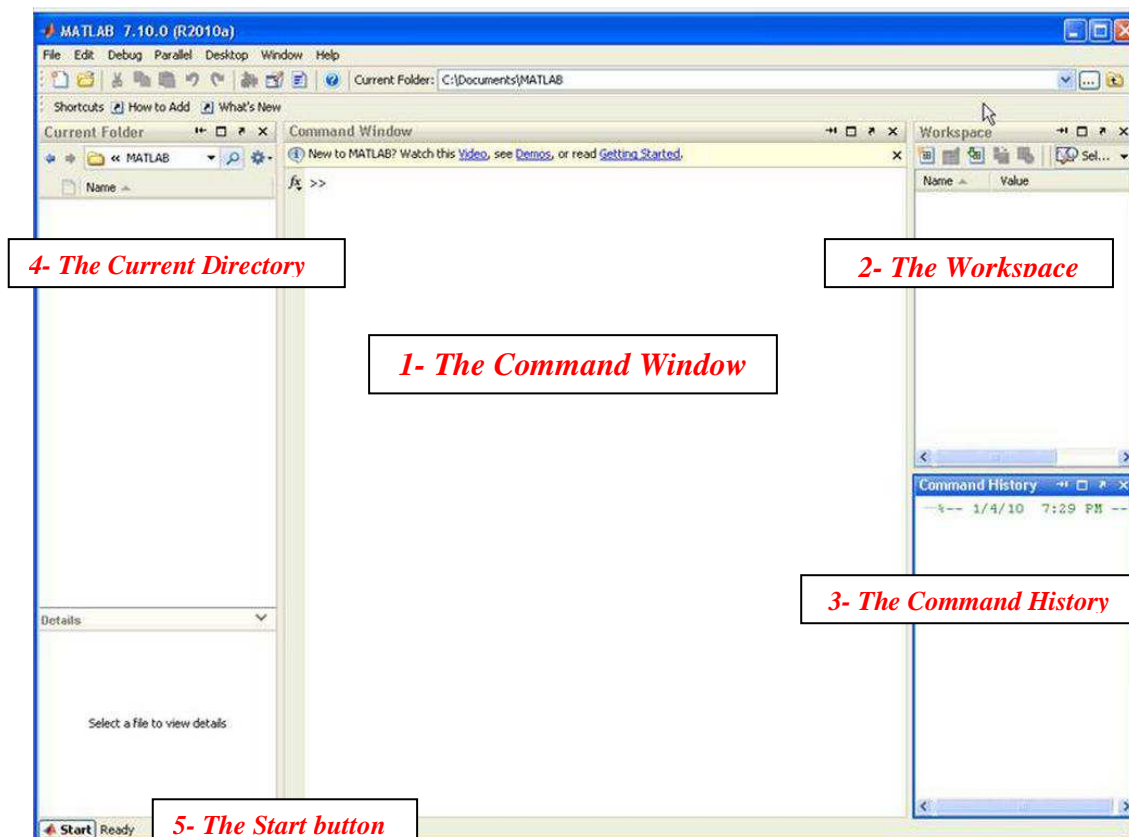
MATLAB has many advantages compared to conventional computer languages (e.g. C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an *array* that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.

It has powerful *built-in* routines that enable a very wide variety of computations.

It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as *toolbox*.

There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

The Matlab window consists of command menu, also some sub-windows as seen below, 1- The command window: all Matlab functions and operations written here, 2- The workspace: hold the current operations, 3- the command history: Keeps all operations (even the errors) done, 4- The current directory (folder) in use, and 5- The Start button: Click for quick access to tools and other options.



Let's start at the very beginning ...

As an example of a simple interactive calculation, just type the expression you want to evaluate. For example, let's suppose you want to calculate the expression, $1 + 2 * 3$. You type it at the prompt command (`>>`) as follows:

Ex 1:

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

← (Enter key)

You will have noticed that if you do not specify an output variable, MATLAB uses a default variable **ans**, short for answer, to store the results of the current calculation. Note that the variable **ans** is created (or overwritten, if it is already existed). To avoid this, you may assign a value to a variable or output argument name. For example:

Ex 2:

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

← (Enter key)

Note: We will not mention Enter key any more for later examples as it was clear stated.

Ex 3:

```
>> 4*x  
ans =  
28.0000
```

Creating MATLAB variables

MATLAB variables are created with an assignment statement. The syntax of variable assignment is: variable name = a value (or an expression)

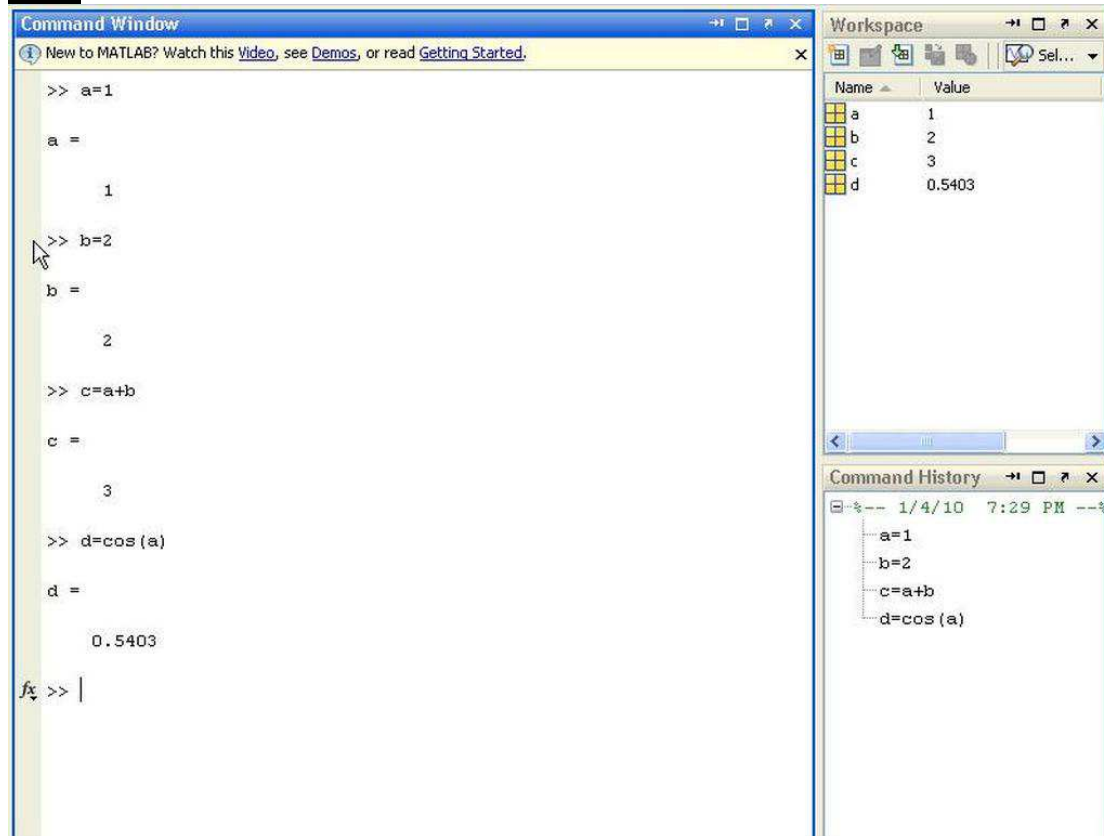
For example,

```
>> x = expression
```

where expression is a combination of numerical values, mathematical operators, variables, and function calls. On other words, expression can involve:

- manual entry
- built-in functions (Ready Matlab functions).
- user-defined functions

Ex 4:



```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

>> a=1
a =
    1

>> b=2
b =
    2

>> c=a+b
c =
    3

>> d=cos(a)
d =
    0.5403

fx >> |

Workspace
Name Value
a 1
b 2
c 3
d 0.5403

Command History
1/4/10 7:29 PM
a=1
b=2
c=a+b
d=cos(a)
```

Once a variable has been created, it can be reassigned. In addition, if you do not wish to see the intermediate results, you can suppress the numerical output by putting a semicolon (;) at the end of the line. Then the sequence of commands looks like this:

Ex 5:

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

Error messages

If we enter an expression incorrectly, MATLAB will return an error message. For example, in the following, we left out the multiplication sign, *, in the following expression:

```
>> x = 10;
>> 5x
??? 5x
|
```

Error: Unexpected MATLAB expression.

Making corrections

To make corrections, we can, of course retype the expressions. But if the expression is lengthy, we make more mistakes by typing a second time. A previously typed command can be recalled with the up-arrow key ↑. When the command is displayed at the command prompt, it can be modified if needed and executed.

Controlling the hierarchy of operations or precedence

Let's consider the previous arithmetic operation, but now we will include *parentheses*. For example, $1 + 2 * 3$ will become $(1 + 2) * 3$

Ex 6:

```
>> (1+2)*3
ans =
9
```

While we have seen that:

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

Ex 7: Find the result of:

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

```
>> 1/(2+3^2)+4/5*6/7
ans =
0.7766
(But, if parentheses are
missing):
```

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

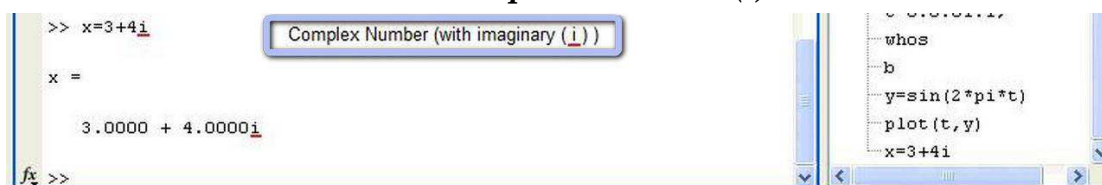
Entering multiple statements per line

It is possible to enter multiple statements per line. Use commas (,) or semicolons (;) to enter more than one statement at once. Commas (,) allow multiple statements per line without suppressing output.

Ex 8:

```
>> a=7; b=cos(a), c=cosh(a)
b =
0.6570
c =
548.3170
```

Complex Number: (i)



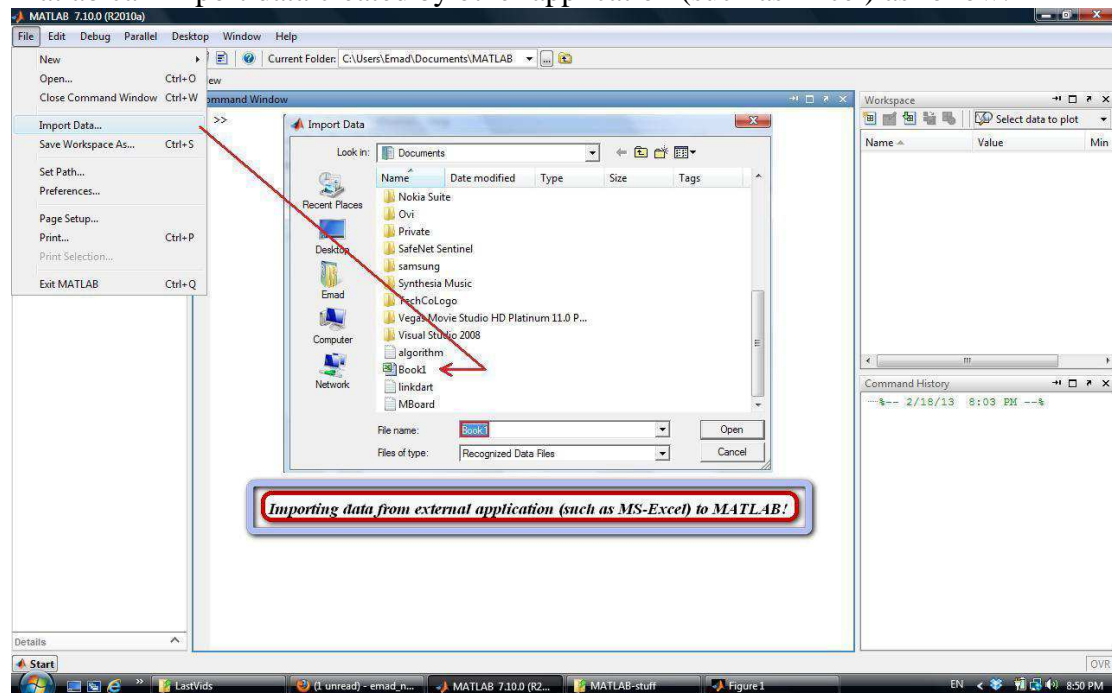
Miscellaneous commands

Here are few additional useful general commands:

- To clear the Command Window, type **clc** (variables not cleared from Workspace)
- To abort a MATLAB computation, type **ctrl+c**
- To Clear all variables from Workspace window, type **Clear** or **Clear All**.
- **Display** command can show any text in command window, (display 'Hello').
- **Who** command gives a List of all variables stored in memory.
- **Whos** give more details which include size, space allocation, and class of the variables.

Importing external data

Matlab can import data created by other application (such as Excel) as follow:



We get:



Lecture prepared by
Assist. Professor/ Emad Jihad

MATLAB® & Simulink®

Chapter 2

Introduction Algebra in Matlab

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**

Algebra-Matlab

```
>>% MATLAB - Part II.
```

```
>>%=====
```

```
>> % Note that % sign is used for only comments and explanations in matlab!
```

```
>> % Algebraic Equations:
```

```
>> %=====
```

```
>> % syms command is used to Declare variables used in the equation, then
```

```
>> % factor command is used to solve an equation:
```

```
>> % Example(1):
```

```
>> syms x y
```

```
>> factor(x^2-y^2)
```

```
ans =
```

```
(x - y)*(x + y)
```

```
>> syms x
```

```
>> factor(x^3-5*x^2+3*x+1)
```

```
ans =
```

```
(x - 1)*(x^2 - 4*x - 1)
```

```
>> % expand command is used to expand the terms of any power
```

```
>> % Example(2):
```

```
>> syms x y
```

```
>> expand((x^2-y^2)^4)
```

```
ans =
```

```
x^8 - 4*x^6*y^2 + 6*x^4*y^4 - 4*x^2*y^6 + y^8
```

```
>>
```

```
>>% simplify command is used to simplify some equations to smallest ones:
```

```
>>% Example(3):
```

```
>> syms x
```

```
>> simplify((x^5-3*x^4)^2/(x-1)^2)
```

```
ans =
```

Algebra-Matlab

$$(x^8(x-3)^2)/(x-1)^2$$

% Example(4):

```
>> syms x
```

```
>> simplify((x^3-5*x)/(x^2-4*x))
```

ans =

$$(x^2 - 5)/(x - 4)$$

```
>>% Solving Algebraic equations:
```

```
>>%=====
```

```
>>% To solve Algebraic equations (solve) command is to be used:
```

```
>>% Example(5): solve the equation: a*x^2+b*x+c for the value of x:
```

```
>> solve('a*x^2+b*x+c','x')
```

ans =

$$-(b + (b^2 - 4*a*c)^{1/2})/(2*a)$$

$$-(b - (b^2 - 4*a*c)^{1/2})/(2*a)$$

```
>>% notice that we used the quote ( ' ) to define the equation and the variable (x)
```

```
>>% But instead, we can still use (syms) to declare the variable without the Quotes in solve:
```

```
>> syms a b c x
```

```
>> solve(a*x^2+b*x+c,x)
```

ans =

$$-(b + (b^2 - 4*a*c)^{1/2})/(2*a)$$

$$-(b - (b^2 - 4*a*c)^{1/2})/(2*a)$$

```
>> % We can solve more than one equations with more than one variable.
```

```
>>% In this case, Vectors will have the solution.
```

```
>>% Example(6):
```

```
>> [x,y]=solve('x^2-2*x+y^2=3','2*x+y=1')
```

Algebra-Matlab

x =

$$\frac{19^{1/2}}{5} + \frac{3}{5}$$
$$\frac{3}{5} - \frac{19^{1/2}}{5}$$

y =

$$-\frac{2 \cdot 19^{1/2}}{5} - \frac{1}{5}$$
$$\frac{2 \cdot 19^{1/2}}{5} - \frac{1}{5}$$

>>% Find the solution in vectors for the following equations simultaneously:

>>% Example(7):

>> [x1,x2]=solve('x1^2+3*x1-x2^2=3','3*x1+x2=5')

x1 =

$$\frac{193^{1/2}}{16} + \frac{33}{16}$$
$$\frac{33}{16} - \frac{193^{1/2}}{16}$$

x2 =

$$-\frac{3 \cdot 193^{1/2}}{16} - \frac{19}{16}$$
$$\frac{3 \cdot 193^{1/2}}{16} - \frac{19}{16}$$

>>% In those previous cases, we got two points for each variable in 2nd. degree equations.

>>% But for 1st. degree equations we will get only one result for each variable!

>>% Example(8): Find the value of x and y in: 5x+10y=33 and 18x-4y=20,

>> [x,y]=solve('5*x+10*y=33','18*x-4*y=20')

x =

$$\frac{83}{50}$$

y =

Algebra-Matlab

247/100

>>% The all above answers can be put in their normal form by simply Re-enter them:

>>% For example,

>> 1/5 - (2*19^(1/2))/5

ans =

-1.5436

>>% or,

>> 83/50

ans =

1.6600

>>% Example(9):

>>% Find the intersection points of two circles equations:

[x,y]=solve('x^2+y^2=4','(x-1)^2+(y-1)^2=3')

x =

$23^{1/2}/4 + 3/4$

$3/4 - 23^{1/2}/4$

y =

$3/4 - 23^{1/2}/4$

$23^{1/2}/4 + 3/4$

>>% Which in turn will be:

>> $23^{1/2}/4 + 3/4$

ans =

Algebra-Matlab

1.9490

```
>> 3/4 - 23^(1/2)/4
```

ans =

-0.4490

```
>> 3/4 - 23^(1/2)/4
```

ans =

-0.4490

```
>> 23^(1/2)/4 + 3/4
```

ans =

1.9490

```
>>% Finally, we may solve for more than two equations and variables, as in this last example.
```

```
>>% Example(10):
```

```
>> [w,x,y,z]=solve('3*w-x-y+z=1','w+5*x-y+z','2*w-3*x-y+2*z','x+z=3')
```

w =

9/4

x =

7/12

y =

91/12

Algebra-Matlab

z =

29/12

>>% Or, in normal form answer, we get:

>> w =9/4

w =

2.2500

>> x =7/12

x =

0.5833

>> y =91/12

y =

7.5833

>> z =29/12

z =

2.4167

>>% =====

>>% = Examples by: Assistant Professor/ Emad Jihad =

>>% = MATLAB - 2013-Baghdad =

>>% =====

MATLAB® & Simulink®

Chapter 3

Differential Equations in Matlab

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**

Differential Equations in Matlab

- First Order Diff. Eq.

In Mathematics, 1st. order diff. eq. is written as dy/dx or y' ... Matlab can solve defined or undefined equations symbolically; Command used to solve such equations is **dsolve**, while **Dy** is used in order to represent the Derivative function.

Example (1): Find the solution for x in the following first order eq.: $xy' - y = 1$?

Matlab solution:

```
>> dsolve('x*Dy-y=1','x')  
  
ans =  
  
C2*x - 1
```

Now, if we have an initial condition for y such as $y(3)=8$, then we get:

```
>> dsolve('x*Dy-y=1','y(3)=8','x')  
  
ans =  
  
3*x - 1
```

Example (2): Find the solution for x in: $y' + y = \sin(x) / 2$.

```
>> dsolve('Dy+y=sin(x)/2','x')  
  
ans =  
  
sin(x)/4 - cos(x)/4 + C6/exp(x)
```

Example (3): Find the solution for: $y' + y = \cos(t)$.

```
>> dsolve('Dy+y=cos(t)')  
  
ans =  
  
cos(t)/2 + sin(t)/2 + C8/exp(t)
```

Notice that Matlab gives different names for constant (C).

Example (4): Find the solution for: $y' = x + y$, where $y(0) = c$.

```
>> dsolve('Dy=x+y','y(0)=c','x')  
  
ans =  
  
exp(x)*(c + 1) - x - 1
```

- Second Order Diff. Eq.

In Mathematics, second order derivative may be written as y'' or d^2y/dx^2
In Matlab, it's written as **D2y** (Thus the Higher is the same, for example D12y)!

Example (5): Solve the following second order equation: $3y'' - 4y' + y = \cos(x)$.

```
>> dsolve('3*D2y-4*y+y=cos(x)','x')  
  
ans =  
  
C14*exp(x) - cos(x)/6 + C13/exp(x)
```

Similar to 1st. order differential equations, we may have initial conditions for y and y'' too.

Example (5): Find the solution for the previous example (5), with initial condition values are $y(0)=3$ and $Dy(0)= -3$.

```
>> dsolve('3*D2y-4*y+y=cos(x)','y(0)=3','Dy(0)=-3','x')  
  
ans =  
  
37/(12*exp(x)) - cos(x)/6 + exp(x)/12
```

Example (6): Solve the equation: $d^2y/dx^2 - dy/dx + 3y = x^3$.

```
>> dsolve('D2y-5*Dy+3*y=x^3','x')
ans =

>> dsolve('D2y-5*Dy+3*y=x^3','x')

ans =

(44*x)/9 + (5*x^2)/3 + x^3/3 + C19*exp(x*(13^(1/2)/2 + 5/2)) +
C20/exp(x*(13^(1/2)/2 - 5/2)) + 190/27
```

Example (7): Solve the equation: $d^2y/dx^2 + 3y = 2\sin(t)$,
Where $y(\pi/2) = 2\pi$, $dy/dt = -5$ at $t = \pi/2$.

```
>> dsolve('D2y+3*y=2*sin(t)', 'y(pi/2)=2*pi','Dy(pi/2)=-5')

ans =

sin(3^(1/2)*t)*((2*3^(1/2)*cos(t*(3^(1/2) - 1)))/(3*(2*3^(1/2) - 2)) - (2*3^(1/2)*cos(t*(3^(1/2) + 1)))/(3*(2*3^(1/2) + 2))) - cos(3^(1/2)*t)*((2*3^(1/2)*sin(t*(3^(1/2) - 1)))/(3*(2*3^(1/2) - 2)) - (2*3^(1/2)*sin(t*(3^(1/2) + 1)))/(3*(2*3^(1/2) + 2))) + (cos(3^(1/2)*t)*(30*sin((pi*3^(1/2))/2) + 3*cos((pi*3^(1/2))/2)^2*sin((pi*(3^(1/2) - 1))/2) + 3*cos((pi*3^(1/2))/2)^2*sin((pi*(3^(1/2) + 1))/2) + 3*sin((pi*3^(1/2))/2)^2*sin((pi*(3^(1/2) - 1))/2) + 3*sin((pi*3^(1/2))/2)^2*sin((pi*(3^(1/2) + 1))/2) + 3*3^(1/2)*cos((pi*3^(1/2))/2)^2*sin((pi*(3^(1/2) - 1))/2) - 3*3^(1/2)*cos((pi*3^(1/2))/2)^2*sin((pi*(3^(1/2) + 1))/2) + 12*pi*3^(1/2)*cos((pi*3^(1/2))/2) + 3^(1/2)*sin((pi*3^(1/2))/2)^2*sin((pi*(3^(1/2) - 1))/2) - 3^(1/2)*sin((pi*3^(1/2))/2)^2*sin((pi*(3^(1/2) + 1))/2) - 2*3^(1/2)*cos((pi*3^(1/2))/2)*sin((pi*3^(1/2))/2)*cos((pi*(3^(1/2) - 1))/2) + 2*3^(1/2)*cos((pi*3^(1/2))/2)*sin((pi*3^(1/2))/2)*cos((pi*(3^(1/2) + 1))/2))/(6*3^(1/2)*cos((pi*3^(1/2))/2)^2 + 6*3^(1/2)*sin((pi*3^(1/2))/2)^2) - (sin(3^(1/2)*t)*(30*cos((pi*3^(1/2))/2) + 3*cos((pi*3^(1/2))/2)^2*cos((pi*(3^(1/2) - 1))/2) + 3*cos((pi*3^(1/2))/2)^2*cos((pi*(3^(1/2) + 1))/2) + 3*sin((pi*3^(1/2))/2)^2*cos((pi*(3^(1/2) - 1))/2) + 3*sin((pi*3^(1/2))/2)^2*cos((pi*(3^(1/2) + 1))/2) + 3^(1/2)*cos((pi*3^(1/2))/2)^2*cos((pi*(3^(1/2) - 1))/2) - 3^(1/2)*cos((pi*3^(1/2))/2)^2*cos((pi*(3^(1/2) + 1))/2) + 3*3^(1/2)*sin((pi*3^(1/2))/2)^2*cos((pi*(3^(1/2) - 1))/2) - 3*3^(1/2)*sin((pi*3^(1/2))/2)^2*cos((pi*(3^(1/2) + 1))/2) - 12*pi*3^(1/2)*sin((pi*3^(1/2))/2) - 2*3^(1/2)*cos((pi*3^(1/2))/2)*sin((pi*3^(1/2))/2)*sin((pi*(3^(1/2) - 1))/2) + 2*3^(1/2)*cos((pi*3^(1/2))/2)*sin((pi*3^(1/2))/2)*sin((pi*(3^(1/2) + 1))/2))/(6*3^(1/2)*cos((pi*3^(1/2))/2)^2 + 6*3^(1/2)*sin((pi*3^(1/2))/2)^2)
```

Prepared and Edited By:
Assist. Professor / Emad Jihad
Matlab -2013

MATLAB® & Simulink®

Chapter 4

Differentiation and Integration in Matlab

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**

Differentiation and Integration in Matlab.

Before using Matlab to perform differentiation and integration, let's show how to represent and find the values of functions.

Representing Functions

In order to represent a function in Matlab, **inline** function is to be used.

Example (1): represent and find the value of function: $x^2 + 5x - 2$, if $x=3$.

```
>> f=inline('x^2+5*x-2')  
  
f =  
  
    Inline function:  
    f(x) = x^2+5*x-2  
  
>> f(3)  
  
ans =  
  
    22
```

We can also represent and find the value of a more than one variable function.

Example (1): represent and find the value of the following two variables function:

$g(x,y) = x^3 - 4x^2 + 3xy - 7$, with values $x=4$, $y=6$.

```
>> g=inline('x^3-4*x^2+3*x*y-7')  
  
g =  
  
    Inline function:  
    g(x,y) = x^3-4*x^2+3*x*y-7  
  
>> g(4,6)  
  
ans =  
  
    65
```

To define a function as a vector, **vectorize** is used along with the function in order to give more than one point at the same time to evaluate that function. (See the example in the next page)...

Example (2): Represent and evaluate the following function at the points 2, 5, and 8 $f(x) = 3x^2 + 2x - 5$. (Hint: in this case *vectorize* is to be used).

```
>> f=inline(vectorize('3*x^2+2*x-5'))
f =
    Inline function:
    f(x) = 3.*x.^2+2.*x-5
>> f([2 5 8])
ans =
    11    80   203
```

To evaluate the function for (n) points at the same time as a (1Xn) vector.

Limits

To compute limits using Matlab, **limit** function is used to evaluate it, when x goes to some value.

Example (3): Find the limit of the function: $\lim_{x \rightarrow 3} ((x^2 - 5) / (x - 1))$. (x goes to 3).

```
>> syms x
>> limit((x^2-5)/(x-1),x,3)
ans =
    2
```

Differentiation

To perform differentiations in Matlab, **diff** function is used.

Example (4): Find the differentiation of the function: $3x^2 + 1$.

(Here also, either **syms** is used to define variable(s) or ('...' used without **syms**).

<pre>>> syms x >> diff(3*x^2+1) ans = 6*x</pre>	OR...	<pre>>> diff('3*x^2+1') ans = 6*x</pre>
---	-------	---

Example (5): Solve the diff. of: $2\sin(x^2) / \cos(x)$.

```
>> diff('2*sin(x)/cos(x)')

ans =

(2*sin(x)^2)/cos(x)^2 + 2
```

The n^{th} Derivative of (f) is written in the form **diff(f, n)**. So if $n=2$ then the second derivative is required and so on...

Example (6): Find the 1st, 2nd, and 3rd derivative of the function: $2x^3 - 4x^2 + 3x$.

<pre>>> diff('2*x^3-4*x^2+3*x') ans = 6*x^2 - 8*x + 3</pre>First derivative
<pre>>> diff('2*x^3-4*x^2+3*x',2) ans = 12*x - 8</pre>Second derivative
<pre>>> diff('2*x^3-4*x^2+3*x',3) ans = 12</pre>Third derivative.

Partial derivative: Matlab can compute partial derivative of a give a given expression with respect to some variable... in this case this variable has to be specified within the *diff* function.

Example (7): Find the derivative of the equation $f(x,y) = x^3 - 3y^2 + \sin(x) - 2\cos(y)$, in terms of x, then for y.

```
>> syms x y
>> diff(x^3-3*y^2+sin(x)-2*cos(y),x) ...for variable x.

ans =

cos(x) + 3*x^2

>> diff(x^3-3*y^2+sin(x)-2*cos(y),y) ...for variable y.

ans =

2*sin(y) - 6*y
```

We can also find the n^{th} . Derivative as we mentioned before for any equation variable.

Example (8): Find the third derivative ($d^3 f x^3$) of the equation: $d^3 f x^3 = x^3 - 3y^2 + \sin(x) - 2\cos(y)$, in terms of x, then for y.

```
>> syms x y
diff(x^3-3*y^2+sin(x)-2*cos(y),x,3) ←...third derivative for x.

ans =

6 - cos(x)

>> syms x y
diff(x^3-3*y^2+sin(x)-2*cos(y),y,3) ←...third derivative for y.

ans =

(-2)*sin(y)
```


Integration

Matlab can find both definite and indefinite integrals. **Int** command is used to compute the symbolic f for x . i.e. $\text{int}(f) \rightarrow \int f(x)dx$, or $\text{int}(f,a,b) \rightarrow \int_a^b f(x)dx$.

Example (9): Find the definite and indefinite integrations (with limits 2,5) for:

$$\int_{-2}^{-5} (x^3 + 2)dx.$$

```
>> int('x^3+2')
```

... definite integral

```
ans =
```

```
(x*(x^3 + 8))/4
```

```
>> int('x^3+2',2,5)
```

... indefinite integral (with limits a=2, b=5)

```
ans =
```

```
633/4
```

```
>> 633/4
```

...just to convert the result to normal value.

```
ans =
```

```
158.2500
```

Example (10): Find the definite integration of $\int (\cos(x) / \sin(x)) dx$?

```
>> int('cos(x)/sin(x)')
```

```
ans =
```

```
log(tan(x)) - log(tan(x)^2 + 1)/2
```

Finally, remember we used **log10** for common base of 10, and **log** to represent Natural Logarithm (ln) before.

But in case of integration we must use (**ln**) for Natural Logarithm instead of log.

Example (11): Find the definite integration with limit (1,3) for: $\int_{-1}^{-3} (\ln(x) + 2)dx$

```
>> int('ln(x)+2',1,3)
ans =
log(27) + 2      ... Result of integral.
>> log(27)+2
ans =
5.2958           ...Normal numeric result.
```

Example (12): Find the indefinite integral of: $\int (2\ln(x) + 2x/(x-3))dx$?

```
>> int('2*ln(x)+2*x/(x-3)')
ans =
6*log(x - 3) + 2*x*log(x)
```

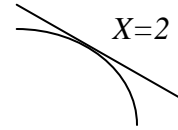
Selected Engineering Exercises:

Exercise 1:

Find the Slope of a tangent line to $x^2 + 3x - 2$, at $x=2$.

Slope means the derivative dy/dx of the curve at $x=2$.

Now, to find the 1st. derivative of the function given:



```
>> diff('x^2+3*x-2')
```

```
ans =
```

```
2*x + 3
```

Then we have to represent the result in a form of a function:

```
>> f=inline('2*x + 3')
```

```
f =
```

```
Inline function:
```

```
f(x) = 2*x + 3
```

Now, substitute the value of $x = 2$ in the function to finally find the slope:

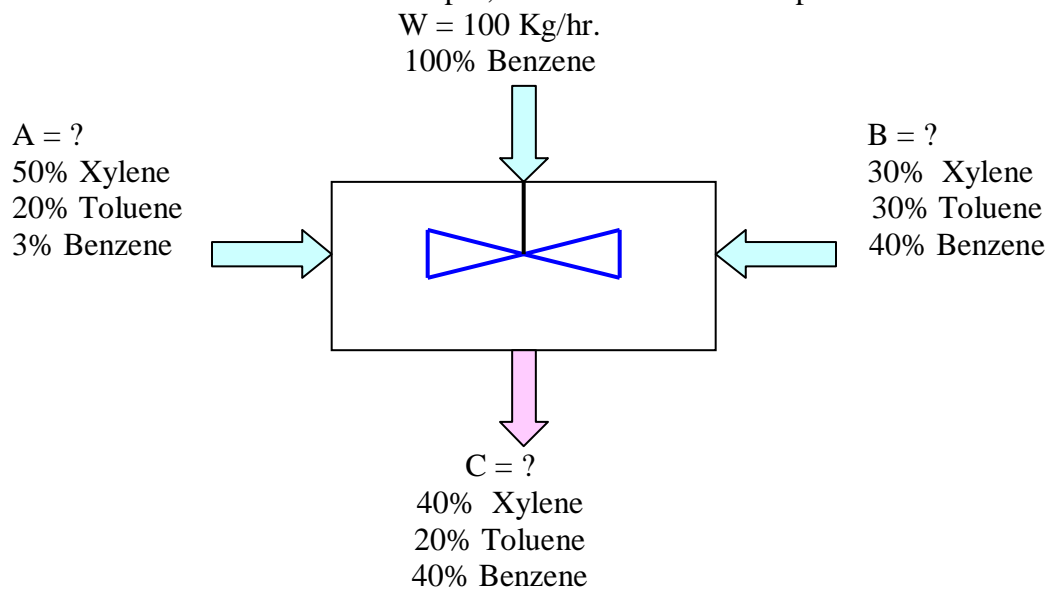
```
>> f(2)
```

```
ans =
```

```
7
```

Exercise 2:

For the mixer shown below, write Matlab code to find the values of streams **A**, **B**, as inputs with stream **W** as another fixed input, and stream **C** as an output.



Solution:

In this exercise we notice that there are Four variables to find their results (although the W input has a constant value). The first value of A will be $(0.5A + 0.3B = 0.4C)$, the second value will be $(0.2A + 0.3B = 0.2C)$, the third one is $(0.3A + 0.4B + W = 0.4C)$, while the stream $W=100$ only.

Since this is Algebraic case, then solve command will be used to put the four equations results as a row vector with four elements A, B, C, and W.

```
>> [A B C W]=solve('0.5*A+0.3*B=0.4*C','0.2*A+0.3*B=0.2*C','0.3*A+0.4*B+W=0.4*C','W=100')
```

A =

600.0

B =

200.0

C =

900.0

W =

100.0

Lecture prepared and edited by: Assist. Professor/ Emad Jihad

MATLAB® & Simulink®

Chapter 5

Vectors and Matrices in Matlab

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**

Vectors and Matrices

Arrays: an array is a mathematical structure that has a collection of numerical **elements**, each of these elements referenced by an **index** which represents the location of that element within the array. This index has integer sequence and any element can be accessed by this sequence.

Arrays can be One Dimension, Two dimensions or even Multi dimension depending on their structure and nature.

A **Vector** is a one dimension array; it can be either **Row vector** which consists of one row and several columns or (nX1) elements size. **Column vector** is also one dimensional array but with one column and several rows or (1Xn) elements size.

A **Matrix** is normally a two dimensional array with (nXm) elements in which n represent the number of rows while m represents the number of columns. In some cases when (n=m) then such a matrix is called a square matrix.

Defining and entering vectors and matrices elements:

There are several methods to enter elements to vectors and matrices in Matlab:-

- 1- Entering an explicit list of elements.
- 2- Loading data from external file (or importing from an application such as Excel).
- 3- Generating elements using function (such as (**rand**) function for random data).
- 4- Building from other arrays elements.

Array elements in rows are represented by semi columns (:), while columns in either spaces or commas (,) in between.

Vectors:

Example (1): Write Matlab code to create a row vector with 5 elements.

```
>> A=[2 5 6 2 7]

A =

     2     5     6     2     7
```

Example (2): Write Matlab code to create a column vector with 6 elements.

```
>> B=[6;3;1;0;1;5]

B =

     6
     3
     1
     0
     1
     5
```

Elements can be also a mathematical or several mathematical operations.

Example (3): Write Matlab code to represent a row vector with 3 elements of $\sin(30)$, $\sin(60)$ and $\sin(90)$.

```
>> X=[sin(30) sin(60) sin(90)]  
  
X =  
  
-0.9880 -0.3048 0.8940 ... (Results will be in radian).
```

Example (3): Build column vector elements from a (3X1) vector and (2X1) vector.

```
>> x=[3;1;5]; ...1st. vector  
>> y=[-6;0]; ...2nd. Vector  
>> z=[x;y] ...resultant vector  
  
z =  
  
3  
1  
5  
-6  
0
```

Transpose of (A) array is represented by (A').

Example (3): From the previous example, Find the transpose of (z).

```
>> z'  
  
ans =  
  
3 1 5 -6 0
```

Example (4): Find the log of each of the 3 elements in a row vector.

```
>> v=[2 5 6];  
>> log(v)  
  
ans =  
  
0.6931 1.6094 1.7918
```

Example (5): Generate a row vector with element values from 1 to 10.

```
>> G=[1:10]

G =

     1     2     3     4     5     6     7     8     9    10
```

Example (6): Generate randomly a column vector with 4 element values between 0 and 10?

```
K=rand(4,1)*10

K =

     4.2176
     9.1574
     7.9221
     9.5949
```

...*rand* function is used to generate randomly numbers between (0 and 1).

Matrices

Normally, Matrices are two dimensional arrays and we will deal with such dimension. However, there are multi dimensional with 3, 4... dimensions.

Example (7): Find the transpose for A(3X4) matrix.

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

A =

     4     6     0
     2     -1     0
     8     0     1
     7     -3     1

>> A'

ans =

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

...Notice that rows separated by semi columns (;).

Operations on Vectors and Matrices:

Both the vectors and Matrices are submitted to some mathematical laws and rules concerning the operations that can be done with... we will not discuss these laws and rules, but will show how Matlab deal with them.

Calling vector or matrix elements:

Since the elements within a vector or a matrix are arranged with indices, then an element can be brought or called by its index.

Example (8): Display the seventh element in a 10 elements vector.

```
>> V=[6 1 2 4 5 0 -4 8 0 1];
>> V(7)

ans =

    -4
```

Example (9): Display from the second to the sixth elements of the previous example.

```
>> V(2:6)

ans =

     1     2     4     5     0
```

Example (10): Display the element B(3,4) in a 5X5 matrix.

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

B =

     2     5     0    -1     4
     6     0     0     1    -4
     3     0     1     0     1
     7     1     0     8     3
     1     0    -4     1     2
>> B(3,4)

ans =

     0
```

Example (11): Display the elements from row (2) Column (4) to row (3) Column (5) in a 2D matrix with elements (5X5)?

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

B =

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

>> B(2:3,4:5)

ans =

     1    -4
     0     1
```

Vectors and matrices Addition and Subtraction:

Example (12): Add 10 to the element 5 in an 8 elements vector.

```
>> A=[2 0 1 2 -4 3 7 5]

A(5)=A(5)+10

A =

     2     0     1     2     6     3     7     5
```

Example (13): Subtract 4 from the elements of previous example.

```
>> A=A-4

A =

    -2    -4    -3    -2     2    -1     3     1
```

Example (14): Add the 6 elements column vector A to column vector B and put result in vector C.

```
>> A=[1;3;5;0;6],B=[5;-1;0;-3;4]

A =

     1
     3
     5
     0
     6

B =

     5
    -1
     0
    -3
     4

>> C=A+B

C =

     6
     2
     5
    -3
    10
```

Remember that rules and laws of matrices and vectors should be applicable.

Example (15): Add two (3X4) matrices and put result into new matrix.

```
>> X=[3 2 1;6 1 7;0 1 0;6 -4 2]

X =

     3     2     1
     6     1     7
     0     1     0
     6    -4     2

>> Y=[5 2 2;-5 2 0;3 -3 2;0 1 0]

Y =

     5     2     2
    -5     2     0
     3    -3     2
     0     1     0

>> Z=Y+X

Z =

     8     4     3
     1     3     7
     3    -2     2
     6    -3     2
```

...The new matrix.

Example (16): Subtract (2.5) from the Y matrix of the previous example.

```
Y-2.5

ans =

    2.5000   -0.5000   -0.5000
   -7.5000   -0.5000   -2.5000
    0.5000   -5.5000   -0.5000
   -2.5000   -1.5000   -2.5000
```

Vectors and Matrices Multiplication:

In example (15), Addition of the matrices was applicable, but not for multiplication. The rule of multiplication needs to have the first matrix elements ($n \times m$) and the second one ($m \times (\text{something})$), thus multiplying X by Y' will be applicable, Otherwise error will occur.

```
>> G=X*Y'  
  
G =  
  
    21   -11    5    2  
    46   -28   29    1  
     2     2   -3    1  
    26   -38   34   -4
```

Same thing happens for vectors, Row vector may be multiplied by column vector (and vice versa) in case they have the same number of elements.

In case that we want to multiply a row vector with another row vector (or elements of column vector with another one), then Matlab has a special operator which is $(.*)$, this is called “**element-wise multiplication operation**” ...Same applicable for Matrices too.

Example (17): Multiply row vector elements with 5 elements by another 5 row vector elements.

```
>> E=[5 1 2 1 4],F=[3 1 0 2 1]  
  
E =  
  
     5     1     2     1     4  
  
F =  
  
     3     1     0     2     1  
  
>> J=E.*F  
  
J =  
  
    15     1     0     2     4
```

...Two row vectors with same no. of elements.

...The special element-wise operation.

Example (18): Multiply Matrix A(2X3) by B(2X3) using The special element-wise operation.

```
>> A=[8 1 2;5 3 4],B=[3 2 0;4 5 1]

A =

     8     1     2
     5     3     4

B =

     3     2     0
     4     5     1

>> C=A.*B

C =

    24     2     0
    20    15     4
```

...The special element-wise operation

Note: To display any matrix size, use command **size**.

Example (19): Display number of rows and columns for the previous example matrices.

```
>> size(A)

ans =

     2     3

>> size(B)

ans =

     2     3

>> size(C)

ans =

     2     3
```

Vectors and Matrices Division:

One should distinguish between two operators in Matlab, the / and the \.

The usual division sign operator like in $X = B / A$ is a solution to $X * A = B$, While if using $X = A \setminus B$ is a solution to $A * X = B$.

Example (20): If A and B are two 5 elements row vectors, Find the solution of $X=B/A$ and $X=A\setminus B$.

```
>> A=[2 2 2 2 2]
A =
     2     2     2     2     2
>> B=(2:2:10)
B =
     2     4     6     8    10
>> X=B/A
X =
     3.0000
>> X=A/B
X =
     0.2727
>> X=A\B
X =
     1     2     3     4     5
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

In case of any two Matrices a and b, the division will be

$a/b = a * \text{inv}(b)$, while $a \setminus b = \text{inv}(a) * b$.

```
>> a=[2 4 1;3 2 4;5 0 1]
a =
     2     4     1
     3     2     4
     5     0     1
>> b=[1 2 0;1 1 0;4 2 1]
b =
     1     2     0
     1     1     0
     4     2     1
>> a/b
ans =
     4.0000    -6.0000     1.0000
     7.0000   -20.0000     4.0000
    -3.0000     4.0000     1.0000
>> a*inv(b)
ans =
     4.0000    -6.0000     1.0000
     7.0000   -20.0000     4.0000
    -3.0000     4.0000     1.0000
```

(See Page 12 for more about **matrix inverse**).

Deleting Elements from a vector or matrix:

Some times we need to delete element or more from a vector, or whole row or column from a matrix, in such cases use the Empty matrix [].

Example (21): Delete element no. 3 from a 5 elements row vector.

```
>> V=[4 1 5 6 3]
V =
     4     1     5     6     3 ...Element no. 3 which is 5 to be deleted.
>> V(3)= [] ...Delete V(3).
V =
     4     1     6     3
```

Example (22): Delete the 2nd. Column from a 3X4 matrix.

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


More Matrices Matlab functions:

Inv function is used to compute Matrix Inverse (Or we may use for example $A^{(-1)}$).

Example (23): Find the Inverse of Matrix (4 X 4).

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

A =

     5     2     1     0
     7     1     0    -2
     6     0    -3     1
     1     0     3     1

>> inv(A)

ans =

   -0.0451    0.0902    0.0827    0.0977
    0.6316   -0.2632   -0.1579   -0.3684
   -0.0376    0.0752   -0.0977    0.2481
    0.1579   -0.3158    0.2105    0.1579
```

Example (24): Prove that the theory of $(A \times A^{-1})$ is true.

```
>> B=[2 1 1; 1 2 2;2 1 2]

B =

     2     1     1
     1     2     2
     2     1     2

>> Binv=inv(B)

Binv =

    0.6667   -0.3333     0
    0.6667    0.6667   -1.0000
   -1.0000     0    1.0000

>> V=B*Binv

V =

     1     0     0
     0     1     0
     0     0     1
```

....Identity matrix generated.

Zeros function puts zeros in a matrix, i.e. all elements are zeros.

```
>> A=zeros(3,4)

A =

    0    0    0    0
    0    0    0    0
    0    0    0    0
```

ones put 1s in a matrix.

```
>> A=ones(3,4)

A =

    1    1    1    1
    1    1    1    1
    1    1    1    1
```

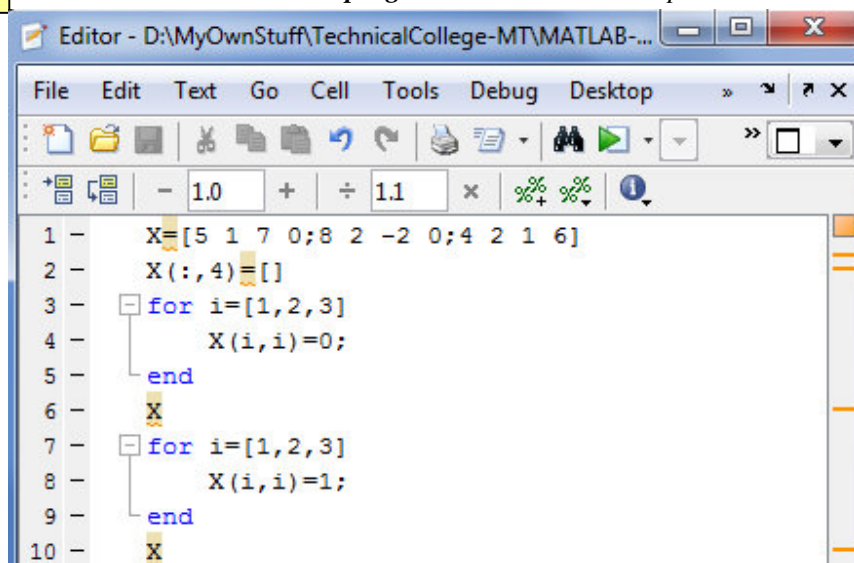
eye gives the identity matrix (Diagonal 1s).

```
>> A=eye(4,4)

A =

    1    0    0    0
    0    1    0    0
    0    0    1    0
    0    0    0    1
```

*But in order to change diagonal element to any fixed value (say 1), then we have to write and run a **program** such as this in script editor.*



Matlab Solved problems of vectors and matrices

Ex(1): Estimate the average boiling point of a Water/Ethanol system at different water compositions (0,0.1,0.2,...1.0) providing that:

Boiling point of water = 100° C

Boiling point of Ethanol= 78.35° C

Mixture boiling point = $X_{\text{water}} \times \text{TBP}_{\text{water}} + X_{\text{ethanol}} \times \text{TBP}_{\text{ethanol}}$.

Solution:

```
>> TBPwater=100;
>> TBPethanol=78.35;
>> Xwater=0:0.1:1

Xwater =

Columns 1 through 8

    0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000

Columns 9 through 11

    0.8000    0.9000    1.0000

>> Xethanol=1-Xwater

Xethanol =

Columns 1 through 8

    1.0000    0.9000    0.8000    0.7000    0.6000    0.5000    0.4000    0.3000

Columns 9 through 11

    0.2000    0.1000     0

>> T=TBPwater*Xwater+TBPethanol*Xethanol

T =

Columns 1 through 8

    78.3500    80.5150    82.6800    84.8450    87.0100    89.1750    91.3400    93.5050

Columns 9 through 11

    95.6700    97.8350    100.0000
```

Ex(2): Calculate the Reynolds Number at two different diameters $D = 0.2$ and 0.5 m, using different velocities as $u=1.2.3\dots,10$ m/s, Providing that:
 $Re = (\rho u d) / \mu$, $\mu = 0.001$, $\rho = 1000$.

Solution

```
>> d1=0.2;d2=0.5;  
>> u=0.1:0.1:1;  
>> m=0.001;  
>> p=1000;  
>> Re1=u.*p*d1/m
```

Re1 =

1.0e+005 *

Columns 1 through 8

0.2000 0.4000 0.6000 0.8000 1.0000 1.2000 1.4000 1.6000

Columns 9 through 10

1.8000 2.0000

```
>> Re2=u.*p*d2/m
```

Re2 =

1.0e+005 *

Columns 1 through 8

0.5000 1.0000 1.5000 2.0000 2.5000 3.0000 3.5000 4.0000

Columns 9 through 10

4.5000 5.0000

Ex(3): Create a vector of even numbers between 31 to 75.

Solution:

```
>> X=(32:2:75)

X =

Columns 1 through 14

    32    34    36    38    40    42    44    46    48    50    52    54    56    58

Columns 15 through 22

    60    62    64    66    68    70    72    74
```

Ex(4): Let $x=[2\ 5\ 1\ 6]$.

- a- Add 16 to each element.
- b- Add 3 to just Odd index elements.
- c- Compute the square root of each element.
- d- Compute the square of each element.

Solution:

Note: We will assume independent operations for the problem.

```
x=[2 5 1 6]
x =
     2     5     1     6

>> x+16                               ...(a)
ans =
    18    21    17    22

16+x(1:2:4)                            ...(b)
ans =
    18    17

>> [sqrt(2) sqrt(5) sqrt(1) sqrt(6)]   ...(c) , or (we may use 2^1/2 5^1/2 ....).
ans =
    1.4142    2.2361    1.0000    2.4495

[2^2 5^2 1^2 6^2]                       ...(d)
ans =
     4    25     1    36
```

Ex(5): Let $x=[3\ 2\ 6\ 8]$ and $y=[4\ 1\ 3\ 5]$

- Add elements in x to y .
- Raise each element of x to the power specified by the corresponding y elements.
- Divide each element of y by the corresponding elements of x .
- Multiply each element in x by the corresponding elements in y and put in z .

Solution

```
>> x=[3 2 6 8]
x =
    3    2    6    8

>> y=[4 1 3 5]
y =
    4    1    3    5

>> x+y                                     ... (1)

ans =

    7    3    9   13

>> [3^y(1) 2^y(2) 6^y(3) 8^y(4)]         ... (2)

ans =
    81     2   216  32768

>> y./x                                    ... (3)

ans =

    1.3333    0.5000    0.5000    0.6250

>> x.*y                                    ... (4)

ans =

    12    2   18   40
```

Ex(6): Write Matlab codes to calculate temperature conversion from Celsius C into Fahrenheit F then to Rankine R. (Rankin).

Solution

```
>> tc=(0:15:100)

tc =

    0    15    30    45    60    75    90

>> tf=1.8*tc+32                                % or: tf=(9/5)*tc+32
tf =
    32    59    86   113   140   167   194
... (F)

>> tr=tf+459.69
tr =
    491.6900   518.6900   545.6900   572.6900   599.6900   626.6900   653.6900
... (R)

>> t=[tc' tf' tr']
t =
    0         32.0000   491.6900
   15.0000   59.0000   518.6900
   30.0000   86.0000   545.6900
   45.0000  113.0000   572.6900
   60.0000  140.0000   599.6900
   75.0000  167.0000   626.6900
   90.0000  194.0000   653.6900
← ...Table of C°, F° and R° .
```

Matrix Algebra

Systems of Linear equations can be solved using matrices.

Consider a set of n linear equations with x_1, x_2, \dots, x_n unknown variables.

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2$$

$$\cdot \quad \cdot \quad \cdot \quad \cdot$$

$$\cdot \quad \cdot \quad \cdot \quad \cdot$$

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n$$

Thus

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ b_n \end{bmatrix} \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

Example: For the matrices, $A = [1 \ -4 \ 3; 3 \ 1 \ -2; 2 \ 1 \ 1]$, $b = [-7; 14; 5]$. Find x values.

```
>> A=[1 -4 3; 3 1 -2; 2 1 1], b=[-7;14;5]

A =

     1     -4     3
     3     1     -2
     2     1     1

b =

    -7
    14
     5

>> X=A\b

X =

     3
     1
    -2
```

...Compute X values.

... x_1 Value.

... x_2 Value.

... x_3 Value.

MATLAB[®] & Simulink[®]

Chapter 6

Introduction to Polynomials in Matlab

**MATLAB[®]
& SIMULINK[®]**

Assist. Professor Emad Jihad

 **MathWorks[®]**

Polynomials in Matlab

One of the important aspects in applied science is the polynomial which has the following general layout format in Numerical Analysis:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

This is an (n^{th}) power polynomial in x , where the (\mathbf{a} 's) are the Coefficients of the polynomial, with Real or Complex number values.

In Matlab, Polynomials are represented by Vectors holding the coefficients of the polynomials, assuming (0) to undefined x values.

Example (1): Represent the following polynomials:

$$y = 3x^3 + 2x^2 - x + 5, \text{ and } w = x^4 - 7x^2 + 2?$$

```
>> y=[3 2 -1 5]
```

```
y =
```

```
    3    2   -1    5
```

```
>> w=[1 0 -7 0 2]
```

```
w =
```

```
    1    0   -7    0    2
```

... Notice that we put (0's) for missed x 's (x^3 and x terms).

Roots of a polynomial

To calculate the roots of a given polynomial, **roots** is used either indirectly or directly.

Example (2): Find the roots of y polynomial in previous example indirectly and directly?

```
>> y=[3 2 -1 5];
```

```
>> r=roots(y)
```

```
r =
```

```
-1.5626
```

```
0.4480 + 0.9306i
```

```
0.4480 - 0.9306i
```

```
>> r=roots([3 2 -1 5])
```

```
r =
```

```
-1.5626
```

```
0.4480 + 0.9306i
```

```
0.4480 - 0.9306i
```

... indirectly use of roots... (imaginary roots can be found)

... Direct use of roots... (imaginary roots can be found)

Note: Always No. of roots found is equal to the polynomial order (n), therefore, the last example showed 3 roots for 3 (x) terms, while the last coefficient is not considered.

Example (3): Find roots of the polynomial $x^4 - 7x^2 + 2$?

```
>> p=[1 0 -7 0 2];
>> r=roots(p)

r =

-2.5887
 2.5887
-0.5463
 0.5463
```

Predicting polynomial values

In order to predict the y value of the data of the fitted polynomial for some other x values, **polyval** is to be used.

Example (4): Determine the value of the fitted polynomial $y= 3x^3 + 2x^2 - x + 5$ for $x=4$?

```
>> p=[3 2 -1 5];
>> y=polyval(p,4)

y =

225
```

...Fitted polynomial for $x = 4$.

Example (5): Find the polynomial value at Multiple points (vectors) of $v1=[3 1 5]$ and $y= [3 2 -1 5]$, then consider the y value for $x=4$ as in previous example?

```
>> y =[3 2 -1 5];
>> v1=[3 1 5];
>>
p=polyval(y,v1)

p =

101 9 425
```

.....

```
>> v1=[3 1 5];
>> y=[3 2 -1 5];
>> y=polyval(y,4);
>> y=polyval(y,v1)

y =

225 225 225
```

... fitted for $x=4$.

Best Fit polynomial coefficients

Use **polyfit** command to determine the polynomial's best fitted coefficients for a given data...the syntax format for this command is: **polyfit(x,y,n)**, where x, y are the data vectors and n is the order of the polynomial for which the Least-Squares Fit is desired.

Example (6): Fit x, y vectors to 4 order polynomial:

$3.2x^4 + 3.5x^3 + 4.1x^2 + 5x + 5.7$ and $-1.7x^4 - 1.5x^3 + 1.1x^2 + 1.5x + 3.6$

```
>> x=[3.2 3.5 4.1 5.0 5.7]
x =
    3.2000    3.5000    4.1000    5.0000    5.7000
>> y=[-1.7 -1.5 1.1 1.5 3.6]
y =
   -1.7000   -1.5000    1.1000    1.5000    3.6000
>> co=polyfit(x,y,4)
co =
    2.2433   -39.1473   251.9575  -706.5981   726.9242
>> yp=polyval(co,x)
yp =
   -1.7000   -1.5000    1.1000    1.5000    3.6000
```

Thus, the polynomial with the best fit coefficients will be:

$$y = 2.2433x^4 - 39.1473x^3 + 251.9575x^2 - 706.5981x + 726.9242$$

We notice in this example that the predicted values of the dependent variable at each value of the independent variable are the same (identical), in other words, the actual y points and the predicted y points are identical.

Exercises

- 1-** Fit the following data describing the accumulation of species A over Time to a second order polynomial, then by using this polynomial, predict the accumulation at 15 hours.

Time(hr)	1	3	5	7	8	10
Mass A acc.	9	55	141	267	345	531

Solution:

```
>> A=[9 55 141 267 345 531];
>> Time=[1 3 5 7 8 10];
>> % Now Fit the data using polyfit command
>> coeff=polyfit(Time,A,2)

coeff =

    5.0000    3.0000    1.0000

>> % So accordingly, Mass A = 5 * (Time)2 + 3 * (Time) + 1
>> % Then we find the Mass A at 15 hours as following:
>> MAPred=polyval(coeff,15)

MAPred =

    1171
```

- 2-** Fit the following vapor pressure vs. temperature data in fourth order polynomial, then calculate the vapor pressure when Temp=100°C.

Temp (C)	-36.7	-19.6	-11.5	-2.6	7.6	15.4	26.1	42.2	60.6	80.1
Pre.(kPa)	1	5	10	20	40	60	100	200	400	760

Solution:

```
>> Temp= [-36.7 -19.6 -11.5 -2.6 7.6 15.4 26.1 42.2 60.6 80.1];
>> Pre= [1 5 10 20 40 60 100 200 400 760];
>> p=polyfit(Temp,Pre,4)

p =

    0.0000    0.0004    0.0360    1.6062    24.6788

>> % Thus VaporPressure = 0.0004 * (Temp)3 + 0.0360 * (Temp)2 + 1.6062 * (Temp) + 24.6788

>> VPressure=polyval(p,100)

VPressure =

    1.3552e+003
```

3- Find the 3rd. order polynomial that satisfies following data of water for saturation temperature and pressure. Using the predicted polynomial, compute the saturation Pressure at 65°C?

Temp C	0	10	20	30	40	50	60	70	80	90	100
Pre(kPa)	.6108	1.227	2.337	4.241	7.375	12.335	19.92	31.16	47.36	70.11	101.33

Solution:

```
>> Temp= [0 10 20 30 40 50 60 70 80 90 100];
>> Pre= [.6108 1.227 2.337 4.241 7.375 12.335 19.92 31.16 47.36 70.11 101.33];
>> p=polyfit(Temp,Pre,3)

p =

    0.0002   -0.0085    0.2604   -0.0534
>>% Thus, SatPressure = 0.0002 * (Temp)3 - 0.0085 * (Temp)2 + 0.2604 * (Temp) - 0.0534

>> SPressure=polyval(p,65)

SPressure =

    24.7565
```

Prepared and Edited By:
Assist. Professor / Emad Jihad
Matlab -2013

MATLAB® & Simulink®

Chapter 7

Introduction to Matlab Programming

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**

Introduction to Matlab Programming

In addition to the command window instructions to execute different needs, Matlab has a powerful Programming capabilities to write and run prewritten and saved programs.


To Start Writing a new program scripts, follow the following steps below:

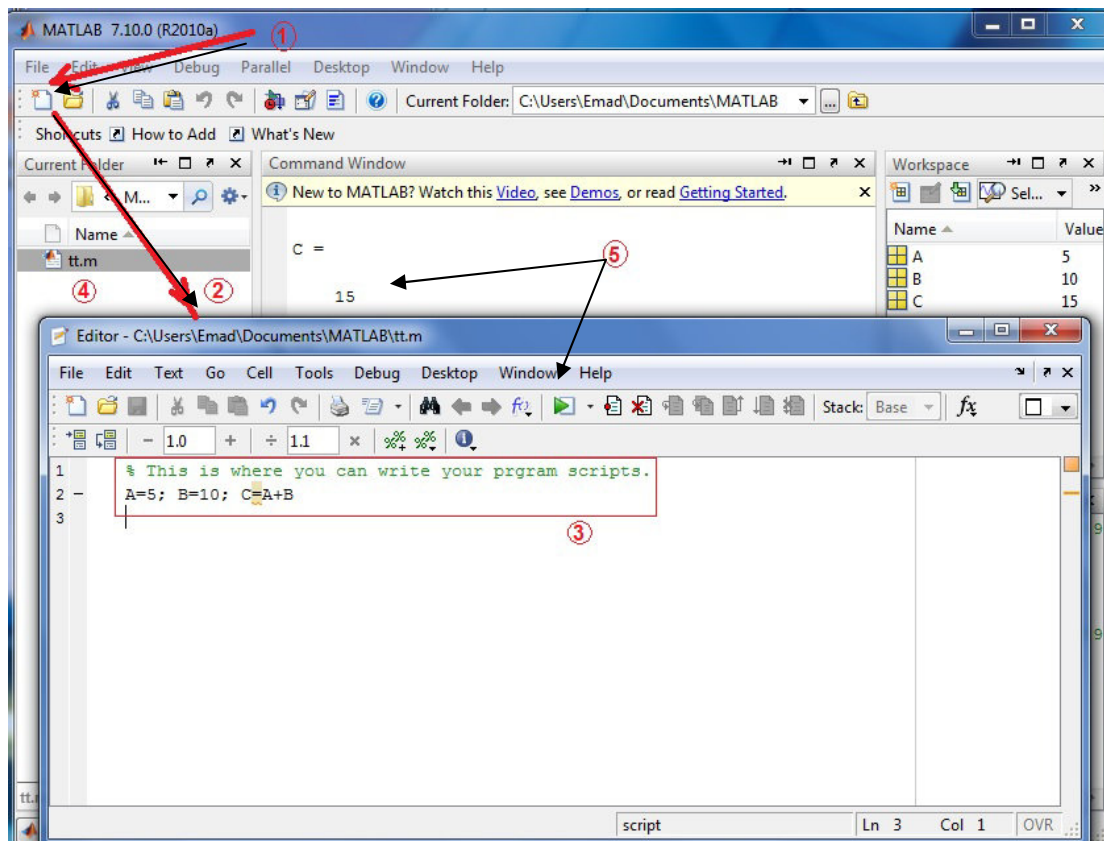
Step 1: Click on New script icon (or, from File command, select New → script).

Step 2: an Editor window displays.

Step 3: Start writing your program scripts.

Step 4: Save the program (Program extension name is **.m**).

Step 5: Run the programs (use  icon), Result will appear back in command window area.

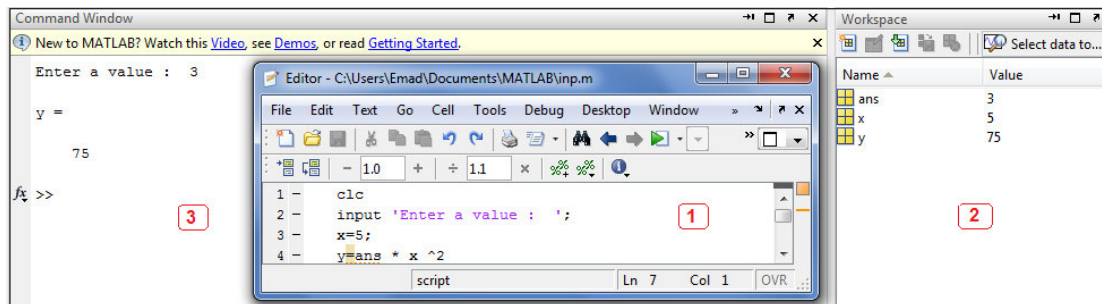


Input function: enables to display a text message along with a numeric value.

Ex1:

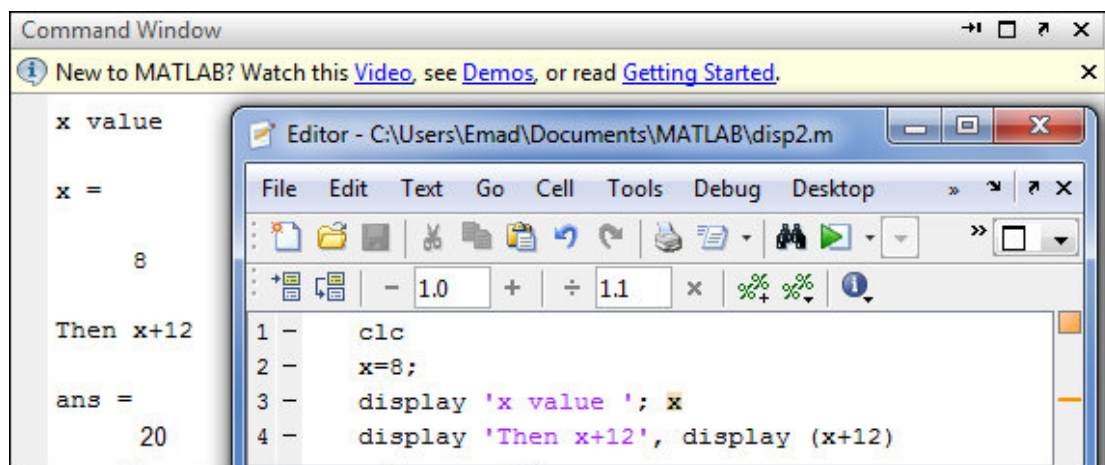
```
>> Input ' Please enter a number: '  
>> Please enter a number: 10  
>>ans =  
    10
```


Ex2: Write a Matlab program to Enter any value to calculate the following equation:
 $y = (\text{value input}) * x ^2$, where x is any defined value.
 Solution: follow the steps in the figure:



Display function: can display a message without any input value.

Ex3: Write a Matlab program to show the following:



Lecture Prepared and Edited by
 Assist. Professor/ Emad Jihad
 Matlab2013

MATLAB® & Simulink®

Chapter 8

Conditions and loops in Matlab Programming

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**

Conditions and Loops in Matlab

Condition

A condition is a circumstance that If happened will give a specific action. Usually in all Programming languages such as VB and other languages, **if** statement is used to check on such conditions. Matlab use the same statement too in a very similar way as in other programming languages.

The General format of simple if statement is:

```
if (relation)  
    (Matlab commands)  
end
```

A more complicated format of if statement (checking for several conditions) will be:

```
if (relation)  
    (Matlab commands)  
    elseif (relation)  
        (Matlab commands)  
    elseif (relation)  
        (Matlab commands)  
    .  
    .  
    .  
else  
    (Matlab commands)  
end
```

Relational operators can be one of the following:

```
== equals  
> Greater than  
< Less than  
<= Less than, equals  
>= Greater than, equals  
~= Not equals
```

While the **Logical** operations are:

```
& And relation  
| Or relation  
~ Not relation
```

Note: It's recommended to construct a Matlab program (option New then Script from File menu), so that you can run it whenever you want (This program will be saved with extension name **.m**).

Loop

By a loop we mean number of iterations that to be repeated to accomplish a specific task. Again, all programming languages have statements of such loops. In Matlab there are two kinds of statement, **for** and **while**.

The **for** statement is usually used for unconditional loop, the **while** is used when checking for a specific condition to happen, so it's used for conditional loop.

The General format concerning the **for loop** is:

```
for variable = expression
    Statements
end
```

The general format of **while loop** is:

```
While condition satisfied
    Statements
end
```

in this case the while statement will work only if the condition relation is true and will keep looping until the condition is no more satisfied.

Note: at any moment loop can be forced to stop using either **Break** statement or **Ctrl+c**.

Example (1): Given a vector with 10 element, check these elements for numbers grater than or equal 5 then change the element to be 100, if the element value is less than 5 then turn this element to (0), otherwise change the element to -100?

Solution:

In program script, write the following code:

```
clc ... the vector elements
a=[1 4 5 1 7 9 2 0 3 8]; ... for statement loop
for i=1:10 ... checking for element < 5
if a(i)<5
    k(i)=0; ... checking for element = 5
elseif a(i)>=5
    k(i)=100; ... If above conditions not satisfied.
else ... end of if statement
    k(i)=-100; ... end of for loop
end ... display vector a elements
end ... display the new vector elements.
a ... Will give results on command window below:
k
```

```
a =
    1     4     5     1     7     9     2     0     3     8
k =
    0     0    100     0    100    100     0     0     0    100
```

Example (2): Write Matlab code to find number between 1 and 3 with incrementing value of 0.5 (using for statement)?

Solution:

```
clc
for i=1:0.5:3
    i
end
```

Will give:

```
i =
    1

i =
    1.5000

i =
    2

i =
    2.5000

i =
    3
```

Example (3): calculate the square root of the summation of odd numbers between (1-50)?

Solution:

```
clc
sum=0;
for i=1:2:50
    sum=sum+i;
end
sum
```

... Odd numbers loop
... sum of odd numbers
...will give: →

```
sum =
    625
```

Example (4): Given a 3 by 3 matrix, write necessary code to change non diagonal elements to zero?

Solution:

```

clc
M=[5 1 2;3 1 5;6 2 1]
for i = 1:3
    for j = 1:3
        if i~=j % if i Not equals j
            M(i,j)=0; % put 0 in non diagonal elements
        end
    end
end
M

```

```

M =
     5     1     2
     3     1     5
     6     2     1

M =
     5     0     0
     0     1     0
     0     0     1

```

Applied Exercise:

Exercise (1): Write Matlab code to calculate the average density, conductivity, and specific heat for water in the range of temperature from (0-50)^o C, providing that these parameters for water are function of temperature as in the following equations:

The Density: $P = 1200.92 - 1.0056 T_k^o + 0.001084 * (T_k^o)^2$

The Conductivity: $K = 0.34 + 9.278 * 10^{-4} * T_k^o$

The Specific heat: $C_p = 0.015539 * (T_k^o - 308.2)^2 + 4180.9$

Note: 1- Take 11 points of temperature.

2- Temperature in Kelvin degree = Celsius degree + 273.

Solution:

```

clc
aP=0; aKc=0; aCp=0;
for T=273:5:323
    P= 1200.92-1.0056 * T + 0.001084 * T^2;
    Kc=0.34+9.278*10^-4 * T;
    Cp=0.015539*(T-308.2)^2 +4180.9;
    aP=aP+P;
    aKc=aKc+Kc;
    aCp=aCp+Cp;
end
Averagedensity=aP/11
Averagecoductivity=aKc/11
Averagespecialheat=aCp/11

```

```

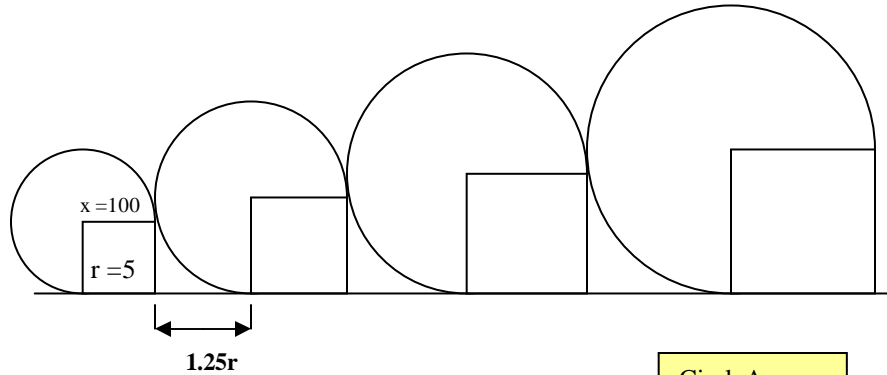
Averagedensity =
    997.7857

Averagecoductivity =
    0.6165

Averagespecialheat =
    4.1864e+003

```

Exercise (2): Write Matlab code to compute the Areas of 4 Circles and Squares shown in the figure below, knowing that the first circle center point in x-axis = 100, radius = 5 and is increasing for others in 1.25 of the previous radius, with center point displacement = radius?



Solution:

```

clc
x=100;
r=5;
CircleArea= r^2*pi
SquareArea=r^2
for i=1:3
    r=r*1.25;
    x=x+r;
    CircleArea= r^2*pi
    SquareArea=r^2
end

```



```

CircleArea =
    78.5398

SquareArea =
    25

CircleArea =
    122.7185

SquareArea =
    39.0625

CircleArea =
    191.7476

SquareArea =
    61.0352

CircleArea =
    299.6056

SquareArea =
    95.3674

```

Exercise (3): Write Matlab code to calculate t values in respect to the following:

- t= 200When y is below 10.000
- = 200 + 0.1 (y - 10.000) When y is between 10.000 and 20.000
- = 1.200 + 0.15 (y - 20.000) When y is between 20.000 and 50.000
- = 5.700 + 0.25 (y - 50.000) When y is above 50.000?

To test the program, try values of y = 5.000, 17.000, 25.000, and 75.000.

Solution:

When applying y test values we get:

```
if y<10.000
    t=200
elseif y>=10.000 & y<=20.000
    t=200+0.1*(y-10.000)
elseif y>=20.000 & y<=50.000
    t=1.200+0.15*(y-20.000)
elseif y>50.000
    t=5.700+0.25*(y-50.000)
end
```

```
>> y=5.000;
t =
    200
>> y=17.000;
t =
    200.7000
>> y=25.000;
t =
    201.5000
>> y=75.000;
t =
    206.5000
```

Lecture prepared and edited by:
Assist. Professor/ Emad Jihad
2013

MATLAB® & Simulink®

Chapter 9

Introduction to Graph plot in Matlab

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**

Graph plot in Matlab

Computer Graphics (CG) is one of the most important uses in all fields. Graph plotting considered one of implementations of CG.

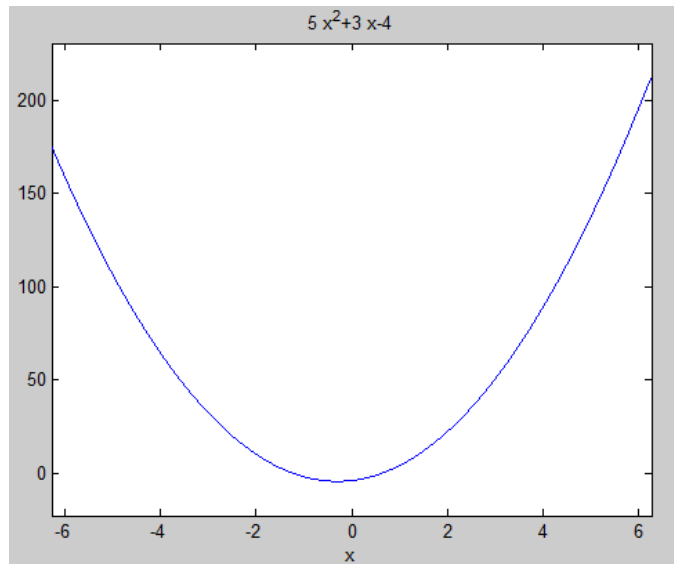
Matlab provides many methods to deal and generate plotted Graphs representing mathematical and statistical data. When plotting a graph, a special Graphic window displayed

1- Plotting symbolic functions

In order to plot a function graph, **ezplot** command (easy plot) is used to plot.

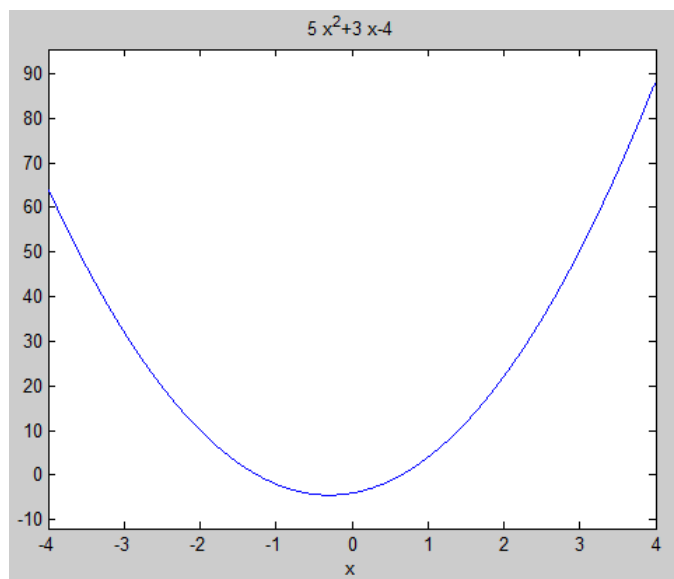
Example (1): Plot the function: $y = 5x^2 + 3x - 4$:

```
>> ezplot('5*x^2+3*x-4')
```



Or, we can plot the same Plot with different axis scale
In case we define values to it.
For example between (-4 and 4)
To the independent variable.

```
>>ezplot('5*x^2+3*x-4',[-4:4])
```

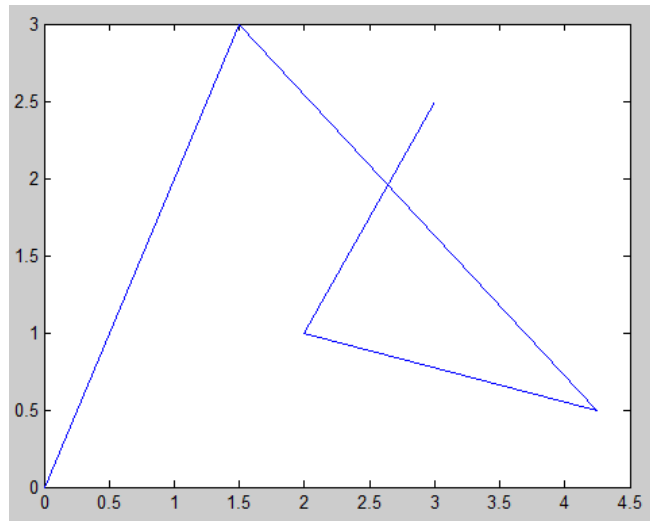


2- X-Y Plots (vectors)

Using vectors to define Cartesian x,y data pairs with same number of elements in both axis. Let x vector be the x-coordinates with (x_1, x_2, \dots, x_n) elements, and y vector the y-coordinates with (y_1, y_2, \dots, y_n) elements then the command **plot(x,y)** will generate the graph of these pair of elements.

Example (2): Plot the graph of two dimensional for the following x,y coordinates: (0,0), (1.5,3), (4.25,0.5), (2,1), (3,2.5)?

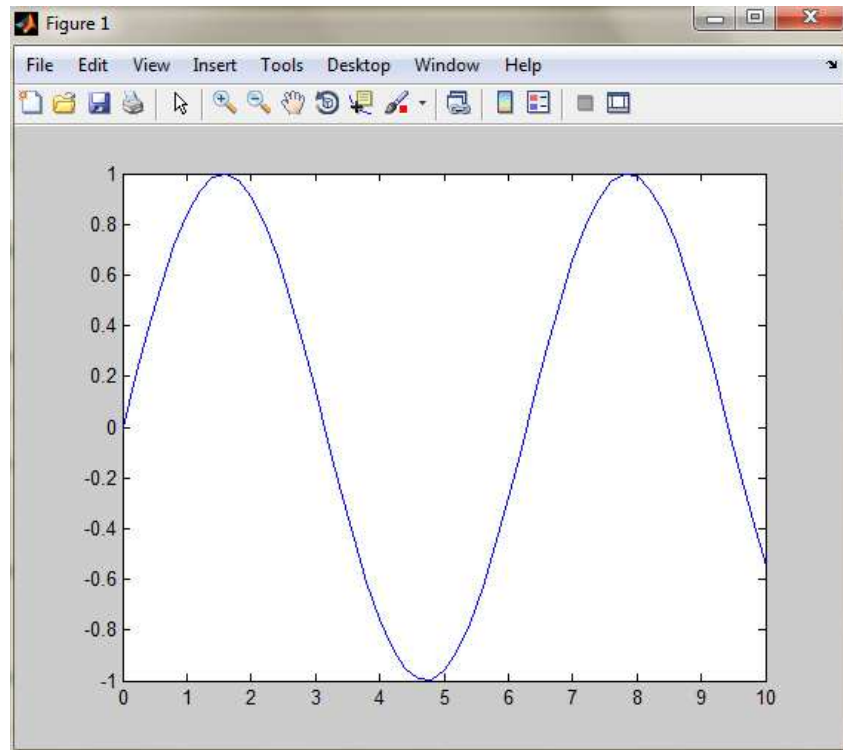
```
>> x=[0 1.5 4.25 2 3];  
>> y=[0 3 0.5 1 2.5];  
>> plot(x,y)
```



Matlab can plot different functions
Such as sin, cos, log, ...etc.

Example (3): Plot sin function for x values 0 to 10 with increasing value of 0.2?

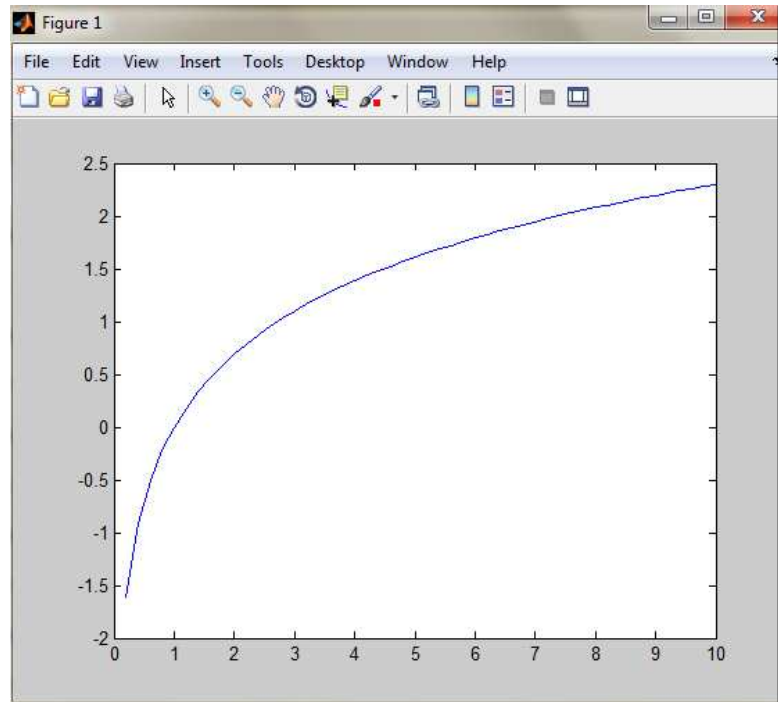
```
>> x=[0:0.2:10];  
>> y=sin(x);  
>> plot(x,y)
```



Notice that the x-axis
Scale according to x
Vector values, while
The depending
y-axis scales according
to function given.

Example (4): For the same x values in previous example, plot the function: $y = \log(x)$?

```
>> y=log(x);  
>> plot(x,y)
```



Decorating the graph:

Matlab has the ability to Display the output graph In various forms.

These decorations can be Done either from menu Bar of the graph window, Or by using items within The plot command, such as:

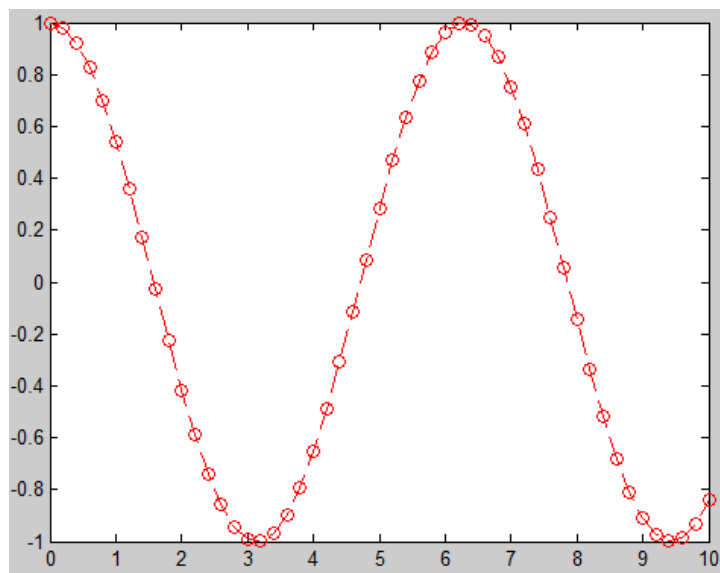
Character colors: k (black), g (green), r (red), b (blue)...

Character symbol: . (point) , o (circle), * (star)...etc

Character Line style: - (solid), : (dotted), -. (dashdot), - - (dashed)...etc

Example (4): Plot function $y=\cos(x)$, with red color, circles and dotted curve?

```
>> y=cos(x);  
>> plot(x,y,'ro--')
```

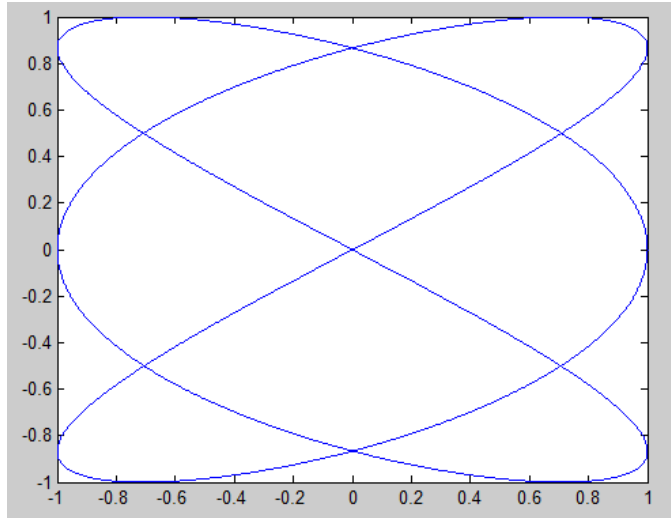


We can also control the plot axis by defining the minimum and maximum values for x and y axis by using **axis([x.min x.max y.min y.max]).** For example `>> axis([-4 4 -10 10]).`

Plots of parametrically defined curves can also be made as in the following example.

Example (5):

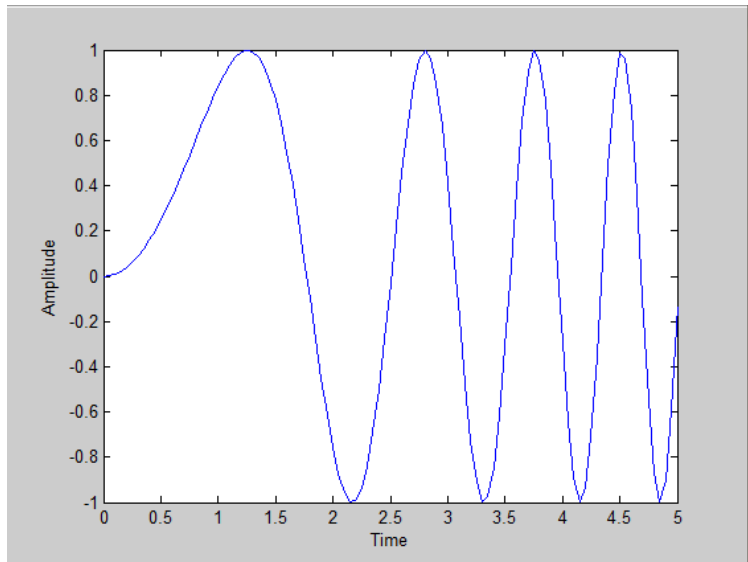
```
>> t=0:0.001:2*pi;  
>> x=cos(3*t);  
>> y=sin(2*t);  
>> plot(x,y)
```



To give titles to axis we can either use the plot window menu bar then choose insert : X Label, Y Label and Title, or use direct command as in the examples below.

Example (6): Plot the function $y = \sin(x^2)$ with x-axis, y-axis label titles Time and Amplitude respectively?

```
y=sin(x.^2);  
plot(x,y);  
xlabel('Time')  
ylabel('Amplitude')
```



Notice that we used $\sin(x.^2)$ to define the Function in this case.

Multiple plots

Another Matlab feature is to plot more one graph without removing each other by using **figure** command. Also more than one graph can be plotted in the same plot window:

Example (7): Plot functions $y=\sin(x)$ and $y=\exp(x.^2)$ and keep both displayed for x value (-1 to 1 with increasing point 0.01)?

```
>> X=[-1:0.01:1];
>> Y=exp(X.^2);
>> plot(X,Y)
>> ylabel('Exp (X^2)')
>> title('Function y = exp(x^2)')
>> figure
>> X=[-1:0.01:1];
>> Y=sin(X);
>> plot(X,Y)
>> ylabel('sin (X) in radian')
>> title('Function y = sin (X)')
```

... Using figure command enables to display more than one plot at the same time in two graph windows.

figure 1.

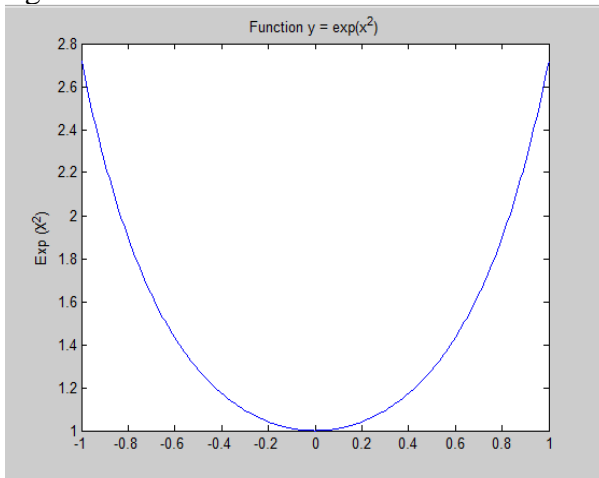
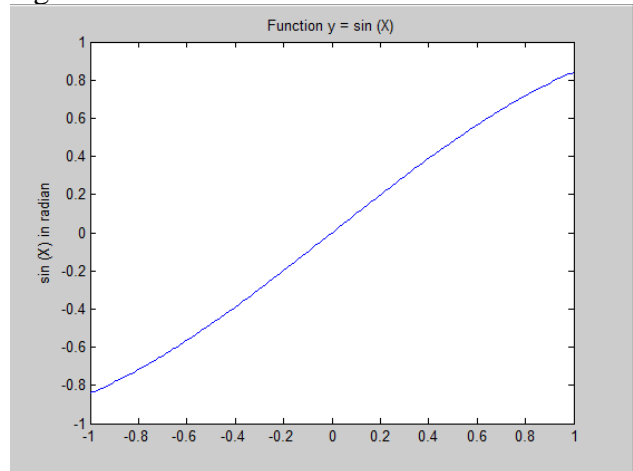
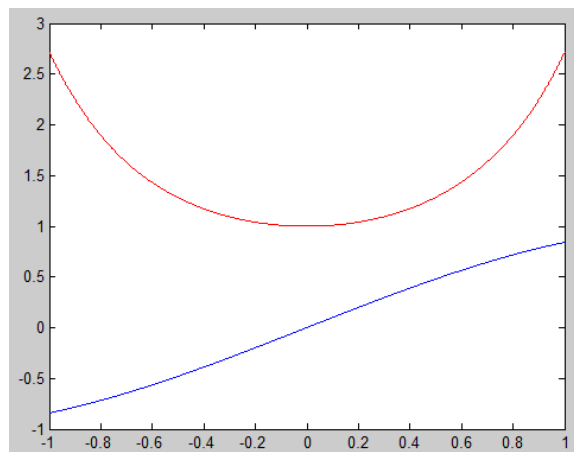


figure 2.



Now to plot two or in general multiple plots in one graph , **hold on** command is used. However, the axes may be rescaled according to this as see in the example below:

```
>> X=[-1:0.001:1];
>> Y=exp(X.^2);
>> plot(X,Y,'r')
>> hold on;
>> X=[-1:0.001:1];
>> Y=sin(X);
>> plot(X,Y,'b')
```



This situation remains active until Switched off by **hold off**.

Subplot

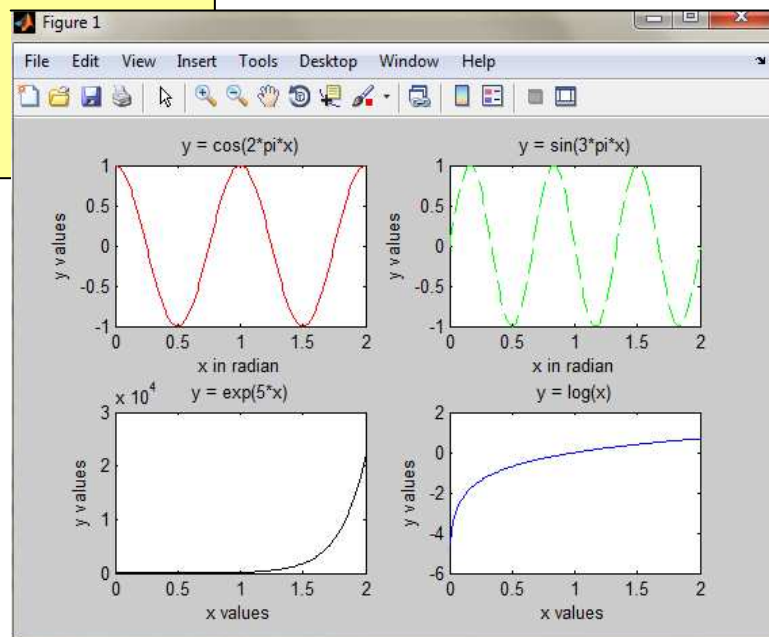
To split the graph plot window to several or multiple (n X m) array of smaller windows, another Matlab command is used which is **subplot**.

The general format is subplot(nmp), where n and m are the number of smaller window plots or the rows and columns, p is the sequence of the plot within the whole graph.

Properties such as hold on, grid, color or others will work on each individual subplots.

Example (8): Suppose we want to plot 4 plots (2X2) in one graph, then the code could be as in below:

```
>> x=0:0.01:2;
>> subplot(221), plot(x,cos(2*pi*x),'r') ... Defining subplot (1)
>> xlabel('x in radian')
>> ylabel('y values')
>> title('y = cos(2*pi*x)')
>> subplot(222), plot(x,sin(3*pi*x),'g--') ... Defining subplot (2)
>> xlabel('x in radian')
>> ylabel('y values')
>> title('y = sin(3*pi*x)')
>> subplot(223)
>> plot(x,exp(3*pi*x),'ko')
>> subplot(223)
>> plot(x,exp(5*x),'k-')
>> xlabel('x values')
>> ylabel('y values')
>> title('y = exp(5*x)')
>> subplot(224)
>> plot(x,log(x),'b')
>> xlabel('x values')
>> ylabel('y values')
>> title('y = log(x)')
```



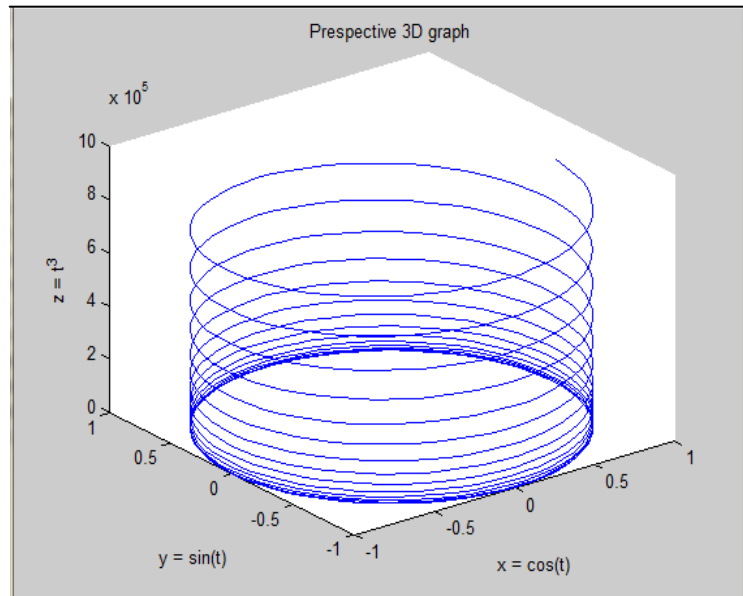
Three Dimensional Plots

Matlab can produce 3 dim. Plots by simply using command **plot3** that will turn curves of 2 dim. Into 3 dim.

To plot a 3D graph, we need to define the x, y, and z axis in order to make the graph. As mentioned before, the number of elements of each of the three vectors must be the same and are usually defined parametrically.

Example (9): Plot a 3D graph if the values of t were ($t=0.01:0.1:30*\pi$), and the three vectors of $x = \cos(t)$, $y = \sin(t)$, and $z=t^3$.

```
>> t=0.01:0.1:30*pi;
>> x=cos(t);
>> y=sin(t);
>> z=t.^3;
>> plot3(x,y,z)
>> xlabel('x = cos(t)')
>> ylabel('y = sin(t)')
>> zlabel('z = t^3')
>> title('Prespective 3D graph')
```

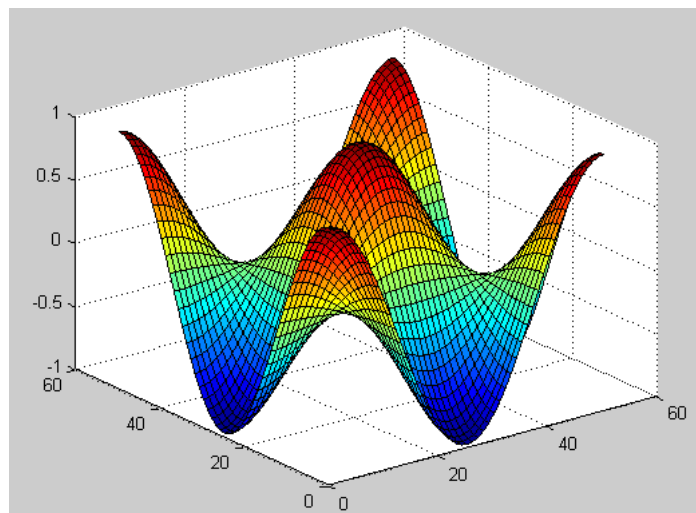


Surface

We can plot the surface plot of square matrix elements forming also a 3D graph using **surf** command as illustrated in the following example.

Example (10): Given a vector $v=[0:0.02:1]$, plot the surface of $y' * y$ matrix, when $y = \cos(v*2*\pi)$?

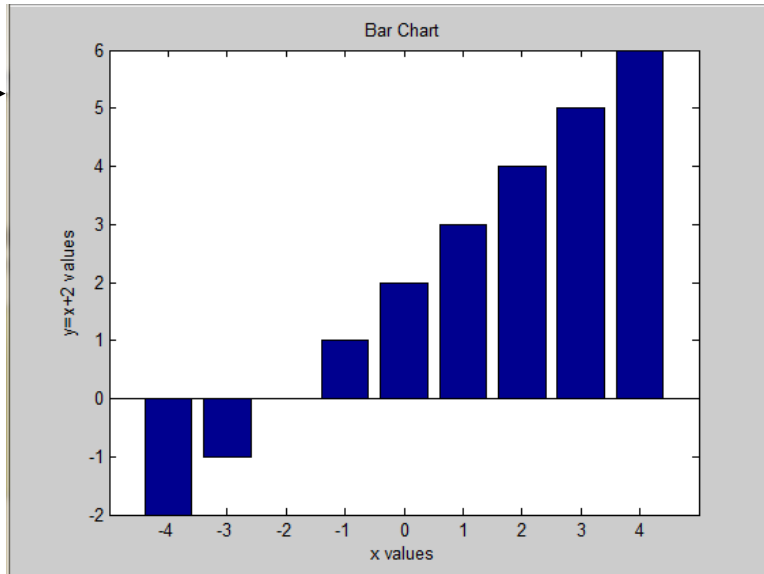
```
>> v=0:0.02:1;
>> y=cos(v*2*pi);
>> m=y'*y;
>> surf(m)
```



Bar chart plot

We can also plot bar chart or graph instead of curves using **bar** command as follows:
Let $x=-4$ to 4 and $y=x+2$, by using bar command we get:

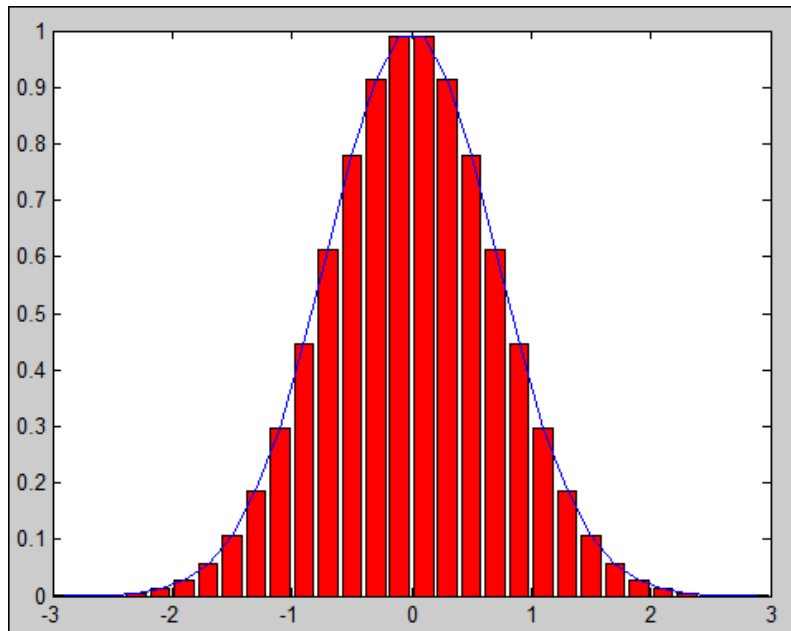
```
>> x=-4:4;  
>> y=x+2;  
>> bar(x,y)  
>> xlabel('x values')  
>> ylabel('y=x+2 values')  
>> title('Bar Chart')
```



We may also combine two or more different plotting styles in one graph window:

```
>> x = -2.9:0.2:2.9;  
>> bar(x,exp(-x.*x),'r')  
>> hold on  
>> plot(x,exp(-x.^2))
```

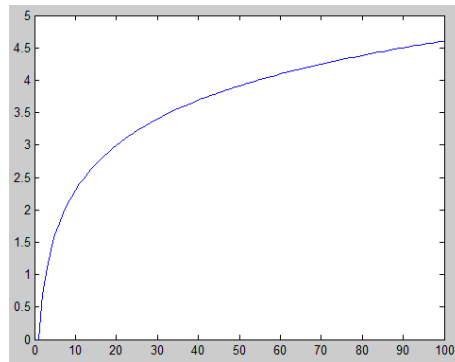
Notice the use of hold on
Command to enable plotting
More than on plot on the
Same graph window as
Mentioned before!



Simple Problems

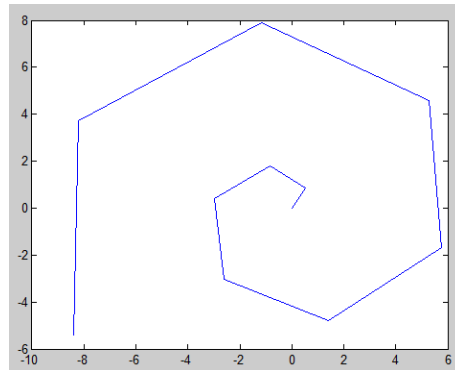
1- Plot the log of the values from 1 to 100?

```
>> x=1:100;  
>> y=log(x);  
>> plot(x,y)
```



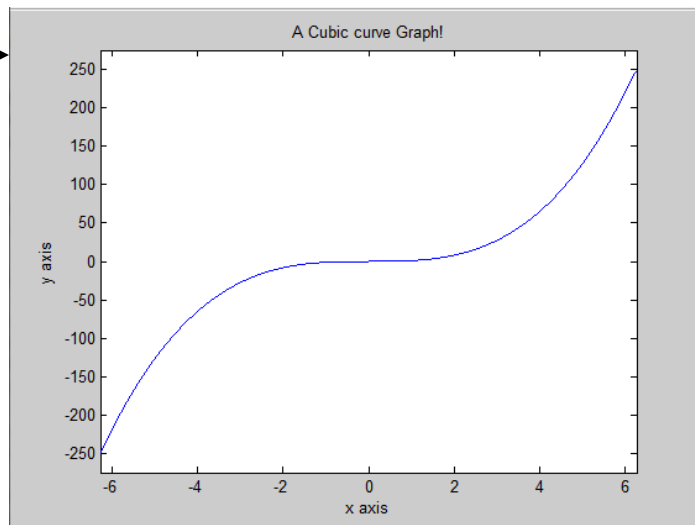
2- Plot the Parametric curve $x=t.\cos(t)$, $y=t.\sin(t)$ for $0 \leq t \leq 10$.

```
>> t=0:10;  
>> x=t.*cos(t);  
>> y=t.*sin(t);  
>> plot(x,y)
```



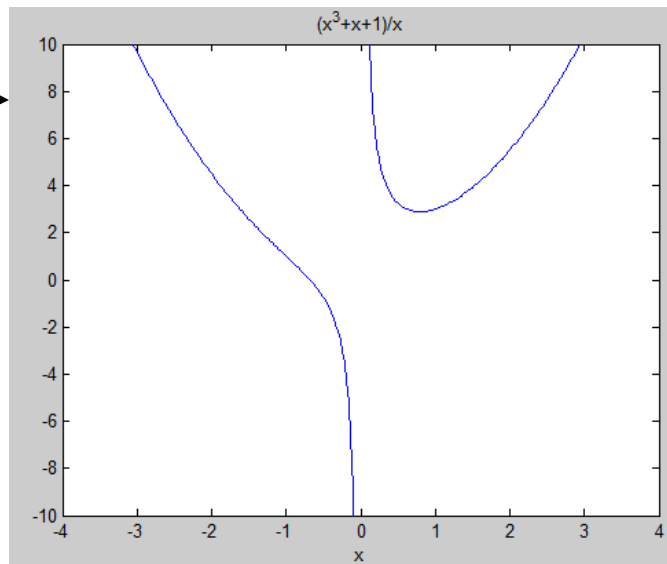
3- Plot the cubic curve $y=x^3$, label the axis and put a title “A Cubic curve Graph”.

```
>> ezplot('x^3')  
>> xlabel('x axis')  
>> ylabel('y axis')  
>> title('A Cubic curve Graph!')
```



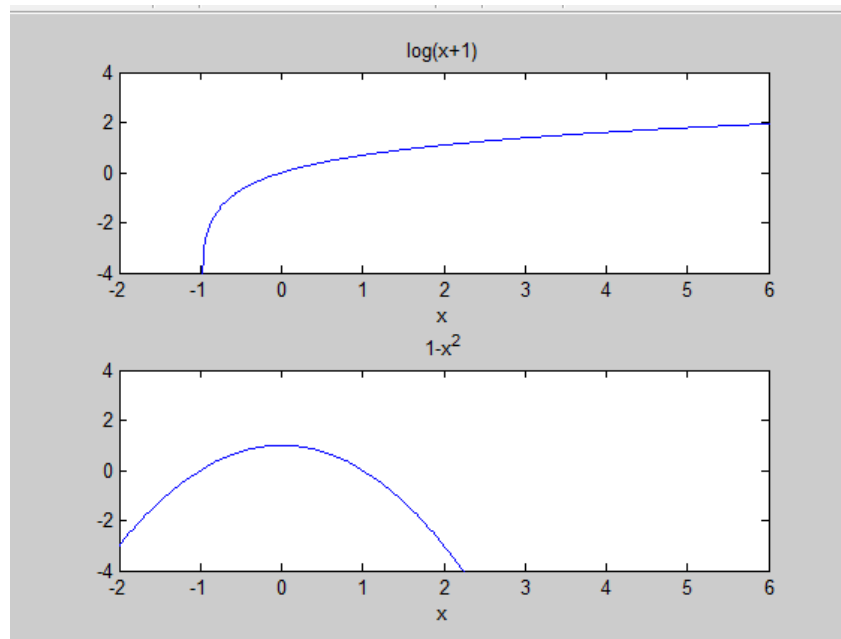
4- Plot $y=(x^3 + x + 1)/ x$, for $-4 \leq x \leq 4$ and $-10 \leq y \leq 10$.

```
>> ezplot('(x^3+x+1)/x')  
>> axis([-4 4 -10 10])
```



5- Plot $y=\ln(x+1)$ and $y=1-x^2$ on the same window for $-2 < x < 6$ and $-4 < y < 4$.

```
>> ezplot('log(x+1)')  
>> subplot(211)  
>> ezplot('log(x+1)')  
>> axis([-2 6 -4 4])  
>> subplot(212)  
>> ezplot('1-x^2')  
>> axis([-2 6 -4 4])
```



Lecture prepared and edited by
Assist. Professor/ Emad Jihad
2013

MATLAB® & Simulink®

Chapter 10

Introduction Image Processing in Matlab

To

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**

Image Processing in Matlab (a basics introduction)

Matlab is widely used in many Information Technology (IT) and computer science fields such as Image processing.

Images are represented in 2 dim arrays. An Array of **Gray scale** image needs to include numbers represent the *Pixels intensity* value and the *Gray level*.

Gray level will be (0) for Black level and (255) for the White level, while other in between levels should be (> 0 and < 255) according to the pixels Gray levels.

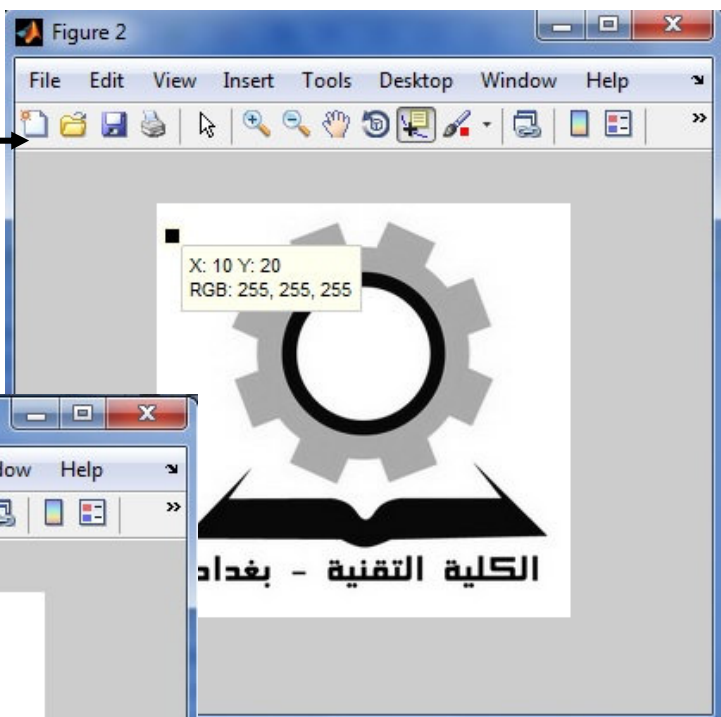
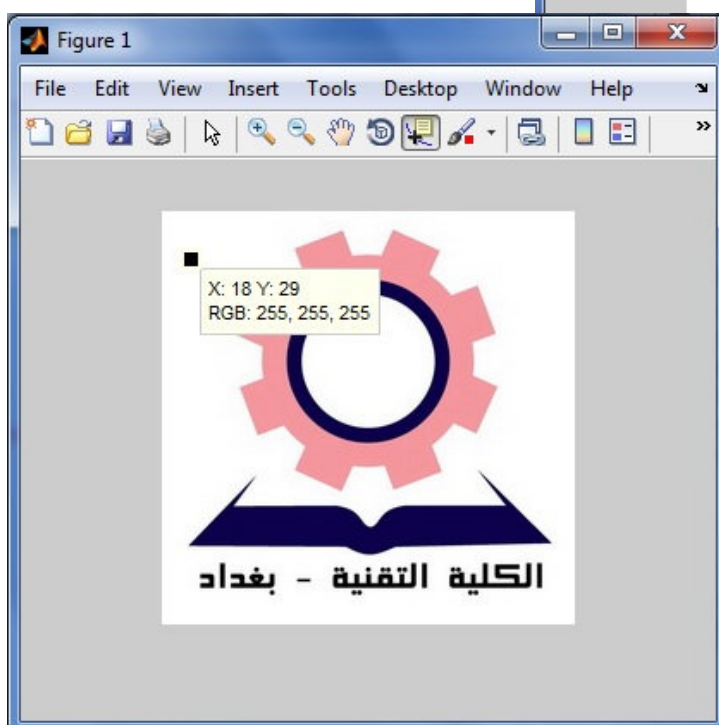
For **Color images**, we need to define (3) 2-dim. Arrays to hold the values of the three main colors **RGB** (Red, Green, and Blue) for each pixel.

To read a (jpg) image format file, **imread** ('imagefilename.jpg') is to be used, then **figure** and **imshow** commands are used to show the image on figures window.

Reading/Showing image files in Matlab

Ex1: Show how Matlab read and display a B/W image ('tcb-bw.jpg') and a color image ('tcb.jpg')?

```
>> % Black and White image
>> I1 = imread('tcb-bw.jpg');
>> figure;
>> imshow(I1);
>>% Color image
>> I2 = imread('tcb.jpg');
>> figure;
>> imshow(I2);
```



For Both cases we picked a pixel Show coordinates and RGB value of (255,255,255) for White color.
Note: For some colored images Converted to Gray scale image, Gray scale might be also (0~255), But (0~1) is possible too.

Also, we can show the image directly without using the figure command as we did.

```
>> image1 = imread('tcb.jpg');  
>> imshow (image1);
```

This is practical only when using 1 figure, otherwise figure command should be used.

Sub imaging

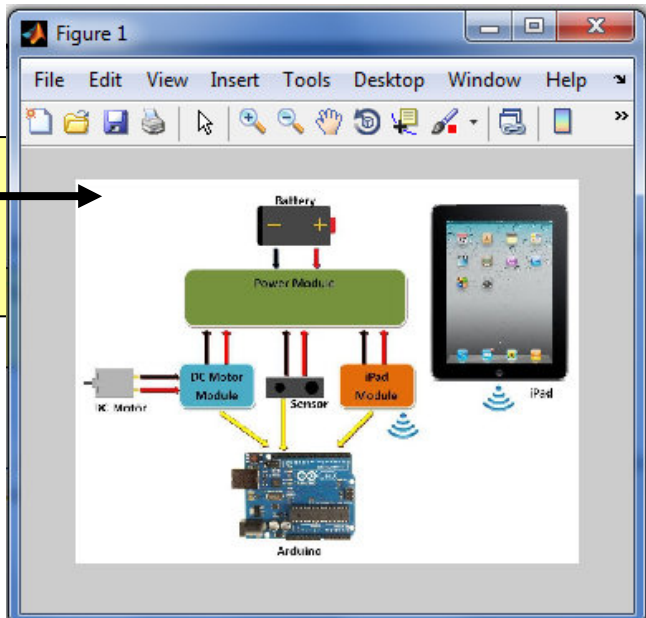
As an image is a 2 dim. Array of pixel values, thus we can get a sub image from the original one as we explained before in the vectors and Matrices chapter.

The new sub image will be defined as the following: $IS=Iss(x1:x2,y1:y2,1:3)$

Where (IS) is any name for the source image, (Iss) is the new sub image, $x1,x2,y1,y2$ are the beginning and ending pixels as rows and columns, while (1:3) is the third dimension which gives the three RGB color channels used (i.e. R=1, G=2, and B=3).

Ex2: Perform sub image segmentation from a given image?

```
>> IS=imread('MechaTron.jpg');  
>> imshow(IS);  
>> Iss=IS(200:400,100:300,1:3);  
>> imshow(Iss);
```

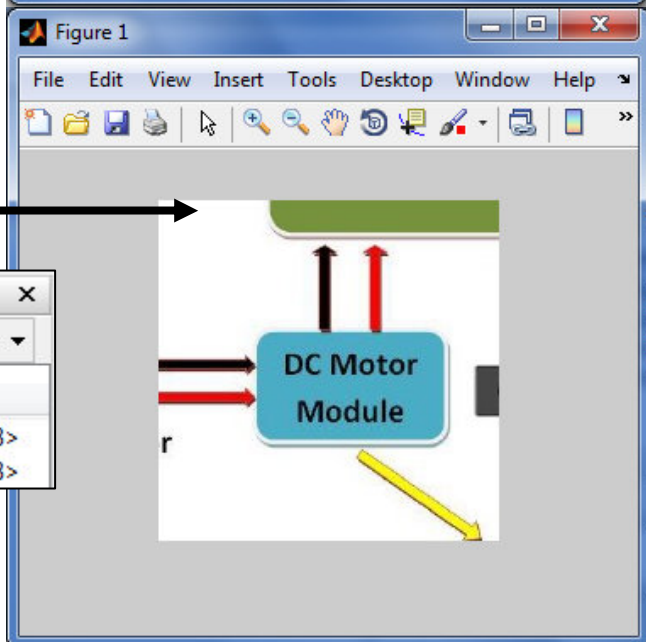


In Matlab, We can know No. Of Image rows and columns from The Workspace window.. as shown Below:

Name	Value
IS	<575x756x3 uint8>
Iss	<201x501x3 uint8>

Or, simply use **size** command:

```
>> size(IS)  
ans =  
    575    756     3
```



Converting Color image to Gray scale image

To convert from color to Gray level image, **rgb2gray**(*image name*) command is used, where image name is any color image file with extension (usually jpg).

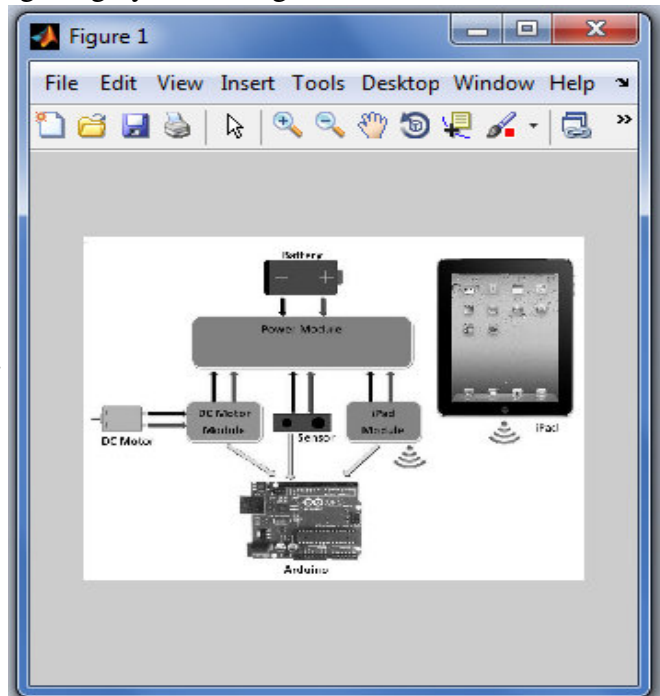
Ex3: Convert the previous color image to gray level image.

```
>> GI = rgb2gray(IS);  
>> imshow(GI);
```

Now to Save the New Gray scale Image, **imwrite** command is used.

Ex4: Save the previous gray scale Image into a new file.

```
>> imwrite(GI,'ISGray.jpg');
```

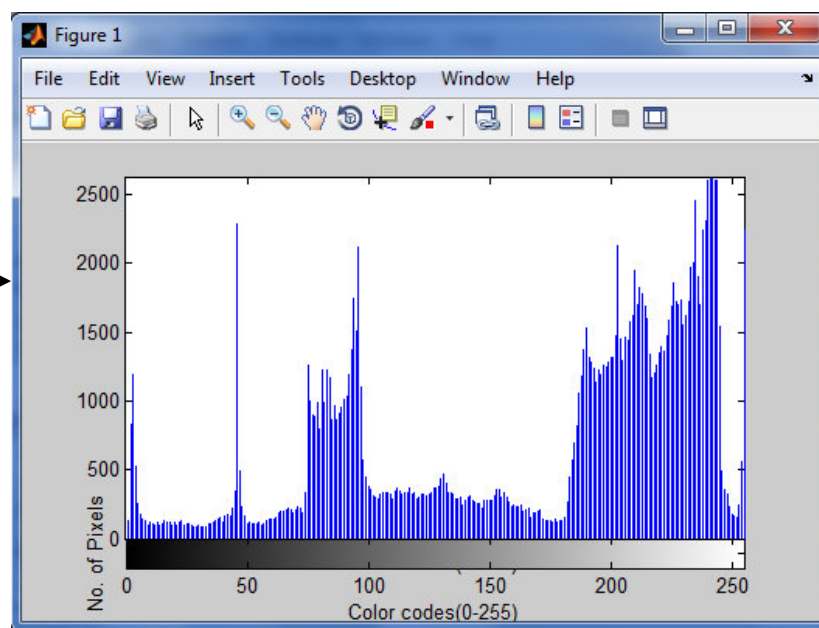


Drawing the Histogram of an image pixel colors

imhist command is used to show a histogram chart of No. of pixels according to colors as shown below:

Ex5: Draw a histogram chart showing the No. of pixels according to colors of an image?

```
>> imhist(GI)
```

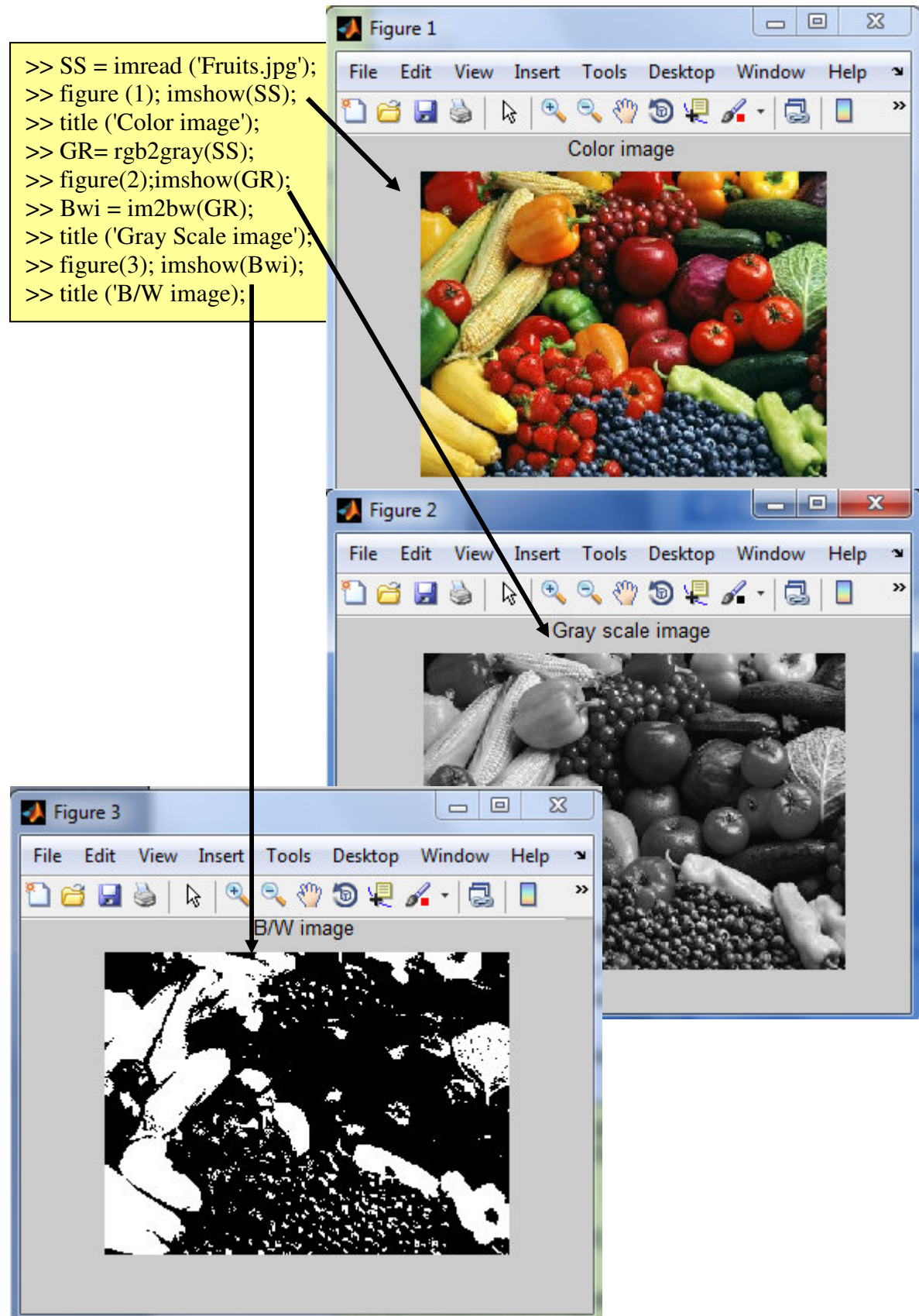


Converting From Gray scale to Black and White (BW) image

To convert an image from Gray level to BW, we use **im2bw** command, Thus only 2 colors image will be generated.

Ex6: Convert an image from Color to Gray level, then to a BW image.

```
>> SS = imread ('Fruits.jpg');  
>> figure (1); imshow(SS);  
>> title ('Color image');  
>> GR= rgb2gray(SS);  
>> figure(2);imshow(GR);  
>> Bwi = im2bw(GR);  
>> title ('Gray Scale image');  
>> figure(3); imshow(Bwi);  
>> title ('B/W image');
```



The image displays three MATLAB figure windows illustrating the conversion process:

- Figure 1:** Shows the original color image of various fruits, titled "Color image".
- Figure 2:** Shows the same fruit image converted to grayscale, titled "Gray scale image".
- Figure 3:** Shows the grayscale image converted to a binary (black and white) image, titled "B/W image".

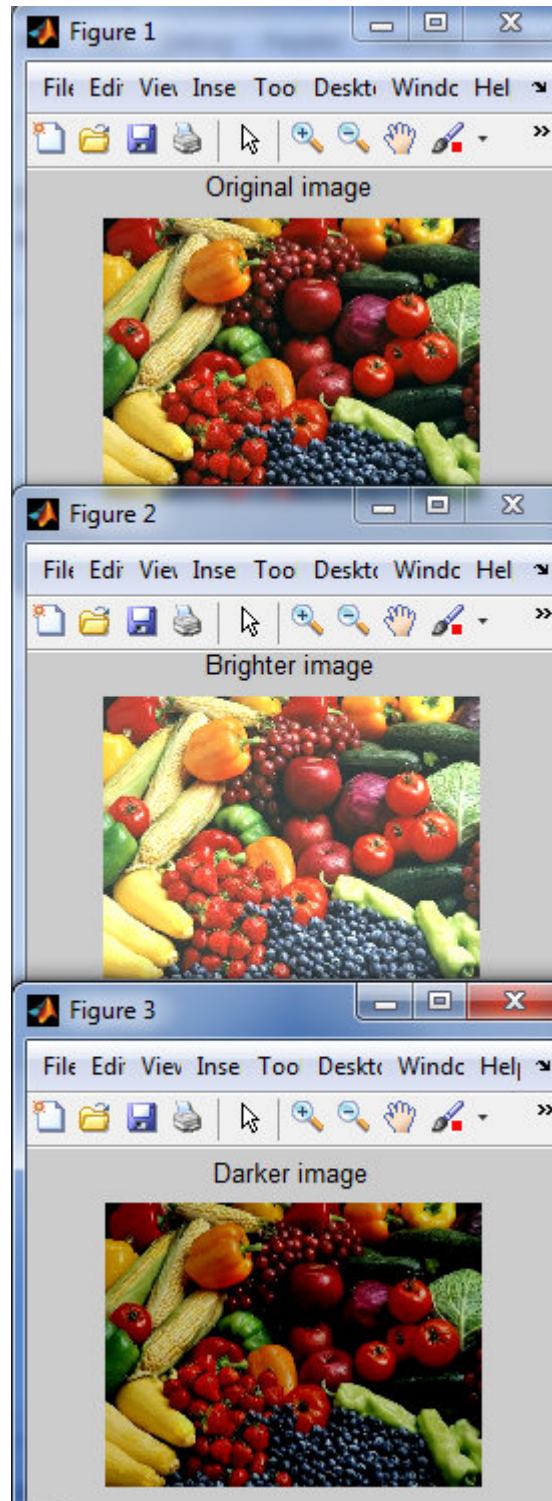
Arrows from the code block point to each corresponding figure window.

Image Brightness

To increase or Decrease the Brightness of an image, just us (+) for Brighter or (-) for Darker image!

Ex7: For the Color image in Ex6. Increase the Brightness first, then Decrease the original image brightness by the value of (50).

```
>> SS=imread('Fruits.jpg');  
>> figure(1);imshow(SS);  
>> SSB=SS+50;  
>> figure(2);imshow(SSB);  
>> SSD=SS-50;  
>> figure(3);imshow(SSD);
```



Lecture prepared and edited by: Assist. Professor / Emad Jihad

MATLAB® & Simulink®

Chapter 11

Introduction to Graphical User Interface (GUI) Design & Programs

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**

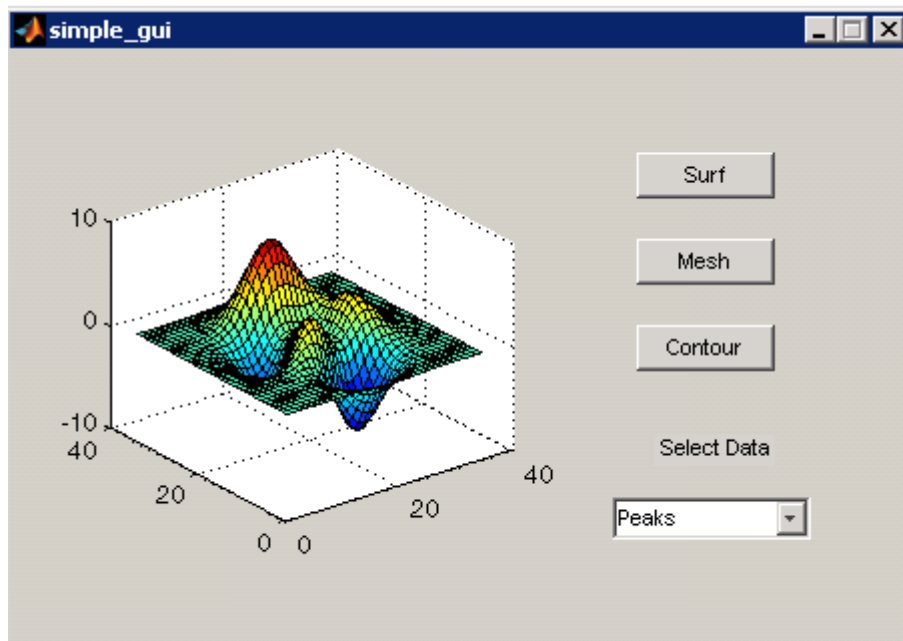
What Is a GUI?

A **Graphical User Interface (GUI)** is a graphical display in one or more windows containing controls, called components that enable a user to perform interactive tasks. Modern Programming languages such as VB, visual C++, .etc. And also Matlab are using several components or Objects along with programming code or script to design an **interactive** application.

The user of the GUI does not have to create a script or type commands at the command line to accomplish the tasks. Unlike coding programs to accomplish tasks, the user of a GUI need not understand the details of how the tasks are performed .

GUI components can include menus, toolbars, push buttons, radio buttons, list boxes, and sliders—just to name a few. GUIs created using MATLAB tools can also perform any type of computation, read and write data files, communicate with other GUIs, and display data as tables or as plots.

The following figure illustrates a simple GUI that you can easily build yourself .



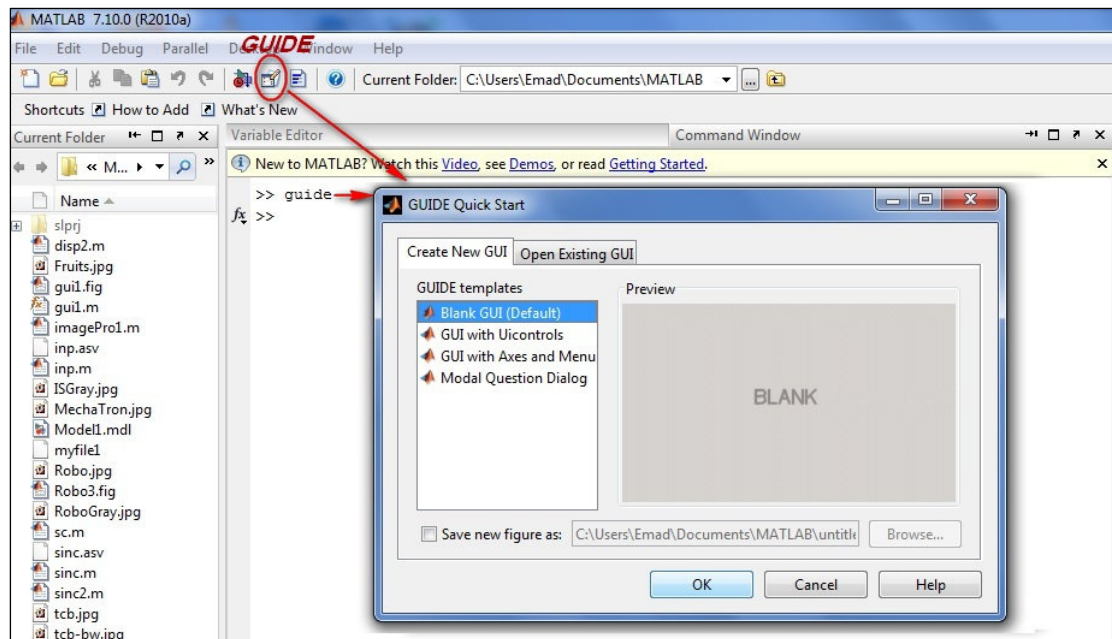
As an example, The above GUI contains:


- An axes component
- A pop-up menu listing three data sets that correspond to MATLAB functions: peaks, membrane, and sinc
- A static text component to label the pop-up menu
- Three buttons that provide different kinds of plots: surface, mesh, and contour

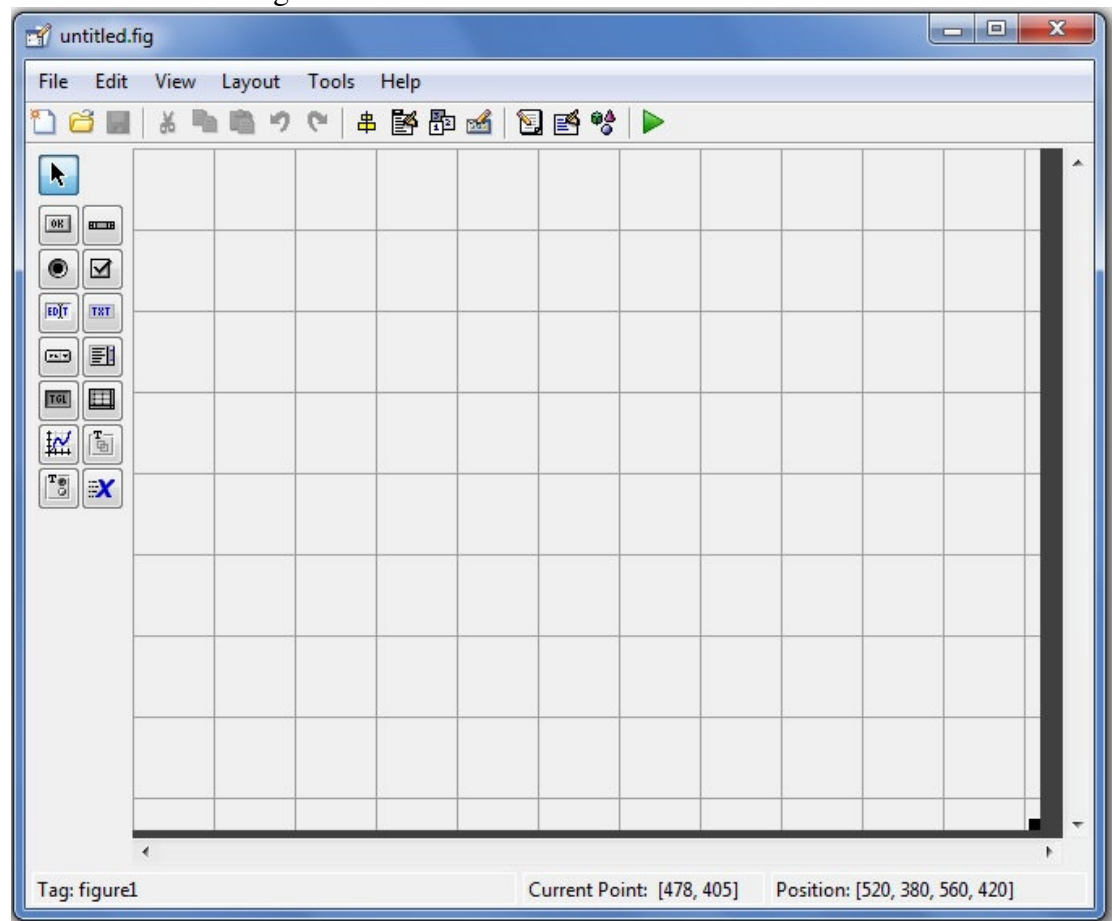
When you click a push button, the axes component displays the selected data set using the specified type of 3-D plot .

Getting started

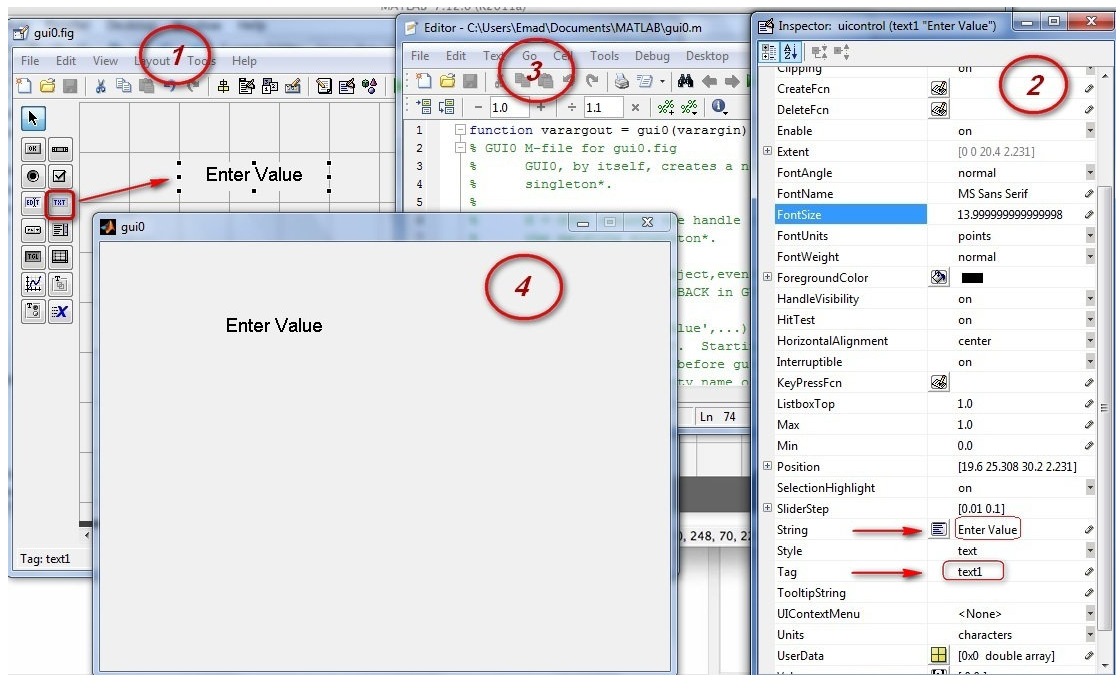
To start using GUI, from menu tool bar, select the (GUIDE) option or directly write the command on Matlab command space, then as a new window appears, select Blank GUI to open a new job, then press (ok) as seen below.




A new GUI window will appear, on the left side you will find the necessary tools or components (objects) to design your job, that when selected, enables to locate and put on the design grid area. After finishing the design with all script, the Run figure  tool will start running it.



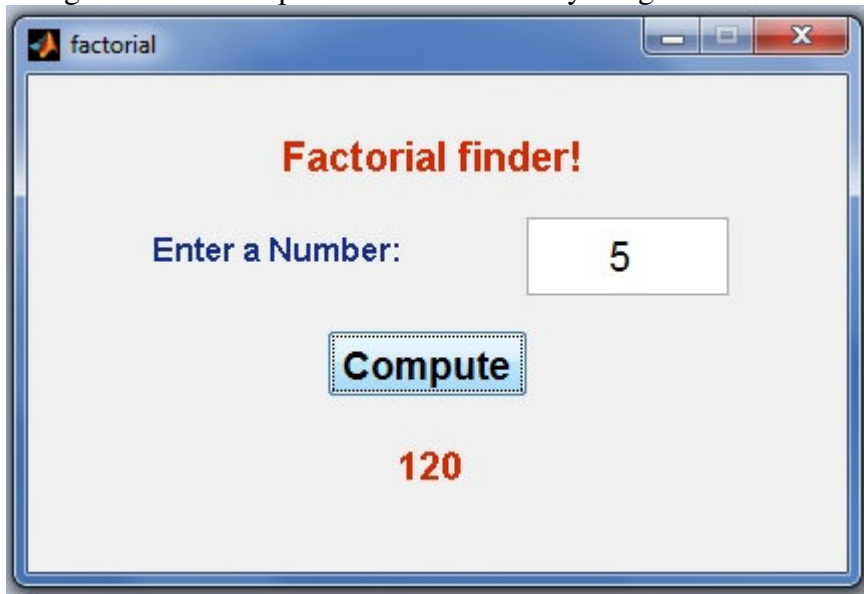
The following figure illustrates the procedures of making a GUI application.



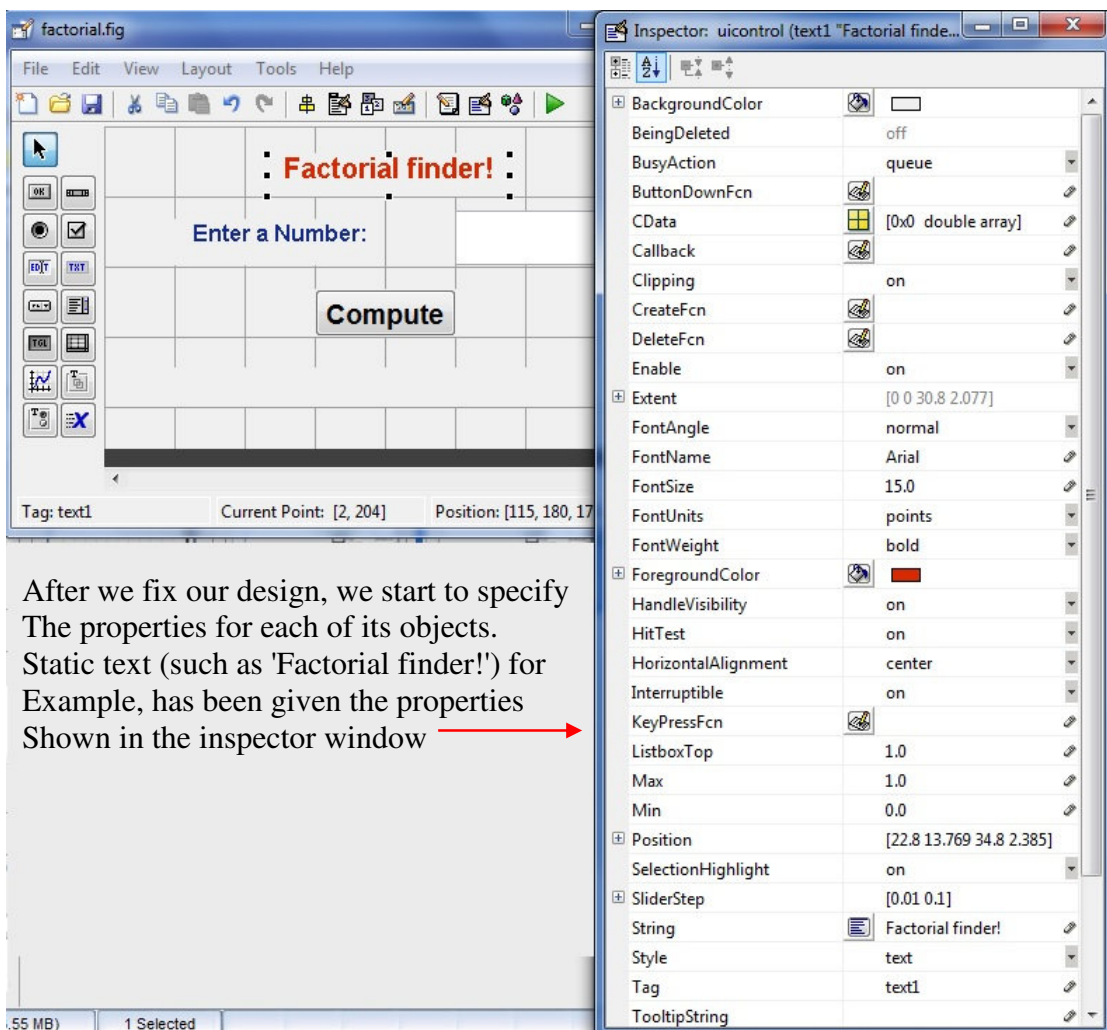
- 1- Select components from tools option and locate them on grid area.
- 2- Double click on any selected component (for example, the Edit text), an inspector window appears with all properties you need to specify (for instance, Font size, color, string (the caption string of the text), Tag...etc.
- 3- Editor script window will have all code of Matlab program, and finally
- 4- The GUI application that will start as you click on Run tool  and save the figure to your folder with extension (.fig).

(Note: for more details on the GUI tools and options, use Matlab Help, or see other references about the subject!).

Ex1: Recall the problem of computing the factorial of any integer number ($N!$). Design a GUI to compute the factorial of any integer number like below:

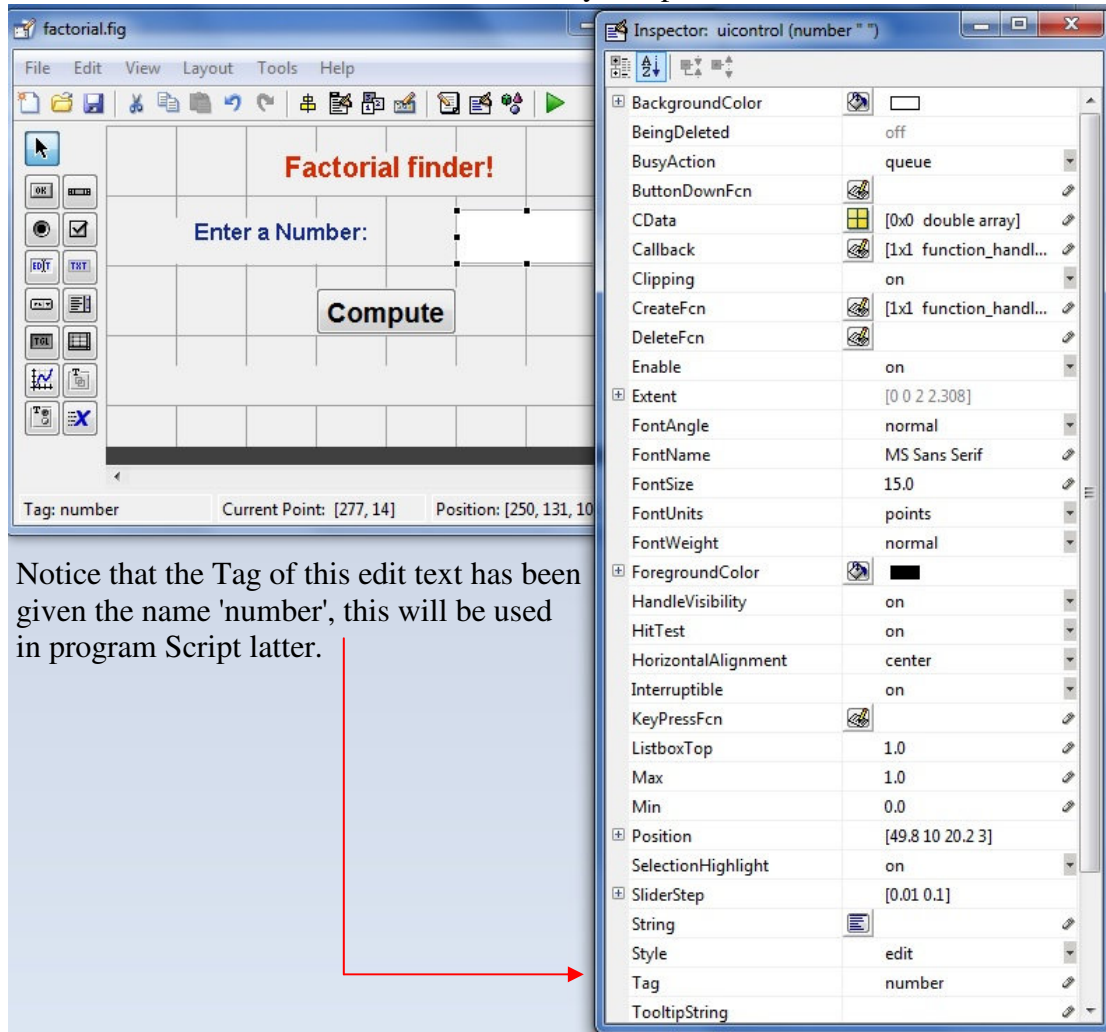


Solution:



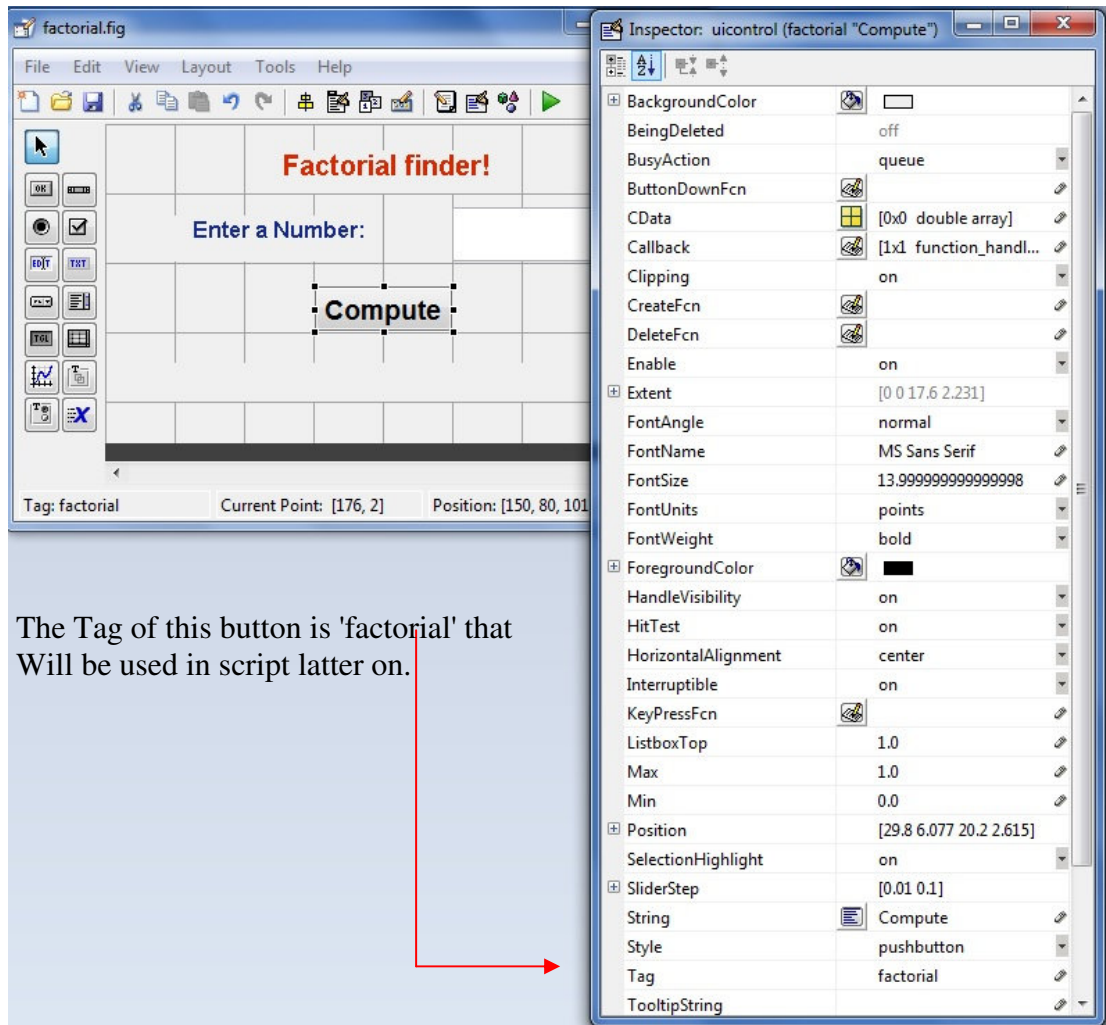
The same is for the second Static text ('Enter a Number :')...

While for the Edit text that to allow user entry, the procedure is like below.

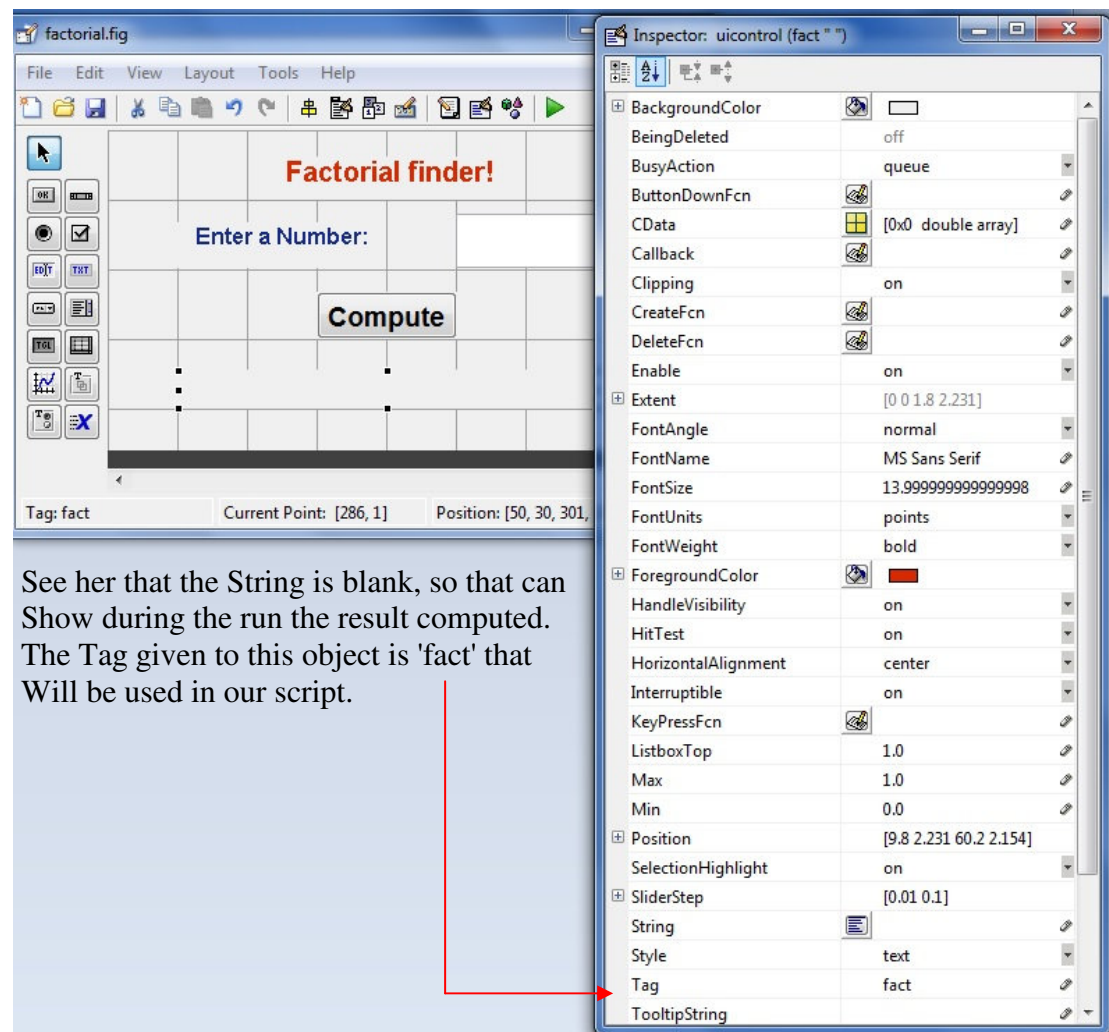


Notice that the Tag of this edit text has been given the name 'number', this will be used in program Script latter.

The most important object or component in the design is the Push button; in which in this example has been given the title 'Compute' to give the Factorial result.



Finally, the result after hitting the Compute button is a Static edit as illustrated:



Now, the steps of writing the program script of this design:

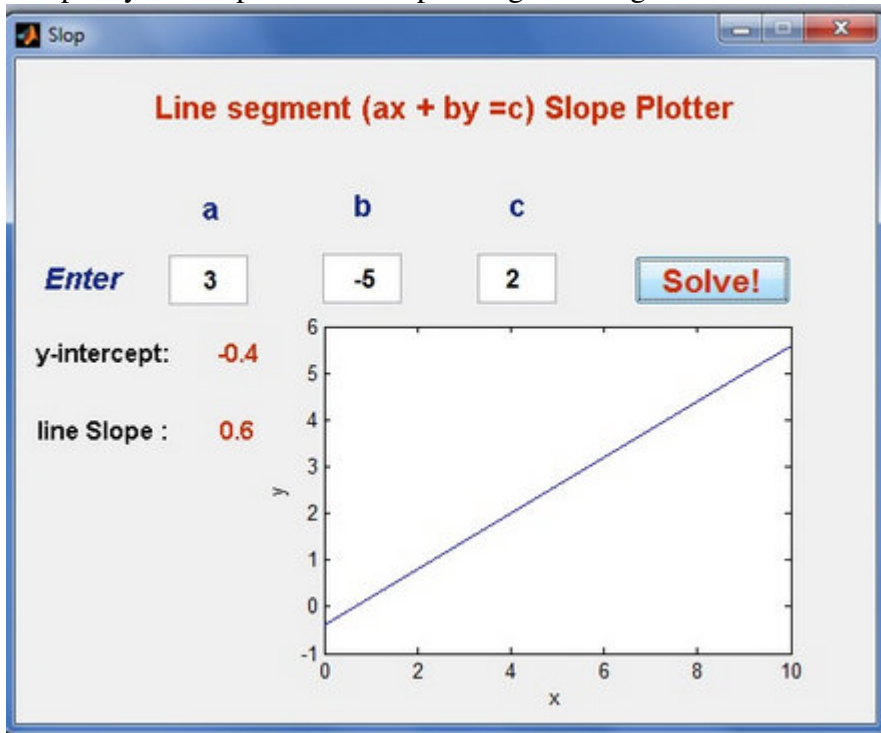
```

% --- Executes on button press in factorial.
function factorial_Callback(hObject, eventdata, handles)
% hObject      handle to factorial (see GCBO)
% eventdata    reserved - to be defined in a future version
of MATLAB
% handles      structure with handles and user data (see
GUIDATA)
n=str2num(get(hObject,'string'));
f=1;
for i=1:n
    f=f*i;
end
ff=num2str(f);
set(handles.fact,'string',ff);

```

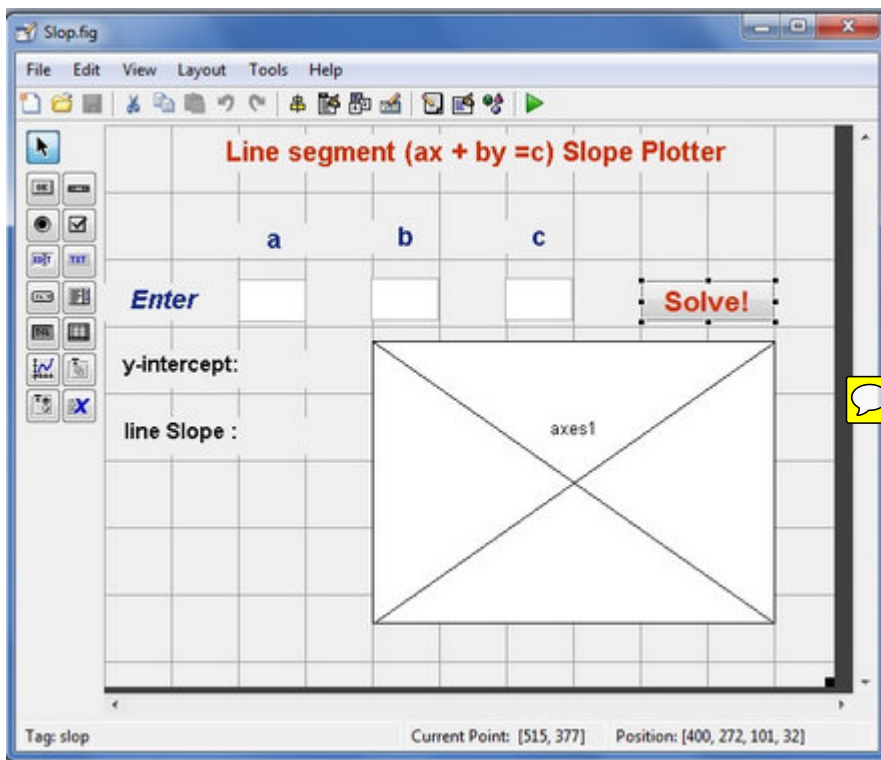
The function name is 'factorial_Callback', the (str2num) is used to turn the string into numeric value so that computing will occur after getting the variable called 'number' that the user will enter, then the algorithm for computing the factorial, and finally, turn the result (f) back into string called (ff) which will give the result.

Ex2: Write the script necessary to plot any line segment ($ax + by = c$), and also compute y-intercept and line slope using the design below:



Solution:

The design can be similar to the following:

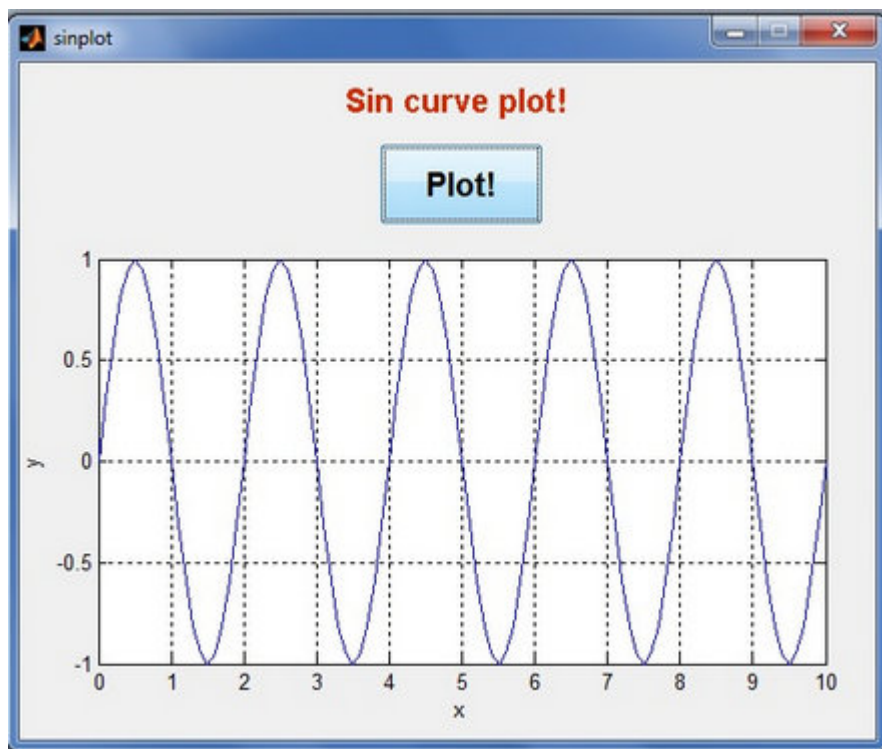


While the script necessary to execute the program is:

```
% --- Executes on button press in slop.
function slop_Callback(hObject, eventdata, handles)
% hObject    handle to slop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=str2num(get(handles.a , 'string'));
b=str2num(get(handles.b , 'string'));
c=str2num(get(handles.c , 'string'));
int=num2str(c/b);
slope=num2str(-a/b);
set(handles.int, 'string', int);
set(handles.slope, 'string', slope);
x=0:10;
y=c/b -a/b*x;
axes(handles.axes1);
plot(x,y);
xlabel('x');
ylabel('y');
guidata(hObject, handles);
```

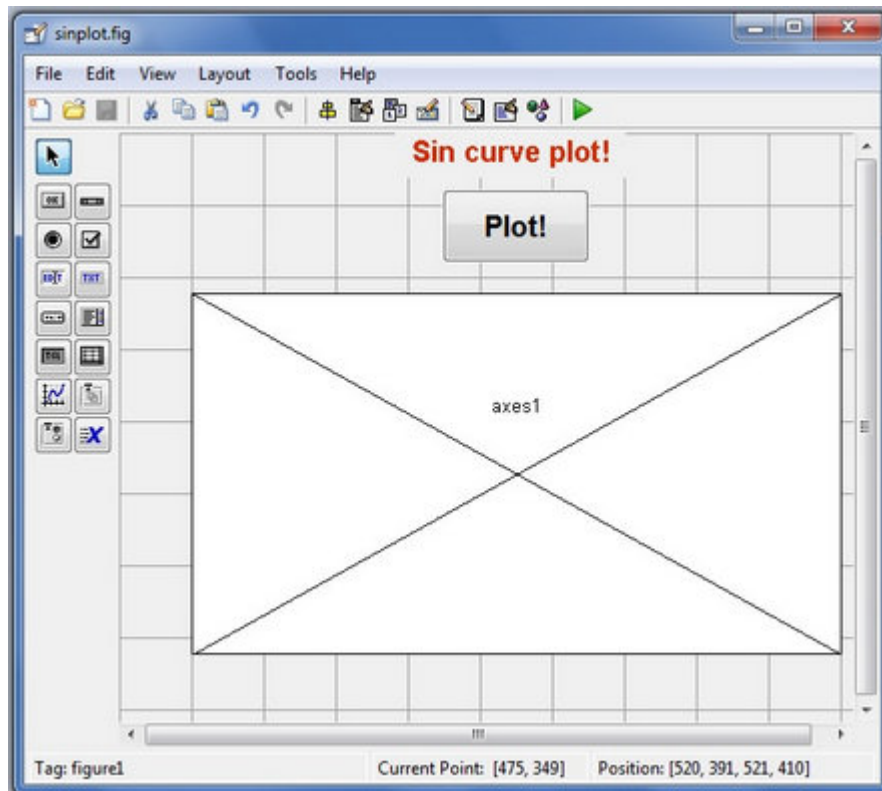
Notice that the command **axes** is used for enabling the plot within the frame called 'axes1' in the tag of the component inspector properties, 'Solve!' button should have tag name same as the function name 'slop'.

Ex3: Design the following and write Matlab script to plot sine curve?



Solution:

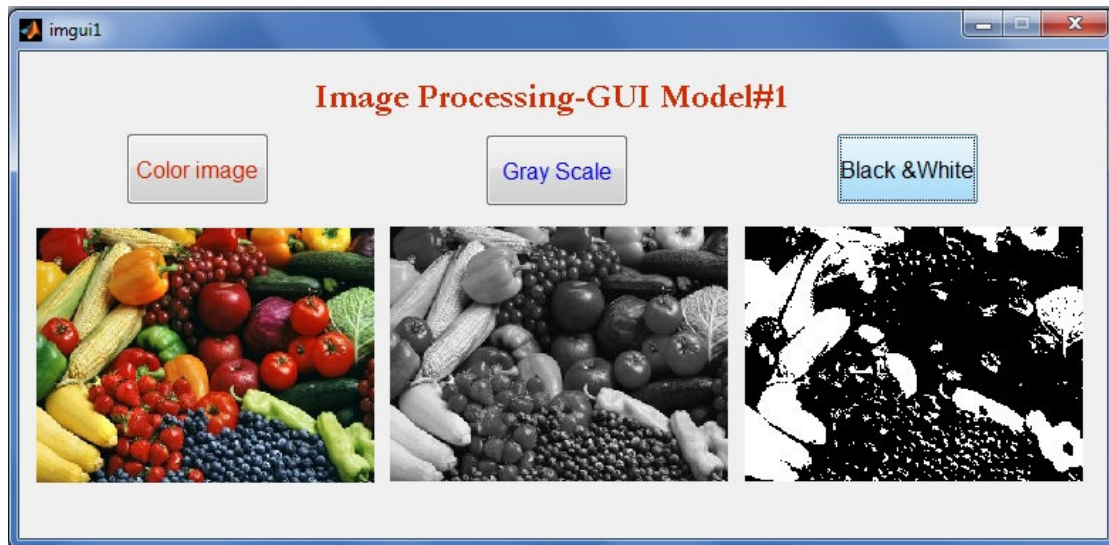
First, design the following layout:



The 'Plot!' button is to be used to plot the sin curve for (x) values between (0-10) times pi, while the axes1 is the graph area that will show the sin curve. Matlab script to do the job can be as follows:

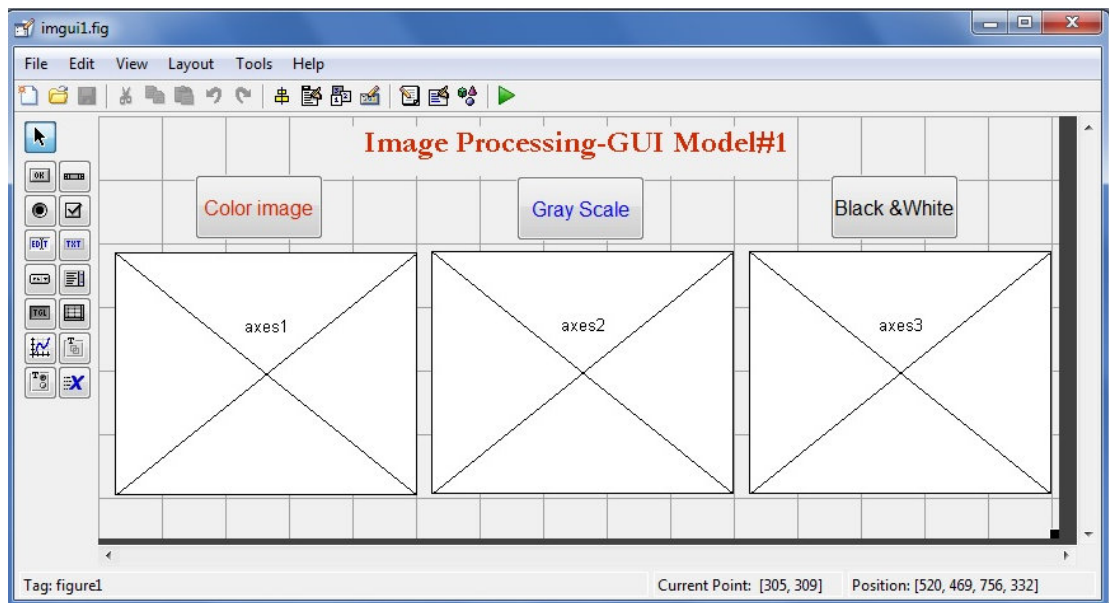
```
% --- Executes on button press in sinplot.
function sinplot_Callback(hObject, eventdata, handles)
% hObject      handle to sinplot (see GCBO)
% eventdata    reserved - to be defined in a future version of
MATLAB
% handles      structure with handles and user data (see GUIDATA)
x=0:0.1:10;
y=sin(x*pi);
axes(handles.axes1);
plot(x,y,'b')
xlabel('x');
ylabel('y');
% if you want to display grid, then use:
grid on
```

Ex(4): Using the techniques of image processing, Design the following layout which displays original color image, a Gray scale and Black and white images?



Solution:

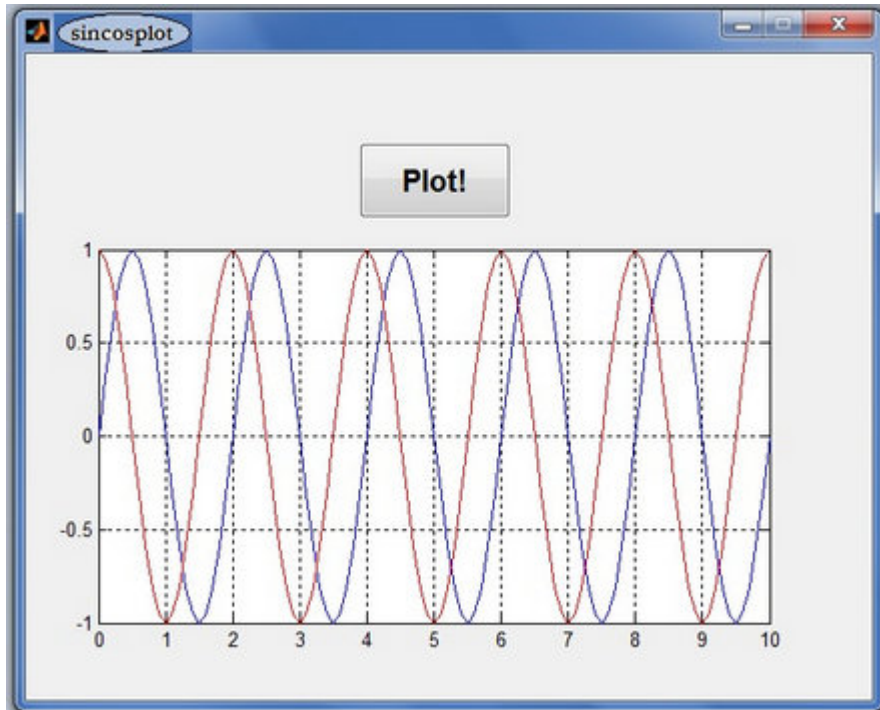
The model design should be like this below, with one note that when working with axes areas, no need for XTicks and YTicks, and therefore delete there data.



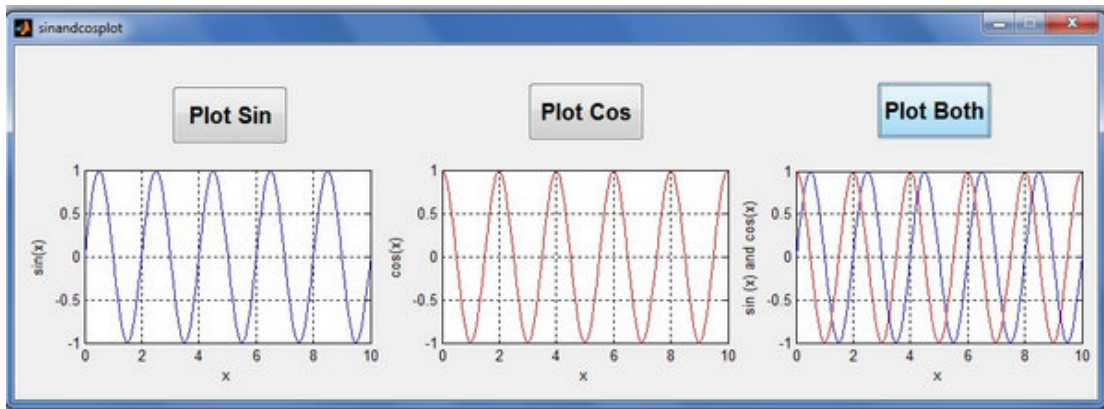
The script for three push buttons is:

```
function colors_Callback(hObject, eventdata, handles)
SS=imread('Fruits.jpg');
axes(handles.axes1);
imshow(SS);
function Gray_Callback(hObject, eventdata, handles)
SS=imread('Fruits.jpg');
GS=rgb2gray(SS);
axes(handles.axes2);
imshow(GS);
function BNW_Callback(hObject, eventdata, handles)
SS=imread('Fruits.jpg');
GS=rgb2gray(SS);
BW=im2BW(GS);
axes(handles.axes3);
imshow(BW);
```

Unsolved Exercise(1): write necessary Matlab script to run the following sin and cos plotting on the same graph:

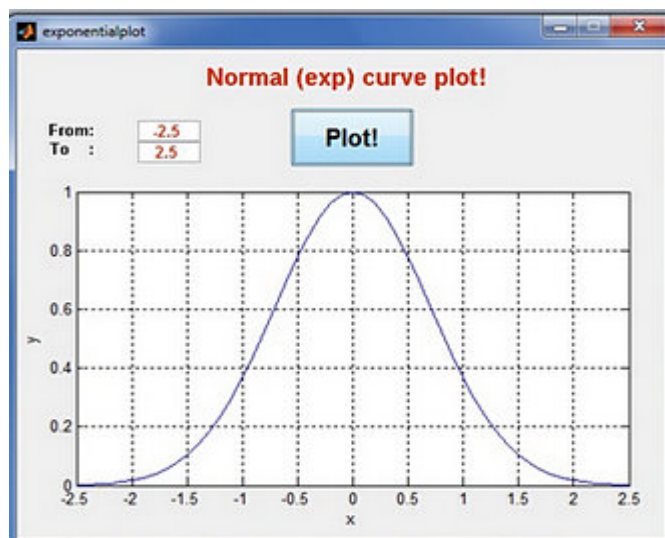


Unsolved Exercise(2): write Matlab script to run the following, with Three buttons: the first when press will plot sin curve, the second when pressed plots cos curve, while the third plots both curves when pressed?

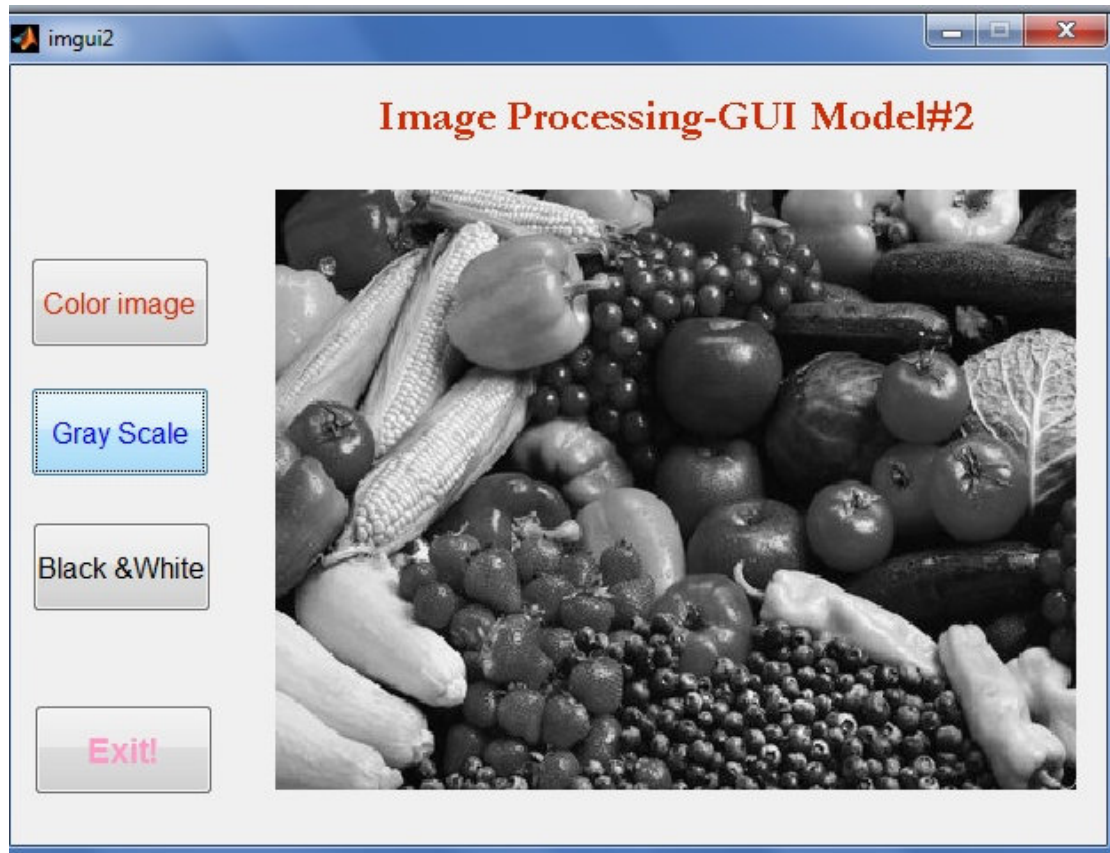


Unsolved Exercise(3):

Write Matlab script for plotting the Exponential (Normal) curve for any start and end values using the design beside.



Unsolved Exercise 4): Using the techniques of image processing, Design the following layout which displays original color image, a Gray scale and Black and white images on the same axes area depending on button pushed. The fourth button is to Exit from program as well as from Matlab?



Lecture edited and prepared
By
Assist. Professor/Emad Jihad
2013-2014

MATLAB® & Simulink®

Chapter 12

Introduction to SIMULINK!

**MATLAB®
& SIMULINK®**

Assist. Professor Emad Jihad

 **MathWorks®**

Simulink

Simulink, developed by MathWorks, is a data flow graphical programming language tool for **modeling, simulating** and **analyzing multidomain dynamic systems**.

Its primary interface is

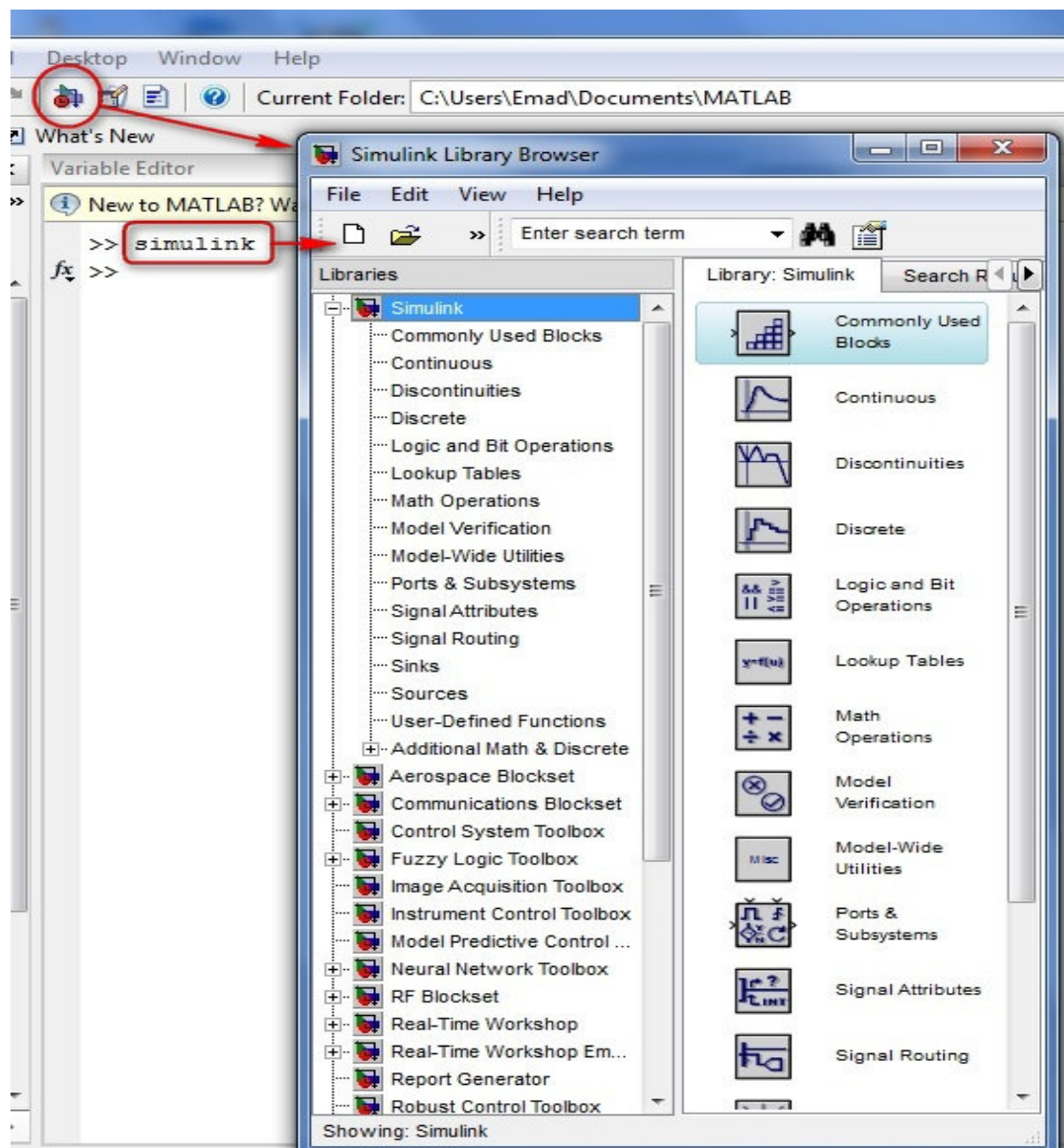
- 1- a **graphical block diagramming tool** and
- 2- a **customizable set of block libraries**.

It offers tight integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it.

Simulink is widely used in control theory and digital signal processing for multidomain simulation and Model-Based Design...

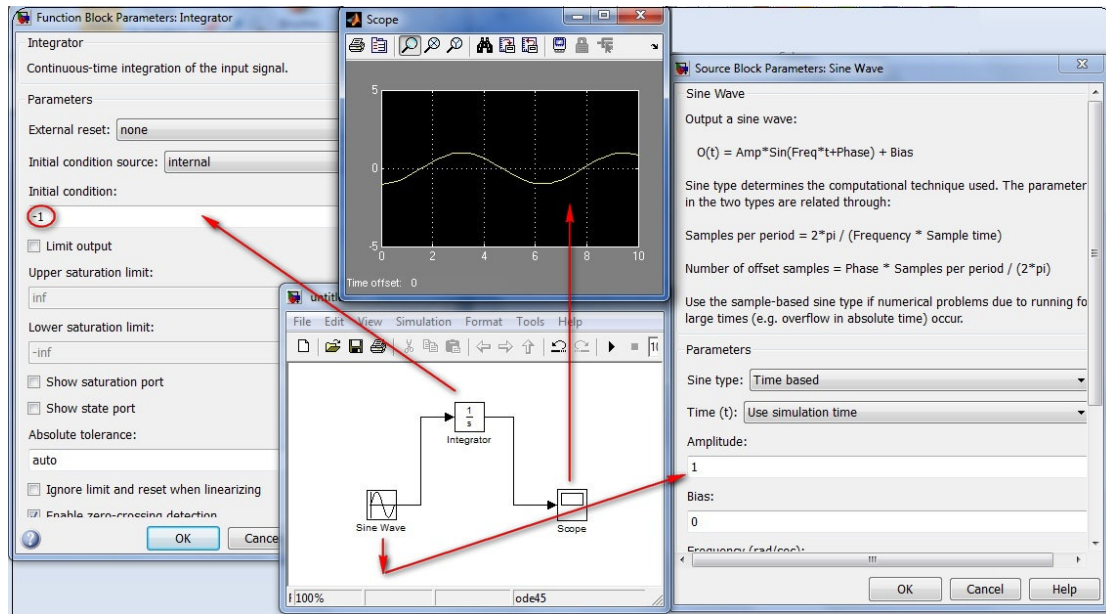
In this lecture, only a simple introduction to Simulink is considered, just to give an idea about this advanced technique in Matlab.

Activating Simulink is either by its icon or by writing its name in command window.



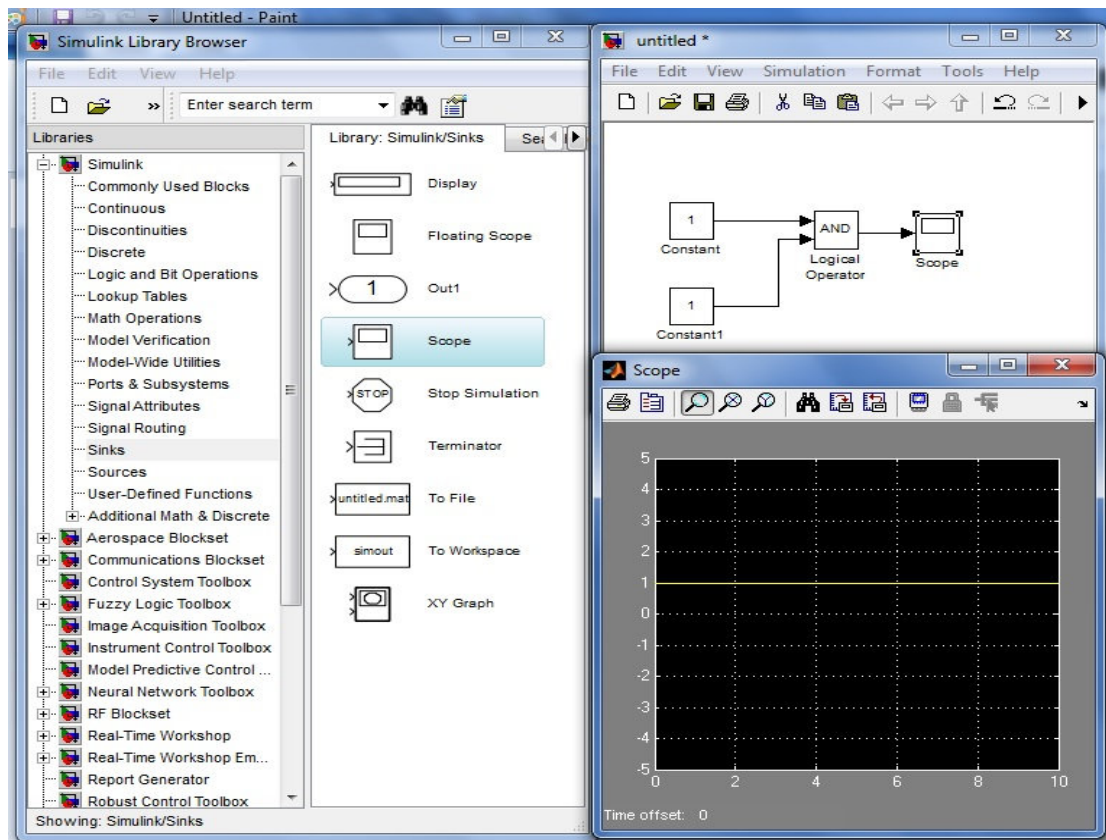
From file command on the menu bar, select (New) for a new Model or (Open) to open an existing one.
 As can be seen that the Simulink Library browser shows many block diagramming tools that the user can select from and making the model.

Ex(1): Make a new model that has blocks represent sin wave, integrator, and a Scope.



First, a new blank model is selected, then graphical blocks of components are selected from the different libraries of Simulink, then necessary values given after connecting all block in the model, and finally double click on scope and get the output.

Ex(2): Use a scope to prove that the logical (AND) is correct?



For all values used to test any condition, these values may either be entered as seen before in Ex(1) or input is possible through the command window or make a program script to do it faster after giving a variable name to define those variables.

Ex(3): Write Matlab script to input variables to be used in a model contains two logical variables as input to a Logical And, and another two input variables to an Add block, then find the output using a scope for each.

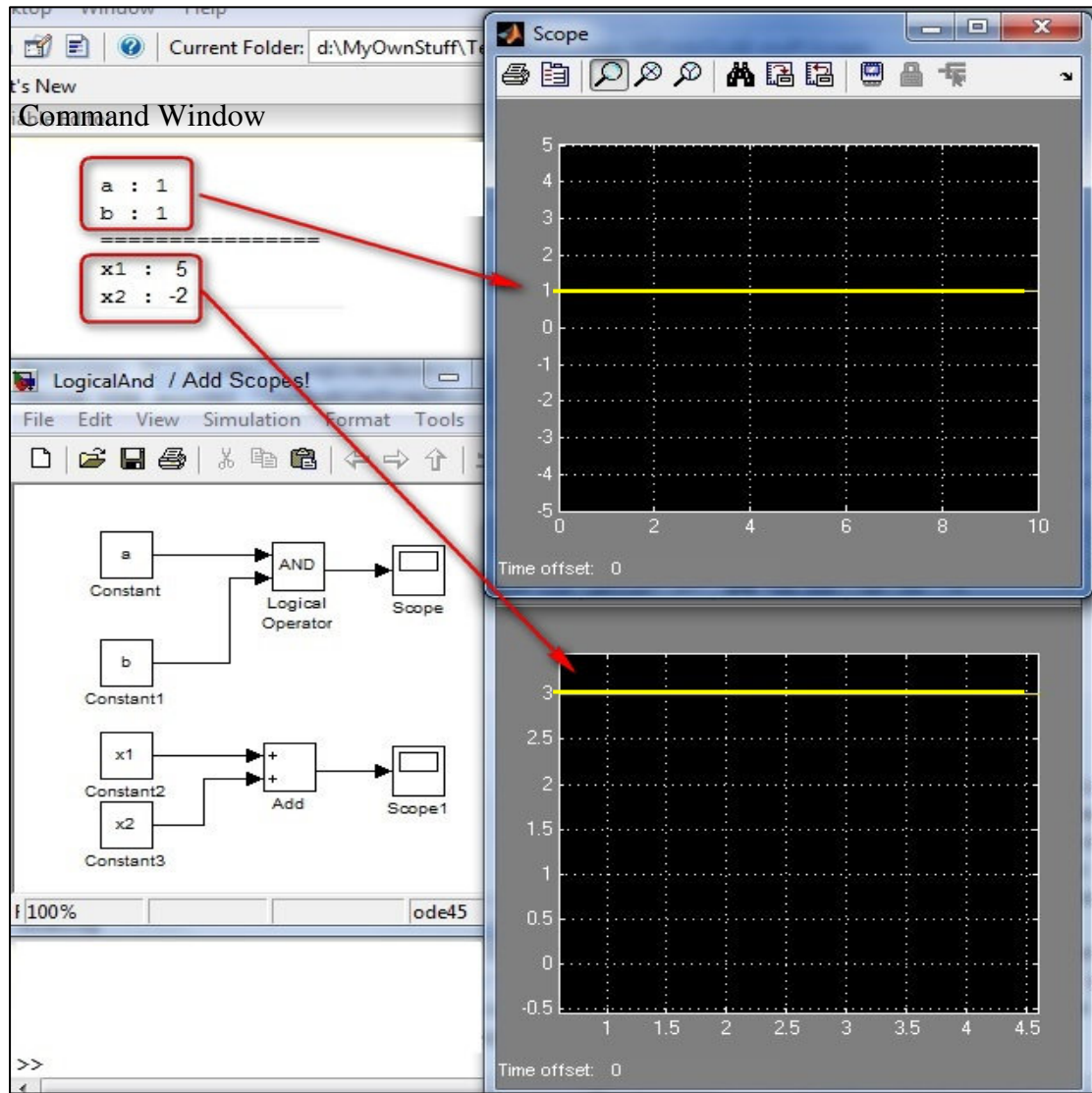
First, changing the variable names of input within block properties, then writing the script to define the interactive input that to be entered through command window:

```

% input variables for Logical And/ Add blocks
model
input('a : ');
a=ans;
input('b : ');
b=ans;
display('-----')
input('x1 : ');
x1=ans;
input('x2 : ');
x2=ans;

```

...Matlab script.



Ex(4): Design a model for an (Accelerometer) using Transfer function with Numerator coefficients [2 2 6], and Denominator coefficients of [1 3 2], receiving input from step block with step time (0), initial time (0) and final value of (1).

Solution:

The model should be like the following after Autoscale is chosen for scope...

The screenshot displays the MATLAB/Simulink environment. On the left, the 'Library' browser shows the 'Transfer Fcn' block under 'Continuous' blocks. The main workspace shows a Simulink model with a 'Step' block connected to a 'Transfer Fcn' block, which is then connected to a 'Scope' block. The 'Transfer Fcn' block is configured with numerator coefficients [2 2 6] and denominator coefficients [1 3 2].

The 'Scope' block shows the resulting plot of the system's response. The plot displays a step response that starts at 1, dips to a minimum of approximately 1.2 at t=0.5, and then rises to a steady-state value of 3. The x-axis represents time from 0 to 7, and the y-axis represents the output from 1 to 3.

Transfer Fcn
Model linear system by transfer function

Library
Continuous

Description

$$\frac{1}{s+1}$$

The Transfer Fcn block models a linear system by a transfer function of the Laplace-domain variable s . The block can model single-input single-output (SISO) and single-input multiple output (SIMO) systems.

Conditions for Using This Block

The Transfer Fcn block assumes the following conditions:

- The transfer function has the form

$$H(s) = \frac{y(s)}{u(s)} = \frac{num(s)}{den(s)} = \frac{num(1)s^{nn-1} + num(2)s^{nn-2} + \dots + num(nn)}{den(1)s^{nd-1} + den(2)s^{nd-2} + \dots + den(nd)},$$
 where u and y are the system input and outputs, respectively. nn and nd are the number of numerator and denominator coefficients, respectively. $num(s)$ and $den(s)$ are the numerator and denominator in descending powers of s .
- The order of the denominator must be greater than or equal to the order of the numerator.
- For a multiple-output system, all transfer functions have the same order.

Modeling a Single-Output System

For a single-output system, the input and output of the block are:

- Enter a vector for the numerator coefficients of the transfer function.
- Enter a vector for the denominator coefficients of the transfer function.

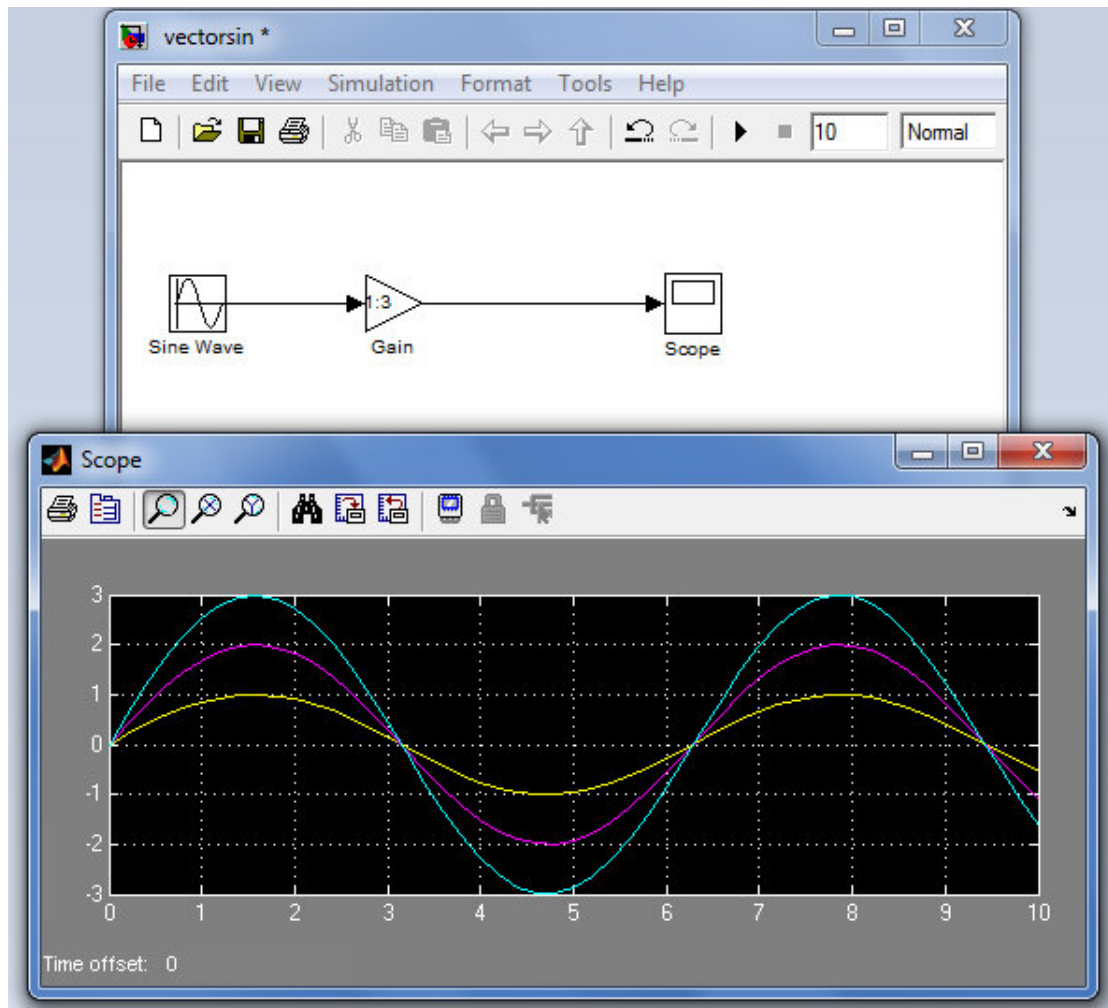
Modeling a Multiple-Output System

For a multiple-output system, the block input is a scalar and the block output is a vector. To model this system:

Ex(5): Display the sin wave for 3 different vector elements?

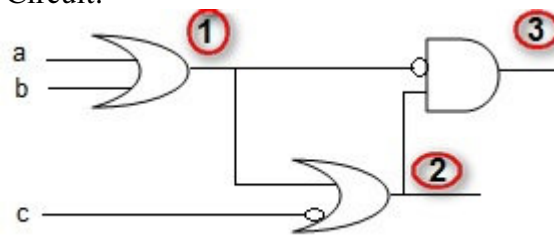
Solution:

Instead of choosing one Amplitude value for the sine wave block, we enter 1:3 for example, and the design and output should be like this:



Ex(6): Design Matlab Simulink model and find the output signal for each of the gates in the following Logic Circuit:

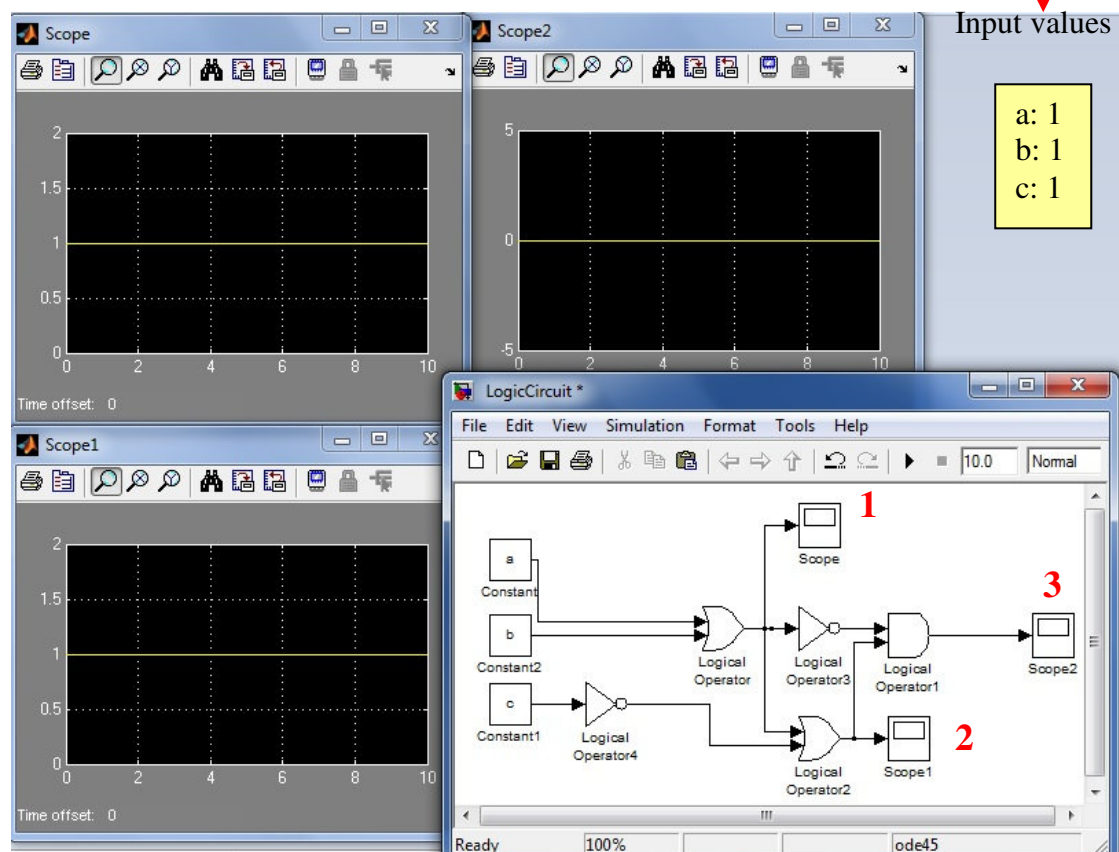
Solution:



Solution:

To make the Model work for all input possible values, first write a program script:

```
% input digital logic variables
a,b and c
input('a: ');
a=ans;
input('b: ');
b=ans;
input('c: ');
c=ans;
display('=====')
```



After executing the program, with input values (1,1,1) for example for (a, b and c), Notice that the Output will be (1, 1, 0) for gate scopes (1,2 and 3) respectively. Notice also, that in order to display normal gate layout instead of block layout, distinctive is to be chosen for Icon shape.

(Try to change the input values and test the output signals, also you may change the circuit for more practicing).

Ex(7): For the simple vibration problem using 2nd. Order differential equation :

$$x'' = - (k/m).x - (c/m).x'$$

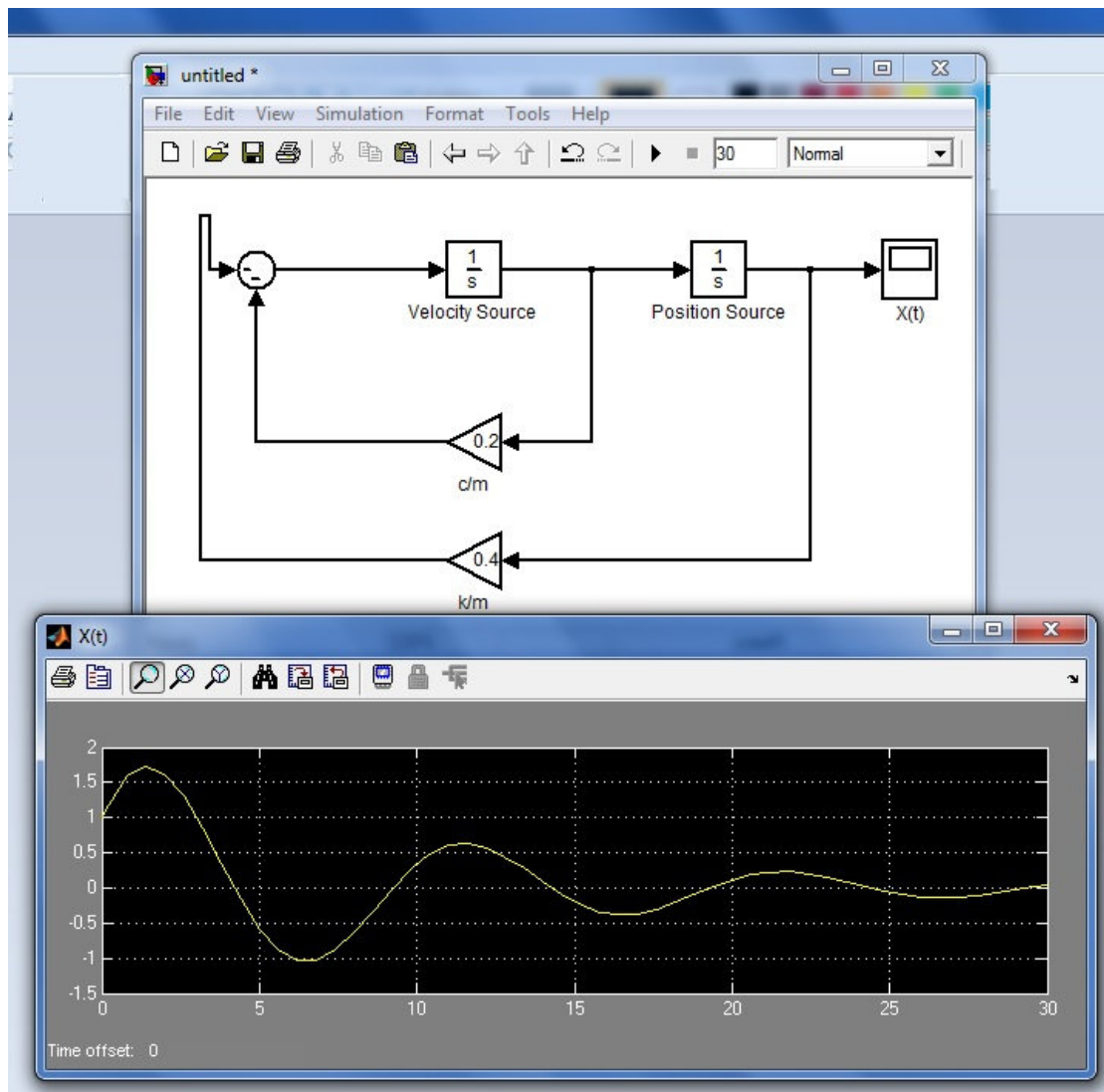
Design a Simulink model and display the output $x(t)$, providing that:

Initial velocity source $(c/m) = 1$, initial position source $(k/m) = 1$.

Set the value of $(c/m) = 0.2$ and $(k/m) = 0.4$...

Time = 30 s.

Solution:



Lecture prepared and edited by
Assistant Professor/ Emad Jihad
Department of Mechatronics, Technical Engineering College – Baghdad
2013-2014