German University in Cairo – GUC
Information Engineering & Technology
Media Engineering & Technology

Signals & Systems Lab.- Manual (1)

A brief overview of:

Introduction to MATLAB

By: Eng. Moustafa Adly

Contents

1. The MATLAB Desktop

1.1. Work space Management

Instruction	Definition
who	Displays variables in the workspace
whos	Displays variables in the workspace with more details
clear	Deletes MATLAB workspace (all variables)
clear('y')	Deletes the variable named y
clear y	Deletes the variable named y (another way)
save *file_name*	Saves workspace variables to a storage device (e.g. HDD)
load *file_name*	Loads workspace variables from a storage device (e.g. HDD)

1.2. Command Window Management

clc	Clears command window
x=1;	';' suppresses echo
UP ↑ & DOWN ↓ arrows	Recall previously executed commands

1.3. Command History Management

Double clicking any command in the command history will execute it again.

2. Variables in MATLAB

Identification	Variable type
A=34	Integer
X=[1 2 3 4]	Vector
Y=[1 2 3; 4 5 6; 7 8 9];	Matrix
S='hello'	String
R=3+2i or R=3+2j	Complex number (both declarations are equivalent)
5*3/9+12	The result will be stored in a variable called "ans"
inf (1/0)	Infinity
pi	3.14 or π
eps	Very small number; tends to zero

Note: MATLAB is case sensitive. So, x=3 doesn't mean that X=3 as they aren't equivalent.

3. Mathematical Functions

3.1. Complex numbers

Function	Output
s=3+4j or s=3+4i	Declaring a complex number s
real(s)	[3] real part of s
imag(s)	[4] imaginary part of s
abs(s)	[5] absolute value of s : $\sqrt{(3)^2+(4)^2}$ or the non-negative value of any number s
angle(s)	[53.1301] radian angle of s : $\tan^{-1}(4/3)$
complex(a,b)	[a+jb] results in a complex number of a, b
conj(s)	[3-4j] conjugate of s (real-j*imag.)

3.2. Trigonometric Functions

Y=sin(x)	Y=asin(x)	Y=sinh(x)	Y=asinh(x)
Y=cos(x)	Y=acos(x)	Y=cosh(x)	Y=acosh(x)
Y=tan(x)	Y=atan(x)	Y=tanh(x)	Y=atanh(x)
Y=sec(x)	Y=asec(x)	Y=sech(x)	Y=asech(x)
Y=csc(x)	Y=acsc(x)	Y=csch(x)	Y=acsch(x)
Y=cot(x)	Y=acot(x)	Y=coth(x)	Y=acoth(x)

Adding 'a' before function name gives the inverse function of the original one.
Adding 'h' after function name gives the hyperbolic function of the original one.
For inverse hyperbolic add both 'a' at the front and 'h' at the end of the function name.

3.3. Exponential Functions

Function	Example	
x^n	x^n :	3^4=81
exp(x)	Exponential function:	exp(5)=148.41
log10(x)	Log(x)/log(10) [base 10 logarithm]:	log10(100)=2
log2(x)	Log(x)/log(2) [base 2 logarithm]:	log2(64)=6
log(x)	Ln (natural logarithm):	log(4)=ln(4)=1.39
pow2(x)	2^x:	pow2(3)=2^3=8
Nextpow2(x)	Produces n where 2^n≥x:	nextpow2(33)=6
sqrt(x)	$\sqrt{x}$:	sqrt(25)=5
factorial(x)	x! the factorial of x such that x≥0 and integer:	factorial(3)=6

4. Vectors or Arrays

4.1. Declaring vectors

X=[1 5 9 -3 5]	X=[*element1 element2 element3 …….*]
Y=0:0.1:5	(start:step:end)
Z=linspace(0,10,11)	(start, end, no. of points) linear space
Z=logspace(0,10,11)	(start, end, no. of points) *[10^0 10^1 . . . 10^{10}]*

4.2. Operations on vectors

x(7)	Displays 7th element of the vector named x
x(5:9)	Displays 5th to 9th elements of x
x(6:end) ≡ x(6:length(x))	Displays elements from the 6th to the last one of x
x(1:1:6)	Displays elements 1,2,3,4,5, and 6 (step one)
x(1:2:7)	Displays elements 1,3,5, and 7 (step two)
x(8:-3:1)	Display elements 8,5, and 2 (step down by three)
x([1 2 4 9])	Display elements 1st,2nd,4th, and 9th elements
x'	The transpose of x, with conjugate if x is complex
x.'	The transpose of x without conjugation at all
[r,c]=size(x)	Returns: r = number of rows of the vector x c = number of columns of the vector x
sum(x)	Adds all elements of the vector x
mean(x)	The mean value of the vector x ≡ sum(x)/length(x)
length(x)	Number of elements in the vector x
find(x==2)	Returns the locations of the elements in x which are equal to 2
P=find(x>4)	Returns the locations of the elements in x which are greater than 4 in a vector P of zeros and ones (T/F)
G=x(P)	G is formed of the elements in x that are greater than 4
P=find(x>4 & x<10)	Returns the locations of the elements in x which are greater than 4 ***and*** in the same time are less than 10 in a vector P
P=find(x==5 | x>=8)	Returns the locations of the elements in x which are equal to 5 ***or*** are greater than or equal to 8 in a vector P
x==5	Asks MATLAB if x equals 5 or not. Answer is 1 if true and 0 if false.
Sol=x==3	The same as (x==3) but the answer (0/1) is stored in Sol.
y=x^2	Squaring the vector x; will lead to error due to dimensions
y=x.^2	Squaring the elements of the vector x; no care for dimensions.

4.3. Plotting two vectors

To plot any two vectors, there are two techniques:

First technique:

You should be aware of the contents of one of the two vectors at least and the relation of the other one to the known vector. An example of that is: plotting the sinusoidal function; you may want to plot Y=sin(X), so you should be aware by either X or Y and then use the relation between them (sin or asin according to the known vector) to plot the relationship between them.

If X is known: Define X by any method of defining vectors, indicate the forward relation of Y to X , which is the sin function, and then ask MATLAB to plot the two vectors.

Note that the computing is discrete, so you can't say that X is [-2π to 2π] but you should indicate the step of calculation. Also, note that minimizing the step size will lead to very smooth curve for the relationship and vice versa.

On the MATLAB command widow, write the following instructions:

```
>> X=[-2*pi:0.1:2*pi];
>> Y=sin(X);
>> plot(X,Y)
```

Here, we indicated that the step size is 0.1, but you may change it once to a greater value and another to a smaller value and note the difference in the curve of the relationship between X and Y in each case.

If Y is known: Define Y by any method of defining vectors, indicate the backward relation of X to Y , which is the arcsin function, and then ask MATLAB to plot the two vectors.

On the MATLAB command widow, write the following instructions:

```
>> Y=[-1:0.01:1];
>> X=asin(Y);
>> plot(X,Y)
```

Also, change the step size (0.01) once to a greater value and another to a smaller value and note the difference in the curve of the relationship between X and Y in each case.

Second technique:

In this one, you don't know a specific relationship between the vectors.
For example, if you would like to plot the average temperature of the months of the year to their order according to the following table:

MONTH	1	2	3	4	5	6	7	8	9	10	11	12
Temperature	17	19	22	24	27	30	34	35	31	26	23	20

Of course, you don't have a specific relation between the month order and the average temperature. So, you should use this way for plotting the two vectors.

On the MATLAB command widow, write the following instructions:

```
>> Month=[1:1:12];
>> Temperature=[ 17 19 22 24 27 30 34 35 31 26 23 20];
>> plot(Month,Temperature)
%% try : >> stem(Month,Temperature)
```

Note the following:

The plot instruction is used to draw the relationship between any two vectors such that the first input argument of the instruction is represented on the X-axis while the second input argument is represented on the Y-axis.

There are many formats for the **PLOT** instruction, check them by typing >>***help plot*** on the MATLAB command window.

Also, there are other instructions for plotting such as the **STEM** instruction. So, try typing it instead of the plot instruction in the previous examples. Furthermore, check the formats of the STEM instruction by typing >>***help stem*** on the MATLAB command window.

In the example of temperature vs. month order, you may get a continuous curve that describes the relation between vectors by finding a function between them using what is called *Curve Fitting* which will be discussed later.

More details will be illustrated in the Plotting section of this manual.

4.4. Polynomials

Sometimes you need to write a polynomial, find its roots, multiply it by another one, divide it by another polynomial, differentiate it, integrate it, substitute by a value in it, put it in partial fractions form, or fit it to get a curve of specific order. Make full use of the following table for that purpose:

P=[1 3 4 4 6]	$P = x^4 + 3x^3 + 4x^2 + 4x + 6$
Q=[1 6 2]	$Q = x^2 + 6x + 2$
C=poly([1 2 3])	$C = (x-1)(x-2)(x-3)$
conv(P,Q)	Multiply P*Q
[s,r]=deconv(P,Q)	Divide P/Q, where P=s+r/Q
roots(P)	Find the roots of P (put P=0 → get x)
polyder(P)	Derivative of P
polyval(P,3)	Substitute in P by x=3
polyint(P,2)	Integrate P with constant of integration =2
polyfit(x,y,n)	Fit x to y by a curve of degree n
[z,p,k]=residue(Q,P)	Partial fractions of Q/P: z=gains of the partial fractions terms (or zeros) p=poles of the partial fractions terms k=free term of the division

Example:

Let $F(x) = x^4 - x^3 - 7x^2 + x + 6$

and $G(x) = x^2 + 5x + 6$

It's required to:

i. Calculate the value of F(3) and G(-1).
ii. Find the roots of F(x) and G(x).
iii. Find F(x)*G(x).
iv. Find F(x)/G(x).
v. Get $\frac{dF(x)}{dx}$ and $\frac{dG(x)}{dx}$.
vi. Get $\int F(x)dx$ and $\int G(x)dx$ setting the constant of integration to be equal to 2.
vii. Find the partial fractions of the ratio $\frac{G(x)}{F(x)}$.

Solution:

Using simple calculus:

i. Substitute by x=3 in F(x) to get F(3) : $F(3)=3^4-3^3-7*3^2+3+6=0$
Substitute by x=-1 in G(x) to get G(-1) : $G(-1)=(-1)^2+5*(-1)+6=2$

ii. Solve the equation: $F(x)=x^4-x^3-7x^2+x+6=0$ to get the roots of F(x) simply, you can write it as: $(x+1)(x-1)(x+2)(x-3)=0$ $\rightarrow$ the roots are x=-1,1,-2,3.
Also, solve the equation $G(x)=x^2+5x+6=0$ to get the roots of G(x) Simply, you can write it as: $G(x)=(x+2)(x+3)=0$ $\rightarrow$ the roots are x=-2,-3.

iii. $$\begin{aligned}F(x)*G(x)&=(x^2+5x+6)(x^4-x^3-7x^2+x+6)\\&=x^6+4x^5-6x^4-40x^3-31x^2+36x+36\end{aligned}$$

iv. Using long division we get: $$\frac{F(x)}{G(x)}=(x^2-6x+27)+\frac{-48x-96}{x^2+5x+6}$$

v. $$\frac{dF(x)}{dx}=4x^3-3x^2-14x+1 \qquad \& \qquad \frac{dG(x)}{dx}=2x+5$$

vi. $$\begin{aligned}\int F(x)dx&=\frac{x^5}{5}-\frac{x^4}{4}-\frac{7x^3}{3}+\frac{x^2}{2}+6x+const.\\&=\frac{x^5}{5}-\frac{x^4}{4}-\frac{7x^3}{3}+\frac{x^2}{2}+6x+2\end{aligned}$$

$$\int G(x)dx=\frac{x^3}{3}+\frac{5x^2}{2}+6x+const.=\frac{x^3}{3}+\frac{5x^2}{2}+6x+2$$

vii. $$\frac{G(x)}{F(x)}=\frac{x^2+5x+6}{x^4-x^3-7x^2+x+6}=\frac{(x+2)(x+3)}{(x+1)(x-3)(x+2)(x-1)}=\frac{(\frac{1}{4})}{x+1}+\frac{(\frac{3}{4})}{x-3}+\frac{(-1)}{x-1}$$

Check all these results using MATLAB as follows:

```
>>F=[1 -1 -7 1 6];
>>G=[1 5 6];
>>polyval(F,3)
>>polyval(G,-1)
>>roots(F)
>>roots(G)
>>conv(F,G)
>>[s,r]=deconv(F,G)
>>polyder(F)
>>polyder(G)
>>polyint(F,2)
>>polyint(G,2)
>>[z,p,k]=residue(G,F)
```

5.Matrices

5.1. Declaring matrices

A=[1 5 9; -3 5 1; -4 2 7]	a=[*row1 elements ; row 2 elements ; ……. *]
B=zeros(3,4)	A 3*4 zeros matrix
C=ones(2,5)	A 2*5 ones matrix
D=eye(4)	A 4*4 identity (unity) matrix
E=rand(4)	A 4*4 random matrix ranging from 0 to 1
f=magic(4)	A 4*4 magic matrix (sum of any row = sum of any column)

5.2. Operations on matrices:

a(2,3)	Displays the element of the matrix named a, which is in row 2 and the column 3
a(2,3)=5	Set the element in row 2 and column 3 to be equal to 5
S=a(2,1:3)	Puts the elements of the 2nd row from the 1st to the 3rd column of the matrix a in a vector named S
X=[12 3 10]	Defines a 1*3 matrix called X
Y=[5 9 13]	Defines a 1*3 matrix called Y
Z=[X;Y]	The 1st row in Z is the vector X & 2nd row is the vector Y
W=[X' Y']	The 1st column in W is the transpose of the vector X & the 2nd column is the transpose of the vector Y [W≡3*2]
a'	The transpose of matrix a
inv(a)	The inverse of a
det(a)	The determinant of a
[r,c]=size(a)	Returns: r = number of rows of the matrix a c = number of columns of the matrix a
[u,v]=eig(a)	u≡eign vectors and the diagonal of v is the eign values for a
sum(a)	Sum of each column
prod(a)	Product of each column
B=a^2	Squaring the matrix a
B=a.^2	Squaring each element in the matrix a
a*b	Multiplying matrices, care for dimensions
a*b-b*a	Will the answer be zero? why?
a.*b	Multiplying each element in a by the one in b of the same location
a.*b-b.*a	Will the answer be zero? why?
a/b	a*inv(b)
a\b	Inv(a)*b
sort(a)	Sorting each column in a ascending, for vectors: sorting the vector ascending.
[q w]=sort(a)	q is the matrix a after sorting but w is the last position of the elements in a (before sorting)
rank(a)	Rank of the matrix a
[q w]=max(a)	q is a row containing the maximum element of each column while w is its position in the column in a
[q w]=min(a)	The same as the previous instruction but for minimum
D=diag(a)	d is a vector containing the elements of the diagonal of a
flipud(a)	Flips a up to down
fliplr(a)	Flips a left to right
rot90(a,*n*)	Rotates a counter-clockwise n times
reshape(a,*m*,*n*)	Resorts a into m rows & n columns such that m*n=size of a

5.3. Some applications

Solving Linear Equations:

Suppose that you would like to solve the following system of equations:

2X+Y-Z=6
X-Y-Z=-3
X+2Y-3Z=-9

In matrix form, you can write:

$$\begin{bmatrix} 2 & 1 & -1 \\ 1 & -1 & -1 \\ 1 & 2 & -3 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 6 \\ -3 \\ -9 \end{bmatrix} \longrightarrow \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 2 & 1 & -1 \\ 1 & -1 & -1 \\ 1 & 2 & -3 \end{bmatrix}^{-1} * \begin{bmatrix} 6 \\ -3 \\ -9 \end{bmatrix}$$

On MATLAB command window write:

```
>>a=[2 1 -1; 1 -1 -1; 1 2 -3];
>>b=[6;-3;-9];
>>solution=inv(a)*b                    % or solution=a\b
```

Curve Fitting

If you would like to draw a curve (of degree 2 for e.g.) to represent the relation between the month order and temperature (previously discussed), you should write on the MATLAB command window the following:

```
>>month=1:1:12;
>>temp=[17 19 22 24 27 30 34 31 26 23 20];
>>n=2;
>>p=polyfit(month,temp,n);      curve fitting between month and temp by degree n=2
>>x=1:0.1:12;
>>y=polyval(p,x);
>>plot(month,temp,'r',x,y,'g')
```

6. Plotting:

6.1. 2-D plotting:

First of all, let's define a vector x=[-π,π] and another two vectors y and z, where y=sin(x) and z=cos(x). Then, you can declare that to MATLAB as follws:

```
>> x=[-pi:pi/10:pi]
>>y=sin(x);
>>z=cos(x);
>>w=100*cos(x)
```

Now, you can use this table to access the PLOT instruction in the form you are desired in.

plot(x,y,'ro:',x,z,'b+-.')	Plotting sin and cos functions by two different line formats
legend('sin(\theta)','cos(\theta)')	Prints legend on the graph using symbolic theta θ
title('My graph')	Prints graph title
xlabel('\theta')	Prints x-axis label
ylabel('\function')	Prints y-axis label
grid on / grid off	Toggle grid on and off
hold on / hold off	The new plot doesn't replace the old one / the new plot replaces the old one
H=axis	Now H is a vector that contains the axis limits
axis([-1 1 -2 4])	New axis limits for x and y respectively
point=ginput(3)	Let the user click on three points on the graph and store their locations in a matrix called point (its size =3*2). Every row would have one point(x,y)
point =ginput	Let the user click any number of times until pressing ENTER and store the locations of the clicked points in a matrix called point.
plotyy(x,y,x,w)	Creates plot with 2 y-axes for different scaled functions.
subplot(235)	Creates a 2*3 graph and set attention to the fifth one
text(x1,y1,'this point')	Write the text 'this point' at point (x1,y1)
stairs(x,y)	Draws the relation between x and y in ladder steps
stem(x,y,'--')	Discretized form of the relation between x and y
bar(x,y)	Draws the relation in bars
bar3(x,y)	Draws the relation in 3-D bars

6.2. 3-D plotting:

On the MATLAB command window, write the following:

```
>>t=-4*pi:0.1:4*pi;
>>x=cos(t);
>>y=sin(t);
>>plot3(x,y,t)
```

7. Symbolic Math:

7.1. Declaration:

Until now, you couldn't deal with MATLAB using symbols such as differentiating the function $F(x) = x * \sin(x)$ with respect to x or integrating it, but you are used to deal with MATLAB using numerical quantities. In symbolic math, you will be able to do that. So, check the following table to use MATLAB in symbolic form.

syms x y z a b c d s t	Defines some symbols
F=5*sin(x)+x	Defining a symbolic function f(x)
G=sin(x)/(cos(y)+2)	Two variable-symbolic function g(x,y)
H=a*x-b*y+z	Five variable-symbolic function h(a,b,x,y,z)

7.2. Operations on symbolic math:

7.2.1. Solving & substitution

subs(f,2)	Getting f(2).
subs(g,x,3)	Getting g(3,y) – substitute by x=3 in g(x,y).
solve(f)	Solve f(x)=0 and get x as an expression of other parameters or return its value if f is of single variable.
solve(h,z)	Solve h=0 with respect to z → get z as an expression of other parameters or variables of h.
solve(h)	Solving for x as a default variable.
[x y]=solve('x+y=10','x-y=3')	Solve two linear simultaneous equations
[x y z]=solve('x+y+z=10', 'x+y-z=2','x-y+z=6')	Solve three linear simultaneous equations
Limit(f,x,inf)	Gets the limit for f when x tends to infinity

7.2.2. Differentiation

diff(f,n)	Differentiate function f with respect to its default variable n times
diff(g,x,n)	Partial derivative of function g with respect to x (∂g/∂x) n times
diff(g,y,n)	Partial derivative of function g with respect to y (∂g/∂y)n times

7.2.3. Integration

int(f)	Integration with respect to x (the default variable of f) and neglecting the constant of integration
int(g,x)	Integration of g with respect to x : $\int g(x,y)dx$
int(f,0,1)	Limited integration of f: $\int_0^1 f(x)dx$
int(f,a,b)	Limited by symbols integration: $\int_a^b f(x)dx$
int(g,x,1,2)	Limited integration of g with respect to x: $\int_1^2 g(x,y)dx$
int(g,y,a,b)	Limited integration of g with respect to y: $\int_a^b g(x,y)dy$

7.2.4. Laplace transform

It's very useful to use MATLAB for getting the Laplace and inverse Laplace transforms directly without mathematical derivations and calculations and loosing a lot of time using the symbolic math as follows.

syms t s	Defining t and s as a symbol
laplace(sin(t))	Getting the laplace transform of the function sin(t)
ilaplace($\frac{1}{s^3}$)	Getting the inverse laplace transform of the function $\frac{1}{s^3}$
Ztrans & iztrans	Gets the z-transform and inverse z-transform for discrete signals

7.2.5. Fourier transform

syms t w n	Defining t, n, and w as symbols
fourier(sin(2t))	Fourier transform of the function sin(2t)
ifourier(1)	Inverse fourier transform of 1
fft(x,n)	Discrete time fast fourier transform

7.2.6. Solving Differential Equations

Sometimes you are required to solve the following differential equation for example:

$$\ddot{y} + x\dot{y} + x^2 = 0$$

Under some initial conditions which are: $\dot{y}(0) = 1$ & $y(0) = 0$

So, you may use MATLAB for that purpose as follows:

Dsolve(‘ $D2y + Dy * x + x^2$ ’,’Dy(0)=1’,’y(0)=0’)	Solve a differential equation
Dsolve(‘Dy=x+y’,’Dx=2*x-y’)	Solving two simultaneous D.E.
Pretty(ans)	Gives you the form of the variable ans that can be written in your sheet

Note that:

δ(t): Delta Function	→	>> dirac(t)
u(t): Step Function	→	>> heaviside(t)

8. Programming conepts:

8.1. Logical Operators:

The famous logical operators are:

- *AND:* logical anding between any two variables.
  Example:
  ```
  >> a=1;
  >> b=0;
  >> c=and(a,b)
  ```
- *OR:* logical oring between any two variables.
  Example:
  ```
  >> a=0;
  >> b=1;
  >> c=or(a,b)
  ```
- *xor:* logical anding between any two variables.
  Example:
  ```
  >> a=1;
  >> b=0;
  >> c=xor(a,b)
  ```

The logical operators AND & OR can be represented by symbols directly as shown in the example below:

Example:
```
>>a=1;
>>b=0;
>> c=a&b              % Logical AND
>> d=a|b              % Logical OR
```

8.2. Conditional statements:

The most famous conditional statement in all programming languages is the *if-statement*, which usually has the form:

```
If condition
  ………………
  ………………      }  Actions to be done if true condition
  ………………
elseif  another_condition
  ………………
  ………………      }  Actions to be done if true another_condition
  ………………
else
  ………………
  ………………      }  Actions to be done if both condition and another_condition are false
  ………………
end
```

8.3. Loops:

Looping is very famous in all programming languages. The most common loops are the *for-loops* and the *while-loops*. Let's start by the *for-loop*:

```
Initialization for the counter
For condition_on_counter
  ………………
  ………………                          }  Actions to be done as long as true condition_on_counter
  ………………
  changing the value of the counter
end
```

It's very common for you not to make an infinite loop☺, but I should remind you not to do so. If happened by error: press **ctrl** while pressing **pause/break** to break it.

Now, what about the *while-loop*:

```
Initialization for a counter
while condition_on_the_counter
  ………………
  ………………                        }  Actions to be done as long as true condition_on_the_counter
 <<operations_on_the_counter>>
  ………………
end
```

8.4. Some useful instructions:

In this part, I will mention some useful instruction for the main programming purposes but I won't mention how to use them and their formats, so it is your turn to do so for your own sake using either MATLAB help or the Internet:

- input
- disp
- num2str
- str2num
- inline
- fminsearch
- fzero
- dirac
- heaviside

8.5. Functions in MATLAB:

Functions are well-known in all programming languages. Also they play the same role in MATLAB. You can easily create a function and save it and then you will be able to call it from the command window or the m-files. When you type **real(3+5*i)** in the command window, you are calling a pre-defined function of MATLAB that was built by the programmers of MATLAB. From now on, you will be able to create your own functions to fulfill your needs. The structure and steps of creating a function is illustrated in this section as follows:

1) Open a new M-file as follows: click File – New – M file.
2) Save the file from the file menu with the same name of your function as "myfunc"
3) Type the following text in the first line of the script m-file editor:

```
function  [out1,out2,………..] = myfunc (in1,in2,………….)
```

4) In1, in2,……… are the input arguments that should be passed to the function while calling it.
5) Out1, out2,…… are the output arguments that are calculated in the function and then returned to the location of calling the function.
6) Note that the function name must be the same as the name of its file that you used in step (2). Also, it should be a new name, i.e. not a reserved word for MATLAB, and must start by a letter not a number.
7) After that first line you can use any MATLAB instructions knowing that you have some input variables that should be used and some output variables that MUST be returned from the function.

Example:

Here is the format of a function that calculates the sum and the product of three input variables.

```
function  [sum,product]=example_func(var1,var2,var3)
sum= var1+var2+var3;
product=var1*var2*var3;
```

Save the name of the m-file to be "example_func.m"

In the command window type:

```
>> a=3;
>> b=4;
>> c=5;
>> example_func(a,b,c)
```

The output of the last instruction will be as follows:

```
>>
    ans =
            12
```

Now try calling it using:

```
>> [x,y]=example_func(a,b,c)
```

The output will be:

```
>>
     x=
        12
     y=
         60
```

It is clear that in the first case you didn't assign output arguments, so the first output only was calculated while in the second case both of them were assigned because of assigning the sum to the variable x and the product to the variable y. Note that you don't have limitations on the names of the function variables and the calling variables, i.e. it needs not to be the same names.
To check the availability of your function name, use `help your_func_name` before creating it.
You can add any help comments for your function by inserting the following line before the first line in by inserting the following line before the first line in your function (i.e. before the word "function"):

```
% This function calculates the sum and the product of three input arguments.
```

Then type :

```
>> help myfunc
```

On the command window to get the help comment you inserted.

Note that: most of the MATLAB instructions that deal with vectors and matrices, previously discussed, can do the same task with the symbolic forms.

MATLAB version: MATLAB 6.5 R13 is preferred (or higher versions).

M-files: you can use the MATLAB editor (M-file editor) instead of using the command window as it's more flexible to change any instructions or parameters in the editor of the M-file then save changes and re-run the code, to get it from the MATLAB toolbar: File → New → M file then save it and write your code or instructions.

Very important: dealing with MATLAB doesn't require full memorization of its instructions and their formats, but you are preferred only to memorize the name of the instructions and then you may use the great help of MATLAB to know every thing about these instructions. Just write the following on the MATLAB command window:

```
>> help instruction_name
```

Toolboxes: MATLAB has many great toolboxes which are simply a group of predefined functions and blocks that may help in the field of engineering such as the Communication, Control, RF, …… toolboxes. Later on, we will focus on these toolboxes. For more details about these toolboxes, just type:

>> help *toolbox_name*   for e.g.:   >> help communication

Java & C++: MATLAB deals with Java and C++ efficiently, so you can switch between them through a compiler that changes the code to the other programming method.

Text book: you can use the following text book as a reference for you:

Duane Hanselman(2001)
Bruce Littlefield
"Mastering MATLAB 6"
A Comprehensive Tutorial and Reference
Prentice Hall, ISBN 0130194689

Visit: http://www.mathworks.com

Instructor: *Eng. Moustafa Adly*
moustafa.adly@guc.edu.eg