



DEPARTMENT OF
COMPUTER TECHNIQUES ENGINEERING

COMPUTER APPLICATIONS

FOR
2ND STAGE STUDENTS

1st Lecture

LECTURER
SAFA HUSSAIN

Contents: -

- Introduction to MATLAB.
- MATLAB Environment.
- MATLAB Windows: -
 - ❖ Command Window.
 - ❖ Workspace Window.
 - ❖ Command History Window.
 - ❖ Help Window.
 - ❖ Editor Window.
 - ❖ Access your files
 - ❖ create variables.

What is the Matlab?

- Is a high-performance, high-level language.
- Stands for Matrix Laboratory.
- Can be used as a fancy calculator.
- Allows you to easily work with entire matrices rather than one number at a time.
- Is useful for anything that requires matrix and vector manipulations such as:
 - ❖ Mathematical, scientific, engineering, statistical and financial problems.
 - ❖ Anything that requires plotting/visualizing and analyzing data.
- Comes with a basic set of tools for visualizing data and for performing calculations on matrices and vectors.

MATLAB ENVIRONMENT

The MATLAB environment (on most computer systems) consists of menus, buttons and a writing area similar to an ordinary word processor. There are plenty of help functions that you are encouraged to use. The writing area that you will see when you start MATLAB, is called the command window. In this window you give the commands to MATLAB. For example, when you want to run a program you have written for MATLAB you start the program in the command window by typing its name at the prompt. The command window is also useful if you just want to use MATLAB as a scientific calculator or as a graphing tool. If you write longer programs, you will find it more convenient to write the program code in a separate window, and then run it in the command window (discussed in Intro to programming).

In the command window you will see a prompt that looks like `>>` . You type your commands immediately after this prompt. Once you have typed the command you wish MATLAB to perform, press `<enter>`.

MATLAB Windows: -

The MATLAB Work Environment

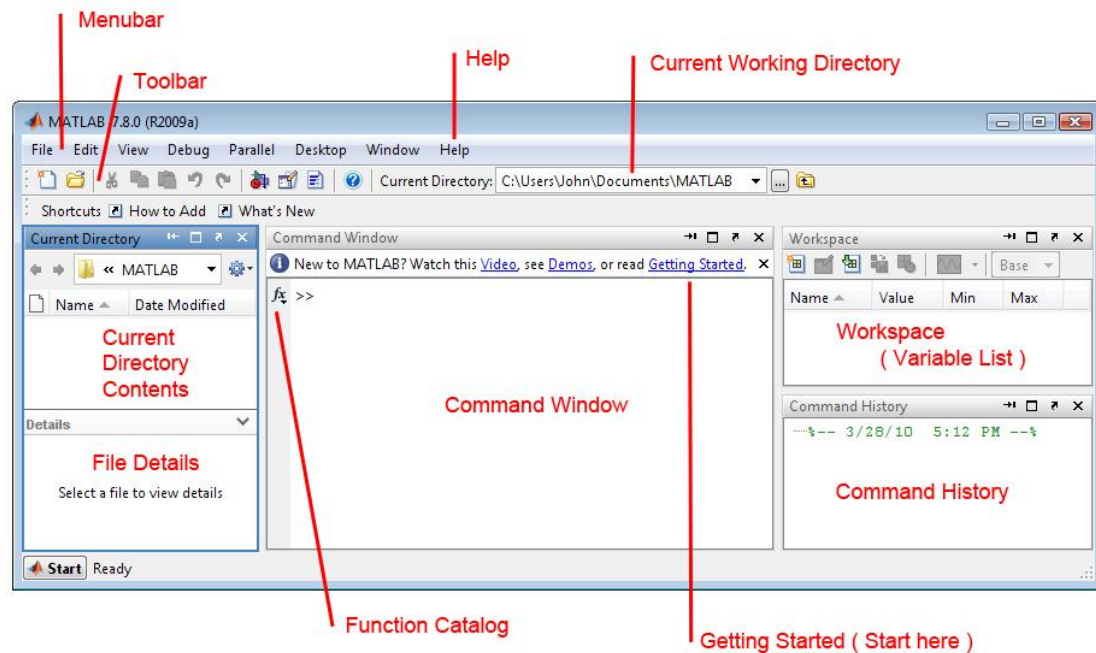


Figure 1

The Command Window

Often times you will work at the command line in the command window. The command window allows you to type commands directly and see the results immediately.

For example at the `>>`, define a variable "a" by typing

```
>> a=[1 2]
```

What is printed out below is the output of the command

```
a =
```

```
1 2
```

The Command History Window

shows the commands you have entered in the past. You can repeat any of these commands by double-clicking on them, or by dragging them from the Command History Window into the Command Window. You can also scroll back to previous commands by using the up arrow in the Command Window. When you learn how to edit Matlab script files or functions, you can also drag commands from the Command History Window into a file.

The Workspace Window

shows the list of variables that are currently defined, and what type of variable each is. (I.e. a simple scalar, a vector, or a matrix, and the size of all arrays.) Depending on the size (i.e. type) of the variable, its value may also be shown.

Getting Help

Matlab has an incredibly good help system. The help is built into Matlab and can be accessed from the help menu, but the full help is available online at mathworks.com. Because this help is so good, it is silly to reproduce everything here. We show some brief descriptions on commands below, but we expect you to READ THE HELP FILES.

To get help on a command, type "doc commandname" in Command Window where commandname is the command of interest. For example:

```
>> doc plot
```

gives

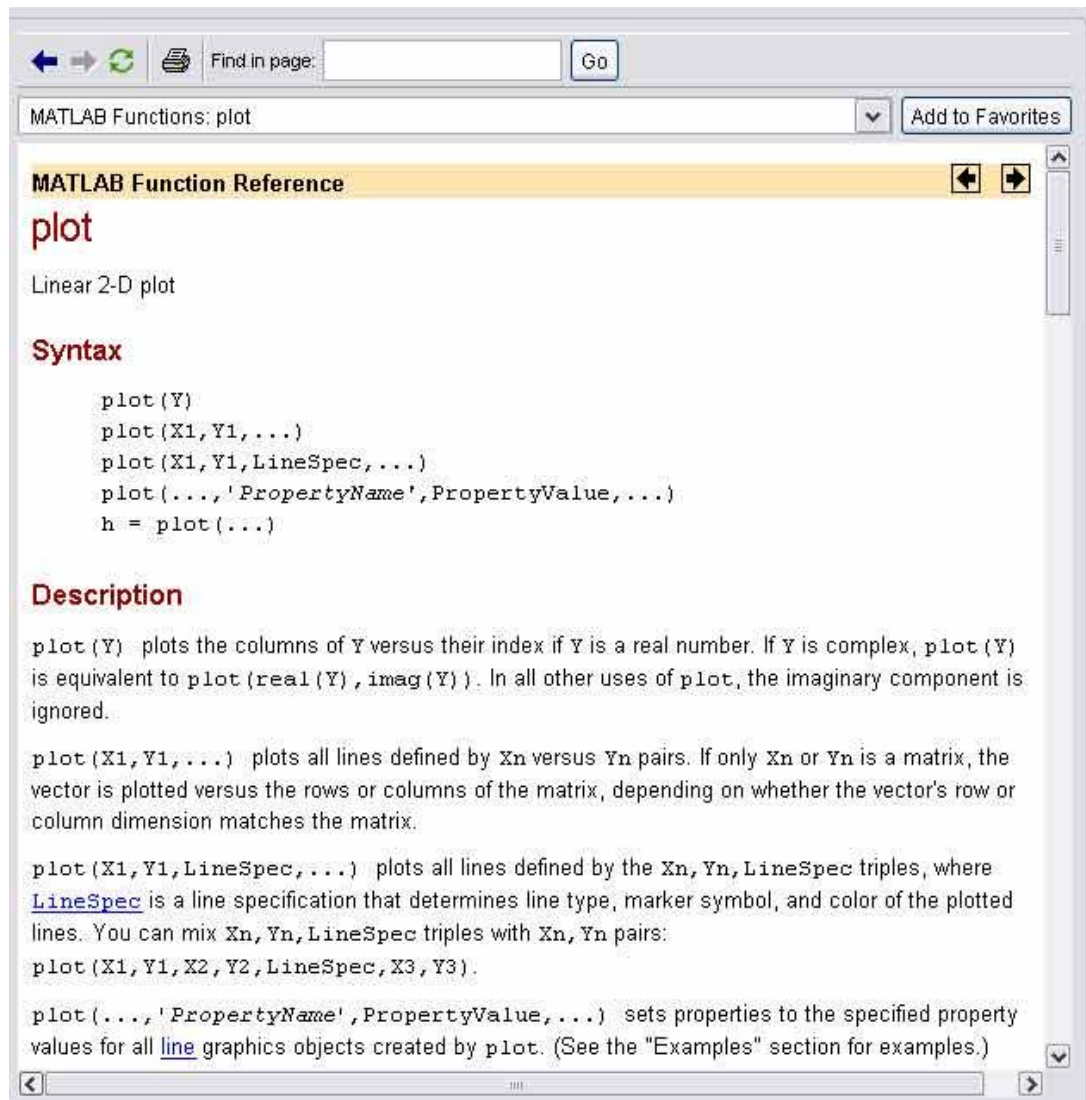


Figure 2

The Editor Window

Alternatively, you can write a program or script called a m-file (has a .m extension) in the Editor Window. The Editor Window is a word processor specifically designed for Matlab commands, so it automatically formats certain things for you such as the command "for" below

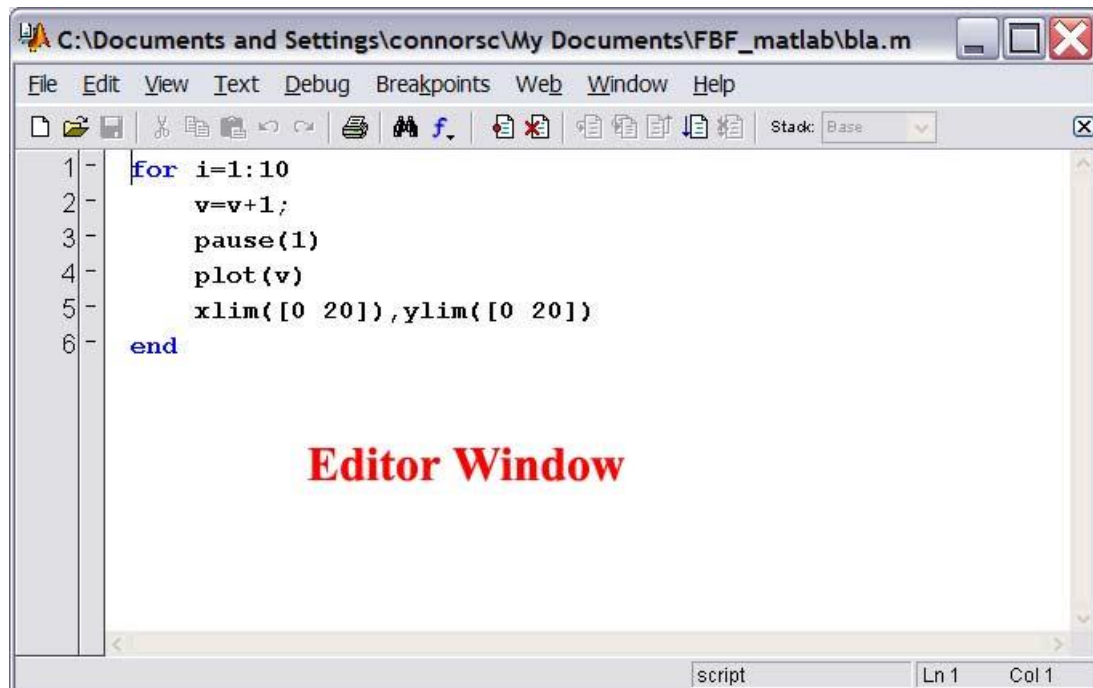
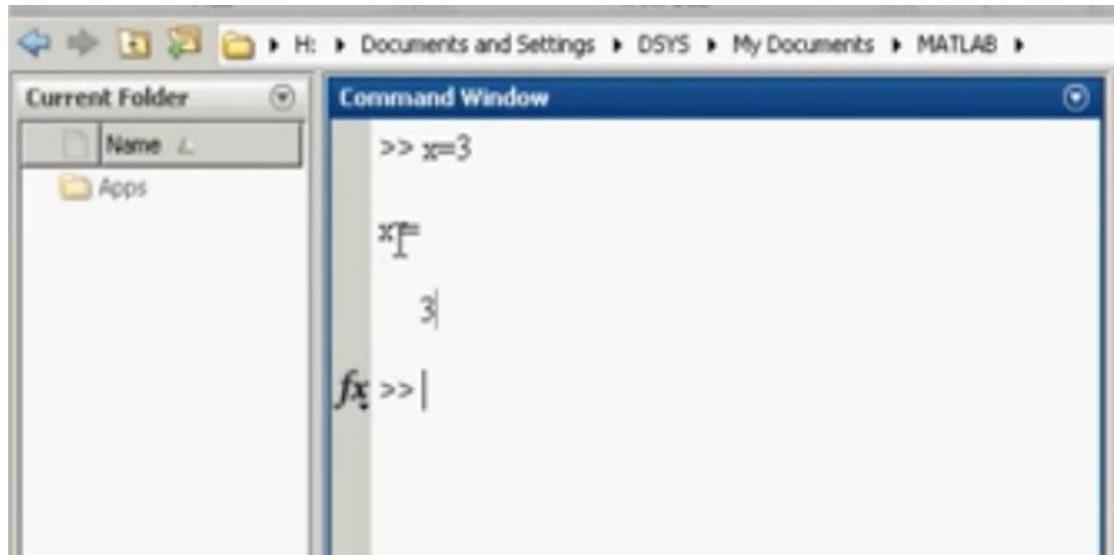


Figure 3

This program may consist of one or many commands to execute. Once the program is saved you can just type the name of the file and all of the commands will execute. It is common to try a series of commands at the command line and once you like them, copy them from the Command History and then paste them into a m-file. If you want to make comments that help explain in English what different code does you put a "%" symbol in front of the text.

Access your files



create variables

As you work in MATLAB, you issue commands that create variables and call functions. For example, create a variable named *a* by typing this statement at the command line:

```
a = 1
```

MATLAB adds variable *a* to the workspace and displays the result in the Command Window.

```
a=
```

```
1
```

Create a few more variables.

```
b = 2
```


b =

2

c = a + b

c =

3

d = cos(a)

d =

0.5403

- When you do not specify an output variable, MATLAB uses the variable `ans`, short for *answer*, to store the results of your calculation.

sin(a)

ans =

0.8415

- If you end a statement with a semicolon, MATLAB performs the computation, but suppresses the display of output in the Command Window.

e = a*b;

- You can recall previous commands by pressing the up- and down-arrow keys, ↑ and ↓. Press the arrow keys either at an empty command line or after you type the first few characters of a command. For example, to recall the command `b = 2`, type `b`, and then press the up-arrow key.



DEPARTMENT OF
COMPUTER TECHNIQUES ENGINEERING

COMPUTER APPLICATIONS

FOR
2ND STAGE STUDENTS

نظري _ عملي

LECTURER
SAFA HUSSAIN

Commonly used Operators and Special Characters

MATLAB supports the following commonly used operators and special characters:

Operator	Purpose
+	Plus; addition operator.
-	Minus; subtraction operator.
*	Scalar and matrix multiplication operator.
.*	Array multiplication operator.
^	Scalar and matrix exponentiation operator.
.^	Array exponentiation operator.
\	Left-division operator.
/	Right-division operator.
.\	Array left-division operator.
./	Array right-division operator.
:	Colon; generates regularly spaced elements and represents an entire row or column.
()	Parentheses; encloses function arguments and array indices; overrides precedence.
[]	Brackets; encloses array elements.

.	Decimal point.
...	Ellipsis; line-continuation operator
,	Comma; separates statements and elements in a row
;	Semicolon; separates columns and suppresses display.
% —	Percent sign; designates a comment and specifies formatting. Quote sign and transpose operator.
·—	Non-conjugated transpose operator.
=	Assignment operator.

>> 14/4

ans =

3.5000

>> ans ^ (-6)

ans =

5.4399e-004

Example:

Evaluate the MATLAB expressions

The expressions are given by:

$$1+2/3*4-5$$

$$1/2/3/4$$

$$1/2+3/4*5$$

$$5-2*3*(2+7)$$

$$(1+3)*(2-3)/3*4$$

$$(2-3*(4-3))*4/5$$

In MATLAB

$$1+2/3*4-5 = 1 + \frac{2}{3}4 - 5 = -\frac{4}{3},$$

$$1/2/3/4 = (((1/2)/3)/4) = \frac{1}{24},$$

$$1/2+3/4*5 = \frac{1}{2} + \frac{3}{4}5 = \frac{17}{4},$$

$$5-2*3*(2+7) = 5 - 6(9) = -49,$$

$$(1+3)*(2-3)/3*4 = \frac{4 \times (-1)}{3}4 = -\frac{16}{3},$$

$$(2-3*(4-3))*4/5 = (2 - 3 \times 1)\frac{4}{5} = -\frac{4}{5};$$

>> 1+2*3

ans =

7

>> x = 1+2*3

x =

7


```
>> 4*x
ans =
28.0000

>> t = 5;

>> t = t+1
t =
6
```

Special Variables and Constants

MATLAB supports the following special variables and constants:

Name	Meaning
ans	Most recent answer.
eps	Accuracy of floating-point precision.
i,j	The imaginary unit $\sqrt{-1}$.
Inf	Infinity.
NaN	Undefined numerical result (not a number).
pi	The number π has the value 3.1415926...

Operations	Description
==	Equal
~=	Not equal
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal

Example:

```
>> a=1:9; b=9-a;
>> t=a>4 %finds elements of (a) that are
greater than 4.
t =
0 0 0 0 1 1 1 1 1
```

Zeros appear where $a \leq 4$, and ones where $a > 4$.

Experiment No. (4) Relational and Logical Operations

```
>> t= (a==b) %finds elements of (a) that are equal to those
in (b).
t =
0 0 0 0 0 0 0 0 0
```

Logical Operation

Operation	Description
&	Logical AND and(a,b)
	Logical OR or(a,b)
~	Logical NOT
xor (a,b)	Logical EXCLUSIVE OR

Example:

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

```
>> c=a&b
c =
0 0 0 0 0 0
```

Example:

let **x**=[2 -3 5 ;0 11 0] , then

- a) find elements in x that are greater than 2
 - b) find the number of nonzero elements in x
- solution

a)

```
>> x>2
```

```
ans =
```

```
0 0 1
```

```
0 1 0
```

b)

```
>> t=~(~x) ;
```

```
>> sum(sum(t))
```

```
ans =
```

```
4
```

Bitwise Operation

MATLAB also has a number of functions that perform bitwise logical operations.

If A, B unsigned integers then:

Operation	Description
bitand (A, B)	BitwiseAND
bitor (A, B)	Bitwise OR
bitset (A, BIT)	sets bit position BIT in A to 1
bitget (A, BIT)	returns the value of the bit at position BIT in A
xor (A, B)	Bitwise EXCLUSIVE OR

Example: if A=5, B=6 then:

```
>> bitget(A,3)
```

```
ans = 1
```

where A= 1 0 1 0 0 000

```
>> bitget(A,(1:8))
```

```
ans =
```

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

```
>> bitand(A,B)
```

```
ans =
```

```
4
```

```
>> and(A,B)
```

```
ans =
```

```
1
```

Housekeeping Functions

Here are some functions that don't really do math but are useful in programming.

`abs(x)` the absolute value of a number (real or complex)

`clc` clears the command window; useful for beautifying printed
output

`ceil(x)` the nearest integer to x looking toward +1

`clear` clears all assigned variables

`close` all closes all figure windows

`fix(x)` the nearest integer to x looking toward zero

`fliplr(A)` flip a matrix A, left for right

`flipud(A)` flip a matrix A, up for down

`floor(x)` the nearest integer to x looking toward -1

`length(a)` the number of elements in a vector

`mod(x,y)` the integer remainder of x/y; see online help if x or y are
negative

rem(x,y) the integer remainder of x/y; see online help if x or y are negative

rot90(A) rotate a matrix A by 90_

round(x) the nearest integer to x

sign(x) the sign of x and returns 0 if x=0

size(c) the dimensions of a matrix

sqrt(x) Square root.

acos(x) Inverse cosine.

angle(x) Phase angle (angle of complex number).

asin(x) Inverse sine.

atan(x) Inverse tangent.

exp(x) Exponential: e^x .

imag(x) Complex imaginary part.

log(x) Natural logarithm: $\ln(x)$.

log10(x) Common (base 10) logarithm: $\log_{10}(x)$.

rand(n) returns an N-by-N matrix containing pseudorandom values drawn from the standard uniform distribution on the open interval(0,1). rand(M,N) returns an M-by-N matrix. rand returns a scalar.

randn(n) returns an N-by-N matrix containing pseudorandom values drawn from the standard normal distribution on the open interval(0,1). randn(M,N) returns an M-by-N matrix. randn returns a scalar.

Elementary math functions

Function	Description	Example
<code>sqrt(x)</code>	Square root.	<code>>> sqrt(81)</code> <code>Ans=9</code>
<code>nthroot(x,n)</code>	Real n th root of a real number x . (If x is negative n must be an odd integer.)	<code>>> nthroot(80,5)</code> <code>ans =</code> <code>2.4022</code>
<code>exp(x)</code>	Exponential(e^X)	<code>>> exp(5)</code> <code>ans =</code> <code>148.4132</code>
<code>abs(x)</code>	Absolute value	<code>. >> abs(-24)</code> <code>ans =</code> <code>24</code>
<code>log(x)</code>	Natural logarithm. Base e logarithm (\ln).	<code>>> log(1000)</code> <code>ans =</code> <code>6.9078</code>
<code>log10(x)</code>	Base 10 logarithm	<code>>> log10(1000)</code> <code>ans =</code> <code>3.0000</code>
<code>factorial(x)</code>	The factorial function $x!$ (x must be a positive integer.)	<code>>> factorial(5)</code> <code>ans =</code> <code>120</code>

Examples of Expressions:-

```
1) >> x = (1+sqrt(5))/2
```

```
x =
```

```
1.6180
```

```
>> a = abs(3+4i)
```

```
a =
```

```
5
```

```
>> y=sin(pi/3)+cos(pi/4)-2*sqrt(3)
```

```
y =
```

```
-1.8910
```

2) Solve the following system

```
x+y=1
```

```
x-y+z=0
```

```
x+y+z=2
```

Solution

```
>> a=[1 1 0; 1 -1 1; 1 1 1];
```

```
    b=[1;0;2];
```

```
>> x=inv(a)*b
```

```
or
```

```
>> x=a\b
```

Another functions:-

clear Removes all variables from the memory.

clear x y z Removes only variables x, y, and z from the memory.

who Displays a list of the variables currently in the memory.

Whos Displays a list of the variables currently in The memory and their sizes together with information about their bytes and class

Try to type the following assignment:

```
>> x = 2;
```

```
y = 4;
```

```
z = x*y
```

```
z=
```

```
8
```

```
>> who
```

Your variables are:

```
x y z
```

```
>> whos
```

Name	Size	Bytes	Class	Attributes
x	1x1	8	double	
y	1x1	8	double	
z	1x1	8	double	

Now Try to do the following:

```
>> a=3
```

```
>> a=3; can you see the effect of semicolon " ; "
```

```
>> a+5 assign the sum of a and 5 to ans
```

```
>> b=a+5 assign the sum of a and 5 to b
```

```
>> clear a
```

```
>> a can you see the effect of clear command
```

```
>>clc clean the screen
```

```
>> b
```

Precedence	Mathematical Operation
First	Parentheses. For nested parentheses, the innermost are executed first.
Second	Exponentiation.
Third	Multiplication, division (equal precedence).
Fourth	Addition and subtraction.

>> 7+8/2

ans =

11



Type and press **Enter**.
8/2 is executed first.

>> (7+8)/2

ans =

7.5000



Type and press **Enter**.
7+8 is executed first.

>> 4+5/3+2

ans =

7.6667



5/3 is executed first.

>> 5^3/2

ans =

62.5000



5^3 is executed first, /2
is executed next.

>> 27^(1/3)+32^0.2

ans =

5



1/3 is executed first,
27^(1/3) and 32^0.2 are
executed next, and + is executed
last.

```
>> 27^1/3+32^0.2
```



27^1 and 32^0.2 are executed first, /3 is executed next, and + is executed last.

```
ans =  
11
```

```
>> 0.7854-(0.7854)^3/(1*2*3)+0.785^5/(1*2*3*4*5)...
```

Type three periods ... (and press **Enter**) to continue the expression on the next line.



```
-(0.785)^7/(1*2*3*4*5*6*7)
```

```
ans =
```

The last expression is the first four terms of the Taylor series for $\sin(\pi/4)$.

```
0.7071
```

Example:

```
>> 1+2
```

```
Ans=
```

```
3
```

```
>> 3 ^ 2
```

% 3 raised to the power of 2

```
ans=
```

```
9
```

```
>> A= 1+2
```

```
A =
```

```
3
```

```
>> B=2*A
```

```
B =
```

```
6
```

```
>> x = 3-2^4
```

```
x = -13
```

```
>> C=B^A
```

```
C =
```

```
216
```

```
>> sin(pi /2)
```

% sine of angle 90°

```
ans =
```

```
1
```

```
>>7/0
```

% Divide by zero

```
ans=
```

```
inf
```

```
>> sqrt(9)
```

% root mean square of 9 i.e. 9

```
ans=
```

```
3
```

Consider the expressions

$$x^3 + bx^2 + cx - 2.3 \quad \frac{a^2 - b^3}{c + 4} \quad \frac{\frac{ab}{x-15}}{d + \frac{3}{5}}$$

The corresponding MATLAB program lines are

`x 3 + b*x 2 + c*x - 2.3, (a 2 - b 3)/(c + 4), ((a*b)/(x - 15))/(d + 3/5)`

MATLAB is case sensitive

(-X) is not equal to (x)

- Variable names must start with a letter

- You'll get an error if this doesn't happen

- After that, can be any combination of letters, numbers and underscores

```
>> x=4
```

```
X =
```

```
4
```

```
>>X = 7
X=
7
>> x - X
ans=
-3
```

Elementary functions

cos(x) Cosine	abs(x) Absolute value
sin(x) Sine	sign(x) Signum function
tan(x) Tangent	max(x) Maximum value
acos(x) Arc cosine	min(x) Minimum value
asin(x) Arc sine	ceil(x) Round towards +1
atan(x) Arc tangent	floor(x) Round towards -1
exp(x) Exponential	round(x) Round to nearest integer
sqrt(x) Square root	rem(x) Remainder after division
log(x) Natural logarithm	angle(x) Phase angle
log10(x) Common logarithm	conj(x) Complex conjugate

Trigonometric math functions

Function Description Example:

```
1)sin(x)
sind(x)
Sine of angle x (x in radians).
Sine of angle x (x in degrees).
>> sin(pi/6)
ans =
0.5000
```

```
2)cos(x)
cosd(x)
Cosine of angle x (x in radians).
Cosine of angle x (x in degrees).
```

```
>> cosd(30)
ans =
0.8660
```

```
3)tan(x)
tand(x)
Tangent of angle x (x in radians).
Tangent of angle x (x in degrees).
>> tan(pi/6)
ans =
0.5774
```

```
4)cot(x)
cotd(x)
Cotangent of angle x (x in radians).
Cotangent of angle x (x in degrees).
>> cotd(30)
ans =
1.7321
```

Function Description Example:

1)round(x) Round to the nearest integer.

```
>> round(17/5)
```

```
ans =
```

```
3
```

2)fix(x) Round toward zero.

```
>> fix(13/5)
```

```
ans =
```

```
2
```

3)ceil(x) Round toward infinity.

```
>> ceil(11/5)
```

```
ans =
```

```
3
```

4)floor(x) Round toward minus infinity.

```
>> floor(-9/4)
```

```
ans =
```

```
-3
```

5)rem(x,y) Returns the remainder after x is

divided by y.

```
>> rem(13,5)
```

```
ans =3
```

Example:

We illustrate here some typical examples which related to the elementary functions previously defined.

As a first example, the value of the expression $y = e^{a \sin(x)} + 10$

p

y , for $a = 5$, $x = 2$, and

$y = 8$ is computed by

```
>> a = 5; x = 2; y = 8;
```

```
>> y = exp(-a)*sin(x)+10*sqrt(y)
```

```
y =
```

```
28.2904
```

The subsequent examples are

```
>> log(142)
```

```
ans =
```

```
4.9558
```

```
>> log10(142)
```

```
ans =
```

```
2.1523
```

Note the difference between the natural logarithm $\log(x)$ and the decimal logarithm (base

10)

$\log_{10}(x)$.

To calculate $\sin(\pi/4)$ and e^{10} , we enter the following commands in MATLAB,

```
>> sin(pi/4)
```

```
ans =
```

```
0.7071
```

```
>> exp(10)
```

```
ans =2.2026e+004
```

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.

Entering a Vector

- A vector is a special case of a matrix. An array of dimension $1 \times n$ is called a row vector, whereas an array of dimension $m \times 1$ is called a column vector.
- The elements of vectors in MATLAB are enclosed by square brackets and are separated by spaces or by commas. For example, to enter a row vector, v, type

Ex:

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

```
v =
```

```
1 4 7 10 13
```

- Column vectors are created in a similar way; however, semicolon (;) must separate the components of a column vector

Ex:

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

```
u =
```

```
1
```

```
4
```

```
7
```

```
10
```

```
13
```

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

MATLAB will execute the above statement and return the following result:

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

To obtain a vector whose entries are 0, 2, 4, 6, and 8, you can type in the following line:

```
>> v = [0:2:8]  
v =  
0 2 4 6 8
```

Here we specified the first entry 0, the increment value 2, and the last entry 8. The two colons (:) “tell” MATLAB to fill in the (omitted) entries using the specified increment value.

- On the other hand, a row vector is converted to a column vector using the transpose operator. The transpose operation is denoted by an apostrophe or a single quote (').

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

Colon Notation and Extracting Parts of a Vector

To access blocks of elements, we use MATLAB's colon notation (:). For example, to access the first three elements of v, we write,

```
v = [1 4 7 10 13]  
>> v (1:3)  
ans =  
1 4 7
```

Or, all elements from the third through the last elements,

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

```
>> u = [0:8]
```

```
u =
```

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

Here we specified the first entry 0 and the last entry 8, separated by a colon (:).

In general, first: step: last produces a vector of entities with the value first, incrementing by the step until it reaches last:

```
>> 0.2 : 0.5 : 2.4
```

```
ans =
```

```
0.2000 0.7000 1.2000 1.7000 2.2000
```

```
>> -3 : 3 : 10
```

```
ans =
```

```
-3 0 3 6 9
```

Parts of vectors can be extracted by using a colon notation :

```
>> r = [-1:2:6, 2, 3, -2]
```

```
r =
```

```
-1 1 3 5 2 3 -2
```

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

```
A =
```

```
1 2 3
```

```
3 4 5
```

```
6 7 8
```

We can avoid separating each row with a semicolon if we use a carriage return instead. In other

words, we could have defined A as follows

```
>> A = [
```

```
1 2 3
```

```
3 4 5
```

```
6 7 8]
```

```
A =
```

```
1 2 3
```

```
3 4 5
```

```
6 7 8
```

which is perhaps closer to the way we would have defined A by hand using the linear algebra notation.

You can refer to a particular entry in a matrix by using parentheses. For example, the number 5 lies in the 2nd row, 3rd column of A, thus

```
» A(2,3)
```

```
ans =
```

```
5
```

The order of rows and columns follows the convention adopted in the linear algebra notation.

This means that $A(2, 3)$ refers to the number 5 in the above example and $A(3, 2)$ refers to the number 7, which is in the 3rd row, 2nd column.

Note MATLAB's response when we ask for the entry in the 4th row, 1st column.

```
» A(4,1)
```

```
??? Index exceeds matrix dimensions.
```

As expected, we get an error message. Since A is a 3-by-3 matrix, there is no 4th row and

MATLAB realizes that. The error messages that we get from MATLAB can be quite informative

```
9
```

when trying to find out what went wrong. In this case MATLAB told us exactly what the problem was.

We can “extract” submatrices using a similar notation as above. For example to obtain the submatrix that consists of the first two rows and last two columns of A we type

```
» A(1:2,2:3)
```

```
ans =
```

```
2 3
```

```
4 5
```

We could even extract an entire row or column of a matrix, using the colon (:) as follows.

Suppose we want to get the 2nd column of A. We basically want the elements $[A(1, 2)$

```
A(2,2) A(3,2)]. We type
```

```
» A(:,2)
```

```
ans =
```

```
2
```

```
4
```

```
7
```

where the colon was used to tell MATLAB that all the rows are to be used. The same can be done when we want to extract an entire row, say the 3rd one.

```
>> A(3, :)
```

```
ans =
```

```
6 7 8
```

MATLAB will allow you to look at specific parts of the vector. If you want, for example, to only

look at the first 3 entries in the vector v , you can use the same notation you used to create the

vector:

```
>> v(1:3)
```

```
ans =
```

```
0 2 4
```

Note that we used parentheses, instead of brackets, to refer to the entries of the vector. Since we

omitted the increment value, MATLAB automatically assumes that the increment is 1. The

following command lists the first 4 entries of the vector v , using the increment value 2 :

```
>> v(1:2:4)
```

```
ans =
```

```
0 4
```

Operations on Vectors

A number of operations can be done on vectors. A vector can be multiplied by a scalar , or added /subtracted to/ from another vector with the same length, or a number can be added/subtracted to/ from a vector. All these operations are carried out element-by-element.

```
>> v = [-1 2 7] ; w = [2 3 4];
```

```
>> z = v + w
```

```
z =
```

```
1 5 11
```

A simple way to compute the inner product is by the use of the dot product, i.e (\cdot) , which performs element-wise multiplication. For two vectors x and y , of the same length, is defined as a vector $[x1$

$y_1, x_2, y_2, \dots, x_n, y_n]$ thus, the corresponding elements of two vectors are multiplied. For instance:

```
>> u = [-1 3 5];  
>> v = [-1 2 7];  
>> u .* v  
ans =  
1 6 35
```

Mathematically, it is not defined how to divide one vector by another. However, in MATLAB, the operator (./) is defined to perform an element-by-element division. It is, therefore, defined for vectors of the same size and type:

```
>> u ./ v  
ans =  
1.0000 1.5000 0.7143
```

Matrices

- Matrices are fundamental to MATLAB. Therefore, we need to become familiar with matrix generation and manipulation. Matrices can be generated in several ways.

Entering a Matrix

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

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

Here is a typical example. To enter a matrix A, such as, type,

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

MATLAB then displays the 3x3 matrix as follows,

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

Matrix Indexing

We select elements in a matrix just as we did for vectors, but now we need two indices. The element of row i and column j of the matrix A is denoted by $A(i, j)$. Thus, $A(i, j)$ in MATLAB refers to the element A_{ij} of matrix A . The first index is the row number and the second index is the column number. For example, $A(1,3)$ is an element of first row and third column. Here, $A(1,3)=3$.

Correcting any entry is easy through indexing. Here we substitute $A(3,3)=9$ by $A(3,3)=0$. The result is

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

```
A =
```

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

Creating a Sub-Matrix

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

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

```
>> B = A ( [2 3] , [1 2] )
```

```
B =
```

```
4 5
7 8
```

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

```
>> C = A ( [2 1 3], :)
```

```
C =
```

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

- To delete a row or column of a matrix, use the empty vector operator, $[]$.

```
>> A(3,:) = []
```

```
A =
```

```
1 2 3
4 5 6
```

Third row of matrix A is now deleted. To restore the third row, we use a technique for creating a matrix

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

A =

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

Matrix A is now restored to its original form.

- To determine the dimensions of a matrix or vector, use the command size. For example,

```
>> size(A)
```

ans =

```
3 3
```

Transposing a Matrix

The transpose operation is denoted by an apostrophe or a single quote ('). It flips a matrix about its main diagonal and it turns a row vector into a column vector. Thus,

```
>> A'
```

ans =

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

Defining a matrix is similar to defining a vector. To define a matrix A, you can treat it like a

column of row vectors. That is, you enter each row of the matrix as a row vector (remember to

separate the entries either by commas or spaces) and you separate the rows by semicolons (;).

Define now another matrix B, and two vectors *s* and *t* that will be used in what follows.

```
>> B = [
```

```
-1 3 10
```

```
-9 5 25
```

```
0 14 2]
```

B =

```
-1 3 10
```

```
-9 5 25
```

```
0 14 2
```

```
>> s = [-1 8 5]
```

```
s =
```

```
-1 8 5
```

```
10
```

```
>> t = [7;0;11]
```

```
t =
```

```
7
```

```
0
```

```
11
```

The real power of MATLAB is the ease in which you can manipulate your vectors and matrices.

For example, to subtract 1 from every entry in the matrix A we type

```
>> A-1
```

```
ans =
```

```
0 1 2
```

```
2 3 4
```

```
5 6 7
```

It is just as easy to add (or subtract) two compatible matrices (i.e. matrices of the same size).

```
>> A+B
```

```
ans =
```

```
0 5 13
```

```
-6 9 30
```

```
6 21 10
```

The same is true for vectors.

```
>> s-t
```

```
??? Error using ==> -
```

Matrix dimensions must agree.

This error was expected, since s has size 1-by-3 and t has size 3-by-1. We will not get an error if

we type

```
>> s-t'
```

```
ans =
```

```
-8 8 -6
```

since by taking the transpose of t we make the two vectors compatible.

We must be equally careful when using multiplication.

```
>> B*s
```

```
??? Error using ==> *
```

Inner matrix dimensions must agree.

```

>> B*t
11
ans =
103
212
22

```

Matrix Generators

MATLAB provides functions that generates elementary matrices. The matrix of zeros, the matrix of ones, and the identity matrix are returned by the functions `zeros`, `ones`, and `eye`, respectively.

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

Make sure you ask for `help` on all the above commands.

To create the identity matrix of size 4 (i.e. a square 4-by-4 matrix with ones on the main diagonal

and zeros everywhere else) we use the command `eye`.

```

>> eye(4,4)

```

```

ans =

```

```

1 0 0 0

```

```

0 1 0 0

```

```

0 0 1 0

```

```

0 0 0 1

```

```

>> eye(3)

```

```

ans =

```

```

1 0 0

```

```

0 1 0

```

```

0 0 1

```

The numbers in parenthesis indicates the size of the matrix. When creating *square* matrices, we

can specify only one input referring to size of the matrix. For example, we could have obtained the above identity matrix by simply typing `eye(4)`. The same is true for the matrix building functions below.

Similarly, the command `zeros` creates a matrix of zeros and the command `ones` creates a matrix of ones.

```
>> zeros(2,3)
```

```
ans =
```

```
0 0 0
```

```
0 0 0
```

```
>> ones(2)
```

```
ans =
```

```
1 1
```

```
1 1
```

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

```
b =
```

```
1
```

```
1
```

```
1
```

We can create a randomly generated matrix using the `rand` command. (The entries will be uniformly distributed between 0 and 1.)

```
>> C = rand(5,4)
```

```
C =
```

```
0.2190 0.3835 0.5297 0.4175
```

```
0.0470 0.5194 0.6711 0.6868
```

```
0.6789 0.8310 0.0077 0.5890
```

```
0.6793 0.0346 0.3834 0.9304
```

```
0.9347 0.0535 0.0668 0.8462
```

As mentioned earlier, the command `diag` has two uses. The first use is to extract a diagonal of a matrix, e.g. the main diagonal. Suppose `D` is the matrix given below. Then, `diag(D)` produces

a column vector, whose components are the elements of D that lie on its main diagonal.

```
>> D = [  
0.9092 0.5045 0.9866  
0.0606 0.5163 0.4940  
0.9047,0.3190,0.2661];  
>> diag(D)  
ans =  
0.9092  
0.5163  
0.2661
```

The second use is to create diagonal matrices. For example,

```
>> diag([0.9092;0.5163;0.2661])  
ans =  
0.9092 0 0  
0 0.5163 0  
0 0 0.2661
```

creates a diagonal matrix whose non-zero entries are specified by the vector given as input. (A

short cut to the above construction is `diag(diag(D))`).

This command is not restricted to the main diagonal of a matrix; it works on off diagonals as well.

See `help diag` for more information.

```
>> A = [9,7,0;0,8,6;7,1,-6]  
A =  
9 7 0  
0 8 6  
7 1 -6  
>> size(A)  
ans =  
3 3
```

Matrix Arithmetic Operations

MATLAB allows arithmetic operations: +, -, *, and ^ to be carried out on matrices.

Those operations work as for vectors: they address matrices in the element-by-element way, therefore they can be performed on matrices of the same sizes. They also allow for scalar – matrix operations.

For the dot product or division, corresponding elements are multiplied together or divided one by another. Thus,
 $A+B$ or $B+A$ is valid if A and B are of the same size
 $A*B$ is valid if A's number of column equals B's number of rows
 A^2 is valid if A is square and equals $A*A$
 $\alpha*A$ or $A*\alpha$ multiplies each element of A by α

OPERATION	MATRIX	ARRAY
Addition	+	+
Subtraction	—	—
Multiplication	*	.*
Division	/	./
Left division	\	.\
Exponentiation	^	.^

- **Matrix additions and subtractions**

A few examples of basic operations are provided below:

```
>>B = [ 1 -1 3; 4 0 7]
```

```
B =
```

```
1  -1  3
4   0  7
```

```
>> B2= [ 1 2; 5 1; 5 6 ];
```

```
>> B= B+B2'
```

%add two matrices

```
B=
```

```
2  4  8
6  1 13
```

```
>> B-2
```

**% subtract 2 from all elements
of B**

```
ans =
```

```
0  2  6
```


4 -1 11

The code for this should be relatively self explanatory:

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

```
>> disp(3*A-B)  
>> disp(A*B)
```

- **Matrix multiplications and divisions**

We can perform array multiplication. It is important to recognize that this is not matrix multiplication. We use the same notation used when multiplying two vectors together (.*) .

$$A.*B = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} (a_{11})(b_{11}) & (a_{12})(b_{12}) \\ (a_{21})(b_{21}) & (a_{22})(b_{22}) \end{pmatrix}$$

Example

```
>> A = [12 3; -1 6];  
      B = [4 2; 9 1];  
>> C = A.*B  
C =  
    48    6  
    -9    6  
• Let B= [ 2 4 8 ; 6 1 13 ]  
>> B=  
    2    4    8  
    6    1   13
```

```
>> B./4                                % divide all elements of the
                                       matrix B by 4
```

```
ans =
0.5000  1.0000  2.0000
1.5000  0.2500  3.2500
```

Vector Functions

Other MATLAB functions operate essentially on vectors returning a scalar value. Some of these

functions are given in the table below.

max	largest component
min	smallest component
length	length of a vector
sort	sort in ascending order
sum	sum of elements
prod	product of elements
median	median value
mean	mean value
Magic	The syntax of this function

Example:

Let z be the following row vector.

```
» z = [0.9347,0.3835,0.5194,0.8310]
```

```
z =
0.9347  0.3835  0.5194  0.8310
```

Then

```
» max(z)
```

```
ans =
0.9347
```

```
» min(z)
```

```
ans =
0.3835
```

```

>> sort(z)
ans =
0.3835 0.5194 0.8310 0.9347
>> sum(z)
ans =
2.6686
>> mean(z)
ans =
0.6671

```

Example:

```

>> M = [
0.7012,0.2625,0.3282
0.9103,0.0475,0.6326
0.7622,0.7361,0.7564];

```

If we used the `max` command on `M`, we will get the row in which the maximum element lies

(remember the vector functions act on matrices in a column-by-column fashion).

```

>> max(M)
ans =
0.9103 0.7361 0.7564

```

To isolate the largest element, we must use the `max` command on the above row vector. Taking

advantage of the fact that MATLAB assigns the variable name `ans` to the answer we obtained,

we can simply type

```

>> max(ans)
ans =
0.9103

```

The two steps above can be combined into one in the following.

```

>> max(max(M))
ans =
0.9103

```

(length(A)) Returns the number of elements in the vector A.

```
>> A=[5 9 2 4];  
>> length(A)  
ans =  
4
```

(size(A)) Returns a row vector [m,n], where m and n are the size of the array A.

```
>> size(A)  
ans =  
2 5
```

(mean(A)) If A is a vector, returns the mean value of the elements of the vector.

```
>> mean(A)  
ans =  
5  
>> A=[6 1 4 0 12; 5 19 6  
8 2]  
A =  
6 1 4 0 12  
5 19 6 8 2
```

Example:

```
>> A=[5 9 2 4 11 6 11 1];  
>> C=max(A)  
C =  
11
```

```
>> [d,n]=max(A)
d =
11
n =
5
```

```
>> A=[5 9 2 4];
min(A)
>> A=[5 9 2 4];
>> min(A)
ans =
2
```

```
sum(A)
>> A=[5 9 2 4];
>> sum(A)
ans =
20
```

```
sort(A)
>> A=[5 9 2 4];
>> sort(A)
ans =
2 4 5 9
```

```
median(A)
>> A=[5 9 2 4];
>> median(A)
ans =
4.5000
```

Example

For b=1 2 3 4 ; 5 6 7 8 ; 9 10 11 12 ; 13 14 15 16 represent a matrix write this matrix in matlab form and find the sum function.

Solution:

```
>> b=[1 2 3 4 ; 5:8 ; 9 10 11 12 ; 13 14 15 16]
```

b=

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

```
>> sum(b)
```

ans=

28 32 36 40

Example:

Find the *min* function for above matrix b and return the result at x variable.

```
>> x= min(b)
```

x=

1 2 3 4

Example:

Find the *max* function for above matrix b and return the result at ENB variable.

```
>> ENB= max(b)
```

ENB=

13 14 15 16

Example:

Find the *diag* function for above matrix b and return the result at k variable.

```
>> k=diag(b)
```

k=

1

6

11

16

Example2:

Fined the *diag* function for below matrix A and return the result at variable B.

A=1 15 2 11 ; 23 1 4 5; 3 1 15 7; 1 4 9 10

Solution:

```
>> A=[1 15 2 11 ; 23 1 4 5; 3 1 15 7; 1 4 9 10]
```

A=

1 15 2 11

23 1 4 5

3 1 15 7

1 4 9 10

```
>> B=diag(A)
```

B=

1

1

15

10

Example: Fined the mean function for above matrix b and return the result at M variable.

```
>> M=mean(b)
```

M= 2.5 6.5 10.5 14.5

Example: Fined the sum(b(:)) function for above matrix b.

```
>>sum(b(:))
```

ans= 136

(size function): number of row and number of column

Example:

Fined the size function of the above matrix b.

```
>> size(b)
```

ans=

4 4

Number of row

Number of column

```
>> A=[3 4 9;2 4 5]

A =

    3    4    9
    2    4    5
```

الأمر size

```
>> size(A)
```

عدد الصفوف

```
ans =
```

عدد الأعمدة

```
    2    3
```

Find the prod command of below matrix A and return the result at B variable.

Solution

```
>> A=[1 15 2 11; 23 1 4 5; 3 1 15 7; 1 4 9 10]
```

```
A =
```

```
    1   15    2   11
   23    1    4    5
    3    1   15    7
    1    4    9   10
```

```
>> B=prod(A)
```

```
B =
```

```
    69    60   1080   3850
```

Example : For the matrix A below find the sum(diag(A)) and prod(diag(A)) for each case return the result at variable B.

Solution


```
>> A=[1 15 2 11; 23 1 4 5; 3 1 15 7; 1 4 9 10]
```

```
A =
```

```
    1    15     2    11
   23     1     4     5
     3     1    15     7
     1     4     9    10
```

```
>> B=sum(diag(A))
```

```
B =
```

```
    27
```

```
>> A=[1 15 2 11; 23 1 4 5; 3 1 15 7; 1 4 9 10]
```

```
A =
```

```
    1    15     2    11
   23     1     4     5
     3     1    15     7
     1     4     9    10
```

```
>> B=prod(diag(A))
```

```
B =
```

```
    150
```

Magic function: The syntax of this function

Variable name = magic(N) where N the number of row that must equal to the number of column i.e. generate rectangular matrix.

```
>> A=magic(3)
```

```
A =
```

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

Note:

- For variable Vector: var1
- var1(:) gives all row or column elements
- var1(1:5) gives the first five row or column elements
- Var1 (2) gives the second element

Example:

For M a vector with elements 2:3:42 write in matlab form and find
k=M(2:5) M(:) R=M(7)

Solution:

```
>>M=[2:3:42]
```

M=

2 5 8 11 14 17 20 23 26 29 32 35 38 41

```
>> k=M(2:5)
```

ans=

5 8 11 14

```
>> M(:)
```

Ans =

2

5

8

11

14

17

20

23

26

29

32

35

38

41

```
>> R=M(7)
```

R=

20

For variable Matrix: mat1

Mat1 (1,3) gives the first row, third column element

Mat1 (:,2) gives all elements in the second column

Mat1 (1,:) gives all elements in the first row

Mat1 (3:4,1:3) gives all elements in the third and fourth rows that

are in the first through third columns

Mat1(2,3) gives one element in second row and third column

Use in place of an index to represent all elements in a row or column of a previously defined matrix

Subscript expressions involving colons refer to portions of a matrix. For example: A(1:k,j)

- refers to the first k elements of the jth column of A.

Example: For S matrix with elements

s= 1 2 3;4 5 6;7 8 9;10 11 12

write this matrix in matlab form and find

R=S(4,:)

S(1:2,2)

S(:,3)

S(6)

K=S(2:4,2:3)

M= (3:4,1:3)

T=(3,2)

Solution:

>>S=[1 2 3;4 5 6;7 8 9;10 11 12]

S= 1 2 3 4 5 6 7 8 9 10 11 12

>>R=S(4,:)

R= 10 11 12

>>S(1:2,2)

ans= 2 5

```

>> S(:,3)
ans=
3
6
9
12
>> S(6)
ans=
5
>> K=S(2:4,2:3)
K=
5 6
8 9
11 12
>> M= (3:4,1:3)
M=
7 8 9
10 11 12
>> T=(3,2)
T=
8

```

```

>> A = [1:3;8:10;11:13]
A = 1    2    3
     8    9   10
    11   12   13
>> A(2,3)
ans = 10
>> A(1:3,2)
ans = 2
     9
    12
>> A(2,:)
ans = 8    9   10

```

Removing Elements of Matrices and Vectors

To remove a row or column of a matrix, set its value to an empty vector:

```
>> x = [7 2 4;9 8 12;4 7 8;12 1 4]
```

```
x= 7 2 4  
    9 8 12  
    4 7 8  
    12 1 4
```

```
>> x(:,2) = [] : All , elements of column two has cancel
```

```
x = 7 4  
    9 12  
    4 8  
    12 4
```

```
>>x(3,:)=[] %means delete all elements of row 3
```

```
x=  
7 2 4  
9 8 12  
12 1 4
```

```
>>x(:,3)=[] %means delete all elements of column 4
```

```
x=  
7 2  
9 8  
4 7  
12 1
```

```
>>x(2,end) % delete end element of second row
```

```
x=  
7 2 4  
9 8  
4 7 8  
12 1 4
```

The same technique can be used to remove single vector elements:

```
>> y = [13 7 12 9 15 8];
```

```
>> y(3) = []
```

$y =$
13 7 9 15 8

Inverse of a matrix:

The matrix B is the inverse of the matrix A if, when the two matrices are multiplied, the product is the identity matrix. Both matrices must be square and the multiplication order can be BA or AB

$$BA = AB = I$$

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

```
A =
```

```
2 1 4
```

```
4 1 8
```

```
2 -1 3
```

```
>> B=inv(A)
```

```
B =
```

```
5.5000    -3.5000    2.0000
```

```
2.0000    -1.0000     0
```

```
-3.0000     2.0000   -1.0000
```

Homework

About the general matrix functions:

For A matrix with the elements as shown

```
[2:2:8; 5 2 4 3 ; 3:6; 1 2 3 4 ]
```

Represent this matrix in MATLAB form and find the following:

```
C=prod(A)
```

```
D=diag(A)
```

```
E=sum(sum(A))
```

```
F=mean(A)
```

```

G=2.*A
H=A(:)
I=A(:,2)=[]
J=max(A)
K=size(A)
L=sum(diag(A))
zeros (2,3)
ones (N,M)
ones (2,3)
size(x)
>>x=[1 2 3 4 5 6];
size(x)
>>v=[1 2 3];
length(v)

>>x=[1 2 3 4 5 6];
max (x)
max(max(x))
min (x)
>> x=[1 2 3 4 5 6];
min (x)
min(min(x))
magic(x)
>> x=[1 2 3 4 5 6];
prod(x)
prod(prod(x))

>>x=[4 6 8 10 9 1 8 2 5];
median(x)
median(x,2)
median(median(x))

```



DEPARTMENT OF
COMPUTER TECHNIQUES ENGINEERING

COMPUTER APPLICATIONS

FOR
2ND STAGE STUDENTS

Third lecture

LECTURER

MSc. Safa Hussain

Introduction to 2-D plot

The drawing is meant to be the relationship that governs the drawing process between two variables only.

*Linspace

It is used by linear vector routing by specifying the smallest number, the largest number and the number of desired points between two numbers

Linspace(a,b,n)

a= it's the small number.

b=it's the large number.

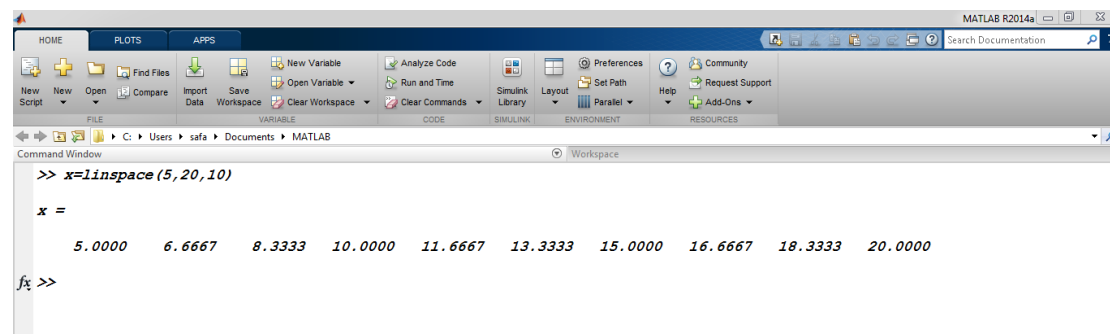
n=number of points required between two digits.

Example:-

Generate a vector (x) where the number of points divided by this vector = 10

X=[5,20]

Sol:



```
>> x=linspace(5,20,10)

x =

    5.0000    6.6667    8.3333   10.0000   11.6667   13.3333   15.0000   16.6667   18.3333   20.0000

fx >>
```

Example:-

Generate a vector where the number of points is divided by 100

X=[0,10]

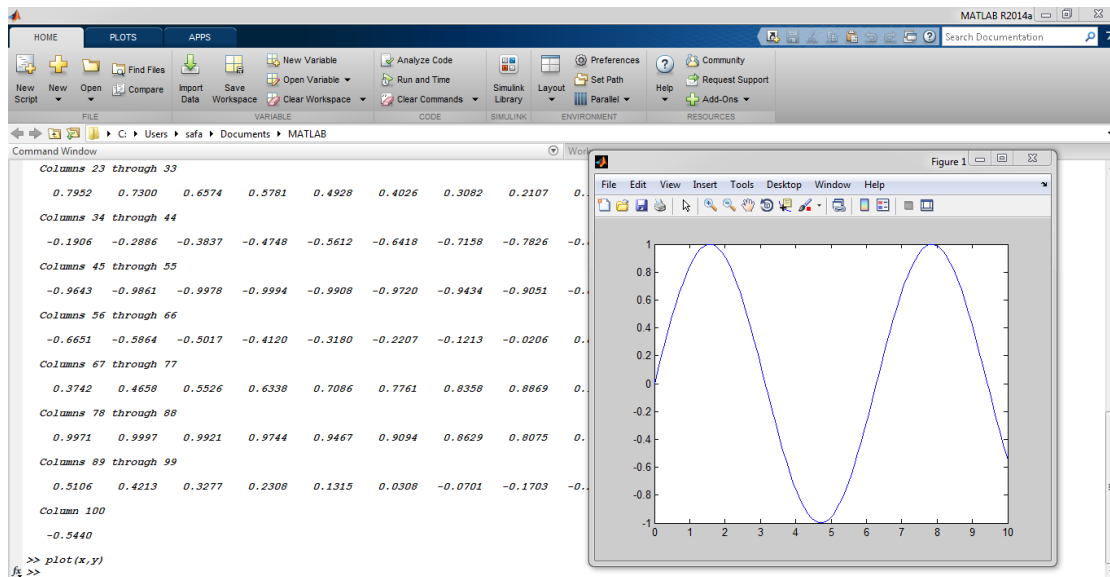
Y=sin(x)

Sol:

X=linspace(0,10,100)

Y=sin(x)

>>plot(x,y)



Setting of 2-D properties:

Some characteristics (change colors and add title to graphic)

How does this property take shape in the Matlab?

This property takes its shape into the matlab, which is included in the order(plot) as it takes the following picture:

Plot(x,y," ")

The property is always between two separators.

Example:

The vector(x) where the number of points divided = 100 and the following equation:

X=[0,1]

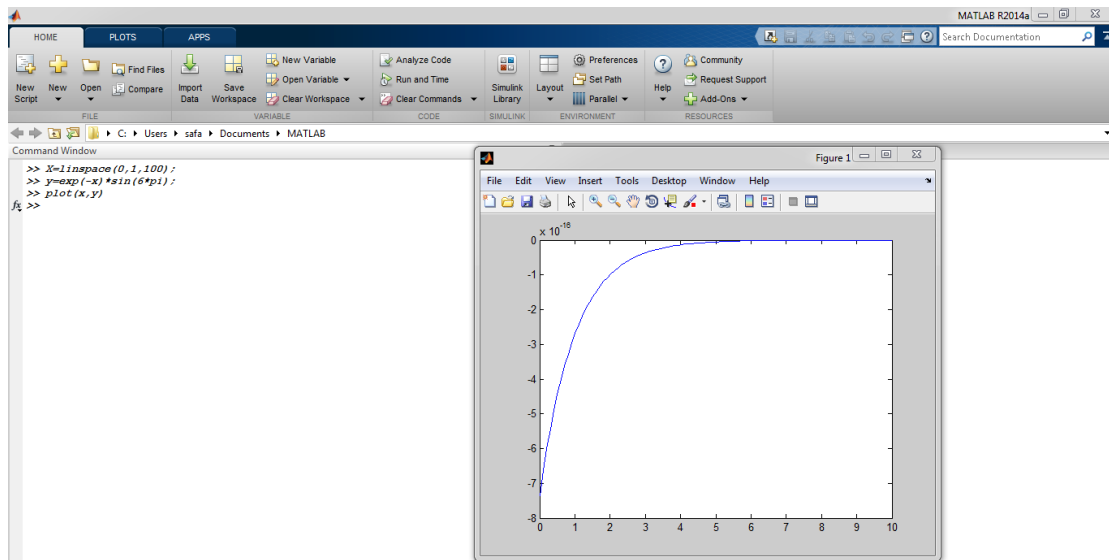
Y=e^{-x} sin(6*pi)

Sol:

X=linspace(0,1,100);

y=exp(-x)*sin(6*pi);

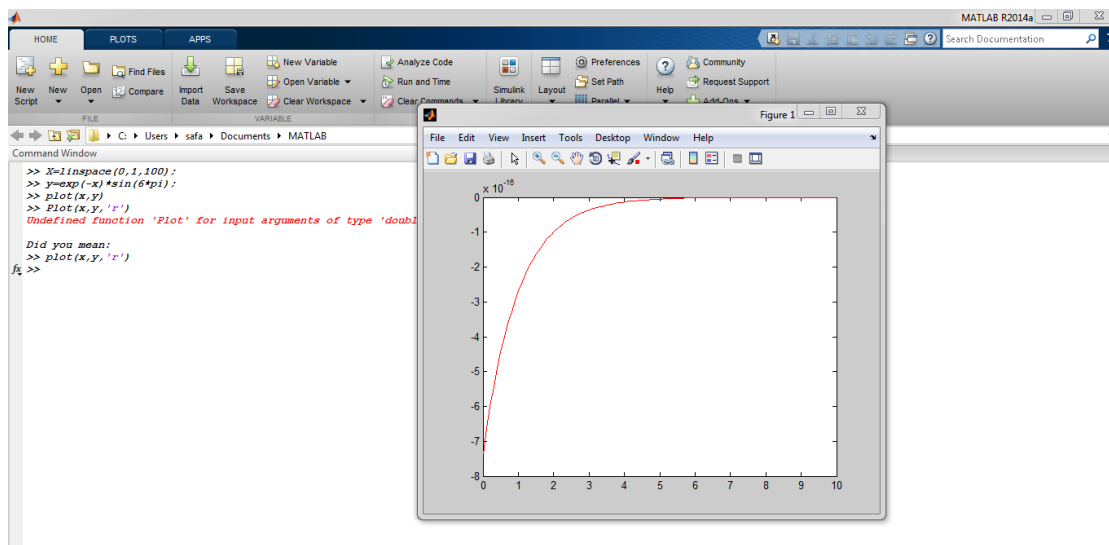
Plot(x,y)



1)First property:

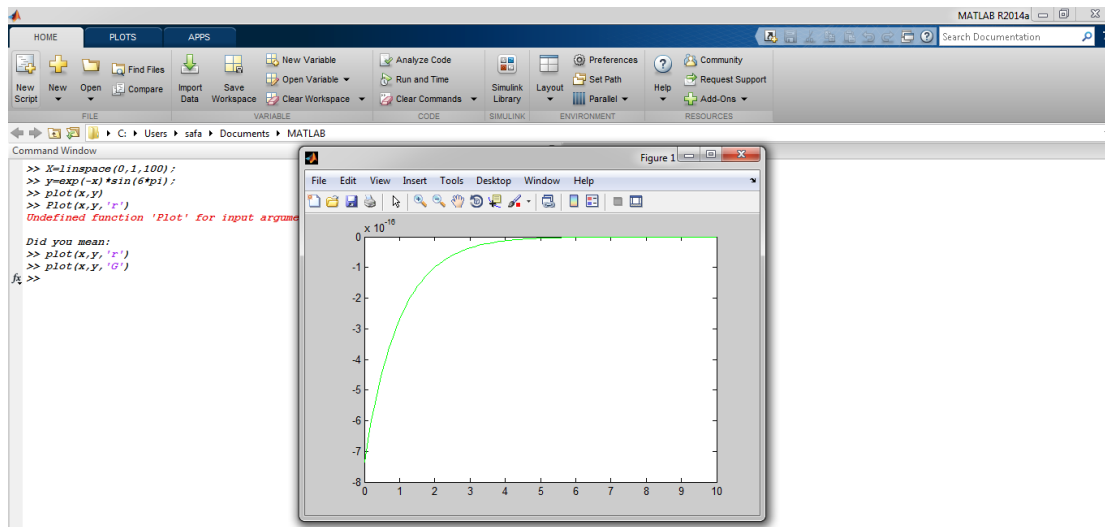
Change the line color

Plot(x,y,'r')



Or

Plot(x,y,'G')

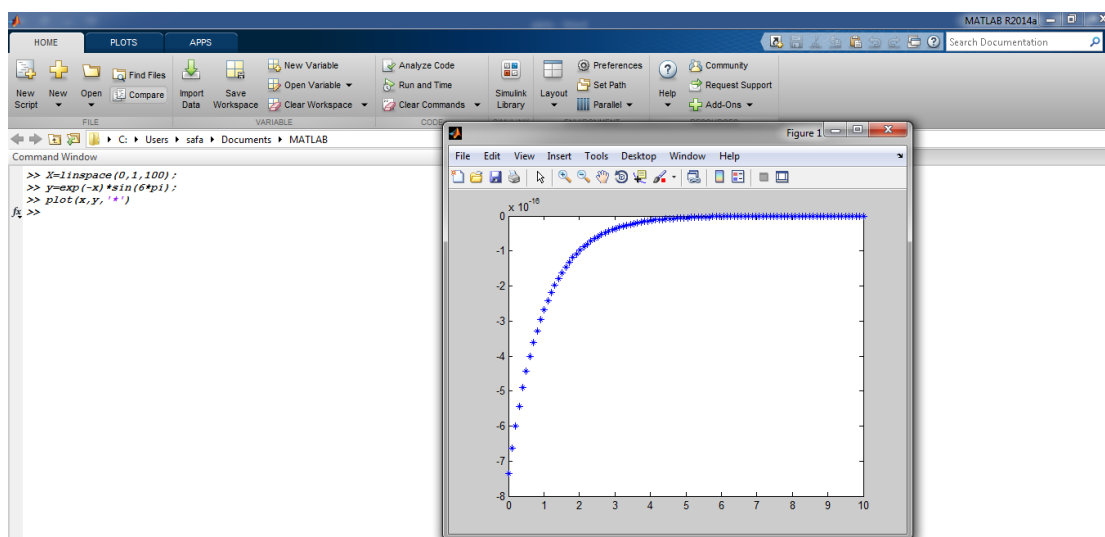


Color		Marker		Line Style	
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star	(none)	no line
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		
		>	triangle (left)		
		p	pentagram		
		h	hexagram		

second property:

Change the shape of the line:

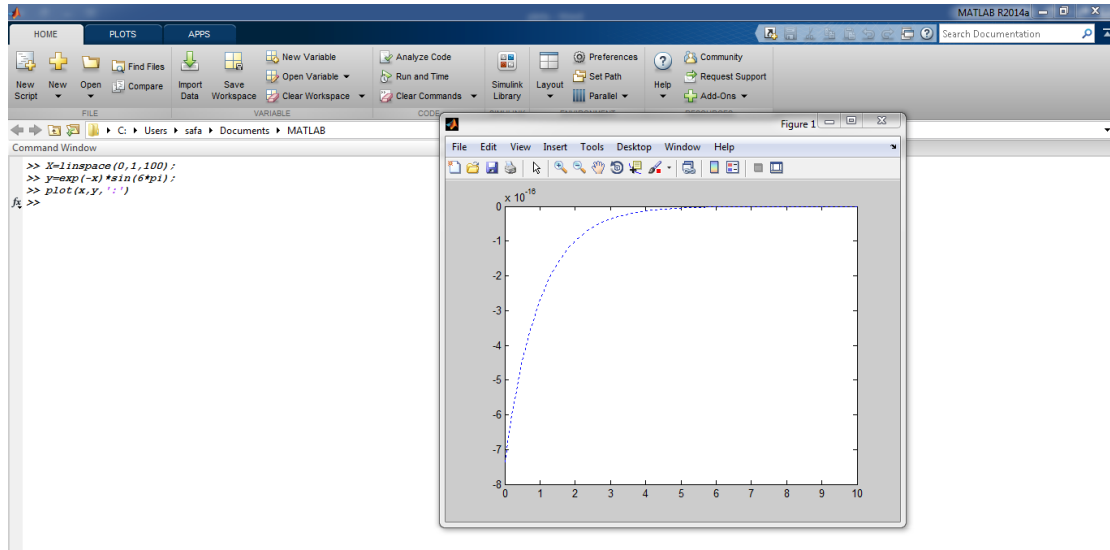
Plot(x,y,'*')



third property:

Change the line type:

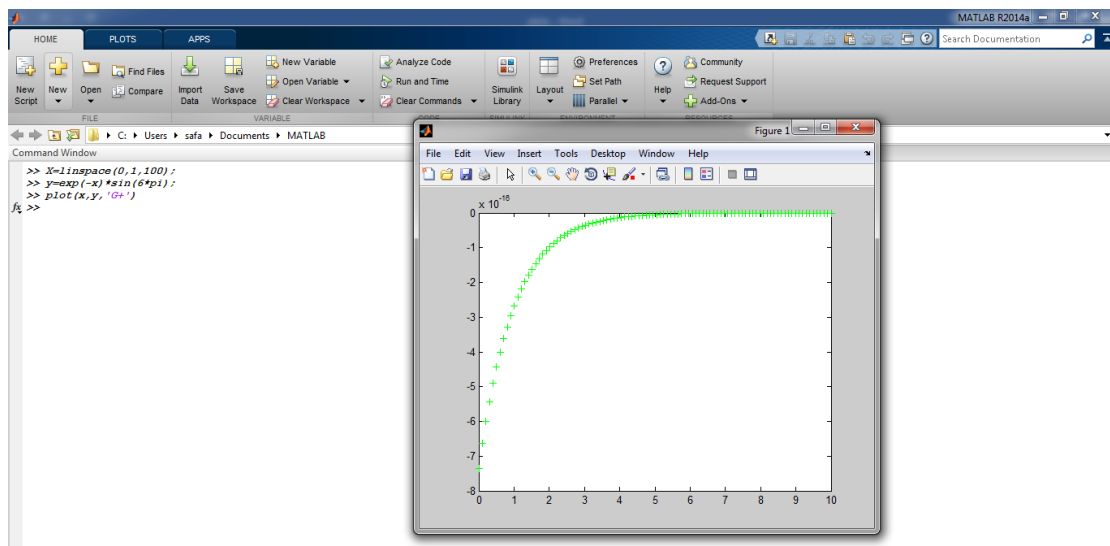
>> plot(x,y,':')



***If we want to merge two properties together**

(Change font color to green and line to + shape)

>> plot(x,y,'G+')

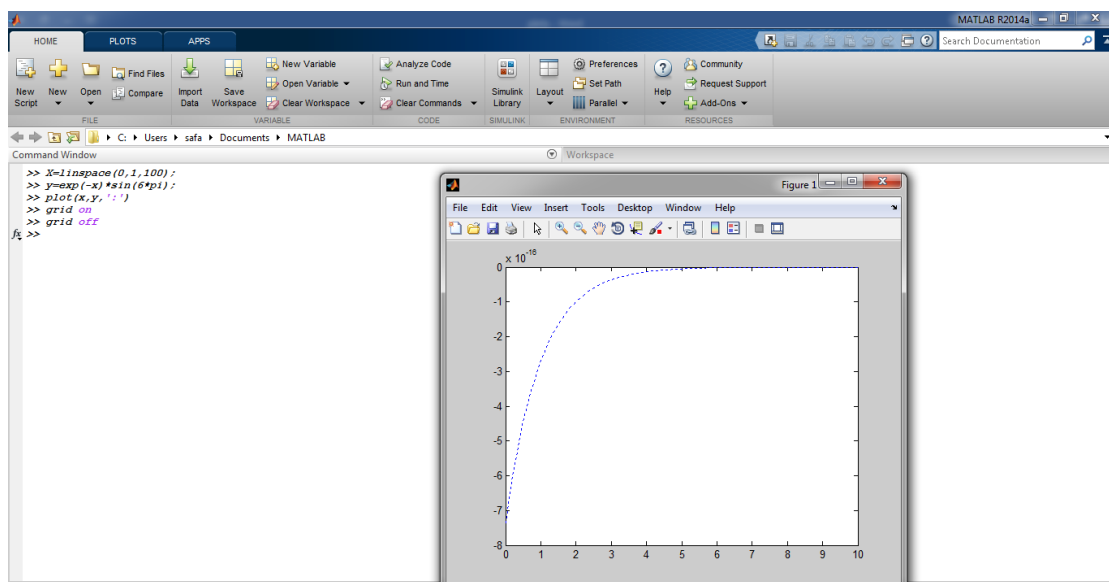
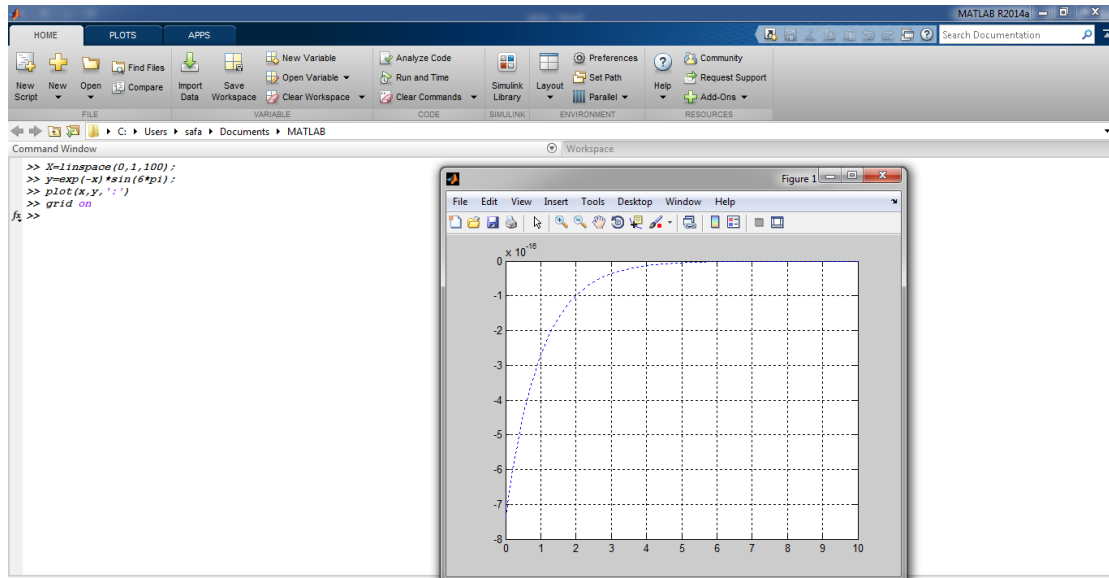


The fourth property (drawing grid)

The matlab places a grid on the drawing so that it is easy to determine the values from the drawing

* To add a grid to the drawing we use the following command **>>grid on**

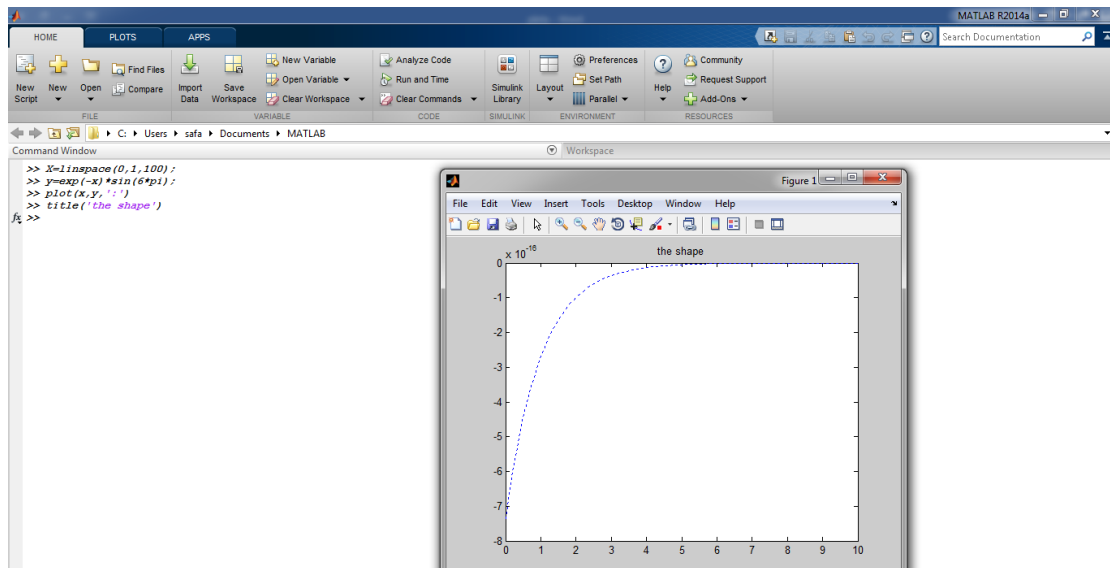
* To remove a grid from the graphic we use the following command **>>grid off**



Fifth property (title and drawing of axis)

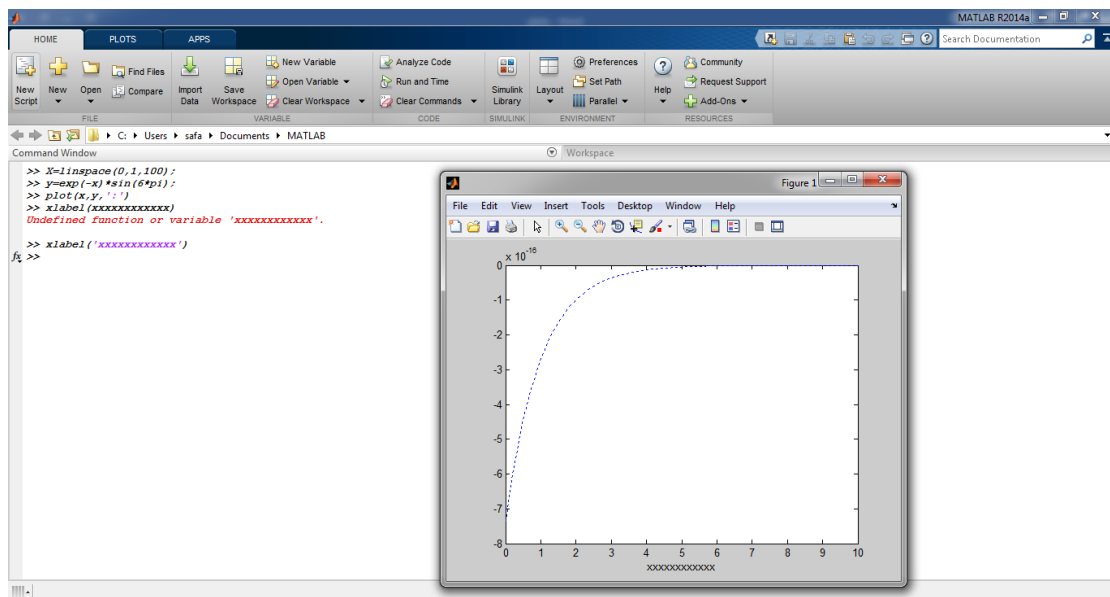
To add a title to the graphic, we use the command (title) where it takes the next image

>>title(' any string')



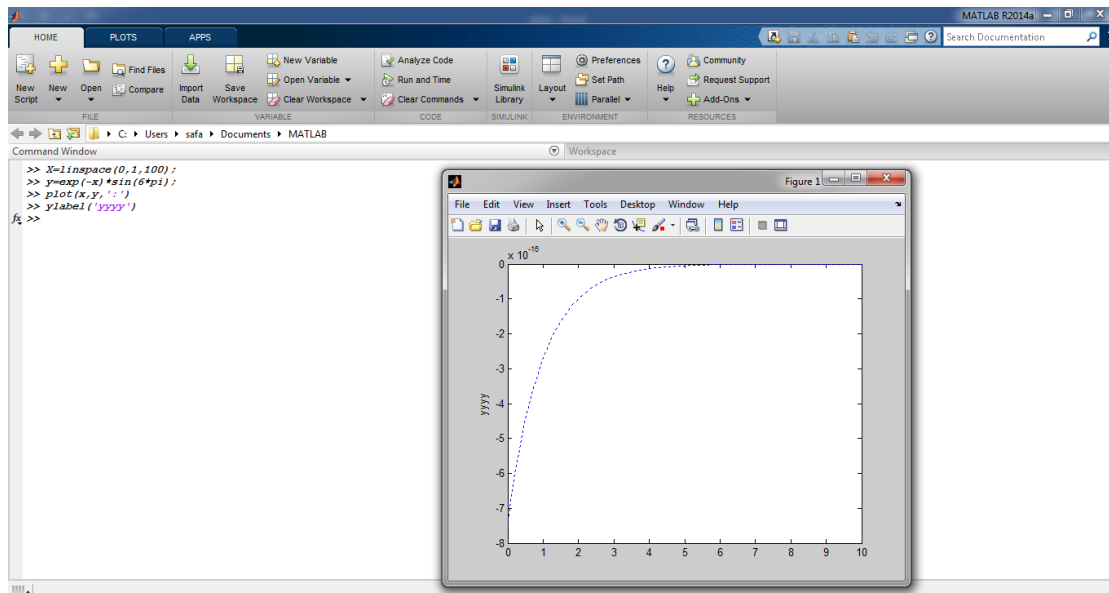
If we want to name the axis of the (X-Axis) we use the command (xlabel):

xlabel('any string')



If we want to name the axis of the (y-Axis) we use the command (ylabel):

ylabel('any string')

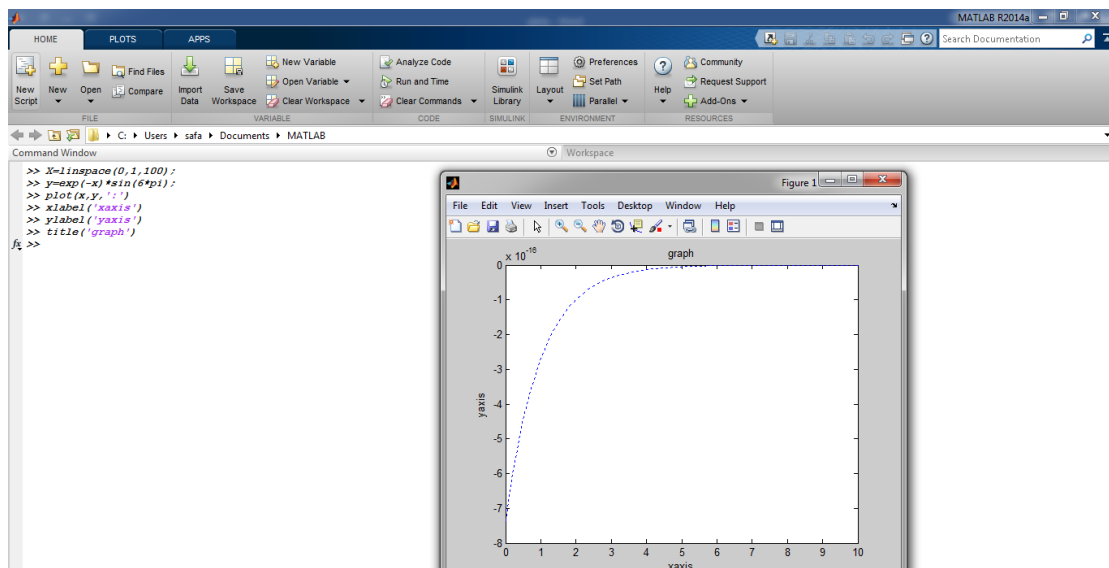


Example:

```
>> xlabel('xaxis')
```

```
>> ylabel('yaxis')
```

```
>> title('graph')
```



The sixth property:

Place text on a dot in the graphic

We can place text on any location in the drawing using the function **(text)**

```
>> text(x,y,'string')
```

X=Position the point on the axis

Y=Position the point on the axis

String=To write the text, be between the commas

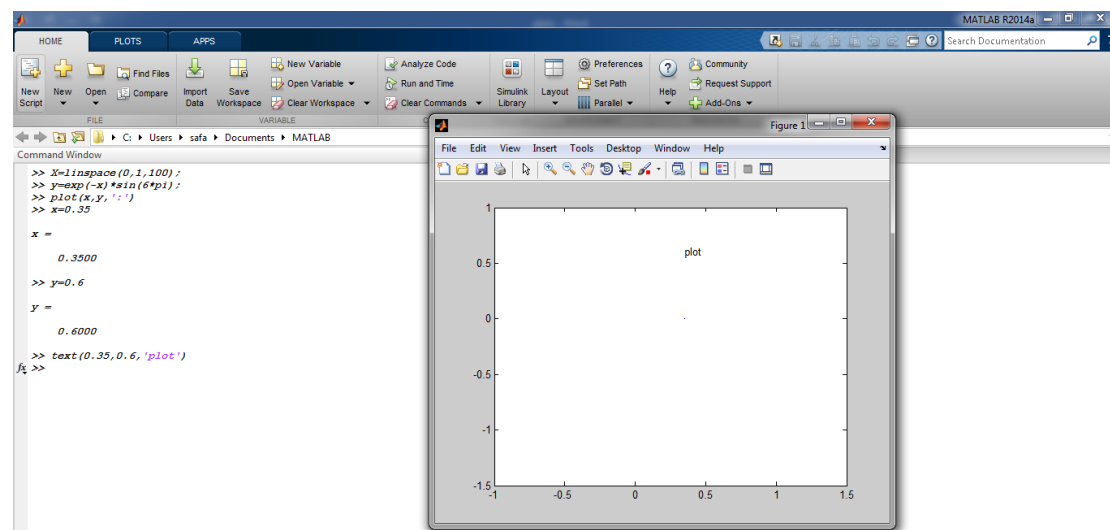
Example:

Place text on the point in the graphic where we want to specify a point between(x,y)

X=0.35

Y=0.6

>>text(0.35,0.6,'plot')



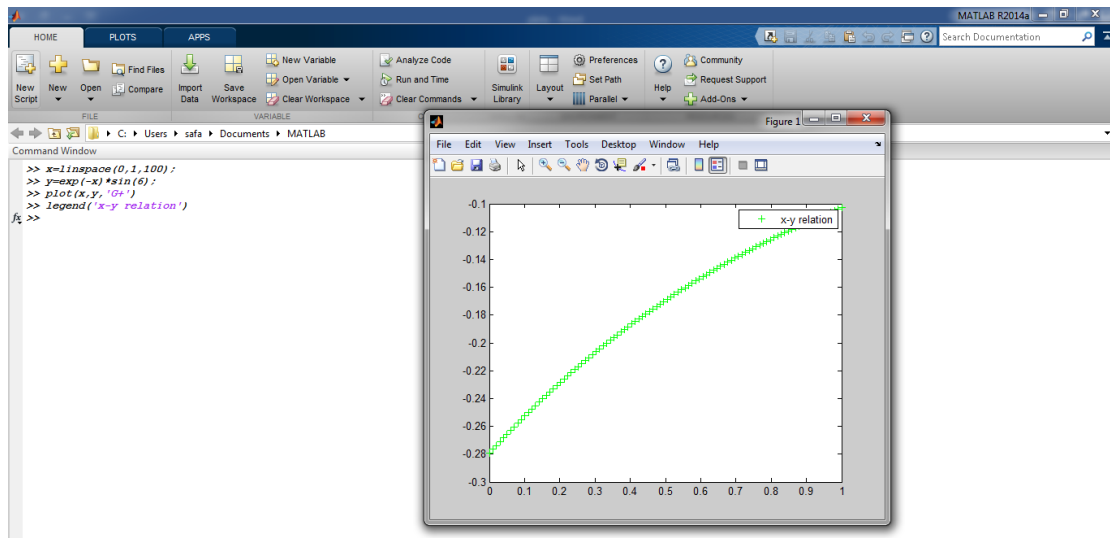
Seven property:

***legend**

This is used by placing a guide on the drawing page to show what each color means on the drawing and take it in the Matlab:

Legend('any string','any string',.....)

>>legend('x-y relation')



Multiple 2-D plots in a window

We can create separate graphics in a single window using the command (subplot) by selecting several graphics that will be illustrated, where (subplot) command to put the image as a matrix or vector.

*When using this command, you must know how many graphics will appear and how to place them on the window

*The general image of the MATLAB (subplot) is used on the following image:

Subplot(m,n,p)

M=Number of rows

N=Number of columns

P=The number of the cell we are running

The following figure shows us the merging of several drawings into a single window

Example:

We have a vector that increases the rate=0.1 of creating these functions drawing to be in one function and several separate drawings $x=[0,10]$:

1-sin(x)

2-sin(2x)

3-cos(x)

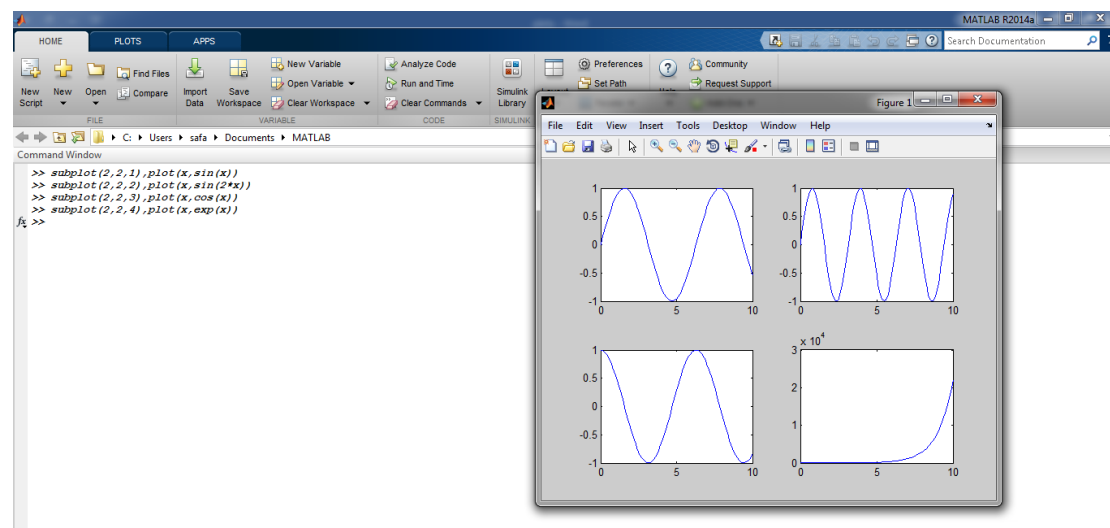
1. `>>x=(0,0.1,10)`

`>>subplot(2,2,1),plot(x,sin(x));`

2.`subplot(2,2,2),plot(x,sin(2*x));`

3.`subplot(2,2,3),plot(x,exp(x));`

4.`subplot(2,2,4),plot(x,exp(x));`



Combining

- Merge over a graphic into a single window

Example:

We have a vector (x) where it increases by=0.1

Create the graphic for the variables (x,y) in a single graphic.

X=[0,10]

X=cos(x) y=sin(x)

Two ways to incorporate :

1-the first graphic using the command(hold on,hold off)

```
>>x=(0:0.1:10)
```

```
>>y=sin(x)
```

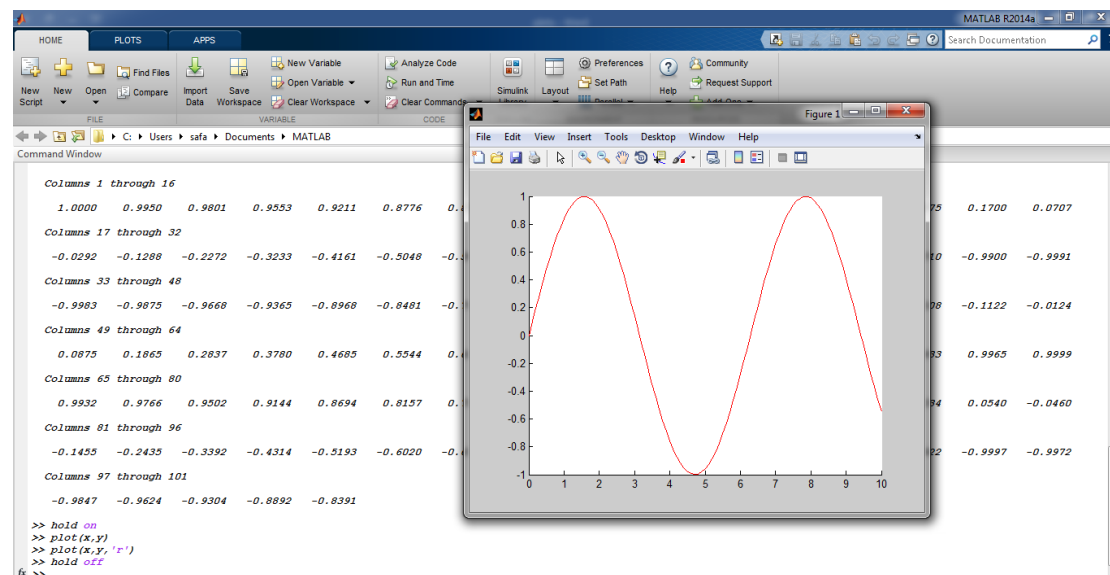
```
>>z=cos(x)
```

```
>>hold on
```

```
>>plot(x,y)
```

```
>>Plot(x,z,'r')
```

```
>>hold off
```



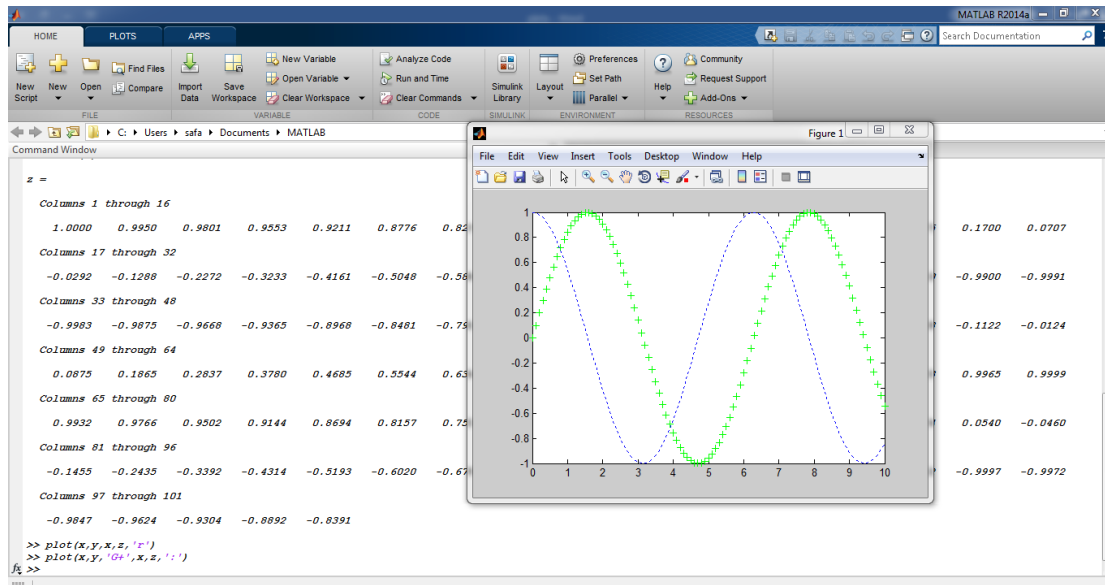
2-Method 2 to merge the drawings

It is about writing variables in one line within (plot)

```

plot(x,y,x,t)
>>x=(0:0.1:10)
>>y=sin(x)
>>z=cos(x)
>>plot(x,y,x,z,'r')
>>plot(x,y,'G+',x,z,':')

```



Statistical plots

Matlab supports statistical graphics and ratio by using two commands (**bar**,**pie**)

Bar(y)

pie(y)

Bar(x,y)

pie(x,y)

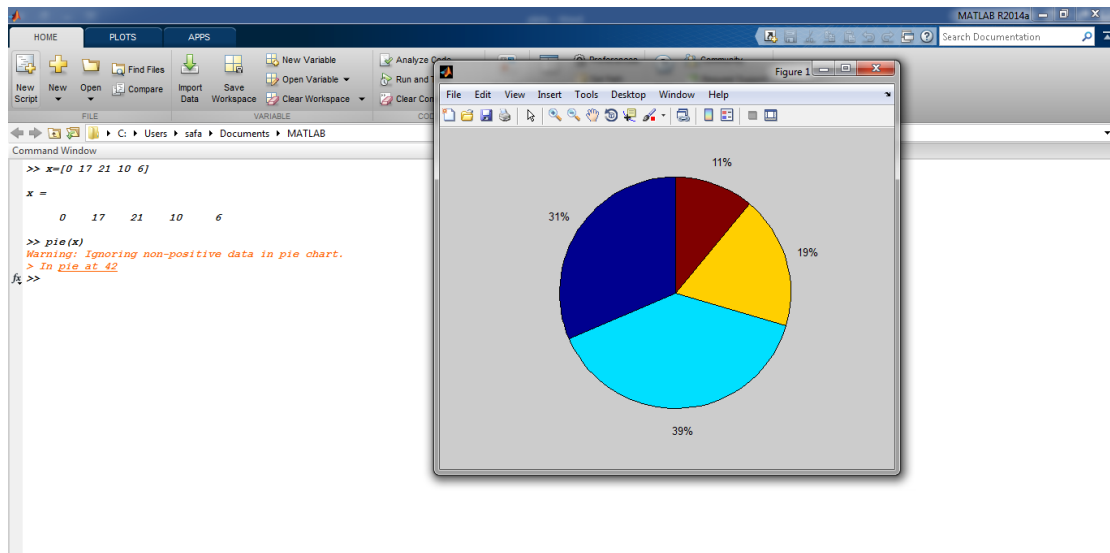
Example:

The distribution of grades for 70 students in the programming course is shown in the following research:

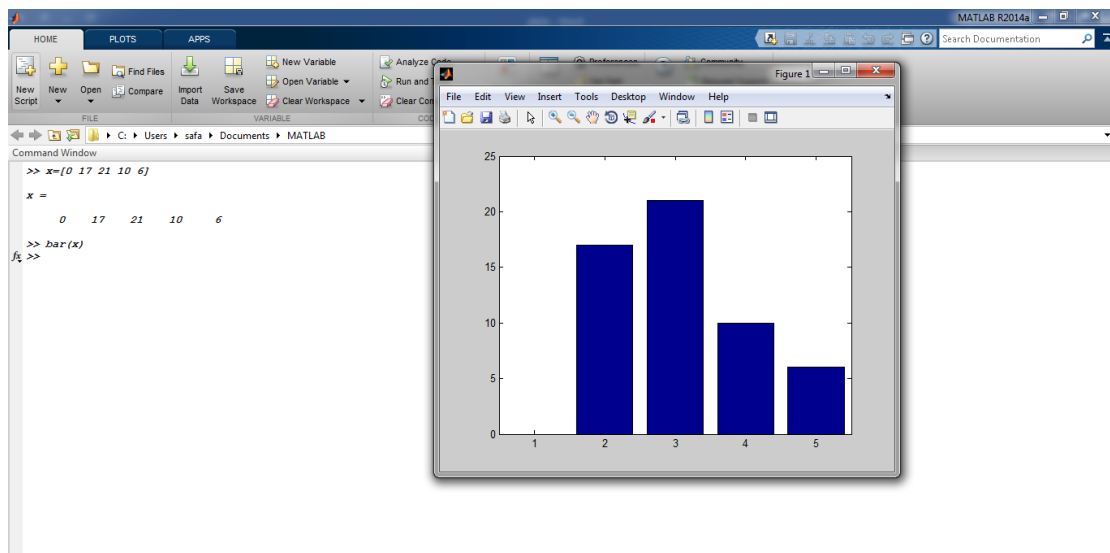
```

>>x=(0 17 21 10 6)
>>pie(x)

```



>>bar(x)



Plotting Discrete signals

Matlab analyzes signals, using the following command(**stem**)

In the Matlab it is the following picture:

Stem(y)

Stem(x,y)

Example:

We have a vector(x) as the number of points

The divider is 60 created by the intermittent signal of the function(sin(x))

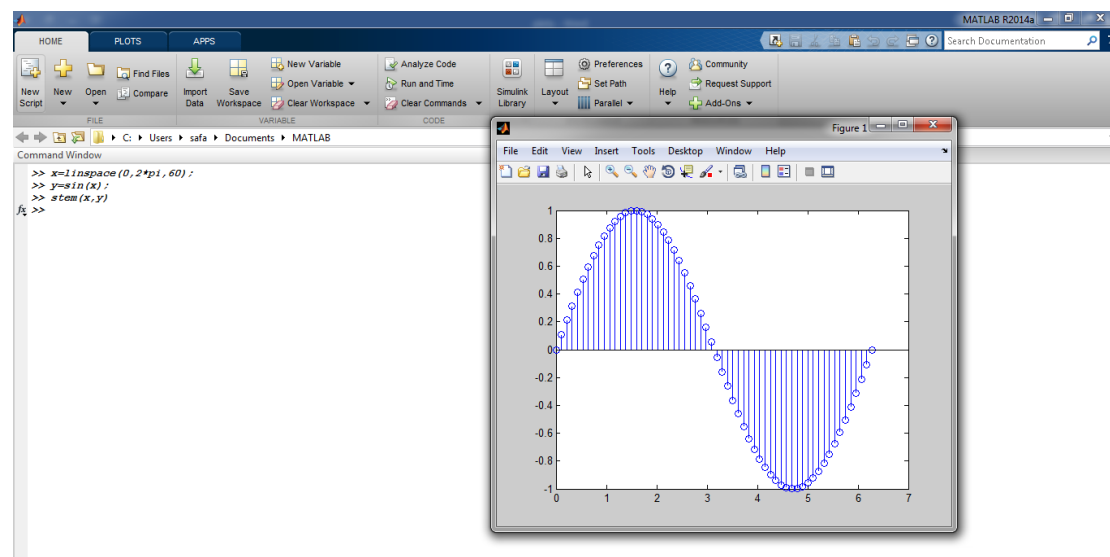
$X=[0,2\pi]$

Sol:

```
>>x=linspace(,2*pi,60);
```

```
>>y=sin(x);
```

```
>>stem(x,y)
```



Example:

How to draw a circle by Matlab?

```
>>t=(0:360);
```

```
>>r=5;
```

```
>>A=2;
```

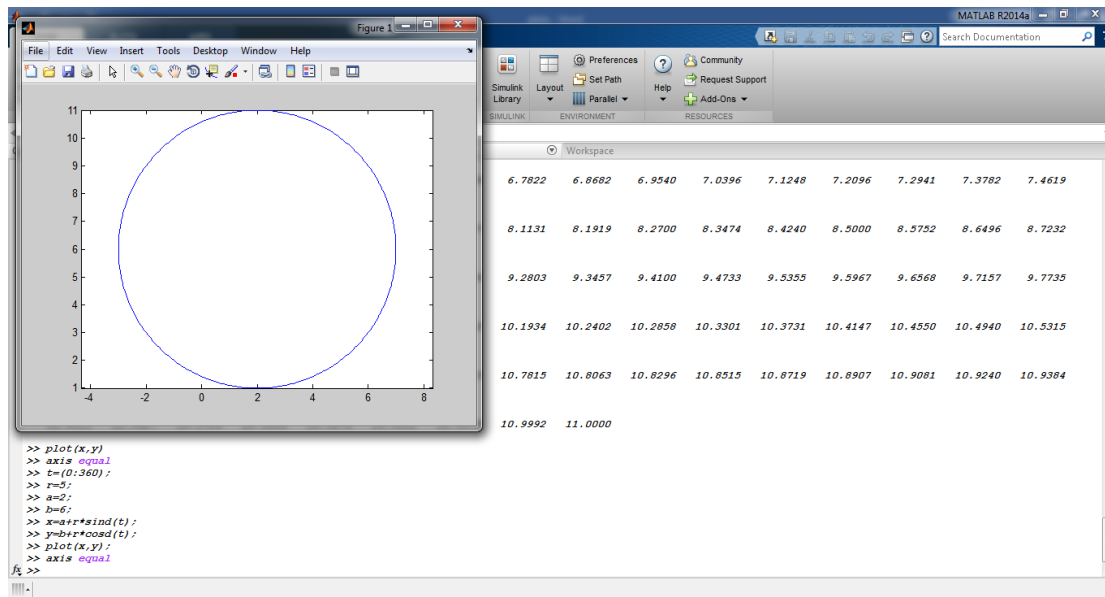
```
>>B=6;
```

```
>>X=a+r*sind(t);
```

```
>>y=b+r*cosd(t);
```

```
>>Plot(x,y);
```

```
>>axis equal
```



Three – Dimentional Graphics

Introduction 3-D plot

Basic Plots

A MATLAB surface is defined by the z coordinates associated with a set of (x,y) coordinates . for example , suppose we have the set of (x, y) coordinates:

$$(x, y) = \begin{pmatrix} 1,1 & 2,1 & 3,1 & 4,1 \\ 1,2 & 2,2 & 3,2 & 4,2 \\ 1,3 & 2,3 & 3,3 & 4,3 \\ 1,4 & 2,4 & 3,4 & 4,4 \end{pmatrix}$$

The points can be plotted as (x,y) pairs

The (x,y) pairs can be split into two matrices:

$$x = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix} \quad y = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 \end{pmatrix}$$

The matrix x varies along its rows and y varies down its column. We define the surface z :

$$z = \sqrt{x^2 + y^2}$$

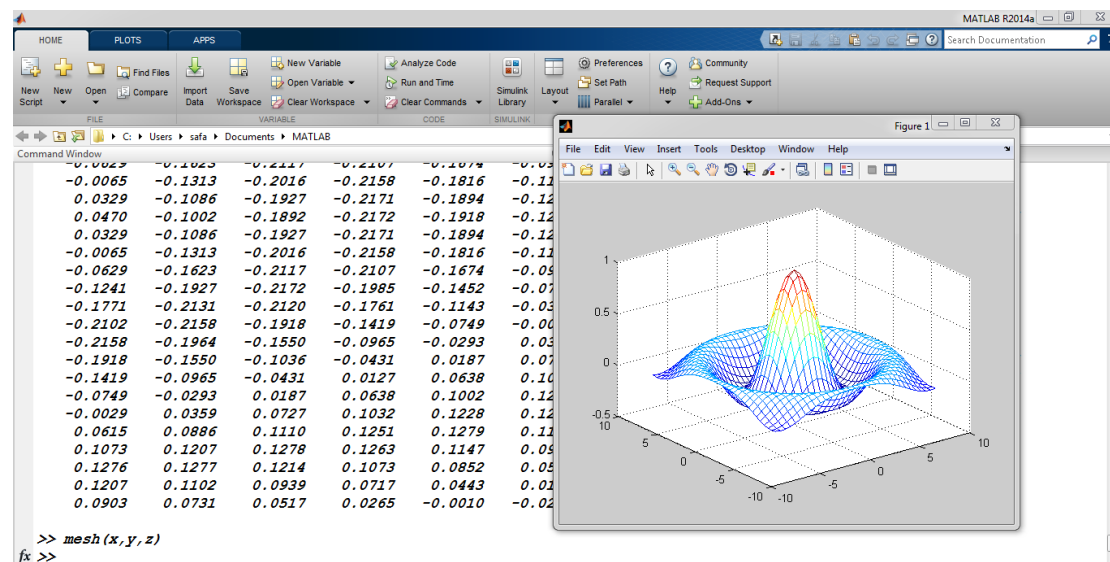
Which is the distance of each (x,y) point from the origin (0,0) . to calculate z in MATLAB for the x and y matrices given above, we

begin by using the mesh grid function, which generates the required , x and y matrices.

```
>> [x,y] = meshgrid (1:4)
```

Example Write a program to draw a 3-D plot of sinc function for x,y in the interval from -8 to 8 in step of 0.5.

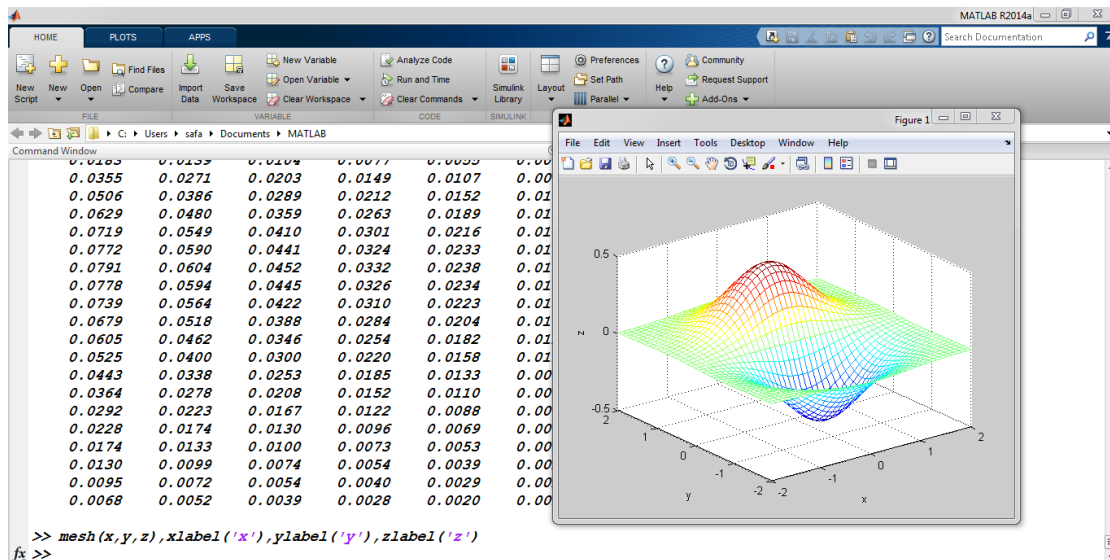
```
>>[x,y]=meshgrid(-8:.5:8);
>> r= sqrt (x.^2 + y.^2 ) + eps;
>> z = sin (r). / r;
>> mesh (x , y , z )
>> z= sin (r)./r
```



Example Plot the function

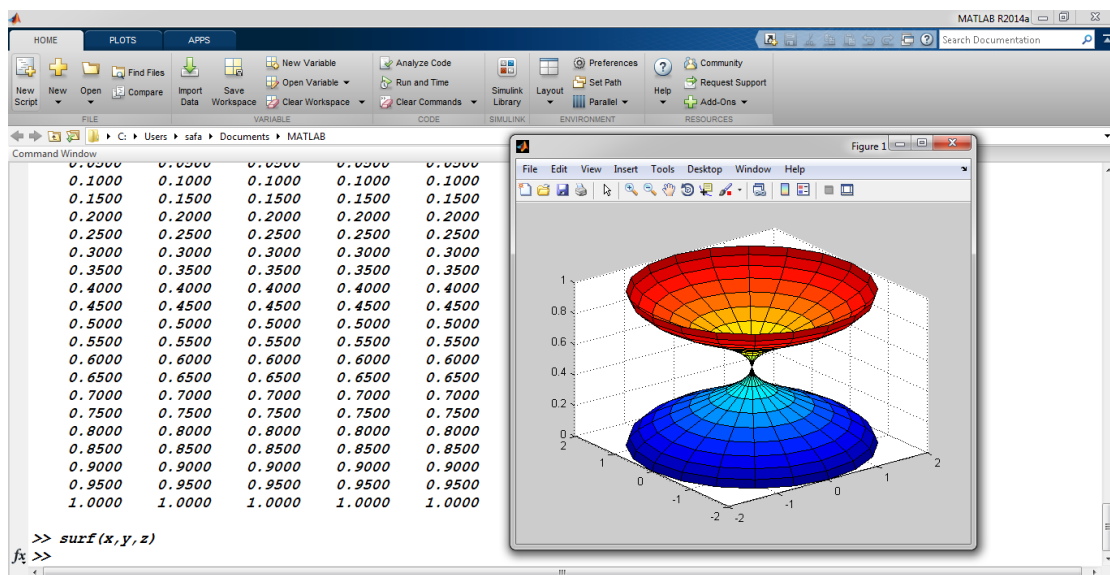
$$z = y e^{-x^2 - y^2}$$

```
>[x,y]=meshgrid(-2:0.1:2);
>>z=y.*exp(-x.^2-y.^2);
>> mesh(x,y,z),xlabel('x'),ylabel('y'),zlabel('z')
```



Example:-

```
>>t=0:pi/10:2*pi;
>>[x,y,z]=cylinder(1+cos(t));
>>surf(x,y,z)
```





DEPARTMENT OF
COMPUTER TECHNIQUES ENGINEERING

COMPUTER APPLICATIONS

FOR
2ND STAGE STUDENTS

fourth lecture

LECTURER

MSc. Safa Hussain

Input / Output of Variables in script file

- *script* is basically just a sequence of commands
- the same kind we enter into the command window
- scripts are stored as a plain text files with an .m extension
- so called *m-files* can also contain functions
- name of the script can be used as a command

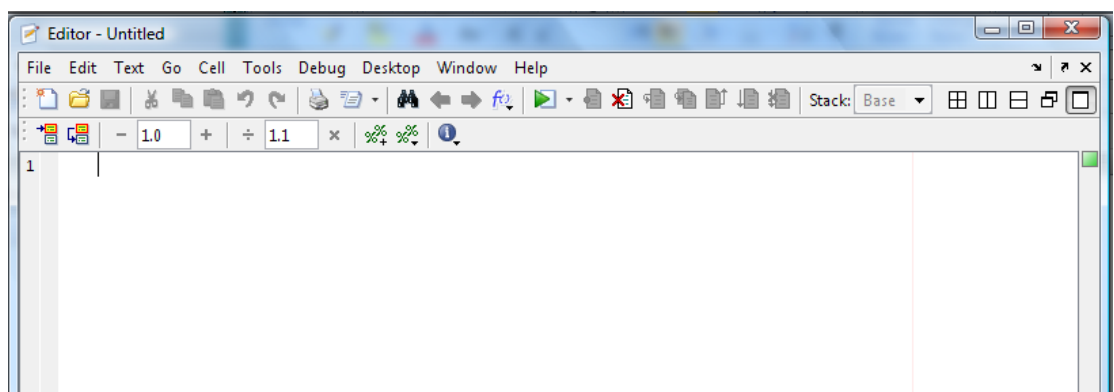
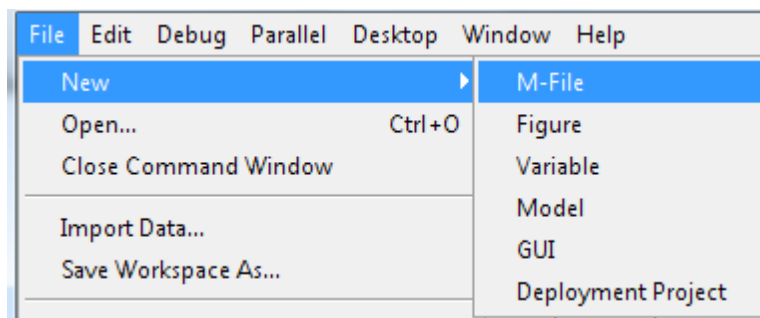
The syntax of the input command is :

variable = input ('a string displayed in the command window tells a user what to do')

As with any variable ,the variable and its assigned value will be displayed in the command window unless a semicolon is type at the input command.

To Input new M-file:

(1)



Example: Calculate the radical equation of the second order?

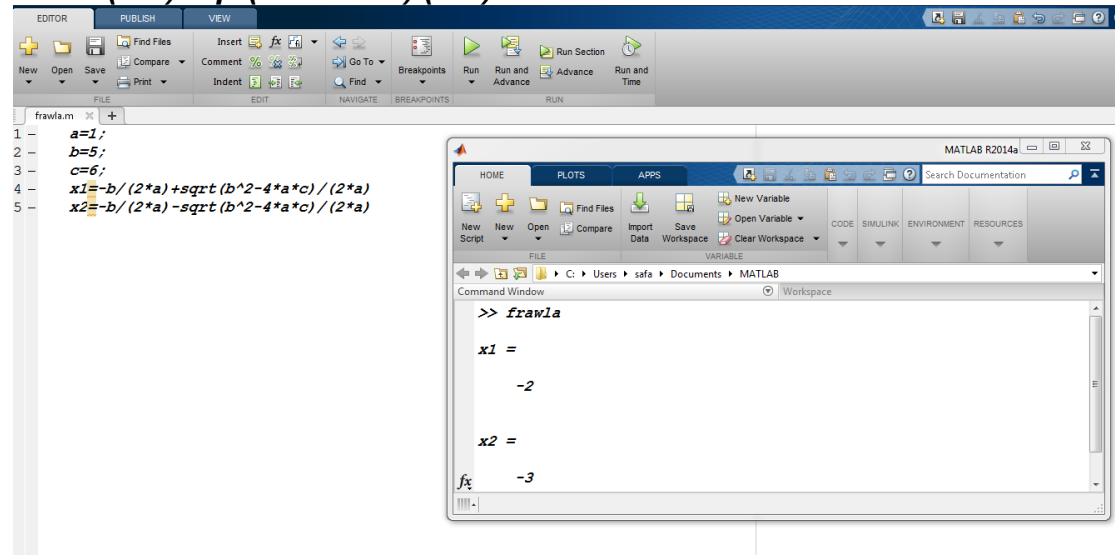
$a=1;$

$b=5;$

$c=6;$

$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$

$x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$



Input of Variable

variable = input ('a string displayed in the command window tells a user what to do')

Example: Finding roots of a second order equation?

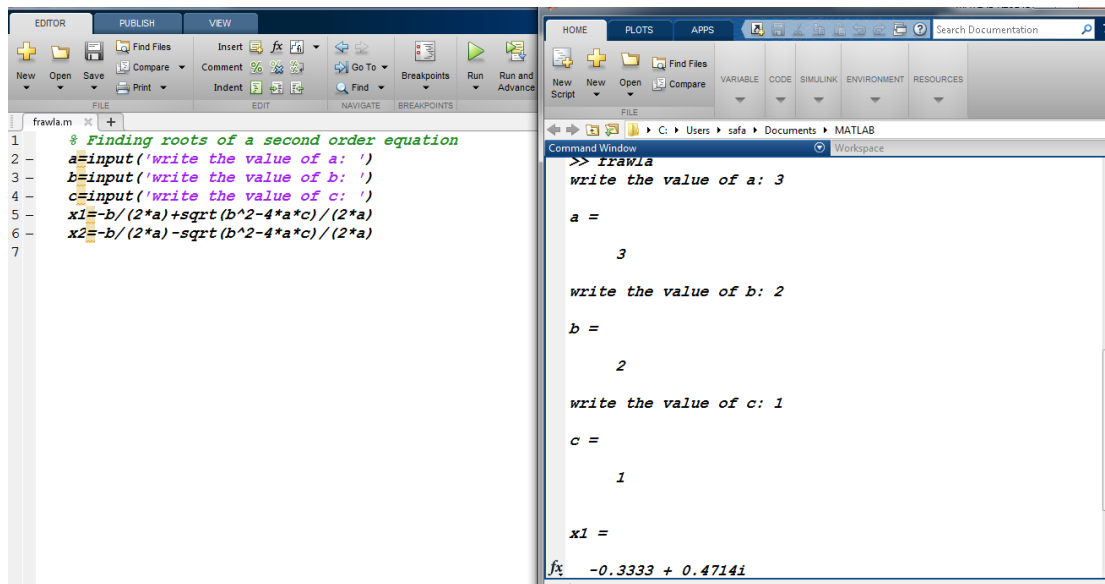
$a = \text{input}(\text{'write the value of a: '})$

$b = \text{input}(\text{'write the value of b: '})$

$c = \text{input}(\text{'write the value of c: '})$

$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$

$x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$



Input() function

- **asking for user input:**

```
>> a = input('Enter a number: ')
Enter a number: 4.7
a =
4.7000
```

 - the 4.7 was entered by keyboard
 - the rest is MATLAB output
- **The input is actually an expression:**

```
>> b = input('Enter an expression: ')
Enter an expression: 2*sin(pi/3)
b =
1.7321
```
- **even variables can be used:**

```
>> c = input('Enter an expression: ')
Enter an expression: 2 * b
c =
3.4641
```
- **so we can input vectors and matrices:**

```
>> A = input('Enter a matrix: ')
Enter a matrix: [ 1 2 3; 4 5 6; 7 8 9 ]
A =
1 2 3
4 5 6
7 8 9
```

- **and strings:**

```
>> s = input('Your name: ')
Your name: 'Thomas'
s =
Thomas
```
- **but more conveniently:**

```
>> t = input('Favorite color: ', 's')
Favorite color: blue
t =
blue
```

EXAMPLE: Calculates the area of a triangle in script file ?

ans:

% knowing the base and height and which are entered using the input command.

```
a=input('Enter the base of the triangle a=');
b=input('Enter the base of the triangle b=');
Triangle area =(a*b)/2
```

```
a=input('input the value of a');
b=input('input the value of b');
Area=0.5*a*b
```

Output of Variable

• **disp (x)**

displays the array (x), without printing the array name. In all other ways it's the same as leaving the semicolon off an expression except that empty arrays don't display.

disp(name of the variable) or disp('text as a string ')

If (x) is a string, the text is displayed.

```
>> x=[1 2 3];
>> x
x =
    1 2 3
>> disp(x)
    1 2 3
```


Example:

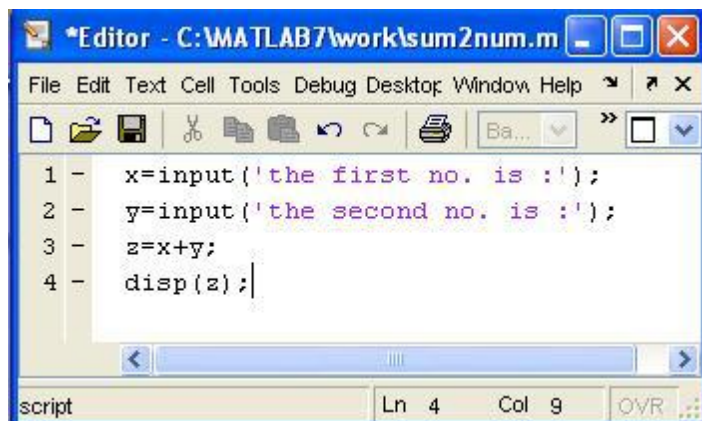
```
>> a=6;
>> b=a;
>> s='Ahmed has ';
>> w='Ali has ';
>> t=' Dinars';
>> disp([ s num2str(a) t]);
>> disp([ w num2str(b) t]);
```

the execution result is:
Ahmed has 6 Dinars
Ali has 6 Dinars

Example: To calculate the area and circumference of a circle?

```
r=input('The radius of the circle(r)= ');
A=pi*r*r;
C=2*pi*r;
disp(['The area of the circle = ',num2str(A)])
disp(['The circumference of the circle = ',num2str(C)])
```

The radius of the circle(r)= 6
The area of the circle = 113.0973
The circumference of the circle = 37.6991

Example: Write a program to calculate summation for two numbers?A screenshot of a MATLAB Editor window titled '*Editor - C:\MATLAB7\work\sum2num.m'. The window has a menu bar with 'File', 'Edit', 'Text', 'Cell', 'Tools', 'Debug', 'Desktop', 'Window', and 'Help'. Below the menu bar is a toolbar with icons for file operations and editing. The main text area contains the following MATLAB code:

```
1 - x=input('the first no. is :');
2 - y=input('the second no. is :');
3 - z=x+y;
4 - disp(z);
```

The status bar at the bottom shows 'script', 'Ln 4', 'Col 9', and 'OVR'.

OR


```

Editor - C:\MATLAB7\work\sum2num.m
File Edit Text Cell Tools Debug Desktop Window Help
1 - x=input('the first no. is :');
2 - y=input('the second no. is :');
3 - z=x+y
4 - z
script Ln 4 Col 2 OVR Pr

```

Example: Write a MATLAB program to find circumference of circle ($c=2\pi r$)?

r=input('enter the value of r :');

c=2*pi*r;

disp('the value is :')

disp(c)

the execution result is:

enter the value of r :4

the value is :

25.1327

Example: The solution of quadratic equation is given by quadratic formula :

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad ?$$

Sol, write a MATLAB to find roots x1 and x2.

a=input('enter the value of a :');

b=input('enter the value of b :');

c=input('enter the value of c :');

x1=(-b+(b^2-4*a*c)^0.5)/(2*a);

x2=(-b-(b^2-4*a*c)^0.5)/(2*a);

disp('the value of x1 :')

disp(x1)

disp('the value of x2 :')

disp(x2)

the execution result is:

enter the value of a :3

enter the value of b :2

enter the value of c :1

the value is x1 :

-0.3333 + 0.4714i

the value of x2 : -0.3333 - 0.4714i

Example: Program to convert temperature from Fahrenheit to Celsius?

```
Tf=input('Enter the temperature in Fahrenheit: ');  
Tc=(5/9)*(Tf-32);  
disp(['The temperature = ',num2str(Tc),' degree celsius'])
```

Homework:

(1) Write a program To calculate the area and perimeter of a rectangle?

(2) write a Program to calculate the area of a triangle?

(3) Write a MATLAB program to find Area of square?

.



DEPARTMENT OF
COMPUTER TECHNIQUES ENGINEERING

COMPUTER APPLICATIONS

FOR
2ND STAGE STUDENTS

Fifth lecture

Assistant Lecturer
Safa Hussain

Flow Control

Computer programming languages offer features that allow you to control the flow of command execution based on decision making structures. MATLAB has several flow control constructions:

- **if statement.**
- **switch and case statement.**
- **for statement.**
- **while statement.**

(1)if statement

The if statement evaluates a logical expression and executes a group of statements

when the expression is *true*. The optional *elseif* and *else* keywords provide for the execution

of alternate groups of statements. An *end* keyword, which matches the *if*, terminates the last group of statements.

if statement

if syntax:

if condition

command

command

...

command

end

Example:

```
x = input('Enter a number: ');  
if x > 0  
    disp('positive')  
end  
if x == 0  
    disp('zero')  
end  
if x < 0  
    disp('negative')  
end
```

(2)if-else statement

if-else syntax:

```
if condition  
command(s)  
else  
command(s)  
end
```

Example:

```
x = input('Enter a number: ');  
if x > 0  
disp('positive')  
else  
if x == 0  
disp('zero')  
else  
disp('negative')  
end  
end
```

Example:

```
x = input('Enter a number: ');  
if mod(x,2)==0  
disp('positive')  
disp('even')  
else  
disp('odd')  
end
```

(3)if-elseif statement

if-elseif syntax:

```
if condition  
command(s)  
elseif condition  
commmand(s)  
elseif condition  
command(s)
```

...

```
else  
command(s)  
end
```

Example:

```
A=input('A=');  
B=input('B=');  
if A > B  
'GREATER'  
elseif A < B  
'LESS'  
elseif A == B  
'EQUAL'  
else  
error ('Unexpected situation')  
end
```

Example:

```
course_mark=input('Mark = ');  
if (course_mark < 50 )  
course_grade='fail';  
elseif (course_mark >=60 && course_mark < 70)  
course_grade='pass';  
elseif (course_mark >=70 && course_mark < 80)  
course_grade='Good';  
elseif (course_mark >=80 && course_mark < 90)  
course_grade='Very Good';  
elseif (course_mark >=90 )  
course_grade='Excellent';  
end  
disp(['course_grade = ' course_grade]);
```

(4)The switch Statement

The general form of **switch** statement is:

```
switch exp

    case value1
        statement1

    case value2
        statement2

    ...

    Case valuen
        statementn

    otherwise
        Statementm

end
```

Example:

Write a Program to reads the number of the month and prints the number of days?

```
m=input('enter month:');
switch (m)
    case {2}
        disp('28 or 29');
    case {1,3,5,7,8,10,12}
        disp('31');
    case {4,6,9,11}
        disp('30');
    otherwise
        error('such month does not exist');
end
```

Assistant lecturer Safa Hussain

(5)for

```
For counter=first;step:last  
  
    statements  
  
end
```

Counter: The counter controlling the circuit.

First: The initial value of the meter.

Step: The amount of increase or decrease in the value of the meter in each cycle.

Last: The final value of the meter.

Example:

```
s=0;  
for r=1:3  
s=s+r;  
end  
s
```

output:

1+2+3=6

Example:

```
s=0;  
for r=1:2:9  
s=s+r;  
end  
s
```

output:

1+3+5+7+9

s=25

Example:Write aprogram to find factorial of the number(n)?

```
n=input('enter the value of factorial n:')
s=1;
for i=n:-1:1
    s=s*i;
end
s
```

output:

n=5

fact=120

Example:Write aprogram to find if the number is prime or not?

```
x=input('=number')
c=0;
for i=1:x
    if mod(x,i)==0
        c=c+1
    end
end
if(c==1)
    disp('number is prime')
else
    disp('number is not prime')
end
```

(6)While:

```
while exp
    statements
end
```

Example:Write aprogram to Find the largest common denominator of two numbers?

```
x=input('input the first no: ')
y=input('input the second no: ')
while (x~=y)
    if x>y
        x=x-y;
    else
        y=y-x;
    end
end
x
```

output:

x=6

y=8

The result:

x=2

Homework:

(1)Write a program to input student's degree and print the average?(use if)

(2)Write a Program takes the number of the month and prints in any of the seasons of the year there is this month?(use switch)

(3)Write a Program to calculate sum of cubes of positive numbers less than 6?(use for)

(4) Write a program to calculate factorial of numbers 1 through 10?(use for)

(5)Write a program to determine the triangle type(Equilateral, Isosceles, Scalene)(use elseif)

Assistant lecturer Safa Hussain