

Mat lab

Lecturer Wala'a Abd ul-Mageed

Introduction:

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 [1] 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 others of applied science and engineering..

Starting MATLAB

When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains *other* windows. The major tools within or accessible from the desktop are:

1. The Command Window
2. The Command History
3. The Workspace
4. The Current Directory
5. The Help Browser

The Start button

Operation	Symbol
Addition	+
Subtraction	-
Multiplication	*
Left division	\
Right division	/
Power	^
Transpose	'
Square root	Sqrt

Ex: Use Mat lab to find the following :

$$6\frac{10}{13} + \frac{18}{5(7)} + 5(9)^2$$

```
>> 6*(10/13)+18/(5*7)+5*9^2
```

Ans: 410.1297

Creating MATLAB variables

MATLAB variables are created with an assignment statement. The syntax of variable as- segment is

variable name = a value (or an expression)

For example,

```
>> x = expression
```

```
>> clear
```

The command `clear` or `clear all` removes all variables from the workspace. This frees up system memory. In order to display a list of the variables currently in the memory, type

```
>> who
```

Controlling the appearance of floating point number

MATLAB by default displays only 4 decimals in the result of the calculations, for example `;163.6667`, as shown in above examples. However, MATLAB does numerical calculations in *double* precision, which is 15 digits. The command `format` controls how the results of computations are displayed. Here are some examples of the different formats together with the resulting outputs.

Example:

```
>> format short
```

```
>> x=-163.6667
```

If we want to see all 15 digits, we use the command format long

```
>> format long
```

```
>> x= -1.636666666666667e+002
```

To return to the standard format, enter format short, or simply format.

The complex number: $z=x+iy$ _

real(z)	Real part of complex number
imag(z)	Imaginary part complex number
abs(z)	Absolute value
conj(z)	The conjugate
angle(z)	The angle

Mathematical functions:

MATLAB offers many pre-defined mathematical functions for technical computing which contains a large set of mathematical functions.

Typing `help elfin` and `help specfun` calls up full lists of *elementary* and *special* functions respectively. There is a long list of mathematical functions that are *built* into MATLAB. These

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

functions are called *built-ins*. Many standard mathematical functions, such as $\sin(x)$, $\cos(x)$, $\tan(x)$, $\ln(x)$, are evaluated by the functions `sin`, `cos`, `tan`, `exp`, and `log` respectively in Mat lab

In addition to the elementary functions, MATLAB includes a number of preened constant values. A list of the most common values is

<code>pi</code>	The π number, $\pi = 3.14159\dots$
<code>i, j</code>	The imaginary unit i , $\sqrt{-1}$
<code>Inf</code>	The infinity, ∞
<code>NaN</code>	Not a number

```
>> x=1;
```

```
>>y=exp(x)
```

```
>>y=
```

```
2.1783
```

```
>> 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
```

Matrix generation

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

For example,

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

v =

1. 4 7 10 13

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

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

w =

1

4

7

10

13

Thus, $v(1)$ is the first element of vector v , $v(2)$ its second element, and so forth. Furthermore, to access *blocks* of elements, we use MATLAB's colon notation (:). For example, to access the first three elements of v , we write,

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

```
ans =
```

```
1    4    7
```

Example:

1. Given $x = [3 \ 1 \ 5 \ 7 \ 9 \ 2 \ 6]$, explain what the following commands "mean" by by summarizing the net result of the command.

a. `x(3)`

b. `x(1:7)`

c. `x(1:end)`

d. `x(6:-2:1)`

sol:

a. `>> x(3)`

5

a. `>> x(2 : 7)`

1. 5 7 9 2 6

a. `>>x(1:end)`

3. 1 5 7 9 2 6

a. `x(6:-2:1)`

1. 2 7 1

Entering a matrix:

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

²begin with a square bracket, [

²separate elements in a row with spaces or commas (,)

²use a semicolon (;) to separate rows

²end the matrix with another square bracket,].

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

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

type,

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

MATLAB then displays the 3×3 matrix as follows,

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

- $A(:, j)$ is the j th column of A , while
- $A(i, :)$ is the i th row, and
- $A(\text{end}, :)$ picks out the last row of A .

Example In a previous example

```
>>A(2,:)
```

```
ans
```

```
4 5 6 >>
```

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

```
ans          3 6 9 >>
```

```
A(2,2)
```

```
ans          5
```

Linear spacing

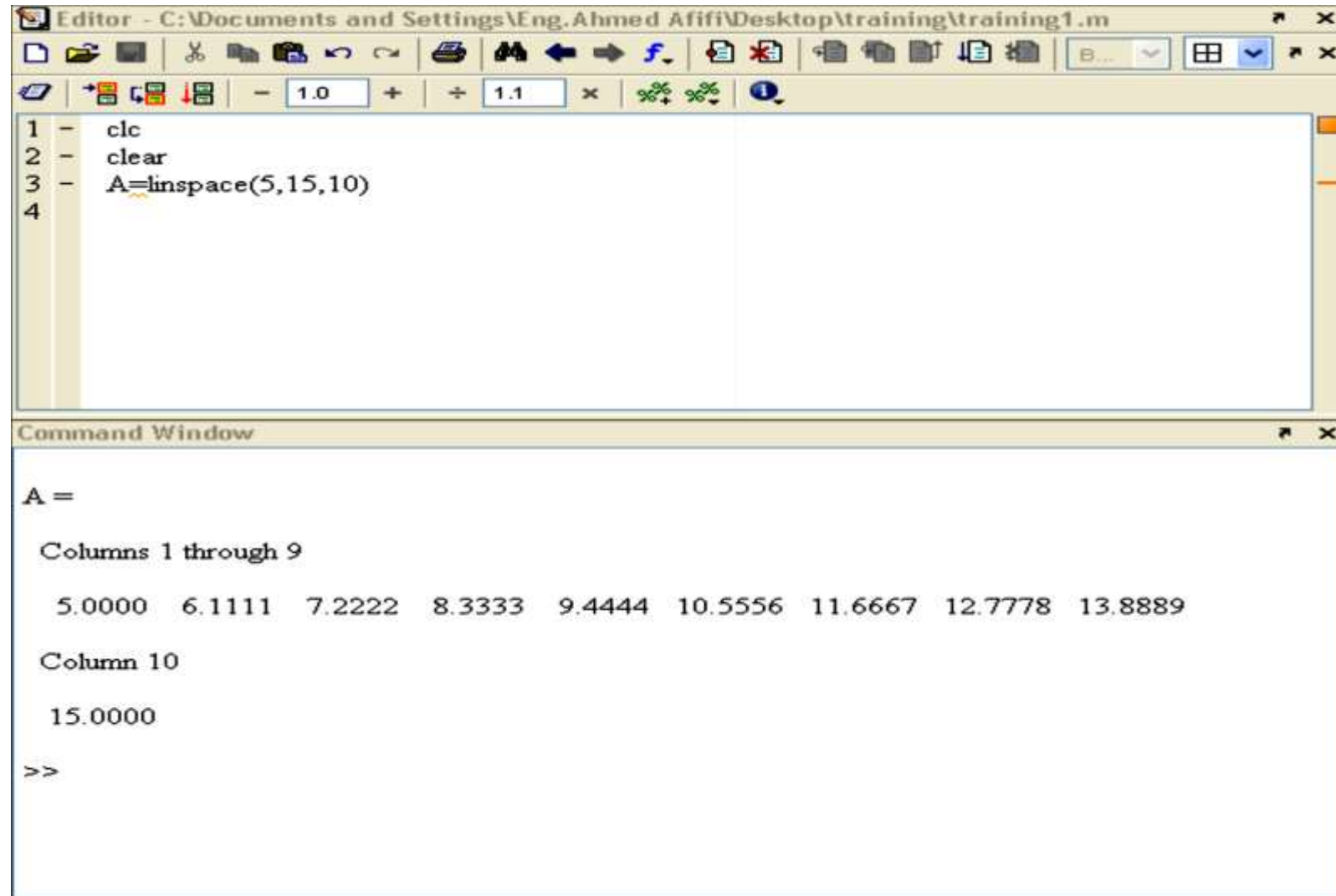
On the other hand, there is a command to generate linearly spaced vectors: `linspace`. It is similar to the colon operator (`:`), but gives direct control over the number of points. For example,

$$y = \text{linspace}(a,b)$$

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

y = linspace(a ,b, n)

`linspace(minimum number,maximum number,number of points in between)`



The image shows a MATLAB environment with an Editor window and a Command Window. The Editor window contains a script with four lines of code: `clc`, `clear`, `A=linspace(5,15,10)`, and a blank line. The Command Window displays the output of the script, showing the matrix `A` with 10 columns. The first 9 columns are displayed in a single row, and the 10th column is displayed in a separate row.

```
1 - clc
2 - clear
3 - A=linspace(5,15,10)
4
```

Command Window

```
A =
Columns 1 through 9
    5.0000    6.1111    7.2222    8.3333    9.4444   10.5556   11.6667   12.7778   13.8889
Column 10
   15.0000
>>
```

Colon operator in a matrix

$A(i,j)$	Location of element in row and column
$A(i,:)$	The row of matrix
$A(:,j)$	The column of matrix
$A(i:k,:)$	Specifies rows i to K
$A(:,j:k)$	Specifies columns j to k
$A(:, :)$	Replace the matrix it selves
$A(:)$	Replace the matrix by one column
$A(i,:)=[]$	Delete the row which number i
$A(:,j)=[]$	Delete the column which number j

Set up any matrix of 3*3 with named S and find:

a- the first row of a matrix .

b- the third column of a matrix.

c- the 6th element in a matrix .

d- write the matrix by one column.

e- delete the first row of a matrix.

```
>> S=[ 2 3 5; -4 5 0; 6 7 9]
```

a- S(1,:)

```
>> ans
```

```
>> 2 3 5
```

```
2 3 5
-4 5 0
6 7 9
```

b- >>S(:,3)

```
>> ans
```

```
>> 5 0 9
```

Matrix operations:

size(A)	The dimension of matrix
diag(A)	Main diagonal
det(A)	Determinate of matrix
inv(A)	Inverse of matrix
sum(A)	Summation of matrix(sum of each column)
A'	The transpose
sum(A')'	Summation of matrix(sum of each row)
Prod(A)	Product of each column
max(A)	Maximum value of each column
min(A)	Minimum value of each column
mean(A)	Average of each column
sort(A)	Sort in a seconding order
fliplr(A)	Flip the matrix from left to right
flipud(A)	Flip the matrix from up to down
tril(A)	Extract the lower part of matrix
triu(A)	Extract the upper part of matrix

Let b= [1 2 3; 4 5 6; 7 8 9]

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

>>b=

1	2	3
4	5	6
7	8	9

>> a= size(b)

3 3

>> d= sum(b)

>> d=

12 15 18

>> e= b'

>>e=

1	4	7
2	5	8
3	6	9

>> f= max(b)

>> f=

7 8 9

```
>> t=fliplr(b)
```

```
>> t=
```

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

```
>> h=tril(b)
```

```
>> h=
```

```
1  0  0  
4  5  0  
7  8  9
```


Matrix generators:

MATLAB provides functions that generate 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

Matrix arthematics oprations :

As we mentioned earlier, MATLAB allows arithmetic operations: +, -, *, and ^ to be carried out on matrices. 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 $A*A$

$a*A$ or $A*a$ multiplies each element of A by

Array arithmetic operations

On the other hand, array arithmetic operations or *array operations* for short, are done *element-by-element*. The period character, ., distinguishes the array operations from the matrix operations. However, since the matrix and array operations are the same for addition

(+) and subtraction (/), the character pairs (:+) and (:/) are not used. The list of array operators is shown below. If A and B are two matrices of the same size with elements $A = [a_{ij}]$ and $B = [b_{ij}]$, then the command

<code>.*</code>	Element-by-element multiplication
<code>./</code>	Element-by-element division
<code>.^</code>	Element-by-element exponentiation

```
>> C = A.*B
```

produces another matrix C of the same size with elements $c_{ij} = a_{ij}b_{ij}$. For example, using the same 3×3 matrices,

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

we have,

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

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

Solving linear equations:

For example, consider the following system of linear equations

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

The coefficient matrix A is

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \text{and the vector } b = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

With matrix notation, a system of simultaneous linear equations is written

$$Ax = b$$

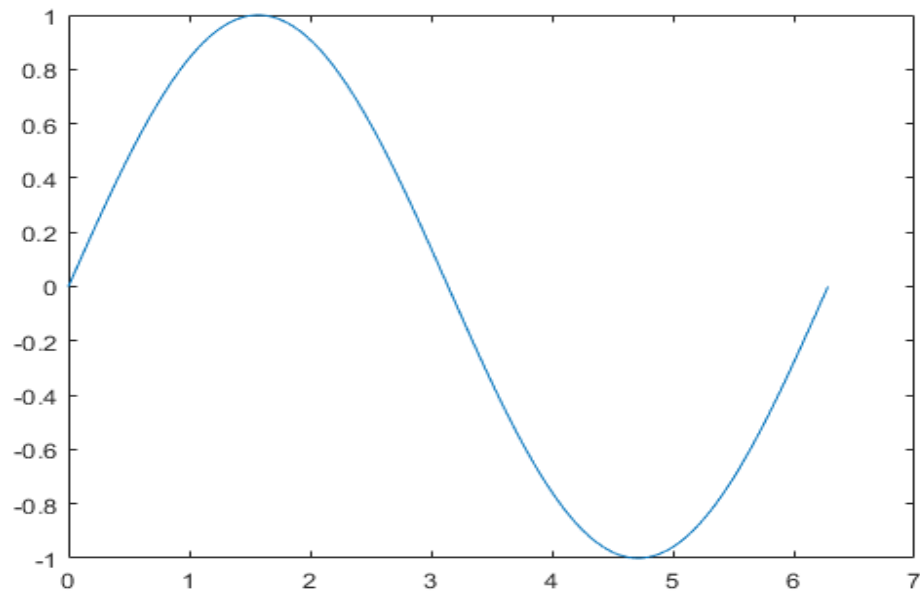
This equation can be solved for x using linear algebra. The result is $x = A^{-1}b$.

There are typically two ways to solve for x in MATLAB:

```
>> A = [1 2 3; 4 5 6; 7 8 0];  
>> b = [1; 1; 1];  
>> x = inv(A)*b  
x =  
    -1.0000  
     1.0000  
    -0.0000
```

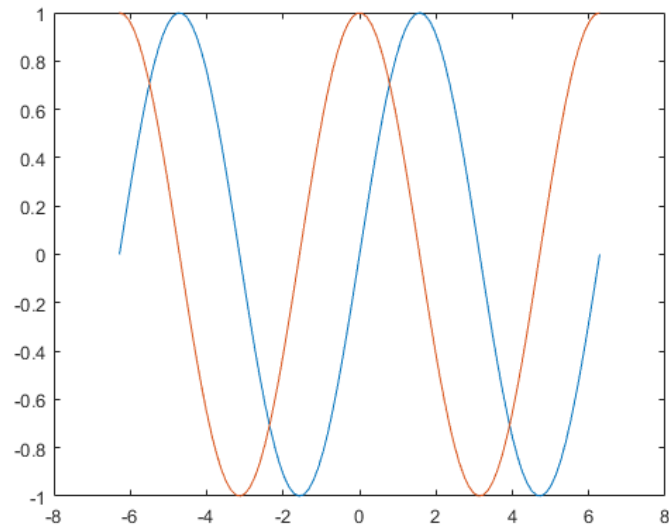
plot2D:

```
x = 0:pi/100:2*pi;  
y = sin(x);  
plot(x,y)
```



Plot two functions:

```
x = linspace(-2*pi,2*pi);  
y1 = sin(x);  
y2 = cos(x);  
plot(x,y1,x,y2)
```



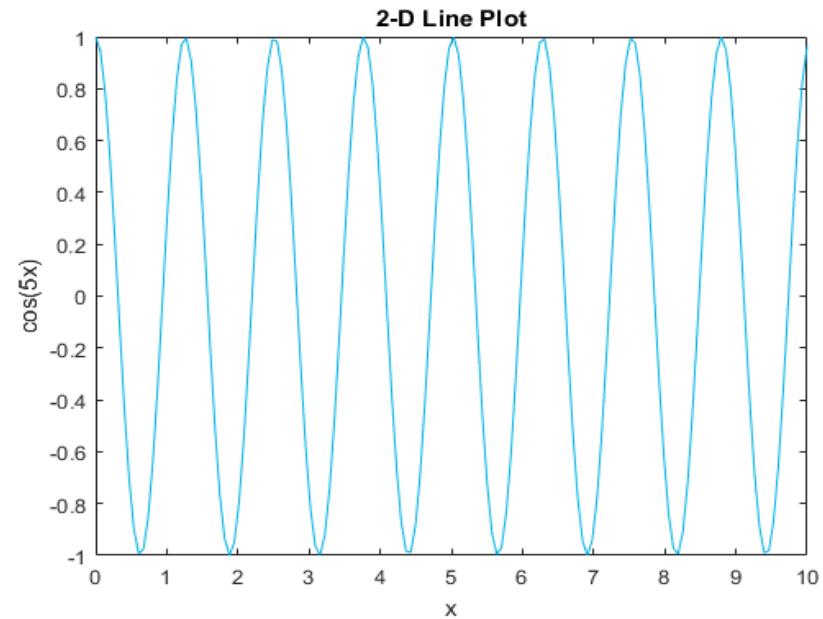
Insert title and labels :

```
x = linspace(0,10,150);
```

```
y = cos(5*x);
```

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

```
title('2-D Line Plot')xlabel('x')ylabel('cos(5x)')
```



Color	Discerption	Line style	Discerption
y	yellow	-	Solid line
m	magenta	--	Dashed line
c	cyan	:	Dotted line
r	red	-.	Dash dot line
g	green		
b	blue		
w	white		
k	black		

Marker	Description
o	circle
+	Plus sign
*	Asterisk
.	Point
x	Cross
s	Square

Insert color – style line –and marker

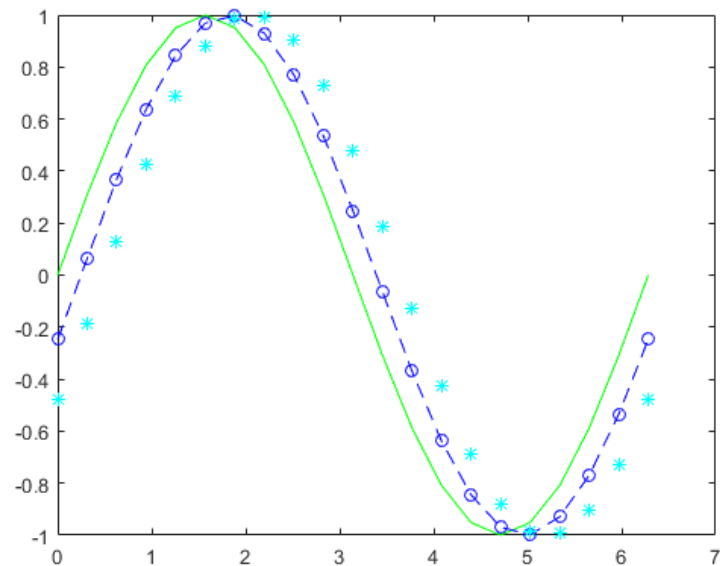
```
x = 0:pi/10:2*pi;
```

```
y1 = sin(x);
```

```
y2 = sin(x-0.25);
```

```
y3 = sin(x-0.5);
```

```
plot(x,y1,'g',x,y2,'b--o',x,y3,'c*')
```



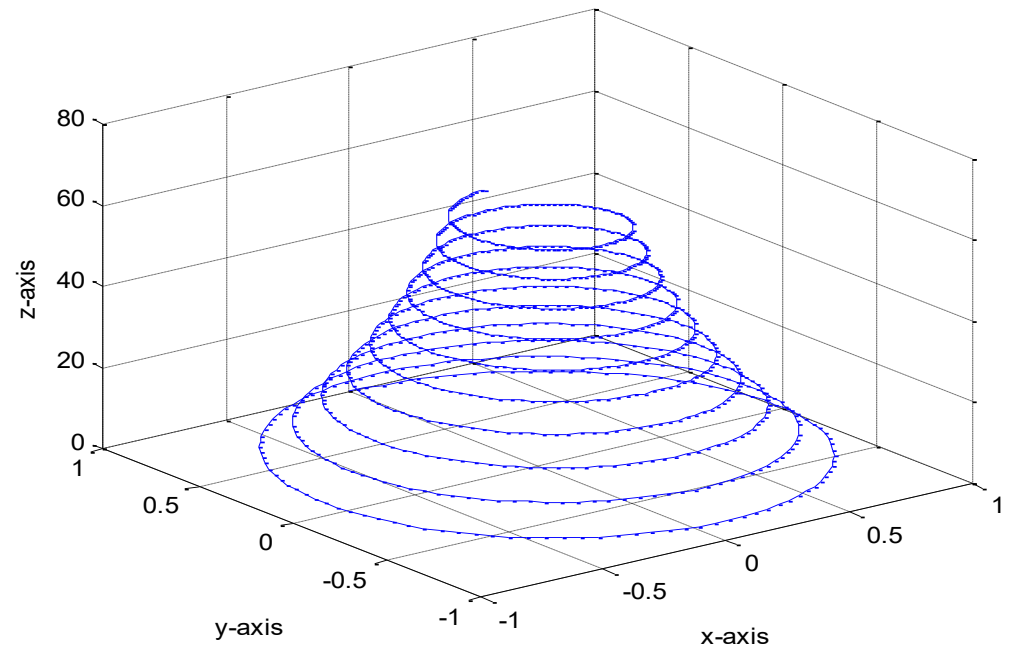
Plot 3D

plot3	three- Plots dimensional graph of the trajectory of a set of three parametric equations $x(t)$, $y(t)$, and $z(t)$ can be obtained using <code>plot3(x,y,z)</code>
meshgrid	If x and y are two vectors containing a range of points for the evaluation of a function, <code>[X,Y] = meshgrid(x, y)</code> returns two rectangular matrices containing the x and y values at each point of a two-dimensional grid.
mesh(X,Y,z)	If X and Y are rectangular arrays containing the values of the x and y coordinates at each point of a rectangular grid , and if z is the value of a function evaluated at each of these points, <code>mesh(X,Y,z)</code> will produce a three-dimensional perspective graph of the points. The same results can be obtained with <code>mesh(x,y,z)</code> can also be used.

Surf

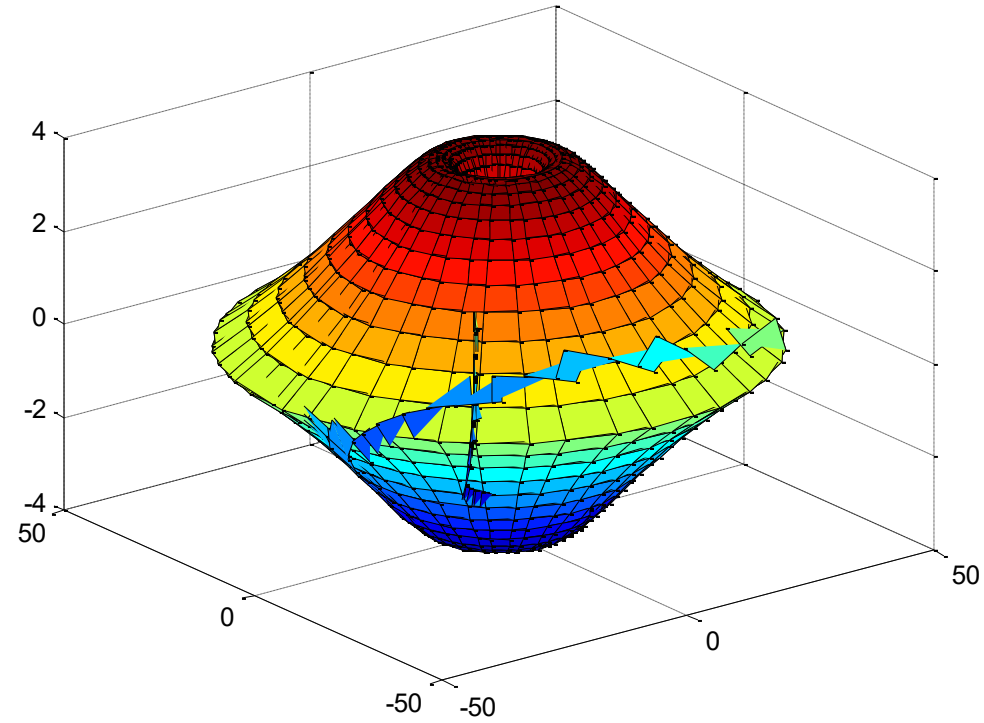
Produces a three-dimensional perspective drawing. Its use is usually to draw surfaces, as opposed to plotting functions, although the actual tasks are quite similar. The output of surf will be a shaded figure. If row vectors of length n are defined by $x = r \cos \theta$ and $y = r \sin \theta$, with $0 \leq \theta \leq 2\pi$, they correspond to a circle of radius r . If r is a column vector equal to $r = [0 \ 1 \ 2]'$; then $z = r * \text{ones}(\text{size}(x))$ will be a rectangular, $3 \times n$, arrays of 0's and 2's, and `surf(x, y, z)` will produce a shaded surface bounded by three circles; i.e., a

```
t=0:pi/50:20*pi;  
plot3(exp(-0.02*t).*sin(t).exp(0.02*t).*cos(t),t,'r'),  
xlabel(x-axis),ylabel(y-axis),zlabel(z-axis), grid
```



EX:

```
x=0:pi/40:2*pi;  
polar(x, sin(2*x));
```



EX:

$r=4$;

$u=\text{linspace}(0,5*\pi,50)$;

$v=\text{linspace}(-\pi,\pi,50)$

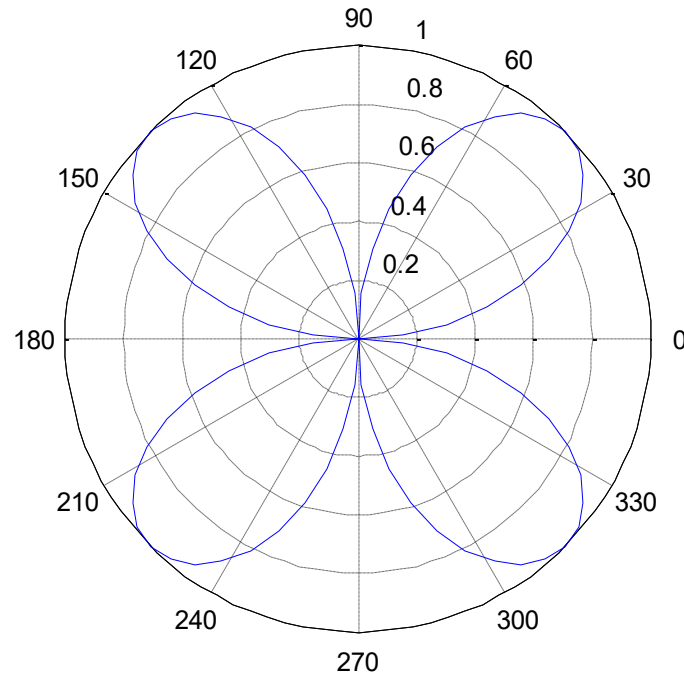
$[U,V]=\text{meshgrid}(u,v)$;

$x=r*\cosh(v).*\cos(u)$;

$y=r*\cosh(v).*\sin(u)$;

$z=r*\sin(v)$;

$\text{surf}(x,y,z)$



المتغيرات

يستخدم في هذا النظام التعابير الحرفية لتعريف الدوال , عملية التعامل مع الحروف بدل الارقام تسمى symbolic method
لنفرض ان لدينا الدوال الاتية

1. $F(x) = 2x^2 + 4x - 1$
2. $g(x,y) = xy^3 + yx^2$

فعملية التعريف تكون بالشكل الاتي

1. `syms x`
`f='2*x^2 + 4 * x - 1'`
2. `syms x y`
`g=' x.*y^3 + y.* x^2`

ان اهم الايعازات المستخدمة في نظام الماتلاب لايجاد الحل للدالة $f(x)=0$ اي إيجاد جذور هذه الدالة هي

1. الايعاز solve حيث يستخدم هذا الايعاز لايجاد جذر الدالة و يكون بالصيغة الاتيه

Solve ('fun')

Ex: find the roots of $f=x^2 -4x +4$

Sol:

Syms x

$F='x^2 - 4 * x + 4 '$

Solve (f)

Ans= -2

يمكن الحصول على المشتقة للدوال من خلال الايعاز التالي $df=diff(f)$

$$\text{Ex: } f=2*x^2 + 4 * x - 1'$$

$$df= diff(f)$$

$$df = 4*x+4$$

$$\text{Ex: } f(x,y) = x^2y + x^2 + y^2 \quad \text{use mat lab to find } \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \text{ and } \frac{\partial^2 f}{\partial x \partial y}$$

Sol: H.W

المشتقات من الرتب العليا

يمكن ذلك باستخدام الايعاز $diff$ والذي يكون معرف بالصيغة $diff(f,n)$

لو اردنا المشتقة الثالثة للمثال السابق

$$df3=diff(f,3)$$

$$df3=0$$

اما اذا كانت الدالة تعتمد على اكثر من متغير $diff(f,'x',n)$ حيث n عدد مرات الاشتقاق

التكامل

التكاملات الغير محددة

يمكن ايجاد التكامل الغير المحدد للدوال باستخدام الايعاز $\text{int}(f)$

$$\text{Ex: find } \int (x^2 + 2) dx$$

$$F = 'x^2 + 2'$$

$$\text{int}(f)$$

$$\frac{1}{3} * x^3 + 2 * x$$

$$\text{Ex: } s = \int (x^2 y + 2y) dy$$

$$g = 'x^2 * y + 2 * y'$$

$$s = \text{int}(g, 'y')$$

$$\frac{1}{2} x^2 * y^2 + y^2$$

العلاقات : ان اهم العلاقات المستخدمة في هذا النظام هي

1. < : اقل less than

2. > : اكبر greater than

3. ≤ : اقل او يساوي less or equal

4. ≥ : اكبر او يساوي greater or equal

5. == : يساوي equal

6. ~= : لايساوي not equal

العوامل المنطقية

1- & And

2- | Or

3- ~ Not

جمل الادخال

يمكن ادخال قيم المتغيرات و المصفوفات المعرفة في البرنامج عن طريق الادخال المباشر او عن طريق جملة الادخال input والتي تعرف بالشكل الاتي :

Variable= input ('تعليق');

X= input('enter value of x=')*

جمل الاخراج out put

هنالك طريقتين لاجراج النتائج المحسوبة داخل البرنامج وذلك عن طريق مساحة الابعازات فيكتب اسم المتغير المراد معرفة قيمته فقط ثم نضغط enter

اما الطريقة الثانية وذلك باستخدام الابعاز disp الذي يستخدم للطباعة وكماياتي :

disp (variable)

disp(x)

ويمكن طباعة اكثر من متغير واحد بشرط وضع []

disp([x y]) كذلك يمكن طباعة جملة تعريف داخل البرنامج

disp('the value of matrix')

الايعايات الشرطيه

if logical expression : Ifend

Statement

end

ex: write a program to find the value of $y = x + \sin x$ if x less than 5

sol:

```
x= input ('enter value of x=');
```

```
if x< 5
```

```
y= x+ sin(x)
```

```
end
```

```
disp(y)
```

ex: write a program to find the roots of equation $ax^2 + bx + c = 0$ where this equation has real roots H.W

الصيغة if...else...end وتأخذ الصيغة الآتية

If logical expression

Statement 1

else

Statement 2

End

Ex: write program to find the roots of equation $ax^2 + bx + c = 0$ where this equation has equal roots or not have

Sol:

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

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

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

```
if(b^2-4*a*c==0)& a~= 0
```

```
x=-b/2*a;
```

```
else
```

```
disp('this equation has no roots')
```

```
end
```

Ex: write program to solve the system $AX=b$ if A a square matrix else find the determent of A H.W

الصيغة if...else if... else....end وتأخذ الشكل التالي :

If logical expression 1

Statement 1;

Else if logical expression 2

Statement 2;

Else if logical expression 3

Statement 3;

End

:

$$\left\{ \begin{array}{ll} n < 0 & \text{display input must be positive} \\ n > 0 \text{ and even} & A = n/2 \\ n > 0 \text{ and odd} & A = n + 1/2 \end{array} \right\}$$

Sol:

```
n= input('enter n=');
```

```
if( n<0)
```

```
disp('input must be positive');
```

```
else if rem(n,2)==0
```

```
A=n/2;
```

```
else
```

```
A=(n+1)/2;
```

```
end
```


العدادات في نظام الماتلاب

For وتأخذ الصيغة التالية :

For variable=initial : final

For variable= initial : step: final

Ex: write a program to find (n!)

```
n=input('n=');
```

```
f=1
```

```
for i=1:n
```

```
f=f*i;
```

```
end
```