CHƯƠNG
Mở Đầu

GIỚI THIỆU VỀ MATLAB

1. MATLAB là gì ?

MATLAB là ngôn ngữ lập trình cấp cao cho mục đích tính toán kỹ thuật, Chương trình tích hợp tính toán, đồ họa và lập trình trong một môi trường để sử dụng nơi mà tất cả các vấn đề và lời giải được biểu diễn dưới dạng các ghi chú toán học.

MATLAB là một hệ thống tương tác mà tất cả các phần tử dữ liệu cơ bản là một mảng không yêu cầu về mặt kích thước. Đây là một thuận lợi cho phép người sử dụng giải quyết các bài toán trong kỹ thuật đặc biệt là các công thức được xây dựng từ ma trận hay là vectơ.

2. *MATLAB System:*

MATLAB gồm năm thành phần chính sau:

2.1 Development Environment: Tập hợp các công cụ và những tiện ích cho phép người dùng sử dụng các ***hàm*** và ***file*** Matlab. Phần lớn các công cụ này là giao tiếp người dùng bao gồm: Matlab Desktop và Command Window, Command History, Edit and Debugger, Workspace, trình duyệt hổ trợ sử dụng help.

2.2 The MATLAB Mathematical Function Library: Đây là thư viện tập hợp các giải thuật tính toán được tập hợp trong các hàm từ cơ bản như: sum, sin, cosine, và số phức cho đến các hàm phức tạp hơn như: nghịch đảo ma trận, trị riêng và véc tơ riêng của ma trận, biến đổi Fourier,...

2.3 Lập trình trong môi trường MATLAB: Lập trình với Matlab. Matlab là ngôn ngữ lập trình cấp cao thực thi theo các đoạn lệnh, các hàm, cấu trúc dữ liệu, ... cho phép xây dựng các chương trình từ đơn giản, nhỏ cho đến các chương trình lớn, phức tạp.

2.4 Graphics: Matlab đã được mở rộng các khả năng cho việc thể hiện vector và ma trận dưới dạng đồ thị cũng như ký hiệu và in ấn chúng. Matlab cung cấp các hàm nâng cao cho việc thể hiện dữ liệu hai, ba chiều, xử lý ảnh, chuyển động, đồ thị. Ngoài ra Matlab còn cung cấp cho người sử dụng các hàm giao tiếp giữa người dùng và ứng dụng.

2.5 The MATLAB Application program Interface(API): Đây là thư viện cho phép người dùng tạo ra những chương trình bằng ngôn ngữ C hoặc Fortran tương tác với MATLAB.

2.6 Sử dụng tài liệu trợ giúp của Matlab: Matlab cung cấp tài liệu hướng dẫn dưới dạng bảng in và Web để hướng dẫn sử dụng tất cả các modul có trong Matlab.

2.7 Nội dung và mục đích khoá học: Khóa học này cung cấp cho người học những kỹ năng khai thác, sử dụng và lập trình với Matlab ở mức độ căn bản.

TỔNG QUAN MÔI TRƯỜNG LÀM VIỆC CỦA MATLAB:

Khởi động và thoát khỏi Matlab

Khởi động Matlab:

Trên cửa sổ Window nhấp đúp lên biểu tượng của Matlab 

Vào Start -> All Programs -> Matlab release 12 -> Matlab 12.
Sau khi khởi động ứng dụng Matlab sẽ được mở, giao diện chính của Matlab như hình
Thoát khỏi Matlab:
Để thoát khỏi ứng dụng Matlab có thể chọn Exit Matlab từ menu File trên Desktop hoặc nhập quit trong Command Window

Giới thiệu môi trường làm việc và các công cụ của Matlab

Giới thiệu khái quát giao diện làm việc chính của Matlab
Khi khởi động Matlab giao diện chính sẽ mở ra như hình, giao diện này chứa các công cụ (giao tiếp người dùng) để quản lý các tập tin, biến và các ứng dụng phụ trợ khác.
Command window.
Help Browser.
Current Directory Browser.
Command history.
WorkSpace Browser.
Editor/Debugger.
Profiler.
Một vài hàm cơ bản khi bắt đầu với Matlab

exit, quit	Thoát khỏi Matlab
finish	Kết thúc Mfile
matlabrc	
startup	

Hàm liên quan đến Command Window

clc : Xoá các dòng lệnh ở command window
diary: Lưu các lệnh thực thi ở command window
dos: Truy xuất DOS command và trả về kết quả
format: Định dạng thể hiện kiểu xuất dữ liệu
home : Di chuyển con trỏ lên vị trí phía trên bên trái của cửa sổ Command Window

Các hàm tìm sự trợ giúp của Matlab

doc: Hiển thị cửa sổ trợ giúp sử dụng MATLAB
demo: Truy suất demo thông qua Matlab Help Browser

docroot: Hiển thị đường dẫn xác định document
whatsnew: Thuộc tính mới trong phiên bản này so với phiên bản trước
help : Truy xuất trợ giúp của Matlab

Workspace

workspace: Hiển thị workspace
who, whos: Hiển thị trong workspace
clear, clear all: Xoá các biến trong Workspace

File

cd: Thay đổi đường dẫn
delete: Xoá tập tin hoặc đối tượng đồ hoạ
dir : Hiển thị danh sách đường dẫn hiện hành
matlabroot: Hiển thị đường dẫn cài đặt Matlab
pwd: Hiển thị đường dẫn hiện hành
mkdir: tạo đường dẫn mới

Path

addpath: thiết lập thư mục hiện hành.
genpath: trả về chuỗi, đường dẫn các thư mục chỉ định.
path2rc: lưu đường dẫn thành một file pathdef.m
pathtool: Hiển thị họp thoại setpath để xem hoặc thay đổi đường dẫn Matlab.
path: Xem đường dẫn của Matlab
rmpath: Gỡ đường dẫn

BIỂU THỨC (EXPRESSION)

Cũng giống như hầu hết các ngôn ngữ lập trình khác, MATLAB cũng cung cấp những biểu thức toán học, nhưng không giống các ngôn ngữ lập trình khác, hầu hết các biểu thức này đều liên quan đến ma trận.

- ❖ Biến số (variables)
- ❖ Số (Numbers)
- ❖ Toán tử (Operaters)
- ❖ Hàm (Functions)

Biến (Variables):

MATLAB không yêu cầu khai báo kiểu và kích thước của biến. Khi MATLAB bắt gặp tên một biến mới, nó sẽ tự động tạo ra biến và phân phát giá trị vùng nhớ cho biến. Nếu biến này đã tồn tại thì nó sẽ lưu giá trị mới và nếu cần thiết Matlab phân phác giá trị mới cho biến.

Biến chỉ sử được tối đa 19 ký tự có nghĩa, biến phân biệt giữa chữ hoa và chữ thường.

Biến bắt đầu bằng một từ theo sau là từ hay số hoặc dấu gạch chân (_).

Có hai loại biến:

- Biến toàn cục: có tác dụng trong toàn bộ chương trình.
- Biến cục bộ: chỉ có tác dụng trong phạm vi khai báo (nội trong hàm)

Ngoài ra MATLAB còn cung cấp một số biến đặc biệt ví dụ như **pi, i, j, ans** ...

Dùng các lệnh **who** và **whos** để kiểm tra biến, lệnh **clear** và **clear all** để xóa biến đã sử dụng

Ví dụ:

```
» clear a
» clear b degree
» a
??? Undefined function or variable a.
```

Số (Numbers):

MATLAB sử dụng các quy ước thập phân. Sử dụng ghi chú khoa học (scientific notation) **e** và số ảo (imaginary Numbers) để biểu diễn giá trị số.

Tất cả những con số đều được lưu kiểu định dạng (***format)***

Dùng hàm **format** để định dạng kiểu số:

Ví dụ:

```
» b=3/26;
» format long; b
b =
  0.11538461538462
» format short e; b
b =
  1.1538e-001
» format bank; b
b =
        0.12
» format +; b
b =
+
» format rat; b
b =
    3/26
» format short; b
b =
   0.1154
```

Toán tử:

+	Cộng
-	Trừ
*****	Nhân
/	Chia
****	Chia trái
^	Mũ
‘	Chuyển ma trận

Hàm:

MATLAB cung cấp một lượng lớn các hàm toán học cơ bản bao gồm **abs, sqrt, exp, sin**,... Trong MATLAB có hai loại hàm:

- Các hàm **build-in** là những hàm chúng ta chỉ được sử dụng chứ không được hiệu chỉnh.
- Các hàm **M-file** chúng ta có thể xem và hiệu chỉnh nội dung của hàm.

Một vài hàm đặc biệt, cho giá trị là hằng số:

pi:	3.141759...
i,j:	đơn vị ảo i,j = $\sqrt{-1}$
eps:	điểm chấm động có quan hệ đến độ chính xác, 2^{-52}
realmin:	Số chấm động nhỏ nhất, 2^{-1022}
realmax:	Số điểm chấm động lớn nhất, 2^{1023}
NaN:	Not a number(Không phải là một con số)
Inf:	Infinity(Số vô cùng lơn, không xác định)

Số vô hạn sinh ra do phép chia của một giá trị khác không cho một giá trị bằng không hoặc các phép tính của các biểu thức chưa được định nghĩa.

Không phải số khi ta tính toán các phép tính: không chia cho không hoặc vô cùng chia cho vô cùng.

Ví dụ:

```
        rho = (1+sqrt(5))/2
rho =
   1.6180
a = abs(3+4i)
a =
    5
z = sqrt(besselk(4/3,rho-i))
z =
  0.3730+ 0.3214i
huge = exp(log(realmax))
huge =
  1.7977e+308
toobig = pi*huge
toobig =
   Inf
```

Thứ tự ưu tiên trong MATLAB

- Dấu ( )

1. Parentheses ()
2. Transpose (.'), power (.^), complex conjugate transpose ('), matrix power (^)
3. Unary plus (+), unary minus (-), logical negation (~)
4. Multiplication (.*), right division (./), left division(.\), matrix multiplication (*), matrix right division (/), matrix left division (\)
5. Addition (+), subtraction (-)
6. Colon operator (:)
7. Less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)
8. Element-wise AND (&)
9. Element-wise OR (|)

Ví dụ:

```
» A=2; B=3; C=4;
» A*B^C
ans =
   162
» (A*B)^C
ans =
        1296
```

Hàm load: đọc một file nhị phân chứa ma trận các phần tử được tạo ra từ một hàm nào đó hoặc đọc một file_text chứa dữ liệu số. Dữ liệu trong file_text sẽ được sắp xếp như bảng chữ nhật của các số được ngăn cách bởi khoảng trắng và mỗi dòng được viết trên mỗi hàng và số phần tử trên mỗi hàng phải bằng nhau.
Chương trình tạo và tải tập tin có phần mở rộng *.dat

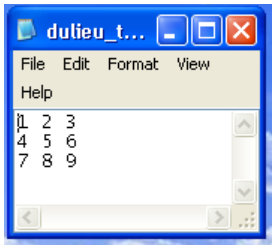
Chương trình chính	Hàm con
clear all; clc	**function file_dulieu**
file_dulieu	**A=[1 2 3;4 5 6;7 8 9];**
load dulieu, A	**save dulieu A**

```
A =

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

Chương trình tạo và tải tập tin có phần mở rộng *.text

Chương trình chính	Tạo tập tin dữ liệu dulieu_text.txt

```
clear all; clc
load dulieu_text.txt
dulieu_text
```




```
dulieu_text =

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

M-Files: ta có thể tạo một tập tin dữ liệu trong Matlab rồi lưu với phần mở rộng *.m

Chương trình chính	Tập tin dữ liệu *.m
clear all; clc **dulieu, A**	**A=[1 2 3;4 5 6;7 8 9];**

```
A =

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

MỘT VÀI HÀM TOÁN HỌC CƠ BẢN TRONG MATLAB

Hàm hình học:

Hàm	Chú thích
sin	Tính sin của một góc
sinh	Hyperbolic sine
asin, asinh	
cos	Tính cosine
cosh	
acos, acosh	
tan, cot	
tanh, coth	
atan, acot	

Hàm mũ và logarit

exp(x): Hàm mũ cơ số e^x

log(x): Hàm logarit cơ số e (ln(x))

log10(x): Hàm logarit cơ số thập phân ($\log_{10}(x)$)

log2(x): Hàm logarit cơ số 2 ($\log_2(x)$)

sqrt(x): Hàm căn bậc hai

Hàm số phức:

abs: Hàm trị tuyệt đối
angle: Hàm tính góc
conj:
real: Phần thực của số phức
imag: Phần ảo của số phức

Các hàm liên quan đến số:

fix: Làm tròn dần về 0
floor: làm tròn về giá trị âm
ceil: làm tròn về giá trị dương
round:
sign:

Ví dụ:

```
» a=[1,-4,6,-7]
a =
    1   -4    6   -7
» b=sqrt(a)
b =
  1.0000            0 + 2.0000i   2.4495            0 + 2.6458i
» real(b)
ans =
   1.0000        0   2.4495        0
» imag(b)
ans =
        0   2.0000        0   2.6458
```

MA TRẬN

Trong Matlab ma trận là một mảng chữ nhật các phần tử. Nếu ma trận có duy nhất một phần tử ta có ma trận vô hướng, ma trận có một hàng hoặc một cột ta có véc tơ. Các phép toán trên ma trận được thiết kế một cách tự nhiên và tiện lợi cho người sử dụng.

Bạn có thể tạo ma trận theo nhiều cách khác nhau:

- Xây dựng ma trận bằng việc nhập từng phần tử một
- Tải một file dữ liệu từ bên ngoài
- Xây dựng các ma trận nhờ các hàm build-in
- Tạo một ma trận bằng hàm do ta tạo ra.

Một vài nguyên tắc cơ bản để xây dựng ma trận là:

Phân biệt giữa các phần tử trong hàng là khoảng trắng hoặc dấu phẩy
Sử dụng dấu chấm phẩy (;) để ngăn cách giữa các hàng.
Bao quanh một ma trận là dấu: []

Ví dụ: cần khai báo một ma trận như sau:

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

Matlab

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

A =

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

Khi nhập vào ma trận như trên, Matlab tự động lưu biến A trong Workspace bạn có thể tham chiếu đến nó chỉ đơn giản nhập vào A.

Khi ta không nhập một biến lấy dữ liệu ra, Matlab tự động sử dụng biến **ans** để gán kết quả của phép tính. Và biến này cũng được sử dụng như những biến khác.

Vec tơ hàng là ma trận 1 x n gồm 1 hàng và n cột

Ví dụ:

```
>> A=[1 2 3 4 5]

A =

     1     2     3     4     5
```

Vec tơ cột là một ma trận cỡ m x1, gồm một cột và m hàng .

```
>> B=[1;2;3;4;5]

B =
   I
      1
      2
      3
      4
      5
```

Sử dụng dấu " ' " để chuyển đổi qua lại giữa các vec tơ hàng và vectơ cột

```
>> A'

ans =

     1
     2
     3
     4
     5
```

```
>> B'

ans =

     1     2     3     4     5
```

Ma trận số phức

```
» b=[4;8-i*23;5;3+i*10];
» b'
ans =
  4.0000          8.0000 +23.0000i   5.0000          3.0000 -10.0000i
» b.'
ans =
  4.0000          8.0000 -23.0000i   5.0000          3.0000 +10.0000i
```

Các hàm MATLAB tạo ma trận đặc biệt

zeros: Tạo ma trận các phần tử bằng đều bằng không

Cú pháp	Chú thích
zeros(n)	Tạo ma trận vuông nxn các phần tử đều bằng không
zeros(m,n)	Tạo ma trận cỡ mxn các phần tử đều bằng không
zeros([m n])	Tạo ma trận cỡ mxn các phần tử đều bằng không
zeros(size(A))	Tạo ma trận không dựa vào kích thước của ma trận A

Ví dụ:

```
>> A=zeros(3)

A =

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

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

B =

     0     0     0
     0     0     0
```

```
>> size(C)

ans =

     3     3

>> D=zeros(size(C))

D =

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

ones: Tạo ma trận các phần tử bằng đều bằng một

Cú pháp	Chú thích
ones(n)	Tạo ma trận vuông nxn các phần tử đều bằng 1
ones(m,n)	Tạo ma trận cỡ mxn các phần tử đều bằng1
ones([m n])	Tạo ma trận cỡ mxn các phần tử đều bằng 1
ones(size(A))	Tạo ma trận phần tử bằng 1 dựa vào kích thước của ma trận A

Ví dụ:

```
>> A=ones(3)

A =

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

```
>> B=ones(2,2)

B =

     1     1
     1     1
```

```
>> size(C)

ans =

     2     3

>> ones(size(C))

ans =

     1     1     1
     1     1     1
```

***eye* :** Tạo ma trận đơn vị

Cú pháp: Y = eye(n)
Y = eye(m,n)
Y = eye(size(A))

Ví dụ:

```
>> eye(3)

ans =

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

```
>> eye(2,3)

ans =

     1     0     0
     0     1     0
```

pascal: Tạo ma trận Pascal

magic: Tạo một ma trận

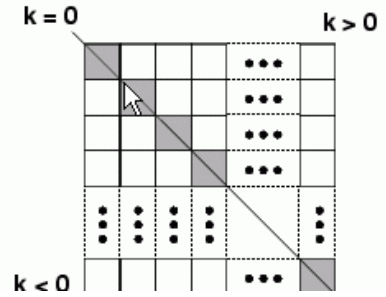
Ví dụ:

```
A = pascal(3)

A =

     1     1     1
     1     2     3
     1     3     6
```

```
>> magic(3)

ans =

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

diag: tạo ma trận đường chéo

Cú pháp	Chú thích	Ví dụ

diag(v,k)		`>> diag([2 1 2],1)`
		 `ans =`
		 `0 2 0 0`
		 `0 0 1 0`
		 `0 0 0 2`
		 `0 0 0 0`

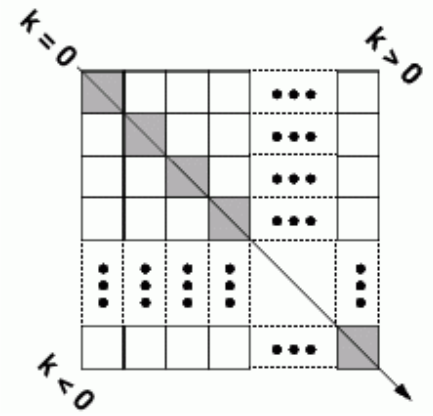
diag(v)		`>> A=diag([5 6 9 ])` `A =`

diag(X,k)

Ví dụ:

```
» A
A =
    1    2    3
    4    5    6
    7    8    9
» diag(A)
ans =
    1
    5
    9
» diag(diag(A))

ans =
    1    0    0
    0    5    0
    0    0    9
```

tril và ***triu***: Tạo ma trận tam giác trên và tam giác dưới

Cú pháp	Chú thích	Ví dụ
tril(v) tril(X,k)		>> tril(5*ones(4,4)) ans = 5 0 0 0 5 5 0 0 5 5 5 0 5 5 5 5 >> tril(5*ones(4,4),-1) ans = 0 0 0 0 5 0 0 0 5 5 0 0 5 5 5 0

linspace, logspace : Tạo vec tơ hàng

Cú pháp	Chú thích
linspace(a,b)	Tạo vec tơ hàng gồm 100 phần tử trong khoảng a,b
linspace(a,b,n)	Tạo vec tơ hàng gồm n phần tử trong khoảng a,b

Ví dụ:

```
>> linspace(1,2,4)
ans =
 1.0000    1.3333    1.6667    2.0000

>> logspace(1,2,4)
ans =
10.0000   21.5443   46.4159  100.0000
```

Tạo dãy bằng dấu hai chấm (**:**)
Ví dụ:

```
» A=1:5
A =
    1    2    3    4    5
» B=1:0.1:1.5
B =
   1.0000   1.1000   1.2000   1.3000   1.4000   1.5000
```

```
» C=1:-0.1:0.75
C =
   1.0000   0.9000   0.8000
```

rand, randn : Tạo ra các phần tử ngẫu nhiên cùng kiểu hoặc thông thường.

```
>> C=rand(2,4)

C =

    0.9501    0.6068    0.8913    0.4565
    0.2311    0.4860    0.7621    0.0185

>> C=randn(2,4)

C =

   -0.4326    0.1253   -1.1465    1.1892
   -1.6656    0.2877    1.1909   -0.0376
```

Các phép tính trên ma trận

Phép tính	Chú thích
+, -	Cộng hoặc trừ hai ma trận cùng kích thước
A*B	Nhân hai ma trận A và B
A/B	Chia hai ma trận (chia phải) A và B
A\B	Chia trái hai ma trận B và A
A.*B	Nhân từng phần tử của hai ma trận A và B
A./B	Chia từng phần tử của hai ma trận A và B
A.\B	Chia từng phần tử của hai ma trận B và A
.^	Mũ cho từng phần tử của mảng

x	1 2 3	y	4 5 6
x’	1 2 3	y’	4 5 6
x+y	5 7	x-y	-3 -3

2\x	1/2 1 3/2	2./x	2 1 2/3
x/y	0 0 1/6 0 0 1/3 0 0 1/2	x./y	1/4 2/5 1/2

	9		-3
x+2	3 4 5	x-2	-1 0 1
x*y	Error	x.*y	4 10 18
x'*y	32	x'.*y	Error
x*y'	4 5 6 8 10 12 12 15 18	x.*y'	Error
x*2	2 4 6	x.*2	2 4 6
x\y	16/7	x.\y	4 5/2 2

x/2	1/2 1 3/2	x./2	1/2 1 3/2
x^y	Error	x.^y	1 39 729
x^2	Error	x.^2	1 4 9
2^x	Error	2.^x	2 4 8
(x+i*y)'	1 - 4i 2 - 5i 3 - 6i		
(x+i*y).'	1 + 4i 2 + 5i 3 + 6i		

Chỉ số ma trận:

Phần tử ở dòng i cột j của ma trận A được kí hiệu A(i,j). Ví dụ A(4,2) là phần tử ở dòng bốn, cột hai của ma trận A.

Ví dụ:

Ma trận	Gán	Lấy giá trị

`>> A=[4 5 6;9 8 7;1 2 3]`
`A =`
`4 5 6`
`9 8 7`
`1 2 3`
	`>> A(2,3)=10`
	`A =`
	`4 5 6`
	`9 8 10`
	`1 2 3`
		`>> B=A(2,1)`
		`B =`
		`9`

Ta cũng có thể tham chiếu đến các phần tử của một ma trận bằng một chỉ số duy nhất A(k), cách này thường dùng để tham chiếu đến các phần tử của vector hàng hoặc cột. Tuy nhiên ta cũng có thể áp dụng cho các ma trận hai chiều, trong trường hợp này chỉ số là vị trí của phần tử trong ma trận.

```
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16

>> A(4)
ans =
    13

>> A(2)
ans =
     5

>> A(10)
ans =
     7

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

Nếu ta truy xuất vào phần tử ở bên ngoài ma trận Matlab sẽ báo lỗi

```
>> A(4,5)
???  Index exceeds matrix dimensions.

>> |
```

Ngược lại, nếu ta gán một giá trị cho một phần tử ở bên ngoài ma trận thì kích thước ma trận sẽ tự động tăng lên để cung cấp vị trí cho các phần tử mới.

```
>> X=A

X =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

```
>> X(4,5)=17

X =

     1     2     3     4     0
     5     6     7     8     0
     9    10    11    12     0
    13    14    15    16    17
```

Truy xuất vào hàng thứ i, cột bất kì của ma trận A ta thực hiện A(i ,:). Ngược lại khi cần truy xuất vào hàng bất kì, cột j của ma trận A ta thực hiện A(:,j)

Ma trận	Phần tử hàng của ma trận	Phần tử cột của ma trận

`>> C=[10 11 12; 3 4 6;9 7 8]`
`C =`
`10 11 12`
`3 4 6`
`9 7 8`
	`>> C(2,:)`
	`ans =`
	`3 4 6`
		`>> C(:,2)`
		`ans =`
		`11`
		`4`
		`7`

Toán tử " **:** " đây là một trong những toán tử quan trọng nhất của Matlab, nó xuất hiện trong các dạng khác nhau.

Biểu thức

```
>> 1:10

ans =

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

Với khai báo như trên ta thu được một vactor có một hàng, mười cột với bước nhảy mặc định là một.

Để định giá trị bước nhảy ta khai báo như sau:

```
>> 100:-7:50

ans =

   100    93    86    79    72    65    58    51
```

Khai báo chỉ số phần tử của ma trận A(1:k,j) nghĩa là lấy các phần tử ở dòng một đến dòng k và cột j của ma trận A.

```
A =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

```
>> A(1:4,2)

ans =

     2
     6
    10
    14
```

Khi khai báo A(:,end) có nghĩa là lấy các phần tử ở hàng bất kỳ, cột cuối cùng của ma trận A

```
A =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

```
>> A(:,end)

ans =

     4
     8
    12
    16
```

Truy xuất vào ma trận con của một ma trận, ví dụ E_{ixj} là ma trận con của ma trận C_{mxn} ta thực hiện như sau:

>> E=C([2 3],[1 3])

Kết quả:

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

E =

     3     6
     9     8
```

Nếu muốn truy xuất vào phần tử cuối của hàng hoặc cột trong một ma trận ta có thể thực hiện như sau:

```
C =

    10    11    12
     3     4     6
     9     7     8
```

```
>> C(2,end)

ans =

     6
```

```
>> C(end,1)

ans =

     9
```

Xóa một hàng hoặc một cột của một ma trận thực hiện phép gán hàng hoặc cột cần xoá bỏ bằng []

Ví dụ:

```
C =

    10    11    12
     3     4     6
     9     7     8
```

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

C =

    10    11    12
     3     4     6
```

```
>> C(:,2)=[]

C =

    10    12
     3     6
```

Thêm vào một ma trận một hàng hoặc một cột ta thực hiện như sau:

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

D =

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

Thêm vào ma trận D một hàng

```
>> D(4,:)=[8 4 6]

D =

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

Thêm vào ma trận D một cột:

```
>> D(:,4)=[8 4 6 4]'
D =

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

Ghép chuỗi: là quá trình xử lý lắp ghép nhiều ma trận con (nhỏ) thành một ma trận lớn, yêu cầu của phép ghép là các ma trận con phải có kích thước phù hợp.

```
clear all; clc
A1=[1 2;5 6], A2=[3 4;7 8], A3=[9 10 11 12;13 14 15 16],
A=[A1 A2;A3]
```

```
A1 =
     1     2
     5     6

A2 =
     3     4
     7     8

A3 =
     9    10    11    12
    13    14    15    16

A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16
```

Các hàm liên quan đến sử lý ma trận:

size: Cho biết kích thướt của ma trận

median: Tính giá trị trung bình của ma trận

max: Tìm phần tử lớn nhất trong ma trận

min: Tìm phần tử nhỏ nhất trong ma trận
mean: Tính giá trị trung bình của dãy
sum: Tính tổng của một ma trận
length: Hàm trả về chiều dài của một mảng

Ví dụ:

```
>> size(D)

ans =

     4     2
```

```
>> size(C)

ans =

     2     4
```

```
A1 =
    1     2     3     4     5     6

>> mean(A1)
ans =

    3.5000

>> sum(A1)

ans =

    21
```

```
>> max(A1)
ans =
     6
>> min(A1)
ans =
     1
>> median(A1)
ans =
     3.5000
```

Hàm ***sort*** sắp xếp giá trị của mảng

```
>> A=[4 5 9 7 2 1]

A =

     4     5     9     7     2     1

>> sort(A)

ans =

     1     2     4     5     7     9
```

<u>Ứng dụng MATLAB trong việc giải phương trình đại số tuyến tính</u>

Giải phương trình đại số là một trong những vấn đề quan trọng trong tính toán kỹ thuật. Có nhiều cách giải tuy nhiên trong phần này chỉ đề cập đến các hàm có sẵn trang MATLAB để giải hệ phương trình đại số tuyến tính.

Cách 1:

Giả sử có hai ma trận A và B, tìm ma trận x sao cho X thoả mản Ax=B hay xA=B

Đối với phương trình thoả mản Ax=B thì ta dùng phép chia trái trong MATLAB được thực hiện như sau: x=A\B còn đối với xA=B thì ta tìm nghiệm x=A/B

Ví dụ:

```
>> A=[4 5 6;9 8 5;4 1 10]

A =

     4     5     6
     9     8     5
     4     1    10
```

```
>> B=[1 8 7]'

B =

     1
     8
     7
```

```
>> x=A\B

x =

    2.3298
   -1.5745
   -0.0745
```

```
>> x=A/B'

x =

    0.7544
    0.9474
    0.7193
```

```
>> a = [3,-1,1;
        2,3,-1;
        -1,2,-1];
>> b=[10,5,-1];
>> x = b/a

x =

   -2.0000    5.0000   -6.0000
```

Chú ý: Đối với phép chia x= A\B yêu cầu hai ma trận A và B bằng nhau về số hàng còn đối với x=A/B thì yêu cầu số hàng của ma trận A bằng số cột của ma trận B.

Cách 2:

Ta có Ax=B ⇨ $x = A^{-1}B$

x= A\ B ta thực hiện như sau bằng hàm inv(A)

x=inv(A)*B

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

x =

    2.3298
   -1.5745
   -0.0745
```

Cách 3:

Dùng phương pháp tách LU bằng hàm ***lu*** có sẵn trong MATLAB

Ví dụ: Giải hệ phương trình đại số Ax=B bằng phương pháp tách LU ta thực hiện như sau

```
>> [L,U]=lu(A)
L =
    0.4444   -0.5652    1.0000
    1.0000         0         0
    0.4444    1.0000         0
U =
    9.0000    8.0000    5.0000
         0   -2.5556    7.7778
         0         0    8.1739
>> x=U\(L\B)

 x =

    2.3298
   -1.5745
   -0.0745
```

Một vài ví dụ ứng dụng các cách trên để giải:

Ví dụ 1: Giải hệ phương trình sau

$$\begin{bmatrix} 4 & -2 & -10 \\ 2 & 10 & -12 \\ -4 & -6 & 16 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -10 \\ 32 \\ -16 \end{bmatrix}$$

Matlab:

Cách 1:

```
A = [4  -2  -10;  2  10  -12;  -4  -6  16];
B = [-10;  32;  -16];
X = A\B
```

Kết quả:

```
X =
    2.0000
    4.0000
    1.0000
```

Cách 2:

```
A = [4  -2  -10;  2  10  -12;  -4  -6  16];
B = [-10;  32;  -16];
C = inv(A)
X = C*B
```

Kết quả:

```
C =
    2.2000    2.3000    3.1000
    0.4000    0.6000    0.7000
    0.7000    0.8000    1.1000
X =
    2.0000
    4.0000
    1.0000
```

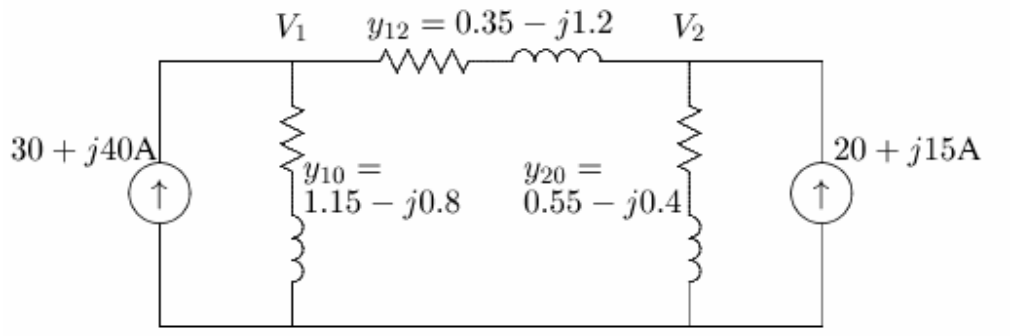
Cách 3:

```
A = [ 4  -2  -10;  2  10  -12;  -4  -6  16 ]
B = [-10;  32  -16];
[L,U] = lu(A)
X = inv(U)*inv(L)*B
```

Kết quả:

```
L =
    1.0000         0         0
    0.5000    1.0000         0
   -1.0000   -0.7273    1.0000

U =
    4.0000   -2.0000  -10.0000
         0   11.0000   -7.0000
         0         0    0.9091

X =
    2.0000
    4.0000
    1.0000
```

Ví dụ 2: Xác định hiệu điện thế V_1 và V_2 của mạch như sau



Áp dụng định luật Kirchhoft cho dòng điện ta có:

$$\begin{bmatrix} 1.5-j2.0 & -.35+j1.2 \\ -.35+j1.2 & 0.9-j1.6 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 30+j40 \\ 20+j15 \end{bmatrix}$$

Công suất của dòng điện được tính S= V*I

Matlab:

```
>> j=sqrt(-1);
>> I=[30+j*40; 20+j*15];
>> Y=[1.5-j*2 -.35+j*1.2; -.35+j*1.2 .9-j*1.6];
>> V=inv(Y)*I;
>> S=V.*conj(I);
>> V

V =

   3.5902 +35.0928i
   6.0155 +36.2212i

>> S

S =

  1.0e+003 *

   1.5114 + 0.9092i
   0.6636 + 0.6342i
```

Polynomials and Interpolation(Hàm đa thức và nội suy)

MATLAB cung cấp một vài hàm cho các phép toán đa thức cơ bản như tìm nghiệm đa thức (roots), ước lượng giá trị đa thức(polyval), phép tính vi phân,... Thêm vào đó có một vài hàm cho những ứng dụng cao hơn như hàm nội suy, sử lý số liệu thực nghiêm, phân rã đa thức.

Tìm nghiệm đa thức:

$$s^6 + 9s^5 + 31.25s^4 + 61.25s^3 + 67.75s^2 + 14.75s + 15$$

Matlab: Sử dụng hàm ***roots*** để tìm nghiệm đa thức trên

```
p = [ 1  9  31.25  61.25  67.75  14.75  15 ]
r = roots(p)
```

Ta thu được nghiệm:

```
r =
    -4.0000
    -3.0000
    -1.0000 + 2.0000i
    -1.0000 - 2.0000i
     0.0000 + 0.5000i
     0.0000 - 0.5000i
```

Tìm hệ số của đa thức khi biết trước tập nghiệm:

Giả sử nghiệm của đa thức là : -1, -2, -3 ± j4

Dùng hàm ***poly*** để tìm hệ số của đa thức, ta thực hiện MATLAB như sau:

```
i = sqrt(-1)
r = [-1  -2  -3+4*i  -3-4*i ]
p = poly(r)
```

Kết quả:

```
p =
     1   9   45   87   50
```

Đa thức của bộ nghiệm trên là

$$s^4 + 9s^3 + 45s^2 + 87s + 50 = 0$$

Tính giá trị của đa thức:

Sử dụng hàm ***polyval(c,x_0)*** để tính giá trị của đa thức c tại x=x_0

Ví dụ 1: Tìm giá trị của đa thức $f(x) = x^3 - 3x - 1$ sau tại x=5

MATLAB:

```
>> p=[1 0 -3 -5];
>> x0=5;
>> polyval(p,x0)

ans =

   105
```

Ví dụ 2:

```
c = [1   2  3  1];
x = 0:1:4;
y = polyval(c, x)
```

Kết quả:

```
y =
     7   23   55   109
```

Nhân và chia đa thức:

Sử dụng hai hàm ***conv*** và ***deconv*** khi ta muốn nhân hoặc chia hai đa thức với nhau:

Ví dụ 1: Cho hai đa thức : $f_1 = s^2 + 7s + 12$ và $f_2 = s^2 + 9$ Hãy tính $f_3 = f_1 * f_2$

MATLAB:

```
>> f1=[1 7 12];
>> f2=[1 0 9];
>> f3=conv(f1,f2)

f3 =

     1     7    21    63   108
```

Vậy đa thức cần tìm là:

$f_3 = s^4 + 7s^3 + 21s^2 + 63s + 108$

Ví dụ 2: Cho hai đa thức $f_4 = s^4 + 9s^3 + 37s^2 + 81s + 52$ và $f_5 = s^2 + 4s + 13$, hãy tính $f_6 = \frac{f_4}{f_5}$

MATLAB:

```
>> f4=[1 9 37 81 52];
>> f5=[1 4 13];
>> [f6,r]=deconv(f4,f5)
f6 =
     1     5     4
r =
   0     0     0     0     0
```

Vậy đa thức cần tìm là: $f_6 = s^2 + 5s + 4$

Với r là phần dư f4=conv(f5,f6)+r

Ví dụ 3:

```
>> g1=[1 3 6 4 5];
>> g2=[1 5 4];
>> [g3,r]=deconv(g1,g2)
g3 =
     1    -2    12
r =
    0     0     0   -48   -43
>> g32=conv(g2,g3)+r
g32 =
     1     3     6     4     5
```

Tính đạo hàm của đa thức:

Ta sử dụng hàm ***polyder(p)*** để tính đạo hàm của đa thức p

Ví dụ:

```
>> h1=[1 2 1];
>> dh1=polyder(h1)
dh1 =
     2     2
```

```
>>  p =[1 0 -2 -5];
>> q = polyder(p)
q =
     3     0    -2
```

Phân rã đa thức:

Sử dụng hàm ***[r,p,k]=residue(b,a)***

$$\frac{b(s)}{a(s)} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \ldots + \frac{r_n}{s-p_n} + k_s$$

Ví dụ: Phân rã đa thức sau

$$F(s) = \frac{2s^3 + 9s + 1}{s^3 + s^2 + 4s + 4}$$

MATLAB:

Ví dụ 1:

```
b = [ 2  0  9  1];
a = [ 1  1  4  4];
[r,p,k] = residue(b,a)
```

Kết quả:

```
r =
      0.0000   -0.2500i
      0.0000   +0.2500i
     -2.0000
p =
      0.0000   +2.0000i
      0.0000   -2.0000i
     -1.0000
K =
      2.0000
```

Cuối cùng đa thức trên sau khi phân rã ta được

$$2 + \frac{-2}{s+1} + \frac{j0.25}{s+j2} + \frac{-j0.25}{s-j2} = 2 + \frac{-2}{s+1} + \frac{1}{s^2+4}$$

Dùng hàm ***[b,a]=residue(r,p,k)*** để chuyển ngược lại

Ví dụ 2:

$$\frac{-4+8s^{-1}}{1+6s^{-1}+8s^{-2}}$$

MATLAB:

```
b = [-4 8];
a = [1 6 8];
[r,p,k] = residue(b,a)

r =
   -12
     8


p =
    -4
    -2


k =
     []
```

Dùng hàm ***[b,a]=residue(r,p,k)*** ta có kết quả sau:

```
[b2,a2] = residue(r,p,k)

b2 =
    -4     8


a2 =
     1     6     8
```

Sử lý số liệu thực nghiệm:

Sử dụng hàm polyfit(x,y,n) để tìm hệ số của đa thức dựa vào một tập dữ liệu có nghĩa, với x và y là dữ liệu còn n là bậc đa thức nội suy

Ví dụ1:

```
>> x = [1 2 3 4 5]; y = [5.5 43.1 128 290.7 498.4];
>> p = polyfit(x,y,3)
p =
   -0.1917   31.5821  -60.3262   35.3400
```

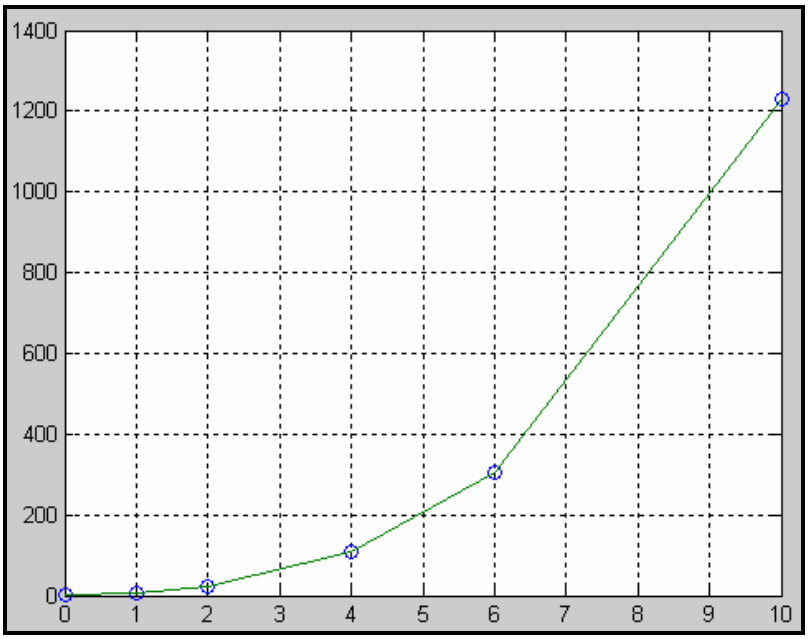
Ví dụ 2: Tìm đa thức bậc 3 dựa vào bảng dữ liệu sau:

x	0	1	2	4	6	10
y	1	7	23	109	307	1231

MATLAB:

```
>> x = [ 0 1 2 4 6 10];
>> y = [ 1 7 23 109 307 1231];
>> c = polyfit(x,y,3)
c =
   1.0000    2.0000    3.0000    1.0000
```

Đồ thị:



NỘI SUY

Nội suy dữ liệu một chiều:

Sử dụng hàm ***interp1*** để nội suy dữ liệu một chiều, cú pháp của lệnh như sau:

Cú pháp:

yi = interp1(x,Y,xi)
yi = interp1(Y,xi)
yi = interp1(x,Y,xi,method)
yi = interp1(x,Y,xi,method,'extrap')
yi = interp1(x,Y,xi,method,extrapval)

Chú thích:

- ***yi = interp1(x,Y,xi)*** trả về vec tơ giá trị yi tương ứng với vec tơ giá trị xi dựa vào phép nội suy của tập dữ liệu x, Y. Nếu Y là một ma trận thì hàm nội suy sẽ xây dựng theo từng cột của ma trận Y.

Ví dụ:

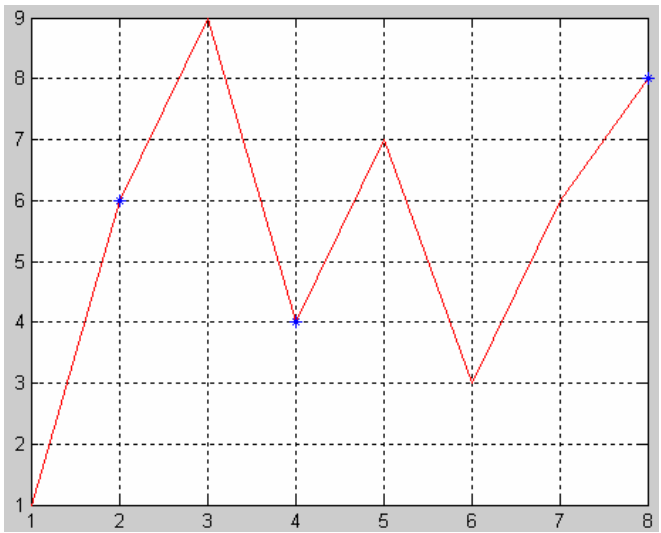
```
>> x = 0:10;
y = sin(x);
xi = 0:.5:10;
yi = interp1(x,y,xi)

yi =

  Columns 1 through 7

         0    0.4207    0.8415    0.8754    0.9093    0.5252    0.1411

  Columns 8 through 14

   -0.3078   -0.7568   -0.8579   -0.9589   -0.6192   -0.2794    0.1888

  Columns 15 through 21

    0.6570    0.8232    0.9894    0.7007    0.4121   -0.0660   -0.5440
```

- ***yi = interp1(Y,xi)*** giống như ***yi = interp1(x,Y,xi)*** tuy nhiên x = 1:N với N= size(Y,1) hoặc N= length(Y)

Ví dụ:

```
>> Y=[1 6 9 4 7 3 6 8];
>> xi=[2 4 8];
>> yi=interp1(Y,xi)

yi =

     6     4     8

>> plot([1:length(Y)],Y,'r',xi,yi,'b*')
>> grid on
```

Kết quả đồ thị:



- ***interp1(x,Y,xi,method)*** Nội suy dữ liệu theo nhiều phương pháp khác nhau

Method:

'nearest':

'linear': Phép nội suy tuyến tính và cũng là ***method*** mặc định của hàm ***interp1***.

'spline': Nội suy theo đường cong bậc 3

'cubic' và 'pchip': Nội suy theo đường cong Hermit

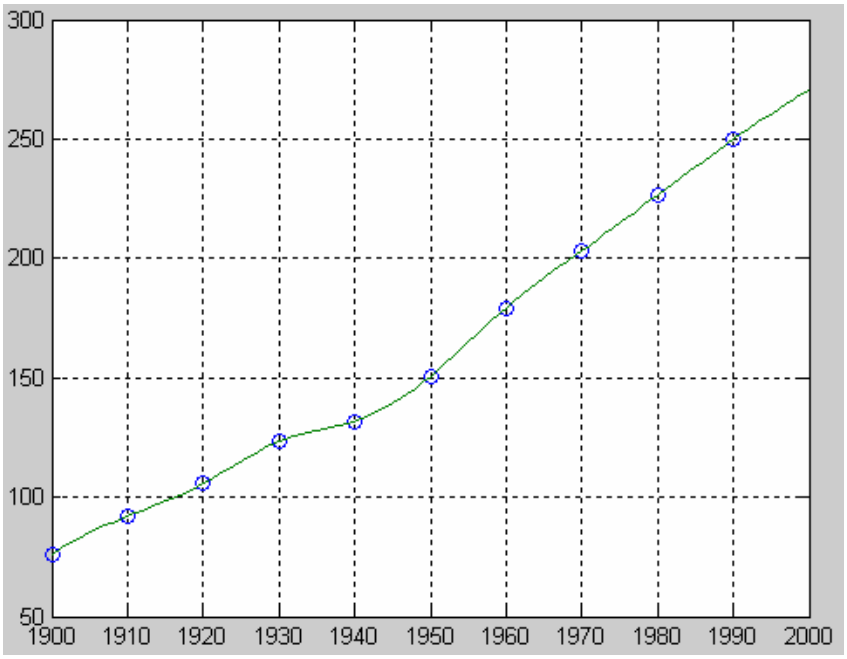
Chú ý:

Đối với các 'nearest' và 'lnear' thì giá trị phần tử của vec tơ xi nằm ngoài vùng giá trị x thì MATLAB hiểu các phần tử này là NaN, còn các phương pháp còn lại MATLAB sẽ thực hiện theo phép ngoại suy.

Ví dụ:

```
>>

t = 1900:10:1990;
p = [75.995  91.972  105.711  123.203  131.669...
     150.697  179.323  203.212  226.505  249.633];
>>  x = 1900:1:2000;
 y = interp1(t,p,x,'spline');
 plot(t,p,'o',x,y)
>> grid on
```

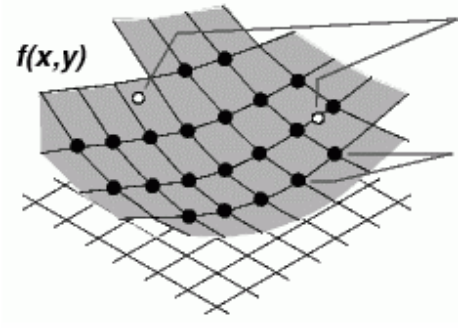
Kết quả bằng đồ thị:



- ***yi = interp1(x,Y,xi,method,'extrap')*** Thực hiện phép ngoại suy ứng với từng giá trị của xi nằm ngoài vùng dữ liệu của x.

- ***yi = interp1(x,Y,xi,method,extrapval)*** Trả về vec tơ giá trị của các phần tử xi ngoài vùng dữ liệu x.

Nội suy dữ liệu hai chiều: Sử dụng hàm ***interp2*** để thể hiện phép nội suy hai chiều. Đây là một hàm quan trọng cho việc sử lý ảnh và các dữ liệu mà ta mường tượng hóa.



Điểm nội suy P(XI,YI,ZI)

Bộ điểm lưới (X,Y,Z)

Cú pháp:

ZI = interp2(X,Y,Z,XI,YI)
ZI = interp2(Z,XI,YI)
ZI = interp2(Z,ntimes)
ZI = interp2(X,Y,Z,XI,YI,method)

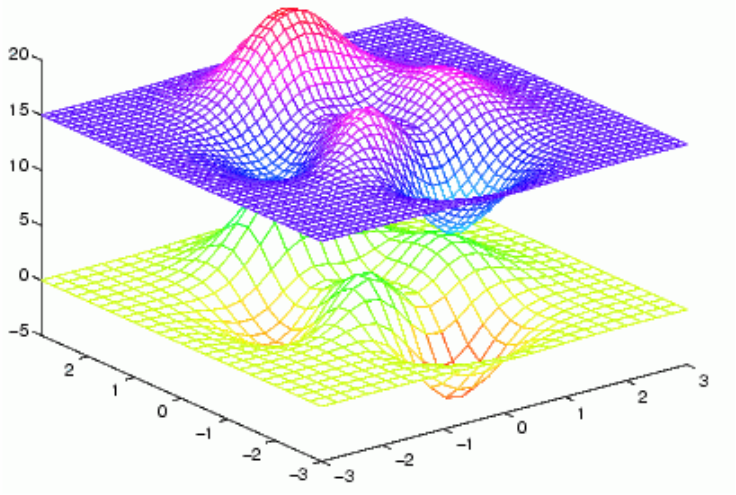
Chú thích:

- ***ZI = interp2(X,Y,Z,XI,YI)*** trả về vec tơ giá trị ZI tương ứng với cặp vec tơ giá trị (XI,YI) dựa vào phép nội suy của tập dữ liệu X, Y,Z.

Ví dụ:

```
>> [X,Y] = meshgrid(-3:.25:3);
Z = peaks(X,Y);
>> [XI,YI] = meshgrid(-3:.125:3);
>> ZI = interp2(X,Y,Z,XI,YI);
>> mesh(X,Y,Z), hold, mesh(XI,YI,ZI+15)
hold off
axis([-3 3 -3 3 -5 20])
```

Đồ thị:



- ***ZI = interp2(Z,XI,YI)*** Thực hiện phép nội suy hai chiều với tập dữ liệu X=1:N, Y=1:M và Z trong đó [M,N]=size(Z).
- ***ZI = interp2(Z,ntimes)*** Thực hiện phép nội suy bằng việc tăng thêm giá trị Z lên n lần
- ***ZI = interp2(X,Y,Z,XI,YI,method)*** Thực hiện phép nội suy 2 chiều dựa vào các phương pháp sau:

Method:

'linear':

'nearest':

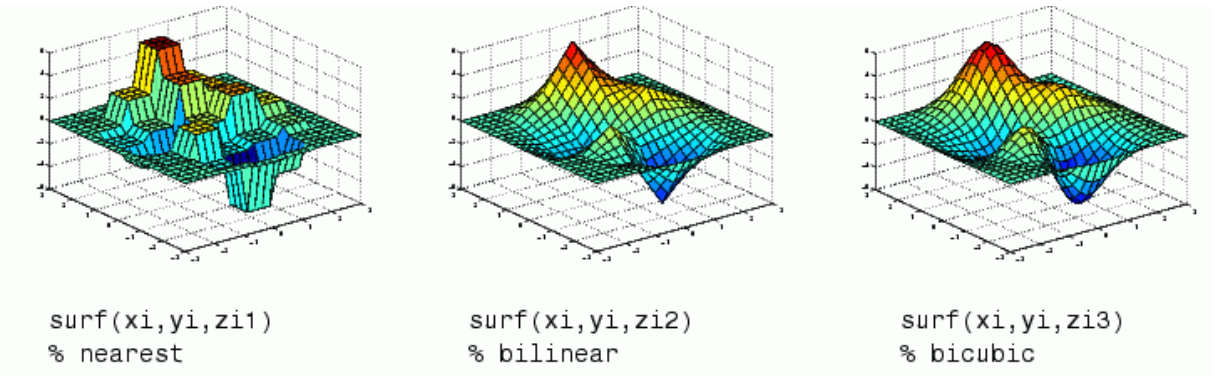
'spline'

'cubic'

So sánh các phương pháp nội suy khác nhau:

MATLAB:

```
[xi,yi] = meshgrid(-3:0.25:3);
zi1 = interp2(x,y,z,xi,yi,'nearest');
zi2 = interp2(x,y,z,xi,yi,'bilinear');
zi3 = interp2(x,y,z,xi,yi,'bicubic');
```



GIẢI PHƯƠNG TRÌNH VI PHÂN THƯỜNG

Có hai lớp bài toán:

- Bài toán giá trị đầu : Nghiệm bài toán phụ thuộc vào thời gian, do vậy nghiệm bài toán ở thời điểm ban đầu phải biết trước.
- Bài toán giá trị biên:Nghiệm bài toán không phụ thuộc thời gian, tuy nhiên nghiệm bài toán trên biên phải được biết trước

Giải bài toán với giá trị đầu:

Giả sử ta sử phương trình vi phân cấp 1 và điều kiện đầu cho như sau:

$$y' = f(t, y)$$

$$y'(t_0) = y_0$$

Các hàm MATLAB sau dùng để giải ptvp điều kiện đầu:

- ode23: Giải ptvp bậc 2-3 theo phương pháp Runge-Kutta
- ode45: Giải ptvp bậc 4-5 theo phương pháp Runge-Kutta

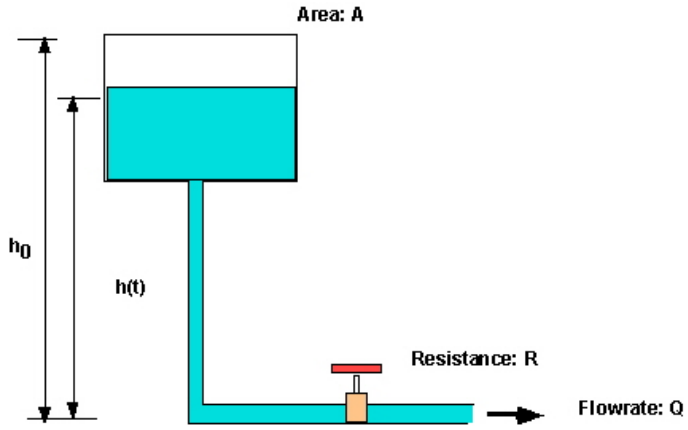
Cú pháp:

[T,Y] = solver(odefun,tspan,y0)
[T,Y] = solver(odefun,tspan,y0,options)
[T,Y] = solver(odefun,tspan,y0,options,p1,p2...)
[T,Y,TE,YE,IE] = solver(odefun,tspan,y0,options)

Chú thích:

solver :	ode23, ode45,...
odefun:	hàm chứa ptvp
tspan:	Vec tơ xác định khoảng tp
y0:	Vec tơ giá trị ban đầu
option:	Thuộc tính tích phân

Ví dụ 1:



Với :

$$\frac{dh}{dt} + \frac{h}{AR} = \frac{Q_{in}(t)}{A}$$

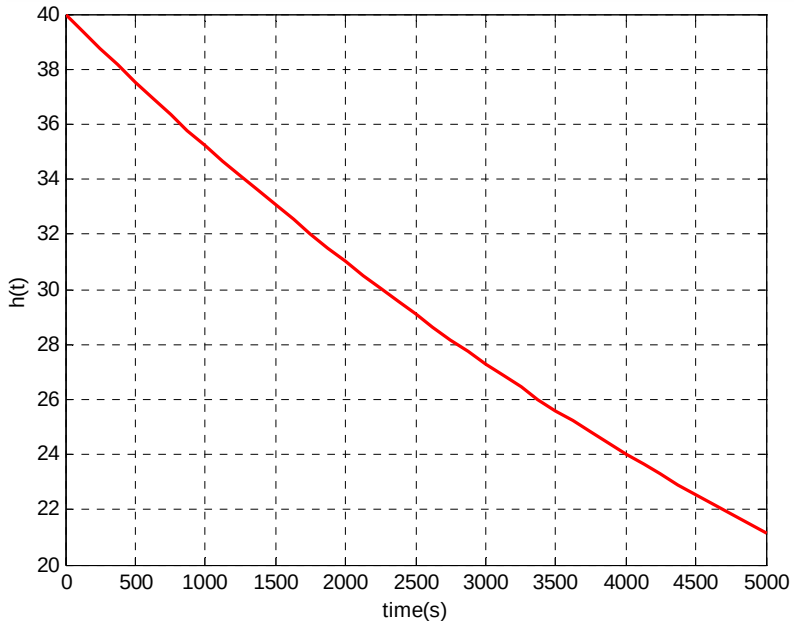
$$h(0) = 40; \quad A = 78.5; \quad R = 100$$

hay:

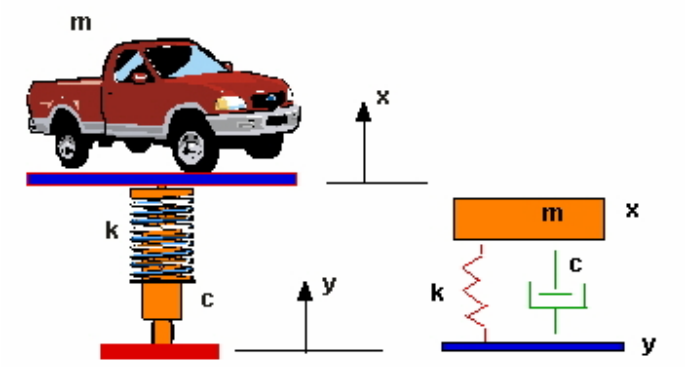
$$\frac{dh}{dt} = -\frac{h}{AR} + \frac{Q_{in}(t)}{A}; \quad h(0) = 40;$$

MATLAB:

```
>> clear all
>> A=78.5;
>> R=100;
>> h0=40;
>> tspan=[0 5000];
>> odefun = inline('-h/(78.5*100)','t','h');
>> [t h] = ode45(odefun,tspan,h0);
>> plot(t,h,'r','LineWidth',2)
>> grid on
```



Ví dụ 2:



CHƯƠNG 3

LẬP TRÌNH VỚI MATLAB

MATLAB cho phép người dùng lập trình theo hai loại : ***scripts*** và ***function***

SCRIPTS: Là hình thức đơn giản nhất của M-file bởi vì nó không có thông số vào và ra. Chúng là một tập hợp các lệnh và các hàm của **MATLAB**. Scripts hoạt động dựa vào đối tượng tồn tại trong Workspace. Tất cả các biến tạo ra trong scripts đều có thể sử dụng sau khi scripts kết thúc.

Ví dụ:

%%% M-file: ***ex_3_1.m***

```
theta=-pi:0.01:pi;
rho(1,:)=2*sin(5*theta).^2;
rho(2,:)=cos(10*theta).^3;
rho(3,:)=sin(theta).^2;
rho(4,:)=5*cos(3.5*theta).^3;
```

Thực hiện ở dòng command line

```
>> theta=-pi:0.01:pi;
>> rho(1,:)=2*sin(5*theta).^2;
>> rho(2,:)=cos(10*theta).^3;
>> rho(3,:)=sin(theta).^2;
>> rho(4,:)=5*cos(3.5*theta).^3;
```

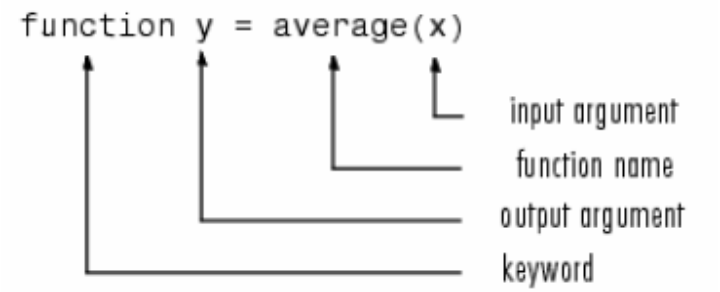
FUNCTION : Là scripts tuy nhiên có thêm đối số vào (input arguments) và đối số ra (output arguments). Hoạt động của biến trong một workspace riêng, các biến này chia rẽ với các biến trong workspace mà ta thực hiện trong scripts hay ở command line.

Các thành phần cơ bản của một hàm (***function***)

- Dòng định nghĩa hàm.
- Dòng cho biết ý nghĩa của hàm (***Dòng H1***)
- Dòng chú thích ý nghĩa của hàm (**Dòng *Help line***)
- Nội dung của hàm

Ví dụ:

Dòng định nghĩa hàm cho biết: Tên của hàm và các đối số của hàm được minh họa như sau:



Việc đặt tên hàm cũng có ràng buộc giống như việc đặt tên biến: bắt đầu bằng tên hàm bằng kí tự tiếp theo là các con số và dấu gạch dưới.

Đối số của hàm: Nếu hàm có nhiều giá trị trả về thì ta đặt tất cả các đối số này trong dấu ngoặc vuông []

```
function [x,y,z]= sphere(theta, phi, rho)
```

Nếu hàm không có giá trị trả về ta có thể để trống hay để dấu []

```
function print_result(x) hay function []=print_result(x)
```

Dòng H1: Đây là dòng trợ giúp đầu tiên, cho biết ý nghĩa của hàm, dòng này ngay sau dòng định nghĩa hàm và bắt đầu với kí tự " ***%***"

Ví dụ : hàm ***average.m***

% AVERAGE means of vectơ Elements

Dòng này sẽ là dòng đầu tiên xuất hiện khi bạn gõ ***help function_name*** (Tên hàm) tại dòng nhắc (>>) của MATLAB. Dòng này chứa thông tin rất quan trọng của hàm.

Các dòng trợ giúp (Help line):

Các dòng này được tạo ra nhmằ mục đích chú thích các thông số của hàm cũng như các ví dụ khi sử dụng hàm này. Các dòng này xuất hiện sau dòng H1 line khi bạn gõ ***help function_name*** tại dòng nhắc của MATLAB. Các dòng này sẽ kết thúc khi có dòng trắng giữa hai dòng có " ***%***"

Ví dụ bạn gõ ***help sin*** tại dòng nhắc MATLAB:

Có kết quả như sau:

```
>> help sin

 SIN    Sine.
    SIN(X) is the sine of the elements of X.

 Overloaded methods
    help sym/sin.m
```

Nội dung của hàm: đây chính là phần chính của hàm

```
function A=average(x)
% Tinh gia tri trung binh cua mot vec to
% x mot vecto co m phan tu
% A gia tri tb cu x
m=size(x,2);
tong=0;
for i=1:m
    tong= tong+x(1,i);
end
A=tong/m;
```

Dòng H1

Các dòng trợ giúp

Nội dung của hàm

Chú thích trong MATLAB :

Như đã đề cập ở trên, Dòng chú thích là dòng bắt đầu với dấu **%** . Dòng chú thích có thể xuất hiện bất cứ đâu trong ***.m** (tập tin của MATLAB) và ta có thể đặt ở cuối dòng lệnh.

Ví dụ:

```
%% Tinh tong cac phan tu
y=sum(x) % su dung ham sum
```

Hàm hoạt động như thế nào:

Người dùng có thể gọi hàm ở dòng nhắc Matlab hay bất cứ nơi nào trong ***.m** (tập tin của MATLAB) chắc rằng các thông số cần thiết cho hàm đủ là được

Khi gặp một tên mới MATLAB :

1. Kiểm tra xem có phải là tên biến hay không.
2. Kiểm tra xem có phải là hàm con (subfunction) và hàm riêng (private function(Hàm nằm trong thư mục con của thư mục hiện hành gọi là hàm riêng)) hay không.
3. Kiểm tra xem hàm này có trong đường dẫn hiện hành hay không.

```
function (avg,med) = newstats(u) % Primary function
% NEWSTATS Find mean and median with internal
functions.
n = length(u);
avg = mean(u,n);
med = median(u,n);
```

Subfunction

```
function a = mean(v,n)
% Calculate average.
a = sum(v)/n;
```

```
function m = median(v,n)
% Calculate median.
w = sort(v);
if rem(n,2) == 1
   m = w((n+1)/2);
else
   m = (w(n/2)+w(n/2+1))/2;
end
```

Kiểm tra biến vào và ra của một hàm:

Ta sử dụng hai hàm sau: ***nargin*** và ***nargout*** để kiểm tra thông số vào và ra của một hàm.

Ví dụ:

```
function c = testarg1(a,b)
if (nargin == 1)
c = a.^2;
elseif (nargin == 2)
c = a + b;
end
```

Các phép toán logic:

Phép toán	Ý nghĩa
&	Và
|	Hoặc
~	Phủ định

Bảng toán logic

A	B	~A	A|B	A&B	A xor B
0	0	1	0	0	0
0	1	1	1	0	1
1	0	0	1	0	1
1	1	0	1	1	0

Ví dụ:

```
>> u = [1 0 2 3 0 5];
v = [5 6 1 0 0 7];
u & v
ans =
     1     0     1     0     0     1
>> u | v
ans =
     1     1     1     1     0     1
>> ~u
ans =
     0     1     0     0     1     0
```

Các hàm logic (Logical function :

1. ***C = xor(A,B)***

Ý nghĩa:

A	B	C
zero	zero	0
zero	nonzero	1
nonzero	zero	1
nonzero	nonzero	0

Ví dụ:

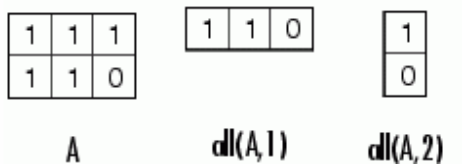
```
>>  A = [0 0 pi eps]

A =

         0         0    3.1416    0.0000

>>  B = [0 -2.4 0 1]

B =

         0   -2.4000         0    1.0000

>> C = xor(A,B)

C =

     0     1     1     0
```

2. ***all(A)*** hoặc ***all (A, dim)***

Kiểm tra tất cả các phần tử trong vectơ A, giá trị sẽ trả về 1 nếu tất cả phần tử khác không ngược lại bằng 0. Nếu A là một ma trận thì all(A) sẽ kiểm tra vectơ cột của ma trận A.

```
>> A=[0 0 0 0];
>> all(A)
ans =
     0

>> B=[7 8 9 10];
>> all(B)
ans =
     1

>> C=[0 5 6 0 7];
>> all(C)
ans =
     0

>> D=[4 5 6;1 2 3; 7 8 0];
>> all(D)
ans =

     1     1     0
```

all(A,dim): Kiểm tra phần tử mảng A theo vec tơ hàng hay cột mà thôi



Ta có thể ứng dụng hàm này để lập trình

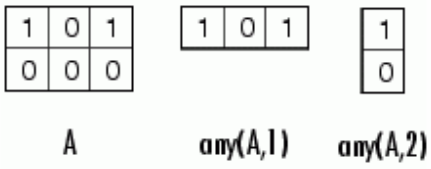
```
if all(A < 5)
        lệnh
end
```

```
>> A=[4 5 6; 1 3 6];
>> all(A<4)

ans =

     0     0     0

>> all(A<5)

ans =

     1     0     0
```

3. Hàm ***any(A)*** và ***any(A,dim)***

Cũng giống như hàm ***all*** nhưng ý nghĩa hàm ***any*** chỉ trả về 0 khi tất cả các phần tử ma trận A đều bằng 0 còn các trường hợp còn lại hàm trả về 1

any(A,dim)



Ví dụ:

```
>> A=[4 9 2 0];
>> all(A)
ans =
     0
>> any(A)
ans =
     1
```

```
>> B=[4 5 9;4 2 8;0 5 6];
>> all(B)
ans =
     0     1     1
>> any(B)
ans =
     1     1     1
```

4. Hàm ***find(A)***

Tìm chỉ số của phần tử A theo một đều kiện nào đó. Ví dụ sau sẽ minh họa cụ thể hơn

```
>> A=[1 5 6 2 ;4 3 0 2.5;7 9 1 0]
A =
    1.0000    5.0000    6.0000    2.0000
    4.0000    3.0000         0    2.5000
    7.0000    9.0000    1.0000         0

>> i=find(A<3)
i =
     1
     8
     9
    10
    11
    12
>> A(i)=10
A =
    10     5     6    10
     4     3    10    10
     7     9    10    10
```

Toán tử quan hệ:

Toán tử	Ý nghĩa
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
>	Lớn hơn
>=	Lớn hơn hoặc bằng
==	Bằng nhau
~=	Không bằng

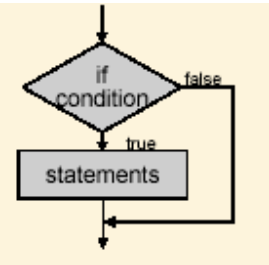
Các câu lệnh điều kiện:

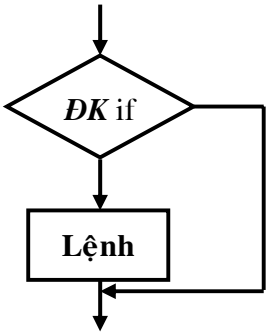
- Dạng 1:

```
if (biểu thức logic)
    lệnh
end
```

Lệnh chỉ thực hiện khi biểu thức logic nhận giá trị đúng. Nếu biểu thức logic này không thoả mãn thì lệnh không được thực hiện.

Lưu đồ:





Ví dụ:

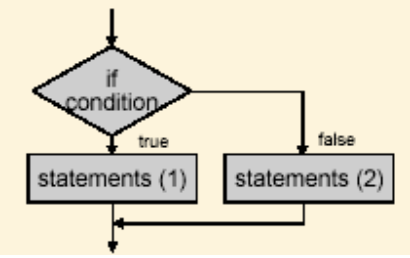
```
x = input('x = ');
if  x<0
x = -x;
end;
disp('|x | = ');
disp(x);
```

- Dạng 2:

```
if  (biểu thức logic)
        lệnh 1
else
        lệnh 2
end
```

Lệnh 1 thực hiện khi biểu thức logic đúng và ngược lại khi biểu thức logic sai thì lệnh 2 thực hiện.

Lưu đồ:



Ví dụ:

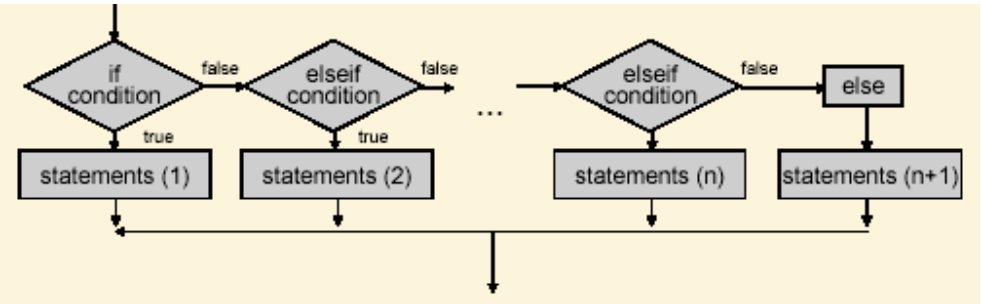
```
x = input('Nhập  x:')
if x<-1
y = abs(x+1)
else
y = abs(x-1)
end
disp('x=')
disp(x)
disp('y=')
disp(y)
```

- Dạng 3:

```
if (biểu thức logic 1)
        lệnh 1
elseif (biểu thức logic 2)
        lệnh 2
else
        lệnh 3
end
```

Ở dạng này lệnh 1 thực hiện khi biểu thức logic 1 thỏa mãn. Nếu biểu thức logic 1 không đúng, chương trình tiếp tục kiểm tra biểu thức logic 2, nếu đúng thực hiện lệnh 2 nếu sai sẽ thực hiện lệnh 3.

Lưu đồ:



Ví dụ:

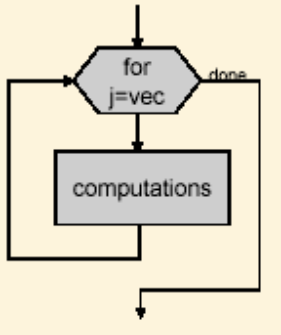
```
if temperature > 100
        disp('Too hot - equipment malfunctioning.')
elseif temperature > 90
        disp('Normal operating range.')
elseif temperature > 50
        disp('Below desired operating range.')
else
        disp('Too cold - turn off equipment.')
end
```

Vòng lặp ***for***:

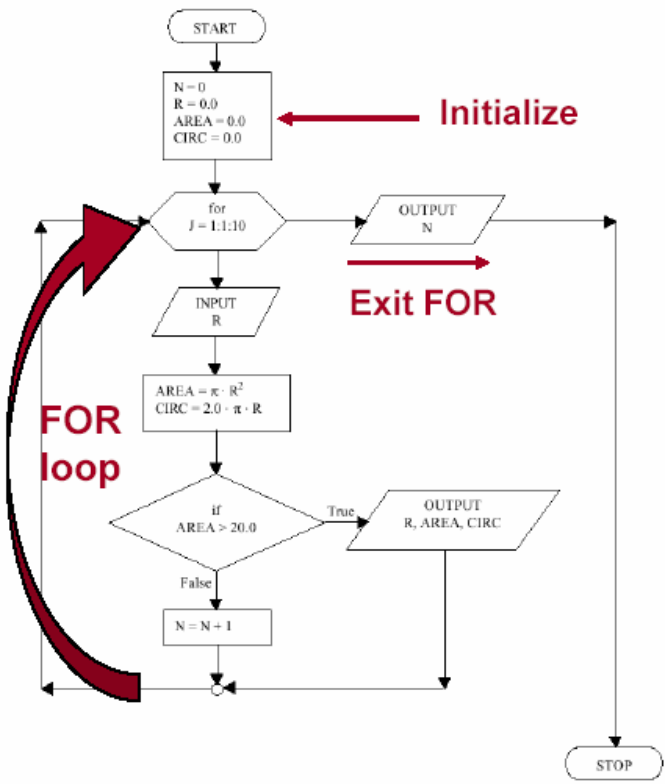
```
for index = start:increment:end
        statements
end
```

start và ***end*** lần lượt là giá trị bắt đầu và kết thúc của ***index, increment*** là bước nhảy, nếu không cài đặt bước nhảy thì MATLAB mặc định ***increment =1,*** mỗi lần lặp sẽ kết thúc khi gặp từ khóa ***end*** . Việc đầu tiên khi thực hiện vòng lặp ***for*** là MATLAB kiểm tra xem sẽ lặp bao nhiêu lần, sau đó mới bắt đầu thực thi vòng lặp.

Lưu đồ:



Ví dụ 1:



MATLAB code:

Ví dụ 2:

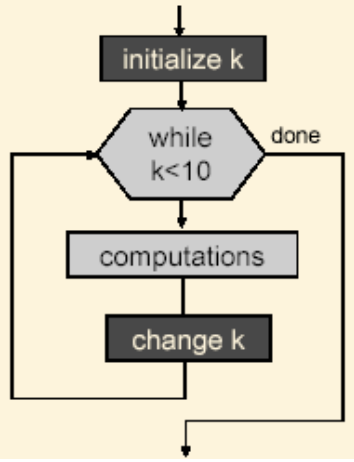
```
n = input('Enter n:');
Sn = 0;
for k = 1:1:n
        Sn = Sn + 1/k;
end;
disp('Sn = ');disp(Sn);
```

Vòng lặp ***while***:

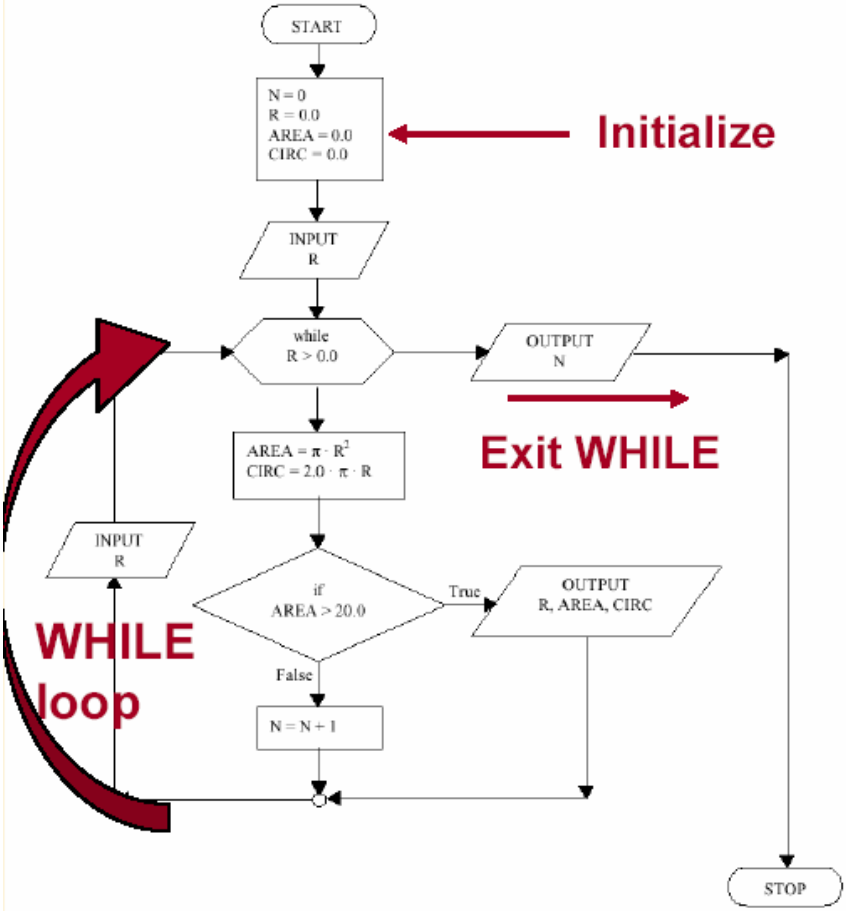
```
while ( điều kiện)
        lệnh
end
```

Đối với vòng lặp ***while*** thì số lần lặp không xác định, số lần lặp phụ thuộc vào ***điều kiện*** đúng hay sai. Trong khi điều kiện còn đúng thì vòng lặp vẫn còn tiếp tục vì vậy trong phần lệnh của vòng lặp ***while*** luôn có điều kiện để dừng vòng lặp. Trong quá trình xây dựng và chạy thử chương trình nếu rơi vào vòng lặp vô hạn thì bạn nhấn tổ hợp phím: ***Ctrl+(pause/brreak)*** để kết thúc chương trình.

Lưu đồ:



Ví dụ 1:



MATLAB code:

Ví dụ 2:

```
n = input('Enter n:');
Sn = 0; k = 0;
while k<= n
    k = k + 1;
    Sn = Sn + 1/k;
end
disp('Sn = '); disp(Sn);
```

Cấu trúc switch ... case

```
switch (biểu thức)
    case điều kiện_1
        Lệnh 1
    case điều kiện_2
        Lệnh 2
    ....
    case điều kiện_n
        Lệnh n
    otherwise
        Lệnh (n+1)
end
```

Ví dụ:

```
fprintf(' \n');
fprintf('Select a case:\n');
fprintf('==============\n');
fprintf(' 1 - pi\n');
fprintf(' 2 - e \n');
fprintf(' 3 - i \n');
fprintf('==============\n');
n = input('');
switch n
    case 1
        disp('Pi = ');disp(pi);
    case 2
        disp('e = ');disp(exp(1));
    case 3
        disp('i = ');disp(i);
    otherwise
        disp('Nothing to display');
end
```

ĐỒ HỌA TRONG MATLAB

Các bước đồ họa:

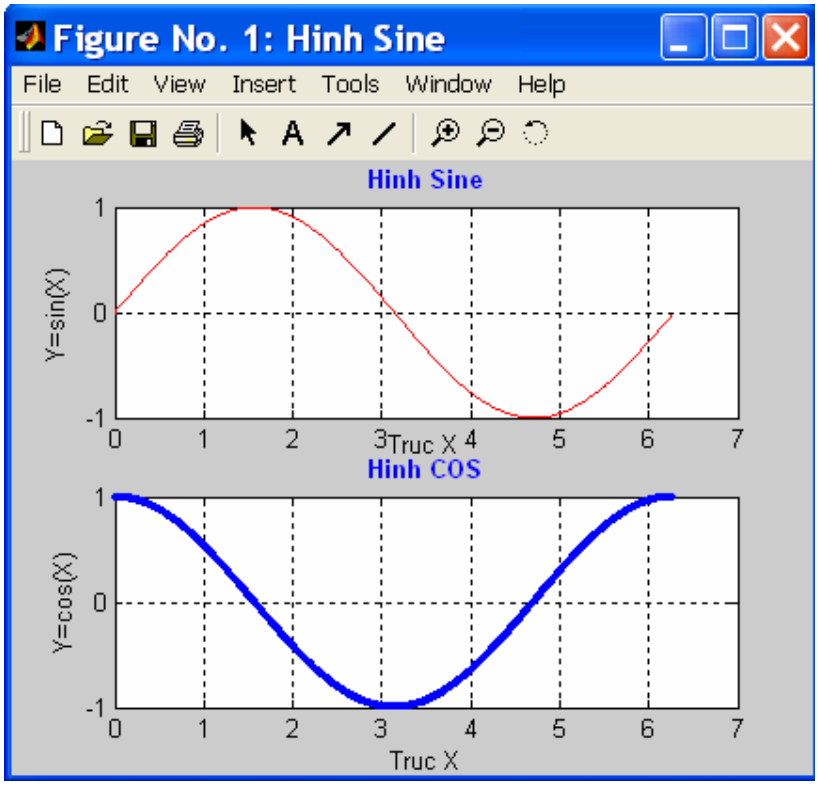
Bước	MATLAB code
1. Chuẩn bị dữ liệu	X= 0:0.1:2*pi; Y= sin(X);
2. Chọn cửa sổ và vị trí của đối tượng đồ họa trong cửa sổ.	figure('Name','Hinh sine');
3. Gọi hàm vẽ đồ thị	subplot(2,2,1); h =plot(X,Y);
4. Xác định thuộc tính cho hình như kiểu đường, màu sắc, ...	set(h, 'LineStyle','-','Color','r')
5. Xác định các thuộc tính của axis, axes, lưới vẽ, ...	axis([-1 6 -1.2 1.2]); grid on;

6. Chú thích cho đồ thị: labels, legend, text, ...	xlabel('X');
	ylabel('Y=SIN(X)');
	title('Đồ thị hình sine');

7. Xuất kết quả	

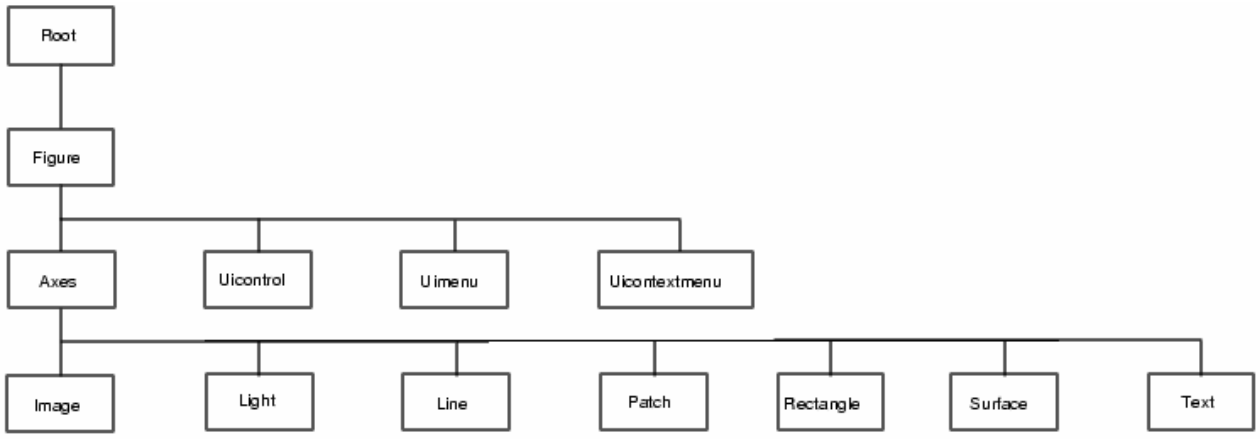
Kết quả các bước trên ta có đồ thị như sau:

MATLAB code	Đồ thị

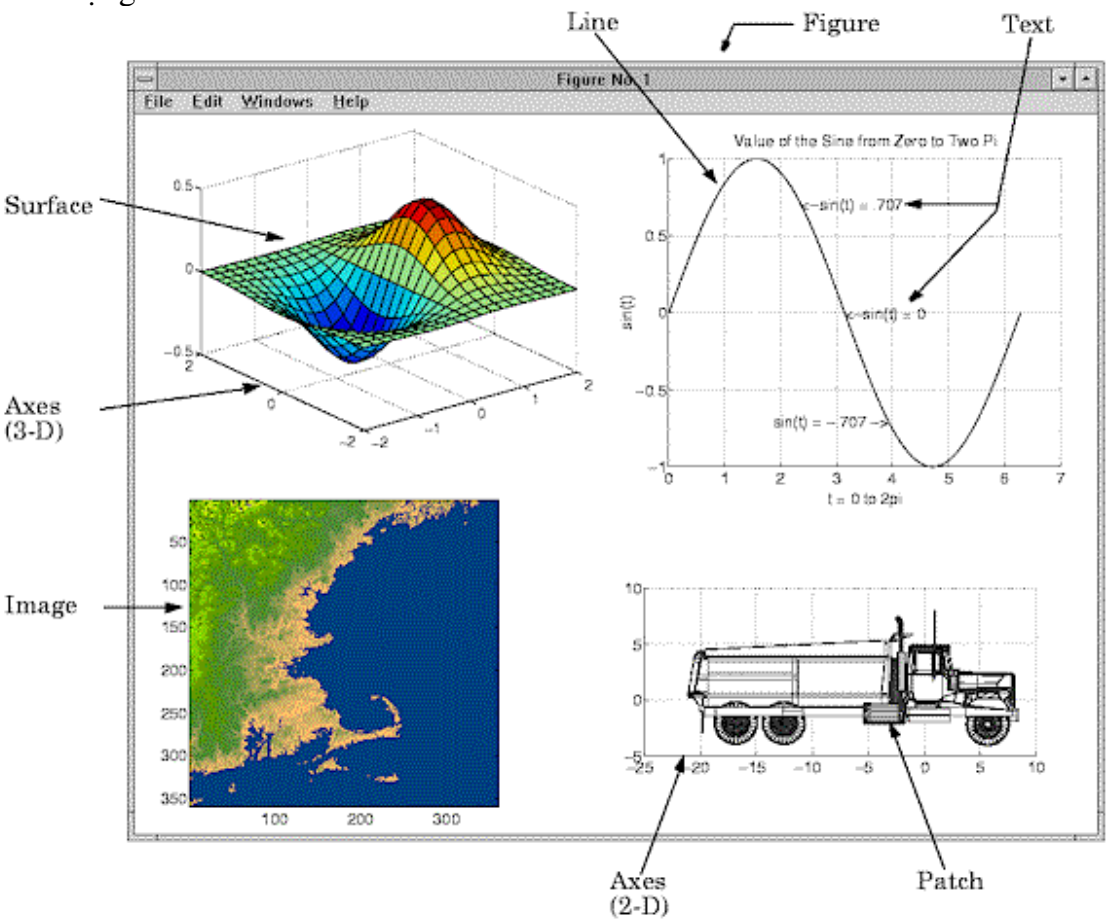
X=0:0.01:2*pi;
Y1=sin(X);
Y2=cos(X);
figure('Name','Hinh Sine');
Subplot(2,1,1);
h=plot(X,Y1);
set(h,'linestyle','-','Color','r');
xlabel('Truc X');
ylabel('Y=sin(X)');
title('Hinh Sine');
grid on
subplot(2,1,2);
h=plot(X,Y2);
set(h,'linestyle','.','Color','b');
xlabel('Truc X');
ylabel('Y=cos(X)');
title('Hinh COS');
	

grid on;

Mô hình cây thư mục quản lý đối tượng đồ họa trong MATLAB



Các đối tượng của MATLAB



Lệnh ***figure***

Cú pháp:

figure

figure('PropertyName',PropertyValue,...)
figure(h)
h = figure(...)

Chú thích:

- ***figure***: tạo một đối tượng đồ họa, đối tượng đồ họa ở đây là cửa sổ để quản lý tất cả các kết quả của MATLAB khi xuất ra dưới dạng đồ họa.
- ***figure('PropertyName',PropertyValue,...)***: Cửa sổ với những thuộc tính và giá trị mà người sử dụng sẽ cài đặt trong cửa sổ này.
- figure(h): Thực hiện một trong hai nhiệm vụ sau:

- Kiểm tra xem figure(h) tồn tại hay chưa.
- Nếu chưa có thì tạo một cửa sổ mới còn nếu có rồi (đã tồn tại) thì làm figure này hiện hành.

Một vài property (thuộc tính) của figure:

- 'Position' : Xác định vị trí của figure ở đâu trên màn hình
- 'Name' : Tên của figure
- 'NumberTitle': Số của thứ tự của figure
- 'Color': màu nền của figure

Các lệnh liên quan đến ***figure***:

clf: Xóa cửa sổ (figure) hiện hành.

gcf: Trả về cửa sổ (figure) hiện hành của figure hiện hành.

close(h): Đóng figure thứ h với h là một con số.

close all: Đóng tất cả các figure.

get(0,'ScreenSize'): Lấy chế độ phân giải màn hình, dùng để định vị vị trí của figure trên màn hình.

Ví dụ:

scrsz = get(0,'ScreenSize');
figure('Position',[1 scrsz(4)/2 scrsz(3)/2 scrsz(4)/2])

set(0,'DefaultFigureProperty',PropertyValue...): Thiết lập giá trị hay thuộc tính mặc định cho figure.

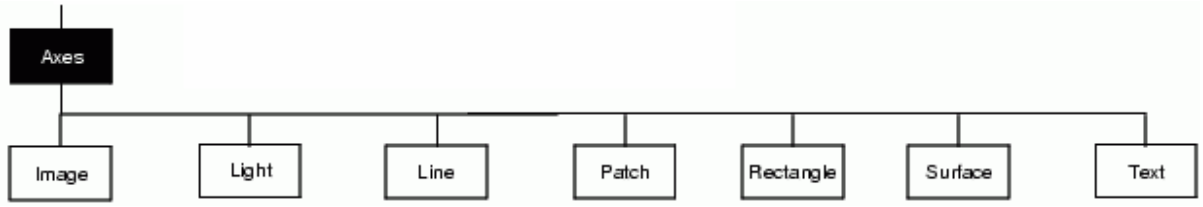
Lệnh ***axes***

Cú pháp:

axes
axes('PropertyName',PropertyValue,...)
axes(h)
h = axes(...)

Chú thích:

Lệnh ***axis*** cho phép người dùng tạo ra một hệ trục mới để chứa đối tượng đồ họa khác như : line, text, ...



Properties : thuộc tính của axes như : box(on/ off), Position([left bottom width height]), Visible(on / off), ...

Các lệnh liên quan đến axes:

cla: Xóa axes hiện hành.

gca: Trả về axes hiện hành của figure hiện hành.

set(0,'DefaultAxesPropertyName',PropertyValue,...): xác lập lại giá trị mặc định cho axes gốc.

set(gcf,'DefaultAxesPropertyName',PropertyValue,...): xác lập lại giá trị mặc định cho axes của cửa sổ hiện hành.

Lệnh ***subplot***:

Cú pháp:

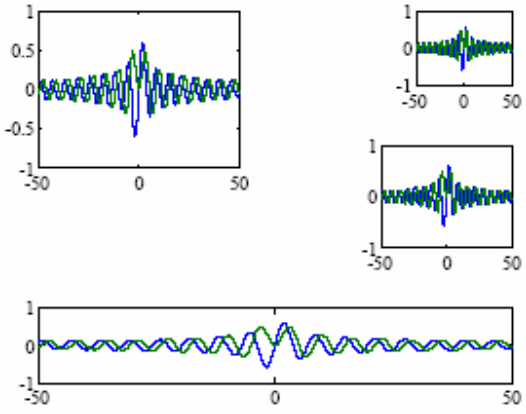
```
subplot(m,n,p)
subplot(h)
subplot('Position',[left bottom width height])
h = subplot(...)
```

Chú thích:

- ***subplot(m,n,p)***: Tạo ra một axes ở vị trí thứ p khi chia cửa sổ ra thành m xn hình chữ nhật và axes này trở thành axes hiện hành.

MATLAB code:

```
%% Tạo dữ liệu
x=linspace(-50,50,250);
y1=real(besselj(1,x));
y2=real(besselj(2,x));
grid on;
% Vẽ hình ứng với dữ liệu trên
subplot(2,2,1), plot(x,y1,x,y2)
subplot(3,3,6), plot(x,y1,x,y2)
subplot(4,4,4), plot(x,y1,x,y2)
subplot(4,1,4), plot(x,y1,x,y2)
```



- ***subplot(h):*** tạo axes ứng với h làm hiện hành.
- ***subplot('Position',[left bottom width height])***: Thiết lập vị trí cho axes

Lệnh ***hold***: Dùng để vẽ được nhiều hình trên cùng một axes gồm có :

hold on : Cho phép vẽ nhiều hình trên một axes.

hold off : kết thúc việc cho vẽ nhiều hình trên một axes

Lệnh ***title***: Tạo tiêu đề cho đồ thị

Cú pháp lệnh:

title('string')

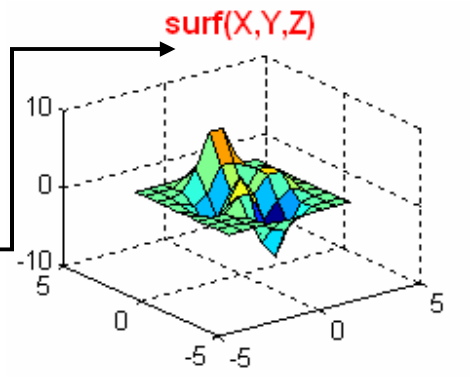
title(fname)

title(...,'PropertyName',PropertyValue,...)

Ví dụ:

Matlab code:

```
clear all;
clc;
[X,Y,Z]=peaks(10);
surf(X,Y,Z);
title('surf(X,Y,Z)');
```



Lệnh ***text***: tạo đối tượng chữ ở một vị trí xác định trong hệ trục tọa độ.

Cú pháp:

text(x,y,'string')

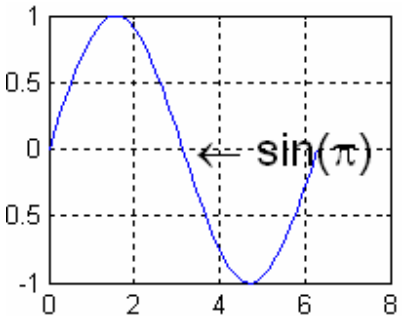
text(x,y,z,'string')

text(...'PropertyName',PropertyValue...)

Ví dụ:

Matlab code:

```
plot(0:pi/20:2*pi,sin(0:pi/20:2*pi));
text(pi,0,' \leftarrow sin(\pi)','FontSize',18);
grid on;
```



Lệnh **xlabel, ylabel, zlabel** : tạo nhãn cho các hệ truc x, y, z

Lệnh ***axis***: Xác định giới hạn trục của các trục tọa độ hoặc tắt các hệ gồm các lệnh sau

axis([xmin xmax ymin ymax zmin zmax)

axis on/ off

axis square

axis auto

axis equal

Lệnh ***grid on/ off*** : bậc tắt lưới của axes hiện hành

LỆNH ĐỒ HỌA 2D

1. Lệnh ***plot***:

Cú pháp:

```
plot(Y)
plot(X1,Y1,...)
plot(X1,Y1,LineSpec,...)
plot(...,'PropertyName',PropertyValue,...)
h = plot(...)
```

Chú thích:

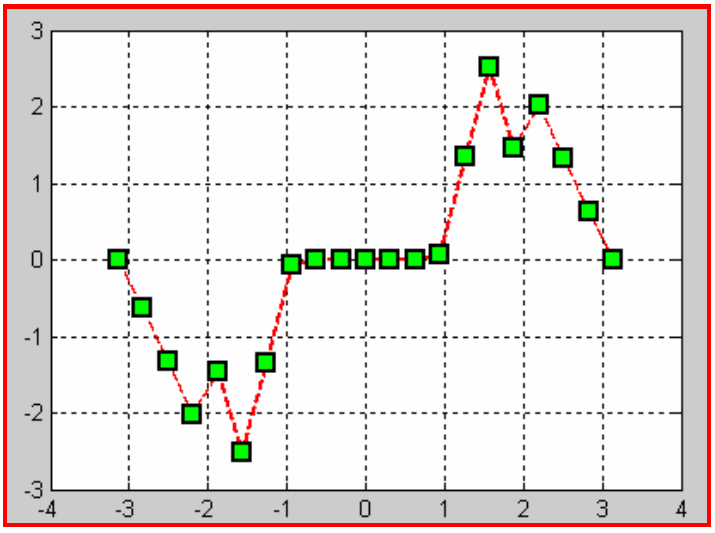
- ***plot(Y)*** : Nếu Y đều là số thực thì *plot(Y)* xây dựng đồ thị dựa trên tập dữ liệu *X=0:size(Y,2)* và Y còn Y là số phức thì *plot(Y)* sẽ xây dựng đồ thị dựa vào tập dữ liệu sau: *X= real(Y), Y=imag(Y)*
- ***plot(X1,Y1,...):*** đồ thị được xây dựng trên tập dữ liệu X, Y
- ***plot(X1,Y1,LineSpec,...):*** Giống như lệnh *plot(X,Y)* tuy nhiên ***LineSpec*** hổ trợ ta định nghĩa (LineStyle, LineWidth,Color, Maker)
- ***plot(...,'PropertyName',PropertyValue,...):*** Thiết lập thuộc tính cho đồ thị

Ví dụ:

MATLAB code:

```
>> x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'--rs','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10)
>> grid on
```

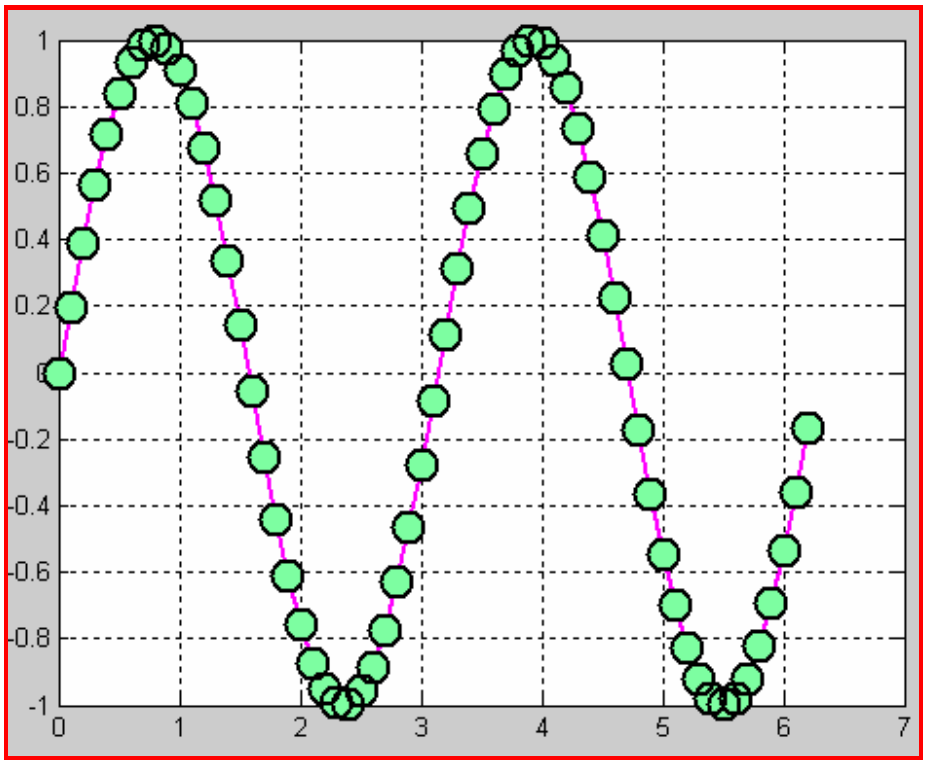
Đồ thị:



Ví dụ 2:

```
t=0:0.1:2*pi;
plot(t,sin(2*t),'-mo',...
                'LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor',[.49 1 .63],...
                'MarkerSize',12);
grid on
```

Đồ thị:



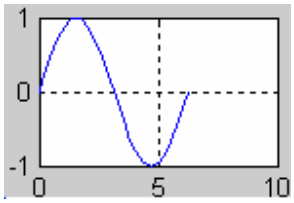
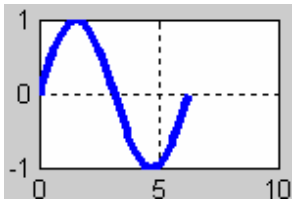
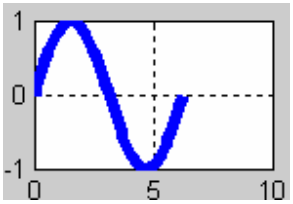
LineSpec: gồm các loại sau

- LineStyle
- LineWidth
- Color
- Marker(type, size, face, Edge)

LineStyle:

Kí hiệu	Kiểu đường	Ví dụ
-	solid line (Mặc định)	———
--	dashed line	– – – – –
:	dotted line	.....
-.	dash-dot line	– . – . –

Linewidth(LW):

Giá trị LW	1	3	5
Hình			

Color:

Có ba cách trong MATLAB xác định màu sắc cho đối tượng đồ họa là:

- RGB Value: Nhập giá trị màu sắc thông qua ba giá trị
- ShortName: Theo tên viết tắc
- LongName: Theo tên đầy đủ của màu

Bảng màu tương ứng với ba kiểu trên

RGB Value	Short Name	Long Name
[1 1 0]	y	yellow
[1 0 1]	m	magenta
[0 1 1]	c	cyan
[1 0 0]	r	red
[0 1 0]	g	green
[0 0 1]	b	blue
[1 1 1]	w	white
[0 0 0]	k	black

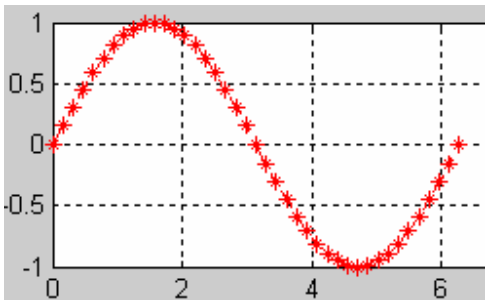
Maker:

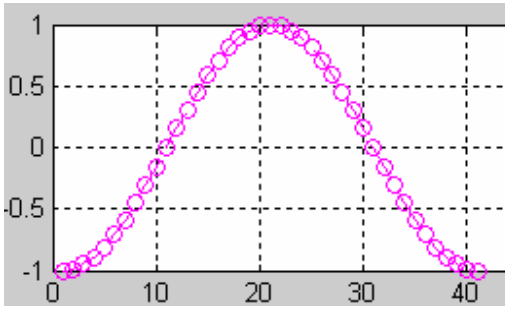
Sau đây là các marker trong MATLAB

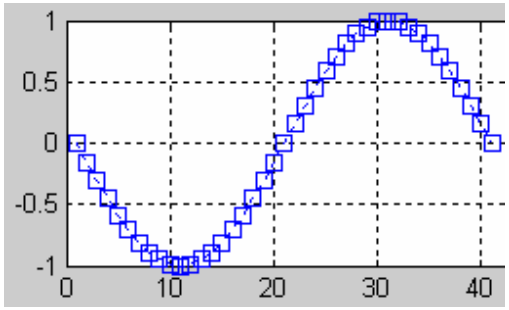
Kí hiệu	Ý nghĩa
+	Dấu cộng
o	Hình tròn
*	Dấu hoa thị
.	Điểm
x	Dấu chéo nhau
s	Hình vuông
d	Hình thoi
^	Tam giác trên
v	Tam giác dưới
>	Tam giác bên phải
<	Tam giác bên trái
p	Sao năm cạnh
h	Sao sáu cạnh

Ví dụ:

MATLAB code	Kết quả

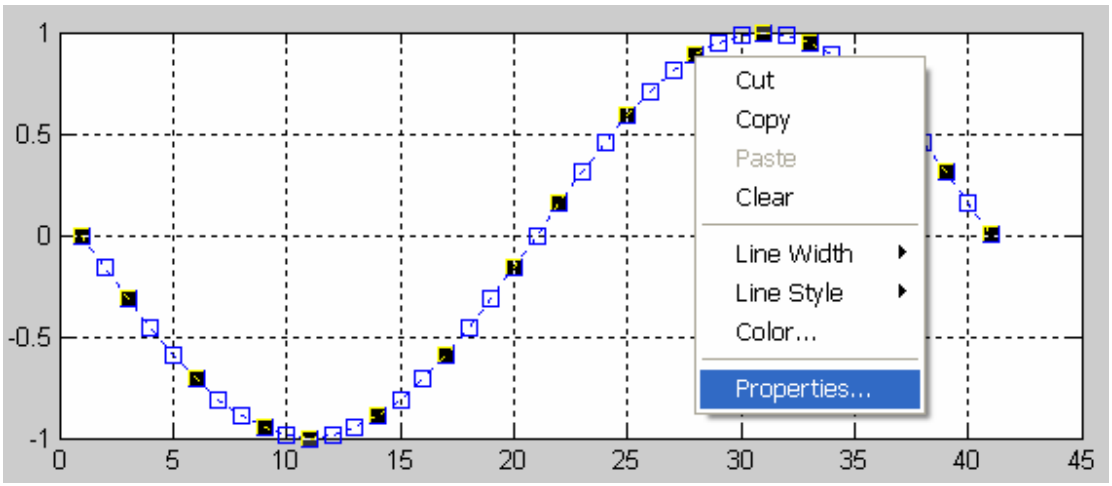
`>> t = 0:pi/20:2*pi;`
`plot(t,sin(t),'-.r*')`
`>> grid on;`
	

`>> t = 0:pi/20:2*pi;`
`>> plot(sin(t-pi/2),'--mo')`
`>> grid on`
	

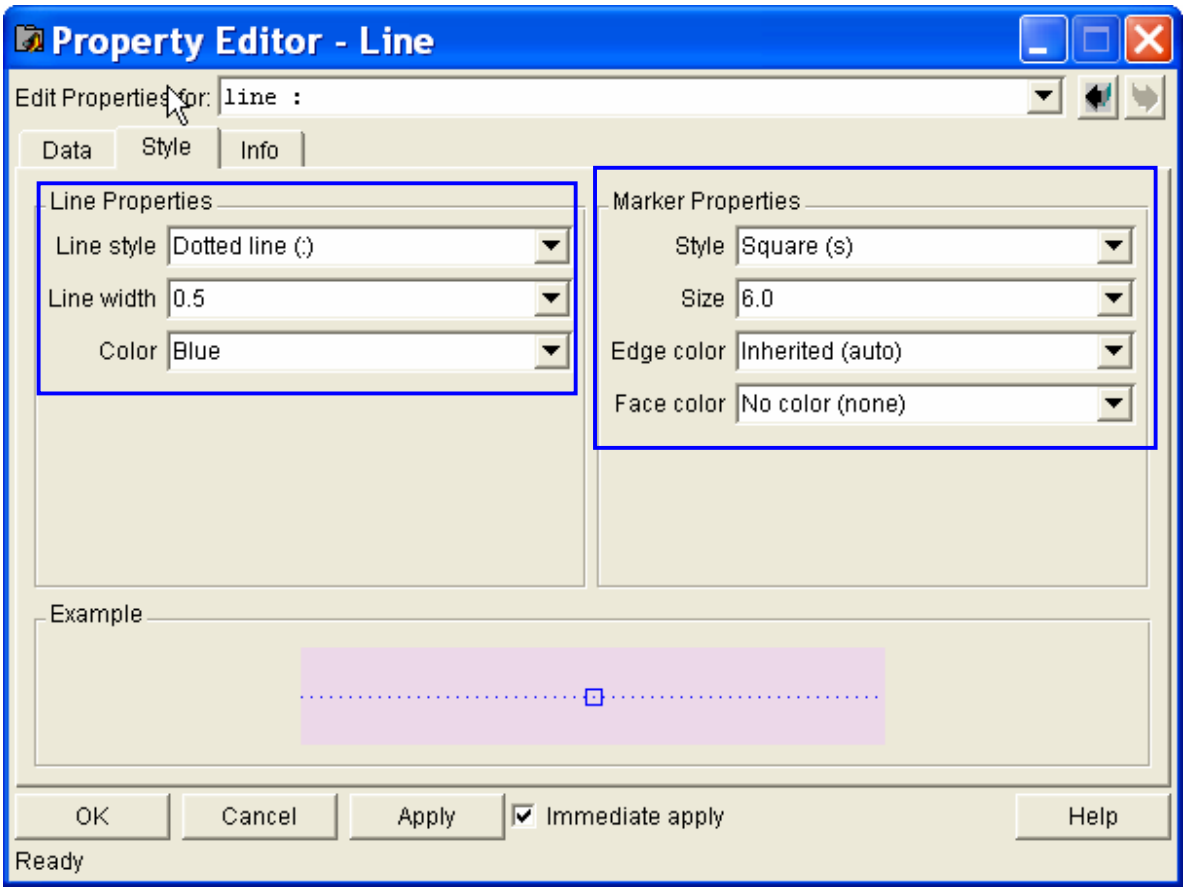
`>> t = 0:pi/20:2*pi;`
`>> plot(sin(t-pi),':bs')`
`>> grid on`
	

Ngoài ra ta còn dùng các thuộc tính hổ trợ cho marker: 'MarkerEdgeColor', 'MarkerFaceColor', 'MarkerSize'.

Để hiệu chỉnh thuộc tính của một đối tượng trực tiếp, ta chọn properties... của đối tượng cần hiệu chỉnh



Hiện thị hộp thoại Properties như sau:



2. *Lệnh Line:* Dùng để vẽ đường thẳng

Cú pháp:

line(X,Y)

line(X,Y,Z)

line(X,Y,Z,'PropertyName',PropertyValue,...)
line('PropertyName',PropertyValue,...)
h = line(...)

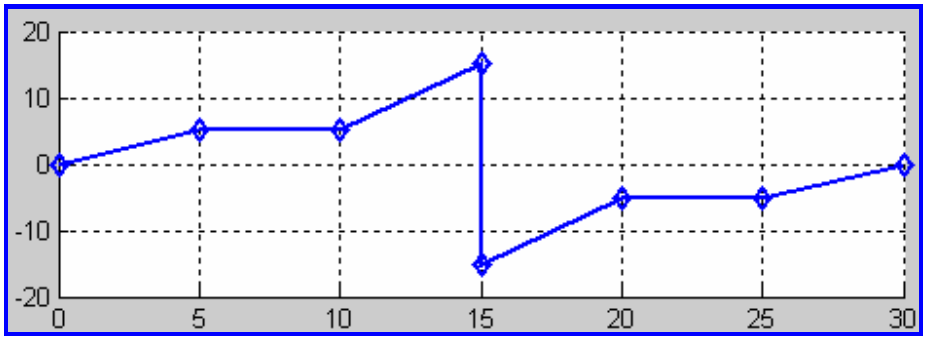
Chú thích:

Cũng giống như lệnh plot đã đề cập ở trên

Ví dụ:

MATLAB code

```
>> x=[0 5 10 15 15 20 25 30];
>> y=[0 5 5 15 -15 -5 -5 0];
>> line(x,y,'Linewidth',2,'LineStyle','-','Marker','d');
>> grid on
```



3. Lệnh vẽ ***plotyy***:

Cú pháp:

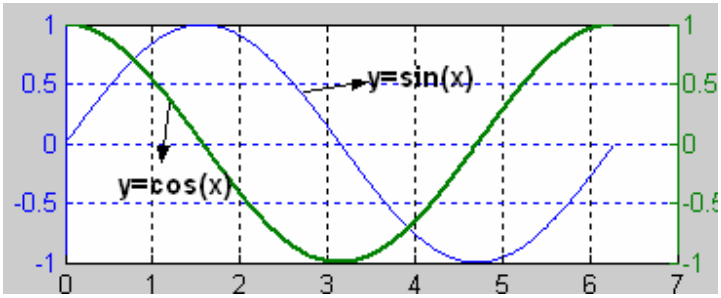
plotyy(X1,Y1,X2,Y2)
plotyy(X1,Y1,X2,Y2,'function')
plotyy(X1,Y1,X2,Y2,'function1','function2')
[AX,H1,H2] = plotyy(...)

Chú thích:

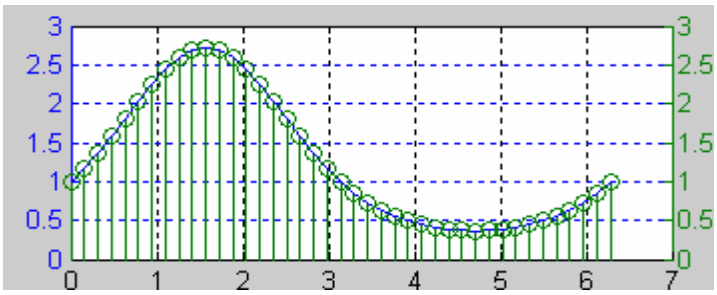
- ***plotyy(X1,Y1,X2,Y2):*** Lệnh vẽ hai hình trên một axes, cùng trục x và hai trục y khác nhau. Hai hình trên ứng với hai tập dữ liệu (X1, Y1) và (X2, Y2) .
- ***plotyy(X1,Y1,X2,Y2,'function')***: Tập dữ liệu trên có thể dùng các hàm (*function*) để vẽ như: ***plot, semilogx, semilogy, loglog, stem*** hoặc bất cứ hàm nào của MATLAB thỏa mãn: **h= function(x,y)**
- ***plotyy(X1,Y1,X2,Y2,'function1','function2')*** : function1(X1, Y1), function (X2, Y2) với function1 và function2 là các hàm đã đề cập ở trên.
- ***[AX,H1,H2] = plotyy(...)*** : Trả về các axes chứa H1 và H2 trong vec tơ AX. AX(1) ứng với H1 nằm bên trái và AX(2) ứng với H2 nằm bên phải.

Ví dụ:

```
>> axis([-1 10 -1.5 1.5]);
>> x=0:0.01:2*pi;
>> yc=cos(x);
>> ys=sin(x);
>> plotyy(x,ys,x,yc);
>> grid on;
```




```
>> t = 0:pi/20:2*pi;
y = exp(sin(t));
plotyy(t,y,t,y,'plot','stem')
>> grid on
```



4. Lệnh ***semilogx*** và ***semilogy***:

Cú pháp:

semilogx(Y)
semilogx(X1,Y1,...)
semilogx(X1,Y1,LineSpec,...)
semilogx(...,'PropertyName',PropertyValue,...)
h = semilogx(...)

Chú thích:

Ví dụ:

5. Lệnh ***stem***:

Cú pháp:

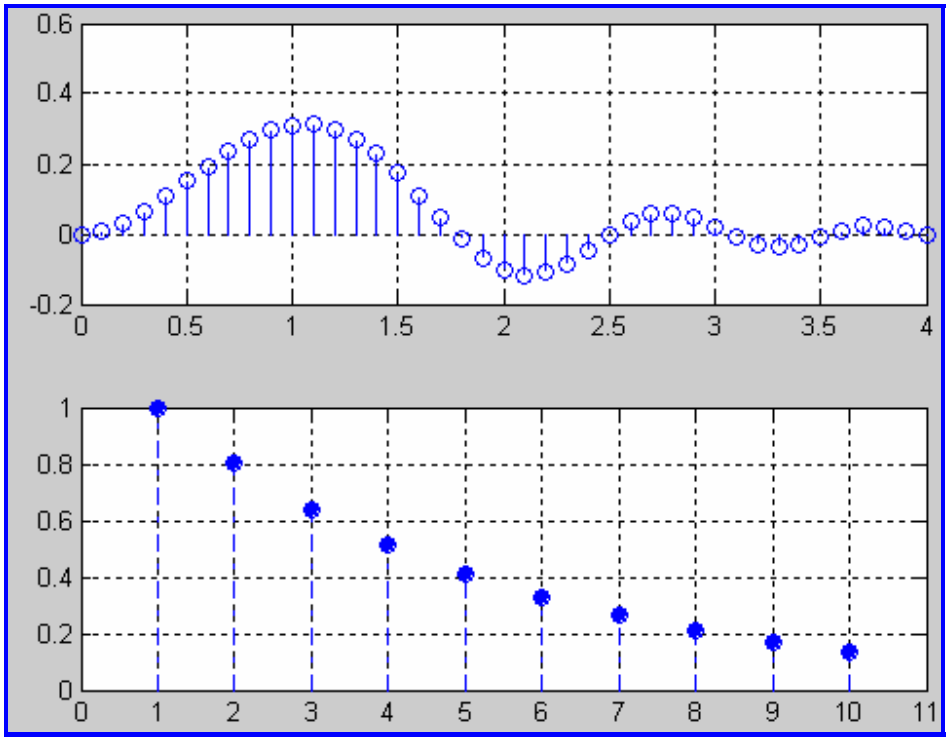
stem(Y)
stem(X,Y)
stem(...,'fill')
stem(...,LineSpec)
h = stem(...)

Chú thích:

- stem(Y):
- stem(X, Y)
- stem(...,'fill')
- stem(...,LineSpec)

Ví dụ 1:

```
subplot(2,1,1);
x = 0:0.1:4;
y = sin(x.^2).*exp(-x);
stem(x,y);grid
subplot(2,1,2);
y = linspace(0,2,10);
stem(exp(-y),'fill','-.')
axis ([0 11 0 1]);
grid
```



6. Lệnh ***loglog***:

Cú pháp:

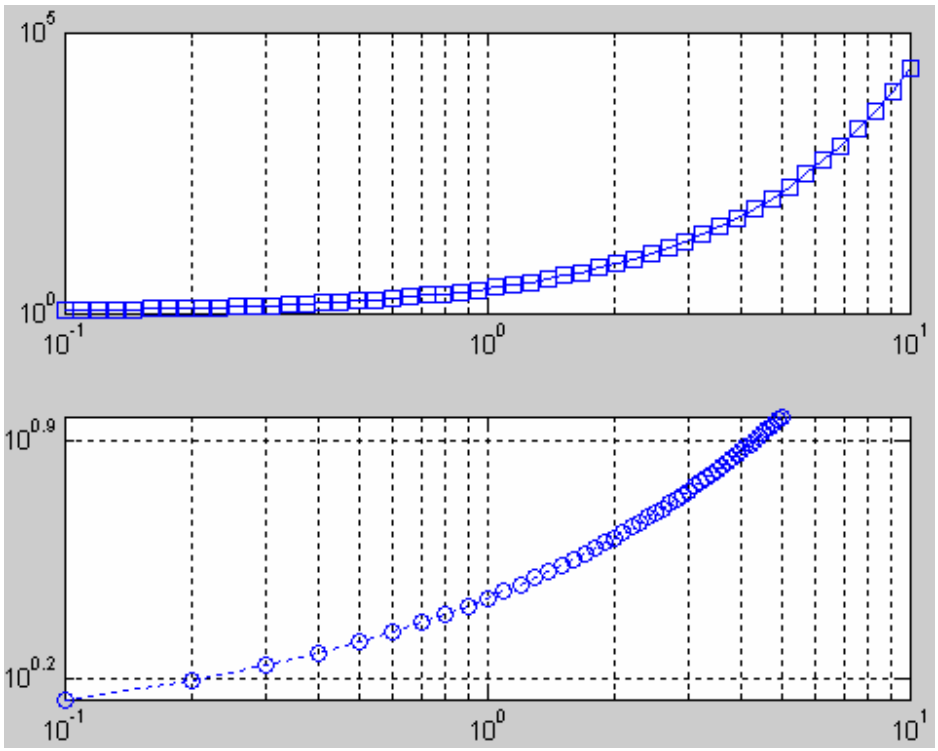
loglog(Y)
loglog(X1,Y1,...)
loglog(X1,Y1,LineSpec,...)
loglog(...,'PropertyName',PropertyValue,...)
h = loglog(...)

Ví dụ:

MATLAB code:

```
subplot(2,1,1);
x = logspace(-1,1);
loglog(x,exp(x),'-s')
grid on
```

```
subplot(2,1,2);
x=0:0.1:5;
y=exp(sqrt(x));
loglog(x,y,':o');
grid
```



7. Lệnh ***bar*** và ***barh***:

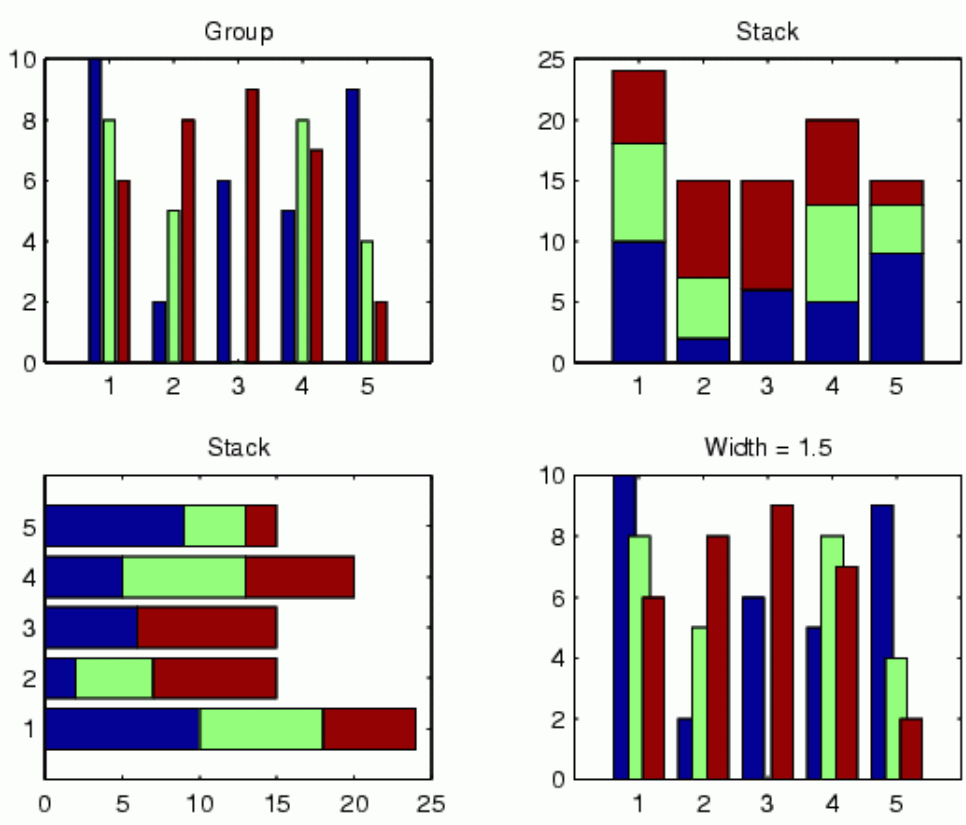
Cú pháp lệnh:

```
bar(Y)
bar(x,Y)
bar(...,width)
bar(...,'style')
bar(...,LineSpec)
[xb,yb] = bar(...)
h = bar(...)
barh(...)
[xb,yb] = barh(...)
h = barh(...)
```

Ví dụ:

MATLAB code:

```
Y =
round(rand(5,3)*10);
subplot(2,2,1)
bar(Y,'group')
title 'Group'
subplot(2,2,2)
bar(Y,'stack')
title 'Stack'
subplot(2,2,3)
barh(Y,'stack')
title 'Stack'
subplot(2,2,4)
bar(Y,1.5)
title 'Width = 1.5'
```



8. Lệnh ***area***:

9. Lệnh ***pie***:

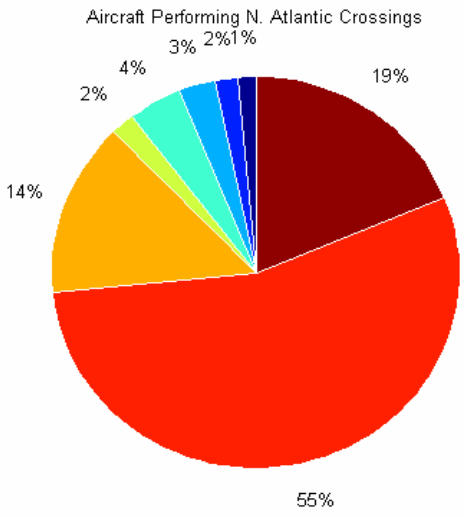
Cú pháp:

```
pie(X)
pie(X,explode)
h = pie(...)
```

Ví dụ:

MATLAB code:

```
% acft = char('A310','A330','MD11','DC-10', 'L1011',...
% 'B747','B767','B777');
numbers=[12 15 24 35 16 120 456 156];
pie(numbers)
% for i=1:8 % array of strings
% gtext(acft(i,:)); % get text from char variable
% end
title('Aircraft Performing N. Atlantic Crossings')
```



10. Lệnh ***histograms***: Gồm hai hàm vẽ là : ***hist*** và ***rose***

Cú pháp lệnh ***hist***:

```
n = hist(Y)
n = hist(Y,x)
n = hist(Y,nbins)
[n,xout] = hist(...)
```

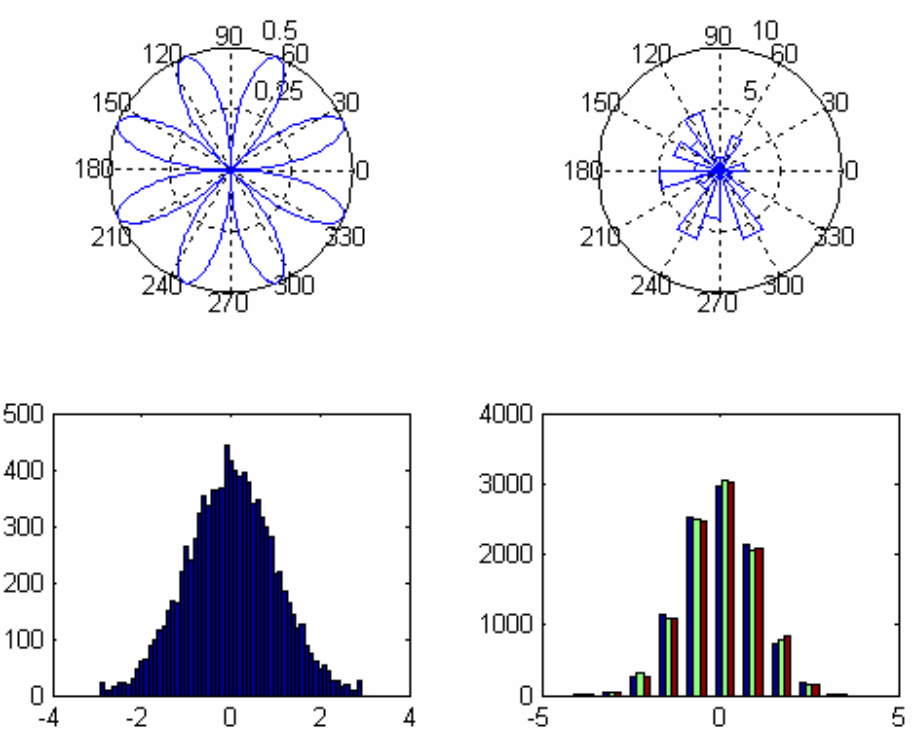
Cú pháp lệnh ***rose***:

```
rose(theta)
rose(theta,x)
rose(theta,nbins)
[tout,rout] = rose(...)
```

Ví dụ:

MATLAB code:

```
subplot(2,2,1);
% Polar plot
t=0:.01:2*pi;
polar(t,sin(2*t).*cos(2*t));
subplot(2,2,2);
theta = 2*pi*rand(1,50);
rose(theta);
subplot(2,2,3);
x = -2.9:0.1:2.9;
y = randn(10000,1);
hist(y,x) ;
subplot(2,2,4);
Y = randn(10000,3);
hist(Y);
```



LỆNH ĐỒ HỌA 3D

1. Lệnh ***plot3***:

Cú pháp:

```
plot3(X1,Y1,Z1,...)
plot3(X1,Y1,Z1,LineSpec,...)
plot3(...,'PropertyName',PropertyValue,...)
h = plot3(...)
```

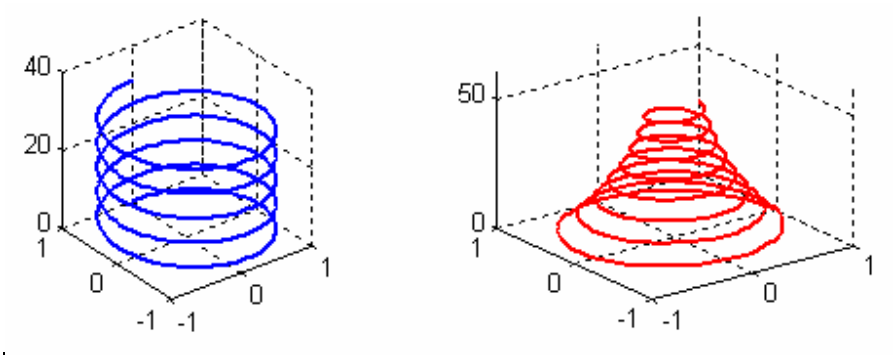
Chú thích: Tất cả các cú pháp này điều giống lệnh plot tuy nhiên plot3 có thêm tập dữ liệu theo phương z

Ví dụ: .

MATLAB code:

```
subplot(2,2,1);
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
grid on
axis square;
subplot(2,2,2);
t= 0:0.1:16*pi;
x=exp(-0.03*t).*cos(t);
y=exp(-0.03*t).*sin(t);
z=t;
```

```
plot3(x, y, z)
grid on
```



2. Lệnh ***meshgrid***:

3. Lệnh ***peak***: một ví dụ của hàm hai biến dùng để tạo ra bộ dữ liệu minh họa cho các lệnh như surf, mesh, surface,...

Cú pháp:

```
Z = peaks;
Z = peaks(n);
Z = peaks(V);
Z = peaks(X,Y);
peaks;
peaks(N);
peaks(V);
peaks(X,Y);
[X,Y,Z] = peaks;
[X,Y,Z] = peaks(n);
[X,Y,Z] = peaks(V);
```

Chú thích:

- Z = peaks: Trả về một ma trận cỡ 49 x 49
- Z = peaks(n): Trả về một ma trận vuông cỡ n.
- Z=peaks(V): Trả về một ma trận vuông cỡ n= length(V).
- Z = peaks(X,Y) : Trả về ma trận z dựa vào dữ liệu (X, Y).
- [X,Y,Z]peaks(...): Như các lệnh peaks trên chỉ khác là có thêm giá trị X, Y.

Ví dụ:

```
>> Z=peaks(5)
Z =
    0.0001    0.0042   -0.2450   -0.0298   -0.0000
   -0.0005    0.3265   -5.6803   -0.4405    0.0036
   -0.0365   -2.7736    0.9810    3.2695    0.0331
   -0.0031    0.4784    7.9966    1.1853    0.0044
    0.0000    0.0312    0.2999    0.0320    0.0000
```

```
>> [x,y,z]=peaks(2)
x =
    -3     3
    -3     3
y =
    -3    -3
     3     3
z =
   1.0e-004 *
   0.6671   -0.0586
   0.3224    0.4103
```

4. Lệnh ***mesh***:
5. Lệnh ***surf*** và ***surfc***:

Cú pháp:

surf(Z)
surf(X,Y,Z)
surf(X,Y,Z,C)
surf(...,'PropertyName',PropertyValue)
surfc(...)
h = surf(...)
h = surfc(...)

Chú thích:

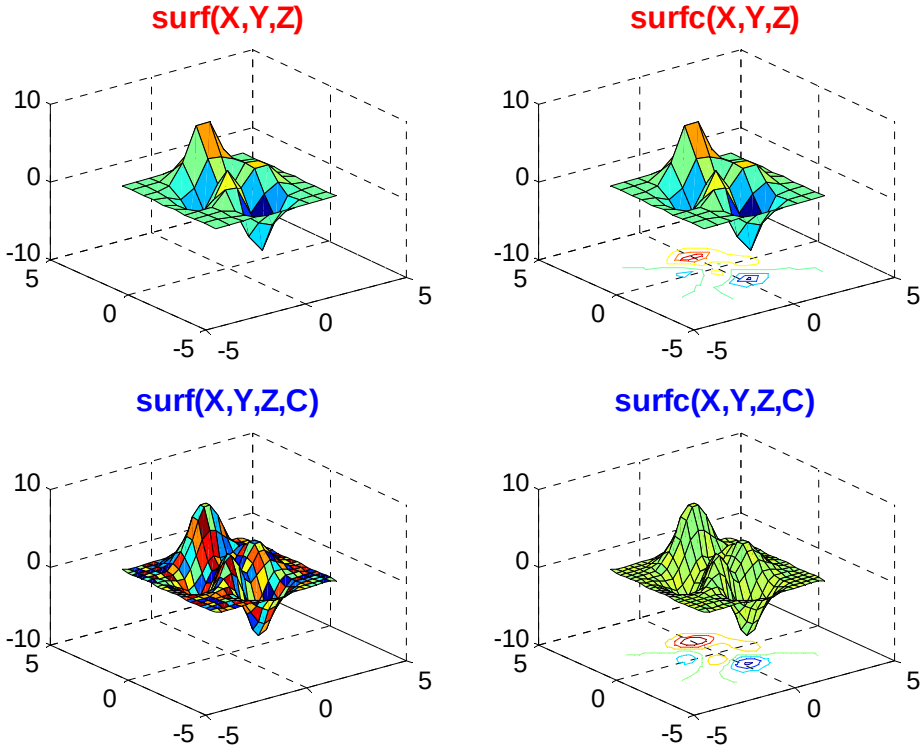
- ***surf(Z):*** tạo mặt đổ bóng 3D từ ma trận Z với bộ dữ liệu X=1:m, Y=1: n trong đó [m, n] = size(Z). Chiều cao là dữ liệu từ ma trận Z cũng là dữ liệu cho việc điều chỉnh màu sắc của mặt đổ bóng 3D này.
- ***surf(X, Y, Z):*** Tạo mặt đổ bóng 3D với bộ dữ liệu (X, Y, Z), màu sắc của mặt được xác định thông qua ma trận Z.
- ***surf(X,Y,Z,C):*** Như surf(X,Y,Z), chỉ khác C là ma trận cho dữ liệu màu của mặt đổ bóng 3D.
- ***surfc(...)*** : Giống như lệnh ***surf()*** chỉ khác là có thêm các đường countour của mặt 3D trên mặt phẳng OXY.

Ví dụ:

MATLAB code:

```
clear all; clc;
subplot(2,2,1);
[X,Y,Z]=peaks(10);
surf(X,Y,Z);
title('surf(X,Y,Z)');
subplot(2,2,2);
[X,Y,Z]=peaks(10);
```

```
surfc(X,Y,Z);
title('surfc(X,Y,Z)');
subplot(2,2,3);
[X,Y,Z]=peaks(20);
C=rand(20);
surf(X,Y,Z,C);
title('surf(X,Y,Z,C)');
subplot(2,2,4);
[X,Y,Z]=peaks(20);
C=rand(20);
surfc(X,Y,Z,C);
title('surfc(X,Y,Z,C)');
```



6. Lệnh ***sphere***:

Cú pháp:

```
sphere
sphere(n)
[X,Y,Z] = sphere(...)
```

Chú thích:

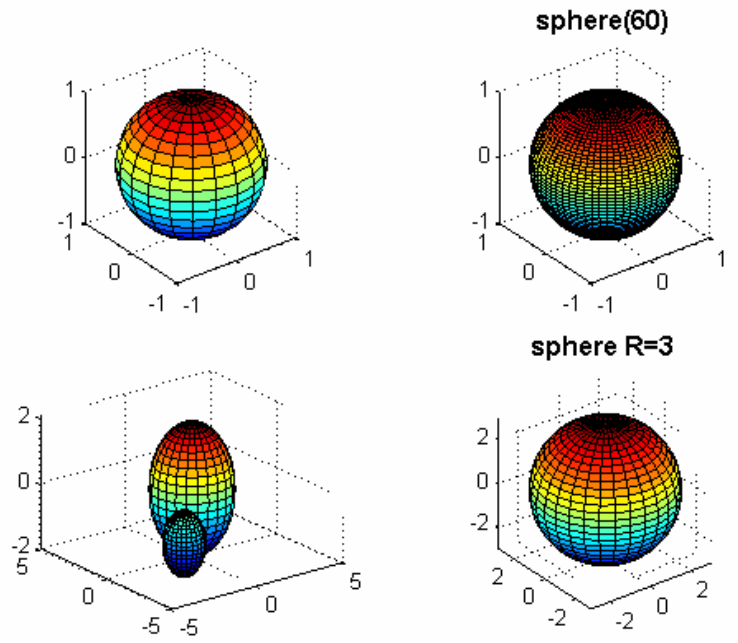
- ***sphere***: tạo hình cầu bằng việc kết hợp hai hàm surf và mesh.
- ***sphere(n):*** hình cầu được tạo thành bằng n mặt kết hợp lại với nhau

- ***[X,Y,Z] = sphere(...):*** Trả về bộ dữ liệu tạo nên hình cầu này bằng các lệnh ***surf(X, Y, Z)*** và ***mesh(X, Y, Z).***

Ví dụ:

MATLAB code:

```
subplot(2,2,1);
sphere;
axis equal;
grid on;
subplot(2,2,2)
sphere(60);
title('sphere(60)');
axis equal;
subplot(2,2,3);
[x,y,z]=sphere(24);
surf(x-2, y-2, z-1);
hold on
surf(2*x, 2*y,2*z);
% axis off
subplot(2,2,4);
[x,y,z]=sphere(30);
surf(3*x,3*y,3*z);
grid on;
title('sphere R=3');
axis equa;
```



7. Lệnh ***cylinder***:

Cú pháp:

[X,Y,Z] = cylinder
[X,Y,Z] = cylinder(r)
[X,Y,Z] = cylinder(r,n)
h= cylinder(...)

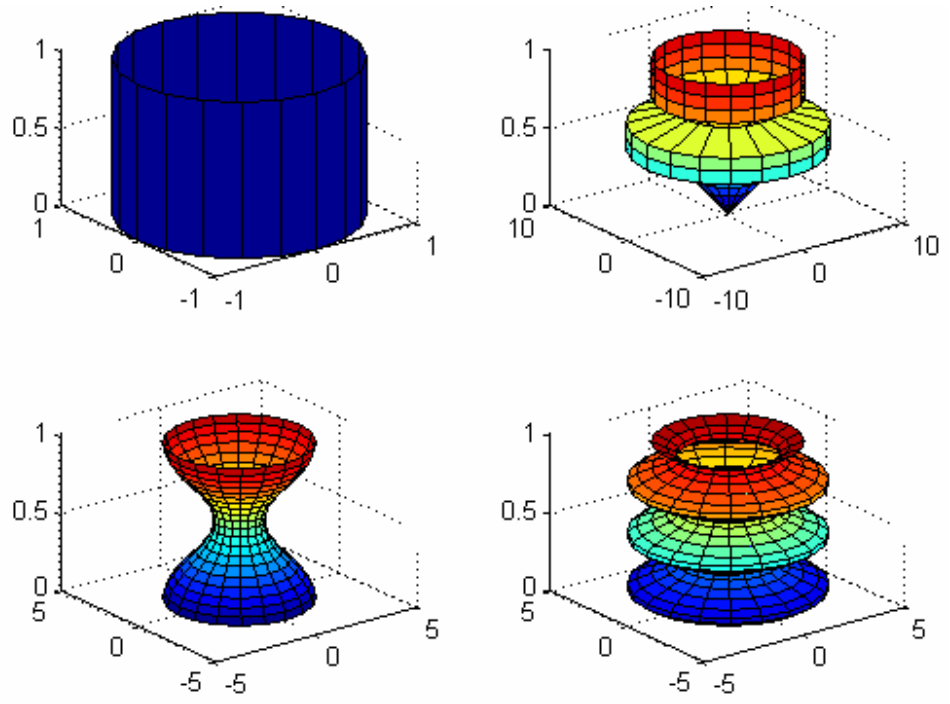
Chú thích:

- ***[X,Y,Z] = cylinder*** : Tạo khối hình trụ bằng hai lệnh ***surf*** và ***mesh***, trả về bộ dữ liệu tạo ra hình trụ bằng hai lệnh ***surf*** và ***mesh***
- ***[X,Y,Z] = cylinder(r):*** Tạo khối trụ với bán kính r
- ***[X,Y,Z] = cylinder(r,n):*** tạo khối trụ với bán kính r và n mặt xung quanh

Ví dụ:

MATLAB code:

```
subplot(2,2,1);
cylinder;
grid on;
subplot(2,2,2)
r=[0 1 2 3 5 8 8 8 4 6 6 6 6];
cylinder(r);
subplot(2,2,3);
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(2+cos(t));
surf(X,Y,Z)
subplot(2,2,4);
t=0:pi/5:6*pi;
p=3+sin(t);
cylinder(p);
```



8. ***stem3***:

Cú pháp:

```
stem3(Z)
stem3(X,Y,Z)
stem3(...,'fill')
stem3(...,LineSpec)
h = stem3(...)
```

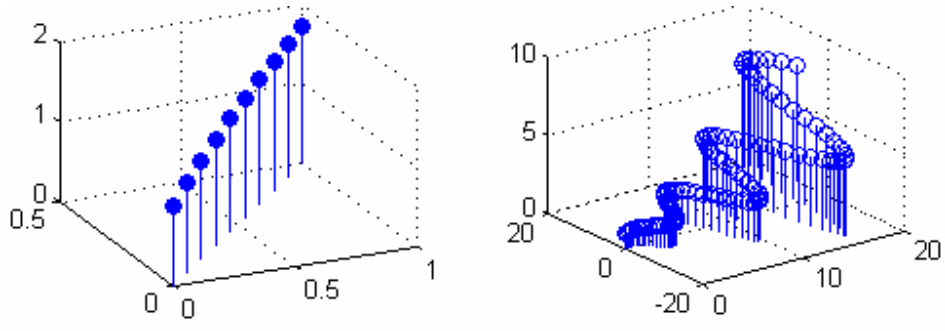
Chú thích:

Giống như lệnh ***stem*** chỉ khác thêm giá trị theo phương z

Ví dụ:

MATLAB code:

```
subplot(2,2,1);
X = linspace(0,1,10);
Y = X./2;
Z = sin(X) + cos(Y);
stem3(X,Y,Z,'fill')
view(-25,30)
subplot(2,2,2);
t=0:.2:20;
x=t; y=t.*cos(t);
z=exp(0.1*t);
stem3(x,y,z);
```



9.

SYMBOLIC MATLAB

Symbolic là một Toolbox(Thư viện) các phép toán kiểu ký tự được đưa vào môi trường tính toán của Matlab. Toolbox này làm phong phú thêm vào tiện ích tính toán và đồ họa của MATLAB. Symbolic Matlab được phát triển dựa trên Symbolic Maple. Đây là phần mềm được viết bởi Đại học Waterloo Canada dựa trên các cơ sở được nghiên cứu của trường Đại học Kỹ thuật Eidgenossiche thành phố Zurich Thụy Sĩ.

1. Khai báo biến trong Symbolic:

Lệnh ***sym*** và ***syms*** cho phép tạo các biến và đối tượng thuộc Symbolic MATLAB

Cú pháp:

S = ***sym***(A)	***syms*** arg1 arg2 ...
x = ***sym***('x')	***syms*** arg1 arg2 ... *real*
x = ***sym***('x','*real*')	***syms*** arg1 arg2 ... *unreal*
x = ***sym***('x','*unreal*')	

Chú thích:

Lệnh ***sym*** khai báo từng biến một, bên trong dấu () là một biến.

Lệnh ***syms*** khai báo đồng thời nhiều biến symbolic

Ví dụ:

syms y z r s t

Lệnh trên tương đương với tập lệnh sau:

```
y= sym( ‘y’)
z= sym( ‘z’)
r= sym( ‘r’)
s= sym( ‘s’)
t= sym( ‘t’)
```

Chú ý:

Một biểu thức symbolic Matlab, tất cả các biến nếu không phải hằng số điều khai báo biến cả trước khi xây dựng biểu thức Matlab.

Ví dụ:

```
>> S=2*x+5*y-3*t
??? Undefined function or variable 'x'.
>> syms x y t
>> S=2*x+5*y-3*t
S =
   2*x+5*y-3*t
```

Để kiểm tra biến của một biểu thức symbolic Matlab ta dùng lệnh ***findsym*** cú pháp lệnh như sau:

r = findsym(S): Trả về lần lượt các biến symbolic trong S, các biến này cách nhau bởi dấu phẩy (,) nếu không có biến nào sẽ trả về một mảng rỗng.

Ví dụ:

```
>> findsym(2*x+5*y-3*t)
ans =
t, x, y
>> findsym(S)
ans =
t, x, y
```

2. Các hàm toán học trong symbolic Matlab:

2.1 Hàm ***diff*** (differentiate): Dùng để tính đạo hàm

Cú pháp:

- ***diff(S)***: đạo hàm biểu thức symbolic với biến được xác định bằng lệnh ***findsym***.
- ***diff(S,'v')***: đạo hàm biểu thức symbolic ứng với biến v
- ***diff(S,n):*** đạo hàm cấp n của biểu thức S
- ***diff(S,'v',n)***: đạo hàm theo biến v đến cấp n

Ví dụ

```
>> syms x t
>> f=5*x^2+2*t;
>> g=3*cos(x^2);
>> diff(f)
ans =
 10*x
>> diff(f,'t')
ans =
 2
>> diff(g,2)
ans =
-12*cos(x^2)*x^2-6*sin(x^2)
```

2.2 Hàm ***int*** (integrate): Dùng để tính tích phân

Cú pháp:

- ***R = int(S)***: lấy tích phân biểu thức S ứng với biến symbolic
- ***R = int(S,v)***: lấy tích phân biểu thức S ứng với biến v
- ***R = int(S,a,b)***: lấy tích phân biểu thức S cận từ a đến b
- ***R = int(S,v,a,b)*** : lấy tích phân biểu thức S cận từ a đến b ứng với biến S

Ví dụ:

```
>> syms x
>> a=0; b= pi;
>> int('sin(x)^2',a,b)

ans =

1/2*pi
```

2.3 Hàm ***jacobian***: Xây dựng ma trận Jacobian

Cú pháp:

R = jacobian(w,v)

Chú thích:

jacobian(w,v) tính Jacobian của hàm w ứng với v. Trong đó w là biểu thức symbolic còn v là vectơ hàng chứa các biến

Ví dụ:

```
>> syms x y z
>> w=5*x^3+6*y+2*z;
>> v=[x y z];
>> jacobian(w,v)
ans =
[ 15*x^2,       6,       2]
```

2.4 Hàm ***limit***: Tính giới hạn của biểu thức

Cú pháp:

- ***limit(F,x,a)***: tìm giới hạn của hàm F khi x → a
- ***limit(F,a)***: tìm giới hạn của hàm F khi biến của hàm F được tìm bởi hàm findsym dần về a
- ***limit(F)***: tìm giới hạn của hàm F khi x → 0
- ***limit(F,x,a,'right')***: tìm giới hạn bên phải của hàm F khi x → a
- ***limit(F,x,a,'left')***: tìm giới hạn bên trái của hàm F khi x → a

Ví dụ:

```
>> syms x y
>> f=(5*x^3+6*x)/(2*x^2+5);
>> limit(f,x,2)
ans =
 4
```

2.5 Hàm ***symsum***: Tổng của chuỗi

Cú pháp:

- r = symsum(s): tổng của biểu thức symbolic s theo biến symbolic k được xác định bằng lệnh ***findsum*** từ 0 → k-1
- r = symsum(s,v): tổng của biểu thức symbolic s theo biến symbolic v được xác định bằng lệnh ***findsum*** từ 0 → v-1
- r = symsum(s,a,b), r = symsum(s,v,a,b): tổng của biểu thức symbolic s theo biến symbolic v được xác định bằng lệnh ***findsum*** từ v=a → v=b

Ví dụ:

```
>> syms k n
symsum(k,1,n)
ans =
1/2*(n+1)^2-1/2*n-1/2
>> pretty(ans)
                    2
      1/2 (n + 1)   - 1/2 n - 1/2
```

2.6 Hàm ***taylor*** : Khai triển chuỗi Taylor

Cú pháp:

- ***r = taylor(f,n,v):*** Cho đa thức xấp xỉ theo Maclaurin bậc n-1 của biểu thức f với v là biến khai triển.
- ***r = taylor(f,n,v,a)***: khai triển Taylor của biểu thức f quanh điểm a. Nếu không cho giá trị thì giá trị mặc nhiên của n là 6. Công thức khai triển Taylor như sau:

$$f(x) = \sum_{n=0}^{\infty} (x-a)^n \cdot \frac{f^{(n)}(a)}{n!}$$

Ví dụ: Khai triển Taylor của hàm f thực hiện trong toán học

$$\sum_{n=0}^{5} x^n \cdot \frac{f^{(n)}(0)}{n!}$$

Matlab:

>> syms x

>>taylor(f)

Ta có kết quả như sau:

```
ans =

6/5*x+13/25*x^3-26/125*x^5
```

3. Các hàm đại số tuyến tính:

Một số hàm ***det, diag, eig, inv, tril, triu,rank*** đã đề cập trong phần ma trận ở chương mở đầu tuy nhiên các lệnh này chỉ dừng lại ở việc các phần tử trong ma trận là số, trong phần Symbolic này các phần tử trong ma trận là các biến. Ví dụ sau sẽ minh họa rõ hơn.

```
>> syms x y z s t
>> A = [x+y,x+z;1+s,1-t]

A =

[ x+y, x+z]
[ 1+s, 1-t]
```

```
>> diag(A)
ans =
[ x+y]
[ 1-t]
```

```
>> triu(A)
ans =
[ x+y, x+z]
[   0, 1-t]
```

```
>> inv(A)
ans =
[ (-1+t)/(x*t-y+y*t+x*s+z+z*s),   (x+z)/(x*t-y+y*t+x*s+z+z*s)]
[  (1+s)/(x*t-y+y*t+x*s+z+z*s), -(x+y)/(x*t-y+y*t+x*s+z+z*s)]
```

Hàm ***expm(A)***: Tạo hàm mũ của ma trận symbolic A

Hàm ***poly(A)***: Biến đổi dạng ma trận thành biểu thức

4. Các hàm đơn giản hóa

4.1 Hàm ***collect***: gôm các số hạng và biến

Cú pháp:

- ***R = collect(S)***: thu gọn các hệ số của biểu thức S với biến mặc định là x
- ***R = collect(S,v)***: thu gọn các hệ số của biểu thức S với biến là v thay vì biến mặc định là x.

Ví dụ:

```
>> syms x y t
>> f=3*x+5*x^2+6*(x-1)*(x+2);
>> collect(f)
ans =
11*x^2+9*x-12
>> g=6*y+y^3+y*(y-1)*(y+1)+7*x+2*x^2;
>> collect(g,y)
ans =
2*y^3+5*y+7*x+2*x^2
```

4.2 Hàm ***expand*** : khai triển biểu thức symbolic

Cú pháp:

expand(S)

Ví dụ:

```
>> syms x y z
>> h=(2*x-1)*(x+2)+ 6*(y-1)+z*(z+1)
h =
(2*x-1)*(x+2)+6*y-6+z*(z+1)
>> expand(h)
ans =
2*x^2+3*x-8+6*y+z^2+z
```

4.3 Hàm ***factor***: phân tích biểu thức thành thừa số

Cú pháp:

factor(X)

Ví dụ:

```
>> syms x y
>> f=(x^3-y^3);
>> factor(f)

ans =

(x-y)*(x^2+x*y+y^2)
```

4.4 Hàm ***numden***: lấy tử số và mẩu số

Cú pháp:

[N, D]=numden(S)

Với N: biểu thức tử số của đa thức S

D: biểu thức mẩu số của đa thức S

Ví dụ:

```
>> syms x y t
>> g=(x*(y-1)*(t+1))/((x-1)*(y+1)*t);
>> [N, D]=numden(g)
N =
x*(y-1)*(t+1)
D =
(x-1)*(y+1)*t
```

4.5 Hàm ***simple*** và ***simlify***: tìm dạng tối giản của đa thức

Cú pháp:

Ví dụ:

```
>> syms x y t
>> g=cos(t)^2+sin(t)^2;
>> simplify(g)
ans =

1
>> f=(x-y)*(x^2+x*y+y^2);
>> simple(f);
ans =

x^3-y^3
```

4.6 Hàm ***subs***: thay thế biến simbolic thành biến khác hoặc thành giá trị

Cú pháp:

R = subs(S,old,new)

Ví dụ:

```
>> syms x y t
>> S=cos(3*x)+2*sin(y);
>> S_new_1=subs(S,x,t)

S_new_1 =

cos(3*t)+2*sin(y)

>> S_new=subs(S,x,pi/4)

S_new =

-1/2*2^(1/2)+2*sin(y)
```

5. Các hàm tìm nghiệm phương trình

5.1 Hàm ***solve*** dùng giải phương trình và hệ phương trình.

Cú pháp:

g = solve(eq)

g = solve(eq,var)

g = solve(eq1,eq2,...,eqn)

g = solve(eq1,eq2,...,eqn,var1,var2,...,varn)

Chú thích:

Ví dụ:

5.2 Hàm dsolve dùng để giải phương trình và hệ phương trình vi phân thường

6. Các hàm đặc biệt

6.1 Hàm tính tích phân cosine ***cosint(X)***

Tính giá trị của tích phân cosine ứng với mỗi phần tử của X. Tích phân của hàm cosine này được định nghĩa bởi hàm sau:

$$Ci(x) = \gamma + \ln(x) + \int_0^x \frac{\cos t - 1}{t} dt$$

Ví dụ:

```
>> cosint(pi/2)

ans =

    0.4720
```

6.2 Hàm tính tích phân sine ***sinint(X)***

Tính giá trị của tích phân sine ứng với mỗi phần tử của X. Tích phân của hàm sine này được định nghĩa bởi hàm sau:

$$Si(x) = \int_0^x \frac{\sin t}{t} dt$$

Ví dụ:

```
>> sinint(pi/2)

ans =

    1.3708
```

7. Các hàm biến đổi Laplace và Fourier

7.1 Hàm biến đổi Fourier (***fourier***)

Cú pháp :

F = fourier(f)

F = fourier(f,v)

F = fourier(f,u,v)

Chú thích:

- F = fourier(f): biến đổi fourier của hàm vô hướng symbolic với biến mặc định x và biến mặc định trả về là w,nghĩa là

$$f = f(x) \Rightarrow F = F(w)$$

Và được tính bằng:

$$F(w) = \int_{-\infty}^{\infty} f(x)e^{-iwx}dx$$

- F = fourier(f,v): giống như fourier(f) chỉ khác là biến trả về là biến v thay vì biến w.
- F = fourier(f,u,v): biến đổi fourier của hàm f có biến là u thành hàm F có biến là v.

Ví dụ:

7.2 Hàm biến đổi Fourier ngược (***ifourier***)

Cú pháp:

f = ifourier(F)

f = ifourier(F,u)

f = ifourier(F,v,u)

Chú thích:

Biến đổi Fourier hàm	Matlab command

$f(x) = e^{-x^2}$	`>> syms x`
	`>> f = exp(-x^2)`
	`f =`
	`exp(-x^2)`

$F[f](w) = \int_{-\infty}^{\infty} f(x)e^{-ixw}dx$
$= \sqrt{\pi}\, e^{-w^2/4}$
	`>> fourier(f)`
	`ans =`
	`pi^(1/2)*exp(-1/4*w^2)`

$f(x) = xe^{-\lvert x\rvert}$	`>> syms x u` `>> f = x*exp(-abs(x));`

$F[f](u) = \int_{-\infty}^{\infty} f(x)e^{-ixu}dx$
$= -\frac{4i}{(1+u^2)^2 u}$
	`>> fourier(f,u)`
	`ans =`
	`-4*i/(1+u^2)^2*u`

Tương tự như hàm fourier(f) chỉ khác là hàm ifourier(F) là phép biến đổi fourier ngược nghĩa là:

$$F = F(w) \Rightarrow f = f(x)$$

Và hàm fourier ngược được định nghĩa như sau:

$$f(x) = 1/(2\pi)\int_{-\infty}^{\infty} F(w)e^{iwx}dw$$

7.3 Hàm biến đổi Laplace (***laplace***)

Cú pháp:

laplace(F)
laplace(F,t)
laplace (F,w,z)

Chú thích :

- ***L=laplace(F)*** : thực hiện phép biến đổi laplace với biến mặc định của hàm F là t và trả về biến mặc định là s nghĩa là :

$$F = F(t) \Rightarrow L = L(s)$$

Phép biến đổi này được định nghĩa như sau :

$$L(s) = \int_0^{\infty} F(t)e^{-st}dt$$

- ***L=laplace(F,t)*** : tương tư như cú pháp trên chỉ khác là biến mặc định của hàm L là t thay vì s.
- L=laplace (F,w,z): giống như trên chỉ khác là w là biến của hàm F còn z là biến của hàm Laplace L nó thay thế các biến mặc nhiên s và t

Ví dụ :

Biến đổi Laplace	Matlab command
$f(t) = t^4$ $L[f] = \int_0^{\infty} f(t)e^{-ts}dt$ $= \frac{24}{s^5}$	>> syms t >> f= t^4; >> L=laplace(f) L = 24/s^5

Biến đổi Fourier ngược	Matlab command
$f(w) = e^{w^2/(4a^2)}$ $F^{-1}[f](x) = \int_{-\infty}^{\infty} f(w)e^{ixw}dw$ $= \frac{a}{\sqrt{\pi}}e^{-(ax)^2}$	>> syms a real >> syms w u >> f = exp(-w^2/(4*a^2)); >> F = simple(ifourier(f)) F = a*exp(-x^2*a^2)/pi^(1/2)

7.4 Hàm biến đổi Laplace ngược(***ilaplace***)

Cú pháp :

F = ilaplace(L)

F = ilaplace(L,y)

F = ilaplace(L,y,x)

Chú thích : Phép biến đổi ngược laplace

$$L = L(s) \Rightarrow F = F(t)$$

Và được định nghĩa như sau :

$$F(t) = \int_{c-i\infty}^{c+i\infty} L(s)e^{st}ds$$

8. Các hàm đồ họa symbolic

8.1 Hàm vẽ chu tuyến (***ezcontour*** và ***ezcontourf***)

8.2 Hàm vẽ mặt 3D có lưới ***ezmesh, ezsurf***

8.3 Hàm vẽ mặt 3D có thêm các đường (***contour ezmeshc, ezsurfc***)

8.4 Hàm vẽ đường trong 2D ***ezplot***

8.5 Hàm vẽ đường trong 3D ***ezplot3***