

## Chương 1 CƠ SỞ MATLAB

### 1.1 Tổng quan về Matlab

#### 1.1.1 Khái niệm về Matlab

**Matlab** là một ngôn ngữ lập trình thực hành bậc cao được sử dụng để giải các bài toán về kỹ thuật. Matlab tích hợp được việc tính toán, thể hiện kết quả, cho phép lập trình, giao diện làm việc rất dễ dàng cho người sử dụng. Dữ liệu cùng với thư viện được lập trình sẵn cho phép người sử dụng có thể có được những ứng dụng sau đây.

- Sử dụng các hàm có sẵn trong thư viện, các phép tính toán học thông thường
- Cho phép lập trình tạo ra những ứng dụng mới.
- Cho phép mô phỏng các mô hình thực tế.
- Phân tích, khảo sát và hiển thị dữ liệu.
- Với phần mềm đồ họa cực mạnh
- Cho phép phát triển, giao tiếp với một số phần mềm khác như C++, Fortran.

#### 1.1.2 Tổng quan về cấu trúc dữ liệu của MATLAB, các ứng dụng

Matlab là một hệ thống tương giao, các phần tử dữ liệu là một mảng ( mảng này không đòi hỏi về kích thước ). Chúng cho phép giải quyết các vấn đề liên quan đến lập trình bằng máy tính, đặc biệt sử dụng các phép tính về ma trận hay véc tor và có thể sử dụng ngôn ngữ C học Fortran lập trình rồi thực hiện ứng dụng lập trình đó bằng các câu lệnh gọi từ MATLAB. MATLAB được viết tắt từ chữ MATRIX LABORATORY tức là thư viện về ma trận, từ đó phần mềm MATLAB được viết nhằm cung cấp cho việc truy cập vào phần mềm ma trận một cách dễ dàng, phần mềm ma trận này được phát triển bởi các công trình Linpack và Eispack. Ngày nay MATLAB được phát triển bởi Lapack và Artpack tạo nên một nghệ thuật phần mềm cho ma trận.

##### a. Dữ liệu

Dữ liệu của MATLAB thể hiện dưới dạng ma trận ( hoặc mảng – tổng quát ), và có các kiểu dữ liệu được liệt kê sau đây

- Kiểu đơn **single**, kiểu này có lợi về bộ nhớ dữ liệu vì nó đòi hỏi ít byte nhớ hơn, kiểu dữ liệu này không được sử dụng trong các phép tính toán học, độ chính xác kém hơn
- Kiểu **double** kiểu này là kiểu thông dụng nhất của các biến trong MATLAB
- Kiểu Sparse.
- Kiểu int8, uint8, int16 . . .

- Kiểu char ví dụ 'Hello'
- Kiểu cell.
- Kiểu Structure.

Trong MATLAB kiểu dữ liệu double là kiểu mặc định sử dụng trong các phép tính số học. Các bạn có thể tham khảo các kiểu dữ liệu khác trong đĩa CD Help MATLAB 6.0

### **b. Ứng dụng**

MATLAB tạo điều kiện thuận lợi cho:

- Các khoá học về toán học
- Các kỹ sư, các nhà nghiên cứu khoa học
- Dùng MATLAB để tính toán ,nghiên cứu tạo ra các sản phẩm tốt nhất trong sản xuất.

### **c.Toolbox là một công cụ quan trọng trong Matlab**

Công cụ này được MATLAB cung cấp cho phép bạn ứng dụng các kỹ thuật để phân tích, thiết kế , mô phỏng các mô hình .

Ta có thể tìm thấy toolbox ở trong môi trường làm việc của .

- Mạng nơron
- Logic mờ
- Simulink

### **1.1.3 Hệ thống MATLAB**

Hệ thống giao diện của MATLAB được chia thành 5 phần

- Môi trường phát triển.

Đây là nơi đặt các thanh công cụ, các phương tiện giúp chúng ta sử dụng các lệnh và các file, ta có thể liệt kê một số như sau.

- + Desktop
- + Command Window
- + Command History
- + Browsers for viewinghelp
- Thư viện, các hàm toán học
  - Bao gồm các cấu trúc như tính tổng, sin cosin atan, atan2 etc..., các phép tính đơn giản đến các phép tính phức tạp như tính ma trận nghịch đảo, trị riêng, chuyển đổi furier ,laplace , symbolic library
- Ngôn ngữ MATLAB
  - Đó là các ngôn ngữ cao về ma trận và mảng, với các dòng lệnh, các hàm, cấu trúc dữ liệu vào , có thể lập trình hướng đối tượng.
- Đồ hoạ trong MATLAB

Bao gồm các câu lệnh thể hiện đồ họa trong môi trường 2D và 3D, tạo các hình ảnh chuyển động, cung cấp các giao diện tương tác giữa người sử dụng và máy tính .

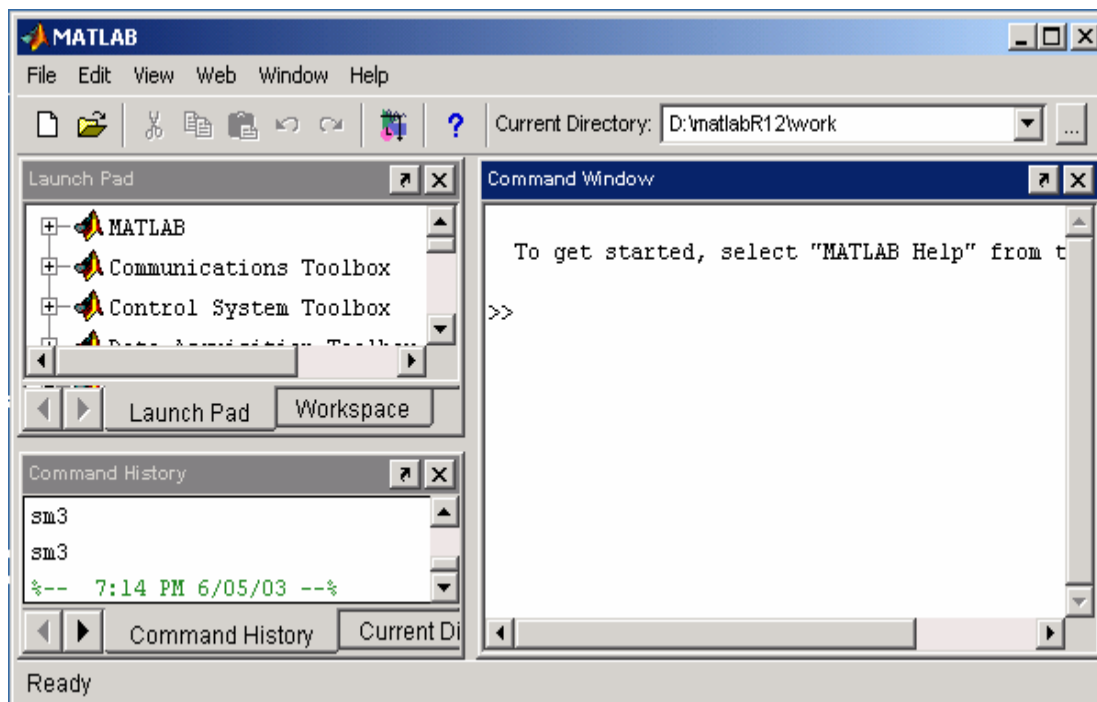
- Giao tiếp với các ngôn ngữ khác.

MATLAB cho phép tương tác với các ngôn ngữ khác như C , Fortran

#### 1.1.4 LÀM QUEN VỚI MATLAB

Trước tiên để khởi động MATLAB bạn kích đúp (hoặc đơn) và biểu tượng file MATLAB.exe ,trên màn hình xuất hiện cửa sổ sau.( Xem hình vẽ 1.1 )

Cửa sổ đó chứa các thanh công cụ( giao diện người và máy) cần thiết cho việc quản lý các files, các biến ,cửa sổ lệnh, có thể coi desktop là các panel gồm các ô, vùng, quản lý và tác dụng của từng cửa sổ nhỏ được quản lý bởi desktop



Hình vẽ 1.1

Trên hình vẽ ta thấy cửa sổ desktop(cửa sổ lớn nhất), và các cửa sổ phụ của nó

#### 1.1.5 LÀM VIỆC VỚI CÁC CỬA SỔ CỦA MATLAB ĐƯỢC QUẢN LÝ BỞI DESKTOP

##### a. Cửa sổ Command window :

Là cửa sổ giao tiếp chính của Matlab bởi đây là nơi nhập giá trị các biến, hiển thị giá trị,tính toán giá trị của biểu thức, thực thi các hàm có sẵn trong thư viện (dạng lệnh), hoặc các hàm(dạng *function*) do người dùng lập trình ra trong M-files.

Các lệnh được nhập sau dấu nhắc ' >> ', và nếu có sai sót trong quá trình gõ(nhập) lệnh thì hãy nhấn phím Enter cho đến khi nhận được dấu nhắc >>. Thực thi lệnh bằng nhấn phím Enter.

Gõ các lệnh sau:

```
>> A= pi/2 ; ↵
```

```
>> B= sin(A) ↵
```

```
B=
```

```
1
```

Hoặc chương trình soạn thảo trong M-file dưới đây:

```
% Chương trình trong M-file
x= 0:pi/6:2*pi;
y=sin(x);
plot(x,y);
% chương trình được lưu với tên file là
ve_sin.m
```

Thực thi chương trình trên trong cửa sổ Command window bằng dòng lệnh sau

```
>> ve_sin ↵
```

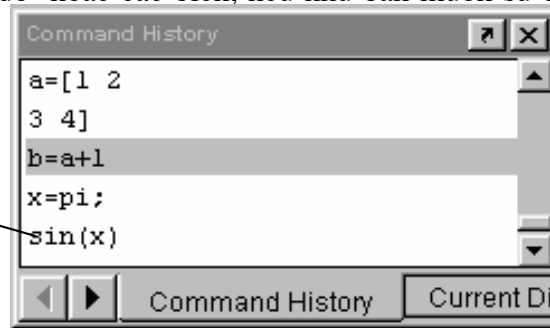
Chúng ta thấy rõ hơn trong mục “ Sử dụng lệnh trực tiếp “ ở phần sau.

##### b. Cửa sổ command History

Các dòng mà bạn nhập vào trong cửa sổ Command window ( các dòng này có thể là dòng nhập biến ,hoặc có thể là dòng lệnh thực hiện hàm nào đó ) được giữ lại trong cửa sổ Command History ,và cửa sổ này cho phép ta sử dụng lại những lệnh đó bằng cách kích

đôi chuột lên các lệnh đó hoặc các biến, nếu như bạn muốn sử dụng lại biến đó. **Xem hình 1.2**

Kích đôi chuột lên lệnh hoặc biến để sử dụng lại

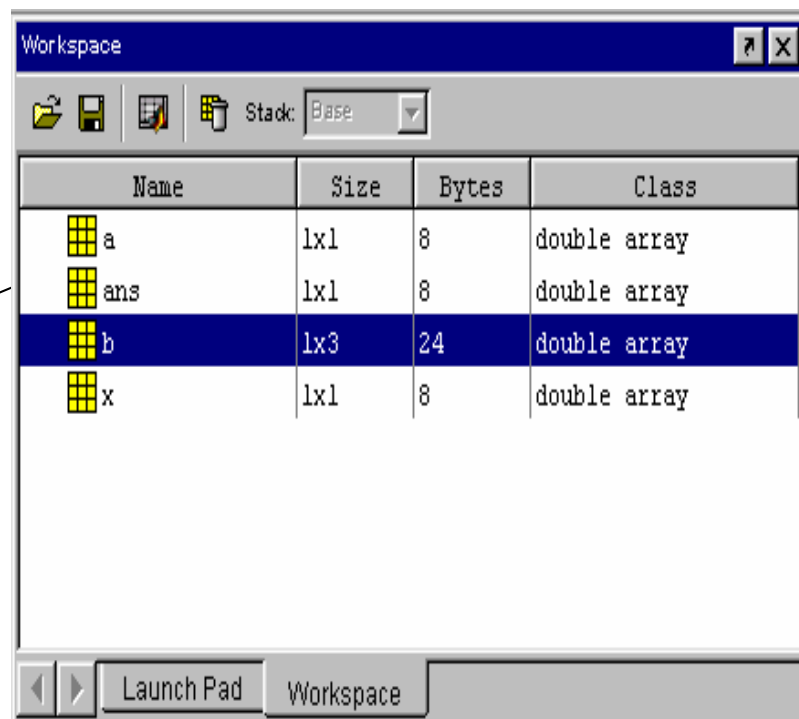


**Hình 1.2**

c. Cửa sổ Workspace:

Là cửa sổ thể hiện tên các biến bạn sử dụng cùng với kích thước vùng nhớ(số bytes), kiểu dữ liệu(lớp) ,các biến được giải phóng sau mỗi lần tắt chương trình.(**xem hình 1.3**)

Kích đôi chuột lên biến để xem dữ liệu(hoặc thay đổi giá trị)



**Hình 1.3**

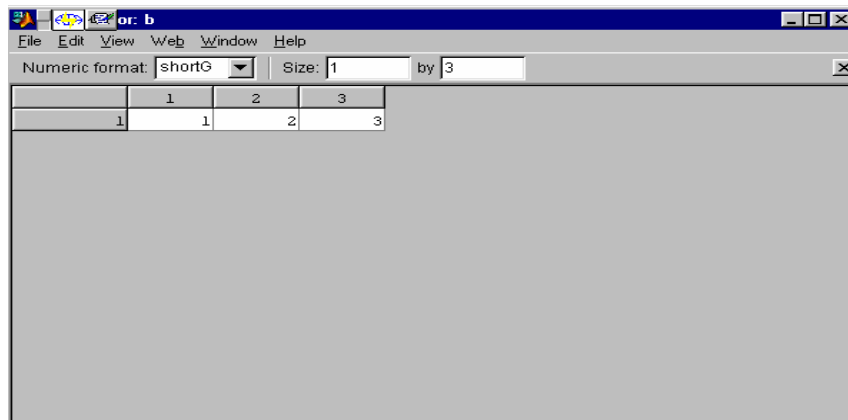
Ngoài ra nó cho phép thay đổi giá trị , cũng như kích thước của biến bằng cách kích đôi chuột lên các biến. Hoặc kích vào nút bên trái ngay cạnh nút save

Ví dụ khi chọn biến(giả thử là biến b) rồi kích đúp(hoặc kích chuột vào nút cạnh nút save) ta được cửa sổ sau gọi là Array Editor: **xem hình 1.4**

Tiêu đề là tên biến  $b$ , *định dạng dữ liệu ở ô có tên là: Numeric format*, mặc định là dạng short, *Kích thước size là 1 by 3 (tức là một hàng và 3 cột)* ta có thể thay đổi kích thước này bằng cách thay đổi số có trong ô kích thước size.

+ Dùng cửa sổ này để lưu các biến

*Ở dưới là dữ liệu của biến  $b$* , ta có thể thay đổi chúng bằng cách thay đổi giá trị trong các ô đó



Hình 1.4

### Ví dụ

Nhập biến `>>b=[1 2 3];`  
`>>x=pi;`

Tất cả các biến đều được lưu trong Workspace trong đó thể hiện cả kích thước (Size), số Bytes và kiểu dữ liệu(class) (8 bytes cho mỗi phần tử dữ liệu kiểu double cụ thể là 24 bytes dành cho  $b$  và 8 bytes dành cho  $a$ )

### d. Cửa sổ M-file

Là một cửa sổ dùng để soạn thảo chương trình ứng dụng, để thực thi chương trình viết trong M-file bằng cách gõ tên của file chứa chương trình đó trong cửa sổ Commandwindow.

Khi một chương trình viết trong M-file, thì tùy theo ứng dụng cụ thể, tùy theo người lập trình mà chương trình có thể viết dưới dạng sau

+**Dạng Script file**: Tức là chương trình gồm tập hợp các câu lệnh viết dưới dạng liệt kê, không có biến dữ liệu vào và biến lấy giá trị ra

+**Dạng hàm function** có biến dữ liệu vào và biến ra.

**e. Đường dẫn thư mục**: Nơi lưu giữ các file chương trình

## 1.2 Nhập biến, lệnh trực tiếp từ cửa sổ Command Window:

Sau khi xuất hiện dấu nhắc `>>` trong cửa sổ command window điều đó đồng nghĩa cho phép bạn nhập biến hoặc thực hiện các câu lệnh mong muốn.

*Do dữ liệu của MATLAB được thể hiện dưới dạng ma trận cho nên các biến dùng trong MATLAB dữ liệu của nó cũng thể hiện dưới dạng ma trận, việc đặt tên biến không được đặt một cách tùy tiện mà phải đặt theo một quy định*

- Tên ma trận(biến) phải bắt đầu bằng một chữ cái, và có thể chứa đến 19 ký tự là số hoặc chữ.
- Bên phải dấu bằng là các giá trị của ma trận
- Dấu chấm phẩy(; ) là để phân cách các hàng, còn các giá trị trong hàng được phân cách nhau bởi dấu phẩy(,) hoặc dấu cách( phím space).
- Kết thúc nhập ma trận thường có dấu chấm phẩy hoặc không tùy theo bạn muốn thể hiện kết quả của nó hay không.

#### a. Nhập các biến, ma trận, các lệnh liệt kê trực tiếp

Thông thường Matlab sử dụng 4 vị trí sau dấu phẩy cho các số thập phân có dấu phẩy chấm động, và sử dụng biến " ans " cho kết quả của phép tính. Ta có thể đăng ký biến thể hiện kết quả này của riêng mình . Xét tập các lệnh sau:

Ví dụ trường hợp không sử dụng biến lưu kết quả, biến ans tự động được gán

```
>> 8+9
```

```
ans =
```

```
17
```

Nhập biến  $r = 8/10$  trong cửa sổ CommandWindow như sau:

```
>> r = 8/10 ←
```

```
r=0.8000
```

Bạn có thể sử dụng các biến này cho các phép tính tiếp theo ví dụ như:

```
>> s=10*r
```

```
s=
```

```
8
```

Ví dụ nhập trực tiếp các số liệu như sau

```
>> a=[1 2;3 4]
```

```
a =
```

```
1 2
```

```
3 4
```

Matlab có hàng trăm hàm được định nghĩa sẵn ví dụ như hàm tính sin ...

```
>> x=pi; %nhập biến x
```

```
>> sin(x) % nhập lệnh sin(x), ấn enter để thực hiện lệnh tính sin(x)
```

```
ans =
```

```
1.2246e-016
```

**+ Các phép tính sử dụng trong Matlab :**

Trong MATLAB cũng sử dụng các phép toán thông thường được liệt kê trong bảng sau

Ký tự	Ý nghĩa	Lệnh Matlab
+	Cộng $a + b$	$a+b$
-	Trừ $a - b$	$a-b$
*	Nhân $ab$	$a*b$
/	Chia phải $a/b = \frac{a}{b}$	$a/b$
\	Chia trái $b \backslash a = \frac{a}{b}$	$b/a$
^	Mũ $a^b$	$a^2$

Thứ tự ưu tiên các phép toán:

Tất cả các biểu thức toán học đều được thực hiện từ trái qua phải, ta có bảng thứ tự ưu tiên như sau:

Thứ tự ưu tiên	Các phép
1	Dấu ngoặc trong biểu thức
2	Toán tử mũ ^, thực thi từ trái qua phải
3	Toán tử nhân, chia có cùng mức ưu tiên, thực hiện từ trái sang phải.
4	Cộng, trừ

Ví dụ 1 :

```
>> a=[1 2;3 4];
>> b=[5 6;7 8];
>> a+b^2
ans =
    68    80
    94   110
```

Ví dụ 2 Giải phương trình bậc hai, các lệnh nhập trong cửa sổ CommandWindow

```
>>a= 1;
>>b=-2;
>>c=1;
>>delta= b^2- 4*a*c;
>>x1=(-b+ sqrt(delta) )/(4*a);
>>x2=(-b- sqrt(delta) )/(4*a);
```



**Chú ý :** + Các lệnh được kết thúc bằng dấu chấm phẩy, Matlab sẽ không thể hiện kết quả trên màn hình, ngược lại không có dấu chấm phẩy Matlab sẽ thể hiện kết quả.

+ Trong quá trình nhập ma trận nếu các phần tử trên một hàng dài quá ta có thể xuống dòng bằng toán tử ba chấm(...)

Ví dụ

```
>>Number_apples=10;Number_Oranges=25,Number_bananas=34;  
>>Fruit_Purchased= Number_apples+ Number_Oranges+ ...  
Number_bananas
```

### 1.3 Sử dụng các lệnh gián tiếp từ các file dữ liệu

Như đã trình bày trong phần cửa sổ M-file, tập hợp các lệnh của MATLAB được soạn thảo trong cửa sổ M-file dưới dạng Script file hoặc dạng hàm function(có biến đầu vào và ra), và được ghi (lưu)vào file dữ liệu có phần mở rộng là .m (Thông thường các chương trình soạn thảo trong M-file thường được lưu theo đường dẫn C:\matlab\work\Tên\_file ), muốn thực thi chương trình soạn thảo đó ta gọi lệnh trong cửa sổ Commandwindow, tùy theo chương trình viết dạng Script file hay function mà trong cửa sổ ta có 2 cách gọi như sau:

- Đối với chương trình viết dạng Script file

```
>> tên_file ;
```

```
a=1;  
b=-2;  
c=1;  
delta=b^2-4*a*c;  
x1=(-b+sqrt(delta))/(2*a)  
x2=(-b-sqrt(delta))/(2*a)  
% lưu vào file GPTB2.m
```

Ví dụ giải phương trình bậc hai tìm nghiệm x1 và x2 viết trong M-file dạng Scriptfile: Thực thi chương trình trên trong cửa sổ CommandWindow bằng lệnh

```
>>GPTB2 ←
```

- Đối với chương trình viết dạng function ,có tham số đầu vào và ra,ta phải truyền đủ các tham số cần thiết.

**Ví dụ :** Giải phương trình bậc hai với ba tham số đầu vào là các hệ số a , b, c và hai biến đầu ra là nghiệm của phương trình x<sub>1</sub> và x<sub>2</sub>

(Xem cách viết hàm **function** ở mục sau)

```
function [x1, x2]  
=GPTB2(a,b,c)  
    x1=(-b+sqrt(delta))/(2*a);  
%Tinh nghiệm x1  
    x2=(-b-  
sqrt(delta))/(2*a);  
%Tinh nghiệm x2
```

Thực hiện bài toán trên trong Command window như sau:

```
>>a= 1;
>>b=-2;
>>c=1;
>>[x1,x2]=GPTB2 (a,b,c) % cấu trúc chung là [x1,x2]=Tên_file (a,b,c)
( hoặc [x1,x2]=GPTB2(1,-2,1) ↵ )
```

Lưu ý rằng khi viết chương trình trong M-file, bạn muốn ghi chú thích ta dùng ký tự % đặt trước dòng chú thích như sau  
%-----dòng chú thích -----

#### Ví dụ 2

```
%Viết trong M-file(dạng Script file)
x=0:0.1:10 ; %Tạo vector x
y=cos(x);
plot(x,y); % Vẽ đồ thị hàm cosin
%lưu vào file có tên là dai1.m
```

Thực thi hàm trên của sổ commandwindow bằng lệnh  
>> dai1 ↵

Viết chương trình trong M-file được dùng là chủ yếu ,đặc biệt đối với những chương trình dài , phức tạp thì bạn nên viết trong M-file.

#### 1.4 Dòng nhắc gán giá trị biến

Đối với bạn đã học lập trình Pascal, bạn muốn nhập giá trị khi thực thi chương trình bạn dùng cặp lệnh:

```
writeln( 'Nhập giá trị của a=');
readln(a);
```

Nhưng đối với MATLAB thì bạn sẽ thấy rất đơn giản chỉ dùng một lệnh duy nhất đó là :  
a=input('Nhập gia tri cua a=');

**Ví dụ:** Trong cửa sổ Commandwindow ta gõ lệnh  
>> a =input('nhap a=');

Nhấn Enter cho kết quả dưới dạng

nhập a= 3; đồng nghĩa với việc gán a=3.

*Sử dụng dòng nhắc gán giá trị biến trong trường hợp ta muốn thay đổi giá trị các biến lúc chạy chương trình*

```
% Chương trình viết trong M-file, bạn có thể viết trong
CommandWindow
a=input('nhập he so a=' );
b=input('nhập he so b=' );
c=input('nhập he so c-=' );
Delta=b^2-4*a*c;
x1=(-b+ sqrt(Delta))/(2*a)
x2=(-b- sqrt(Delta))/(2*a)
```

### 1.5 Cách tạo một hàm function

Trước hết ta thống nhất rằng, để tạo một hàm function ta phải soạn thảo nó trong M-file. Cấu trúc hàm như sau:

```
%Khai báo hàm có từ khoá function
function[danh sách tên kết quả]= Tên_hàm(danh sách các biến đầu vào)
% Thân chương trình
câu lệnh 1;
câu lệnh 2;
câu lệnh 3;
.....
câu lệnh n;
%kết thúc chương trình khi kết thúc câu lệnh
```

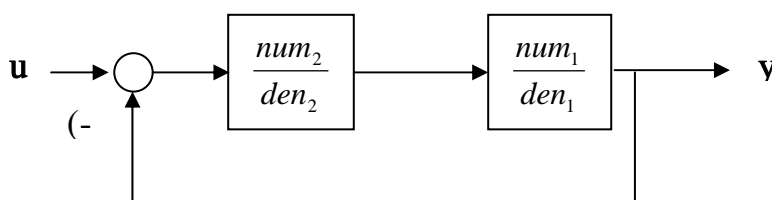
**Chú ý:**

- Danh sách tên kết quả, và tham số đầu vào được cách nhau bằng dấu phẩy.

Ví dụ : `function[x1,x2,x3]=dai2(a,b,c,d)`

- Thân chương trình **không** bắt đầu bằng từ khoá **Begin** và **không** kết thúc bằng từ khoá **End** như Ngôn ngữ lập trình Pascal.
- *Ta nên lưu vào file có tên trùng với tên hàm*

Ví dụ: Cho sơ đồ khối của hệ thống điều khiển tự động như hình dưới đây



Nhiệm vụ: Tính hàm truyền kín của hệ thống

Chương trình có thể được viết như sau:

```
function[numk, denk]=ham_truyen(num1, den1, num2, den2)
numh=conv(num1, num2);% conv là hàm nhân, hàm này được định nghĩa sẵn
denh=conv(den1,den2);
numk=numh;
m=length(denh)- length(numh);
numh1=[zeros(:,m), numh];
denk= numh1+denh;
%kết thúc chương trình tại đây bạn nên lưu vào file có tên là ham_truyen.
```

Thực thi hàm:

```
>> num1=[1 1];
>>den1=[1 2 1];
>>num2=[1 2];
>>den2=[1 2 1 4];
>>[numk,denk]=ham_truyen(num1,den1,num2,den2);
```

### 1.6 Sử dụng hàm có sẵn

Có rất nhiều hàm có sẵn, đó là các hàm đã được lập trình sẵn, và được đưa vào thư viện, để xem một hàm cũng như cấu trúc, cách sử dụng ta dùng lệnh >>help tên\_hàm

Ví dụ Ta muốn xem cấu trúc hàm ode23

```
>>help ode23
```

### 1.7 Vẽ các hàm

Dùng lệnh fplot để vẽ các hàm, hàm này có thể có sẵn(ví dụ như sin, cos . . .), hoặc các hàm tạo bởi người dùng viết trong M-file dạng function

Cấu trúc:

```
fplot('Tên_hàm',[Xmin,Xmax]
,tol,N,'LineStyle');hoặc

fplot(@Tên_hàm,[Xmin,Xmax]
,tol,N,'LineStyle');
```

Lưu ý:Đối với các hàm toán học có sẵn(không phải định nghĩa) ví dụ như sin, cos ,... thì có thể thực hiện như sau:

```
+ fplot('sin(x)',2*pi*[-1 1] ) %vẽ y=sin(x) với x=[-2*pi 2*pi];
+ fplot([sin(x),tan(x),cos(x)]', 2*pi*[-1 1] );
```

%vẽ ba đồ thị trên cùng một cửa sổ với  $x=[-2*\pi \ 2*\pi]$  ;

- Dùng hàm inline ví dụ : `f=inline('x+2');` `fplot(f,[0 2])` ;
- Đối với các hàm trong M-file có thể sử dụng các cách sau

Ví dụ: Tính  $f_1, f_2, f_3$

```
function [f1,f2,f3]= FUNC(x)
    f1= x+3;
    f2=x;
    f3=x.^2;
    %lưu vào file FUNC.m
```

Hàm FUNC sẽ trả về một vector hàng ứng với mỗi giá trị của  $x$ , ví dụ  $x=[x_1;x_2]$  thì hàm FUNC sẽ trả về ma trận sau đây.

$f_1(x_1), f_2(x_1), f_3(x_1)$

$f_1(x_2), f_2(x_2), f_3(x_2)$

***Lợi dụng đặc điểm này ta có thể vẽ nhiều đồ thị trên cùng một cửa sổ thông qua ví dụ sau:***

```
%Tạo hàm Y
function Y=myfun(x)
    Y(:,1)=200*sin(x(:))./(x(:));
    Y(:,2)=x(:).^2;
    %lưu vào file có tên là
myfun.m
```

Thực thi chương trình trên trong Commandwindow

`>>fplot('myfun',[-20 20])`;

(hoặc dùng `>>fplot(@myfun,[-20 20])` )

***Các thông số*** `tol`, `N`, `LineStyle` ***lần lượt là sai số liên quan(tương đối), số điểm ít nhất, biểu diễn thuộc tính của đường.***

***Chú ý:Khi bạn muốn hạn chế khoảng biểu diễn cả trục x và y thì dùng*** `[Xmin Xmax Ymin Ymax]` .

### 1.8 Lưu và lấy dữ liệu

Với Matlab khi thoát khỏi chương trình(tắt),các biến dữ liệu(trongWorkspace) sẽ bị mất,do vậy khi thực hiện lại chương trình bạn phải khai báo lại các biến cần thiết trên,

điều này gây mất thời gian, và biện pháp tốt là bạn lưu tất cả các biến cần thiết cho chương trình của bạn vào file riêng, và khi cần chúng ta gọi chúng ra bằng một lệnh

***Lưu dữ liệu có thể là :***

- Lưu tất cả các biến trong vùng làm việc( Workspace) hoặc
- Một số biến nhất định tùy theo nhu cầu .

Sau đây là các cách lưu các biến dữ liệu:

### 1.8.1 Lưu và lấy dữ liệu dưới file nhị phân(binary)

***Lưu dữ liệu:***

```
>>save('C:\matlabR12\work\ten_file') %lưu toàn bộ biến trong Workspace
```

```
>>save('C:\matlabR12\work\ten_file', 'x','y')% chỉ lưu biến x và y
```

Chú ý: C:\matlabR12\work\ten\_file là đường dẫn tới file, thông thường khi cài đặt chương trình thì mặc định là cài vào ổ C (nếu bạn cài vào ổ D, khi sử dụng lệnh save, bạn chỉ cần thay đổi thành :D:\matlabR12\work\ten\_file)

Ví dụ:

```
%Viết trong Command Window  
>>a=1;  
>>b=1;  
>>c=-2;  
>>save('C:\matlabR12\work\Bien', 'x','y')
```

***Khôi phục lại dữ liệu dùng lệnh sau:***

```
load ('C:\matlabR12\work\ten_file') % lấy dữ liệu
```

Ví dụ: Bây giờ ta xoá hai biến a và b ra khỏi chương trình và thực hiện lệnh load để lấy lại dữ liệu:

```
>>clear a ; %xoá biến a  
>>clear b ; %xoá biến b  
>> load ('C:\matlabR12\work\ten_file')  
>>a %kiểm tra xem a đã khôi phục lại  
chưa  
a=1  
>>b%kiểm tra xem b đã khôi phục lại  
chưa  
b=1
```

### 1.8.2 Lưu và lấy dữ liệu dưới file ASCII

```
>>save('C:\matlabR12\work\ten_file','-ASCII').
```

Lưu toàn bộ biến trong workspace vào file

```
>>save('C:\matlabR12\work\ten_file','x','y','-ASCII').
```

Lưu hai biến x và y vào file

```
>>load('C:\matlabR12\work\ten_file','-ASCII').
```

khi thực hiện lệnh này thì trong Workspace sẽ xuất hiện biến có tên là tên của file , kích đúp chuột lên biến này sẽ xuất hiện dữ liệu của toàn bộ biến được lưu giữ, việc truy nhập đến biến lưu giữ thông qua việc truy nhập kiểu Matrận

Ví dụ Command window

```
>>a=2;
```

```
>>b=3;
```

```
>>c=4;
```

```
>>save('C:\matlabR12\work\ save ')%lưu 3 biến trong file tên save
```

```
>> load('C:\matlabR12\work\ save ')%khôi phục dữ liệu
```

hoặc

```
>> save('C:\matlabR12\work\ save', 'a','b')%lưu hai biến a và b  
trong file %tên save
```

Tương tự:

```
>>a=3;
```

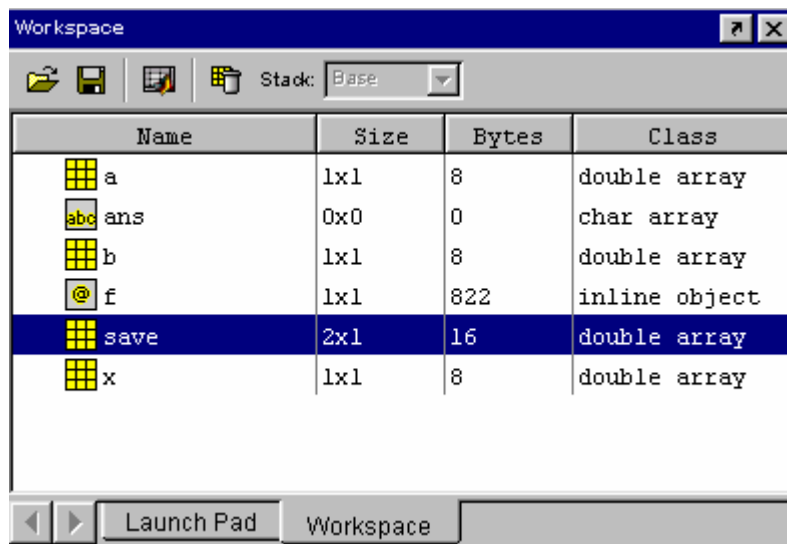
```
>>b=4;
```

```
>>save('C:\matlabR12\work\save','a','b','-ASCII')
```

```
>>load('C:\matlabR12\work\save','-ASCII') %khôi phục dữ liệu
```

Trong workspace sẽ có biến save như sau:

Kích đúp vào save sẽ xuất hiện dữ liệu của hai biến a và b



*Hoặc đơn giản để lưu biến bạn có thể chọn biến rồi kích vào nút save trong cửa sổ Workspace*

## 1.9 Các toán tử logic và các lệnh điều kiện

### 1.9.1 Các toán tử quan hệ

Một biểu thức logic trong MATLAB có được từ sự so sánh các đại lượng khác nhau (ví dụ hai đại lượng A và B). Những ký hiệu thể hiện sự so sánh được gọi là các toán tử quan hệ, sau đây là liệt kê các toán tử

#### Bảng liệt kê các toán tử quan hệ

Toán tử quan hệ	Ý nghĩa
<	Nhỏ hơn vd $A < B$
>	Lớn hơn vd $A > B$
<=	Nhỏ hơn hoặc bằng $A <= B$
>=	Lớn hơn hoặc bằng $A >= B$
==	Bằng vd $A == B$
~=	Không bằng vd $A ~= B$

Các toán tử quan hệ thực hiện việc so sánh từng phần tử của mảng, chúng trả lại một mảng có cùng kích thước với hai mảng trên (hai mảng ban đầu phải có cùng kích thước nếu không sẽ gây ra lỗi), với các phần tử trong mảng là 0 hoặc 1 tương ứng với các quan hệ so sánh là sai hay đúng

#### Trường hợp đặc biệt so sánh hai số phức:

+ Khi dùng các toán tử quan hệ là < hoặc > thì chỉ so sánh phần thực của nó mà thôi.

+ Khi dùng các toán tử quan hệ <= hoặc >= thì so sánh cả phần thực lẫn phần ảo

#### Khi so sánh hai chuỗi.

Dùng toán tử strcmp

#### Cấu trúc:

strcmp( chuỗi1, chuỗi2)

Ví dụ :

```
>> Chuoi1 = 'Pham Duc Dai';
>> Chuoi2 = 'Vu van van';
>> ss = strcmp(Chuoi1,
Chuoi2);
ss = 0
```

**Chú ý :** Khi so sánh một số vô hướng với một ma trận thì số đó được nhân với một ma trận ones(size(ma trận so sánh)) sao cho nó có kích thước giống với ma trận cần so sánh rồi mới so sánh.

Ví dụ:

$X=5$ ;  $X>=[1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9] \Leftrightarrow X=5*\text{ones}(3,3)$ ;  $X>[1\ 2\ 3; 4\ 5\ 6; 7\ 8\ 9]$

Kết quả trả về :

ans =



```

1 1 1
1 1 0
0 0 0
>>X=5;
>>X >=[1 2 3 ; 4 5 6; 7 8
9]
ans
1 1 1
1 1 0
0 0 0

```

**1.9.2 Các toán tử logic**

(Logical Operator & | ~)

Cấu trúc:

Toán tử logic	Ý nghĩa
&	Và vd A&B
	Hoặc vd A B
~	Đảo vd ~A

Các ký hiệu & , | ,~ là các toán tử logic và hoặc đảo. Chúng thực hiện trên từng phần tử của của các mảng so sánh( toán tử logic cho phép thực hiện trên nhiều mảng với yêu cầu là các mảng phải có cùng kích thước), kết quả trả về là một ma trận có cùng kích thước với các ma trận so sánh trên. Các toán tử logic thường dùng để liên kết các biểu thức quan hệ.

***Bảng chân lý:***

Đầu vào		And	Or	Xor	Not
A	B	A&B	A B	xor(A,B)	~A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Mức ưu tiên cao nhất đối với toán tử logic đảo( not ,~) , hai toán tử and và | có cùng mức ưu tiên , trong một biểu thức toán học thì chúng được thực hiện theo thứ tự từ trái sang phải.

Ta có thể sử dụng các toán tử 'and' , 'or' , 'not' ⇔ & , | , ~ như bảng sau:

A&B	and(A,B)
A B	or(A,B)
~A	not(A)

Chú ý trong các biểu thức sử dụng các toán tử logic thì ta nên dùng dấu ngoặc để xác định rõ ràng, và đảm bảo tính tương thích trong các phiên bản mới của Matlab

#### Tổng kết:

- Các phép tính số học sẽ được thực hiện trước khi thực hiện các biểu thức logic.
- Khi tính toán ta nên thêm dấu ngoặc đơn để làm biểu thức trở nên sáng sủa hơn.
- Gặp những biểu thức phức tạp sẽ sử lý các tính toán số học trước, sau đó các toán tử logic được xem xét từ trái qua phải .

### 1.10 Các câu lệnh điều kiện, rẽ nhánh

#### 1.10.1 Câu lệnh điều kiện *if*.

Cấu trúc

% Đây là cấu trúc đơn giản nhất.

```
if expression
    Statements;
end;
```

% Cấu trúc sử dụng lệnh elseif ,else và if được viết liền

```
if expression1
    Statements;
elseif expression2
    Statement;
else
    Statements;
end
```

Biểu thức expression bao gồm các toán tử quan hệ ví dụ như (count<limit) hoặc (height-offset)>0), Ngoài ra nó còn kết hợp với các toán tử logic để liên kết các biểu thức quan hệ.

Ví dụ 1: if (count<limit)&((height-offset)>0)

Ví dụ 2: Cho khoảng [a b], viết chương trình chia khoảng này thành n khoảng bằng nhau

```
function v= lnearspace(a,b,n)
if n<2
error('Ban nhap sai, it nhat n phai lon hon 1');
end;
h=(b-a)/(n-1);
v=a:h:b;
```

-----  
Thực thi chương trình trên trong Command window như sau:

```
>> v=Soan1(5,1,5)
```

```
v =
```

```
5 4 3 2 1
```

với n  
cho  
trước.

Ví dụ 3: Chương trình xác định dấu của số nhập vào :

```
function s= sign(x)
if x>0
s=1;      % so duong
elseif x<0
s=-1;    % so am
else
s=0;     % so =0
end;
```

Nguyên tắc làm việc của lệnh if như sau :Khi biểu thức **expression** đúng thì câu lệnh ngay sau lệnh if được thực hiện nếu sai câu lệnh đó sẽ được bỏ qua đến lệnh elseif

**Một biểu thức số sánh là đúng nếu tất cả các phần tử so sánh của hai mảng (hoặc hai ma trận) trả về giá trị 1**

Ví dụ:

A=[ 1 0; 2 3]; B=[1 1; 3 4];

Thì : + A<B là sai vì A(1,1) không nhỏ hơn B(1,1);

+ A<(B+1) là đúng bởi vì không phần tử nào trong A lớn hơn phần tử trong B tương ứng

% Cấu trúc khi sử dụng lệnh else if

```
if expression1
statements1;
else
if expression2
statements2;
end
end
```

**Ta thấy trong cấu trúc trên thì có hai lệnh if riêng biệt cho nên phải có hai từ kết thúc là end.**

Bài tập ví dụ: Nhập vào bàn phím điểm của một học sinh rồi in ra đánh giá :

```
Diem= input('nhap diem vao=');
if ( Diem< 5)
fprintf('Hoc luc yeu');
elseif( Diem>=5)&(Diem<7)
fprintf('Hoc luc trung binh');
```

### 1.10.2 Vòng lặp for

Cấu trúc:

```
for i= imin : $\Delta$ i: imax
    statements;
end
```

$\Delta$ i : Là bước nhảy của vòng lặp for, giá trị mặc định là =1;

Ví dụ: Tính tổng  $s = 1 + 2^p + 3^p + \dots + n^p$ ; ( p là số mũ )

```
function s= Sump(n , p)
s=0;
for i=1:n ; s=s+i^p ;end;
```

Hai chương trình sau đây là giống nhau

```
for i=1:100;  
y(i)=sin(i);  
end;
```

⇔

```
x=1:100;  
y=sin(x);
```

### 1.10.3 Vòng lặp while

Cấu trúc:

```
while( bieu_thuc_logic)
    statements;
end;
```

Trước hết vòng lặp kiểm tra xem nếu biểu thức logic đúng thì thực hiện các câu lệnh statements.

Ví dụ:

```
n=input( 'Nhap n=' );
s=0; i=0;
while( i<n)
s=s+i;
end;
```

### 1.10.4 Lệnh ngắt break , error, return

- **Lệnh break** :Tác dụng điều khiển chương trình nhảy ra khỏi vòng lặp for hay while gần nó nhất.

Ví dụ:Nhập một số dương nếu âm thì nhập lại

```
while 1
n= input('nhap n=');
while(n<0)      %Vong lap while2
break;         %Thoat khoi vong lap while2
end
if n>0
break;        %Thoat khoi vongwhile chinh
end
end
```

#### 1.10.5 Lệnh error và lệnh return

- **Lệnh error**: Dùng để thông báo lỗi , hiển thị cho người lập trình biết đó là lỗi gì ?

Ví dụ: error('error message'); hiển thị thông điệp lỗi khi thực hiện câu lệnh này.

```
a=input('Nhap a=');
b=input('Nhap b=');
% Thuc hien a: b
if b==0
error('divide by zeros');
end;
```

Khi thực thi chương trình trên ( nhập b=0) thì xuất hiện dòng chữ đỏ

Như sau:

??? Error using ==> soan1

divide by zeros

Chú ý rằng soan1 là tên file lưu chương trình trên

- **Lệnh return**: Thường được sử dụng trong các hàm của MATLAB. Lệnh return sẽ cho phép quay trở về thực thi những lệnh nằm trong tác dụng của lệnh return.

#### 1.10.6 Biến toàn cục

Biến toàn cục được dùng trong phạm vi toàn bộ các chương trình, nếu các chương trình đó khai báo biến toàn cục đó.

**Cấu trúc:**

```
global x y z % khai báo ba biến toàn cục x y z
```

Ví dụ đơn giản sau:

```
function[ u,v]=Main(x,y)
    global a b;
    Tinhham(x,y,z);
    u=a;v=b;
```

%Chương trình con tính hàm

```
function Tinhham(x,y,z)
    global a b;
    a=x^2 +y^2+ z^2;
    b=x^3 +y^3+
z^3;
```

Thực thi chương trình:

```
>> x=2,y=3,z=4;
>> [u,v]=Main(x,y,z);
```

### 1.10.6 Định dạng dữ liệu ra

Các phép tính trong MATLAB được thực hiện với độ chính xác cao, ta có thể định dạng cho các số xuất ra màn hình tùy từng yêu cầu cụ thể:

Ví dụ số  $a = 4/3$  với

- format short (đây là chế độ mặc định gồm 4 số sau dấu phẩy)  
a=1.3333
- format short e  
a= 1.3333e+00
- format long  
a=1.33333333333333
- format lang e  
a=1.33333333333333e+000

Ngoài cách này ra ta định dạng dữ liệu bằng thanh tool công cụ trên màn hình

Ví dụ: Gõ các lệnh sau trong cửa sổ CommandWindow

```
>>format long
>>a=4/3
a=
1.33333333333333
```

### 1.10.7 Một số hàm toán học thông thường hay sử dụng

Tên hàm	Ý nghĩa
Sin	Hàm sin
Cos	Hàm cos
Tan	Hàm tan
Asin	Hàm acsin
Acos	Hàm accos
Atan	Hàm tính arctg
Angle	Lấy góc pha
Fix	Làm tròn hướng 0
Floor	Làm tròn hướng $-\infty$
Exp	Hàm e mũ
Ceil	Làm tròn hướng $-\infty$
Log	Logarit cơ số e
log10	Logarit cơ số 10
sqrt(x)	Căn bậc hai

### 1.11 Các hằng số được sử dụng trong Matlab

- **Ký tự** `inf` thay thế cho  $\infty$  trong toán học

Inf : là số vô cùng lớn mà Matlab không thể hiện được

ví dụ:

```
>> 5/0
```

```
ans =
```

```
Inf
```

- **Ký tự** `NaN` thay thế cho một số không xác định

ví dụ:

```
>> 0/0
```

```
ans=
```

```
NaN
```

- **Ký tự** `pi` thể hiện là số  $\pi = 3.14159..$
- **Ký tự** `eps`

### 1.12 Số phức trong Matlab

Sử dụng `i` và `j` để thể hiện phần ảo với `i = j = sqrt(-1)`

Ví dụ:

```
>> 5+6i
```

```
ans =
```

```

5.0000 + 6.0000i
>> 5+6j
ans =
5.0000 + 6.0000i

```

**Chú ý** khi làm với số phức cần phân biệt :  $y=7/2*i$  và  $x=7/2i$  cho hai kết quả khác nhau

```

>> y= 7/2*i
y =
3.5*i
>> x=7/2i
x=
-3.5i

```

Ví dụ các phép tính về số phức:

```

>> s=3+7i
>>w=5-9i
>>w+s
>>w-s
>>w*s
>>w/s

```

### **1.13 Các lệnh thoát khỏi chương trình, liệt kê các biến, xoá biến .**

- Lệnh `exit` : Tác dụng thoát khỏi chương trình .
- Lệnh `clc` (clear command) xoá tất cả các lệnh trong cửa sổ CommandWindow
- Lệnh `clear` xoá toàn bộ các biến trong bộ nhớ hiện thời .
- Lệnh `clear <danhsáchbiến>` Xoá danh sách biến được liệt kê ra.
- Lệnh `whos` cho biết tất cả các biến hiện thời, kích thước ô nhớ biến đó.
- Lệnh `quit` cũng giống như lệnh `exit`
- Các phím mũi tên lên xuống (trên bàn phím) được dùng để gọi lại các lệnh đã thực hiện trước đó.

### **Câu hỏi& Bài tập cuối chương**

Nêu đặc điểm của các cửa sổ ( CommandWindow, history .)

1. Sử dụng Lệnh trực tiếp từ cửa sổ command window và gián tiếp từ các file
2. Các câu lệnh điều kiện , vòng lặp
3. Lưu và lấy dữ liệu cho một hoặc nhiều biến.

**Bài tập1** : Giải hệ phương trình bậc nhất dùng dòng nhắc gán giá trị biên nhập các hệ số a,b,c,d,m,n

hệ cho như sau:  $ax + by = m$



$$cx+dy=n$$

**Bài tập 2: Sinh viên vẽ các hàm sau**

a. Vẽ đặc tính diode với quan hệ dòng điện và điện áp trên diode như sau

$$i=I_0*(\exp(40*v)-1)(A); I_0=1.E-6; \text{ vector } v=[-10:0.005:0.8] .$$

Sinh viên thực hiện theo hai cách viết trực tiếp trong CommandWindow và viết hàm trong M-file.

b. Vẽ các hàm cơ bản  $\sin(x)$  ,  $\cos(x)$  ,  $\tan(x)$ ... dùng lệnh `fplot` .

c. Vẽ hàm  $y= \sin(x)/x$  ,  $y=x$  ,  $y=\sin(x)$  trên cùng một đồ thị .

**Bài tập 3** : Lưu tất cả các biến bạn đã dùng trong quá trình thực hành vào file riêng của mình, để lần sau lấy ra dùng lại.

Bài 4:Lập chương trình dạng hàm ***function*** để giải phương trình bậc hai

Bài 5:Lập hàm tính hàm truyền kín các sơ đồ hệ thống điều khiển trong sách lý thuyết điều khiển tự động

**Bài 5:Lập hàm *function*  $[Q, R]=\text{divide}(a,b)$  tìm thương và số chia hai số a và b**

## CHƯƠNG 2

# THƯ VIỆN TOÁN HỌC KIỂU KÝ TỰ (SYMBOLIC MATLAB)

### 2.1 Giới thiệu về thư viện toán học kiểu ký tự

Symbolic matlab là thư viện các phép toán kiểu ký tự được đưa vào môi trường tính số học của matlab , thư viện này làm phong phú và tiện ích thêm với nhiều kiểu tính toán về toán học khác cho phần tính số học và đồ hoạ đã có trước đây trong thư viện Matlab.

### 2.2 Các lệnh cơ bản khai báo biến symbolic

#### 2.2.1 Lệnh syms và lệnh sym

+ Nhiệm vụ tạo đối tượng (bao gồm cả biến) symbolic

Cấu trúc:

```
syms arg1 arg2 ...  
syms arg1 arg2 ... real  
syms arg1 arg2 ... unreal
```

Mô tả

Khai báo các biến  $arg1$  ,  $arg2$  là các biến symbolic có hai cách khai báo dùng lệnh `syms` hoặc lệnh `sym` như sau:

`syms arg1 arg2 ...`

Khai báo các thông số  $arg1$ ,  $arg2$  là các biến symbolic , ta có thể khai báo như sau

```
arg1 = sym('arg1');  
arg2 = sym('arg2'); ...
```

Tương tự :

`syms arg1 arg2 ... real` là ký hiệu ngắn gọn cho

```
arg1 = sym('arg1','real');  
arg2 = sym('arg2','real'); ...
```

Các biến khai báo như trên là các biến thực kiểu symbolic .Vậy thì các biến này khác gì các biến khai báo không có đặc tính real?

Ta phân biệt như sau: Đối với một biến thực symbolic thì nó có các tính chất của số thực ví dụ như  $(arg)^2 > 0$  (khi khai báo là `syms arg real`) còn khi bạn khai báo là `syms arg` thì các biến này chỉ đơn thuần là biến symbolic không có các tính chất của số thực tức là  $(arg)^2$  sẽ không có dấu ,mà chỉ coi là các ký tự symbolic mà thôi

Tiếp tục

`syms arg1 arg2 ... unreal` là ký hiệu ngắn gọn cho

```
arg1 = sym('arg1','unreal');  
arg2 = sym('arg2','unreal'); ...
```

Ví dụ:

`syms x beta real` giống như việc khai báo

```
x = sym('x','real');  
beta = sym('beta','real');
```

Để xoá đối tượng symbolic `x` và `beta` khỏi (trạng thái) `'real'` ta làm như sau

```
syms x beta unreal
```

**Chú ý :** `clear x` sẽ không xoá đối tượng symbolic `x` khỏi trạng thái `'real'`. Bạn có thể thực hiện được điều trên (tức là xoá `x` khỏi trạng thái số thực) bằng cách sử dụng các lệnh `syms x unreal or clear mex or clear all`.

### 2.2.2.Lệnh `sym`

Tạo một số, một biến và một đối tượng symbolic

Cấu trúc như sau

```
S = sym(A)  
x = sym('x')  
x = sym('x','real')  
x = sym('x','unreal')  
S = sym(A,flag) where flag is one of 'r', 'd', 'e', or 'f'.
```

Mô tả:

`S = sym(A)` Tạo một đối tượng `S` của lớp `'sym'` từ `A`. Nếu thông số đầu vào là một chuỗi, kết quả là một số, một biến symbolic. Nếu thông số đầu vào là một số vô hướng hay một ma trận, kết quả là một thể hiện của các số đã cho dưới dạng symbolic

`x = sym('x')` Tạo biến symbolic với tên là `x` chứa kết quả trong `x`

`x = sym('x','real')` cho rằng `x` là thực cho nên `conj(x)` bằng với `x` (có thể coi đây là phương pháp kiểm tra số thực)

Ví dụ: `x = sym('x','unreal')` làm cho biến `x` (trong sạch) và không có đặc tính nào thêm (đảm bảo rằng `x` không phải là biến thực)

*Ví dụ*

+ `pi = sym('pi')` kết quả cho lại giá trị số `pi` (đầu vào là một chuỗi)

+ Lệnh `pi = sym('pi')` và `delta = sym('1/10')`

Kết quả `delta = 1/10` ;

Cấu trúc sau cho phép chuyển đổi số symbolic sang các dạng số thực và các dạng số khác tùy thuộc vào flag là `'r'`, `'d'`, `'e'` hoặc `'f'`

$S = \text{sym}(A, \text{flag})$  ở đó flag là một trong 'r', 'd', 'e', or 'f'.

**Ví dụ : Tạo ma trận symbolic A**

```
A=[ 1 2 3 ; 4 5 6];  
>>A=[ 1 2 3; 4 5 6];  
>>A=sym(A)  
Kết quả trả về ma trận A= [ 1 2 3]  
[ 4 4 6]
```

**Ví dụ: Tạo biến symbolic x ,y, z**

```
>> syms x y z ;% hoặc sym('x' ) hoặc sym('y') . . .  
>> f= x^2 + y^2 +z^2;
```

**Ví dụ Tạo số symbolic a= 5**

```
>> a= sym('5')  
a =  
5
```

Thông thường hiệu quả của việc sử dụng lệnh sym là để chuyển đổi một ma trận từ số sang dạng phom symbolic .Lệnh

```
A = hilb(3)
```

**Tạo ma trận Hilbert**

```
A =  
1.0000 0.5000 0.3333  
0.5000 0.3333 0.2500  
0.3333 0.2500 0.2000
```

**áp dụng sym cho A**

```
A = sym(A)
```

**Bạn có thể đạt được ma trận symbolic Hilbert có kích thước 3-by-3**

```
A =  
[ 1, 1/2, 1/3]  
[ 1/2, 1/3, 1/4]  
[ 1/3, 1/4, 1/5]
```

Ta thấy rằng khi áp dụng lệnh symbolic cho số hoặc ma trận thì kết quả thu lại sẽ chính xác hơn

2.2.3 Phép Tính với các biến và số symbolic

Các phép tính cơ bản về ma trận đều được dùng đối với các số và biến symbolic.

**Ví dụ về phép cộng hai Ma trận symbolic(hoặc với một ma trận không phải là symbolic)**

```
>> syms a b c;  
>> a=[a b c ; b c a];  
>> d=[1 2 3 ;4 5 6];
```

```

>> a+d
ans =
    [ a+1, b+2, c+3]
    [ b+4, c+5, a+6]
>> A=sym([1 2 3 ; 4 5 6]);
>> B=sym([2 3 4 ;5 6 7]);
>> A+B
ans =
    [ 3, 5, 7]
    [ 9, 11, 13]
Tương tự cho phép nhân và phép chia ( * / \ ./ .\ )

```

### 2.3 Tạo hàm symbolic

Thông thường có hai cách tạo hàm Symbolic

- Tạo hàm bằng biểu thức symbolic  $f = f(x,y,z,...)$  trong đó  $x, y, z$  được khai báo là các biến symbolic
- Tạo trong M-file
- Tạo trực tiếp các hàm .

#### 2.3.1 Tạo hàm từ các biểu thức symbolic

Hàm tạo ra chứa các biến phải là biến symbolic

Ví dụ tạo hàm  $f = 3*x^2 + 2*x + 1$  ta làm như sau

```

>> syms x % khai báo x là biến symbolic
>> f= 3* x^2 + 2*x +1 % f là hàm symbolic

```

ví dụ:

```

syms x y z
r = sqrt(x^2 + y^2 + z^2)
t = atan(y/x)
f = sin(x*y)/(x*y)

```

**Tạo biểu thức symbolic r và t và f .**

Chú ý Chỉ khi tạo một hàm symbolic thì Bạn mới được phép sử dụng lệnh limit ,diff, int, subs, và các hàm toán học symbolic khác

#### 2.3.2 Tạo Hàm Symbolic từ M-file

Tạo một hàm bằng cấu trúc function , trong đó đầu vào là các biến cần để thiết lập hàm , đầu ra là biến chứa hàm như vậy cách tạo hàm giống với tạo hàm thông thường , Vì thế để

Matlab hiểu rằng đây là hàm symbolic thì ta phải lưu vào file có đường dẫn như sau  
C:\matlabR12\toolbox\symbolic\@sym\ten\_ham

Ví dụ tạo hàm symbolic  $z = \sin(x)/x$

```
function z = sinc(x)
%SINC The symbolic sinc function
% sin(x)/x. This function
% accepts a sym as the input argument.
if isequal(x,sym(0))
    z = 1;
else
    z = sin(x)/x;
end
```

Ví dụ : Muốn tạo hàm symbolic  $f = 3x^2 + 2x + 1$

```
function f= tao_ham( x)
f= 3*x^2 + 2*x + 1
```

%Lưu vào đường dẫn C:\matlabR12\toolbox\symbolic\@sym\tao\_ham

%-----gọi hàm trong command window-----

```
>> syms x
>>f= tao_ham(x)
f= 3*x^2 + 2*x + 1
```

### 2.3.3 Tạo hàm trực tiếp

Ta có thể tạo hàm trực tiếp như sau  $f = 3x^2 + 2x + 1$

```
>> f=sym('3*x^2 + 2*x +1')
```

Tuy nhiên tạo hàm như trên thì f là hàm symbolic, nhưng bản thân biến x lại không phải là biến symbolic

Khi khai báo hàm kiểu này ,muốn sử dụng biến x ta thêm hai dấu ' x '

```
>> f= sym('3*x^2+ 2*x +1');
```

```
>> g=subs(f,'x','x+h')
```

```
g =
```

```
3*(x+h)^2+ 2*(x+h) +1
```

```
>> df=(subs(f,'x','x+h')-f)/h'
```

```
df =
```

```
(3*(x+h)^2+2*h-3*x^2)/h
```

```
>> diff(f,'x')    ans = 6*x+2
```

Ví dụ : Tính 6!

Ta tạo hàm tính trực tiếp như sau

```
>> f=sym('x!');
```

```
>> subs(f,'x',6)
```

```
ans =
```

```
720
```

Ví dụ tạo hàm  $1/x!$

```
>> f=1/sym('x!');
```

```
>> subs(f,'x',n)
```

```
>> subs(f,'x','n')
```

```
ans =
```

```
1/(n)!
```

## 2.4 Tạo biến thực và biến phức

Tạo biến phức ví dụ  $z = x + i * y$  thì ta phải khai báo  $x$  và  $y$  là các biến symbolic thực tức là:

```
syms x y real
```

```
z = x + i*y
```

### I. Giải thích

Tạo biến symbolic  $x$  và  $y$ , các biến này có được sự công thêm các tính chất toán học của một biến thực. Cụ thể nó có ý nghĩa rằng biểu thức

```
f = x^2 + y^2
```

$f \geq 0$ . Cho nên,  $z$  là một biến phức

```
conj(x)= x;conj(z)=x-i*y;expand(z*conj(z))=x^2+y^2
```

Để xoá  $x$  khỏi là một biến thực, bạn phải dùng lệnh như sau

```
syms x unreal
```

hoặc

```
x = sym('x','unreal')
```

Lệnh sau

```
clear x
```

không làm cho  $x$  khỏi là một số thực

## 2.5 Lệnh findsym

Tìm các biến trong biểu thức symbolic hoặc ma trận

Syntax

```
r = findsym(S)
```

```
r = findsym(S,n)
```

Mô tả

`findsym(S)` Trả về tất cả các biến symbolic trong  $S$  được cách nhau bởi dấu phẩy (trong in alphabetical order). Nếu  $S$  không chứa bất kỳ một biến nào `findsym` trả về một chuỗi rỗng `findsym(S,n)` trả về n biến alphabetically gần x nhất

Ví dụ

```
syms a x y z t
```

```
findsym(sin(pi*t)) returns pi, t.
```

`findsym(x+i*y-j*z)` returns x, y, z.

`findsym(a+y,1)` returns y.

## 2.6 Tính toán

Công cụ toán dọc symbolic cung cấp các hàm để thực hiện các toán tử cơ bản của phép toán

Đạo hàm , giới hạn , tích phân, tổng và mở rộng chuỗi Taylor.

### 2.5.1 Lệnh symsum

Symbolic summation.

Syntax

`r = symsum(s)`

`r = symsum(s,v)`

`r = symsum(s,a,b)`

`r = symsum(s,v,a,b)`

Mô tả

\*`symsum(s)` là tổng của biểu thức symbolic s theo biến symbolic của nó là k được xác định bởi lệnh `findsym` từ 0 đến k-1

\*`symsum(s,v)` là tổng của biểu thức symbolic theo biến symbolic v được xác định từ 0 đến v-1

\*`symsum(s,a,b)` and `symsum(s,v,a,b)` Định nghĩa tổng của biểu thức symbolic theo biến v từ v=a đến v=b

Ví dụ

Các lệnh sau:

`syms k n x`

`symsum(k^2)`

trả về kết quả

$\frac{1}{3}k^3 - \frac{1}{2}k^2 + \frac{1}{6}k$

`symsum(k)` trả về

$\frac{1}{2}k^2 - \frac{1}{2}k$

`symsum(sin(k*pi)/k,0,n)` trả về

$-\frac{1}{2} \frac{\sin(k(n+1))}{k} + \frac{1}{2} \frac{\sin(k)}{k} / (\cos(k)-1) \cos(k(n+1)) -$

$\frac{1}{2} \frac{\sin(k)}{k} / (\cos(k)-1)$

`symsum(k^2,0,10)` trả về kết quả sau

385

Ví dụ:

`>> syms x k;`

`>> symsum(x^k/sym('k!'), k, 0,inf)%inf la +vo cung`

`ans =`



```

exp(x)
>> symsum(x^k/sym('k!'), k, 0,5)
ans =
1+x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5

```

**Chú ý : Các ví dụ trước sử dụng sym để tạo biểu thức symbolic .k!**

### 2.5.2 Tính đạo hàm

Bây giờ chúng ta tạo các biến và hàm

```

syms a x
f = sin(a*x)

```

sau đó

```
diff(f)
```

Lệnh này sẽ tính đạo hàm của f với biến symbolic của nó (trong trường hợp này là x), như được định nghĩa bởi lệnh findsym

```

ans =
cos(a*x)*a

```

Để tính đạo hàm với biến a ta làm như sau

```
diff(f,a)
```

Nó trả về  $df/da$ .

```

ans =
cos(a*x)*x

```

Để tính đạo hàm bậc hai với biến x và a ta làm như sau

```
diff(f,2)
```

hoặc

```
diff(f,x,2)
```

Trả về

```

ans =
-sin(a*x)*a^2

```

và

```
diff(f,a,2)
```

Nó trả về

```

ans =
-sin(a*x)*x^2

```

Định nghĩa a,b,x,n,t và theta trong Matlab workspace, sử dụng lệnh sym. Bảng sau cho thấy tác dụng của lệnh diff

f	diff(f)
---	---------

$X^n$	$x^{n+1}/x$
$\sin(a*t+b)$	$\cos(a*t+b)*a$
$\exp(i*theta)$	$i*\exp(i*theta)$

Example:

```
syms a x
```

```
A = [cos(a*x),sin(a*x);-sin(a*x),cos(a*x)]
```

**Nó trả lại**

```
A =
```

```
[ cos(a*x), sin(a*x)]
```

```
[ -sin(a*x), cos(a*x)]
```

**Lệnh**

```
diff(A)
```

**Trả về**

```
ans =
```

```
[ -sin(a*x)*a, cos(a*x)*a]
```

```
[ -cos(a*x)*a, -sin(a*x)*a]
```

### 2.5.3 sym2poly

**Biến đổi đa thức symbolic sang vec tơ hệ số đa thức của đó**

*Cấu trúc*

```
c = sym2poly(s)
```

**Mô tả**

sym2poly trả về một vector hàng, vec tơ này chứa hệ số của đa thức symbolic. Các hệ số này được xếp theo thứ tự tương ứng với số mũ của biến độc lập của đa thức

**Ví Dụ**

**Các lệnh sau đây:**

```
syms x u v;
```

```
sym2poly(x^3 - 2*x - 5)
```

**Trả về**

```
1 0 -2 -5
```

**trong khi**

```
sym2poly(u^4 - 3 + 5*u^2) Trả về
```

```
1 0 5 0 -3
```

và sym2poly(sin(pi/6)\*v + exp(1)\*v^2) trả về

2.7183 0.5000 0

#### 2.5.4 Tính giới hạn Limit

Công cụ toán học symbolic cho phép bạn tính giới hạn của hàm theo cách thông thường

.Các lệnh sau

```
syms h n x
```

```
limit( (cos(x+h) - cos(x))/h,h,0 )
```

Trả về kết quả

```
ans =
```

```
-sin(x)
```

và

```
limit( (1 + x/n)^n,n,inf ) % n tiến tới vô cùng
```

Nó trả về kết quả

```
ans =
```

```
exp(x)
```

Thể hiện hai trong tất cả giới hạn quan trọng nhất trong toán học, đạo hàm (trong trường hợp này là  $\cos(x)$ ) và hàm e mũ x

giới hạn tồn tại khi cho biến tiến tới hai phía (đó là, kết quả là giống nhau bất kể tiến bên phải hay bên trái ).Nếu kết quả khác nhau hai phía thì đạo hàm đó không tồn tại

Cho nên đạo hàm sau kết quả là không xác định và Công cụ toán học symbolic trả về giá trị là NaN

Lệnh

```
limit(1/x,x,0)
```

hoặc

```
limit(1/x)
```

returns

```
ans =NaN
```

Lệnh

```
limit(1/x,x,0,'left')
```

Trả về

```
ans =
```

```
-inf
```

Trong khi lệnh.

```
limit(1/x,x,0,'right')
```

Trả về:

```
ans =
```

```
inf
```

Quan sát thấy rằng trường hợp mặc định,  $\text{limit}(f)$  giống với  $\text{limit}(f,x,0)$ .

Lựa chọn cho lệnh limit trong bảng trên, chúng ta giả sử rằng f là một hàm symbolic với đối tượng x

II. 2.5.5 Tính Tích phân

Nếu f là một biểu thức symbolic thì tích phân của hàm f là

$\text{int}(f)$

Tìm một biểu thức symbolic F thoả mãn  $\text{diff}(F)=f$ , thì F là giá trị trả về của  $\text{int}(f)$

Tương tự hàm  $\text{int}(f,v)$

$\text{int}(f,v)$  Sử dụng đối tượng symbolic v như là biến của tích phân,

Ví dụ Tạo các biến symbolic sau

`syms a b theta x y n x1 u`

F	Int(f)
$x^n$	$x^{(n+1)}/(n+1)$
$y^{(-1)}$	$\text{Log}(y)$
$n^x$	$1/\log(n)*n^x$
$\text{Sin}(a*\text{theta}+b)$	$-\cos(a*\text{theta}+b)/a$
$\text{Exp}(-x1^2)$	$1/2*\pi^{(1/2)}*\text{erf}(x1)$
$1/(1+u^2)$	$\text{Atan}(u)$

Bảng thể hiện kết quả tích phân của một số hàm

Định nghĩa tích phân còn được thể hiện như sau

$\text{int}(f,a,b)$

hoặc  $\text{int}(f,v,a,b)$  % Tính tích phân f theo biến v từ a đến b

2.6 Giải phương trình - Hệ phương trình đại số

**Giải phương trình-hệ phương trình dùng lệnh solve**

**Mục đích:** Giải một hoặc nhiều phương trình đại số tuyến tính symbolic

**Cấu trúc**

`g = solve(eq)`

`g = solve(eq, var)`

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

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

### Mô tả

Eq là biểu thức đơn hoặc một phương trình. Đầu vào để giải (tìm nghiệm) có thể là biểu thức hoặc chuỗi symbolic. Nếu eq là một biểu thức symbolic ( $x^2-2*x+1$ ) hoặc một chuỗi, chuỗi này không chứa một phương trình, như ( $x^2-2*x+1$ ), thì `solve(eq)` là giải phương trình  $eq=0$  Với biến mặc định của nó được xác định bởi hàm `findsym.solve(eq,var)` tương đương với việc giải phương trình eq (hoặc  $eq=0$  trong hai trường hợp ở trên) đối với biến var (giải phương trình với biến là var)

Ví dụ : `>> solve(' x^2 + 2*x + 1 ', 'x')` tức là giải phương trình  $x^2+2*x+1=0$  với biến là x

```
>> solve(' y*x^2 + x *y+1 ', 'y')
```

Hệ phương trình. Đầu vào là các biểu thức symbolic hoặc các chuỗi xác định phương trình.

`solve(eq1,eq2,...,eqn)` giải hệ các phương trình tạo bởi `eq1,eq2,...,eqn` trong n biến được xác định bằng cách áp dụng lệnh `findsym` cho toàn hệ (in the n variables determined by applying `findsym` to the system)

Ba loại khác nhau của đầu ra có thể.

+ Đối với một phương trình và một đầu ra, kết quả (sau khi giải ) được trả về với nhiều kết quả cho phương trình tuyến tính (with multiple solutions for a nonlinear equation)

+ Đối với hệ thống phương trình có số đầu ra cân bằng, kết quả được chứa trong alphabetically và được ký hiệu như là đầu ra.(chứa trong alphabetically tức là chứa theo thứ tự chữ cái)

+ Đối với hệ thống phương trình có số đầu ra là đơn, kết quả trả về là một cấu trúc

Ví dụ

`solve('a*x^2 + b*x + c')` trả về

```
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2)),
```

```
1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

`solve('a*x^2 + b*x + c','b')` trả về

```
-(a*x^2+c)/x
```

```
>> n=solve('x + y = 1','x - 11*y = 5')
```

```
n =
```

```
  x: [1x1 sym]
```

```
  y: [1x1 sym]
```

```
>> n.y
```

```
ans =
```

```
-1/3
```

```
>> n.x
```

```
ans =
```

4/3

```
>> [x, y]=solve('x + y = 1','x - 11*y = 5')
```

kết quả:

x= 4/3

y=-1/3

```
>>A = solve('a*u^2 + v^2', 'u - v = 1', 'a^2 - 5*a + 6')
```

Trả về dạng cấu trúc

A =

a: [1x4 sym]

u: [1x4 sym]

v: [1x4 sym]

ở đó

A.a =

[ 2, 2, 3, 3]

A.u =

[ 1/3+1/3\*i\*2^(1/2), 1/3-1/3\*i\*2^(1/2),

1/4+1/4\*i\*3^(1/2), 1/4-1/4\*i\*3^(1/2)]

A.v =

[ -2/3+1/3\*i\*2^(1/2), -2/3-1/3\*i\*2^(1/2),

-3/4+1/4\*i\*3^(1/2), -3/4-1/4\*i\*3^(1/2)]

## 2.7 Biến đổi laplace

### 2.7.1 Biến đổi thuận Laplace

Cấu trúc

laplace(F)

laplace(F,t)

Mô tả

$L = \text{laplace}(F)$  là biến đổi laplace của  $F$  với biến độc lập mặc định là  $t$ . kết quả mặc định trả lại là hàm của  $s$ . Biến đổi laplace được áp dụng cho một hàm của biến  $t$  và trả lại một hàm của biến  $s$

Nếu  $F = F(s)$ , laplace trả lại một hàm của  $t$

#### Bằng cách định nghĩa

$t$  là biến kiểu symbolic trong  $F$  được xác định bởi hàm findsym.

$L = \text{laplace}(F,t)$  tạo ra  $L$ , một hàm của  $t$  thay mặc định là hàm của  $s$ .

$L = \text{laplace}(F,w,z)$  tạo ra  $L$ , một hàm của  $z$  trong đó  $F$ , một hàm của  $w$  thay thế biến mặc định là  $s$  và  $t$  tương ứng

### 2.7.2 Biến đổi ngược laplace

Mục đích: Biến đổi ngược laplace

### Cấu trúc

$F = \text{ilaplace}(L)$

$F = \text{ilaplace}(L,y)$

$F = \text{ilaplace}(L,y,x)$

### Mô tả

$F = \text{ilaplace}(L)$  là phép biến đổi ngược Laplace của đối tượng vô hướng symbolic  $L$  với biến độc lập là  $s$ . trả lại mặc định là một hàm của  $t$ . Biến đổi ngược laplace được áp dụng cho một hàm của  $s$  và trả về một hàm của  $t$ . Nếu  $L = L(t)$ ,  $\text{ilaplace}$  trả về một hàm của  $x$ .

### Bằng cách định nghĩa

ở đó  $c$  là một số thực được chọn cho nên tất cả all singularities of  $L(s)$  are to the left of the line  $s = c$ ,  $i$ .

$F = \text{ilaplace}(L,y)$  tạo ra  $F$  là một hàm của  $y$  thay vì mặc định  $t$ .

$y$  là một đối tượng symbolic vô hướng.

$F = \text{ilaplace}(L,y,x)$   $F$  là một hàm của  $x$  và  $L$  là một hàm of  $y$  thay vì mặc định là  $s$  và  $t$ .

### 2.8 Vấn đề tích phân với hằng số thực

Một trong những tình huống liên quan tới đạo hàm các hàm symbolic là

đấu của các biến (coi là hằng số) khi bạn bình phương biến đó. ở đây ta hiểu rằng khi bạn coi một biến nào đó trong biểu thức là biến (ví dụ biến lấy tích phân) thì các biến còn lại được coi là hằng số và Matlab sẽ không hiểu được là nó dương hay âm (coi chỉ là ký tự).

Ví dụ, biểu thức

$$e^{-(kx)^2}$$

Là dương, đồ thị có hình chuông cong tiến tới 0 khi  $x$  tiến tới  $\pm \infty$  với mọi số thực  $k$ .

Một ví dụ về đường cong được cho thấy dưới đây với

$$k = \frac{1}{\sqrt{2}}$$

được tạo ra, sử dụng những lệnh sau

```
syms x
k = sym(1/sqrt(2));
f = exp(-(k*x)^2);
ezplot(f)
```

The Maple kernel, không coi  $k^2$  hoặc  $x^2$  là các số dương. Maple cho rằng biến symbolic  $x$  và  $k$  là không xác định. Có nghĩa rằng, chúng là biến và không có thêm đặc tính toán học nào.

Thông thường tính tích phân hàm trên ta làm như sau

$$\int_{-\infty}^{\infty} e^{-(kx)^2} dx$$

Trong công cụ toán học symbolic , sử dụng hàm

```
syms x k;
f = exp(-(k*x)^2);
int(f,x,-inf,inf)
```

và kết quả là

*Definite integration: Can't determine if the integral is convergent.*

*Need to know the sign of --> k^2*

*Will now try indefinite integration and then take limits.*

*Warning: Explicit integral could not be found.*

*ans =*

*int(exp(-k^2\*x^2),x= -inf..inf)*

Trong lời cảnh báo trên bạn chú ý thấy dòng lệnh “ Need to know the sign of----> k<sup>2</sup> “ tạm dịch là không hiểu dấu của k<sup>2</sup>. Mà hợp lý toán học là k<sup>2</sup> phải dương do vậy bạn phải khai báo sao cho k<sup>2</sup>>0 bằng cách

---> Tạo biến Real sử dụng lệnh sym

Chú ý rằng Maple không thể định nghĩa dấu của biểu thức k<sup>2</sup>. Bằng cách nào có thể vượt qua trở ngại này? Câu trả lời là tạo biến k biến thực. Sử dụng lệnh sym.

```
syms k real
int(f,x,-inf,inf)
```

trả về

```
ans =
signum(k)/k*pi^(1/2)
```

## 2.9 Vẽ Đồ thị Dùng hàm ezplot cho các biến, số symbolic

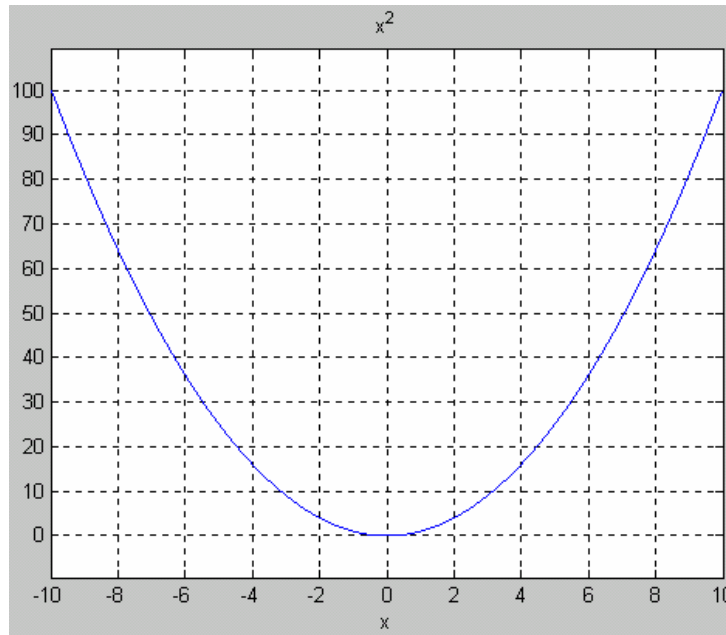
Cờu trúc: ezplot( y ,[ xo xm]): Vẽ y theo biến x thuộc khoảng [ xo xm]

Ví dụ:

```
>> syms x y;
>> y= x.^2;
```



>> ezplot(y,[1 10]), grid on



Các bạn chú ý rằng lệnh ezplot trên dùng để vẽ trong không gian 2D ( không gian 2 chiều ) , còn để vẽ trong không gian 3D không có gì khó khăn ta dùng lệnh ezplot3 ,các bạn tự tham khảo thêm sách .

### Câu hỏi ôn tập

1. Những tiện ích khi sử dụng thư viện toán học symbolic là gì ?.
2. lệnh findsym có tác dụng gì ?.
3. Thứ tự ưu tiên các biến khi sử dụng biến mặc định ? .
4. Có mấy cách tạo hàm symbolic? Em hãy so sánh các cách .
5. Dấu của các biến symbolic như thế nào ?
6. Vẽ đồ thị hàm symbolic, bằng hàm vẽ thông thường plot có được không ?

### Bài tập

1. Tạo hàm symbolic sau  
 $Y = x^2 + x + y + z + 1;$   
Bạn hãy nêu thứ tự ưu tiên các biến .
2. Tạo hàm symbolic sau dùng các cách tạo hàm khác nhau rồi tích đạo hàm , tích phân của nó  $Y = 1/(5 + 4 * \cos(x))$
3. Vẽ đồ thị hàm trên, theo hai cách thông thường và sử dụng symbolic

## CHƯƠNG 3

### MA TRẬN VÀ MẢNG TRONG MATLAB

#### 3.1 Nhập ma trận trong Matlab

##### 3.1.1 Các Cách nhập ma trận trong Matlab

Matlab cung cấp một vài phương tiện cho người sử dụng để tạo ra một ma trận, mỗi phương tiện có những ưu điểm của nó và được sử dụng tùy theo từng yêu cầu bài toán. Nói chung Matlab cung cấp ba phương tiện.

- Nhập Ma trận trực tiếp từ cửa sổ command Window.
- Nhập Ma trận từ một file( sử dụng M-file hoặc load)
- Nhập ma trận từ những hàm có sẵn trong Matlab.

##### a. Nhập Ma trận trực tiếp từ cửa sổ command Window

Trong môn học toán cao cấp chúng ta đã biết nhập một ma trận như sau

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

Đây là một ma trận có số hàng  $m = 3$  và số cột  $n = 3$

Để nhập ma trận trên trong Matlab ta nhập trực tiếp như sau

Từ dòng nhắc lệnh trong cửa sổ command Window  $>>$  ta nhập

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

```
4 5 6
```

```
7 8 9];
```

Các hàng được cách nhau bằng một dấu chấm phẩy (;) như trên, các phần tử trong một hàng được cách nhau bằng dấu cách (thanh space) hoặc dấu phẩy(.). Kết thúc dòng lệnh có hoặc không có dấu ;

Nếu không có dấu chấm phẩy ở cuối dòng thì Matlab sẽ in ra kết quả ma trận vừa nhập

Như ví dụ trên:

```
>> A=[ 1,2,3 ; 4 5 ,6;7 8 9] nhấn Enter sẽ cho kết quả là
```

```
A=
```

```
1 2 3
```

```
4 5 6
```

Trong trường hợp số phần tử trên một hàng quá dài ta có thể xuống dòng bằng dấu ba chấm ...

Ví dụ 

```
>> b=[1,2,3,4,...
```

```
5 6 7 8 9] % đây ma trận 9 hàng và một cột
```

Lưu ý rằng trong một số trường hợp ma trận hoặc mảng dữ liệu dài thì việc không thêm dấu chấm phẩy sau câu lệnh nhập, Matlab sẽ in ra số liệu dài trong cửa sổ command Window, gây khó nhìn cho người dùng

### **b. Nhập Ma trận từ M-file**

Ta có thể nhập một ma trận bằng cửa sổ soạn thảo M-file, mở cửa sổ này bằng cách vào File- New- M-file. Một cửa sổ soạn thảo sẽ được hiện ra cho phép bạn soạn thảo dưới dạng text, do là cửa sổ soạn thảo dạng text cho nên bạn có thể soạn thảo từ file word sau đó copy vào cửa sổ M-file. Để nhập ma trận ta soạn thảo tương tự như trong cửa sổ command window sau đó lưu vào file như sau:

Ví dụ:

```
A=[1 2 3 ; 4 5 6 ; 7, 8,9]; % không có dấu chấm phẩy sẽ in ra kết quả
```

Cũng tương tự như trên nếu số phần tử trên một hàng quá nhiều thì ta có thể xuống dòng

```
A=[1 2 3 4 ...  
5 6 7 8 9 10];
```

Sau khi kết thúc soạn thảo ta lưu vào tên\_file .

Để thực thi các lệnh nhập trong M-file ta dùng lệnh sau trong command window như sau: >> ten\_file ;

### **c. Nhập ma trận từ các hàm có sẵn**

Matlab có một thư viện các hàm cho phép tạo ma trận. Sau đây là một số hàm

- ones(m,n) tạo ma trận m hàng và n cột , với các phần tử đều bằng 1, ones(m) tạo ma trận vuông cấp m, với các phần tử đều là 1.
- zeros(m,n) tạo ma trận kích thước m x n, với các phần tử đều bằng 0, zeros(m) tạo ma trận vuông cấp m.
- eyes(m,n) tạo ma trận kích thước m x n với các phần tử đều bằng 1, eyes(m) tạo ma trận vuông cấp m .

ví dụ:

```
ones(2,3)
```

```
ans=
```

```
1 1 1
```

```
1 1 1
```

```
eyes(2,3)
```

```
ans=
```

```
1 0 0
```

```
0 1 0
```

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

```
ans=
```

```
0 0 0
```

0 0 0

### 3.2 Ma trận số phức

Số phức trong matlab được viết như sau:

Ví dụ số phức  $3+4*i$  dùng  $i$  để chỉ số ảo

```
>> a=3+ 4*i
```

a=

3+ 4\*i

Nếu muốn  $ii$  để chỉ số ảo

Ta định nghĩa  $ii= \text{sqrt}(-1)$

Sau đó bạn viết:

```
>> a=3+ 4*ii
```

a=

3+ 4\*i

```
>>A=[ 1+2*i , 3+4*i ; 5+6*i, 4+5*i ]
```

A=[ 1+2\*i 3+ 4\*i

5+6\*i 4+5\*i ]

### 3.3 Tạo vec tơ

Khi ta cần khảo sát đặc tính của đồ thị nào đó trong một khoảng xác định, khoảng xác định này được biểu diễn dưới dạng vectơ

Ví dụ khảo sát đặc tính đồ thị trong khoảng  $x=1$  đến  $100$

```
>> x= 1:100; % x lấy giá trị từ 1 đến 100, bước tăng của x là 1
```

```
>>t=0: 0.1 : 10;% bước nhảy là của t là 0.1
```

Công thức chung tạo vec tơ là  $X=X_{\min} : \text{bước\_tăng} : X_{\max}$

### 3.4 Truy nhập các phần tử của ma trận

Để truy nhập các phần tử của ma trận ta làm như sau:

Giả sử ma trận

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

Thì `>> A(i,j)` ; sẽ truy nhập phần tử hàng thứ  $i$  và cột thứ  $j$

Ví dụ để truy nhập đến phần tử thứ nhất ta :

```
>> A(1,1)
```

ans=

1

Đặc biệt để gọi toàn bộ số hàng hoặc toàn bộ số cột dùng toán tử (`:`)

```
>> A(:,1) % gọi toàn bộ số hàng tương ứng với cột 1
```

ans=

```

1
4
7
>>A(1,:) % gọi toàn bộ số cột tương ứng hàng 1
ans=
    2    3
>> A(1:2,1) % gọi hàng 1 đến hàng 2 tương ứng với cột thứ nhất
ans=
    1
    4
>>A(1:2,:) % gọi hàng 1 đến hàng 2 tương ứng với tất cả các cột
ans=
    1 2 3
    4 5 6

```

### 3.5 Phép tính ma trận và mảng

#### a. Phép tính ma trận

- Phép tính cộng , phép tính trừ :Điều kiện hai ma trận A và B phải có cùng kích thước hoặc một trong hai là số vô hướng

ví dụ:

```

>>a=[1 2 3 ;4 5 6; 7 8 9];
>>b=[2 3 4; 5 6 7; 8 9 10];
>>a+b;
ans=
    5    7
    9   11  13
   15   17  19

```

- Nhân hai ma trận

$A*B$  lưu ý rằng số cột của ma trận A phải bằng số cột của ma trận B, ngoại trừ một trong hai là số vô hướng

- Chia trái ma trận (\)

$X=A\backslash B$  tương đương với việc giải hệ phương trình tuyến tính  $A*X=B$ , gần tương đương với  $X=inv(A)*B$

- Chia phải ma trận (/)

$X=B/A$  tương đương với việc giải phương trình tuyến tính

$X*A=B$  gần tương đương với  $X= B*inv(A)$

#### b. Phép tính dãy

Cho hai mảng sau:

```
>>x=[1 2 3];
```

```
>>y=[2 3 4];
```

- Phép tính cộng , trừ giống như phép tính đối với ma trận

```
>>x+y
```

```
ans=
```

```
5 7
```

- Phép tính nhân(.\*)

```
>>x.*y
```

```
ans=
```

```
2 6 12
```

- Phép tính chia(/ hoặc ./)

```
>> x./y
```

```
ans=
```

```
0.5 0.66 0.75
```

```
>>x ./y
```

```
ans= 2 1.5 0.75
```

3.6 Giải hệ phương trình tuyến tính

3.6.1 Hệ phương trình tuyến tính :

**Xét hệ phương trình sau:**

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

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

.

.

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m$$

Bài toán đặt ra là tìm véc tơ  $x=[x_1;x_2;x_3;\dots;x_n]$  sao cho thoả mãn bài toán trên

3.6.2 Hệ Phương trình tuyến tính không đồng nhất

Phương trình như sau gọi là phương trình tuyến tính KĐN

$$a_1x_1 + a_2x_2 + \dots + a_nx_n = b$$

**b đứng độc lập (nó không nhân với biến nào cả)**

**Xét hệ thống sau:**

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

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

.

.

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m$$

Viết theo ma trận  $A= [a_{11} a_{12} \dots a_{1n}; a_{21} a_{22} \dots a_{2n}, \dots, a_{m1} a_{m2} \dots a_{mn}]$

$$X=[x_1 \ x_2 \dots \ x_n]; B=[b_1 \ b_2 \dots \ b_n];$$

Trong đó A được gọi là ma trận hệ số, X là vector kết quả

3.6.2.1 Giải hệ phương trình bằng hàm nghịch đảo inv

Nếu  $m=n$  thì A là ma trận vuông, và nếu  $\det(A)$  là khác 0 thì tồn tại  $A^{-1}$  và vector kết quả X được cho bởi :

$$A^{-1} \cdot A \cdot X = X = A^{-1} \cdot B$$

Ví dụ Giải hệ sau:

$$2 \cdot x_1 - x_2 = 2$$

$$x_1 + x_2 = 5$$

Matlab command

```
>> A=[ 2 -1 ; 1 1 ];
```

```
>> B=[ 2 ; 5];
```

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

```
>> X=
```

```
2.3333
```

```
2.667
```

```
>> X= rats(X)
```

```
X=
```

```
7/3
```

```
8/3
```

Tuy nhiên chúng ta không thể áp dụng phương pháp trên cho

$$2 \cdot x_1 - x_2 = 2$$

$$2 \cdot x_1 - x_2 = 0$$

Ma trận hệ số  $A=[ 2 \ -1 ; 2 \ -1];$

Vì  $\det(A)=0 \Rightarrow$  không áp dụng được hàm nghịch đảo cho ma trận A

3.6.3 Hệ phương trình tuyến tính đồng nhất

Biểu diễn dưới dạng ma trận như sau

$$A \cdot x = 0$$

- Nếu  $\det(A) \neq 0$  hệ có nghiệm duy nhất là  $X=0$

Ví dụ xét hệ phương trình tuyến tính sau

$$2 \cdot x_1 - x_2 = 0$$

$$x_1 + x_2 = 0$$

ở đây  $\det(A)=3$  cho nghiệm  $x_1=0, x_2=0$

- Đối với hệ phương trình thuần nhất có  $\det(A)=0$  thì hệ này có vô số nghiệm

Ví dụ Xét hệ phương trình tuyến tính sau

$$-6 \cdot x_1 + 3 \cdot x_2 = 0$$

$$2 \cdot x_1 - x_2 = 0$$

Ma trận hệ số  $A = \begin{bmatrix} -6 & 3 \\ 2 & -1 \end{bmatrix}$ ,  $\det(A) = 0$  biểu diễn trên đồ thị thấy rằng hai đường này trùng nhau do vậy hệ trên có vô số nghiệm

- Trường hợp số biến  $n <$  số phương trình  $m$

Ví dụ như sau:

$$3x_1 + 4x_2 - 2x_3 = 0$$

$$-2x_1 + 3x_2 - 4x_3 = 0$$

$$5x_1 + x_2 + 2x_3 = 0$$

$$-9x_1 + 5x_2 - 10x_3 = 0$$

Ma trận hệ số là ma trận  $4 \times 3$ , định thức lớn nhất có thể được xây dựng từ ma trận  $A$  là định thức ma trận  $3 \times 3$ , nhưng định thức của ma trận kích thước  $3 \times 3 = 0$  ( $A_1 = \begin{bmatrix} 3 & 4 & -2 \\ -2 & 3 & -4 \\ 5 & 1 & 2 \end{bmatrix} \Rightarrow \det(A_1) = 0$ )

Do đó ta xác định tiếp ma trận  $2 \times 2$

Ví dụ như sau

$A_2 = \begin{bmatrix} 3 & 4 \\ -2 & 3 \end{bmatrix}$  và  $\det(A) \neq 0$  ta nói rằng hạng của ma trận  $A$  (ma trận hệ số) là bằng 2 đồng nghĩa với việc ta chỉ giải hai phương trình bất kỳ trong số tất cả các phương trình trên, và số biến chúng ta gán giá trị tùy ý là  $n - r$  (trong đó  $n$  là số biến còn  $r$  là hạng của ma trận  $A$ )

Giải hai phương trình :

$$3x_1 + 4x_2 - 2x_3 = 0$$

$$-2x_1 + 3x_2 - 4x_3 = 0$$

Kết quả :  $x_1 = (-10/17)x_3$  và  $x_2 = (16/17)x_3$ , với  $x_3$  lấy giá trị tùy ý

3.6.4 Giải hệ phương trình tuyến tính bằng Matlab (Dùng toán tử \)

$$2x_1 - x_2 = 2$$

$$x_1 + x_2 = 5$$

```
>> A=[ 2 -1 ; 1 1];
```

```
>> B=[2 ; 5];
```

```
>> X=A\B
```

Phương pháp giải này gọi là phương pháp Gaussian elimination

Toán tử (\) thông thường cung cấp một kết quả trong Matlab, trong một số trường hợp nó là phương pháp giải riêng

3.7 Điều kiện có nghiệm

Theo Kronecker-Capelli thì

Một hệ phương trình có một lời giải khi và chỉ khi ma trận hệ số  $A$  và ma trận  $[A \ B]$  có cùng hạng.

Giả sử hạng của hai ma trận đều là  $r$  thì xảy ra các trường hợp sau đây

- $r = n$  Hệ phương trình có nghiệm duy nhất,



- $r < n$  Hệ phương trình có vô số nghiệm, chúng ta có thể giải cho  $r$  biến như là hàm của  $n-r$  biến khác, các biến khác này có thể lấy giá trị tùy ý

Ví dụ trên

$\text{rank}(a) = \text{rank}([a \ b]) = n$  cho nên hệ nghiệm duy nhất

```
>> rank(A), rank([A B])
```

```
ans=
```

```
2
```

```
ans=
```

```
2
```

Chúng ta xem xét ví dụ sau:

$$2 * x_1 + 3 * x_2 + 4 * x_3 = 4$$

$$x_1 + x_2 + x_3 = 5$$

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

```
>>B=[ 4 ; 5];
```

```
>>rank(A), rank([A B])
```

```
ans=
```

```
2
```

```
ans=
```

```
2
```

```
>> X= A\B
```

```
X=
```

```
8
```

```
0
```

```
3
```

Hạng của hai ma trận  $A$  và  $[A \ B]$  bằng nhau và bằng 2 cho nên hệ có một lời giải, nhưng do  $\text{rank}(A) < n$  cho nên ta chỉ giải cho hai biến như là hàm của biến còn lại. Kết quả Matlab cho trên chỉ là một trường hợp riêng ( $n-r$  biến được gán =0)

Xét hệ sau

$$x_1 + 2 * x_2 + 3 * x_3 = 12$$

$$3 * x_1 + 2 * x_2 + x_3 = 15$$

$$3 * x_1 + 4 * x_2 + 7 * x_3 = 13$$

$$10 * x_1 + 9 * x_2 + 8 * x_3 = 17$$

Tính toán bằng Matlab như sau

```
>> A=[1 2 3 ; 3 2 1 ; 3 4 7; 10 9 8];
```

```
>>B= [12 ; 15; 13 ; 17 ];
```

```
>>rank(A), rank([A B])
```

```
ans=
```

3

```
ans=
```

```
4
```

```
>> X= A\B
```

```
ans=
```

```
1.0887
```

```
-0.2527
```

```
1.5349
```

Khi thử lại như sau

```
>> A* ans
```

```
ans= 5.1882
```

```
4.2957
```

```
13.0000
```

```
20.8925
```

Kết quả không bằng B

Hệ phương trình trên vô nghiệm ,tuy nhiên Matlab vẫn cho nghiệm ,nghiệm này không phải nghiệm đúng mà là nghiệm xấp xỉ giải theo tiêu chuẩn bình phương tối thiểu( ta không đề cập tới)

### 3.8 Hệ điều kiện yếu

Chúng ta nói rằng một vấn đề được coi là điều kiện yếu nếu một sự thay đổi nhỏ trong dữ liệu sẽ dẫn đến thay đổi lớn trong kết quả. Điều này là rất nguy hiểm đối với các kỹ sư làm việc với các thiết bị , sai số ở các thiết bị , sai số do làm tròn (điều này chắc chắn xảy ra) Nếu dữ liệu này là đầu vào đối với vấn đề trên thì kết quả thu được sẽ khủng khiếp Vấn đề chúng ta bàn tới là Điều kiện yếu của hệ phương trình tuyến tính

Ma trận yếu điển hình là ma trận Hibert có dạng như sau:

$$A = \begin{bmatrix} 1 & 1/2 & 1/3 & \dots & 1/n \\ 1/2 & 1/3 & 1/4 & \dots & 1/(n+1) \\ 1/3 & 1/4 & 1/5 & \dots & 1/(n+2) \\ \dots & \dots & \dots & \dots & \dots \\ 1/n & \dots & \dots & \dots & 1/(2n) \end{bmatrix}$$

Ví dụ sau đây: Giải hệ phương trình tuyến tính có ma trận hệ số sau

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1.01 \end{bmatrix} \quad B = \begin{bmatrix} 2 \\ 2.01 \end{bmatrix};$$

```
>> X= A\B
```

```
X=
```

```
1.0000
```

```
1.0000
```

Một sai số nhỏ được thể hiện trong long format

```
>> format long; X= A\B
```

```
X=
```

```
1.000000000002
0.999999999998
```

Nếu ta thay đổi một phần tử của A ví dụ  $A(1, 2)=1.005$

```
>> A(1,2)=1.005 ; X= A\b
```

```
X=
```

```
-0.0000000099991
1.9999999999991
```

Thay đổi  $A(1,2) = 1.005$  so với giá trị cũ là 1 tức là tăng 0.5% tương ứng với giá trị  $x(1)$  giảm 101%, và tăng  $x(2)$  tăng 100%

Cách giải hệ phương trình điều kiện yếu  $A*X=B$

Nếu A là ma trận Hillbert sử dụng hàm tính nghịch đảo  $\text{invhilb}(n)$  trong đó n là kích thước của ma trận đó

Ví dụ >>A= [ 1/1 1/2 ; 1/2 1/3];

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

```
>>X= invhilb(2)* b
```

Nếu A không phải là ma trận hilbert thì sử dụng hàm symbolic

Ví dụ A= [ 1 1.01; 0.5 1.02];

```
A=sym( [1 1.01 ; 0.5 1.02] );
```

```
B=[ 1.1; 1.2];
```

```
X= A\b
```

3.9 Lệnh cond Tính điều kiện của ma trận

Cấu trúc:

```
>> cond(A) % A là ma trận
```

kết quả trả lại dạng như sau:  $a \cdot 10^k$ ;  $0 < a < 9$

k là số digits không tin cậy trong kết quả giải hệ phương trình tuyến tính và trong việc nghịch đảo ma trận. Nếu k xấp xỉ 1 thì đó là ma trận có well-condition

Ví dụ >>A=[1/2 1/3 1/4 ; 1/3 1/4 1/5; 1/4 1/5 1/6];

```
>> cond(A)
```

```
ans=
```

```
1.3533e+003
```

Ta thấy rằng  $k=3$  tức là có 3 số không đáng tin cậy

Tổng kết

Định nghĩa : Hàng ma trận

$A_r$  là một ma trận r hàng r cột được xây dựng từ A, không nhất thiết theo thứ tự trong ma trận A và  $\det(A_r) \neq 0$ . Nếu bất kỳ ma trận  $A_{r+1}$  nào được xây dựng từ r+1 hàng và r+1 cột của A,  $\det(A_{r+1})=0$  thì chúng ta nói rằng Ma trận A có hạng bằng r

Một hệ thống m phương trình tuyến tính trong n biến (chưa biết)

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

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

.

.

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m$$

Có thể viết dưới dạng form ma trận  $AX=B$

Trong đó A là ma trận hệ số và X là vector kết quả

Điều kiện có nghiệm Ma trận  $[A \ B]$  được gọi là ma trận mở rộng của hệ. Theo Kronecker-Capelli thì hệ phương trình tuyến tính có nghiệm khi và chỉ khi hạng của ma trận A bằng hạng của ma trận bổ xung

- Nếu  $r = n$  thì nghiệm trên là duy nhất
- Nếu  $r < n$  thì hệ trên không xác định và r biến có thể được biểu diễn dưới dạng hàm của n-r biến khác, các biến khác này có thể cho giá trị bất kỳ (nói cách khác hệ vô số nghiệm)

Nghiệm của hệ phương trình tuyến tính được tính trong Matlab bằng toán tử ( $\backslash$ ). Nếu hệ có nghiệm duy nhất Matlab sẽ cung cấp cho nó, nếu hệ là không xác định ( $r < n$ ) thì toán tử ( $\backslash$ ) sẽ cung cấp cho chúng ta một nghiệm riêng trong đó n-r biến sẽ được đặt =0.

Một nghiệm, nghiệm này làm thoả mãn tổng bình phương của các nghiệm bé nhất

Dùng lệnh  $X = \text{pinv}(A) * B$

Nếu hạng của A  $\neq$  hạng ma trận mở rộng thì toán tử ( $\backslash$ ) cung cấp một kết quả nhưng kết quả này không phải là nghiệm của hệ

Hệ thuần nhất khi vector  $B=0$ . Một hệ thuần nhất có một nghiệm tầm thường khi  $\det(A) \neq 0$ . Nếu  $\det(A)=0$  hệ có nhiều hơn một nghiệm trong trường hợp này Matlab sẽ cảnh báo người dùng :

### *Câu hỏi ôn tập*

1. Các cách nhập một ma trận ?
2. Điều kiện có nghiệm của hệ phương trình đại số tuyến tính, cách tính
3. Lập chương trình mà đầu vào là hai ma trận A và b, đầu ra là kết quả thông báo hệ có nghiệm hay không.
4. Hệ phương trình điều kiện yếu là gì?, những ảnh hưởng của nó.

### *Bài tập*

Thực hiện các phép toán sau

1. Nhập hai ma trận  $a = [1 \ 2 \ 3; 4 \ 5 \ 6]$ ,  $b = [5 \ 6 \ 7; 8 \ 9 \ 10]$ .

- Tính  $\text{Addab} = a + b$ .

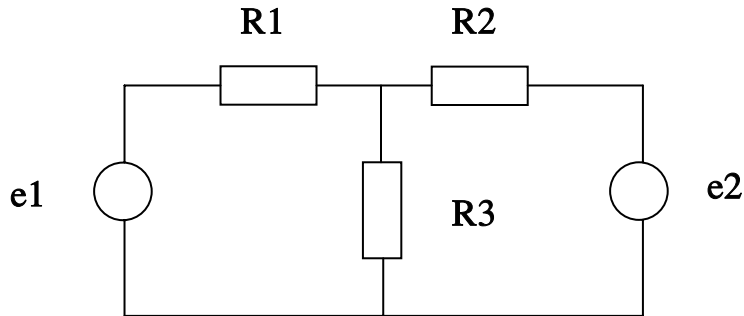
- Tính  $\text{Subsab} = a - b$ ;

- Tính  $\text{Multab} = a * b$

-Tính  $Mulba=b*a$ ;

-Tính  $Divab= a/b$ ; và  $b/a$

2. Cho mạch điện sau



Cho thông số:  $R1= 10(\text{omh})$  ,  $R2= 20(\text{omh})$  ,  $R3= 10(\text{omh})$

$e1= 20(\text{v})$  ,  $e2= 30(\text{v})$

Tính dòng điện  $I1$  và  $I2$  và  $I3$  bằng cách lập theo dạng hệ phương trình đại số tuyến tính ba ẩn số

**Bài tập giải hệ phương trình tuyến tính sau:**

$$A*X= B$$

Trong đó:  $A=[1/2 \ 1/3 \ 1/4 ; 1/3 \ 1/4 \ 1/5; 1/4 \ 1/5 \ 1/6]$ ;  $B=[0.95 \ 0.67 \ 0.52]$

1. Giải hệ đã cho

2. Thay đổi  $B(3)=0.53$  rồi giải hệ phương trình, so sánh với trường hợp trên

3. Tính điều kiện của ma trận này và đưa ra nhận xét

Chú ý khi giải hệ phương trình tuyến tính với ma trận hệ số là ma trận Hilbert (ma trận điều kiện yếu) thì ta dùng hàm tính nghịch đảo của nó là hàm invhilb(n)

Định nghĩa Ma trận Hilbert là:  $A=[ 1 \ 1/2 \ 1/3.....1/n; 1/2 \ 1/3 \ ...1/(n+1);$

# Chương 4

## Đồ hoạ Trong Matlab

### 4.1 Điểm và đường trong đồ hoạ matlab

Dùng hàm Plot để vẽ điểm -Đường thẳng trong mặt phẳng

Để vẽ các đường trong mặt phẳng,các hàm số phụ thuộc vào biến ví dụ như  $y=f(x)$  thì trong matlab cung cấp cho ta hàm plot(x,y) để vẽ ,trong không gian ba chiều thì dùng hàm plot3(x,y,z) .

trước hết ta nói qua về cách dùng hàm plot và các ví dụ minh hoạ cụ thể để hiểu rõ hơn về vấn đề này:

#### 4.1.1 Lệnh plot

Syntax

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

Mô tả:

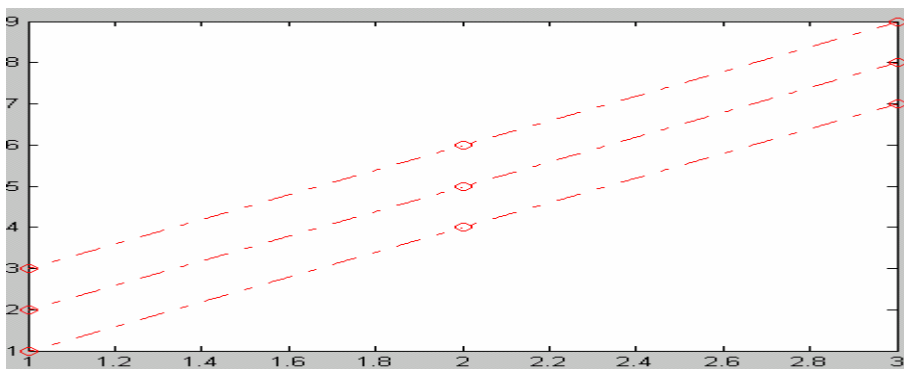
Hàm plot có nhiều cách dùng như bạn đã thấy ở trên

plot(y): Hàm này để biểu diễn các cột của y theo các chỉ số tương ứng của chúng nếu y là ma trận các số thực, nếu y là số phức thì plot(y) tương ứng với plot(real(y),image(y)).

Ta có thể lấy ví dụ sau:

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

plot(A) sẽ được kết quả như sau(giao điểm ký hiệu là dấu o tròn )



Các dấu tròn trên hình vẽ thể hiện các giao điểm giữa các phần tử của các cột và các chỉ số tương ứng của chúng trong từng cột .

Cụ thể các giao điểm (1,1) và (2,1);(3,1) tương ứng là phần tử thứ nhất của các cột,do là phần tử thứ nhất cho nên có chỉ số là 1

**plot(x,y...):** Vẽ các đường thẳng tương ứng với các cặp điểm (x,y) của véc tơ x và véc tơ y.Nếu chỉ một trong x hoặc y là ma trận thì nó sẽ vẽ theo vector cột hoặc hàng tương ứng với vector còn lại phù hợp với kích thước hàng hay cột của ma trận đó.

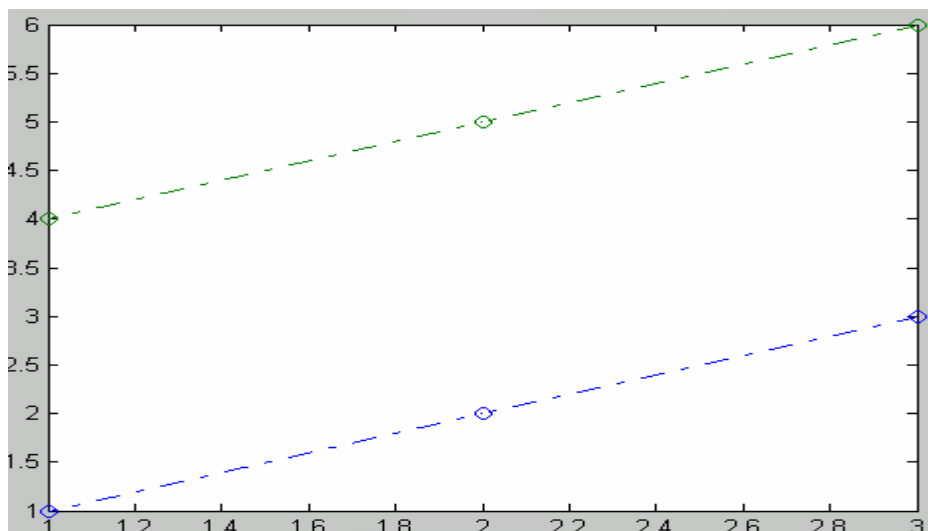
Cụ thể :

Giả sử x: là vector cot

$$x=[1 \ 2 \ 3]; \text{ và}$$

y là ma trận  $y=[1 \ 2 \ 3;4 \ 5 \ 6];$

rõ ràng là x có kích thước bằng với kích thước hàng của ma trận do vậy mà nó sẽ biểu diễn các hàng của y theo x .Kết quả plot(x,y) như sau:



plot(x,y,linespec...) cũng có thể viết như sau plot(x,y,linespec,x1,y1,linespec1,...);

Hàm này giống như hàm trên ,nhưng các thuộc tính về đường được thể hiện ở trong linespec .Sau đây ta có thể liệt kê các thuộc tính về đường

Matlab cho phép bạn sử dụng một số ký tự sau đây để xác định thuộc tính của đường

- Line style
- Line width
- Color
- Marker type
- Marker size
- Marker face and edge coloring (for filled markers)

Matlab định nghĩa các chuỗi xác định cho kiểu đường, Marker types và colors

### 1. Line Style Specifiers

Specifier	Line Style
-	đường liền(default)
--	đường nét đứt
:	dotted line
-.	dash-dot line

### 2. Marker Specifiers

Specifier	Marker Type
+	plus sign
O	Circle
*	Asterisk
.	Point
X	Cross
S	Square
D	Diamond
^	upward pointing triangle
V	downward pointing triangle
>	right pointing triangle



<	left pointing triangle
P	five-pointed star (pentagram)
H	six-pointed star (hexagram)

### 3. Color Specifiers

Specifier	Color
R	Red
G	Green
B	Blue
C	Cyan
M	Magenta
Y	Yellow
K	Black
W	White

Các lệnh plot chấp nhận một thông số Linespec ,thông số này định nghĩa ba phân tử ,các phân tử này xác định đường

- Line style (kiểu đường)
- Marker symbol (Kiểu đánh dấu)
- Color (kiểu màu)

Chú ý rằng khi kết hợp chúng ta có thể để theo một thứ tự bất kỳ

For example

```
plot(x,y,'-or')
```

Vẽ y theo x sử dụng kiểu đường là dash-dot đặt vòng tròn(o) tại các giao điểm (x,y) ,và màu của đường và màu của vòng tròn đánh dấu là màu đỏ  
Nếu bạn xác định một điểm đánh dấu, không phải là kiểu đường, Matlab chỉ vẽ các điểm đánh dấu

I. Ví dụ  
plot(x,y,'d')

#### 4.1.2 Hàm plot(x,y, protypename,prototypevalue....)

Hàm này xác định rõ các thuộc tính của đường thẳng ví dụ như Chiều rộng của đường thẳng

**LineStyle** {-} | -- | : | -. | none

Độ rộng của đường(Linewidth) mặc định là 0.5 points( 1point=1/72 inch)

Ví dụ về Lineproperty:

```
plot(t,sin(2*t),'-mo',...  
      'LineWidth',2,...  
      'MarkerEdgeColor','b',...  
      'MarkerFaceColor','r',...  
      'MarkerSize',12)
```

Giải thích như sau:

Hàm trên vẽ đồ thị  $f=\sin(2*t)$  theo biến t, đặc tính của đường là màu (magne) ,giao hai điểm là hình tròn(s), đường là liên tục(solid line)

Line width là 2(point) ( 1point=1/72 inches) default là 0.5 points

MarkerEdgeColor là màu đen( blue)

Mẫu trong (mặt) của các điểm nút giao là màu đỏ:

**Chúng ta thấy rằng đồ thị được xây dựng từ việc nối các điểm có tọa độ (x,y) bằng các đoạn thẳng**

**\*tỷ lệ các trục sẽ được matlab tự động tạo ra sao cho phù hợp**

4.1.3 Để vẽ nhiều đồ thị trên cùng một hình vẽ thì chúng ta có hai cách

+Vẽ đồ thị thứ nhất

+ Dùng lệnh Hold on

+Vẽ tiếp đồ thị thứ hai + hold off

hoặc Dùng hàm plot(x1,y1,x2,y2)

Ví dụ ta vẽ hai hàm  $y=\sin(x)$  và  $y1=\cos(x)$

Ta dùng lệnh plot như sau:

```
plot(x,y,x,y1);
```

#### 4.1.4 Title , xlabel, ylabel, gtext, legend, grid

Title dùng để viết tiêu đề cho đồ thị

Ví dụ Title(.....)

Xlabel( string) đặt tên nhãn cho trục x

Ylabel(string) đặt tên nhãn cho trục y

Gtext(string) để viết text vào đồ thị

Legend(string,-1) để ghi chú thích cho đồ thị, số -1 để ghi chú thích bên ngoài các trục của hình vẽ

Grid on hoặc Grid off để mở hoặc tắt Grid

t=0:pi/6:pi;

plot(t,sin(2\*t),'-mo',...

'LineWidth',2,...

'MarkerEdgeColor','b',...

'MarkerFaceColor','r',...

'MarkerSize',14)

legend('y=sin(2\*t)','y=cos(2\*t)',-1);( chú ý viết đúng thứ tự)

grid on;

xlabel('trục thời gian');

ylabel('trục Sin và cos');

title('Đồ thị hàm sin(2\*t)');

hold on;

j=cos(2\*t);

plot(t,j,'-b+');

hold off;

#### 4.2 Hàm plot3(x,y,z) để vẽ các điểm và đường trong không gian

Ngoài việc thêm trục z các hàm này sử dụng giống như hàm plot(x,y)

Cấu trúc

plot3(X1,Y1,Z1,...)

plot3(X1,Y1,Z1,LineSpec,...)

plot3(...,'PropertyName',PropertyValue,...)

h = plot3(...)

Việc sử dụng các hàm này giống với hàm Plot trong 2D do vậy ta không đề cập tới nữa

Chú ý tới hàm View(a,b) để quan sát góc nhìn của đồ thị trong đó a là góc tính theo chiều ngược chiều kim đồng hồ từ phía âm của trục y còn b là góc nhìn tính bằng độ xuống mặt phẳng x,y

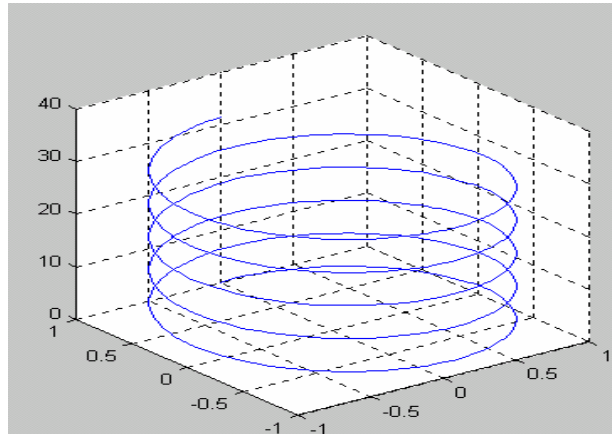
Giá trị mặc định của a và b là  $-37.5^{\circ}$  và  $30^{\circ}$

%Ví dụ

```

%Plot a three-dimensional helix.
% Vẽ mặt phẳng  $x = \sin(t)$ ,  $y = \cos(t)$ ,  $z = t$ 
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
grid on;
axis square
view(-80,30);
xét hai trường hợp
view(-80,30);
view(-40,30);
Khi cho  $a=0$  và  $b=90$  thì hình vẽ
trở về hình vẽ trong mặt phẳng
toạ độ hai chiều

```



#### 4.3 Hàm semilogx, semilogy

##### Semi-logarithmic plots

##### Cấu trúc

```

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

```

##### Mô tả

**Semilogx(y)** vẽ giống như **plot(y)** nhưng chỉ khác rằng tỷ lệ trên trục x là logarit cơ số 10, tương tự như vậy đối với **Semilogy(y)** thì tỷ lệ trên trục y theo logarit cơ số 10 ứng với trục x.

<code>X=0:10:1000;</code>	<code>subplot(2,2,2);</code>	<code>subplot(2,2,3);</code>	<code>subplot(2,2,4</code>
<code>Y=100*x;</code>	<code>semilogy(x,y,'.');</code>	<code>loglog(x,y,'.');</code>	<code>);</code>
<code>Subplot(2,2,1);</code>	<code>xlabel('tuyen</code>	<code>xlabel('log');</code>	<code>plot(x,y,'.');</code>
<code>Semilogx(x,y,'.');</code>	<code>ting');</code>	<code>ylabel('log');</code>	<code>xlabel('tuyen</code>
<code>Xlabel('log');</code>	<code>ylabel('log');</code>	<code>title('</code>	<code>ting');</code>
<code>Ylabel('tuyen ting');</code>	<code>title('</code>	<code>loglog(x,y)');</code>	<code>ylabel('tuyen</code>
<code>Title(' semilogx(x,y)');</code>		<code>grid on;</code>	<code>ting');</code>
<code>Grid on;</code>			<code>title('</code>

Trong không gian 3 chiều thì ta không dùng các hàm trên để vẽ ,mà ta sử dụng hàm plot3 và dùng hàm set để đặt trục tọa độ

```
t=1:1:100;  
x=sin(20*pi*t);  
y=cos(20*pi*t);  
z=t;  
subplot(1,2,1);  
plot3(x,y,z);  
set(gca,'Zscale','log');grid on;view(125,7);
```

4.4 Vẽ trong hệ tọa độ cực

**Cấu trúc**

```
polar(theta,rho)  
polar(theta,rho,LineStyle)
```

**Mô tả**

The polar function accepts polar coordinates, plots them in a Cartesian plane, and draws the polar grid on the plane.

polar(theta,rho) creates a polar coordinate plot of the angle theta versus the radius rho. theta is the angle from the x-axis to the radius vector specified in radians; rho is the length of the radius vector specified in dataspace units.

polar(theta,rho,LineStyle) LineStyle specifies the line type, plot symbol, and color for the lines drawn in the polar plot.

**Examples**

Create a simple polar plot using a dashed, red line:

```
t = 0:.01:2*pi;  
polar(t,sin(2*t).*cos(2*t),'--r')
```

this is a figure for plotting Polar(phi, r);

Khi chuyển từ hệ tọa độ cực sang hệ tọa độ Đêcart ta làm như sau [x,y]=pol2cart(phi, r) sau đó dùng lệnh Plot(x,y)

```
t = 0:.01:2*pi;  
h=sin(2*t).*cos(2*t);  
[x,y]=pol2cart(t,h); axis(equal)
```

```
plot(x,y);grid on;
```

notice : command Axis('equal') set unit which is divided in X and Y axis

Đối với hệ tọa độ cầu Trong Matlab không có hàm để vẽ .tuy nhiên ta có thể chuyển đổi từ hệ tọa độ cầu thành hệ tọa độ ĐêCart trong không gian

```
[x,y,z]=sph2cart(theta,phi,r) sau đó dùng hàm vẽ trong không gian là plot3(x,y,z)
```

#### 4.5 Đồ thị cột

bar, barh

#### Cấu trúc

```
bar(Y)
bar(x,Y)
bar(...,width)
bar(...,'style')
bar(...,LineStyle)
```

#### II. Mô tả

Vẽ biểu đồ các giá trị trong véc tor hoặc trong Matrxn như là thanh ngang hoặc thanh thẳng đứng

`bar(Y)` vẽ một đồ thị cột cho mỗi phần tử trong `Y`. Nếu `Y` là một ma trận, `bar` nhóm các thanh được tạo ra bởi mỗi phần tử trong mỗi hàng.

Tỷ lệ trục `x` từ 1 to `length(Y)` khi `Y` là một vector, và 1 đến `size(Y,1)`, đó chính là số hàng, khi `Y` là một ma trận.

`bar(x,Y)` vẽ một đồ thị cột cho mỗi phần tử trong `Y` tại các vị trí xác định trong `x`, ở đó `x` là vector tăng định nghĩa các khoảng cho các cột thẳng đứng. Nếu `Y` là một ma trận, `bar` gộp các cột tương ứng trong cùng một hàng trong `Y` tại vị trí tương ứng với một phần tử trong `x`.

`bar(...,width)` hàm này giống các hàm trên nhưng có thêm đặc tính đặt độ rộng của cột. Giá trị mặc định của `width` là 0.8. Nếu `width` is 1, các cột trong một nhóm chạm vào

`bar(...,'style')` Xác định kiểu của cột

'`style`' là 'group' hoặc 'stack'. 'group' là chế độ mặc định

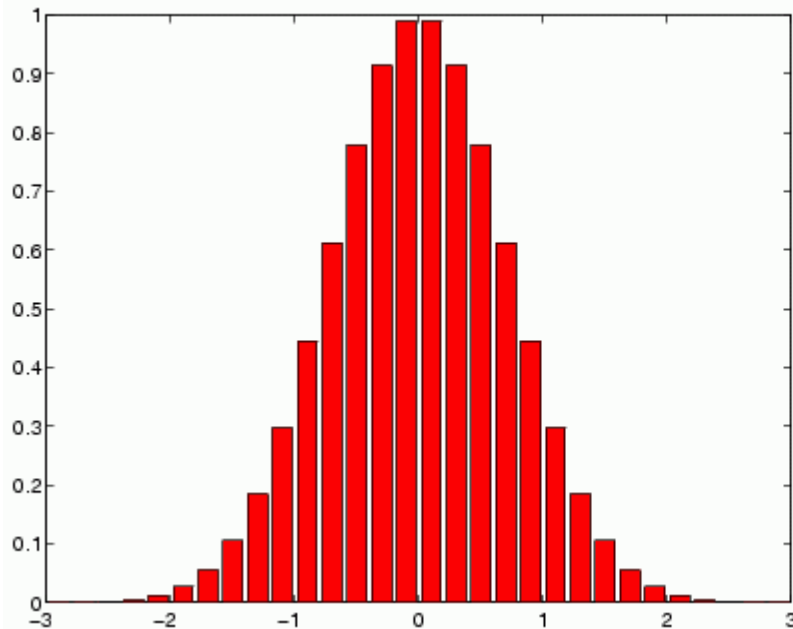
- 'group' biểu diễn  $n$  nhóm của  $m$  cột thẳng đứng, ở đó  $n$  là số hàng và  $m$  là số cột trong `Y`.
- 'stack' Biểu diễn đồ thị cột cho mỗi hàng của `Y`. Chiều cao của cột là tổng các phần tử trong một hàng

`bar(...,LineStyle)` displays all bars using the color specified by `LineStyle`.

Ví dụ

Plot a bell shaped curve:

```
x = -2.9:0.2:2.9;
bar(x,exp(-x.*x))
colormap hsv
```



Tương tự đồ thị cột trong không gian lệnh vẫn giữ nguyên nhưng thay vì bar ta thay lệnh `bar3(x,y,z)`

#### 4.6 Đồ thị bánh (Pie)

Cấu trúc:

`pie(x)` :Hàm này vẽ đồ thị bánh với các 'khoanh' được xác định bởi phần trăm các giá trị trong vector `x` ví dụ `x=[ 1 2 3 4]` phân thành 4 khoanh trên toàn bộ vòng tròn ,mỗi khoanh tương ứng phần trăm các phần tử trong `x`

```
>>x=[ 1 2 3 4];
```

```
>>pie(x)
```

cho đồ thị như sau:

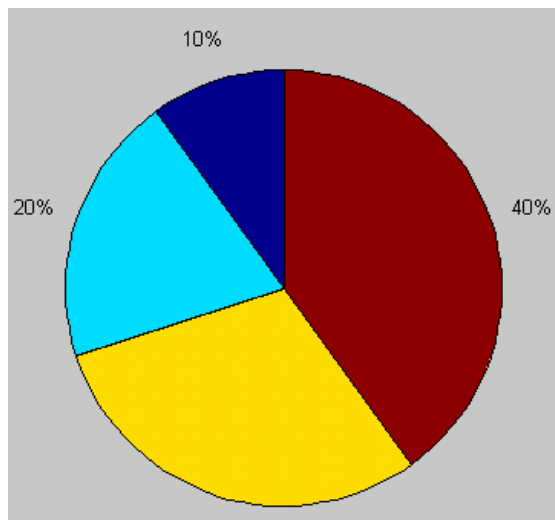
Đỉnh cao nhất ứng với phần tử đầu tiên của véc tor ,các phần tử tiếp theo được bố trí theo chiều ngược chiều kim đồng hồ .

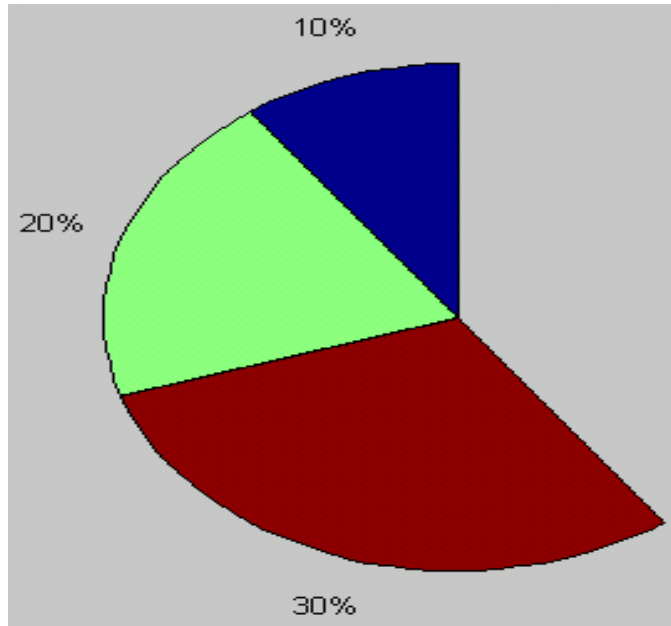
Nếu tổng các phần tử trong véc tor `x` <1 thì trên đồ thị bánh biểu diễn phần trăm chính là các phần tử đó .

Ví dụ

```
>> x=[ 0.1 0.2 0.3 ] % tương ứng với 10% 20% 30%
```

```
>>pie(x);
```







## chương 4

### Ma trận - các phép toán về ma trận.

#### 4.1 KHÁI NIỆM:

- Trong MATLAB dữ liệu để đưa vào xử lý dưới dạng ma trận.
- Ma trận A có n hàng, m cột được gọi là ma trận cỡ  $n \times m$ . Được ký hiệu  $A_{n \times m}$
- Phần tử  $a_{ij}$  của ma trận  $A_{n \times m}$  là phần tử nằm ở hàng thứ i, cột j .
- Ma trận đơn ( số đơn lẻ ) là ma trận 1 hàng 1 cột.
- Ma trận hàng (  $1 \times m$  ) số liệu được bố trí trên một hàng.

$$a_{11} \ a_{12} \ a_{13} \ \dots \ a_{1m}$$

- Ma trận cột (  $n \times 1$  ) số liệu được bố trí trên 1 cột.

$$a_{11}$$

$$a_{21}$$

$$a_{31}$$

$$\vdots$$

$$\vdots$$

$$a_{n1}$$

#### 4.1.1 Các qui định để định nghĩa một ma trận:

- Tên ma trận có thể gồm 31 ký tự. Bắt đầu phải bằng chữ cái sau đó có thể là số, chữ cái, các ký tự đặc biệt ... Tên đặt bên trái dấu bằng , bên phải dấu bằng là các phần tử của ma trận.
- Bao quanh các phần tử của ma trận bằng dấu ngoặc vuông.
- Các phần tử trong ma trận được cách nhau bởi ký tự trống hoặc dấu phẩy ( , ).
- Kết thúc một hàng trong ma trận bởi dấu ( ; ).

#### 4.1.2 Các cách để nhập một ma trận:

- Liệt kê trực tiếp: VD  $\gg A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$

$$\gg B = [1 \ 2 \ 3; \\ 4 \ 5 \ 6; \\ 7 \ 8 \ 9]$$

- Nhập thông qua lệnh. Dùng lệnh input

```
>> input('Nhap gia tri cho ma tran C = ')
Nhap gia tri cho ma tran C = [1 3 4;4 5 7;7 5 8]
ans =
     1     3     4
     4     5     7
     7     5     8
```

Chú ý khi kết thúc một câu lệnh có thể dùng dấu (;) hoặc không dùng dấu (;).

- Nếu dùng dấu (;) câu lệnh được thực hiện nhưng kết quả không hiện ra màn hình.
- Nếu không dùng dấu (;) câu lệnh được thực hiện và kết quả được hiện ra màn hình.
- Trong cả 2 trường hợp trên sau khi câu lệnh được thực hiện kết quả đều được lưu vào trong bộ nhớ và có thể sử dụng cho các câu lệnh tiếp theo.

Vd

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

Cả 2 ma trận A, B đều được lưu vào trong bộ nhớ và có thể được sử dụng cho những câu lệnh tiếp theo.

```
>> c = a*b
c =
    30    36    42
    39    48    57
    31    41    51
```

#### 4.1.3 Hiển thị lại ma trận:

- Để hiển thị lại ma trận ta gõ tên ma trận sau đó enter.

VD >> c

c =

```

30 36 42
39 48 57
31 41 51

```

- Để hiển thị nội dung của ma trận hoặc lời thông báo (trong dấu nháy đơn) ta dùng lệnh: disp

VD >> disp(c)

c =

```

30 36 42
39 48 57
31 41 51

```

>> disp('hiển thị lời thông báo này')

hiển thị lời thông báo này

### Chú ý:

- Các phần tử trong ma trận có thể là các số phức:

VD >> a=[1+3i 2+2i;3+i 1+i]

a =

```

1.0000 + 3.0000i 2.0000 + 2.0000i
3.0000 + 1.0000i 1.0000 + 1.0000i

```

- Các phần tử trong ma trận có thể là các ký tự. Nhưng trước tiên ta phải khai báo các phần tử bằng lệnh syms

VD >> syms sinx cosx a

>> b = [ sinx cosx; a cosx]

b =

```

[ sinx, cosx]
[ a, cosx]

```

>> c=[a sinx; a a]

```
c =  
    [ a, sinx]  
    [ a, a]
```

## 4.2. XỬ LÝ TRONG MA TRẬN:

### 4.2.1 Tạo vectơ từ ma trận:

Công thức tổng quát: Biến = giới hạn đầu : bước chạy : giới hạn cuối

Giới hạn đầu, giới hạn cuối, bước chạy: là các số thực

Bước chạy có thể dương hoặc âm.

VD Tạo 1 vectơ t chạy từ 0 đến 0.6 với bước chạy tiến là 0.1

```
>> t=0:0.1:0.6
```

```
t =
```

```
    0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000
```

VD: Tạo 1 vectơ t chạy từ 0.6 đến 0 với bước chạy lùi là 0.1

```
>>t=0.6:-0.1:0
```

```
t =
```

```
    0.6000    0.5000    0.4000    0.3000    0.2000    0.1000     0
```

**Chú ý :** Trong trường hợp giới hạn trên, giới hạn dưới là các số nguyên và bước chạy bằng 1 thì ta không cần đưa bước chạy vào trong biểu thức.

VD >> C = 1:5

```
C =
```

```
    1     2     3     4     5
```

### 4.2.2 Gọi các phần tử trong ma trận.

MATLAB cho phép ta xử lý đến từng phần tử của ma trận. Để truy cập đến từng phần tử của ma trận ta phải gọi được chúng thông qua chỉ số của từng phần tử.

Tên của ma trận( Chỉ số hàng, chỉ số cột)

VD:

```
>> A = [1:3; 4:6; 7:9]
```

```
A =
```

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

```
>> B = A(1,1)
```

```

B =
      1
>> A(3,3) = A(2,2) + B
A =
      1      2      3
      4      5      6
      7      8      6

```

Chú ý: Trong trường hợp ta muốn gọi tất cả các hàng hoặc tất cả các cột ta có thể dùng toán tử hai chấm ( : )

VD:

```

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

```

#### 4.2.3 Gọi 1 ma trận con từ một ma trận lớn.

VD

```

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

```

```

5
7 8
>> c=[a(1,1) a(3,3); a(2,3) a(3,1)]
c =
1 9
6 7

```

### 4.3 CÁC MA TRẬN ĐẶC BIỆT:

**4.3.1 Ma trận zeros.** Tất cả các phần tử trong ma trận đều bằng 0.

VD

```

>> C = zeros (2,3)
C =
0 0 0
0 0 0

```

```

>> d = zeros(3)
d =
0 0 0
0 0 0
0 0 0

```

**4.3.2 Ma trận ones.** Tất cả các phần tử trong ma trận đều bằng 1

VD

```

>> C = ones (2,3)
C =
1 1 1
1 1 1

```

```

>> d = ones(3)
d =
1 1 1
1 1 1
1 1 1

```

**4.3.3 Ma trận ma phương Magic:** Tổng tất cả giá trị các phần tử trên hàng = Tổng tất cả giá trị các phần tử trên cột = Tổng tất cả giá trị các phần tử trên đường chéo của ma trận

Vd

```
>> A = Magic (3)
```

```
A =
```

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

**4.3.4 Ma trận eye.** Tất cả các phần tử trên đường chéo có giá trị 1, các phần tử khác có giá trị 0.

VD

```
>> B = eye (3)
```

```
B =
```

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

#### 4.4 CÁC PHÉP TOÁN VECTOR:

Phép toán	Công thức	Matlab
Cộng, trừ	$A+B, A-B$	$A+B, A-B$
Nhân mảng	$A.B = C$	$A.*B$
Chia trái mảng	$B \setminus A$	$B \setminus A$
Chia phải mảng	$A/B$	$A./B$
Luỹ thừa mảng	$A^B$	$A.^B$

##### 4.4.1 Các phần tử là các số thực:

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

```
a =
```

```
1    1    2
2    1    1
```

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

```
b =
```

```
1    2    2
1    1    1
```

```
>> c=a.*b
```

```
c =
```

```
1 2 4  
2 1 1
```

```
>> d=a./b
```

```
d =
```

```
1.0000 0.5000 1.0000  
2.0000 1.0000 1.0000
```

```
>> e=a.\b
```

```
e =
```

```
1.0000 2.0000 1.0000  
0.5000 1.0000 1.0000
```

```
>> f=a.^b
```

```
f =
```

```
1 1 4  
2 1 1
```

#### 4.4.2 Các phân tử là các số phức.

```
>>a=[1+i 2+3i;3-4i 1+3i]
```

```
a =
```

```
1.0000 + 1.0000i 2.0000 + 3.0000i  
3.0000 - 4.0000i 1.0000 + 3.0000i
```

```
>> b=[2+i 2+2i;1-4i 3+3i]
```

```
b =
```

```
2.0000 + 1.0000i 2.0000 + 2.0000i  
1.0000 - 4.0000i 3.0000 + 3.0000i
```

```
>> c=a.*b
```

```
c =
```

```
1.0000 + 3.0000i -2.0000 +10.0000i  
-13.0000 -16.0000i -6.0000 +12.0000i
```

#### 4.4.3 Các phân tử là các tham số:

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



```

>>A=[a b; b c]
A =
[ a, b]
[ b, c]
>> B=A
B =
[ a, b]
[ b, c]
>> C=A.*B
C =
[ a^2, b^2]
[ b^2, c^2]

```

## 4.5 CÁC PHÉP TOÁN VỀ MA TRẬN:

### 4.5.1 Phép chuyển vị:

Phép chuyển đổi vectơ hàng thành vectơ cột gọi là phép chuyển vị. Thực hiện phép chuyển vị bằng toán tử dấu nháy đơn ( ' ).

VD

```

>> A = [1:3; 4:6; 7:9]
A =
     1     2     3
     4     5     6
     7     8     9
>> B = A'
B =
     1     4     7
     2     5     8
     3     6     9

```

Ma trận B được gọi là ma trận chuyển vị của ma trận A

### 4.5.2 Phép cộng - trừ ma trận.( + , - )

Phép cộng và trừ ma trận được thực hiện với các ma trận có cùng kích cỡ.

$$C_{ij} = A_{ij} + B_{ij}$$

$$D_{ij} = A_{ij} - B_{ij}$$

```
>> A = [1:3; 4:6; 7:9]
```

```
A =
```

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

```
>> B = A'
```

```
B =
```

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

```
>> C = A + B
```

```
C =
```

```
     2     6    10
     6    10    14
    10    14    18
```

#### 4.5.3 Phép nhân, chia ma trận:

$C = A \cdot B$ .

Để thực hiện được phép nhân trên thì số cột của ma trận A phải bằng số hàng của ma trận B.

$$C_{ij} = \sum_{k=1}^n A_{ik} \cdot B_{kj}$$

Các phần tử trong ma trận C được tính như sau:

*VD các phần tử trong ma trận là các số thực.*

```
>> A = [1 2 1; 1 0 1]
```

```
A =
```

```
     1     2     1
     1     0     1
```

```
>> B = [1 0 2; 2 1 1; 1 1 1]
```

```

B =
     1     0     2
     2     1     1
     1     1     1
>> C = A * B
C =
     6     3     5
     2     1     3

```

*VD các phần tử trong ma trận là các số phức.*

```

>> a=[1+2i 2+2i;1+3i 2+2i]
a =
 1.0000 + 2.0000i 2.0000 + 2.0000i
 1.0000 + 3.0000i 2.0000 + 2.0000i
>> b=[1+i 2+i;1+3i 2+i]
b =
 1.0000 + 1.0000i 2.0000 + 1.0000i
 1.0000 + 3.0000i 2.0000 + 1.0000i
>> c=a*b
c =
-5.0000 +11.0000i 2.0000 +11.0000i
-6.0000 +12.0000i 1.0000 +13.0000i

```

**VD c, c phÇn tử trong ma trÛn lµ c, c tham sè**

```

>> syms a b c
>> d=[2*a b c; a b c; 0 0 a]
d =
 [ 2*a, b, c]
 [ a, b, c]
 [ 0, 0, a]
>> e=[a b c; 2*a 2*b^2 c; a 0 b]
e =

```

```

[ a, b, c]
[ 2*a, 2*b^2, c]
[ a, 0, b]
>> f=d*e
f =
[ 2*a^2+2*b*a+c*a, 2*b*a+2*b^3, 2*c*a+2*c*b]
[ a^2+2*b*a+c*a, b*a+2*b^3, c*a+2*c*b]
[ a^2, 0, b*a]

```

Phép chia ma trận thực chất là phép nhân với ma trận nghịch đảo.

$$C = \frac{A}{B} = A * \frac{1}{B}$$

Lấy ma trận nghịch đảo thực hiện bằng hàm inv.

```

>> A = [1 2 1; 1 0 1]
A =
     1     2     1
     1     0     1
>> B = [1 0 2; 2 1 1; 1 1 1]
B =
     1     0     2
     2     1     1
     1     1     1
>> C = inv(B)
C =
     0     1.0000    -1.0000
    -0.5000    -0.5000     1.5000
     0.5000    -0.5000     0.5000
>> D = A*C
D =
    -0.5000    -0.5000     2.5000
     0.5000     0.5000    -0.5000

```

Chú ý: Trong các phép tính trên nếu thực hiện với một số thực thì tất cả các phần tử trong ma trận sẽ được cộng, trừ, nhân, chia ( / ) với số thực đó tùy thuộc vào phép toán tương ứng.

```
>> A = [1 2 1; 1 0 1]
```

```
A =
```

```
    1    2    1
    1    0    1
```

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

```
B =
```

```
    2    4    2
    2    0    2
```

**4.5.4 Phép quay ma trận:** Quay ma trận B đi 1 góc 90 độ theo ngược chiều kim đồng hồ.

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

```
a =
```

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

```
>> b=rot90(a)
```

```
b =
```

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

**4.5.5.Phép đảo ma trận:** Đảo các phần tử của ma trận từ trái sang phải.

```
>> c=fliplr(b)
```

```
c =
```

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

## Chương 5

### *Cơ sở phương pháp tính*

#### 5.1 Nội suy và thuật toán nội suy

Vì sao phải nội suy: Trong thực tế khi đo một đại lượng vật lý bất kỳ tại những điều kiện môi trường thay đổi (còn có nhiều đại lượng khác thay đổi) ta nhận được các giá trị rời rạc, và có tính thống kê, ứng với mỗi thời điểm ta nhận được một giá trị đo như vậy khi ta muốn xác định giá trị đo ở một thời điểm bất kỳ thì ta phải dùng phép nội suy.

Trong chương này ta chỉ tìm hiểu và tính toán cho 2 phép nội suy đó là :

+Nội suy lagrange cho bài toán một chiều

+Nội suy lagrange cho bài toán hai chiều

#### 5.1.1 Nội suy lagrange cho bài toán một chiều

##### I. Lý thuyết

Giả sử có n điểm đo rời rạc tương ứng với kết quả đo như sau:

x	x0	x1	x2	.....	xn
f	f0	f1	f2	.....	fn

Công thức nội suy lagrange bậc N tính giá trị đo được tại một điểm bất kỳ là :

Thuật toán nội suy:

% thuật toán nội suy cho bài toán một chiều

```
function T=NS1C(x,f,xa);
    i=length(x);
    j=length(f);
    T=0;n=i;
    if(i~=j)
        error('Ban nhap sai');
    end
    i=1;
    while(i<=n)
        g=1;j=1;
        while(j<=n)
            if(i~=j)
                g=g*(xa-x(j))./(x(i)-x(j));
            end
            j=j+1;
        end
        T=T+g*f(i);
```



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

### Mô tả

yi = interp1(x, Y, xi) trả về vector yi chứa các phần tử tương ứng với các phần tử của xi và giá trị trả về đó được xác định bằng cách sự nội suy(interpolation) trong vectors x and Y. The vector x xác định các điểm tại đó dữ liệu Y được cho trước (the points at which the data Y is given). Nếu Y là một ma trận, thì việc nội suy được thực hiện cho mỗi cột của Y và Yi có kích thước là yi is length(xi)-by-size(Y,2).

(the interpolation is performed for each column of Y and yi is length(xi)-by-size(Y,2))

yi = interp1(Y, xi) giả sử rằng x = 1:N, ở đó N =length(y) là chiều dài của Y nếu Y là vector, hoặc size(Y,1) nếu Y là ma trận .

yi = interp1(x, Y, xi, method) interpolates using alternative methods:

'nearest'	Nearest neighbor interpolation
'linear'	Linear interpolation (default)
'spline'	Cubic spline interpolation
'pchip'	Piecewise cubic Hermite interpolation
'cubic'	(Same as 'pchip')
'v5cubic'	Cubic interpolation used in MATLAB 5

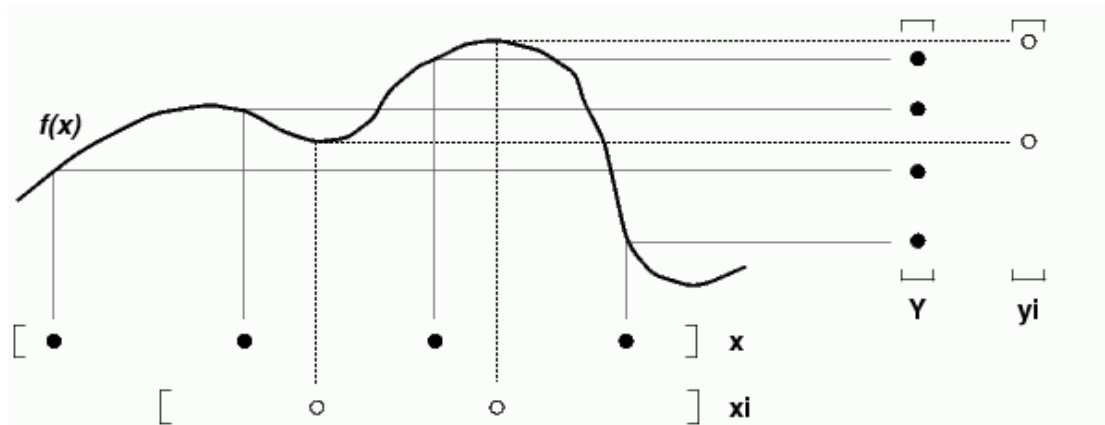
For the 'nearest', 'linear', and 'v5cubic' methods, interp1(x, Y, xi, method) trả về NaN cho tất cả các phần tử của xi mà nằm ngoài khoảng xác định của x. Đối với tất cả các phương pháp, interp1 đề cập đến việc xác định dữ liệu(nội suy cho cả các điểm nằm ngoài vùng của x) nằm ngoài phạm vi biểu diễn

yi = interp1(x, Y, xi, method, 'extrap') uses the specified method to perform extrapolation for out of range values.

yi = interp1(x, Y, xi, method, extrapval) returns the scalar extrapval for out of range values. NaN and 0 are often used for extrapval.

Lệnh interp1 nội suy giữa các điểm. Nó tìm giá trị tại các điểm ở giữa các điểm đã xác định, của hàm một chiều(of a one-dimensional function  $f(x)$ ) hàm này được xác định dưới dữ liệu cho trước ( underlies the data. ) Hàm này được biểu diễn dựa trên quan hệ các cặp véc tor x, Y, xi, Yi



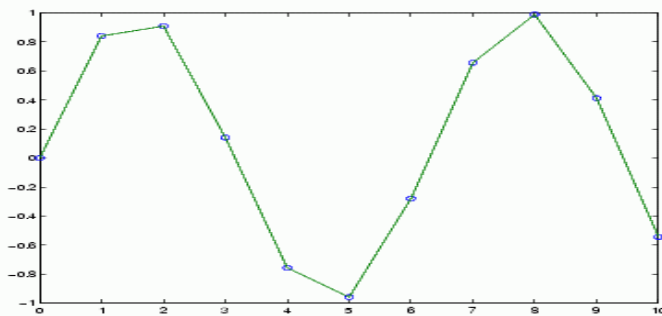


Interpolation is the same operation as *table lookup*. Described in table lookup terms, the *table* is  $[x, Y]$  and `interp1` *looks up* the elements of  $x_i$  in  $x$ , and, based upon their locations, returns values  $y_i$  interpolated within the elements of  $Y$ .

### Examples

**Example 1.** Generate a coarse sine curve and interpolate over a finer abscissa.

```
x = 0:10;
y = sin(x);
xi = 0:.25:10;
yi = interp1(x,y,xi);
plot(x,y,'o',xi,yi)
```



**Example 2.** Here are two vectors representing the census years from 1900 to 1990 and the corresponding United States population in millions of people.

```
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];
```

The expression `interp1(t,p,1975)` interpolates within the census data to estimate the population in 1975. The result is

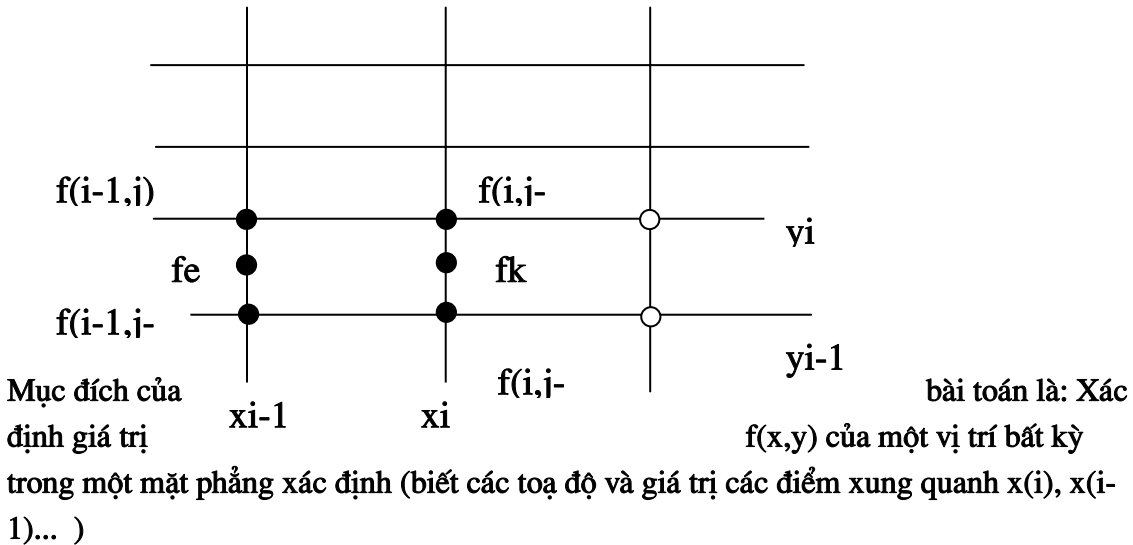
```
ans =
     214.8585
```

```

Ví dụ : >> x=[1 2 3 4];
        >> f=[0.671 0.620 0.567 0.512];
        >> interp1(x,f,1.5)
ans =
        0.6455

```

### 5.1.2 Nội suy cho bài toán hai chiều



Mục đích của định giá trị bài toán là: Xác định giá trị  $f(x,y)$  của một vị trí bất kỳ trong một mặt phẳng xác định (biết các tọa độ và giá trị các điểm xung quanh  $x(i)$ ,  $x(i-1)$ ... )

Muốn xác định giá trị tại một điểm có vị trí  $x_{i-1} < x < x_i$  và  $y_{i-1} < y < y_i$  ta dùng phương pháp nội suy hai chiều, thực chất của phương pháp này là thực hiện hai lần bài toán nội suy một chiều

Bài 1: Nội suy theo phương y tìm ra  $f_e$  và  $f_k$

Bài 2: Nội suy theo phương x từ  $f_e$  đến  $f_k$  để tìm  $g(x,y)$

$f_e =$

$$\frac{y_j - y}{y_j - y_{j-1}} f(i-1, j-1) + \frac{y - y_{j-1}}{y_j - y_{j-1}} f(i-1, j); f_f = \frac{y_j - y}{y_j - y_{j-1}} f(i, j-1) + \frac{y - y_{j-1}}{y_j - y_{j-1}} f(i, j)$$

$$g(x,y) = \frac{x_i - x}{x_i - x_{i-1}} f_e + \frac{x - x_{i-1}}{x_i - x_{i-1}} f_f$$

Chương trình có thể được viết như sau:

Quy ước

Đầu vào là  $f=[f(1) f(2) f(3) f(4)]$  tương ứng với  $[f(i-1,j-1) f(i-1,j) f(i,j-1) f(i,j)]$ ;  $x=[x(i-1) x(i)]$

function  $g = C5(x,y,f,x_a,y_a)$

% trong đó  $x=[x(i-1) x(i)]$   $y=[y(i-1) y(i)]$

%  $x_a$   $y_a$  là tọa độ của điểm cần tìm

```

% f=[f1 f2 f3 f4] là véc tor f tương ứng với f(xy)
i=length(x);
j=length(y);
fe=(1/(y(i)-y(i-1)))*((y(j)-ya)*f(1)+(ya-y(j-1))*f(2));% tính fe
fk=(1/(y(i)-y(i-1)))*((y(j)-ya)*f(3)+(ya-y(j-1))*f(4));% tính fk
g=(1/(x(i)-x(i-1)))*((x(i)-xa)*fe+(xa-x(i-1))*fk);% tính g

```

Thực hiện trong command window như sau

```

>> x=[1 2 ];
>> y=[3 4];
>> f=[5 6 7 8];
>> xa=1.5,ya=3.5;
>> g=C5(x,y,f,xa,ya)
g =
    6.5000

```

Có nhiều cách nội suy tuy nhiên chúng ta chỉ xem xét hai phương pháp trên mà thôi

## 5.2 Giải phương trình phi tuyến

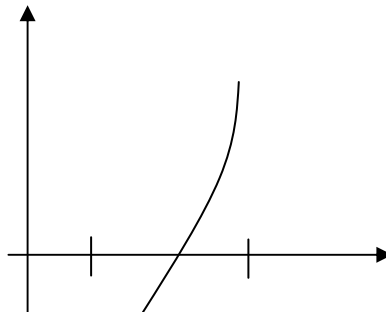
Dùng phương pháp chia đôi để xác định nghiệm của phương trình

Nội dung toán học của phương pháp như sau: xét phương trình  $f(x)=0$

Trên khoảng phân ly nghiệm  $[a b]$ , chia đôi  $[a b]$  bởi  $c=(a+b)/2$

Nếu  $f(c)=0$  thì  $c$  là nghiệm của phương trình, nếu  $f(c) \neq 0$  thì so sánh dấu của  $f(c)$  với  $f(a)$  và  $f(b)$ ,  $f(a)*f(c)<0$  khoảng phân ly nghiệm mới là  $[a c]$ ,  $f(c)*f(b)<0$  thì khoảng phân ly nghiệm là  $[c b]$

Tiếp tục chia đôi các khoảng phân ly nghiệm cho đến khi tìm được giá trị  $cn$  nào đó mà  $f(cn)=0$  thì  $cn$  chính là nghiệm. Tuy nhiên việc tìm chính xác  $cn$  là rất khó khăn người ta chỉ tìm nghiệm gần đúng trong một sai số cho phép là  $tol$



### Đồ thị biểu diễn phương pháp chia đôi

Nếu sai số cho trước thì số bước lặp đòi hỏi là  $(b-a)/2^n < tol$

Suy ra  $n \geq (\ln(b-a)/\text{tol})/0.6931$ ;

Trong đó b và a tương ứng là các khoảng phân ly nghiệm mới

Thuật toán để giải như sau:

```
%-----  
function x= C5(a,b,t)  
% n la so lan lap  
% a la can duoi b la can tren  
i=1;  
if( f(a)*f(b)>0 )  
    disp('nhap lai a va b ');  
end  
while(abs(a-b)>t)  
    c=(a+b)/2;  
    if( f(c)==0)  
        disp('nghiem la x=');  
        x=c;  
        break;  
    end  
    if(f(c)*f(a)<0)  
        b=c;  
    end  
    if(f(c)*f(b)<0)  
        a=c;  
    end  
end  
x=c;
```

Phương pháp Newton

Công thức tính nghiệm của phương pháp Newton là

$X_n = X_{n-1} - f(X_{n-1})/f'(X_{n-1})$

$f=f(x)$  là hàm cần tính nghiệm, chúng ta sẽ tính các giá trị của  $X_n$  đến khi đạt được sai số cần thiết ( tức là  $\text{abs}(X_n - X_{n-1}) < \text{tol}$ ) thì  $X_n$  chính là nghiệm gần đúng của phương trình trên.

```
%-----  
%Thuật toán giải nghiệm gần đúng theo phương pháp Newton  
function[x]=f(t,xb)  
N=input('nhap buoc lap N=');  
tol=1.e-5;
```

```

x=xb;
i=1;
while(i<=N)
    [f,fd]=T1(t,x);
    x=x-(f./fd);
    sol=[i xb x f./fd ]
    if(abs(xb-x)<=tol)
        break;
    end
    xb=x;
    i=i+1;
end
disp('nghiem cuoi cung la')
x
//-----//
function[f,fd]=T(t,x)
f= t^3+t+1;
fd=diff(f,t);
f=subs(f,t,x);
fd=subs(fd,t,x);
//-----Gọi chương trình từ cửa sổ Command Window-----//
>>syms t;
>> xb=4;
>>[x]=f(t,xb)
%-----

```

### 5.3 Tích phân số

#### a.Phương pháp hình thang

$$I = \frac{h}{2} (f_0 + 2f_1 + 2f_2 + \dots + f_N)$$

$$h = \frac{b-a}{N}$$

$$f_0 = f(a), f = f(a + ih)$$

Ví dụ tính tích phân:  $I = \text{int}(f,a,b)$ ;  $f=2*x^2* \cos(x)$

```

%-----
function I= C5(a,b,n)
% a va b la hai can
% n la so buoc tinh

```

```

h=(b-a)/n;
I=0;
for i=0:n
    x=a+h*i;
    c=2;
    if((i==0)|(i==n))
        c=1;
    end
    I=I+c*(2*x^2*cos(x));
end
I=I*h/2;

```

%-----Thuc hien trong command window-----

```
>> I=C5(0,1,20)
```

```
I =
```

```
0.4784
```

Dùng Matlab để tính tích phân hình thang:  $\text{trapz}(x,y)$

Eg1:

```
>> x=[0:0.05 1]';
```

```
>> y=2*x.^2.*cos(x);
```

```
>> trapz(x,y)
```

```
ans =
```

```
0.5403
```

```
>> t=[0:15:90]';
```

```
>> x=t*pi/180;
```

```
>> y=[sin(x) cos(x)];
```

```
>> trapz(x,y)
```

```
ans =
```

```
0.9943 0.9943
```

Để sử dụng công thức trên thì  $x$  là vector cột có cùng chiều dài với vector  $y$ , hoặc  $y$  là một mảng mà các phần tử có chiều dài giống  $x$

Tính theo phương pháp thông thường chuẩn:

```
>> syms x
```

```
>> int(2*x^2*cos(x),0,1)
```

```
ans =
```

```
-2*sin(1)+4*cos(1)
```

```
>> eval(ans)
```

```
ans =
```

```
0.4783
```

Kết luận rằng : phương pháp hình thang giải theo trapz thì độ chính xác kém hơn:

b. Phương pháp Simpson 1/3

$$I = \frac{h}{3} (f_0 + 4f_1 + 2f_2 + 4f_3 + \dots + 2f_{N-2} + 4f_{N-1} + f_N)$$

$H=(b-a)/N;$

$$f_0 = f(a), f_1 = f(a + i * h)$$

%----- Chương trình viết theo Simpson-----

```
function I= C5(a,b,n)
```

```
% a va b la hai can
```

```
% n la so buoc tinh
```

```
h=(b-a)/n;
```

```
I=0;
```

```
for i=0:n
```

```
    x=a+h*i;
```

```
    c=4;
```

```
    if((i==0)|(i==n))
```

```
        c=1;
```

```
    end
```

```
    if(mod(i,2)==0)
```

```
        c=2;
```

```
    end
```

```
    I=I+c*(2*x^2*cos(x));
```

```
end
```

```
I=I*h/3;
```

Cách giải Dùng matlab( for Simpson)

#### 5.4 Dùng Laplace để giải bài toán trong Lý thuyết Mạch

Trong Lý thuyết mạch có rất nhiều các đại lượng đạo hàm ,các đại lượng đó có thể được biến đổi qua Laplace và thay thế bài toán lý thuyết mạch về bài toán giải bằng Laplace.

% Các ví dụ

ví dụ 1

```
syms t s;
```

```
I1= sym('I1(t)');
```

```
k=laplace(I1,t,s); % Chuyển đổi I1(t) sang Laplace
```

```

syms t s;
I1=sym('I1(t)');
laplace(i,t,s)
dI1=sym('diff(I1(t),t)')
l=laplace(dI1,t,s) % chuyển đạo hàm I1(t) sang Laplace

```

Các lệnh phụ trợ cần chú ý để giải một bài toán ký thuyết mạch

1. Lệnh collect( f , x ) : là lệnh nhóm thừa số chung theo biến

Ví dụ  $f = 2*x + 3*x$ ;

```
>>f= collect(f,x)
```

```
f=
5*x
```

2. Lệnh thay thế subs( f, { x,y,z }, { 1,2,3 }) thay thế x , y , z bằng 1 2 3

```
>> syms x;
```

```
>> syms R1 R2 R3;
```

```
>> f= R1+R2 + R3*x;
```

```
>> subs(f,{R1,R2,R3},{1,2,3})
```

```
ans =
3+3*x
```

3. Trong khi giải phương trình :

Chúng ta thay thế phân tử laplace(I1(t),t,s) bằng LI1 như sau

```
>> syms t s;
```

```
>> sym(' diff( I1(t),t)');
```

```
>> l=sym(' diff( I1(t),t)');
```

```
>> l=laplace(l,t,s)
```

```
l =
```

```
s*laplace(I1(t),t,s)-I1(0)
```

(chú ý khi thay I1(0) bằng giá trị nào đó thì ta phải gán như sau

ví dụ subs(1, { 'I1(0)', 'laplace(I1(t),t,s)' }, {2, LI1})

kết quả sẽ được như sau: l= s\* LI1 -2

4. Sau khi giải ra nghiệm dòng , áp theo laplace thì ta chuyển đổi ngược lại dùng hàm

biến đổi ngược laplace( hàm ngược là illaplace)

Ví dụ cụ thể

Cho mạch điện có các phương trình như sau:  $(dI1/dt)*R1 + R2 = I1*R3$

% giải hệ phương trình trên bằng cách biến đổi sang laplace



```

%chương trình viết trong M-file và được ghi trong file C5.m
syms R1 R2 R3 real;
I1=sym('I1(t)');
dI1=sym('diff(I1(t),t)');
eq1= dI1*R1 +R2-I1*R3;
syms t s ;
q1=laplace(eq1,t,s)
syms I1p;
q2=subs(q1,{R1,R2,R3,'I1(0)', 'laplace(I1(t),t,s)'},{1,2,3,2,I1p})
q2=collect(q2,I1p);% nhóm lại thừa số chung là I1p
I1p=solve(q2,I1p)% Giải phương trình trên với biến I1p
ilaplace(I1p)% biến đổi ngược lại sang I1(t)

```

Kết quả khi thực hiện chương trình trên là:

```

>>C5
q1 =
R1*(s*laplace(I1(t),t,s)-I1(0))+R2/s-R3*laplace(I1(t),t,s)
q2 =
s*I1p-2+2/s-3*I1p
I1p =
2*(s-1)/s/(s-3)
% kết quả I1(t)
ans=
2/3+4/3*exp(3*t) % kết quả I1(t)

```

Sau đây là một số bài tập để giải.

### 5.5 Giải hệ phương trình đại số tuyến tính

Phần này đã trình bày ở chương II 'Thư viện toán học Symbolic'

Muốn giải trước hết hàm phải là hàm symbolic của một hoặc nhiều biến nào đó

```

>>syms x y;
>> [x,y]=solve('x+y=1','x-11*y=5',x,y)
x =
    4/3
y =
   -1/3
> syms x y;
>> n=solve('x+y=1','x-11*y=5',x,y) % kết quả dạng cấu trúc
n =

```

```

x: [1x1 sym]
y: [1x1 sym]
>> n.x % truy nhập cấu trúc biến x
ans =
4/3
>> n.y % Truy nhập cấu trúc biến y
ans =
-1/3

```

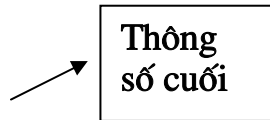
### 5.6 Phương trình vi phân thường

DSOLVE Symbolic tìm nghiệm của phương trình vi phân DSOLVE('eqn1','eqn2', ...) chỉ chấp nhận các biểu thức vi phân dạng symbolic ('eq1'....) và điều kiện đầu. Một số phương trình hoặc các điều kiện đầu có thể được nhóm lại với nhau và cách nhau bằng dấu phẩy(comma), đối với một thông số đầu vào, mặc định là biến 't' biến độc lập này có thể được thay đổi từ 't' đến các biến symbolic khác bằng cách thêm biến đó như là thông số đầu vào cuối cùng

Ví dụ như sau: giả sử ta cần giải phương trình vi phân  $dy/dx = x*y$  biến lấy tích phân(phải là) x cho nên ta coi x là thông số đầu vào cuối cùng ta viết như sau

```
syms x
```

```
y=dsolve('Dy=x*y','Dy(0)=1','x');
```



ký hiệu 'D' định nghĩa phương trình vi phân tương ứng với biến độc lập ví dụ thông thường sử dụng  $dy/dt$ . "D" được theo sau bởi một số, thì số đó định nghĩa bậc vi phân ví dụ  $D^2y$  nghĩa là  $d^2y/dt^2$  ví dụ sau:

```
y = dsolve('D2y+y=1','y(0) = 0')
```

kết quả:  $y = 1 + C1 * \sin(t) - \cos(t)$

Còn  $D^3y$  tức là  $d^3y/dt^3$

chú ý rằng biến symbolic không được chứa trong D ví dụ như không thể ghi như sau :

```
syms y; dsolve('Dy') (sai)
```

Điều kiện đầu xác định bởi biểu thức 'y(a)=b' hoặc 'Dy(a)=b'

ở đó y là một trong những biến phụ thuộc và a và b là số không đổi nếu số điều kiện đầu nhỏ hơn số biến phụ thuộc thì Kết quả sẽ được cho trong mảng C1,C2

Có ba kiểu đầu ra. Đối với một phương trình vi phân thì có một đầu ra, đối với hệ có nhiều phương trình vi phân thì có số đầu ra tương ứng (đầu ra có thể là một structer)

Examples:

```
dsolve('Dx = -a*x') returns
```

```
ans = exp(-a*t)*C1
```

```
x = dsolve('Dx = -a*x','x(0) = 1','s') returns
```

```

x = exp(-a*s)
y = dsolve('(Dy)^2 + y^2 = 1','y(0) = 0') returns
y =
[ sin(t)
[ -sin(t)
S = dsolve('Df = f + g','Dg = -f + g','f(0) = 1','g(0) = 2')
returns a structure S with fields
S.f = exp(t)*cos(t)+2*exp(t)*sin(t)
S.g = -exp(t)*sin(t)+2*exp(t)*cos(t)
Y = dsolve('Dy = y^2*(1-y)')
Warning: Explicit solution could not be found; implicit solution returned.
Y =
t+1/y-log(y)+log(-1+y)+C1=0
dsolve('Df = f + sin(t)', 'f(pi/2) = 0')
dsolve('D2y = -a^2*y', 'y(0) = 1, Dy(pi/a) = 0')
S = dsolve('Dx = y', 'Dy = -x', 'x(0)=0', 'y(0)=1')
S = dsolve('Du=v, Dv=w, Dw=-u','u(0)=0, v(0)=0, w(0)=1')
w = dsolve('D3w = -w','w(0)=1, Dw(0)=0, D2w(0)=0')
y = dsolve('D2y = sin(y)'); pretty(y)

```

Sử dụng ode23 và ode45 dùng để giải phương trình vi phân thường

### Cấu trúc

$[T, Y] = \text{ODE23}(\text{ODEFUN}, \text{TSPAN}, \text{Y0})$  với  $\text{TSPAN} = [\text{T0} \text{TFINAL}]$  tổ hợp hệ phương trình vi phân  $y' = f(t, y)$  từ thời gian  $\text{T0}$  đến  $\text{TFINAL}$  với giá trị ban đầu  $\text{Y0}$  (with initial conditions  $\text{Y0}$ ). Hàm  $\text{ODEFUN}(T, Y)$  chắc chắn trả về một véc tor cột tương ứng với  $f(t, y)$ . Mỗi hàng trong mảng kết quả  $Y$  tương ứng thời điểm(t) trả về trong column vector  $T$

Để lấy kết quả tại các thời điểm  $\text{T0}, \text{T1}, \dots, \text{TFINAL}$  (tất cả là tăng đều hoặc giảm đều) sử dụng  $\text{TSPAN} = [\text{T0} \text{T1} \dots \text{TFINAL}]$ .

$[T, Y] = \text{ODE23}(\text{ODEFUN}, \text{TSPAN}, \text{Y0}, \text{OPTIONS})$  solves as above with default integration properties replaced by values in  $\text{OPTIONS}$ , an argument created with the  $\text{ODESET}$  function. See  $\text{ODESET}$  for details.

Thông thường sử dụng options là một số vô hướng để nói về sai số liên quan ('RelTol') Nếu không có thông số trên thì mặc định sai số liên quan là  $=1 \cdot e^{-3}$  và sai số tuyệt đối mặc định với tất cả các phân tử là  $1 \cdot e^{-6}$

### Example

```
[t,y]=ode23(@vdp1,[0 20],[2 0]);
```

```

    plot(t,y(:,1));
    solves the system  $y' = \text{vdp1}(t,y)$ , using the default relative error
    tolerance  $1e-3$  and the default absolute tolerance of  $1e-6$  for each
    component, and plots the first component of the solution.
% giai phuong trinh vi phan bac hai sau
%  $L \cdot d^2q/dt^2 + R \cdot dq/dt + q/c = E_0 \cdot \cos(w \cdot t)$ 
% nguyen tac giai
global R L C Eo omega
L=100;
R=100;
C=0.25;
Eo=10;
omega=1;
%t0=0;
%ta=3;
%x0=[1 1]'
tol=1e-3;
[t,x]=ode23(@Mach1,[0 3],[1 1],tol);
kq=[t x(:,1) x(:,2)]
plot(t,x(:,1));
%----- Ham Mach1-----
function f= Mach1(t,x)
global R L C omega Eo
f=[(Eo/L)*cos(omega*t)-x(1)/(C*L)-R*x(2)/L x(2)]';
%-----ket qua thuc hien trong command window-----
kq =
    0  1.0000  1.0000
    0.0800  0.9216  1.0833
    0.3585  0.5926  1.4308
    0.5589  0.2895  1.7484
    0.7093  0.0171  2.0319
    0.8596 -0.3011  2.3615
    1.0069 -0.6642  2.7362
    1.1900 -1.1987  3.2858
    1.4006 -1.9498  4.0557
    1.6323 -2.9833  5.1132
    1.8804 -4.3902  6.5518

```

2.1408 -6.2933 8.4990  
2.4104 -8.8576 11.1269  
2.6868 -12.3044 14.6669  
2.9682 -16.9303 19.4292  
3.0000 -17.5398 20.0576

Nguyên tắc giải bài toán : Đưa phương trình vi phân cấp n về n phương trình vi phân cấp một trong ví dụ trên ta đặt  $x_1=q$  , $x_2=diff(x_1)$  như vậy ta có hai phương trình vi phân( giống như phương pháp đặt biến trạng thái trong lý thuyết điều khiển tự động )

## CHƯƠNG 5

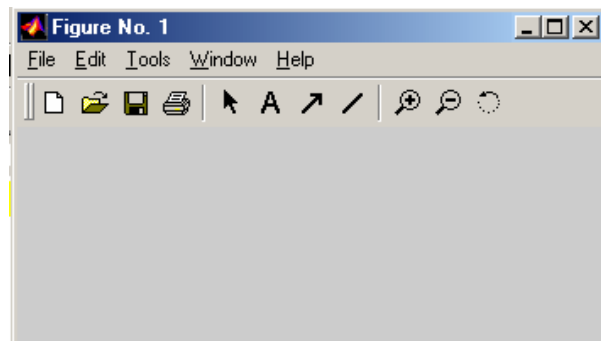
### ĐỒ HOẠ TRONG MATLAB

#### 5.1 MÀN HÌNH ĐỒ THỊ:

Đây là nơi trình bày mọi hình ảnh , đồ thị .... đã được giải trình từ khung của sổ Command của Matlab để xử lý theo các lệnh , công cụ màn hình.

Có hai cách để hiện khung màn hình đồ thị trắng:

- Từ khung cửa sổ command kích File/New và chọn Figure từ menu xổ.
- Cũng trong khung cửa sổ Command gõ lệnh figure và ấn enter.



#### 5.2 CÁC LỆNH MENU ĐỒ HOẠ TRONG MATLAB:

Để giúp các bạn có thể nắm vững và sử dụng chương trình Matlab; phần này giới thiệu các lệnh trên thanh menu cùng với các chức năng và công dụng của từng menu con nằm trong các menu chính.

##### 5.2.1 File:

Hiện menu xổ chứa các lệnh con có chức năng tạo, quản lý, điều hành cũng như thay đổi các thông tin các thông số mặc định của chương trình cho phù hợp với từng công việc.

##### New Figure

New Figure dùng để mở trang màn hình đồ họa mới.

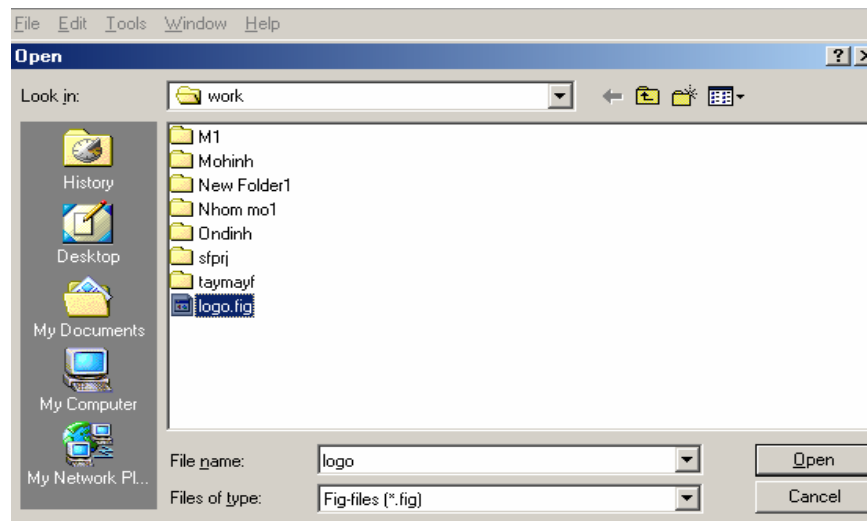
Để mở trang màn hình mới chồng lên màn hình đồ thị cũ trong khi đồ thị cũ vẫn còn hiện diện trên màn hình, chọn New Figure từ menu xổ. Một cửa sổ mới sẽ xuất hiện ra nằm chồng lên màn hình cũ

### Open:

Mở tập tin đồ thị cũ trong khung màn hình đồ thị để xử lý theo nhu cầu công việc.

Các bước thực hiện mở đồ thị đã lưu:

- **K**ích File/ Open từ menu xổ, xuất hiện hộp thoại Open .



- **Look**  
Nơi

in:  
chứa

các tập tin đồ thị của Matlab. Nơi chứa có thể là ổ đĩa, thư mục hoặc chương trình khác.

- **File name:** Tên tập tin muốn mở trong khung màn hình đồ thị .

- **Files of type:** Thể loại tập tin đồ thị là **.fig**

- **K**ích **O**p từ menu xổ, xuất hiện hộp thoại Open. Nhập tên tập tin vào khung File name hoặc kích một lần vào tên tập tin, kích Open. Tập tin đã mở sẽ hiển thị trên màn hình.

### Close:

Đóng khung màn hình đồ thị để về khung cửa sổ nhập lệnh của Matlab ( Biểu tượng có chức năng tương đương với lệnh Close trong menu File)

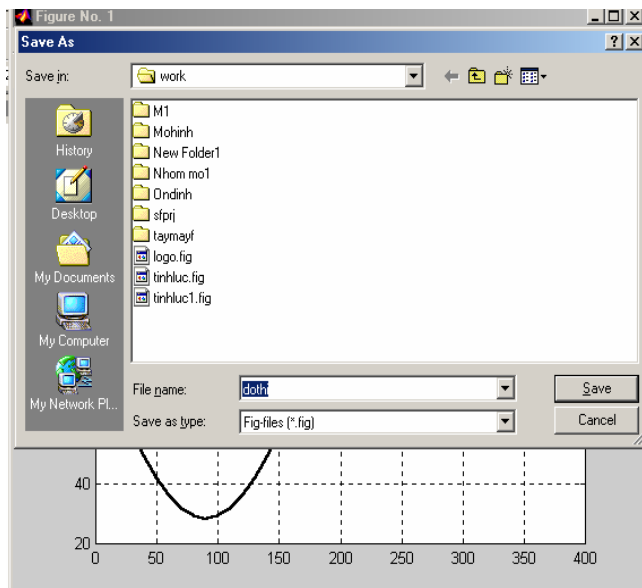
### Save:

Lưu lại những thay đổi trong khung màn hình đồ thị hiện hành. Tuy nhiên, có một điều khác biệt là lệnh này lưu lại ngay những thay đổi trong tập tin mới sau khi đã được đặt tên và đang hiện diện trên màn hình để tiếp tục xử lý.

Nếu bạn mở tập tin cũ với lệnh Open để xử lý và nếu đã có những thay đổi bất kỳ trong nội dung hiện hành và sau khi kích lệnh save, màn hình hiện khung thoại save as. Từ khung thoại này bạn có thể lưu lại những thay đổi theo tên tập tin cũ hoặc với một tên mới.

**Save As:** Hiện khung thoại Save As để bạn lưu tập tin đồ thị mới vẽ theo một tập tin mới hoặc lưu lại những thay đổi trong nội dung của tập tin cũ được mở với lệnh Open theo tên cũ hoặc với tên mới. Các bước thực hiện như sau:

- Sau khi thay đổi, kích File / Save as
- **Save in** Nơi chứa các tập tin muốn. Lưu. Nơi chứa các tập tin đồ thị của Matlab. Nơi chứa có thể là ổ đĩa, thư mục hoặc chương trình khác.
- **File name** Tên tập tin tùy chọn để lưu cho đồ thị vừa tạo
- **Save as type** Thể loại tập tin muốn lưu. Mặc định là **.fig** đối với các tập tin đồ thị

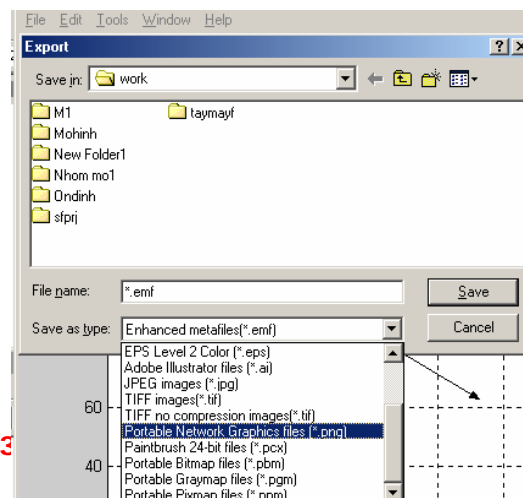


- Sau khi chọn nguồn chứa (nếu cần thiết), đặt tên mới cho đồ thị, kích vào **Save** để lưu.

### Export:

Lưu lại tập tin đồ thị hiện hành thành một dạng tập tin khác để sau này có thể chuyển sang chương trình ứng dụng khác.

- Tạo một đồ thị mới hoặc mở tập tin đồ thị cũ lên màn hình.
- Kích menu File và chọn Export. Màn hình hiện khung thoại Export.





**Save in** Nơi chứa các tập tin muốn chuyển.

**File name** Tên tập tin muốn lưu lại để chuyển. Bạn có thể đặt tên theo tên cũ nhưng phân mở rộng lại là một tên khác .

**Save as type** Chọn loại tập tin muốn lưu lại để chuyển.

- Sau khi chọn xong, kích vào Save để ghi lại tập tin theo dạng khác.

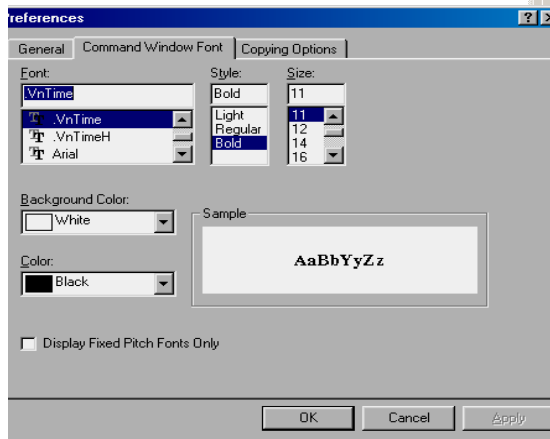
### Property Editor:

Hiện khung thoại Graphics Property để người sử dụng thay đổi các khung thuộc tính mặc định cho phù hợp với tác vụ.

**Preferences:** Hiện khung thoại với ba tùy chọn để người sử dụng có thể thay đổi tham số cho phù hợp nhiệm vụ.

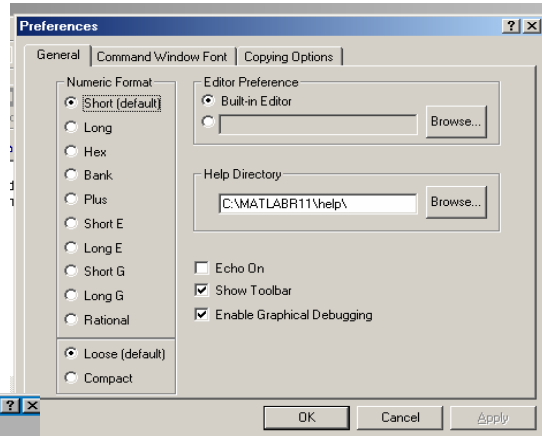
- **General:** Hiện khung thoại ngay khi kích chọn lệnh Preferences từ menu File của khung cửa sổ lệnh MATLAB.

K



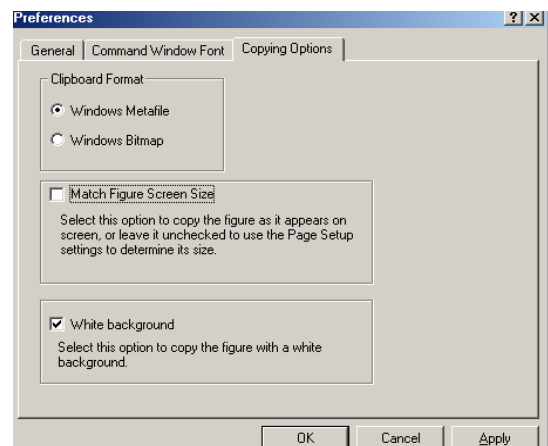
n

khung danh mục font cùng thuộc tính để người sử dụng thay đổi font mặc định thành font quen thuộc.



ch chọn các loại tham số muốn thay đổi hoặc gán thêm sau đó kích OK.

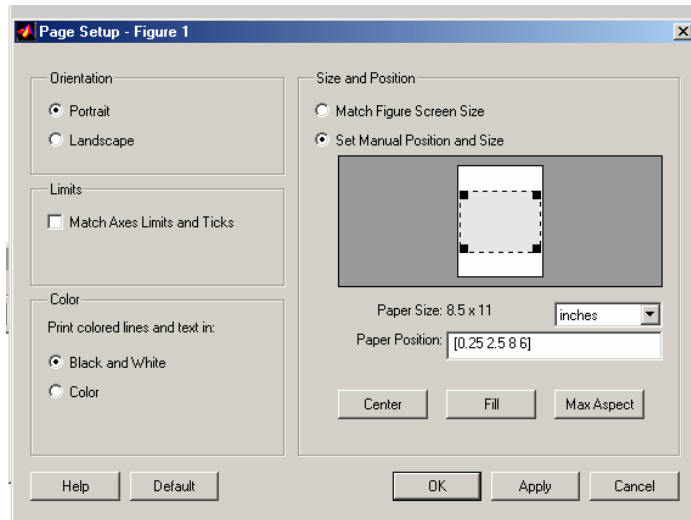
- **Command Windows Font:**



- **Copy Options** : Hiện khung thông tin để người sử dụng tùy chọn những thành phần đối tượng muốn sao chép.

**Page Setup:** Hiện khung thoại Page Setup để định lại trang in cho phù hợp với đồ thị khi vẽ hoặc in.

- **Orientation:** Chọn hướng giấy in dọc hoặc in ngang.
- **Limits:** Đánh dấu chọn để áp dụng hoặc không áp dụng chế độ vừa khớp với các trục và giá trị theo thanh đo trên trục.
- **Color:** Chọn chế độ màu hoặc đen trắng khi in hoặc vẽ trên thiết bị.
- **Size and Position:** Có hai tham số tùy chọn, sử dụng theo mặc định hoặc chỉnh theo tùy chọn.
  - **Center** In đồ thị ở giữa trang
  - **Fill** In đồ thị trọn cả trang
  - **MaxAspect** In đồ thị theo tỷ lệ tối đa

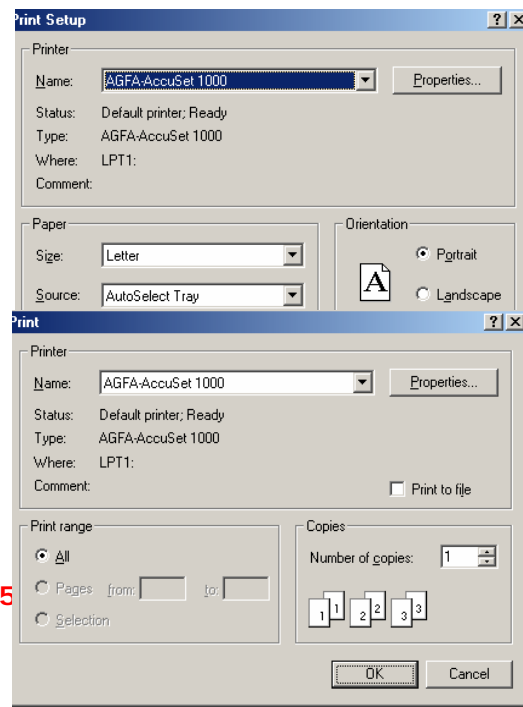


### Print Setup:

Nếu có nhiều loại thiết bị in( vẽ ), thì nên khai báo lại trước khi chuyển bản đồ thị sang máy in hoặc máy vẽ.

Các bước thực hiện vẽ một đồ thị:

- Kích File / Open.
- Kích File / Print Setup, màn hình hiện khung thoại Print Setup.



- Kích chọn loại máy in, khổ giấy.... Theo ý muốn.
- Kích Ok để in.

**Print Preview:** Xen lại đồ thị hiện hành trước khi chuyển sang in


**Print:**

Hiện khung thoại print để người sử dụng chọn loại máy in hoặc máy vẽ cho phù hợp; chọn số lượng bản in với Copies. Nếu cần thiết kích Properties để chọn thêm các thông tin bổ sung cần thiết.


**5.2.2 Edit:** Hiện menu xổ chứa các lệnh con liên quan đến việc xử lý các đối tượng cũng như các thông số đã gán cho đối tượng .

**Undo:** Huỷ lệnh hoặc chức năng đã được thi hành trước đó. Sau khi kích lệnh Undo để thực hiện việc huỷlệnh, Undo sẽ chuyển đổi thành Redo để có thể phục hồi những chức năng đã huỷ.

**Cut ( Ctrl + x):** Cắt đối tượng đã chọn trong bản đồ thị hiện hành đưa vào Cliboard để sau đó dán vào đồ thị khác. Sau khi cắt đối tượng đã chọn sẽ biến mất tại vị trí hiện hành. Các bước thực hiện như sau:

- Kích vào biểu tượng Enable Plote Editing trên thanh công cụ. 
- Kích chọn đối tượng muốn cắt.
- Kích Edit / Cut. Đối tượng sẽ biến mất tại vị trí hiện hành và lưu vào Cliboard.

**Copy ( Ctrl + c):** Sao chép đối tượng đã chọn trong đồ thị hiện hành đưa vào Cliboard để sau đó dán vào đồ thị khác. Đối tượng sau khi sao chép vẫn hiện diện tại vị trí cũ. Các bước thực hiện như sau:


- Kích vào biểu tượng Enable Plote Editing trên thanh công cụ. 
- Kích chọn đối tượng muốn cắt.
- Kích Edit / Copy. Đối tượng sẽ vẫn hiện tại vị trí hiện hành và lưu vào Cliboard.

**Paste ( Ctrl + v):** Dán đối tượng đã được Cut hoặc Copy vào Cliboard sang đồ thị khác. Các bước thực hiện như sau:

- Mở đồ thị muốn được dán đối tượng đã được lưu vào Cliboard.
- Ấn Ctrl + v hoặc kích vào Paste. Đối tượng sẽ được dán vào đồ thị mới hoặc vị trí mới của đồ thị cũ.

- Để di chuyển đối tượng đến vị trí mới ta kích vào đối tượng muốn di chuyển. Khi con trỏ đổi thành mũi tên bốn đầu thì kéo, giữ đối tượng đến vị trí mới rồi thả nút chuột.

**Clear:** Xoá đối tượng được chọn trong đồ thị.

- Kích vào biểu tượng Enable Plot Editing trên thanh công cụ. 
- Kích chọn đối tượng muốn xoá.
- Kích Edit / Clear. Đối tượng sẽ xoá mất tại vị trí hiện hành.

**Select All( Ctrl + a):** Chọn tất cả các đối tượng trong đồ thị hiện hành để xử lý theo nhu cầu.

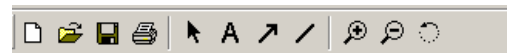
**Copy Figure:** Sao chép cả đồ thị sang Clipboard để rồi sau đó dán sang trang màn hình đồ thị hoặc sang chương trình khác. Các bước thực hiện như sau:

- Mở đồ thị muốn được sao chép sang đối tượng khác.
- Kích Edit / Copy Figure. Đồ thị sẽ được ghi vào Clipboard.
- Thoát khỏi Matlab, mở chương trình cần Copy đồ thị đến.
- Kích chọn vị trí muốn dán đồ thị đến, sau đó kích Paste. Đồ thị sẽ được dán ngay vị trí đã chọn.

**Copy Option:**


Hiện khung thoại Preferences để người sử dụng thay đổi (Xem File / Preferences).

**5.2.3 Tools ( Ctrl + t):** Chứa các lệnh công cụ hỗ trợ việc thực hiện, xử lý các thuộc tính minh họa trong đồ thị để người quan sát dễ hiểu hơn.

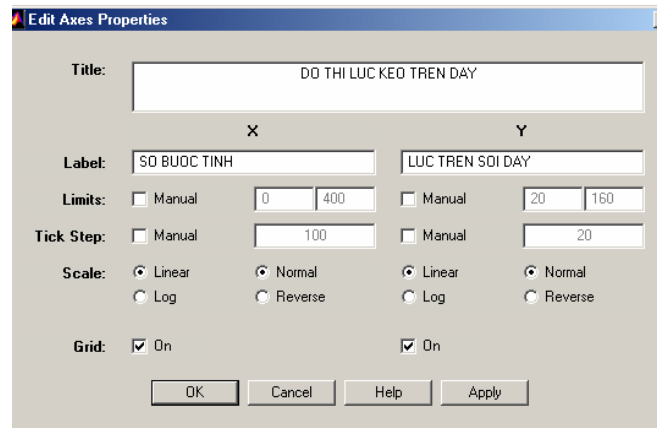


**Show Toolbar:**

Cho hiện hoặc giấu các biểu tượng công cụ Standard trên màn hình

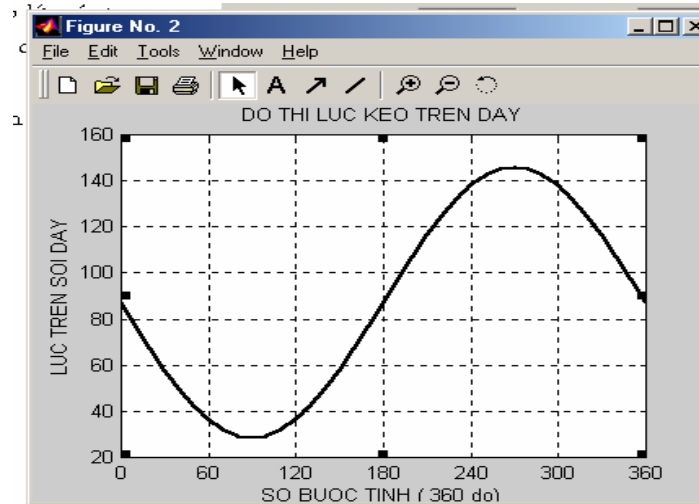
**Enable Plot Editing:** Đánh dấu chọn hoặc bỏ chọn đối tượng. 

**Axes Properties:** Gán thuộc tính cho các trục tọa độ cùng với tiêu đề đồ thị hiện hành. Thí dụ với đồ thị lực kéo sợi dây khi quay tròn qua cầu. Thay vì dùng các thuộc tính



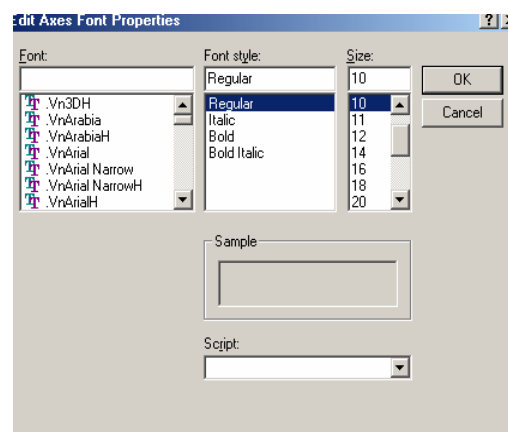
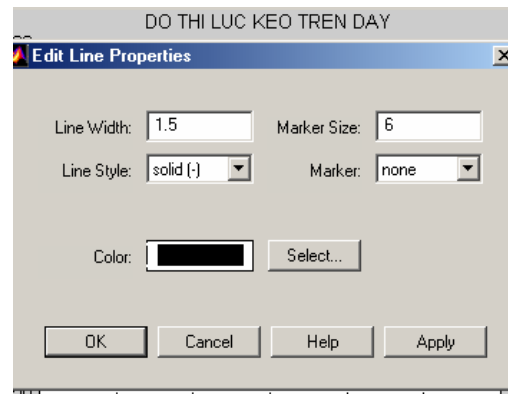
lệnh để gán chung với số liệu, ta có thể dùng khung thoại Axes Properties để gán thuộc tính. Các bước như sau:

- Sau khi vẽ đồ thị , kích chọn **Tool / Axes Properties**. Màn hình xuất hiện khung thoại Axes Properties.
- Trong khung thoại Axes Properties có thể: đặt tiêu đề cho đồ thị ( Title), gán nhãn cho trục x và trục y ( Label), giới hạn độ lớn các trục ( Limits), tỷ lệ xích các trục..... cho đồ thị.
- Kích Apply và kích OK.



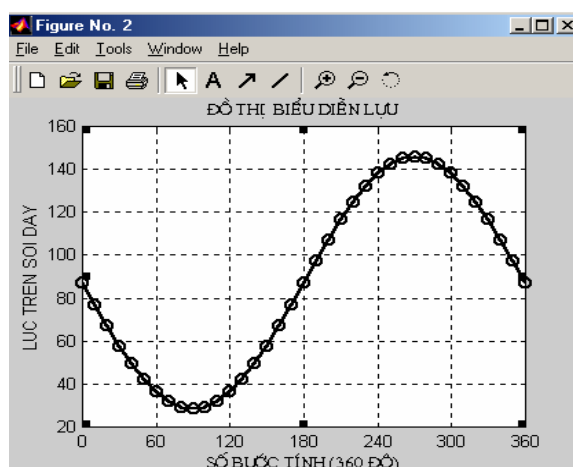
**Line Properties:** Gán thuộc tính như các điểm đỉnh dữ liệu, tạo nét đồ thị dày hay mỏng cũng như màu sắc cho đường đồ thị hiện hành. Các bước thực hiện như sau:

- Kích **Enable Plot Editing** và kích chọn đường đồ thị. Những điểm đỉnh dữ liệu trên đường đồ thị sẽ được đánh dấu bằng những hình vuông nhỏ.
- Kích **Tool / Line Properties**. Màn hình hiện khung thoại line properties.
- Chọn độ dày mỏng của đường đồ thị **Line Width**; kích cỡ dấu đỉnh dữ liệu trên đồ thị **Marker Size**; Kiểu dáng đồ thị **Line Style**; màu sắc **Color** .....
- Kích Apply và kích OK.



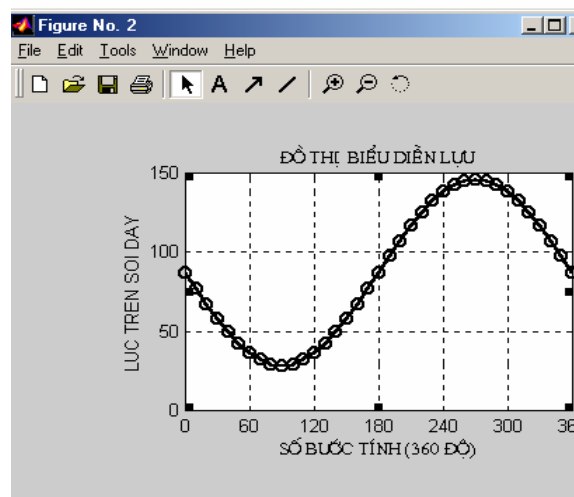
**Text Properties:** Thay đổi các thuộc tính cho các ký tự trong đồ thị. Các bước thực hiện như sau:

- Mở đồ thị muốn thay đổi thuộc tính ký tự.
- Kích menu **Tool / Axes Properties**. Xoá các ký tự cần thay đổi trước đó. Nhập lại các ký tự theo cách gõ tiếng việt, đừng quan tâm đến các ký tự hiện ra có đúng không. Kích **Apply** và kích **OK**.
- Kích biểu tượng **Enable Plot Editing**, chọn chuỗi ký tự muốn thay đổi thuộc tính.
- Kích menu **Tool / Text Properties**. Màn hình xuất hiện khung thoại **Edit Font Properties**.
- Trong khung thoại, chọn loại font chữ Vn Time; kích chọn các thuộc tính cho font chữ (kích thước, kiểu dáng).
- Kích **Ok**.



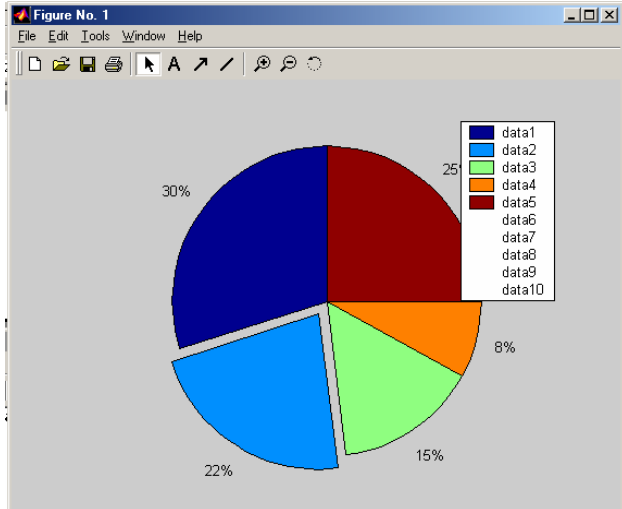
**Unlock Axes Position:** Chỉnh lại tỷ lệ trục tọa độ hoặc di chuyển đồ thị đến vị trí khác trong khung cửa sổ đồ họa.

- Kích chọn biểu tượng **Enable Plot Editing**.
- Chọn trục tọa độ x (y) nếu là đồ thị 2D hoặc trục z nếu đồ thị là 3D.
- Kích menu **Tool / Unlock Position**.
- Muốn chỉnh tỷ lệ cho trục nào, kích chuột vào điểm chỉnh trong bốn góc của đồ thị và kéo vào trong hay ra ngoài theo hướng tâm. Nếu muốn di chuyển đồ thị đến vị trí khác, kích chuột vào điểm chỉnh nằm trên một cạnh và kéo đến vị trí khác.
- Sau khi thực hiện xong, kích lại menu **Tool**. Lần này lệnh **Unlock.....** đổi thành **Lock Position**. Kích **Lock Position** để khoá lại.




**Show Legend:** Cho hiện hoặc giấu bảng phân loại số liệu của đồ thị trên màn hình

- Mở đồ thị muốn hiện bảng phân loại dữ liệu trên màn hình.
- Sau khi đồ thị hiện trong khung màn hình đồ họa, kích **Tool / Show Legend**. Ngay cạnh đồ thị sẽ xuất hiện khung phân loại số liệu của đồ thị cùng với mỗi loại mang một màu khác nhau.

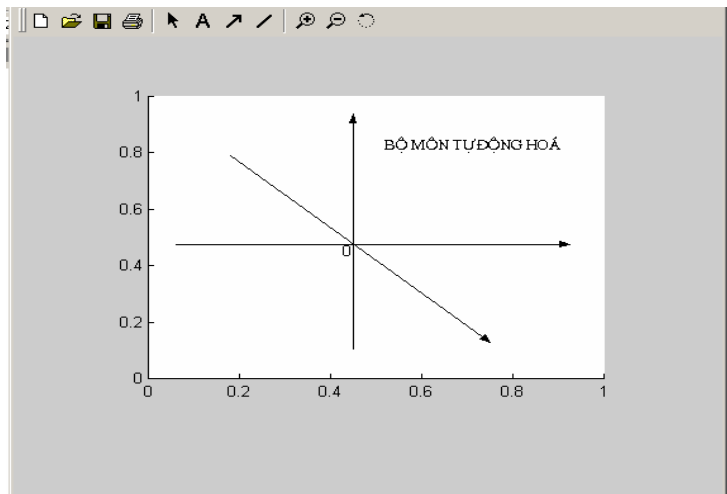


- Để giấu phân loại dữ liệu đồ thị, kích Tool / Hide legend

**Add:** Hiện bốn thanh công cụ hỗ trợ cho việc thêm các thuộc tính vào đồ thị:

- **Axes:** Vẽ trục tọa độ mới trong khung màn hình đồ họa.
  - Kích **File / New / Figure**.
  - Kích **Tool / Add / Axes**. Con trỏ mouse đổi thành hình chữ thập.
  - Kích chọn vị trí đầu tiên, kích và kéo mouse. Thả mouse sau khi có kích thước của các trục tọa độ đã định.
- **Line:** Thêm đường kẻ vào đồ thị. Các bước thực hiện như sau:
  - Mở đồ thị muốn thêm đường kẻ.
  - Kích **Tool / Add / Line** hoặc kích biểu tượng  Trên thanh công cụ.

- Kích chọn điểm đầu tiên, kéo mouse đến vị trí thứ hai. Kích lại mouse để định vị.
- Để gán thuộc tính cho đường kẻ, kích biểu tượng **Enable**




Plot Editing trên thanh công cụ và kích đúp vào đường vẽ đó. Màn hình hiện khung thoại Edit line Properties.



- Khai báo tham số cho từng loại thuộc tính ( giống như trong Line Properties).

Kích Apply và OK.


- **Arrow:** Thêm các mũi tên vào đường đồ thị tại những vị trí bất kỳ.

- Mở đồ thị cần thêm mũi tên.
- Kích **Tool / Add / Arrow**, hoặc kích biểu tượng  trên thanh công cụ.
- Kích chọn vị trí để thêm mũi tên, kéo mouse để lấy độ dài cho mũi tên.
- Có thể di chuyển mũi tên đến vị trí bất kỳ bằng cách kích mouse vào mũi tên rồi kéo đến vị trí đã định vị.


- **Text:** Thêm các ký tự vào đồ thị. Các bước thực hiện như sau:

- Mở đồ thị cần thêm ký tự.
- Kích **Tool / Add / Text** hoặc kích biểu tượng  trên thanh công cụ.
- Kích chọn vị trí cần thêm ký tự, nhập ký tự từ bàn phím.
- Kích vào  rồi kích đúp vào chuỗi ký tự để thay đổi font chữ và thuộc tính cho phù hợp.
- Để di chuyển, kích vào chuỗi ký tự rồi kéo.
- Để sửa đổi nội dung kích đúp vào chuỗi ký tự rồi thực hiện như trong word.

**Zoom In:** Phóng to đồ thị để quan sát từng chi tiết.

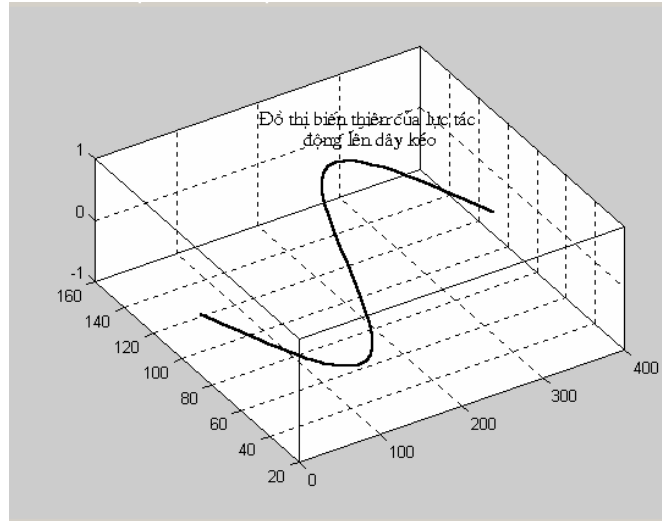
- Kích vào **Zoom In** hoặc kích vào biểu tượng  Trên thanh công cụ.

**Zoom out:** Thu nhỏ đồ thị theo yêu cầu.

- Kích vào **Zoom out** hoặc kích vào biểu tượng  Trên thanh công cụ.



## Rotate 3D: Xoay đối tượng 2D thành 3D



## 5.3 THỰC HÀNH VẼ CÁC LOẠI ĐỒ THỊ 2- D

### 5.3.1 Đồ thị tuyến tính:

Đồ thị tuyến tính là loại đồ thị 2-D dùng các đoạn thẳng nối các điểm dữ liệu lại với nhau để tạo thành một biểu đồ liên tục.

- lệnh vẽ Plot:

>> Plot ( tên biến , tên hàm)

VD 1: vẽ hàm  $y = \sin(x)$

>> x = 0 : 0.1 : 10 ;

% Tạo vector x từ 0 → 10 với bước 0.1.

>> y = sin(x); % Nhập hàm.

>> plot (x,y) % Vẽ hàm y theo biến x.

>>grid on % Tạo chia ô cho đồ thị.

VD 2: vẽ đồ thị  $y = ax + bx$  với  $a = \sin$ ,  $b = \cos$

x biến thiên từ 0 đến  $2\pi$ .

>> x = 0: pi/100: 2\*pi;

>> y = sin(x)+cos(x);

>> plot(x,y)

>>grid on

VD 3: Tạo biến từ hàm linspace :

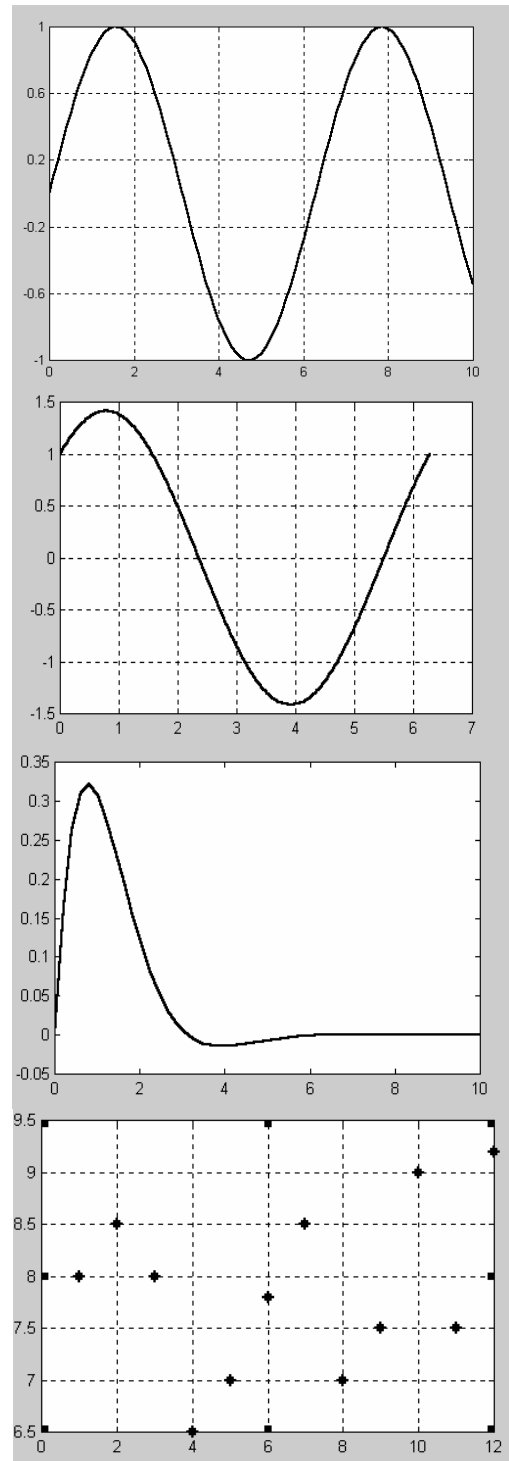
Tên biến = linspace ( Điểm đầu, điểm cuối, số điểm cần vẽ )

% vẽ hàm  $y = e^{-x} \cdot \sin(x)$  với x chạy từ 0 → 50 với số điểm cần vẽ 50 điểm.

>> x=linspace(0,10,50);

>> y=exp(-x).\*sin(x);

>> plot(x,y)



### 5.3.2 Đồ thị dạng đánh dấu:

Đồ thị dạng đánh dấu là loại đồ thị chỉ dùng các điểm như vòng tròn, hình thoi .... Thay vì dùng các đoạn thẳng nối lại với nhau.VD 4:

```
>> a = [8 8.5 5 8 6.5 7 7.8 8.5 7 7.5 5 9 7.5 9.2];
>> plot ( a,'*');
>>grid on
```

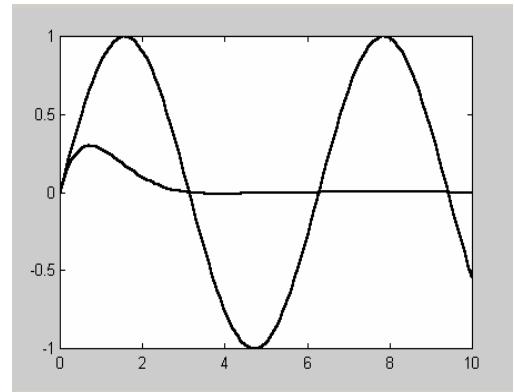
### 5.3.3 Vẽ nhiều đường biểu diễn trên cùng một đồ thị:

Cùng một bản đồ thị ta có thể vẽ nhiều đồ thị với các dữ liệu khác nhau và loại đường minh họa. Theo mặc định Matlab sẽ tự động gán loại màu sắc cho từng dữ liệu để phân biệt. Công thức tổng quát khi vẽ nhiều đồ thị trên cùng một hệ tọa độ:

### Plot ( tên biến 1, tên hàm1, tên biến 2, tên hàm 2....)

VD 5:

```
>>x=0:0.1:10;
>> y1=sin(x);
>> y2=sin(x).*3.^(-x);
>> plot(x,y1,x,y2)
```



### 5.3.4 Chú thích và kiểm soát đồ thị:

- title (' Tên tiêu đề đồ thị ')
- xlabel (' Tên trục x')
- ylabel (' Tên trục y')
- text (x,y, 'chuỗi ký tự') đưa một chuỗi ký tự vào điểm có tọa độ x,y trên đồ thị.
- gtext('chuỗi ký tự') đưa một chuỗi ký tự được xác định bởi dấu + hay con trỏ chuột.
- legend('chuỗi 1','chuỗi 2'...) đưa ra màn hình đồ họa một khung chú thích bao gồm các chuỗi. Vị trí của khung có thể được di chuyển bởi chuột.
- legend off: loại bỏ chức năng legend khỏi màn hình đồ họa.
- Grid on: bật chế độ lưới trong màn hình đồ họa.
- Grid off: tắt chế độ lưới trong màn hình đồ họa.
- Hold on: giữ lại các đồ thị đã vẽ ( dùng để vẽ nhiều đồ thị trên một hệ trục tọa độ)
- Hold off: ngược lại với hold on

Trong Matlab ta có thể chọn đường vẽ và màu theo 1 trong các kiểu sau:

Ký hiệu	Màu	Ký hiệu	Kiểu
y	vàng	.	Chấm điểm
m	đỏ tươi	o	Vòng tròn
c	xanh	x	Dấu x

r	đỏ	+	Dấu cộng
g	xanh lá cây	*	Dấu sao
b	xanh thẫm	-	Nét liền
w	trắng	-.	Gạch chấm
k	đen	--	Gạch gạch

Khi đó ta dùng lệnh: **plot(tên biến, tên hàm, 'ký hiệu màu ký hiệu kiểu đường')**

VD 6: vẽ hàm  $\cos(x)$ ,  $\cos(2x)$

```
>> x=linspace(0,10,50);
```

```
>> y=cos(x);y1 = cos(2*x);
```

% vẽ y bằng dấu x màu đen, y1 bằng dấu \* màu xanh thẫm

```
>> plot(x,y,'xk',x,y1,'*b');
```

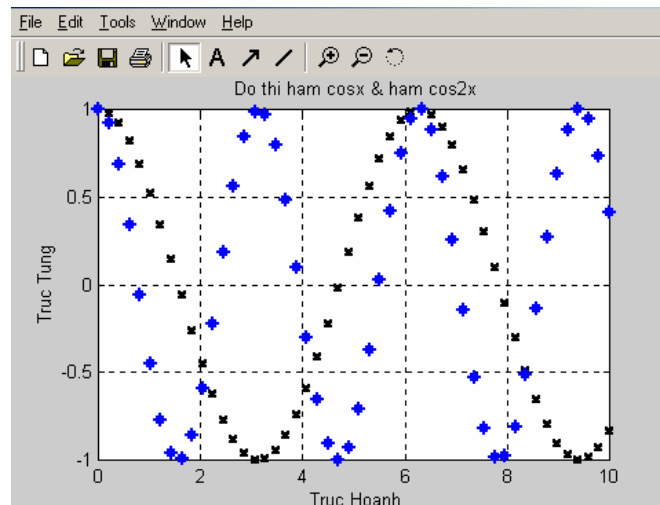
% Tên đồ thị

```
>> title(' Do thi ham cosx & ham cos2x')
```

```
>> xlabel(' Truc Hoanh')
```

```
>> ylabel(' Truc Tung')
```

```
>> grid on
```



**Gán giá trị thang đo:** Ngoài giá trị thang đo theo mặc định của chương trình, có thể tự chia thang đo theo dữ liệu riêng.

VD 7:

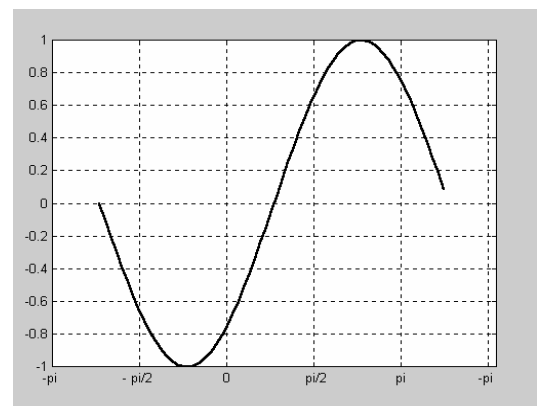
```
>> x = -pi : .1 : pi;
```

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

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

```
>> set(gca,'Xtick',-pi : pi/2 : pi)
```

```
>> set(gca,'Xticklabel', { '-pi','- pi/2','0',' pi/2','pi' })
```



### 5.3.5 Đồ thị hình thanh:

Loại đồ thị này thường dùng để minh họa các số liệu theo dạng thanh, có thể theo trục x hoặc trục y. VD8 : Vẽ biểu đồ khối lượng nhập hàng trong 12 tháng.

```
>> x = [230 255 270 210 170 240 265 280 240 300 320 345];
```

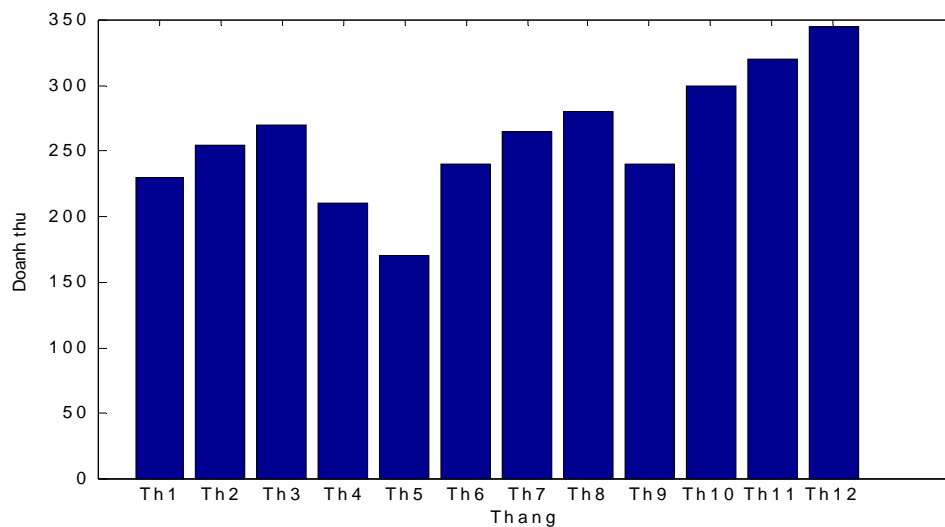
```
>> bar (x)
```

```
>> xlabel('Thang')
```

```
>> ylabel('Doanh thu')
```

```
>>set(gca,'Xticklabel',...
```

```
{ 'Th1','Th2','Th3','Th4','Th5','Th6','Th7','Th8','Th9','Th10','Th11','Th12' })
```



### 5.3.6 Đồ thị tọa độ cực:

Thường được áp dụng trong lĩnh vực thiên văn như hướng gió, hướng di chuyển của cơn bão...VD

9:

```
>> th = [0.:1:10];
```

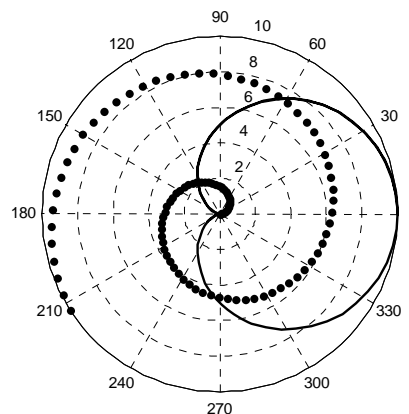
```
>> r1 = th;
```

```
>> r2 = 5*cos(th)+ 5;
```

```
>> % màu đen, đường chấm.
```

```
>> h1 = polar(th,r1,'k.');
```

```
>> set(h1,'Markersize',15)
```

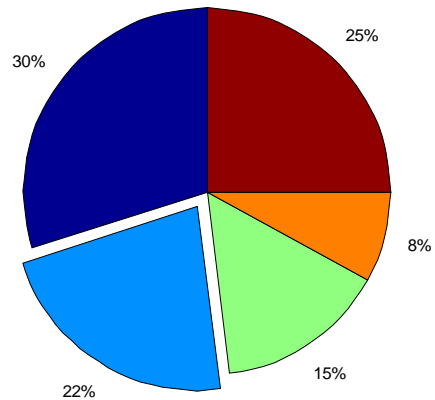


```
>> hold on
>> h2 = polar(th,r2,'k');% màu đen, liền.
```

### 5.3.7 Đồ thị hình Pie:

Là loại đồ thị tỷ lệ bách phân của từng loại dữ liệu để minh họa. Theo mặc định Matlab sẽ tô màu khác nhau cho từng thành phần dữ liệu.VD 10:

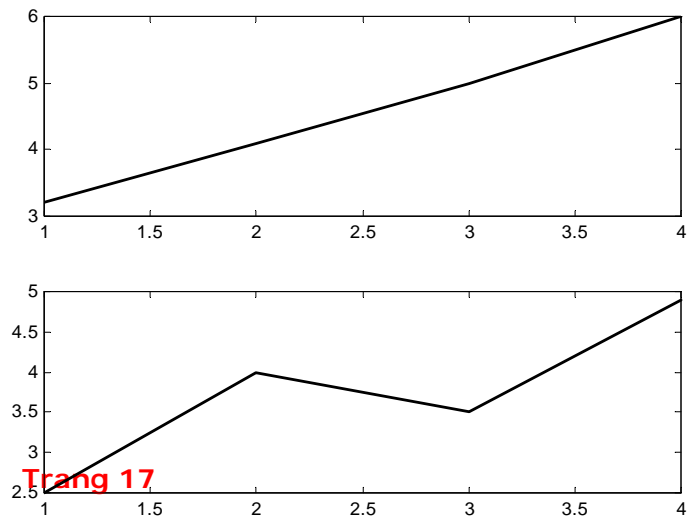
```
>> x = [30 22 15 8 25];
>> explot = [0 1 0 0 0];
>> pie(x,explot)
>> colormap jet
```



### 5.3.8 Hiện nhiều đồ thị trong một màn hình:

Trong một màn hình đồ thị, có thể cho hiện nhiều đồ thị với mỗi đồ thị là một loại dữ liệu khác nhau.VD 11:

```
>> a = [3.2 4.1 5 6];
>> b = [2.5 4 3.5 4.9];
>> subplot(2,1,1);plot(a)
% tạo trục tạo độ
>> subplot(2,1,2);plot(b)
% tạo trục tạo độ
```



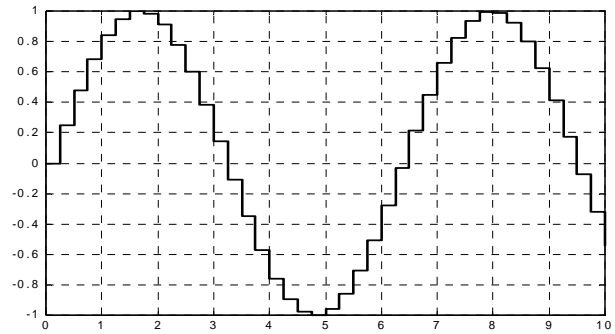
### 5.3.9 Lệnh stairs:

Để vẽ đồ thị bậc thang.

VD 12:

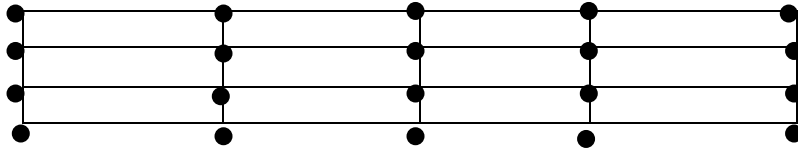
```
>>x = 0: .25: 10;
```

```
>>stairs (x,sin(x))
```



## 5.4 THỰC HÀNH VẼ CÁC LOẠI ĐỒ THỊ 3- D

### 5.4.1 Lệnh xác định vùng vẽ:



```
>>a = linspace(1,5,50);
```

```
>>b = linspace(1,10,100);
```

```
>>[ x,y] = meshgrid(a,b);
```

```
>> z = sin(x)+cos(y);
```

### 5.3.2 Lệnh vẽ : plot3(x,y,z) :

Tạo các đối tượng tuyến tính trong môi trường 3-D. VD 8:

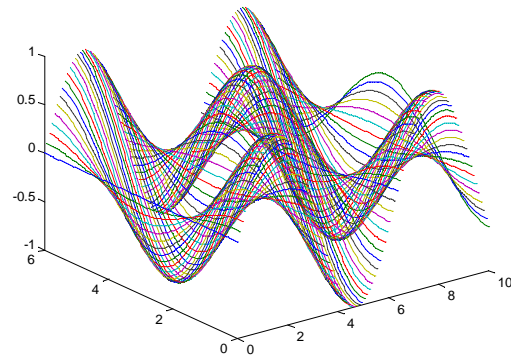
```
>> a =linspace(0,10,100);
```

```
>> b=linspace(0,6,100);
```

```
>>[x,y]=meshgrid(a,b);
```

```
>>z=sin(x)+cos(y);
```

```
>> plot3(x,y,z)
```

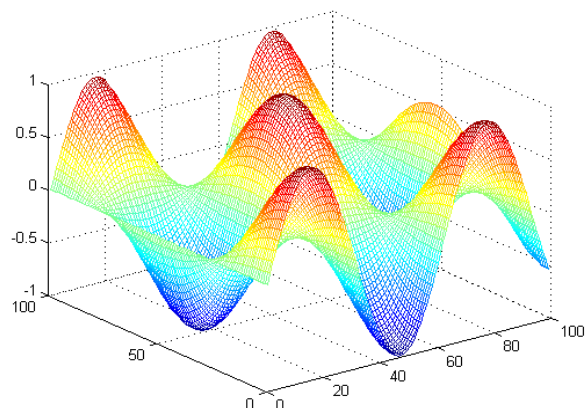


### *Người ta dùng lệnh*

*mesh(z): để hình vẽ có lưới.*

Vd

```
>>a=linspace(0,10,100);
```



```
>> b=linspace(0,6,100);  
>> [x,y]=meshgrid(a,b);  
>> z=sin(x).*cos(y);  
>> mesh(z)
```



## Chương 6

### Mô hình hoá, Mô phỏng hệ thống động sử dụng Simulink

#### Mục lục chương :

1. Khái niệm về simulink
  2. Phương pháp xây dựng mô hình
  3. ứng dụng simulink mô phỏng các bài toán kỹ thuật
- 6.1 Khái niệm về simulink

#### **What Is Simulink?**

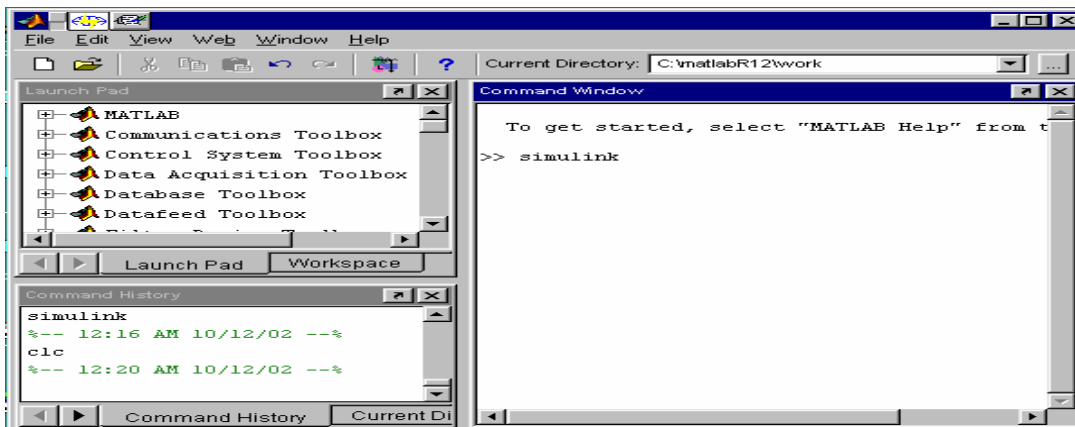
Simulink là một phần mềm gói gọn được sử dụng để xây dựng mô hình và mô phỏng , tính toán phân tích hệ thống động .Simulink cho phép mô tả hệ thống tuyến tính, hệ phi tuyến , các mô hình trong thời gian liên tục hoặc gián đoạn(lấy mẫu ) hay kết hợp cả hai. Đối với mô hình , Simulink cung cấp một giao diện đồ hoạ (GUI) cho việc xây dựng mô hình như là các khối (block diagrams), người sử dụng chỉ cần kích chuột và drag( chọn khối rồi giữ nguyên chuột trái rồi rê chuột đến vị trí đặt các khối). Với giao diện giao tiếp như vậy, bạn có thể vẽ Mô hình như là mô hình bạn vẽ trên "giấy"

Thư viện simulink bao gồm các khối thư viện sinks, sources( tạo tín hiệu), linear. . . . Và bạn cũng có thể tự tạo ra một khối block riêng của mình (viết trong S-function)

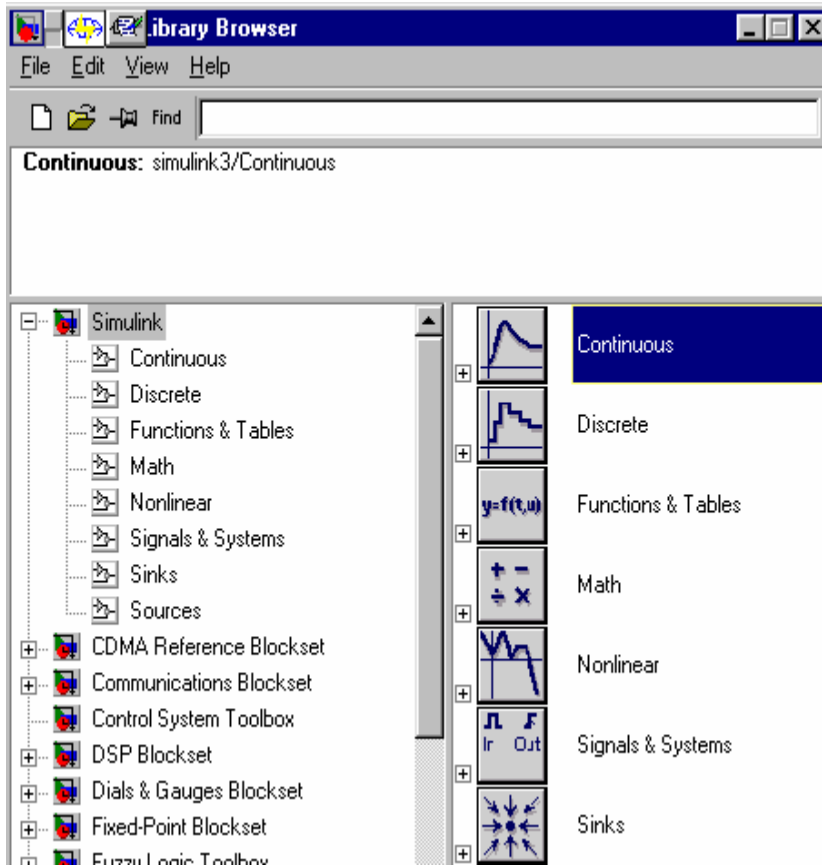
Xây dựng mô hình 'từ trên xuống' hoặc 'từ dưới lên trên' để xem kỹ các khối trong thư viện các khối source hoặc sink linear . . . bạn kích đúp chuột vào các khối đó. Sau khi định nghĩa mô hình bạn có thể mô phỏng mô hình đó , sử dụng scope để xem biểu diễn mô hình đó , ví dụ như một khối phát hình sin , đầu ra của khối đó được mắc với một scope để thể hiện kết quả của khối đó

#### 6.2 Thư viện simulink và môi trường làm việc (nơi xây dựng mô hình)

Để Bắt đầu vào vùng làm việc của simulink trong cửa sổ command window ta gõ lệnh >>simulink như sau:



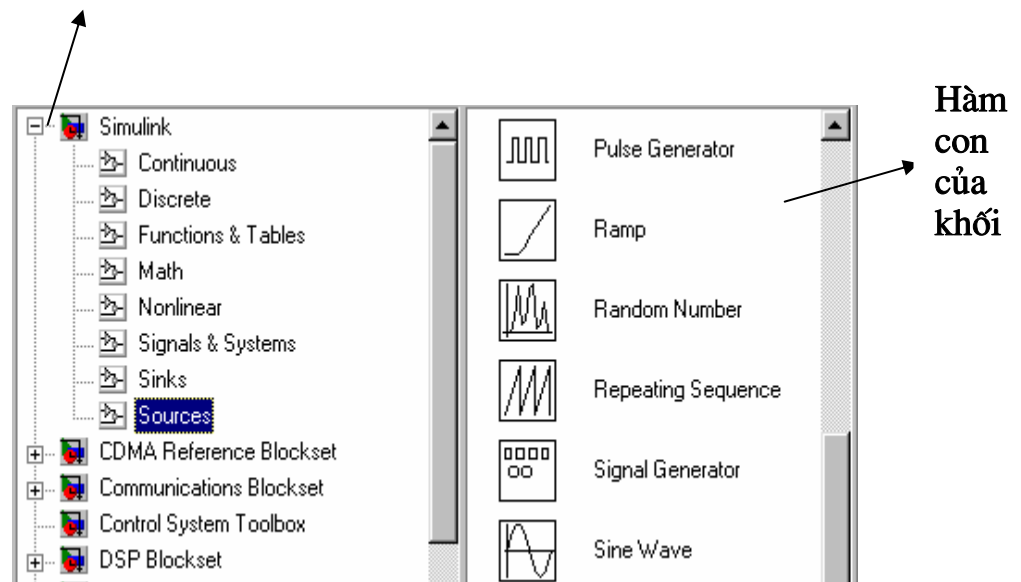
Trên màn hình sẽ xuất hiện thêm một cửa sổ mới , cửa sổ này chứa toàn bộ dữ liệu thư viện của Simulink, nó có thể di chuyển được bằng chuột như sau:



Các khối  
thư viện Của  
Simulink

Bạn có thể kích  
đúp chuột vào từng  
khối để xem các

khối con của nó (hoặc bạn nhấp đơn chuột vào danh mục tương ứng với khối từ simulink)  
*ví dụ bạn chọn khối source*



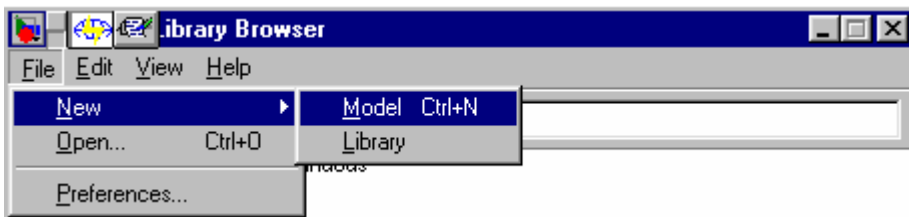
Hàm  
con  
của  
khối

## Tạo môi trường làm việc

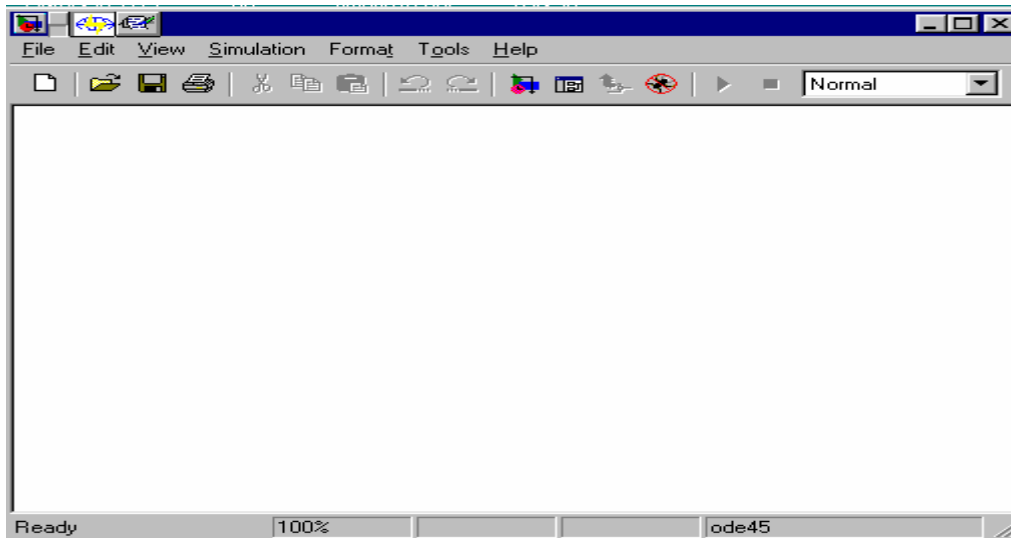
Từ cửa sổ Library Browser (xem hình trên) ta kích chuột vào file danh sách các mục New, Open, Preferences xuất hiện.

Để tạo môi trường làm việc (vùng để vẽ mô hình) ta chọn mục New rồi chọn Model

Ctrl+N một cửa sổ làm việc xuất hiện



Cửa sổ làm việc như sau

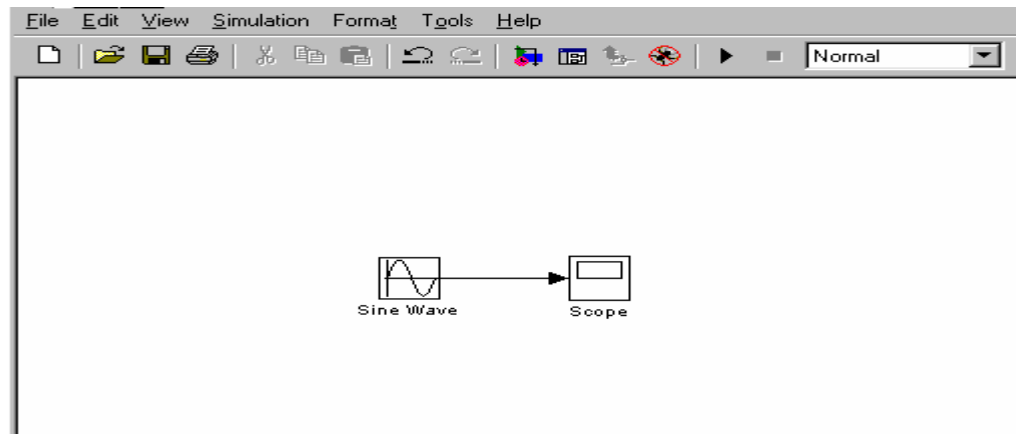


Đặt lại tên cho mô hình bằng cách vào mục file -> Save as

### 6.3 Phương pháp xây dựng mô hình

Tất cả các bước trên là chuẩn bị cho việc xây dựng mô hình mô phỏng

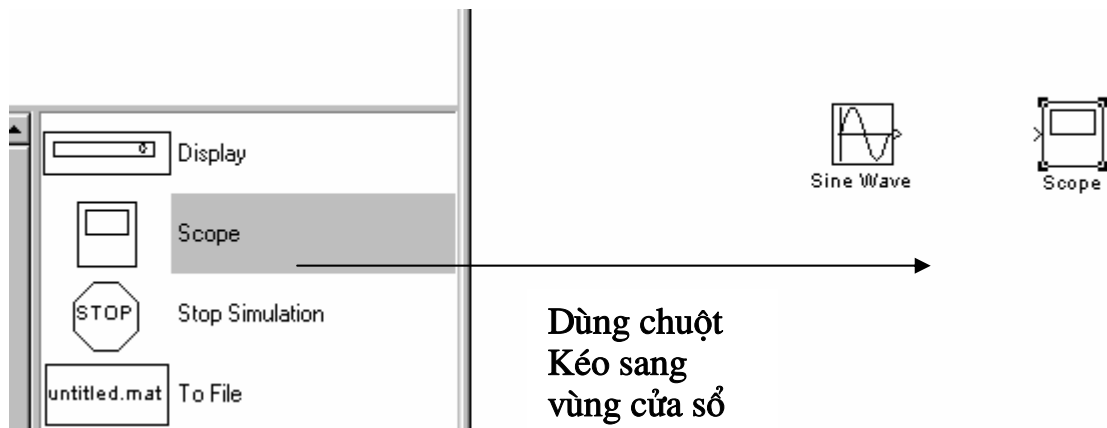
Giả sử ta muốn xây dựng mô hình phân tích sóng sin trên cửa sổ làm việc như sau:(xem



hình vẽ dưới đây)

### Các bước:

Sau khi **tao môi trường làm việc mới** (các bước giới thiệu ở trên) tiếp đến

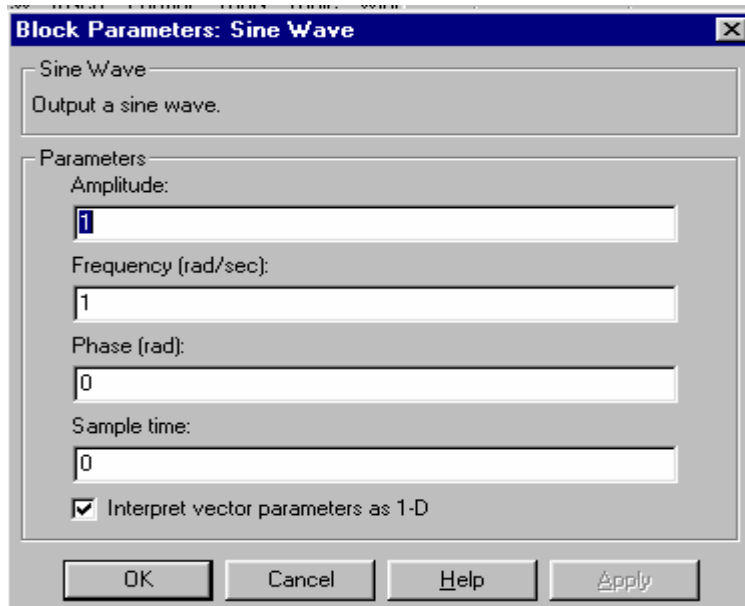


Tìm khối hàm sin ở trong khối thư viện nào( bằng cách chọn từng khối bằng chuột từ các mục dưới Simulink) ở ví dụ này hàm tạo sin ở trong khối Source (xem hình trên), dùng chuột chọn vào khối SineWave giữ nguyên chuột rồi kéo sang vùng cửa sổ làm việc , trên cửa sổ làm việc xuất hiện khối hàm SineWave, tương tự ta làm như vậy với Scope trong khối Sink việc nối các khâu với nhau có các mũi tên , dùng chuột nối các mũi tên đó lại.

**Đặt lại các thông số của các hàm** bằng cách kích đúp lên các khối(các khối đã ở trong vùng cửa sổ làm việc). Đối với khối SinWave thì có các thông số có thể thay đổi được là:

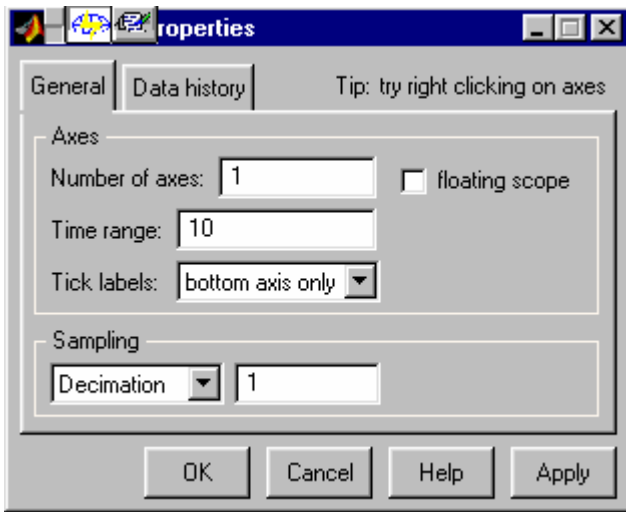
+ Chu kỳ( tần số) Frequency(rad/s)

+ Biên độ Amplitude



+ Sample time (thời gian lấy mẫu)

Khối Scope:

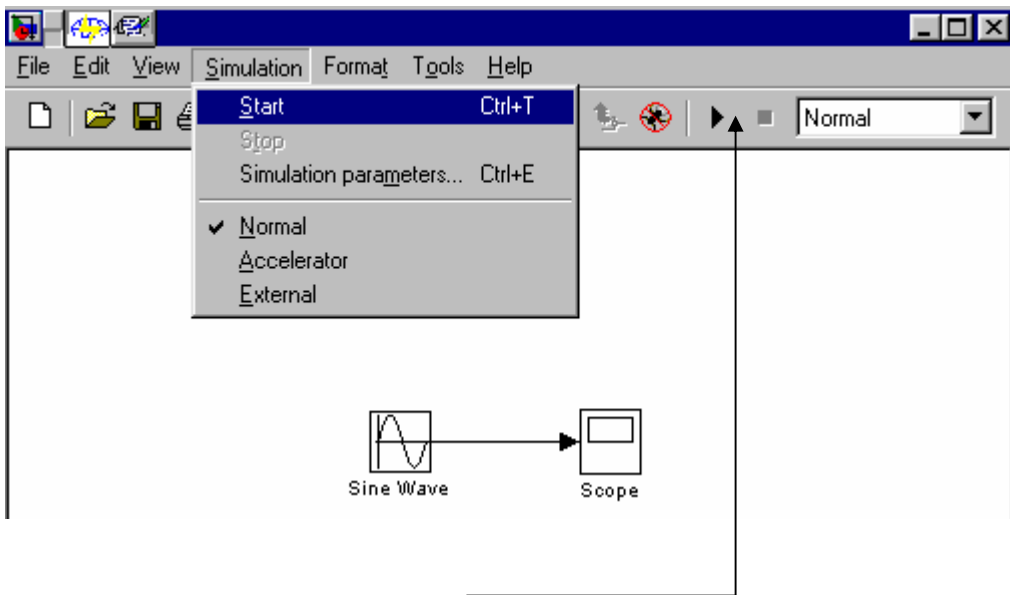


Sau khi hoàn tất đặt lại các thông số , đến công việc quan trọng nhất là kết quả mô phỏng

:

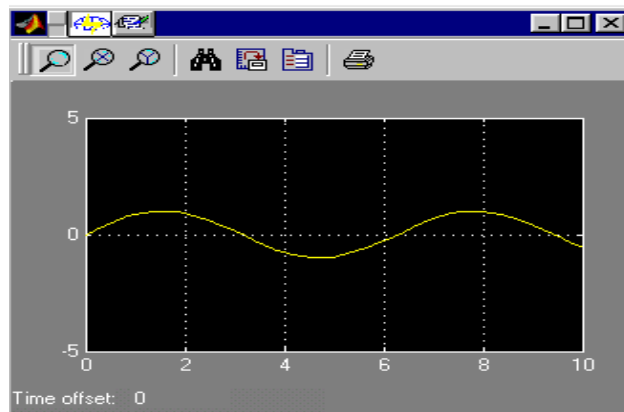
1. Trên thanh công cụ nhấp chuột vào mục Simulation

Và chọn Start



2 . Hoặc nhấp vào nút tamgiác như trên(tác dụng lệnh giống như vào lệnh Start)

3.Muốn Dừng quá trình đang mô phỏng ta kích vào nút vuông bên cạnh nút tam giác(nút



này chỉ xuất hiện khi ta đang mô phỏng)

4.Để xem kết quả của khối SineWave ta kích đôi chuột vào Scope

## CHƯƠNG 6

### ỨNG DỤNG TÍNH TOÁN TRONG MATLAB

#### 6.1 LỆNH SYM, SYMS:

**Mục đích:** Biến đổi các số, biến, đối tượng thành Symbolics.

Ví dụ:

```
>> sym x y
>> x = sym('x'); y = sym('y');
% x,y là các biến symbolic.
>> syms x y real
>> x = sym('x', 'real'); y = sym('y', 'real')
% x,y là biến kiểu thực symbolics
syms x real y
x = sym('x', 'real'); y = sym('y')
% x là biến kiểu thực, y là biến bất kỳ kiểu symbolic
syms x y unreal
% x, y không phải là biến thực
syms t
Q = sym('Q(t)');
% t biến symbolic và Q là hàm symbolic.
```

#### 6.2 NHÂN 2 ĐA THỨC: ( Dùng lệnh conv)

$$y1 = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0$$

$$y2 = b_n x^n + b_{n-1} x^{n-1} + \dots + b_0$$

**Bước 1:** Lập 2 ma trận hàng tên y1, y2 có các phần tử là các hệ số từ  $a_n$  đến  $a_0$  và  $b_n$  đến  $b_0$  giảm dần theo bậc của phương trình ( Nếu hệ số nào không có ghi 0 )

**Bước 2:** dùng lệnh conv để nhân 2 đa thức.

```
>> y3 = conv(y1, y2)
```

VD:  $y1 = 2x^2 + 3x + 1$

$y2 = 3x^2 + 4x$

```
>> y1 = [2 3 1]
```



```
>> y2 = [3 4 0]
>> y3 = conv(y1,y2)
>> y3 = 6 17 15 4 0
```

**Chú ý:** hàm conv chỉ thực hiện nhân 2 đa thức. Muốn nhân nhiều đa thức với nhau ta phải thực hiện nhiều lần hàm conv.

## 6.3 CÁC TÌNH TOÁN CHO PHƯƠNG TRÌNH:

### 6.3.1 Giải phương trình bậc cao: (Lệnh Roots)

$$y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0$$

Bước 1: Lập 1 ma trận hàng có các phần tử là các hệ số từ  $a_n$  đến  $a_0$  giảm dần theo bậc của phương trình ( Nếu hệ số nào không có ghi 0 )

Bước 2: Dùng lệnh Roots để giải ma trận vừa tạo được.

VD giải phương trình sau:  $y = x^5 - 2x^4 + 5x^2 - 1$

```
>> y = [ 1      -2      0      5      0      -1]
```

```
y =
```

```
1 -2 0 5 0 -1
```

```
>> kq=roots(y)
```

```
kq =
```

```
1.5862 + 1.1870i
```

```
1.5862 - 1.1870i
```

```
-1.1606
```

```
-0.4744
```

```
0.4627
```

### 6.3.2 Biết nghiệm tìm lại phương trình: ( lệnh poly )

Lấy kết quả của ví dụ trên

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

```
>>poly(A)
```

```
ans =
```

```
1 -5 8 14
```

### 6.3.3 Chuyển từ phương trình hệ số sang phương trình có chứa cả tham số: ( poly2sym)

```
>> poly2sym([1 0 -2 -5])
```

```

ans =
x^3-2*x-5
>> y = [1 2 3 0 1]
y = 1 2 3 0 1
>> poly2sym(y)
ans = x^4+2*x^3+3*x^2+1

```

### 6.3.4 Giải hệ phương trình tuyến tính:

Vd giải hệ phương trình tuyến tính sau:

$$\begin{cases} 2x + 3y + z = 7 \\ 3x + 6y - 4z = 19 \\ x + y + z = 2 \end{cases}$$

Thực chất hệ phương trình trên có thể đưa về phép toán ma trận sau:

$$\begin{bmatrix} 2 & 3 & 1 \\ 3 & 6 & -4 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 7 \\ 19 \\ 2 \end{bmatrix}$$

Như vậy việc giải hệ PT tuyến tính thực chất là thực hiện phép toán về ma trận.

```

>> A=[2 3 1;3 6 -4;1 1 1]
A =
    2    3    1
    3    6   -4
    1    1    1
>> B=[7;19;2]
B =
    7
   19
    2
>> C=inv(A)
C =
  -2.5000  0.5000  4.5000
   1.7500 -0.2500 -2.7500
   0.7500 -0.2500 -0.7500

```

```
>> kq=C*B
kq =
    1.0000
    2.0000
   -1.0000
```

VD2:

```
>>A=[1+i 2i;3+i 1]
A =    1.0000 + 1.0000i    0 + 2.0000i
       3.0000 + 1.0000i    1.0000

>> C=inv(A)
C =
    0.0882 + 0.1471i    0.2941 - 0.1765i
   -0.1176 - 0.5294i   -0.0588 + 0.2353i

>> B=[1; 2+i]
B =
    1.0000
    2.0000 + 1.0000i

>> KQ=C*B
KQ =
    0.8529 + 0.0882i
   -0.4706 - 0.1176i
```

VD3

```
>>syms a1 a2 b1 b2 c1 c2
>>A=[a1 a2;b1 b2]
A =
    [ a1, a2]
    [ b1, b2]

>> B=[c1;c2]
B =
    [ c1]
    [ c2]

>> C=inv(A)
```

```

C =
    [-b2/(-a1*b2+b1*a2), a2/(-a1*b2+b1*a2)]
    [ b1/(-a1*b2+b1*a2), -a1/(-a1*b2+b1*a2)]
>> KQ=C*B
KQ =
    [-b2/(-a1*b2+b1*a2)*c1+a2/(-a1*b2+b1*a2)*c2]
    [ b1/(-a1*b2+b1*a2)*c1-a1/(-a1*b2+b1*a2)*c2]

```

### 6.3.5 Giải hệ phương trình phi tuyến:( Lệnh solve)

Ví dụ:

```

sin(x)+y^2+log(z)=7
3*x+2^y+z^3=4
x+y+z=2
>>[x,y,z]=solve('sin(x)+y^2+log(z)=7','3*x+2^y+z^3=4','x+y+z=2')
x =    -2.3495756224572032187410536400368
y =     2.6835269194785219427270239079010
z =     1.666048702978681276014029732135

```

Ví dụ:

```

x^2 + x*y + y = 3
x^2 - 4*x + 3 = 0
>>[x,y] = solve('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0')
x =
[ 1]
[ 3]
y =
[ 1]
[-3/2]

```

### 6.3.6 Giải hệ phương trình tham số:

```

>>[a,u] = solve('a*u^2 + v^2= 0','u - v = 1','a,u')
a =
    -v^2/(v^2+2*v+1)
u =
    v+1

```

```
>>[a,v] = solve('a*u^2 + v^2','u - v = 1','a,v')
a = -(u^2-2*u+1)/u^2
v = u-1
```

### 6.3.7 Giải hệ phương trình vi phân thường: ( lệnh dsolve)

```
>>y = dsolve('(D2y) =1','y(0) = 1')
y = 1/2*t^2+C1*t+1
>>[x,y]=dsolve('Dx = y', 'Dy = -x')
x= cos(t)*C1+sin(t)*C2
y = -sin(t)*C1+cos(t)*C2
```

### 6.3.8 Giải hệ phương trình vi phân theo hàm có sẵn của Matlab:

Ví dụ: Cho hệ phương trình vi phân

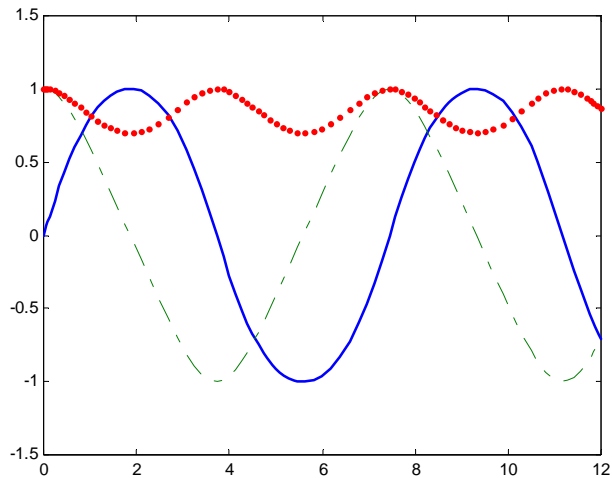
$$\begin{aligned} y_1' &= y_2 y_3 & y_1(0) &= 0 \\ y_2' &= -y_1 y_3 & y_2(0) &= 1 \\ y_3' &= -0.51 y_1 y_2 & y_3(0) &= 1 \end{aligned}$$

Chương trình mô tả phương trình vi phân dạng M-file:

```
function dy = rigid(t,y)
dy = zeros(3,1);
dy(1) = y(2) * y(3);
dy(2) = -y(1) * y(3);
dy(3) = -0.51 * y(1) * y(2);
```

Thời gian giải phương trình vi phân Tspan =[0 12], vector điều kiện đầu [0 1 1]

```
>>options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
>>[t,y] = ode45('rigid',[0 12],[0 1 1],options);
>>plot(t,y(:,1),'-',t,y(:,2),'-.',t,y(:,3),'-.')
```



## 6.4 LỆNH VÀ HÀM TRONG SYMBOLIC MATLAB:

Symbolics Matlab là thư viện các phép tính toán kiểu ký tự được đưa vào môi trường tính

học của Matlab.

Ta cần chú ý rằng, khi viết một lệnh hay một hàm trong Matlab phải viết bằng chữ thường.

### 6.4.1 Tính toán (Calculus):

- **Tính đạo hàm (diff):**

- `diff(S)`: Đạo hàm biểu thức symbolic S với biến của đạo hàm tự do.
- `diff(S,'v')` hay `diff(S,sym('v'))`: Đạo hàm biểu thức symbolic S với biến lấy đạo hàm là biến symbolic v.
- `diff(S,n)`: Đạo hàm cấp n biểu thức S, n là số nguyên dương.

Ví dụ:

```
>>syms x t
>> y = sin(x^2);
>>z = diff(y);
z = 2*cos(x^2)*x
pretty(z)% hiển thị dạng quen thuộc
2.cos^2x.x
>>y = diff(t^6,6) % đạo hàm bậc 6 của hàm t^6.
```

$$Y = 720$$

Ví dụ:

```
>>syms u v
>>y = u^2*v - u*v^3;
>> y2u = diff(y,u,2) %dao ham cap 2 theo u
>> y3u = diff(y,v,3) %dao ham cap 3 theo v
y2u = 2*v
y3u = -6*u
```

- **Tính tích phân( int):**

- $\text{int}(S)$ : Tích phân không xác định của biểu thức symbolic S với biến tự do mặc định. Muốn biết biến mặc định ta dùng lệnh `findsym`.
- $\text{int}(S,v)$ : Tích phân không xác định của biểu thức symbolic S với biến tích phân v.
- $\text{int}(S,a,b)$ : Tích phân không xác định của biểu thức symbolic S với biến tự do và cận lấy tích phân từ [a,b].
- $\text{int}(S,v,a,b)$ : Tích phân không xác định của biểu thức symbolic S với biến tích phân v và cận lấy tích phân từ [a,b].

Ví dụ:

```
>>syms x t z alpha
>>int(-2*x/(1+x^2)^2)
ans = 1/(1+x^2)
>>int(x/(1+z^2),z)
ans = x*atan(z)
>>int(x*log(1+x),0,1)
ans = 1/4
>>int(-2*x/(1+x^2)^2)
ans = 1/(1+x^2)
>> int([exp(t),exp(alpha*t)])
ans = [ exp(t), 1/alpha*exp(alpha*t)]
```

Ví dụ: Tính tích phân  $I = \int_{-\infty}^{\infty} e^{-(sx)^2} dx$

```
>>Syms x s real
```

```
>>f = exp(-(s*x)^2);
>>I = int(f,x,-inf,inf)% inf là vô cùng lớn
I =
      Signum(s)/s*pi^(1/2)
```

Hàm signum chính là hàm sign (hàm dấu), nghĩa là sign(s) cho ta:  
 $\text{sign}(s) = 1$  khi  $s > 0$ ;  $\text{sign}(s) = 0$  khi  $s = 0$ ;  $\text{sign}(s) = -1$  khi  $s < 0$ ;

- **Tính giới hạn(limit):**

- $\text{limit}(F, x, a)$  : Tìm giới hạn của biểu thức F khi  $x \rightarrow a$ .
- $\text{limit}(F, a)$  : Tìm giới hạn của biểu thức F với biến độc lập.
- $\text{limit}(F)$  : Tìm giới hạn của biểu thức F khi  $a = 0$ .
- $\text{limit}(F, x, a, \text{'right'})$  hoặc  $\text{Limit}(F, x, a, \text{'left'})$  : Tìm giới hạn phải hoặc bên trái

Ví dụ:

```
>>syms x a t h
>>limit(sin(x)/x)
ans = 1
>>limit(1/x,x,0,'right')
ans = inf
>>limit(1/x,x,0,'left')
ans = -inf
>>limit((sin(x+h)-sin(x))/h,h,0)
ans = cos(x)
>>v = [(1+a/x)^x,exp(-x)];
>>limit(v,x,inf,'left')
ans = [exp(a),0]
```

- **Tính tổng của dãy số là các biến symbolic(symsum):**

- $\text{symsum}(S)$ : Tổng của biểu thức symbolic theo biến symbolic k , k được xác định bằng lệnh `findsym` từ  $0 \rightarrow k - 1$ .
- $\text{symsum}(S,v)$ : Tổng của biểu thức symbolic S theo biến symbolic v,v được xác định từ  $0 \rightarrow k - 1$ .



- `symsum(S,a,b)`, `symsum(S,v,a,b)`: Tổng của biểu thức symbolic S theo symbolic v, v được xác định từ  $v = s$  đến  $v = b$ .

Ví dụ:

```
>>syms k n x
>>symsum(k^2)
ans = 1/3*k^3-1/2*k^2+1/6*k
>>symsum(k)
ans = 1/2*k^2-1/2*k
>>symsum(sin(k*pi)/k,0,n)
ans = -1/2*sin(k*(n+1))/k+1/2*sin(k)/k/(cos(k)-1)*cos(k*(n+1))-
1/2*sin(k)/k/(cos(k)-1)
>>symsum(k^2,0,10)
ans = 385
>>symsum(x^k/sym('k!'),k,0,inf)
ans = exp(x)
```

Ví dụ: Cho tổng của 2 dãy

$$S1 = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots$$

$$S2 = 1 + x + x^2 + \dots$$

```
>>syms x k
>>s1 = symsum(1/k^2,1,inf) %inf là vô cùng.
s1 = 1/6*pi^2
>>s2 = symsum(x^k,k,0,inf)
```

- **Tìm hàm ngược (finverse):**

- `finverse(f)`: Tìm hàm ngược của f. f là hàm symbolic với một biến x
- `finverse(f,u)`: Tìm hàm ngược của f. f là hàm symbolic với một biến u.

Ví dụ:

```
>>syms u v x
>>finverse(1/tan(x))
ans = atan(1/x)
>>finverse(exp(u-2*v),u)
ans = 2*v+log(u)
```

$$s2 = -1/(x-1)$$

- **Khai triển taylor(taylor):**

- taylor(f)
- taylor(f,n,v): Cho ta xấp xỉ đa thức theo Maclaurin bậc (n-1) của biểu thức, hàm khai triển symbolic f và v là biến độc lập trong biểu thức. v có thể là một xâu (string) hay là biến symbolic.
- taylor(f,n,v,a): Khai triển Taylor của biểu thức hay hàm symbolic f quanh điểm a. Đối số có thể là giá trị số, một hàm symbolic hay một xâu.....Nếu không cho giá trị n thì mặc nhiên trong Matlab n = 6.

Vi dụ: Khai triển Taylor của hàm  $f = e^{x\sin(x)}$  quanh điểm  $x_0 = 2$  (Nếu  $x_0 = 0$  ta có khai triển Maclaurin).

```
>>syms x
>> f = exp(x*sin(x));
>>t = taylor(f,4,2)% khai triển 4 số hạng đầu tiên khác 0 và xung quanh điểm x0 = 2
```

Kết quả:

$$\begin{aligned} & \exp(2*\sin(2))+\exp(2*\sin(2))*(2*\cos(2)+\sin(2))*(x-2)+\exp(2*\sin(2))*(- \\ & \sin(2)+\cos(2)+2*\cos(2)^2+2*\cos(2)*\sin(2)+1/2*\sin(2)^2)*(x-2)^2+\exp(2*\sin(2))*(- \\ & 1/3*\cos(2)-1/2*\sin(2)-\cos(2)*\sin(2)+2*\cos(2)^2- \\ & \sin(2)^2+4/3*\cos(2)^3+2*\cos(2)^2*\sin(2)+\cos(2)*\sin(2)^2+1/6*\sin(2)^3)*(x-2)^3 \end{aligned}$$

Bây giờ ta có thể vẽ hàm đã cho và hàm đã khai triển bằng chuỗi Taylor quanh điểm  $x_0 = 2$  và cho nhận xét.

```
>>syms x
>> f = exp(x*sin(x));
>>t = taylor(f,10,2);
>>xd= 1:0.05:3;
>>yd = subs(f,x,xd);% thay thế biến x bằng xd
>>ezplot(t,[1,3])% vẽ hàm symbolic
>> hold on
>>plot(xd,yd,'r-')
```

#### 6.4.2 Các hàm làm đơn giản hoá các biểu thức:

- **Gom số hạng, biến(collect):**

- collect(S): S là đa thức, gom các số hạng chứa biến x
- collect(S,v): S là đa thức, gom các số hạng chứa biến v

Ví dụ:

```
>>syms x y;
>>R1 = collect((exp(x)+x)*(x+2))
>>R2 = collect((x+y)*(x^2+y^2+1), y)
>>R3 = collect([(x+1)*(y+1),x+y])
```

Kết quả:

$$R1 = x^2 + (\exp(x) + 2)x + 2\exp(x)$$

$$R2 = y^3 + x^2y + (x^2 + 1)y + x(x^2 + 1)$$

$$R3 = [(y + 1)x + y + 1, x + y]$$

- **Khai triển biểu thức(expand):**

➤ expand: Khai triển biểu thức symbolic S.

Ví dụ:

```
>>syms x y a b c t
>>expand((x-2)*(x-4))
ans = x^2 - 6*x + 8
>>expand(cos(x+y))
ans = cos(x)*cos(y) - sin(x)*sin(y)
>>expand(exp((a+b)^2))
ans = exp(a^2)*exp(a*b)^2*exp(b^2)
>>expand(log(a*b/sqrt(c)))
ans = log(a) + log(b) - 1/2*log(c)
>>expand([sin(2*t), cos(2*t)])
ans = [2*sin(t)*cos(t), 2*cos(t)^2 - 1]
```

- **Phân tích biểu thức thành thừa số(factor):**

➤ Factor(X): Phân tích biểu thức mảng symbolic X thành thừa số.

Ví dụ:

```
>>syms x y a b
>>factor(x^3 - y^3)
(x-y)*(x^2 + x*y + y^2)
>>factor([a^2 - b^2, a^3 + b^3])
[(a-b)*(a+b), (a+b)*(a^2 - a*b + b^2)]
>>factor(sym('12345678901234567890'))
```

```
(2)*(3)^2*(5)*(101)*(3803)*(3607)*(27961)*(3541)
```

- **Phân tích đa thức ra dạng thừa số(horner):**

- $R = \text{horner}(p)$ :

Ví dụ:

```
>>syms x y
>>horner(x^3-6*x^2+11*x-6)
ans = -6+(11+(-6+x)*x)*x
>>horner([x^2+x;y^3-2*y])
ans = [ (1+x)*x
[(-2+y^2)*y]
```

- **Lấy tử số và mẫu số(numden):**

- $[n,d] = \text{numden}(A)$ : lấy ra tử số của A là n, mẫu số của A là d

Ví dụ:

```
>>syms x y a b
>>A = (4-x)/5;
>>[n,d] = numden(A)
n = 4-x
d = 5
>>[n,d] = numden(x/y + y/x)
n = x^2+y^2
d = y*x
>>A = [a, 1/b]
>>[n,d] = numden(A)
n = [a, 1]
d = [1, b]
```

- **Tìm dạng tối giản của đa thức( simple, simplify):**

- $R = \text{simplify}(S)$
- $R = \text{simple}(S)$
- $[r, \text{how}] = \text{simple}(S)$

Ví dụ:

```
>>syms x y a b c
>>simplify(sin(x)^2 + cos(x)^2)
```

```

ans = 1
>>simplify(exp(c*log(sqrt(a+b))))
ans = (a+b)^(1/2*c)
>>S = [(x^2+5*x+6)/(x+2),sqrt(16)];
>>R = simplify(S)
R = [x+3,4]

```

### 6.4.3 Biến đổi :

- **Biến đổi fourier:**

- $F = \text{fourier}(f)$ : Biến đổi fourier của hàm vô hướng  $f$  với biến độc lập mặc nhiên  $f$  và cho ta hàm mặc nhiên qua phép biến đổi này là  $w$ .
- $F = \text{fourier}(f,v)$ :  $F$  là hàm của biến  $v$  thay thế biến mặc nhiên  $w$ .
- $F = \text{fourier}(f,u,v)$ :  $f$  là hàm của  $u$  và  $F$  là hàm của  $v$  chúng thay thế các biến mặc nhiên  $x$  và  $w$ .

Ví dụ:

```

>>syms x w u
>>f = exp(-x^2)
>>fourier(f)
ans = pi^(1/2)*exp(-1/4*w^2)
>>g = exp(-abs(w))
>>fourier(g)
ans = 2/(1+t^2)
>>f= x*exp(-abs(x))
>>fourier(f,u)
ans = -4*i/(1+u^2)^2*u
>>syms x v u real
>>f= exp(-x^2*abs(v))*sin(v)/v
>>fourier(f,v,u)
ans = -atan((u-1)/x^2)+atan((u+1)/x^2)

```

- **Biến đổi ngược fourier:**

- $f = \text{ifourier}(F)$ : Biến đổi ngược của hàm mục tiêu vô hướng  $F$  với biến độc lập mặc nhiên  $w$ . phép biến đổi ngược này là hàm của  $x$ .
- $f = \text{ifourier}(F,u)$ :  $f$  là hàm của biến  $u$  thay thế biến mặc nhiên  $x$ .
- $f = \text{ifourier}(F,v,u)$ :  $F$  là hàm của  $v$  và  $f$  là hàm của  $u$  chúng thay thế các biến mặc nhiên  $w$  và  $x$  tương ứng.

Ví dụ:

```
>>syms a w x t v real
>>f = exp(-w^2/(4*a^2))
>>F = ifourier(f);
>>F = simple(F)
F = a*exp(-x^2*a^2)/pi^(1/2)
>>g=exp(-abs(x))
>>ifourier(g)
ans = 1/(1+t^2)/pi
>>f=2*exp(-abs(w))-1
>>simplify(ifourier(f,t))
ans = (2-pi*Dirac(t)-pi*Dirac(t)*t^2)/(pi+pi*t^2)
>>f=exp(-w^2*abs(v))*sin(v)/v;
>>ifourier(f,v,t)
ans = 1/2*(atan((t+1)/w^2) - atan((-1+t)/w^2))/pi
```

- **Biến đổi laplace:**

- $L = \text{laplace}(F)$ : Biến đổi Laplace của hàm  $F$  với biến mặc nhiên độc lập  $t$ . nó cho ta một hàm của  $s$
- $L = \text{laplace}(F,t)$ :  $L$  là một hàm của  $t$  thay thế biến mặc nhiên  $s$ .
- $L = \text{laplace}(F,w,z)$ :  $L$  là hàm của  $z$  và  $F$  là hàm của  $w$ , nó thay thế các biến symbolic mặc nhiên  $s$  và  $t$  tương ứng.

Ví dụ:

```
>>syms t v x a
>>f = t^4
>>laplace(f)
ans = 24/s^5
>>g=1/sqrt(s)
>>laplace(g)
ans = 1/s^(1/2)*pi^(1/2)
>>f=exp(-a*t)
>>laplace(f,x)
ans= 1/(x + a)
```

```
>>f=1- cos(t*v)
>>laplace(f,x)
ans = 1/x-x/(x^2+v^2)
```

- **Biến đổi Laplace ngược**

- $F = \text{ilaplace}(L)$ : Biến đổi Laplace ngược của hàm symbolic L với biến mặc nhiên độc lập s. Nó cho ta một hàm của t.
- $F = \text{ilaplace}(L,y)$ : F là hàm của y thay thế biến mặc nhiên t.
- $F = \text{ilaplace}(L,y,x)$ : F là hàm của x và L là hàm của y, nó thay thế các biến symbolic mặc nhiên t và s.

Ví dụ :

```
>>syms s a t
>>f=1/s^2
>>ilaplace(f)
ans = t
>>g=1/(t-a)^2
>>ilaplace(g)
ans = x*exp(a*x)
>>syms u a x
>>f=1/(u^2-a^2)
>>ilaplace(f,x)
ans = 1/(-a^2)^(1/2)*sin((-a^2)^(1/2)*x)
>>syms s v x
>>f=s^3*v/(s^2+v^2)
>>ilaplace(f,v,x)
ans = s^3*cos(s*x)
```

#### 6.4.4 Áp dụng vẽ đồ thị:

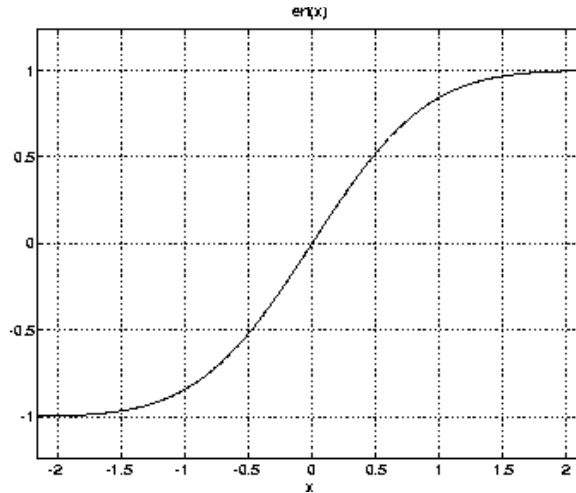
- **ezplot (vẽ đồ thị)**

- $\text{ezplot}(f)$ : Vẽ hàm  $f = f(x)$  với miền mặc nhiên  $-2\pi < x < 2\pi$ .
- $\text{ezplot}(f,[\text{min},\text{max}])$ : Vẽ hàm  $f = f(x)$  trong miền giá trị  $[\text{min},\text{max}]$  của biến.
- $\text{ezplot}(x,y)$ : Vẽ đường cong tham số  $x = x(t)$ ;  $y = y(t)$  với biến mặc nhiên

$$0 < t < 2\pi.$$

Ví dụ:

```
>>syms x
>>ezplot(erf(x))
>>grid
```

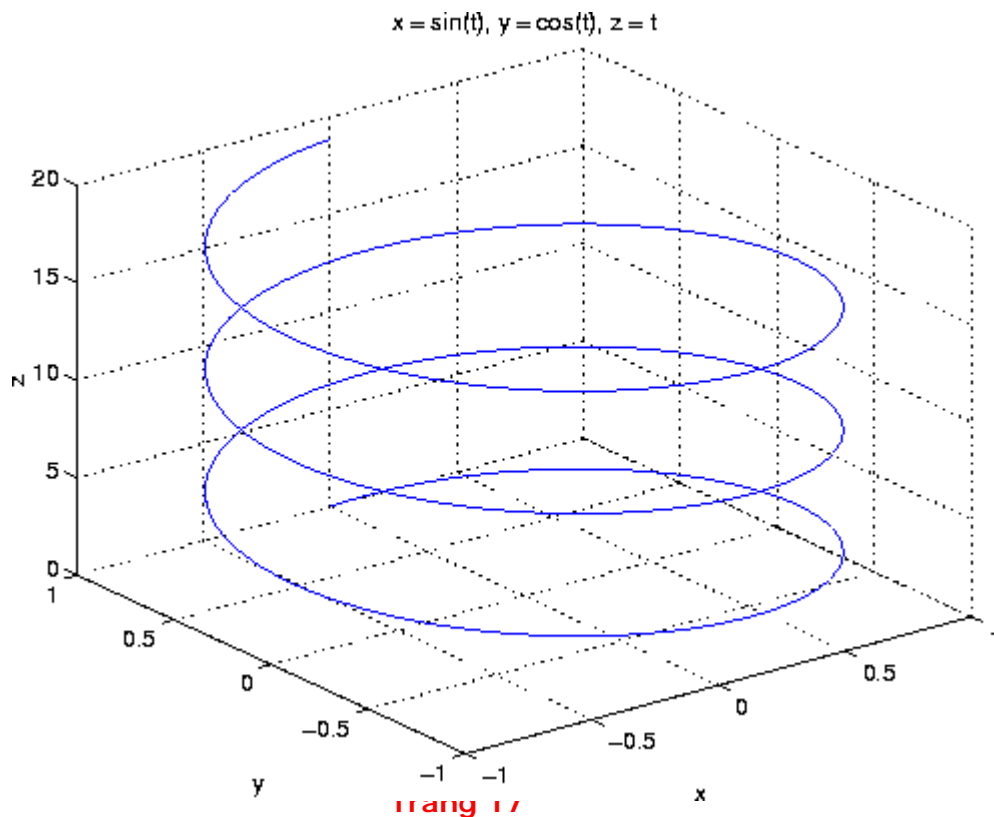


- ezplot3( vẽ đường trong 3 chiều)

- ezplot3(x,y,z): Vẽ các hàm  $x = x(t)$ ,  $y = y(t)$ , và  $z = z(t)$  với miền mặc định là  $0 < t < 2\pi$ .
- ezplot3(x,y,z,[tmin,tmax]): Vẽ các hàm  $x = x(t)$ ,  $y = y(t)$ , và  $z = z(t)$  trong khoảng giá trị  $t_{min} < t < t_{max}$ .

Ví dụ:

```
>>syms t; ezplot3(sin(t), cos(t), t,[0,6*pi])
```



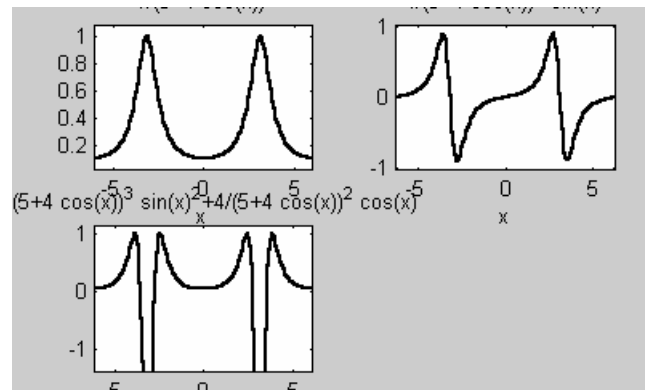


- **Ví dụ :** Cho hàm  $f(x) = \frac{1}{5+4\cos(x)}$  với  $x \in [a,b]$ .

➤ Vẽ đồ thị và các đạo hàm bậc 1, bậc 2. miền xác định mặc nhiên trong symbolic Matlab là  $-2\pi \leq x \leq 2\pi$ .

Hãy vào cửa sổ soạn thảo và trong cửa sổ này ta viết chương trình như sau:

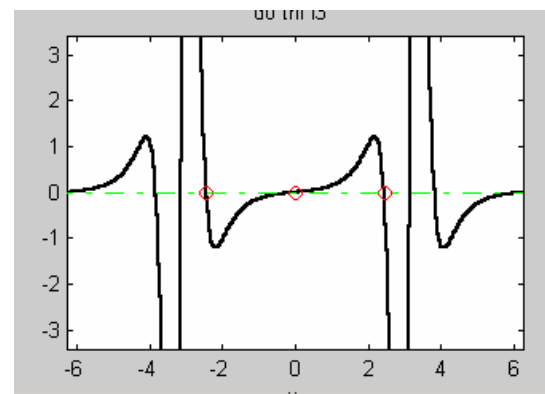
```
syms x
f1 = 1/(5+4*cos(x));
f2 = diff(f1);
% đạo hàm bậc 1 của f1.
f3 = diff(f2);
% đạo hàm bậc 2 của f1.
subplot(2,2,1)
ezplot(f1)
subplot(2,2,2)
ezplot(f2)
subplot(2,2,3)
ezplot(f3)
```



➤ Tìm các điểm x làm cho đạo hàm bậc 3 của hàm  $f(x) = \frac{1}{5+4\cos(x)}$  bằng không và vẽ đồ thị.

Hãy vào cửa sổ soạn thảo và trong cửa sổ này ta viết chương trình như sau:

```
syms x
f = 1/(5+4*cos(x));
f3 = diff(f,3);
% đạo hàm bậc 3 của f.
pretty(f3);
% đưa về dạng quy tắc toán học
f3 = simplify(f3);
pretty(f3);
z = solve(f3);
% Giải phương trình f3 = 0
format;
% lấy 5 chữ số thập phân
zr = double(z);
% chuyển mã trận, biểu thức symbolic về dạng số
ezplot(f3)
% vẽ hàm f3
hold on;
% lưu đồ thị đã vẽ
```



```
plot(zr,0*zr,'ro')
% ve diem "o" mau do cho cac diem co gia tri thuc de f3 =
0.
plot([-2*pi,2*pi],[0,0],'g-.')
```