Phần 1
cơ sở Mat lab

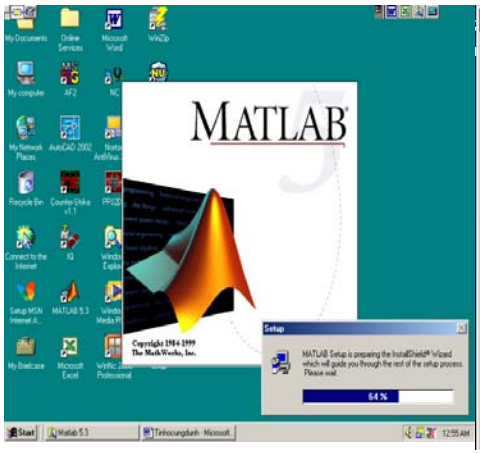
Chương 1:
Cài đặt matlab

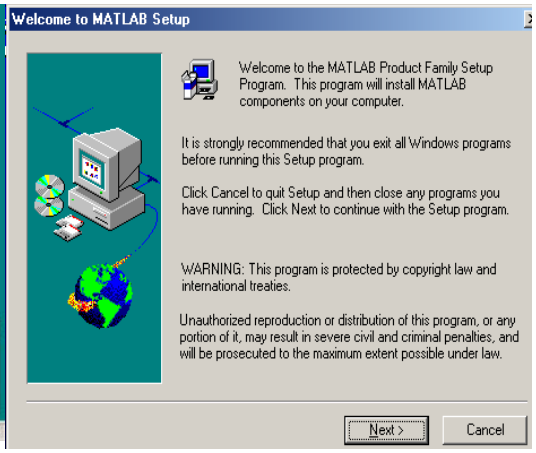
1.1.Cài đặt chương trình:

Qui trình cài đặt Matlab cũng tương tự như việc cài đặt các chương trình phần mềm khác, chỉ cần theo các hướng dẫn và bổ xung thêm các thông số cho phù hợp.

1.1.1 Khởi động windows.

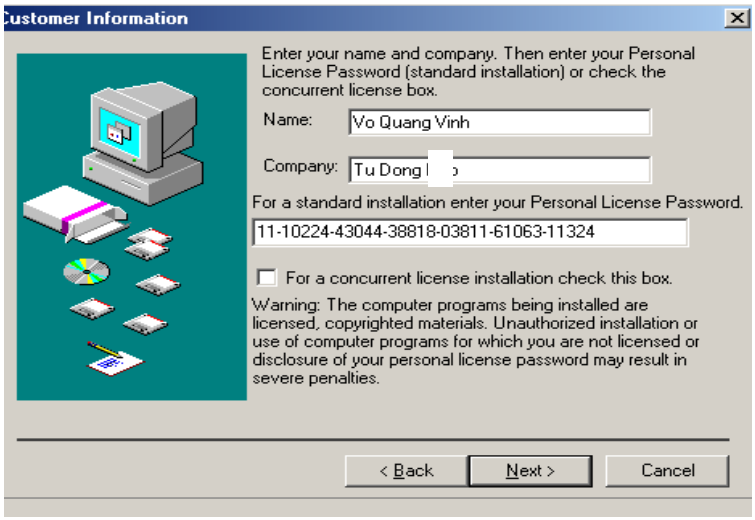
1.1.2 Do chương trình được cấu hình theo Autorun nên khi gắn dĩa CD vào ổ đĩa thì chương trình tự hoạt động, cửa sổ đón mừng hiện lên trong giây lát. Kích vào Next



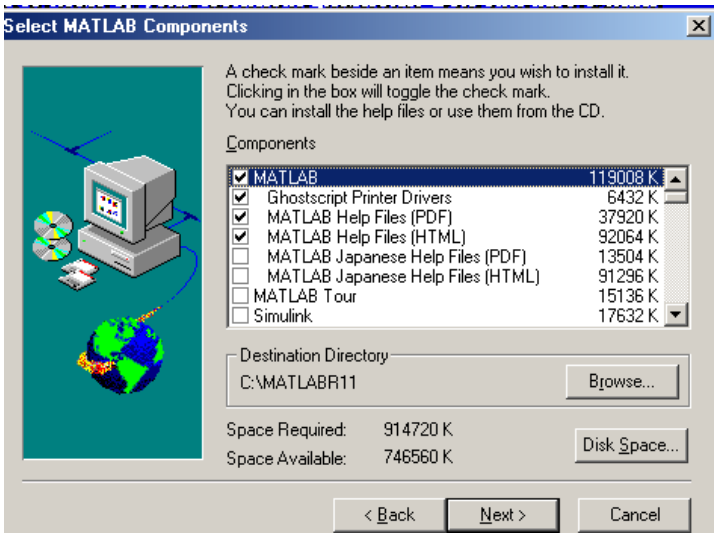


chuyển sang cửa sổ cài đặt kế tiếp.

1.1.3 Cửa sổ thứ hai chỉ các thông tin về bản quyền của chương trình. Kích Yes để sang cửa sổ cài đặt kế tiếp.



1.1.4 Trong cửa sổ Costumer Information, nếu cần thiết đánh tên họ, công ty. Nhập mã khoá của chương trình vào khung Persnal license password. Kích tiếp Next.



1.1.5 Trong cửa sổ select Matlab Components, kích xoá những thành phần không cần thiết sử dụng trong chương trình để dành thêm dung lượng trống. Muốn kiểm tra dung lượng của chương trình , kích vào Disk space để quan sát.

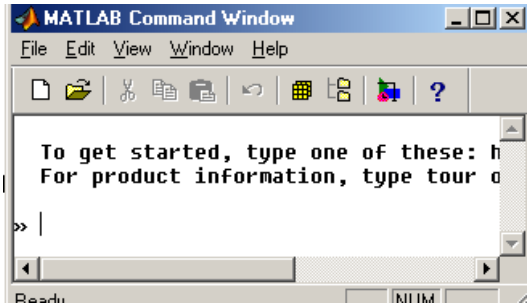
1.1.6 Sau khi kích Next, màn hình hiện khung thông tin Setup và chương trình bắt đầu cài đặt.

1.1.7 Sau khi hoàn thành Setup, màn hình hiện khung thoại Setup complete. Kích Finish để kết thúc qui trình cài đặt.

1.1.8 Sau khi kết thúc cài đặt cửa sổ kế tiếp là Internet explorer. Kích Close để về cửa sổ nền windows, biểu tượng tự động gán trong nền windows.

1.2. Khởi động và thoát khỏi chương trình

1.2.1 Khởi động MATLAB: Kích vào biểu tượng Matlab trên màn hình Windows hoặc kích:



Start/ Programs/ Matlab/ Matlab 5.3, sau đó hiện cửa sổ Matlab và cửa sổ lệnh (MATLAB Command Widowd)

Cũng như các chương trình chạy trong môi trường windows khác Matlab cũng có những thành phần giao diện của chương trình.

- Dòng trên cùng là thanh tiêu đề gồm:

  + Tận trái là biểu tượng chương trình. Khi kích vào biểu tượng này Matlab hiện menu xổ chứa các lệnh liên quan tới việc xử lý khung cửa sổ chương trình cũng như thoát chương trình.

  + Kế biểu tượng là tên chương trình cùng với cửa sổ chương trình. Tận cùng là ba biểu tượng có chức năng phong to, thu nhỏ và thoát chương trình.

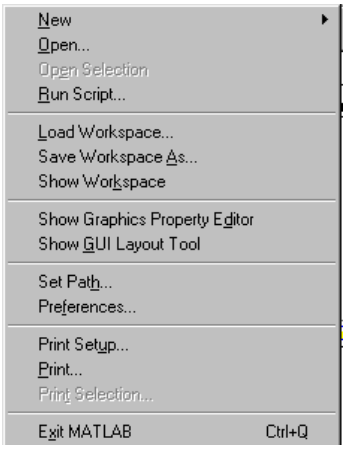
- Dòng thứ hai là thanh trình đơn gồm các menu chính chưa các lệnh liên quan đến việc tạo, xử lý, gán thuộc tính vào đồ thị.
- Dòng thứ ba là thanh công cụ chứa các hộp công cụ chứa các lệnh nhanh.
- Phần chiếm gần chọn màn hình là nơi nhập các lệnh .
- Cuối cùng là thanh tác vụ hiện thông tin về tình trạng đang xử lý, thực hiện đối tượng.

1.2.2 Thoát khỏi MATLAB: Từ cửa sổ Matlab command winwods đánh lệnh **quit** hoặc kích biểu tượng (X) nằm ngay góc phải thanh tiêu đề Matlab.

Chương 2

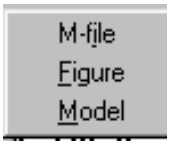
Các lệnh menu của MATLAB

2.1 File

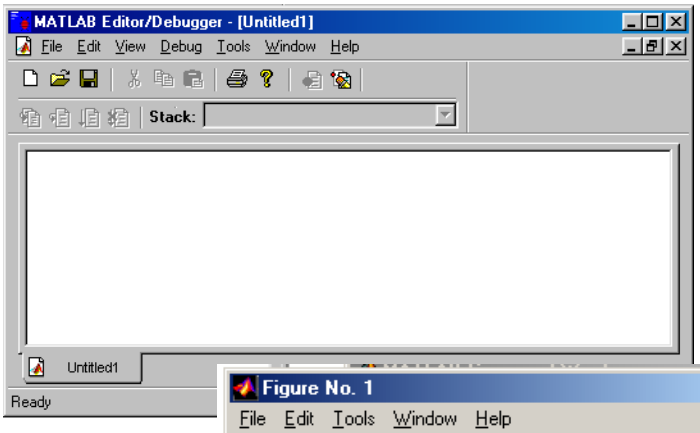


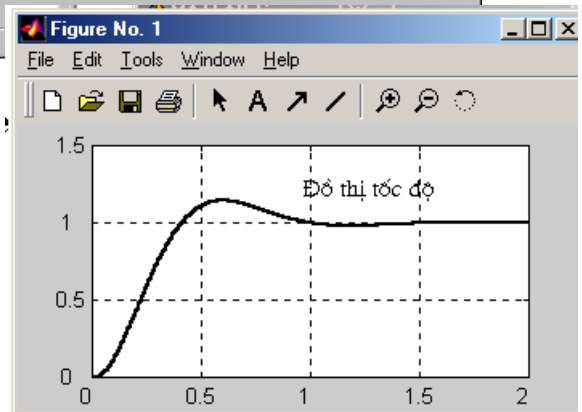
Kích vào File hiện menu chứa các lệnh liên quan đến việc tạo mới, quản lý, gán thuộc tính cho đối tượng, đồ thị cũ

2.1.1 New: Hiện menu chứa ba lện để tạo đối tượng mới



- **M-file:** Hiện cửa sổ **Editor/ Debugger**. Đây là môi trường bạn tạo vũng như sửa đổi, gỡ rối các tập tin chương trình như M-file hoặc MEX-file hoặc các đối tượng nào đó để thực hiện cho một nhiện

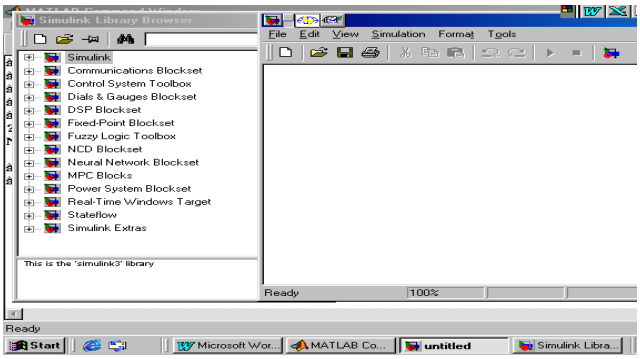




vụ nào đó. Để có thể tạo được các tập tin này, bạn phải dùng ngôn ngữ lập trình C hoặc FORTRAN

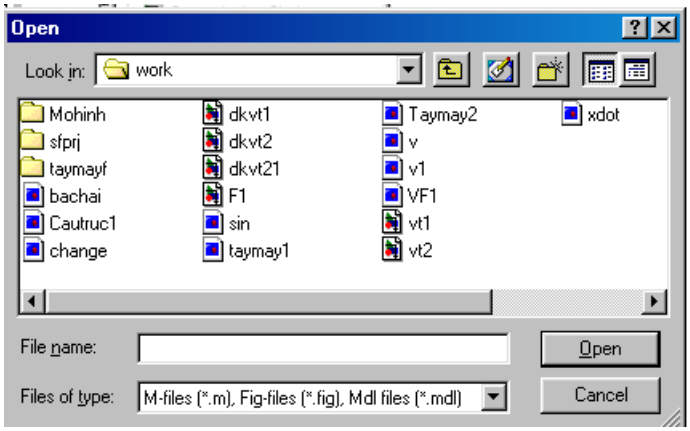
- **Figure:** Đây là môi trường đồ hoạ để bạn tự vẽ các đối tượng hoặc để Matlab vẽ các đồ thị theo hàm lệnh bạn nhập từ khung cửa sổ lệnh của MATLAB hoặc hiện đồ thị cũ với lệnh **Open** từ lệnh của Matlab.

- **Model :** Hiện cửa sổ Simulink nằm bên phải và cửa sổ thư viện nằm bên trái để ban tự chọn mà thiết kế đối tượng.



2.1.2 Open : Mở tập tin đồ thị hoặc hình ảnh trong cửa sổ **Figure** để sử lý.

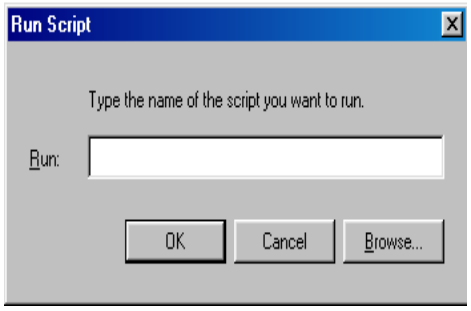
- Trong cửa sổ Matlab Command windows kích: **File/ Open**
- Từ khung thoại, kích chọn tên tập tin cùng thể loại tập tin (nếu cần thiết) muốn mở và kích OK.



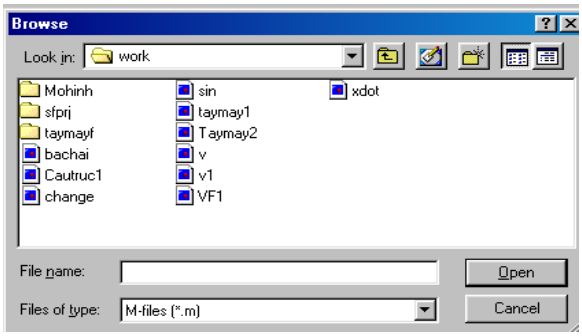
2.1.3 Open Selection:

Có chức năng mở những đối tượng đã chọn riêng và lưu thành một tập tin riêng

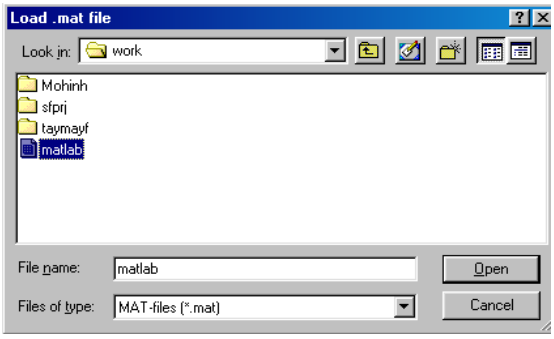
2.1.4 Run Script : Chạy một chương trình đã lưu thành một tập tin.



- Trong cửa sổ Matlab Command windows kích: **File/ Run Script.** Màn hình xuất hiện khung thoại **Run Script.**
- Đánh tên tập tin vào trong khung nhập lệnh **Run** và kích OK. Để tìm kiếm các tập tin đã lưu kích vào **Browse.**
- Từ khung thoại chọn nguồn chứa và tên tập tin cần mở rồi kích vào Open để về khung thoại Run Script với toàn bộ đường dẫn cùng tên tập tin muốn mở. Kích OK.



2.1.4 Load Workspace: Tải tên tập tin . **mat** lên màn hình.

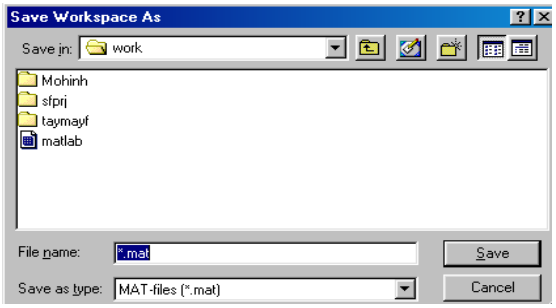


- Trong cửa sổ Matlab Command windows kích: **File/ Load Workspace.** Màn hình xuất hiện khung thoại **Load. Mat file.**
- Từ khung thoại, kích chọn tên tập tin và kích Open.

Bạn có thể dùng lệnh **Load Workspace** để du nhập các tập tin từ những nguồn chứa khác nhau bằng cách cũng trong khung thoại Load.mat file, kích vào mũi tên của Look in để tìm nguồn chứa, chọn tập tin muốn du nhập.

Ghi chú : *Để có thể du nhập các tập tin khác vào MTALAB, điều bắt buộc là các tập tin đó phải được soạn theo hệ mã ASCII.*

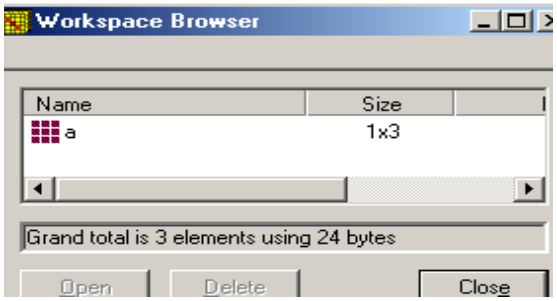
2.1.5 Save Workspace as : Lưu các nội dung trong cửa sổ lệnh của MATLAB theo workspace.



- Nhập các lệnh trong cửa sổ lệnh của MATLAB như bình thường.
- Từ menu File, chọn Save Workspace as. Màn hình hiện khung thoại Save Workspace as.
- Từ khung thoại, đánh tên tập tin muốn lưu (nếu cần thiết chọn nguồn chứa với Save in). Kích Save. Nên nhớ là tập tin này được lưu theo dạng *.mat và cũng là tập tin chương trình gốc.

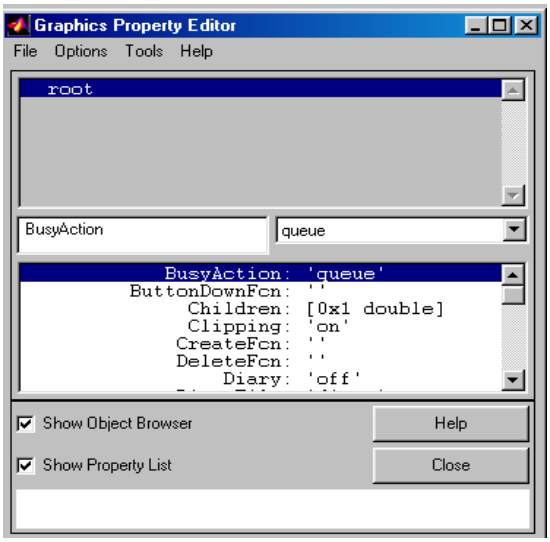
2.1.5 Show Workspace :

Hiện nội dung của tập tin workspace lên màn hình bằng cách từ menu File, chọn Save Workspace. Trong cửa sổ **Save Workspace** sẽ hiện nội dung của tập tin Workspace đã ghi.



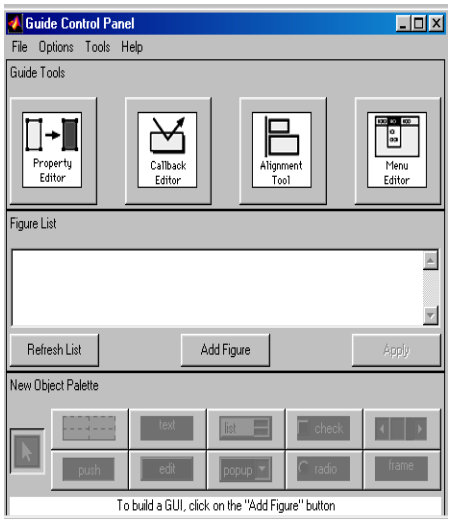
2.1.6 Show graphics proprty editor :

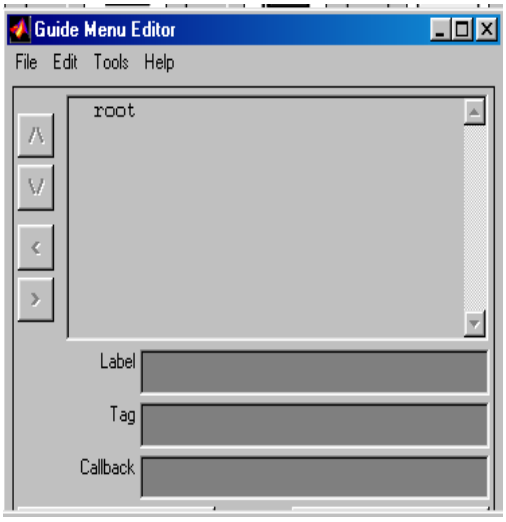
Hiện cửa sổ thay đổi thuộc tính mặc định của hình ảnh để áp dụng cho tất cả các hình ảnh sẽ được thiết kế sau này hoặc chỉ riêng theo từng nhiệm vụ.



- Từ cửa sổ lệnh MATLAB kích : **File/Show graphics proprty editor.** Màn hình xuất hiện cửa sổ thông tin Show graphics proprty editor.
- Chọn thuộc tính cần thay đổi và kích Close. Thí dụ, thay đổi thuộc tính Echo từ mặc định là **on** thành **off**

2.1.7 Show gui layout tool : Hiển thị danh mục các biểu tượng công cụ trong cửa sổ Guide Control Panel.

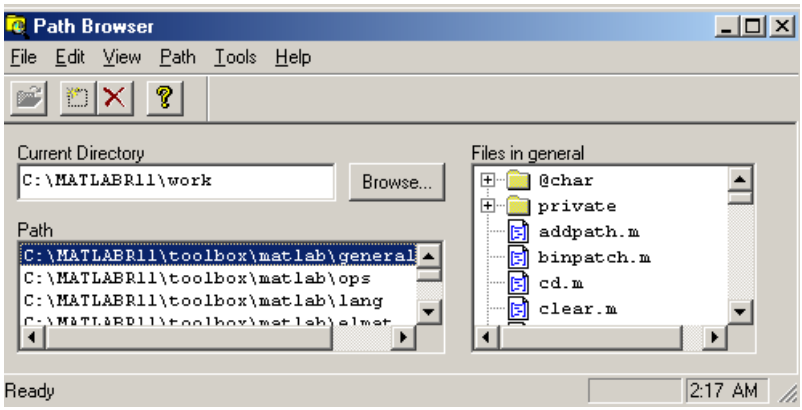




Từ cửa sổ này, bạn có thể thêm các biểu tượng công cụ tự tạo mới, gán biểu tượng biểu tượng công cụ vào đối tượng để thực hiện nhanh bằng cách kích chọn biểu tượng trong khung biểu tượng Guide tool để hiện thông tin và từ đây gán nhãn cũng như các thuộc tính khác cho công cụ vừa chọn và kích Apply.

2.1.8 Set path: Gán tập tin vào thư mục bất kỳ trong MATLAB.

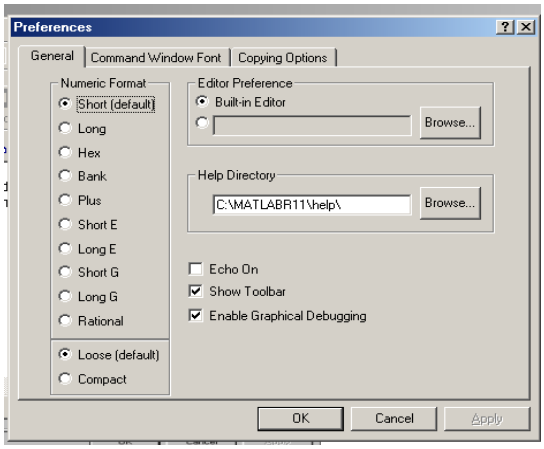
- Từ menu File chọn Set Path. Màn hình hiện cửa sổ Path Browser. Cửa sổ này liệt kê các danh sách các đường dẫn thư mục nằm trong khung cửa bên trái và danh sách tên tập tin nằm bên phải.



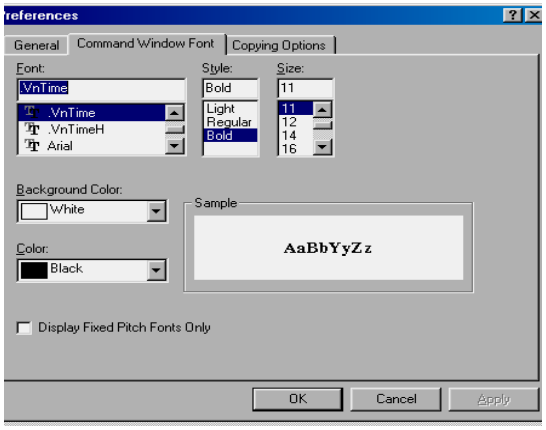
- Chọn tên đường dẫn thư mục trong khung cửa sổ Path và chọn tên tập tin muốn thay đổi vị trí chứa trong khung cửa sổ hoặc đánh trực tiếp vào khung nhập lệnh Current Directory.
- Từ menu File của cửa sổ Path Browser, chọn Save để lưu nhữnh thay đổi.

2.1.9 Preferenes: Hiện khung thoại với ba tuỳ chọn để người sử dụng có thể thay đổi tham số cho phù hợp nhiệm vụ.

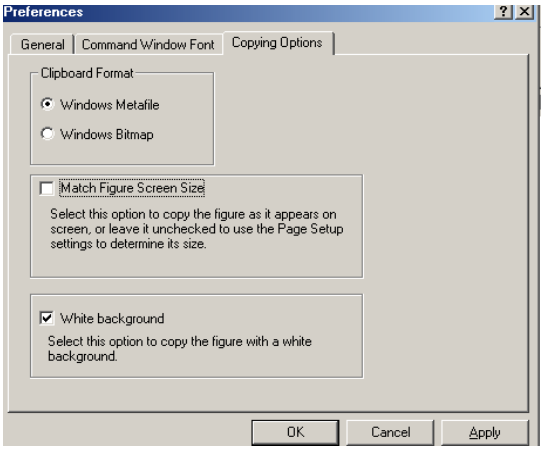
- **General:** Hiện khung thoại ngay khi kích chọn lệnh Preferences từ menu File của khung cửa sổ lệnh MATLAB. Kích chọn các loại tham số muốn thay đổi hoặc gán thêm và kích OK.



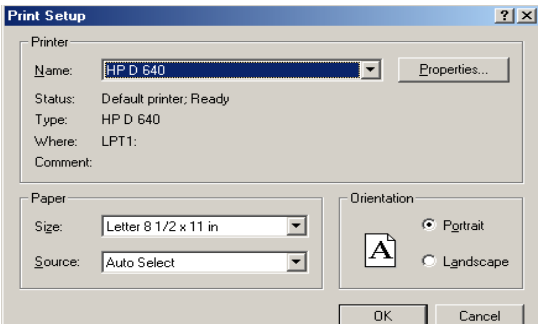
- **Command Windows Font:** Hiện khung danh mục font cùng thuộc tính để người sử dụng thay đổi phông mặc định thành font quên thuộc.



- **Copy Options :** Hiện khung thông tin để người sử dụng tuỳ chọn những thành phần đối tượng muốn sao chép.



2.1.9 Print Setup :

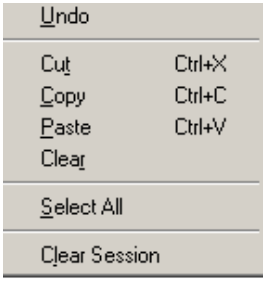


Hiện khung thoại Print cùng các chức năng phụ liên quan đến việc in.

2.1.10 Exit MATLAB (Ctrl + Q) :

Thoát khỏi chương trình MATLAB để về lại màn hình nền hệ điều hành windows.

2.2 Edits : Hiện menu con chứa các lệnh kiên quan đến việc xử lý các đối tượng.



2.2.1 Undo : Huỷ lệnh đã thực hiện trước đó. Sau khi chọn lệnh Undo để huỷ lệnh, lệnh sẽ đổi thành Redo để người sử dụng khôi phục những gì đã huỷ trươc đó với lệnh Undo.

2.2.2 Cut (Ctrl +X) : Cắt đối tượng trong khung cửa sổ lệnh hiện hànhvà sai đó dán vào vị trí đã chọn. Đối tượng được cắt có thể là một công thức, một chuỗi ký tự, hàm lệnh□. .

- Chọn đối tượng muốn cắt trong khung cửa sổ lệnh MATLAB và ấn Ctrl + C hoặc chon Copy từ menu Edit. Thí dụ công thức.
- Chọn vị trí bất kỳ để dán tạmvà ấn Ctrl + V hoặc Paste. Đối tượng sẽ được dán vào vị trí đã chọn . Nên nhớ đây chỉ là phiên bản đã được sao chép và dán.
- Chọn vị trí thực sự muốn dán. Kích lênh Cut hoặc ấn Ctrl +X, đối tượng vừa dán lúc trước sẽ tạm biến mất tại vị trí hiện hành. ấn Ctrl + V để dán.

2.2.3 Copy (Ctrl + C) : Sao chép đối tượng trong khung cửa sổ lệnh MATLAB và sau đó dán vào vị trí đã chọn. Đối tượng được copy có thể là một công thức, một chuỗi ký tự, hàm lệnh □.

- Chọn đối tượng cần copy tại cửa sổ lệnh MATLAB và sau đó ấn Ctrl + V hoặc Copy. Thí dụ , chọn cônh thức.
- Chọn vị trí cần copy đến và âns Ctrl + V.

2.2.4 Paste (Ctrl + V) : Dán đối tượng đã được Cut hoặc Copy vào vị trí đã chọn. Ngoài ra , bạn còn có thể dùng lệnh Paste của MATLAB để dán các đối tượng khác vào MATLAB.

2.2.5 Clear : Xoá đối tượng đã được chọn trong khung cửa sổ MATLAB.

2..2.6 Select Att : Chọn toand bộ nội dung trong khung cửa sổ lệnh của MATLAB.

2.2.7 Clear Session : Xoá toàn bộ nội dung của cửa sổ lệnh MATLAB sau khi chọn với lệnh **Select Att.**

2.3 View : Đánh dấu chọn hoặc xoá để hiện hay giấu thanh công cụ trong khung cửa sổ lệnh MATLAB.

2.4 Window : Hiện thông tin về số tập tin đã thực hiện trong khung cửa sổ lệnh MATLAB.

Chương 3

Các khái niệm cơ bản

MATLAB (Matrix laboratory) là1 phần mềm ứng dụng chạy trong trong môi trường Windows. Dùng để mô phỏng các hệ thống điều khiển dưới dạng hệ phương trình trạng thái hoặc sơ đồ cấu trúc. Thực hiện các phép toán về xử lý ma trận, xử lý tín hiệu cũng như xử lý đồ hoạ□

3.1 Một số phím chuyên dụng và lệnh thông dụng :

- ↑ hoặc Ctrl + p : Gọi lại các lệnh đã thực hiện trước đó.
- ↓ hoặc Ctrl +n : Gọi lại lệnh vừa thực hiện trước đó.
- → hoặc Ctrl + f : chuyển con trỏ sang bên phải 1 ký tự.
- ← hoặc Ctrl + b: chuyển con trỏ sang trái một ký tự.
- Dấu (;) để kết thúc một dòng lệnh
- ↵nhảy xuống dòng dưới
- Ctrl + A hoặc Home : chuyển con trỏ về đầu dòng.
- Ctrl + E hoặc End: Chuyển con trỏ đến cuối dòng.
- BackSpace: Xoá ký tự bên trái con trỏ.
- Esc: xoá dòng lệnh.

- Ctrl + K : Xoá từ vị trí con trỏ đến cuối dòng.
- Ctrl + C : Dừng chương trình đang thực hiện.
- Clc : lệnh xoá màn hình.
- Clf: Lệnh xoá màn hình đồ hoạ.
- Input: lệnh nhập dữ liệu vào từ bàn phím.
- Demo: lệnh cho phép xem các chương trình mẫu.
- Help: lệnh cho phép xem phần trợ giúp.
- Ctrl — c: Dừng chương trình khi nó bị rơi vào trạng thái lặp không kết thúc.
- Dòng lệnh dài: Nếu dòng lệnh dài quá thì dùng □ ↵ để chuyển xuống dòng dưới.

3.2 Biến trong MATLAB

3.2.1Tên: Là một dãy ký tự bao gồm các chữ cái hay các chữ số hoặc một số ký tự đặc biệt dùng để chỉ tên của biến hoặc tên của hàm. Chúng phải được bắt đầu bằng chữ cái sau đó có thể là các chữ số hoặc một vài ký tự đặc biệt. Chiều dài tối đa của tên là 31 ký tự.

Bình thường Matlab có sự phân biệt các biến tạo bởi chữ cái thường và chữ cái hoa. Các lệnh của Matlab nói chung thường sử dụng chữ cái thường. Việc phân biệt đó có thể được bỏ qua nếu chúng ta thực hiện lệnh : >> **casensen off**

3.2.2 Một số lệnh với biến:

- clear: lệnh xoá tất cả các biến đã được định nghĩa trước trong chương trình .
- clear biến1, biến 2... : xoá các biến được liệt kê trong câu lệnh.
- Who: hiển thị các biến đã được định nghĩa trong chương trình.

- Whos: hiển thị các biến đã được định nghĩa trong chương trình cùng với các thông số về biến.
- Size (tên biến đã được định nghĩa): cho biét kích cỡ của biến dưới dạng ma trận với phần tử thứ nhất là số hàng của ma trận, phần tử thứ 2 là số cột của ma trận.
- Save: Lưu giữ các biến vào một File có tên là Matlab. mat.
- Load: Tải các biến đã được lưu dữ trong một File đưa vào vùng làm việc.

3.2.3 Một số biến đã được định nghĩa trước:

- **ans**: tự động gán tên này cho kết quả của một phép tính mà ta không đặt tên.

VD >> [1 2] ↵

ans =

2

- **pi**
- **realmax**: đưa ra giá trị của số lớn nhất mà máy tính có thể tính toán được.
- **realmin**: đưa ra giá trị của số nhỏ nhất mà máy tính có thể tính toán được.

3.3 Các phép toán vô hướng:

3.3.1 Biểu thức số học:

Matlab có thể thực hiện tất cả các biểu thức toán học thông qua các toán tử sau:

+ phép cộng.

- Phép trừ.

* phép nhân.

/ Phép chia phải

\ phép chia trái

^ phép luỹ thừa.

= để gán một giá trị nào đó cho một biến.

3.3.2 Thứ tự ưu tiên trong phép toán số học:

ngoặc đơn.

luỹ thừa

nhân, chia.

Cộng ,trừ.

3.3.3 Các ví dụ:

- **Ví dụ 1**: Giải phương trình bậc hai $ax^2 + bx + c = 0$

Ta biết các nghiệm của phương trình này có dạng:

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

Vì Matlab là một chương trình tính toán số nên chúng ta phải xác định các giá trị a, b, c.

Dấu = được sử dụng để gán giá trị của a, b, c như sau (gõ phím Enter ở cuối mỗi hàng)

```
>>a = 2
a =
     2
>>b = 5;
>>c = -3;
```

⇒ ***Dấu (;) ở cuối dòng thì Matlab sẽ không hiển thị lại giá trị vừa nhập.***

```
>> x1= (-b + sqrt(b^2- 4*a*c))/(2*a)
x1 =
      0.5000
```

```
>> x2= (-b - sqrt(b^2- 4*a*c))/(2*a)
x2 =
     -3
```

- **Ví dụ 2**: Tính giá trị của đa thứ.

```
>> a = x^3 -2*x^2 - 6;
>>b = x^2 + 5*x -7;
>>x=3;
>> w = a/b
w =
   0.1765
```

3.4Sử dụng lệnh từ các file lệnh:

Những lệnh của Matlab có thể được đưa vào một file. Sau đó bạn sẽ hướng dẫn Matlab làm việc với các lệnh đó. Bây giờ, với ví dụ 1, chúng ta sẽ đưa toàn bộ các lệnh trên vào một file lấy tên là vidu.m. Tên của file phải được bắt đầu bằng một ký tự và phần mở rộng là .m. Các bước như sau:

Bước 1: File / New/ M-file, Môi trường soạn thảo Editor / Debugger sẽ xuất hiện

Bước 2: Trên màn hình soạn thảo, ta gõ các lệnh của Matlab.

```
a = 2;
b = 5;
c=-3;
x1= (-b + sqrt(b^2- 4*a*c))/(2*a)
x2= (-b - sqrt(b^2- 4*a*c))/(2*a)
```

Bước 3: Ghi lại nội dung tập tin với tên vidu.m rồi thoát khỏi môi trường soạn thảo để trở về cửa sổ lệnh (Matlab Command window.

Bước 4: Tại cửa sổ lệnh ta gõ tên tập tin.

```
>>vidu.m↵
```

- ***Các file *.m có 2 loại:***
  - Script file: là một chương trình con không có giá trị trả về.
  - Function file: là một chương trình con có giá trị trả về.

3.5 Dòng nhắc gán giá trị các biến:

Để thay đổi các giá trị a,b,c ta phải soạn thảo lại file vidu.m rồi chạy lại. Ta sửa lại chương trình để có dòng nhắc nhập a,.b,c với các lần chạy chương trình khác nhau.

Bước 1: File / New/ M-file, Môi trường soạn thảo Editor / Debugger sẽ xuất hiện

Bước 2:

```
a=input('nhap a= ');
b=input('nhap b= ');
c=input('nhap c= ');
x1= (-b + sqrt(b^2- 4*a*c))/(2*a)
x2= (-b - sqrt(b^2- 4*a*c))/(2*a)
```

Bước 3: Ghi lại nội dung tập tin với tên vidu.m.

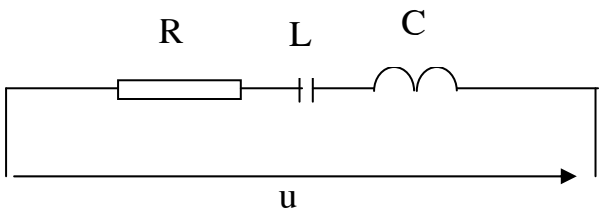
Khi chạy chương trình ta thu được kết quả như sau:

Bước 4: Tại cửa sổ lệnh ta gõ tên tập tin.

```
>>vidu↵
nhap vao a= 1
nhap vao b= -2
nhap vao c= 5
x1 =
1.0000  + 2.0000i
x2 =
1.0000 — 2.0000i
```

Hai nghiệm ứng với các gia trị a,b,c vừa nhập vào và là nghiệm ảo.

VD về Script file: Giải bài tập mạch: cho mạch điện như hình vẽ



Hãy tính dòng trong mạch và điện áp trên từng phần tử .

Hãy vào cửa sổ soạn thảo và trong cửa sổ này ta viết chương trình như sau:

```
R=input( 'nhap gia tri cho R = ')
C=input( 'nhap gia tri cho C = ')
L=input( 'nhap gia tri cho L = ')
U=input( 'nhap gia tri cho U = ')
ZL=2*50*pi*L*i
ZC=1/(2*50*pi*C*i)
Z=R+ZL+ZC
i= U/Z
UR=i*R
UL=i*ZL
UC=i*ZC
```

Sau khi viết xong chương trình ta kích vào biểu tượng save trong cửa sổ soạn thảo và tên là vd1.

Muốn chạy ta trở lại cửa sổ MATLAB command Window và từ dấu nhắc lệnh:

```
>> vd1
nhap gia tri cho R = 1000
R =       1000
nhap gia tri cho C = 0.1
C =    0.1000
nhap gia tri cho L = 0.2
L =    0.2000
nhap gia tri cho U = 220
U =   220
ZL =       0 +62.8319i
ZC =       0 - 0.0318i
```

```
Z =  1.0000e+003 +6.2800e+001i
i =   0.2191 - 0.0138i
UR =  2.1914e+002 -1.3762e+001i
UL =   0.8647 +13.7687i
UC =  -0.0004 - 0.0070i
```

3.6Cách tạo một hàm:

Mỗi một file hàm của Matlab (M - file) đều được khai báo như sau:

Function [Tên kết quả] = tên hàm (danh sách các biến).

Phần thân của chương trình trong hàm là các lệnh của Matlab thực hiện việc tính toán giá trị của đại lượng được nêu trong phần tên kết quả theo các biến được nêu tronhg phần danh sách biến. Các biến chỉ có tác dụng nội trong hàm vừa được khai báo Tên của các biến dược cách nhau bằnh dấu phẩy (,).

Ví dụ ta thành lập hàm đổi từ độ sang radian:

```
function rad = change(do)
rad = do*pi/180; % doi do sang radian
```

Trong Matlab các dòng ghi chú sau dấu % không có tác dụng thực thi, chúng đơn giản là những dòng nhắc để người đọc chương trình dễ hiểu mà thôi.

File.m thường lấy tên là tên của hàm, ta đặt tên file hàm vừa lập là change.m.

Nếu muốn đổi 45^0 sang radian, chỉ cần gõ:

```
>>rad = change(45)
rad =
    0.7854
```

Ví du: tạo hàm giải phương trình bậc hai, tên tập tin được đặt là bachai.m.

```
function [x1,x2] = bachai(a,b,c)
delta = b^2-4*a*c;
x1 = (-b + sqrt(delta))/(2*a);
```

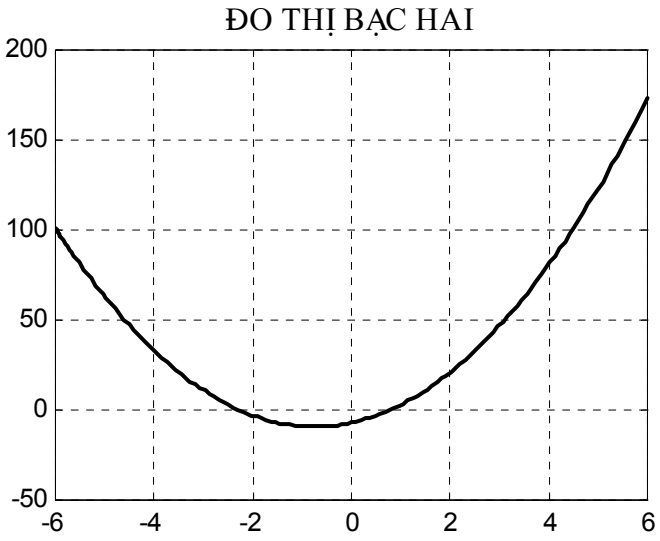
```
x2 = (-b - sqrt(delta))/(2*a);
>>[x1,x2]=bachai1(4,6,-7)
x1 =
      0.77707
x2 =
      -2.2707
```

3.7Vẽ các hàm .

Khi muốn vẽ một hàm nào đó, phải xác định hàm đó trong một file.m sau đó sử dụng lệnh :

Fplot('tên hàm',[khoange vẽ])

Ví dụ vẽ hàm $y = 4x^2+6x-7$ trong đoạn [-6, 6], ta lập file bachai1.m.




```
function y = bachai1(x)
a = 4;b = 6; c = -7;
y =a*x^2 + 6*x + c;
>>fplot('bachai1', [-6,6])
```

3.8Lưu và lấy dữ liệu:

Ta có thể có thể tạo lập một file dữ liệu, sau đó khi cần dùng thì lấy ra. Ví dụ tạo lập một ma trận A:

```
1   1.1   1.2
2   2.1   2.2
3   3.2   3.2
```

Sau đó ta lưu ma trận vào một file có tên là dulieu1.

```
>>A = [1  1.1  1.2;2  2.1  2.2;3  3.2  3.2]
 A =
```

1.0000 1.1000 1.2000

2.0000 2.1000 2.2000

3.0000 3.2000 3.2000

>>save dulieu1

Như vậy, ta đã có một file dữ liệu (file này nằm trong thư mục work của Matlab). Khi cần sử dụng file dữ liệu này, ta lấy như sau:

>>load dưlieu

Sau lệnh load, ta có thể lấy dữ liệu để sử dụng:

>>A

A =

1.0000 1.1000 1.2000

2.0000 2.1000 2.2000

3.0000 3.2000 3.2000

3.9 số phức và các phép toán về số phức.

Matlab chấp nhận số phức biểu thị bởi hàm đặc biệt i và j, cả hai là toán tử và các hàm.

3.9.1 Biểu diễn số phức:

Tên = Phần thực + phần ảo i hoặc Tên = Phần thực + phần ảo j

Vidụ:

>> a = 3 +4i ↵

3.9.2 Các phép toán đối với số phức:

- **cộng, trừ, nhân số phức:**

C1= a1+b1i ; C2=a2+b2i

Cộng : c1+c2

Trừ : c1 - c2

Nhân: c1*c2

Chia: c1/c2

- **Các hàm đặc biệt:**

real(x) : tìm phần thực của số phức x.

imag(x): tìm phần ảo của số phức x.

conj(x): tìm số phức liên hợp của số phức x.

abs(x): Tìm giá trị tuyệt đối của số phức x (độ lớn)

angle(x): góc tạo bởi giữa trục thực và ảo.

Ví dụ:

```
>> a =3.0000 + 4.0000i
a =
    3.0000 + 4.0000i
>> thuc=real(a);
 >> ao=imag(a);
 >> lienhop=conj(a);
 >> dolon=abs(a);
 >> goc =angle(a);
```

3.10 Các phép toán quan hệ và phép toán logic

3.10.1 Các phép toán quan hệ:

Các phép toán quan hệ bao gồm:

- Nhỏ hơn: <
- Nhỏ hơn hoặc bằng: <=
- Lớn hơn: >
- Lớn hơn hoặc bằng: >=
- Bằng: ==
- Không bằng (khác): ~=

Biểu thức có các toán tử quan hệ nhận gia trị đúng là (true) hoặc sai (false). Trong Matlab, biểu thức đúng sẽ có giá trị là 1, biểu thức sai có gia trị là 0.

Ví dụ1:

```
>>12.2>12
ans =
     1
>>1~=1
```

```
ans =
     0
 >> A=[ 1:3;4:6;7:9]
A =
    1    2    3
    4    5    6
    7    8    9
>> B=[1:3;2:4;10:12]
B =
     1    2    3
     2    3    4
    10   11   12
>> A==B
ans =
     1    1    1
     0    0    0
     0    0    0
```

Ví dụ 2:

Cho một quả cầu mặng có khối lượng là m = 2kg được buộc vào một sợi dây có chiều dài

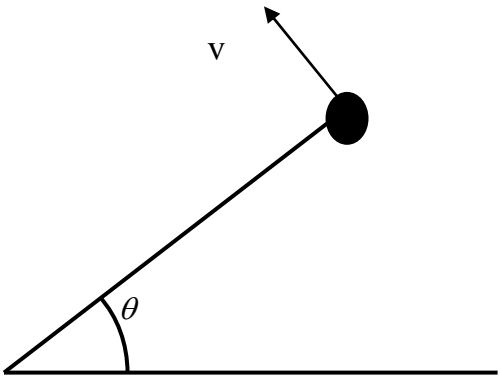
l = 1.5m. Một người nắm vào đầu dâyvà quay tròn quả cầu theo phương thẳng đứng.Vận tốc dài của quả cầu ở vị trí cao nhất là v =6m/s. lực kéo T trên sợi dây thay đổi theo góc nghiêng của dây với phương nằm ngang xác định theo công thức sau:

$$T = \frac{mv^2}{l} + mg(2 - 3\sin\theta)$$

Cho biết khi lực kéo T> 145N thì sơi dây bị đứt . Hãy xác định những vị trí dây bị đứt.

Chương trình như sau:

```
function y = tinhluc(theta,T)
m=2;l=1.5;Tmax=145;v=6;g=9.8;
theta=0:10:360;
```



```
rad=pi*theta/180;
T=m*(v^2/l+g*(2-3*sin(rad)));
bang=[theta;T>Tmax]
plot(theta,T);
grid on
>>Tinhluc
bang =
 Columns 1 through 12
    0   10   20   30   40   50   60   70   80   90  100  110
    0    0    0    0    0    0    0    0    0    0    0    0
 Columns 13 through 24
  120  130  140  150  160  170  180  190  200  210  220  230
    0    0    0    0    0    0    0    0    0    0    0    0
 Columns 25 through 36
  240  250  260  270  280  290  300  310  320  330  340  350
    0    0    1    1    1    0    0    0    0    0    0    0
 Column 37
  360
    0
```

Từ kết quả trên ta thấy tại các vị trí ứng với các góc $260^{0,}$, $270^{0,}$, $280^{0,}$ thì biểu thức logic T>Tmax có giá trị là 1, tức tại đó dây có thể sẽ bị đứt.

3.10.2 Các phép toán logic

Các phép toán logic và, hoặc, đảo được thực hiện bởi các toán tử sau:

- Phép và(and): **Ký hiệu là &**

VD: phép & 2 ma trận cùng cỡ A, B là một ma trận có các phần tử bằng 1 nếu các phần tử tương ứng của cả 2 ma trận đầu đều khác 0 và bằng 0 nếu 1 trong 2 phần tử tương ứng của 2 ma trận bằng 0.

```
>>A=[1 2 7; 0 4 9;1 3 5];
 >> B=[0 2 4; 2 4 6; 3 0 7]
>> C=A&B
                C =
                   0     1     1
```

0 1 1

1 0 1

- Phép hoặc (or) : **Ký hiệulà |**

VD : phép or 2 ma trận cùng cỡ A,B là một ma trận có các phần tử bằng 0 nếu các phần tử tương ứng của cả 2 ma trận đầu đều bằng 0 và bằng 1 nếu 1 trong 2 phần tử tương ứng của 2 ma trận khác 0.

```
>>A=[0 2 7; 0 4 9;1 3 0];
>> B=[0 2 4; 2 4 6; 3 0 0];
>> C=A | B
```

C =

0 1 1

1 1 1

1 1 0

- Phép đảo(not): ***Ký hiệu là ~***

Ví Dụ : phép đảo của một ma trận là một ma trận có cùng cỡ với các phần tử có giá trị bằng 1 nếu các phần tử của ma trận đầu có giá trị bằng 0 và bằng 0 nếu các phần tử của ma trận đầu có giá trị khác 0.

```
>>A=[0 2 7; 0 4 9;1 3 0]
>> B=~A
```

B =

1 0 0

1 0 0

0 0 1

3.10.3 Cấu trúc câu lệnh điều kiện:

- ***Lệnh if đơn:***

Cú pháp: **if <biểu thức logic>**

Nhóm lệnh;

End

Nếu biểu thức logic đúng nhóm lệnh sẽ được thực hiện . Nếu biểu thức logic sai thì chương trình chuyển đến lệnh sau end.

VD:

```
function y=f(a,b,c)
    if a<b
disp(' hinh thang nguoc')
    end
y=(a+b)*c/2;
```

Sau đó ghi tên là ht

Quay lại cửa sổ MatLab command Window:

```
>> ht(2,4,2)
 hinh thang nguoc
ans =
   6
```

- ***Cấu trúc lệnh if lồng nhau:***

Cú pháp: **if <biểu thức logic1>**

Nhóm lệnh 1;

if <biểu thức logic 2>

Nhóm lệnh 2;

End

Nhóm lệnh 3;

End

Nhóm lệnh 4;

Nếu biểu thức logic 1 đúng thì

- Thực hiện nhóm lệnh 1.
- Kiểm tra biểu thức logic 2.
  - Nếu đúng thực hiện nhóm lệnh 2
  - Nếu sai bỏ qua nhóm lệnh 2
- Thực hiện nhóm lệnh 3.

Nếu biểu thức logic 1 sai thì

- Thực hiện nhóm lệnh 4.

- ***Lệnh else:***

Cú pháp: **if <biểu thức logic >**

Nhóm lệnh A;

else

Nhóm lệnh B;

End

Nhóm lệnh A sẽ được thực hiện nếu biểu thức logic đúng. Nếu không nhóm lệnh B sẽ được thực hiện.

- ***Lệnh elseif***

Cú pháp: **if <biểu thức logic1 >**

Nhóm lệnh A;

Elseif< BT logic 2>

Nhóm lệnh B;

Elseif< BT logic 3>

Nhóm lệnh C;

.....

End

Nếu BT logic 1 đúng nhóm lệnh A sẽ được thực hiện. Nếu sai kiểm tra Btlogic 2, nếu đúng thực hiện nhóm lệnh B. Nếu sai kiểm tra BT logic3,nếu đúng thực hiện nhóm lệnh C□Nếu không có biểu thức logic nào đúng thì không có lệnh nào trong cấu trúc trên được thực hiện.

- ***Kết hợp cấu trúc elseif và else***

Cú pháp: **if <biểu thức logic1 >**

Nhóm lệnh A;

Elseif< BT logic 2>

Nhóm lệnh B;

Elseif< BT logic 3>

Nhóm lệnh C;

.....

Else< BT logic n>

Nhóm lệnh n;

End

Nếu BT logic 1 đúng nhóm lệnh A sẽ được thực hiện. Nếu sai kiểm tra Btlogic 2, nếu đúng thực hiện nhóm lệnh B. Nếu sai kiểm tra BT logic3, nếu đúng thực hiện nhóm lệnh C□Nếu không có biểu thức logic nào đúng thì nhóm lệnh n được thực hiện.

```
a=input(' vao a=')
b=input(' vao b=')
c=input(' vao c=')
d=b^2-4*a*c
if d < 0
  disp(' pt vo nghiem')
elseif d==0
  disp (' pt co nghiem kep')
  x12=-b/2*a
else
  disp (' pt co 2 nghiem phan biet')
  x1=(-b+sqrt(d))/2*a
  x2=(-b-sqrt(d))/2*a
end
```

- ***câu điều kiện và lệnh Break***

  cú pháp: if< biểu thức logic>,break, end

  thoát khỏi vòng lặp nếu điều kiện logic đúng. Ngược lại sẽ thực hiện lệnh tiếp theo trong vòng lặp.

3.10.4 Cấu trúc vòng lặp.

- ***vòng lặp for:***

  cú pháp: for chỉ số = biểu thức

  nhóm lệnh A;

  end

  qui tắc thực hiện:

- Chỉ số của vòng lặp phải là biến.
- Nếu ma trận biểu thức là rỗng thì không thực hiện vòng lặp.
- Nếu biểu thức là một đại lượng vô hướng vòng lặp thực hiện 1 lần và chỉ số nhận giá trị của đại lượng vô hướng.

- Nếu biểu thức là một vector hàng, số lần lặp bằng số phần tử trong véctor sau mỗi lần lặp chỉ số nhận giá trị tiếp theo của vector.
- Nếu biểu thức là một ma trận thì số vòng lặp bằng số cột của ma trận sau mỗi vòng lặp chỉ số sẽ nhận giá trị cột tiếp theo của ma trận
- Khi kết thúc vòng lặp chỉ số nhận giá trị cuối cùng.
- Có thể dùng toán tử **:** trong vòng lặp for i = chỉ số đầu: gia số: chỉ số cuối. Và số vòng lặp được tính ceil((chỉ số cuối-chỉ số đầu)/gia số+1)

```
n=input ('vao n=')
    for k=n
     s=1;
 s=s+k
    end
```

đặt tên là vd

```
>>vd
vao n=3
n =
     3
s =
     4
>>vd
vao n=[1 4 7]
n =
     1     4     7
s =
     2
s =
     5
s =
     8
```

```
>>vd
vao n=[1 2 5;5 7 8;9 5 6];
 s =
    2
    6
   10
s =
    3
    8
    6
s =
    6
    9
    7
```

- ***Vòng lặp While***

Cú pháp: While < biểu thức>

Nhóm lệnh A;

End

Nếu biểu thức đúng thì thực hiện nhóm lệnh A. Khi thực hiện xong thì kiểm tra lại điều kiện. Nếu điều kiện vẫn đúng lại thực hiện nhóm lệnh A. Nếu sai vòng lặp kết thúc.

3.10.5 Các lệnh break, return, error:

Lệnh **break**: kết thúc sự thự thi vòng lặp for hoặc while

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.**

Lệnh **error** (‘dòng nhắn): kết thúc thực thi lệnh và hiển thị dòng nhắn trên màn hình.

Ví dụ: Chọn một số dương bất kỳ. Nếu số đó là số chẵn thì chia hết cho hai. Nếu số đó là số lể thì nhân với 3 rồi cộng 1. Lặp lại quá trình đó cho đến khi kết quả là 1.

Chương trình:

```
while 1
```

```
  n=input ('Nhap vao mot so : ');
  if n<=0
    break
      end
  while n>1
    if rem(n,2)== 0% phan du cua n chia cho 2
      n=n/2
    else
      n= 3*n+1
    end
  end
end
```

Khi chạy chương trình ta sẽ thấy tác dụng của lệnh break (dừng chương trình khi nhập số âm hoặc số 0)

3.10.6 Biến toàn cục (global variables)

Matlab cho phép sử dụng cùng một biến cho các hàm hoặc giữa các hàm và chương trình chính của Matlab, điều này được thực hiện thông qua việc khai báo biến toàn cục:

Global tên1 tên2 tên3 □.

(Tên các hàm cách nhau bằng dấu khoảng trống, không sử dụng dấu phẩy).

Việc khai báo biến toàn cục phải được thực hiện ở chương trình chính hoặc ở file lệnh (script) hoặc ở file hàm (function) có sử dụng các biến. Biến toàn cục có tác dụng cho đến khi kết thúc quá trình tính toán hoặc khi toàn bộ workspace được xoá. Không được đưa tên biến toàn cục vào danh sách các đối số của hàm. Khi sử dụng biến toàn cục các lệnh sau tỏ ra rất cần thiết:

- Clear glolal : Lệnh này cho phép loại bỏ các biến toàn cục.
- Isglobal(Tên biến) : Lệnh này cho phép kiểm tra xem một biến nào đó có phải là biến toàn cục hay không. Nếu là biến toàn cục thì giá trị trả về sẽ là 1.

3.10.7 Một số hàm toán học:

Mục này chỉ giới thiệu một số hàm đơn giản và thướng gặp trong khi lập trình:

- sin: hàm sin
- cos: hàm cos
- tan: hàm tang
- asin: hàm arcsin
- acos: hàm arccos
- atan: hàm arctang

- sinh: hàm sin hyperbolic
- cosh: hàm cos hyperbolic
- tanh: hàm tang hyperbolic
- asinh: hàm arcsin hyperbolic
- acosh: hàm arccos hyperbolic
- atanh: hàm arctang hyperbolic
- abs: Lấy giá trị tuyệt đối hoặc độ lớn của số phức
- round: làm tròn đến số nguyên gần nhất
- fix: làm tròn hướng về không
- floor: làm tròn hướng về $-\infty$
- ceil : làm tròn hướng về $+\infty$
- rem: phần dư sau khi chia
- gcd: ước số trung lớn nhất
- lcm: bội số trung nhỏ nhất
- exp: luỹ thừa e
- log: logarit cơ số e
- log10: logarit cơ số 10

3.10.8 Định dạng số:

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 tuỳ theo yêu cầu cụ thể:

Ta lấy ví dụ với số 4/3

- format short (đây là chế độ mặc định):

a = 1.3333

- format short e

a = 1.3333e + 000

- format long

a = 1.33333333333333

- format long e

a = 1.33333333333333e + 000

- format bank

a = 1.33

- format hex

a = 3ff5555555555555

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 hoạ 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 goị 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 cáh 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 dung

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 môt công cu quan trong 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 mô 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 nghich 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 hạo 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 .

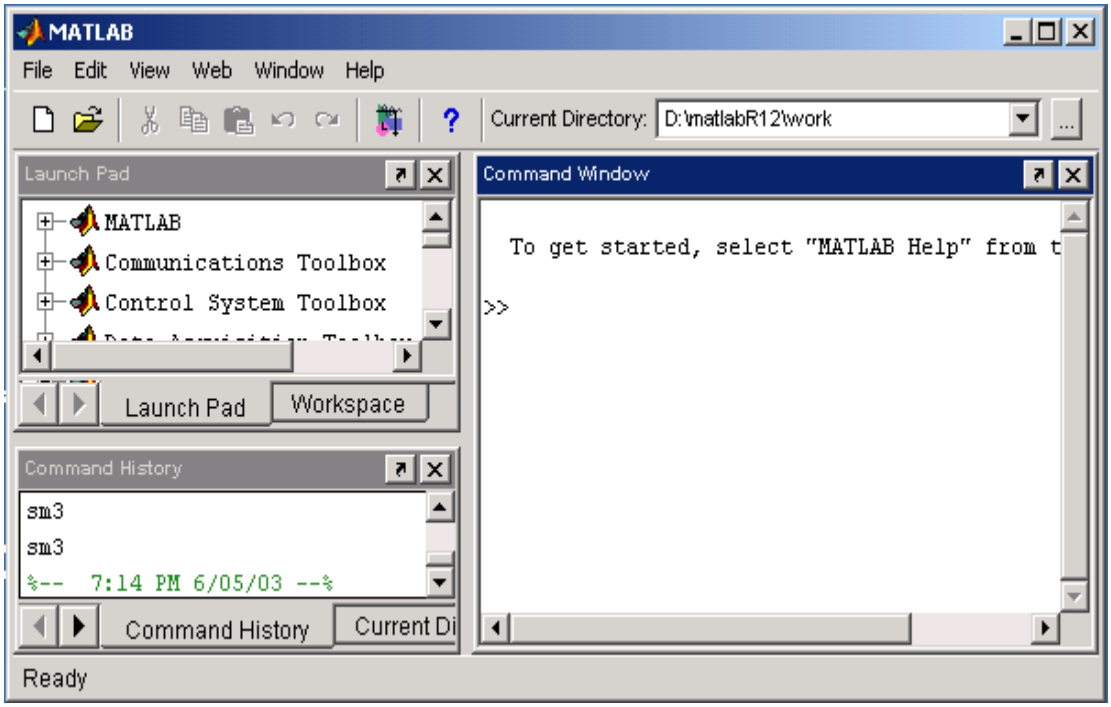
- Giao tiếp với các ngôn ngữ khác.

  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 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:

```
% Chuong trinh trong M-file
x= 0:pi/6:2*pi;
y=sin(x);
plot(x,y);
```

III.

IV. thực thi chương trình trên trong cửa sổ Command window bằng dòng lệnh sau

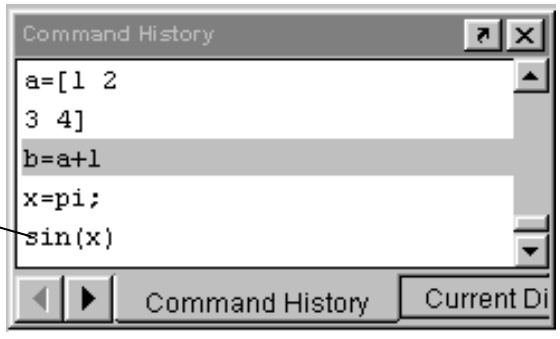
```
>> ve_sin↵
```

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 dung lai

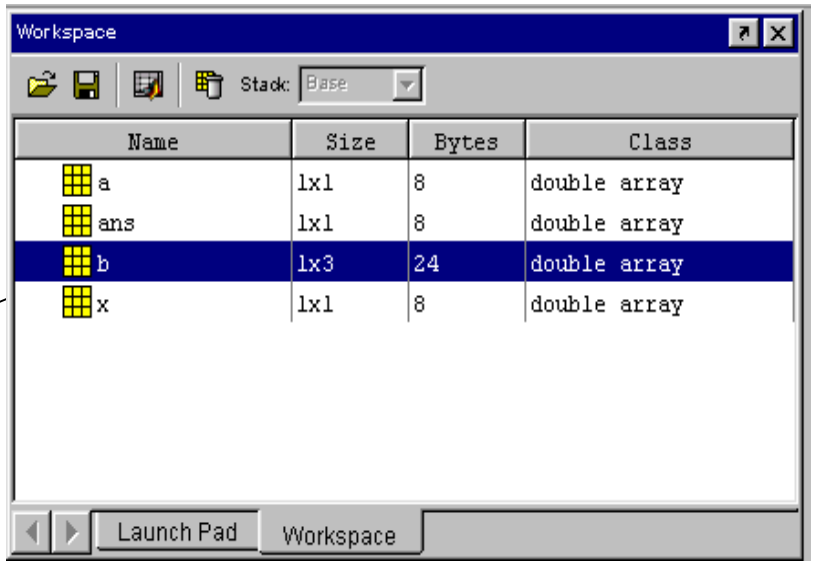


Hình 1.2

c. Cửa sổ Workspace:

Là cửa sổ thể hiện tên các biến bạn sử dụng cùng với kích thươc 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 Yêu



Hình 1.3

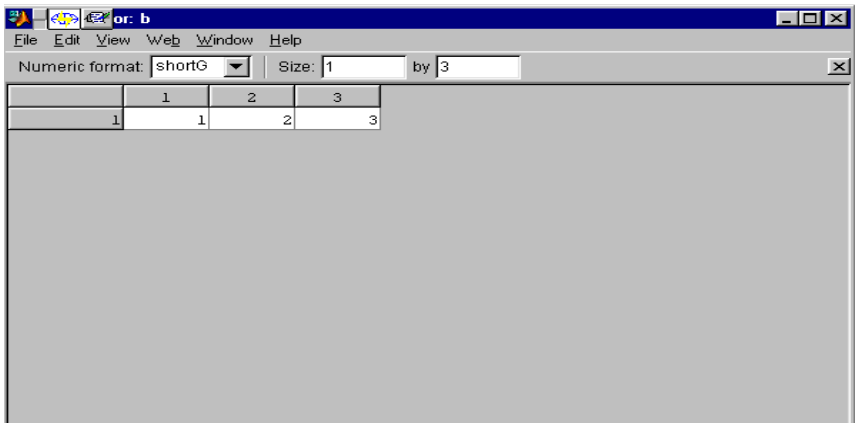
Ngoài ra nó cho phép thay đổi giá tri , cũng như kích thước của biến bằng cách kích đôi chuột lên các biến. Hoặc kích vào nút bên trái ngay cạnh nút save

Ví dụ khi chọn biến(giả thử là biến b) rồi kích đúp(hoặc kích chuột vào nút cạnh nút save) ta đươc 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 tuỳ theo ứng dụng cụ thể, tuỳ 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 matrận 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 tuỳ 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 tuỳ theo bạn muốn thể hiện kết quả của nó hay không.

a. Nhập các biến, matrận, 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 bảngsau

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\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 dụ1 :

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

Ví dụ2 Giải phương trình bậc hai, các lệnh nhập trong của 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:\matlabR12\ work\Tên_file), muốn thực thi chương trình soạn thảo đó ta gọi lệnh trong cửa sổ Commandwindow, tuỳ 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 ;

Ví dụ giải phương trình bậc hai tìm nghiệm x1 và x2 viết trong M-file dạng Scriptfile:

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

% lưu vào file GPTB2.m

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_1 và x_2

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

```
function [x1, x2]
=GPTB2(a,b,c)
   x1=(-b+sqrt(delta))/(4*a);
%Tinh nghiem x1
```

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 cửa 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 biên

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('Nhap gia tri cua a=');

Ví dụ: Trong cửa sổ Commandwindow ta gõ lệnh

>> a =input('nhap a=');

Nhấn Enter cho kết quả dưới dạng

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

Sử dụng dòng nhắc gán giá trị biên trong trường hợp ta muốn thay đổi giá trị các biến lúc thực thi chương trình.

Ví dụ : sử dụng dòng nhắc gán giá trị biên để giải phương trình bậc hai

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

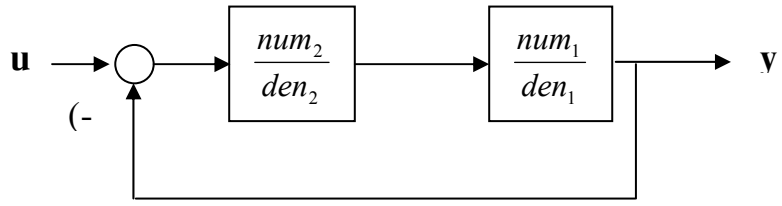
Chú ý:

- 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 **<u>không</u>** bắt đầu bằng từ khoá **<u>Begin</u>** và **<u>không</u>** 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□,[X_{min} ,X_{max}] ,tol,N,□LineSpec□);hoặc

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 f1, f2, f3

```
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=[x1;x2] thì hàm FUNC sẽ trả về ma trận sau đây.

```
f1(x1) ,f2(x1), f3(x1)
f1(x2) ,f2(x2), f3(x2)
```

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
```

Thực thi chương trình trên trong Commandwindow

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

(hoặc dùng >>fplot(@myfun ,[-20 20])

Các thông số tol, N **,** LineSpec ***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 tuỳ theo nhu cầu .

Sau đây là các cách lưu các biến dữ liệu:

1.8.1 Lưu và lấy dữ liệu dưới file nhi 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;
```

<u>Khôi phuc lai dữ liêu dùng lênh sau:</u>

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
```

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
```

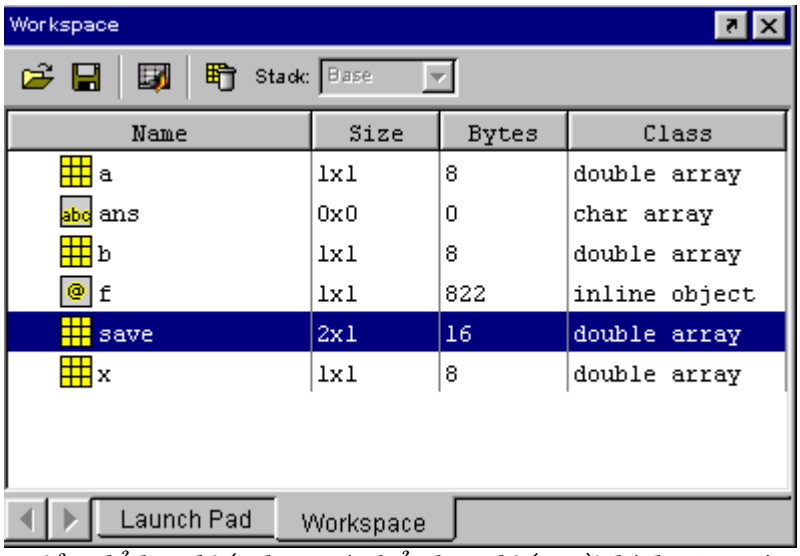
trong file %tên save

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 só 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 m trạn 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] ⇔ X=5*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
```

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 trạn 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 locgic 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 với n cho trước.

```
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=lsan1(5,1,5)
```

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
```

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 só sánh là đúng nếu tất cả các phần tử so sánh của hai mảng (hoặc hai ma trận) trả về gía 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');
     elseif(Diem>=7)&(Diem<8)
      fprintf('Hoc luc kh¸');
     else   fprintf('Hoc luc gioi');
    end;
```

1.10.2 Vòng lặp for

Cấu trúc:

```
for i= imin :Δi: imax
  statements;
end
```

Δi : Là bước nhảy của vòng lặp for, giá trị mặc định là =1;

Ví dụ: Tính tổng s= 1+2^p +3^p+ ... 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');
```

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+
```

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 tuỳ 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=1.3333333333333
- format lang e
  a=1.3333333333333e+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=
    1.3333333333333
```

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 hường -∞
Exp	Hàm e mũ
Ceil	Làm tròn hướng -∞
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 ∞ 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 π =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` <danh sách biến> 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 điode với quan hệ dòng điện và điện áp trên điode như sau
i=I0*(exp(40*v)-1)(A); I0=1.E-6; vector v=[-10:0.005:0.8] .
Sinh viên thực hiện theo hai cách viết trực tiếp trong CommadWindow 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 tập3 : 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]=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 matrận, 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 tuỳ thuộc vào flag là ' r ' , ' d ' ,' e ' hoặc ' f '

S = sym(A,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 matrận 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 pahỉ 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= 3*x^2 + 2*x + 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= 3*x^2+ 2*x+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 >=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 matrận

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 rông 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

1/3*k^3-1/2*k^2+1/6*k

symsum(k) trả về

1/2*k^2-1/2*k

symsum(sin(k*pi)/k,0,n) trả về

-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) 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 hao 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*n/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, véc 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 trong nhất trong toán học,đạo hàm (trong trường hợp này là cos(x)) và hàm e mũ x

giới hạntồn 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 quảlà 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, limit(f) giống với 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

int(f)

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

Tương tự hàm int(f,v)

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)	Log(y)
n^x	1/log(n)*n^x
Sin(a*theta+b)	-cos(a*theta+b)/a
Exp(-x1^2)	1/2*pi^(1/2)*erf(x1)

1/(1+u^2)	Atan(u)

Bảng thể hiện kết quả tích phân của một số hàm

Định nghĩa tích phân còn được thể hiện như sau

int(f,a,b)

hoặc 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 làmột 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 phuơng 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 = 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

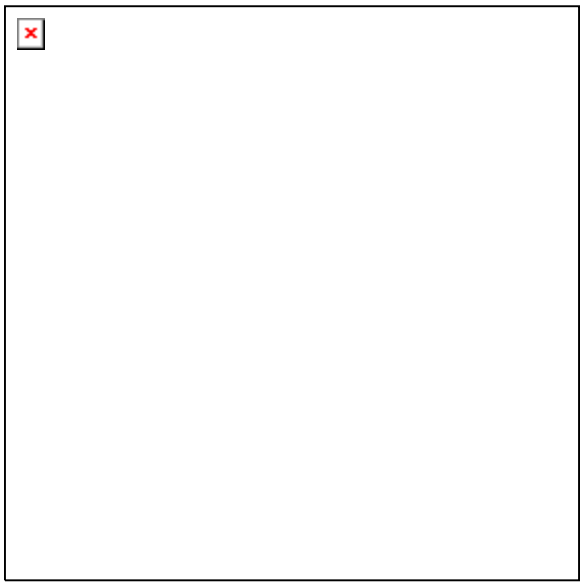
Nếu F = F(s), laplace trả lại một hàm của t

Bằng cách định nghĩa

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

L = laplace(F,t) tạo ra L,một hàmcủa t thay mặc định là hàm của s.

L = 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 = ilaplace(L)

F = ilaplace(L,y)

F = ilaplace(L,y,x)

Mô tả

F=ilaplace(L) là phép biến đổi ngược Laplace của đối tượng vô hướng symbolic Lvới 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), 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 = 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 = 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 tinh tế liên quan tới đạo hàm các hàm symbolic là
dấu 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).

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

Ví dụ, biểu thức

Là dương,đồ thị có hình chuông cong tiến tới 0 khi *x* tiến tới ± inf với mọi số thực k.
Một ví dụ về đường cong được cho thấy dưới đây với
được tạo ra, sử dụng những lệnh sau

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

```
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^2 "

tạm dịch là không hiểu dấu của k^2. Mà hợp lý toán học là k^2 phải dương do vậy bạn phải khai báo sao cho $k^2 > 0$ bằng cách

---> Tạo biến Real sử dụng lệnh sym

Chú ý rằng Maple không thể định nghĩa dấu của biểu thức k^2. 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)
```

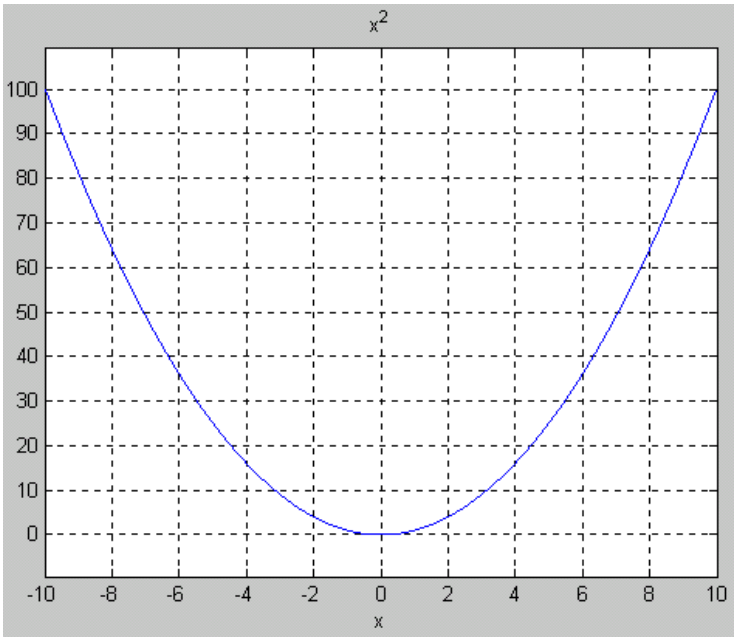
2.9 Vẽ Đồ thị Dùng hàm ezplot cho các biến, số symbolic

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 matrận trong Matlab

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

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

a. Nhập Matrận trực tiếp từ cửa sổ command Window

Trong môn học toán cao cấp chúng ta đã biết nhập một matrận 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 matrận 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 matrận vừa nhập

Như ví du 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 matrận 9 hàng và một cột
```

Lưu ý rằng trong một số trường hợp matrận 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 Matrận từ M-file

Ta có thể nhập một matrận 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 matrận ta soạn thảo tương tự như trong cửa sổ command window sau đó lưu vào file như sau:

Ví du:

```
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 matrận 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 xn 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= 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 đến100, 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=Xmin : bước_tăng: Xmax

3.4 Truy nhập các phần tử của ma trận

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

Giả sử ma trận

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

Thì >> A(i,j) ; sẽ truy nhập đến 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\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 dẫy

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:

a11*x1 + a12*x2+ . . . +a1n*xn=b1

a21*x2 + a22*x2+ . . . +a2n*xn=b2

.

.

am1*x1 + am2*x2+ . . . +amn*xn=bm

Bài toán đặt ra là tìm véc tor x=[x1;x2;x3....;xn] 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

a1*x1 + a2*x2 + . . . + an*xn = b

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

<u>Xét hệ thống sau:</u>

a11*x1 + a12*x2+ . . . +a1n*xn=b1

a21*x2 + a22*x2+ . . . +a2n*xn=b2

.

.

am1*x1 + am2*x2+ . . . +amn*xn=bm

Viết theo ma trận A= [a11 a12...a1n; a21 a22...a2n,....am1 am2...amn]

X=[x1 x2.... xn]; B=[b1 b2 ... bn];

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 :

$$A^{-1}*A*X=X=A^{-1}*B$$

Ví dụ Giải hệ sau:

```
2*x1  -  x2  =  2
  x1  + x2  =  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*x1  -  x2  =  2
2*x1  -  x2  =  0
```

Ma trận hệ số A=[2 -1 ; 2 -1];

Vì det(A)=0 => 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

$$A*x=0$$

- **Nếu det(A)#0 hệ có nghiệm duy nhất là X=0**

Ví dụ xét hệ phương trình tuyến tính sau

```
2*x1 -  x2=0
  x1+  x2=0
```

ở đây det(A)= 3 cho nghiệm x1=0 , x2=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* x1 + 3*x2 = 0

2* x1 - x2 = 0

Ma trận hệ số A= [-6 3 ; 2 -1] , 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:

3*x1 + 4*x2 - 2*x3= 0

-2*x1 + 3*x2 - 4*x3= 0

5*x1 + x2 + 2*x3= 0

-9*x1 + 5*x2 - 10*x3= 0

Ma trận hệ số là ma trận 4 x 3 ,định thức lớn nhất có thể được xây dựng từ ma trận A là định thức ma trận 3 x 3, nhưng định thức của ma trận kích thước 3 by 3 =0 (A1=[3 4 - 2; -2 3 - 4 ; 5 1 2]=> det(A1)=0)

Do đó ta xác định tiếp ma trận 2 x 2

Ví dụ như sau

A2=[3 4; -2 3] và det(A) # 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 tuỳ ý là = n- r (trong đó n là số biến còn r là hạng của ma trận A)

Giải hai phương trình :

3*x1 + 4*x2 - 2*x3= 0

-2*x1 + 3*x2 - 4*x3= 0

Kết quả : x1= (-10/17)*x3 và x2=(16/17)*x3 , với x3 lấy giá trị tuỳ ý

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

2*x1 - x2 = 2

x1 + x2 = 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 tuỳ ý**

Ví dụ trên

rank(a)= 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* x1 + 3* x2 + 4*x3 = 4
     x1 +    x2  +    x3 = 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 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

$$x1 + 2*x2 + 3*x3 = 12$$
$$3*x1 + 2*x2 + x3 = 15$$
$$3*x1 + 4*x2 + 7*x3 = 13$$

10*x1 + 9 *x2 + 8 *x3 = 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=[ 1 1/2   1/3.....1/n;1/2 1/3 ...1/(n+1)
    1/3 1/4   1/5....              1/(n+2)
    1/n ..                         1/(2n)]
```

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

```
A=[1   1;  1  1.01]   B=[2 ; 2.01];
>> 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 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* 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 đólà 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

Đinh nghĩa :Hang ma trận

Ar 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(Ar)#0 .Nếu bất kỳ ma trận Ar+1 nào được xây dựng từ r+1 hàng và r+1 cột của A, det(Ar+1)=0 thì chúng ta nói rằng Matrận A có hạng bằng r

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

a11*x1 + a12*x2+ . . . +a1n*xn=b1
a21*x1 + a22*x2+ . . . +a2n*xn=b2
.
.
am1*x1 + am2*x2+ . . . +amn*xn=bm

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 kiên có nghiệm Matrận [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 (\ ) .Nếu hệ có nghiệm duy nhất Matlab sẽ cung cấp cho nó , nếu hệ là không xác định(r<n) thì toán tử (\ ) 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= pinv(A)*B

Nếu hạng của A # hạng matrận mở rộng thì toán tử (\ ) 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) # 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 bào hệ có nghiệm hay không.**
4. **Hệ phương trình điều kiện yếu là gi?, 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 Addab= a+b .

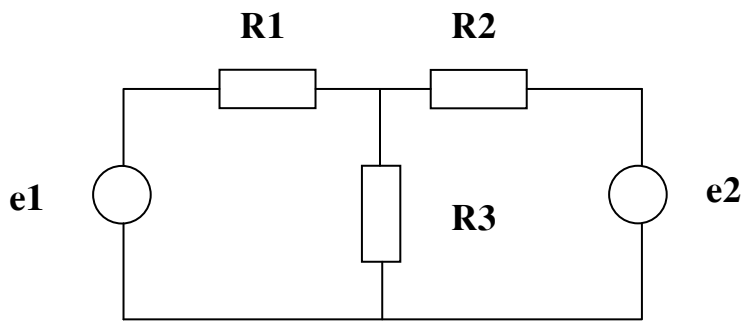
-Tính Subsab= a-b;

-Tính Multab= a*b

-Tính Mulba=b*a;

-Tính Divab= a/b; và b/a

2. Cho mạch điện sau



Cho thông số: R1= 10(omh) , R2= 20(omh) , R3= 10(omh)

e1= 20(v) , e2= 30(v)

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

<u>Bài tập giải hệ phương trình tuyến tính sau:</u>

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 <u>invhilb(n)</u>

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

Mô tả:

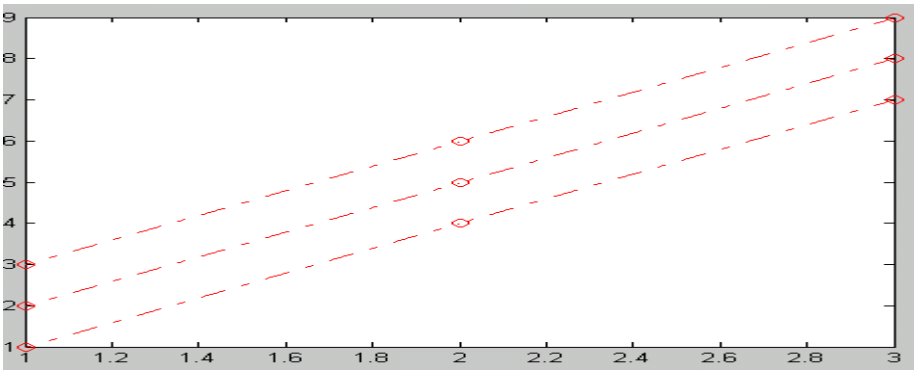
Hàm plot có nhiều cách dùng như bạn đã thấy ở trên

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 vec 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 matrận đó.

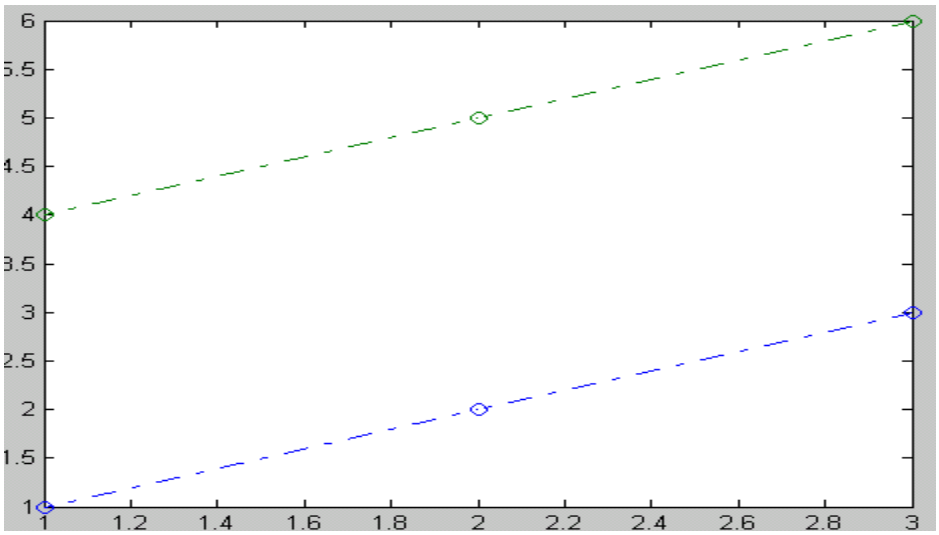
Cụ thể :

Giả thử x: là vector cot

x=[1 2 3]; và

y là martrận y=[1 2 3;4 5 6];

rõ ràng là x có kích thước bằng với kích thước hàng của matrận 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 phần tử ,các phần tử này xác định đường

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

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 mầu của đường và mầu của vòng tròn đánh dấu là mầu 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,protypevalue....)**

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(Linewith) mặc định là o.5 points(1point=1/72 inch)

Ví dụ về Linepropertype:

```
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 mầu (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 mầu đen(blue)

Mỗu trong (mặt) của các điểm nút giao là mầu đỏ:

Chúng ta thấy rằng đồ thị được xây dựng từ việc nối các điểm có toạ 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('truc thoi gian');
                 ylabel('truc Sin va cos');
                 title('Do thi ham 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 bắng độ xuống mặt phẳng x,y**

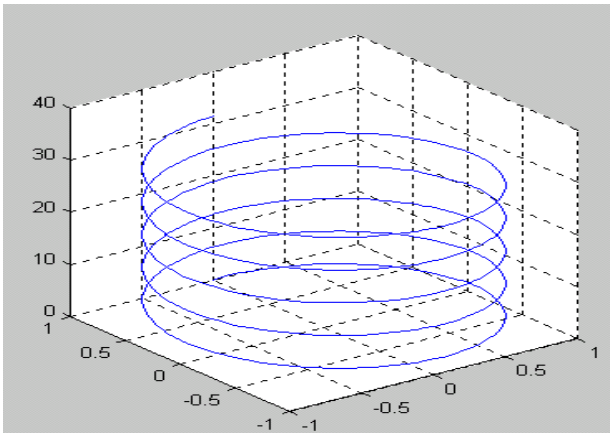
Giá trị mặc định của a và b là -37.5 ^{0} và 30 ^{0}

```
%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,LineSpec,...)
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 truc x.

```
X=0:10:1000;
subplot(2,2,2);
semilogy(x,y,'.');
xlabel('tuyen tinh');
ylabel('log');
subplot(2,2,3);
loglog(x,y,'.');
xlabel('log');
ylabel('log');
subplot(2,2,4);
plot(x,y,'.');
xlabel('tuyen tinh');
```

```
Y=100*x;
Subplot(2,2,1);
Semilogx(x,y,'.');
Xlabel('log');
Ylabel('tuyen tinh');
Title(' semilogx(x,y)');
Grid on;
```

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 toạ 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 toạ độ cực

Cấu trúc

polar(theta,rho)

polar(theta,rho,LineSpec)

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,LineSpec) LineSpec 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 toạ độ cực sang hệ toạ 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 toạ độ cầu Trong Matlab không có hàm để vẽ .tuy nhiên ta có thể chuyển đổi từ hệ toạ độ cầu thành hệ toạ độ ĐèCart 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(...,LineSpec)

II. Mô tả

Vẽ biểu đồ các giá trị trong véc tor hoặc trong Matrận 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.**

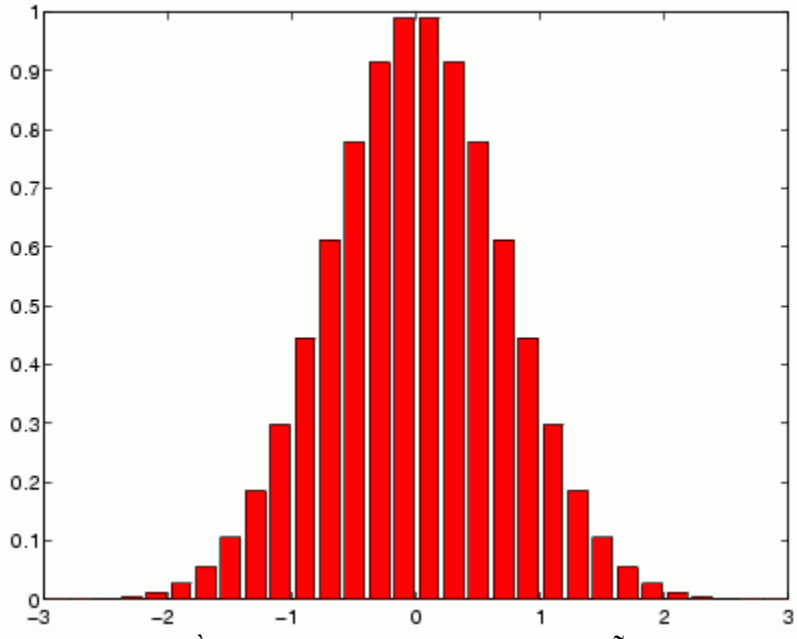
- **'stack' Biểu diễn đồ thị cột cho mỗi hàng của Y. Chiều cao của cột là tổng các phần tử trong một hàng**

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

Ví dụ

Plot a bell shaped curve:

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

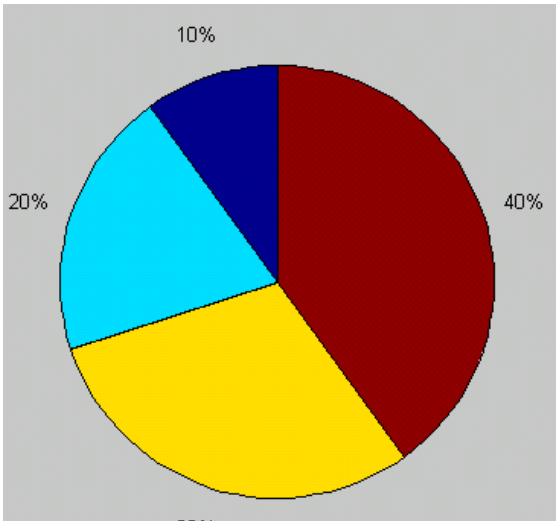


Tương tự đồ thị cột trong không gian lệnh vẫn giữ nguyên nhưng thay vì bar ta thay lệnh bar3(x,y,z)

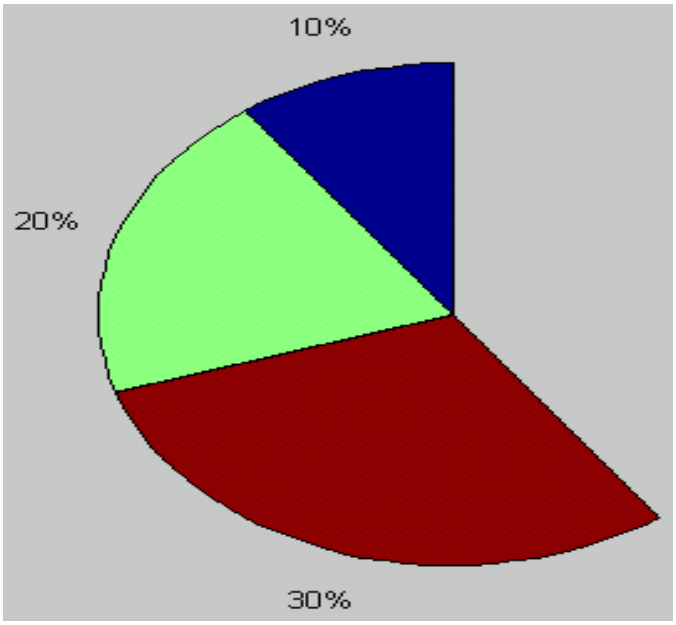
4.6 Đồ thị bánh (Pie)

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 vec 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 × 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 × m) số liệu được bố trí trên một hàng.

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

- Ma trận cột (n × 1) số liệu được bố trí trên 1 cột.

$$a_{11}$$
$$a_{21}$$
$$a_{31}$$
$$.$$
$$.$$
$$a_{n1}$$

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

```
>> 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 véctơ từ ma trận:

Công thức tổng quát: Biến = giới hạn đầu : bước chạy : gới 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, gới 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\A	B.\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 phần 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 phần 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 véctơ hàng thành véctơ 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*B.

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

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

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 tö trong ma trËn 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        -1.000
           -0.5000     -0.5000        1.5000
            0.500      -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 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 tuỳ 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:

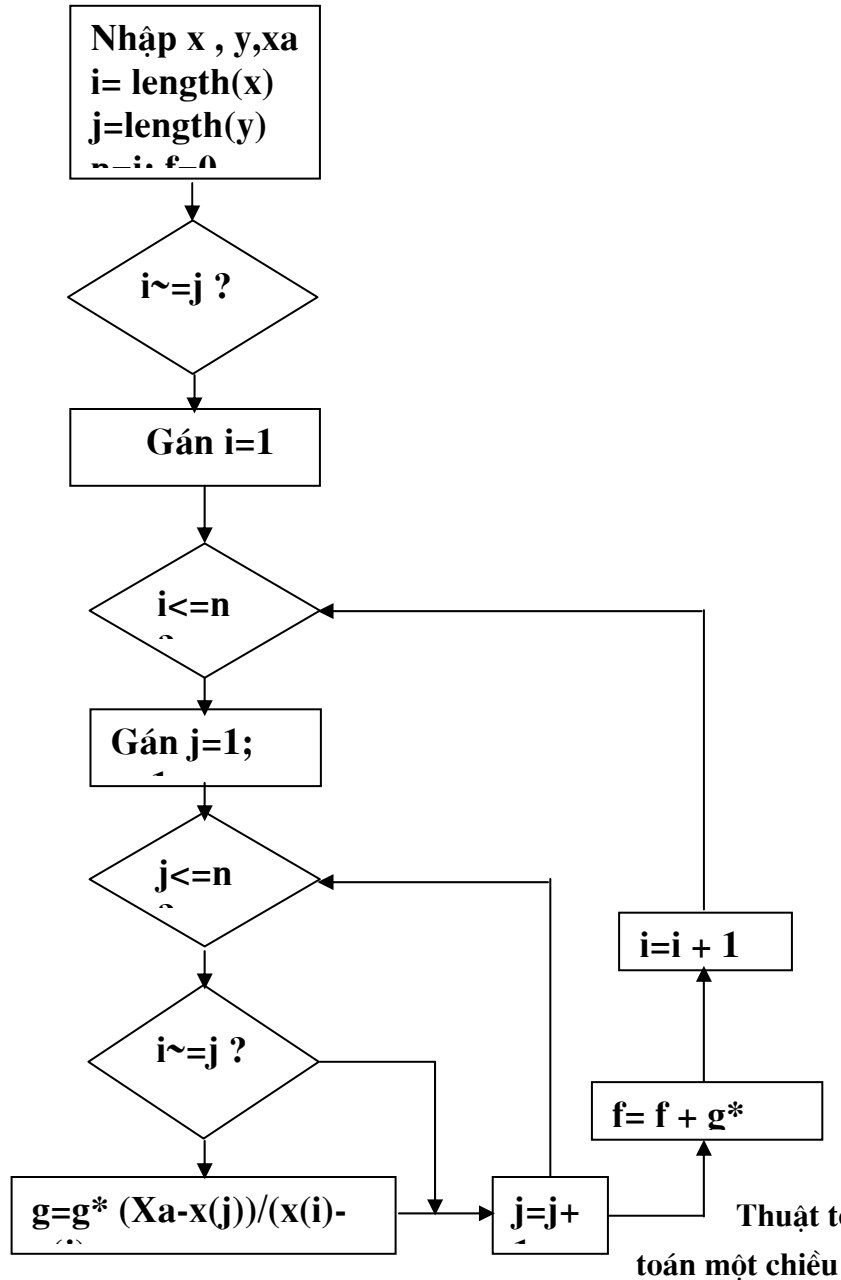
x	**x0**	**x1**	**x2**	**. . . . . .**	**xn**
f	**f0**	**f1**	**f2**	**. . . . . .**	**fn**

Công thức nội suy lagrange bậc N tính giá trị đo được tại một điểm bất kỳ là :

Thuật toán nội suy:

```
% thuat toan noi suy cho bai toan mot chie
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);
% in ra so lieu
        sl=[i x(i) f(i)]
        i=i+1; end
```



Thuật toán nội suy cho bài toán một chiều lagrange

interp1(nội suy theo spline)

One-dimensional data interpolation (table lookup)

Syntax

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

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 matrận .

yi = interp1(x,Y,xi,method) interpolates using alternative methods:

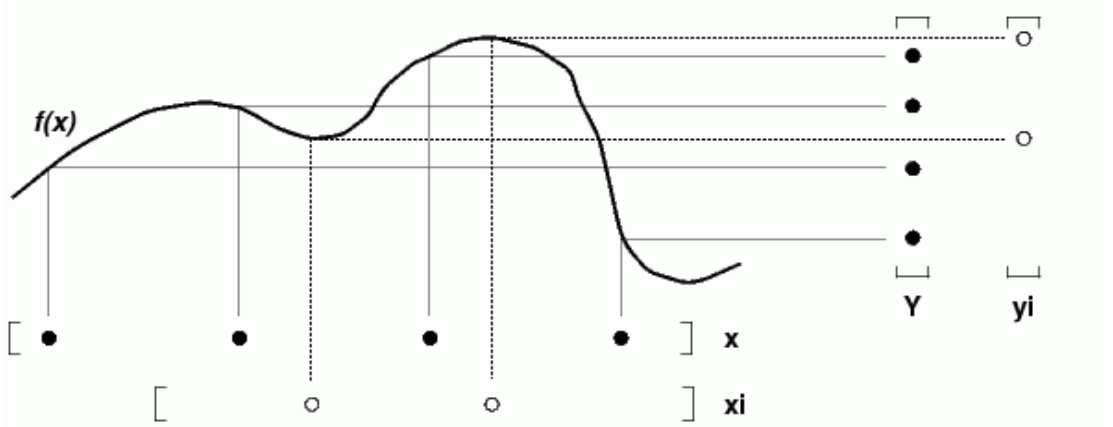
{PRIVATE} 'nearest'	**Nearest neighbor interpolation**
'linear'	**Linear interpolation (default)**
'spline'	**Cubic spline interpolation**
'pchip'	**Piecewise cubic Hermite interpolation**
'cubic'	**(Same as 'pchip')**
'v5cubic'	**Cubic interpolation used in MATLAB 5**

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

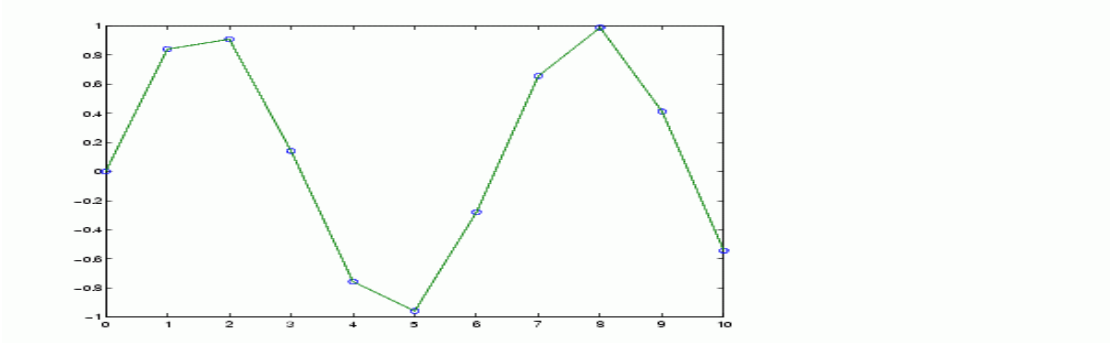


Interpolation is the same operation as *table lookup*. Described in table lookup terms, the *table* is [x,Y] and interp1 *looks up* the elements of xi in x, and, based upon their locations, returns values yi interpolated within the elements of Y.

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];

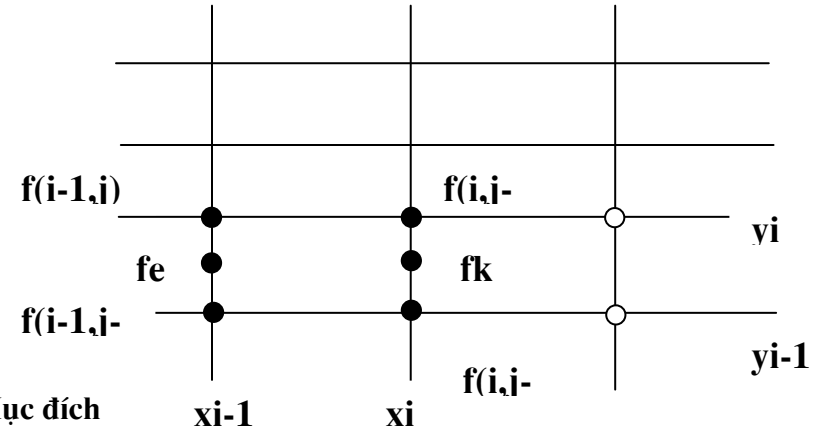
>> f=[0.671 0.620 0.567 0.512];

>> interp1(x,f,1.5)

ans =

0.6455

5.1.2 Nội suy cho bài toán hai chiều



Mục đích của bài toán là: Xác định giá trị f(x,y) của một vị trí bất kỳ trong một mặt phẳng xác định (biết các toạ độ và giá trị các điểm xung quanh x(i), x(i-1)...)

Muốn xác định giá trị tại một điểm có vị trí xi-1<x<xi và yi-1<y<yi 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 fe và fk

Bài 2: Nội suy theo phương x từ fe đến fk để tìm g(x,y)

fe=

$$\frac{yj-y}{yj-yj-1}f(i-1,j-1)+\frac{y-yj-1}{yj-yj-1}f(i-1,j);f_f=\frac{yj-y}{yj-yj-1}f(i,j-1)+\frac{y-yj-1}{yj-yj-1}f(i,j)$$

$$g(x,y)=\frac{xi-x}{xi-xi-1}fe+\frac{x-xi-1}{xi-xi-1}f_f$$

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,xa,ya)
% trong do x=[x(i-1) x(i)] y=[y(i-1) y(i)]
% xa ya la toạ 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

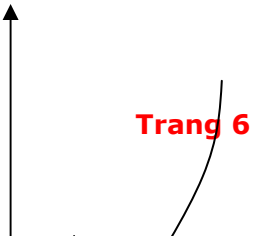
Dùng phương pháp chí đôi để xác định nghiệm của phương trình

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)~=0 thì so sanh 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 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>=(ln(b-a)/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

Xn=Xn-1 -f(Xn-1)/f(Xn-1)'

f=f(x) là hàm cần tính nghiệm, chúng ta sẽ tính các giá trị của Xn đến khi đạt được sai số cần thiết (tức là abs(Xn- Xn-1)< tol) thì Xn 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= $\frac{h}{2}(f_0 + 2f_1 + 2f_2 + ... + f_N)$

h= $\frac{b-a}{N}$

$f_0 = f(a), f = f(a+ih)$

Ví dụ tính tích phân: I= 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: *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 véctor cột có cùng chiều dài với vector y, hoặc y làmột 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

$$\mathbf{I=}\frac{h}{3}(f_0+4f_1+2f_2+4f_3+.....2f_{N-2}+4f_{N-1}+f_N)$$

H=(b-a)/N;

$f_0=f(a), f_1=f(a+i*h)$

%---------- Chuong trinh viet 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(fix(i/2)*2==i)
   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 Mach

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.

```
% Cac vi du
```

ví dụ1

```
syms t s;
I1= sym('I1(t)');
k=laplace(I1,t,s);     % Chuyen doi 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) % chuyen dao ham I1(t) sang Laplace
```

Các lệnh phụ trợ cần chú ý để giải một bài toán kỹ 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 phần 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(l ,{ '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 banừg 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
```

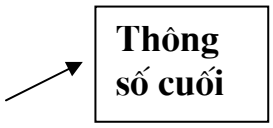
5.6 Phương trình vi phân thường

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 D2y 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 D3y 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 biếnphụ 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] = ODE23(ODEFUN,TSPAN,Y0) với TSPAN = [T0 TFINAL] tổ hợp hệ phương trình vi phân y' = f(t,y) từ thời gian T0 đến TFINAL với giá trị ban đầu Y0(with initial conditions Y0). Hàm 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 T0,T1,...,TFINAL(tất cả là tăng đều hoặc giảm đều) sử dụng TSPAN = [T0 T1 ... TFINAL].

[T,Y] = ODE23(ODEFUN,TSPAN,Y0,OPTIONS) solves as above with default integration properties replaced by values in OPTIONS, an argument created with the ODESET function. See ODESET for details.

Thông thường sủ 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-e3 và sai số tuyệt đối mặc định với tất cả các phần tử là 1e-6

Example

```
    [t,y]=ode23(@vdp1,[0 20],[2 0]);
    plot(t,y(:,1));
```

solves the system y' = 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*d2q/dt2  + R * dq/dt + q/c = Eo* cos(w*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 x1=q ,x2=diff(x1) như vậy ta có hai phương trình vi phân(giống như phương pháp đặt biến trạng thái trong lý thuyết điều khiển tự động)

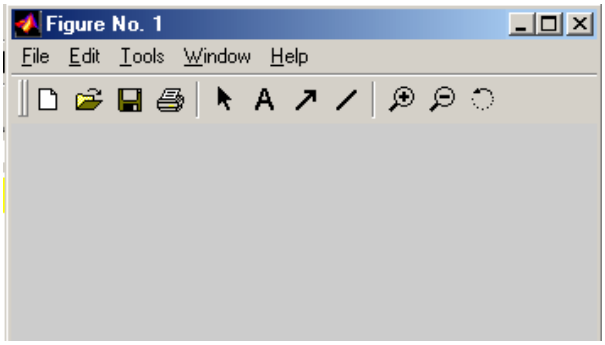
CHƯƠNG 5
ĐỒ HOẠ TRONG MATLAB

5.1 MÀN HÌNH ĐỒ THỊ:

Đây là nơi trình bày mọi hình ảnh , độ 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 cáclệnh 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 hoạ mới.

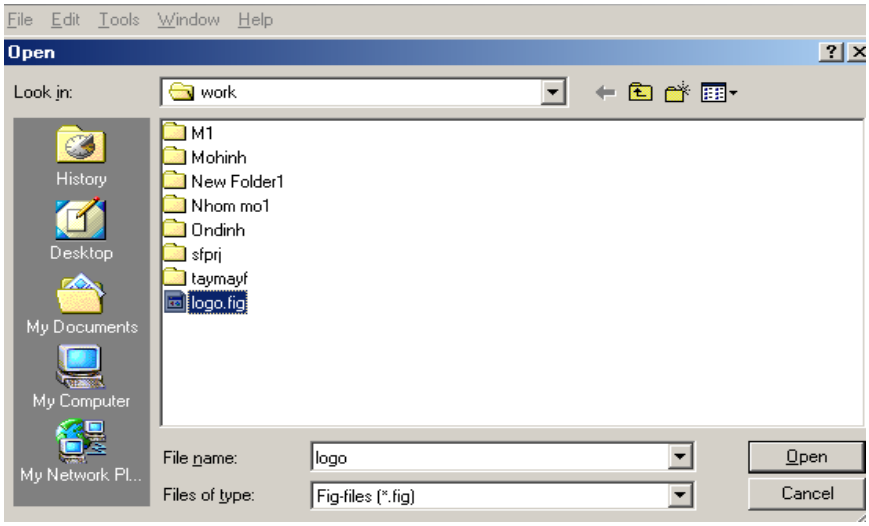
Để mở trang màn hình mới chồng lên màn hình đồ thị cũ trong khi đồ thị cũ vẫn còn hiện diện trên màn hình, chọn New Figure từ menu xổ. Một cửa sổ mới sẽ xuất hiện ra nằm chồng lên màn hình cũ

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:

- KÝch File/ Open tõ menu xæ, xuÊt hiÖn mµn h×nh tho¹i Open .



- **Look in**: Nơi chứa các tập tin đồ thị của Matlab. Nơi chứa có thể là ổ đĩa, thư mục hoặc chương trình khác.
- **File nane**: Tên tập tin muốn mở trong khung màn hình đồ thị .
- **Files of type**: Thể loại tập tin đồ thị là **.fig**

- KÝch ®óp vµo tªn tËp tin muèn më hoÆc ®,nh tªn tËp tin vµo khung File name hoÆc kÝch mét lÇn vµo tªn tËp tin, kÝch Open. TËp tin ®å thÞ võa chän sÏ hiÖn lªn mµn h×nh.

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)

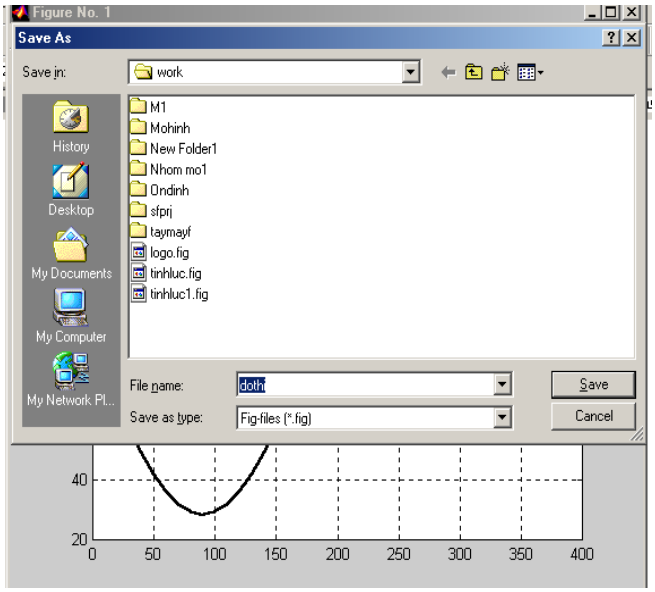
Save:

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 nhữnh 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

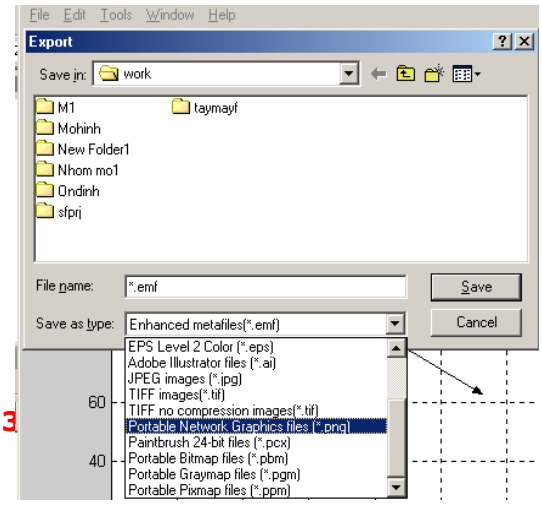


- **Save in** Nơi chưa các tập tin muốn. Lưu. Nơi chứa các tập tin đồ thị của Matlab. Nơi chứa có thể là ổ đĩa, thư mục hoặc chương trình khác.
- **File name** Tên tập tin tuỳ chọn để lưu cho đồ thị vừa tạo
- **Save as type** Thể loại tập tin muốn lưu. Mặc định là **.fig** đối với các tập tin đồ thị
- 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 chưa các tập tin muốn chuyển.

File name Tên tập tin muốn lưu lại để chuyển. Bạn có thể đặt tên theo tên cũ nhưng phần mở rộng lại là một tên khác .

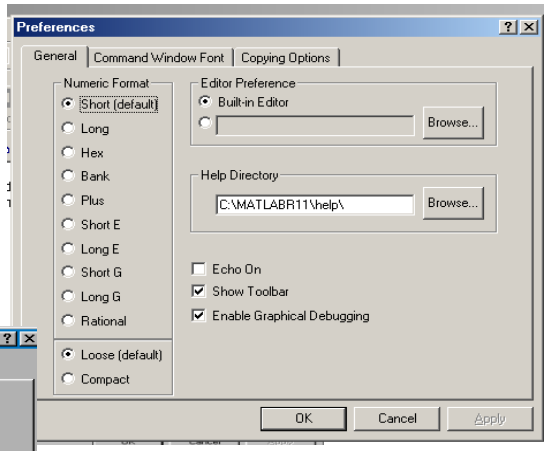
Save as type Chọn loại tập tin muốn lưu lại để chuyển.

- Sau khi chọn song, kích vào Save để ghi lại tập tin theo dạng khác.

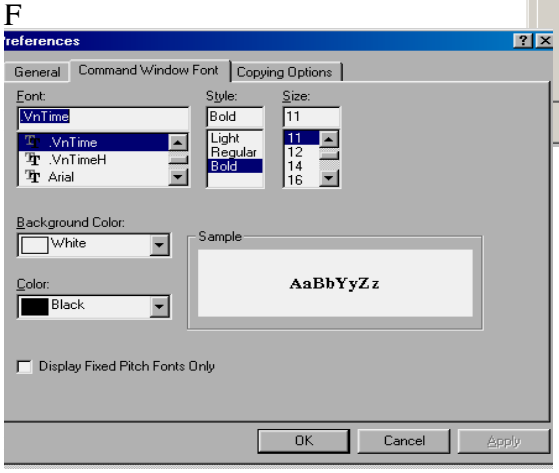
Property Editor:

Hiện khung thoại Graphics Property để người sử dụng thay đổi các khung thuộc tính mặc định cho phù hợp với tác vụ.

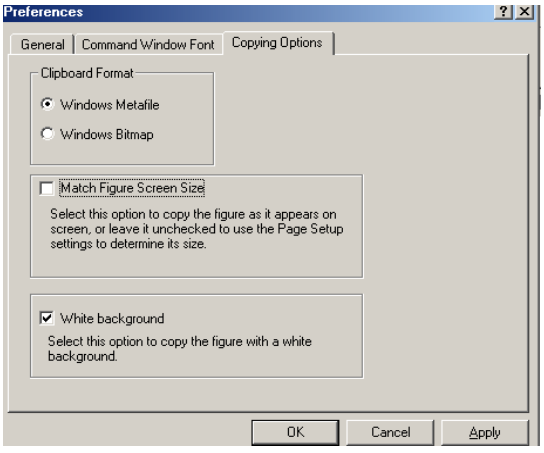
Preferences: Hiện khung thoại với ba tuỳ chọn để người sử dụng có thể thay đổi tham số cho phù hợp nhiệm vụ.



- **General:** Hiện khung thoại ngay khi kích chọn lệnh Preferences từ menu File của khung cửa sổ lệnh MATLAB. Kích chọn các loại tham số muốn thay đổi hoặc thêm sau đó kích OK.

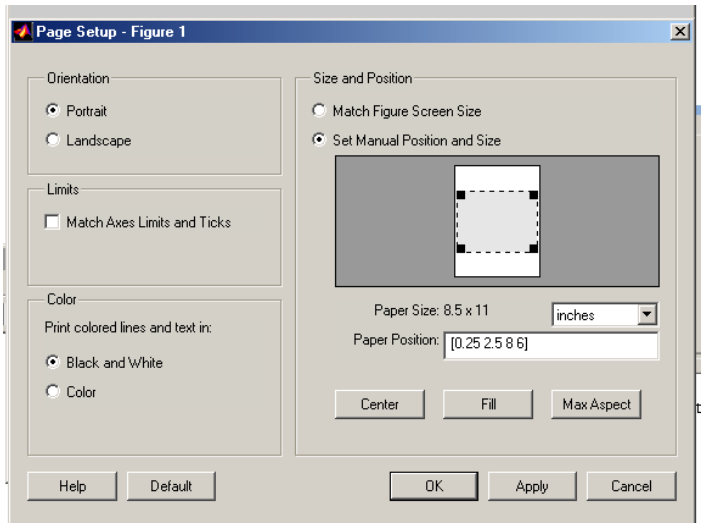


- **Command Windows Font:** Hiện khung danh mục font cùng thuộc tính để người sử dụng thay đổi font mặc định thành font quên thuộc.



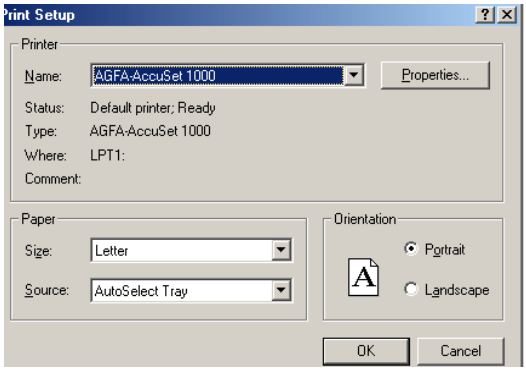
- **Copy Options :** Hiện khung thông tin để người sử dụng tuỳ chọn những thành phần đối tượng muốn sao chép.

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 gia trị theo thanh đo trên trục.
- **Color:** Chọn chế độ mầu hoặc đen trắng khi in hoặc vẽ trên thiết bị.
- **Size and Position:** Có hai tham số tuỳ chọn, sử dụng theo mặc định hoặc chỉnh theo tuỳ 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:

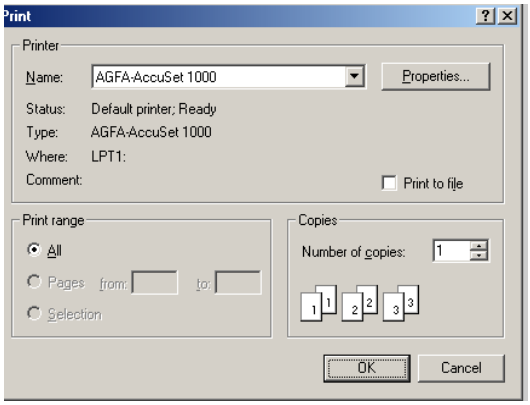


Nếu có nhiều loại thiết bị in(vẽ), thì nên khai báo lại trước khi chuyển bản đồ thị sang máy in hoặc máy vẽ.

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 bạn 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 sử 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 mời 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.

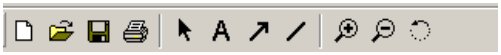
Copy Figure: Sao chép cả đồ thị sang Cliboard để rồi sau đó dán sang trang màn hình đồ thị hoặc sang chương trình khác. Các bước thực hiện như sau:

- Mở đồ thị muốn được sao chép sang đối tượng khác.
- Kích Edit / Copy Figure. Đồ thị sẽ được ghi vào Cliboard.
- 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 hoạ trong đồ thị để người quan sát dễ hiểu hơn.

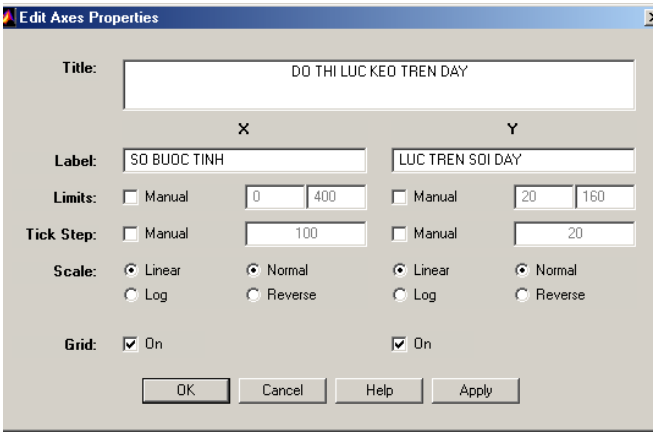


Show Toolbar:

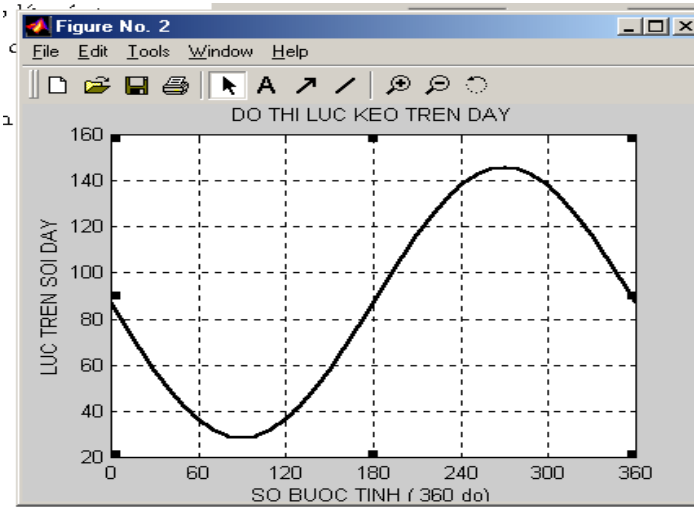
Cho hiện hoặc giấu các biểu tượng công cụ Standard trên màn hình

Enable Plot Editing: Đánh dấu chọn hoặc bỏ chọn đối tượng. 

Axes Properties: Gán thuộc tính cho các trục toạ độ cùng với tiêu đề đồ thị hiện hành. Thí dụ với đồ thị lực kéo sợi dây khi quay tròn quả 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:

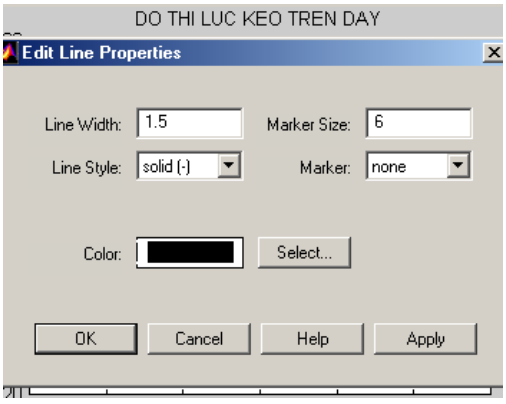


- Sau khi vẽ đồ thị , kích chọn **Tool / Axes Properties.** Màn hình xuất hiện khung thoại Axes Properties.
- Trong khung thoại Axes Properties có thể: đặt tiêu đề cho đồ thị (Title), gán nhãn cho trục x và trục y (Label), giới hạn độ lớn các trục (Limits), tỷ lệ xích các trục….. . cho đồ thị.
- Kích Apply và kích OK.

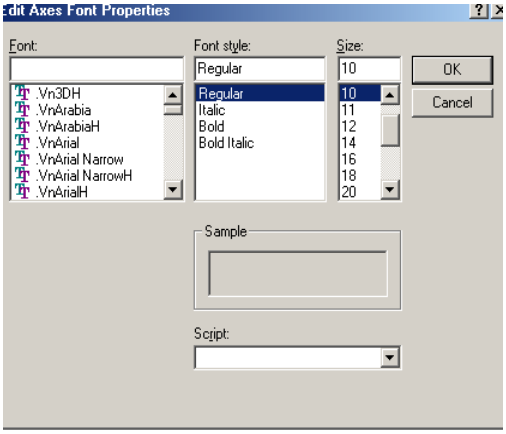


Line Properties: Gán thuộc tính như các điểm đỉnh dữ liệu, tạo nét đồ thị dầy hay mỏng cũng như mầu sắc cho đường đồ thị hiện hành. Các bước thực hiện như sau:

- 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.

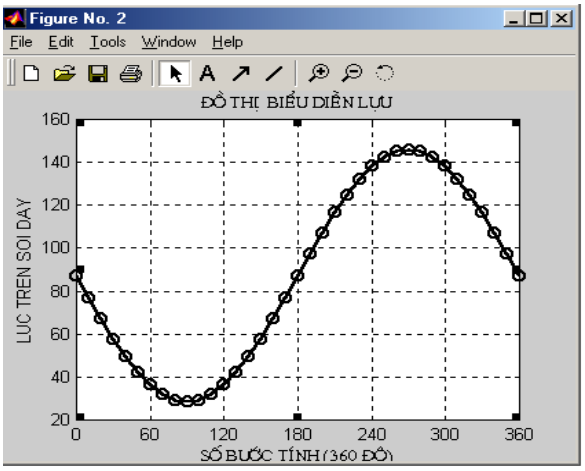


- Kích **Tool / Line Properties.** Màn hình hiện khung thoại line properties.
- Chọn độ dầy mỏng của đường đồ thị **Line Width**; kích cỡ dấu đỉnh dữ liệu trên đồ thị **Marker Size**; Kiểu dáng đồ thị **Line Style**; mầu sắc **Color** ……
- Kích Apply và kích OK.



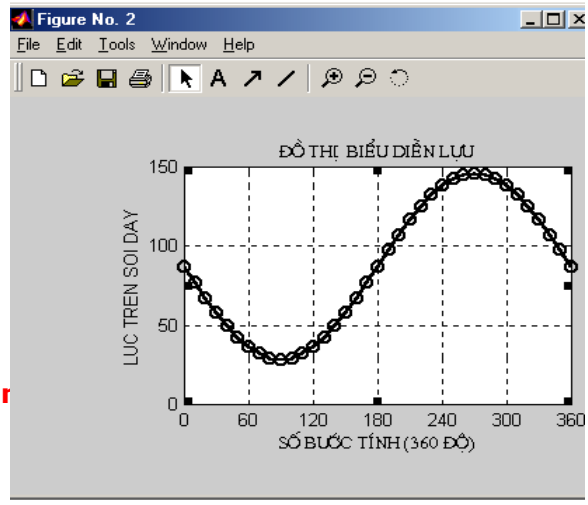
Text Properties: Thay đổi các thuộc tính cho các ký tự trong đồ thị. Các bước thực hiện như sau:

- 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 toạ độ hoặc di chuyển đồ thị đến vị trí khác trong khung cửa sổ đổ hoạ.

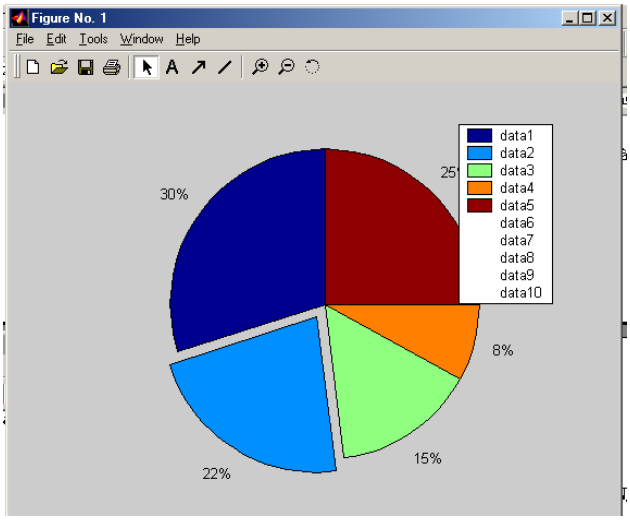
- Kích chọn biểu tượng **Enable Plot Editing**.
- Chọn trục toạ độ x (y) nếu là đồ thị 2D hoặc trục z nếu đồ thị là 3D.



- Kích menu Tool / Unloock 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 hoạ, 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 mầu khác nhau.

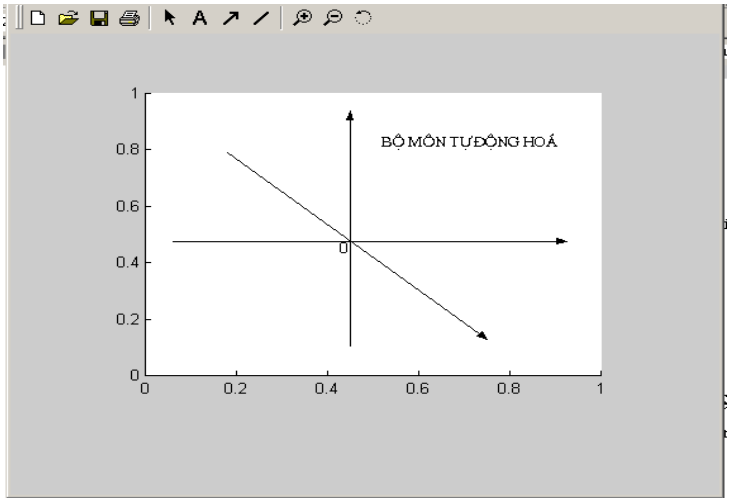


- Để giấu phân loại dữ liệu đồ thị, kích Tool / Hide legend

Add: Hiện bốn thanh công cụ hỗ trợ cho việc thêm các thuộc tính vào đồ thị:

- Axes: Vẽ trục toạ độ mời trong khung màn hình đồ hoạ.
  - 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 toạ độ đã định.
- **Line:** Thêm đường kẻ vào đồ thị. Các bước thực hiện như sau:
  - Mở đồ thị muốn thêm đường kể.

- Kích **Tool /Add /Line** hoặc kích biểu tượng Add Line 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 Propertise.



- 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.

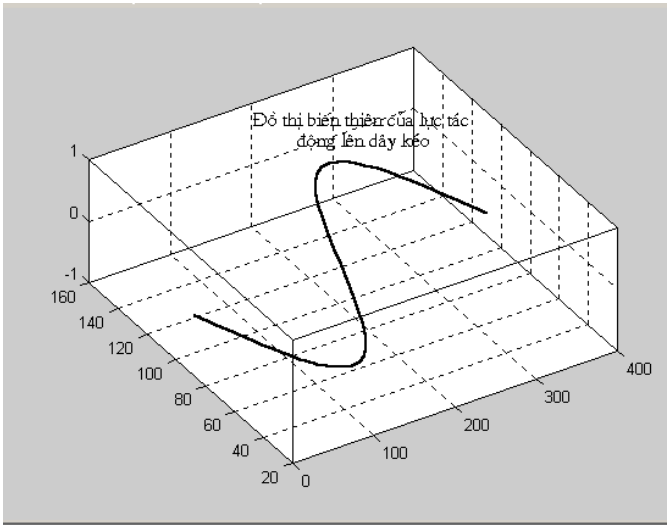
Zoom In: Phóng to đồ thị để quan sát từng chi tiết.

- 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 vecter 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** :

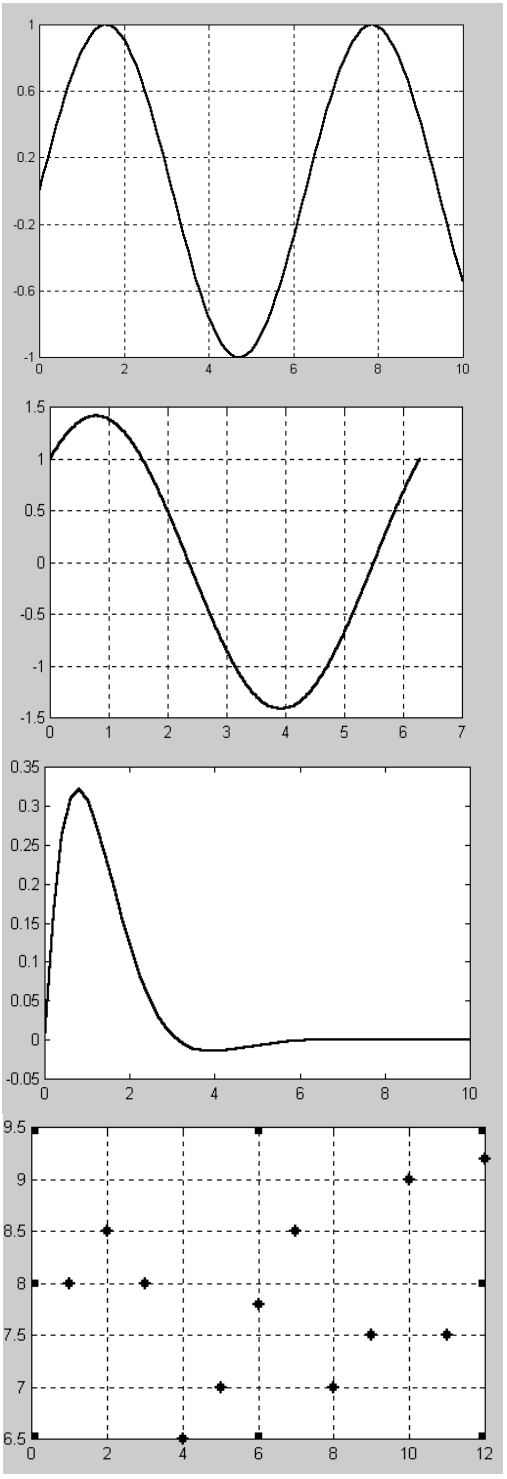
Tên biến = linspace (Điểm đầu, điểm cuối, số điểm cần vẽ)

% vẽ hàm y = e^{-x}.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:

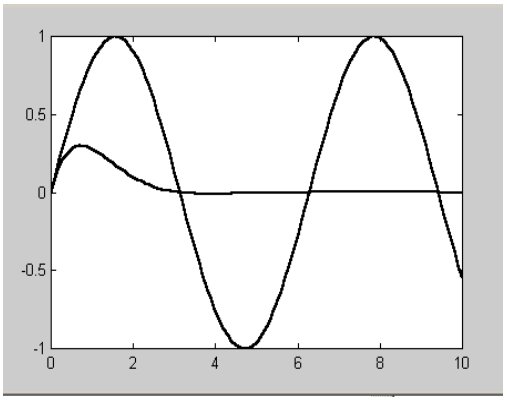
Đồ thị dạng đánh dấu là loại đồ thị chỉ dùng các điểm như vòng tròn, hình thoi …. Thay vì dùng các đoạn thẳng nối lại với nhau.VD 4:

```
>> 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 hoạ. Theo mặc định Matlab sẽ tự động gán loại mầu sắc cho từng dữ liệu để phân biệt. Công thức tổng quát khi vẽ nhiều đồ thị trên cùng một hệ toạ 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 toạ 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 hoạ 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 hoạ.
- Grid on: bật chế độ lưới trong màn hình đồ hoạ.
- Grid off: tắt chế độ lưới trong màn hình đồ hoạ.
- Hold on: giữ lại các đồ thị đã vẽ (dùng để vẽ nhiều đồ thị trên một hệ trục toạ độ)
- Hold off: ngược lại với họd on

Trong Matlab ta có thể chọn đường vẽ và mầu theo 1 trong các kiểu sau:

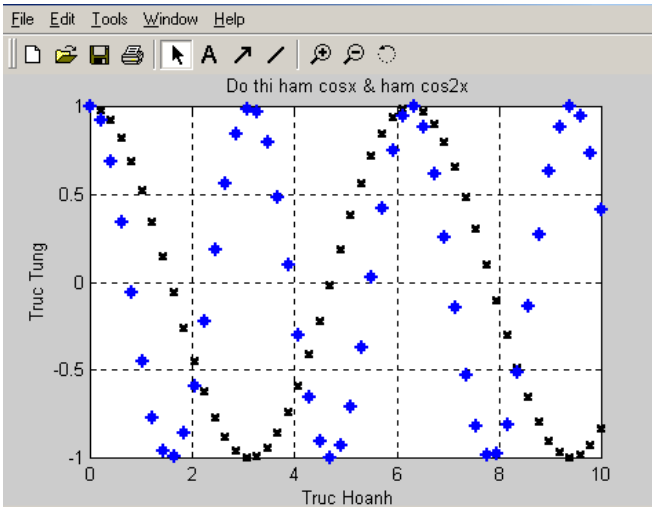
Ký hiệu	Mầu	Ký hiệu	Kiểu
y	vàng	.	Chấm điểm
m	đỏ tươi	o	Vòng tròn
	xanh	x	Dấu x

c	đỏ	+	Dấu cộng
r	xanh lá cây	*	Dấu sao
g	xanh thẫm	-	Nét liền
b	trắng	-.	Gạch chấm
w	đen	--	Gạch gạch
k			

Khi đó ta dùng lệnh: **plot(tên biến, tên hàm,'ký hiệu mầu ký hiệu kiểu đường')**

VD 6: vẽ hàm Cos(x), cos (2x)

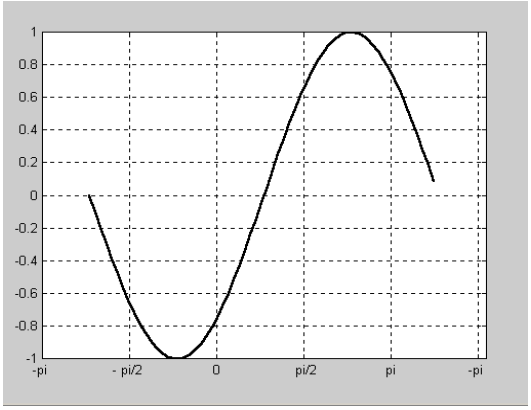
```
>> x=linspace(0,10,50);
>> y=cos(x);y1 = cos(2*x);
% vẽ y bằng dấu x mầu đen, y1 bằng dấu * mầu xanh thẫm
>> plot(x,y,'xk',x,y1,'*b');
% Tên đồ thị
>> title(' Do thi ham cosx & ham cos2x')
>> xlabel(' Truc Hoanh')
>> ylabel(' Truc Tung')
>> grid on
```



Gán giá trị thanh đo: Ngoài giá trị thanh 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)
```

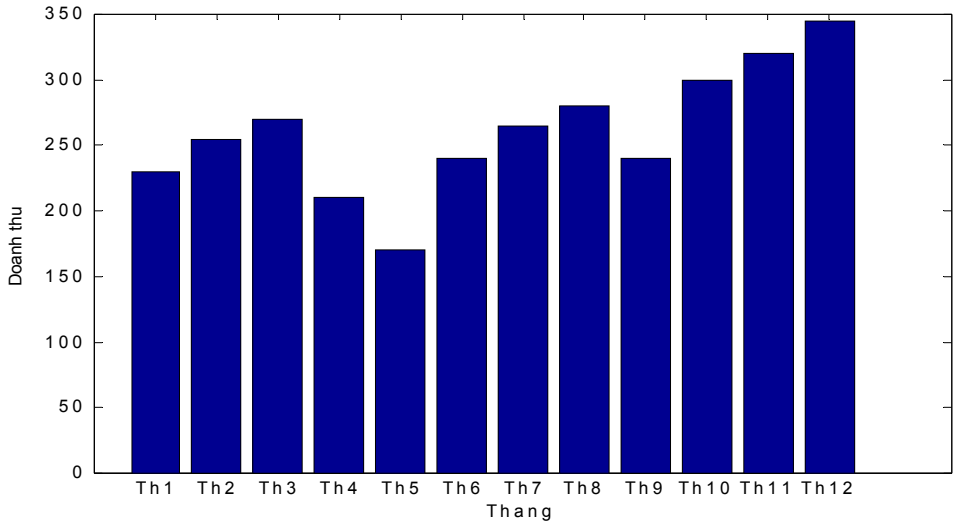


>> set(gca,'Xticklabel',{ '-pi','- pi/2','0',' pi/2','pi' })

5.3.5 Đồ thị hình thanh:

Loại đồ thị này thường dùng để minh hoạ 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.

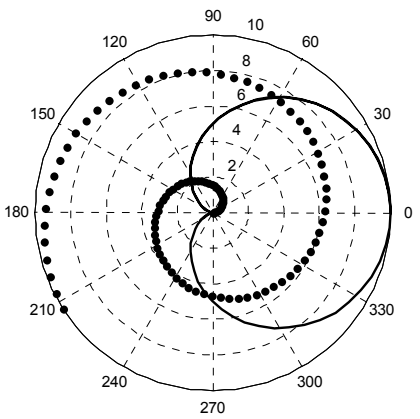
```
>> x = [230 255 270 210 170 240 265 280 240 300 320 345];
>> bar (x)
>> xlabel('Thang')
>> ylabel('Doanh thu')
>>set(gca,'Xticklabel',…
   { 'Th1','Th2','Th3','Th4','Th5','Th6','Th7','Th8','Th9','Th10','Th11','Th12' })
```



5.3.6 Đồ thị toạ 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:

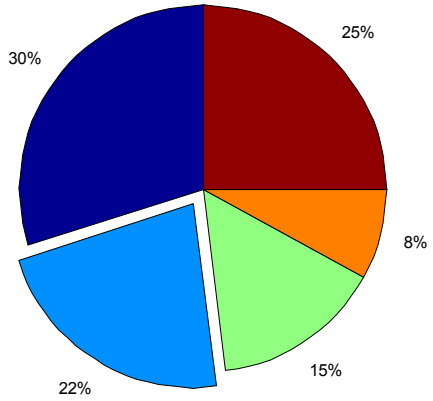
Là loại đồ thị tỷ lệ bách phân của từng loại dữ liệu để minh hoạ. Theo mặc định Matlab sẽ tô mầu khác nhau cho từng thành phần dữ liệu.VD 10:

>> 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:

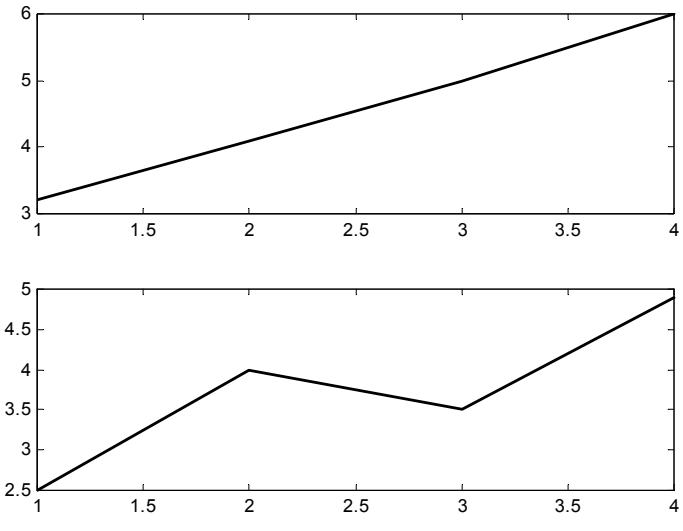
>> a = [3.2 4.1 5 6];

>> 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

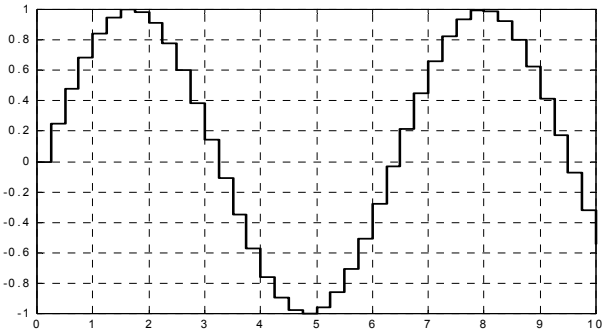
5.3.9 Lệnh staris:

Để vẽ đồ thị bậc thang.

VD 12:

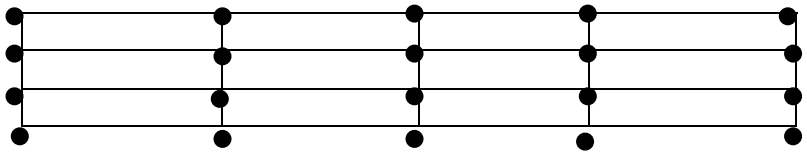
>>x = 0: .25: 10;

>>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:

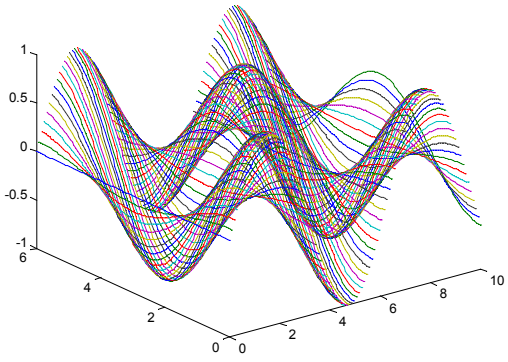
>> a =linspace(0,10,100);

>> 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

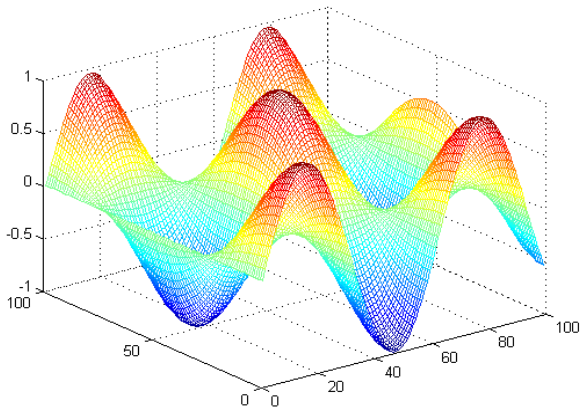
>>a=linspace(0,10,100);

>> 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

Muc luc 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?

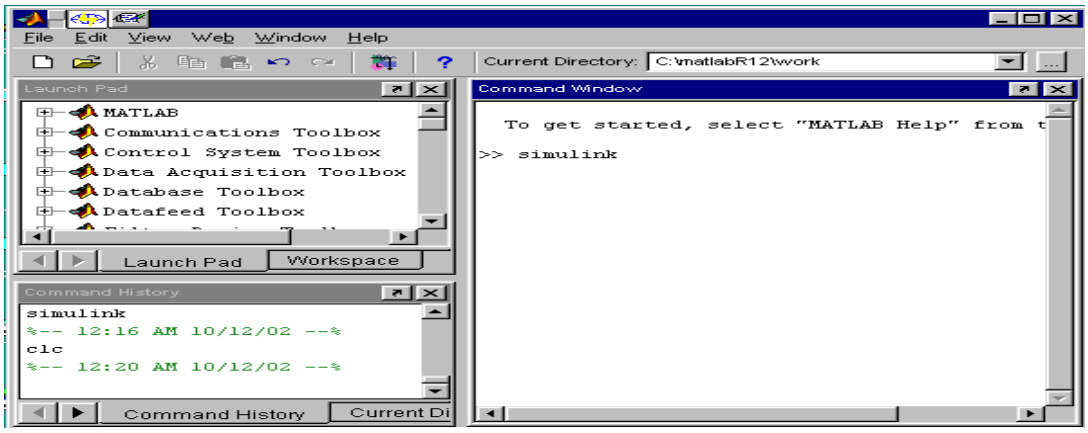
Simulink là một phần mềm gói gọn được sử dụng để xây dựng mô hình và mô phỏng , tính toán phân tích hệ thống động .Simulink cho phép mô tả hệ thống tuyến tính, hệ phi tuyến , các mô hình trong thời gian liên tục hoặc gián đoạn(lấy mẫu) hay kết hợp cả hai. Đối với mô hình , Simulink cung cấp một giao 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)**

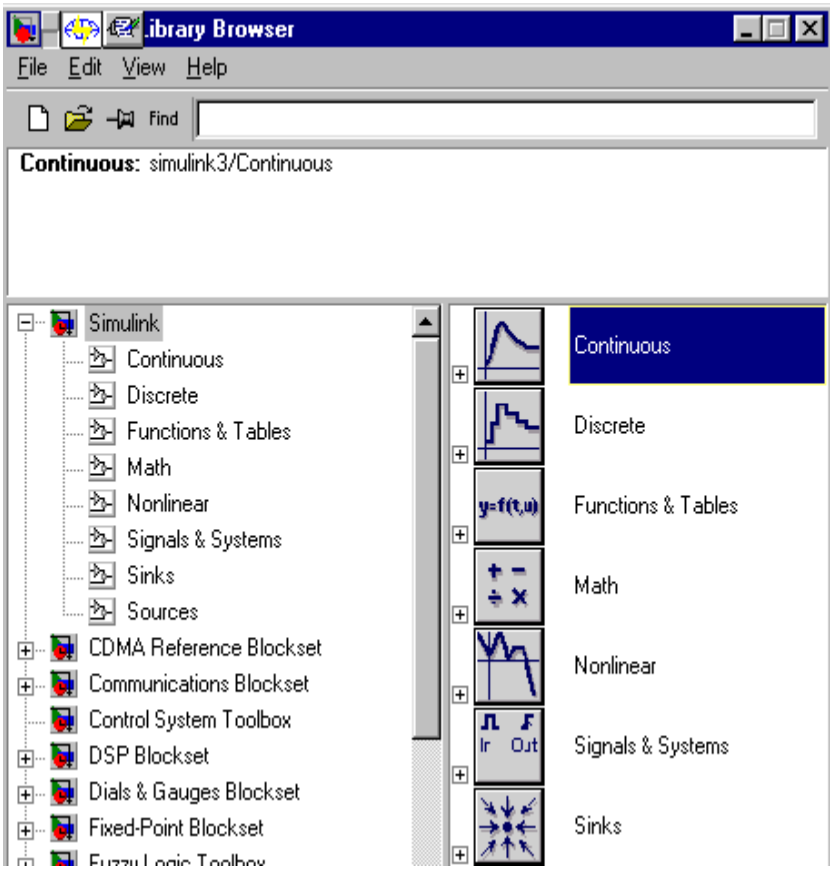
Xây dựng mô hình 'từ trên xuống 'hoặc 'từ dưới lên trên ' để xem kỹ các khối trong thư viện các khối source hoặc sink linear . . . bạn kích đúp chuột vào các khối đó. Sau khi định nghĩa mô hình bạn có thể mô phỏng mô hình đó , sử dụng scope để xem biểu diễn mô hình đó ,ví dụ như một khối phát hình sin , 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:

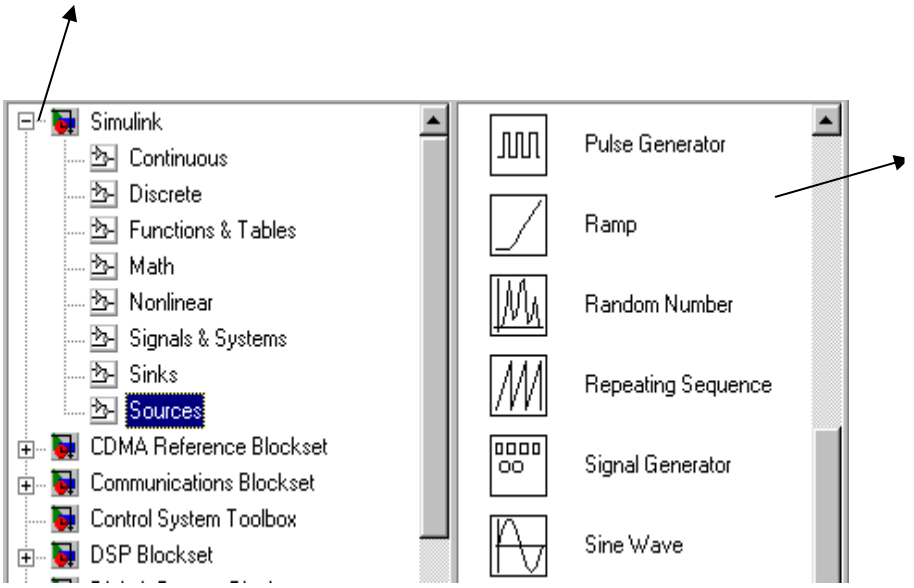


Trên màn hình sẽ xuất hiện thêm một cửa sổ mới , cửa sổ này chứa toàn bộ dữ liệu thư viện của Simulink, nó có thể di chuyển được bằng chuột như sau:



Các khối thư viện Của

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 ban chọn khối source*

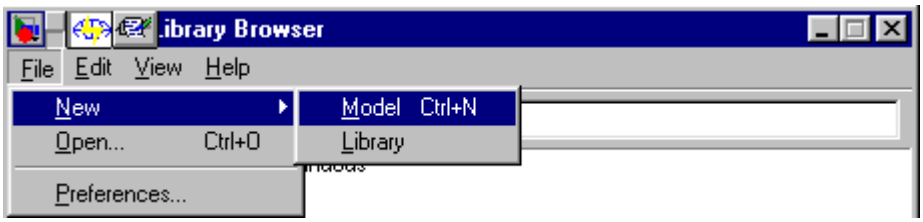


Hàm con của khối

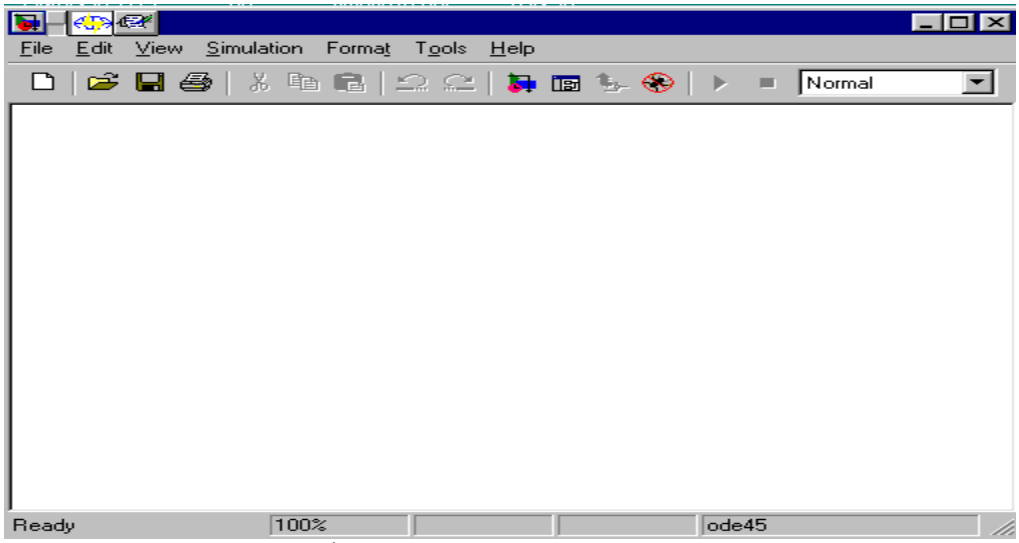
Tạo mội trường làm việc

Từ cửa sổ Library Browser (xem hình trên) ta kích chuột vào file danh sách các mục New , Open , Preferences xuất hiện .

<u>Để tao môi trường làm viêc (vùng để vẽ mô hình) ta chon muc New rồi chon Model Ctr+N một cửa sổ làm viêc xuất hiên</u>



Cửa sổ làm việc như sau

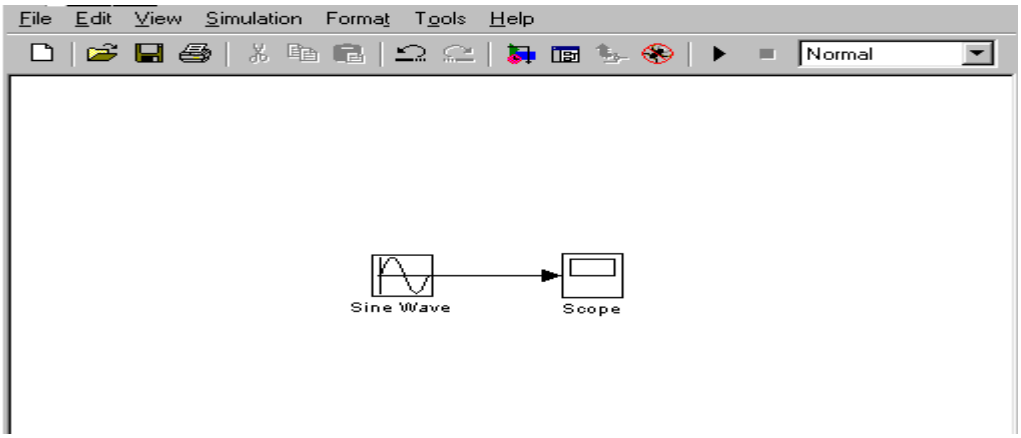


Đặt lại tên cho mô hình bằng cách vào mục file -> Save as

6.3 Phương pháp xây dựng mô hình

Tất cả các bước trên là chuẩn bi cho viêc xây dựng mô hình mô phỏng

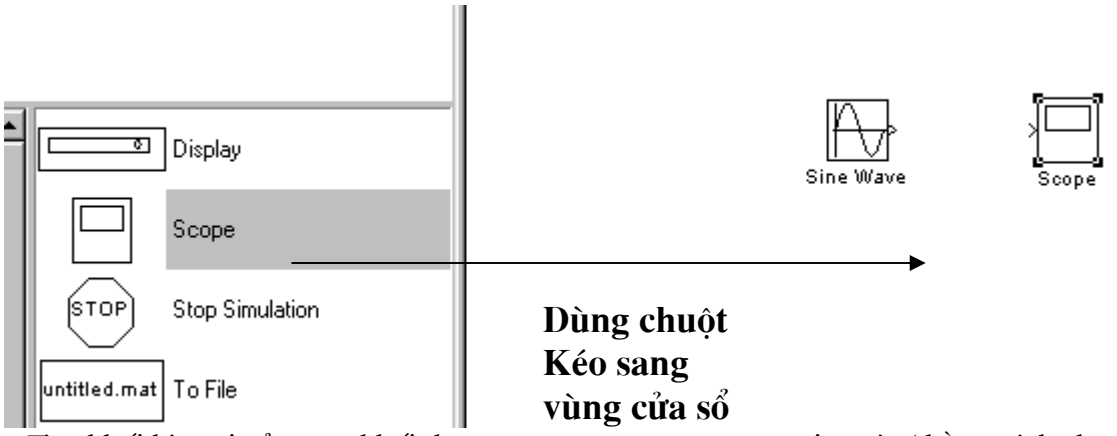
Giả sử ta muốn xây dựng mô hình phân tích sóng sin trên cửa sổ làm việc như sau:(xem



hình vẽ dưới đây)

Các bước:

Sau khi ***tao môi trường làm viêc mới*** (các bước giới thiệu ở trên) tiếp đến

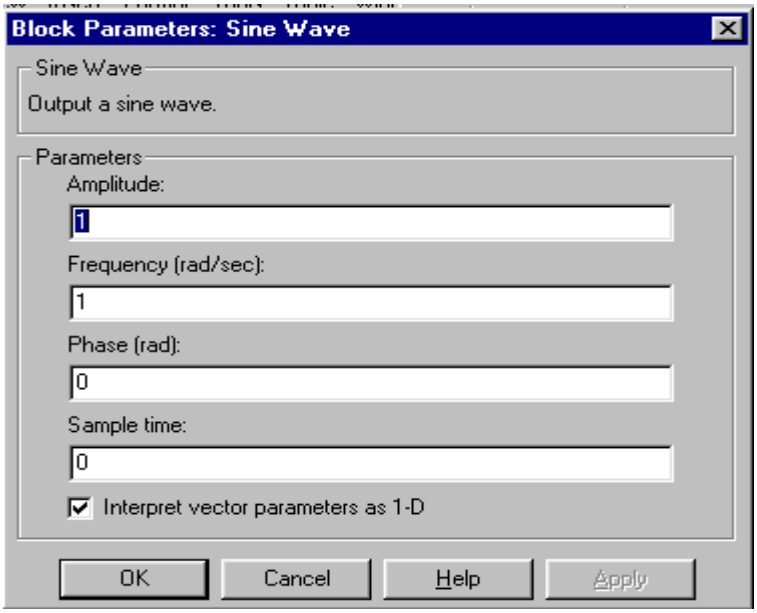


Tìm khối hàm sin ở trong khối thư viện nào(bằng cách chọn từng khối bằng chuột từ các mục dưới Simulink) ở ví dụ này hàm tạo sin ở trong khối Source (xem hình trên), dùng chuột chọn vào khối SineWave giữ nguyên chuột rồi kéo sang vùng cửa sổ làm việc , trên của sổ làm việc xuất hiện khối hàm SineWave, 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 lai các thông số của các hàm bằng cách kích đôi 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:

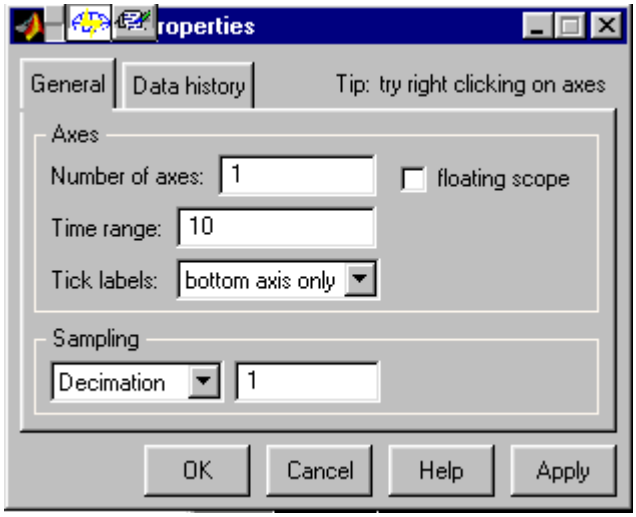
+ Chu kỳ(tần số) Frequency(rad/s)

+ Biên độ Amplitude



+ Sample time (thời gian lấy mẫu)

Khối Scope:

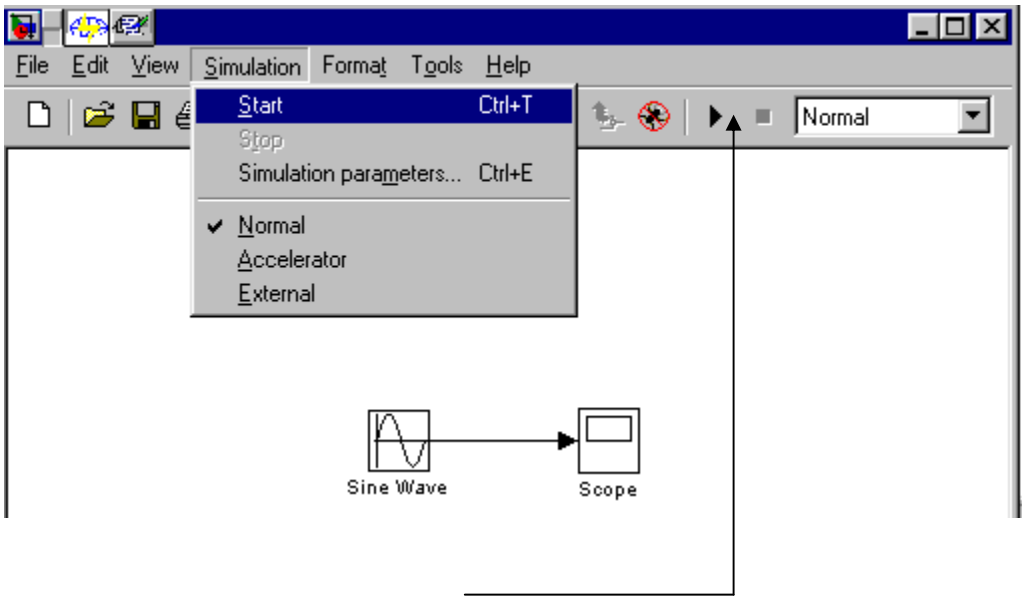


Sau khi hoàn tất đặt lại các thông số , đến công việc quan trong nhất là kết quả mô phỏng

:

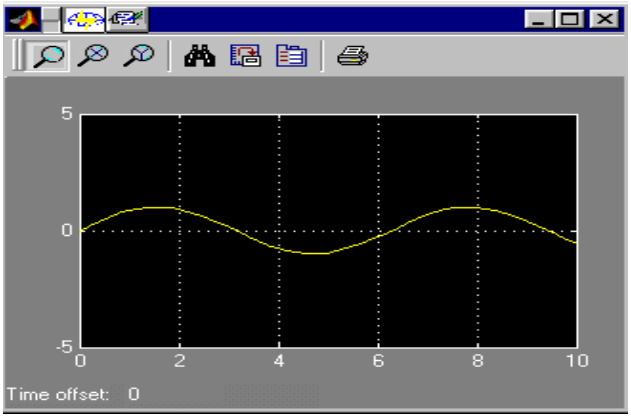
1. Trên thanh công cụ nhấp chuột vào mục Simulation

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 phảil 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)

$$y1 = a_n x^n + a_{n-1}x^{n-1} + ... + a_0$$

$$y2 = b_n x^n + b_{n-1}x^{n-1} + ... + b_0$$

Bước1: 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)

$y = a_n x^n + a_{n-1}x^{n-1} + ... + a_0$

Bước1: 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:

$$\begin{vmatrix} 2 & 3 & 1 \\ 3 & 6 & -4 \\ 1 & 1 & 1 \end{vmatrix} \cdot \begin{vmatrix} x \\ y \\ z \end{vmatrix} = \begin{vmatrix} 7 \\ 19 \\ 2 \end{vmatrix}$$

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 =   -2.34957562245720321874105364003 68
y =   2.68352691947852194272702390790 10
z =   1.66604870297868127601402973213 5
```

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 phần theo hàm có sẵn của Matlab:

Ví dụ: Cho hệ phường trình vi phân

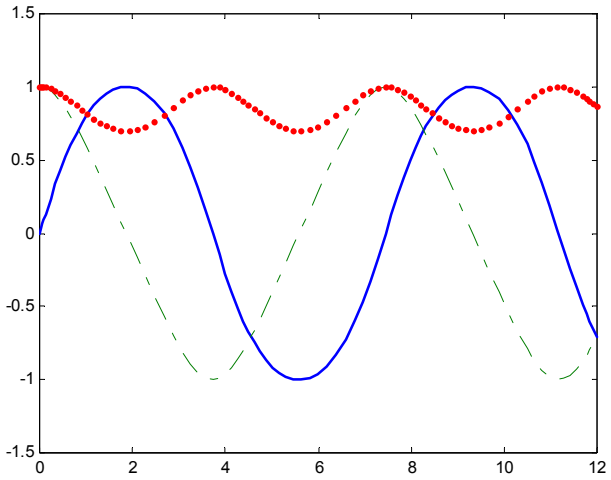
$$y'_1 = y_2 y_3 \qquad y_1(0) = 0$$
$$y'_2 = -y_1 y_3 \qquad y_2(0) = 1$$
$$y'_3 = -0.51 y_1 y_2 \qquad y_3(0) = 1$$

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.cos2x.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):**
  - int(S): Tích phân không xác định của biển thức symbolic S với biến tự do mặc định. Muốn biết biến mặc định ta dùng lệnh findsym.
  - int(S,v): Tích phân không xác định của biểu thức symbolic S với biến tích phân v.
  - 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].
  - 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].

Vidụ:

```
>>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)]
```

Vídụ: 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:
sign(s) = 1 khi s>0; sign(s) = 0 khi s =0; sign(s) = -1 khi s<0;

- **Tính giới hạn(limit):**
  - limit(F, x, a) : Tìm giới hạn của biểu thức F khi $x \to a$.
  - limit(F, a) : Tìm giới hạn của biểu thức F với biến độc lập.
  - limit(F) : Tìm giới hạn của biểu thức F khi a = 0.
  - limit(F, x, a, 'right') hoặc Lim it(F, x, a, '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):**
  - symsum(S): Tổng của biểu thức symbolic theo biến symbolic k , k được xác định bằng lệnh findsym từ $0 \to k$ -1.
  - symsum(S,v): Tổng của biểu thức symbolic S theo biến symbolic v,v được xác định từ $0 \to k$ - 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)
```

Vi dụ: Cho tổng của 2 dãy

$S1 = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ldots.$

$S2 = 1 + x + x^2 + \ldots..$

```
>>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 gía 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 o và xung quanh điểm x0 = 2
```

Kết quả:

```
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
```

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 biễn 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*y^2+(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 = 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] = 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 = simplify(S)
  - R = simple(S)
  - [r, how] = 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 = 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 nàylà w.
  - F = fourier(f,v): F là hàm của biến v thay thế biến mặc nhiên w.
  - F = 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 = 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 = ifourier(F,u): f là hàm củabiến u thay thế biến mặc nhiên x.
  - f = 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 = laplace(F): Biến đổi Laplace của hàm F với biến mặc nhiên độc lập t. nó cho ta một hàm của s
  - L = laplace(F,t): L là một hàm của t thay thế biến mặc nhiên s.
  - L = 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)
```

- **BiÕn ®æi laplace ng-îc**

  - F = 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 = ilaplace(L,y): F là hàm của y thay thế biến mặc nhiên t.
  - F = 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.

VÝ dô :

```
>>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 dông ®å ho¹:

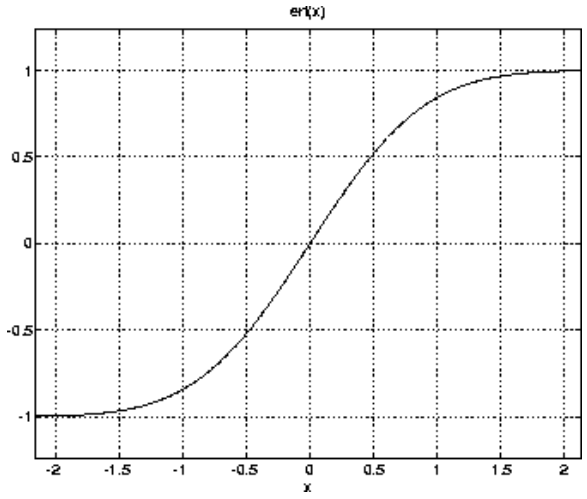
- **ezplot(vÏ ®-êng)**

  - ezplot(f): VÏ hµm $f = f(x)$ với miền mặc nhiên $-2\pi < x < 2\pi$.
  - ezplot(f,[min,max]) : Vẽ hàm f = f(x) trong miền giá trị [min,max] của biến.
  - ezplot(x,y): Vẽ đường cong ham số x = x(t); y = y(t) với biến mặc nhiên

$0 < t < 2\pi$.

Ví dụ:

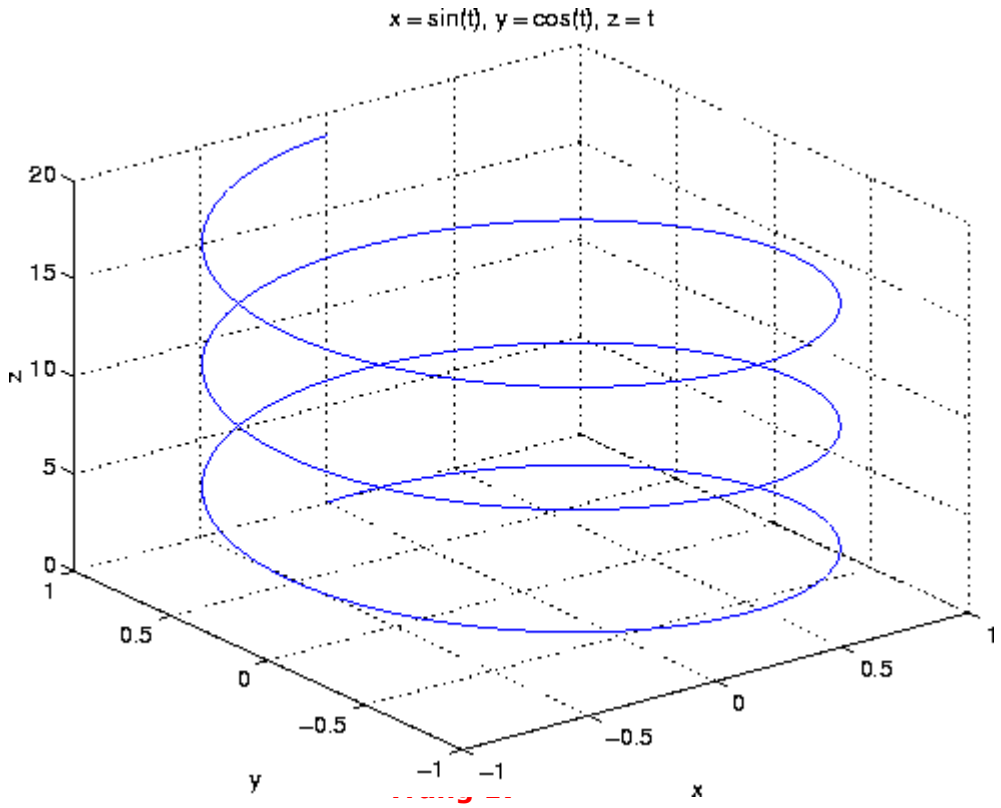
```
>>syms x
>>ezplot(erf(x))
>>grid
```



- **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 tmin < t < tmax.

VÝ dô:

>>syms t; ezplot3(sin(t), cos(t), t,[0,6*pi])

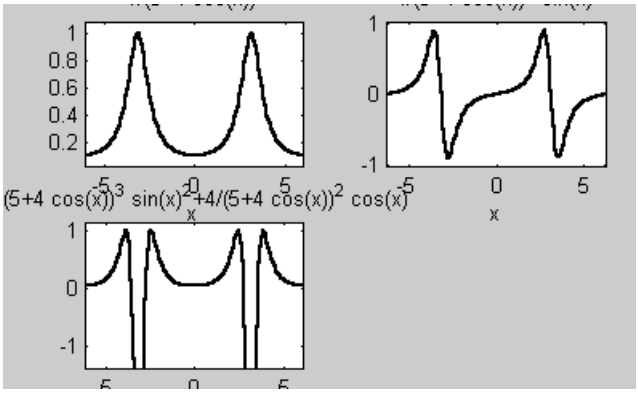


- **Ví dụ :** Cho hàm f(x) = $\frac{1}{5+4\cos(x)}$ với x∈[a,b].

  ➢ Vẽ đồ thị và các đạo hàm bậc 1, bậc 2. miền xác định mặc nhiên trong symbolic Matlab là $-2\pi \le x \le 2\pi$.

Hãy vào cửa sổ soạn thảo và trong cửa sổ này ta viết chương trình như sau:

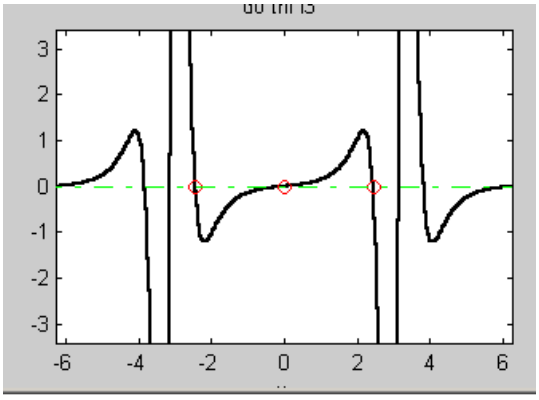
```
syms x
f1 = 1/(5+4*cos(x));
f2 = diff(f1);
% dao ham bac 1 cua f1.
f3 = diff(f2);
% dao ham bac 2 cua f1.
subplot(2,2,1)
ezplot(f1)
subplot(2,2,2)
ezplot(f2)
subplot(2,2,3)
ezplot(f3)
```



➢ 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:

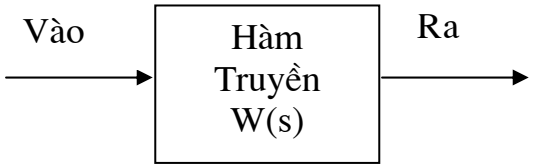
```
syms x
f = 1/(5+4*cos(x));
f3 = diff(f,3);
% dao ham bac 3 cua f.
pretty(f3);
%dua ve dang quyen toan hoc
f3 = simplify(f3);
pretty(f3);
z = solve(f3);
%Giai phuong trinh f3 = 0
format;
%lay 5 chu so so le
zr = double(z);
%chuyen ma tran, bieu thuc symbolic ve dang so
ezplot(f3)
%ve ham f3
hold on;
```



```
%luu do thi da ve
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-.')
```

♣6 HỆ THỐNG ĐIỀU KHIỂN
(SYSTEM CONTROL)

Hệ thống điều khiển có thể được mô tả bằng sơ đồ cấu trúc sau:



Khi quan hệ giữa lượng ra và lượng vào được mô tả bởi hàm truyền W(s)

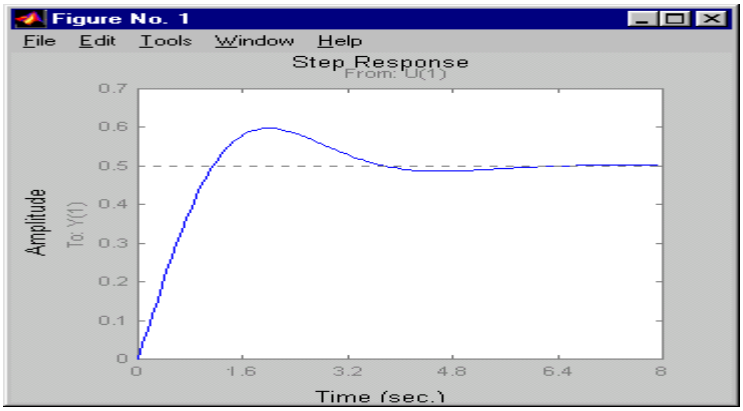
$$W(s) = \frac{b_0 s^m + b_1 s^{m-1} + ... + b_m}{a_0 s^n + a_1 s^{n-1} + ... + a_n}$$

VD cho hệ thống điêù khiển có hàm truyền:

$$W(s) = \frac{s+2}{2s^2 + 3s + 4}$$

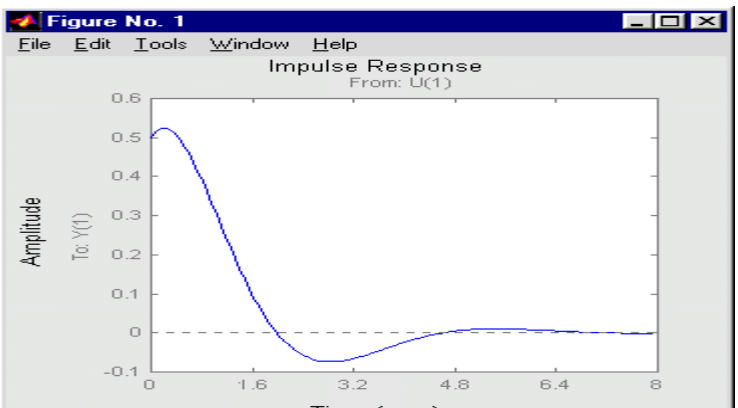
Mô phỏng hệ thống bằng hàm quá độ:

```
>> W = tf ([1   2],[2      3       4]);
( Hoặc >> tuso = [1    2];
          >> mauso = [ 2          3          4]
         >> W=tf(tuso,mauso);          )
>> step(W)
```



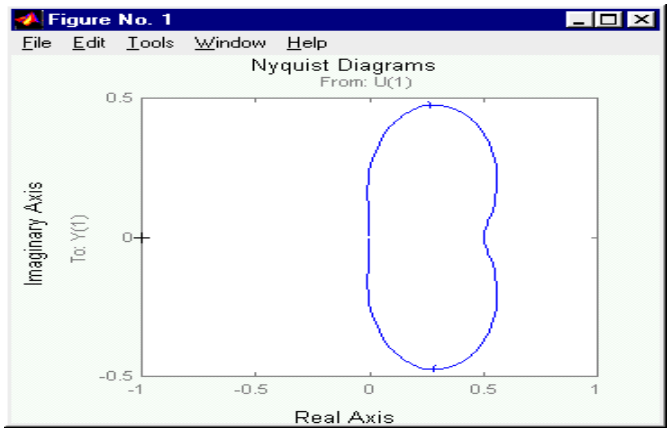
1. Mô phỏng hệ thống bằng hàm quá độ xung:

```
>> W = tf ([1   2],[2      3       4]);
>> impulse(W)
```



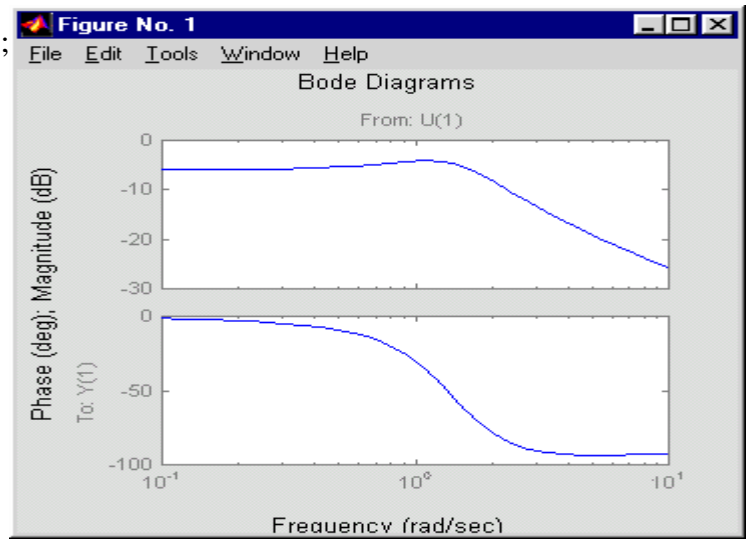
2. *Mô phỏng hệ thống bằng đặc tính tần số biên pha:*

```
>> W = tf ([1   2],[2      3      4]);
>> nyquist(W)
```



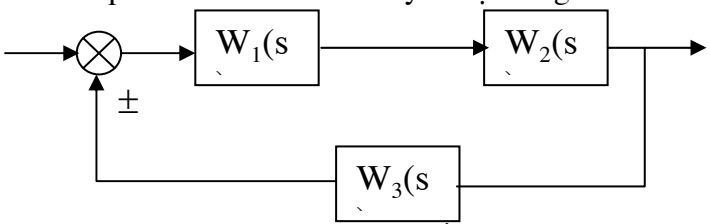
3. *Mô phỏng hệ thống bằng các đặc tính biên độ tần số và pha tần số:*

```
>> W = tf ([1   2],[2      3      4]);
>> Bode(W)
```



Chú ý: khi hệ thống gồm nhiều khâu mắc nối tiếp ta có thể tính hàm truyền hệ thống:

```
>> W = W1*W2*....Wn
```



4. *Mô phỏng hệ thống có phản hồi.*

a. hệ thống có phản hồi dương: sau khi đã khai báo xong các hàm truyền W_1, W_2, W_3. Hàm truyền của hệ thống được tính như sau:

```
>> W = feedback ( W1*W2 , W3  , 1 )
```

(số 1 nhằm chỉ phản hồi dương)

b. hệ thống có phản hồi âm: sau khi đã khai báo xong các hàm truyền W_1, W_2, W_3. Hàm truyền của hệ thống được tính như sau:

>> W = feedback (W_1*W_2 , W_3)

5. Tìm điểm cực của hệ thống khi biết hàm truyền W.

điểm cực = pole(W)

```
>> W = tf ([1      2],[2     3      4]);
>> D = pole(W)
    D =
        -   0.7500 + 1.1990i
        -   0.7500 - 1.1990i
```

7. Từ hàm truyền hệ thống dạng $W(s) = \frac{b_0 s^m + b_1 s^{m-1} + ... + b_m}{a_0 s^n + a_1 s^{n-1} + ... + a_n}$

đổi sang dạng zero - pole - gain. $W(s) = \frac{K(c_0 - s)(c_1 - s)...}{(d_0 - s)(d_1 - s)...}$

Tên = zpk (W)

```
>> W = tf ([1      2],[2     3      4]);
>>  R = zpk(W)
    zero / pole / gain
                0.5(s + 2)
            (s^2 + 1.5s + 2)
```

9. Tìm giá trị điểm cực và zero của hệ thống dưới dạng hàm truyền và bố trí chúng trên mặt phẳng phức.

Tìm các giá trị sau khi đã có hàm truyền W hệ thống.

[p z] = pzmap (W)

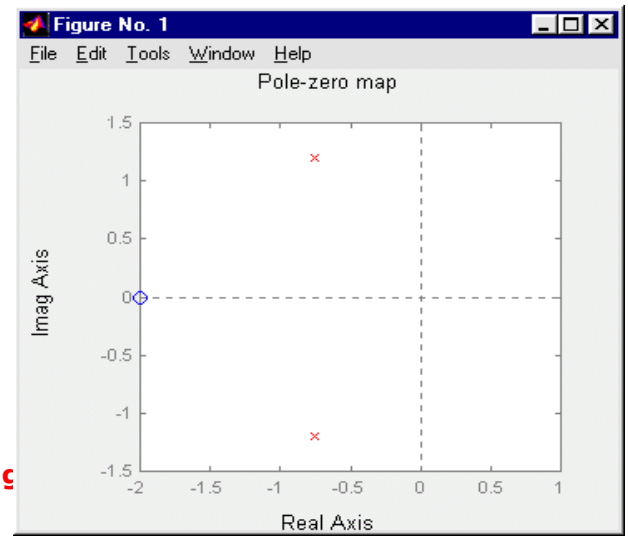
p =

z =

VD

```
>> W = tf ([1  2],[2     3      4]);
>> [p      z] = pzmap (W)
        p =

          -0.7500 + 1.1990i
          -0.7500 - 1.1990i
```



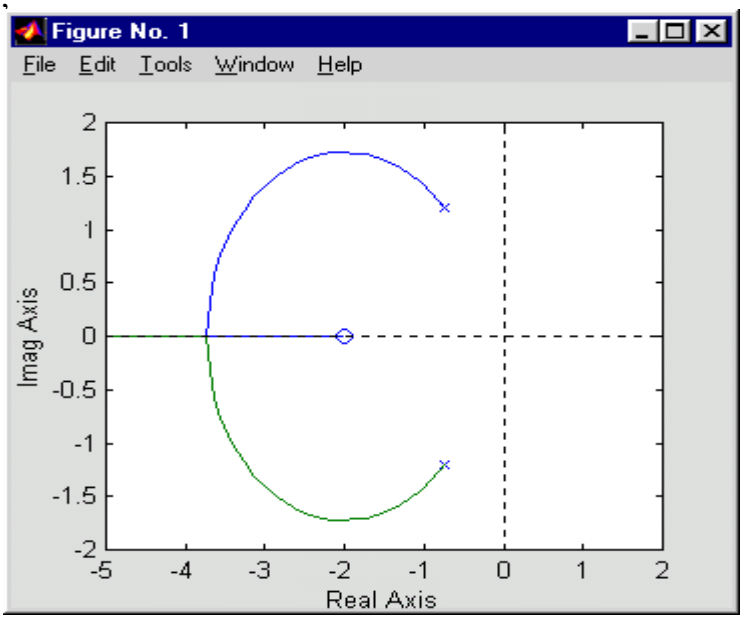
```
z =
    -2
```

Xem bố trí trên mặt phẳng phức.

```
>> W = tf ([1   2],[2      3       4]);
>> pzmap (W)
```

10. Tìm qũi đạo nghiệm số trên mặt phẳng phức.

```
>> W = tf ([1   2],[2      3       4]);
>> rlocus (W)
```

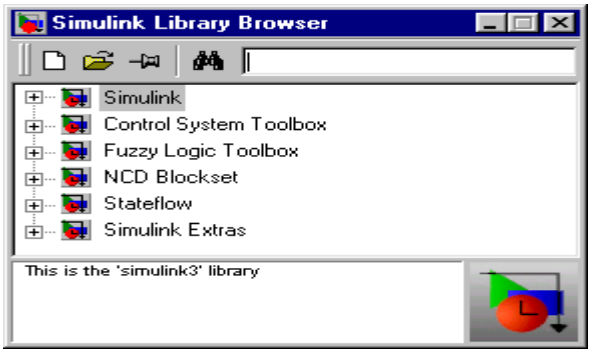


♣7 MÔ PHỎNG HỆ THỐNG DÙNG SƠ ĐỒ KHỐI (SIMULINK)

Ngoài việc mô phỏng hệ thống bằng câu lệnh trong MATLAB còn cho phép ta mô phỏng hệ thống dưới dạng sơ đồ khối. Đa số các hệ thống sau khi đã mô tả toán học, thành lập hàm truyền thường đưa về dạng sơ đồ cấu trúc. Vì vậy mô phỏng hệ thống bằng SIMULINK giúp ta dễ tưởng tượng và quan sát hệ thống hơn.

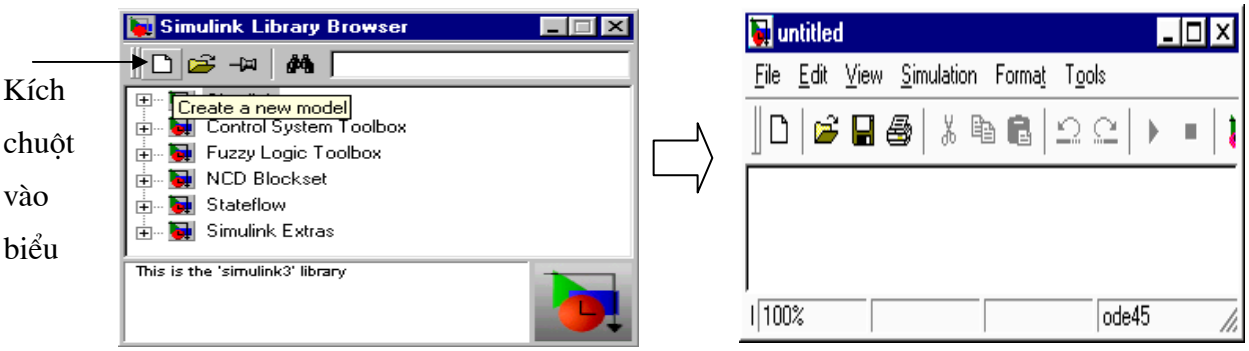
I. Các bước để mô phỏng hệ thống bằng SIMULINK.

1. *Bước 1*: Gọi phần ứng dụng SIMULINK.
   Từ dấu nhắc lệnh của MATLAB ta gõ dòng lệnh SIMULINK
   >> SIMULINK → Simulink Library Browser



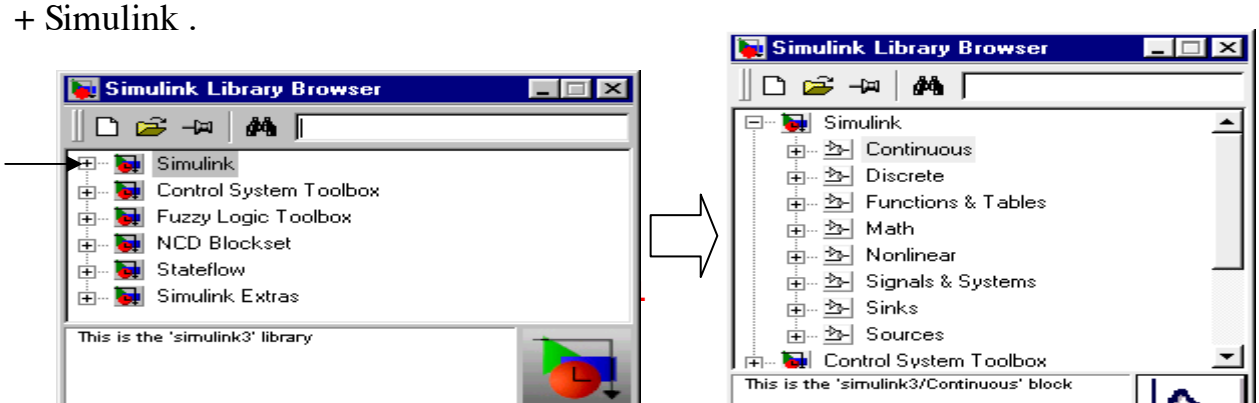
2. *Bước 2*: mở cửa sổ làm việc

Kích chuột vào biểu tượng create a new model trong cửa sổ Simulink Library Browser để mở cửa sổ mới (Cửa sổ mà chúng ta sẽ xây đựng mô hình mô phỏng nó có tên là untitled)



3. *Bước 3:* Mở thư viện chính SIMULINK bằng cách kích chuột vào biểu tượng

+ Simulink .



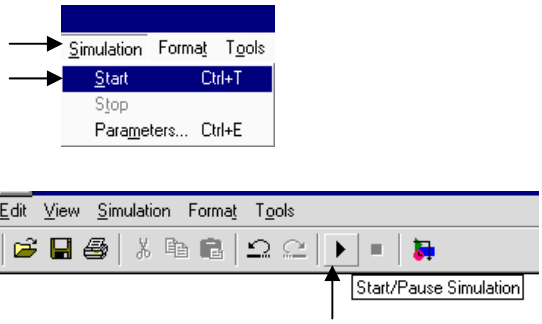
Trong thư viện chính này bằng cách tương tự ta có thể mở các thư viện con, chọn các khối cần thiết và rê chuột đưa chúng ra cửa sổ làm việc.

4. Bước 4: nối các khối theo sơ đồ cấu trúc.
Sau khi các khối đã được đưa ra cửa sổ làm việc ta dùng chuột để nối các khối theo đúng sơ đồ cấu trúc cần mô phỏng.

5. Bước 5: Mở các khối bằng cách kích đúp chuột vào khối đó. Lúc này sẽ xuất hiện cửa sổ Block Parameters .... Tại đây ta có thể thay đổi dữ liệu theo mong muốn.
6. Bước 6: Thực hiện quá trình mô phỏng bằng các cách sau chọn các công việc sau trong cửa sổ làm việc.

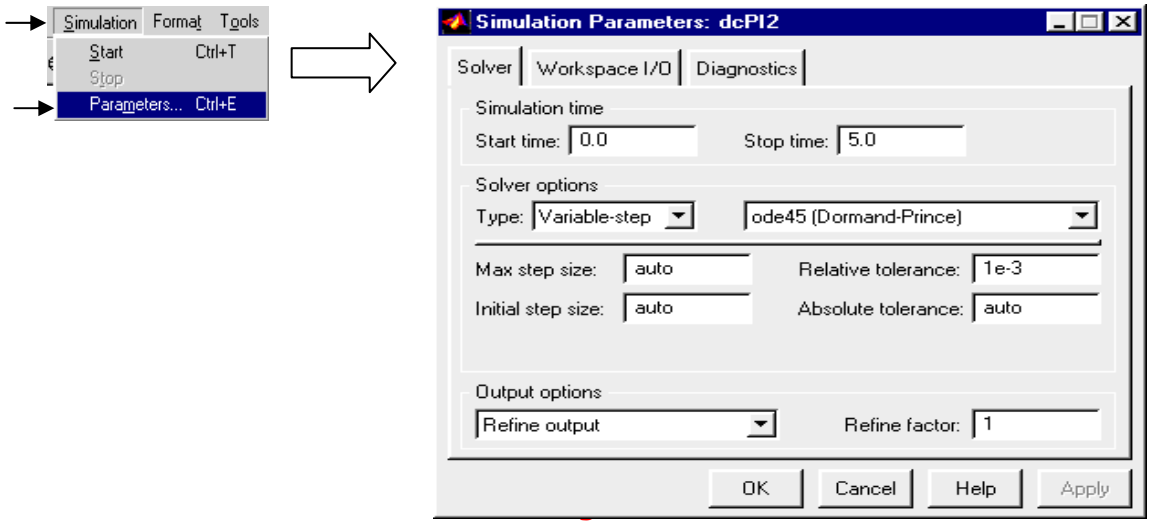
- Simulation / start.

- Kích vào biểu tượng
  Start / Pause Simulation.



7. Bước 7: Ta có thể thay đổi thông số của quá trình mô phỏng

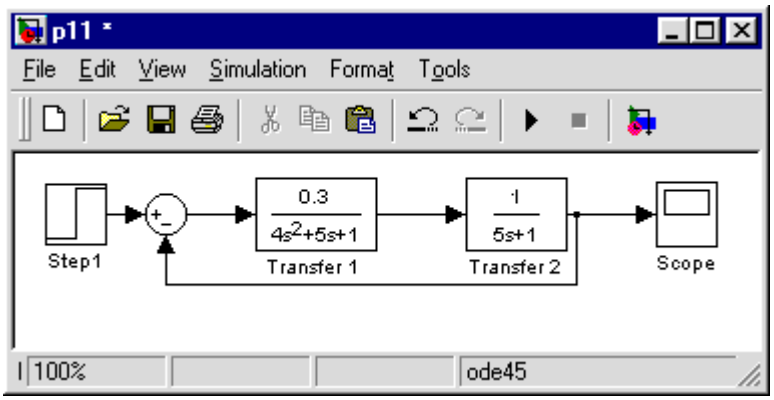
- Simulation / Parameters → Simulation Parameters...

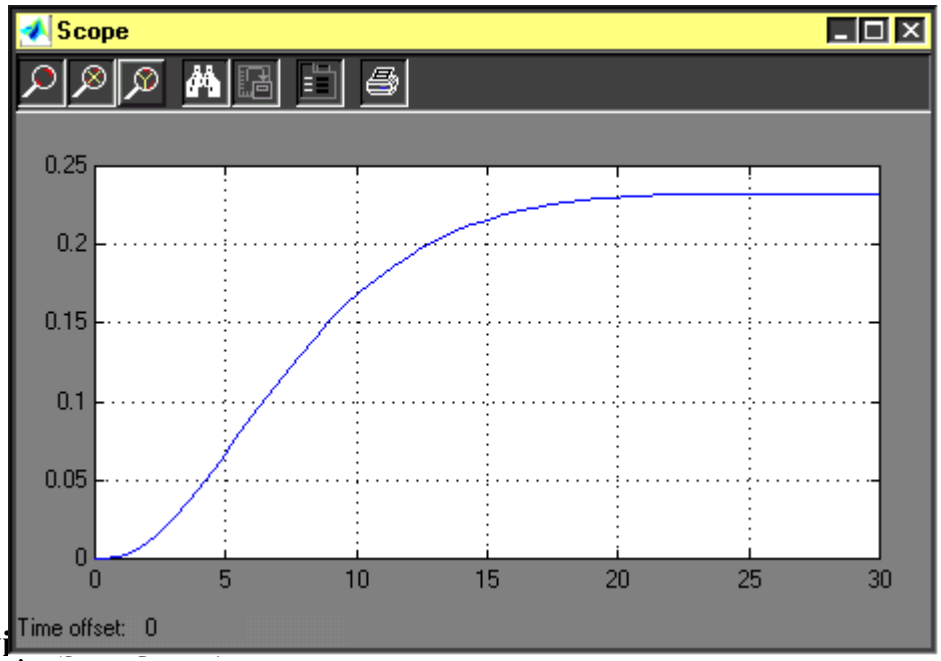


Thay đổi thời gian bắt đầu tại ô Start time
Thay đổi thời gian kết thúc tại ô Stop time

8. *Bước 8:* Có thể ghi lại mô hình mô phỏng vừa tạo được bằng cách chọn Save trong menu FILE hoặc kích vào biểu tượng đĩa mềm trên thanh công cụ của cửa sổ làm việc.

VD sơ đồ và đồ thị mô phỏng của hệ thống điều khiển như hình vẽ :





II. Thư vi

1. Continuous: Các khối liên tục

- Derivative: Khối đạo hàm
- Integrator: Khối tích phân

- State - Space: Phương trình trạng thái
- Transfer Fcn: Hàm truyền tuyến tính
- Transport Delay: Làm trễ tín hiệu
- Zero - Pole: Hàm truyền dạng Zero-pole

2. Discrete: Các khối gián đoạn

3. Math : các hàm toán học

- Gain : Khối khuyếch đại
- Sum: Khối cộng tín hiệu.

4. Nonlinear : các khối phi tuyến

5. Sinks: Khối quan sát: Thông thường hay sử dụng khối Scope để quan sát quá trình mô phỏng.

6. Sources: Khối nguồn

- Constant: Khối hàm không đổi.
- Step: Khối hàm bước nhẩy.
- Sine Wave : Tạo tín hiệu hình Sin.
- Signal Generator: Máy phát tín hiệu.

Serial Number:
11-10224-43044-38818-03811-61063-11324
11-30608-63868-33547-31267-29961-41669
11-57331-07688-52528-02324-05637

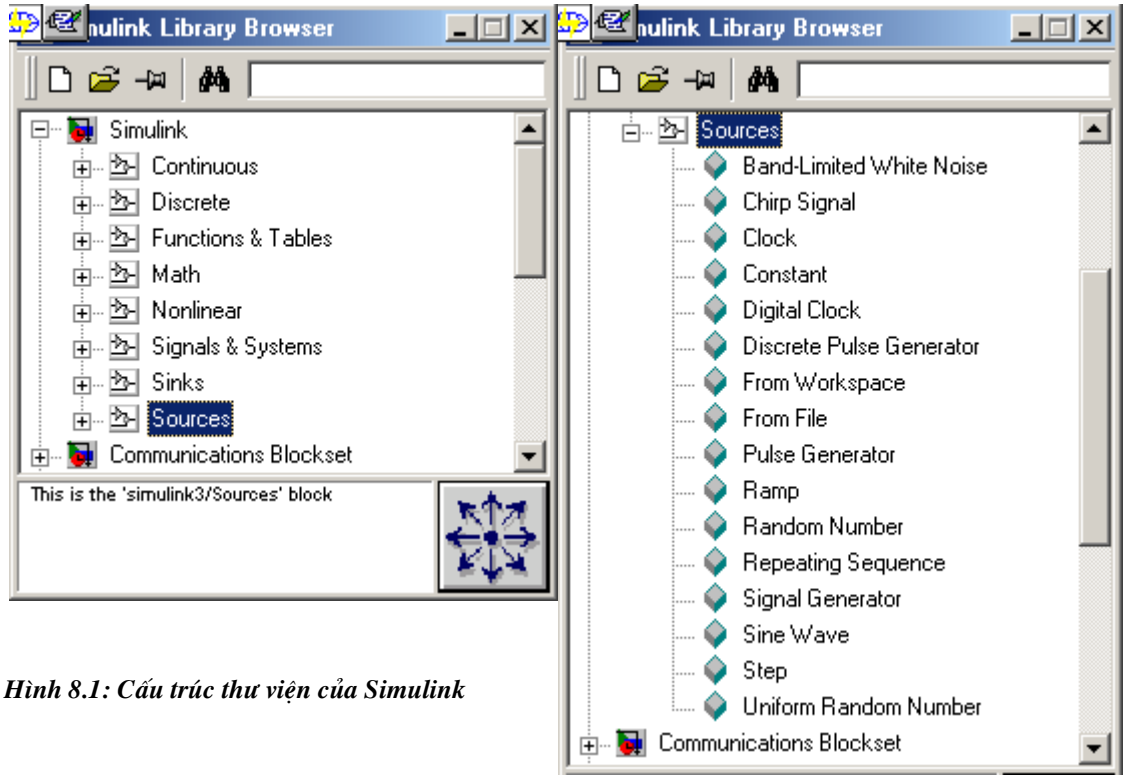
CHƯƠNG 8
MÔ PHỎNG HỆ THỐNG ĐỘNG HỌC SỬ DỤNG SIMULINK

Simulink là phần mềm mô phỏng các hệ thống động học trong môi trường Matlab. Đặc điểm của Simulink là lập trình ở dạng sơ đồ cấu trúc của hệ thống. Nghĩa là , để mô phỏng một hệ thống đang được mô tả ở dạng phương trình vi phân, phương trình trạng thái, hàm truyền đạt hay sơ đồ cấu trúc thì chúng ta cần chuyển sang chương trình Simulink dưới dạng các khối cơ bản khác nhau theo cấu trúc cần khảo sát. Với cách lập trình như vậy người nghiên cứu hệ thống sẽ thấy trực quan và dễ hiểu.

Trong môi trường Simulink có thể tận dụng được các khả năng tính toán, phân tích dữ liệu, đồ hoạ của Matlab và sử dụng các khả năng của toolbox khác như toolbox xử lý tín hiệu số, logic mờ và điều khiển mờ, nhận dạng, điều khiển thích nghi, điều khiển tối ưu …v v.Việc Simulink kết hợp được với các toolbox đã tạo ra công cụ rất mạnh để khảo sát động học các hệ tuyến tính và phi tuyến trong một môi trường thống nhất.

8.1 Thư viện khối chuẩn của Simulink:

Môi trường lập trình Simulink được tạo nên từ các khối chuẩn trong các thư viên của Simulink. Các thư viện Simulink bao gồm các khối sau:



Hình 8.1: Cấu trúc thư viện của Simulink

8.1.1 Thư viện các khối Sources (Khối phát tín hiệu):

Thư viện này gồm các khối tạo nguồn tín hiệu khác nhau. Trong thư viện Sources có các khối như trong bảng dưới đây:

Tên khối	Chức năng

Band-Limited White Noise	Đưa nhiễu trắng vào hệ
Chirp- Signal	Tạo sóng sin tần số bất kỳ
Clock	Cấp thời gian thực
Constant	Tạo đại lượng không đổi, tín hiệu đầu vào không đổi
Digital Clock	Cấp thời gian, với thời gian lấy mẫu
Discrete Pulse Generator	Khối phát tín hiệu dao động rời rạc
From Workspace	Đọc dữ liệu trong vùng nhớ đệm
From file	Đọc dữ liệu từ một file
Pule Generator	Tạo các xung với các chu kỳ khác nhau
Ramp	Phát tín hiệu đường y= ax +b
Random Number	Tạo các số ngẫu nhiên phân bố chuẩn
Repeating Sequence	Tạo tín hiệu tuỳ ý lặp lại theo chu kỳ
Signal Generator	Tạo các dạng tín hiệu khác nhau
Sine Wave	Tạo tín hiệu hình sin
Step	Tạo tín hiệu dạng hàm bậc thang đơn vị (hàm bước nhảy)
Uniform Random Number	Tạo các số ngẫu nhiên phân bố đều

8.1.2 Thư viện các khối Sinks.

Ở đây gồm các khối dùng để hiển thị hoặc ghi lại kết quả mô phỏng ở đầu ra một khối trong hệ thống được khảo sát. Trong thư viện Sinks có các khối sau:

Tên khối	Chức năng
Display	Hiển thị tín hiệu dưới dạng chữ số
Scope	Khối quan sát
Stop simulation	Ngừng quá trình mô phỏng khi lượng vào khác không
To File	Ghi dữ liệu vào File
To Workspace	Ghi dữ liệu vào vùng làm việc
XY graph	Hiển thị đồ thị XY của tín hiệu trên cử sổ đồ thị MATLAB

8.1.3 Thư viện các khối Dicrete (tín hiệu rời rạc hay tín hiệu số Z)

Thư viện này có các khối cơ bản của hệ thống rời rạc, các khối tính toán trong miền thời gian rời rạc. Cụ thể bao gồm các khối như trong bảng sau:

Tên khối	Chức năng
Discrete Transfer Ecn	Biểu diễn hàm truyền trong hệ rời rạc
Discrete Zero- pole	Biểu diễn hàm truyền trong hệ rời rạc thông qua Pole và Zero
Discrete -Filter	Biểu diễn các bộ lọc HR và FIR
Discrete State- Space	Biểu diễn hệ thống trong không gian trạng thái rời rạc
Discrete- Time Integrator	Biểu diễn tích phân tín hiệu rời rạc theo thời gian
Fist Order Hold	Khâu tạo dạng bậc nhất
Unit Display	Hiển thị tín hiệu trong một chu kỳ rời rạc
Zero order Hold	Khâu tạo dạng bậc thang không

8.1.4 Thư viện các khối Continuous.

Trong thư viện này có các khối của hệ thống liên tục tuyến tính, các khối biểu diễn các hàm tuyến tính chuẩn. Thư viện Linear gồm các khối sau:

Tên khối	Chức năng
Derivative	Tính vi phân theo thời gian của lượng vào (d/dt)
Integrator	Tích phân tín hiệu
Memory	Bộ nhớ ghi lại dữ liệu
State- Space	Biểu diễn hệ thống trong không gian trạng thái tuyến tính
Transfer Fcn	Hàm truyền đạt tuyến tính của các khâu hoặc hệ thống
Transport Delay	Giữ chậm lượng vào theo giá trị thời gian cho trước.
Variable Transport Delay	Giữ chậm lượng vào với khoảng thời gian biến đổi
Zero- pole	Hàm truyền theo Pole(điểm cực) và Zero(điểm không)

8.1.5 Thư viện các khối Nonlinear (các khâu phi tuyến).

Thư viện Nonlinear có các khối biểu diễn các hàm phi tuyến điển hình các khối trong hệ thống phi tuyến. Cụ thể bao gồm các khối sau:

Dead Zone	Mô tả vùng không nhạy (vùng chết).
Quantizer	Lượng tử hoá tìn hiệu vào trong các khoảng xác định.
Rate Limiter	Hạn chế phạm vi thay đổi của tín hiệu
Relay	Khâu rơle.
Saturation	Khâu bão hoà tín hiệu (khâu hạn chế).
Switch	Chuyển mạch giữa hai lượng vào.

8.1.6 Thư viên khối Signal & System:

Thư viện Signal & System có các khối biểu diễn tín hiệu và hệ thống. Cụ thể bao gồm các khối chính như sau:

Tên khối	Chức năng
Sub&Systems	Xây dựng hệ thống con bên trong hệ thống lớn
In1	Tạo cổng vào cho một hệ thống
Demux (phân kênh)	Tách tín hiệu véctơ thành các tín hiệu vô hướng
Mux (Dồn kênh)	Gộp các tín hiệu thành một véctơ
Out1	Tạo cổng ra cho một hệ thống

8.1.7 Thư viện chứa các khối toán học Math:

Thư viện Math có các khối biểu diễn hàm toán học. Cụ thể bao gồm các khối chính như sau:

Tên khối	Chức năng
Abs	Biểu diễn giá trị tuyệt đối của lượng vào
Combuanatoril logic	Biểu diễn bảng chân lý.
Dot product	Nhân giữ hai véctở
Product	Thực hiện nhân các lượng vào
Gain	Bộ (khâu) khuyếch đại
Matrix gain	BKĐ có hệ số khuyếch đại là một Ma trận
Math function	Các hàm toán học
MinMax	Tìn giá trị min, max

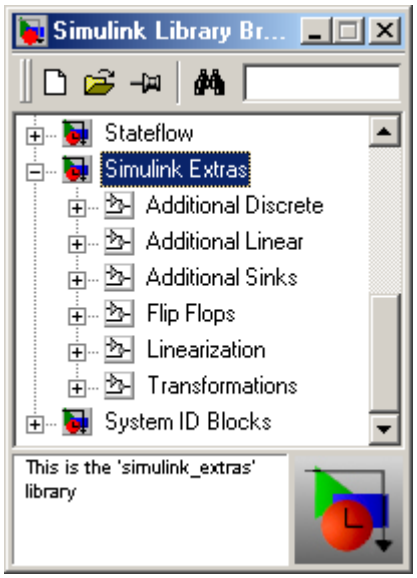
Relational	Toán tử quan hệ
Sum	Tính tổng của các lượng vào
Trigonometric Function	Hàm lượng giác

8.1.8 Thư viện chứa các khối Function & Tables:

Tên khối	Chức năng
Fcn	Ứng dụng biểu thức toán nhất định cho lượng vào.
Matlab Fcn	Ứng dụng hàm Matlab cho lượng vào.
look- Up Table 2-D	Biểu diễn tuyến tính từng đoạn của hai lượng vào
S -Function	Đưa một S-Function vào trong một khối

8.2 Thư viện các khối mở rộng của Simulink:

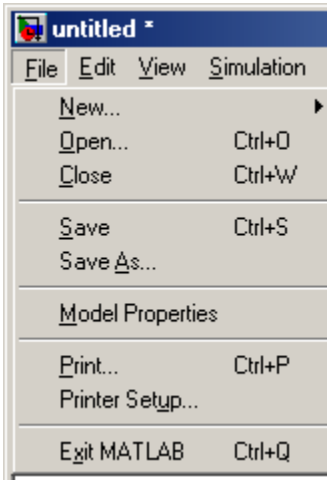
Additional Discrete: Khối mở rộng khối tín hiệu rời rạc.
Additional linear: Khối mở rộng khối tín hiệu tuyến tính
Additional Sinks: Khối mở rộng khối quan sát.
Filp Flops: Khối mở rộng chứa khối Trigơ.
Linearization: Khối mở rộng tuyến tính hoá.
Transformations: Khối mở rộng các khối biến đổi toán học.



Hình 8.2: Cấu trúc thư việnmở rộng của Simulink

8.3 Các trình đơn thông dụng của Simulink (cửa sổ làm việc untitled):

Các trình đơn của Simulink nằm ở phía trên cửa sổ làm việc. Khi một mục trong trình đơn mà theo sau có mũi tên mầu đen trỏ sang phải thì sẽ mở ra trình đơn con, còn khi phía sau là dấu ba chấm thì sẽ mở ra một hộp thoại. Những mục đứng riêng biệt thì kết quả sẽ là một tác vụ trực tiếp. *Trong phạm vi phần này chỉ giới thiệu các trình đơn thông dụng trong cửa sổ untitled.*

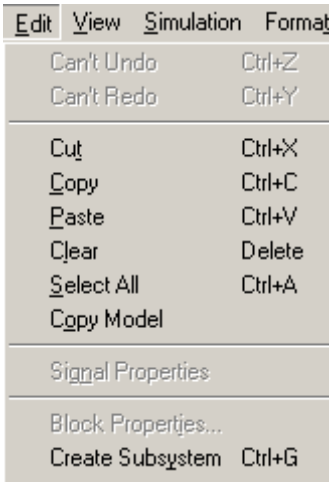


8.3.1 Trình đơn File:

- **New**:

- New/ untitled: Mở cửa sổ làm việc mới để xây dựng một hệ thống mới
- New/ library: Mở cửa sổ để tự xây dựng một thư viên riêng cho người sử dụng.

- **Open**: HIển thị một hộp thoại với danh sách các file đã lưu, ta tìm file cần mở rồi kích vào nút Open để mở file.
- **Close**: Đóng cửa sổ đang mở.
- **Save**: Ghi lại nội dung của một file.
- Save as: Ghi lại nội dung của một file mới.
- **Model Properties**: Các thông số của mô hình.
- **Print…**: In một sơ đồ khối.
- **Print Seup…**: Hiển thị danh sách lựa chọn máy in và kích thước giấy…..
- **Exit Matlab**: Thoát khỏi Matlab.

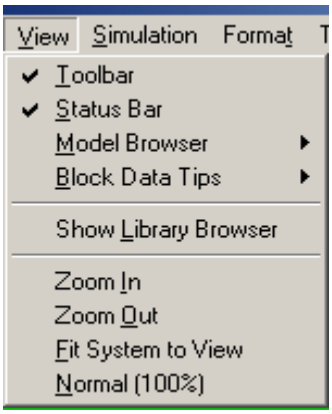


8.3.2 Trình đơn Edit:

- **Cut**: Di chuyển các đối tượng từ cửa sổ làm việc vào vùng nhớ Clipboard.
- **Copy**: Sao chép đối tượng vào vùng nhớ Clipboard.
- **Paste**: Dán nội dung từ Clipboard vào vị trí cần chèn đến.
- **Clear**: Xoá các đối tượng đã chọn.
- **Select All**: Chọn tất cả các đối tượng trong cửa sổ hiện hành.
- **Copy Model**: Sao chép mô hình trong cửa sổ hiện hành vào vùng nhớ Clipboard. Khi cần đem dán sang chương trình khác.
- **Create Subsystem**: Tạo một hệ thống con từ những đối tượng đã chọn trong cửa sổ hiện hành.

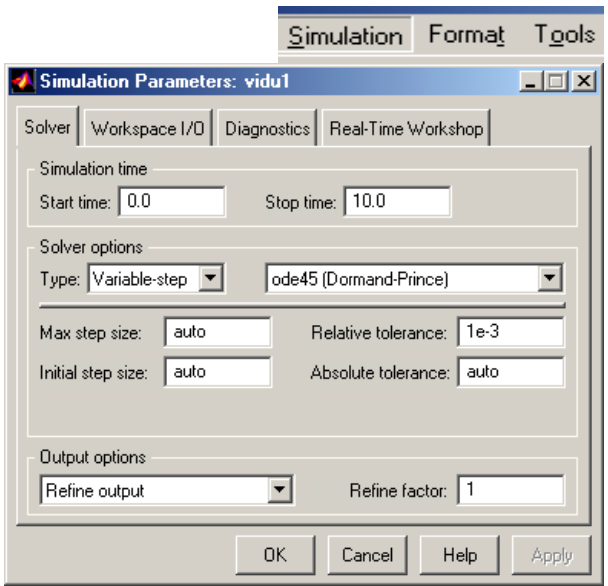
8.3.3 Trình đơn View:

- **Toolbar**: Hiển thị thanh công cụ.
- **Status bar**: Hiển thị thanh trạng thái.
- **Zoom in**: phóng to mô hình.
- **Zoom out**: Thu nhỏ mô hình.
- **Normanl(100%)**: Hiển thị mô hình ở 100%.



8.3.4 Simulation:

- **Start/ Stop**: Khởi động/ dừng mô phỏng.
- **Parameters…**: Đặt thông số cho quá trình mô phỏng.

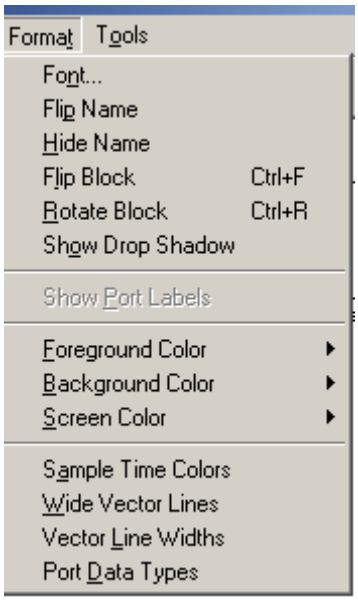


- Max step size: Bước tính lớn nhất.
- Initial step size: Bước tính lúc đầu.
- Relative tolerance: Sai số cho phép.
- Start time: Thời gian bắt đầu mô phỏng.
- Stop time: Thời gian kết thúc mô phỏng.

- Solver option: Đặt biến, thuật toán mô phỏng

8.3.5 Format:

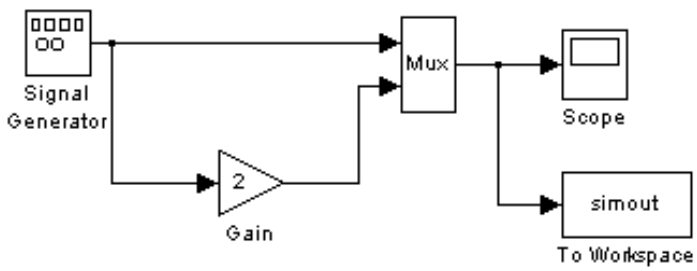
- **Font…**: Chọn kích thước, kiểu dáng, độ đậm nhạt của chữ.
- **Filp Name**: Chuyển vị trí trên, dưới tên của khối.
- **Hide Name**: Đặt ẩn tên của một khối.
- **Show Name**: Hiện tên của một khối.
- **Flip Block**: Xoay khối đi một góc180^0.
- **Rotate Block**: Xoay khối đi một góc 90^0.
- **Show Drop Shadow**: Tạo vết bóng cho khối Simulink.
- **Foreground Color**: Chọn mầu cho đường tín hiệu và đường khung các khối.
- **Background Color**: Chọn mầu nền cho các khối
- **Screen Color**: Chọn mầu nền khung cửa sổ.



8.4 Các bước thực mô phỏng hệ thống bằng Simulink:

Để hiểu rõ cách thức xây dựng một mô hình và cách thức chạy mô phỏng trong Simulink ta xét một ví dụ đơn giản, mô hình có các khối :

Signal Generator : Thuộc thư viện Sources
Gain : Thuộc thư viện Math
Mux : Thuộc thư viện Signals & Systems
Scope & To Workspace: Thuộc thư viện Sinks



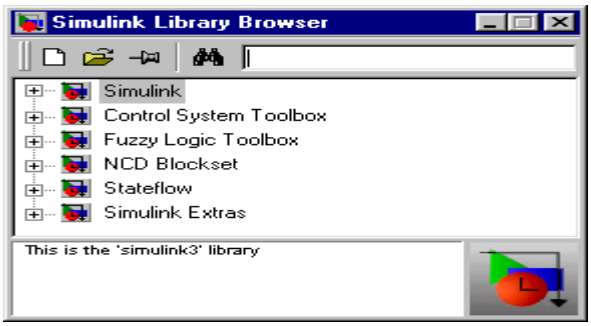
Hình 8.3: Cấu trúc mô hình cần mô phỏng

Bước 1: Gọi phần ứng dụng Simulink .

Kích chuột vào biểu tượng Simulink Library Browser 

Hoặc từ dấu nhắc lệnh trong cửa sổ Matlab Command Window ta gõ dòng lệnh simulink

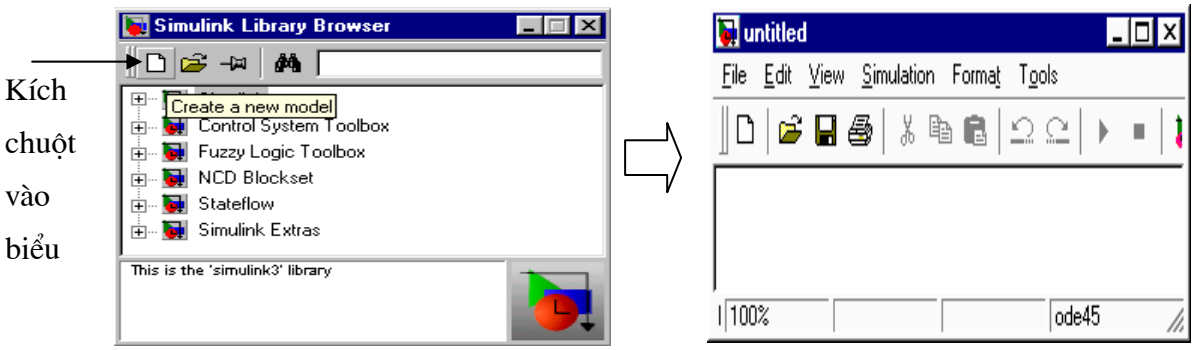
>> **simulink** → Simulink Library Browser



Hình 8.4: Cấu trúc thư viện của Simulink

Bước 2: mở cửa sổ làm việc:

Kích chuột vào biểu tượng create a new model trong cửa sổ Simulink Library Browser để mở cửa sổ mới (Cửa sổ mà chúng ta sẽ xây đựng mô hình mô phỏng nó có tên là untitled)

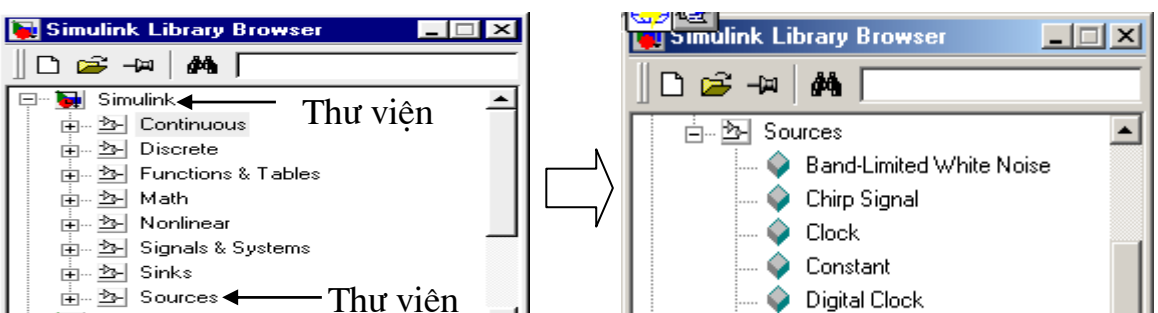


Hình 8.5: Cách thức tạo cửa sổ làm việc

Bước 3: Xây dựng mô hình Simulink:

Các thao tác tìm các khối để xây dựng mô hình như sau:

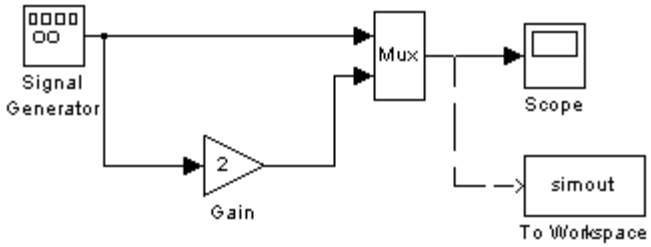
- Kích đúp chuột vào thư viện chính Simulink.
- Kích đúp chuột vào thư viện Sources.
- Kích và kéo thư viện khối Sin Wave sang cửa sổ làm việc (**untitled**)
- Cách thức xây dựng các khối còn lại làm tương tự



Bước 4: Nối các khối theo sơ đồ cấu trúc.

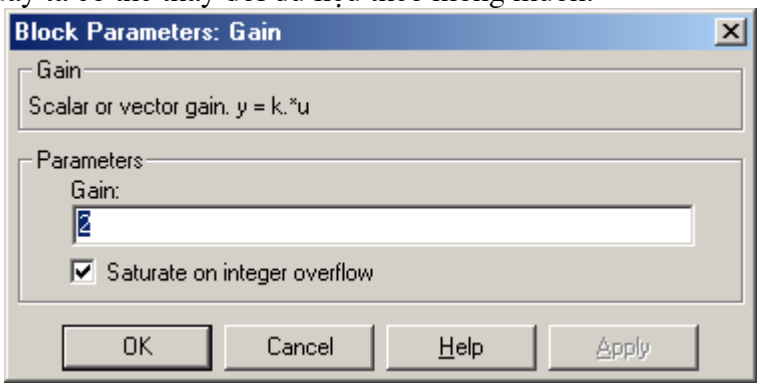
Sau khi các khối đã được đưa ra cửa sổ làm việc ta dùng chuột để nối các khối theo đúng sơ đồ cấu trúc cần mô phỏng, cách làm như sau:

- Nối giữa hai khối: Đưa chuột đến đầu vào hoặc đầu ra của một khối, khi con trỏ suy biến thành dấu cộng thì kích phím trái chuột rồi kéo trỏ chuột đến đầu vào hoặc đầu ra của khối của khối cần nối. Nếu ta nhả phím chuột trước đường nối các khối hoàn thành thì đoạn thẳng sẽ kết thúc bằng mũi tên chỉ hướng truyền tín hiệu.
- Trích đường nối giữa các khối: Kích phải chuột vào điểm cần trích rồi kéo chuột ta sẽ được một đường truyền tín hiệu.
- Thay đổi kích thước của các khối: Kích chuột vào khối sau đó đưa trỏ chuột đến góc của khối rồi kích và kéo theo chiều mũi tên để thay đổi kích thước.
- Di chuyển các khối:
- Copy các khối:

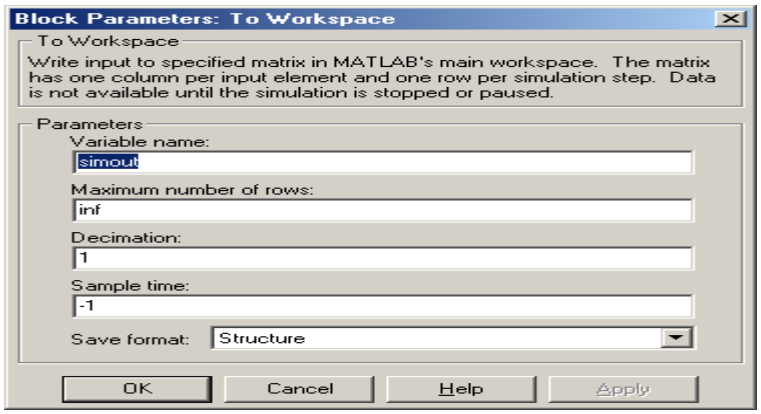


Hình 8.7: Cách nối các khối theo sơ đồ cấu trúc.

Bước 5: Mở các khối bằng cách kích đúp chuột vào khối đó. Lúc này sẽ xuất hiện cửa sổ Block Parameters .... Tại đây ta có thể thay đổi dữ liệu theo mong muốn.



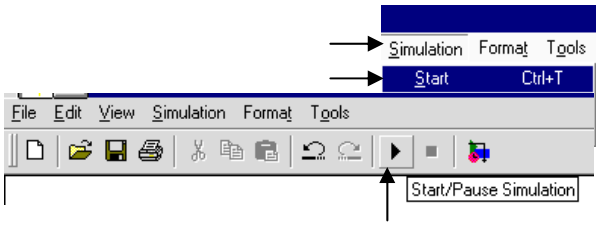
Hình 8.8: Khối thông số của khâu khuyếch đại.



Hình 8.9: Khối thông số của khâu To

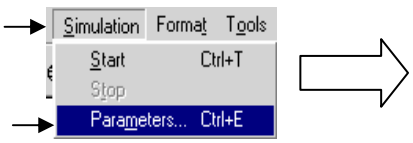
Bước 6: Thực hiện quá trình mô phỏng bằng các cách sau chọn các công việc sau trong cửa sổ làm việc.

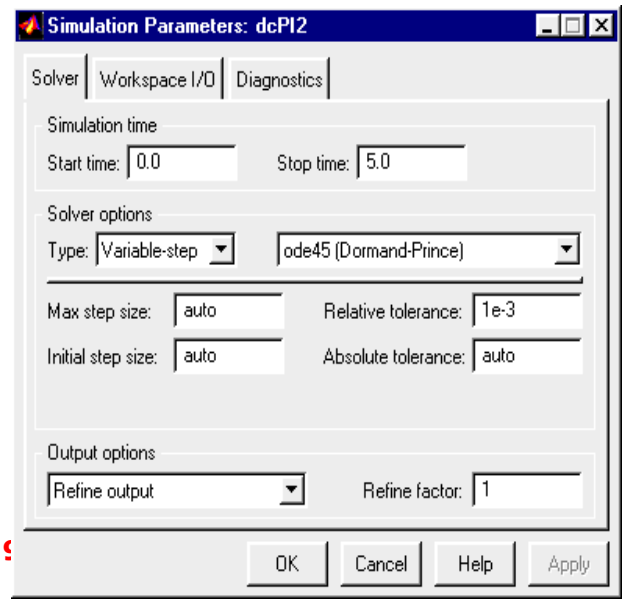
- Simulation / start.
- Kích vào biểu tượng
  Start / Pause Simulation.



Bước 7: Ta có thể thay đổi thông số của quá trình mô phỏng

- Simulation / Parameters → Simulation Parameters...

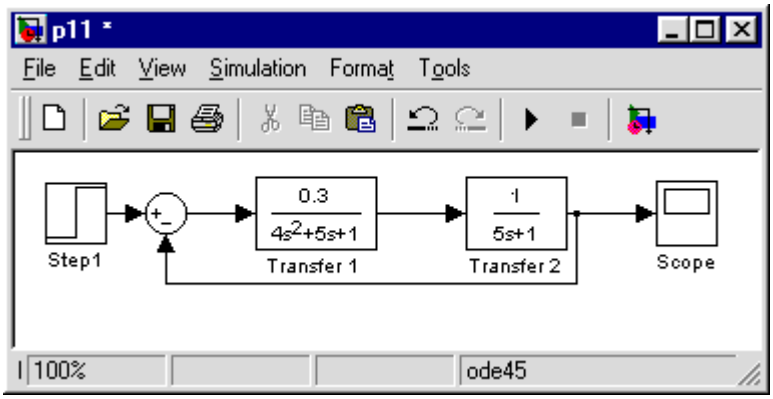


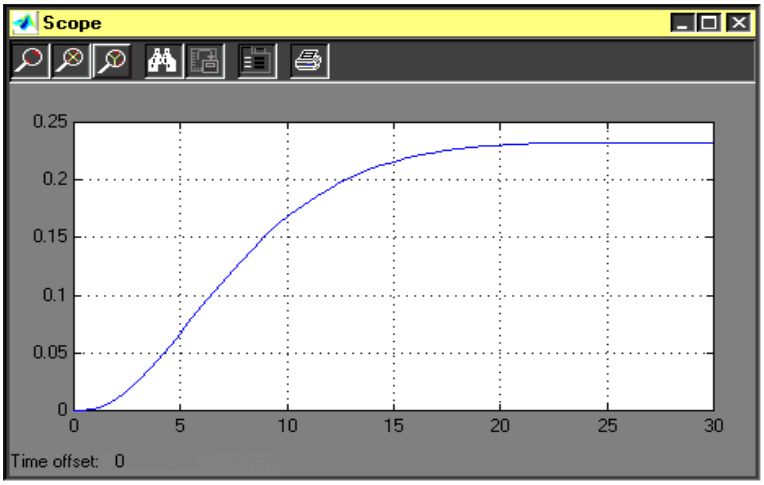


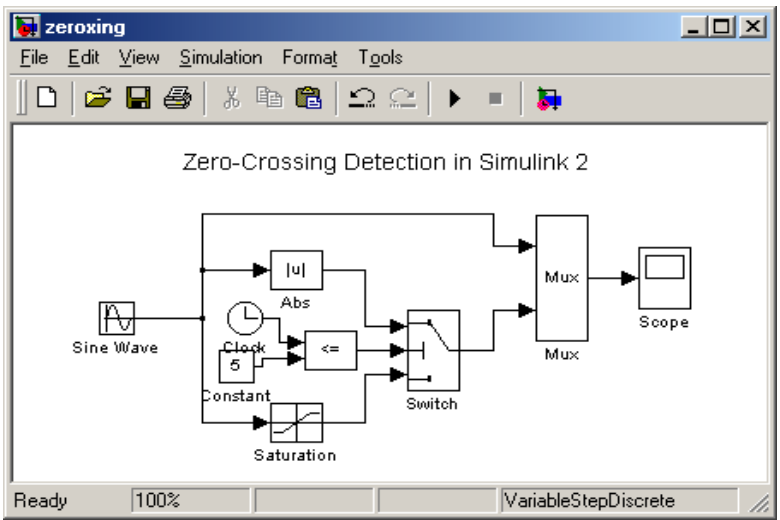
Hình 8.10: Hộp thoại đặt thông số mô

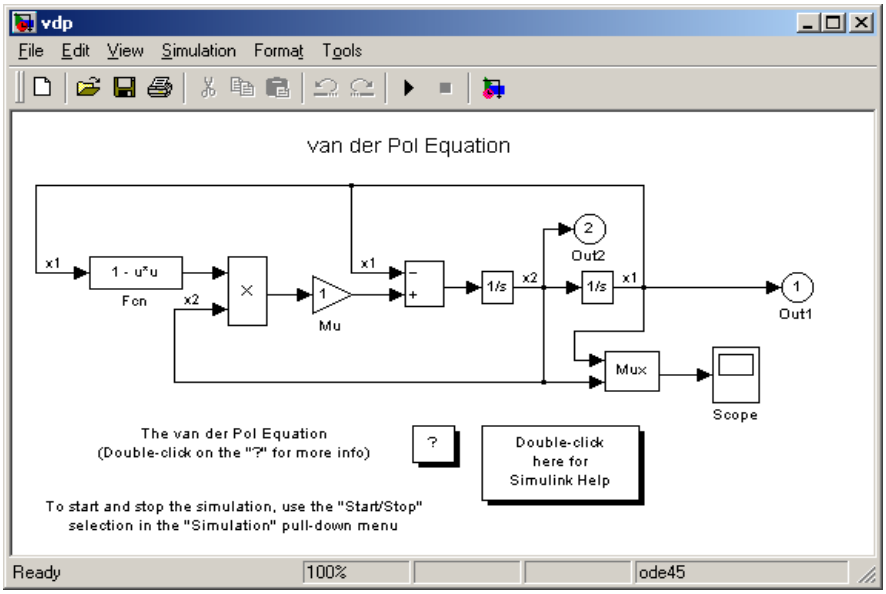
Bước 8: Có thể ghi lại mô hình mô phỏng vừa tạo được bằng cách chọn Save trong menu File hoặc kích vào biểu tượng đĩa mềm trên thanh công cụ của cửa sổ làm việc.

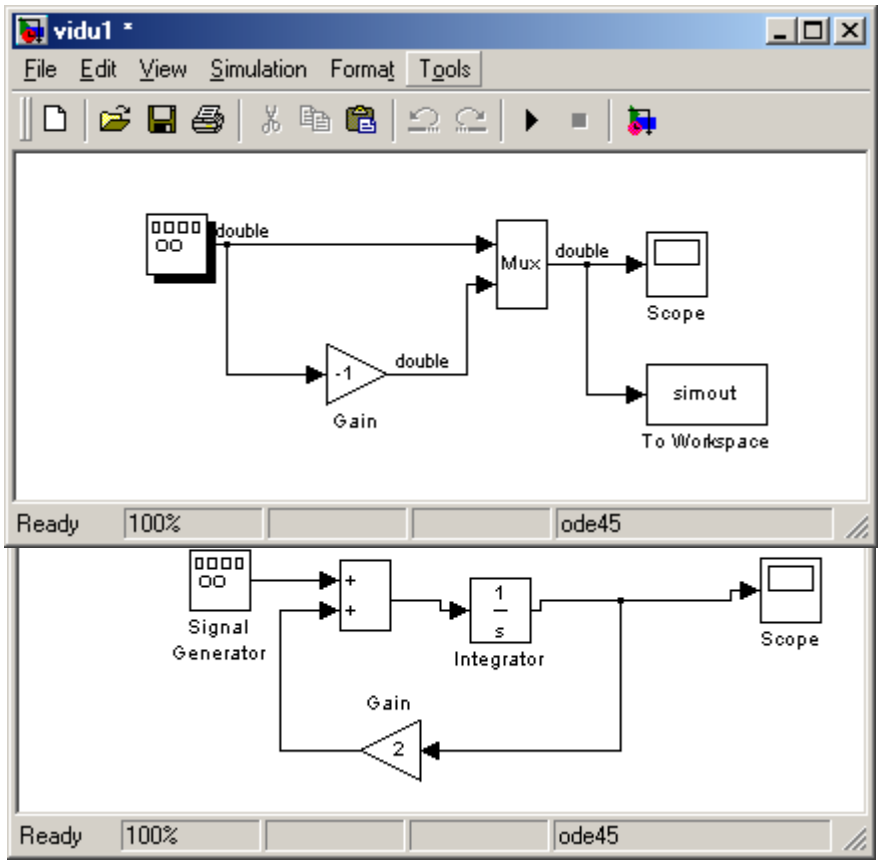
8.5 Các ví dụ:

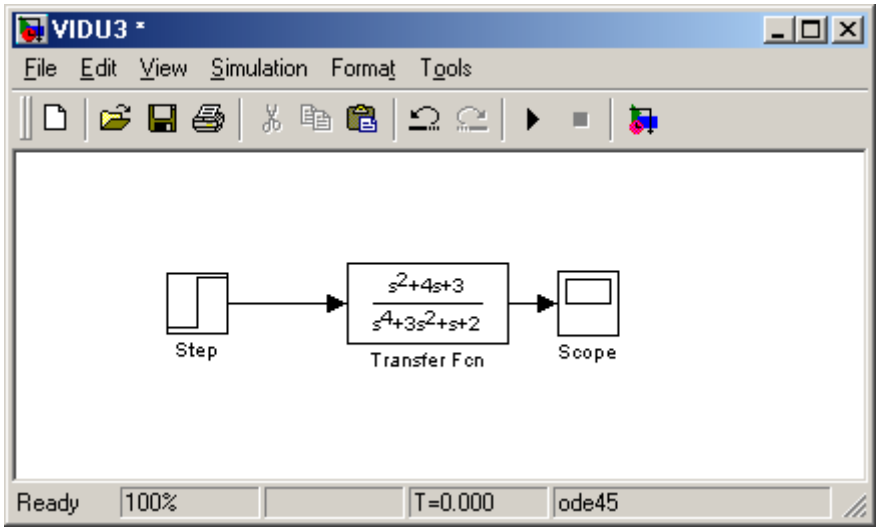


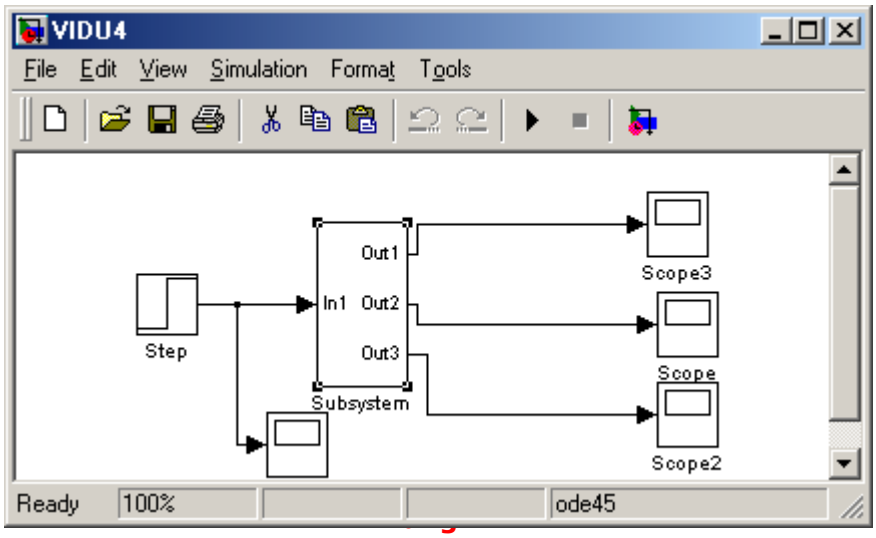


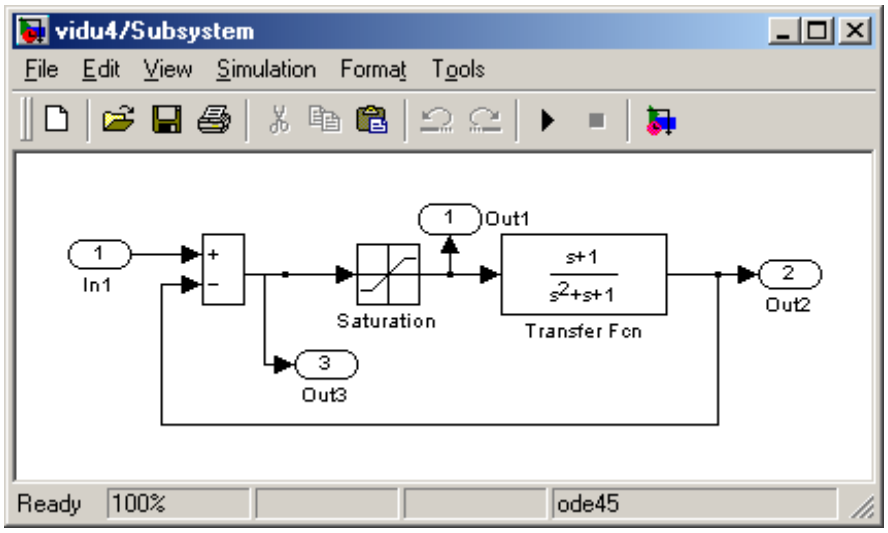


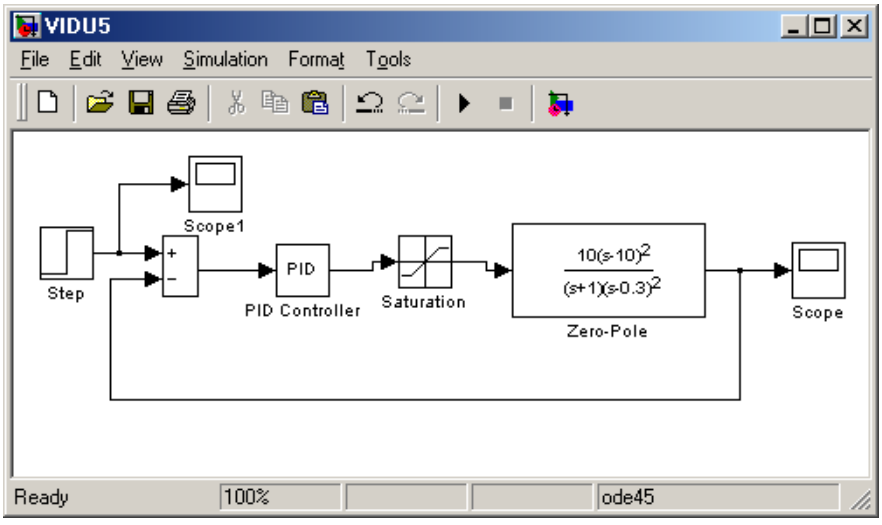

vdp
File Edit View Simulation Format Tools
van der Pol Equation
x1
1 - u*u
Fcn
x2
1
Mu
x1
1/s
x2
2
Out2
1/s
x1
1
Out1
Mux
Scope
The van der Pol Equation
(Double-click on the "?" for more info)
?
Double-click
here for
Simulink Help
To start and stop the simulation, use the "Start/Stop"
selection in the "Simulation" pull-down menu
Ready
100%
ode45


vidu1 *
File Edit View Simulation Format Tools
double
Mux
double
Scope
-1
Gain
double
simout
To Workspace
Ready
100%
ode45
Signal
Generator
1
s
Integrator
Scope
Gain
2
Ready
100%
ode45


VIDU3 *
File Edit View Simulation Format Tools
Step
s2+4s+3
s4+3s2+s+2
Transfer Fcn
Scope
Ready
100%
T=0.000
ode45


VIDU4
File Edit View Simulation Format Tools
Step
In1
Out1
Out2
Out3
Subsystem
Scope3
Scope
Scope2
Ready
100%
ode45





Serial Number:
11-10224-43044-38818-03811-61063-11324
11-30608-63868-33547-31267-29961-41669
11-57331-07688-52528-02324-05637