CHƯƠNG 1

GIỚI THIỆU VỀ MATLAB

MATLAB là sản phẩm phần mềm của công ty MathWorks Inc. Ưu điểm nổi bật của MATLAB là khả năng tính toán và biểu diễn đồ hoạ kỹ thuật nhanh chóng, đa dạng và chính xác cao. Thư viện hàm của MATLAB bao gồm rất nhiều chương trình tính toán con; Các chương trình con này giúp người sử dụng giải quyết nhiều loại bài toán khác nhau, đặc biệt là các bài toán về ma trận, số phức, hệ phương trình tuyến tính cũng như phi tuyến. MATLAB cũng cho phép xử lý dữ liệu và biểu diễn đồ hoạ trong không gian 2D và 3D với nhiều dạng đồ thị thích hợp, giúp người sử dụng có thể trình bày kết quả tính toán một cách trực quan và thuyết phục hơn. Thêm vào đó, các phiên bản MATLAB ngày càng phát triển nhiều module phần mềm bổ sung, gọi là các Toolbox (bộ công cụ) với phạm vi chức năng chuyên dụng cho từng chuyên ngành cụ thể.

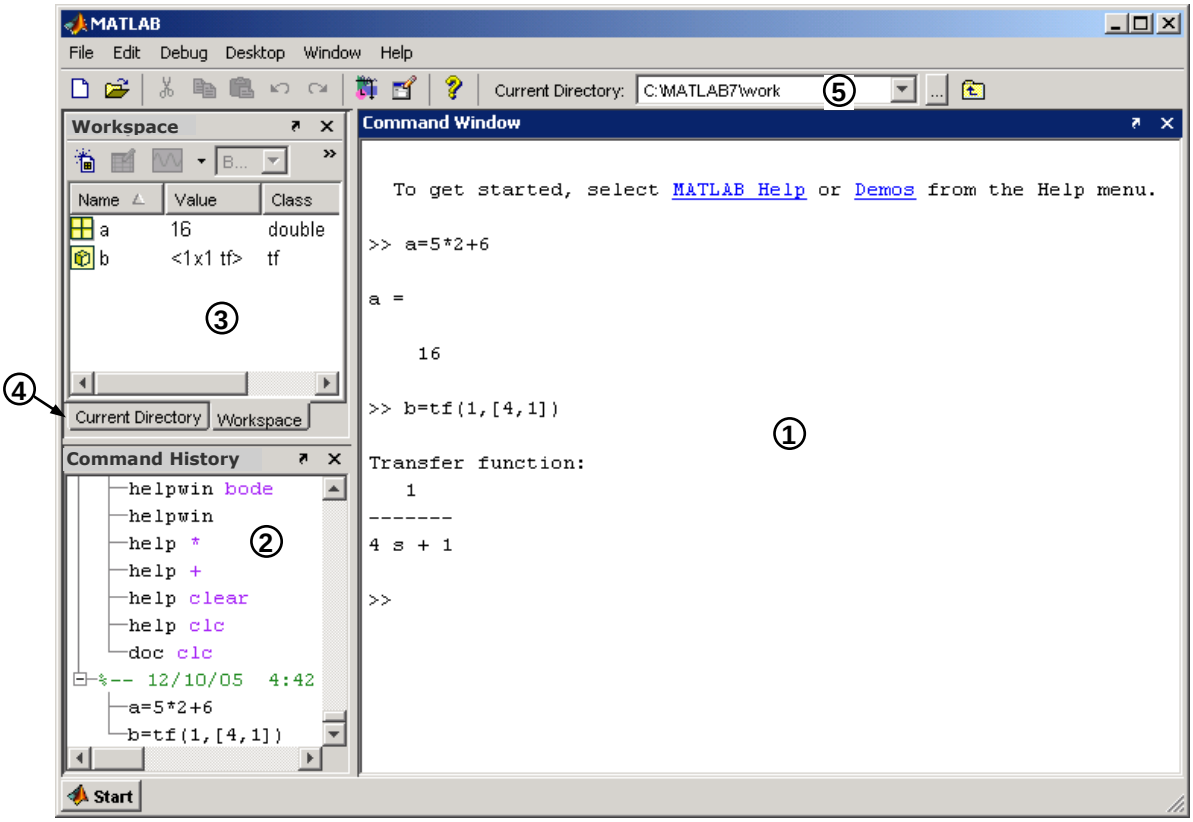
Tài liệu này giới thiệu cách sử dụng MATLAB phần căn bản và ứng dụng các bộ công cụ Control system toolbox và SIMULINK để mô phỏng, phân tích động học các hệ thống điều khiển. Các ví dụ và hình minh hoạ trong tài liệu được thực hiện với phiên bản MATLAB 7 Release 14.

1.1 KHỞI ĐỘNG

- Nhắp đúp chuột vào biểu tượng MATLAB trên màn hình Desktop; hoặc :
- Chọn Start > Programs > MATLAB 7.0 > MATLAB 7.0

1.2 MÀN HÌNH MATLAB

Sau khi khởi động MATLAB ta thu được màn hình MATLAB, bao gồm các cửa sổ tích hợp như hình dưới đây :



Cửa sổ lệnh **Command Window** ① : Đây là cửa sổ chính của MATLAB. Tại đây ta thực hiện toàn bộ việc nhập lệnh và nhận kết quả tính toán. Dấu >> là dấu đợi lệnh. Sau khi nhập lệnh và kết thúc bằng động tác **nhấn phím ENTER**, MATLAB sẽ xử lý lệnh và xuất kết quả liền ngay dưới dòng lệnh. Ví dụ :

```
>> a=5*2+6      (nhập lệnh và nhấn Enter )
a=              (kết quả)
   16
```

Cửa sổ **Command History** ② : Tất cả các lệnh đã sử dụng trong Command Window được lưu giữ và hiển thị tại đây. Có thể lặp lại lệnh cũ bằng cách nhắp đúp chuột vào lệnh đó. Cũng có thể cắt dán, sao chép, xoá cả nhóm lệnh hoặc từng lệnh riêng rẽ.

Cửa sổ **Workspace Browser** ③ : Khái niệm Workspace (không gian làm việc) là một vùng nhớ động trong bộ nhớ của chương trình, tự động hình thành khi MATLAB được khởi động và tự động xóa khi thoát MATLAB. Workspace lưu giữ các biến khi ta sử dụng MATLAB. Tất cả các biến tồn tại trong Workspace đều được hiển thị tại cửa sổ Workspace Browser với các thông tin về tên biến, giá trị, kích cỡ Byte và loại dữ liệu.

Cửa sổ thư mục hiện hành **Current Directory** ④ : Được hiển thị khi nhắp chuột vào ô Current Directory. Nhờ cửa sổ này người sử dụng có thể nhanh chóng nhận biết các thư mục con và các tập tin (file) đang có trong thư mục hiện hành. Các thao tác mở file, lưu file, tìm M-file để thực thi...có mức ưu tiên cao nhất cho thư mục hiện hành. Mặc định khi khởi động MATLAB thì thư mục hiện hành là '...\Thư mục cài đặt MATLAB**work**'.

Tên thư mục hiện hành cũng được chỉ rõ trên thanh toolbar (vị trí⑤).

Trên đây chỉ là một cách hiển thị tổ hợp các cửa sổ trong màn hình MATLAB. Tùy theo thói quen và nhu cầu sử dụng, người dùng có thể thay đổi linh hoạt cách hiển thị thông qua menu ***Desktop*** **>** ***Desktop layout*** **>...** . (Với các phiên bản trước như MATLAB 6 R12 và MATLAB 6.5 R13 chọn menu View > Deskstop Layout >…)

1.3 TIỆN ÍCH TRỢ GIÚP

Tiện ích trợ giúp (Help) của MATLAB rất phong phú. Có thể gọi từ menu ***help*** trên thanh menu hoặc nhập lệnh tại Command window theo cú pháp:

```
help  tênlệnh        % xem trợ giúp tại command window.
doc   tênlệnh        % xem trợ giúp trong cửa sổ Help.
```

Ví dụ, để tìm hiểu chức năng và cách dùng của lệnh **input** ta có thể nhập :

```
>> help input
hoặc :
>> doc  input
```

Ngoài ra, chúng ta cũng có thể xem các ví dụ minh hoạ có sẵn trong MATLAB bằng cách nhập lệnh **demo**.

1.4 THOÁT KHỎI MATLAB

Thực hiện một trong các cách sau đây :

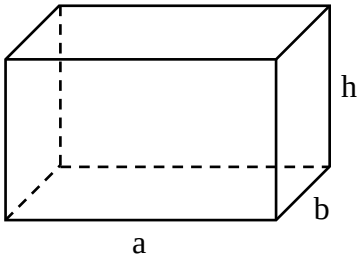
- Nhắp chuột vào nút ☒ ở góc trên, phải của màn hình MATLAB.
- Chọn menu File > Exit MATLAB.
- Nhấn tổ hợp phím Ctrl + Q
- >>quit hoặc >>exit .

1.5 TÍNH TOÁN TẠI COMMAND WINDOW

Với các bài toán đơn giản, chỉ cần dùng ít câu lệnh MATLAB, chúng ta thường giải bằng cách trực tiếp nhập từng lệnh tại cửa sổ Command window.

Ví dụ : Tính thể tích hình hộp :

a = 5 m
b = 2 m
h = 4 m



Tại dấu nhắc lệnh ta nhập:

```
>>  5*2*4
```

MATLAB sẽ tính và hiện kết quả :

```
ans= 40
```

ans – là biến mặc định của MATLAB dùng để chứa dữ liệu hay kết quả tính toán nếu người dùng không đặt tên.

Sử dụng dấu = ta có thể khai báo một biến, đồng thời gán giá trị cho biến đó. Các biến được phân biệt với nhau bởi tên biến.

```
>> a=5; b=2; h=4;
>> S=a*b          % diện tích đáy
S = 10
>> V= S*h         % thể tích
V = 40
```

a, b, h, S, V – là các biến do người dùng đặt tên (user variable).

> **Quy tắc đặt tên biến:**
> + Tên biến phải bắt đầu bằng kí tự chữ. Kế tiếp có thể là chữ, số và dấu _
> + Không được dùng khoảng trống và các dấu ( ),' ,*,-,&,@ ,…
> + Có sự phân biệt chữ hoa và chữ thường.

Ví dụ :

Tên biến hợp lệ : a; b; A; A1; A2; chieu_cao; TT; TT_1; TT_2

Tên biến không hợp lệ : 1B; 2B; G(s); G'; G*, chieu cao; chieu-cao

Các tên biến sau đây là khác nhau: S; s ; the_tich; The_tich; THE_TICH

Lưu ý:

- Không nên đặt tên biến trùng với các biến đặc biệt của MATLAB như: pi (số 3,14159…), i hay j (số ảo đơn vị), inf (số ∞), NaN hay nan (số bất định 0/0).
- Chiều dài tối đa (hay số ký tự tối đa) của tên biến có thể kiểm tra bằng lệnh **:**

```
>> namelengthmax
ans =63
```

Một số lưu ý khi nhập lệnh:

- Bình thường MATLAB luôn hiển thị kết quả của câu lệnh trên màn hình. Nếu muốn MATLAB không hiển thị kết quả thì cuối câu lệnh ta đặt thêm dấu chấm phẩy (;).
- Nhiều câu lệnh có thể đặt chung trên một dòng nhưng bắt buộc phải phân cách nhau bởi dấu phẩy (,) hoặc chấm phẩy (;). Không cho phép phân cách các lệnh bằng khoảng trống. Nếu cuối lệnh nào có dấu phẩy thì MATLAB hiển thị kết quả, còn dấu chấm phẩy thì không hiển thị kết quả.

  Ở ví dụ trên, nếu nhập lệnh :

  >> S=a*b; V=S*h

  thì MATLAB chỉ hiển thị giá trị của V, không hiển thị giá trị của S.
- Các phím mũi tên ↑ ↓ ← → trên bàn phím rất hữu ích khi nhập lệnh. Để gọi lại lệnh vừa gõ, bạn có thể nhấn phím mũi tên ↑, tiếp tục nhấn phím này, nó sẽ gọi tiếp lệnh trước đó. Phím mũi tên ↓ có tác dụng ngược với ↑. Các phím mũi tên ← và → có thể dùng để thay đổi vị trí con trỏ trong dòng lệnh tại dấu nhắc của MATLAB, giúp bạn dễ dàng chỉnh sửa nội dung dòng lệnh.

Xem nội dung của Workspace:

Cách 1: Vào cửa sổ Workspace Browser xem danh sách liệt kê.

Cách 2: Dùng lệnh **who** hoặc **whos**

```
>> who        % liệt kê tên các biến đang có trong Workspace ra màn hình Command
Your variables are:
S  V  a  ans  b  h
>> whos       % liệt kê cả tên biến và các thông tin liên quan
```

Name	Size	Bytes	Class
S	1x1	8	double array
V	1x1	8	double array
a	1x1	8	double array
ans	1x1	8	double array
b	1x1	8	double array
h	1x1	8	double array

Grand total is 6 elements using 48 bytes

Lưu nội dung của Workspace thành tập tin dữ liệu :

- Cách 1: Vào menu File > Save Workspace As > chọn thư mục khác (nếu cần) > nhập tên tập tin > ấn nút Save. Tập tin dữ liệu có tên tổng quát là ***.mat**
- Cách 2: Nhập lệnh >>**save ' đường dẫn\ tênfile.mat '**

  Ví dụ: >>save ' C:\ MATLAB 7\ Work \ mydata1.mat '

Nếu bạn không nhập đường dẫn thì mặc định là lưu vào thư mục hiện hành.

Tải nội dung của một tập tin dữ liệu vào lại Workspace :

- Cách 1: Vào menu File > Import Data > MATLAB Data File (*.mat) > chọn tên tập tin > ấn nút Open.
- Cách 2: Nhập lệnh >>**load ' đường dẫn\ tênfile.mat '**
- Cách 3: Vào cửa sổ Current Directory, nhắp đúp chuột vào tên tập tin cần mở .

Thao tác trên các biến có trong Workspace :

- Xem lại giá trị của biến: Gõ tên biến tại dấu nhắc lệnh.

```
>> tênbiến
>> tênbiến_1, tênbiến_2, ..., tênbiến_n      % giữa các tên biến có dấu phẩy
```

- Chỉnh sửa giá trị đã có của biến : Gõ lệnh gán mới.

  Ví dụ, thay đổi giá trị chiều cao h (đang là 4) thành 6 và tính lại thể tích :

```
>> h=6
h= 6
>>V=S*h
V=60
```

- Xoá sạch nội dung đang có trên màn hình Command window (nhưng không xoá biến) và đưa con trỏ về đầu màn hình :

```
>> clc
```

- Xoá một số biến :

```
>> clear  tênbiến_1  tênbiến_2  ...  tênbiến_n
% chú ý là trường hợp này, giữa các tên biến có khoảng trống.
```

  Ví dụ, để xoá hai biến S và V ta gõ lệnh :

```
>> clear  S  V
```

- Xoá hết mọi biến trong Workspace :

```
>> clear
```

Các thao tác xem nội dung, xoá, lưu, đổi tên, chỉnh sửa giá trị (edit value) của biến cũng có thể thực hiện tại cửa sổ Workspace Browser.

Thao tác trên thư mục:

- Xem đường dẫn và tên thư mục hiện hành: >> **cd**

Khi mới khởi động MATLAB7, thư mục hiện hành mặc định là 'C:\MATLAB7\work'.

- Tạo thư mục mới : >> **mkdir('đường dẫn', 'tên thư mục mới')**

  Ví dụ: >> mkdir('C:\matlab7\work','Nguyen Van A')

  Nếu bạn không nhập đường dẫn thì mặc định là lưu vào thư mục hiện hành.

Lưu ý: tên thư mục cho phép có khoảng trống giữa các từ, nhưng tên biến và tên file thì không được phép.

Bạn cũng có thể nhắp phải chuột trong cửa sổ Current Directory**,** chọn **new** > **folder** > nhập (gõ) tên thư mục muốn tạo mới > nhấn Enter.

- Chuyển thư mục mới tạo trở thành thư mục hiện hành :

```
>>cd   'C:\matlab7\work','Nguyen Van A'
hoặc   >>cd   'Nguyen Van A'
```

Bạn cũng có thể thực hiện bằng cách vào cửa sổ Current Directory, nhắp đúp chuột vào tên thư mục cần chuyển (ví dụ thư mục 'Nguyen Van A') .

- Chuyển lên thư mục cấp trên : >> **cd . .** % giữa cd và .. có khoảng trống

CHƯƠNG 2

M-FILE

Trong MATLAB, M-file là các file chương trình được soạn thảo và lưu ở dạng văn bản. Có hai loại M-file là Script file (file lệnh) và Function file (file hàm). Cả hai đều có phần tên mở rộng là "**.m** ". MATLAB có rất nhiều M-file chuẩn được xây dựng sẵn. Người dùng cũng có thể tạo các M-file mới tuỳ theo nhu cầu sử dụng.

2.1 LẬP TRÌNH DẠNG SCRIPT FILE

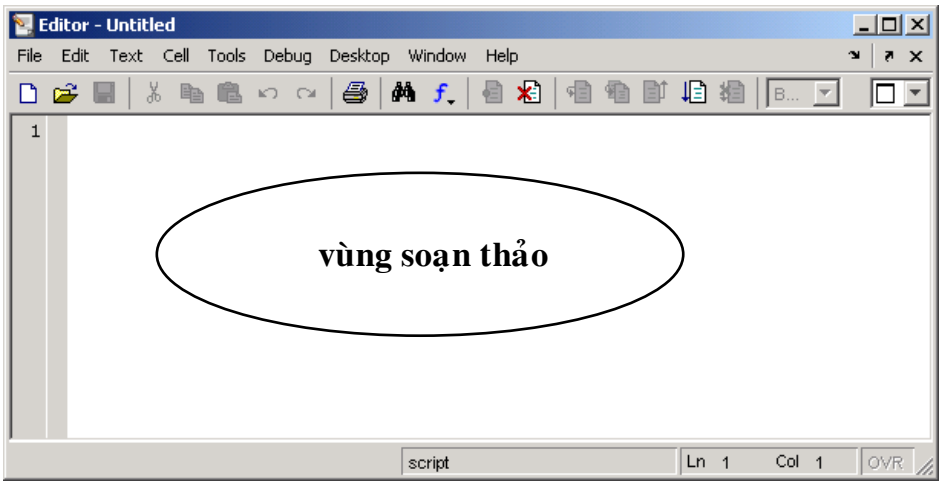
Thay vì nhập và thực thi từng câu lệnh tại cửa sổ Command window, bạn có thể soạn và lưu tất cả các câu lệnh cần thiết để giải bài toán vào một Script file. Sau đó bạn chỉ cần gõ tên file để thực thi toàn bộ chương trình.

1) Mở cửa sổ Editor :

Cách 1: Trong command window gõ lệnh **edit**

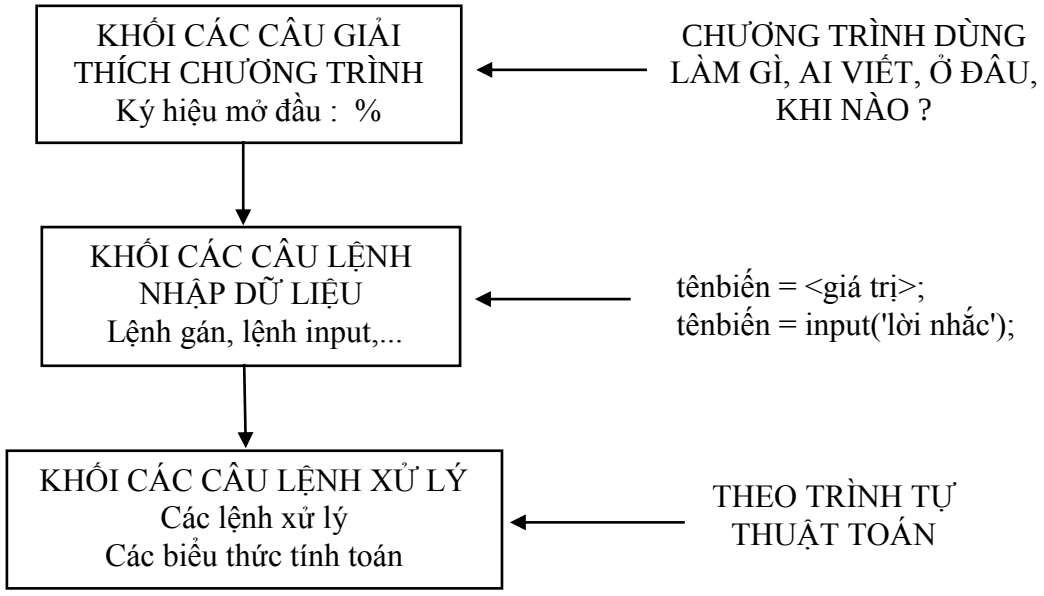
Cách 2: Vào menu File >New >M-File

Cách 3: Nhắp chuột vào icon (icon đầu tiên trên thanh toolbar)



2) Soạn thảo

Cấu trúc tổng quát của một Scrift file :



- Ký hiệu % có thể dùng ở bất cứ chỗ nào trong chương trình để tạo câu ghi chú, giải thích. Các câu ghi chú đặt phía trên dòng lệnh đầu tiên sẽ hiện trên màn hình khi bạn gõ lệnh **help tênfile.**
- Lệnh gán : dùng để gán giá trị cho biến.

  Cú pháp: **tênbiến =<giá trị>**
- Lệnh input : dùng để nhận một giá trị từ bàn phím.

  Cú pháp: **tênbiến = input('lời nhắc')**

  Ví dụ: a=input('Nhap chieu dai a:')

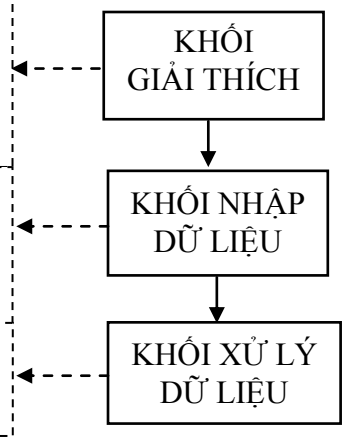
  Khi thực thi, MATLAB sẽ hiện dòng nhắc :

  Nhap chieu dai a:

  sau đó chờ người dùng nhập một giá trị số từ bàn phím, nhận giá trị vừa nhập và gán vào biến a.

Ví dụ 1: Soạn thảo tập tin **vd1.m** với nội dung như sau :

```
% Chuong trinh tinh the tich hinh hop
% DHSPKT, 10/2004
% Nguoi viet: Nguyen Van A
% ---------------------------------------------
a = input('Nhap chieu dai a:');
b = input('Nhap chieu rong b:');
h = input('Nhap chieu cao h:');
S=a*b;          % Dien tich day
V=S*h           % The tich
```



3) Lưu: Vào menu File > Save > đặt tên tập tin > nhắp nút save.

Tập tin Scrift file có phần mở rộng là "**.m**", và được lưu vào thư mục hiện hành. Nếu không có sự lựa chọn khác thì thư mục hiện hành được mặc định là thư mục work của MATLAB. ***Tên tập tin phải bắt dầu bằng ký tự chữ, không có khoảng trống giữa các ký tự*** (giống như quy định về tên biến). Lưu ý kiểm tra và tắt các phần mềm gõ tiếng Việt như Vietkey, Unikey,…nếu cần.

4) Gọi thực hiện SCRIPT FILE:

Cách 1: Trong cửa sổ soạn thảo nhắp chuột vào nút **run** trên thanh toolbar.

Cách 2: Trở về màn hình Command window và gõ tên file (không có phần mở rộng **.m**), sau đó ***nhấn Enter để thực thi***.

Ví dụ để thực thi file "**vd1.m** " vừa tạo, tại dấu nhắc lệnh ta gõ : >> vd1

Lưu ý là dù gọi thực hiện theo cách 1 hay cách 2 thì MATLAB cũng đều xuất kết quả tính toán tại cửa sổ Command Window.

5) Mở một M-file đang có để xem lại hay chỉnh sửa:

Cách 1: Trong cửa sổ Editor hoặc Command window, vào menu File >open >…

Cách 2: Vào cửa sổ Workspace, nhắp đúp chuột vào tên M-file cần mở.

Cách 3: Tại Command window, gõ lệnh **edit ('đường dẫn\tên file')**

2.2 LẬP TRÌNH DẠNG FUNCTION FILE

Tương tự như trong toán học, các hàm (function) trong MATLAB sẽ nhận vào giá trị của các đối số và trả về giá trị tương ứng của hàm. Trình tự tạo và thực thi một file hàm bao gồm các bước như sau:

1) Mở cửa sổ Editor : thực hiện tương tự như Scrift file

2) Soạn thảo:

Cấu trúc chuẩn của một hàm:

```
function [danh sách tham số ra] = tên hàm (danh sách tham số vào)
% khối các câu ghi chú, giải thích
câu lệnh xử lý 1;
câu lệnh xử lý 2;
. . . . . .
tham số ra 1 = biểu thức tính 1;
tham số ra 2 = biểu thức tính 2;
. . . . . .
```

- **Tham số ra** (cũng gọi là tham biến) dùng để chứa các kết quả xử lý của hàm. Khi gọi thực hiện hàm ta có thể thay chúng bằng các tên biến cụ thể.
- **Tham số vào** (cũng gọi là tham trị, hay đối số) là các biến trong hàm mà sẽ nhận các giá trị vào cụ thể khi gọi thực hiện hàm.

Tên các tham số vào, ra trong phần danh sách được phân cách nhau bằng dấu phẩy. Nếu chỉ có 1 tham số ra thì không cần dùng dấu ngoặc vuông [ ].

3) Lưu: như cách lưu của Scrift file. Khi lưu hàm, MATLAB sẽ lấy tên hàm làm tên file, người lập trình ***không nên sửa lại tên này*** để tránh lẫn lộn khi gọi thực hiện hàm.

4) Gọi thực hiện Function file: từ cửa sổ Command thực hiện như sau:

Nếu chỉ có một tham số ra:

>> **tênbiến** = **tênfile** (danh sách các giá trị vào)

Nếu có nhiều tham số ra:

>> **[tênbiến1, tênbiến2,...]** = **tênfile** (danh sách các giá trị vào)

Ví dụ 2: Tạo tập tin **tt_hinhhop.m**

```
function V = tt_hinhhop(a,b,h)
% chuong trinh tinh the tich hinh hop
% khi biet cac canh a(dai),b(rong),h(cao)
S=a*b;
V=S*h;
```

Thực thi trong Command window :

```
>> TT=tt_hinhhop(5,2,4)          % tham biến V có thể đổi tên tuỳ ý, ví dụ đổi là TT.
TT = 40
```

Chú ý:

- Khi bạn gõ lệnh **help tênfilehàm** thì các câu ghi chú ở phía trên dòng khai báo function hoặc dưới dòng function nhưng trước dòng lệnh đầu tiên trong file hàm sẽ được hiện trên màn hình .
- Khi gọi thực thi một file hàm ta ***dùng tên file, không phải tên hàm***. Do đó nếu ở ví dụ 2 ta đặt tên file là "vd2.m" thì khi thực thi phải dùng lệnh:

  >> TT=vd2(5,2,4)

Ví dụ 3: Tạo file hàm **dttt_hinhhop.m** với 2 tham số ra.

```
% CTr tinh dien tich day va the tich hinh hop
% khi biet cac canh a(dai),b(rong),h(cao)
function [S,V] = dttt_hinhhop(a,b,h)
S=a*b;      % dien tich day
V=S*h;      % the tich
```

Thực thi trong Command window :

```
>> [DT,TT] = dttt_hinhhop(5,2,4)          % Các tham biến S,V có thể đổi tên tuỳ ý.
DT =
     10
TT =
     40
```

Ví dụ 4 : Tạo file hàm **change.m**

```
% chương trình đổi độ sang radian
function  rad = change(do)
rad=do*pi/180;
```

Thực thi trong Command window :

```
>> rad = change (30)
rad =
   0.5236
```

Đặc điểm của hàm :

Các hàm chỉ thông tin với MATLAB thông qua các biến truyền vào cho nó và các biến ra mà nó tạo thành, các biến trung gian ở bên trong hàm thì không tương tác với môi trường MATLAB.

Khi MATLAB thực hiện lần đầu các file hàm, nó sẽ mở file và dịch các dòng lệnh của file đó ra một dạng mã lưu trong bộ nhớ nhằm mục đích tăng tốc độ thực hiện các lời gọi hàm tiếp theo. Nếu sau đó không có sự thay đổi gì trong M file, quá trình dịch sẽ không xảy ra lần thứ hai. Nếu trong hàm có chứa lời gọi hàm M-file khác thì các hàm đó cũng được dịch vào trong bộ nhớ. Bằng lệnh **clear function** ta có thể xoá cưỡng bức các hàm đã dịch, nhưng vẫn giữ nguyên các M-file.

Mỗi hàm có không gian làm việc riêng của nó (local workspace), tách biệt với môi trường MATLAB (sử dụng base workspace), mối quan hệ duy nhất giữa các biến trong hàm với môi trường bên ngoài là các biến vào và ra của hàm đó. Nếu bản thân các biến của hàm bị thay đổi thì sự thay đổi này chỉ tác động bên trong của hàm đó và mà không làm ảnh hưởng đến các biến của môi trường MATLAB. Các biến của hàm sẽ được giải phóng ngay sau khi hàm thực thi xong nhiệm vụ, vì vậy không thể sử dụng thông tin của lần gọi trước cho lần gọi sau.

Các hàm có thể sử dụng chung các biến với hàm khác hay với môi trường MATLAB nếu các biến được khai báo là biến toàn cục. Để có thể truy cập được các biến bên trong một hàm thì các biến đó phải được khai báo là biến toàn cục trong mỗi hàm sử dụng nó.

Một M-file có thể chứa nhiều hàm. Hàm chính (main function) trong M-file này phải được đặt tên trùng với tên của M-file. Các hàm khác được khai báo thông qua câu lệnh **function** được viết sau hàm đầu tiên. Các hàm con (local function) chỉ được sử dụng bởi hàm chính, tức là ngoài hàm chính ra thì không có hàm nào khác có thể gọi được chúng. Tính năng này cung cấp một giải pháp hữu hiệu để giải quyết từng phần của hàm chính một cách riêng rẽ, tạo thuận lợi cho việc lập một file hàm duy nhất để giải bài toán phức tạp.

Ví dụ 5 : Tạo file hàm **tinh_gia_tien.m** có nội dung sau

```
function gia = tinh_gia_tien(L,d)
% CTr tinh gia tien khoi thep hinh tru
% khi biet chieu dai L (mm),duong kinh d (mm)
%=====================================================
gama=7800;                    % khoi luong rieng (kg/m^3)
gia_don_vi= 10000;            % (dong/kg)
TT= tt_hinhtru(L,d)           % (mm^3)
TT=(1e-9)* TT                 % (m^3)
gia= gia_don_vi*gama*TT;      % dong
%=====================================================
% Local function (ham con)
function V = tt_hinhtru(L,d)
V=L*pi*d^2/4;     % the tich (mm^3)
```

2.3. Biến cục bộ và biến toàn cục

a) Biến cục bộ

Biến cục bộ chỉ có phạm vi sử dụng trong một hàm. Các biến cục bộ không lưu giữ trong Workspace. Tại Command window ta không thể truy cập được các biến cục bộ. Các biến trong các file hàm đều là biến cục bộ, trừ phi có sự chủ động khai báo khác đi.

Ví dụ: các biến a, b, h, S,V trong file hàm **tt_hinhhop.m** là các biến cục bộ.

b) Biến toàn cục

Biến toàn cục có phạm vi sử dụng trong nhiều hàm hoặc nhiều M-file. Các biến toàn cục được lưu giữ trong Workspace của MATLAB và hiển thị tại cửa sổ Workspace browser. Tại Command window ta chỉ có thể truy cập được các biến toàn cục.

Ví dụ: Các biến trong các Script-file là các biến toàn cục.

Các biến tạo trực tiếp tại Command window là các biến toàn cục.

Để các biến trong Script-file trở thành biến cục bộ, ta có thể ***chuyển một Script-file thành một Function-file*** đơn giản không có các tham số vào, ra. Ví dụ, từ Scrift-file **vd1.m** ta tạo file hàm **vd1B.m** có nội dung như sau:

```
function vd1B()
a = input('Nhap chieu dai a:');
b = input('Nhap chieu rong b:');
h = input('Nhap chieu cao h:');
S=a*b      % dien tich day
V=S*h      % the tich
```

Khi gọi thực hiện bằng lệnh >>vd1B hoặc >>vd1B( ), ta vẫn có các kết quả tương tự >>vd1, chỉ khác là các biến a, b, h, S, V bây giờ là biến cục bộ nên không còn truy cập được từ cửa sổ Command window. Sau khi hàm thực thi xong, nếu gõ lệnh:

>> a

Bạn sẽ nhận được dòng thông báo sau :

??? Undefined function or variable 'a '.

% hàm hoặc biến 'a' chưa được định nghĩa.

CHƯƠNG 3

CÁC KIỂU DỮ LIỆU VÀ PHÉP TÍNH

MATLAB có khả năng tính toán trên mọi kiểu dữ liệu số và chữ. Dữ liệu số có thể là số thực, số phức, vectơ, ma trận. Dữ liệu chữ có thể là chuỗi ký tự, biểu thức logic, biểu thức chữ, ...

3.1 SỐ THỰC

Khi nhập số thập phân, ta dùng dấu chấm để tách phần nguyên và phần lẻ. Lũy thừa của 10 biểu diễn bằng ký hiệu **e.**

Ví dụ 1. Cho các số thực: a =5 ; b =2,54 ; c =10^6 ; d =-4x10^{-3} =-0,004

Nhập vào MATLAB:

```
>> a = 5
a =
     5
>> b = 2.54
b =
    2.5400
>>c = 10^6              % hoặc >> c=1e6
c=
   1.0000e+006
>>d = -4e-3
d=
   -0.0040
```

1) Các phép tính số học

Trong MATLAB, các phép tính số học có mức ưu tiên giống như trong tính toán thông thường. Nếu trong câu lệnh có các phép tính cùng mức ưu tiên thì thứ tự thực hiện là từ trái qua phải. Khi cần thay đổi mức độ ưu tiên ta dùng thêm dấu ngoặc đơn ( ).

PHÉP TÍNH	KÍ HIỆU	MỨC ƯU TIÊN	VÍ DỤ
Lũy thừa	^	1	3^2 ; a^(1/2)
Nhân	*	2	3*5 ; a*b
Chia	/	2	2/4 ; a/b
Chia trái	\	2	2\4 (nghĩa là 4/2) ; a\b
Cộng	+	3	2+4 ; a+b
Trừ	-	3	2-4 ; a-b

Ví dụ 2. Giải phương trình bậc hai $ax^2 + bx + c = 0$; với $a = 1; b = -5; c = 2$.

Ta biết các nghiệm của phương trình bậc hai có dạng :

$$x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2a} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Thực hiện trong MATLAB như sau:

```
>> a = 1; b = - 5; c = 2;
>> delta = b^2-4*a*c;
>> x1 = (- b + delta^(1/2))/(2*a)
x1 =
    4.5616
>> x2 = (- b - delta^(1/2))/(2*a)
x2 =
    0.4384
```

2) Các biến, hằng, ký tự đặc biệt trong MATLAB

+ pi : số 3.14159265...

+ i, j: số ảo đơn vị, $i^2 = j^2 = -1$

+ realmin : số chấm động dương nhỏ nhất, bằng 2^{-1022} hay 2.2251e-308

+ realmax: số chấm động dương lớn nhất, $\approx 2^{1023}$ hay 1.7977e+308

+ Inf hay inf : số vô cùng (Infinity), với Inf = $+\infty$; -Inf = $-\infty$

+ NaN hay nan : (Not-a-number) không phải là số, ví dụ kết quả phép chia 0/0

+ eps : độ chính xác tương đối của số có chấm động, bằng 2^{-52} hay 2.2204e-016

3) Định dạng số

Khi hiển thị kết quả tính toán ra màn hình, MATLAB dùng định dạng số mặc định là **format short**. Tùy theo yêu cầu ta có thể định dạng lại các con số khi hiển thị. Việc định dạng này không ảnh hưởng đến độ chính xác tính toán vì MATLAB vẫn lưu giữ các biến trong bộ nhớ với giá trị thực sự của nó.

Lệnh sử dụng	**Dạng hiển thị**	**Ví dụ: hiển thị số π:** >> pi
format bank	2 chữ số thập phân	3.14
format	4 chữ số thập phân	3.1416
format short	4 chữ số thập phân	3.1416
format short e	4 chữ số thập phân với dấu chấm động	3.1416e+00 (= 3.1416*10^0)
format long	14 chữ số thập phân	3.14159265358979
format rat	dạng tỉ số	355/113

Để đọc giải thích chi tiết về lệnh format, bạn gõ lệnh: **help format**

4) **Một số hàm toán cơ bản** :

Khi sử dụng các hàm này, đối số x là số thực, phức, vectơ, hay ma trận đều được.

TÊN HÀM	**CHỨC NĂNG**
sqrt(x)	Căn bậc hai của x, tương đương lệnh x^(1/2)
exp(x)	Hàm mũ cơ số e của x (= e^x) *Ví dụ:* exp(0)= e^0 =1; exp(1)= e^1 =2.7182
log(x)	Logarit cơ số e của x (= lnx)
log10(x)	Logarit thập phân của x
abs(x)	- Tìm giá trị tuyệt đối của x nếu x là số thực - Tìm môđun của x nếu x là số phức
round(x)	Làm tròn x tới số nguyên gần nhất
rem(x,y)	Tìm phần dư của x /y , có dấu lấy theo x
mod(x,y)	Tìm phần dư của x /y, có dấu lấy theo y
sign(x)	Hàm lấy dấu của x (hàm signum); trả về 1 nếu x>0; trả về -1 nếu x<0; trả về 0 nếu x=0 *Ví dụ:* sign(5)=1; sign (-5)=-1; sign(0)=0
sin(x)	sin của x, với x là radian
cos(x)	cos của x, với x là radian
tan(x)	tang của x, với x là radian
asin(x)	arcsin của x, với x là radian
acos(x)	arccos của x, với x là radian
atan(x)	arctg của x, với x là radian
sinc(x)	= (sin(πx)) /πx nếu x ≠0 = 1 nếu x=0
sind(x); cosd(x); tand(x)	sin; cos; tang của x, với x là độ
asind(x); acosd(x); atand(x)	arcsin; arccos; arctg của x, với x là độ
sinh(x)	sinhyperbol của x ; sinh(x) = $(e^x - e^{-x})/2$
cosh(x)	coshyperpol của x ; cosh(x) = $(e^x + e^{-x})/2$ $\Rightarrow$ cosh(x) + sinh(x) = e^x cosh(x) - sinh(x) = e^{-x}
tanh(x)	tanhyperpol(x) ; tanh(x) = sinh(x) / cosh(x)
asinh(x)	arcsinhyperpol(x); $= \ln(x + \sqrt{x^2 + 1})$
acosh(x)	arcoshyperpol(x); $= \ln(x + \sqrt{x^2 - 1})$
atanh(x)	arctanhyperpol(x); = (1/2)ln [(1+x)/(1-x)]

MATLAB có rất nhiều hàm toán học được xây dựng sẵn. Để tìm hiểu kỹ hơn, bạn có thể gõ lệnh **help elfun**, **help elmat**, **help specfun** hoặc **help datafun** .

3.2 SỐ PHỨC

3.2.1 Cơ sở lý thuyết

Trong toán học, một số phức z thường được biểu diễn theo 1 trong 3 dạng sau:

1) Dạng đại số : z = a + bi hoặc z = a + ib

a – phần thực; b – phần ảo; i – toán tử ảo ($i = \sqrt{-1}$)

2) Dạng môđun-pha : $z = |z|(\cos\theta + i\sin\theta)$

$|z| = \sqrt{a^2 + b^2}$: môđun

θ = arctg (b/a) : góc pha

3) Dạng cực : $z = |z| e^{i\theta}$

3.3.2 Thể hiện trong MATLAB

- Trong Matlab các ký tự i và j dùng để ký hiệu toán tử ảo

Nếu i hoặc j đã được sử dụng cho các giá trị khác thì ta phải định nghĩa lại như sau:

i=sqrt(-1) hoặc j = sqrt(-1)

Ví dụ:

```
>> z = 3 - 5i              %  hoặc  >> z = 3 - i*5
z =
   3.0000 - 5.0000i
>> z=5*exp(4i)          %  tức là  z=5e^4i
z=
    -3.2682 - 3.7840i
```

- Các phép tính số học trên số phức được dùng tương tự như trong số thực.
- Một số hàm tính thành phần của số phức :

TÊN HÀM	CHỨC NĂNG
z = complex(a,b)	Tạo số phức z =a+ib
real (z)	Tính phần thực của z
imag(z)	Tính phần ảo của z
abs(z)	Tính môđun |z|
angle(z)	Tính góc θ, với $-\pi \le \theta \le \pi$
atan2(imag(z),real(z))	Tính góc θ, với $-\pi \le \theta \le \pi$
conj(z)	Tạo số phức liên hợp của z

Ví dụ:

```
>> z1=2+3i;      % hoặc    z1=complex(2,3)
>>theta=angle(z1)
theta=  0.9828
>>modul=abs(z1)
modul= 3.6056
>> z2=modul*exp(theta*i)      % thử lại, nếu tính đúng thì sẽ có z2=z1
z2= 2.0000 + 3.0000i
```

3.3 CHUỖI

Trong MATLAB, chuỗi là dãy ký tự đặt trong cặp dấu nháy đơn ' '.
Mỗi ký tự của chuỗi chiếm 2 byte trong bộ nhớ.

```
>>s = '46'                  %  s là biến chuỗi, chứa 2 ký tự 4 và 6, chiếm 4 byte
>>str = 'the tich hinh hop' % str là biến chuỗi, chứa 17 phần tử, chiếm 34 byte
>>a=46                      % a là biến số thực, chiếm 8 byte bộ nhớ
>>c=46+32i                  % c là biến số phức, chiếm 16 byte bộ nhớ
```

Các hàm xử lý chuỗi thông dụng :

TÊN HÀM	CHỨC NĂNG
upper	Đổi ra ký tự hoa
lower	Đổi ra ký tự thường
str2num	Đổi chuỗi ra số
num2str	Đổi số ra chuỗi
ischar(s); isstr(s)	Hàm trả về 1 (True) nếu s là chuỗi
strcat (s1, s2,...)	Nối các chuỗi thành hàng, tự động ngắt bớt khoảng trống cuối chuỗi nếu có.
strvcat (s1,s2,...)	Ghép các chuỗi thành cột (ma trận ký tự), tự động thêm khoảng trống, bỏ qua chuỗi rỗng
str2mat (s1,s2,...)	Ghép các chuỗi thành cột (ma trận ký tự), tự động thêm khoảng trống, chuỗi rỗng cũng tính là 1 cột.
disp(s)	Hiển thị nội dung của biến s ra màn hình
fprintf	Đưa dữ liệu có định dạng ra file hoặc màn hình
strcmp(s1,s2)	So sánh hai chuỗi, true nếu s1 giống s2
strncmp(s1,s2,N)	True nếu N ký tự đầu của s1, s2 giống nhau
eval('chuỗi')	Xử lý chuỗi như một lệnh MATLAB

Ứng dụng:

Ghép mảng chuỗi với số và hiển thị kết quả :

```
>> a= 12;
>> str=['Gia tri cua a la:  ',  num2str(a)] ;
>> disp(str)   %  hiện nội dung, không cho hiện tên biến
```

Kết quả các câu lệnh trên sẽ là:

```
Gia tri cua a la:  12
```

Dùng lệnh fprintf

```
>> R=45;
>> fprintf ( 'Dien tich = %7.3f  m^2 \n', pi*R^2)
```

Giải thích : - Có %7.3f thì hiển thị ít nhất 7 ký tự với 3 chữ số thập phân.
- Có \n thì in xong xuống hàng, đưa dấu nhắc lệnh về đầu dòng kế tiếp.

Kết quả hiển thị :

```
Dien tich =  6361.725 m^2
```

```
>> B = [8.8  7.7; 8800  7700];
>> fprintf(1,'X is %6.2f meters or %8.3f mm\n',9.9,9900,B)
```

Kết quả hiển thị:

```
X is 9.90 meters or 9900.000 mm
X is 8.80 meters or 8800.000 mm
X is 7.70 meters or 7700.000 mm
```

```
>> fprintf ('\n')      % tạo một dòng trống, tương đương lệnh disp(' ').
```

Ghép các chuỗi và cho hiển thị :

```
>>s=str2mat('Chao cac ban', 'Chung ta bat dau nhe !');   %  tạo hai hàng chuỗi
>>disp(s)
```

Kết quả hiển thị :

```
Chao cac ban
Chung ta bat dau nhe !
```

Lưu ý :

- Kết quả lệnh **strvcat ('Hello','Yes')** tương tự như **['Hello';'Yes ']**
  % strvcat tự động thêm hai khoảng trống sau chuỗi Yes để ghép hợp lệ, tức là hai hàng phải có số phần tử (số ký tự) bằng nhau là 5.
- Kết quả lệnh **strvcat('Hello','','Yes')** khác với **str2mat ('Hello', '','Yes')**
  % ở ví dụ này strvcat tạo 2 hàng vì bỏ qua chuỗi rỗng'', còn str2mat tạo 3 hàng.
- Kết quả lệnh **strcat('Chao ','Ban')** khác với **['Chao ', 'Ban']**
  % strcat tạo kết quả ChaoBan, các khoảng trống sau từ Chao bị bỏ qua.

Ví dụ : Soạn thảo tập tin: **tthinhtru.m**

```
% Chuong trinh tinh the tich hinh tru
% khi biet ban kinh R ,chieu cao h
function V = tthinhtru(R,h)
S=pi*R^2;
V=S*h;
disp (['Ban kinh  R = ',num2str(R)])
disp (['Chieu cao h = ',num2str(h)])
disp ('The tich :') %fprintf('The tich :\n')
```

Thực thi tại Command window :

```
>> TT= tthinhtru(5,2) ↵
Ban kinh  R  = 5
Chieu cao  h  = 2
The tich :
 TT =  157.0796
```

3.4 VÉCTƠ

Trong MATLAB, các thuật ngữ **véctơ** và **mảng** được dùng không phân biệt.

- Để khai báo một **véctơ cột** (mảng cột) ta nhập các phần tử nằm trong dấu ngoặc vuông [ ], phân cách nhau bởi dấu chấm phẩy.

Ví dụ :

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

- Để khai báo một **véctơ hàng** (mảng hàng) ta nhập các phần tử nằm trong dấu ngoặc vuông [ ], phân cách nhau bởi khoảng trắng hoặc dấu phẩy.

Ví dụ :

```
>> b = [2  3  4  7]        % hoặc  >>b= [2,3,4,7]
b=
    2    3    4    7
```

- Để tạo vectơ hàng có giá trị các phần tử cách đều nhau, MATLAB cho phép khai báo bằng toán tử **(:)** như sau :

x=x1: Δx: x2 hoặc **x=[x1: Δx: x2]**

Trong đó **x1** là giá trị đầu, **Δx** là gia số**, x2** là giá trị cuối của vectơ **x**

Nếu **Δx =1** thì có thể khai báo đơn giản:

x=x1: x2 hoặc **x=[x1: x2]**

Ví dụ:

```
>> x =[0:10]          % Vectơ x biểu diễn 11 số tự nhiên từ 0 đến 10.
x =
    0    1    2    3    4    5    6    7    8    9   10
>> k=15 ; v=[k:k:1000*k]     % Vectơ v biểu diễn 1000 số là bội số của k
```

Một số hàm về véctơ (mảng) :

HÀM	CHỨC NĂNG
length(a)	Tìm số phần tử của véctơ a
size(a)	Tìm kích thước véctơ, có dạng (1x n) hoặc (n x 1)
a(i)	Tìm phần tử thứ i của véctơ a (i=1,2,3,…)
a(i : j)	Tìm các phần tử từ thứ i tới thứ j của véctơ a
norm(a)	Tính chuẩn Euclid của véctơ a (= ‖a‖ = căn bậc hai của tổng bình phương các phần tử của a)
sum(a)	Tổng các phần tử
prod(a)	Tích các phần tử

min(a)	Phần tử bé nhất của vectơ a
max(a)	Phần tử lớn nhất của vectơ a
mean(a)	Trung bình cộng của các phần tử

Ví dụ: a) Cho n=50. Tìm tổng n số chẵn đầu tiên
b) Cho n=50. Tìm tổng n số lẻ đầu tiên
b) Cho n=5. Tìm n!

```
>> n=50; a=[2:2: 2*n];
>> tong=sum(a)
  tong=
        2550
>> n=50; b=[1:2: 2*n-1]; tong=sum(b)
  tong=
        2500
>>n=5; c=[1: n];  giai_thua=prod(c)
  giai_thua=
              120
```

Các phép tính giữa véc tơ (mảng) với một số vô hướng:

PHÉP TÍNH	KÍ HIỆU	VÍ DỤ	Ý NGHĨA
Lũy thừa	.^	c=a.^2	c= [(a_1^2);(a_2^2); ...; (a_n^2)]
Nhân	* hoặc .*	c=a*2	c= [(a_1*2);(a_2*2); ...; (a_n*2)]
Chia	/ hoặc .\	c=a/2	c= [(a_1/2);(a_2/2); ...; (a_n/2)]
Chia trái	.\	c=a.\2	c= [(2/a_1);(2/a_2); ...; (2/a_n)]
Cộng	+	c=a+2	c= [(a_1+2);(a_2+2); ...; (a_n+2)]
Trừ	-	c=a-2	c= [(a_1-2);(a_2-2); ...; (a_n-2)]

Các phép tính giữa hai véc tơ :

PHÉP TÍNH	KÍ HIỆU	VÍ DỤ	Ý NGHĨA
Lũy thừa	.^	c=a.^b	c= [(a_1^b_1);(a_2^b_2); ...; (a_n^b_n)]
Tích có hướng	.*	c=a.*b	c= [(a_1*b_1);(a_2*b_2); ...; (a_n*b_n)]
Chia phải	./	c=a./b	c= [(a_1/b_1);(a_2/b_2); ...; (a_n/b_n)]
Chia trái	.\	c=a.\b	c= [(a_1\$b_1$);($a_2$\$b_2$); ...; ($a_n$\$b_n$)]
Cộng hai vectơ	+	c=a+b	c= [(a_1+b_1);(a_2+b_2); ...; (a_n+b_n)]
Trừ hai véctơ	-	c=a-b	c= [(a_1-b_1);(a_2-b_2); ...; (a_n-b_n)]
Chuyển vị (cột thành hàng hay ngược lại)	'	a'	a=[a_1; a_2 ; ...;a_n] thì a'=[a_1 a_2 ... a_n] a=[a_1 a_2 ... a_n] thì a'=[a_1; a_2 ;...;a_n]
Tích vô hướng	'*	c=a'*b	c= (a_1*b_1)+(a_2*b_2)+ ... + (a_n*b_n)

Ví dụ : >> a=[3;2;5] ; % a là vectơ cột

>> b=[-1;3;-6] ; % b là vectơ cột

>> c=a.*b % Tích có hướng ⇒ Kết quả là một véc tơ.

```
c =
    -3
     6
   -30
```

>> d=a'*b % Tích vô hướng ⇒ Kết quả là một số vô hướng.

d = -27

- **Góc giữa 2 vectơ cột a và b**

- Biểu diễn trong toán học : $\theta = \arccos \frac{a^T.b}{\|a\|.\|b\|}$

trong đó : a^T là vectơ chuyển vị của a,
||a|| và ||b|| là chuẩn Euclid của a và b.

- Biểu diễn trong MATLAB:

```
>> theta = acos(a'*b/(norm(a).*norm(b)))    % tính góc giữa hai véctơ cột a và b
   theta =
            2.2729      % radian
```

- **Các lệnh tạo véc tơ hàng đặc biệt**

- Có thể tạo một véctơ hàng tuyến tính bằng cách dùng lệnh:

linspace(giá trị đầu, giá trị cuối, số phần tử)

Nếu ta không nhập số phần tử thì mặc định là 100 phần tử.

Ví dụ:

```
>> x = linspace (0, 20,11)
x =
     0    2    4    6    8   10   12   14   16   18   20
```

- Có thể tạo một véctơ hàng có thang chia logarit bằng cách dùng lệnh:

logspace(giá trị đầu, giá trị cuối, số phần tử)

Đối với hàm logspace, giá trị đầu và giá trị cuối được nhập bởi số mũ thập phân, ví dụ: thay vì nhập 100 (10^2) ta chỉ cần nhập 2. Nếu ta không nhập số phần tử thì mặc định là 50 phần tử.

Ví dụ:

```
>> w = logspace(1,2,5)
w =
   10.0000   17.7828   31.6228   56.2341  100.0000
```

3.5 MA TRẬN

Ma trận có dạng tổng quát :

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}_{m \times n}$$

Trong MATLAB, một ma trận được biểu diễn bằng một dãy số trong ngoặc vuông. Các phần tử trong mỗi hàng được phân cách nhau bởi khoảng trống hoặc dấu phẩy (,) . Các hàng phân cách nhau bởi dấu chấm phẩy (;).

Ví dụ 1:

```
>> A = [1 2 3 ; 4 5 6 ; 7 8 9] ↵        % A là ma trận vuông cấp 3
```

Cũng có thể nhập vào MATLAB theo từng hàng :

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

Cả hai cách nhập trên đều được MATLAB trả về kết quả là :

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

3.5.1 Các phép tính trên ma trận

- Các phép tính cộng, trừ, nhân, luỹ thừa, chia phải, chia trái giữa ma trận và một số vô hướng có thể thực hiện với mọi ma trận. Cú pháp tương tự như vectơ với số vô hướng, ví dụ: A+2; A-2; A.*2; A.^2; A./2 ; A.\2; Các phép tính này sẽ được thực hiện cho từng số hạng của ma trận. Ở đây có hai trường hợp cần lưu ý là **A.*2 = A*2** và **A./2 = A/2**
- Các phép tính giữa hai ma trận như cộng, trừ, chấm nhân, chấm chia chỉ thực hiện được với các ma trận có cùng kích thước (cùng số hàng và số cột). Cụ thể là:

  - Phép tính **A+B** hoặc **A-B** thực hiện cộng hoặc trừ tương ứng từng số hạng .

  - Phép chấm nhân **A .*B** thực hiện nhân tương ứng từng số hạng .

  - Phép chấm chia **A./B** hoặc **A.\B** thực hiện chia phải hoặc trái tương ứng từng số hạng.
- Phép nhân **A*B** được hiểu là phép nhân ma trận như trong toán học, chỉ thực hiện được với các ma trận tương thích (số cột của A bằng số hàng của B).
- Phép chia phải **A/B** tương ứng với trong toán học là $A.B^{-1}$
- Phép chia trái **A\B** tương ứng với trong toán học là $A^{-1}.B$ nhưng A\B dùng được cả khi ma trận A vuông hay không vuông, còn inv(A)*B chỉ dùng được khi A vuông.
- Phép luỹ thừa **A.^2** (có dấu chấm) thực hiện luỹ thừa từng số hạng tương ứng, có thể thực hiện với ma trận A bất kỳ. Còn phép luỹ thừa **A^2** ***tương đương với*** **A*A** , chỉ có nghĩa khi A là ma trận vuông.

Ví dụ 2: Kiểm chứng các phép tính trên ma trận

```
>> syms a1 a2 a3 a4 b1 b2 b3 b4;          % hàm syms để khai báo các biến chữ
>> A=[a1 a2; a3 a4] , B=[b1 b2; b3 b4]
A =                          B=
    [ a1, a2]                    [ b1, b2]
    [ a3, a4]                    [ b3, b4]
>> Tong=A+B
Tong=
    [ a1+b1, a2+b2]
    [ a3+b3, a4+b4]
>> Hieu=A-5
Hieu=
    [ a1-5, a2-5]
    [ a3-5, a4-5]
>> C1=A*B      % giống phép nhân ma trận trong toán học
C1=
    [ a1*b1+a2*b3,   a1*b2+a2*b4]
    [ a3*b1+a4*b3,   a3*b2+a4*b4]
>>C2=A.*B       % phép chấm nhân, thực hiện nhân từng phần tử tương ứng
C2=
    [ a1*b1,  a2*b2]
    [ a3*b3,  a4*b4]
>> A./B              % phép chấm chia phải
ans =
    [ a1/b1, a2/b2]
    [ a3/b3, a4/b4]
>> A.\B              % phép chấm chia trái
ans =
    [ b1/a1, b2/a2]
    [ b3/a3, b4/a4]
>>  D=A\B           % phép chia trái
D=
    [ -(a2*b3-b1*a4)/(a1*a4-a3*a2), -(a2*b4-b2*a4)/(a1*a4-a3*a2) ]
    [ (-a3*b1+a1*b3)/(a1*a4-a3*a2),  (a1*b4-a3*b2)/(a1*a4-a3*a2)]
```

3.5.2 Các hàm tìm kích thước, thành phần của ma trận :

TÊN HÀM	CHỨC NĂNG
size (A)	Tìm kích thước ma trận A
size (A,1)	Tìm số hàng của ma trận A
size (A,2)	Tìm số cột của ma trận A
rank(A)	Tìm số cột hoặc số hàng độc lập tuyến tính. Với ma trận vuông sẽ tìm hạng(cấp) của ma trận.
A(1,:)	Tìm hàng thứ nhất
A(:,2)	Tìm cột thứ hai
max(A)	Tạo vectơ hàng chứa các phần tử lớn nhất của mỗi cột.
min(A)	Tạo vectơ hàng chứa các phần tử bé nhất của mỗi cột.
numel(A)	Tìm tổng số phần tử của ma trận A
A(1,2)	Tìm phần tử ở hàng thứ 1, cột 2
A(1,2)=6	Thay phần tử ở hàng thứ 1 cột 2 bằng 6

Ví dụ 2: Cho 2 ma trận :

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad ; \quad Y = \begin{bmatrix} 0 & 2 & 4 \\ 6 & 5 & 1 \\ 3 & 9 & 7 \end{bmatrix}$$

(a) Tạo vectơ a chứa các phần tử ở hàng hai của ma trận X.
(b) Tạo vectơ b chứa hai phần tử cuối ở hàng hai của ma trận X.
(c) Tạo vectơ c chứa các phần tử ở cột ba của ma trận Y.
(d) Tạo vectơ hàng d chứa các phần tử ở hàng hai của X và cột ba của Y.
(e) Tạo vectơ cột e chứa các phần tử ở hàng hai của X và cột ba của Y.

Giải.

```
>> X=[1 2 3;4 5 6;7 8 9];   Y=[0 2 4;6 5 1;3 9 7];
>> a= X(2, :)
a=  4    5    6
>> b= a(2: 3)        % b là vectơ chứa 2 phần tử cuối của vectơ a
b=  5    6
>> c= Y(:, 3)
c=  4
    1
    7
>> d= [X(2, :) , Y(:, 3)']
d=   4    5    6    4    1    7
>> e= [X(2, :)' ; Y(:, 3)]      %   hoặc e=d'
e=   4
     5
     6
     4
```

1
7

3.5.2 Các hàm tạo ma trận :

TÊN HÀM	CHỨC NĂNG
zeros(m,n)	Tạo ma trận không (m x n)
ones(m,n)	Tạo ma trận (m x n) = 1
eye(n)	Tạo ma trận đơn vị cấp n
A=[]	Tạo ma trận rỗng A
magic(n)	Tạo ma trận magic cấp n
rand(n)	Ma trận cấp n với các phần tử ngẫu nhiên từ 0 đến 1
rand(m,n)	Ma trận (m x n), các phần tử ngẫu nhiên từ 0 đến 1
inv(A)	Ma trận nghịch đảo của ma trận vuông A
A'	Ma trận chuyển vị A^T của ma trận A
det(A)	Tính định thức của ma trận vuông A
poly(A)	Tìm đa thức đặc trưng của ma trận vuông A
eig(A)	Tìm giá trị riêng của ma trận vuông A
diag(A)	Lấy đường chéo chính của ma trận A
tril(A)	Lấy các phần tử từ đường chéo chính trở xuống
triu(A)	Lấy các phần tử từ đường chéo chính trở lên
fliplr(A)	Đảo ngược cột của ma trận A
flipud(A)	Đảo ngược hàng của ma trận A
jordan(A)	Chuyển ma trận A về dạng chính tắc (ma trận chéo)

Ví dụ 3. Tìm ma trận chuyển vị **B** = $\mathbf{A^T}$**,** với ma trận **A** được cho trong ví dụ 1.

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

Ví dụ 4. Tạo ma trận đơn vị có kích thước bằng kích thước ma trận A.

```
>> E  = eye(size(A)) ↵
E =
     1     0     0
     0     1     0
     0     0     1
```

Ví dụ 5. Tìm giá trị riêng và vectơ riêng của ma trận

Nếu ma trận A là ma trận vuông cấp n thì có n số λ thoả mãn Ax = λx . Giá trị λ gọi là giá trị riêng và vectơ cột x gọi là vectơ riêng của ma trận A. Tương ứng với mỗi giá trị riêng λ là một vectơ riêng x.

Trong Matlab, các giá trị riêng có thể tìm bằng cách dùng lệnh **eig(A)**. Để tìm đồng thời cả giá trị riêng và véctơ riêng của A có thể dùng lệnh **[X,D]=eig(A)**. Các phần tử trên đường chéo chính của ma trận chéo D là các λ, còn các cột của ma trận X là các véctơ riêng làm thoả mãn AX=XD.

```
>>A = [1 2 3;4 5 6;7 8 9] ; ↵
>>[X,D]=eig(A)
X =
   -0.2320   -0.7858    0.4082
   -0.5253   -0.0868   -0.8165
   -0.8187    0.6123    0.4082
D =
   16.1168         0         0
         0   -1.1168         0
         0         0   -0.0000
```

Ví dụ 6. Giải hệ phương trình tuyến tính cho ở dạng ma trận :

$$\begin{pmatrix} 1 & -3 & 1 \\ 2 & 1 & -4 \\ 4 & -2 & -1 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} ;$$

$$A.x = b \quad \Rightarrow \qquad x = \frac{b}{A} = A^{-1}.b$$

Chương trình MATLAB:

```
>>A=[1 -3  1 ; 2  1 -4 ; 4 -2  1]
>>b=[3; -1; 1]
>>x=inv(A)* b      % cũng có thể dùng phép chia trái :   x=A\b
```

Kết quả :

```
x =
   -0.2821
   -1.1538
   -0.1795
```

Ví dụ 7: Giải hệ phương trình tuyến tính có số phương trình = số ẩn

$$\begin{cases} 2x_1 + 2x_2 - 3x_3 + x_4 = 4 \\ 4x_1 + 3x_2 - x_3 + 2x_4 = 6 \\ 8x_1 + 5x_2 - 3x_3 + 4x_4 = 12 \\ 3x_1 + 3x_2 - 2x_3 + 2x_4 = 6 \end{cases}$$

Tính toán trong MATLAB :

```
>>A=[2 2 -3 1; 4  3 -1 2; 8 5 -3 4; 3 3 –2 2] ;   % tạo ma trận
>>b=[4;6;12; 6];
>>x=inv(A)*b    % cũng có thể dùng phép chia trái :    x=A\b
x =
     0.3333
     0.3333
    -0.3333
     1.6667
```

Ví dụ 8: Giải hệ 4 phương trình 3 ẩn :

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 366 \\ 4x_1 + 5x_2 + 6x_3 = 804 \\ 7x_1 + 8x_2 = 351 \\ 2x_1 + 5x_2 + 8x_3 = 514 \end{cases}$$

Trường hợp giải hệ phương trình có **số phương trình khác với số ẩn** cần tìm (không phải hệ Cramer) thì ma trận A không vuông nên không dùng được hàm inv(A). **Bắt buộc phải dùng phép chia trái x=A\b** .

```
>> A=[1 2 1;4 5 6; 7 3 0; 2 5 8]  ↵
A =
     1     2     3
     4     5     6
     7     8     0
     2     5     8
>>  b = [320; 880;452; 514];  ↵
>> x=A\b ↵
x =
    -14.1925
    203.8830
    -43.4226
```

3.6 ĐA THỨC

Cho đa thức bậc n :

$$p = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

Trong MATLAB, đa thức được biểu diễn như một vectơ hàng với các phần tử là các hệ số của đa thức sắp theo thứ tự giảm dần **từ bậc cao nhất đến bậc 0** .

Nhận xét: Đa thức bậc n tương ứng với **véctơ hàng có (n+1) phần tử**.

>> p = a_n a_{n-1} ... a_1 a_0

Ví dụ, nhập đa thức: $p = x^4 + 3x^3 - x^2 - 5x + 1$

```
>> p = [1  3 -1  -5  1] ↵
p =   1    3   -1    -5     1
```

Các phép tính với đa thức:

HÀM	Ý NGHĨA
conv(p1,p2)	Nhân hai đa thức
[k,d]=deconv (p1,p2)	Chia hai đa thức (k= kết quả; d =phần dư)
k=polyder(p)	Tìm đạo hàm của đa thức p
k=polyder(p,q)	Tìm đạo hàm của đa thức tích (p*q)
[n,d]=polyder(num,den)	Tìm đạo hàm (dạng n/d) của phân thức (num/den)
roots(p)	Tìm nghiệm đa thức p
p=poly(r)	Lập đa thức p từ vectơ r chứa các nghiệm.
polyval(p,x)	Tính giá trị của đa thức tại x (x có thể là mảng)
[r,p,k]= residue(num,den)	Tìm các thành phần tối giản của phân thức
[num,den]=residue(r,p,k)	Chuyển các thành phần tối giản thành 1 phân thức
printsys(num,den,'s')	in phân thức có dạng tỉ số 2 đa thức theo s
[z,p,k]=tf2zp(num,den)	Tìm các zero z, cực p, độ lợi k của phân thức

Ví dụ 1:

```
>> p = [1  3 -1  -5  1] ; r =roots(p)
r =
   -2.5764
   -1.7475
    1.1268
    0.1971
>> p= poly(r)            % Tái tạo đa thức p từ vectơ nghiệm r
p =
    1    3   -1    -5    1
>> q=[1 2]               %  q=x+2
>> w =conv(p,q)       % w=(x^4 + 3x^3 - x^2 -5x + 1)(x+2)
w =
    1    5    5   -7   -9    2
```

Kết quả trên tương ứng với đa thức : $x^5 + 5x^4 + 5x^3 - 7x^2 - 9x + 2$

```
>>w2=conv(conv(p,q), [4 1])      %  w2= (x^4 - 3x^3 - x^2 - 5x - 1)(x-2)(4x-1)
w2=
     4    21    25   -23   -43    -1     2
```

Hai đa thức cùng bậc có thể cộng trừ theo phép cộng trừ thông thường. MATLAB không có hàm để cộng trừ các đa thức khác bậc. Trường hợp này ta có thể tạo chương trình riêng để dùng hoặc áp dụng cách đơn giản sau đây :

Ví dụ cần cộng hai đa thức

$p = x^4 + 3x^3 – x^2 – 5x + 1$

$q = x+2$

Ta nhập vào MATLAB:

```
>> p=[1 3 -1 -5 1];
>> q=[0 0  0  1 2]; tong = p + q       % hoặc   >> q= [1 2] ; tong = p + [0  0  0  q]
tong =
       1     3    -1    -4     3
```

Ví dụ 2: Tìm các thành phần tối giản của phân thức $G(s) = \dfrac{2s+1}{(s+2)(s^2+5s+4)}$

```
>>num= [2 1];
>>den= conv( [1 2],[1  5  4] ) ;
>>[r,p,k]=residue(num,den)
```

Kết quả:

```
r =
    -1.1667
     1.5000
    -0.3333
p =
    -4.0000
    -2.0000
    -1.0000
k =
     []
```

Trường hợp này G(s) chỉ có các cực riêng biệt p(1), p(2), p(3).
(nói cách khác là mẫu số của G(s) chỉ có nghiệm đơn)

Do đó có thể phân tích :

$$G(s) = \frac{r(1)}{s-p(1)} + \frac{r(2)}{s-p(2)} + ... + \frac{r(n)}{s-p(n)} + k(s) = \frac{-1.1667}{s+4} + \frac{1.5}{s+2} + \frac{-0.3333}{s+1}$$

trong đó k(s) là đa thức theo s, tương ứng với vectơ k.

Ví dụ: k=[1;3] thì k(s)= s+3 ; k=3 thì k(s)=3 ; k=[] thì k(s)=0 .

Ví dụ 3: cho ảnh Laplace $Y(s) = \dfrac{2s^3 + s^2 + 3s + 5}{s(s+1)^2}$

Hãy phân tích Y(s) thành tổng các thành phần tối giản.

```
>> num=[2 1 3 5];
>>den=conv([1 0],conv([1 1],[1 1]));
>> printsys(num,den,'s')
   num/den =
                    2 s^3 +  s^2 + 3 s + 5
                  ------------------------
                    s^3 + 2 s^2 +  s
>> [r,p,k]=residue(num,den)
   r =
       -8
       -1
        5
   p =
       -1
       -1
        0
   k =
        2
```

Trường hợp này Y(s) có cực bội p(1) = p(2) = -1
(nói cách khác là mẫu số của Y(s) có nghiệm bội p(1) = p(2) = -1)

Do đó Y(s) có thể phân tích thành tổng :

$$Y(s) = \frac{r(1)}{s - p(1)} + \frac{r(2)}{[s - p(1)]^2} + \frac{r(3)}{s - p(3)} + k = \frac{-2}{(s+1)} + \frac{-3}{(s+1)^2} + \frac{3}{s} + 2$$

Ví dụ 4: Biểu diễn phân thức sau dưới dạng zero - cực (zero-pole-gain):

$$G(s) = \frac{4s^2 + 16s + 12}{s^4 + 12s^3 + 44s^2 + 48s}$$

```
>> num=[4 16 12]
>> den=[1 12 44 48 0]
>> [z,p,k]=tf2zp(num,den)
   z =
      -3
      -1
   p =
         0
    -6.0000
    -4.0000
    -2.0000
   k =
       4
```

Do đó :

$$G(s) = \frac{K(s - z_1)(s - z_2)}{(s - p_1)(s - p_2)(s - p_3)} = \frac{4(s+3)(s+1)}{(s+6)(s+4)(s+2)}$$

CHƯƠNG 4

LỆNH ĐIỀU KIỆN VÀ VÒNG LẶP

4.1 Biểu thức logic

Biểu thức logic thường được sử dụng để biểu diễn **điều kiện** trong các vòng lặp hay trong các câu lệnh điều kiện. Các biểu thức logic trong MATLAB được thành lập trên cơ sở các toán tử quan hệ và toán tử logic. Toán tử quan hệ là các ký hiệu thể hiện sự so sánh, toán tử logic là các ký hiệu dùng để liên kết các biểu thức logic.

TOÁN TỬ QUAN HỆ	Ý NGHĨA
<	nhỏ hơn
<=	nhỏ hơn hoặc bằng
>	lớn hơn
>=	lớn hơn hoặc bằng
==	bằng
~=	khác

TOÁN TỬ LOGIC	Ý NGHĨA
&	và
|	hoặc
~	không

Biểu thức logic cho kết quả chân trị là đúng (true) hoặc sai (false). Trong MATLAB, biểu thức đúng sẽ có giá trị là 1, biểu thức sai có giá trị là 0.

Ví dụ:

12.5>12 là biểu thức logic, có giá trị là 1.

6~=6 là biểu thức logic, có giá trị là 0.

b==6 có giá trị là 1 nếu b=6, có giá trị là 0 nếu b khác 6.

(12.5>12)& (5>6) có giá trị là 0.

MATLAB cũng cung cấp các hàm có chức năng kiểm tra, so sánh và trả về kết quả logic là 1 (true) hoặc 0 (false). Các hàm thông dụng nhất là:

HÀM	Ý NGHĨA
ischar(s)	True nếu s là chuỗi ký tự
isstr(s)	True nếu s là chuỗi ký tự
isnumeric(x)	True nếu x là số (con số, mảng số,...)
isempty(x)	True nếu x (chuỗi, mảng, ma trận,...) là rỗng
strcmp(s1,s2)	True nếu 2 chuỗi s1, s2 giống nhau
isglobal(x)	True nếu x là biến toàn cục

4.2 Các câu lệnh điều kiện

1) Cấu trúc if – end

```
if <điều kiện>
    Khối các lệnh thực hiện nếu điều kiện là đúng
end
```

2) Cấu trúc if – else –end

```
if <điều kiện>
    Khối các lệnh thực hiện nếu điều kiện là đúng
else
    Khối các lệnh thực hiện nếu điều kiện là sai
end
```

3) Cấu trúc if – elseif – else – end

```
if <điều kiện 1>
    Khối các lệnh thực hiện nếu điều kiện 1 đúng
elseif <điều kiện 2>
    Khối các lệnh thực hiện nếu điều kiện 2 đúng
elseif <điều kiện 3>
    Khối các lệnh thực hiện nếu điều kiện 3 đúng
else
    Khối các lệnh thực hiện nếu không có điều kiện nào đúng
end
```

Ví dụ : Viết chương trình yêu cầu người dùng nhập vào từ bàn phím điểm số của một học sinh. Nếu điểm số từ 1 đến 4 thì xuất ra dòng nhắn "loại yếu", nếu điểm số là 5 hoặc 6 thì xuất ra dòng nhắn "loại trung bình", nếu điểm số là 7 hoặc 8 thì xuất dòng nhắn "loại khá", nếu điểm số là 9 hoặc 10 thì xuất dòng nhắn "loại giỏi". Nếu điểm số nằm ngoài phạm vi từ 1 đến 10 thì xuất dòng nhắn "Số liệu không hợp lệ".

```
diem= input('Nhap diem so: ');
if (diem>=1)&(diem<=4)
       fprintf('loai yeu')
elseif (diem==5)|(diem==6)
       fprintf('loai trung binh')
elseif (diem==7)|(diem==8)
       fprintf('loai kha')
elseif (diem==9)|(diem==10)
       fprintf('loai gioi')
else
       fprintf('So lieu khong hop le')
end
```

4) Cấu trúc switch-case

```
switch  <điều kiện>
    case  giá trị thử 1
          khối lệnh 1
    case  {giá trị thử 2, giá trị thử 3,...}
          khối lệnh 2
    otherwise
          khối lệnh 3
end
```

Điều kiện trong cấu trúc **switch-case** phải có giá trị dạng số nguyên hoặc dạng chuỗi. lệnh **case** sẽ so sánh giá trị của điều kiện với các giá trị thử để thực hiện khối lệnh tương ứng. Nếu không có giá trị thử nào phù hợp thì thực hiện khối lệnh 3.

Ví dụ:

```
diem=input('Nhap diem so: ');
switch (diem)
    case {1,2,3,4}
       disp('loai yeu')
    case {5,6}
       disp('trung binh')
    case {7,8}
       disp('loai kha')
    case {9,10}
       disp('loai gioi')
  otherwise
       disp('So lieu khong hop le')
end
```

4.3 Vòng lặp

4.3.1 Vòng lặp for

```
for  i= i1: Δi : i2          % hoặc i=i1:i2   nếu  Δi = 1
    Khối các lệnh
end
```

Ví dụ 1:

```
disp('chương trình tính tổng giai thừa 1!+2! +3!+...+n!')
n = input ('Nhap gia tri n:');
if n==0, tong=1;
else
      tong=0;  % gia tri khoi dau
      for k=1:n
          tong= tong+prod(1:k);
      end
end
disp(['Tong can tim la:',num2str(tong)]);
```

Ví dụ 2:

```
% chương trình giải tìm nghiệm đa thức bậc n
% Các hệ số nhập theo dạng véctơ hàng. Ví dụ: [2 5 3 1]
p = input ('Nhap cac he so cua da thuc:');
x = roots(p)
disp(['Da thuc bac ',num2str(length(x)),', co cac nghiem la:'])
for i=1:length(x)
    disp(['Nghiem thu ',num2str(i),'= ',num2str(x(i))])
end
```

Ví dụ 3:

```
% chương trình in ra và tính tổng n số chẵn đầu tiên
% S=2+4+...+2*n    bằng vòng lặp for
n=input('nhap gia tri n: ');
S=0;
for i=1:n
     S=S+2*i;
     fprintf('%2d la so chan thu:%3d\n',2*i,i)
end
fprintf('Tong %2d so chan dau tien la:%3d\n',n,S)
```

4.3.2 Vòng lặp while

while <điều kiện>
Khối các câu lệnh thực hiện nếu điều kiện còn đúng
end

Khác với vòng lặp for, số lần lặp của vòng lặp while không được xác định. Ta xét ví dụ tính tổng n số chẵn ở phần trên bằng vòng lặp while (thay vì for) :

```
% chương trình in ra và tính tổng n số chẵn đầu tiên
% S=2+4+...+2*n  bằng vòng lặp while
n = input ('Nhap gia tri n: ');
while n<=0,
      disp('Dieu kien: n>0'),
      n = input ('Nhap lai gia tri n: ');
end
S=0;i=1;
while  i<=n
      S=S+2*i;
      fprintf('%2d la so chan thu:%3d\n',2*i,i)
      i=i+1;
end
disp(['Tong can tim la: ',num2str(S)]);
```

Ghi chú : Việc kiểm tra điều kiện nhập n>0 cũng được thực hiện bằng vòng lặp while để có thể lặp vô hạn lần. Nếu kiểm tra bằng cấu trúc if – end thì chỉ lặp được 1 lần.

4.4 Các lệnh tạo sự gián đoạn

- **Lệnh continue**

Trong vòng lặp for hay while, khi gọi **continue** ngay lập tức chu trình tính chuyển sang bước lặp kế tiếp, mọi lệnh chưa được thực hiện của vòng lặp (***thuộc về bước lặp hiện tại***) sẽ bị bỏ qua.

- **Lệnh break**

Lệnh **break** mạnh hơn lệnh continue, nó làm ngừng ngay lập tức vòng lặp đang tính. Lệnh break có tác dụng cả trong các cấu trúc điều kiện if, switch. Nếu break được sử dụng ngoài vòng lặp for hay while trong phạm vi của một M-file, khi ấy M-file sẽ bị ngừng tại vị trí của break.

Ví dụ: Viết đoạn chương trình tìm và in ra màn hình các số nguyên tố bé hơn 100.

```
clear, clc
disp('================================================')
disp('Chuong trinh tim va in ra cac so nguyen to < 100')
fprintf('%3d la so nguyen to thu:%3d\n',2,1)
count=1; % bien dung de dem so thu tu
for m= 3:1:100
    for k=2:1:m-1
       if mod(m,k)==0, break, end
    end
    if k == m-1,
      count=count+1;
      fprintf('%3d la so nguyen to thu:%3d\n',m,count)
    end
end
```

- **Lệnh return**

Việc thi hành các M-file hàm sẽ kết thúc khi gặp dòng cuối cùng của file đó hoặc gặp dòng lệnh **return**. Lệnh return giúp ta kết thúc một hàm mà không cần phải thi hành hết các lệnh của hàm đó.

- **Hàm error**

Hàm **error** sẽ hiển thị một chuỗi lên cửa sổ lệnh và dừng thực hiện hàm, trả điều khiển về cho cửa sổ lệnh và bàn phím. Hàm này rất hữu dụng để cảnh báo việc sử dụng hàm không đúng mục đích.

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

errordlg('dòng nhắn') : kết thúc thực thi lệnh và hiển thị hộp thoại chứa dòng nhắn.

Ví dụ:

```
% chương trình tính định thức ma trận A
A = input('Nhap ma tran vuong A:')
if isempty(A)    % Nếu A rỗng
   errordlg('Nhap lai ma tran A','Dialog')
   return
else
       DET=det(A)
end
```

CHƯƠNG 5

ĐỒ HOẠ VỚI MATLAB

Phần I. ĐỒ HỌA 2D

5.1 VẼ ĐỒ THỊ BẰNG HÀM PLOT

Hàm **plot** vẽ đồ thị 2D dựa trên hai mảng dữ liệu số do người dùng tạo trước. Nếu dùng hàm plot để vẽ đồ thị hàm số thì số điểm dữ liệu càng nhiều, hình vẽ càng đúng với đồ thị hàm số liên tục (đường cong trơn và liên tục).

5.1.1. Vẽ căn bản

Lệnh **plot(x,y)** : vẽ đồ thị y theo x.

Ví dụ: Vẽ đồ thị các hàm sau:

y1= x^2+3x + 5 trong khoảng [0,10]

y2= sin(x) trong khoảng [0,3π]

Thực hiện trong MATLAB:

```
>> x=[0: 0.01: 10];        % tạo mảng x có giá trị từ 0 đến 10 với gia số 0,01
>> y1=x.^2+3*x+5;          % tạo mảng y (= tính các giá trị tương ứng của y)
>> plot(x,y1)              % vẽ đồ thị y1 theo x

>> x=[0: 0.01: 3*pi];
>> y2=sin(x);
>> plot(x,y2)
```

5.1.2. Vẽ có khai báo màu, kiểu nét và đánh dấu điểm dữ liệu

Cú pháp lệnh: **plot(x,y,S)**

trong đó tham số S là chuỗi ký tự tuỳ chọn để khai báo màu vẽ, kiểu nét và/ hoặc ký hiệu đánh dấu tại các điểm dữ liệu. Nếu không dùng tham số S thì mặc định là màu xanh dương (blue), nét liền (solid), không có ký hiệu đánh dấu.

a) Màu (color)

b	= blue	m	= magenta
g	= green	y	= yellow
r	= red	k	= black
c	= cyan	w	= white

b) Kiểu nét (linestyle) và ký hiệu đánh dấu (marker)

Linestyle	Marker	
- solid	x x-mark	v triangle (down)
: dotted	+ plus	^ triangle (up)
-. dashdot	* star	< triangle (left)
-- dashed	s square	> triangle (right)
. point	o circle	p pentagram
	d diamond	h hexagram

Ví dụ: Vẽ đồ thị hàm số

$y1 = 1 - 3e^{-2t} + 2e^{-3t}$ % trong khoảng [0,10], chọn màu blue, nét liền

$y2 = 1 + 2e^{-t}\sin(2t - \pi/6)$ % trong khoảng [0,6], chọn màu đỏ, nét gạch chấm

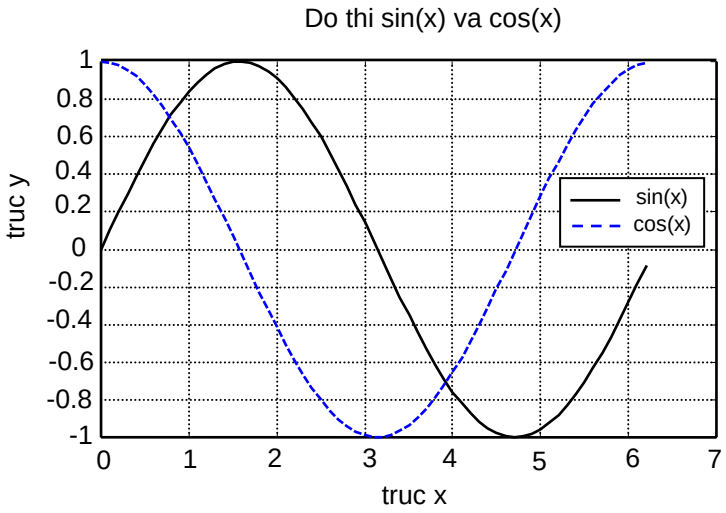
```
t =[0: 0.001:10];
    y1=1-3*exp(-2*t)+2*exp(-3*t);
   plot(t,y1)              % mặc định là màu blue, nét liền

    t =[0: 0.01:6];
    y2= 1+2*exp(-t).*sin(2*t-pi/6);
    plot(t,y2,'r-. ')       % màu đỏ, nét gạch chấm.
```

5.1.3. Các lệnh tiện tích

1. title('tên đồ thị') % tạo tiêu đề đồ thị
2. xlabel('nhãn trục x') % tạo nhãn cho trục x
3. ylabel('nhãn trục y') % tạo nhãn cho trục y
4. text(x,y,'nhãn') % tạo nhãn tại vị trí có toạ độ (x,y)
5. grid *hoặc* grid on % hiện các ô lưới toạ độ.
6. hold *hoặc* hold on % giữ nguyên đồ thị để vẽ tiếp trên cùng hệ trục toạ độ.
7. legend('chúthích1', 'chúthích2',...) % tạo ô chú thích khi vẽ nhiều đồ thị .

Ví dụ :

```
>> x=[0:0.1:2*pi];
>> y1=sin(x);
>> plot(x,y1,'k');
>> hold on
>> y2=cos(x);
>> plot(x,y2,'b--');
>> title('Do thi sin(x) va cos(x)')
>> xlabel('truc x')
>> ylabel('truc y')
>> legend('sin(x)','cos(x)')
>> grid
```



Để vẽ hai hay nhiều đồ thị trên cùng một hệ trục toạ độ ta có thể dùng lệnh **hold on** như trên hoặc dùng lệnh **plot** với cú pháp tổng quát :

plot(x1,y1,S1,x2,y2,S2,…)

trong đó x1,y1,S1 ứng với đồ thị thứ nhất; x2,y2,S2 ứng với đồ thị thứ hai,…

Ở ví dụ trên, thay vì dùng lệnh **hold on** ta có thể vẽ kết hợp hai đồ thị y1 và y2 bằng một lệnh **plot** duy nhất như sau:

```
>> plot(x,y1,'k',x,y2,'b--')
```

5.2 HỆ TRỤC TOẠ ĐỘ (axis), CỬA SỔ VẼ (figure), ĐỒ THỊ CON (subplot)

Lệnh **axis** là công cụ dùng để quản lý hình dáng và thang chia của cả hai trục đứng và ngang. Lệnh này có nhiều tuỳ chọn, để biết một cách đầy đủ về nó, bạn có thể gõ lệnh **help axis** hay **doc axis**. Một số cách thường dùng của lệnh axis là:

LỆNH	Ý NGHĨA
axis([xmin xmax ymin ymax])	Thiết lập các giá trị min, max của hệ trục 2D
axis([xmin,xmax,ymin,ymax])	
axis([xmi xma ymi yma zmi zma])	Thiết lập các giá trị min, max của hệ trục 3D
axis square	Lấy độ dài hai trục bằng nhau (tạo vùng bao vuông, so với mặc định là chữ nhật)
axis equal	Lấy thang chia giống nhau cho cả hai trục
axis off	Tắt bỏ chế độ nền trục, nhãn, ô lưới,...
axis on	Ngược lại với axis off

Nếu muốn vẽ nhiều đồ thị trên các figure (cửa sổ vẽ) khác nhau, ta tạo figure mới bằng lệnh **figure** hoặc chọn menu **file > new > figure** trong cửa sổ figure đang vẽ. Mỗi đối tượng đồ họa tạo mới như figure, axis, line,... được MATLAB tự động gán cho một ***số hiệu*** để quản lý, gọi là **handle**. Trường hợp tổng quát thì giá trị handle là một số thực. Riêng đối với figure thì mỗi figure tạo mới sẽ được gán với handle là một số nguyên dương, ví dụ: 1, 2, 3,.... Bạn có thể chuyển qua lại giữa các figure đang có bằng cách dùng chuột để chọn hoặc dùng lệnh **figure(H)** trong đó H là số hiệu của figure.

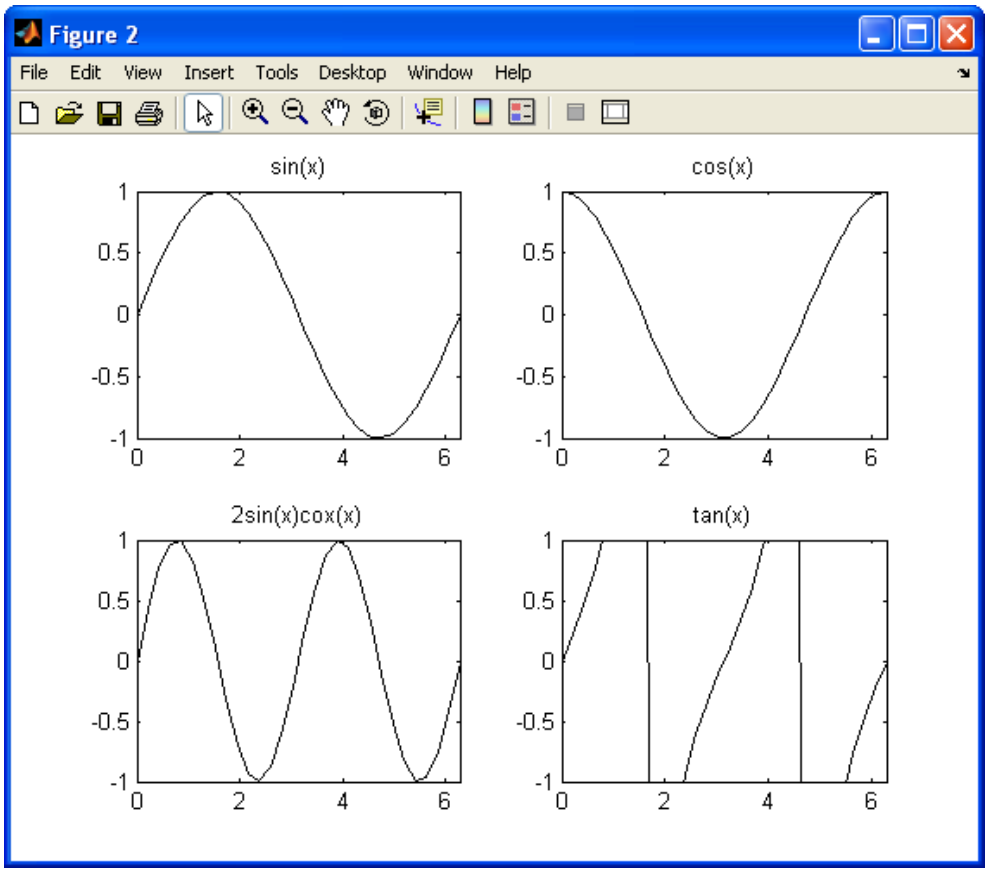
Các lệnh tìm handle thường dùng là **gcf** (Get handlle to current figure – tìm handle của figure hiện hành), **gca** (Get handle to current axis – tìm handle của hệ trục hiện hành), **findobj(gcf, 'Type','Line')** (tìm handle của các đường đồ thị trong figure hiện hành).

Mặt khác, một cửa sổ figure có thể chứa nhiều hơn một hệ trục. Lệnh **subplot(m,n,p)** chia figure thành một ma trận (m x n) vùng đồ hoạ con gọi là subplot, và chọn p là subplot hiện hành. Các subplot được đánh số lần lượt từ trái qua phải, từ trên xuống dưới.

Ví dụ:

```
x= linspace(0,2*pi,30);   % tạo mảng x từ 0 đến 2*pi  có 30 phần tử
y= sin(x);  z= cos(x);
u= 2*sin(x).*cos(x);  v= tan(x);
figure                    % mở mới một cửa sổ figure trống.
subplot(2,2,1)            % tạo 4 subplot và chọn subplot 1 ở góc trên trái .
plot(x,y), axis([0  2*pi  -1  1]), title('sin(x)')
subplot(2,2,2)            % chọn subplot 2 ở góc trên phải .
plot(x,z), axis([0  2*pi  -1  1]), title('cos(x)')
subplot(2,2,3)            % chọn subplot 3 ở góc dưới trái .
plot(x,u), axis([0  2*pi -1  1]), title('2sin(x)cox(x)')
subplot(2,2,4)            % chọn subplot 4 ở góc dưới phải .
plot(x,v), axis([0  2*pi  -1  1]), title('tan(x)')
```

Kết quả :



5.3 VẼ ĐỒ THỊ BẰNG HÀM EZPLOT

Hàm **ezplot** dùng để vẽ đồ thị của hàm số cho bởi biểu thức chữ.

LỆNH	Ý NGHĨA
ezplot(y)	Vẽ đồ thị hàm y=f(x) , mặc định là x ∈ [-2π,2π]
ezplot(y, [a,b])	Vẽ đồ thị hàm y=f(x) trong khoảng x ∈ [a,b]
ezplot(f)	Vẽ đồ thị f(x,y) = 0 , mặc định là x và y ∈ [-2π, 2π]
ezplot(f, [a,b])	Vẽ đồ thị f(x,y) = 0 với x và y ∈ [a,b]
ezplot(f, [xmin,xmax,ymin,ymax])	Vẽ đồ thị f(x,y) = 0 với x ∈ [xmin,xmax] ; y ∈ [ymin,ymax]
ezplot(x,y)	Vẽ đồ thị hàm tham số x=x(t); y=y(t) với t ∈[-2π, 2π]
ezplot(x,y, [tmin,tmax])	vẽ đồ thị x=x(t); y=y(t) với t ∈[tmin, tmax]

Hàm số cần vẽ có thể nhập theo nhiều cách. Hai cách thường dùng là:

Cách 1. Nhập biểu thức hàm trong cặp dấu nháy ' '

Cách 2. Khai báo biến bằng hàm syms ; sau đó nhập biểu thức hàm

Ví dụ, hàm y=2sinxcosx có thể nhập bằng một trong hai cách:

Cách 1: >> y= '2*sin(x)*cos(x)'

Cách 2: >> syms x ; y=2*sin(x)*cos(x)

Khi vẽ bằng hàm ezplot thì tiêu đề đồ thị sẽ được tạo tự động (không cần dùng lệnh title). Sau khi vẽ bạn cũng có thể dùng các lệnh tiện ích **hold, grid, legend, axis,…**tương tự đồ thị của hàm plot. Các tuỳ chọn về màu, kiểu nét và marker không có trong cú pháp của lệnh **ezplot** do đó nếu muốn thiết đặt theo ý riêng, bạn phải điều chỉnh thông qua **handle** và lệnh **set** như sau:

Cách 1:

h=ezplot(y) % vẽ và lưu handle của đường đồ thị vào biến h

set(h, 'Color','màu','Linestyle','kiểunét',…) **%** thiết đặt lại màu, kiểu nét,…

Cách 2: (MATLAB 6.x chỉ dùng được cách này)

ezplot(y) **%** vẽ đồ thị

h= findobj(gcf,'Type','Line') % tìm handle của đường đồ thị

set(h, 'Color','màu','Linestyle','kiểunét',…) **%** thiết đặt lại màu, kiểu nét,…

Cũng có thể kết hợp hai lệnh trên thành một lệnh :

set(findobj(gcf,'Type','Line'),'Color','màu','Linestyle','kiểunét',…)

Ví dụ:
```
>> set(findobj(gcf,'Type','Line'),'Color','r','Linestyle','--')
>> set(findobj(gcf,'Type','Line'),'Color','k','Marker','*')
```

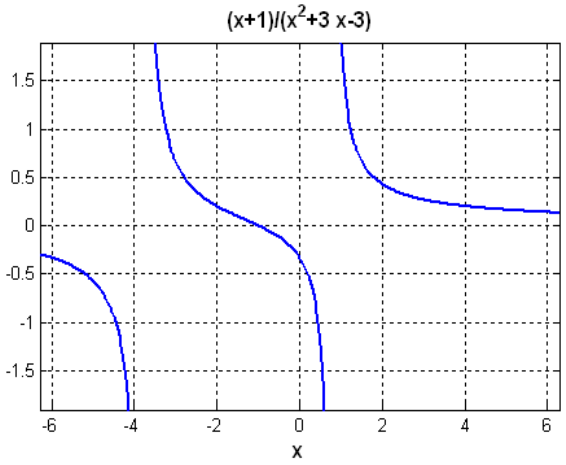
1) Vẽ đồ thị hàm y=y(x) hay y=y(t) trong khoảng mặc định

Ví dụ: Vẽ đồ thị hàm số

$$y = \frac{x+1}{x^2+3x-3}$$

```
% nhập hàm theo cách 1
>> y= '(x+1)/(x^2+3*x-3)'
>> ezplot(y)
>> grid
```

Khoảng mặc định mà Matlab tự chọn là $x \in [-2\pi, 2\pi]$



2) Vẽ đồ thị hàm y=y(x) hay y=y(t) trong khoảng tùy chọn

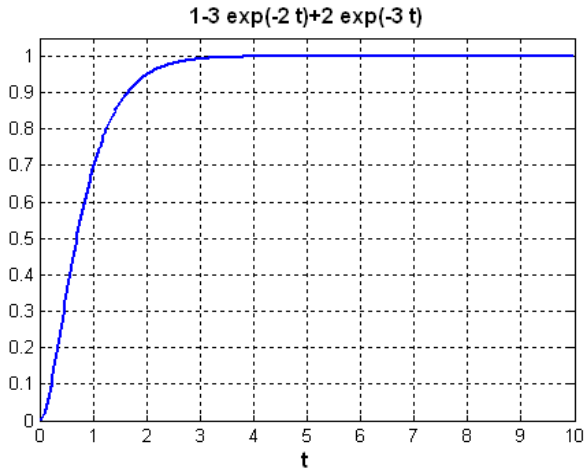
Ví dụ:

- Vẽ đồ thị hàm số

$$y = 1 - 3e^{-2t} + 2e^{-3t}$$

trong khoảng $t \in [0, 10]$

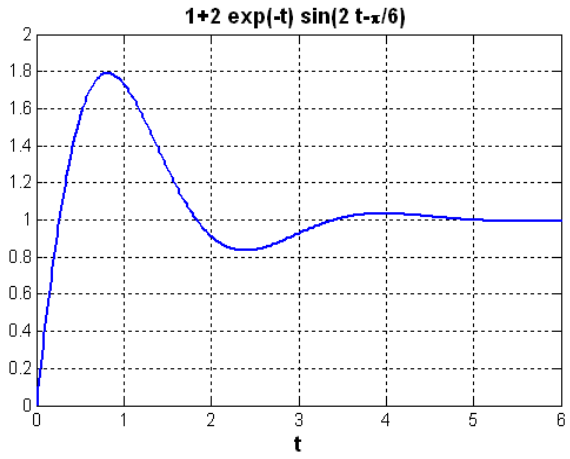
```
% nhập hàm theo cách 2
 syms t
y=1- 3*exp(-2*t) +2*exp(-3*t)
 ezplot(y,[0,10])
 axis([0,10, 0, 1.05])
 grid
```



- Vẽ đồ thị hàm số

$$y = 1 + 2e^{-t}\sin(2t - \pi/6)$$

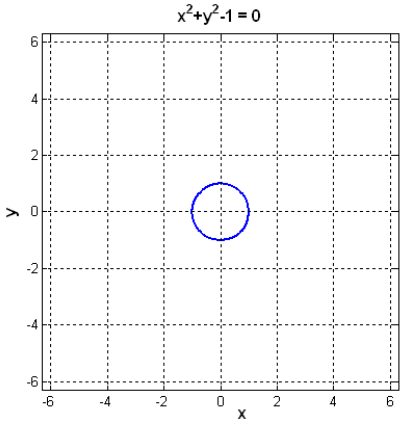
trong khoảng $t \in [0, 6]$




```
%  nhập hàm theo cách 1
>> y= '1+2*exp(-t)*sin(2*t-pi/6)'
>> ezplot(y,[0,6])
>> axis([0,6,0,2]); grid
```

3) Vẽ đồ thị hàm f(x,y)=0 trong khoảng mặc định

Ví dụ: Vẽ đồ thị hàm số $x^2 + y^2 = 1$



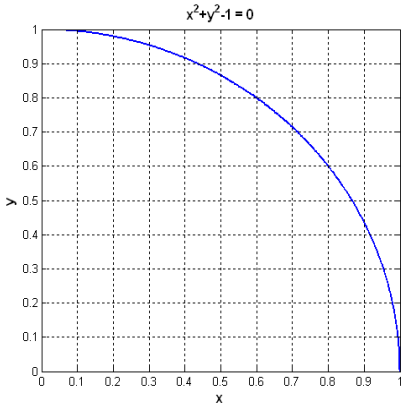

```
>> f= 'x^2+ y^2 - 1'
>> ezplot(f)
>> axis square ; grid
```

Khoảng mặc định mà Matlab tự chọn là $x \in [-2\pi, 2\pi]$, $y \in [-2\pi, 2\pi]$

4) Vẽ đồ thị hàm f(x,y)=0 trong khoảng tùy chọn

Ví dụ: Vẽ đồ thị hàm số : $x^2 + y^2 = 1$

Với $x \in [0,1]$, $y \in [0,1]$

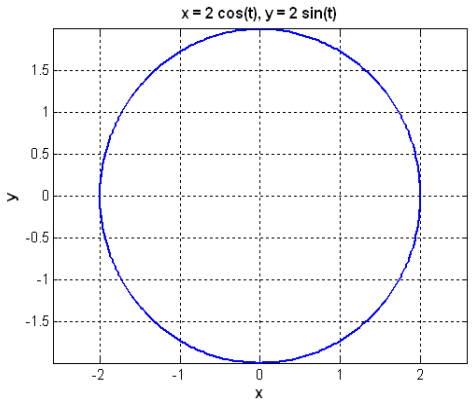



```
>> f= 'x^2 + y^2 -1'
>> ezplot(f,[0,1,0,1])
>> axis square ; grid
```

5) Vẽ đồ thị hàm x=x(t); y=y(t) trong khoảng mặc định $t \in [-2\pi, 2\pi]$

Ví dụ: Vẽ đồ thị hàm số

$$\begin{cases} x = 2\cos t \\ y = 2\sin t \end{cases}$$




```
>> x= '2*cos(t)';
>> y= '2*sin(t)';
>> ezplot(x,y) ; grid
```

Matlab mặc định là $t \in [-2\pi, 2\pi]$

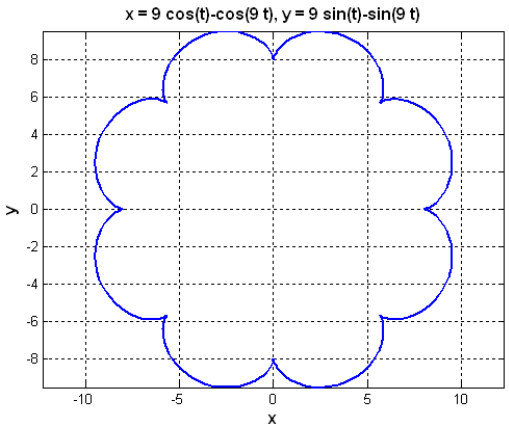
6) Vẽ đồ thị hàm x=x(t); y=y(t) trong khoảng tùy chọn của t

Ví dụ: Vẽ đồ thị hàm số

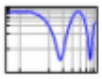
$$\begin{cases} x = 9\cos t - \cos 9t \\ y = 9\sin t - \sin 9t \end{cases}$$

trong khoảng $t \in [0, 2\pi]$

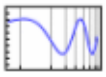
```
>> syms t    % cách nhập 2
>> x = 9*cos(t) - cos(9*t) ;
>> y = 9*sin(t) - sin(9*t) ;
>> ezplot(x,y,[0,2*pi])
>> grid
```



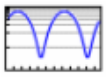
5.4 MỘT SỐ HÀM VẼ 2D KHÁC :



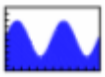
- Hàm **loglog** tương tự như plot ngoại trừ thang chia là logarit cho cả hai trục x, y.



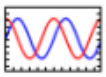
- Hàm **semilogx** tương tự như plot ngoại trừ thang chia của trục x là logarit còn thang chia của trục y là tuyến tính



- Hàm **semilogy** tương tự như plot ngoại trừ thang chia của trục y là logarit còn thang chia của trục x là tuyến tính

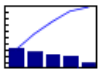


- Hàm **area(x,y)** tương tự như plot(x,y) nhưng có tô màu phần diện tích giới hạn bởi đường cong y, các đường thẳng x=xmin, x=xmax và trục hoành.

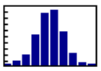


- Hàm **plotyy** vẽ hai đồ thị khác nhau trên cùng một hệ trục nhưng dùng 2 trục y, 2 trục này có thể dùng thang chia khác nhau.

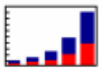
- Hàm **pie(a,b)** vẽ sơ đồ hình bánh pie, với a là một vectơ giá trị và b là một vectơ logic tuỳ chọn.



- Hàm **pareto(y)** vẽ biểu đồ paretogram, với y là vectơ giá trị.



- Hàm **hist (y,x)** vẽ biểu đồ phân bố dữ liệu histogram.



- Các hàm **bar**, **barh**, **stairs, stem** tạo đồ thị dạng bar (dạng thanh đứng hoặc ngang), stair (dạng bậc thang), stem (dạng hình que)

- Hàm **rose(v)** vẽ đồ thị trong hệ toạ độ cực cho các góc trong vectơ v. Các hàm **rose(v,n)** và **rose(v,x)** trong đó x là một vectơ có chức năng tương tự .

- Hàm **polar(theta,rho)** vẽ đồ thị trong hệ toạ độ cực dựa trên hai mảng dữ liệu là vectơ góc quay theta và vectơ bán kính rho.

Phần II. ĐỒ HOẠ 3D

5.6 ĐỒ THỊ ĐƯỜNG 3D

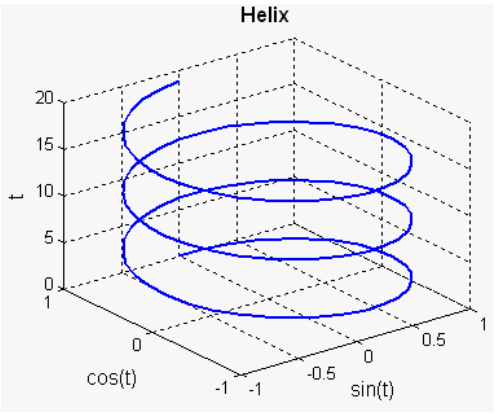
Hàm **plot3** và **ezplot3** là các phiên bản 3D của **plot** và **ezplot**

plot3(x,y,z,S) trong đó x, y, z là các véctơ, S là chuỗi ký tự tuỳ chọn dùng cho việc khai báo màu, kiểu nét và marker.

ezplot3(x,y,z,[tmin,tmax]) vẽ đồ thị hàm f(x,y,z) với x=x(t),y=y(t), z=z(t) trong khoảng t ∈ [tmin,tmax].

Ví dụ:

```
t=linspace (0,6*pi)
plot3(sin(t),cos(t),t)
title('Helix'), xlabel('sin(t)')
ylabel('cos(t)'), zlabel('t')
grid
```



Đường xoắn ốc 3D này cũng có thể vẽ bằng hàm ezplot3 như sau:

```
>> x='sin(t)' ; y='cos(t)' ; z='t' ;  ezplot3(x,y,z,[0, 6*pi])
```

hoặc:

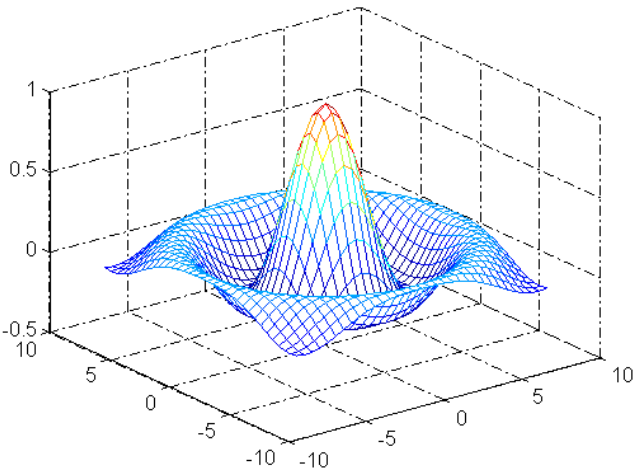
```
>> ezplot3('sin(t)','cos(t)','t',[0, 6*pi])
```

5.7 ĐỒ THỊ LƯỚI VÀ BỀ MẶT 3D

Hàm **mesh(X,Y,Z)**, với X,Y,Z là các ma trận, vẽ Z theo X và Y. Nó sắp xếp giá trị các phần tử ma trận vào các điểm (X,Y,Z) trong không gian 3D và tạo nên một mặt cong có dạng lưới.

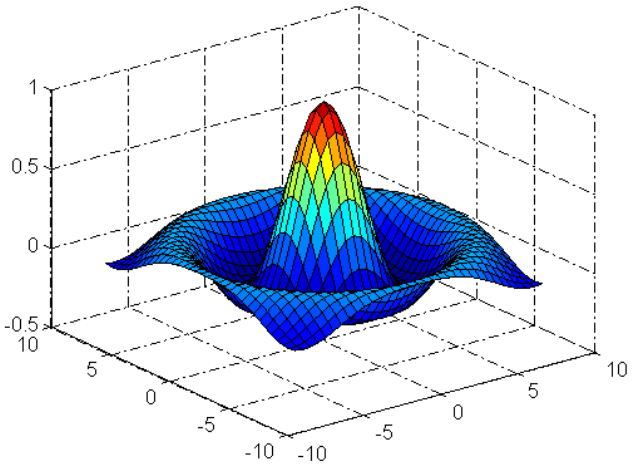
Ví dụ 1: Vẽ đồ thị hàm z=sinr/r với $r = \sqrt{x^2 + y^2}$

```
[X,Y]= meshgrid(-8: 0.5:8);
% Tạo lưới dữ liệu X,Y. meshgrid tạo nên ma trận X gồm các hàng giống nhau,
% ma trận Y gồm các cột giống nhau. Y là ma trận chuyển vị của X.
R=sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R    ; % tính ma trận Z
mesh(X,Y,Z)
```



Đồ thị bề mặt của cùng một ma trận Z trông giống như đồ thị lưới trước đó, chỉ khác là không gian giữa các đường lưới đã được điền đầy bằng màu. Đồ thị loại này được vẽ bằng hàm **surf**, nó có tất cả các đối số như hàm mesh.

>> surf(X,Y,Z)



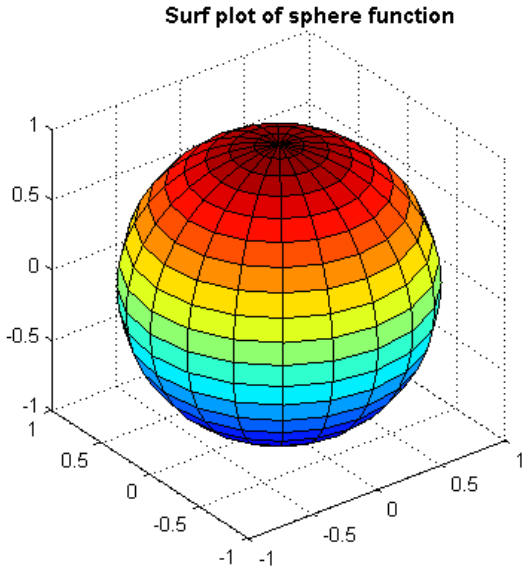
Trong MATLAB có một số hàm dữ liệu 3D được xây dựng sẵn như hàm **sphere** (hình cầu), hàm **cylinder** (hình trụ), hàm ellipsoid (hình ellip), hàm **peaks**,…Để biết chi tiết về các hàm này bạn gõ **help tênhàm**.

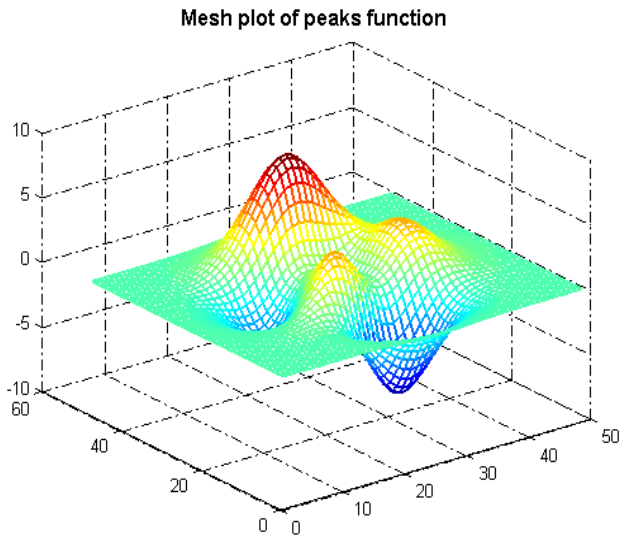
Ví dụ 2:

```
[X,Y,Z]= sphere(20);    % tạo lưới dữ liệu của hình cầu ( 3 ma trận X,Y,Z cấp 21)
surf(X,Y,Z)             % vẽ đồ thị bề mặt quả cầu
axis equal
title('Surf plot of sphere function' )   % tạo tiêu đề đồ thị
```

Ví dụ 3:

```
mesh (peaks)
title('Mesh plot of peaks function' )
% Đồ thị của hàm peaks có đường viền, thể hiện rõ độ nâng hoặc độ cao của hình.
```





Thao tác với đồ thị:

Các thao tác với đồ thị có thể điều khiển bằng chuột thông qua menu và toolbar trong cửa sổ figure hoặc bằng cách nhập lệnh. Một số lệnh thao tác thường dùng là:

- Hàm **view** cho phép khai báo góc để từ đó quan sát được đồ thị trong không gian ba chiều. Hàm view thường dùng ở dạng view(AZ,EL) hoặc view([X,Y,Z]). Để tìm hiểu chi tiết, bạn gõ **help view.**

  view(3) là góc quan sát 3D mặc định, tương đương với AZ=-37.5 và EL=30.

  view(2) là góc quan sát 2D mặc định, tương đương với AZ=0 và EL=90.
- Lệnh **rotate3d on** cho phép điều khiển góc quan sát bằng chuột, **rotate3d off** không cho phép.
- Lệnh **hidden** dấu các nét khuất, **hidden off** có tác dụng ngược lại.
- Lệnh **box on** để tạo khung bao cho hệ trục 3D.

Đặc điểm một số hàm vẽ 3D khác:

- Hàm **ribbon (x,y)** tương tự như hàm **plot(x,y)** ngoại trừ cột của y được vẽ như một dải riêng biệt trong không gian 3D.
- Hàm **contourf** sẽ vẽ một đồ thị đường viền kín, không gian giữa các đường viền được lấp đầy bằng màu.
- Hàm **clabel** tăng thêm độ cao cho đồ thị đường viền.
- **Hàm fill3** vẽ một đa giác đều trong không gian 3 chiều. Cú pháp tổng quát của nó là fill3(x,y,z,c), trong đó chiều đứng của đa giác được xác định bởi 3 thành phần x, y, z. Nhiều đa giác có thể tạo ra bằng cách cho thêm nhiều đối số như **fill3(x1,y1,z1,c1,x2,y2,z2,c2,...).**
- Hàm **bar3** và **bar3h** là phiên bản 3D của bar và barh.
- Hàm **pie3** là phiên bản 3D của pie
- Hàm **ezmesh, ezmeshc, ezsurf, ezsurfc, ezcontour, ezcontourf, ezpolar** vẽ các dạng đồ thị 3D cho các hàm số khai báo bằng biểu thức chữ.

Ví dụ 4: Vẽ đồ thị 3D và hiệu chỉnh màu sắc, góc quan sát, tỉ lệ đồ thị,...

Bước 1: Nhập hàm cần vẽ, ví dụ :

```
>>Z= peaks(20);
```

Bước 2: Mở cửa sổ đồ hoạ figure

```
>>figure(1)
```

Bước 3: Vẽ đồ thị với handle h

```
>>h=surf(Z)
```

Bước 4: Thiết lập màu sắc, độ sáng

```
>>colormap hot                % chọn bảng màu
>>shading interp              % kiểu đổ bóng
>>set(h, 'EdgeColor','k')     % chọn màu các mắt lưới là k=black
>>light('Position',[-2,2,20]) % vị trí nguồn sáng
>>set(h,'FaceColor',[0.7 0.7 0], 'backFaceLighting','lit')
```

Bước 5: Chọn góc quan sát

```
>>view([40,30])
>>view(3)
```

Bước 6: Hiệu chỉnh hệ trục tọa độ

```
>> axis([5 15 5 15 -8  8])
>>set(gca,'ZTickLabel', ' NegativePosition')
```

Bước 7: Chọn tỉ lệ

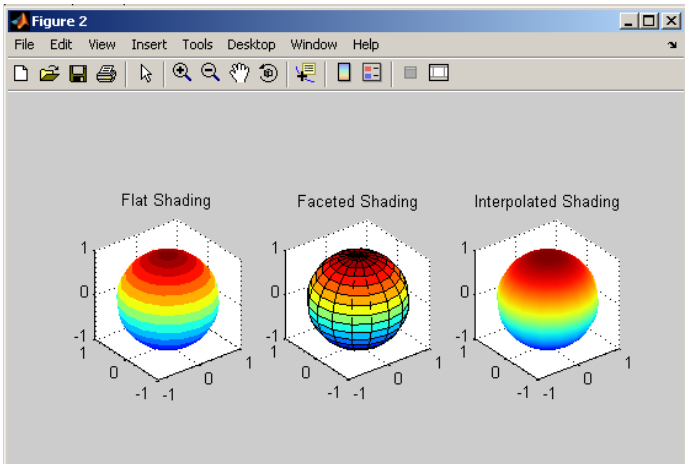
```
>>set(gca,'PlotBoxAspectRatio', [2.5 2.5 1])
```

Bước 8: Tạo nhãn

```
>>xlabel('X Axis')
>>ylabel('Y Axis')
>>zlabel('Z Funcyion Value')
>>title('Peaks')
```

Ví dụ 5: Vẽ hình cầu và so sánh các kiểu đổ bóng (Shading)

```
figure(2)
subplot(1,3,1)
sphere(16)
axis square
shading flat
title('Flat Shading')
%----------------------
subplot(1,3,2)
sphere(16)
axis square
shading faceted
title('Faceted Shading')
%----------------------
subplot(1,3,3)
sphere(16)
axis square
shading interp
title('Interpolated Shading')
```



CHƯƠNG 6

BIỂU THỨC CHỮ

6.1 KHÁI NIỆM BIỂU THỨC CHỮ (SYMBOLIC EXPRESSION)

Ví dụ: Cho hàm số

$$y = x^2 + 12x + 4$$

Tính đạo hàm $\frac{dy}{dx}$, tính giới hạn của y khi $x \to 0$

Giải:

$$\frac{dy}{dx} = 2x + 12$$

$$\lim_{x \to 0}(x^2 + 12x + 4) = 4$$

Hàm y được cho ở dạng trên gọi là **hàm chữ**, vế phải của y được gọi là **biểu thức chữ**, các phép tính đạo hàm hay giới hạn như trên là những phép tính trên biểu thức chữ.

6.2 TÍNH TOÁN TRÊN BIỂU THỨC CHỮ

Nguyên tắc chung trong tính toán biểu thức chữ qua 3 bước sau:

B1- Khai báo biến chữ.

B2- Nhập hàm chữ

B3- Dùng các hàm chuẩn để xử lý các hàm chữ.

Khai báo biến chữ :

Có hai cách khai báo biến :

+ Dùng hàm **syms** để khai báo **một hoặc nhiều** biến chữ cùng lúc :

```
syms  biến_1  biến_2  . . .  biến_n
```

+ Dùng hàm **sym** để khai báo từng biến :

```
tênbiến =sym('tênbiến')
```

Nhập hàm chữ :

Có hai cách nhập:

+ Cách 1: nhập sau các khai báo syms hoặc sym.

+ Cách 2: nhập đồng thời với khai báo sym:

```
>>  tênhàm = sym('biểu thức chữ')
```

Ví dụ, nhập hàm $y = x^2 + 12x + 4$

Cách 1:

```
>>syms x        % hoặc >> x = sym('x')
>>y = x^2+12*x+4
```

Cách 2:

```
>>y = sym('x^2+12*x+4')
```

Các hàm xử lý hàm chữ : chứa trong thư mục **toolbox\symbolic**

Một số hàm giải tích thường dùng:

TÊN HÀM	CHỨC NĂNG
diff	Tính đạo hàm
int	Tính tích phân
limit	Tính giới hạn
dsolve	Giải [hệ] phương trình vi phân
solve	Giải [hệ] phương trình dạng đa thức
numden	Xác định tử số và mẫu số của hàm hữu tỷ
poly2sym	Tạo đa thức từ véctơ hàng chứa các hệ số
sym2poly	Tìm véctơ hàng chứa các hệ số của đa thức
symsum(f,a,b)	Tính tổng của hàm f đi từ a đến b
finverse	Tìm hàm ngược
taylor	Khai triển chuỗi Taylor
eval	Xử lý biểu thức chữ như một câu lệnh MATLAB
ezplot	Vẽ đồ thị một biểu thức chữ và điền nhãn, tiêu đề

Các phép biến đổi :

TÊN HÀM	CHỨC NĂNG
laplace	Biến đổi Laplace
ilaplace	Biến đổi Laplace ngược
fourier	Biến đổi Fourier
ifourier	Biến đổi Fourier ngược
ztrans	Biến đổi Z
iztrans	Biến đổi Z ngược

Định dạng và đơn giản hoá các biểu thức :

TÊN HÀM	CHỨC NĂNG
collect	Rút gọn biểu thức, gom các số hạng theo nhóm
expand	Khai triển biểu thức
factor	Đặt thừa số chung, Đưa đa thức về dạng tích các thừa số
pretty	Hiển thị biểu thức theo cách viết trong toán học
simple	Tối giản hoá biểu thức
simplify	Đơn giản biểu thức

Ví dụ 1: Cho hàm $y = x^2+2x+3$

a/ Tìm đạo hàm bậc 1 và bậc 2 của y

b/ Tìm giới hạn của tỉ số y/x khi $x \to \infty$

c/ Tính tích phân hàm y trong đoạn [1 , 3]

d/ Tìm nghiệm của phương trình y = 0

Giải:

```
>> syms x ; y=x^2+2*x+3 ;              % hoặc >> y=sym('x^2+2*x+3');
>> Dy=diff(y)          % đạo hàm bậc nhất
Dy = 2*x+2
>> D2y=diff(y, 2)      % đạo hàm bậc hai
D2y = 2
>> b= limit(y/x,x,inf,'left')     % tìm giới hạn trái
 b = inf
>> c=int(y,1,3)        % tích phân xác định
 c = 68/3
>> r = solve(y)         %  tìm nghiệm
 r =
    [ -1+i*2^(1/2)]
    [ -1-i*2^(1/2)]
```

Chú ý: Cũng có thể tìm nghiệm của phương trình trên bằng nhiều cách khác. Ví dụ dùng lệnh:

```
>> r=solve('x^2+2*x+3=0')
```

hay

```
>> r= roots([1 2 3])
```

Ví dụ 2: Cho hàm thời gian g(t) = sin(3t), tìm biến đổi Laplace G(s).

```
>> syms t
>> g=sin(3*t)
 g = sin(3*t)
 >> G=laplace(g)
G = 3/(s^2+9)
```

Ví dụ 3: Tìm ảnh Laplace G(s) của hàm $g(t)=\cos^2 t$. Tìm các hệ số (véctơ hàng) của đa thức tử số và mẫu số của G(s) . Tính giá trị hàm G(s) tại $s_1 = 1$; $s_2 = -5j$.

```
>> syms t
>> g=(cos(t))^2 ;
 >> G=laplace(g)              % tìm ảnh Laplace G(s)
 G = 2/s/(s^2+4)*(1+1/2*s^2)
>> G=simplify(G)              % rút gọn biểu thức
G= (2+s^2)/s/(s^2+4)
>> [n,d]=numden(G)            % xác định đa thức tử số và mẫu số ở dạng symbolic
 n = 2+s^2
```

```
 d = s*(s^2+4)
>> p=sym2poly(n)          % xác định đa thức tử số ở dạng vectơ
p =    1    0    2
>> q=sym2poly(d)          % xác định đa thức mẫu số ở dạng vectơ
q =    1    0    4    0
>> s=1; G1= eval(G)       %  >> G1= polyval(p,1)/polyval(q,1)
G1 =   0.6000
>> s=-5j ; G2= eval(G)    %  >>G2= polyval(p,-5j)/polyval(q,-5j)
G2 =   0 + 0.2190i
```

Ví dụ 4: Tìm hàm thời gian y(t) khi biết ảnh Laplace Y(s) :

$$Y(s) = \frac{4}{s(s+3)(s+4)}$$

Giải:

```
>> syms s                          % khai báo s là biến symbolic
>> Y=4/(s*(s+3)*(s+4)) ;           % nhập biểu thức của Y(s)
 >> y=ilaplace(Y)                 % biến đổi Laplace ngược
 y =
    -4/3*exp(-3*t)+exp(-4*t)+1/3   %  y = 1/3 - 4/3 e^-3t + e^-4t
```

Ví dụ 5: Tìm nghiệm của phương trình vi phân :

(a) $\dot{y} + 2y = 0$

(b) $\dot{y} + 2y = t$ với điều kiện đầu: $y(0) = 1$

Giải: Dùng lệnh **dsolve** với cú pháp:

nghiệm = dsolve ('phương trình')

nghiệm = dsolve ('phương trình', ' Điều kiện 1', 'Điều kiện 2',...)

Lưu ý: Khi nhập phương trình ta phải dùng ký hiệu **Dy** để biểu diễn đạo hàm bậc nhất dy/dt , dùng ký hiệu **D2y** để biểu diễn đạo hàm bậc hai d^2y / dt^2, dùng ký hiệu **D3y** để biểu diễn đạo hàm bậc ba d^3y / dt^3 , ...

a/ Tìm nghiệm của phương trình thuần nhất:

```
>> y =dsolve('Dy + 2*y = 0')      %  hoặc  >> f='Dy+2*y=0' ;  y=dsolve(f)
y =
   C1*exp(-2*t)
```

b/ tìm nghiệm phương trình không thuần nhất với điều kiện đầu: y(0) = 1

```
>> y =dsolve('Dy+2*y = t ' , ' y(0)=1' )
y =
   1/2*t-1/4+5/4*exp(-2*t)          %  y = 1/2 t - 1/4 + 5/4 e^-2t
>> pretty(x)
```

$$1/2\ t - 1/4 + 5/4\ \frac{1}{\exp(t)^2}$$

Ví dụ 6 : Giải phương trình bậc hai ở dạng biểu thức chữ :

```
>> r=solve('a*x^2+b*x+c=0')
 r =
    1/2/a*(-b+(b^2-4*a*c)^(1/2))
    1/2/a*(-b-(b^2-4*a*c)^(1/2))
```

Ví dụ 7 : Giải hệ phương trình tuyến tính sau đây ở dạng biểu thức chữ:

$$\begin{cases} 2x_1 + 2x_2 - 3x_3 + x_4 = 4 \\ 4x_1 + 3x_2 - x_3 + 2x_4 = 6 \\ 8x_1 + 5x_2 - 3x_3 + 4x_4 = 12 \\ 3x_1 + 3x_2 - 2x_3 + 2x_4 = 6 \end{cases}$$

Giải : Dùng lệnh solve. Cú pháp tổng quát:

solve('eqn1','eqn2',...,'eqnN')

solve('eqn1','eqn2',...,'eqnN','var1,var2,...,varN')

solve('eqn1','eqn2',...,'eqnN','var1','var2',...'varN')

trong đó eqn1, eqn2,... là các phương trình được nhập ở dạng chuỗi hoặc dạng symbolic.

var1, var2,... là các ẩn số (nghiệm) cần xác định.

```
>>y1=sym('2*x1+2*x2-3*x3+x4-4') ;
>>y2=sym('4*x1+3*x2-x3+2*x4-6') ;
>>y3=sym('8*x1+5*x2-3*x3+4*x4-12') ;
>>y4=sym('3*x1+3*x2-2*x3+2*x4-6') ;
[x1,x2,x3,x4]=solve(y1,y2,y3,y4,'x1,x2,x3,x4')
Kết quả:
x1 = 1/3
x2 = 1/3
x3 = -1/3
x4 = 5/3
```

So sánh với cách giải bằng ma trận số :

```
>>A=[2 2 -3 1; 4  3 -1 2; 8 5 -3 4; 3 3 –2 2] ;   % tạo ma trận
>>b=[4;6;12; 6];
>>x=inv(A)*b
x =       0.3333
```

0.3333
-0.3333
1.6667

Ta thấy hai cách giải có kết quả tương tự nhưng cách giải dùng biến symbolic không biểu diễn nghiệm dưới dạng số thập phân mà mặc định là dạng phân số.

Ví dụ 8: Tìm tổng của n số chẵn đầu tiên và tính giá trị của tổng khi n=20 :

$$\sum_{k=1}^{n} 2k = ? \; ; \qquad \sum_{k=1}^{20} 2k = ?$$

```
>> y= symsum (sym('2*k'),1,'n')        % hoặc   >>syms k n ; y= symsum (2*k,1,n)
```

Kết quả: y = (n+1)^2-n-1

```
>> y=factor(y)    % rút gọn kết quả bằng cách đặt thừa số chung
```

Kết quả: y = n*(n+1)

```
>> y2= symsum (sym('2*k'),1,20)        %  hoặc  >> n=20; y2 = eval(y)
```

Kết quả: y2=420

Nhận xét : Bài toán cũng có thể giải bằng chương trình vòng lặp ở ví dụ 3 trang 33 và ví dụ tính tổng bằng hàm sum trên trang 19.

Ví dụ 9: Tìm tổng hữu hạn :

$$\sum_{k=1}^{n} (2n-1)^2 = \frac{n(2n-1)(2n+1)}{3}$$

```
>> y= symsum (sym('(2*n-1)^2'),1,'n')
y =
   11/3*n+8/3-4*(n+1)^2+4/3*(n+1)^3
>> y=factor(y)
y =
   1/3*n*(2*n-1)*(2*n+1)
>>pretty(y)
    1/3 n (2 n - 1) (2 n + 1)
```

Ví dụ 10: Tìm hàm ngược của hàm f(x) :

```
>>y= finverse (sym('exp(x)'))          % =>  y =  log(x)
>>syms a x;  y= finverse(a^x)          % =>  y= log(x)/log(a)
>>y= finverse(sym('sin(x)'))           % =>  y= asin(x)
>>y= finverse(sym('sqrt(x)'))          % =>  y= x^2
>>y= finverse(sym('1/tan(x)'))         % =>  y= atan(1/x)
```

CHƯƠNG 7

ỨNG DỤNG MATLAB TRONG ĐIỀU KHIỂN TỰ ĐỘNG

7.1 MÔ TẢ PHẦN TỬ VÀ HỆ THỐNG TUYẾN TÍNH

7.1.1 Mô tả phần tử liên tục

Các phần tử và hệ thống tuyến tính bất biến (LTI – linear time-invariant System) có thể mô tả trong Matlab bằng các lệnh : tf, zpk,ss.

1) Phần tử được mô tả toán bằng mô hình hàm truyền đạt :

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + ... + b_0}{a_n s^n + a_{n-1} s^{n-1} + ... + a_0}$$

có thể mô tả trong Matlab bằng 2 cách :

Cách 1 : Dùng lệnh **tf** với cú pháp :

SYS = tf(NUM,DEN)

Trong đó,
- SYS là tên của phần tử hay hệ thống.
- NUM =[b_m b_{m-1} … b_o] % đa thức tử số
- DEN = [a_n a_{n-1} … a_o] % đa thức mẫu số

Lưu ý: Khi sử dụng, các chữ in hoa trong cú pháp lệnh có thể đổi tên tuỳ ý.

Ví dụ 7-1:

```
>> sys1= tf(5,[1 6 8])      %  do tử số là bo=5 nên có thể nhập [5] hay 5 đều được.
Transfer function:
          5
    -------------
    s^2 + 6 s + 8
```

Cách 2 : Dùng lệnh **s = tf('s')** để khai báo mô hình hàm truyền và biến s, sau đó nhập biểu thức toán của hàm truyền.

Ví dụ 7-2:

```
>> s = tf('s') ;  sys2 = 5*(s+1)/((s+4)*(s+3)^2)
Transfer function:
              5 s + 5
    --------------------------
    s^3 + 10 s^2 + 33 s + 36

>> Kp=5; Ki=0.1; Kd= 3;
>> s = tf('s') ;  Gpid=Kp+Ki/s+Kd*s
Transfer function:
          3 s^2 + 5 s + 0.1
          -----------------
                  s
```

2) Phần tử được mô tả toán bằng mô hình zero-cực :

$$G(s) = K\frac{(s-z_1)(s-z_2)...(s-z_m)}{(s-p_1)(s-p_2)...(s-p_n)}$$

có thể mô tả trong Matlab bằng lệnh **zpk** với cú pháp :

SYS = zpk(Z,P,K)

Trong đó:

$\begin{cases} Z = [z_1 \;\; z_2 \;\ldots\; z_m] & \% \text{ véctơ các zero (nghiệm của tử số)} \\ P = [p_1 \;\; p_2 \;\ldots\; p_n] & \% \text{ véctơ các cực (pole, nghiệm của mẫu số)} \\ K = b_m / a_n & \% \text{ hệ số khuếch đại (gain, độ lợi)} \end{cases}$

Nếu tử số hàm truyền không có nghiệm thì lấy Z=[ ] (ma trận rỗng)

Ví dụ 7-3:

```
>> sys3 = zpk ( [ ], [-2 -4] , 5 )
        Zero/pole/gain:
                5
            -----------
            (s+2) (s+4)
```

Nếu biết hàm truyền đạt, ta có thể tìm các zero và cực như sau:

z= zero(SYS) % Tìm vectơ z chứa các zero của hệ SYS

[z,K]= zero(SYS) % Tìm vectơ z và hệ số khuếch đại K của hệ SYS

p= pole(SYS) % Tìm vectơ p chứa các cực của hệ SYS

Ví dụ 7-4:

```
>>z= zero(sys2)      % sys2 đã mô tả ở ví dụ 7-2
z=-1
>>p=pole(sys2)
p=
    -4
    -2
    -2
```

3) Phần tử được mô tả toán bằng mô hình trạng thái :

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

Trong đó: A,B,C,D là các ma trận trạng thái.
u là tín hiệu vào, y là tín hiệu ra, x là biến trạng thái

có thể mô tả trong Matlab bằng lệnh **ss** với cú pháp:

SYS = ss (A,B,C,D)

Ví dụ 7-5:

```
>> A=[-2 -4;2 0] ; B=[1;0] ; C=[0.5 1] ; D=0;
>> sys4 = ss (A,B,C,D)
```

```
a =
          x1      x2
   x1     -2      -4
   x2      2       0
b =
          u
   x1     1
   x2     0
c =
          x1      x2
   y     0.5      1
d =
          u
   y      0
```

Continuous-time model.

4) Phần tử có trễ : Cũng được mô tả bằng các lệnh tf, zpk, ss nhưng có thêm tham số 'inputdelay' hoặc 'outputdelay' để khai báo thời gian trễ.

Ví dụ7.6: Mô tả phần tử trễ có hàm truyền $G(s) = e^{-0,2s} * 4/(s+50)$

```
>> G_delay = tf(4,[1 50],'inputdelay',0.2)

Transfer function:
                4
exp(-0.2*s) * ---------
              s + 50
```

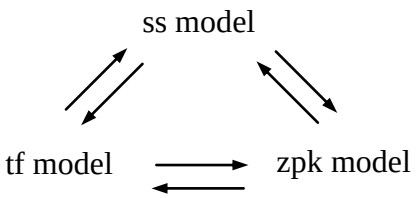
5) Chuyển đổi giữa các dạng mô hình

Các dạng mô hình có thể chuyển đổi qua lại bằng các lệnh ss, tf, zpk :

SYSS= ss(SYS); SYST=tf(SYS); SYSZ=zpk(SYS)

trong đó:

SYS : mô hình bất kỳ
SYSS : mô hình ss
SYST : mô hình tf
SYSZ : mô hình zpk



Ví dụ 7-7 :

```
>> G3 = tf(sys3)     % chuyển sys3 có mô hình zpk ở ví dụ 7-3 về dạng hàm truyền

Transfer function:
          5
---------------
s^2 + 6 s + 8
```

```
>> G4 = tf(sys4)     % chuyển sys4 có mô hình ss ở ví dụ 7-5 về dạng hàm truyền

Transfer function:
        0.5s+2
---------------
s^2 + 2s + 8
```

7.1.2 Mô tả phần tử rời rạc

- Các hàm mô tả phần tử tuyến tính rời rạc có dạng tương tự như khi mô tả phần tử liên tục nhưng có thêm thông tin về thời gian lấy mẫu T (T>0). Nếu chưa xác định thời gian lấy mẫu thì đặt T= -1.

  SYS = tf(NUM,DEN,T)

  SYS = zpk(Z,P,K,T)

  SYS = ss (A,B,C,D,T)

Ví dụ 7-8:

```
>> sys= tf([1 4],[1 2 8],0.1)      % hay  >> z= tf('z',0.1);  sys= (z+4)/ (z^2 + 2* z + 8)
    Transfer function:
            z+4
      ------------------
      z^2 + 2 z + 8
    Sampling time: 0.1
>> sys= tf(4,[1 3 5],-1)
    Transfer function:
            4
      ------------------
      z^2 + 3 z + 5
    Sampling time: unspecified
```

- Có thể chuyển mô hình hệ liên tục thành mô hình hệ rời rạc bằng lệnh **c2d**:

  SYSD=c2d(SYSC,T)

  trong đó: **SYSC** là mô hình hệ liên tục (continuous system)
  SYSD là mô hình hệ rời rạc (discrete system)
  T là thời gian lấy mẫu

Hoặc chuyển mô hình hệ rời rạc thành mô hình hệ liên tục bằng lệnh **d2c** :

SYSC=d2c(SYSD)

Ví dụ 7-9:

```
>> Gs= tf([1 3],[1 2 8]);    % mô tả hệ liên tục có hàm truyền đạt Gs
>> Gz=c2d(Gs,0.1)            % chuyển Gs thành hệ rời rạc có hàm truyền đạt Gz
    Transfer function:
      0.1034 z - 0.07638
      ----------------------
    z^2 - 1.747 z + 0.8187
    Sampling time: 0.1
>> Gs= d2c(Gz)               % chuyển trở lại hệ liên tục
    Transfer function:
        s + 3
      -------------
      s^2 + 2 s + 8
```

7.2 KẾT NỐI CÁC PHẦN TỬ

Các dạng mô hình có thể chuyển đổi và kết nối lẫn nhau, tuy nhiên thứ tự ưu tiên trong Matlab lần lượt là: ss model > tf model > zpk model . Nghĩa là: khi kết nối một mô hình ss với mô hình tf hoặc zpk thì kết quả cuối cùng sẽ được Matlab biểu diễn ở dạng ss; Tương tự, khi kết nối một mô hình tf với mô hình zpk ta nhận được kết quả ở dạng tf .

1) Ghép nối tiếp 2 phần tử : Dùng lệnh **series** hoặc toán tử "*"

SYS = series (SYS1,SYS2)

⇔ **SYS = SYS1* SYS2**

Lệnh **series** chỉ tính được hàm truyền tương đương của 2 phần tử nối tiếp, còn toán tử "*" có thể áp dụng cho số phần tử nối tiếp bất kỳ (2,3,4,…).

Ví dụ: Tìm hàm truyền của ba phần tử nối tiếp

```
>> G1= tf(1,[1 4]) ; G2= tf(1,[1 0]); G3 = tf(1,[1 3]) ;
>>G12= series (G1,G2) ; G = series (G12,G3)
% hoặc  >> G = series(series (G1,G2),G3)
% hoặc  >> G = G1*G2*G3
```

Kết quả :

```
Transfer function:
          1
  ------------------
s^3 + 7 s^2 + 12 s
```

2) Tối giản hóa hàm truyền :

Lệnh **minreal** có tác dụng làm tối giản hóa hàm truyền của hệ thống bằng cách loại bỏ bớt các cặp zero/cực giống nhau. Với hệ có mô hình trạng thái, lệnh **minreal** sẽ loại bỏ các biến không điều khiển được hoặc không quan sát được.

Cú pháp : **SYS=minreal(SYS)**

Ví dụ:

```
>> G1= tf([1 2],[1 4]) ;   G2= tf(2,[1 2]);
>> G12= series (G1,G2)
Transfer function:
          2 s + 4
  -------------
s^2 + 6 s + 8
>>G12= minreal(G12)
Transfer function:
          2
  -----
s + 4
```

3) Ghép song song 2 phần tử : Dùng lệnh **parallel** hoặc toán tử "**+**"

SYS = parallel (SYS1,SYS2)

⇔ **SYS = SYS1 + SYS2**

Lệnh **parallel** chỉ tính được hàm truyền tương đương của 2 phần tử ghép song song, còn toán tử "+" có thể áp dụng cho số phần tử song song bất kỳ (2,3,4,…).

Ví dụ: >> G1= tf(1,[1 4]) ; G2= tf(1,[1 0]); G3 = tf(1,[1 3]) ;
>>G12= parallel (G1,G2) ; G = parallel (G12,G3);

% hoặc >>G = parallel (parallel (G1,G2), G3)

% hoặc >>G = G1 + G2 + G3

Kết quả :

Transfer function:

```
3 s^2 + 14 s + 12
-----------------------
s^3 + 7 s^2 + 12 s
```

4) Tìm mô tả toán của mạch vòng kín : Dùng lệnh **feedback**

- Phản hồi âm:

```
u --->O---->[ SYS1 ]----+---> y
     -|                 |          y = SYSK * u
      +-----[ SYS2 ]<---+
```

SYSK = feedback(SYS1,SYS2)

Với mạch phản hồi âm đơn vị (SYS2=1) thì: **SYSK = feedback(SYS1,1)**

- Phản hồi dương:

SYSK = feedback(SYS1,SYS2,1) % có thêm ký kiệu ",1" sau SYS2

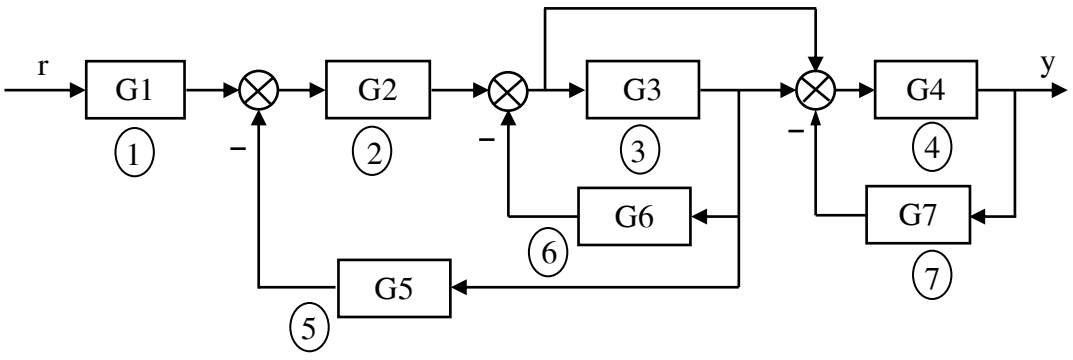
Với mạch phản hồi dương đơn vị (SYS2=1) thì: **SYSK = feedback(SYS1,1,1)**

5) Tìm mô tả toán của hệ thống phức tạp

Cách 1: Biến đổi sơ đồ khối để làm xuất hiện các dạng kết nối đơn giản rồi dùng các lệnh **series** , **parallel**, **feedback** hay các toán tử **"*", "+"** để lần lượt rút gọn sơ đồ khối từ trong ra ngoài.

Cách 2: Dùng các lệnh **append** và **connect**, theo thứ tự các bước như sau:

1) Vẽ sơ đồ khối của hệ thống và đánh số thứ tự các khối (các phần tử, các hệ con).
Ví dụ cần xác định hàm truyền đạt của hệ thống gồm 7 khối sau :



Việc đánh số thứ tự các khối theo quy tắc : từ trái qua phải, từ trên xuống dưới (hoặc từ dưới lên trên).

2) Mô tả các hệ con SYSi trong Matlab bằng cách dùng các lệnh tf, ss, zpk đã nêu ở phần trước.

3) Dùng lệnh **append** để khai báo cho Matlab các hệ con tham gia vào hệ thống:

```
SYSA= append(SYS1,SYS2,SYS3,SYS4,SYS5,SYS6,SYS7);
```

4) Lập ma trận kết nối các hệ con và chỉ định các ngõ vào, ra của hệ thống:

Mỗi hàng của ma trận kết nối Q tương ứng với một hệ con. Số hạng đầu của mỗi hàng là chỉ số của hệ con, các số hạng tiếp theo biểu thị kết nối giữa ngõ vào của hệ con đó với ngõ ra của các hệ con khác. Ví dụ ngõ vào của hệ 2 là ngõ ra của hệ 1 và hệ 5, hệ 5 lại là phản hồi âm, do đó số hạng đầu trong hàng là 2, hai số hạng kế trong hàng là 1 và –5, các số 0 được thêm vào để tạo Q là ma trận chữ nhật.

```
Q= [1  0  0  0  0
    2  1 -5  0  0
    3  2 -6  0  0
    4  2 -6  3 -7
    5  3  0  0  0
    6  3  0  0  0
    7  4  0  0  0 ];
input=1;      % vì ngõ vào của hệ thống là ngõ vào của khối 1
output=4;     % vì ngõ ra của hệ thống là ngõ ra của khối 4
```

5) Dùng lệnh **connect** tìm mô tả toán của toàn hệ thống theo cú pháp:

```
SYS = connect(SYSA,Q,input,output)
```

Cách 3: Vẽ sơ đồ hệ thống trong môi trường SIMULINK của Matlab rồi dùng lệnh **linmod** để trích xuất các ma trận **A,B,C,D** của hệ thống :

```
[A,B,C,D] = linmod ('model_filename');
```

Trong đó tham số 'model_filename' là tên của file mô hình. Ví dụ, sau khi vẽ sơ đồ hệ thống trong SIMULINK ta lưu lại thành file "ht1.mdl" thì dùng lệnh **linmod** như sau :

```
[A,B,C,D] = linmod ('ht1');
```

Sau đó tuỳ nhu cầu có thể tìm mô tả hệ thống dưới dạng mô hình trạng thái:

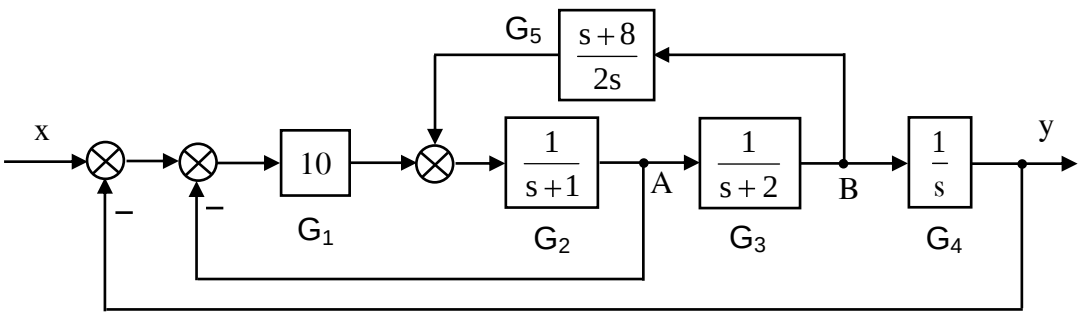
```
SYS = ss (A,B,C,D)
```

Hoặc chuyển về dạng hàm truyền đạt : **SYS = tf(SYS)**

Với hệ thống rời rạc ta dùng lệnh **dlinmod** thay cho lệnh **linmod**

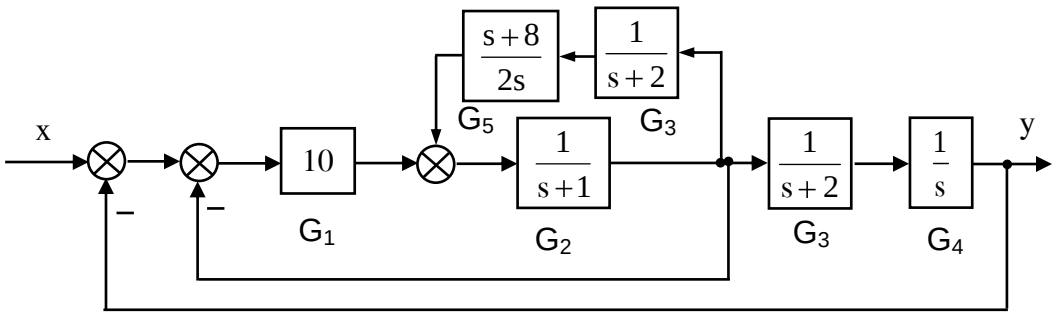
```
[A,B,C,D] = dlinmod ('model_filename',T );
```

Ví dụ: Tìm hàm truyền đạt của hệ thống có sơ đồ khối như hình vẽ :



Giải:

Cách 1: Trước tiên ta cần biến đổi sơ đồ khối về dạng tương đương để xuất hiện các dạng kết nối cơ bản, sau đó mới áp dụng được các hàm kết nối, tương tự như tính toán đại số sơ đồ khối trong lý thuyết điều khiển tự động.



Chương trình Matlab:

```
g1=20;
g2=tf(1,[1 1]);
g3=tf(1,[1 2]);
g4=tf(1,[1 0]);
g5=tf([1 8],[2 0]);                % hoặc >> s=tf('s'); g5=2*(s+1);
gtd1=feedback(g2,g5*g3,1);         % mạch hối tiếp dương
gtd2=feedback(g1*gtd1,1);          % mạch hối tiếp âm đơn vị
gtd2=minreal(gtd2);                % tối giản hoá hàm truyền
SYS=feedback(gtd2*g3*g4,1);        % mạch hối tiếp âm đơn vị
SYS=minreal(SYS)
```

Kết quả :

Transfer function:

```
                10
-----------------------------
s^3 + 13 s^2 + 21.5 s + 6
```

Cách 2: Có thể tính trực tiếp với sơ đồ đã cho, không cần thiết phải biến đổi sơ đồ.

Chương trình Matlab :

```
g1=20;
g2=tf(1,[1 1]);
g3=tf(1,[1 2]);
g4=tf(1,[1 0]);
g5=tf([1 8],[2 0]);
SYSA=append(g1,g2,g3,g4,g5);
```

```
Q =[1 -2 -4      %  ngõ vào của khối 1 là ngõ ra của các khối phản hồi âm  2 và 4
    2  1  5      %  ngõ vào của khối 2 là ngõ ra của các khối  1 và 5
    3  2  0      %  ngõ vào của khối 3 là ngõ ra của khối  2
    4  3  0      %  ngõ vào của khối 4 là ngõ ra của  khối  3
    5  2  0] ;   %  ngõ vào của khối 5 là ngõ ra của  khối  2
input=1;                 %  ngõ vào của hệ thống là ngõ vào của khối 1
output=4;                %  ngõ ra của hệ thống là ngõ ra của khối 4
SYS=connect(SYSA, Q, input,output);
SYS=minreal(SYS)
```

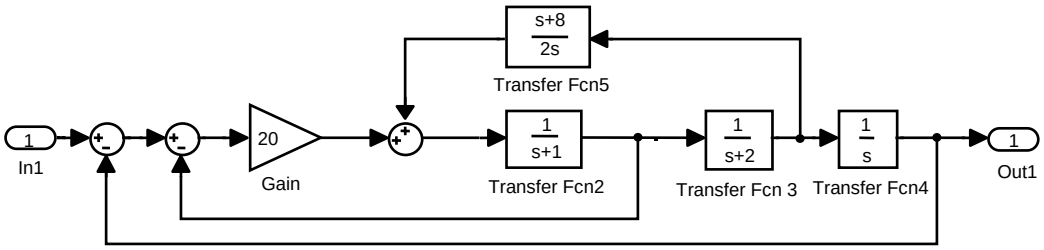
Kết quả :

Transfer function:

$$\frac{10}{s^3 + 13\,s^2 + 21.5\,s + 6}$$

Cách 3 : Thực hiện hai bước :

- Bước1: Vẽ sơ đồ hệ thống trong môi trường SIMULINK của MATLAB, lưu tên file là ‘sodo1.mdl’.



- Bước 2: Nhập và thực thi đoạn chương trình sau :

```
>>[A,B,C,D]=linmod('sodo1');
>>sys=ss(A,B,C,D);
>>sys=tf(sys);
>>sys=minreal(sys)
```

Kết quả :

Transfer function:

$$\frac{10}{s^3 + 13\,s^2 + 21.5\,s + 6}$$

7.3. TÍNH TOÁN VÀ VẼ BIỂU ĐỒ ĐÁP ỨNG THỜI GIAN

7.3.1. Đáp ứng bậc thang

Lệnh **step** dùng để tính toán và vẽ đáp ứng bậc thang (cũng gọi là đáp ứng quá độ hay hàm quá độ, là đáp ứng của hệ thống với tín hiệu vào bậc thang đơn vị 1(t)) .

LỆNH	Ý NGHĨA
step(SYS)	Vẽ biểu đồ đáp ứng bậc thang của hệ thống, khoảng thời gian do Matlab tự động xác định.
step(SYS,T) :	Vẽ biểu đồ đáp ứng bậc thang của hệ thống trong thời gian từ 0 đến T
step(SYS1,SYS2,...,T)	Vẽ đáp ứng của nhiều hệ thống trên cùng 1 hệ trục toạ độ
[y, t] = step(SYS)	Trả về dãy giá trị y tương ứng với vectơ thời gian t

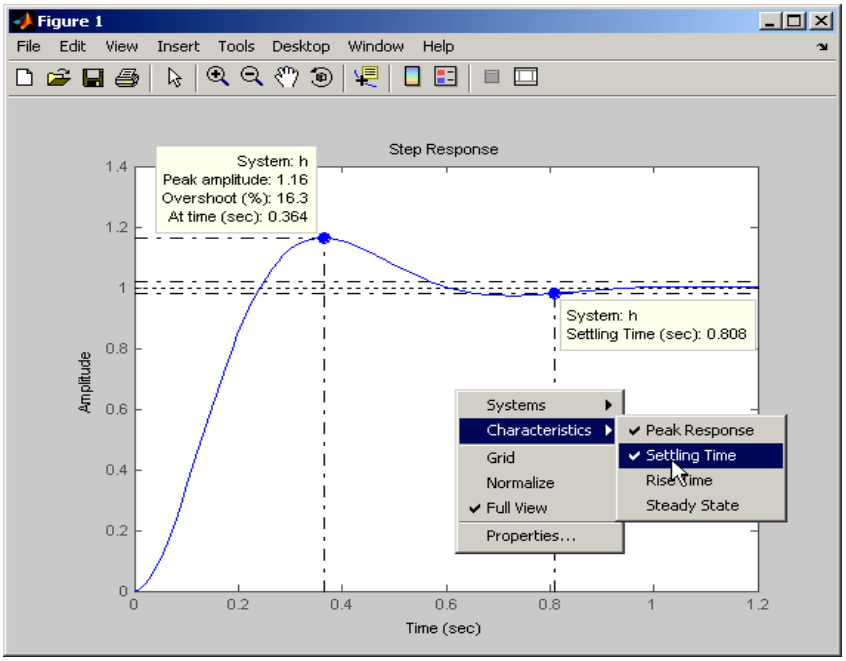
SYS,SYS1,SYS2,... có thể là hệ liên tục hoặc rời rạc có mô hình dạng tf, zpk, hoặc ss bất kỳ như đã trình bày ở phần trước.

Sau khi vẽ biểu đồ ta có thể xác định các thông số quan trọng của đường đáp ứng như thời gian tăng (Risetime, 10 ÷ 90%), thời gian quá độ (Settlingtime, ± 2%), độ vọt lố (Overshoot), giá trị xác lập y(∞)… bằng cách nhắp phải chuột vào vùng trống bất kỳ trên đồ thị để xuất hiện menu danh sách thả xuống (popup-menu) và chọn mục tương ứng.

Hình dưới đây minh hoạ kết quả sau khi nhắp chuột phải vào vùng trống trên đồ thị và lần lượt chọn Characteristics > Settling Time, sau đó nhắp chuột trái vào dấu tròn vừa xuất hiện trên đồ thị để hiển thị giá trị thời gian quá độ.

- Để hiển thị độ vọt lố, chọn Characteristics > Peak Response >...

- Để hiển thị giá trị xác lập y(∞), chọn Characteristics > Steady State >...



Nếu bạn nhắp chuột trái vào điểm bất kỳ trên đường đáp ứng, MATLAB sẽ hiển thị các giá trị hoành độ, tung độ (tương ứng là thời gian và biên độ) tại điểm đó. Nếu bạn nhắp trái rồi rê chuột dọc theo đường đáp ứng, các giá trị trên sẽ hiển thị liên tiếp.

Để tìm đáp ứng bậc thang của hệ thống **dưới dạng hàm số theo thời gian t** (thay vì dạng đồ thị) khi biết hàm truyền đạt G của hệ, ta có thể áp dụng các hàm xử lý biểu thức chữ như sau:

```
[n,d]=tfdata(G,'v');      % tìm véctơ hệ số của tử và mẫu số hàm truyền G.
syms s                    % khai báo biến symbolic (biến chữ) là s
num=poly2sym(n,s);        % tạo biểu thức symbolic của tử số theo biến s
den=poly2sym(d,s);        % tạo biểu thức symbolic của mẫu số theo biến s
G=num/den                 % biểu diễn hàm truyền G dưới dạng biểu thức symbolic
Y=G/s                     % ảnh Laplace Y(s) của hàm quá độ
y=ilaplace(Y)             % hàm quá độ y(t)
```

- **Đáp ứng xung**

Dùng lệnh **impulse** để tính và vẽ đáp ứng xung (hàm trọng lượng) của hệ thống.

Cú pháp của lệnh **impulse** giống như lệnh **step.**

impulse(SYS)

impulse(SYS,T)

impulse(SYS1,SYS2,...,T)

[y,t]=impulse(SYS)

- **Đáp ứng tín hiệu vào bất kỳ**

- Dùng lệnh **lsim** để tìm đáp ứng thời gian của hệ thống đối với tín hiệu vào bất kỳ.

lsim(SYS,u,t) : vẽ đáp ứng với tín hiệu vào u bất kỳ trong khoảng thời gian t.

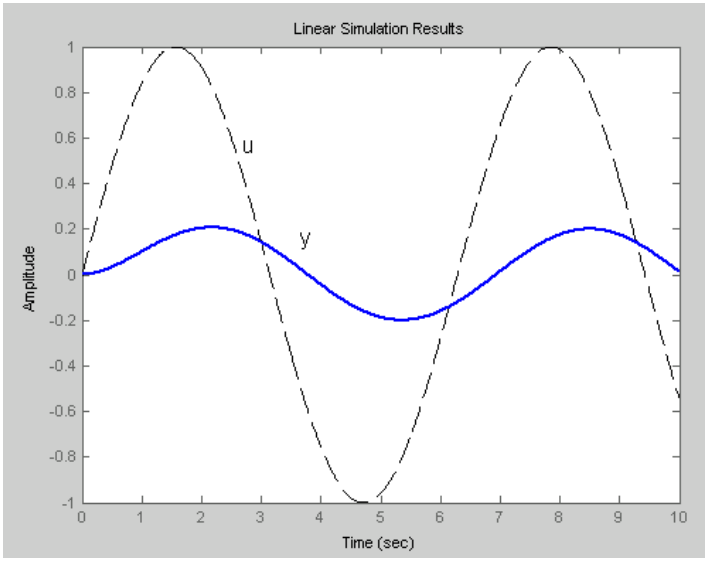
[y,t] = lsim(SYS,u,t) : trả về dãy giá trị đáp ứng y tương ứng với vectơ thời gian t.

- Một số ví dụ tạo tín hiệu vào:

```
>>t= 0:0.01:1;
>>u= ones(size(t)); plot(t,u)        % unit step (bậc thang đơn vị) u=1(t)
>>u=  5*ones(size(t));               % step (bậc thang)  u=5(t)
>>u= [1;zeros(100,1)];               % impulse (xung nhọn đơn vị)
>>u= t ;                             % ramp (hàm dốc)
>>u= t.^2 ;                          % parabol
>>u= square(4*t) ;                   % xung vuông
>>u =sin(t)                          %  sóng sin

>>t=0: 1e-4: 1.5;
>>u=sawtooth(2*pi*50*t)              % sóng răng cưa chu kỳ 2pi, biên độ ±1
```

Ví dụ sau đây minh hoạ đáp ứng của khâu PT_1 đối với tín hiệu vào sin(t) trong 10 giây.

```
>> t = 0:0.01:10;   u = sin(t);
>> sysPT1=tf(1,[3 4]);  lsim(sysPT1,u,t)
```



Có thể tạo nhanh các tín hiệu vào dạng sóng bằng hàm **gensig** :

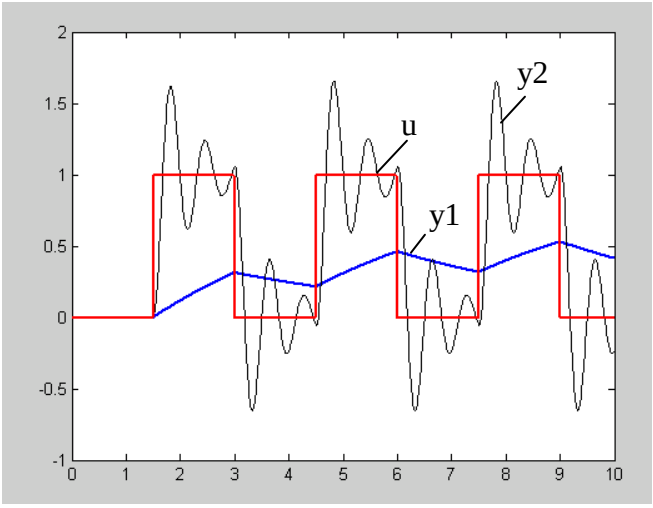
[u,t] = gensig(typ,tau)

[u,t] = gensig (typ,tau,Tf,Ts)

Bằng tham số **typ** ta có thể khai báo loại tín hiệu: sóng sin (**'sin'**), sóng vuông (**'square'**), hoặc dãy xung nhọn (**'pulse'**). Tín hiệu do **gensig** tạo ra có biên độ chuẩn là 1đơn vị. Chu kỳ của tín hiệu được khai báo nhờ tham số **tau**. **Tf** là khoảng thời gian tác động và **Ts** là thời gian lấy mẫu (chu kỳ lấy mẫu). Véctơ thời gian t được Matlab tự động chọn hoặc tính theo Tf và Ts.

Hình sau đây minh hoạ đáp ứng của hai khâu PT1, PT2 khi bị kích thích bởi tín hiệu vào sóng vuông chu kỳ 3s, thời gian tác động 10s, lấy mẫu mỗi 0,01s do **gensig** tạo nên:

```
>> sysPT1=tf(1,[4 1]); sysPT2=tf(100,[1 3 100]);
>> [u,t]=gensig('square',3,10,0.01);  % tạo tín hiệu vào sóng vuông
>> [y1,t]=lsim(sysPT1,u,t);
>> [y2,t]=lsim(sysPT2,u,t);
>> plot(t,y1,t,y2,t,u)
```



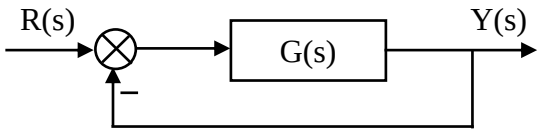
7.4. TÍNH TOÁN VÀ VẼ BIỂU ĐỒ ĐÁP ỨNG TẦN SỐ

7.4.1 Biểu đồ Nyquist

Các cấu trúc thường dùng của hàm **nyquist** :

LỆNH	Ý NGHĨA
nyquist(SYS)	Vẽ biểu đồ Nyquist của hệ SYS
nyquist(SYS1,SYS2,...)	Vẽ biểu đồ Nyquist của nhiều hệ thống trên cùng hệ trục
nyquist([k1;k2;...;kN],[DEN])	Vẽ biểu đồ Nyquist của hệ SYS có hàm truyền dạng G(s)=k/DEN với N giá trị khác nhau của k
nyquist(SYS,w)	Vẽ biểu đồ Nyquist ứng với véctơ tần số w định trước. Nhập vectơ w theo cú pháp **wđầu: gia số : wcuối**
[Re,Im]= nyquist(SYS,w)	Tính phần thực, phần ảo của đáp ứng tần số (= tính giá trị Re và Im tại các điểm nằm trên đường Nyquist).

Ví dụ: Xét hệ thống kín hồi tiếp âm có hàm truyền đạt vòng hở tương ứng là:



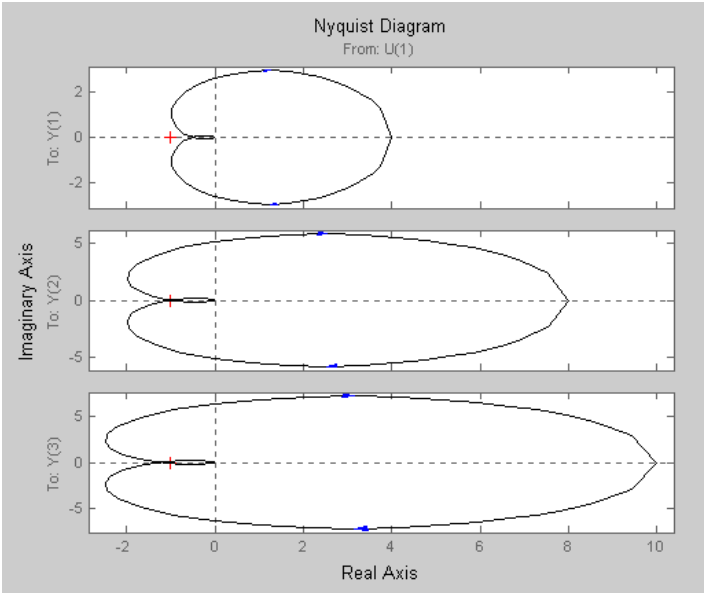
$$G(s) = \frac{K}{s+1^{3}} = \frac{K}{s^3 + 3s^2 + 3s + 1}$$

Từ hàm truyền đạt ta thấy hệ hở ổn định vì các nghiệm của phương trình đặc tính đều là nghiệm thực -1 < 0. Tuy nhiên, theo tiêu chuẩn ổn định Nyquist thì hệ thống kín cũng có thể không ổn định nếu chọn hệ số khuếch đại K quá lớn. Có thể sử dụng hàm **nyquist** để khảo sát ảnh hưởng của tham số K như sau:

Dùng lệnh

```
>> nyquist([4;8;10],[1 3 3 1])
```

ta thu được ba biểu đồ Nyquist như hình dưới đây, tương ứng với K=4 , K=8 , và K=10.

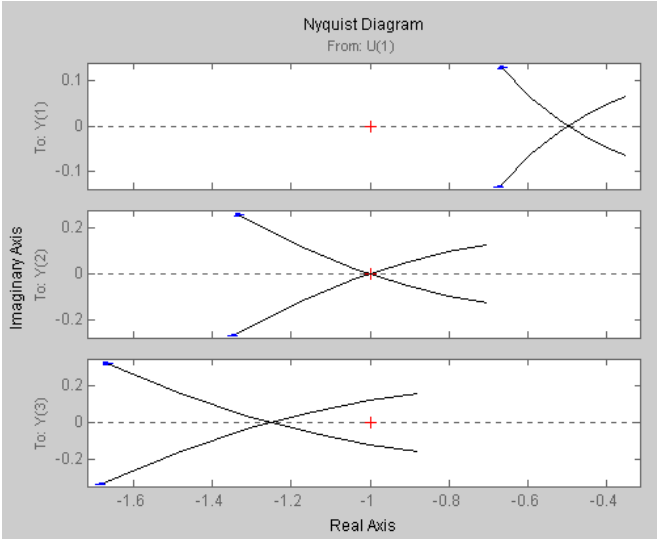


Chọn dải tần số khảo sát từ 1,5 đến 2 rad/s với gia số 0,1 và vẽ lại biểu đồ Nyquist:

```
>> nyquist([4;8;10],[1 3 3 1],1.5:0.1:2)
```

Với dải tần số được chọn thích hợp ta thu được đồ thị phóng to để dễ quan sát những vùng lân cận của điểm tới hạn (-1; j0). Ta thấy:

- Với K< 8 thì đường Nyquist hệ hở không bao điểm (-1; j0) nên hệ kín ổn định,
- Với K=8 thì đường Nyquist hệ hở đi qua điểm (-1; j0) nên hệ kín ở giới hạn ổn định.
- Với K> 8 thì đường Nyquist hệ hở bao điểm (-1; j0) nên hệ kín không ổn định



7.4.1 Biểu đồ Bode

Các chỉ tiêu chất lượng của hệ thống kín có thể xác định từ đặc tính tần số của hệ thống hở, đặc biệt là biểu đồ Bode của hệ hở. Ví dụ: độ dự trữ biên độ, độ dự trữ pha, tần số cắt biên, tần số cắt pha….

LỆNH	Ý NGHĨA
bode(SYS)	Vẽ biểu đồ Bode của hệ SYS
bode(SYS,{WMIN,WMAX})	Vẽ biểu đồ Bode của hệ SYS trong phạm vi tần số chỉ định
bode(SYS1,SYS2,...)	Vẽ biểu đồ Bode của nhiều hệ thống chung trên một hệ trục
[MAG,PHASE] = bode(SYS,W) **[MAG,PHASE,W] = bode(SYS)**	Tính giá trị biên độ và pha của hệ thống. W là vectơ các điểm tần số và nhập theo cú pháp **wđầu: gia số : wcuối**
margin(SYS)	Vẽ biểu đồ Bode và hiển thị giá trị dự trữ biên độ, dự trữ pha,... ngay trên biểu đồ
[Gm,Pm,Wgm,Wpm]= margin(SYS)	Tính toán độ dự trữ biên độ, dự trữ pha, tần số cắt biên, tần số cắt pha

Giá trị dự trữ biên độ Gm do hàm margin tính ra là giá trị tự nhiên (không đơn vị) , để chuyển đổi sang giá trị dB (deciBel) ta dùng công thức **Gm_dB= 20*log10(Gm).**

Với hệ rời rạc , hàm **bode** sẽ áp dụng thuật toán biến đổi $z = e^{j\omega T}$ để ánh xạ vòng tròn đơn vị thành trục tần số thực (trong đó T là thời gian lấy mẫu). Do hàm truyền tần số của hệ rời rạc có tính tuần hoàn với chu kỳ $2\pi/T$ nên với hệ rời rạc, hàm **bode** chỉ tính đáp ứng cho những điểm tần số nhỏ hơn tần số tới hạn π/T.

7.5 Quỹ đạo nghiệm

Quỹ đạo nghiệm (hay biểu đồ nghiệm) là tập hợp các nghiệm của phương trình đặc tính, thể hiện trên mặt phẳng phức, khi độ khuếch đại K biến thiên từ 0 đến $+\infty$.

Phương pháp quỹ đạo nghiệm dùng để khảo sát nghiệm của phương trình đặc tính của hệ kín như là một hàm của hệ số khuếch đại K. Nó không những cho biết hệ thống có ổn định hay không, mà còn cho biết về hệ số giảm chấn hay tỉ số tắt dần dao động.

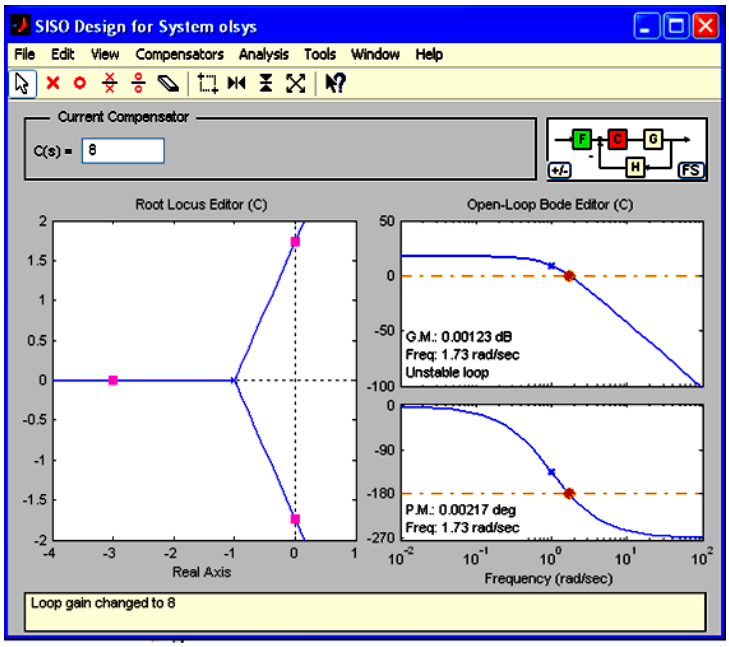
Các lệnh thường dùng để vẽ và khảo sát quỹ đạo nghiệm :

LỆNH	Ý NGHĨA
rlocus(SYS)	Vẽ quỹ đạo nghiệm
root =rlocus(SYS,K)	Tính và xuất dãy giá trị nghiệm ứng với K cho trước.
[root,K]=rlocus (SYS)	Tính và xuất dãy giá trị K và dãy nghiệm tương ứng.
rlocfind	Chọn K bằng cách nhấn chuột trên biểu đồ nghiệm.
sgrid	Tạo các đường lưới của giảm chấn ξ và tần số riêng ω_n
pzmap(SYS)	Tính và vẽ vị trí các cực và zero của hệ thống
[P,Z]=pzmap(SYS)	Tính và xuất các vectơ P, Z chứa giá trị các điểm cực và zero.

Để tạo tiện ích cho người dùng, Matlab hỗ trợ công cụ SISO Design Tool. Đây là công cụ thiết kế hệ SISO theo phương pháp quỹ đạo nghiệm kết hợp với biểu đồ Bode. Áp dụng cho hệ đã nêu ở phần trước ta cũng dễ dàng xác định được trị số K để hệ ở giới hạn ổn định. Đoạn chương trình Matlab chỉ gồm 2 câu lệnh :

```
>>olsys=tf(1,[1 3 3 1]);   % mô tả hệ hở với K=1
>>sisotool(olsys);         % vào cửa sổ SISO Design
```

Tại cửa sổ SISO Design ta có thể nhập giá trị độ khuếch đại K vào ô C(s) hoặc dùng chuột di chuyển các núm vuông trên biểu đồ quỹ đạo nghiệm (tương ứng với việc thay đổi trị số độ khuếch đại k), ta dễ dàng tìm được giá trị K=8 hệ ở biên giới ổn định với độ dự trữ ổn định G.M. $\approx$ 0 dB. Độ khuếch đại càng tăng, độ dự trữ ổn định càng âm, tức là hệ càng mất ổn định.



7.6 Giao diện LTIViewer

Giao diện đồ hoạ LTIViewer được kích hoạt bằng lệnh **ltiview.** Với giao diện LTIViewer bạn có thể cùng một lúc khảo sát đặc tính động học của nhiều hệ thống tuyến tính bất biến, và đối với mỗi hệ thống lại có thể vẽ được tất cả các dạng đặc tính động học. Do có thể vẽ được trên cùng một cửa sổ nên bạn có thể dễ dàng nhận thấy được mối liên hệ giữa các dạng đặc tính động học, ví dụ đáp ứng xung là đạo hàm của đáp ứng bậc thang, đỉnh cộng hưởng trên biểu đồ Bode có biên độ càng cao thì độ vọt lố trên đáp ứng bậc thang càng cao, sự liên hệ giữa biểu đồ Bode và biểu đồ Nyquist,…

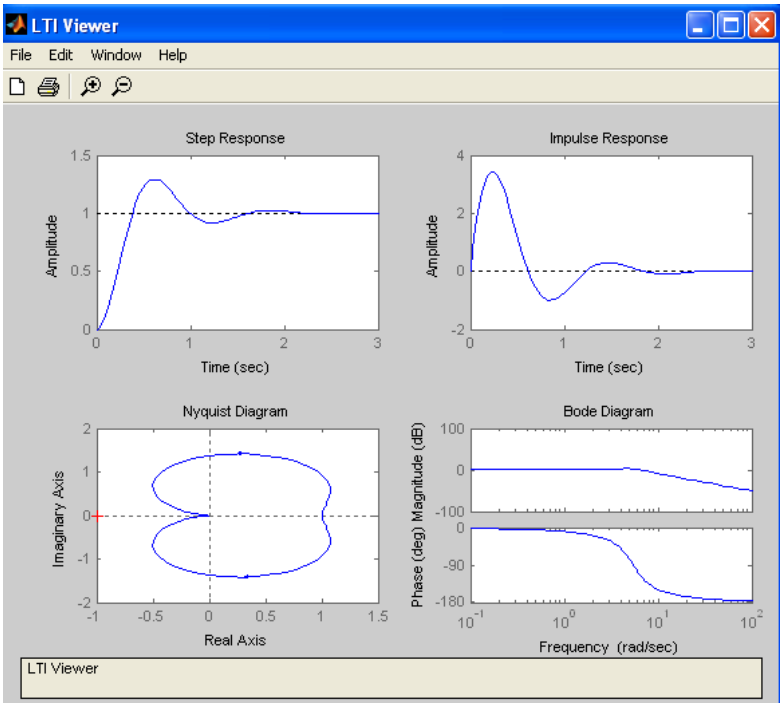
Một số cách thường dùng của lệnh **ltiview** :

LỆNH	Ý NGHĨA
ltiview(PLOTTYPE, SYS)	Vẽ các biểu đồ chỉ định bởi tham số PLOTTYPE . PLOTTYPE có thể là 'step', 'impulse', 'nyquist', bode',… hoặc tổ hợp {'step'; 'impulse'; 'nyquist'; bode',…}
ltiview(SYS1,SYS2,…,SYSN)	Vẽ biểu đồ step của nhiều hệ thống
ltiview(PLOTTYPE,SYS1,SYS2,…,SYSN)	Vẽ các biểu đồ chỉ định bởi PLOTTYPE của nhiều hệ thống

Ví dụ:

```
>> G=tf(10,[1 2 10])

Transfer function:
        10
 --------------
s^2 + 2 s + 10

>> ltiview({'step';'impulse';'nyquist';'bode'},G)
```

Từ menu edit hoặc từ popup-menu khi nhắp chuột phải trong cửa sổ LTIViewer, bạn có thể chọn lựa các thiết đặt về cấu hình như số lượng đồ thị, loại đồ thị, hiển thị hoặc tắt hiển thị các thông số chất lượng trên các đồ thị,…



BÀI TẬP THỰC HÀNH

Chương 1&2

1) Thực hiện lần lượt các bước sau đây:
   - Khởi động phần mềm MATLAB, kiểm tra tên thư mục hiện hành bằng cách quan sát ô Current Directory trên thanh Toolbar hoặc gõ lệnh >>**cd**
   - Tạo một thư mục mới có tên là ***tên của bạn*** (Ví dụ: 'Tran Van Tuan'), vị trí thư mục mới nằm trong thư mục ***work*** của MATLAB.
   - Chuyển thư mục mới tạo trở thành thư mục hiện hành.
   - Kiểm tra lại tên thư mục hiện hành trong ô Current Directory trên thanh Toolbar.
2) Trong Command window, thực hiện lần lượt các yêu cầu sau:
   a. Đặt tên các biến để biểu diễn: chiều dài, chiều rộng, chiều cao và thực hiện phép gán: chiều rộng= 3 (cm); chiều dài=4 (cm); chiều cao= 5 (cm)
   b. Tính diện tích mặt đáy và thể tích hình hộp với các thông số trên.
   c. Mở cửa sổ Workspace hoặc dùng lệnh whos kiểm tra lại các kết quả đã tính. Lưu nội dung của Workspace thành tập tin bai2_data.mat (Lưu ý kiểm tra và tắt các phần mềm gõ tiếng Việt như Vietkey, Unikey,... để tránh lỗi khi nhập tên tập tin).
   d. Quan sát cửa sổ Current Directory để chắc chắn là tập tin bai2_data.mat đã có.
   e. Xoá tất cả các biến trong Workspace hiện hành. Mở lại tập tin bai2_data.mat và kiểm tra lại các biến.
   f. Thay đổi giá trị các biến chiều rộng, chiều dài, chiều cao tuỳ ý. Thực hiện lại các phép tính trên. (Lưu ý: Có thể thay đổi bằng lệnh gán ở cửa sổ Command Window hoặc thay đổi trực tiếp trên cửa sổ Workspace)
3) Thực hiện lại bài 2 bằng SCRIFT FILE với yêu cầu là giá trị chiều rộng, chiều dài, chiều cao được nhập từ bàn phím.
4) Thực hiện lại bài 2 bằng FUNCTION FILE với chiều rộng, chiều dài, chiều cao là các đối số của hàm. Các giá trị hàm trả về là:
   a. Thể tích của hình hộp chữ nhật
   b. Diện tích mặt đáy và thể tích hình hộp chữ nhật
5) Sử dụng SCRIFT FILE và SCRIFT FILE viết các chương trình tính khối lượng của một chi tiết máy hình trụ trơn làm bằng vật liệu thép có khối lượng riêng là 7,8 kg/dm^3.
6) Cho phương trình bậc hai $ax^2 + bx + c = 0$
   a. Thực hiện phép gán từ Command window a=1; b=-8; c=15. Tính nghiệm của phương trình.
   b. Tương tự với a=1 ; b=2; c =13.
   c. Thay đổi tuỳ ý giá trị a, b, c. Tính nghiệm của phương trình.
7) Sử dụng SCRIFT FILE giải phương trình bậc hai với a, b, c được nhập từ bàn phím.
8) Sử dụng FUNCTION FILE giải phương trình bậc hai với a, b, c là các đối số của hàm. Giá trị hàm trả về là hai nghiệm.
9) Viết file hàm tính chu vi và diện tích tam giác khi biết độ dài 3 cạnh.
10) Viết các đoạn chương trình tính thể tích hình lăng trụ đứng, hình tứ diện đều, hình chóp, hình chóp cụt, hình nón, hình nón cụt, hình trụ, hình cầu.

Chương 3

1) Sử dụng các hàm toán học cơ bản có trong tài liệu để thực hiện các phép tính sau :

(a) $\sqrt{625}$; $\sqrt{-9}$; $\sqrt[3]{100}$

(b) e^{0} ; e ; e^{-1} ; $e^{-\infty}$

(c) $(1/2)e^{i5}$

(d) $(3+2i)^{i30^\circ}$

(e) a^{x} với $a=8.10^{-4}$ và $x=3/4$

(f) $0/0$; $1/0$; $1/\infty$

(g) $5^{-2}.2^{10}.\sqrt{3}.\ln 5.\lg 2$

(h) $2\sin(\pi/5).\ \mathrm{arctg}^2(3)$

(i) $\dfrac{2\sqrt{3}}{5}\sin(30^\circ)\cos(45^\circ)$

(j) $2^{\sqrt[3]{\sin(30^\circ)}}.\mathrm{sign}(\sin(5\pi/3).\cos(135^\circ))$

(k) $\dfrac{\cos\sqrt{2}}{4\sqrt{2\sin\sqrt[3]{2}}}$

(l) $\dfrac{2-\left|\sin(-30^\circ)\right|e}{1-\cos 2^{\,3/2}}$

2) Cho biến x được gán giá trị tuỳ ý từ bàn phím (x có thể là một số hoặc vectơ).Viết scrift file tính giá trị của các hàm số sau:

$$f_1(x)=\frac{e^{x}+e^{-x}}{2} \quad ; \quad f_2(x)=\cosh(x)$$

$$g_1(x)=\frac{e^{x}-e^{-x}}{2} \quad ; \quad g_2(x)=\sinh(x)$$

Chạy chương trình. Nhận xét kết quả và giải thích ?

3) Viết chương trình tính giá trị của các biểu thức sau :

$$h(t)=1-\frac{3}{7}e^{-2t}-\frac{4}{7}e^{-9t}$$

$$y(t)=1+e^{-\frac{11}{2}t}\left[\frac{1}{7}\sinh\left(\frac{7}{2}t\right)-\cosh\left(\frac{7}{2}t\right)\right]$$

Chạy chương trình với biến t được gán giá trị tuỳ ý (t có thể là một số hoặc vectơ).

Nhận xét kết quả và giải thích ?

4) Cho hai số phức :

$$z_1=3+2i$$

$$z_2=3e^{i45^\circ}$$

Hãy thực hiện trong MATLAB các phép toán sau đây với hai số phức trên.

(a) z_1+z_2

(b) z_1-z_2

(c) z_1*z_2

(d) z_1/z_2

(e) Tính môđun và góc pha của z_1 .

(f) Nhập z_1 theo dạng môđun-pha. So sánh kết quả trả về với giá trị ban đầu của z_1.

5) Cho hai vectơ u, v như sau:

$$u = [\,2\ \ 4\ \ 6\ \ 8\,] \quad ; \qquad v = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 7 \end{bmatrix}$$

(a) Tìm phần tử lớn nhất và phần tử bé nhất của vectơ u.

(b) Tìm phần tử lớn nhất và phần tử bé nhất trong mọi phần tử của cả hai vectơ.

(c) Tìm trung bình cộng của các phần tử trong vectơ u.

(d) Tính tổng và tích các phần tử của vectơ u.

(e) Tính tích vô hướng và góc hợp bởi hai vectơ u và v.

(f) Viết chương trình cho phép người dùng nhập vào hai vectơ cột u, v bất kỳ và trả về kết quả là tích vô hướng và góc hợp bởi hai vectơ đó.

6) Hãy tạo các vectơ để biểu diễn các tập hợp sau:

a) Tập các số tự nhiên ≤ 200; b) Tập các số nguyên dương ≤ 200;

c) Tập các số chẵn ≤ 200 ; d) Tập các số lẻ < 200;

e) Tập các số ≤ 200 và là bội số của số k, với k được gán giá trị trước, tuỳ ý.

f) Tập các số ≤ 200 và đem chia cho 4 luôn có số dư là 3.

g) Tập các số chính phương ≤ 400 ;

7) Sử dụng các hàm về tổng, tích các phần tử của mảng, hãy thực hiện các yêu cầu sau:

(a) Cho n = 100. Tính tổng n số chẵn đầu tiên. (ĐS: 10100)

(b) Cho n = 100. Tính tổng n số lẻ đầu tiên. (ĐS: 10^4)

(c) Cho n=20. Tính tổng n số tự nhiên đầu tiên chia hết cho 12. (ĐS: 2520)

(d) Cho n=20. Tính tổng n số chính phương đầu tiên. (ĐS: 2870)

(d) Tính tổng S = 8! + 9! +10! (ĐS: $4032*10^3$)

(e) Tính tổng S = 1/3! + 1/4! + 1/5! (ĐS: 0.2167)

(f) Tính tổng $T = 1+3+5+...+97+99+100$ (ĐS: 1717)

(g) Tính tổng $T = 1+4+7+...+100+8!$ (ĐS: 41545)

(h) Tính tổng T = 1+ 1/2 + 1/4 + 1/8 +1/16 (ĐS: 1.9375)

8) Cho các ma trận sau:

$$A = \begin{bmatrix} 1 & 2 \\ -1 & 2 \end{bmatrix} \quad ; \qquad B = \begin{bmatrix} 7 & 2 \\ 1 & 0 \end{bmatrix} \quad ; \qquad C = \begin{bmatrix} 1+2i & 5-2i \\ 3+i & 1+3i \end{bmatrix}$$

Hãy thực hiện các phép toán:

(a) A .*B ; B./A

(b) A.^B ; B.\A

(c) A*B*C ; A*B.*C

(d) Tính định thức của A, B, C.

9) Cho ma trận :

$$A = \begin{bmatrix} 6 & 43 & 2 & 11 & 87 \\ 12 & 6 & 34 & 0 & 5 \\ 24 & 18 & 7 & 41 & 9 \end{bmatrix}$$

Sử dụng các phép toán trên mảng và vectơ như ( ), : , ...thực hiện các yêu cầu sau:

(a) Tạo một vectơ hàng có 5 phần tử là hàng 2 của ma trận A.

(b) Tạo một vectơ cột có 3 phần tử là cột 4 của ma trận A.

(c) Tạo một vectơ hàng có 10 phần tử là hàng 1 và 2 của ma trận A.

(d) Tạo một vectơ hàng chứa các phần tử ở cột 2 và cột 5 của ma trận A.

10) Tạo ba ma trận như sau:

$$A = \begin{bmatrix} 5 & 2 & 4 \\ 1 & 7 & 3 \\ 6 & -10 & 0 \end{bmatrix} \quad ; \quad B = \begin{bmatrix} 11 & 5 & -3 \\ 0 & -12 & 4 \\ 2 & 6 & 1 \end{bmatrix} \quad ; \quad C = \begin{bmatrix} 7 & 14 & 1 \\ 10 & 3 & -2 \\ 8 & -5 & 9 \end{bmatrix}$$

Thực hiện các phép toán :

(a) A+B và B+A. Nhận xét ?

(b) A+(B+C) và (A+B)+C. Nhận xét ?

(c) 5*(A+C) và 5*A + 5*C. Nhận xét ?

(d) A*(B+C) và A*B + A*C. Nhận xét ?

(e) A*B và B*A . Nhận xét ?

(f) A*B và B' *A' . Nhận xét ?

(g) (A+B)' và A' + B' . Nhận xét ?

11) Nhập vào hai đa thức:

$$P(x) = x^4 + 3x^3 - x^2 - 5x + 1$$

$$Q(x) = 3x^2 - 5x + 4$$

(a) Tính tổng hai đa thức trên.

(b) Nhân hai đa thức trên.

(c) Tìm nghiệm của 2 đa thức trên.

(d) Tính giá trị P(1) ; Q(0).

(e) Tính đạo hàm bậc 1, bậc 2 của P(x) và Q(x).

12) Phân tích các ảnh Laplace sau đây thành tổng các phân thức tối giản:

(a) $G(s) = \dfrac{10}{s^3 + 15s^2 + 68s + 96}$

(b) $H(s) = \dfrac{10s + 30}{s(s^3 + 15s^2 + 68s + 96)}$

(c) $H(s) = \dfrac{120s + 10}{s(s^2 + 13s + 30)}$

(d) $G(s) = \dfrac{20}{(s+2)(s^2 + 4s + 13)}$

Chương 4

1) Viết lại các chương trình ví dụ có trong tài liệu về cách sử dụng các cấu trúc **if-elseif-else-end, switch-case, vòng lặp for, vòng lặp while,** ...
2) Viết chương trình tính nghiệm của phương trình bậc hai $ax^2 + bx + c = 0$ với a, b, c nhập từ bàn phím. Các trường hợp phương trình có hai nghiệm thực, nghiệm kép, hai nghiệm phức đều phải xuất thông báo ra màn hình.
3) Viết chương trình in ra màn hình 50 số chẵn đầu tiên và tính tổng, tích các số đó.
4) Viết chương trình in ra màn hình n số lẻ đầu tiên, với n được nhập từ bàn phím. Tính tổng các số đã in. Chương trình phải cảnh báo nếu người dùng nhập giá trị $n \leq 0$.
5) Viết chương trình in ra màn hình n số tự nhiên đầu tiên chia hết cho a , với n và a được nhập từ bàn phím. Tính tổng các số đã in.
6) Viết chương trình tìm và in ra màn hình các số tự nhiên có ba chữ số, biết rằng các số đó chia cho 3 thì dư 1, chia cho 5 thì dư 2, chia cho 4 thì dư 3.
7) Viết chương trình tìm và in ra màn hình các số tự nhiên có ba chữ số, biết rằng các số đó chia cho 8, 9, 30 đều dư 3.
8) Viết chương trình in ra và tính tổng các số chính phương từ 1 đến 400.
9) Sử dụng vòng lặp để tính tổng bình phương n số tự nhiên; tổng bình phương n số chẵn; tổng bình phương n số lẻ; tổng bình phương n số tự nhiên chia hết cho k, với n và k là số dương bất kỳ;
10) Sử dụng vòng lặp để tính tổng $S = 1! + 2! + \ldots + n!$
11) Sử dụng vòng lặp để tính tổng $S = 1/1! + 1/2! + \ldots + 1/n!$
12) Sử dụng vòng lặp để tính tổng $S = 1/1^2 + 1/2^2 + \ldots + 1/n^2$
13) Viết chương trình in ra màn hình các số nguyên tố <150 và tính tổng các số đó.
14) Viết chương trình in ra màn hình n số nguyên tố đầu tiên với n được nhập từ bàn phím và tính tổng, tích các số nguyên tố đó.

Chương 5

1) Sử dụng lệnh **plot** vẽ đồ thị của các hàm sau:

(a) $f(x) = 5 - 4x - x^2 \quad , \quad x \in [-6, 2]$

(b) $g(x) = 2x^2 - 8x - 11 \quad , \quad x \in [-1, 5]$

(c) $y(t) = 1 - 3e^{-4t} + 2e^{-2t} \quad , \quad x \in [0, 2]$

(d) $h(t) = 1 - e^{-2t}\cos t - e^{-2t}\sin t \quad , \quad t \in [0, 4]$

(e) $h(t) = 1 - e^{-2t}\cos \omega t \quad ; \; t \in [0, 4] \; ; \; \omega = 1, 2, 4, 6.$

(f) $h(t) = 1 - e^{-5t}\cos\sqrt{10}\, t + \frac{2}{\sqrt{10}} e^{-5t}\sin\sqrt{10}\, t \quad t \in [0, 2]$

(g) $h(t) = \frac{29}{40}\, 1 - e^{-12t}\cos 4t - 3e^{-12t}\sin 4t \qquad t \in [0, 1]$

2) Sử dụng lệnh **ezplot** vẽ đồ thị 2 cặp hàm tham số sau trên cùng hệ trục, $t \in [0, 2\pi]$

$$\begin{cases} x = 7\cos t - \cos 7t \\ y = 7\sin t - \sin 7t \end{cases} \quad \text{và} \quad \begin{cases} x = 2\cos t \\ y = 2\sin t \end{cases}$$

Thiết đặt màu sắc, marker riêng cho từng đường đồ thị thông qua **handle** và lệnh **set.** Đồ thị thứ nhất vẽ màu vàng, marker là (*). Đồ thị thứ hai vẽ màu đỏ, marker là (o).

3) Sử dụng lệnh **ezplot** vẽ đồ thị 2 cặp hàm tham số sau trên cùng hệ trục, $t \in [0, 2\pi]$

$$\begin{cases} x = 6\cos t - \cos 9t \\ y = 6\sin t - \sin 9t \end{cases} \quad \text{và} \quad \begin{cases} x = 2\cos^3 t \\ y = 2\sin^3 t \end{cases}$$

Thiết đặt màu sắc, kiểu nét, marker. Đặt lại tiêu đề của đồ thị.

4) (a) Sử dụng lệnh **ezplot** vẽ đồ thị hàm số: $y(t) = 2(1 - e^{-t/T})$

Các giá trị của hằng số thời gian T lần lượt chọn là 2, 3, 4, 6. Các đồ thị được vẽ trên cùng hệ trục có trục hoành $t \in [0, 20]$, trục tung $y \in [0, 2.2]$. Thiết đặt màu sắc riêng cho từng đường đồ thị. Quan sát các đồ thị và nhận xét ?

(b) Tạo một figure mới, dùng lệnh **plot** để vẽ đồ thị và thực hiện các yêu cầu trên.

5) (a) Dùng lệnh **plot** vẽ trên cùng một hệ trục toạ độ đồ thị ba hàm số :

$y1=\sin x$; $y2=\sin(x+\pi/3)$; $y3=\sin(x+2\pi/3)$ với $x \in [0, 3\pi]$

Thiết đặt màu sắc, ghi chú cho đồ thị. Đặt tiêu đề là 'Dòng điện hinh sin ba pha'.

(b) Tạo một figure mới, dùng lệnh **ezplot** để vẽ đồ thị và thực hiện các yêu cầu trên.

6) Vẽ đồ thị từng cặp hàm số sau đây trên cùng một đồ thị. Mỗi cặp hàm số được vẽ trong một figure riêng. Trong mỗi figure hãy tạo chú thích bằng lệnh **legend** để chỉ rõ mỗi đường đồ thị tương ứng với hàm số nào:

(a) $u(x) = \cos 2x + \sin 3x$; $v(x) = 3\cos(x) - 2\sin(2x)$; $x \in [0, 4\pi]$

(b) $f(x) = ex^{-3x}$; $g(x) = x^{-3x}(1-3x)$; $x \in [0, 2]$

(c) $f(t) = \sin 3t \cos 2t$; $g(t) = (1/2)\cos t + (5/2)\cos 5t$; $t \in [-2\pi, 2\pi]$

(d) $y_1(t) = (1+2\sin t)\cos t$; $y_2(t) = (1+2\sin t)\sin t$; $t \in [0, 2\pi]$

7) Sử dụng lệnh ezplot vẽ đồ thị các hàm số sau trên cùng một hệ trục tọa độ :

$$y_1(t) = 1 - 3e^{-2t} + 2e^{-3t} \ ;$$

$$y_2(t) = 1 - e^{-4t}\cos 3t + \frac{1}{3}e^{-4t}\sin 3t$$

Thiết đặt hệ trục toạ độ tương ứng với $t \in [0, 4]$; y_1 và $y_2 \in [0, 1.2]$

8) Vẽ trên cùng một hệ trục toạ độ đồ thị của hai hàm số :

$$h1(t) = 1 + \frac{18}{11}e^{-58t} - \frac{29}{11}e^{-80t}$$

$$h2(t) = 1 + e^{-69t}\left[\frac{47}{11}\sinh(11t) - \cosh(11t)\right]$$

Thiết đặt hệ trục toạ độ với $t \in [0, 0.4]$; h1 và $h2 \in [0, 1.2]$.

Nhận xét kết quả và giải thích?

9) Vẽ trên cùng một hệ trục toạ độ đồ thị của hai hàm số :

$$h1(t) = 1 - \frac{3}{4}e^{-2t} - \frac{1}{4}e^{-6t}$$

$$h2(t) = 1 - e^{-4t}\left[\frac{1}{2}\sinh(2t) + \cosh(2t)\right]$$

Thiết đặt hệ trục toạ độ với $t \in [0, 3]$; h1 và $h2 \in [0, 1.05]$.

Chương 6

1) Tìm giá trị bé nhất và lớn nhất của các hàm số sau:

a. $y = x^3 - 6x^2 + 5$; x ∈ [-1, 5]

b. $y = 2e^{-x}\sin(x)$; x ∈ [-6, 0]

c. $y = \sqrt{100 - x^2}$; x ∈ [-4, 4]

d. $y = \dfrac{1 - x - x^2}{1 + x + x^2}$; x ∈ [-200,200]

e. $y = 2tgx - tg^2x$; x ∈ [0, π/2]

f. $y = xe^{-x^2}$; x ∈ [-1, 1]

a	xmin= 4 ymin=-27	xmax= 0 ymax= 5
b	xmin=-2.356 ymin=-14.92	xmax=-5.498 ymax= 345.28
c	xmin= 4 ymin= 9.165	xmax= 0 ymax= 10
d	xmin= -200 ymin= -1	xmax=-0.5 ymax= 1.667
e	xmin= 1.57 ymin=-inf	xmax= 0.785 ymax= 1
f	xmin=-0.707 ymin=-0.429	xmax= 0.707 ymax= 0.429

Hướng dẫn :

Cách 1: Vẽ và sử dụng đồ thị.

Cách 2: Dùng lệnh **[xmin,ymin] = fminbnd (y,x1,x2)**

Ví dụ:

```
>> y='x^3-6*x^2+5'
>> [xmin, ymin] = fminbnd (y,-1,5)          % tìm  xmin, ymin của hàm y=f(x)
>> yN='-(x^3-6*x^2+5)'  % yN= ['-',y]       % tạo yN = - y
>> [xNmin, yNmin] = fminbnd (yN,-1,5)       % tìm xmin, ymin của yN
>> xmax=xNmin,  ymax= -yNmin                % đảo dấu của  yNmin
```

2) Tính các giới hạn sau :

a. $\lim\limits_{x\to 0}\left(\dfrac{\sin x}{x}\right)$

b. $\lim\limits_{x\to 0-}\left(\dfrac{1}{x}\right)$; $\lim\limits_{x\to 0+}\left(\dfrac{1}{x}\right)$

c. $\lim\limits_{x\to \frac{\pi}{2}} 2\text{tg}x - \text{tg}^2x$

d. $\lim\limits_{t\to +\infty} 1 - e^{-5t}$; $\lim\limits_{x\to +\infty}\left(1 + \dfrac{a}{x}\right)^x$

e. $\lim\limits_{h\to 0}\left(\dfrac{\sin(x+h) - \sin x}{h}\right)$

f. $\lim\limits_{x\to 1}\left(\dfrac{p}{1-x^p} - \dfrac{q}{1-x^q}\right)$

g. $\lim\limits_{x\to a}\left(\dfrac{x^x - a^a}{x - a}\right)$

h. $\lim\limits_{x\to 0-}\left(\dfrac{1}{1+e^{1/x}}\right)$; $\lim\limits_{x\to 0+}\left(\dfrac{1}{1+e^{1/x}}\right)$

i. $\lim\limits_{x\to 0}\left(\dfrac{a^x - x\ln a}{b^x - x\ln b}\right)^{\frac{1}{x^2}}$

k. $\lim\limits_{x\to +\infty} = \dfrac{1 - x - x^2}{1 + x + x^2}$; $\lim\limits_{x\to -\infty} = \dfrac{1 - x - x^2}{1 + x + x^2}$

Chú ý: Khi x → 0- lấy giới hạn bên trái ('left') điểm 0
Khi x → 0+ lấy giới hạn bên phải ('right') điểm 0
Khi x → -∞ lấy giới hạn tại -Inf hoặc giới hạn bên phải ('right') điểm -Inf
Khi x → +∞ lấy giới hạn tại Inf hoặc giới hạn bên trái ('left') điểm Inf

3) Dùng hàm **symsum** tính các tổng sau :

a. $\sum_{1}^{n} n$; $\sum_{1}^{n} n^2$; $\sum_{1}^{n} n^3$; $\sum_{1}^{n} n^4$

b. $\sum_{1}^{n} 2n$; $\sum_{1}^{n} 3n$; $\sum_{1}^{n} (3n+1)$; $\sum_{1}^{n} 4n^2$

c. $\sum_{1}^{n} (2k-1)$; $\sum_{1}^{n} (2k-1)^2$; $\sum_{k=1}^{n} (2k-1)^3$

d. $\sum_{k=1}^{n} (1/k)$; $\sum_{k=1}^{n} (1/ak)$; $\sum_{k=1}^{n} (1/k^2)$

f. $\sum_{k=0}^{n} k!$; $\sum_{k=0}^{\infty} \frac{x^k}{k!}$; $\sum_{k=0}^{n} \frac{\sin(k\pi)}{k}$

Cho n=10. Hãy tính giá trị của mỗi tổng đã cho bằng các phương pháp sau :

1- Dùng hàm **symsum** tính tổng S theo n . Sau đó gán n=10 ; eval(S)
2- Dùng hàm **symsum** tính tổng trực tiếp với n=10.
3- Dùng vòng lặp **for** hoặc **while**
4- Dùng các hàm **sum**, **prod** **(**tính tổng, tích các phần tử của vectơ) .

Nhận xét về phạm vi ứng dụng và kết quả của các phương pháp trên.

4) Dùng lệnh **dsolve** giải các phương trình vi phân sau đây :

a. $y' = ay$

b. $y' - y = \sin t$

c. $y'' + 2y' + y = (2x+4)e^{-x}$

d. $2xyy' - y^2 + x = 0$

e. $y'^{\,2} + y^2 = 1$ với điều kiện (ĐK) đầu $y(0) = 0$

f. $y'' + 3y' + 2y = 0$ với ĐK đầu $y'(0) = a$; $y(0) = b$

g. $y'' + 2y' + 5y = 3$ với ĐK đầu $y'(0) = y(0) = 0$

h. $y'' - a^2 y = 0$ với ĐK $y(0)=1;\ y'(\pi/a) = 0$

i. $y'' + 3y' = x + \cos x$ với ĐK đầu $y'(0) = y(0) = 0$

5) Tìm hàm quá độ h(t) nếu biết ảnh Laplace H(s) như sau:

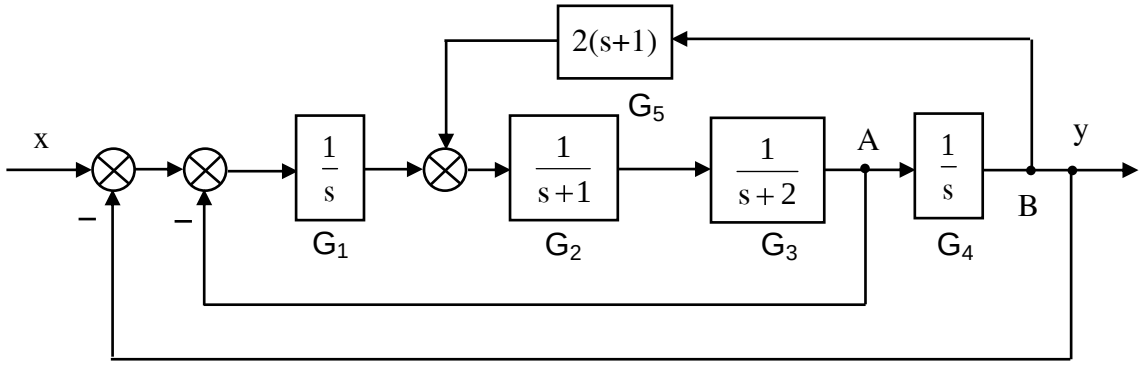
(a) $H(s) = \frac{40}{s(s^2+13s+40)}$ (b) $H(s) = \frac{10s+25}{s(s^2+6s+25)}$

(c) $H(s) = \frac{3s+15}{s(s^2+6s+15)}$ (d) $H(s) = \frac{116(s+40)}{s(s^2+138s+4640)}$

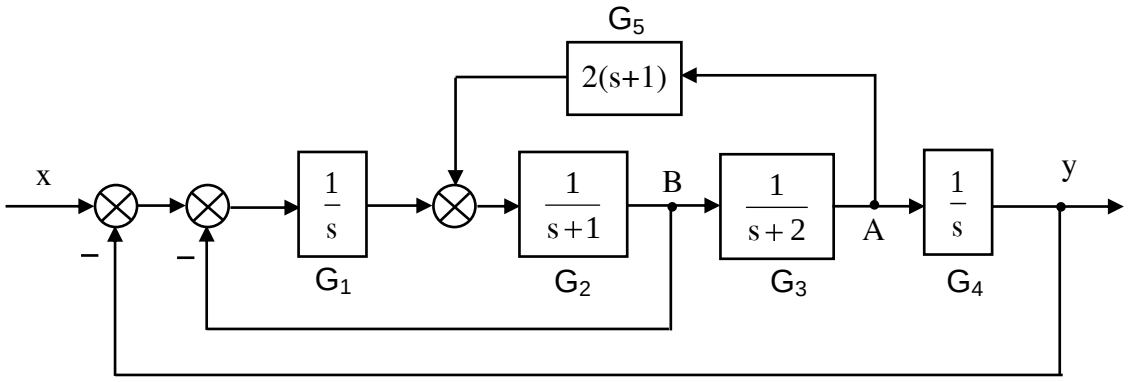
(e) $H(s) = \frac{5s+25}{s(s^2+8s+25)}$ (f) $H(s) = \frac{122(s+43)}{s(s^2+157s+5246)}$

Chương 7

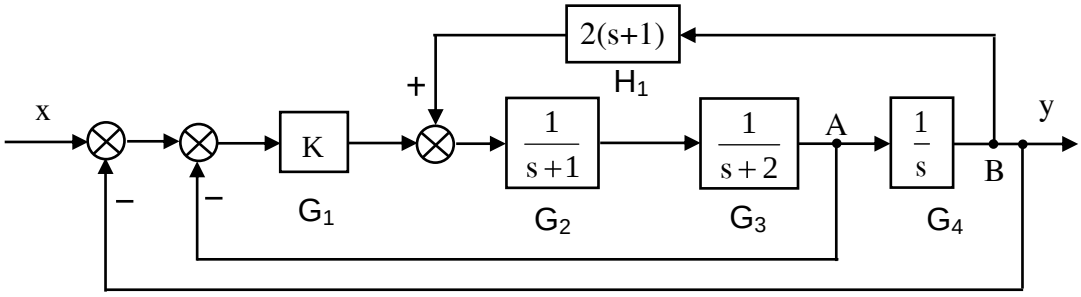
1) Tìm hàm truyền đạt của hệ thống có sơ đồ khối sau đây:



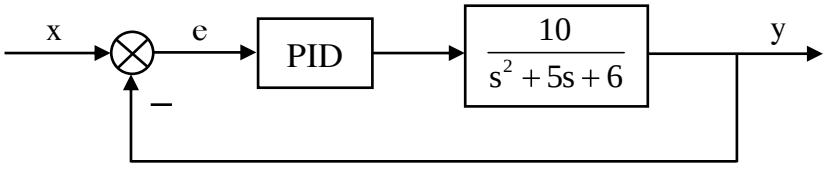
2) Tìm hàm truyền đạt của hệ thống có sơ đồ khối sau đây:



3) Tìm hàm truyền đạt của hệ thống có sơ đồ khối sau đây:



4) Cho hệ thống có sơ đồ khối như hình vẽ .



Bộ điều khiển PID có K_P =7; K_I =12; K_D =1.

Tín hiệu vào bậc thang đơn vị x= 1(t).

Hãy viết chương trình Matlab thực hiện các yêu cầu a, b, c, d, e sau đây:

a. Tìm hàm truyền đạt G(s) của hệ thống.
b. Xác định giá trị các cực và zero của hệ thống.
c. Vẽ biểu đồ đáp ứng quá độ h(t) của hệ thống.
d. Tìm biểu thức của hàm quá độ h(t) = ?
e. Tìm giá trị xác lập h(∞) và sai số xác lập e(∞)

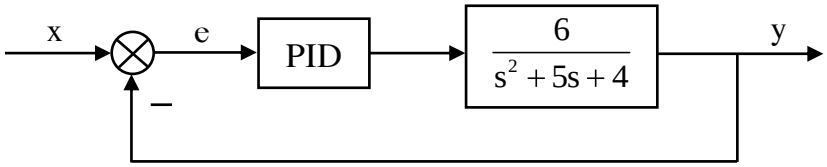
Lưu chương trình, đặt tên tập tin là Bai7_4.m . Gọi thực hiện chương trình và thực hiện tiếp yêu cầu sau đây:

f. Thao tác với đồ thị để xác định các thông số chất lượng: thời gian quá độ (Settling time), độ vọt lố (Overshoot), giá trị xác lập h(∞), sai số xác lập e(∞).

Vào menu file> save as> lưu chương trình với tên mới là Bai7_4B.m . Chỉnh sửa chương trình trên tập tin mới để thực hiện tiếp các yêu cầu sau đây:

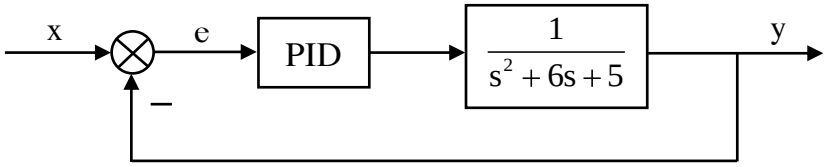
g. Giữ nguyên giá trị ban đầu của K_P và K_D. Thay đổi giá trị K_I tuỳ ý. Xác định giá trị giới hạn của K_I để hệ thống luôn ổn định. (nói cách khác, h(∞) có giá trị hữu hạn). Trong phạm vi hệ thống còn ổn định, có nhận xét gì về giá trị của h(∞) và e(∞) khi K_I thay đổi ?
h. Đặt giá trị K_I =0 và K_D=1. Thay đổi giá trị K_P tuỳ ý. Xác định giá trị giới hạn của K_P để hệ thống luôn ổn định. Trong phạm vi hệ thống còn ổn định, có nhận xét gì về giá trị của e(∞) và tính dao động của hệ khi K_P thay đổi ?
i. Đặt giá trị K_I =0 và K_P=7. Thay đổi giá trị K_D tuỳ ý. Có nhận xét gì về giá trị e(∞) và tính dao động của hệ khi K_D thay đổi ?

5) Thực hiện tương tự bài tập 4 với hệ thống có sơ đồ khối như hình vẽ .



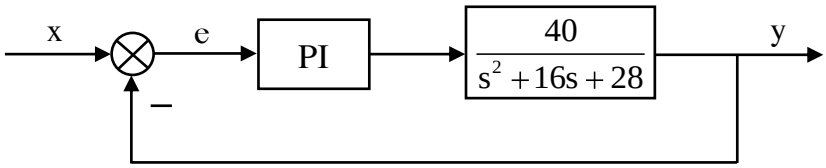
Bộ điều khiển PID có K_P =6; K_I =8; K_D =1.

6) Thực hiện tương tự bài tập 4 với hệ thống có sơ đồ khối như hình vẽ .



Bộ điều khiển PID có K_P =9; K_I =20; K_D =1.

7) Thực hiện tương tự bài tập 4 với hệ thống có sơ đồ khối như hình vẽ .



Bộ điều khiển PI có K_P =1; K_I =2 .

CÂU HỎI TRẮC NGHIỆM

1) Tên biến nào sau đây là không hợp lệ:
   a) Chieucao b) Chieu-cao
   c) ChieuCao d) ChIeUcao1
2) Tên biến nào sau đây là hợp lệ:
   a) G1(s) b) G_1s
   c) 1Gs d) G1s
3) Để xoá hai biến u và v trong Workspace có thể dùng lệnh :
   a) >> clear u, v b) >> clear u v
   c) >> clear u ; v d) cả a và b đều đúng
4) Để xoá mọi biến trong Workspace và làm sạch màn hình MATLAB có thể dùng lệnh :
   a) >> clear clc b) >> clear , clc
   c) >> clear (clc) d) cả 3 cách đều được
5) Câu lệnh nào sau đây là hợp lệ, không bị báo lỗi :
   a) s= ' input ' b) s= input (Hay nhap gia tri s:)
   c) s= input d) s= Input ('Hay nhap gia tri s: ')
6) Để mở mới cửa sổ soạn thảo Editor từ cửa sổ MATLAB có thể dùng cách :
   a) gõ lệnh edit b) vào menu File -> New-> M-File
   c) a và b đều được d) a và b đều sai
7) M-file có thể là:
   a) scrift file b) function file
   c) file chương trình d) bạn có ý kiến khác
8) Để thực hiện file "baitap1.m", tại command window có thể đánh lệnh:
   a) >>baitap1 b) >>baitap1.m
   c) cả hai đều được d) cả hai đều sai
9) Dòng khai báo hàm nào sau đây là không hợp lệ :
   a) function TT= bai_1(a,b,c) b) function [TT]= bai_1(a,b,c)
   c) function = bai_1(a,b,c) d) function [TT, DT]= bai_1(a,b,c)
10) >> A= [1 2] là câu lệnh dùng để nhập A với A là :
   a) vectơ cột b) vectơ cột bước tăng bằng 2
   c) véctơ hàng d) vectơ hàng bước tăng bằng 2
11) >> B= [1 2 3;-4 6 1] là câu lệnh dùng để nhập B với B là :
   a) ma trận b) ma trận hàng
   c) ma trận cấp 2 d) ma trận cấp 3
12) Cho 2 véctơ u = [1 3 5] và v= [2, 4, 6], có thể thực hiện được phép tính nào sau đây:
   a) u' .*v b) u*v
   c) cả hai d) không phép tính nào

13) Cho 2 véctơ u = [1;3 ;5]; v= [2 4 6], có thể thực hiện được phép tính nào sau đây:
   a) u.*v
   b) v' *u
   c) cả hai
   d) không phép tính nào

14) Lệnh nào sau đây dùng để tìm véctơ chứa các phần tử ở hàng một của ma trận A ?
   a) A(1,:)
   b) A(:,1)
   c) A(1 :)
   d) A(1)

15) Lệnh nào sau đây dùng để tìm véctơ chứa các phần tử ở cột hai của ma trận A ?
   a) A(2,:)
   b) A(:,2)
   c) A(2 :)
   d) A(2)

16) Lệnh >>X= linspace(0,10) cho kết quả X là một :
   a) véctơ hàng
   b) véctơ hàng có 100 phần tử
   c) véctơ cột
   d) véctơ hàng có bước tăng bằng 1

17) Các trường hợp nào sau đây sẽ báo lỗi:
   a) >>x=12; disp(x)
   b) >> x=12; fprintf (x)
   c) cả hai
   d) không trường hợp nào

18) Lệnh nào sau đây sẽ không thực hiện được ?
   a) G=tf(6,[1 5 6])
   b) G=tf([6],[1 5 6])
   c) s=tf(s); G=6/s^2+5*s+6
   d) G=zpk([ ],[-2 -3],6)

19) Lệnh vẽ đồ thị nào sau đây sẽ bị báo lỗi ?
   a) syms x; y=x^2; ezplot(y,[0, pi])
   b) syms x; y=x^2; ezplot(x,y,[0, pi])
   c) cả hai
   d) không lệnh nào

20) Lệnh vẽ đồ thị nào sau đây sẽ bị báo lỗi ?
   a) x=0:100; y=x.^2 ; plot(x,y)
   b) y=x.^2 ; plot(x,y,[0:100])
   c) cả hai
   d) không lệnh nào

21) Có thể dùng lệnh nào để cộng hai đa thức (x+1) và (x^2+3) ?
   a) >>[1 1] + [1 3]
   b) >> [1 1] + [1 0 3]
   c) >>[0 1 1] + [1 0 3]
   d) >>[1 1 0] + [1 0 3]

22) Để tìm hàm truyền của hai phần tử nối tiếp, có thể dùng :
   a) toán tử +
   b) Hàm series
   c) a và b đều được
   d) a và b đều sai

23) Để tìm hàm truyền của hai phần tử song song, có thể dùng :
   a) toán tử *
   b) Hàm parallel
   c) cả hai đều được
   d) cả hai đều sai

24) Hệ sys mô tả bởi các lệnh: >> G1=tf(1,[1,4]); sys=feedback(G1,1,1)
   a) là hệ phản hồi dương
   b) là hệ phản hồi âm
   c) là hệ phản hồi âm đơn vị
   d) là hệ phản hồi dương đơn vị