GIỚI THIỆU CHUNG

Điều khiển tự động hoá đóng vai trò quan trọng trong sự phát triển của khoa học và kỹ thuật. Lĩnh vực này hữu hiệu khắp nơi từ hệ thống phi thuyền không gian, hệ thống điều khiển tên lửa, máy bay không người lái, người máy tay, máy trong các quá trình sản xuất hiện đại và ngay cả trong đời sống hàng ngày: điều khiển nhiệt độ, độ ẩm

Trong lý thuyết điều khiển tự động cổ điển các nhà bác học Jame Watt, Hazen, Minorsky, Nyquist, Evan…. đã đưa ra những phương pháp giải quyết nhiều vấn đề đơn giản như: bộ điều tốc ly tâm để điều chỉnh nhiệt độ máy hơi nước, chứng minh tính ổn định của hệ thống có thể được xác định từ phương trình vi phân mô tả hệ thống, xác định tính ổn định của hệ thống vòng kín trên cơ sở đáp ứng vòng hở đối với các tín hiệu vào hình Sin ở trạng thái xác lập…

Khi các máy móc hiện đại ngày nay càng phức tạp hơn nhiều tín hiệu vào và ra thì việc mô tả hệ thống điều khiển hiện đại này đòi hỏi một lượng rất lớn các phương trình. Lý thuyết điều khiển cổ điển liên quan các hệ thống một ngõ vào và một ngõ ra trở nên bất lực để phân tích hệ thống nhiều đầu vào, nhiều đầu ra. Kể từ khoảng năm 1960 trở đi nhờ máy tính số cho phép ta phân tích các hệ thống phức tạp trong miền thời gian, lý thuyết điều khiển hiện đại phát triển để đối phó với sự phức tạp của hệ thống hiện đại. Lý thuyết điều khiển hiện đại dựa trên phân tích miền thời gian và tổng hợp dùng các biến trạng thái, cho phép giải các bài toán điều khiển có các yêu cầu chặt chẽ về độ chính xác, trọng lượng và giá thành của các hệ thống trong lĩnh vực kỹ nghệ không gian và quân sự.

Sự phát triển gần đây của lý thuyết điều khiển hiện đại là trong nhiều lĩnh vực điều khiển tối ưu của các hệ thống ngẫu nhiên và tiền định. Hiện nay máy vi tính ngày càng rẻ, gọn nhưng khả năng xử lý lại rất mạnh nên nó được dùng như là một phần tử trong các hệ thống điều khiển.

MATLAB là một chương trình phần mềm lớn của lĩnh vực tính toán số. Matlab chính là chữ viết tắt từ MATrix LABoratory, thể hiện định hướng chính của chương trình bao gồm một số hàm toán các chức năng nhập / xuất cũng như các khả năng lập trình với cú pháp thông dụng mà nhờ đó ta có thể dựng nên các Scripts. MATLAB có rất nhiều phiên bản như: 3.5, 4.0, 4.2, 5.0, 5.2,…6.0, 6.5 . Hiện tại đã có phiên bản mới nhất 7.1. Trong bài tiểu luận này chúng ta chủ yếu tìm hiểu về phiên bản 6.5.

Simulink là một phần mềm mở rộng của Matlab (1 Toolbox của Matlab) dùng để mô hình hoá, mô phỏng và phân tích một hệ thống động. Thông thường dùng để thiết kế hệ thống điều khiển, thiết kế DSP, hệ thống thông tin và các ứng dụng mô phỏng khác.

Simulink là thuật ngữ mô phỏng dễ nhớ được ghép hai từ Simulation và Link, Simulink cho phép mô tả hệ thống tuyến tính, hệ phi tuyến, các mô hình trong miền thời gian liên tục, hay gián đoạn hoặc một hệ gồm cả liên tục và gián đoạn.

PHẦN I : CƠ SỞ VỀ MATLAB

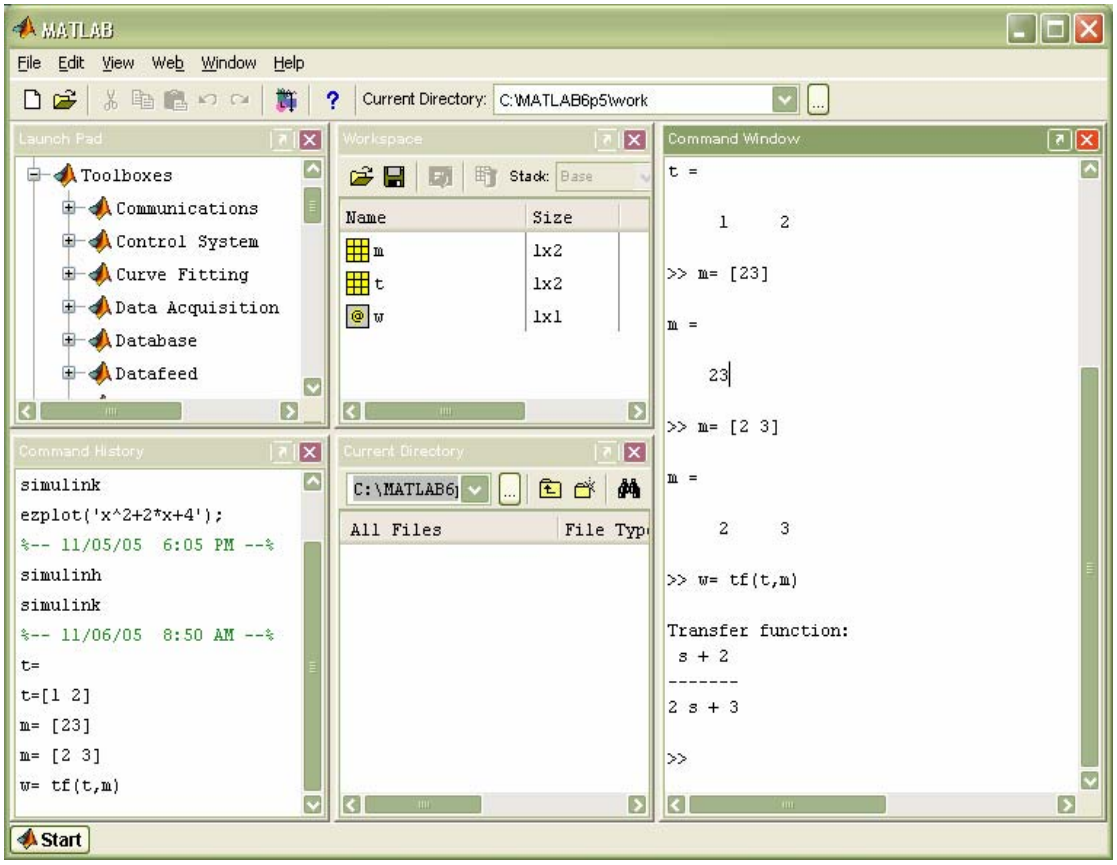
MATLAB là một chương trình phần mềm lớn về lĩnh vực toán số . Tên bộ chương trình chính là chữ viết tắt từ MATrix LABoratory, thể hiện định hướng của chương trình là những phép tính vector và ma trận . Phần cốt lõi của chương trình bao gồm một số hàm toán , các chức nănng nhập /xuất cũng như các khả năng điều khiển chu trình mà nhờ đó có thể dựng trên các Scripts .

Trong phần nay bao gồm các Toolbox liên quan tới Điều Khiển –Tự Động hóa như: Control System Toolbox, Signal Processing Toolbox, Optimization Toolbox, Stateflow Blockset, Power System Blockset , Real – Time Workshop va SIMULINK. SIMULINK là một toolbox có vai trò bặc biệt quan trọng: Vài trò của một công cụ mạnh phục vụ *mô hình hóa và mô phỏng các hệ thống Kỹ thuật – Vật lý* trên cơ sở sơ đồ cấu trúc dạng khối . Cùng với SIMULINK , Statefow Blockset tạo cho ta khả năng mô hình hóa và mô phỏng các automat trạng thái hữu hạn.

1.1. Những bước đi dầu tiên với MATLAB

1.1.1 Màn hình MATLAB

Sau khi khỏi động MATLAB , môi trường tích hợp với những cửa sổ chính như hình dưới :



- Cửa sổ **Launch Pad** : Cửa sổ này cho phép người sử dụng truy cập nhanh các công cụ của MATLAB, Phần Help (trợ giúp) hoặc Online Documents (tài liệu trực tuyến), mở Demos (chương trình trình diễn).

- Cửa sổ thư mục hiện tại **Current Directory Browser** : Nhờ cửa sổ này người sử dụng nhanh chóng nhận biết, chuyển đổi thư mục hiện tại của môi trường công tác, mở File, tạo thư mục mới.

- Cửa sổ môi trường công tác **Workspace Browser** : Tất cả các biến, các hàm tồn tại trong môi trường công tác đều được hiển thị tại cửa sổ nàyvới đầy đủ các thông tin như: Tên loại biến/hàm, kích thước tùy theo Bytes và loại dữ liệu. Ngoài ra còn có thể cất vào bộ nhớ các dữ liệu đó , hoặc sử dụng chức năng Array Editor (soạn thảo mảng) để thay đổi các biến

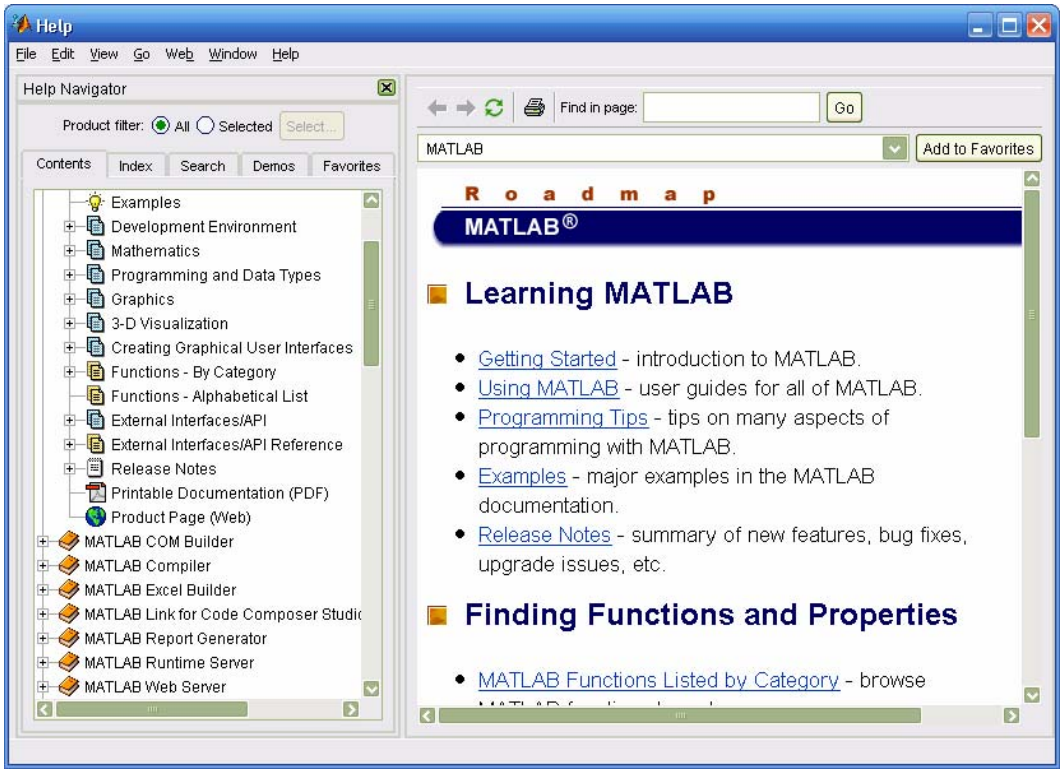
- Cửa sổ lệnh **Command Windows** : Đây là cửa sổ chính của MATLAB . Tại đây ta thực hiện toàn bộ việc nhập dữ liệu và xuất kết quả tính toán. Dấu nhấp nháy >> báo hiệu chương trình sắp hoạt động:

- Mỗi lần nhập dữ liệu được kết thúc bằng động tác nhấn phím ENTER. Nguyên tắc " nhân, chia thực hiện trước cộng , trừ " và thư tự ưu tiên của dấu ngoặc vẫn như bình thường . Số có giá trị lớn thường được nhập với hàm e mũ (có thể viết E) . Có thể kết thúc chương trình bằng cách đóng màn hình MATLAB , hoặc gọi lệnh *quit, exit* hoặc nhấn tổ hợp phím *Ctrl+q*

- Cửa sổ quá khứ **Command History** : Tất cả các lệnh đã sử dụng trong Command Windows được lưu giữ và hiển thị tại đây, có thể lặp lại lệnh cũ bằng cách nháy chuột kép vào lệnh đó . Cũng có thể cắt, sao hoặc xóa cả nhóm lệnh hoặc từng lệnh riêng rẽ.

1.1.2 Tiện ích trợ giúp (Help) của MATLAB

Tiện ích trợ giúp của MATLAB là vô cùng phong phú . Tùy theo nhu cầu , hoặc gọi Help [command] để xem nội dung hỗ trợ của lệnh command trực tiếp trên Command Windows hoặc sử dụng công cụ truy cập Help



Có thể gọi của sổ Help bằng cách gọi trên Menu , gọi lệnh *helpwin* hay *doc* trực tiếp trên của sổ Command Windows . Bằng lệnh *loockfor searchstring* ta có thể tìm chuỗi ký tự *searchstring* trong dòng đầu của mọi MATLAB File trong thư mục MATLAB

```
>> help log


 LOG    Natural logarithm.
   LOG(X) is the natural logarithm of the elements of X.
   Complex results are produced if X is not positive.


   See also LOG2, LOG10, EXP, LOGM.


 Overloaded methods
   help gf/log.m
   help sym/log.m
   help fints/log.m
   help designdev/log.m
>>
```

Các lệnh liên quan tới tiện ích help được tập hợp trong bảng sau:

	Help
help[command]	Tiện ích Help trực tuyến của MATLAB trong cửa sổ lệnh Commmand Workspace
helpwin[command]	Tiện ích Help trực tuyến của MATLAB trong cửa Sổ truy cập Help
doc[command]	Tư liệu trực tuyến của MATLAB trong cửa sổ truy cập Help
*lookfor*searchstring	Tìm chuỗi ký tự searchstringtrong dòng đầu tiên của mọi MATLAB Files trong thư mục MATLAB

1.1.3 Các biến

Thông thường , kết quả cảu các biến được gán cho *ans* . Sủ dụng dấu bằng ta có thẻ định nghĩa một biến , đồng thời gán giá trị cho biến đó . Khi nhập tên của một biến mà không gán giá trị , ta thu giả trị hiện tại của biến đó . Tất cả các biến đều là biến *global* trong Workspace. Tên của biến có thể chứa tới 32 chữ cái , gạch ngang thấp (_) cũng như chữ số . Chữ viết hoa to và chữ viết nhỏ đều được phân biệt .

Việc nhập giá trị có thể được thực hiện thành một chuỗi trong cùng một dòng , chỉ cách nhau bởi dấu (;) . Nếu sử dụng dấu phẩy(,) để tách các lệnh khi ấy các giá trị sẽ được xuất ra màn hình :

```
>> x=25; y=10;
>> x
```

```
x =
   25
>> a=x+y,A=x/y
a =
   35
A =
   2.5000
```

Một số biến như : *pi , i , j* và *inf* đã được MATLAB dùng đêr chỉ các hằng số hay ký hiệu, vậy ta phải tránh sử dụng chúng . Đối với các phép tính bất định (ví dụ 0/0), trên màn hình sẽ xuất hiện kết quả *NaN* (Not a Number) . *esp* cho ta biết cấp chính xác tương đối khi biểu diễn số với dấu phẩy động (ví dụ : *esp* = 2.2204e-016):

```
>> 1/0
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this warning.)
ans =
  Inf                    Inf: infinite (vô cùng)
>> 0/0
Warning: Divide by zero.
(Type "warning off MATLAB:divideByZero" to suppress this warning.)
ans =
  NaN                    NaN: not – defined (bất định )
```

Các ký hiệu

=	Gán giá trị cho biến
+ - * / ^	Các phép tính
;	Nhập giá trị (còn giữ vai trò dấu cách khi nhập nhiều giá trị trong cùng một dòng)
,	Dấu cách khi xuất nhiều giá trị trong cùng một dòng
esp	Cấp chính xác tương đối khi sử dụng giá trị dấu phẩy động
i j	Toán tử ảo
inf	Vô cùng (∞)
NaN	Not a Number
pi	Hằng số π

1.1.4 Các hàm toán học

Chương trình MATLAB có sẵn rất nhiều hàm toán tập hợp trong bảng sau đây . Tất cả các hàm trong bảng đều có khả năng sử dụng tính của vector

Các hàm toán			
sqrt(x)	Căn bậc hai	rem(x,y)	Số dư của phép chia x/y
exp(x)	Hàm mũ cơ số e	round(x)	Làm tròn số
log(x)	Logarit tự nhiên	ceil(x)	Làm tròn lên
log10(x)	Logarit cơ số thập phân	floor(x)	Làm tròn xuống
abs(x)	Giá trị tuyệt đối	sum(*v*)	Tổng các phần tử vector
sign(x)	Hàm dấu	prod(*v*)	Tích các phần tử vector
real(x)	Phần thực	min(*v*)	Phần tử vector bé nhất
imag(x)	Phần ảo	max(*v*)	Phần tử vector lớn nhất
phase(x)	Góc pha của số phức	mean(*v*)	Giá trị trung bình cộng
Các hàm lượng giác			
sin(x)	Hàm sin	atan(x)	Hàm arctg $\pm 90^0$
cos(x)	Hàm cos	atan2(x,y)	Hàm arctg $\pm 180^0$
tag(x)	Hàm tg	sinc(x)	Hàm $\sin(\pi x)/(\pi x)$

1.2 Vector và ma trận

MATLAB có một số lệnh đặc biệt để khai báo hoặc sử lý vector và ma trận . Cách đơn giản nhất để khai báo , tạo lên vector hoặc ma trận là nhập trực tiếp . Khi nhập trực tiếp các phần tử của một hàng được cách bởi dấu phẩy hoặc vị trí cách bỏ trống[1,] các hàng được cách bởi dấu (;) hoặc ngắt dòng.

```
>> vector=[3 4 5]
vector =
    3    4    5
>> matran=[vector; 1 2 3]
matran =
    3    4    5
    1    2    3
```

Vector có các phần tử tiếp diễn với một bước nhất định , có thể nhập một cách đơn giản nhờ

Toán tử (:) như sau (start: increment; destination) "(xuất phát : bước; đích)". Nếu chỉ nhập start và destination , MATLAB sẽ tự động đặt increment là +1.

Cũng có thể nhập các vector tuyến tính cũng như vector có phân hạng logarithm bằng cách dùng lệnh *linspace*(start, destination, number) "(Trong đó number là số lương phần tử của vector)". Ta cũng có thể nhập bằng lệnh *logspace*, start và destination được nhập bởi số mũ thập phân , ví dụ : thay vì nhập 100 = (10^2)ta chỉ cần nhập 2.

```
>> long=1:5
long =
    1    2    3    4    5
>> deep = 10:-2:2
deep =
   10    8    6    4    2
>> longer=linspace(1,15,5)
```

```
longer =
   1.0000    4.5000    8.0000   11.5000   15.0000
>> licreace=logspace(1,2,5)
licreace =
  10.0000   17.7828   31.6228   56.2341  100.0000
```

Bằng các hàm *ones(line,column)*và *zeros(line, column)* ta tạo các ma trận có phần tử là 1 hoặc 0. Hàm *eye(line)* tạo ra ma trận đơn vị, ma trận toàn phương với các phần tử 1 thuộc đường chéo , tất cả các phần tử còn lại là 0. Kích cỡ của ma trận hoàn toàn phụ thuộc người nhập:

```
>> M= ones(2, 3)
M =
    1     1   1
    1     1   1
```

Việc truy cập từng phần tử của vector hoặc ma trận được thực hiện bằng cách khai báo chỉ số của phần tử , trong đó cần lưu ý rằng : chỉ số bé nhất là 1 chứ không phảI là 0. Đặc biệt , khi cần xuất từng hàng hay từng cột , có thể sử dụng toán tử (:) đứng một mình , điều ấy có nghĩa là : phải xuất mọi phần tử của hàng hay cột :

```
>> matran(2,2)
ans =
    2
>> matran(2,:)
ans =
    1     2     3
```

MATLAB có một lệnh rất hữu ích , phục vụ tạo ma trận với chức năng tín hiệu thử đó là : *rand(m,n)*. Khi gọi ta thu được ma trận *m* hàng và *n* cột với phần tử mang các giá trị ngẫu nhiên:

```
>> mt_ngaunhien=rand(2,3)
mt_ngaunhien =
   0.4565    0.8214    0.6154
   0.0185    0.4447    0.7919
```

Khai báo vector và ma trận

[x1 x2 …; x3 x4 …]	Nhập giá trị cho vector và ma trận
start: increment: destination	Toán tử (:)
linspace (start,destination ,number)	Khai báo tuyến tính cho vector
logspace (start,destination ,number)	Khai báo logarithm cho vector
eye(line)	Khai báo ma trận đơn vị
ones(line,column)	Khai báo ma trận với các phần tử 1
zeros(line,column)	Khai báo ma trận với các phần tử 0
rand(line,column)	Khai báo ma trận với các phần tử nhập ngẫu nhiên

1.2.1 Tính toán với vector và ma trận

Nhiều phép tính có thể áp dụng cho vector và ma trận . Ví dụ : Phép nhân với ký hiệu(*) được dùng để tính tích của vector và ma trận . Việc chuyển vị của vector và ma trận được thực hiện nhờ lệnh *transpose* hoặc (‘) . Nếu vector và ma trận là phức , ta dùng thêm lệnh là *ctranspose* hoặc (‘) để tìm giá trị phức liên hợp. Đối với các giá trị thực hai lệnh trên như nhau

```
>> M*matran
ans =
   4    6    8
   4    6    8
```

Nếu như trong các phép tính * / ^ cần được thực hiện cho từng phần tử của vector và ma trận , ta sẽ phải đặt thêm vào trước ký hiệu của phép tình đó ký hiệu (.). Phép tính đối với các biến vô hướng luôn được thực hiện cho từng phần tử một :

```
>> M ./ matran
ans =
   0.3333    0.2500    0.2000
   1.0000    0.5000    0.3333
```

Phép tính trên cũng có hiệu lực cả khi ma trận có các phần tử phức:

```
>> matranphuc = [1+i 1-i; 1 2 ]
matranphuc =
  1.0000 + 1.0000i   1.0000 - 1.0000i
  1.0000             2.0000
>> matranphuc*matranphuc
ans =
  1.0000 + 1.0000i   4.0000 - 2.0000i
  3.0000 + 1.0000i   5.0000 - 1.0000i
>> matranphuc.*matranphuc
ans =
      0 + 2.0000i        0 - 2.0000i
  1.0000             4.0000
```

Lệnh *diff*(vector [n]) tính vector sai phân. Bằng lệnh *conv*(vector_1, vvector_2) ta chạp hai vector *vector_1* và *vector_2*. Nếu hai vector cần chập có phần tử là các hệ số của hai đa thức, kết quả thu được sẽ ứng với các hệ số sau khi nhân hai đa thức đó với nhau :

```
>> diff(vector)
ans = 1   1
```

Hai lệnh *inv* và *det* dùng để nghịch đảo ma trận toàn phương và tính định thức của ma trận . Giá trị riêng của ma trận *matrix* được tính bởi lệnh *eig(matrix)* và hạng của nó được tính bởi lệnh *rank(matrix)*, Nếu cần chuyển vị ma trận ta dùng lệnh *transpose(matrix)*:

```
>> matrix=[1 2 3;3 4 5;5 6 7]
```

```
matrix =
    1    2    3
    3    4    5
    5    6    7
>> rank(matrix)
ns =
    2
>> eig(matrix)
ans =
   12.9282
   -0.9282
    0.0000
>> det(matrix)
ans =
    0
>> inv(matrix)
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.850372e-018.
ans =
  1.0e+016 *
    0.4504   -0.9007    0.4504
   -0.9007    1.8014   -0.9007
    0.4504   -0.9007    0.4504
>> transpose(matrix)
ans =
    1    3    5
    2    4    6
    3    5    7
```

Tính toán với vector và ma trận

.* ./ .^	Các phép tính với từng phần tử
transpose(matrix) hoặc *matrix'*	Chuyển vị ma trận *matrix*
ctranspose(matrix) hoặc *matrix'*	Chuyển vị ma trận *matrix* có phần tử phức liên hợp
inv(matrix)	Đảo ma trận
det(matrix)	Tính định thức của ma trận
eig(matrix)	Tính giá trị riêng của ma trận
rank(matrix)	Xác định hạng của ma trận
diff(vector[n]*)*	Tính vector sai phân
conv(vector_1,vector_2)	Chập vector (nhân đa thức)

1.2 Cấu trúc và trường

1.2.1 Cấu trúc

Để thuận tiện cho việc quản lý và sử dụng , ta có thể tập hợp nhiều biến lại trong một cấu trúc . Trong đó môi mảng có một tên riêng (một chuỗi ký tự string) đặt giữa hai dấu (‘ ‘) có kèm theo giá trị . Một cấu trúc được tạo nên bởi lệnh *struct*(‘name_1’,value_1,’name_2’,value_2,…..):

```
>>my_structure = struct(‘data’, matrix, ‘size’, [2 3]);
```

Việc truy cập vào dữ liệu được thực hiện bởi với dấu cách(.);

```
>>my_structure (2) . data = matrix.^(-1) ;

ans  =

        1.0000            0.5000            0.3333
```

Ngoài ra MATLAB còn có các lệnh về cấu trúc móc vòng như cấu trúc nhập bởi lệnh *componist* .

1.2.2 Trường

Tổng quát ở một mức độ cao hơn cấu trúc là *trường* (Cell Array). Đó chính là các Array (mảng nhiều chiều), chứa Cell (tế bào) với dữ liệu thuộc các loại và kích cỡ khác nhau . Ta có thể tạo ra Cell Array bằng lệnh *cell* , hoặc đơn giản hơn bằng cách ghép các phần tử bên trong dấu ngoặc {}. Từng phần tử của Cell Array có thể được truy cập như các vector , ma trận thông thương như các Array nhiều chiều , chỉ cần lưu ý rằng : Thay vì dùng dấu ngoặc tròn ( ) ta sử dụng dấu ngoặc móc {}.

Giả sử ta tạo ra một Cell Array rỗng có tên *my_cell* như sau :

```
>> my_cell = cell(2,2)

my_cell =

    []    []

    []    []
```

Bây giờ ta lần lượt gán cho từng mảng của my_cell các giá trị sau đây:

```
>> my_cell{1,1} ='chao cac ban';

>> my_cell{1,2} ='chuc cac ban hoc tap tot';

>> my_cell{2,1} =[1 2; 3 4];

>> my_cell{2,2} =10;
```

Khi nhập tên của Cell Array trên màn hình xuất hiện lên đầy đủ cấu trúc của nó. Có thể biết nội dung (hay giá trị) của một hay nhiều Cell khi ta nhập các chỉ số của Cell:

```
>> my_cell

my_cell =

    'chao cac ban'    [1x24 char]

     [2x2 double]    [       10]

>> my_cell{1,1}

ans =

chao cac ban

>> my_cell{1,2}
```

```
ans =
chuc cac ban hoc tap tot
>> my_cell{2,1}
ans =
    1    2
    3    4
```

Cấu trúc (Structure) và trường (Cell Array)

Structure('n1','v1','n2','v2', …)	Khai báo cấu trúc
Structure.name	Truy cập vào phần tử name
My_cell = {}	Tạo Cell Array rỗng
Cell(n)	Tạo n×n Cell Array
Cell(m,n)	Tạo m×n Cell Array

Phần trên là những khái niệm khái quát và những ví dụ cụ thể giới thiệu một phần nhỏ những ứng dụng mà phần mềm MATLAB có thể thực hiện .MATLAB là một phần mềm lớn trong lĩnh vực toán số và còn có khả năng của một ngôn ngữ lập trình bậc cao với tính năng đồ họa phong phú. MATLAB với những công cụ như : Control System Toolbox (công cụ khảo sát thiết kế hệ thống điều khiển), Optimization Toolbox (công cụ tính toán tối ưu) và Signal Processing Toolbox (công cụ sử lý tín hiện). MATLAB đang là phần mềm mà các kỹ sư các sinh viên sử dụng rộng rãi nhờ vào tình năng ưu việt của phần mềm này

PHẦN II

GIỚI THIỆU MỘT SỐ NHÓM LỆNH CƠ BẢN MATLAB

I. LỆNH CƠ BẢN

1. Lênh ANS

a) Công dung: **(Purpose)**

Là biến chứa kết quả mặc định.

b) Giải thích: **(Description)**

Khi thực hiện một lệnh nào đó mà chưa có biến chứa kết quả, thì MATLAB lấy biến Ans làm biến chứa kết quả đó.

2. Lênh CLOCK

a) Công dung: **(Purpose)**

Thông báo ngày giờ hiện tại.

b) Cú pháp:**(Syntax)**

c = clock

c) Giải thích: **(Description)**

Để thông báo dễ đọc ta dùng hàm fix.

3. Lênh COMPUTER

a) Công dung: **(Purpose)**

Cho biết hệ điều hành của máy vi tính đang sử dụng Matlab.

b) Cú pháp: **(Syntax)**

computer

[c,m] = computer

c) Giải thích: **(Description)**

c: chứa thông báo hệ điều hành của máy.

m: số phần tử của ma trận lớn nhất mà máy có thể làm việc được với Matlab.

4. Lênh DATE

a) Công dung: **(Purpose)**

Thông báo ngày tháng năm hiện tại

b) Cú pháp: **(Syntax)**

s = date

5. Lênh CD

a) Công dung:

Chuyển đổi thư mục làm việc.

b) Cú pháp:

cd

cd diretory

cd ..

c) Giải thích:

cd: cho biết thư nục hiện hành.

diretory: đường dẫn đến thư mục muốn làm việc.

cd .. chuyển đến thư mục cấp cao hơn một bậc.

6. Lênh CLC

a) Công dung:

Xóa cửa sổ lệnh.

b) Cú pháp:

clc

7. Lênh CLEAR

a) Công dung:

Xóa các đề mục trong bộ nhớ.

b) Cú pháp:

clear

clear name

clear name1 name2 name3

clear functions

clear variables

clear mex

clear global

clear all

c) Giải thích:

clear: xóa tất cả các biến khỏi vùng làm việc.

clear name: xóa các biến hay hàm được chỉ ra trong name.

clear functions: xóa tất cả các hàm trong bộ nhớ phụ

clear variables: xóa tất cả các biến ra khỏi bộ nhớ.

clear mex: xóa tất cả các tập tin .mex ra khỏi bộ nhớ.

clear: xóa tất cả các biến chung.

clear all: xóa tất cả các biến, hàm, và các tập tin .mex khỏi bộ nhớ. Lệnh này làm cho bộ nhớ trống hoàn toàn.

8. Lênh DELETE

a) Công dung:

Xóa tập tin và đối tượng đồ họa.

b) Cú pháp:

delete filename

delete (n)

c) Giải thích:

file name: tên tập tin cần xóa.

n: biến chứa đối tượng đồ họa cần xóa. Nếu đối tượng là một cửa sổ thì cửa sổ sẽ đóng lại và bị xóa.

9. Lênh DEMO

a) Công dụng:

Chạy chương trình mặc định của Matlab.

b) Cú pháp:

demo

c) Giải thích:

demo: là chương trình có sẵn trong trong Matlab, chương trình này minh họa một số chức năng của Matlab.

10. Lênh DIARY

a) Công dụng:

Lưu vùng thành file trên đĩa.

b) Cú pháp:

diary filename

c) Giải thích:

filename: tên của tập tin.

11. Lênh DIR

a) Công dụng:

Liệt kê các tập tin và thư mục.

b) Cú pháp:

dir

dir name

c) Giải thích:

dir: liệt kê các tập tin và thư mục có trong thư mục hiện hành.

dir name: đường dẫn đến thư mục cần liệt kê.

12. lênh DISP

a) Công dụng:

Trình bày nội dung của biến (x) ra màn hình

b) Cú pháp:

disp (x)

c) giải thích:

x: là tên của ma trận hay là tên của biến chứa chuỗi ký tự, nếu trình bày trực tiếp chuỗi ký tự thì chuỗi ký tự được đặt trong dấu ‘’

13. Lênh ECHO

a) Công dụng:

Hiển thị hay không hiển thị dòng lệnh đang thi hành trong file *.m.

b) Cú pháp:

echo on

echo off

c) Giải thích:

on: hiển thị dòng lệnh.

off: không hiển thị dòng lệnh.

14. Lênh FORMAT

a) Công dụng:

Định dạng kiểu hiển thị của các con số.

Cú pháp	Giải thích	Ví dụ
Format short	Hiển thị 4 con số sau dấu chấm	3.1416
Format long	Hiển thị 14 con số sau dấu chấm	3.14159265358979
Format rat	Hiển thị dạng phân số của phần nguyên nhỏ nhất	355/133
Format +	Hiển thị số dương hay âm	+

15. Lênh HELP

a) Công dụng:

hướng dẫn cách sử dụng các lệnh trong Matlab.

b) Cú pháp:

help

help topic

c) Giải thích:

help: hiển thị vắn tắt các mục hướng dẫn.

topic: tên lệnh cần được hướng dẫn.

16. Lênh HOME

a) Công dụng:

Đem con trỏ về đầu vùng làm việc.

b) Cú pháp:

home

17. Lênh LENGTH

a) Công dụng:

Tính chiều dài của vectơ.

b) Cú pháp:

l = length (x)

c) Giải thích:

l: biến chứa chiều dài vectơ.

18. Lênh LOAD

a) Công dụng:

Nạp file từ đĩa vào vùng làm việc.

b) Cú pháp:

load

load filename

load filename

load finame.extension

c) Giải thích:

load: nạp file matlap.mat

load filename: nạp file filename.mat

load filename.extension: nạp file filename.extension

Tập tin này phải là tập tin dạng ma trận có nghĩa là số cột của hàng dưới phải bằng số cột của hàng trên. Kết quả ta được một ma trận có số cột và hàng chính là số cột và hàng của tập tin văn bản trên.

19. Lênh LOOKFOR

a) Công dụng:

Hiển thị tất cả các lệnh có liên quan đến topic.

b) Cú pháp:

lookfor topic

c) Giải thích:

topic: tên lệnh cần được hướng dẫn.

20. Lênh PACK

a) Công dụng:

Sắp xếp lại bộ nhớ trong vùng làm việc.

b) Cú pháp:

pack

pack filename

c) Giải thích:

Nếu như khi sử dụng Matlap máy tính xuất hiện thông báo "Out of memory" thì lệnh pack có thể tìm thấy một số vùng nhớ còn trống mà không cần phải xóa bớt các biến.

Lệnh pack giải phóng không gian bộ nhớ cần thiết bằng cách nén thông tin trong vùng nhớ xuống cực tiểu. Vì Matlab quản lý bộ nhớ bằng phương pháp xếp chồng nên các đoạn chương trình Matlab có thể làm cho vùng nhớ bị phân mảnh. Do đó sẽ có nhiều vùng nhớ còn trống nhưng không đủ để chứa các biến lớn mới.

Lệnh pack sẽ thực hiện:

+ lưu tất cả các biến lên đĩa trong một tập tin tạm thời là pack.tmp.

+ xóa tất cả các biến và hàm có trong bộ nhớ.

+ lấy lại các biến từ tập tin pack.tmp.

+ xóa tập tin tạm thời pack.tmp.

kết quả là trong vùng nhớ các biến được gộp lại hoặc nén lại tối đa nên không bị lãng phí bộ nhớ.

Pack.finame cho phép chọn tên tập tin tạm thời để chứa các biến. Nếu không chỉ ra tên tập tin tạm thời thì Matlab tự lấy tên tập tin đó là pack.tmp.

Nếu đã dùng lệnh pack mà máy vẫn còn báo thiếu bộ nhớ thì bắt buộc phải xóa bớt các biến trong vùng nhớ đi.

21. Lênh PATH

a) Công dụng:

Tạo đường dẫn, liệt kê tất cả các đường dẫn đang có.

b) Cú pháp:

path

p = path

path (p)

c) Giải thích:

path: liệt kê tất cả các dường dẫn đang có.

p: biến chứa đường dẫn.

path (p): đặt đường dẫn mới.

22. Lênh QUIT

a) Công dung:

Thoát khỏi Matlab.

b) Cú pháp:

quit

23. Lênh SIZE

a) Công dung:

Cho biết số dòng và số cột của một ma trận.

b) Cú pháp:

d = size (x)

[m,n] = size (x)

m = size (x,1)

n = size (x,2)

c) Giải thích:

x: tên ma trận.

d: tên vectơ có 2 phần tử, phần tử thứ nhất là số dòng, phần tử còn lại là số cột.

m,n: biến m chứa số dòng, biến n chứa số cột

24. Lệnh TYPE

a) Công dụng:

Hiển thị nội dung của tập tin.

b) Cú pháp:

type filename

c) Giải thích:

filename: tên file cần hiển thị nội dung.

Lệnh này trình bày tập tin được chỉ ra.

25. Lệnh WHAT

a) Công dụng:

Liệt kê các tập tin *.m, *.mat, *.mex.

b) Cú pháp:

what

what dirname

c) Giải thích:

what: liệt kê tên các tập tin .m, .mat, .mex có trong thư mục hiện hành.

dirname: tên thư mục cần liệt kê.

26. Lệnh WHICH

a) Công dụng:

Xác định chức năng của funname là hàm của Matlab hay tập tin.

b) Cú pháp:

which funname

c) Giải thích:

funname: là tên lệnh trong Matlab hay tên tập tin

d) Ví dụ:

which inv

inv is a build-in function

which f

c:\matlab\bin\f.m

27. Lệnh WHO, WHOS

a) Công dụng:

Thông tin về biến đang có trong bộ nhớ.

b) Cú pháp:

who

whos

who global

whos global

c) Giải thích:

who: liệt kê tất cả các tên biến đang tồn tại trong bộ nhớ.

whos: liệt kê tên biến, kích thước, số phần tử và xét các phần ảo có khác 0 không.

who global và whos: liệt kê các biến trong vùng làm việc chung.

II. CÁC TOÁN TỬ VÀ KÝ TỰ ĐẶC BIỆT

1. Các toán tử số hoc (Arithmetic Operators):

Toán tử	Công dụng
+	Cộng ma trận hoặc đại lượng vô hướng (các ma trận phải có cùng kích thước).
-	Trừ ma trận hoặc đại lượng vô hướng (các ma trận phải có cùng kích thước).
*	Nhân ma trận hoặc đại lượng vô hướng (ma trận 1 phải có số cột bằng số hàng của ma trận 2).
.*	Nhân từng phần tử của 2 ma trận hoặc 2 đại lượng vô hướng (các ma trận phải có cùng kích thước).
\	Thực hiện chia ngược ma trận hoặc các đại lượng vô hướng (A\B tương đương với inv (A)*B).
.\	Thực hiện chia ngược từng phần tử của 2 ma trận hoặc 2 đại lượng vô hướng (các ma trận phải có cùng kích thước).
/	Thực hiện chia thuận 2 ma trận hoặc đại lượng vô hướng (A/B tương đương với A*inv(B)).
./	Thực hiện chia thuận từng phần tử của ma trận này cho ma trận kia (các ma trận phải có cùng kích thước).
^	Lũy thừa ma trận hoặc các đại lượng vô hướng.
.^	Lũy thừa từng phần tử ma trận hoặc đại lượng vô hướng (các ma trận phải có cùng kích thước).

2.. Toán tử quan hê (Relational Operators):

Toán tử	Công dụng
<	So sánh nhỏ hơn.
>	So sánh lớn hơn.
>=	So sánh lớn hơn hoặc bằng.
<=	So sánh nhỏ hơn hoặc bằng.
= =	So sánh bằng nhau cả phần thực và phần ảo.
-=	So sánh bằng nhau phần ảo.

Giải thích:

Các toán tử quan hệ thực hiện so sánh từng thành phần của 2 ma trận. Chúng tạo ra một ma trận có cùng kích thước với 2 ma trận so sánh với các phần tử là 1 nếu phép so sánh là đúng và là 0 nếu phép so sánh là sai.

Phép so sánh có chế độ ưu tiên sau phép toán số học nhưng trên phép toán logic.

3. Toán tử logig (Logical Operators):

Toán tử	Công dụng
&	Thực hiện phép toán logic AND.
|	Thực hiện phép toán logic OR.
~	Thực hiện phép toán logic NOT.

a) Giải thích:

Kết quả của phép toán là 1 nếu phép logic là đúng và là 0 nếu phép logic là sai.

Phép logic có chế độ ưu tiên thấp nhất so với phép toán số học và phép toán so sánh.

b) Ví du:

Khi thực hiện phép toán 3>4 & 1+ thì máy tính sẽ thực hiện 1+2 được 3, sau đó tới 3>4 được 0 rồi thực hiện 0 & 3 và cuối cùng ta được kết qủa là 0.

4. Ký tư đăc biêt (Special Characters):

Ký hiệu	Công dụng
[]	Khai báo vector hoặc ma trận.
()	Thực hiện phép toán ưu tiên, khai báo các biến và các chỉ số của vector.
=	Thực hiện phép gán.
‘	Chuyển vị ma trận tìm lượng liên hiệp của số phức.
.	Điểm chấm thập phân.
,	Phân biệt các phần tử của ma trận và các đối số trong dòng lệnh.
;	Ngăn cách giữa các hàng khi khai báo ma trận.
%	Thông báo dòng chú thích.
!	Mở cửa sổ MS – DOS.

III. CÁC HÀM LOGIC (LOGICAL FUNCTION)

1. Lênh ALL

a) Công dung:

Kiểm tra vector hay ma trận có giá trị 0 hay không.

b) Cú pháp:

y = all(x)

c) Giải thích:

y: biến chứa kết quả

x: tên vedtor hay ma trận

y = 1 khi tất cả các phần tử khác 0

y = 0 khi có 1 phần tử bằng 0

2. Lênh ANY

a) Công dung:

Kiểm tra vector hay ma trận có giá trị khác 0 hay không.

b) Cú pháp:

y = any(x)

c) Giải thích:

y: biến chứa kết quả.

x: tên vector, hay ma trận.

y = 1 khi có 1 phần tử khác 0.

y = 0 khi có 1 phần tử bằng 0.

3. Lệnh EXIST

a) Công dụng:

Kiểm tra biến hay file có tồn tại hay không.

b) Cú pháp:

e = exist('item')

c) Giải thích:

item: là tên file hay tên biến.

e: biến chứa giá trị trả về.

e	Ý nghĩa
0	item không tồn tại trong vùng làm việc
1	item là biến đang tồn tại trong vùng làm việc
2	item đang tồn tại trên đĩa (chỉ kiểm tra trong thư mục hiện hành)
3	item là MEX-file
4	item là file được dịch từ phần mềm Simulink
5	item là hàm của Matlab

4. Lệnh FIND

a) Công dụng:

Tìm phần tử trong vector hay ma trận theo yêu cầu.

b) Cú pháp:

k = find(x)

[i,j] = find(x)

[i,j,s] = find(x)

c) Giải thích:

k: chỉ vị trí của phần tử cần tìm trong vector.

i,j: chỉ số hàng và số cột tương ứng của phần tử cần tìm.

s: chứa giá trị của phần tử cần tìm.

x: tên vector, ma trận hay là yêu cầu đề ra. Nếu không nêu ra yêu cầu thì mặc nhiên là tìm các phần tử khác 0.

IV. NHÓM LỆNH LẬP TRÌNH TRONG MATLAB

1. Lênh EVAL

a) Công dung:

Chuyển đổi chuỗi ký tự thành biểu thức.

b) Cú pháp:

kq = eval('string')

c) Giải thích:

kq: biến chứa kết quả.

Nếu 'string' là các ký số thì chuyển thành những con số.

Nếu 'string' là câu lệnh thì chuyển thành các lệnh thi hành được.

2. Lênh FOR

a) Công dung:

Dùng để thực hiện 1 công việc cần lặp đi lặp lại theo một quy luật, với số bước lặp xác định trước.

b) Cú pháp:

```
for biến điều khiển = giá trị đầu : giá trị cuối,
        thực hiện công việc;
end
```

c) Giải thích:

Công việc chính là các lệnh cần thi hành, có thể có nhiều lệnh, kết thúc lệnh phải có dấu;

3. Lênh FUNCTION

a) Công dung:

Tạo thêm hàm mới.

b) Cú pháp:

function s = n(x)

c) Giải thích:

s: tên biến chứa giá trị trả về sau khi thi hành hàm.

n: tên gợi nhớ.

4. Lênh INPUT

a) Công dụng:

Dùng để nhập vào 1 giá trị.

b) Cú pháp:

tên biến = input ('promt')

tên biến = input ('promt', 's')

c) Giải thích:

tên biến, là nơi lưu giá trị ngập vào.

‘promt’: chuỗi ký tự muốn nhập vào.

‘s’: cho biết giá trị nhập vào là nhiều ký tự.

5. Lênh IF …ELSEIF …ELSE

a) Công dung:

Thực hiện lệnh khi thỏa điều kiện.

b) Cú pháp:

```
if biểu thức luận lý 1
        thực hiện công việc 1;
elseif biểu thức luận lý 2
        thực hiện công việc 2;
else
        thực hiện công việc 3;
end
```

c) Giải thích:

Khi biểu thức luận ký 1 đúng thì thực hiện công việc 1 tương tự cho biểu thức luận lý 2. Nếu cả hai biểu thức sai thì thực hiện công việc sau lệnh else.

Biểu thức luận lý là các phép so sánh ==, <, >, <=, >=

công việc chính là các lệnh cần thi hành, có thể có nhiều lệnh, kết thúc lệnh phải có dấu ;

6. Lênh MENU

a) Công dung:

Tạo menu để chọn chức năng.

b) Cú pháp:

tên biến = menu (‘Tên menu’,‘chức năng1’,‘chức năng2’, …. , ‘chức năng n’)

c) Giải thích:

tên menu: là tiêu đề của menu.

tên biến: là nơi cất giá trị nhận được sau khi chọn chức năng của menu.

Chức năng 1, 2, ….,n:khi chọn chức năng nào thì tên biến có giá trị là số thứ tự của chức năng đó.

7. Lênh PAUSE

a) Công dung:

Dừng chương trình theo ý muốn.

b) Cú pháp:

```
pause on
pause off
pause (n)
```

c) Giải thích:

pause on: dừng chương trình, và chờ nhấn 1 phím bất kỳ (trừ các phím điều khiển) chương trình thực hiện tiếp.

pause off: tắt chức năng pause.

pause (n): dừng chương trình tại n giây.

d) Ví dụ:

```
for n = 1 : 3;
        disp('Press any key to continue…')
        pause
end
```

```
Press any key to continue…
Press any key to continue…
Press any key to continue…
```

8. Lệnh WHILE

a) Công dụng:

Dùng để thực hiện 1 công việc cần lặp đi lặp lại theo một quy luật, với số bước lặp không xác định, phụ thuộc vào biểu thức luận lý.

b) Cú pháp:

```
while biểu thức luận lý
        thực hiện công việc;
end
```

c) Giải thích:

Biểu thức luận lý là các phép so sánh = =, <, >, <=, >=

Công việc chính là các lệnh cần thi hành, có thể có nhiều lệnh, kết thúc lệnh phải có dấu ;

Khi thực hiện xong công việc thì quay lên kiểm tra lại biểu thức luận lý, nếu vẫn còn đúng thì tiếp tục thực hiện, nếu sai thì kết thúc.

V. TẬP LỆNH XỬ LÝ CHUỖI

1. Lệnh ABS

a) Công dụng:

Tạo vector đơn có giá trị của mỗi phần tử là số thứ tự tương ứng với ký tự trong bảng mả ASCII.

Lấy trị tuyệt đối của một số âm.

b) Cú pháp:

```
n = ABS(s)
x = ABS(a)
```

c) Giải thích:

n: tên vector.

s: chuỗi ký tự, hoặc là tên biến chứa chuỗi ký tự.

a: số âm, hoặc là tên biến chứa số âm.

x: trị tuyệt đối của a.

2. Lênh BLANKS

a) Công dung:

Tạo khoảng trắng giữa hai hay nhiều chuỗi ký tự theo mong muốn.

b) Cú pháp:

[S1 BLANKS(b1) S2 BLANKS(b2) …BLANKS(bn) Sn]

c) Giải thích:

S1, S2, …Sn: các chuỗi ký tự.

b1, b2: số khoảng trắng.

3. Lênh DEC2HEX

a) Công dung:

Đổi con số của hệ 10 sang hệ 16.

b) Cú pháp:

s = dec2hex(n)

c) Giải thích:

s: biến chứa chuỗi ký số của hệ 16

n: con số nguyên hệ 10.

4. Lênh HEX2DEC

a) Công dung:

Đổi chuỗi ký số của hệ 16 sang con số của hệ 10.

b) Cú pháp:

n = hex2dec('s')

c) Giải thích:

n: con số của hệ 10.

s: chuỗi ký số hệ 16.

5. Lênh INT2STR

a) Công dụng:

Chuyển số nguyên sang dạng chuỗi.

Chuyển các ký tự trong một chuỗi sang số thứ tự tương ứng trong bảng mã ASCII.

b) Cú pháp:

kq = INT

c) Giải thích:

kq: biến STR(n)chứa kết quả.

n: tên biến cần chuyển.

Nếu n là số nguyên thì kq là chuỗi ký số.

Nếu n là chuỗi ký tự thì kq là số tương ứng trong bảng mã ASCII

6. Lênh ISSTR

a) Công dung:

Kiểm tra nội dung biến có phải là chuỗi ký tự không.

b) Cú pháp:

kq = isstr(n)

c) Giải thích:

kq: biến chứa kết quả.

n: tên biến cần kiểm tra.

kq = 1 nếu n là chuỗi ký tự.

0 nếu n không là chuỗi ký tự.

7. Lênh LOWER

a) Công dung:

Cho ra chuỗi ký tự viết thường.

b) Cú pháp:

b = lower(s)

c) Giải thích:

b: biến chứa kết quả.

s: tên biến chứa chuỗi ký tự hay chuỗi ký tự.

8. Lênh NUM2STR

a) Công dung:

Chuyễn số thực sang dạng chuỗi.

Chuyển các ký tự trong một chuỗi sang số thứ tự tương ứng trong bảng mã ASCII.

b) Cú pháp:

kq = num2tr(n)

c) Giải thích:

kq: biến chứa kết quả.

n: tên biến cần chuyển.

Nếu n là số thực thì kq là số tương ứng trong bảng mã ASCII.

9. Lênh SETSTR

a) Công dung:

Cho ra ký tự tương ứng với số thứ tự trong bảng mã ASCII.

b) Cú pháp:

x = Set Str(n)

c) Giải thích:

x: biến chứa ký tự tương ứng (thuộc bảng mã ASCII).

n: số nguyên ($0 \le n \le 255$).

10. Lênh STR2MAT

a) Công dung:

Tạo ma trận có các phần tử dạng chuỗi.

b) Cú pháp:

s = str2mat('s1', 's2', …)

c) Giải thích:

s: tên ma trận kết quả.

s1, s2: chuỗi ký tự.

11. Lênh STR2NUM

a) Công dung:

Chuyển chuỗi (dạng số) sang số thực.

b) Cú pháp:

n = str2num(s)

c) Giải thích:

s: chuỗi dạng số.

n: số thực.

12. Lênh STRCMP

a) Công dung:

So sánh 2 chuỗi ký tự.

b) Cú pháp:

l = strcmp(s1, s2)

c) Giải thích:

l: biến chứa kết quả.

s1, s2: chuỗi cần so sánh.

13. Lênh UPPER

a) Công dung:

Cho ra chuỗi viết hoa.

b) Cú pháp:

b = upper

c) Giải thích:

b: biến chứa kết quả.

s: tên biến chứa chuỗi ký tự.

VI. CÁC HÀM TOÁN HỌC CƠ BẢN

1. **Một số hàm lương giác:**

a) Cú pháp:

kq = hlg(x)

b) Giải thích:

kq: tên biến chứa kết quả.

x: đơn vị radian.

hlg: tên hàm lượng giác.

Tên hàm lượng giác	Giải thích
sin	Tính giá trị sine
cos	Tính giá trị cosine
tan	Tính giá trị tangent
asin	Nghịch đảo của sine
atan	Nghịch đảo của tangent
sinh	Tính giá trị hyperbolic sine
cosh	Tính giá trị hyperbolic cosine
tanh	Tính gía trị hyperbolic tangent

2. **Lênh ANGLE**

a) Công dụng:

Tính góc pha của số phức.

b) Cú pháp:

p = angle(z)

c) Giải thích:

p: tên biến chứa kết quả, đơn vị radians

z: số phức

3. **Lênh CEIL**

a) Công dụng:

Làm tròn số về phía số nguyên lớn hơn.

b) Cú pháp:

y = ceil(x)

c) Giải thích:

y: số sau khi được làm tròn.

x: số cần được làm tròn.

4. **Lênh CONJ**

a) Công dung:

Tính lượng liên hiệp của số phức.

b) Cú pháp:

y = conj(z)

c) Giải thích:

y: tên biến chứa lượng liên hiệp

z: số phức

5. **Lênh EXP**

a) Công dung:

Tính giá trị ex.

b) Cú pháp:

y = exp(x)

6. **Lênh FIX**

a) Công dung:

Làm tròn số về phía zero.

b) Cú pháp:

y = fix(x)

c) Giải thích:

y: số sau khi được làm tròn.

x: số cần được làm tròn.

7. **Lênh FLOOR**

a) Công dung:

Làm tròn số về phía số nguyên nhỏ hơn.

b) Cú pháp:

y = floor(x)

c) Giải thích:

y: số sau khi được làm tròn .

x: số cần được làm tròn

8. **Lênh IMAG**

a) Công dung:

Lấy phần ảo của số phức.

b) Cú pháp:

y = imag(z)

9. **Lênh LOG**

a) Công dung:

Tìm logarithm cơ số e.

b) Cú pháp:

y = log(x)

10. **Lệnh LOG2**

a) Công dụng:

Tìm logarithm cơ số 2.

b) Cú pháp:

y = log2(x)

11. **Lệnh LOG10**

a) Công dụng:

Tìm logarithm cơ số 10.

b) Cú pháp:

y = log10(x)

12. **Lệnh REAL**

a) Công dụng:

Lấy phần thực của số phức.

b) Cú pháp:

y = real(z)

13. **Lệnh REM**

a) Công dụng:

Cho phần dư của phép chia.

b) Cú pháp:

r = rem(a,b)

c) Giải thích:

r: biến chứa kết quả

a, b: số chia và số bị chia

14. **Lệnh ROUND**

a) Công dụng:

Làm tròn số sao cho gần số nguyên nhất.

b) Cú pháp:

y = round(x)

15. **Lệnh SIGN**

a) Công dụng:

Xét dấu số thực.

b) Cú pháp:

y = sign(x)

c) Giải thích:

x: số thực cần xét dấu.

y: kết quả trả về.

y	x
0	số 0
1	số dương
-1	số âm

VII. TẬP LỆNH THAO TÁC TRÊN MA TRẬN

1. Cộng, trừ, nhân, chia từng phần tử của ma trân với hằng số

a) Cú pháp:

Ma trận kết quả = ma trận [+] [-] [.] [/] hằng số.

2. Lênh DET

a) Công dung:

Dùng để tính định thức của ma trận.

3. Lênh DIAG

a) Công dụng:

Tạo ma trận mới và xử lý đường chéo theo quy ước.

b) Cú pháp:

v = diag(x)

v = diag(x,k)

c) Giải thích:

x: là vector có n phần tử.

v: là ma trận được tạo ra từ x theo quy tắc: số hàng bằng số cột và các phần tử của x nằm trên đường chéo của v.

k: tham số định dạng cho v, số hàng và cột của v = n + abs(k).

Nếu k = 0 đường chéo của v chính là các phần tử của x

Nếu k > 0 các phần tử của x nằm phía trên đường chéo v

Nếu k < 0 các phần tử của x nằm phía dưới đường chéo v

4. Lênh EYE

a) Công dung:

Tạo ma trận đơn vị.

b) Cú pháp:

y = eye(n)

y = eye(n,m)

c) Giải thích:

n: tạo ma trận có n hàng, n cột.

m, n: tạo ma trận có m hàng, n cột.

5. Lệnh FLIPLR

a) Công dụng:

Chuyển các phần tử của các ma trận theo thứ tự cột ngược lại.

b) Cú pháp:

b = fliplr(a)

c) Giải thích:

b: tên ma trận được chuyển đổi.

a: tên ma trận cần chuyển đổi.

6. Lệnh FLIPUD

a) Công dụng:

Chuyển các phần tử của ma trận theo thứ tự hàng ngược lại.

b) Cú pháp:

b = flipud(a)

c) Giải thích:

b: tên ma trận được chuyển đổi.

a: tên ma trận cần chuyển đổi.

7. Lệnh INV

a) Công dụng:

Tìm ma trận nghịch đảo.

b) Cú pháp:

Ma trận nghịch đảo = inv (ma trận)

8. Lệnh tao ma trân

a) Công dụng:

Dùng để tạo 1 ma trận gồm có n hàng và m cột.

b) Cú pháp:

Tên ma trận = [a_{11} a_{12}…a_{1m} ; a_{21} a_{22}… a_{2m} ;…;…]

c) Giải thích:

a_{11}, a_{12}, a_{1m} là các giá trị tại hàng 1 cột 1 đến các giá trị tại hàng 1 cột m, có n dấu (;) là có n hàng.

9. Lệnh tao vector đơn

a) Công dụng:

Lệnh này dùng để tạo 1 vector đơn gồm có n phần tử.

b) Cú pháp 1:

Tên vector = [pt1 pt2 pt3 …ptn]

c) Giải thích:

pt1 pt2 …ptn: là các số thực.

d) Cú pháp 2:

Tên vector = gtđ:csc:gtkt

e) Giải thích:

gtđ: là giá trị bắt đầu của vector.

csc: cấp số cộng.

gtkt: giá trị kết thúc.

10. Lệnh LINSPACE

a) Công dụng:

Tạo vector có giá trị ngẫu nhiên giới hạn trong khoảng định trước.

b) Cú pháp:

y = linspace(x1, x2)

y = linspace(x1, x2, n)

c) Giải thích:

y: tên của vector.

x1, x2: giới hạn giá trị lớn nhất và nhỏ nhất của vector y.

n: số phần tử của vector y.

Nếu không có giá trị n thì mặc định n = 100.

11. Ma trân chuyển vi

a) Công dụng:

Ma trận chuyển vị = ma trận đang có.

b) Cú pháp:

Tạo 1 ma trận chuyển vị từ 1 ma trận đang có.

12. Lệnh MAGIC

a) Công dụng:

Tạo 1 ma trận vuông có tổng của các phần tử trong 1 hàng, 1 cột hoặc trên đường chéo bằng nhau.

b) Cú pháp:

Tên ma trận = magic(n)

c) Giải thích:

n: kích thước ma trận.

Giá trị của mỗi phần tử trong ma trận là một dãy số nguyên liên tục từ 1 đến 2^n.

Tổng các hàng, cột và các đường chéo đều bằng nhau.

13. Nhân ma trân

a) Công dụng:

Ma trận kết quả = ma trận 1* ma trận 2.

14. Lênh ONES

a) Công dụng:

Tạo ma trận mà giá trị của các phần tử là 1.

b) Cú pháp:

y = ones(n)

y = ones(m,n)

c) Giải thích:

y = tên ma trận.

n: tạo ma trận có n hàng

m, n: tạo ma trận có m hàng, n cột.

15. Lênh PASCAL

a) Công dụng:

Tạo ma trận theo quy luận tam giác Pascal.

b) Cú pháp:

pascal (n)

c) Giải thích:

n: là số hàng (cột)

16. Lênh RAND

a) Công dụng:

Tạo ma trận mà kết mà giá trị của các phần tử là ngẫu nhiên.

b) Cú pháp:

y = rand(n)
y = rand(m,n)

c) Giải thích:

y: tên ma trận.

n: tạo ma trận có n hàng, n cột.

m, n: tạo ma trận có m hàng, n cột.

Giá trị của các phần tử nằm trong khoảng [0 1]

17. Lênh RESHAPE

a) Công dụng:

Định dạng lại kích thước ma trận.

b) Cú pháp:

b = reshape(a,m,n)

c) Giải thích:

b: ma trận được định dạng lại.

a: ma trận cần được định dạng.

m, n: số hàng và số cột của b.

Ma trận a phải có số phần tử là: m*n.

18. Lênh ROT90

a) Công dung:

Xoay ma trận 90^0.

b) Cú pháp:

b = rot90(a)

c) Giải thích:

b: ma trận đã được xoay 90^0

a: ma trận cần xoay.

19. Lênh TRACE

a) Công dung:

Tính tổng các phần tử của đường chéo ma trận.

b) Cú pháp:

d = trace(a)

c) Giải thích:

d: biến chứa kết quả.

a: tên ma trận.

20. Lênh TRIL

a) Công dung:

Lấy phân nửa dưới ma trận theo hình.

b) Cú pháp:

I = tril(x)

I = tril(x,k)

c) Giải thích:

I: tên ma trận kết quả.

k: tham số.

Nếu k = o lấy từ đường chéo trở xuống.

Nếu k = n lấy từ đường chéo trở lên n đơn vị.

Nếu k = -n lấy từ đường chéo trở xuống n đơn vị.

21. Lênh TRIU

a) Công dung:

Lấy phân nửa trên ma trận theo hình tam giác.

b) Cú pháp:

I = triu(x)

I = triu(x,k)

c) Giải thích:

I: tên ma trận kết qủa.

k: tham số

Nếu k = 0 lấy từ đường chéo trở lên.

Nếu k = n lấy từ đường chéo trở xuống n đơn vị.

Nếu k = -n lấy từ đường chéo trở lên n đơn vị.

22. Lênh ZEROS

a) Công dung:

Tạo ma trận mà giá trị của các phần tử

b) Cú pháp:

y = zeros(n)

y = zeros(m,n)

c) Giải thích:

y: tên ma trận.

n: tạo ma trận có n hàng và n cột.

m, n: tạo ma trận có m hàng, n cột.

16. Lênh SQRT

a) Công dung:

Tính căn bậc hai.

b) Cú pháp:

y = sqrt(x)

VII. TẬP LỆNH ĐỒ HỌA

1. Lênh AXES

a) Công dung:

Đặt các trục tọa độ tại vị trí định trước.

b) Cú pháp:

axes('propertyname', propertyvalue …)

c) Giải thích:

Tương ứng với một propertyname đi kèm với 1 propertyvalue.

1. 'position',[left, bottom, width, height]: định vị trí và kích thước của trục.

left: khoảng cách từ mép trái cửa sổ đến trục đứng.

bottom: khoảng cách từ mép dưới cửa sổ đến trục ngang.

width: chiều dài của trục ngang.

height: chiều cao trục đứng.

Ghi chú:

Luôn lấy điểm [0,0] làm gốc tọa độ.

Trục ngang và trục đứng có giá trị trong khoảng [0 1] và chia theo tỷ lệ thích hợp

2. Lệnh AXIS

a) Công dụng:

Chia lại trục tọa độ.

b) Cú pháp:

axis([xmin xmax ymin ymax])

axis([xmin xmax ymin ymax zmin zmax])

axis on

axis off

c) Giải thích:

xmin, ymin, zmin: là giá trị nhỏ nhất của các trục x, y, z.

xmax, ymax, zmax: là giá trị lớn nhất của các trục x, y, z.

on: cho hiển thị trục tọa độ.

off: không cho hiển thị trục tọa độ.

3. Lệnh BAR

a) Công dụng:

Vẽ đồ thị dạng cột.

b) Cú pháp:

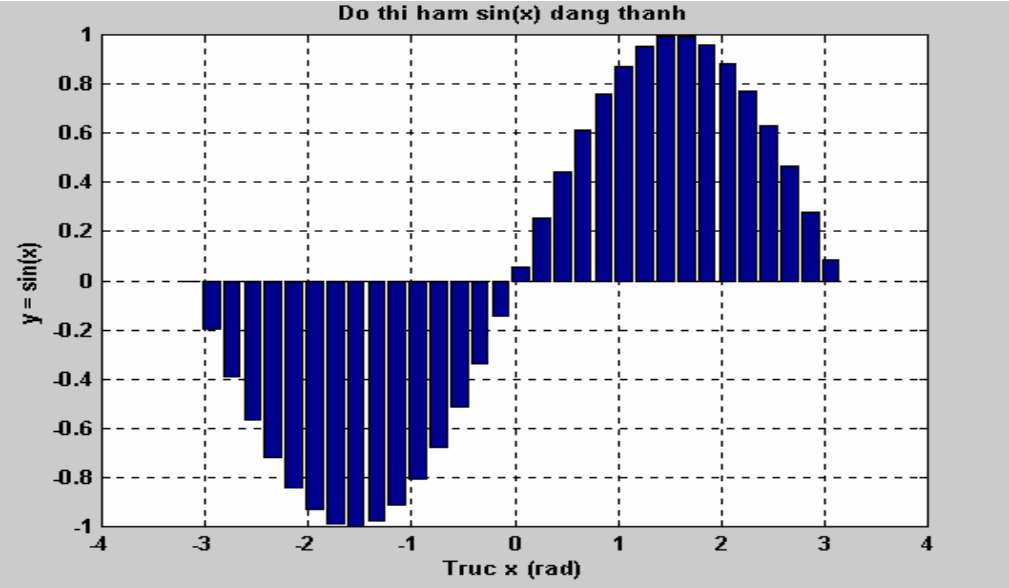
bar(x,y)

c) Giải thích:

Vẽ giá trị x theo giá trị y.

d) Ví dụ:

```
x = -pi:0.2:pi;
bar(x,sin(x));
grid on
title('Do thi ham sin(x) dang thanh')
xlabel('truc x (rad)')
ylabel('y = sin(x)')
```



4. Lênh CLA

a) Công dung:

Xóa tất cả các đối tượng như: đường đồ thị, tên đồ thị…nhưng không xóa trục tọa độ.

b) Cú pháp:

cla

5. Lênh CLF

a) Công dung:

Xóa hình ảnh (đồ thị) hiện tại.

b) Cú pháp:

clf

6. Lênh CLOSE

a) Công dung:

Đóng hình ảnh (đồ thị) hiện tại.

b) Cú pháp:

close

7. Lênh COLORMAP

a) Công dung:

Tạo màu sắc cho đồ thị trong không gian 3 chiều.

b) Cú pháp:

colormap(map)

colormap('default')

c) Giải thích:

Colormap là sự trộn lẫn của 3 màu cơ bản: red, green, blue. Tùy theo tỷ lệ của 3 màu cơ bản mà cho ra các màu sắc khác nhau.

'default': màu có được là màu mặc định.

map: biến chứa các thông số sau:

Map	màu có được
Bone	gray + blue
Cool	cyan + magenta
Flag	red + white + blue + black
Gray	gray
Hot	black + red + yellow + white
Pink	pink

8. Lênh FIGURE

a) Công dung:

Tạo mới hình ảnh (đồ thị).

b) Cú pháp:

figure

9. Lênh GCA

a) Công dung:

Tạo các đặc tính cho trục.

b) Cú pháp:

h = gca

c) Giải thích:

h: là biến gán cho lệnh cga.

Các đặc tính của trục gồm có:

Cú pháp	Giải thích
Set(gca,'XScale','log', 'Yscale','linear')	Định đơn vị trên trục tọa độ: trục x có đơn vị là log và trục y có đơn vị tuyến tính.
Set(gca,'Xgrid','on','YGrid', 'nomal')	Tạo lưới cho đồ thị: trục x có tạo lưới và trục y không tạo lưới.
Set(gca,'XDir','reverse', 'YDir','normal')	Đổi trục tọa độ: đổi trục x về phía đối diện, trục y giữ nguyên.
Set(gca,'XColor','red', 'Ycolor','yellow')	Đặt màu cho lưới đồ thị: đặt lưới trục x màu đỏ, lưới trục y màu vàng. Gồm có các màu: yellow, magenta, cyan, red, green, blue, white, black.

10. Lênh GRID

a) Công dung:

Tạo lưới tọa độ.

b) Cú pháp:

grid on

grid off

c) Giải thích:

on: hiển thị lưới tọa độ.

off: không hiển thị lưới tọa độ.

11. Lênh PLOT

a) Công dung:

Vẽ đồ thị tuyến tính trong không gian 2 chiều.

b) Cú pháp:

plot(x,y)

plot(x,y,'linetype')

c) Giải thích:

x,y: vẽ giá trị x theo giá trị y.

linetype: kiểu phần tử tạo nên nét vẽ bao gồm 3 thành phần:

- Thành phần thứ nhất là các ký tự chỉ màu sắc:

Ký tự	Màu
y	Vàng
m	Đỏ tươi
c	Lơ
r	Đỏ
g	Lục
b	Lam
w	Trắng
k	Đen

- Thành phần thứ hai là các ký tự chỉ nét vẽ của đồ thị:

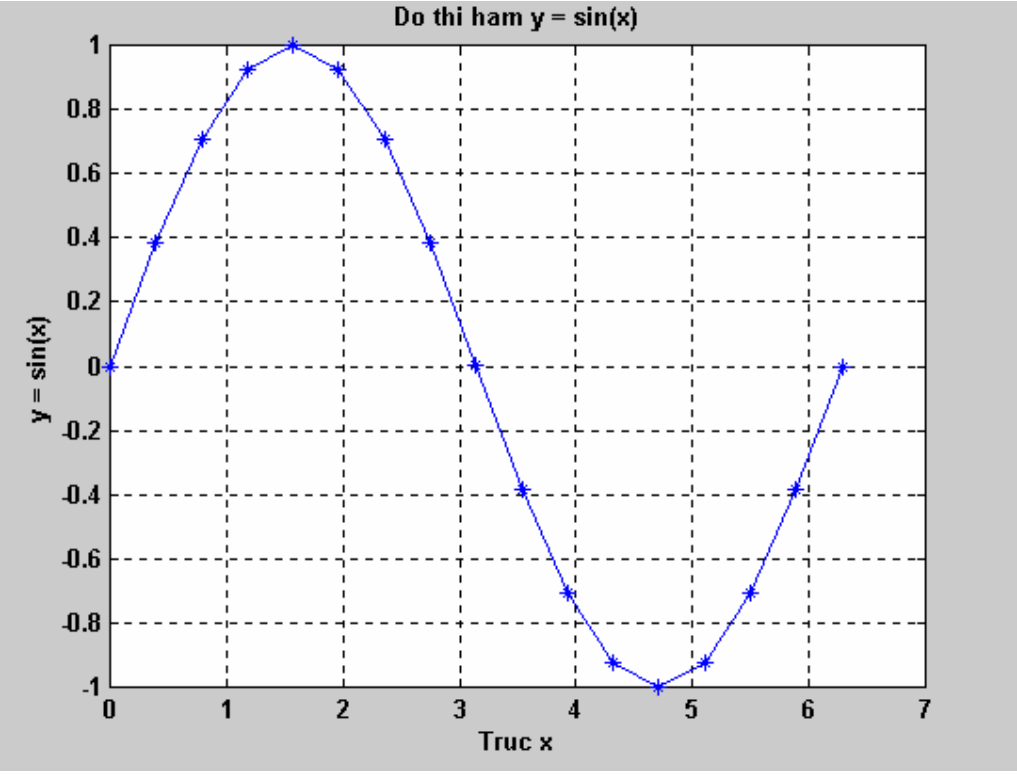
Ký tự	Loại nét vẽ
-	Đường liền nét
:	Đường chấm chấm
-.	Đường gạch chấm
--	Đường nét đứt đoạn

- Thành phần thứ ba là các ký tự chỉ loại điểm đánh dấu gồm:., o, x, +, *

d) Ví dụ:

Vẽ đồ thị hàm y = sin(x) với đồ thị màu lam, đường liền nét và đánh dấu các điểm được chọn bằng dấu *, trục x thay đổi từ 0 tới 2π, mỗi bước thay đổi là π/8

```
x = 0:pi/8:2*pi;
y = sin(x);
plot(x,y, 'b-* ')
ylabel('y = sin(x)')
xlabel('Truc x')
title('Do thi ham y = sin(x)')
grid on
```



12. Lênh SUBPLOT

a) Công dung:

Tạo các trục trong một phần của cửa sổ đồ họa.

b) Cú pháp:

subplot(m,n,p)

subplot(mnp)

c) Giải thích:

subplot(m,n,p) hoặc subplot(mnp)thành cửa sổ đồ họa thành m×n vùng để vẽ nhiều đồ thị trên cùng một cửa sổ.

m: số hàng được chia.

n: số cột được chia

p: số thứ tự vùng chọn để vẽ đồ thị.

Nếu khai báo p > m×n thì sẽ xuất hiện một thông báo lỗi.

d) Ví du:

Chia cửa sổ đồ họa thành 2×3 vùng và hiển thị trục của cả 6 vùng.

subplot(231)

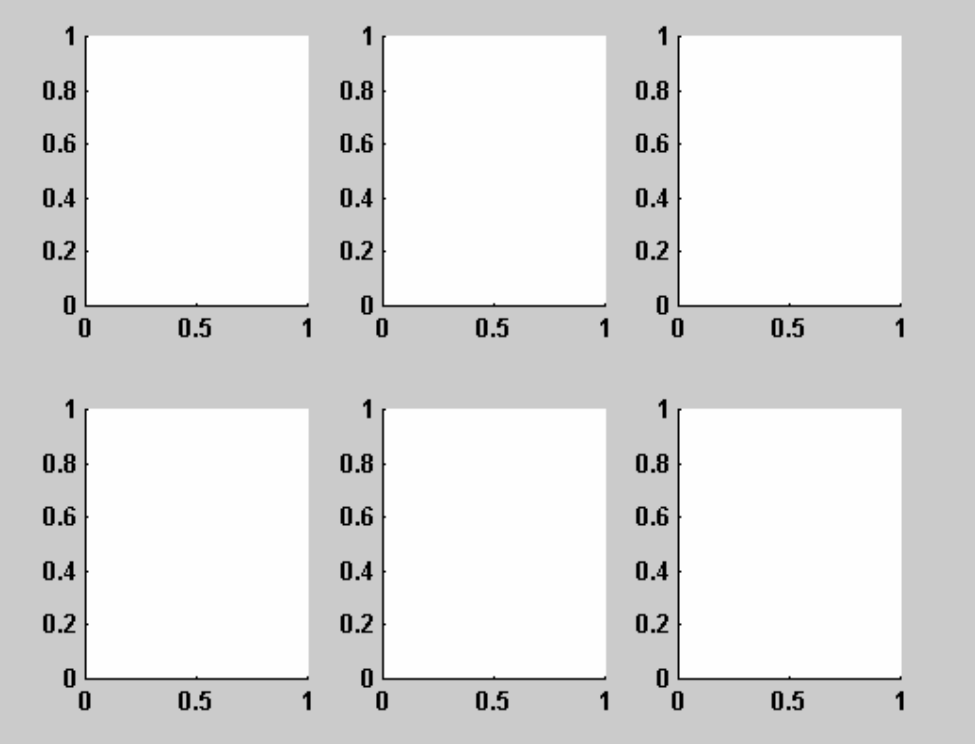
subplot(232)

subplot(233)

subplot(234)

subplot(235)

subplot(236)



13. Lệnh SEMILOGX, SEMILOGY

a) Công dung:

Vẽ đồ thị theo logarith.

b) Cú pháp:

semylogx(x,y)

semylogx(x,y,'linetype')

semylogy(x,y)

semylogy(x,y,'linetype')

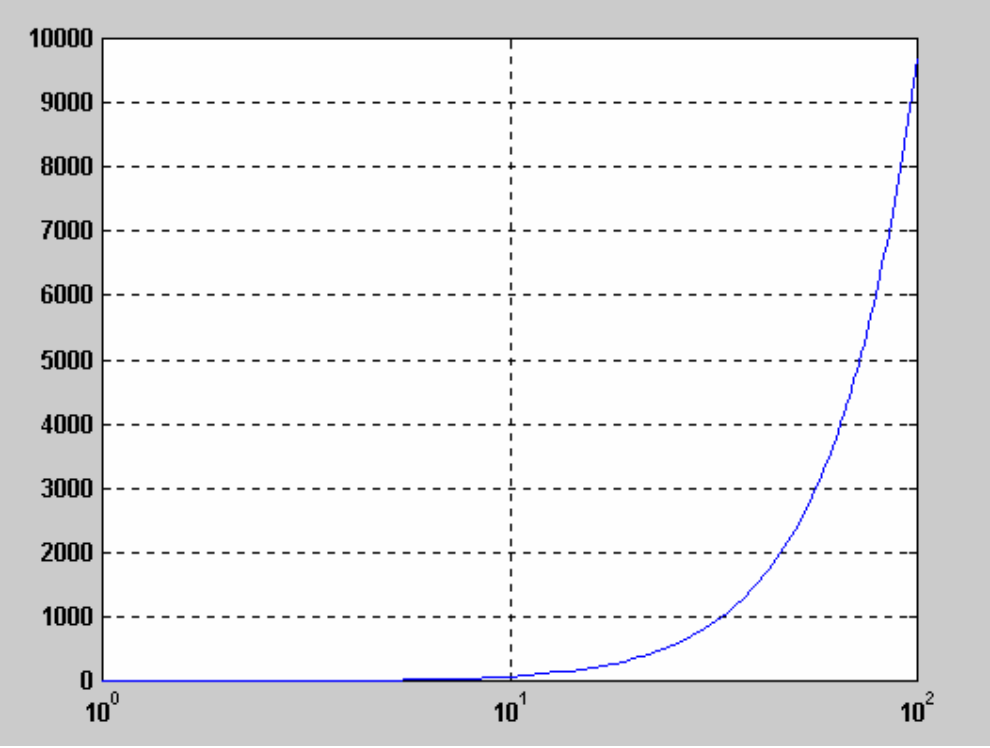
c) Giải thích:

semylogx và semylogy giống như lệnh plot nhưng chỉ khác một điều là lệnh này vẽ đồ thị theo trục logarith. Do đó, ta có thể sử dụng tất cả các loại 'linetype' của lệnh plot.

d) Ví dụ:

Vẽ đồ thị hàm $y = x^2 - 3x + 2$ theo trục logarith của x.

```
x = 0:100;
y = x.^2-3*x+2;
semylogx(x,y,'b')
grid on
```



14. Lệnh POLAR

a) Công dung:

Vẽ đồ thị trong hệ trục tọa độ cực.

b) Cú pháp:

polar(theta,rno)

c) Giải thích:

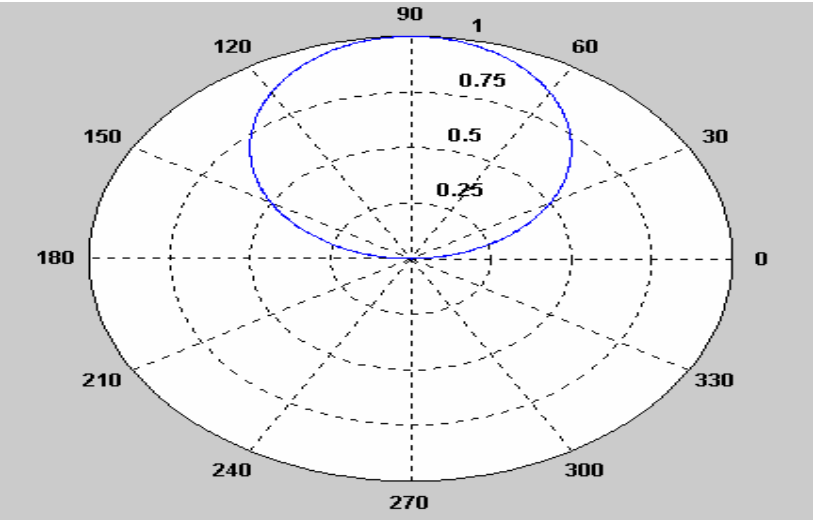
Vẽ giá trị x theo giá trị y.

d) Ví dụ:

t = -pi:0.01:pi;

polar(t, sin(t))

Và ta thu được dạng đồ thị sau:



15. Lệnh SET

a) Công dung:

Thiết lập các đặc tính chất cho đối tượng nào đó.

b) Cú pháp:

set(h, ‘propertyname’, propertyvalue,…)

c) Giải thích:

h: biến chứa đối tượng.

PropertyName và PropertyValue được cho trong bảng sau:

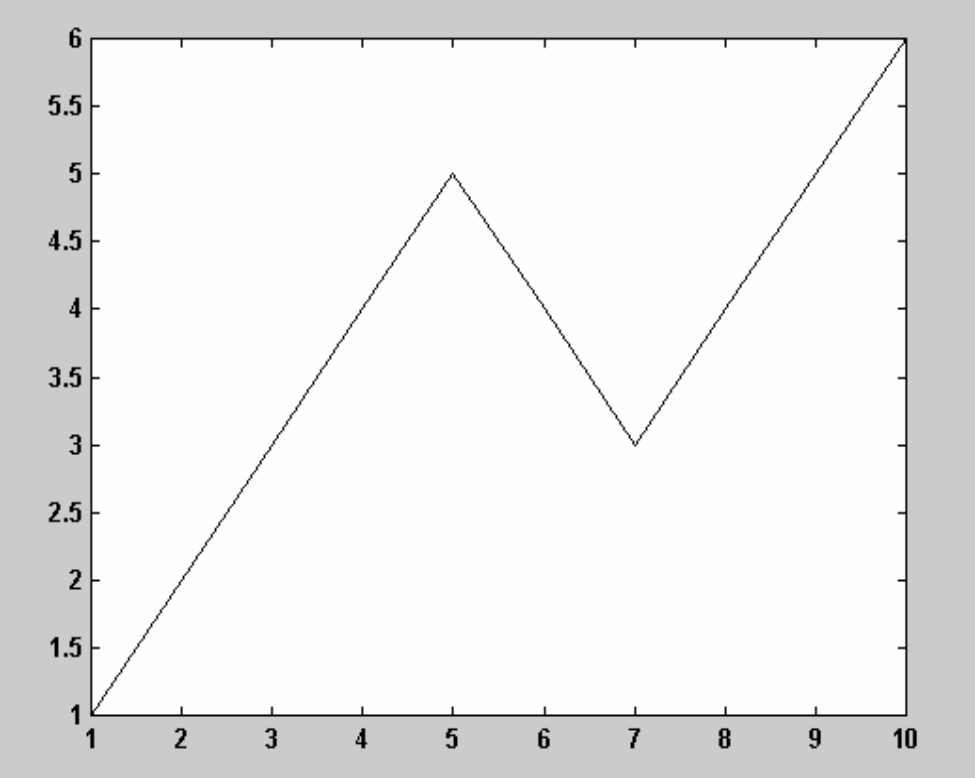
Cú pháp	PropertyName	PropertyValue	Giải thích
Set(h,’Marker’,’+’)	Marker	-, --, :, -. , o, ×, +, *	Chọn kiểu phần tử
Set(h,’LineWidth’,1)	LineWidth	1, 2, 3,…	Độ dày nét vẽ
Set(h,’MarkerSize’,9)	MarkerSize	1, 2, 3,…	Kích thước các điểm tạo nên h
Set(h,’color’,’cyan’)	Color	yellow,magenta, red,green,blue, cyan,white,black	**Chọn màu cho đối tượng h**

d) Ví du:

a = [1 2 3 4 5 4 3 4 5 6];

h = plot(a)

set(h,'color','black')



16. Lênh STAIRS

a) Công dung:

Vẽ đồ thị dạng bậc thang.

b) Cú pháp:

stairs(x,y)

c) Giải thích:

Vẽ giá trị x theo giá trị y.

d) Ví du:

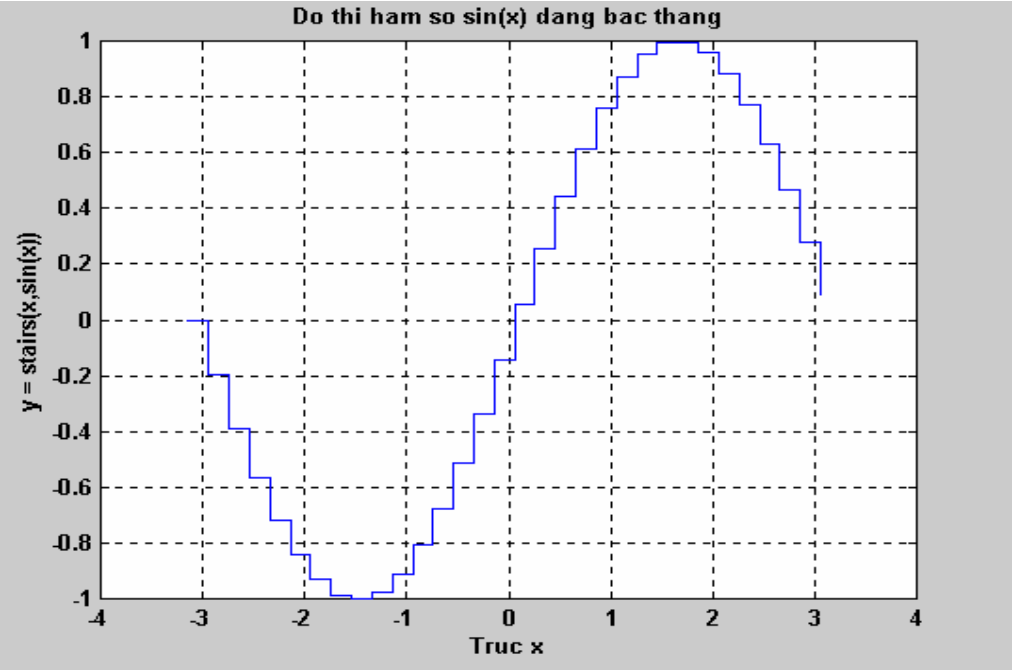
x = -pi:0.2:pi;

stairs(x,sin(x))

xlabeL('Truc x')

ylabel('y = stairs(x,sin(x)')

grid on



17. Lênh TITLE

a) Công dung:

Đặt tiêu đề cho đồ thị.

b) Cú pháp:

title('text')

c) Giải thích:

text: tên tiêu đề.

18. Lênh XLABEL, YLABEL, ZLABEL

a) Công dung:

Đặt tên cho trục X, Y, Z.

b) Cú pháp:

xlabel('nx')

ylabel('ny')

zlabel('nz')

c) Giải thích:

nx, ny, nz: tên trục x, y, z

19. Lênh WHITEBG

a) Công dung:

Thay đổi màu nền của cửa sổ đồ họa.

b) Cú pháp:

whitebg

whitebg('color')

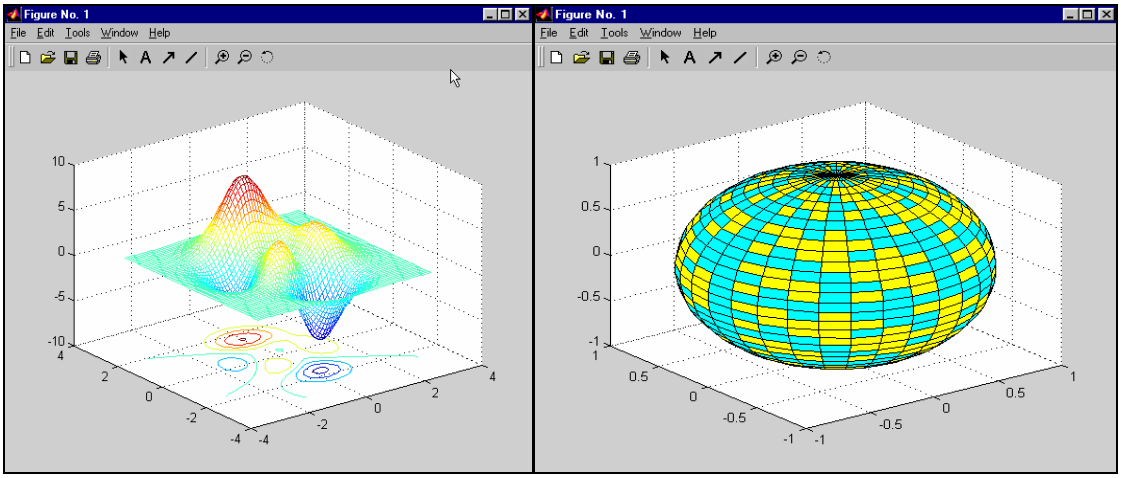
c) Giải thích:

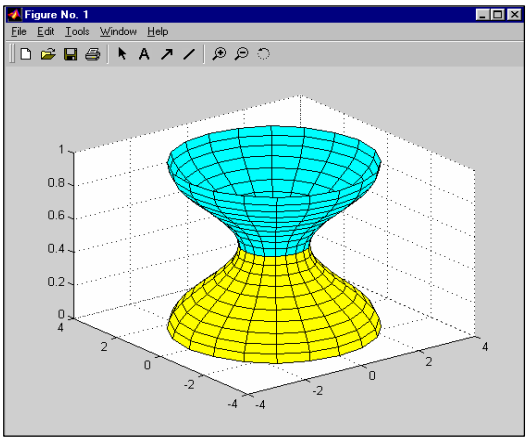
whitebg chuyển đổi qua lại màu nền cửa sổ đồ họa giữa trắng và đen.

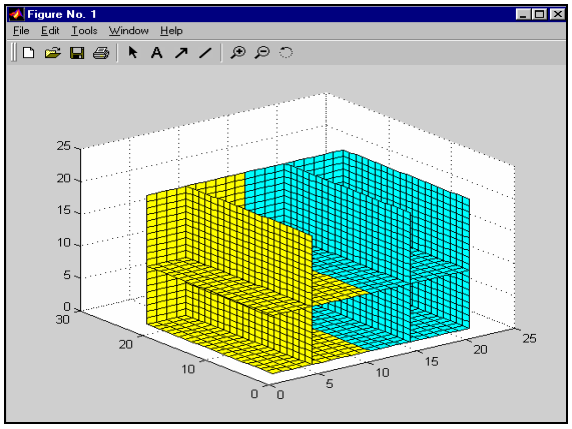
whitebg('color') chuyển màu nền cửa sổ đồ họa thành màu của biến color.

color có thể là các màu: yellow (vàng), magenta (đỏ tươi), cyan (lơ), red (đỏ), green (lục), blue (lam), white (trắng), black (đen).

MATLAB còn vẽ được các đồ thị như sau :







VII. VẼ GIẢN ĐỒ BODE, NYQUIST, NICHOLS

LÝ THUYẾT:

Giản đồ Bode gồm hai đồ thị: Đồ thị logarith biên độ của hàm truyền và góc pha theo logarith tần số. (một đơn vị ở trục hoành gọi là một decade).

Biên độ : $|G(j\omega)|_{dB} = 20 \log_{10} |G(j\omega)|$ (2.22)

Pha : $\varphi = \angle G(j\omega)$ (hay arg G(jω)) (2.23)

Giản đồ Bode của các khâu cơ bản:

* Khâu khuếch đại:

Hàm truyền đạt G(s) = K

Giản đồ Bode L(ω) = 20 lgM(ω) = 20 lgK là 1 đường thẳng song song với trục hoành.

* Khâu quán tính bậc 1:

Hàm truyền đạt $G(s) = \frac{K}{Ts+1}$

Biểu đồ Bode $L(\omega) = 20 \lg M(\omega) = 20 \lg K - 20\lg\sqrt{T^2\omega^2+1}$ có độ dốc giảm –20dB/decade

* Khâu vi phân bậc 1:

Hàm truyền đạt G(s) = K(Ts + 1)

Giản đồ Bode $L(\omega) = 20 \lg M(\omega) = 20 \lg K + 20\lg\sqrt{T^2\omega^2+1}$ có độ dốc tăng 20dB/decade

* Khâu tích phân:

Hàm truyền đạt $G(s) = \frac{K}{s}$

Giản đồ Bode L(ω) = 20 lgM(ω) = 20 lgK – 20lgω

* Khâu bậc 2:

Hàm truyền đạt $G(s) = \frac{\omega_n^2}{s^2 + 2\varepsilon\omega_n s + \omega_n^2}$

Giản đồ Bode $L(\omega) = -20\lg\sqrt{\left(1-\omega^2t^2\right)^2 + 4\varepsilon^2\omega^2t^2}$

VÍ DỤ

Vẽ giản đồ Bode hệ thống hồi tiếp đơn vị của hàm truyền vòng hở sau:

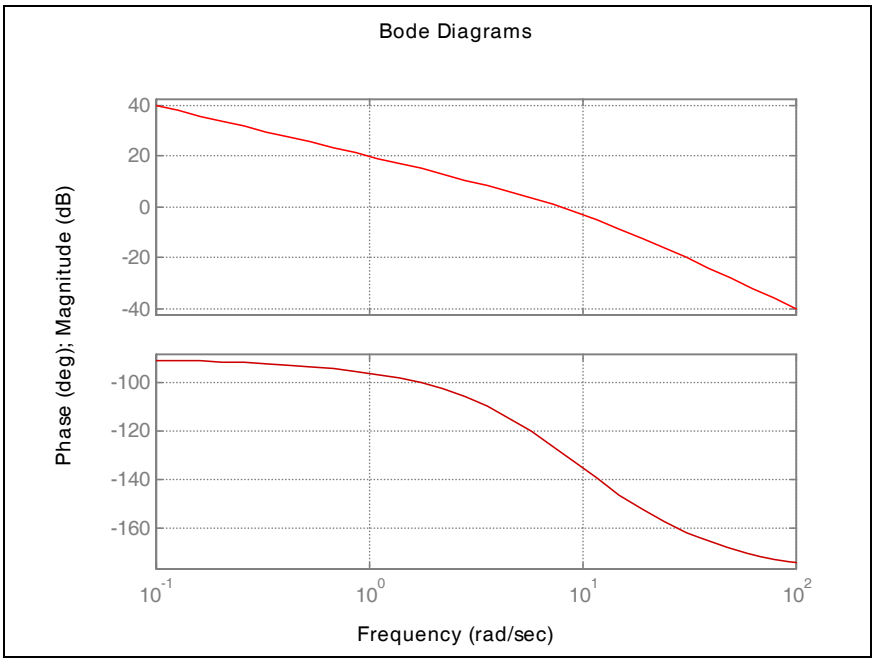
$$G(s) = \frac{10}{s(1+0.1s)}$$

>> num = 10;

>> den = [0.1 1 0];

>> bode(num,den)

Kết quả:



Hệ thống gồm 1 khâu khuếch đại bằng 10, một khâu tích phân và một khâu quán tính bậc 1

Tần số gãy: 10.

$|G(jw)|_{dB} = 20dB - 20\log\omega$

Tại tần số $\omega = 1rad/sec$ $|G(jw)|_{dB} = 20dB$ và độ dốc –20dB/decade (do khâu tích phân).

Độ dốc –20dB/decade tiếp tục cho đến khi gặp tần số cắt $\omega = 10rad/sec$, tại tần số này ta cộng thêm –20dB/decade (do khâu quán tính bậc nhất) và tạo ra độ dốc -40dB/dec.

Vẽ giản đồ Bode hệ thống hồi tiếp đơn vị của hàm truyền vòng hở sau:

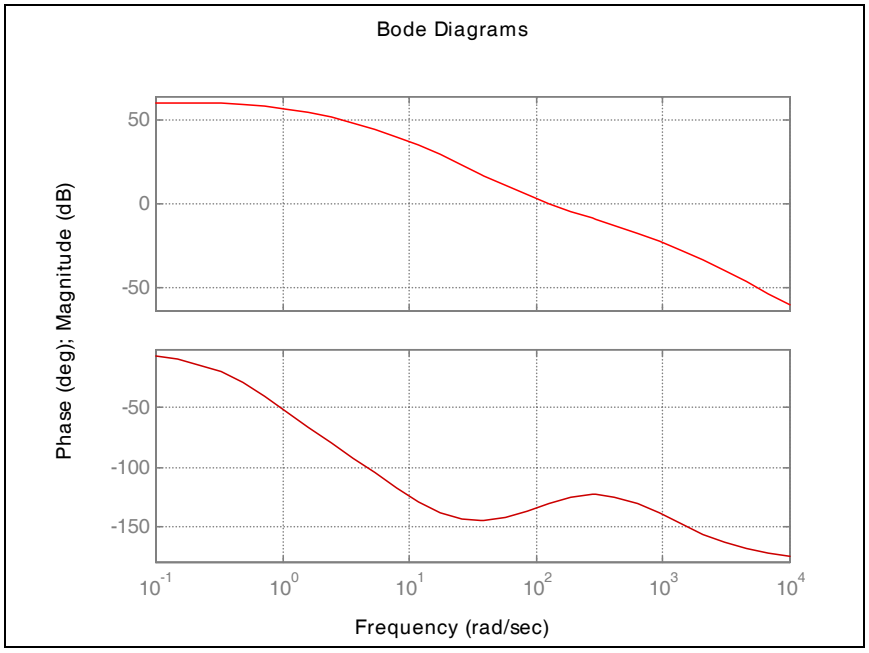
$$G(s) = \frac{10^5(s+100)}{(s+1)(s+10)(s+1000)}$$

>> num = 100000*[1 100];

>> den = [1 1011 11010 10000];

>> bode(num,den)

Kết quả:



Hệ thống gồm một khâu khuếch đại 10^5, một khâu vi phân bậc nhất và 3 khâu quán tính bậc 1.

Tần số gãy: 1,10,100,1000.

$|G(jw)|_{dB|w=0} = 60dB$

Tại tần số gãy ω = 1rad/sec có độ lợi 60dB và độ dốc –20dB/decade (vì khâu quán tính bậc 1). Độ dốc –20dB/decade được tiếp tục đến khi gặp tần số gãy ω = 10rad/sec tại đây ta cộng thêm -20dB/decade(vì khâu quán tính bậc 1), tạo ra độ dốc –40dB/dec. Độ dốc - 20dB ở tần số ω = 100rad/dec (do khâu vi phân bậc 1). Tại tần số gãy ω = 100rad/sec tăng 20dB (vì khâu vi phân bậc 1). Tạo ra độ dốc có độ dốc -20dB.

Tại tần số gãy ω = 1000rad/sec giảm 20dB (vì khâu quán tính bậc 1). Tạo ra độ dốc - 40dB.

Vẽ giản đồ Bode hệ thống hồi tiếp đơn vị của hàm truyền vòng hở sau:

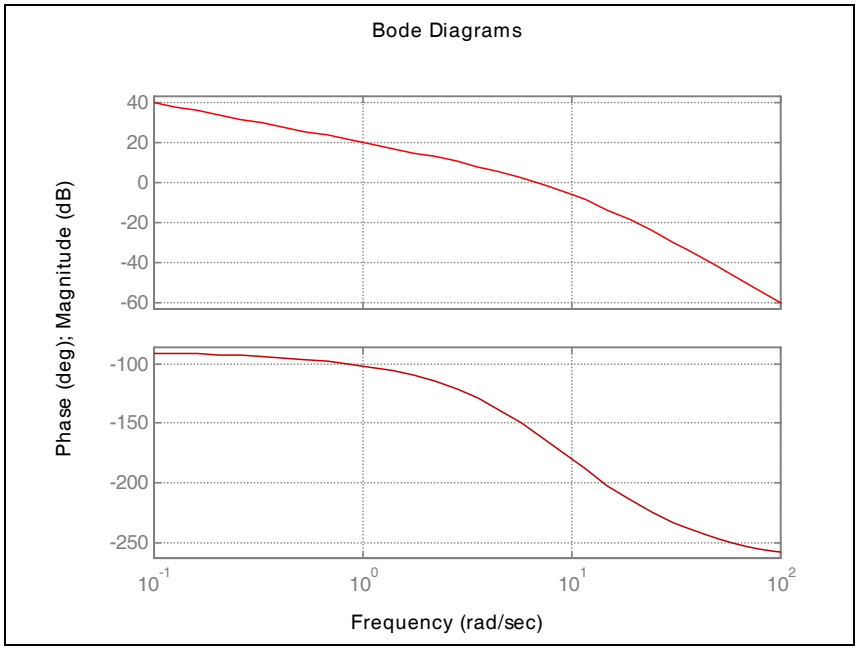
$$G(s) = \frac{10}{s(1+0.1s)^2}$$

>> num = 10;

>> den = [0.01 0.2 1 0];

>> bode(num,den)

Kết quả:



Hệ thống gồm một khâu khuếch đại 10, một khâu tích phân và 1 thành phần cực kép.

Tần số gãy: 10.

$|G(jw)|_{dB} = 20dB - 20\log\omega$

Tần số gãy nhỏ nhất ω = 0.1 rad/sec tại tần số này có độ lợi 40dB và độ dốc –20dB (do khâu tích phân). Độ dốc này tiếp tục cho tới tần số gãy kép ω = 10. ở tần số này sẽ giảm 40dB/decade, tạo ra độ dốc –60dB/dec.

Vẽ giản đồ Bode hệ thống hồi tiếp đơn vị của hàm truyền vòng hở sau:

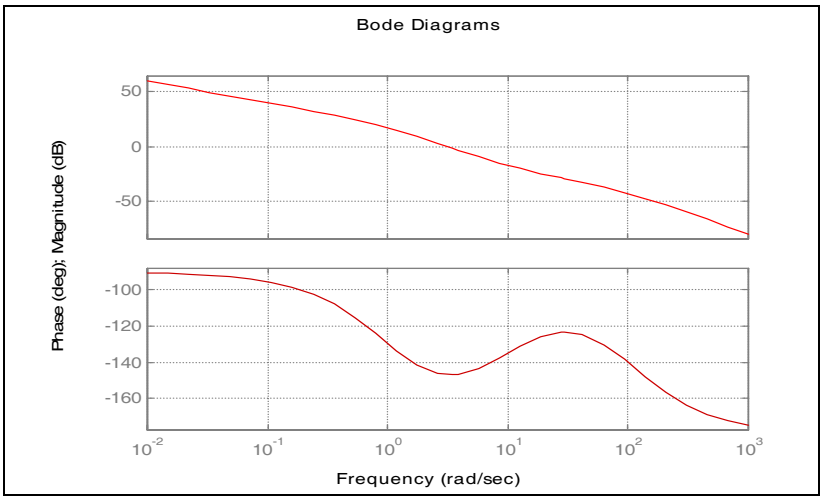
$$G(s) = \frac{10^2(s+10)}{s(s+1)(s+100)}$$

>> num = 100*[1 10];

>> den = [1 101 100 0];

>> bode(num,den)

Kết quả:



Hệ thống gồm một khâu khuếch đại 100, một khâu tích phân và 2 khâu quán tính bậc 1, 1 khâu vi phân.

Tần số gãy: 1,10,100

$|G(jw)|_{dB|w=0} = 20\log 10 - 20\log\omega$

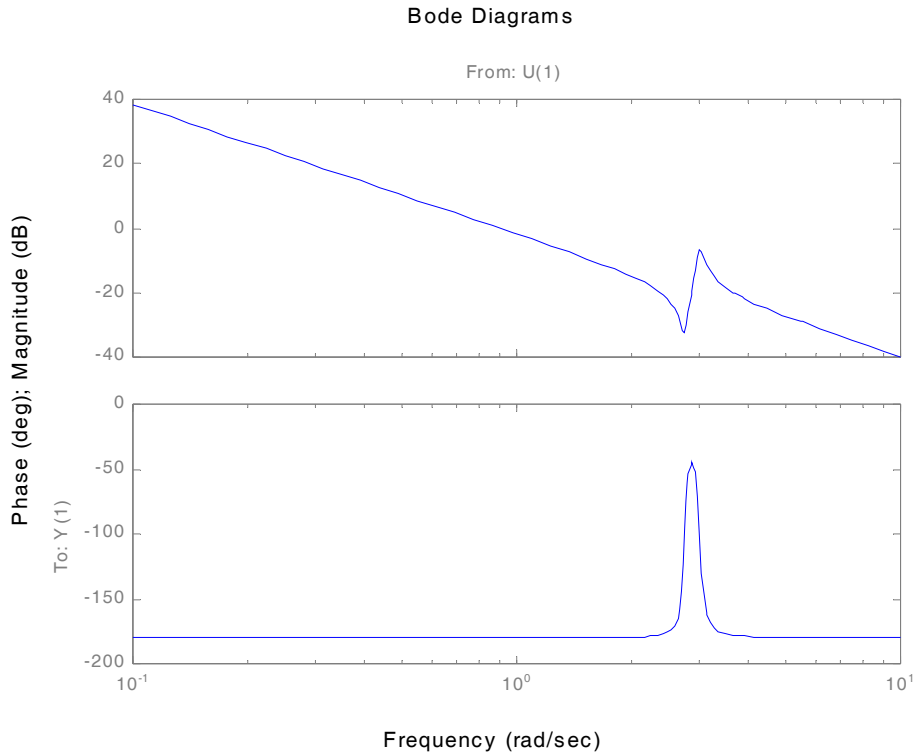
Ta chỉ xét trước tần số gãy nhỏ nhất 1decade. Tại tần số gãy ω = 0.1rad/sec có độ lợi 40dB và độ dốc –20dB/dec, độ dốc –20dB/dec tiếp tục cho đến khi gặp tần số gãy ω = 1rad/sec, ta cộng thêm –20dB/dec (vì khâu quán tính bậc 1) và tạo ra độ dốc –40dB/dec. Tại tần số ω =10 sẽ tăng 20dB/dec (vì khâu vi phân) tạo ra độ dốc –20dB/dec, độ dốc –20db/dec được tiếp tục cho đến khi gặp tần số gãyω = 100rad/sec sẽ giảm 20dB/dec (vì khâu quán tính bậc 1) sẽ tạo độ dốc –40dB/decade.

Bài 5: Bài này trích từ trang 11-21 sách **'Control System Toollbox'**

Vẽ giản đồ bode của hệ thống hồi tiếp SISO có hàm sau:

$$H(s) = \frac{S^2+01.s+7.5}{S^2+0.12s^3+9s^2}$$

```
>> g=tf([1 0.1 7.5],[1 0.12 9 0 0]);
>> bode(g)
```



BIỂU ĐỒ NICHOLS

LÝ THUYẾT

Công dụng: Để xác định độ ổn định và đáp ứng tần số vòng kín của hệ thống hồi tiếp ta sử dụng biểu đồ Nichols. Sự ổn định được đánh giá từ đường cong vẽ mối quan hệ của độ lợi theo đặc tính pha của hàm truyền vòng hở. Đồng thời đáp ứng tần số vòng kín của hệ thống cũng được xác định bằng cách sử dụng đường cong biên độ và độ di pha vòng kín không đổi phủ lên đường cong biên độ – pha vòng hở.

CÚ PHÁP

[mod,phase,puls]= nichols(A,B,C,D);

[mod,phase,puls]= nichols(A,B,C,D,ui);

[mod,phase]= nichols(A,B,C,D,ui,w);

mod,phase,puls]= nichols(num,den);

[mod,phase]= nichols(num,den,w);

Những cấu trúc trên cho độ lớn là những giá trị tự nhiên, pha là độ và vectơ của điểm tần số là rad/s. Sự tồn tại của điểm tần số mà đáp ứng tần số được định giá bằng vectơ w, và ui là biến khai báo với hệ thống nhiều ngõ vào

Chú ý:

+ Khi sử dụng lệnh nichols với cấu trúc không có biến ngỏ ra thì ta được biểu đồ nichols

+ Lệnh nichols luôn luôn cho pha trong khoảng [-360^0,0^0]

Vẽ biểu nichols cho hệ thống có hàm truyền sau:

$$G(s) = 30\frac{s^2 + 7s + 1}{s(s+1)^3}$$

Các bước thực hiện:

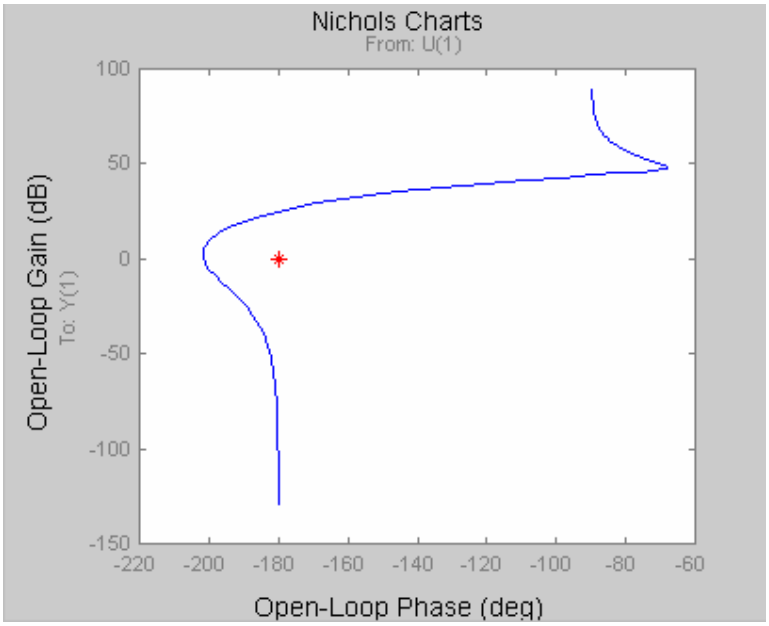
>> num=30*[1 7 1];

>> den=[poly([-1 -1 -1]) 0];

>> hold on, plot(-180,0,'*r'), hold on;

>> nichols(num,den)

Trả về biểu đồ nichols với điểm tới hạn **"critical point"** (-180^0 ,0) được biểu diễn như hình sau:



VẼ BIỂU ĐỒ NYQUYST VÀ KHẢO SÁT ỔN ĐỊNH

DÙNG GIẢN ĐỒ BODE

LÝ THUYẾT

+ Hệ thống ổn định ở trạng thái hở, sẽ ổn định ở trạng thái kín nếu biểu đồ Nyquist không bao điểm (-1+i0) trên mặt phẳng phức.

+ Hệ thống không ổn định ở trạng thái hở, sẽ ổn định ở trạng thái kín nếu biểu đồ Nyquist bao điểm (-1+i0)p lần ngược chiều kim đồng hồ (p là số cực GH nằm ở phải mặt phẳng phức).

Cấu trúc lệnh:

>> num = [*nhập các hệ số của tử số theo chiều giảm dần của số mũ*].

>> den = [*nhập các hệ số của mẫu số theo chiều giảm dần của số mũ*].

>> nyquist(num,den)

Vẽ biểu đồ Nyquyist của hệ thống có hàm truyền sau:

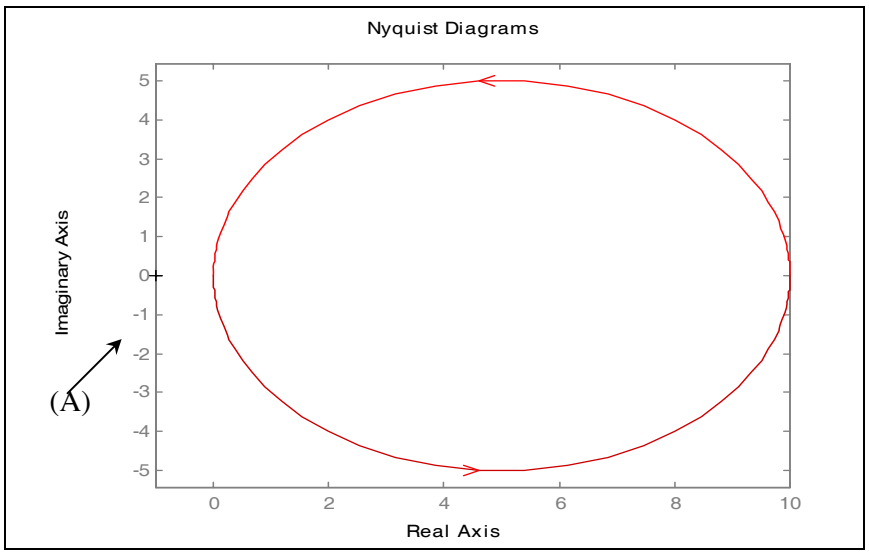
$$GH(s) = \frac{k}{1-st} \text{ (với k =10, t =1)}$$

>> num = 10;

>> den = [-1 1];

>> nyquist(num,den)

Kết quả:



Nhận xét: hàm truyền vòng hở có 1 cực nằm bên phải mặt phẳng phức. Biểu đồ Nyquist không bao điểm A (-1+j0).

Điểm –1 ký hiệu (+) nằm trên trục thực âm (Real Axis), điểm 0 nằm trên trục ảo (Imaginary Axis).

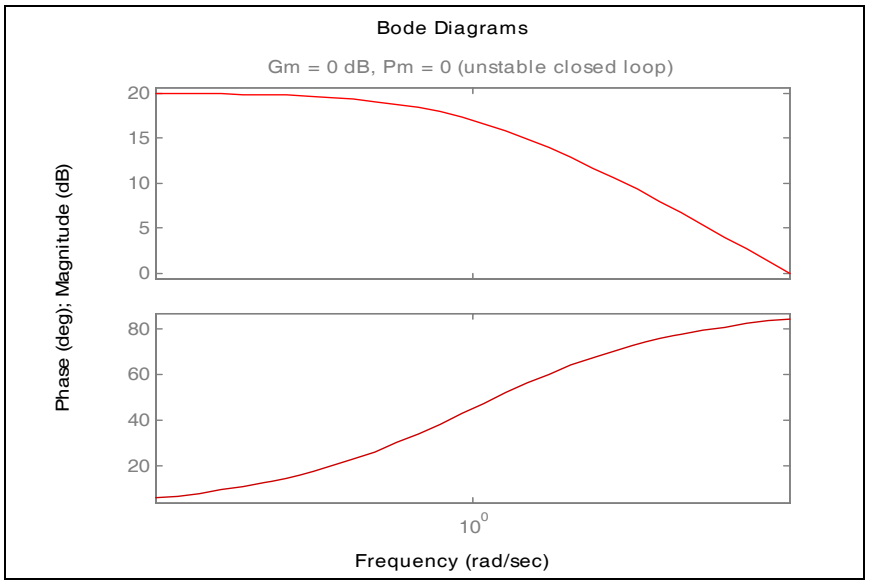
Kết luận: hệ không ổn định.

*** Dùng lệnh margin để tìm biên dư trữ và pha dư trữ.**

>> num = 10;

>> den = [-1 1];

>> margin(num,den);



Kết luận:

Độ dự trữ biên (Gm = 0 dB).

Độ dự trữ pha (Pm = 0°).

Warning: Closed loop is unstable (hệ vòng kín không ổn định).

Vẽ biểu đồ Nyquyist của hệ thống có hàm truyền sau:

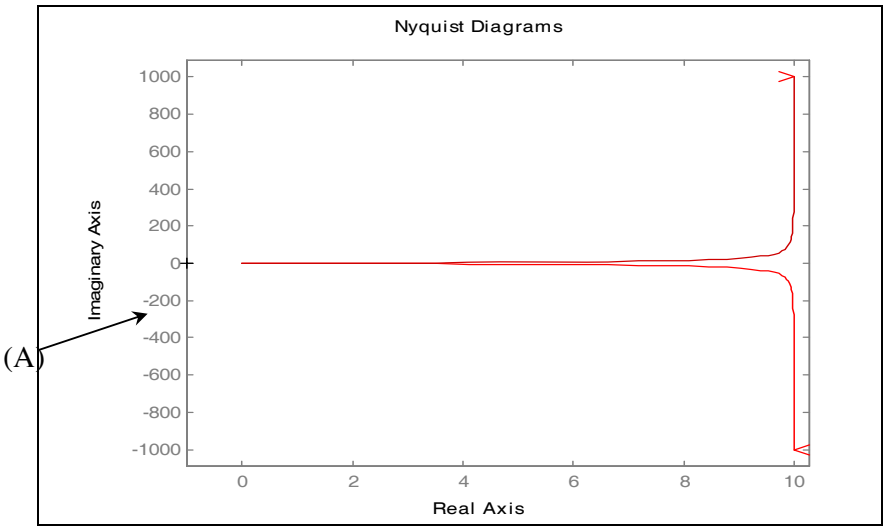
$$GH(s) = \frac{k}{s(1-st)} \quad (k = 10, t = 1)$$

>> num = 10;

>> den = [-1 1 0];

>> nyquist(num,den)

Ta thu được dồ thị sau:



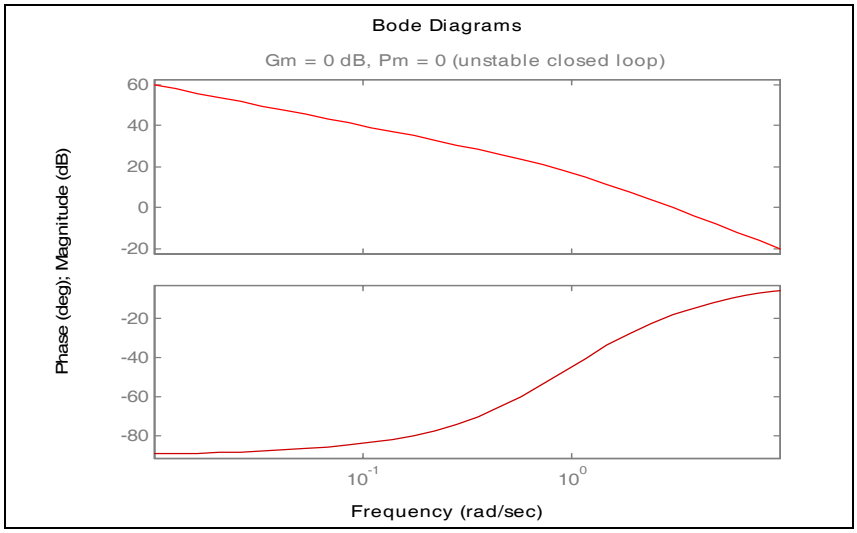
Nhận xét: hàm truyền vòng hở có 1 cực nằm bên phải mặt phẳng phức và 1 cực nằm tại gốc tọa độ. Biểu đồ Nyquist không bao điểm A (-1+j0).

Điểm –1 ký hiệu (+) nằm trên trục thực âm (Real Axis) , điểm 0 nằm trên trục ảo (Imaginary Axis).

Kết luận: hệ không ổn định.

*** Dùng lệnh margin để tìm biên dư trữ và pha dư trữ.**

```
>> num = 10;
>> den = [-1 1 0];
>>margin(num,den)
```



Kết luận:

Độ dự trữ biên (Gm = 0 dB).

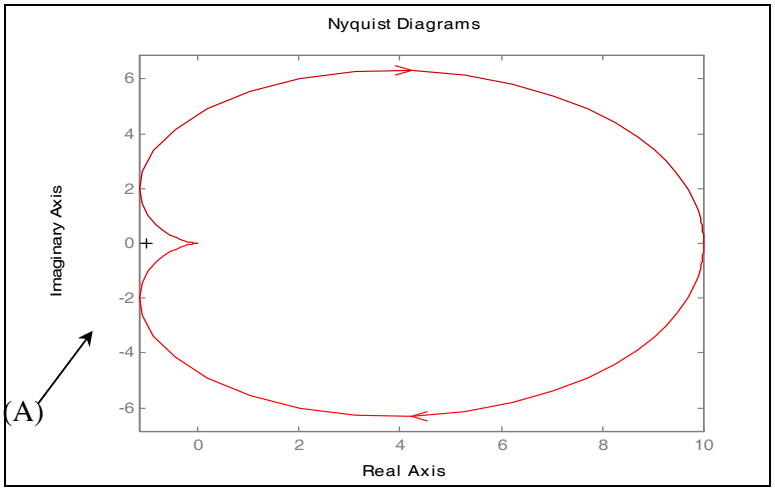
Độ dự trữ pha (Pm = 0°).

Warning: Closed loop is unstable (hệ vòng kín không ổn định).

Vẽ biểu đồ Nyquyist của hệ thống có hàm truyền sau:

$$GH(s) = \frac{k}{(t_1 s + 1)(t_2 s + 1)} \quad (k = 10,\ t_1 = 1,\ t_2 = 2)$$

```
>> num = 10;
>> den = [2 3 1];
>> nyquist(num,den)
```



Nhận xét: hàm truyền vòng hở có 2 cực nằm bên trái mặt phẳng phức. Biểu đồ Nyquist không bao điểm A (-1+j0).

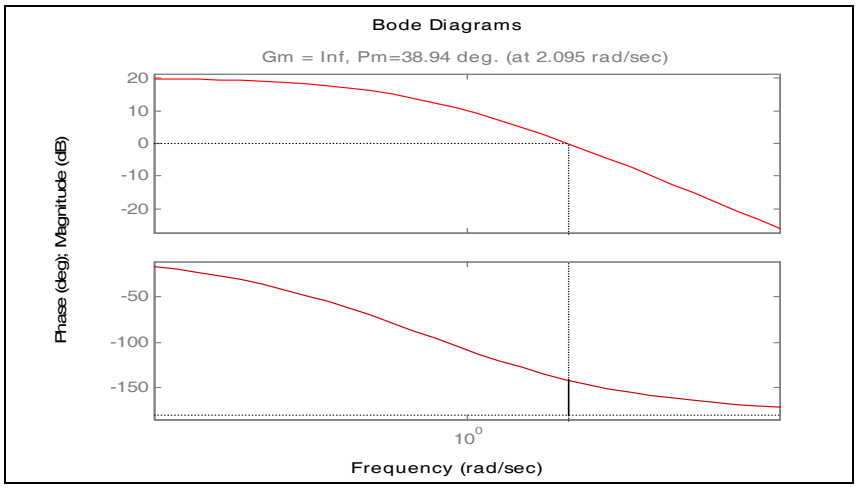
Điểm –1 ký hiệu (+) nằm trên trục thực âm (Real Axis) , điểm 0 nằm trên trục ảo (Imaginary Axis).

Kết luận: hệ thống ổn định.

*** Dùng lệnh margin để tìm biên dư trữ và pha dư trữ.**

```
>> num = 10;
>> den = [2 3 1];
>> margin(num,den)
```

Ta thu được dạng đồ thị sau:



Kết luận: hệ thống ổn định.

Độ dự trữ biên (Gm = ∞).

Độ dự trữ pha (Pm = 38.94°), tại tần số cắt biên 2.095 rad/sec.

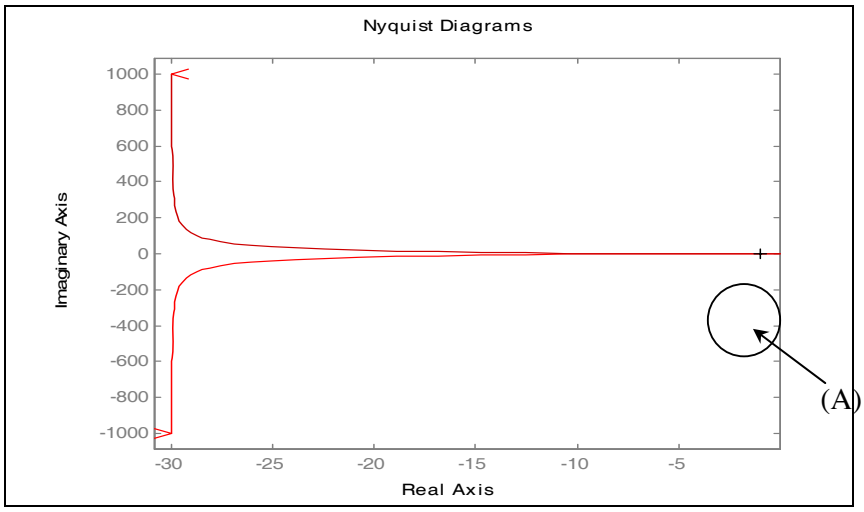
Vẽ biểu đồ Nyquyist của hệ thống có hàm truyền sau:

$$GH(s) = \frac{k}{s(t_1 s + 1)(t_2 s + 1)} \qquad (k = 10 \quad t_1=1, t_2 =2)$$

>>num = 10;

>> den = [2 3 1 0];

>> nyquist(num,den)



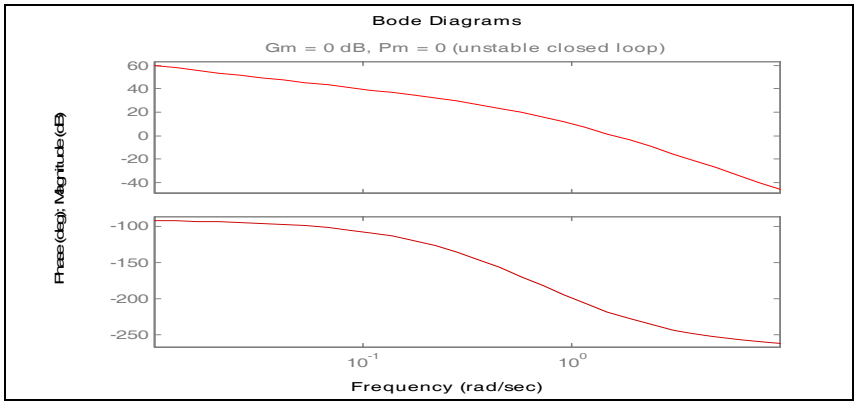
Nhận xét: hàm truyền vòng hở có 2 cực nằm bên trái mặt phẳng phức và 1 cực ở zero. Biểu đồ Nyquist bao điểm A(-1+j0).

Điểm –1 ký hiệu (+) nằm trên trục thực âm (Real Axis) , điểm 0 nằm trên trục ảo (Imaginary Axis).

Kết luận: hệ không ổn định.

*** Dùng lệnh margin để tìm biên dư trữ và pha dư trữ.**

```
>> num = 10;
>> den = [2 3 1 0];
>>margin(num,den)
```



Kết luận: hệ thống không ổn định.

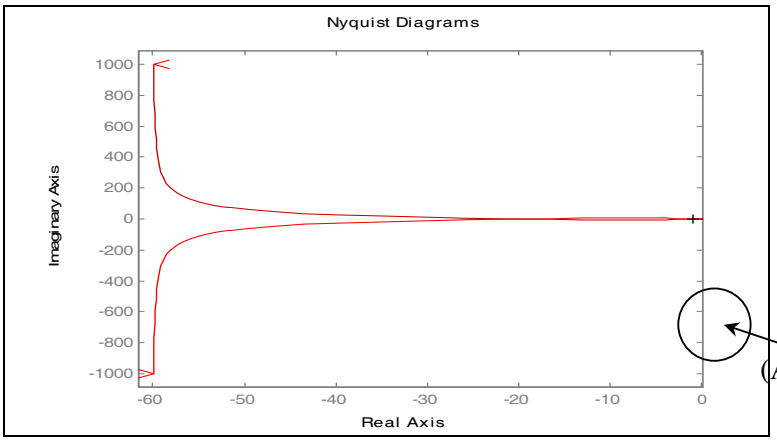
Độ dự trữ biên (Gm = 0 dB).

Độ dự trữ pha (Pm = 0°)

Vẽ biểu đồ Nyquyist của hệ thống có hàm truyền sau:

$$GH(s) = \frac{k}{s(t_1 s+1)(t_2 s+1)(t_3 s+1)} \qquad (t_1 = 1,\ t_2 = 2,\ t_3 = 3,\ k = 10)$$

```
>> num = 10;
>> den = [6 11 6 1 0];
>> nyquist(num,den)
```



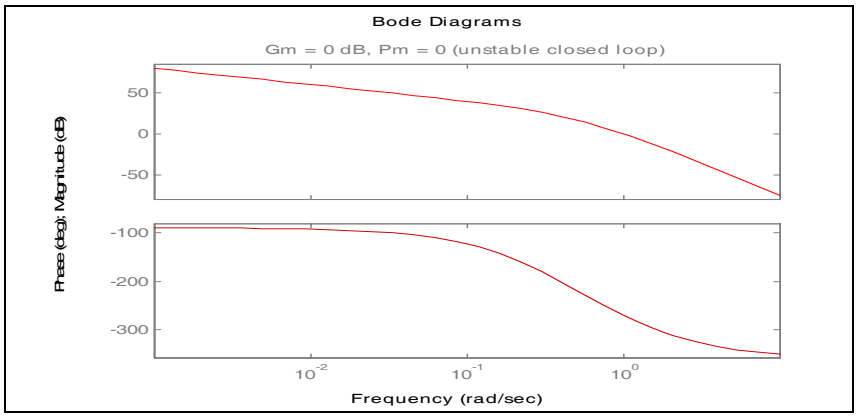
Nhận xét: hàm truyền vòng hở có 3 cực nằm bên trái mặt phẳng phức và 1 cực ở zero. Biểu đồ Nyquist bao điểm A (-1+i0).

Điểm –1 ký hiệu (+) nằm trên trục thực âm (Real Axis) , điểm 0 nằm trên trục ảo (Imaginary Axis).

Kết luận: hệ không ổn định.

*** Dùng lệnh margin để tìm biên dư trữ và pha dư trữ.**

```
>> num = 10;
>> den = [6 11 6 1 0];
>> margin(num,den)
```



Kết luận: hệ thống không ổn định.

Độ dự trữ biên (Gm = 0 dB).

Độ dự trữ pha (Pm = 0°).

NHÓM LỆNH VỀ QUỸ ĐẠO NGHIỆM
(Roots Locus)

1. Lệnh PZMAP

a) Công dụng:

Vẽ biểu đồ cực-zero của hệ thống.

b) Cú pháp:

[p,z]= pzmap(num,den)

[p,z]= pzmap(a,b,c,d)

[p,z]= pzmap(a,b,c,d)

c) Giải thích:

Lệnh pzmap vẽ biểu đồ cực-zero của hệ LTI. Đối với hệ SISO thì các cực và zero của hàmtruyền được vẽ.

Nếu bỏ qua các đối số ngõ ra thì lệnh pzmap sẽ vẽ ra biểu đồ cực-zero trên màn hình.

pzmap là phương tiện tìm ra các cực và zero tuyền đạt của hệ MIMO.

pzmap(a,b,c,d) vẽ các cực và zero của hệ không gian trạng thái trong mặt phẳng phức. Đối với các hệ thống MIMO, lệnh sẽ vẽ tất cả các zero truyền đạt từ tất cả các ngõ vào tới tất cả các ngõ ra. Trong mặt phẳng phức, các cực được biểu diễn bằng dấu × còn các zero được biểu diễn bằng dấu o.

pzmap(num,den) vẽ các cực và zero của hàm truyền trong mặt phẳng phức. Vector num và den chứa các hệ số tử số và mẫu số theo chiều giảm dần số mũ của s.

pzmap(p,z) vẽ các cực và zero trong mặt phẳng phức. Vector cột p chứa tọa độ các cực và vector cột z chứa tọa độ các zero trong mặt phẳng phức. Lệnh này vẽ các cực và zero đã được tính sẵn trong mặt phẳng phức.

Nếu giữ lại các đối số ngõ ra thì :

[p,z]= pzmap(num,den)

[p,z]= pzmap(a,b,c,d)

[p,z]= pzmap(a,b,c,d)

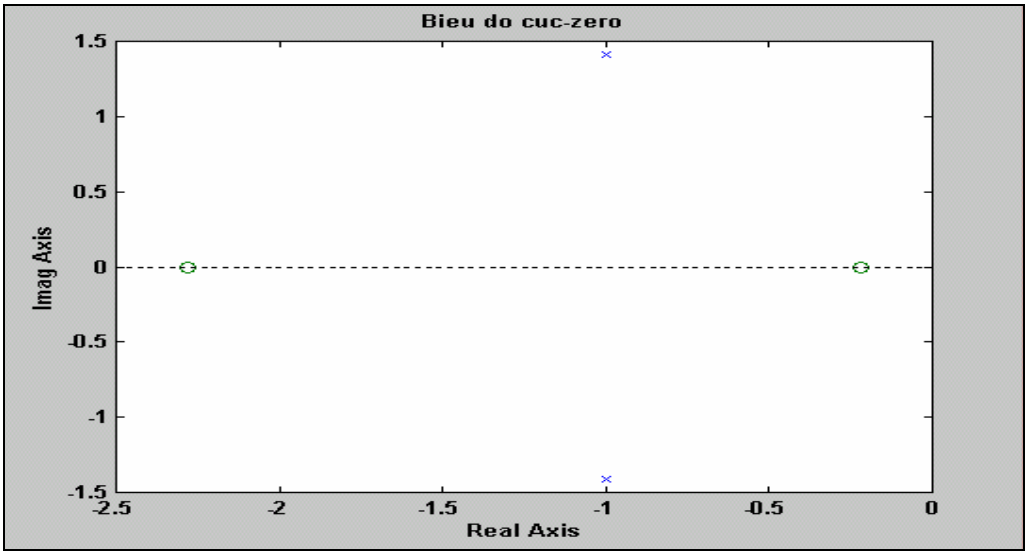
tạo ra các ma trận p và z trong đó p chứa các cực còn z chứa các zero.

d) Ví du: (Trích trang 11-174 sách '**Control system Toolbox'**)

Vẽ các cực và zero của hệ liên tục có hàm truyền :

$$H(s) = \frac{2s^2 + 5s + 1}{s^2 + 2s + 3}$$

```
num = [2    5    1];
den  = [1    2    3];
pzmap(num,den)
title('Bieu do cuc-zero')
```



2. Lênh RLOCFIND

a) Công dung:

Tìm độ lợi quỹ đạo nghiệm với tập hợp nghiệm cho trước.

b) Cú pháp:

[k,poles]= rlocfind(a,b,c,d)

[k,poles]= rlocfind(num,den)

[k,poles]= rlocfind(a,b,c,d,p)

[k,poles]= rlocfind(num,den,p)

c) Giải thích:

Lệnh rlocfind tạo ra độ lợi quỹ đạo nghiệm kết hợp với các cực trên quỹđạo nghiệm. Lệnh rlocfind được dùng cho hệ SISO liên tục và gián đoạn.

[k,poles]= rlocfind(a,b,c,d) tạo ra dấu x trong cửa sổ đồ họa mà ta dùng để chọn một điểm trên quỹ đạo nghiệm có sẵn. Độ lợi của điểm này được tạo ra trong k và các cực ứng với độ lợi này nằm trong poles. Để sử dụng lệnh này thì quỹ đạo nghiệm phải có sẵn trong cửa sổ đồ họa.

[k,poles]= rlocfind(num,den) tạo ra dấu x trong cửa sổ đồ họa mà ta dùng để chọn một điểm trên quỹ đạo nghiệm của hệ thống có hàm truyền G = num/den trong đó có num và den chứa các hệ số đa thức theo chiều giảm dần số mũ của s hoặc z.

[k,poles]= rlocfind(a,b,c,d,p) hoặc [k,poles]= rlocfind(num,den,p) tạo ra vector độ lợi k và vector các cực kết hợp pole với mỗi thành phần trong mỗi vector ứng với mỗi nghiệm trong p.

d) Ví du:

Xác định độ lợi hồi tiếp để các cực vòng kín của hệ thống có hệ số tắt dần $\zeta = 0.707$ và có hàm truyền :

$$H(s) = \frac{2s^2 + 5s + 1}{s^2 + 2s + 3}$$

num = [2 5 1];

den = [1 2 3];

+) Vẽ quỹ đạo nghiệm:

rlocus(num,den); title('Do loi quy dao nghiem');

+)Tìm độ lợi tại điểm được chọn:

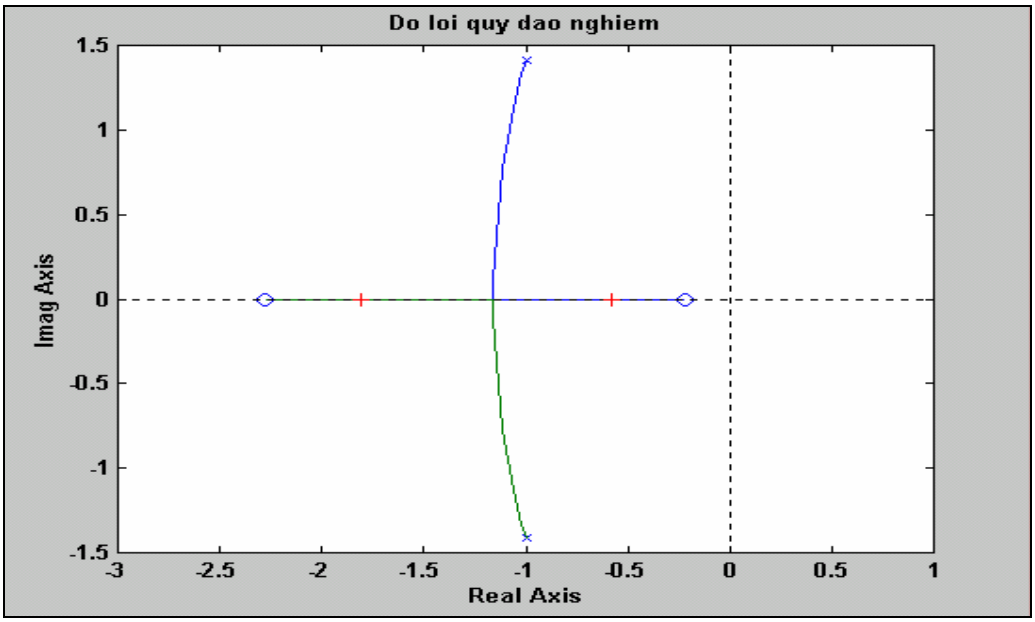
rlocfind(num,den);

Sau khi nhập xong lệnh, trên màn hình của Matlab sẽ xuất hiện dòng chữ:

Select a point in the graphics window

và trên hình vẽ có thước để ta kéo chuột và chọn điểm

ta có quỹ đạo nghiệm:



3. Lệnh RLOCUS

a) Công dụng:

Tìm quỹ đạo nghiệm Evans.

b) Cú pháp:

r = rlocus(num,den)

r = rlocus(num,den,k)

r = rlocus(a,b,c,d)

r = rlocus(a,b,c,d,k)

c) Giải thích:

Lệnh rlocus tìm quỹ đạo nghiệm Evans của hệ SISO. Quỹ đạo nghiệm được dùng để nghiên cứu ảnh hưởng của việc thay đổi độ lợi hồi tiếp lên vị trí cực của hệ thống, cung cấp các thông tin về đáp ứng thời gian và đáp ứng tần số. Đối với đối tượng điều khiển có hàm truyền G(s) và khâu bổ chính hồi tiếp k*f(s), hàm truyền vòng kín là :

$$h(s) = \frac{g(s)}{1 + kg(s)f(s)} = \frac{g(s)}{q(s)}$$

Nếu bỏ qua các đối số ngõ ra thì lệnh rlocus sẽ vẽ ra quỹ đạo trên màn hình. Lệnh rlocus dùng cho cả hệ liên tục và gián đoạn.

r = rlocus(num,den) vẽ quỹ đạo nghiệm của hàm truyền :

$$\text{q(s)} = 1 + \text{k}\frac{num(s)}{den(s)} = 0$$

với vector độ lợi k được xác định tự động. Vector num và den chỉ ra hệ tử số và mẫu số theo chiều giảm dần số của s hoặc z.

$$\frac{num(s)}{den(s)} = \frac{num(1)s^{nn-1} + num(2)s^{nn-2} + ... + num(nn)}{den(1)s^{nd-1} + den(2)s^{nd-2} + ... + den(nd)}$$

r = rlocus(a,b,c,d) vẽ ra quỹ đạo nghiệm của hệ không gian trạng tái SISO liên tục và gián đoạn với vector độ lợi được xác định tự động

r = rlocus(num,den,k) hoặc r = rlocus(a,b,c,d,k) vẽ ra quỹ đạo nghiệm với vector độ lợi k do người sử dụng xác định. Vector k chứa các giá trị và độ lợi mà nghiệm hệ vòng kín được tính.

Nếu sử dụng các đối số ngõ ra thì :

[r,k] = rlocus(num,den)

[r,k] = rlocus(num,den,k)

[r,k] = rlocus(a,b,c,d)

[r,k] = rlocus(a,b,c,d,k)

tạo ra ma trận ngõ ra chứa các nghiệm và vector độ lợi k. Ma trận r có length(k) hàng và (length(den) –1) cột, ngõ ra chứa vị trí các nghiệm phức. Mỗi hàng trong ma trận tương ứng với một độ lợi trong vector k. Quỹ đạo nghiệm có thể được vẽ bằng lệnh plot(r,'x').

d) Ví dụ: Tìm và vẽ quỹ đạo nghiệm của hệ thống có hàm truyền :

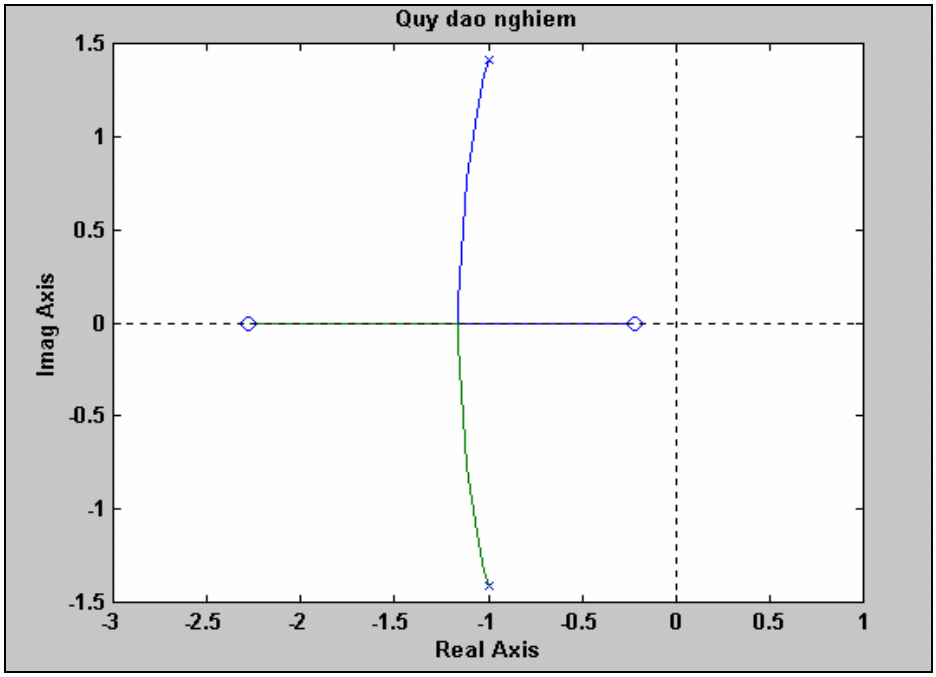
$$H(s) = \frac{2s^2 + 5s + 1}{s^2 + 2s + 3}$$

+) Xác định hàm truyền :

```
>>num = [2   5   1];
>>den  = [1   2   3];
```

+)Vẽ quỹ đạo nghiệm :

```
>>rlocus(num,den)
title('Quy dao nghiem')
```



4. Lênh SGRID

a) Công dung:

Tạo lưới cho quỹ đạo nghiệm và biểu đồ cực-zero liên tục.

b) Cú pháp:

sgrid

sgrid('new')

sgrid(z,wn)

sgrid(z,wn,'new')

c) Giải thích:

Lệnh sgrid tạo lưới cho quỹ đạo nghiệm và biểu đồ cực-zero liên tục trong mặt phẳng s. Đường lưới vẽ là các đường hằng số tỉ số tắt dần (ζ) và tần số tự nhiên (ω_n). Đường tỉ số tắt dần được vẽ từ 0 tới 1 theo từng nấc là 0.1.

sgrid('new') xóa màn hình đồ họa trước khi vẽ và thiết lập trạng thái hold on để quỹ đạo nghiệm hay biểu đồ cực-zero được vẽ lên lưới bằng các lệnh :

sgrid('new')

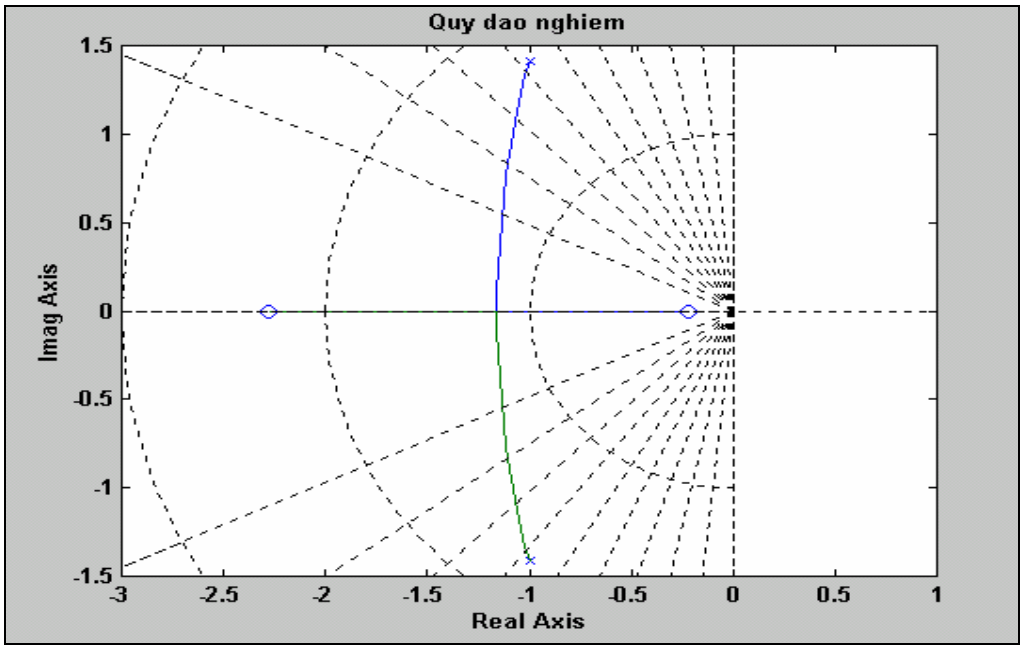
rlocus(num,den) hoặc pzmap(num,den)

sgrid(z,wn) vẽ các đường hằng số tỉ lệ tắt dần được chỉ định trong vector z và vẽ đường tần số tự nhiên được chỉ định trong vector wn.

sgrid(z,wn,'new') xóa màn hình đồ họa trước khi vẽ các đường tỉ số tắt dần và tần số tự nhiên được chỉ định trong vector z và wn. Trạng thái hold on được thiết lập.

d) Ví du:Vẽ lưới trong mặt phẳng s trên quỹ đạo nghiệm của hệ thống có hàm truyền :

```
>>num = [2    5    1];  % ta có thể thay đổi 2 dòng num=…, den=… thành dòng lệnh sau:
>>den  = [1    2    3];  %  H(s)=tf([2 5 1],[1 2 3]);
>>rlocus(num,den)
title('Quy dao nghiem')
sgrid
```



5. Lệnh ZGRID

a) Công dụng:

Vẽ lưới tỉ lệ tắt dần và tần số tự nhiên cho quỹ đạo nghiệm gián đoạn.

b) Cú pháp:

zgrid

zgrid('new')

zgrid(z,wn)

zgrid(z,wn,'new')

c) Giải thích:

Lệnh zgrid tạo lưới quỹ đạo cho nghiệm hoặc biểu đồ cực-zero trong mặt phẳng z. Các đường hằng số tỉ lệ tắt dần (ζ) và tần số tự nhiên chuẩn hóa sẽ được vẽ. ζ được thay đổi từ 0 tới 1 theo từng nấc thay đổi là 0.1 và tần số tự nhiên được vẽ từ 0 tới π với từng nấc thay đổi là π/ω.

zgrid('new') xóa màn hình đồ họa trước khi vẽ lưới và thiết lập trạng thái hold on để quỹ đạo nghiệm hoặc biểu đồ cực-zero được vẽ lên lưới sử dụng các lệnh :

zgrid('new')

rlocus(num,den) hoặc pzmap(num,den)

zgrid(z,wn) vẽ hằng số tắt dần được chỉ định trong vector z và vẽ hằng số tần số tự nhiên cho các tần số chuẩn hóa được chỉ định trong vector wn. Các tần số chuẩn hóa có thể được vẽ bằng lệnh zgrid(z,wn/Ts) với tần số là thời gian lấy mẫu.

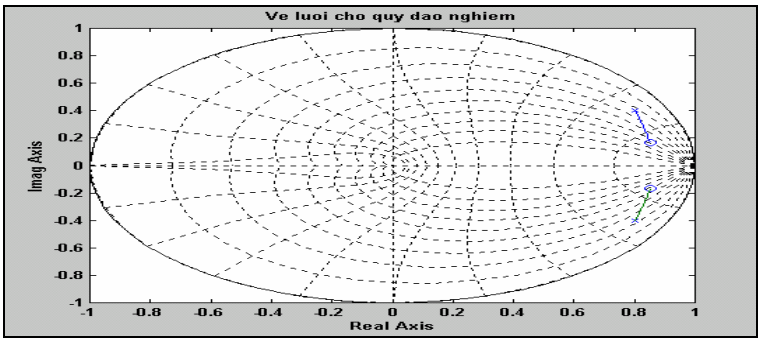
zgrid(z,wn,'new') xóa màn hình đồ họa trước khi vẽ tỉ số tắt dần và tần số tự nhiên được chỉ định trong vector z và wn. Trạng thái hold on được thiết lập.

zgrid([ ],[ ]) sẽ vẽ ra vòng tròn đơn vị.

d) Ví du:Vẽ lưới trong mặt phẳng cho quỹ đạo nghiệm của hệ thống có hàm truyền :

$$H(z) = \frac{2z^2 - 3.4z + 1.5}{z^2 - 1.6s + 0.8}$$

```
>>num = [2    -3.4    1.5];
>>den  = [1    -1.6    0.8];
>>axis('square')
>>zgrid('new')
>>rlocus(num,den)
title('Ve luoi cho quy dao nghiem')
```

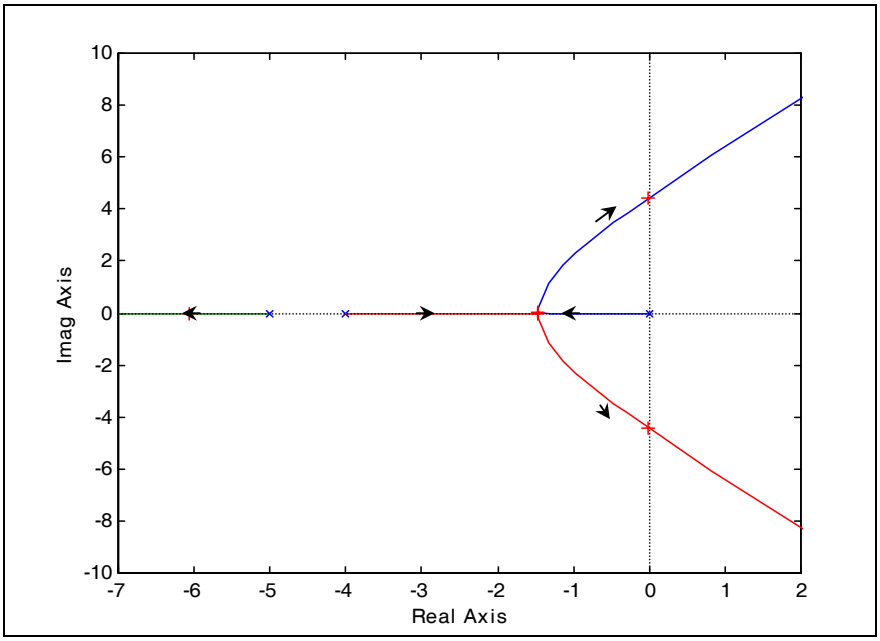


CÁC BÀI TẬPVỀ QUỸ ĐẠO NGHIỆM

Ví dụ: Cho hàm truyền sau:

$$KGH = \frac{k}{s(s+4)(s+5)} \quad \text{với } k = 2$$

```
>> num = 2;
>> den = [1 9 20 0];
>> rlocus(num,den)
```



Từ đồ thị cho ta:

1. Điểm cực: 0 ,-4,-5.
2. Quỹ đạo nghiệm có 3 nhánh.
3. Điểm zero ở vô cùng (∞).
4. Điểm tách được xác định bằng cách từ cửa sổ MATLAB ta nhập:

```
>> num = 2;
>> den = [1 9 20 0];
>> rlocus(num,den);
>> rlocfind(num,den)
```

Sau khi nhập lệnh thì trên cửa sổ lệnh sẽ xuất hiện hàng chữ:

Select a point in the graphics window (hãy chọn 1 điểm trên đồ thị minh họa).

Trên đồ thị sẽ có thước cho ta chọn điểm – kéo rê chuột để chọn điểm cần chọn.

selected_point = -1.4516

Điểm tách có giá trị: -1.4516

Giao điểm của quỹ đạo nghiệm với trục ảo (tương tự như tìm điểm tách): +4.472j, -4.472j.

Từ giá trị tại giao điểm của quỹ đạo nghiệm với trục ảo ta thế vào phương trình đặc trưng:

$F(s) = s^3+9s^2+20s+k = 0$

$F(jw) = -jw^3-9w^2+20jw+k = 0$

$\Rightarrow$ **kgh = 180**

Kết luận: hệ thống sẽ ổn định khi $0 < k < 180$

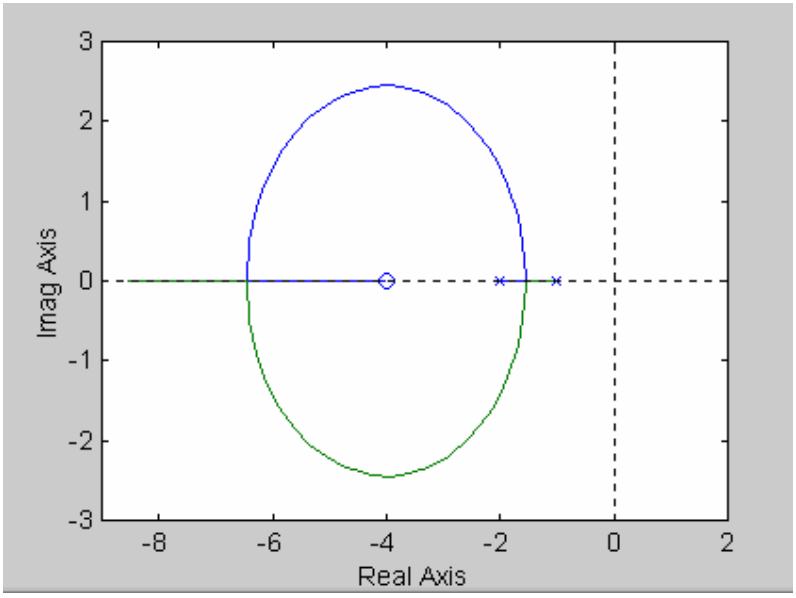
Ví dụ: Cho hàm truyền như sau:

$$G(s) = \frac{s+4}{(s+1)(s+2)}$$

Viết theo cấu trúc sau ta có được đồ thị biểu diễn quỹ đạo nghiệm:

```
>>à num=[1 4];
>> den=conv([1 1],[1 2])
>> rlocus(num,den)
```

Kết quả như hình sau:



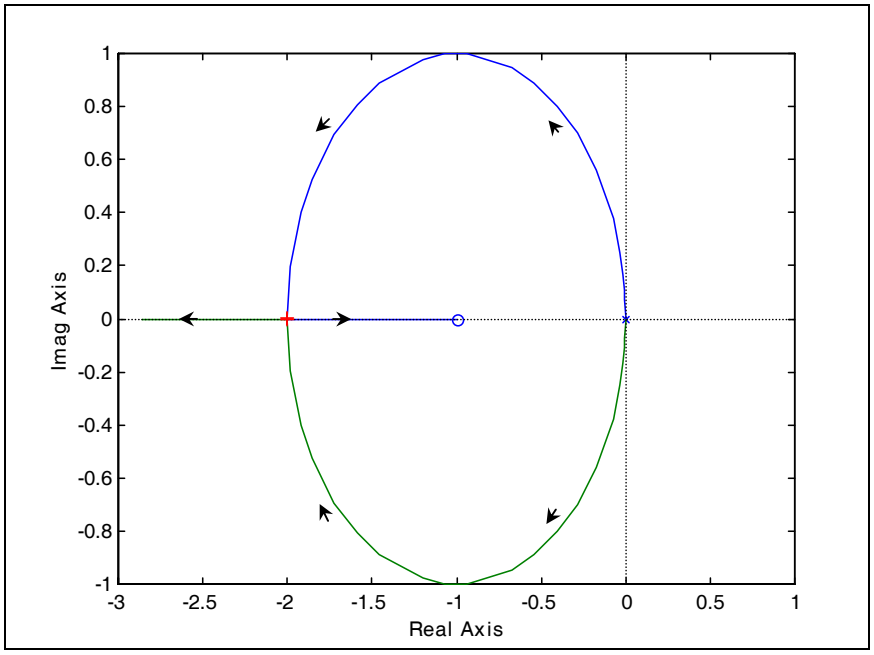
Ví dụ : Cho hàm truyền sau :

$$KGH = \frac{k(ts+1)}{s^2} \quad (k = 1, t = 1)$$

>> num = [1 1];

>>den = [1 0 0];

>> rlocus(num,den)



1. Điểm cực: 0
2. Quỹ đạo nghiệm có 2 nhánh
3. Điểm zero ở ∞, -1
4. Điểm tách được được xác định bằng cách từ cửa sổ MATLAB ta nhập:

ầ num = [1 1];

ầ den = [1 0 0];

ầ rlocus(num,den);

ầ rlocfind(num,den)

Sau khi nhập lệnh thì trên cửa sổ lệnh sẽ xuất hiện hàng chữ:

Select a point in the graphics window (hãy chọn 1 điểm trên đồ thị minh họa).

Trên đồ thị sẽ có thước cho ta chọn điểm – kéo rê chuột để chọn điểm cần chọn.

selected_point = -2

Điểm tách có giá trị: -2.

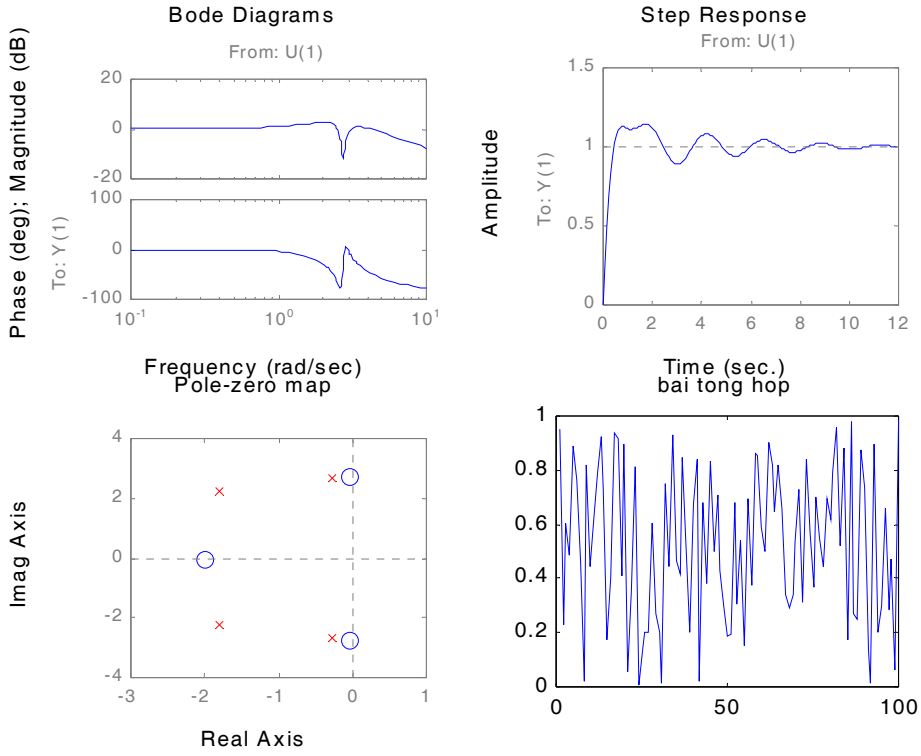
Kết luận: hệ thống ở biên ổn định.

Ví dụ: Trích từ trang 5-19 sách **'Control System Toolbox'**

Bài này tổng hợp các lệnh:

```
>> h=tf([4 8.4 30.8 60],[1 4.12 17.4 30.8 60]);
>> subplot(221)
>> bode(h)
>> subplot(222)
>> step(h)
>> subplot(223)
>> pzmap(h)
>> subplot(224)
>> plot(rand(1,100)
>> plot(rand(1,100))
```

Kết quả ta thu được dạng đồ thi sau:



PHẦN III

KHÁI QUÁT VỀ SIMULINK

1. Khởi động Simulink: khởi động vào Matlab, sau đó có hai cách vào cửa sổ Simulink

Cách 1: vào trực tiếp Simulink bằng cách nhấp chuột vào biểu tượng trong menu của Matlab

Cách 2: gõ lệnh Simulink/ Enter (↵)

2. Đặc điểm của Simulink

Simulink phân biệt (không phụ thuộc vào thư viện con) hai loại khối chức năng: khối ảo (virtual) và khối thực (notvirtual). Các khối thực đóng vai trò quyết định khi chạy mô phỏng mô hình Simulink. Việc thêm hay bớt một khối thực sẽ thay đổi đặc tính động học của hệ thống đang được mô hình Simulink mô tả. Có thể nêu nhiều ví dụ về khối thực như: khối tích phân Integrator hay khối hàm truyền đạt Tranfer Fcn của thư viện Continuous, khối Sum hay khối Product của thư viện con Math. Ngược lại, các khối ảo không có khả năng thay đổi đặc tính của hệ thống, chúng chỉ có nhiệm vụ thay đổi diện mạo đồ hoạ của mô hình Simulink. Đó chính la các khối như Mux, Demucx hay Enable thuộc thư viện con Signal và System. Một số chức năng mang đặc tính ảo hay thực tuỳ thuộc theo vị trí hay cách thức sử dụng chúng trong mô hình Simulink, các mô hình đó được xếp vào loại ảo có điều kiện .

3. Các thao tác cơ bản sử dụng trong Simulink

Simulink gần như chỉ có thể sử dụng được nhờ chuột. Bằng cách nháy kép phím chuột trái vào một trong số các thư viện con thuộc cửa sổ thư viện chính Library ta sẽ thu được một cửa sổ mới có chứa các khối thuộc thư viện con đó. Hoặc cũng có thể thu được kết quả tương tự bằng cách nháy kép chuột trái nhánh của thư viện con, nằm ở phần bên phải của cửa sổ truy cập Library Browser. Từ các khối chứa trong thư viện con ta có thể xây dựng được lưu đồ tín hiệu mong muốn. Để tạo định dạng (Format) và soạn thảo ta có các khả năng sau đây:

- **Copy** (sao chép): bằng cách gắp và thả " Drag &Drop" nhờ phím chuột phải ta có thể chép một khối từ thư viện (cũng có thể từ một thư viện khác)

- **Move** (di chuyển): ta có thể dễ dàng di chuyển một khối trong phạm vi cửa sổ của khối đó nhờ phím chuột trái.

- **Đánh dấu** : bằng cách nháy phím chuột trái vào khối ta co thể đánh dấu, lựa chọn từng khối, hoặc kéo chuột đánh dấu nhiều khối một lúc.

- **Delete** (xoá) : có thể xoá các khối và các đường nối đã bị đánh dấu bằng cách gọi lệnh menu Edit / Clear . Bằng menu Eđit / Undu hoặc tổ hợp phím Ctrl + Z ta có thể cứu vãn lại động tác xoá vừa thực hiện.

- **Hệ thống con**: bằng cách đánh dấu nhiều khối có quan hệ chức năng, sau đó gom chúng lại thông qua menu Edit / Creat Subsystem, ta có thể tạo ra một hệ thống con mới.

- **Nối hai khối** : dùng phím chuột trái nháy vào đầu ra của một khối, sau đó di mũi tên của chuột tới đầu vào cần nối. Sau khi thả ngón tay khỏi phím chuột, đường nối tự động được tao ra. Có thể rẽ nhánh tín hiệu bằng cách nháy phím chuột phải vào một đường nối có có sẵn kéo đường nối mới xuất hiện tới đầu vào cần nối.

- **Di chuyển đường nối**: để lưu đồ tín hiệu thoáng và dễ theo dõi, nhiều khi ta phải di chuyển, bố trí lại vị trí các. Sau khi nhả ngón tay khỏi phím chuột, đường nối tự dộng được tạo ra

có thể rẽ nhánh tín hiệu bằng cách nháy phím chuột phải vào một đường nối có sẵn và kéo đường nối mới xuất hiện tới đầu vào cần nối.

- **Di chuyển đường nối**: để lưu đồ tín hiệu thoáng và dễ theo dõi, nhiều khi ta phải di chuyển, bố trí lại các đường nối. Khi nháy chọn bằng chuột trái ta có thể di chuyển tuỳ ý các điểm góc hoặc di chuyển song song đoạn thẳng của đường nối.

- **Chỉ thị kích cỡ và dạng dữ liệu của tín hiệu**: lệnh chọn qua menu Format/ Signal dimensions sẽ hiển thị kích cỡ của tín hiệu tín hiệu đi qua đường nối. Lệnh menu Format / Port data types chỉ thị thêm loại dữ liệu của tín hiệu qua đường nối.

- **Định dạng** (Format) **cho một khối**: sau khi nháy phím chuột phải vào một khối, cửa sổ định dạng khối sẽ mở ra. Tại mục Format ta có thể lựa chọn kiểu và kích cỡ chữ, cũng như vị trí của tên khối, có thể lật hoặc xoay khối. Hai mục Foreground Color và Background Color cho phép ta đặt chế độ màu bao quanh cũng như mầu nền của khối.

- **Định dạng cho đường nối**: sau khi nháy phím chuột phải vào một đường nối, cửa sổ định dạng đường(của cả đường dẫn tới đường nối đó) sẽ mở ra. Tại đây ta có các lệnh cho phép cắt bỏ, copy hoặc delete đường nối

- **Hộp đối thoại** (Dialog Box) **về đặc tính của khối** (Block Properties): hoặc đi theo menu của cửa sổ mô phỏng Edit/Block Properties, hoặc chọn mục Block Properties của cửa sổ định dạng khối, ta sẽ thu được hộp đối thoại cho phép đặt một vài tham số tổng quát về đặc tính của khối.

- **Hộp đối thoại về đặc tính của tín hiệu** (Signal properties): có thể tới được hộp thoại như Signal properties của một đường nối hoặc bằng cách nháy chuột đánh dấu trên cửa sổ mô phỏng, sau đó đi theo menu Edit/ Signal properties, hoặc chọn mục Signal properties từ cửa sổ định dạng đường. Trong hộp đối thoại ta có thể đặt tên cho đường nối hoặc nhập một đoạn văn bản mô tả. Tuy nhiên, để đặt tên cho đường nối cũng còn có cách khác đơn giản hơn: nháy kép phím chuột trái vào đường nối ta sẽ tự động tới được chế độ nhập văn bản.

II. Tín hiệu và các loại dữ liệu

1. Làm việc với tín hiệu

Đối với Simulink, khái niệm tín hiệu nhằm chỉ vào dữ liệu xuất hiện ở đầu ra của các khối chức năng trong quá trình mô phỏng: các dữ liệu đó chạy dọc theo đường nối từ đầu ra của khối chức năng này tới đầu vào của các khối chức năng khác mà không tốn thơi gian. Tín hiệu trong khuôn khổ Matlab có những đặc điểm riêng do người sử dụng xác định.

Trong Simulink ta phân biệt ba loại kích cỡ tín hiệu:

- Tín hiệu đơn(Scalar).

- Vector tín hiệu: còn được gọi là tín hiệu 1-D, vì kích cỡ của tín hiệu được xác định theo hai chiều [m x n]. Cả vector hàng [1 x n] và vector cột [m x 1] cũng thuộc về phạm trù ma trận tín hiệu. Đôi khi, ví dụ: lúc khai báo định dạng, ma trận cũng được gọi là mảng

Khi tạo một cấu trúc Simulink, các khối ảo sẽ tạo nên các đường tín hiệu ảo, duy nhất nhằm mục đích làm cho sơ đồ cấu trúc chở nên đỡ rối mắt, người sử dụng dễ quản lí hơn. Tín hiệu ảo có thể coi là sự tập hợp hình ảnh của nhiều tín hiệu ảo, không ảo, hay hỗn hợp cả hai loại. Trong quá trình mô phỏng, Simulink sử dụng một thủ tục tên Signal properties để nhận biết: những tín hiệu thực nào được ghép vào tín hiệu ảo. Diễn đạt một cách khác: những khối chức năng nào được ghép thực sự ở đầu cuối của tín hiệu

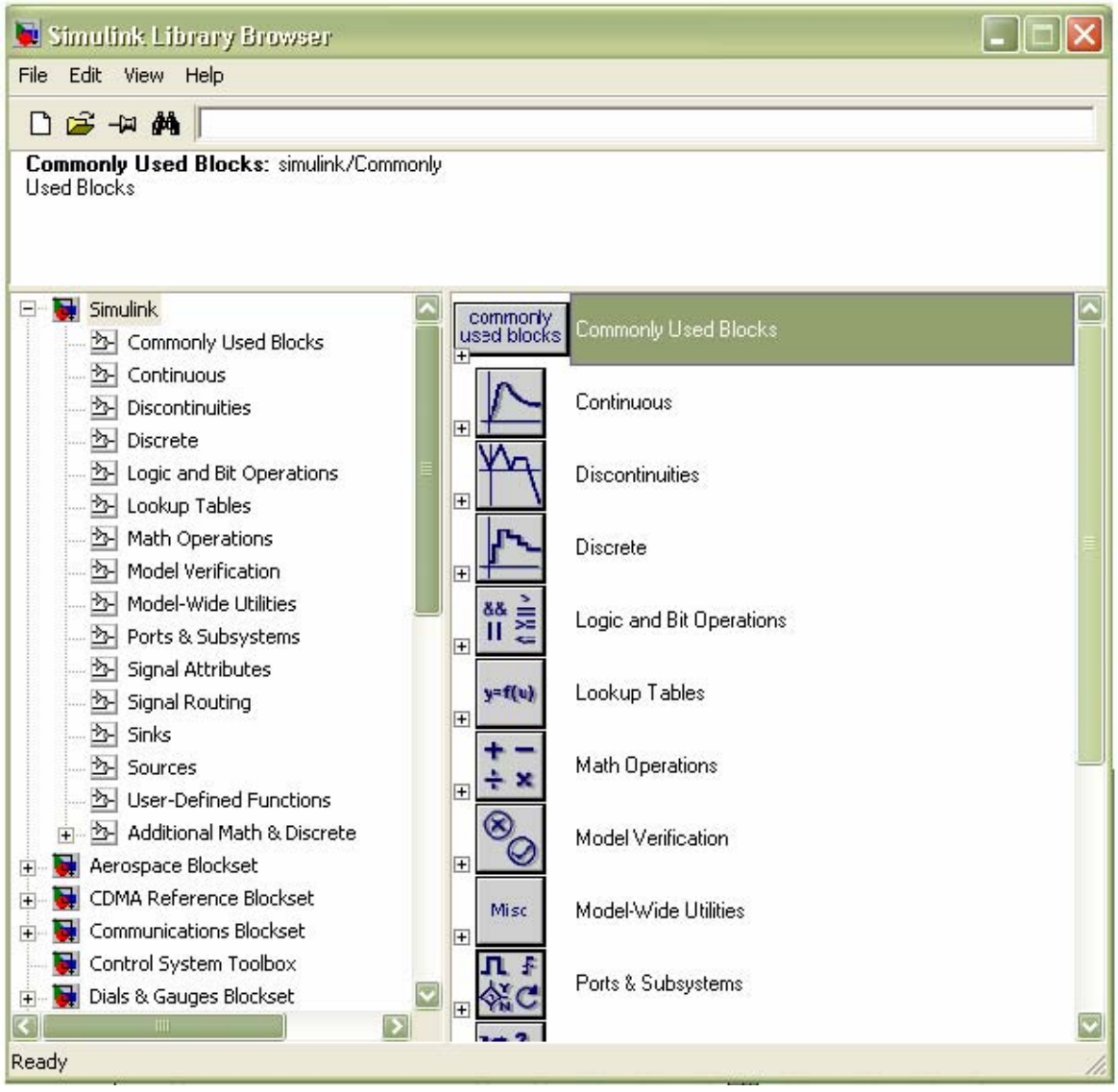
2. Làm việc với các loại số liệu

Bên cạnh các đặc điểm đã được giới thiệu, mỗi tín hiệu thuộc sơ đồ cấu trúc Simulink đều được gán một loại số liệu nhất định, và đó quyết định đến dung lượng bộ nhớ dành cho một tín hiệu. Simulink cũng hỗ trợ tất cả các loại số liệu của Matlab

- Double: chính xác cao, dấu phẩy động
- Sigle: chính xác vừa, dấu phẩy động
- Boolean (0 hoặc 1, logic, được Simulink sử lí như uint8)

Loại số mặc định sẵn của Simulink là Double. Trong quá trình mô phỏng, Simulink sẽ kiểm tra xem việc đảo giữa các loại số liệu có đúng hay không nhằm loại trừ các kết quả sai lầm có thể xảy ra.

Khả năng khai báo, xác định loại số liệu của tín hiệu cũng như của tham số thuộc các khối chức năng trong Simulink là đặc biệt có ý nghĩa, nếu ta dự định tạo ra từ mô hình Simulink mã chạy cho các ứng dụng thời gian thực. Nhu cầu về bộ nhớ và tốc độ tính toán phụ thuộc vào loại số liệu được ta chọn



III. Thư viện của Simulink

1.Thư viện Sources

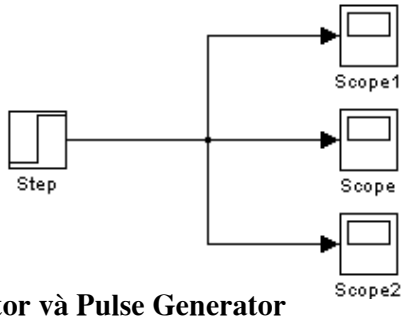
Trong thư viện này bao gồm các nguồn phát tín hiệu, các khối cho phép nhập số liệu từ một File, hay từ Matlab Workspace. Sau đây ta lần lượt điểm qua ý nghĩa từng khối.

a. Constant: khối này tạo nên một hằng số (không phụ thuộc thời gian) thực hoặc phức. Hằng số đó có thể là vecter hay ma trận…. Ta có thể khai báo tham số constsnt value là vecter hàng hay cột với kích cỡ [n *x* 1] hay [1 x n] dưới dạng ma trận

b. Step và Ramp: nhờ hai khối này ta có thể tạo nên các tín hiệu dạng bậc thang hay dạng dốc tuyến tính dùng để kích thích các mô hình Simulink. Trong hộp thoại Block Parameters của khối Step ta có thể khai báo giá trị đầu- giá trị cuối và cả thời điểm bắt đầu của tín hiệu bước nhảy. Đối với Ramp ta có thể khai báo độ dốc, thời điểm mà giá trị xuất phát của tín hiệu ở đầu ra.

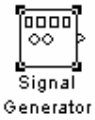
(Chú ý: hai khối Step và Ramp không chỉ tạo ra một tín hiệu mà có thể tạo ra một tập các tín hiệu được xử lý dưới dạng vector hoặc ma trận.)

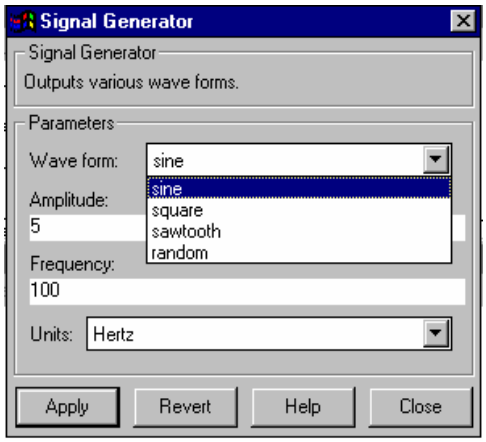
Ví dụ:



c. Signal Generator và Pulse Generator

Nhờ Signal Generator ta tạo ra các dạng tín hiệu kích thích khác nhau.

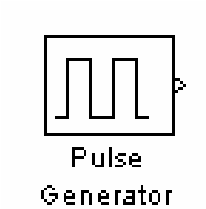


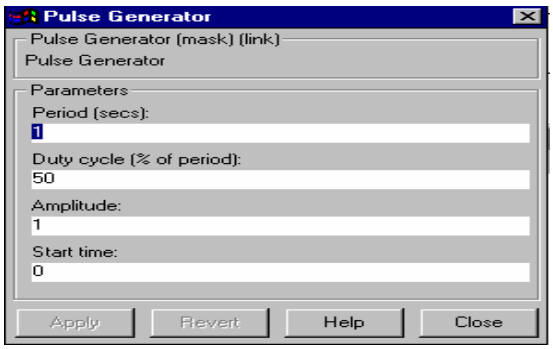


Cung cấp cho 4 dạng sóng khác nhau (giống như máy phát sóng)

+ Sóng Sin
+ Sóng vuông (Square)
+ Sóng răng cưa (Sawtood)
+ Sóng ngẫu nhiên (Random)

Với Pulse Generator tạo chuỗi xung hình chữ nhật. Biên độ và tần số có thể khai báo tùy ý. Đối với Pulse Generator ta còn có khả năng chọn tỉ lệ cho bề rộng xung(tính bằng phần trăm cho cả chu kì)

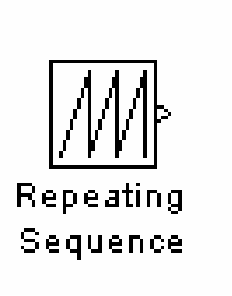


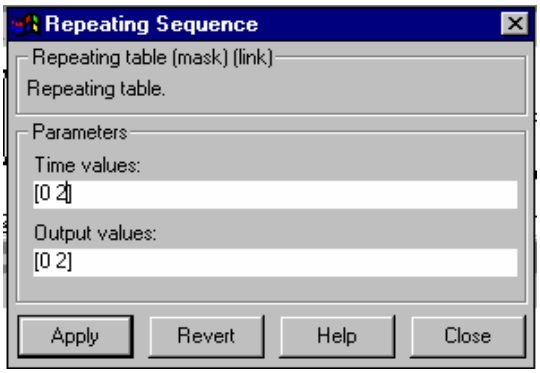


d. Repeating sequence

Khối này cho phép ta tạo nên một tín hiệu tuần hoàn tùy ý. Tham số Time values phải là một vecter thời gian với các giá trị đơn điệu tăng.

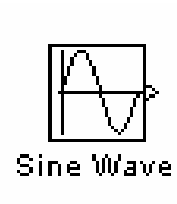
Vecter biến ra Output values phải có kích cỡ (chiều dài) phù hợp với chiều dài của tham số Time values.Giá trị lớn nhất của vecter thời gian quyết định chu kì lặp lại của vecter biến ra.

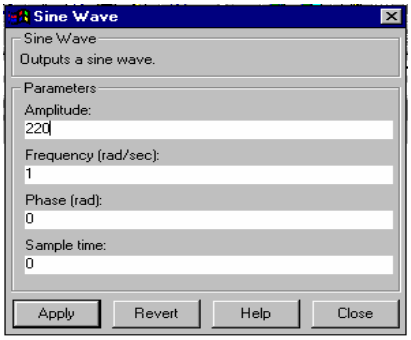




e. Sine Wave

Khối này được sử dụng để tạo tín hiệu hình Sin cho cả hai loại mô hình: liên tục (tham số Sample time = 0)và gián đoạn (tham số sample time = 1)

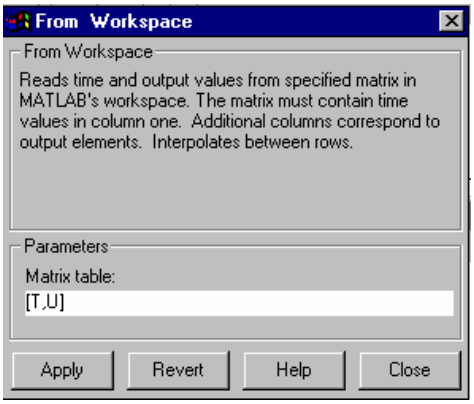




Màn hình cài đặt thông số cho khối Sine Wave

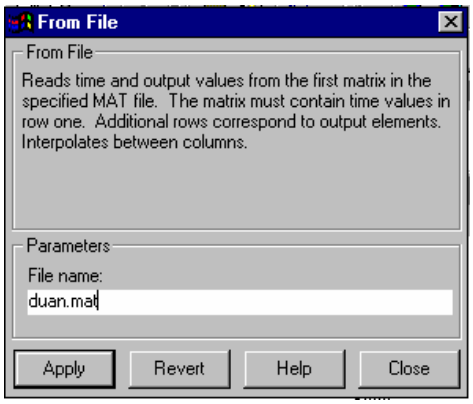
f. From Workspace

Khối From Workspace có nhiệm vụ lấy số liệu từ cửa sổ Matlab Workspace để cung cấp cho mô hình Simulink. Các số liệu lấy vào phải có dạng của biểu thức Matlab, khai báo tại dòng Data.



g. From File

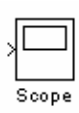
Bằng khối From File ta có thể lấy số liệu từ một MAT-File có sẵn. MAT-File có thể là kết quả của một lần mô phỏng trước đó, đã được tạo nên và cất đi nhờ khối To file trong sơ đồ Simulink



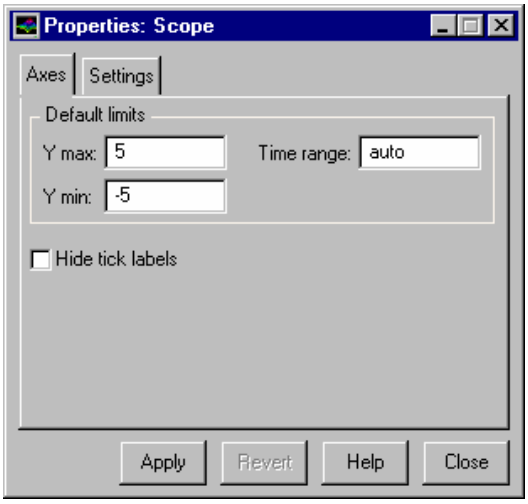
2. Thư viện Sinks

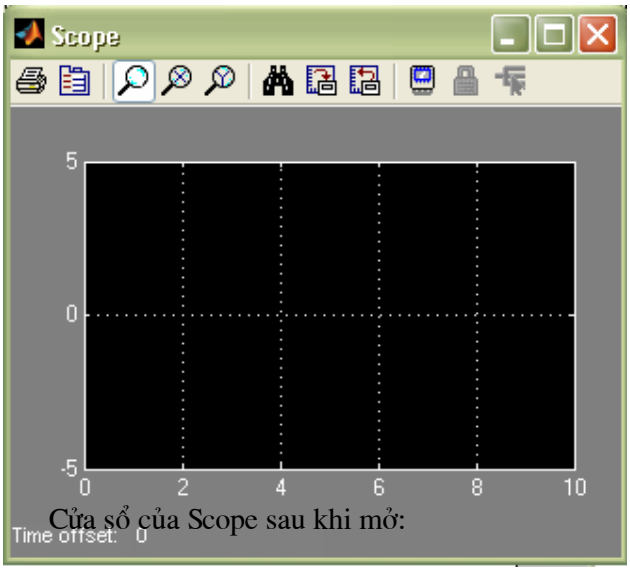
Thư viện này bao gồm các khối xuất chuẩn của Simulink. Ngoài khả năng hiển thị đơn giản bằng số, còn có các khối dao động kí để biểu diễn các tín hiệu phụ thuộc thời gian hay biểu diễn hai tín hiệu trên hệ tọa độ XY.

a. Scope



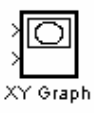
Nhờ khối Scope ta có thể hiển thị các tín hiệu của quá trình mô phỏng. Khi nhấn vào nút Properties, hộp thoại Scope Properties (đặc điểm của Scope) sẽ mở ra. Chọn general ta có thể đặt chế độ cho các trục. Khi đặt Number of axes > 1, cửa sổ Scope sẽ có nhiều đồ thị con giống tương tự như lệnh Subplot của Matlab. Nếu điền một số cụ thể vào ô time range, đồ thị sẽ chỉ được biểu diễn tới thời điểm do giá trị của số xác định.





Cửa sổ của Scope sau khi mở:

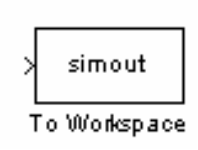
b. XY Graph

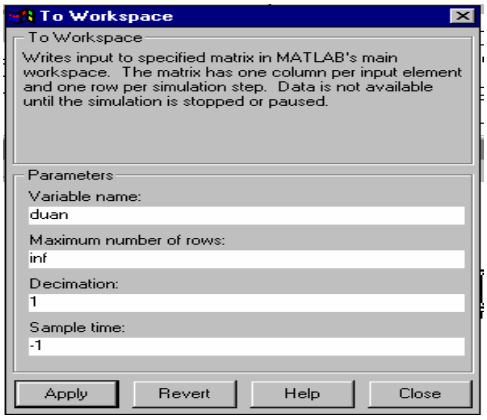


Khối này biểu diễn hai tín hiệu đầu vào trên hệ tọa độ XY dưới dạng đồ họa Matlab đầu vào thứ nhất (bên trên). Ứng với trục X đầu thứ hai ứng với trục Y.

c. To Workspace

Khối To Workspace gửi số liệu ở đầu vào của khối tới môi trường Matlab Workspace dưới dạng mảng (Array), Stracture hay Stracture with time và lấy chuỗi kí tự khai tại variable name để đặt tên cho tập số liệu được ghi.

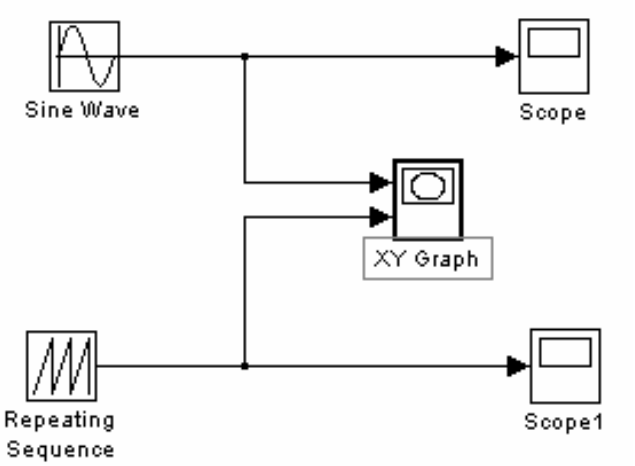




d. To File

Khối này giúp ta cất tập số liệu (mảng hay ma trận) ở đầu vào của khối cùng với véctơ thời gian dưới dạng Mat- File. Array định dạng giống như định dạng mà khối From File cần, vì vậy số liệu do To File cất có thể được From File đọc trực tiếp mà không cần phải xử lí gì.

Ví dụ: hai tín hiệu hình Sin và tín hiệu hình răng cưa được hiển thị độc lập, đồng thời trên hệ tọa độ XY, được thiết lập như hình bên.



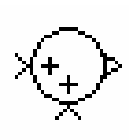
3. Thư viện Math

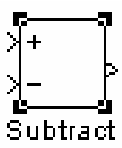
Thư viện này có một số khối có chức năng ghép toán học các tín hiệu khác nhau, có những khối đơn giản chỉ nhằm cộng hay nhân tín hiệu còn có các hàm phức tạp như lượng giác và logic…Sau đây ta xét chức năng của một số khối quan trọng trong thư viện này.

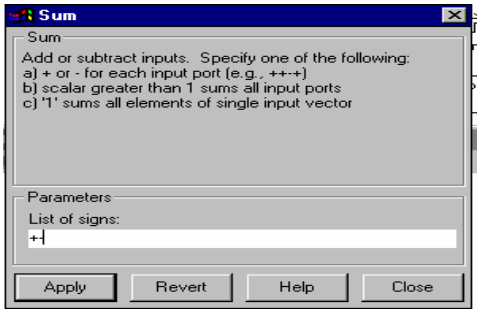
a. Sum

Tín hiệu ra của khối Sum là tổng của các tín hiệu đầu vào (Ví dụ như tín hiệu đầu vào là các tín hiệu hình Sin thì tín hiệu đầu ra cũng là các tín hiệu hình Sin). Khối Sum cũng có thể tính tổng từng phần tử(ví dụ tín hiệu vào gồm hai tín hiệu: Sin(x)

và [5 9 3] thì tín hiệu ra sẽ có dạng [Sin(x)+5 Sin(x)+9 Sin(x)+3])

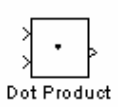






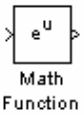
b. Product và Dot Product

Khối Product thực hiện phép nhân từng phần tử hay nhân ma trận cũng như phép chia giữa các tín hiệu vào (dạng 1-D hay 2- D) của khối ví dụ: nếu một khối Product có tham số Number of Inputs = */*, với ba tín hiệu vào là 5, sinx và
[4 4 5 6] khi ấy tín hiệu đầu ra có dạng [20/Sinx 20/sinx 25/Sinx 30/Sinx].



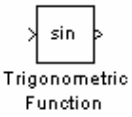
Khối Dot Product tính tích vô hướng của các Vector đầu vào. Giá trị đầu ra của khối tương đương với lệnh Matlab y = Sum(conj(u_1)*u_2).

c. Math Function và Trigonometric Function

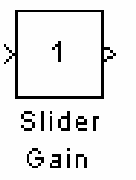


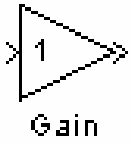
Cả hai khối này đều có thể xử lí tín hiệu 2-D. Khối Math Function có một lượng lớn các hàm toán đã được chuẩn bị sẵn cho phép ta lựa chọn theo nhu cầu sử dụng. Còn khối Trigonometric Function có tất cả các hàm lượng giác quan trọng.

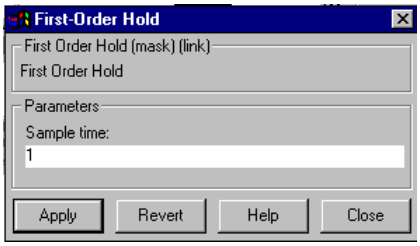
d. Gain và Slider Gain



Khối Gain có tác dụng khuyếch đại tín hiệu đầu vào (định dạng 1-D hay 2-D) bằng biểu thức khai báo tại ô Gain. Biểu thức đó chỉ có thể là một biến hay một số biến. Biến đó phải tồn tại trong môi trường Matlab Workspace thì khi ấy Simulink mới tính toán được với biến.





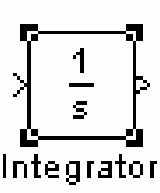


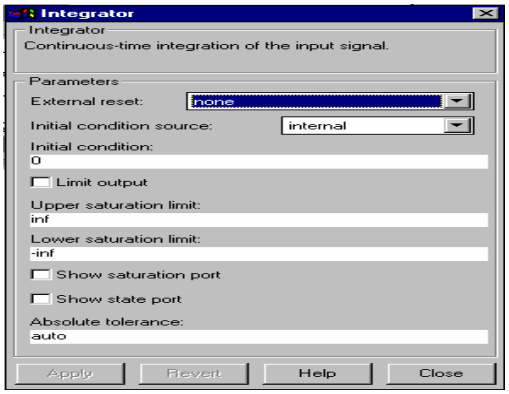
Khối Slider Gain cho phép thay đổi hệ số khuyếch đại vô hướng trong quá trình mô phỏng.

4. Thư viện Constinuous

a.Integrator

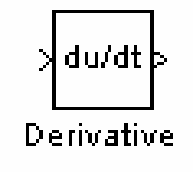
Khối Integrator lấy tích phân tín hiệu đầu vào của khối. Giá trị ban đầu được khai báo hoặc trực tiếp tại hộp thoại Block Parameters hoặc thông qua chọn giá trị Internal tại ô Initial condition Source để sau đó điền giá trị ban đầu vào dòng viết của ô Initial condition. Đầu ra của khối Integrator có thể được một tín hiệu bên ngoài lập về một giá trị ban đầu biến trạng thái của khối. Biến trạng thái của khối thực chất đồng nhất về giá trị với biến đầu ra nhưng với Simulink tính hai biến đó (biến ra và biến trạng thái) tại những thời điểm ít nhiều có khác nhau. Nếu mô hình Simulink chứa các biến trạng thái chênh lệch nhau về kích cỡ giá trị, khi ấy nên khai báo tham số Absolute Tolerance riêng rẽ thêm cho từng khối Integrator của mô hình, mặc dù đã khai báo Absolute Tolerance chung tại hộp thoại Simulation Parameters. Việc khai báo thêm sẽ buộc Simulink bảo đảm đúng giá trị sai số yêu cầu đối với từng khối.





b. Derivative

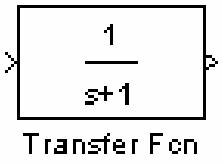
Khối này cho phép ta tính đạo hàm tín hiệu đầu vào. Tín hiệu tìm được ở đầu ra có dạng Δu/Δt với Δ là biến thiên của đại lượng cần tính kể từ bước tích phân liền trước đó. Giá trị của ra ban đầu là 0

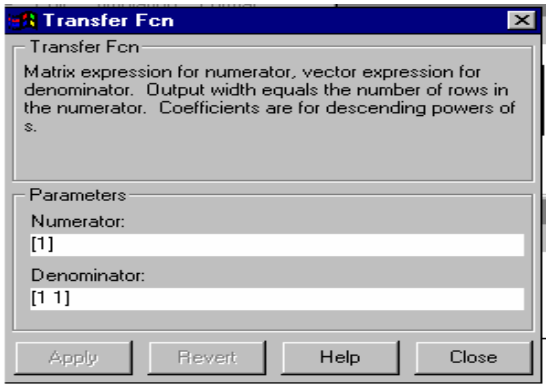


c. Transfer Fcn và Zero-Pole

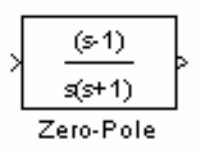
Khối Transfer Fcn cho phép có thể mô hình hóa hàm truyền đạt của một hệ tuyến tính. Tham số của khối là các hệ số của đa thức tử số và mẫu số, khai báo theo thứ tự số mũ của s giảm dần. Bậc của mẫu số phải lớn hoặc bằng bậc của tử số. Ví dụ: nếu nhập cho tử số [5 7 3 1] và mẫu số [6 8 3 2 1] khối sẽ tạo ra hàm truyền đạt:

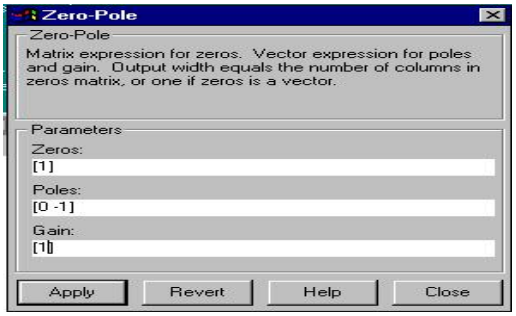
$$W(s) = \frac{y(s)}{u(s)} = \frac{5s^3 + 7s^2 + 3s + 1}{6s^4 + 8s^3 + 3s^2 + 2s + 1}$$





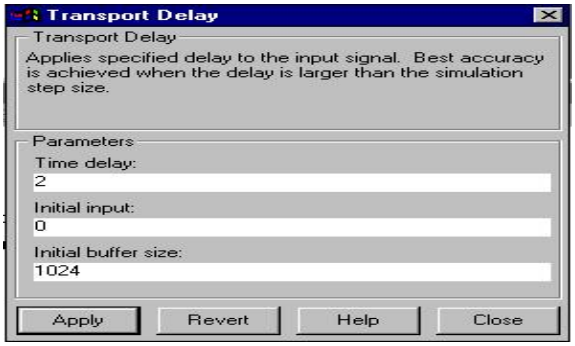
Khối Zero Pole sẽ tạo nên từ các tham số Zeros, Poles và Gain một hàm truyền đạt dưới dạng hệ số hóa theo điểm không, điểm cực.



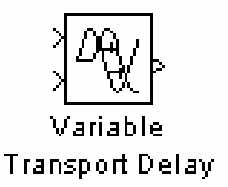


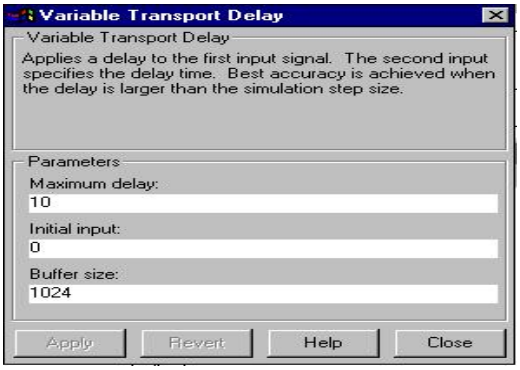
d. Transport Delay và Variable Transport Delay

Khối Trasnport Delay làm trễ tín hiệu vào khoảng thời gian ≥ 0 khai báo tại ô Time Delay trước khi xuất tới đầu ra. Chỉ đến khi thời gian mô phỏng bắt đầu vượt quá thời gian trễ (so với lúc bắt đầu mô phỏng), khối Transport Delay mới xuất giá trị khai tại Initial Input tới đầu ra.



Bằng khối Variable Transport Delay có thể điều khiển trễ tín hiệu một cách rất linh hoạt: tín hiệu chứa thời gian trễ được đưa tới đầu vào thứ hai (đầu vào phía dưới) của khối. Tại ô Maxmum Delay ta phải khai một giá trị trễ tối đa, có tác dụng giới hạn (chặn trên) giá trị của tín hiệu điều khiển thời gian trễ.



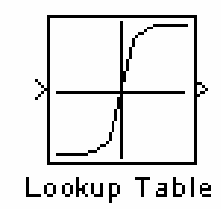


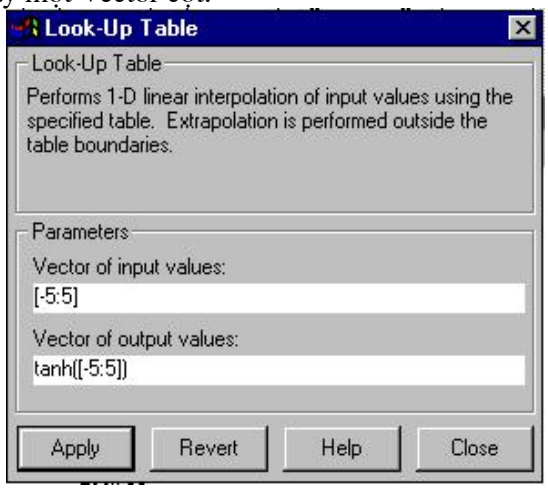
5. Thư viện Tables

a. Lookup Table

Khối này tạo tín hiệu ra từ tín hiệu vào trên cơ sở thông tin cất trong một bảng tra (Vector of input values x Vector of output values). Nếu giá trị hiện tại của tín hiệu vào trùng với một giá trị thuộc Vector of input values, giá trị tương ứng trong bảng thuộc Vevtor of output values sẽ được đưa tới đầu ra. Nếu giá trị của tín hiệu vào nằm giữa hai giá trị thuộc Vector of output values, Simulink thực hiện nội suy hai giá trị tương ứng của Vector of output values. Nếu giá trị của tín hiệu vào bé hơn / lớn hơn giá trị đầu tiên / giá trị cuối cùng của Vector of input values,

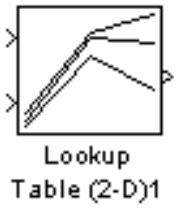
Simulink sẽ thực hiện ngoại suy hai gía trị đầu tiên / cuối cùng của Vector of output values. Vector of input values có thể là một Vector hàng hay một Vector cột.

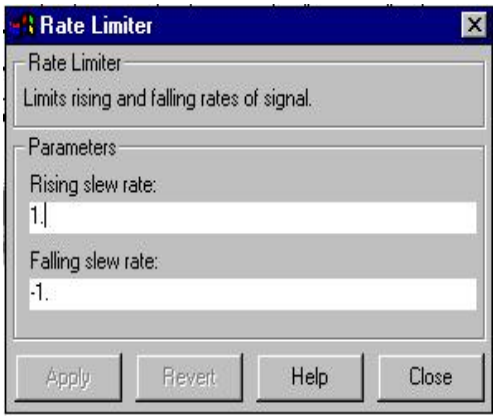




b. Lookup Table (2-D)1

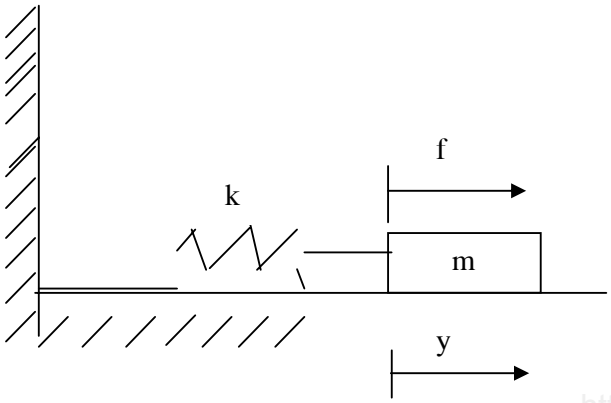
Khối này cho phép tạo nên một bảng tra hai chiều. Bằng tham số Table ta khai báo một ma trận cất các tín hiệu đầu ra. Muốn tìm được giá trị đưa tới đầu ra ta cần biết Row để tìm hàng và Column để tìm cột của ô trong giá trị ma trận. Tín hiệu đặt ở đầu vào phía trên được so với Row tín hiệu đặt ở đầu vào phía dưới được so với Column.





II. Áp dụng Simulink vào thiết kế và phân tích

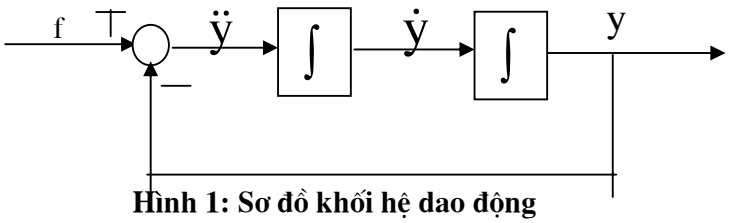
Trong việc khảo sát những ứng dụng ta thử xây sơ đồ mô phỏng cho hệ dao động lò xo khối lượng sau:



Hệ lò xo khối lượng trên được mô tả bởi phương trình vi phân:

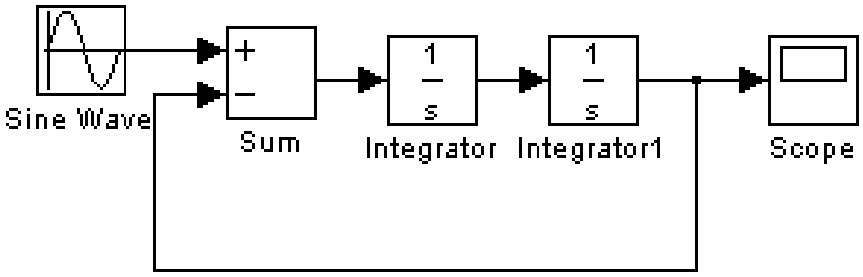
$$m\ddot{y}+ky= f$$

m: khối lượng; k: độ cứng lò xo
Từ đó ta có sơ đồ khối sau:

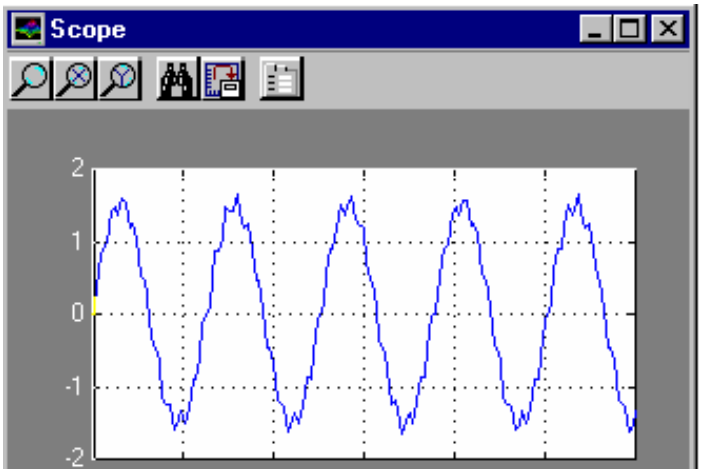


Hình 1: Sơ đồ khối hệ dao động

Sau đó ta thử xây dựng sơ đồ mô phỏng trong Simulink



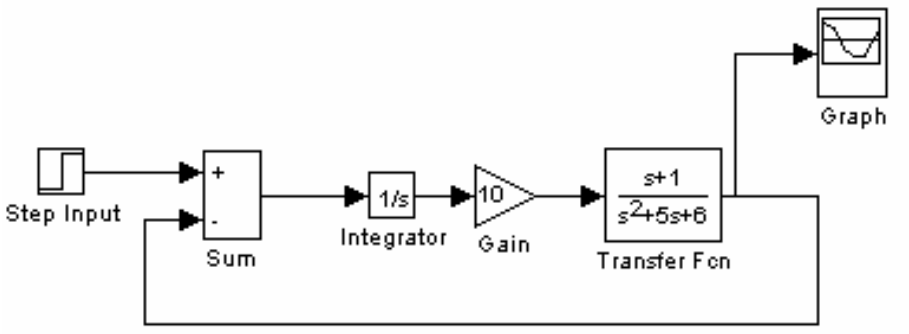
Hình 2. Sơ đồ khối mô phỏng hệ khối lượng lò xo

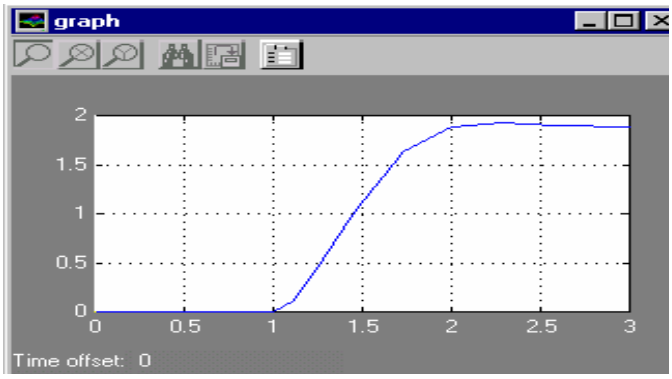


Hình 3. Dạng sóng ngõ ra của Scope

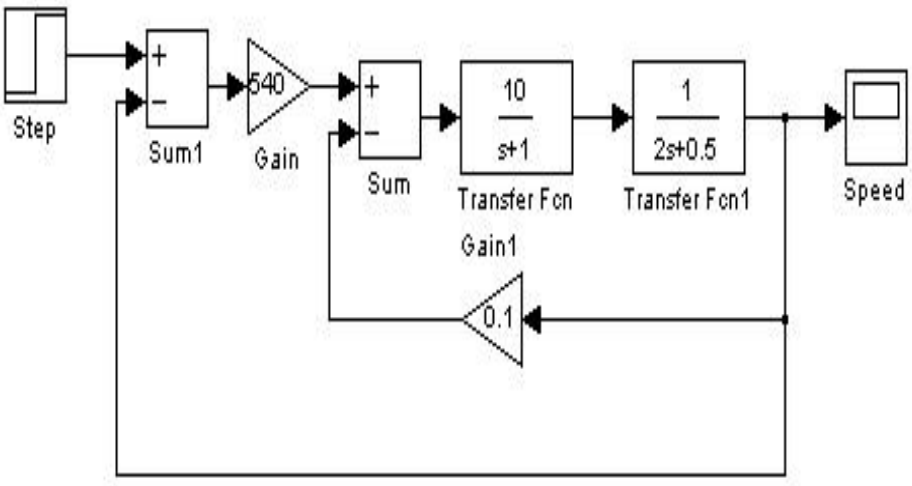
Từ sơ đồ thiết kế ta có thể thêm vào các khối để khảo sát hệ như: hệ số cứng (gain), thay đổi dạng sóng...

Trong các sách bài tập điều khiển tự động có các khối nhưng không biết được đáp ứng của hệ như thế nào. Việc dùng Simulink để khảo sát rất thuận tiệncho việc phân tích bài toán

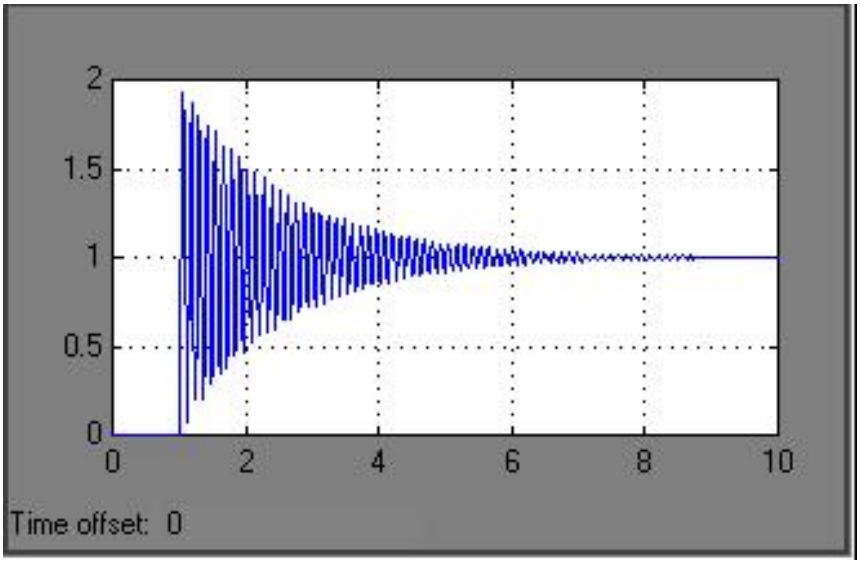




Hình 5. Đáp ứng của sơ đồ ở hình 2.35 và 2.36 là sơ đồ của động cơ điện và đáp ứng vận tốc quay



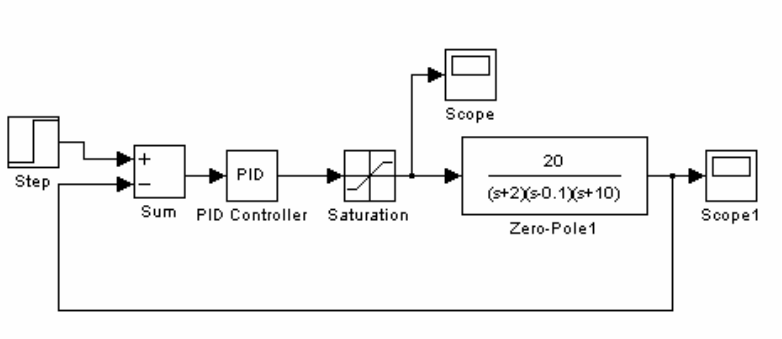
Hình 6. Sơ đồ khối của động cơ điện



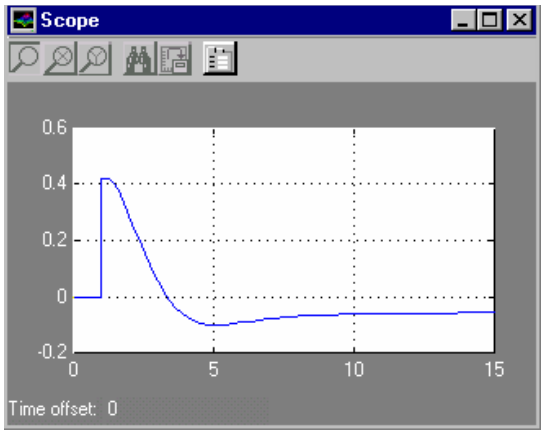
Hình 7. Đáp ứng vận tốc quay

Trong điều khiển tự động thên vào khâu các P, PI, PID làm cho hệ thống hoạt động tốt và ổn định hơn.

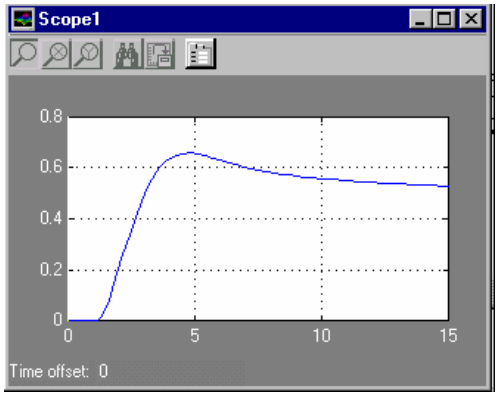
Các sơ đồ điều khiển và mô phỏng

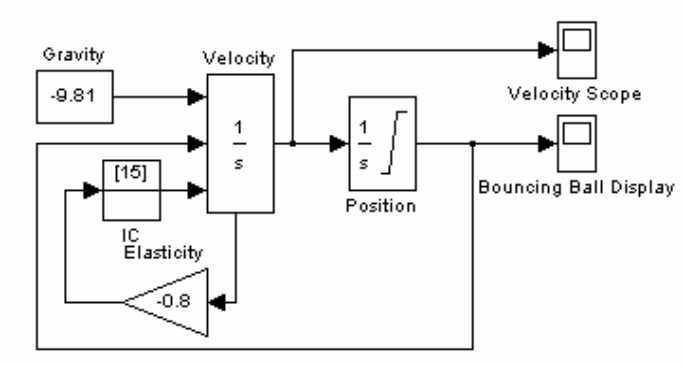


Hình 8. Hệ thống điều khiển có khâu PI

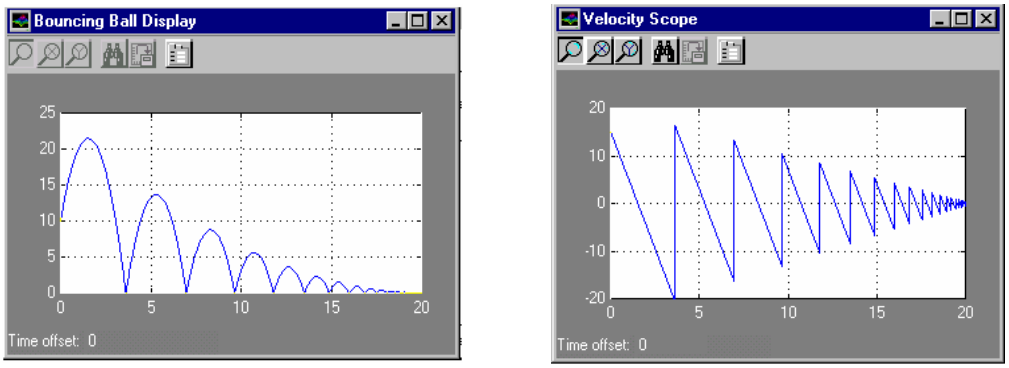


Hình 9. Đáp ứng tại khâu bão hòa

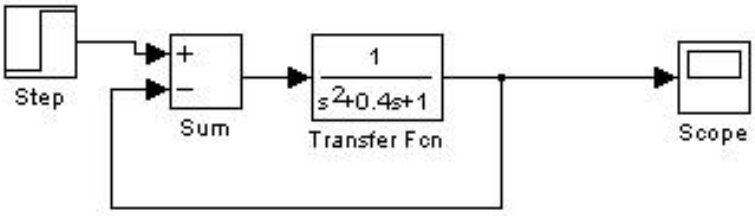




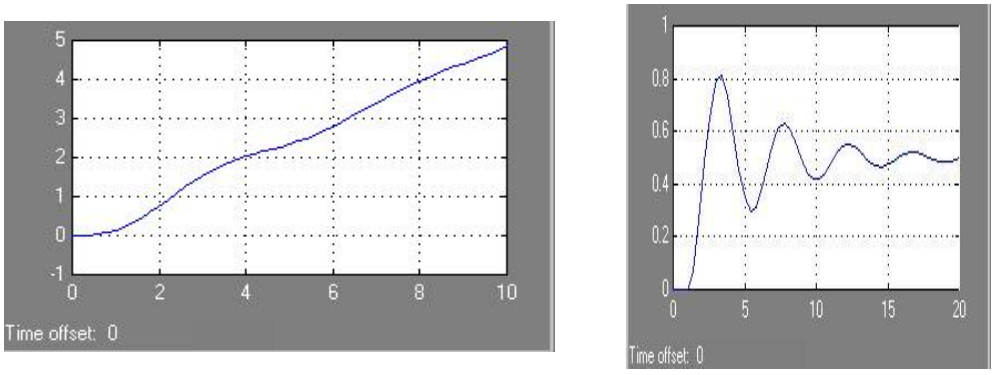
Hình 10. Hệ thống mô phỏng của chuyển động rơi quả banh



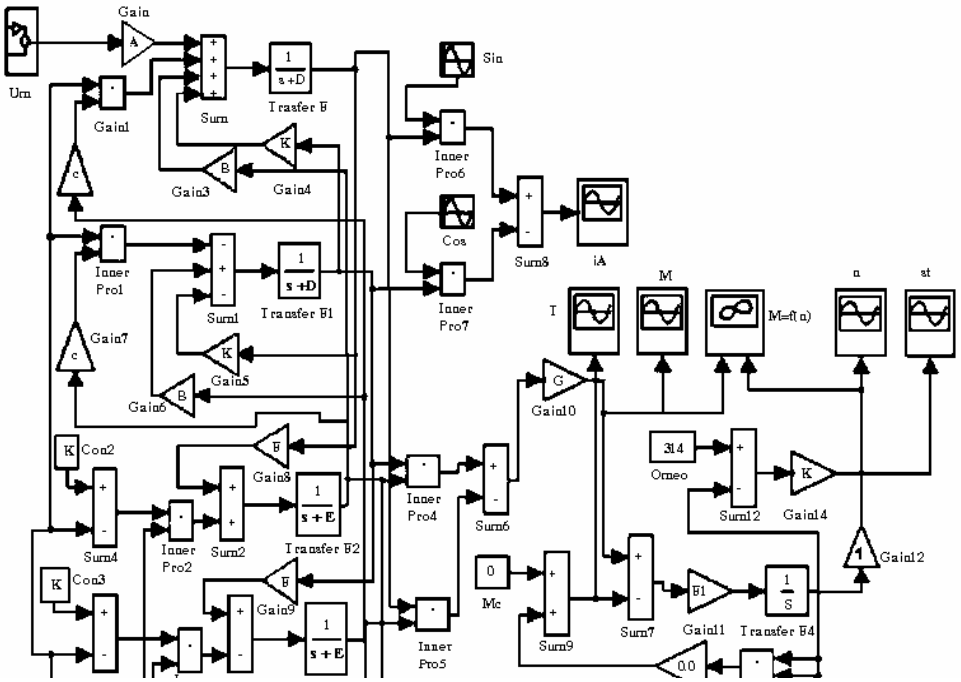
Hình 11. Kết quả mô phỏng



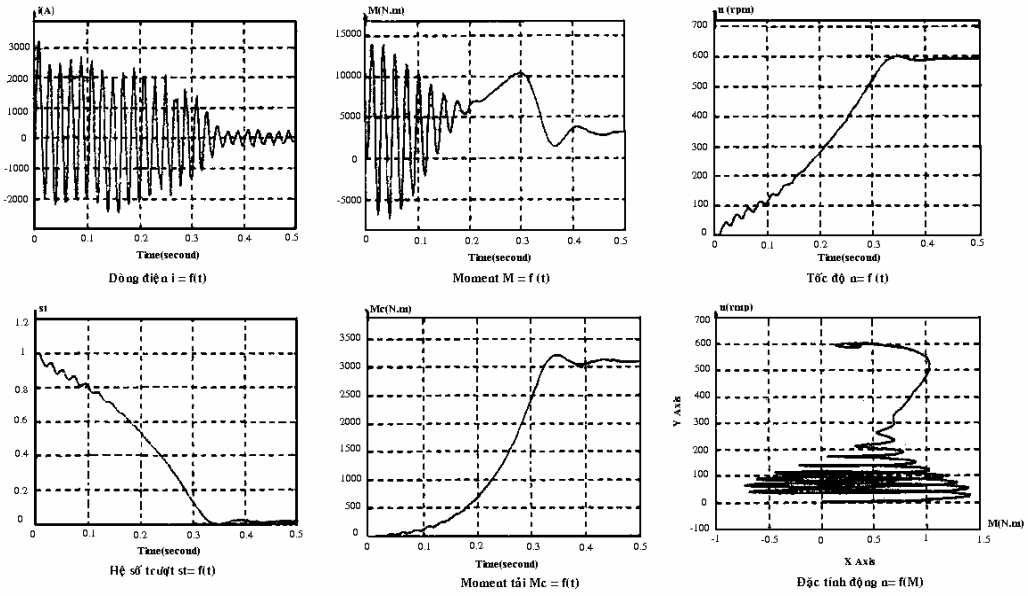
Hình 12. Sơ đồ mô phỏng một khâu bậc hai



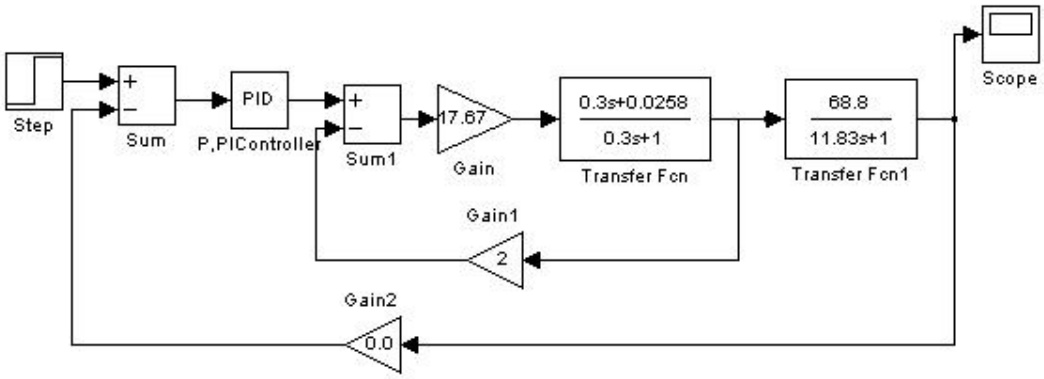
Hình 13: Đáp ứng của khâu bậc hai dưới ngõ vào là hàm dốc và bước

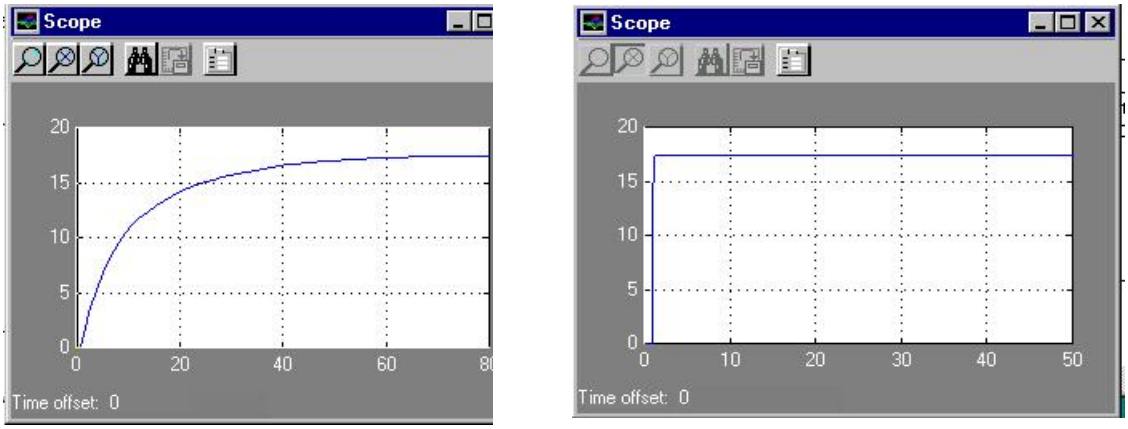


Hình 14. Mô hình động cơ không đồng bộ



Sơ đồ điều khiển động cơ DC kích từ độc lập đang kéo tải với các khâu P,PI,PID





Hình 15 : Đáp ứng của động cơ với khâu P