Th.s. NGUYỄN CHÍ NGÔN

Thí nghiệm CAD

(Computer-Aided Design)

Được biên soạn trong khuôn khổ dự án ASVIET002CNTT
"Tăng cường hiệu quả đào tạo và năng lực tự đào tạo của sinh viên khoa Công nghệ Thông tin - Đại học Cần thơ"

ĐẠI HỌC CẦN THƠ - 12/2003

Lời nói đầu

Cùng với sự phát triển nhanh chóng của máy tính, CAD (Computer-Aided Desgin) được xây dựng ngày càng hoàn thiện và ứng dụng trong hầu hết các lĩnh vực khoa học kỹ thuật. Đối với chuyên ngành Điện tử, nhiều phần mềm CAD cho phép thiết kế mạch, mô phỏng và vẽ mạch in một cách nhanh chóng và hiệu quả như OrCAD/Pspice, Multisim (Electronics WorkBench), MicroSim, ExpeditionPCB, …

Tuy nhiên, đây là các phần mềm đóng gói chỉ được ứng dụng trong chuyên môn hẹp là Điện tử, nó không cho phép lập trình mô phỏng các hệ thống động (Dynamic systems) bất kỳ khác. Vì vậy, chương trình đào tạo môn CAD cho sinh viên Điện tử chuyên ngành Viễn thông và Tự động hóa đã hướng tới phần mềm **Matlab**. Đây là một ngôn ngữ lập trình cấp cao dạng nguồn mở, nó hỗ trợ rất nhiều thư viện chức năng chuyên biệt từ Toán học, Kinh tế, Logic mờ, Truyền thông, Điều khiển tự động, … đến điều khiển phần cứng cho các thiết bị. Đồng thời, nó cho phép người sử dụng bổ sung các công cụ tự tạo làm phong phú thêm khả năng phân tích, thiết kế và mô phỏng các hệ thống động liên tục và rời rạc, tuyến tính và phi tuyến bất kỳ. Với những ưu điểm nổi bậc của mình, Matlab đã được nhiều trường Đại học hàng đầu trên thế giới áp dụng và giảng dạy.

Tham vọng thì nhiều nhưng trong phạm vi 30 tiết thực hành, chúng ta chỉ có thể đề cập đến những vấn đề cơ bản nhất. Hy vọng từ đó sinh viên tự nghiên cứu, học hỏi để có thể thiết kế và mô phỏng các mô hình Hệ thống Viễn thông hay các Hệ điều khiển tự động. Đồng thời cũng nắm sơ lược về các phần mềm mô phỏng mạch và vẽ mạch in như Multisim, OrCad,…

Giáo trình này gồm 7 bài, mỗi bài 5 tiết. Sinh viên chọn 6 bài để thực tập:

1. Thao tác trong cửa sổ lệnh của Matlab.
2. Hàm và Script files.
3. Symbolic và Simulink.
4. Mô hình hệ thống Viễn thông. *(sinh viên chuyên ngành VT)*
5. Mô hình hệ thống Điều khiển tự động. *(sinh viên chuyên ngành ĐKTĐ)*
6. Tạo giao diện trong Matlab.
7. Thiết kế - Mô phỏng và vẽ mạch in *(các sinh viên không chọn bài 4 hoặc 5)*

Mặc dù đã hết sức cố gắng, song do trình độ hạn chế của người viết mà nhiều vấn đề chắc chưa được trình bày tốt cũng như chưa bố cục hợp lý. Xin chân thành cảm ơn mọi ý kiến đóng góp của sinh viên và các bạn đồng nghiệp.

TcAD, tháng 11 năm 2003

Nguyễn Chí Ngôn

Địa chỉ liên hệ:

Bộ môn Viễn thông và Tự động hóa
Khoa Công nghệ Thông tin, Đại học Cần thơ
01 Lý Tự Trọng, Tp. Cần thơ, tỉnh Cần thơ
Tel: (71) 831301 Fax: (71) 830841
Email: ncngon@ctu.edu.vn
URL: http://www.cit.ctu.edu.vn/department/ac/ncngon.html

Mục lục

Lời nói đầu **2**

Mục lục **3**

BÀI 1: THAO TÁC TRONG CỬA SỔ LỆNH CỦA MATLAB **5**

I. Mục tiêu 5

II. Tham khảo 5

III. Thực hành 5

III.1 Ma trận 5

III.2 Vectơ 6

III.3 Các đa thức 7

III.4 Đồ họa 7

IV. Tự chọn 12

BÀI 2: HÀM VÀ SCRIPT FILES **13**

I. Mục tiêu 13

II. Tham khảo 13

III. Thực hành 13

III.1 Script files 13

III.2 Sử dụng các hàm xây dựng sẵn 15

III.3 Xây dựng hàm 17

IV. Tự chọn 21

BÀI 3: SYMBOLIC VÀ SIMULINK **22**

I. Mục tiêu 22

II. Tham khảo 22

III. Thực hành 22

III.1 Symbolic 22

III.2 Simulink 24

IV. Tự chọn 30

BÀI 4: MÔ HÌNH HỆ THỐNG VIỄN THÔNG **32**

I. Mục tiêu 32

II. Tham khảo 32

III. Thực hành 32

III.1 Hệ thống thông tin liên tục (Analog Communications) 34

III.2 Hệ thống thông tin rời rạc (Digital Communications).....36
IV. Tự chọn.....37
BÀI 5: MÔ HÌNH HỆ THỐNG ĐIỀU KHIỂN TỰ ĐỘNG 38
I. Mục tiêu.....38
II. Tham khảo.....38
III. Thực hành.....38
III.1 Hàm truyền và phương trình trạng thái của hệ thống.....39
III.2 Bộ điều khiển PID.....42
III.3 Hiệu chỉnh thông số của bộ điều khiển PID.....45
IV. Tự chọn.....47
BÀI 6: TẠO GIAO DIỆN TRONG MATLAB 49
I. Mục tiêu.....49
II. Tham khảo.....49
III. Thực hành.....49
IV. Tự chọn.....55
BÀI 7: THIẾT KẾ – MÔ PHỎNG MẠCH VÀ VẼ MẠCH IN 57
I. Mục tiêu.....57
II. Tham khảo.....57
III. Thực hành.....57
III.1 Multisim.....57
III.2 OrCAD.....59
IV. Tự chọn.....64

BÀI 1

THAO TÁC TRONG CỬA SỔ LỆNH CỦA MATLAB

I. Mục tiêu

Bài thí nghiệm này giúp sinh viên làm quen nhanh với Matlab 6.5 bằng các thao tác đơn giản trên ma trận, vectơ, biểu thức toán học, các lệnh đồ họa, …, thực hiện ngay trên cửa sổ lệnh (command window) của Matlab.

II. Tham khảo

[1]. Nguyễn Hứa Duy Khang, Bài giảng môn CAD, Bộ môn Viễn thông & Tự động hóa, khoa Công nghệ thông tin, Đại học Cần thơ, 2001.

[2]. The Mathworks Inc., Matlab Notebook User's Guide, 2003.

[3]. Nguyễn Hoài Sơn - Đỗ Thanh Việt - Bùi Xuân Lâm, Ứng dụng MATLAB trong tính toán kỹ thuật, Tập 1, NXB ĐHQG Tp. HCM, 2000

[4]. Nguyễn Hữu Tình - Lê Tấn Hùng - Phạm Thị Ngọc Yến - Nguyễn Thị Lan Hương, Cơ sở Matlab & ứng dụng, NXB KH và Kỹ thuật, 1999.

[5]. http://www.facstaff.bucknell.edu/maneval/help211/exercises.html

[6]. http://www.glue.umd.edu/~nsw/ench250/matlab.htm

III. Thực hành

Từ cửa sổ lệnh của Matlab, sinh viên lần lượt thực hiện các thao tác sau:

III.1 Ma trận

Để tạo ma trận trong Matlab ta chỉ cần liệt các phần tử của ma trận trong cặp dấu ngoặc vuông ([…]). Các phần tử trên cùng hàng được phân biệt bởi dấu phẩy (,) hoặc khoảng trắng (space). Các hàng của ma trận, phân cách nhau bởi dấu chấm phẩy (;). Ví dụ, nhập ma trận A có 4 hàng, 4 cột như sau:

```
>> A=[16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
>> size(A)
```

Để truy xuất đến từng phần tử của ma trận ta dùng chỉ số phần tử tương ứng. Ví dụ, phần tử ở hàng thứ 2, cột thứ 3 của A là A(2,3).

```
>> A(2,3)
```

① Cho ma trận A=[2 4 1; 6 7 2; 3 5 9], sinh viên dùng các lệnh cần thiết để:

a. Lấy dòng đầu tiên của ma trận A.

b. Tạo ma trận B bằng 2 dòng cuối cùng của A.
c. Tính tổng các phần tử trên các cột của A. (gợi ý: tính tổng các phần tử trên cột 1: *sum(A(:,1))*).
d. Tính tổng các phần tử trên các dòng của A.

② Cho ma trận A=[2 7 9 7; 3 1 5 6; 8 1 2 5], sinh viên giải thích kết quả của các lệnh sau:

a. A'
b. A(:,[1 4])
c. A([2 3],[3 1])
d. reshape(A,2,6)
e. A(:)
f. [A A(end,:)]
g. A(1:3,:)
h. [A ; A(1:2,:)]
i. sum(A)
j. sum(A')
k. [[A ; sum(A)] [sum(A,2) ; sum(A(:))]]

③ Giải hệ phương Ax=b, với: $A=\begin{bmatrix}1 & 0 & -1\\ 2 & 5 & 3\\ 3 & -1 & 0\end{bmatrix}$ và $b=\begin{bmatrix}1\\ 1\\ -2\end{bmatrix}$. Gợi ý: x=A\b.

III.2 Vectơ

Vectơ thực chất cũng là ma trận có kích thước (n x 1) hay (1 x n), nên ta có thể tạo ra vectơ như cách tạo ra ma trận. Ngoài ra, có thể dùng một số cách sau:

```
>>x=0:0.1:1
>>y=linspace(1, 10, 20)   % vecto 20 phan tu cach deu nhau tu 1 den 10
>>z=rand(10,1)
```

① Cho vectơ x = [3 1 5 7 9 2 6], giải thích kết quả của các lệnh sau:

a. x(3)
b. x(1:7)
c. x(1:end)
d. x(1:end-1)
e. x(6:-2:1)
f. x([1 6 2 1 1])
g. sum(x)

② Sinh viên hãy tạo một vectơ x có 100 phần tử, sao cho:

$$x_n = \frac{(-1)^{n+1}}{(2n-1)}.$$

Gợi ý: Tạo vectơ n có 100 phần tử từ 1 đến 100, dùng toán tử dấu chấm (.) để xác định x.

III.3 Các đa thức

Các đa thức trong Matlab được mô tả bằng các vectơ hàng với các phần tử của vectơ chính là các hệ số của đa thức, xếp theo thứ tự số mũ giảm dần. Ví dụ, đa thức m = s^4-s^3+4s^2-5s-1 được biểu diễn là:

```
>>m=[1  -1  4  5  -1]
```

Để xác định giá trị của đa thức, ta dùng lệnh **polyval**. Ví dụ, xác định giá trị của đa thức tại điểm s=2:

```
>>polyval(m,2)
```

Để xác định nghiệm của đa thức, ta dùng lệnh **roots**. Ví dụ:

```
>>roots(m)
```

① Cho phương trình x^2-4x+5=0, giải phương trình theo 2 cách, cách 1 – tính delta theo phương pháp cổ điển, cách 2 – dùng hàm **roots**, hãy so sánh kết quả.

a. Cách1:

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

b. Cách 2:

```
>>m=[a b c];
>>x=roots(m)
```

Hãy thay đổi các giá trị khác nhau của a, b và c tương ứng trong 2 cách giải trên. So sánh kết quả và nhận xét.

② Giải phương trình x^3- 2x^2+4x+5=0. Kiểm chứng kết quả thu được bằng hàm **polyval**. Sinh viên có nhận xét gì về kết quả kiểm chứng.

③ Lặp lại câu ② cho phương trình x^7-2=0.

④ Sinh viên thử dùng hàm **poly** để tạo đa thức từ các nghiệm cho trước.

```
>>help poly
```

III.4 Đồ họa

Matlab hỗ trợ chế độ đồ họa rất mạnh, bao gồm đồ họa 2D và 3D, với các trục tọa độ tuyến tính và phi tuyến bất kỳ.

III.4.1. Đồ họa 2 D

Đồ họa 2D chủ yếu dựa trên lệnh **plot**. Để được giúp đỡ, ta gõ:

```
>>help plot
```

① Vẽ đồ thị hàm số y_1=sinx.cos2x và hàm số y_2=sinx^2 trong [0-2π], trên cùng hệ trục tọa độ, ta lần lượt thực hiện như sau:

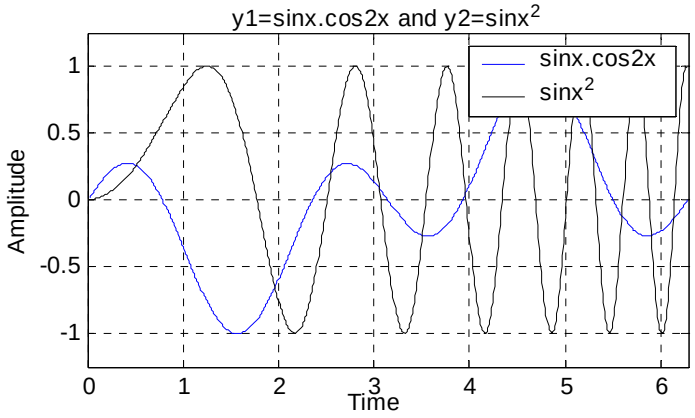
```
>>x=0:0.01:2*pi;
>>y1=sin(x).*cos(2*x);          %nhan tuong tung tung phan tu
>>plot(x,y1)
>>grid on                       %hien thi luoi
```

Sau khi thu được đồ thị hàm y1, để vẽ y2 trên cùng đồ thị, ta thực hiện:

```
>>hold on                       %giu hinh, mac nhien la hold off
>>y2=sin(x.^2);                 %luy thua tung phan tu
>>plot(x,y2,'k')                %duong ve co mau den
 >>axis([0  4*pi  -1.25  1.25])  %dinh lai toa do hien thi
```

Ta có thể đặt nhãn cho các trục cũng như tiêu đề cho đồ thị:

```
>>xlabel('Time')
>>ylabel('Amplitude')
>>title('y1=sinx.cos2x and y2=sin(x^2)')
>>legend('sinx.cos2x','sinx^2')
```



Hình 1.1 – Biểu diễn đồ thị các hàm số trên cùng hệ trục tọa độ

② Matlab hỗ trợ rất nhiều thuộc tính đồ họa, để có thể kiểm soát các thuộc tính này ta cần dùng đến thẻ đồ họa. Ví dụ:

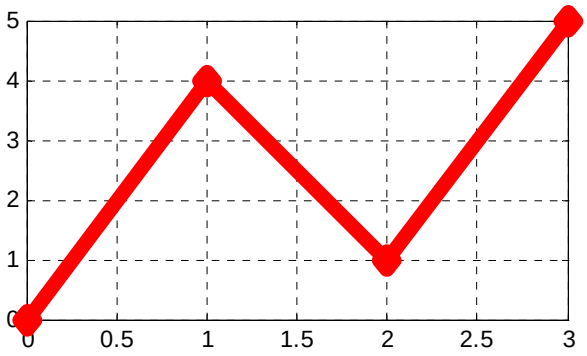
```
>>close all
>>x=[0 1 2 3];
>>y=[0 4 1 5];
>>h=plot(x,y)
```

h chính là thẻ đồ họa của hàm plot, để thấy các thuộc tính đồ hoạ, ta dùng lệnh:

```
>>set(h)
```

Bây giờ ta thử đặt một số thuộc tính đồ họa cho h.

```
>>set(h,'Color', 'r')                    %dat lai mau do
>>set(h,'LineWidth',6)                   %dat do rong duong
>>set(h,'Marker','v','MarkerSize',6)
```

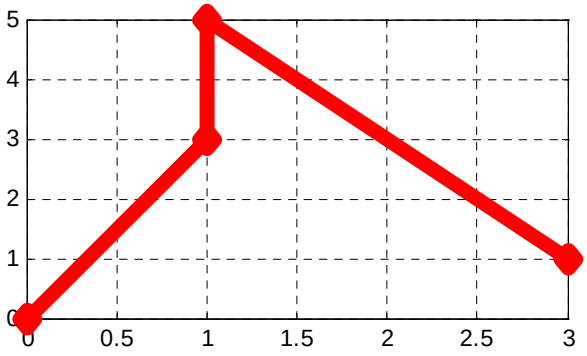


Hình 1.2 – Thay đổi thuộc tính đường biểu diễn

Dữ liệu dùng để vẽ được đặt trong 'XData' và 'YData'.

```
>> set(h,'XData',[0 1 1 3])
>>set(h,'YData',[0 3 5 1])
```

Từ kết quả này, ta nhận thấy nếu dữ liệu trong 'XData' và 'YData' biến thiên theo thời gian, thì trên cửa sổ đồ họa (figure) ta sẽ thấy hình ảnh sinh động kiểu 'animation'.

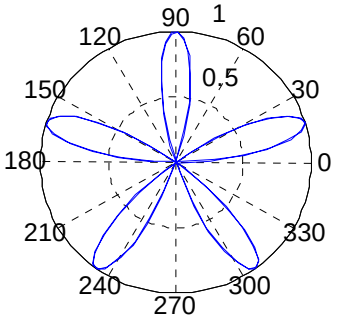


Hình 1.3 - Thay đổi dữ liệu trong cửa sổ đồ họa

③ Tương tự như hàm **plot**, sinh viên thử dùng các hàm **semilogx**, **semilogy** và **loglog** cho trường hợp trục tọa độ phi tuyến.

④ Ngoài các lệnh biểu diễn đường cong trong tọa độ Descartes, Matlab cũng hỗ trợ việc vẽ đồ thị hàm số trong hệ tọa độ cực bằng hàm **polar**.

```
>>theta=0:0.05:2*pi;
>>r=sin(5*theta);
>>polar(theta,r)
```



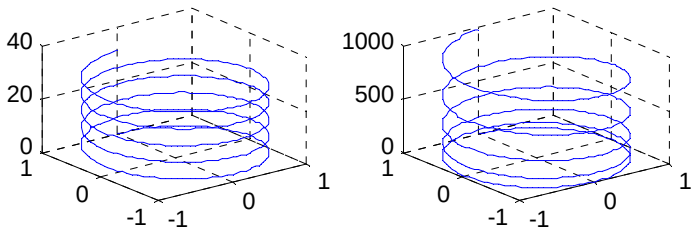
Hình 1.4 – Biểu diễn đồ thị hàm số trong hệ tọa độ cực

III.4.2. Đồ họa 3 D

Matlab cung cấp nhiều hàm vẽ đồ thị 3D, chẳng hạn: **plot3** - dùng để vẽ các đường trong không gian 3 chiều; **mesh** và **surf** - dùng để vẽ vật thể 3D (gõ **help mesh** và **help surf** để biết thêm các hàm 3D có liên quan).

① Vẽ đồ thị 3D bằng hàm **plot3**:

```
>>t=0:pi/50:10*pi;
>>x=sin(t);
>>y=cos(t);
>>z=t;
>>subplot(121), plot3(x,y,z)          %ve tren o thu nhat
>>grid on
>>subplot(122), plot3(x,y,t.^2)       %ve tren o thu hai
>> grid on
```

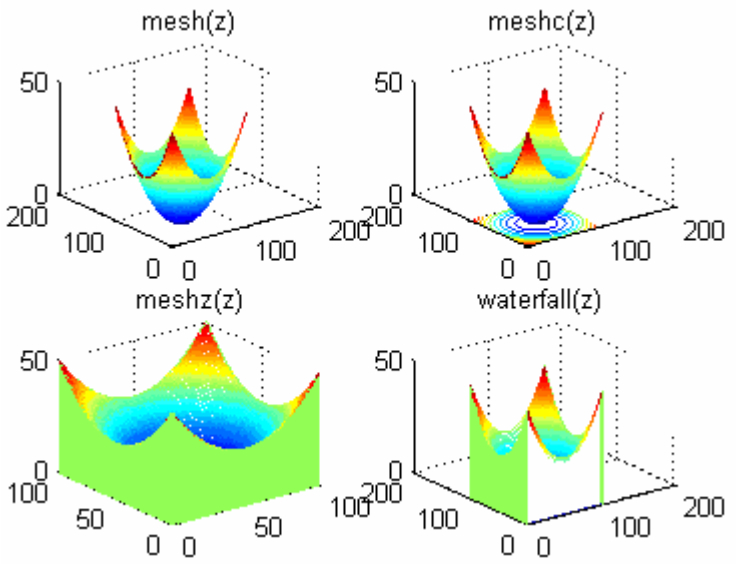


Hình 1.5 – Vẽ đồ thị 3D bằng hàm plot3

② Vẽ mặt paraboloid $z=x^2+y^2$ trong không gian 3 chiều:

```
>>close all
>>t=-5:0.1:5;
>> [x,y]=meshgrid(t);          %dinh luoi ve
>>z=x.^2+y.^2;
>> subplot(2,2,1), mesh(z)     %ve mat luoi 3D
```

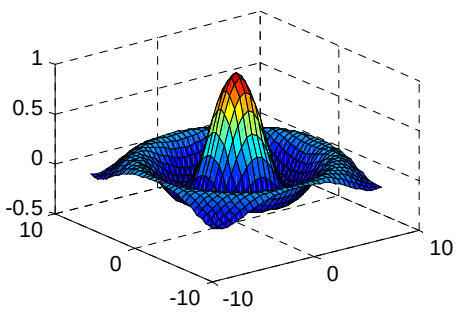
```
>> title('mesh(z)')
>> subplot(2,2,2), meshc(z)      %giong mesh nhung co them duong vien
>> title('meshc(z)')
>> subplot(2,2,3), meshz(z)      %co them luoi tren mat x,y
>> title('meshz(z)')
>> subplot(2,2,4), waterfall(z)  %chi ve luoi theo 1 huong
>> title('waterfall(z)')
```



Hình 1.6 - Vẽ mặt paraboloid

③ Vẽ mặt $z = \frac{\sin\left(\sqrt{x^2 + y^2}\right)}{\sqrt{x^2 + y^2}}$ trong không gian 3 chiều:

```
>>x=-8:0.5:8;
>>y=x;
>>[x,y]=meshgrid(x,y);
>>r=sqrt(x.^2+y.^2);
>>z=sin(r)./r;
>>surf(x,y,z)
```



Hình 1.7 – Một biểu diễn đồ thị 3D khác

④ Sinh viên thử vẽ mặt trụ $z = \sqrt{x^4 + y^2}$ bằng hàm **mesh** và hàm **surf**.

IV. Tự chọn

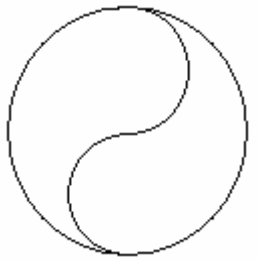
① Giải hệ phương trình sau:

$$\begin{aligned} 2x_1 + 4x_2 + 6x_3 - 2x_4 &= 0 \\ x_1 + 2x_2 + x_3 + 2x_4 &= 1 \\ 2x_2 + 4x_3 + 2x_4 &= 2 \\ 3x_1 - x_2 + 10x_4 &= 10. \end{aligned}$$

② Chứng tỏ rằng (A+B)C=AC+BC, với:

$$A=\begin{bmatrix} 10 & -2 \\ 20 & 4 \\ 3 & 6 \end{bmatrix}, B=\begin{bmatrix} 3 & 1 \\ -10 & 2 \\ 0 & 5 \end{bmatrix} \text{ và } C=\begin{bmatrix} -3 & 4 \\ 6 & 1 \end{bmatrix}.$$

③ Sinh viên thử vẽ hình sau (Hình 1.8):



(Hình 1.8)

④ Từ cửa sổ lệnh của Matlab, nhập: **demos**. Chọn **MATLAB → Graphics → 3D-plots**. Chạy chương trình demo này.

Bài 2

HÀM VÀ SCRIPT FILES

I. Mục tiêu

Bài thí nghiệm này giúp sinh viên tiếp cận với kỹ thuật lập trình trong Matlab thông qua việc sử dụng các hàm có sẵn, xây dựng các hàm mới dựa trên các cấu trúc **if - for - while** và tìm hiểu cách xây dựng script files, làm cơ sở cho việc lập trình mô phỏng các hệ thống Viễn thông và Tự động ở các bài sau.

II. Tham khảo

[1]. Nguyễn Hứa Duy Khang, Bài giảng môn CAD, Bộ môn Viễn thông & Tự động hóa, khoa Công nghệ thông tin, Đại học Cần thơ, 2001.

[2]. The Mathworks Inc., Matlab Notebook User's Guide, 2003.

[3]. Nguyễn Công Định, Phân tích và tổng hợp các hệ thống điều khiển bằng máy tính, NXB Khoa học và Kỹ thuật, 2002.

[4]. Nguyễn Hữu Tình - Lê Tấn Hùng - Phạm Thị Ngọc Yến - Nguyễn Thị Lan Hương, Cơ sở Matlab & ứng dụng, NXB KH và Kỹ thuật, 1999.

[5]. http://www.mines.utah.edu/gg_computer_seminar/matlab/

[6]. http://www.glue.umd.edu/~nsw/ench250/matlab.htm

III. Thực hành

Hàm và Script files trong Matlab đều được quản lý dưới dạng các tập tin có phần mở rộng **.m**, thường được soạn thảo bởi Matlab Editor. Khởi động Matlab Editor bằng một trong các cách sau:

1. Nhấp chuột vào biểu tượng trên menu bar của cửa sổ lệnh.
2. File → New → M-file
3. Nhấp vào biểu tượng Matlab Editor trên desktop (nếu có).

Chúng ta lần lượt khảo sát qui cách xây dựng các hàm và script file. Sinh viên đọc và thực hành lần lượt theo các phần sau.

III.1 Script files

Tập hợp các dòng lệnh của Matlab được sắp xếp theo một cấu trúc nào đó và lưu thành file có phần mở rộng *.m được gọi là script file (file kịch bản, file chương trình). Ta có thể chạy file này từ cửa sổ lệnh giống hệt như các lệnh của Matlab. Cấu trúc của một script file như sau:

```
%  ------------------------------------------------------------------------------
%  Phần viết sau dấu '%' ở đây dùng cho lệnh help
%  Thông thường phần này mô tả chức năng, cách sử dụng,
%  ví dụ minh họa hay những lưu ý đặc biệt mà tác giả mong muốn trợ
%  giúp cho người sử dụng.
%  ------------------------------------------------------------------------------
[global     tênbiến1, tênbiến2,… ]          % Khai báo biến toàn cục
                                            % (nếu có)
<các câu lệnh>                              % phần trình bày câu lệnh
```

① Hãy khởi động Matlab Editor và tạo một script file có tên **bai21.m**, với nội dung như sau:

```
% Doan script file nay hien thi loi chao trong 2s. Sau do
% hien thi logo cua matlab mot cach sinh dong roi thoat
close all
%  ---------------------- Tao mot cua so do hoa -------------------------------
figure('Color',[0 0 0],...
        'Name','Welcome to Matlab Experiments',...
        'NumberTitle','off',...
        'MenuBar','none');
%  ----------------------- Hien thi loi chao ----------------------------------
text(  'String','Welcome to MATLAB',...
        'Color',[.25 .25 .25],...
        'Position',[0.01 .501],...
        'Fontsize',32,...
        'FontAngle','italic');
text(   'String','Welcome to MATLAB',...
        'Color','w',...
        'Position',[0 .5],...
        'Fontsize',32,...
        'FontAngle','italic');
axis off;
pause(2);                                       % dung trong 2 giay
%  ---------------------- Hien thi logo cua Matlab ----------------------------
logospin
%  ---- Thoat - xoa cac bien trong workspace va dong cua so lai -----
clear
close
```

```
% ket thuc script file
```

Sau khi lưu file này, từ cửa sổ lệnh của Matlab, sinh viên hãy nhập:

```
>>help bai21
```

Để thi hành script file vừa soạn, hãy nhập:

```
>>bai21
```

② Soạn thảo script file có tên **bai22.m** để lập biểu đồ tổng số sinh viên Điện Tử - Tin học tốt nghiệp tại khoa Công nghệ Thông tin, Đại học Cần thơ từ năm 1996 đến 2001 với dữ liệu như sau:

Năm	Kỹ sư Điện tử	Kỹ sư Tin học
1996	38	48
1997	33	54
1998	36	120
1999	31	92
2000	60	110
2001	70	131

Nội dung của script file như sau:

```
%
% Lap bieu do tong so sinh vien Dien tu - Tin hoc tot nghiep tại
% Khoa Cong nghe Thong tin, Dai hoc Can tho, tu 1996-2001
%
svdt=[38 33 36 31 60 70];          % sinh vien Dien tu
svth=[48 54 120 92 110 131];       %sinh vien Tin hoc
nam=1996:2001;
subplot(211), bar(nam,svdt);
title('Sinh vien Dien tu tot nghiep tu 1996-2001');
subplot(212), bar(nam,svth);
title('Sinh vien Tin hoc tot nghiep tu 1996-2001');
Colormap(cool);
%
```

Thi hành file này từ cửa sổ lệnh. Sinh viên có thể dùng lệnh **help bar** để biết các thông số của hàm này và hiệu chỉnh (tùy thích) một số chức năng hiển thị của hàm **bar** trong file **bai22.m**.

③ Sinh viên hãy tạo một script file có tên **bai23.m** để vẽ các đồ thị trên hình 2.1.

III.2 Sử dụng các hàm xây dựng sẵn

Matlab hổ trợ một thư viện hàm rất phong phú, xây dựng trên các giải thuật nhanh và có độ chính xác cao. Ngoài các hàm cơ bản của Matlab, tập hợp các hàm dùng để giải quyết một ứng dụng chuyên biệt nào đó gọi

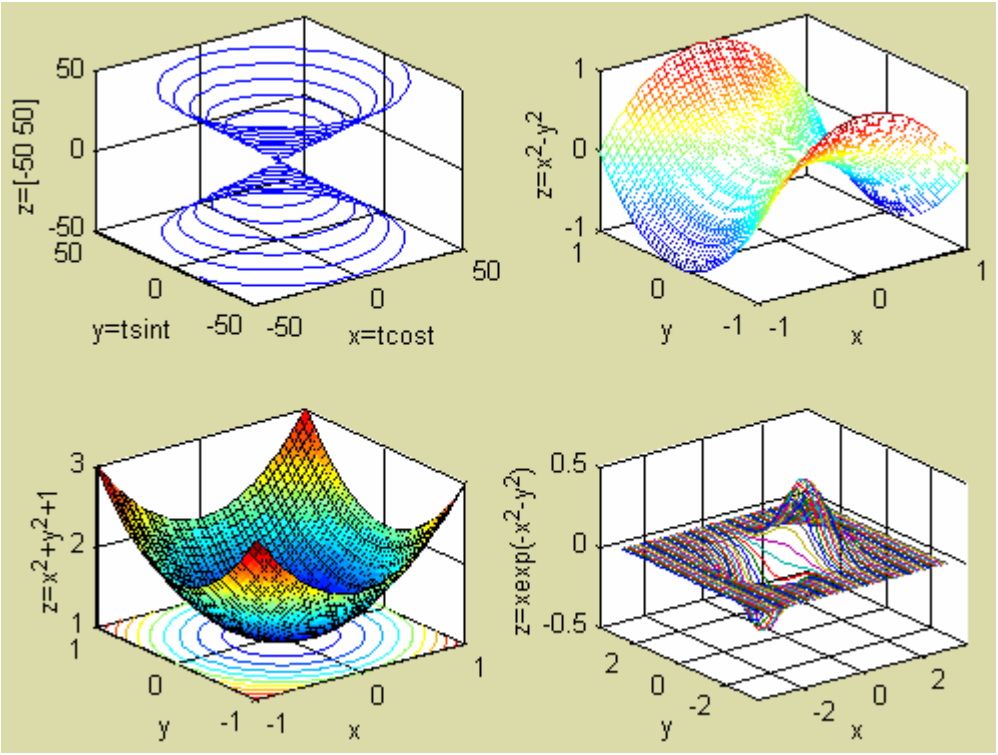
là Toolbox, ví dụ: Xử lý số tín hiệu (Digital Signal Processing), Điều khiển tự động (Control), Mạng Nơron nhân tạo (Neural networks), …

Sinh viên xem lại giáo trình để biết thêm về các hàm. Ngoài ra, có thể dùng lệnh **help** để biết chức năng của toolbox và hàm cũng như cách thức sử dụng chúng.

```
help <ten toolbox>          % chuc nang toolbox
>>help control              % liet ke ham cua  control toolbox
help <ten ham>              % chuc nang ham
>>help plot                 % chuc nang ham plot
```

Ta có thể tìm kiếm các hàm liên quan bằng cách cung cấp cho hàm **lookfor** của Matlab một từ khóa:

```
lookfor  <tu khoa tim kiem>
>>lookfor filter            % tìm các hàm liên quan đến mạch lọc
```



(Hình 2.1 – dùng cho câu III.1.3)

① Hàm [Y I]=max(X) cho biết phần tử lớn nhất của vectơ (mãng) X với chỉ số tương tứng I.

```
>>help max
>> x=[1 5 9 7 6 4];
>> [y,i]=max(x)
y =
    9
i =
    3
Nghĩa là phần tử thứ i=3 của vectơ x có giá trị lớn nhất, y=9.
```

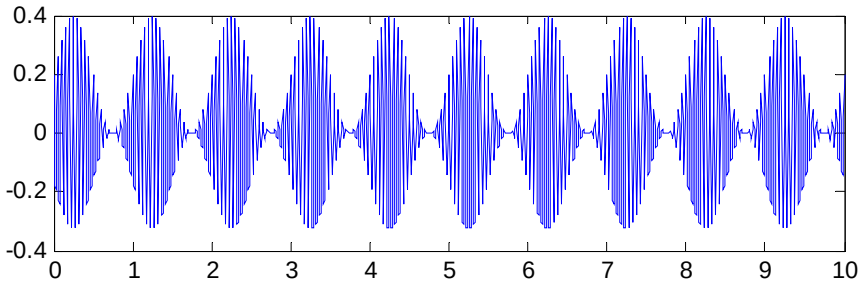
② Hàm Y=exp(X), tính e^X, kết quả trả về cho Y.

```
>>help exp
>>x=0:0.05:1;
>>y=exp(x);
>>stem(x,y)      % giong ham plot nhung ve cho tin hieu roi rac
```

③ Giả sử ta muốn điều chế biên độ sóng mang được truyền hai băng cạnh (Double sideband transmission carrier Amplitude Modulation) với:

- Tín hiệu điều chế X=0.2*sin(2*pi*t) với t:0:0.01:10
- Tần số sóng mang fc=620KHz

Sinh viên hãy tìm hàm thích hợp và vẽ tín hiệu thu được sau khi điều chế, có dạng như hình 2.2:



Hình 2.2 – Tín hiệu thu được sau khi điều chế biên độ

III.3 Xây dựng hàm

Việc xây dựng hàm cũng được thực hiện tương tự như script file. Tuy nhiên, đối với hàm ta cần quan tâm đến các tham số truyền cho hàm và các kết quả trả về sau khi thực hiện. Có 3 điểm cần lưu ý:

- Tên hàm phải được đặt trùng với tên file lưu trữ.
- Phải có từ khóa **function** ở dòng đầu tiên.
- Trong một hàm có thể xây dựng nhiều hàm con (điều này không có trong script file). Kết thúc hàm con phải có từ khóa **end** (điều này không cần trong hàm 'cha').

Qui cách xây dựng hàm được mô tả như sau:

```
function [out1,out2,...]=tenham(in1,in2,...)
% ---------------------------------------------------------------------------
%     Phần này sẽ hiển thị khi người sử dụng dùng lệnh help tenham
% ---------------------------------------------------------------------------
[global  <tênbiến1, tênbiến2, ...>]    %khai báo biến toàn cục (nếu có)

<Các câu lệnh thực hiện hàm>

out1=kết quả1                          %kết quả trả về của hàm
out2=kết quả2
...
```

```
% Các hàm con (nếu có)
[ function [subout1,subout2,...]=tenhamcon(subin1,subin2,...)
 <Các câu lệnh của hàm con>
 end ]              %từ khóa end khong can doi voi Matlab Version 6.x
```

① Xây dựng hàm **gptb2** để giải phương trình bậc hai $ax^2+bx+c=0$. Nội dung hàm như sau:

```
function [x1,x2]=gptb2(a,b,c)
% Giai phuong trinh bac hai ax^2+bx+c=0
%         [x1,x2]=gptb2(a,b,c)
% Trong do: x1,x2 nghiem thuc hoac phuc
%         a,b,c la 3 he so cua phuong trinh
%
% Vi du: [x1,x2]=gptb2(1,-3,2)
%
% Copyright 2003 Nguyen Chi Ngon TcAD - CIT - Cantho University
% Email: ncngon@cit.ctu.edu.vn

if nargin<3
       error('Vui long nhap du 3 he so cua phuong trinh')
elseif  a==0
       x1=-c/b;
       x2=[];
else
       D = b^ 2 - 4*a*c;
       x1 = (-b+sqrt(D))/(2*a);
       x2 = (-b-sqrt(D))/(2*a);
end
```

Sau khi lưu file này với tên **gptb2.m**, sinh viên thử kiểm chứng kết quả:

```
>>help gptb2
>>[x1,x2]=gptb2(1,6,-7)
>>[x1,x2]=gptb2(2,7,14)
>>[x1,x2]=gptb2(0,4,3)
>>[x1,x2]=gptb2(1,6)
```

✓ Cho biết ý nghĩa của từ khóa **nargin**?

✓ Sinh viên hãy viết lại hàm này để kết quả chỉ trả về nghiệm số thực.

② Xây dựng hàm **vdcongdb(a,m,method)** để vẽ một số đường cong đặc biệt trong hệ tọa độ cực, với a là bán kính và m là số đường cong vẽ trên cùng trục tọa độ. Trường hợp này hàm không trả về giá trị nên ta không cần biến ngõ ra.

Tuỳ theo giá trị của tham số **'method'** mà ta vẽ đồ thị tương ứng:

Nếu method = 'Becnulli': Vẽ đường Lemniscat Becnulli: $r = a\sqrt{|2\cos 2\theta|}$

Nếu method = 'Astroit': Vẽ đường Astroit: $r = a\sqrt{\left|1 - \frac{\sin 3\theta}{4}\right|}$

Nếu method = 'Xoanoc': Vẽ đường xoắn ốc: $r = \cos\theta + 1$

Nội dung hàm như sau:

```
function vdcongdb(a,m,method)
% Ve duong cong trong toa do cuc: vdcongdb(a,m,method)
% method =      'Becnulli' - Ve duong Lemniscat Becnulli:
%                       r=a*sqrt(abs(2*cos(2*theta)))
%                   'Astroit'  - Ve duong Astroit:
%                       r=a*sqrt(abs(1-sin(3*theta)/4))
%                   'Xoanoc'   - Ve duong xoan oc:
%                       r=a*cos(theta)+1
% Voi:  a-ban kinh;   m-so duong cong ve tren cung he truc
% Vi du: vdcongdb(0.5, 4, 'Becnulli')
%
% Copyright 2003 Nguyen Chi Ngon, TcAD - CIT - CTU
% Email: ncngon@cit.ctu.edu.vn

if nargin<3
        error('Vui long nhap du 3 thong so cua ham')
else
        theta=0:0.01:2*pi;    method=upper(method);
        switch method
          case 'BECNULLI'
                r=a*sqrt(abs(2*cos(2*theta)));
          case 'ASTROIT'
                r=a*sqrt(abs(1-sin(3*theta)/4));
          case 'XOANOC'
                r=a*cos(theta)+1;
          otherwise
                error('Chon: "Becnuli", "Aristod" hoac "Xoanoc"')
        end                             % end of switch

        % ve do thi
        close all;    figure('Color','w');
        for k=1:m
                hold on
                r1=r*k;
                mau=[rand(1,1) rand(1,1) rand(1,1)];
                h=polar(theta,r1);
```

```
            set(h,'color',mau,'LineWidth',2);
            axis equal;
        end                         % end of for
        hold off;
        axis off
end                                 % end of if
```

Sinh viên hãy kiểm chứng lại hoạt động của hàm, ví dụ:

```
>>help vdcongdb
>>vdcongdb(1,5,'Becnulli')
>>vdcongdb(1,5,' Astroit')
>>vdcongdb(1,5,'Xoanoc')
>> vdcongdb(1,5,'saikieu')
>> vdcongdb(5,'becnulli')
>> ....
```

③ Xây dựng hàm **dudoan()** để dự đoán kết quả sau mỗi lần tung một xúc xắc đồng nhất, 6 mặt. Nội dung hàm như sau:

```
function dudoan()
% Du doan ket qua sau moi lan tung ngau nhien mot xuc xac 6 mat
% Chuong trinh lap lai cho den khi nguoi su dung khong doan tiep
%
% Copyright 2003, Nguyen Chi Ngon TcAD - CIT, Cantho University
% Email: ncngon@ctu.edu.vn

tiep = 'y'; sai=0; dung=0;
disp('Chao mung ban den voi Casino nay!')
while(lower(tiep)=='y')
        doan=input('Moi ban du doan ket qua (1-6):');
        kqua=tungxx;
        if (doan ~= kqua)
                disp('Xin loi, ban da doan sai!')
                sai=sai+1;
        else
                disp('Xin chuc mung!')
                dung=dung+1;
        end
        tiep=input('Ban muon choi tiep("y"/"n"):');
end
disp(['Dung ' num2str(dung) ' trong tong so ' num2str(sai+dung) ' lan doan'])

% subfunction ----------
function mat = tungxx()
        mat=floor(6*rand(1,1))+1;
% end
```

Sinh viên thử sử dụng hàm này và cho biết sự khác nhau giữa script file và hàm không có tham số vào.

IV. Tự chọn

① Viết chương trình tìm nghiệm thực của phương trình bậc 2.

② Viết chương trình in tam giác Pascal n dòng trong màn hình đồ họa với n được nhập từ bàn phím.



Bài 3

SYMBOLIC VÀ SIMULINK

I. Mục tiêu

Hiểu một cách cơ bản về hai công cụ mạnh và hữu hiệu của Matlab, đó là **Symbolic** và **Simulink**, để từ đó sinh viên có thể tự mình phát huy các chức năng cao hơn của hai công cụ này trong tính toán và mô phỏng hệ thống.

II. Tham khảo

[1]. The Mathworks Inc., Matlab Notebook User's Guide, 2003.

[2]. Nguyễn Hoài Sơn - Đỗ Thanh Việt - Bùi Xuân Lâm, Ứng dụng MATLAB trong tính toán kỹ thuật, Tập 1, NXB ĐHQG Tp. HCM, 2000

[3]. Nguyễn Chí Ngôn, Bài thí nghiệm Kỹ thuật mô phỏng trong Điều khiển tự động, Bộ môn Viễn Thông & Tự động hóa, khoa Công nghệ thông tin, Đại học Cần thơ, 2002.

[4]. Nguyễn Công Định, Phân tích và tổng hợp các hệ thống điều khiển bằng máy tính, NXB Khoa học và Kỹ thuật, 2002.

[5]. http://www-h.eng.cam.ac.uk/help/tpl/programs/Matlab/Symbolic.html

III. Thực hành

Symbolic và **Simulink** đều chứa thư viện chức năng rất phong phú, bài thí nghiệm này chỉ có thể đề cập đến những gì gọi là cơ bản nhất. Từ đó, sinh viên có thể tự mình nghiên cứu và phát triển tiếp.

III.1 Symbolic

Phiên bản mới nhất của Symbolic toolbox được Mathworks giới thiệu trong Matlab 6.5 vào tháng 6-2003. Đó là một thư viện toán học kiểu ký tự, được phát triển từ Symbolic Maple của trường Đại học Waterloo, Canada. Để có cái nhìn tổng quát về các chức năng của Symbolic, sinh viên hãy gõ:

```
>>help symbolic
```

Một số hàm thông dụng của Symbolic:

Tên hàm	Chức năng	Tên hàm	Chức năng
diff	Đạo hàm	**fourier**	Biến đổi Fourier
int	Tích phân	**ifourier**	Biến đổi Fourier ngược
taylor	Khai triển Taylor	**laplace**	Biến đổi Laplace
det	Định thức của ma trận	**laplace**	Biến đổi Laplace ngược

numden	Tử và mẫu của phân số	**ezplot**	Vẽ hàm, ≡ **plot**
subs	Thay biến sym bằng trị số	**ezpolar**	Vẽ hàm, tọa độ cực ≡ **polar**
dsolve	Giải phương trình vi phân	**ezmesh**	Vẽ mặt lưới ≡ **mesh**
solve	Giải phương trình đại số	**ezsurf**	Vẽ mặt ≡ **surf**

Để biến đổi một số, một biến hay một đối tượng nào đó thành kiểu **Symbolic** ta có thể sử dụng một trong các cách sau:

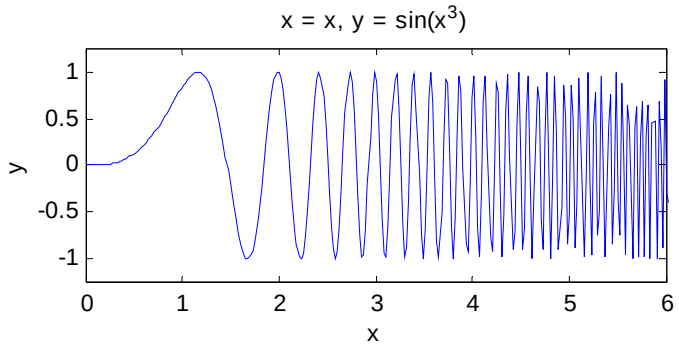
```
>>s=sym(A)
>>x=sym(x)
>>syms x y z          % khai báo kết hợp → x, y và z là biến symbolic
```

① Tính đạo hàm bằng hàm **diff** của symbolic: Nếu S là biểu thức symbolic thì:

diff(S)	đạo hàm của S theo biến tự do
diff(S,'v')	đạo hàm của S theo biến v
diff(S,'v',n)	đạo hàm cấp n của S theo v.

Ví dụ: Tính đạo hàm của $y = \sin x^3$.

```
>> syms x                    % khai bao x la bien kieu symbolic
>> y=sin(x^3);
>> z=diff(y)                 % dao ham cua y
   z =
        3*cos(x^3)*x^2       % sinh vien kiem tra ket qua
>>pretty(z)                  % hien thi dang quen thuoc
        3 cos(x^3) x^2
>>ezplot(x,y)                % ve y theo x
```



Hình 3.1 – Vẽ đồ thị hàm symbolic

② Tính vi phân bằng hàm **int -** Nếu S là biểu thức Symbolic thì:

int(S)	tích phân không xác định của S theo biến mặc nhiên (muốn biết biến mặc nhiên này ta dùng hàm **findsym**).
int(S,v)	tích phân không xác định của S theo v.
int(S,a,b)	tích phân xác định của S trên cận [a,b]
int(S,v,a,b)	tích phân xác định của S theo v trên cận [a,b]

Ví dụ: Tính $\int_0^1 \frac{2x^2(19+12x^2)}{7(x^2+1)}dx$

```
>>syms x
>>S=2*x^2*(19+12*x^2)/(7*(x^2+1))
>>y=int(S,x,0,1)                % tích phân S theo x trên cận [0,1]
>>subs(y)                       % đổi sang kiểu số
```

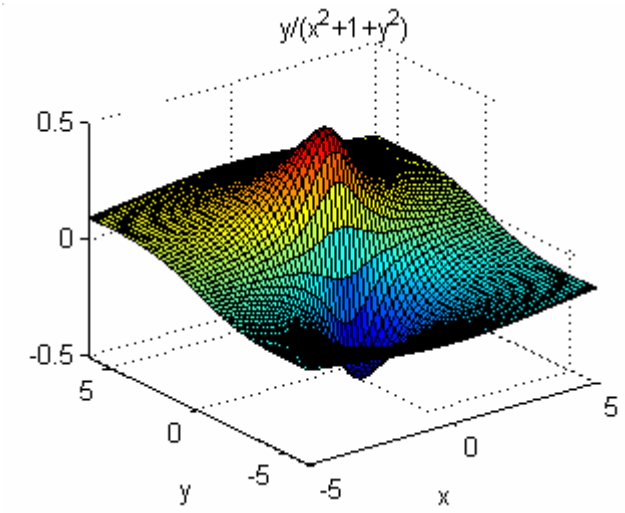
③ Giải hệ phương trình bằng hàm **solve:**

```
>>help solve
>>syms x  y
>>[x,y]= solve('x^2*sin(x^2)-3*y=7','x+y=1')
```

Sau khi thu được nghiệm x và y, sinh viên hãy thay vào 2 phương trình trên và nhận xét kết quả.

④ Vẽ mặt 3D bằng hàm **ezsurf**: Ví dụ vẽ mặt $S = f(x,y) = \frac{y}{1+x^2+y^2}$ trên miền xác định: -5<x<5; -2π<y<2π.

```
>>syms x y
>>S=y/(1+x^2+y^2)
>>ezsurf(S, [-5 5 -2*pi 2*pi])
```



Hình 3.2 – Vẽ đồ thị 3D cho hàm số symbolic

III.2 Simulink

Simulink 5.0 (Simulation and Link - R13) được MatWorks giới thiệu vào tháng 6 năm 2003. Nó cho phép phân tích, mô hình hóa và mô phỏng các hệ thống động tuyến tính và phi tuyến, liên tục và rời rạc một cách trực quan trong môi trường giao tiếp đồ họa, bằng các thao tác chuột đơn giản. Có thể nói, không tận dụng được Simulink là một thiệt thòi lớn cho người làm công tác mô phỏng!

Khởi động Simulink bằng một trong các cách sau:

nhập: >>simulink

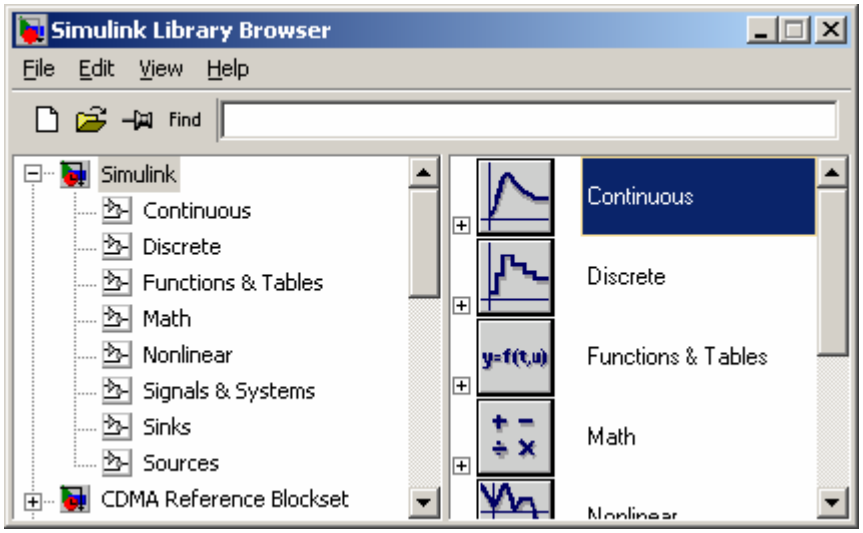
hoặc nhấp chuột vào  trên menubar của Matlab

Thư viện simulink hiện ra như hình 3.3:

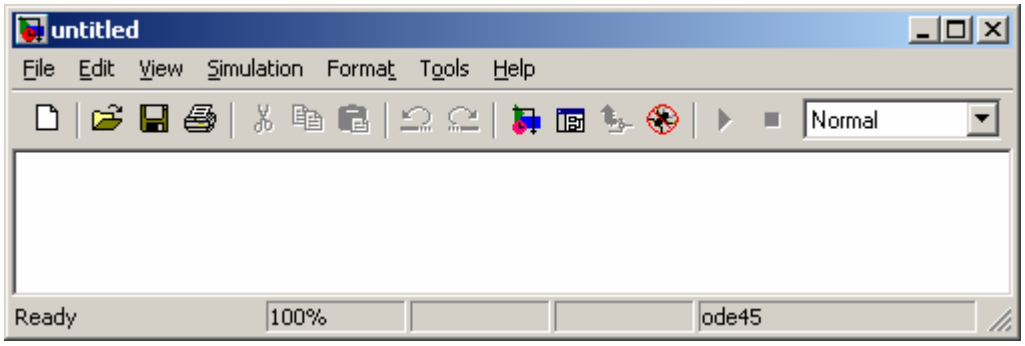
Trước tiên, sinh viên hãy nhấp chuột vào các thanh cuộn của thư viện để có cái nhìn thân thiện về simulink.

Từ đây, để có thể tạo mô hình bằng simulink, hãy:

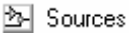
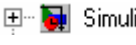
- nhấp chuột vào biểu tượng  của thư viện simulink
- chọn: File - New - Model trong Menu của thư viện Simulink
- chọn: File - New - Model trong cửa sổ lệnh của Matlab

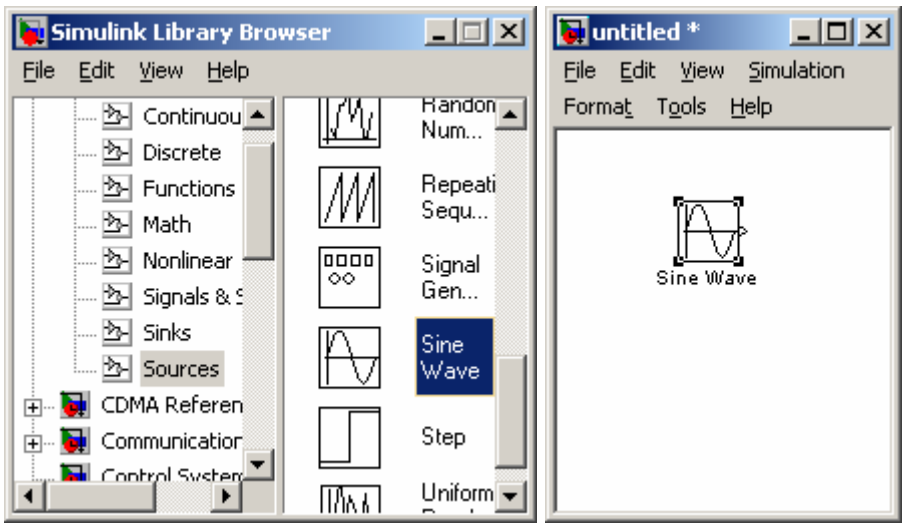


Hình 3.3 – Cửa sổ chính của thư viện Simulink



Hình 3.4 – Môi trường soạn thảo của Simulink

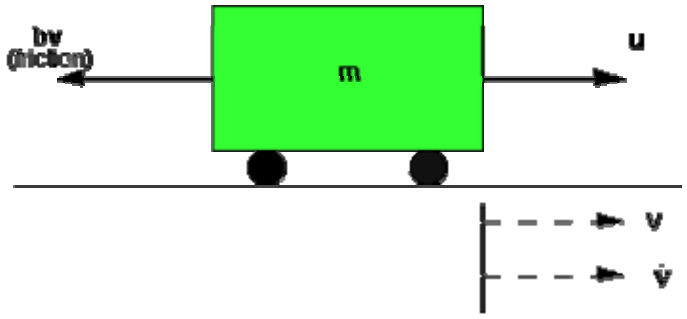
Cửa sổ này (hình 3.4) cho phép ta 'nhấp - kéo - thả' vào từng khối chức năng trong thư viện simulink. Ví dụ, đặt vào đây khối 'Sine Wave' trong thư viện  của  (hình 3.5):



Hình 3.5 – Lấy một khối từ thư viện

Sau khi đặt tất cả các khối cần thiết của mô hình, ta nối chúng lại bằng cách 'nhấp - giữ' và kéo một đường từ ngõ ra của khối này đến ngõ vào của khối khác rồi thả phím trái chuột, một kết nối sẽ được thiết lập.

① Xây dựng mô hình hệ thống xe tải:



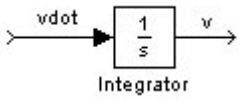
Hình 3.6 – Mô hình xe tải

cho bởi phương trình: $m\frac{dv}{dt} = u - bv$ hay $\frac{dv}{dt} = \frac{1}{m}(u - bv)$.

Trong đó m là khối lượng xe, u là lực tác động của động cơ (ngõ vào của mô hình), b là hệ số ma sát và v là vận tốc đạt được (ngõ ra của mô hình).

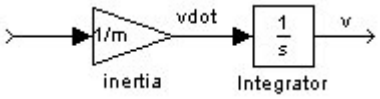
Xuất phát điểm của việc xây dựng các mô hình hệ thống từ các phương trình vi phân tương ứng là các khối tích phân (Integrator). Nếu trong phương trình mô tả hệ thống có **vi phân bậc n** thì ta sẽ đặt vào mô hình **n khối tích phân**, do quan hệ $\int \frac{dv}{dt} = v$.

- Mở một cửa sổ mô hình mới.
- Đặt vào mô hình khối 'Integrator' từ thư viện 'Continuous' và kẻ các đường thẳng nối đến ngõ vào và ngõ ra của khối này.
- Đặt nhãn 'vdot' (dv/dt) cho cho đường nối đến ngõ vào và 'v' cho đường nối đến ngõ ra bằng cách nhấp đúp chuột ngay phía trên các đường này.



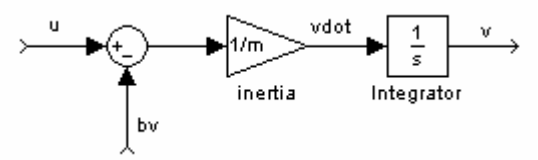
Từ phương trình hệ thống ta thấy dv/dt (vdot) bằng tích của thành phần (1/m) và thành phần tổng (u-bv), nên ta thêm khối 1/m ngay trước khối tích phân:

- Đặt vào khối 'Gain' trong thư viện Math.
- Nhấp đúp chuột vào khối này để thay đổi độ lợi thành 1/m.
- Đặt nhãn 'inertia' cho khối này để tượng trưng cho quán tính của xe (nhấp đúp vào nhãn 'Gain' bên dưới khối).



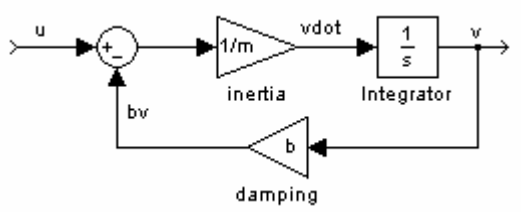
Bây giờ ta đặt khối tổng với 2 ngõ vào '+-', ngõ vào '+' sẽ được nối với u, ngõ vào '-' sẽ được nối với thành phần bv để được (u-bv)

- Đặt vào khối 'Sum' trong thư viện Math
- Nhấp đúp vào khối này để đổi ngõ vào từ '++' sang '+-'



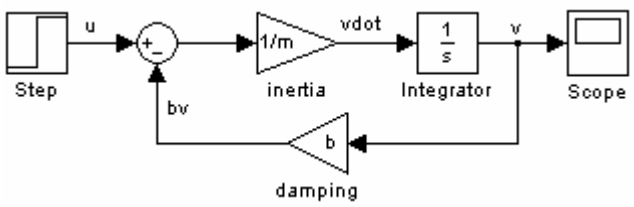
Để được thành phần bv ta chỉ cần đặt thêm khối 'Gain' với độ lợi b:

- Đặt khối 'Gain' có độ lợi b
- Đặt nhãn là 'damping' tượng trưng cho thành phần lực cản của xe.



Đến đây việc xây dựng mô hình xe tải với ngõ vào u và ngõ ra v coi như hoàn thành. Tuy nhiên, để mô phỏng mô hình này, ta cần đặt thêm khối 'Step' vào u và hiển thị v trên khối 'Scope'

- Đặt khối 'Step' trong thư viện Sources biên độ u ngay ngõ vào.
- Đặt khối 'Scope' trong thư viện Sinks ngay ngõ ra v.



Nhớ rằng m, b và u là các biến cần được gán trị trước khi mô phỏng.

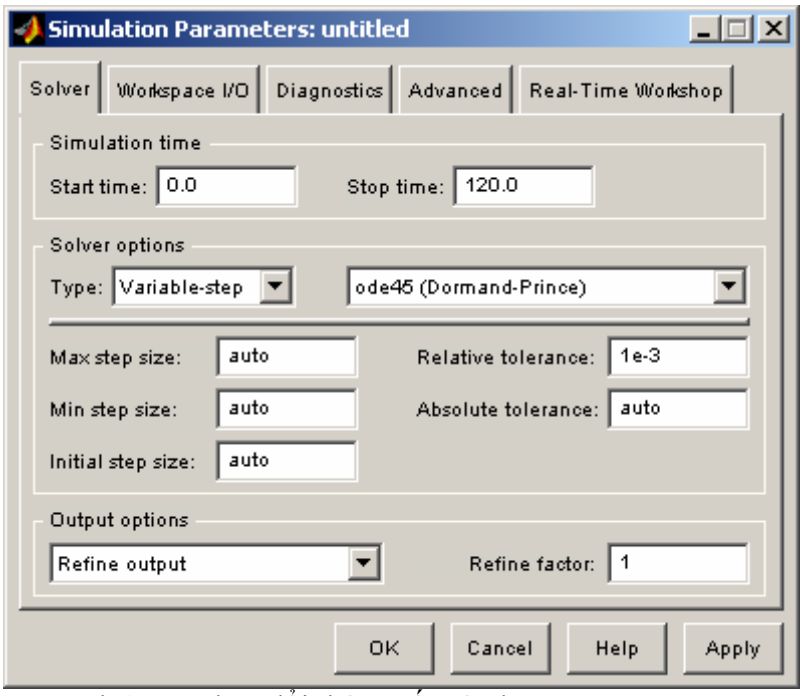
```
>>m=1000
>>b=50
>>u=500
```

Thời gian mô phỏng hệ thống tùy thuộc vào thông số **'Stop time'** trong menu **'Simulation→Simulation paramrters'**, giả sử đặt 120 (Hình 3.7).

Để chạy mô phỏng, ta có thể thực hiện bằng 1 trong các cách:

- nhấp chuột vào biểu tượng ▶ trên menubar của mô hình
- chọn: Simulation → Start
- Ctrl-T

Nhấp đúp vào **'Scope'** để xem kết quả mô phỏng.



Hình 3.7 – Thay đổi thông số mô phỏng

② Xây dựng mô hình hệ thống điều khiển vị trí motor DC cho bởi phương trình vi phân sau:

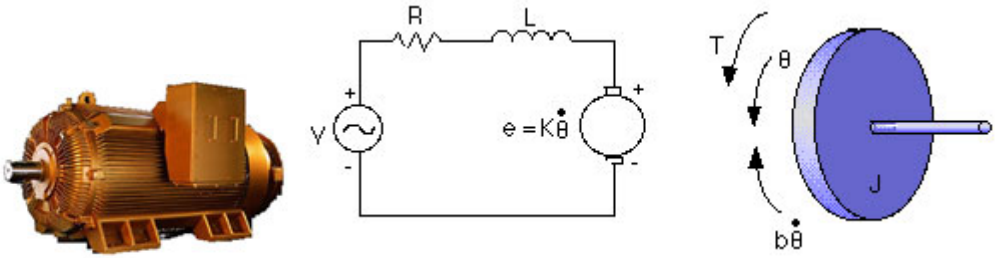
$$\frac{d^2\theta}{dt^2} = \frac{1}{J}\left(K_t i - b\frac{d\theta}{dt}\right)$$

$$\frac{di}{dt} = \frac{1}{L}\left(-Ri + V - K_e\frac{d\theta}{dt}\right)$$

Trong đó:

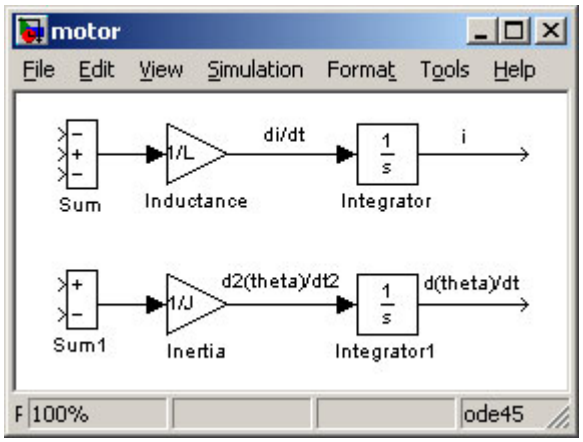
$J = 0.01\ Kgm^2/s^2$	là moment quán tín của rotor
$b = 0.1\ Mms$	là hệ số ma sát của các bộ phận cơ khí
$K = K_e = K_t = 0.01\ Nm/A$	là hằng số sức điện động
$R = 10\ ohm$	là điện trở dây quấn
$L = 0.5\ H$	là hệ số tự cảm

V	là điện áp đặt lên cuộn dây của motor
θ	là vị trí trục quay (ngõ ra của mô hình)
i	là dòng điện chạy trong cuộn dây của motor.



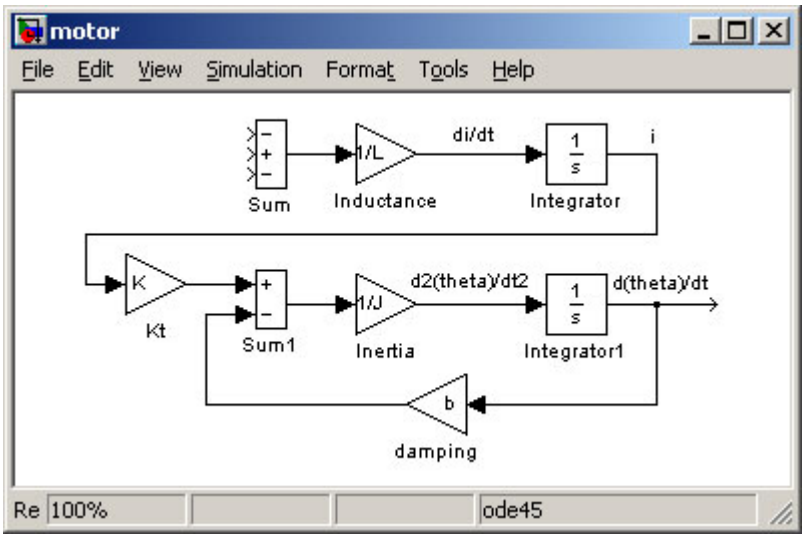
Hình 3.8 – Mô hình toán hệ điều khiển vị trí motor DC

Quan sát từng phương trình mô tả hệ thống ta thấy cấu trúc của chúng cũng tương tự như phương trình của câu ①. Sinh viên lần lượt thực hiện trên từng phương trình để được (hình 3.9):



(Hình 3.9)

Kết hợp 2 phương trình:



(Hình 3.10)

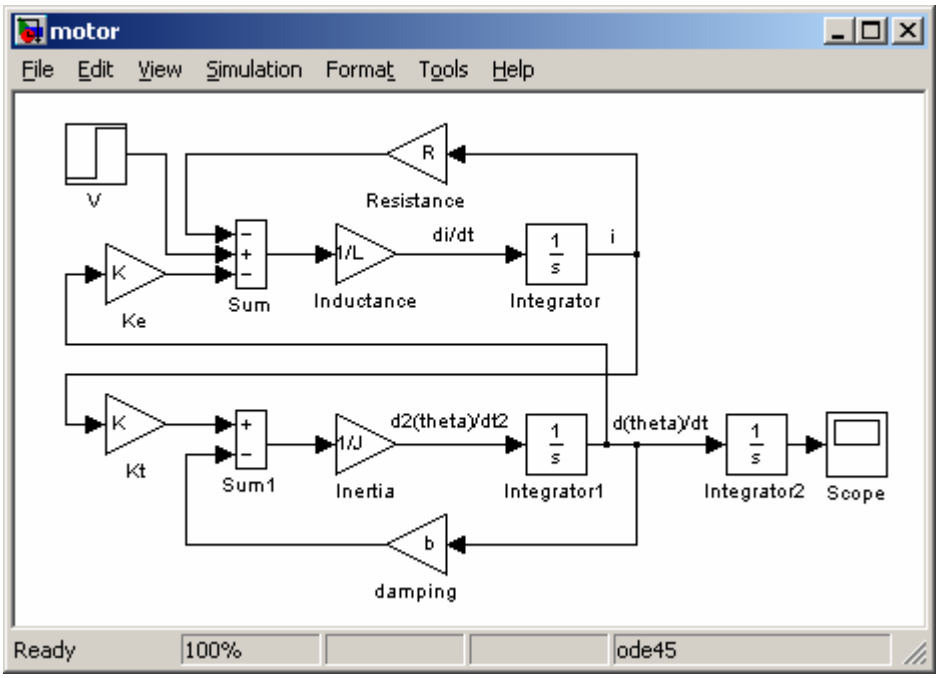
Đặt vào mô hình khối 'Step' để làm tín hiệu tham khảo, khối 'Scope' để quan sát đáp ứng. Sinh viên hãy gán trị cho tất cả các thông số của mô hình, thực hiện mô phỏng và quan sát đáp ứng (Hình 3.11).

Hãy thay đổi ngõ vào mô hình bằng khối tạo **xung vuông**. Mô phỏng, quan sát kết quả và nhận xét.

IV. Tự chọn

① Sinh viên hãy tính đạo hàm cấp 2 của hàm $y = xe^{(1-x^2)}$ bằng tay và kiểm chứng kết quả bằng **symbolic**.

② Tính tích phân sau và kiểm chứng kết quả bằng symbolic: $\int_0^{\infty} e^{-x^2}$



Hình 3.11 - Mô hình Simulink hoàn chỉnh của hệ điều khiển vị trí motor DC

③ Xây dựng mô hình hệ thống xe lửa cho bởi phương trình:

$$M_1 \frac{d^2x_1}{dt^2} = F - k(x_1 - x_2) - \mu M_1 g \frac{dx_1}{dt}$$

$$M_2 \frac{d^2x_2}{dt^2} = k(x_1 - x_2) - \mu M_2 g \frac{dx_2}{dt}$$

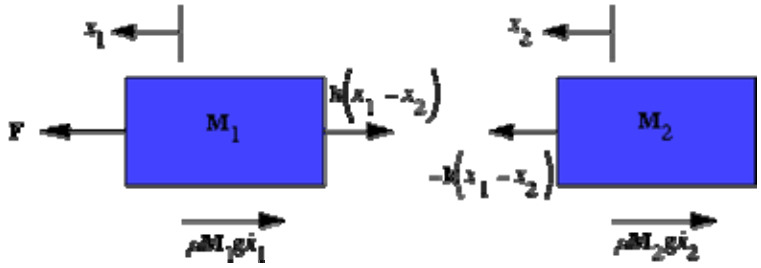


(Hình 3.12 Photo courtesy: Dr. Howard Blackburn)

Trong đó các thông số tượng trưng như sau:

M_1=1 kg	là khối lượng toa kéo;
M_2=0.5 kg	là khối lượng toa khách;

k=1 N/sec	là độ cứng lò xo kết nối giữa 2 toa;
F=1 N	là lực tác động của đầu máy (ngõ vào mô hình);
μ=0.002 sec/m	là hệ số ma sát lăn;
g = 9.8 m/s^2	là gia tốc trọng trường
x_1, x_2	vị trí 2 toa (ngõ ra).



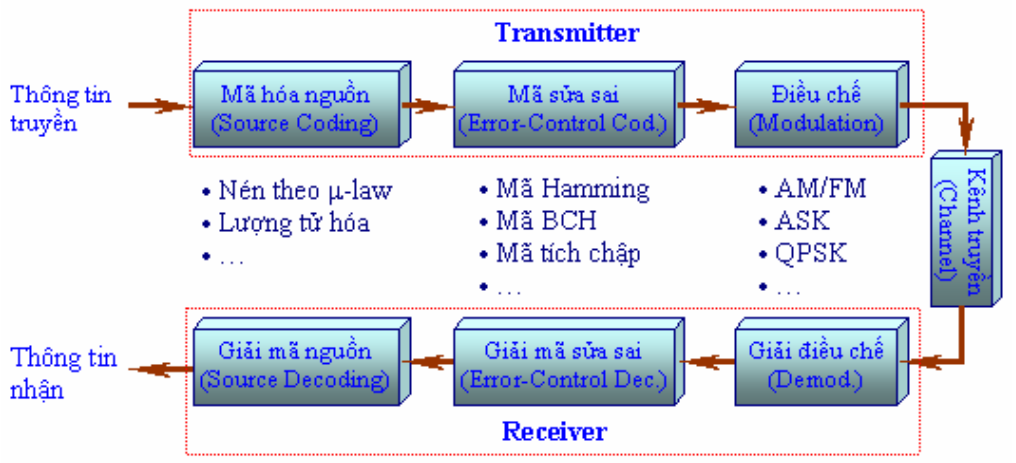
Hình 3.13 – Mô hình toán của hệ thống xe lửa

BÀI 4

MÔ HÌNH HỆ THỐNG VIỄN THÔNG

I. Mục tiêu

Bài thí nghiệm này tìm hiểu phương pháp mô hình hóa một hệ thống thông tin đơn giản như hình 4.1:



Hình 4.1 - Một mô hình truyền tin tiêu biểu

Chúng ta sẽ xử lý tín hiệu nguồn (mã hóa, kiểm soát lỗi, điều chế) và truyền qua một kênh truyền có nhiễu sau đó khôi phục tín hiệu nhận được (giải điều chế, giải mã) và xác định xác suất lỗi tín hiệu.

II. Tham khảo

[1]. The Mathworks Inc., Matlab User's Guide – Communications Toolbox, 2003.

[2]. Nguyễn Hữu Tình - Lê Tấn Hùng - Phạm Thị Ngọc Yến - Nguyễn Thị Lan Hương, Cơ sở Matlab & ứng dụng, NXB KH và Kỹ thuật, 1999.

[3]. http://murray.newcastle.edu.au/uers/staff/eemf/ELEC352/notes.htm

III. Thực hành

Để có thể thực tập tốt bài thí nghiệm, sinh viên cần có kiến thức về **Xử lý số tín hiệu** (Digital Signal Processing), **Truyền dữ liệu** (Data transmittion) và **Cơ sở Viễn thông**. Do đó, bài thí nghiệm này không bắt buộc đối với sinh viên chuyên ngành Tin học (nếu có) và sinh viên Điện tử hướng ĐKTĐ. Trong trường hợp đó, sinh viên có thể chọn **bài 5** hoặc **bài 7** để thực tập.

Xây dựng mô hình hệ thống thông tin nói chung là một việc rất lớn, với nhiều vấn đề cần giải quyết. Do đó trong phạm vi bài thí nghiệm này chúng ta chỉ có thể khảo sát một số khâu trong tổng thể của một hệ thống truyền thông.

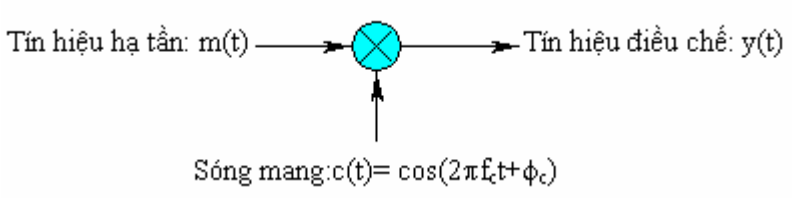
III.1 Hệ thống thông tin liên tục (Analog Communications)

Trong phần này chúng ta sẽ tìm hiểu các phương pháp điều chế và giải điều chế tín hiệu liên tục (Analog Modulation and Demodulation).

① **Điều chế biên độ sóng mang bị nén hai băng cạnh** (Double-Sideband Suppressed Carrier Amplitude Modulation – DSB-SC): Phương pháp điều chế này dựa theo công thức sau:

$$y(t) = m(t)\cos(2\pi f_c t+\phi_c)$$

Trong đó: m(t) là tín hiệu hạ tần (thông tin) dùng để điều chế sóng mang tần số cao $c(t) = \cos(2\pi f_c t+\phi_c)$ để thu được tín hiệu điều chế y(t).



Hình 4.2 – Nguyên tắc điều chế biên độ

Ta có thể thực hiện phương pháp điều chế này trong Matlab như sau (sinh viên nên ghi trong **file .m** để thuận tiện cho việc sử dụng lại ở các câu sau):

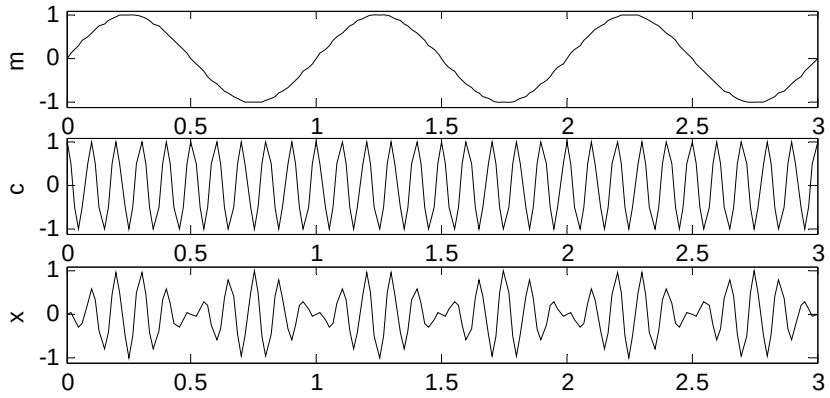
```
>>Fc=10;                                  % tần số sóng mang [Hz]
>>Fs=60;                                  % tần số lấy mẫu để mô phỏng
>>Fm=1;                                   % tần số tín hiệu hạ tần
>>t=[0:1/Fs:3];                           % time window
>>c=cos(2*pi*Fc*t);                       % tín hiệu sóng mang
>>m=sin(2*pi*Fm*t);                       % tin hiệu hạ tần 1Hz
>>subplot(311), plot(t,m)                 % vẽ tín hiệu hạ tần
>>subplot(312), plot(t,c)                 % vẽ sóng mang
>>y=m.*c;                                 % điều chế
>>subplot(312), plot(t,y)                 %  vẽ tín hiệu điều chế
>>z=abs(fft(y));                          % phổ tín hiệu điều chế
>>z=z(1:length(z)/2+1);                   % lấy 1/2
>>frq=[0:length(z)-1]*Fs/length(z)/2;     % frequency window
>>figure                                  % new figure
>>plot(frq,z);                            % vẽ phổ
```

✓ Sinh viên hãy quan sát phổ tín hiệu điều chế, và hãy so sánh với tần số sóng mang.

✓ Thử dùng hàm **amod** của Communications toolbox thực hiện công việc trên.

```
>>help amod
```

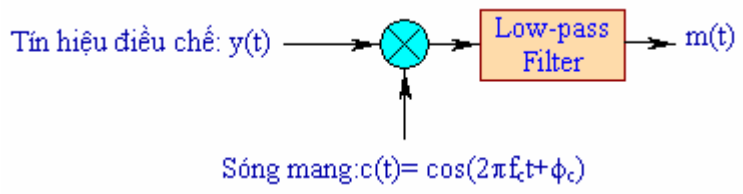
```
>>Y=amod(m, Fc, Fs, 'amdsb-sc');          % điều chế
>>subplot(313),plot(t,Y)
```



Hình 4.3 – Tín hiệu điều chế biên độ

✓ Tương tự như trên, sinh viên hãy thực hiện **điều chế biên độ sóng mang bị nén đơn băng cạnh** (Single-Sideband Suppressed Carrier Amplitude Modu-lation – SSB-SC). So sánh với phương pháp điều chế biên độ DSB-SC.

② **Giải điều chế biên độ:** Có nhiều phương pháp giải điều chế biên độ. Ở đây ta xét phương pháp giải điều chế đồng bộ (Synchronous Demodulation), áp dụng cho phương pháp điều chế biên độ với sóng mang bị nén SC (Suppressed Carrier) và phương pháp điều chế biên độ với sóng mang được truyền TC (Transmission Carrier), theo mô hình sau:



Hình 4.4 – Nguyên tắc giải điều chế biên độ

Thực hiện giải điều chế như sau (phải đảm bảo rằng tín hiệu điều chế y còn trong WorkSpace của Matlab):

```
>>y_mix=y.*c;                    % trộn tín hiệu điều chế với sóng mang
>>[num,den]=butter(3, (Fc/2)/(Fs/2));   % low-pass filter tần số cắt Fc/2
>>m_r=filter(num,den,y_mix);      % cho tín hiệu đã trộn qua bộ lọc
>>close all                       %
>>plot(t,m)                       % vẽ tín hiệu hạ tần ban đầu
>>hold on                         %
>>plot(t,m_r,'r')                 % vẽ tín hiệu hạ tần vừa hồi phục
```

✓ Giải thích vì sao ta dùng bộ lọc thông thấp với tần số cắt Fc/2.

✓ Sinh viên hãy quan sát và so sánh sự khác biệt giữa tín hiệu giải điều chế và tín hiệu hạ tần ban đầu. Giải thích.

✓ Thử dùng hàm **ademod** của **Communications toolbox** để giải điều chế:

```
>>help ademod
>>M_R=ademod(y,Fc,Fs,'amdsb-sc');  % giải điều chế
>>plot(t,M_R,'g')
```

III.2 Hệ thống thông tin rời rạc (Digital Communications)

Trong phần này, chúng ta sẽ thực hiện mô hình một hệ thống truyền thông rời rạc như sơ đồ sau:



Hình 4.5 – Mô hình một hệ thống truyền tin rời rạc

Sinh viên cần đọc phần help của từng lệnh hiểu và biết cách sử dụng.

① **Tạo tín hiệu nguồn:**

```
>>clear
>>close all
>>K=11;                          % chiều dài từ tin (message length)
>>msg=randint(K*2,1);            % tạo tín hiệu nguồn (digital message)
>>subplot(411),stem(msg,'.')     % vẽ tín hiệu nguồn
>>ylabel('digital message')
```

② **Mã hóa BCH:**

```
>>N=15;                          %chiều dài từ mã (codeword length)
>>code = encode(msg,N,K,'bch');      %mã hóa BCH
>>subplot(412), stem(code,'.')
>>ylabel('BCH')
```

③ **Điều chế QASK:** Sử dụng bộ điều chế số M-QASK với M=16 (sinh viên cần xem lại lý thuyết về bộ điều chế này)

```
>>M=16;                                  % multiple number
>>Fd=1;                                  % tốc độ nguồn tin
>>Fc=10;                                        % tần số sóng mang
>>Fs=30;                                        % tần số lấy mẫu
>>modu = dmod(code,Fc,Fd,Fs,'qask',M);      % điều chế M-QASK
>>subplot(413),plot(modu)                % vẽ tín hiệu điều chế
>>ylabel('QASK')
```

④ **Kênh truyền có nhiễu:** Tín hiệu sau khi điều chế được truyền trên kênh truyền có nhiễu trắng Gaussian công suất 0.1W.

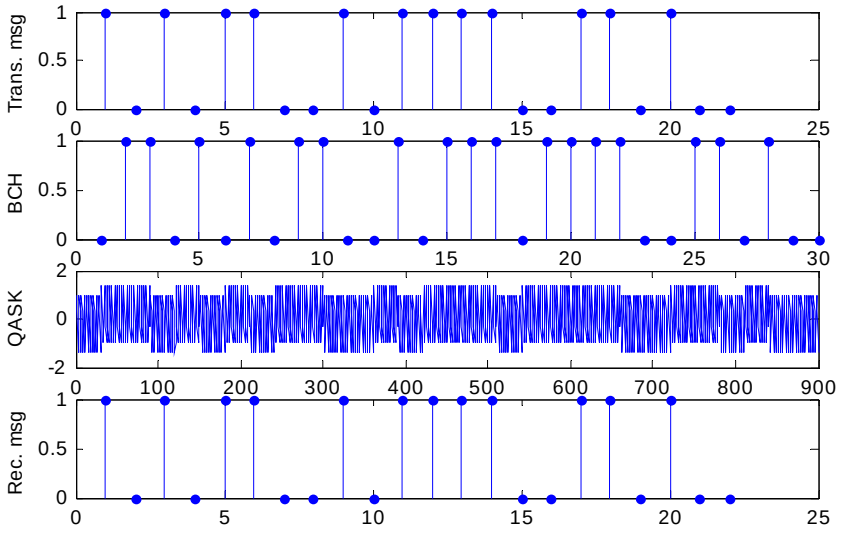
```
>>std_value=0.1;          % cong suat nhieu
>>modu_noise = modu+randn(length(modu),1)*std_value; % add noise
```

⑤ **Giải điều chế QASK:**

```
>>demo = ddemod(modu_noise,Fc,Fd,Fs,'qask', M);  % giải điều chế
```

⑥ **Giải mã BCH:**

```
>>msg_r = decode(demo,N,K,'bch');          % tín hiệu thu
>>subplot(414), stem(msg_r,'.')            % vẽ tín hiệu số thu được
>>ylabel('received message')
```



Hình 4.6 – So sánh tín hiệu truyền/nhận

⑦ **Xác định lỗi:**

```
>>rate=biterr(msg,msg_r,M)
```

⑧ Sinh viên hãy dùng hàm **channel** của Communications toolbox để mô hình hóa kênh truyền ở bước ④. Lặp lại các quá trình giải điều chế và giải mã, so sánh kết quả (lưu ý: băng thông của kênh truyền phải đủ lớn).

IV. Tự chọn

① Thực hiện điều chế và giải điều chế tần số **FM** với Fm, Fc, Fs ở câu III.1.①.

② Thực hiện mô hình Digital Communications như hình sau:



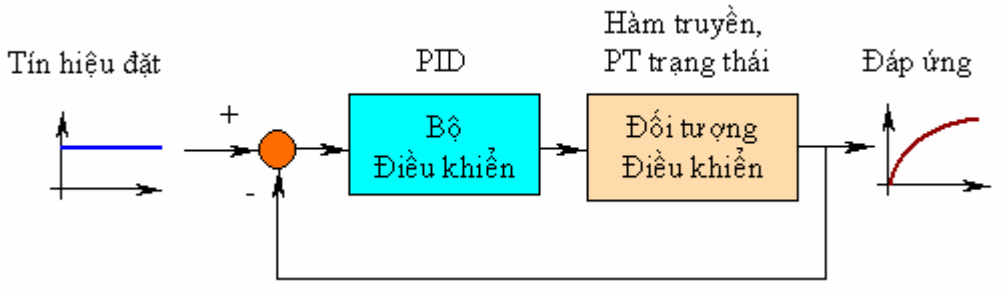
Bài 5

MÔ HÌNH HỆ THỐNG ĐIỀU KHIỂN TỰ ĐỘNG

I. Mục tiêu

Trong bài thí nghiệm này chúng ta sẽ tìm hiểu phương pháp mô hình hóa một hệ điều khiển tự động, bao gồm:

- Hàm truyền và phương trình trạng thái của hệ thống
- Đáp ứng vòng hở và đáp ứng vòng kín của hệ thống
- Xây dựng bộ điều khiển PID
- Chỉnh định thông số của bộ điều khiển và khảo sát đáp ứng của hệ thống.



Hình 5.1 – Một mô hình hệ thống điều khiển tiêu biểu

II. Tham khảo

[1]. The Mathworks Inc., Matlab Notebook User's Guide – Control toolbox, 2003.

[2]. Phạm Văn Tấn, Bài giảng môn Cơ sở Tự động học, Bộ môn Viễn Thông và Tự động hóa, khoa Công nghệ Thông tin, Đại học Cần Thơ, 2001.

[3]. Nguyễn Công Định, Phân tích và Tổng hợp các hệ thống Điều khiển bằng máy tính, NXB Khoa học và Kỹ thuật, 2002.

[4]. http://www.engin.umich.edu/group/ctm

[5]. http://www.shu.ac.uk/schools/eng/teaching/rw/pidtutorial.htm

III. Thực hành

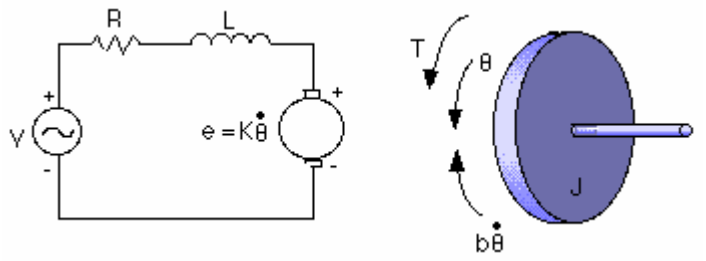
Để có thể thực hiện tốt bài thí nghiệm, sinh viên cần nắm vững các kiến thức cơ bản về Điều khiển tự động (Cơ sở tự động học). Do đó, bài này không bắt buộc đối với các sinh viên Tin học (nếu có) và các sinh viên Điện tử theo hướng Viễn thông. Trong trường hợp đó, có thể sinh viên đã thực tập **bài 4** hoặc sinh viên có thể chuyển sang **bài 7**.

III.1 Hàm truyền và phương trình trạng thái của hệ thống

Trong điều khiển tự động, người ta thường biểu diễn một hệ thống vật lý bằng hàm truyền (transfer function) hay phương trình trạng thái (state-space equation) của nó (đối với các hệ phi tuyến, để đạt được điều này, người ta phải dùng phương pháp tuyến tính hóa từng đoạn).

Giả sử có hệ thống điều khiển tốc độ motor DC như hình vẽ 5.2 [4]. Trong đó:

J = 0.01 kgm^2/s^2	là moment quán tính của rotor
b = 0.1 Nms	hệ số ma sát
K=K$_e$=K$_t$=0.01 Nm/Amp	các hằng số sức điện động
R = 1 ohm	điện trở
L = 0.5 H	điện cảm
I:	dòng điện chạy trong cuộn dây của motor
V:	điện áp trên hai đầu cuộn dây motor - ngõ vào
θ:	vị trí trục - ngõ ra



Hình 5.2 – Mô hình toán một hệ điều khiển tốc độ motor DC

Phương trình vi phân mô tả hệ thống như sau:

$$J\frac{d^2\theta}{dt^2} + b\frac{d\theta}{dt} = Ki$$

$$L\frac{di}{dt} + Ri = V - K\frac{d\theta}{dt}$$

① **Hàm truyền:** Biến đổi Laplace 2 vế của phương trình trên ta được:

$$s(Js+b)\boldsymbol{\Theta(s)} = K\mathbf{I(s)}$$

$$(Ls+R)\mathbf{I(s)} = V - Ks\boldsymbol{\Theta(s)}$$

Suy ra: $\left[(Ls+R)(Js+b)+K^2\right]s\boldsymbol{\Theta} = K\mathbf{V}$ hay $\dfrac{\dot{\theta}}{V} = \dfrac{K}{(Ls+R)(Js+b)+K^2}$

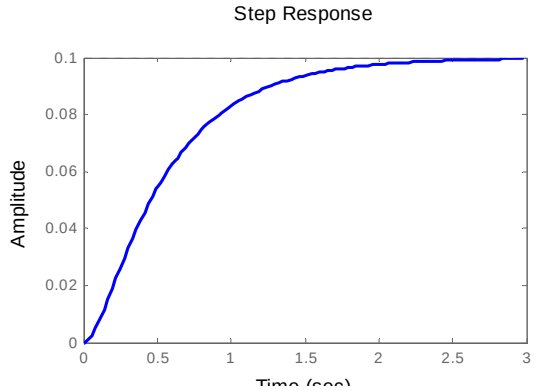
Biểu diễn hàm truyền này trong Matlab ta thực hiện như sau (sinh viên nên lưu thành **file.m**):

```
>>J=0.01;
>>b=0.1;
>>K=0.01;
>>R=1;
>>L=0.5;
>>num=K;                    % tử số của hàm truyền
```

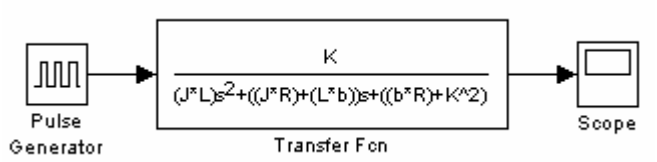
```
>>den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];        % mẫu số hàm truyền
>>hamtruyen = tf(num,den)
```

✓ Đáp ứng bước vòng hở:

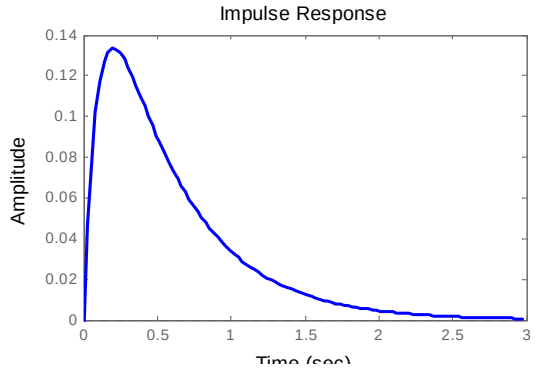
```
>>step(num,den)                                  % hoac
>>step(hamtruyen)
```



✓ Đáp ứng xung vòng hở:




```
>>impulse(hamtruyen)
```



② **Phương trình trạng thái:** Dạng tổng quát:

$$\dot{X} = AX + BU$$
$$Y = CX + DU$$

với X là véctơ trạng thái, U là véctơ tín hiệu vào và Y là véctơ tín hiệu ra.

✓ Biến trạng thái và phương trình trạng thái: Từ phương trình vi phân mô tả hệ thống, nếu đặt $x_1 = \dot{\theta}$ và $x_2 = i$, ta có:

$$\dot{x}_1 = -\frac{b}{J}x_1 + \frac{K}{J}x_2 + 0V$$
$$\dot{x}_2 = -\frac{K}{L}x_1 - \frac{R}{L}x_2 + \frac{1}{L}V$$
$$Y = x_1 + 0x2 + 0\ V$$

hay

$$\dot{X} = \begin{bmatrix} -\frac{b}{J} & \frac{K}{J} \\ -\frac{K}{L} & -\frac{R}{L} \end{bmatrix} X + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} V$$
$$Y = [1 \quad 0]X$$

✓ Biểu diễn phương trình trạng thái trong Matlab như sau:

```
>>J=0.01;
>>b=0.1;
>>K=0.01;
>>R=1;
>>L=0.5;
>>A = [-b/J   K/J;  -K/L   -R/L];
>>B = [0;1/L];
>>C=[1   0];
>>D=0;
```

✓ Đáp ứng bước vòng hở:

```
>>step(A,B,C,D)
```

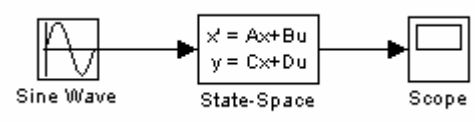
✓ Đáp ứng xung vòng hở:

```
>>impulse(A,B,C,D)
```

③ Ta có thể chuyển đổi qua lại giữa hàm truyền và phương trình trạng thái bằng lệnh sau:

```
>>[num,den]=ss2tf(A,B,C,D)     % từ PT trạng thái sang hàm truyền
>>[A,B,C,D]=tf2ss(num,den)     % từ hàm truyền sang PT trạng thái
```

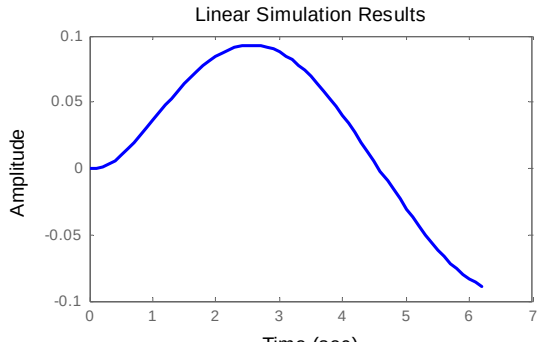
④ Khảo sát đáp ứng vòng hở của hệ thống đối với tín hiệu bất kỳ



(Hình 5.3)

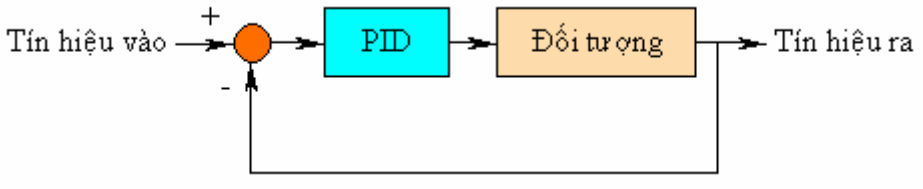
Phải đảm bảo rằng trong Workspace còn biến **hamtruyen** của câu ①, sinh viên có thể dùng lệnh **lsim** để khảo sát đáp ứng của hệ đối với tín hiệu bất kỳ. Giả sử đó là tín hiệu sin:

```
>>close  all
>>t=0:0.1:2*pi;
>>u=sin(pi/4*t);
>>lsim(hamtruyen,u,t)   % mo phong dap ung voi tin hieu vao u
```



III.2 Bộ điều khiển PID

Cấu trúc một hệ thống điều khiển PID như hình sau:



Hình 5.4 – Sơ đồ khối hệ điều khiển PID

Trong đó hàm truyền của khâu PID là: $K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$

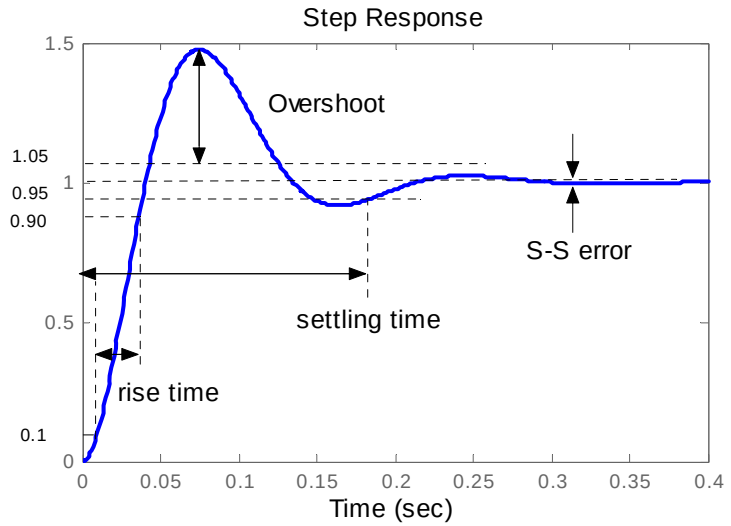
với: K_P là độ lợi của khâu tỉ lệ (Proportional gain)

K_I là độ lợi của khâu tích phân (Integral gain)

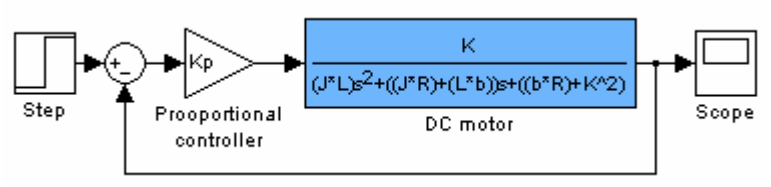
K_D là độ lợi khâu vi phân (Derivative gain)

Việc hiệu chỉnh phù hợp 3 thông số K_P, K_I và K_D sẽ làm tăng chất lượng điều khiển. Ảnh hưởng của 3 thông số này lên hệ thống như sau:

Đ. ứng vòng kín (C.L. response)	T. gian tăng (Rise time)	Vọt lố (Overshoot)	T. gian quá độ (Settling time)	Sai số xác lập (Steady-state err.)
K_P	Giảm	Tă ng	Ít thay đổI	Tă ng
K_I	Giảm	Tăng	Tăng	Không xác định
K_D	Ít thay đổi	Giảm	Giảm	Thay đổi ít



① Bộ điều khiển tỉ lệ P:

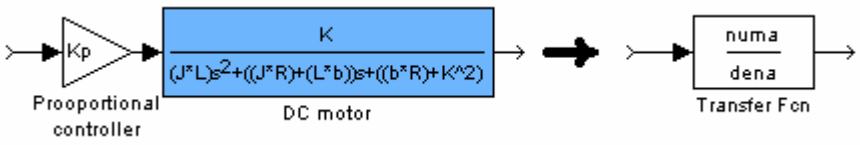


Hình 5.5 – Bộ điều khiển tỉ lệ P

✓ Thực hiện trong Matlab: Ta có hàm truyền của motor DC như III.1.①:

```
>>J=0.01;
>>b=0.1;
>>K=0.01;
>>R=1;
>>L=0.5;
>>num=K;
>>den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];
```

Khi thêm vào khâu tỉ lệ P, ta có hàm truyền vòng hở:



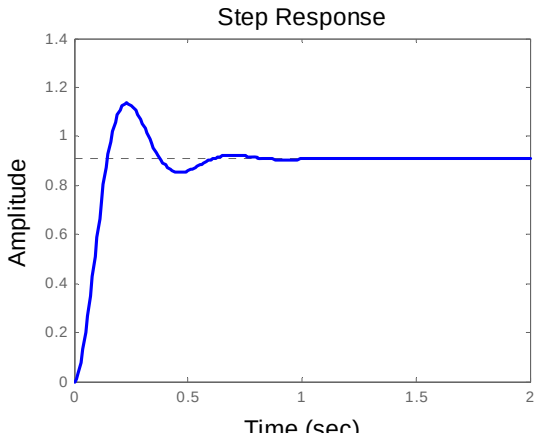

```
>>Kp=100;
>>numa=Kp*num;
>>dena=den;
```

Xác định hàm truyền vòng kín của hệ thống ta dùng lệnh **cloop**:

```
>>[numac,denac]=cloop(numa,dena)
```

Đáp ứng Step vòng kín của bộ điều khiển tỉ lệ như sau:

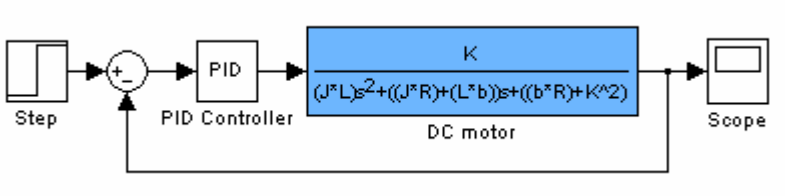
```
>>t=0:0.01:2;
>>step(numac,denac)
```



✓ Sinh viên hãy so sánh với đáp ứng của hệ khi chưa có bộ điều khiển tỉ lệ, ở câu III.1.① (lưu ý đến các thông số: thời gian lên, độ vọt lố, thời gian quá độ).

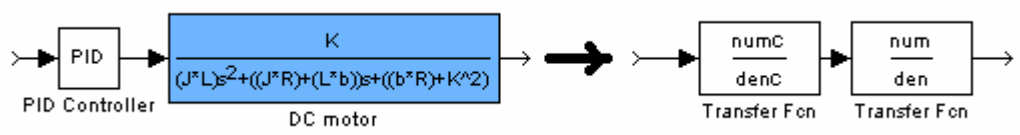
✓ Tương tự, sinh viên hãy so sánh với đáp ứng xung.

② Bộ điều khiển Vi tích phân tỉ lệ PID:



Hình 5.6 – Bộ điều khiển PID

Khi thêm bộ điều khiển PID, hàm truyền hở của hệ thống là:




```
>>Kp=100;
>>Ki=1;
>>Kd=1;
>>numc=[Kd, Kp, Ki];
>>denc=[1 0];
>>numa=conv(num,numc);          % tích chập tử số
>>dena=conv(den,denc);          % tích chập mẫu số
```

Hàm truyền vòng kín hồi tiếp âm đơn vị:

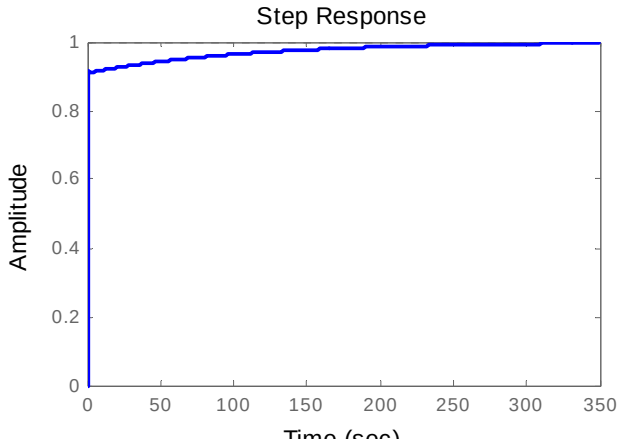
```
>>[numac,denac]=cloop(numa,dena);
```

Đáp ứng Step của hệ điều khiển PID:

```
>>step(numac,denac)
```

✓ Sinh viên so sánh với đáp ứng của bộ điều khiển tỉ lệ P ở câu ①, nhận xét.

✓ Dựa vào bảng tổng kết ảnh hưởng của K_P, K_D và K_I đối với hệ thống điều khiển, sinh viên hãy thay đổi 3 thông số này và kiểm chứng đáp ứng của hệ thống.



III.3 Hiệu chỉnh thông số của bộ điều khiển PID

Một phương pháp cổ điển nhưng đơn giản và hiệu quả để chỉnh định 3 thông số K_P, K_I và K_D của bộ điều khiển PID là phương pháp Ziegler-Nichols (Ziegler Nichols Tuning Method). Thủ tục chỉnh định như sau:

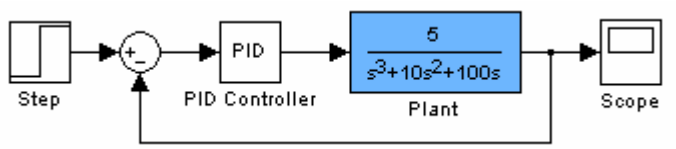
1. Chỉ điều khiển hệ thống bằng bộ điều khiển tỉ lệ K_P (đặt K_I=K_D=0).
2. Tăng K_P đến giá trị K_C mà ở đó hệ thống bắt đầu bất ổn (bắt đầu xuất hiện sự giao động - điểm cực của hàm truyền kín nằm trên trục ảo $j\varpi$). Xác định tần số ϖ_c của giao động vừa đạt.

   Từ 2 giá trị K_C và ϖ_c vừa đạt, các thông số số K_P, K_I và K_D được xác định như bảng sau:

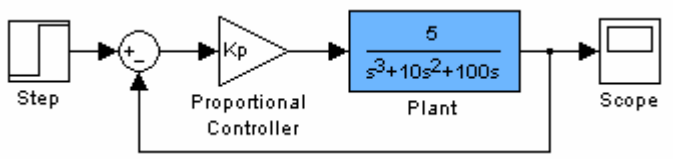
Bộ điều khiển	K_P	K_I	K_D
P (tỉ lệ)	0.5 K_C		
PI (tích phân tỉ lệ)	0.45 K_C	0.191$K_P\varpi_c$	
PID (vi tích phân tỉ lệ)	0.6 K_C	0.318$K_P\varpi_c$	0.785K_P/ϖ_c
PID với một ít vọt lố	0.33 K_C	0.318$K_P\varpi_c$	2.07K_P/ϖ_c
PID không vọt lố	0.2 K_C	0.53$K_P\varpi_c$	3.14K_P/ϖ_c

3. Tinh chỉnh lại 3 thông số này để đạt được đáp ứng như mong muốn.

① **Ví dụ:** Giả sử cần thiết kế bộ điều khiển PID cho hệ thống sau:

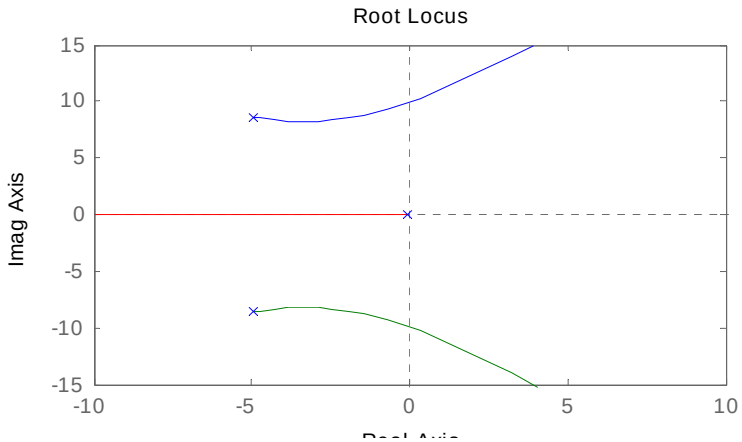


✓ Bước 1: Điều khiển hệ thống chỉ với bộ điều khiển tỉ lệ:



✓ Bước 2: Xác định K_C và ϖ_c mà ở đó hệ thống bắt đầu giao động - dùng hàm **rlocus** của Matlab (sinh viên nên lưu thành file **.m** hoặc thao tác trong Matlab Editor sau đó copy và dán vào Workspace cả đoạn lệnh để dễ dàng cho việc hiệu chỉnh các thông số ở phần sau):

```
>>close all
>>num=5;
>>den=[1 10 100 0];
>>[numc,denc]=cloop(num,den);
>>htkin=tf(numc,denc)          % ham truyen vong kin
>>rlocus(htkin);               %ve qui dao nghiem
>> axis([-10 10 -15 15])
```



Xác định Kc và ϖ_c bằng hàm **rlocfind**:

```
>>[Kc,Omegac] = rlocfind(htkin)
```

Nhấp chuột vào điểm giao nhau giữa quĩ đạo nghiệm và trục ảo của đồ thị, trong WorkSpace ta được:

```
Kc =

    199.5793

Omegac =

   -10.0145
     0.0072 +10.0072i
     0.0072 - 10.0072i
```

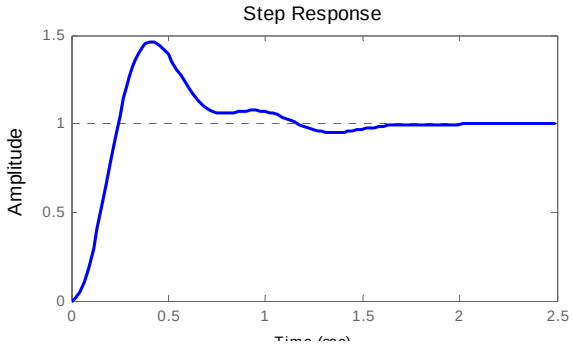
Như vậy ta được K_C=200 và ϖ_c = 10. Suy ra thông số của bộ điều khiển PID:

$K_P = 0.6K_C = 120$
$K_I = 0.318K_P\varpi_c = 381.6$
$K_D = 0.785K_P/\varpi_c = 9.4$

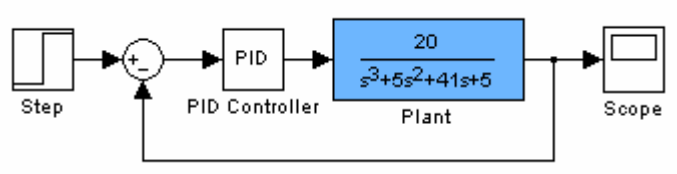
Thử đáp ứng của hệ:

```
>>Kp=120; Ki=381.5; Kd=9.4;
>>numc=[Kd, Kp, Ki];
>>denc=[1 0];                          % ham truyen cua PID
>>[numac,denac]=cloop(conv(num,numc),conv(den,denc))
>>step(numac,denac)
```

✓ Bước 3: Thực hiện tương tự như III.2.②, sinh viên hãy điều chỉnh một lượng nhỏ 3 thông số K_P, K_D và K_I để được đáp ứng tốt hơn.

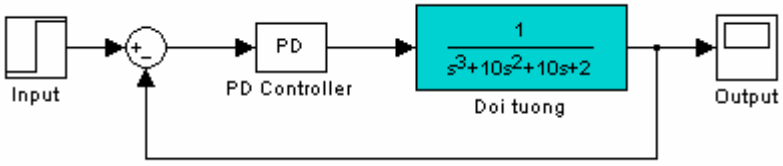


② Sinh viên hãy thiết kế bộ điều khiển PID cho hệ thống sau:

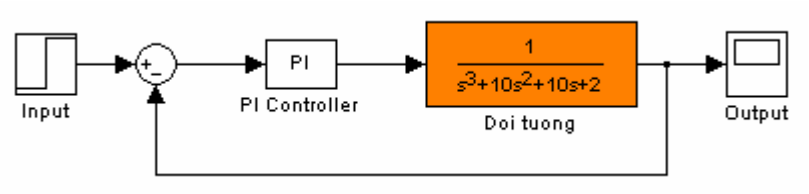


IV. Tự chọn

① Sinh viên hãy thiết kế bộ điều khiển Vi phân tỉ lệ (Proportional-Derivative controller):



② Sinh viên hãy thiết kế bộ điều khiển Tích phân tỉ lệ (Proportional-Integral controller):


Input
PI
PI Controller
1
s3+10s2+10s+2
Doi tuong
Output

Bài 6

TẠO GIAO DIỆN TRONG MATLAB

I. Mục tiêu

Bài thí nghiệm này trang bị cho sinh viên một số kiến thức cơ bản để xây dựng giao diện người dùng trong môi trường Matlab, nhằm hoàn thiện một chương trình ứng dụng nhất định.

II. Tham khảo

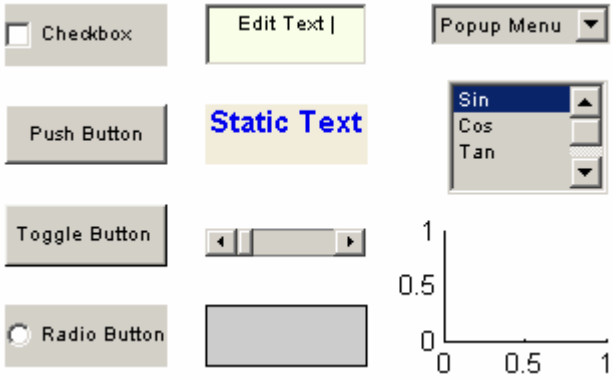
[1]. The Mathworks Inc., Matlab Notebook User's Guide, 2003.

[2]. Nguyễn Hữu Tình - Lê Tấn Hùng - Phạm Thị Ngọc Yến - Nguyễn Thị Lan Hương, Cơ sở Matlab & ứng dụng, NXB KH và Kỹ thuật, 1999.

[3]. Nguyễn Hoài Sơn - Đỗ Thanh Việt - Bùi Xuân Lâm, Ứng dụng MATLAB trong tính toán kỹ thuật, Tập 1, NXB ĐHQG Tp. HCM, 2000 (trang 328-344).

III. Thực hành

Cũng như các ngôn ngữ cấp cao khác, Matlab hổ trợ nhiều công cụ chức năng cho phép lập trình tạo giao diện sử dụng đẹp và nhanh chóng. Ví dụ, các dạng nút ấn, cửa sổ soạn thảo, các dạng menu, … như hình 6.1.



Hình 6.1 – Các công cụ hổ trợ giao diện

Trong bài thí nghiệm này, sinh viên lần lượt thực hiện các bước được nêu ra để được một giao diện đơn giản như hình vẽ 6.2. Trong mỗi bước thực hành, sinh viên hãy quan sát kỹ đáp ứng của chương trình, từ đó tự mình rút ra kinh nghiệm về việc điều chỉnh các thuộc tính đồ hoạ của Matlab.

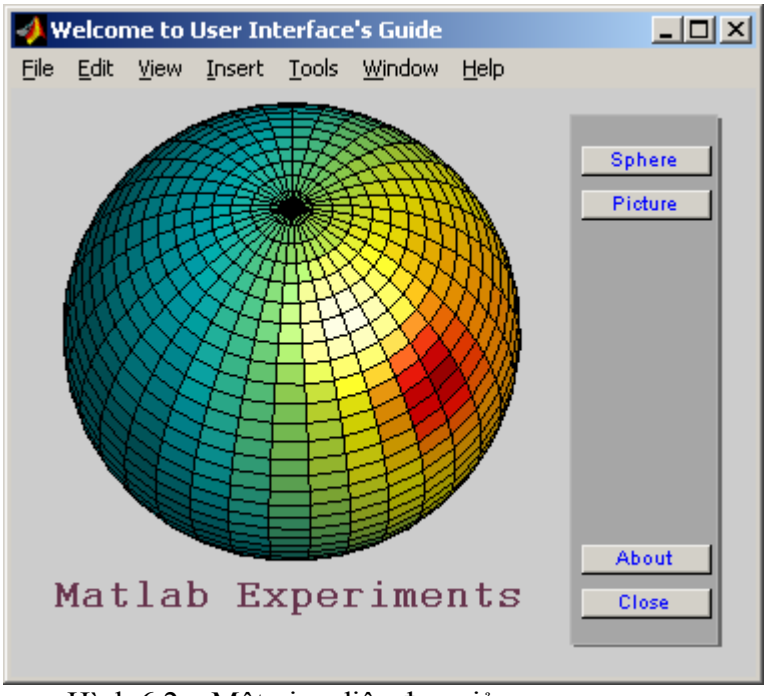
① Tạo cửa sổ chính **figure** – 'Welcome to User Interface's Giude' theo cách sau:

- Mở Matlab Editor
- Nhập nội dung sau:

```
% User Interface's Guide
%
% Matlab Experiments 2003
% TcAD, CIT, Cantho University

% ------------------------------------------------------------------------
% Initialize whole figure...
   namefig  = 'Welcome to User Interface''s Guide';
    figpos    = get(0,'DefaultFigurePosition');   % lay vi tri mac nhien
    figpos(1)= figpos(1)-10;    figpos(2)= figpos(2)-10;
    figpos(3)= figpos(3)+10;    figpos(4)= figpos(4)+10;

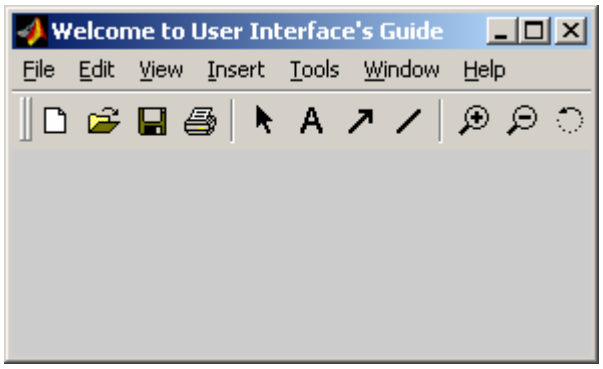
% Tao figure
   fig=figure( ...
            'Name', namefig, ...
            'NumberTitle','off', ...
            'Position',figpos);
```



Hình 6.2 – Một giao diện đơn giản

- Lưu thành file **giaodien.m**
- >>giaodien <↵> % thi hanh

Quan sát kết quả trên hình 6.3 (Lưu ý các thuộc tính: Name, Position)



(Hình 6.3)

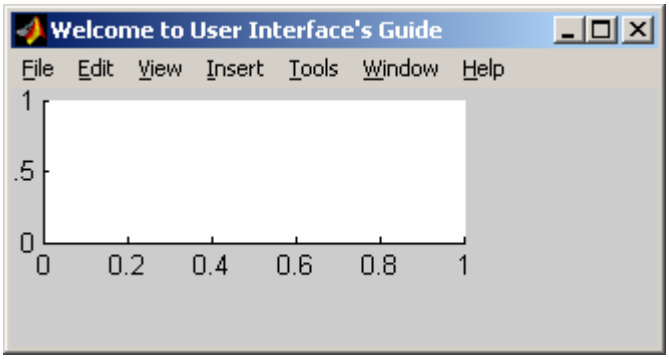
- >>set(fig) % xem thuộc tính của figure trong cửa sổ lệnh

② Thêm vào figure một axes cho phép hiển thị đồ họa:

- Nhập tiếp nội dung sau vào file **giaodien.m**

```
%  ------------------------------------------------------------------------
%  main axes
   axs=axes('Position',[0.05 0.4 0.65 0.55]);
```

- Lưu và thi hành file này.



(Hình 6.4)

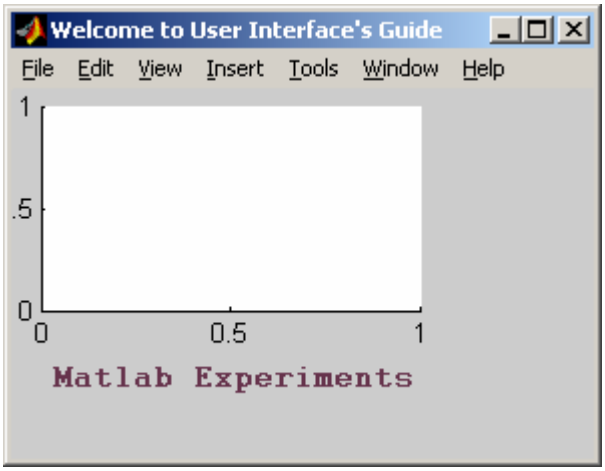
③ Hiển thị dòng **'Matlab Experiments'** bên dưới **axes**:

- Nhập tiếp nội dung sau vào file **giaodien.m**

```
% ------------------------------------------------------------------------
% text
   txtpos=[10 50 425 50];
   txt=uicontrol(...
            'Style','text',...
            'BackgroundColor',[0.8 0.8 0.8],...
            'ForegroundColor',[0.4 0.5 0.3],...
            'String','Matlab Experiments',...
```

```
        'Position',txtpos,...
        'Fontname','Courier',...
        'FontWeight','Bold',...
        'FontSize',26);
```

- Lưu và thi hành file này.



(Hình 6.5)

④ Tạo một **frame** có shadow bên phải **figure** để đặt các nút chức năng:

- Nhập tiếp nội dung sau vào file **giaodien.m**

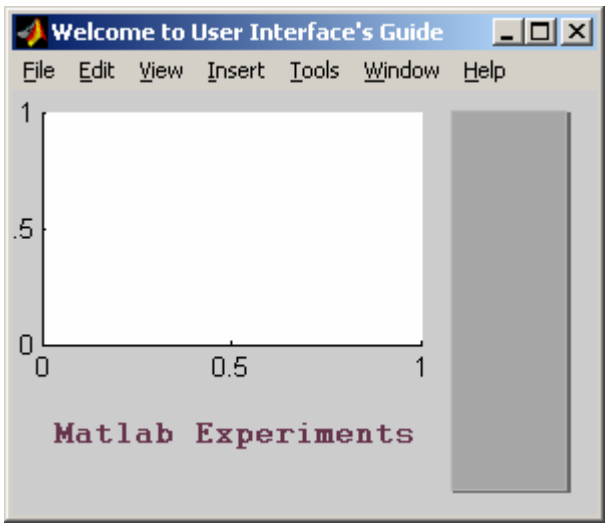
```
% ------------------------------------------------------------------------
% Console frames
   p1=0.755; p2=0.05; p3=0.2; p4=0.90;
   frm1pos = [p1 p2 p3 p4];
   frm2pos = [p1-0.005 p2+0.005 p3 p4];

% shadow frame
  frm1=uicontrol( ...
        'Style','frame', ...
        'Units','normalized', ...
        'Position',frm1pos, ...
        'ForegroundColor',[0.4 0.4 0.4],...
        'BackgroundColor',[0.4 0.4 0.4]);

% main frame
   frm2=uicontrol( ...
        'Style','frame', ...
        'Units','normalized', ...
        'Position',frm2pos, ...
        'ForegroundColor',[0.7 0.7 0.7],...
```

```
        'BackgroundColor',[0.65 0.65 0.65]);
```

- Lưu và thi hành file này.



(Hình 6.6)

⑤ Tạo nút **'Close'** có chức năng đóng cửa sổ figure hiện hành:

- Nhập tiếp nội dung sau vào file **giaodien.m**

```
% ------------------------------------------------------------------------
% Close button
   closeHndl=uicontrol(...
         'Style','pushbutton', ...
         'Units','normalized', ...
         'Position',[p1+0.01 p2+0.05 p3-0.025 0.05], ...
         'String','Close', ...
         'Foregroundcolor','b',...
         'Fontsize',9,...
         'Callback','close');
```

- Lưu và thi hành file này.

Vấn đề quan trọng nhất đối với một nút chức năng là thi hành công việc tương ứng khi người sử dụng thao tác. Thuộc tính **'CallBack'** cho phép: thi hành một lệnh của Matlab dưới dạng chuỗi (bao gồm lệnh gọi hàm, script file, biểu thức toán, …). Sinh viên thử thay lệnh **close** bằng một lệnh khác, chẳng hạn **demos** và quan sát đáp ứng.

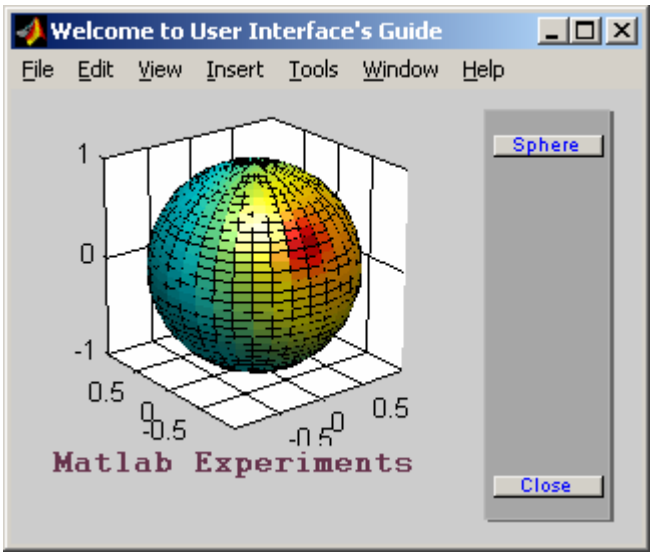
⑥ Tạo nút **Sphere** để vẽ một hình cầu 3D:

- Nhập tiếp nội dung sau vào file giaodien.m

```
% ------------------------------------------------------------------------
% Sphere button

   sph=uicontrol( ...
         'Style','pushbutton', ...
         'Units','normalized', ...
         'Position',[p1+0.01 p4-0.05 p3-0.025 0.05], ...
         'String','Sphere', ...
         'Foregroundcolor','b',...
         'Fontsize',9,...
         'Callback',['[x,y,z]=sphere(35);', 'surfl(x,y,z);', 'axis equal']);
```

- Lưu và thi hành file này.



(Hình 6.7)

⑦ Tạo nút **Picture** để hiển thị ảnh màu:

- Nhập tiếp nội dung sau vào file giaodien.m

```
% ------------------------------------------------------------------------
% Picture button
   pic=uicontrol( ...
         'Style','pushbutton', ...
         'Units','normalized', ...
         'Position',[p1+0.01 p4-0.125 p3-0.025 0.05], ...
         'String','Picture', ...
         'Foregroundcolor','b',...
         'Fontsize',9,...
         'Callback',['imshow(imread("flowers.tif"))']);
```

- Lưu và thi hành file này.



(Hình 6.8)

Sinh viên lưu ý cách biểu diễn nhiều lệnh ở dạng chuỗi cho **'CallBack'**. Trong trường hợp có quá nhiều lệnh phục vụ chức năng này, ta nên đưa chúng vào một **script file** hoặc một **hàm** khác. Ngoài ra, nếu ta tạo giao diện dưới dạng một hàm (function) thì thuộc tính **'CallBack'** cho phép gọi một **hàm con** được viết ngay trong file này.

⑧ Sinh viên hãy tạo thêm nút **'About'** sao cho khi người sử dụng ấn nút sẽ mở cửa sổ mới nằm ngay giữa màn hình 6.9.



(Hình 6.9)

IV. Tự chọn

Sinh viên hãy xem một số file tạo giao diện **demo** của các toolbox trong Matlab chẳng hạn: **graf3d.m** trong **\toolbox\matlab\demos**, để tham khảo cách xây dựng giao diện theo dạng hàm. Với cách này, thuộc tính **'CallBack'** trong **'uicontrol'** cho phép gọi trực tiếp một hàm mà hàm đó được xây dựng ngay trong file giao diện (nghĩa là không cần tạo một file ***.m** khác phục vụ cho **'CallBack'**).

Thử sửa lại giao diện của bài thí nghiệm này theo dạng trên.



BÀI 7

THIẾT KẾ – MÔ PHỎNG MẠCH VÀ VẼ MẠCH IN

I. Mục tiêu

Giúp sinh viên bước đầu làm quen với các phần mềm sau: **Multisim V6.20** (Electronics WorkBench) dùng để mô phỏng các mạch tương tự và mạch số; **OrCAD Release 9** dùng để vẽ Schematic, mô phỏng và vẽ mạch in (PCB).

II. Tham khảo

[1]. OrCAD® Inc., User's Guide - Capture, 1998.

[2]. OrCAD® Inc., User's Guide - Layout, 1998.

[3]. http://www.electronicsworkbench.com/tutorial/

[4]. http://www.rit.edu/~labeee/

III. Thực hành

Mặc dù **Multisim** và **OrCAD/Pspice** là các phần mềm rất lớn với nhiều tính năng áp dụng mạnh và phức tạp, nhưng sinh viên chuyên ngành Điện tử hoàn toàn có thể tự nghiên cứu được. Do đó, bài thí nghiệm này là không bắt buộc đối với tất cả sinh viên (đặc biệt là sinh viên Tin học, nếu có).

Trong trường hợp sinh viên không chọn bài thí nghiệm này thì có thể chọn **Control System Toolbox** (đối với sinh viên ĐKTĐ) hoặc **Communications Toolbox** (đối với sinh viên Viễn Thông) hoặc **Image Processing Toolbox** (đối với sinh viên Tin học) của MATLAB để khảo sát.

Bài thí nghiệm này tương đối dài, sinh viên cần sắp xếp thời gian hợp lý để thực hiện các công việc sau:

III.1 Multisim

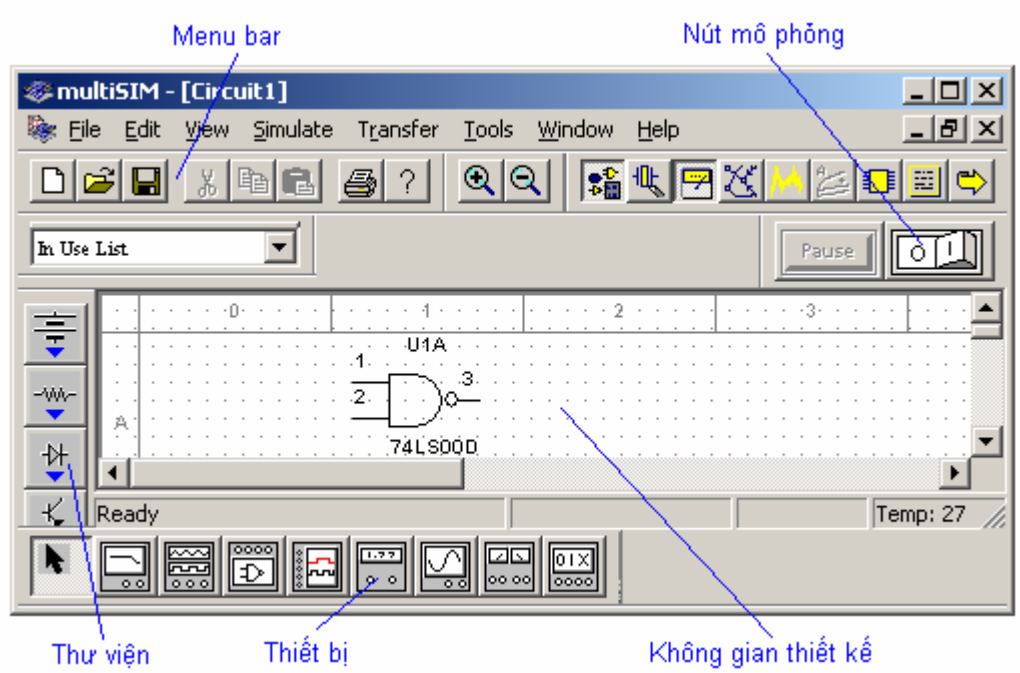
Trong phần này, chúng ta sẽ làm quen với Multisim 6.20 (hiện đã có Multisim 2001 hoàn thiện hơn), đây là Version mới của Electronics WorkBench 5.x, nó cho phép sử dụng kết hợp thư viện linh kiện Analog và Digital với nhiều công cụ thân thiện, hỗ trợ tuyệt vời cho việc mô phỏng các mạch điện tử.

① Khởi động Multisim:

- Chạy file Multisim.exe từ Start menu của Windows
- Nhấp vào biểu tượng  trên Desktop.

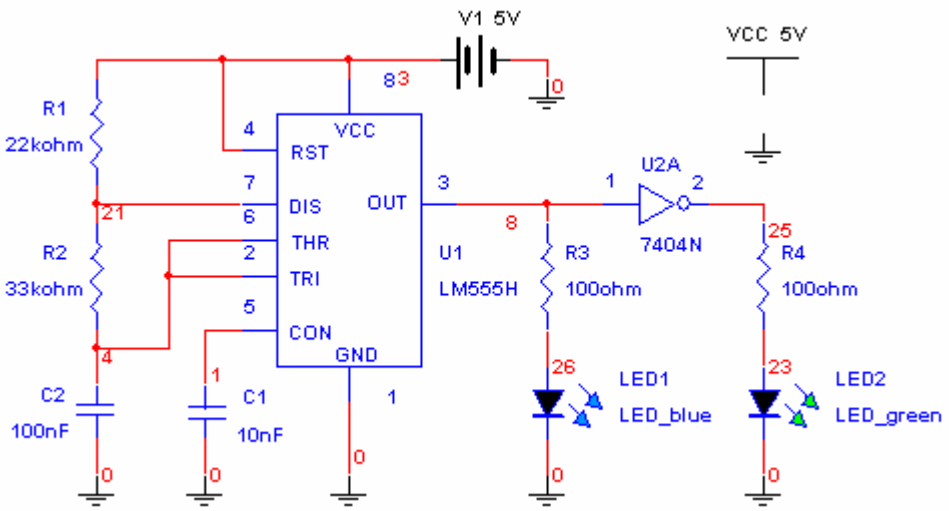
② Sinh viên hãy nhấp chuột vào các thư viện linh kiện của Multisim trên Menu bên trái của cửa sổ chính để có cái nhìn tổng quát về các linh kiện chức năng.

- Khi muốn đặt linh kiện nào vào không gian thiết kế, chỉ cần nhấp chuột vào linh kiện đó, rồi chọn thông số phù hợp và đặt vào vị trí mong muốn.
- Để tạo một kết nối giữa 2 điểm, chỉ cần nhấp chuột vào điểm thứ nhất rồi kéo đến điểm thứ hai và nhấp chuột lần nữa.



Hình 7.1 – Giao diện chính của MultiSIM

③ Sinh viên hãy vẽ và mô phỏng mạch điện sau:



Hình 7.2 – Một mạch điện điển hình

- Dùng Osilsoscope quan sát dạng tín hiệu ở chân 2-6 và chân 3 của LM555.
- Hãy ước lượng tần số giao động và chu kỳ thao tác của mạch.

Kinh nghiệm: Đôi lúc Multisim đưa ra bảng thông báo lỗi trong quá trình mô phỏng hay hiệu chỉnh thông số của các linh kiện. Trường hợp đó, ta gở bỏ khối Osilloscope, hiệu chỉnh và mô phỏng vài lần rồi gắn Osilloscope trở lại để quan sát dạng tín hiệu. Ngoài ra, nếu trong mạch có sử dụng các IC số (74xxx, 40xxx, …) ta phải có nguồn Vcc=5V như hình vẽ trên.

III.2 OrCAD

Các chức năng chính của OrCad gồm: Vẽ sơ đồ mạch (OrCAD Capture), mô phỏng mạch (OrCAD Pspice) và vẽ mạch in (OrCAD Layout). Tuy nhiên, trong phần thí nghiệm này chúng ta không đề cập đến OrCAD Pspice (đây là một trong những chức năng rất mạnh của OrCAD phiên bản 9.0 trở lên, đề nghị sinh viên tự nghiên cứu).

III.2.1 Vẽ sơ đồ mạch (Schematic)

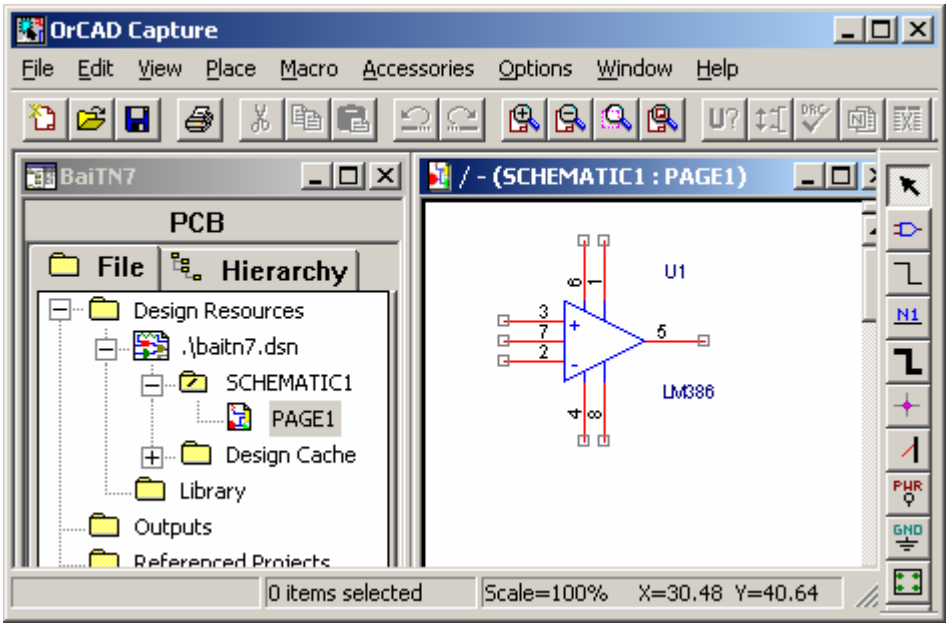
① Khởi động OrCAD Capture bằng một trong các cách sau:

- Chọn **Capture CIS** từ Start Menu của Window
- Nhấp vào biểu tượng  trên desktop

Để tạo một Project mới cho việc vẽ sơ đồ mạch ta thực hiện:

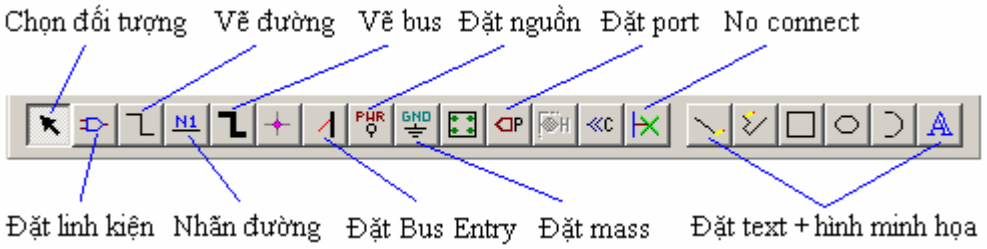
- Chọn **File** → **New Project…** hoặc nhấp vào icon  trên Menu bar
- Khi hộp thoại **'New Project'** xuất hiện: Chọn **'Schematic'** và đặt tên project trong menu **'Name'** (giả sử là BaiTN7). Có thể chọn thư mục lưu project này trong menu **'Location'**.

Môi trường vẽ mạch xuất hiện khi ta nhấp đúp vào 'PAGE1' (hình 7.3). Từ đây ta lần lượt đặt các linh kiện vào và kết nối để được sơ đồ mong muốn.



Hình 7.3 – Môi trường vẽ mạch điện của OrCAD

Sử dụng Menu tắt sau:

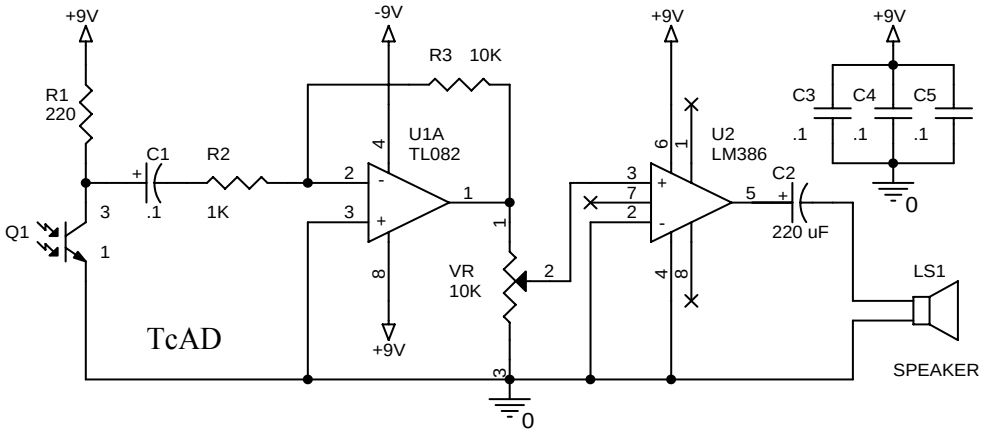


(Hình 7.4)

② Sinh viên hãy vẽ sơ đồ mạch như hình sau. Trong đó các thư viện linh kiện được sử dụng như sau:

- **Discrete:** R, Capacitor Pol, Capacitor Non-Pol, Resistor Var, Photo PNP và Speaker.
- **Opamp:** TL082
- **Amplifier:** LM386

Lưu Schematic vừa vẽ với tên **BaiTN7.**

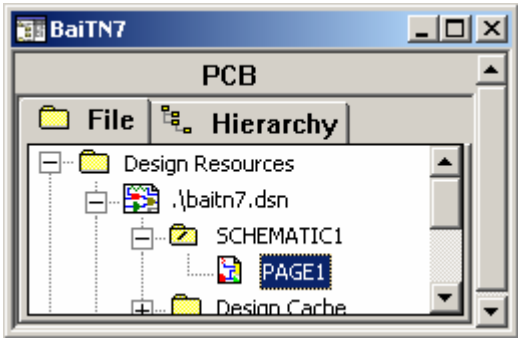


(Hình 7.5)

III.2.2 Tạo Netlist

Sau khi lưu sơ đồ vừa vẽ, để tạo Netlist phục vụ cho việc vẽ mạch in ta thực hiện như sau:

- Đóng cửa sổ **'Schematic1:Page1'**, quay về Project Management **'BaiTN7'**



(Hình 7.6)

- Chọn **Tools - Annotate...** để đánh số thứ tự cho linh kiện.
- Chọn **Tool - Create Netlist....** để tạo Netlist.
- Khi cửa sổ '**Create Netlist**' xuất hiện, ta chọn **layout Tab** (file dữ liệu netlist là BaiTN7.MNL), **OK**.

Nếu đảm bảo rằng có file BaiTN7.MNL trong thư mục hiện hành thì thoát khỏi OrCAD Capture để chuẩn bị cho việc vẽ mạch in.

Lưu ý: Trong phần trên chúng ta không đề cập đến chức năng Kiểm tra mạch của OrCAD và một số chức năng hỗ trợ khác.

III.2.3 Vẽ mạch in (Printed Circuit Board-PCB)

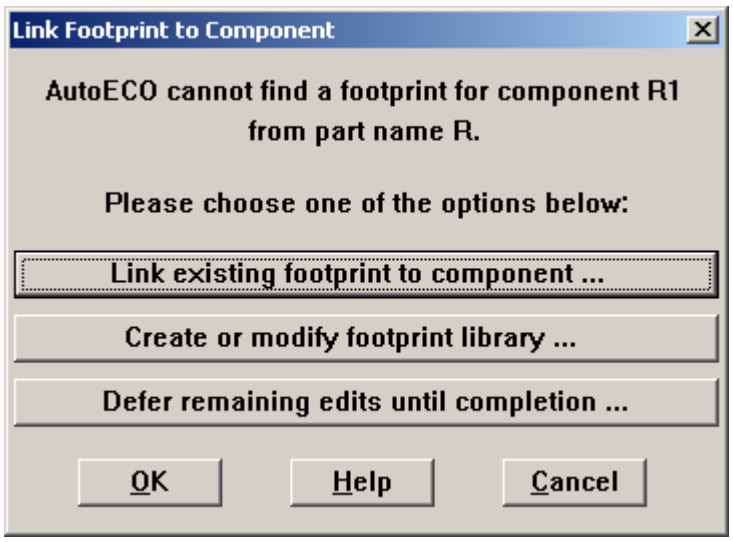
① Khởi động OrCAD Layout bằng một trong các cách sau:

- Chọn **Layout Plus** trong Start Menu của Window
- Nhấp vào biểu tượng  trên desktop.
- Chọn **File → New** để tạo một PCB layout mới

Sau thao tác này, cửa sổ **'Load Template File'** xuất hiện, ta cần đặt đường dẫn chỉ đến file **DEFAULT.tch** trong OrCAD/Layout_plus/DATA. Sau đó thực hiện nạp file Netlist vừa tạo ở bước III.3.2 (ví dụ: BaiTN7.MNL).

② Đặt **footprint** cho các linh kiện:

Nếu trong quá trình vẽ Schematic ta không chọn footprint cho các linh kiện mỗi khi đặt vào thì ở bước này OrCAD sẽ yêu cầu chọn kiểu chân cho các linh kiện theo menu sau:

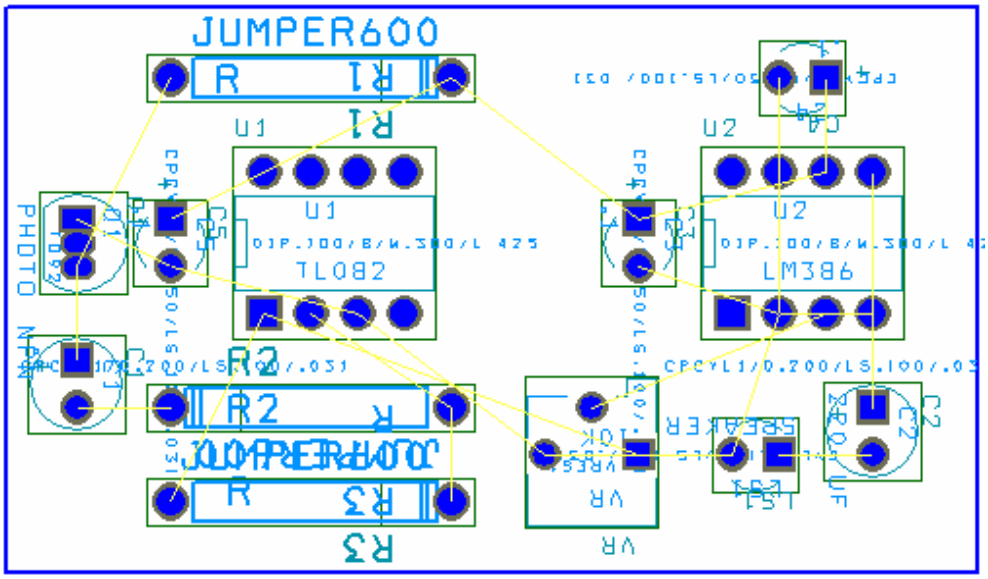


(Hình 7.7)

Nhấp vào menu **'Link existing footprint to component …'**, chọn lựa kiểu chân phù hợp cho từng loại linh kiện. Ở bước này, kiểu chân linh kiện cần chọn đúng kích thước (có thể phải tạo các footprint mới). Đối với bài thí nghiệm này, sinh viên có thể tham khảo bảng sau:

Linh kiện	Footprint	Thư viện
R	Jumper600	JUMPER
C	CPCYL1/D.200/LS.100/.031	TM_CAP_P
Photo NPN	TO92	TO
TL082, LM386	DIP.100/8/W.300/LS.100/.425	DIP100T
VR	VRES1	VRES
Speaker	CYL/D.150/LS.100/.31	TM_CYLND

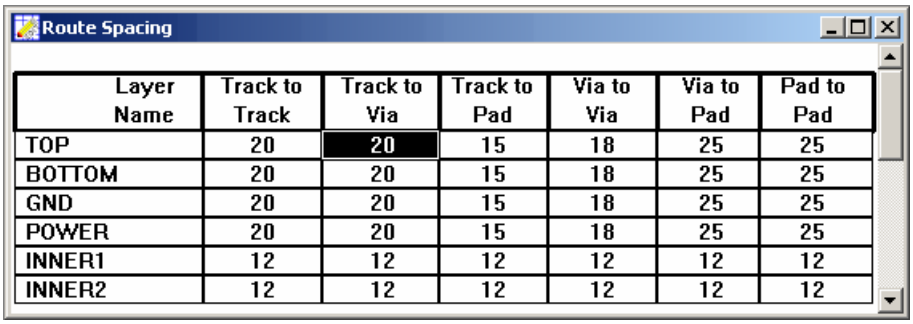
③ Sắp xếp linh kiện: Quan sát menu bar, chọn chức năng thích hợp và sắp xếp các linh kiện trên board mạch. Ở bước này, nếu cần thiết ta có thể đưa vào thêm các linh kiện mới, chẳng hạn các lỗ (hold) để bắt ốc cho board mạch.



(Hình 7.8)

④ Đặt kích thước các đường mạch và khai báo số lớp:

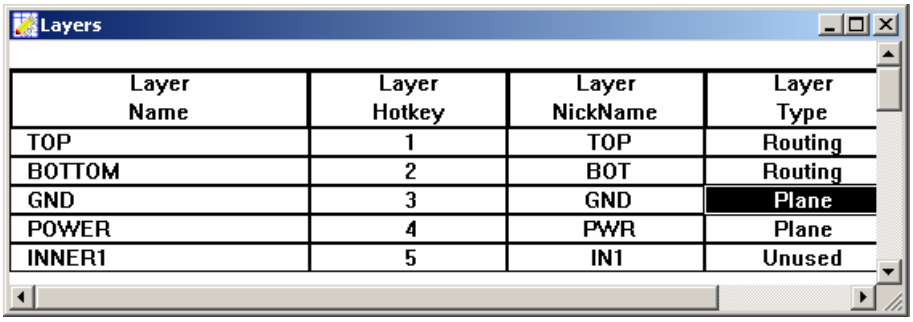
- Chọn Options → Global Spacing … để đặt khoảng cách giữa Track, Via và Pad:



Layer Name	Track to Track	Track to Via	Track to Pad	Via to Via	Via to Pad	Pad to Pad
TOP	20	20	15	18	25	25
BOTTOM	20	20	15	18	25	25
GND	20	20	15	18	25	25
POWER	20	20	15	18	25	25
INNER1	12	12	12	12	12	12
INNER2	12	12	12	12	12	12

(Hình 7.9)

- Chọn **Options → Jumper Settings …, OK** hoặc **View SpreadSheet → Layer** để báo số lớp vẽ. Trong bài thí nghiệm này, ta vẽ board mạch 2 lớp, nên chỉ có TOP và BOTTOM là Routing.



Layer Name	Layer Hotkey	Layer NickName	Layer Type
TOP	1	TOP	Routing
BOTTOM	2	BOT	Routing
GND	3	GND	Plane
POWER	4	PWR	Plane
INNER1	5	IN1	Unused

(Hình 7.10)

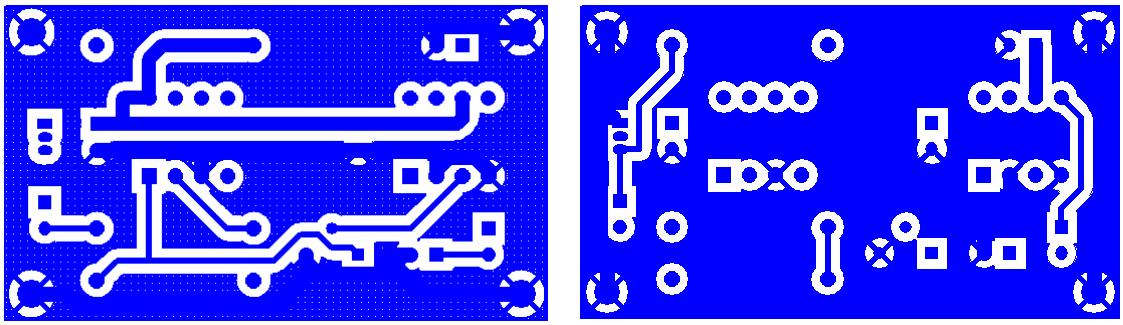
- Chọn **View SpreadSheet → Nets** để đặt kích thước các đường mạch:

Nets

Net Name	Color	Width Min Con Max	Routing Enabled
+9V		45, 55, 60	Yes
-9V		45, 55, 60	Yes
0		55, 60, 65	Yes
N00085		20, 25, 30	Yes
N00331		20, 25, 30	Yes
N00334		20, 25, 30	Yes

(Hình 7.11)

⑤ **Vẽ mạch:** OrCAD hỗ trợ nhiều nhiều chiến lược vẽ mạch. Để khai thác tốt các chiến lược này đòi hỏi phải có một số kinh nghiệm nhất định. Thông thường, người ta hay vẽ các đường Nguồn – Mass với các tụ lọc nguồn thích hợp trước, sau đó thực hiện vẽ tự động từng phần và cuối cùng kết nối các phần vừa vẽ. Nên nhớ rằng, chế độ vẽ tự động chỉ giúp giảm thời gian, không có phần mềm nào vẽ mạch tự động tốt hơn tư duy của con người.



(Hình 7.12)

Một số thông số cần lưu ý (áp dụng cho các sinh viên vẽ và thuê gia công mạch Luận văn tốt nghiệp tại một số cơ sở ở Tp. HCM):

- Đường kính tối thiểu của các lỗ chân cắm và các via xuyên mạch: 55 mils (1000 mils=1 inches). Các lỗ chân linh kiện nên đặt thành hình oval.
- Độ rộng tối thiểu của các đường mạch (track width): 15 mils (hiện nay tại Tp. HCM một số công ty có thể thực hiện mạch với độ rộng đường 6 mils).
- Khoảng cách Track to Track nên >15 mils
- Để có thể vẽ xuyên qua giữa 2 chân IC, phải đặt khoảng cách Track to Pad thích hợp, nên 12 ÷ 15 mils.

IV. TỰ chỌn

① Tự thiết kế và mô phỏng một mạch điện tử bằng Multsim

② Vẽ mạch in cho mạch vừa thiết kế bằng OrCAD.



Nguyễn Chí Ngôn, *ThS., Giảng viên*
Bộ môn Viễn Thông - Tự động hóa
Khoa Công nghệ Thông tin, Đại học Cần thơ
Email: ncngon@ctu.edu.vn
Mobile Tel: 0913199766

© TcAD - 2003