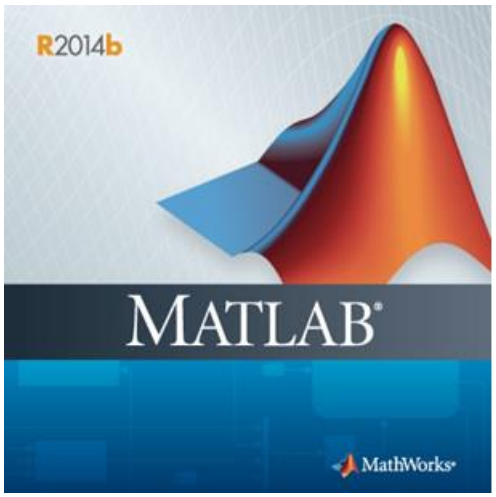
CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN VỀ MATLAB

1.1. GIỚI THIỆU MATLAB:

MATLAB (Matrix Laboratory) là một công cụ phần mềm của Math Work dùng để giải các bài toán kỹ thuật, đặc biệt là các bài toán liên quan đến ma trận. MATLAB cho phép tính toán số với ma trận, vẽ đồ thị hàm số hay biểu đồ thông tin, thực hiện thuật toán, tạo các giao diện người dùng và liên kết với những chương trình máy tính viết trên nhiều ngôn ngữ lập trình khác. MATLAB giúp đơn giản hóa việc giải quyết các bài toán tính toán kĩ thuật so với các ngôn ngữ lập trình truyền thống như C, C++, và Fortran. MATLAB được sử dụng trong nhiều lĩnh vực, bao gồm xử lý tín hiệu và ảnh, truyền thông, thiết kế điều khiển tự động, đo lường kiểm tra, phân tích mô hình tài chính, hệ thống điều khiển, mạng neuron, fuzzy logic, tính toán sinh học….

Cửa sổ biểu tượng của chương trình MATLAB:



Hình 1.1 - Cửa sổ khởi động của MATLAB

1.2. CÁC PHẦN CƠ BẢN TRONG LẬP TRÌNH MATLAB:

1.2.1 Các phép toán và toán tử

- Các phép toán: + , - , * , / , \ (chia trái) , ^ (mũ) , ‘ (chuyển vị hay số phức liên hiệp).
- Các toán tử quan hệ : < , <= , > , >= , == , ~=

- Các toán tử logic : & , | (or) , ~ (not)
- Các hằng trong Matlab:

pi: 3.14159265

i, j: số ảo

eps: sai số 2-52

realmin: số thực nhỏ nhất 2-1022

realmax: số thực lớn nhất 21023

inf: vô cùng lớn

NaN: Not a number

❖ **Chú ý :**

+ Các lệnh kết thúc bằng dấu chấm phẩy, MATLAB sẽ không thể hiện kết quả trên màn hình.

+ Các chú thích được đặt phía sau dấu %.

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

1.2.2. Khai báo biến

- Phân biệt chữ hoa và chữ thường.
- Không cần phải khai báo kiểu biến.
- Tên biến phải bắt đầu bằng ký tự và không được có khoảng trắng.
- Không đặt tên trùng với các tên đặc biệt của MATLAB.
- Để khai báo biến toàn cục (sử dụng được trong tất cả chương trình con), phải dùng thêm từ khoá ***global*** phía trước.

1.2.3 Các lệnh thường dùng

```
>>help tên_hàm     % tham khảo help của hàm
>>lookfor 'chuỗi'  %Tìm kiếm chuỗi
>>clc              % Xoá màn hình
>>clear tên_biến   % Xoá biến
>>clear all        %Xoá tất cả các biến
>>clf              %Xoá figure
```

```
>>save              % Lưu các biến hiện có trong bộ nhớ
>>load              % Lấy nội dung các biến đã lưu
>>who          % liệt kê các biến trong bộ nhớ
>>whos         % liệt kê chi tiết các biến trong bộ nhớ
>>which        % Xác định vị trí của hàm hay file
>>what         % Liệt kê các file có trong một thư mục
```

Ví dụ:

```
>>which plot  %Xác định vị trí của hàm plot
>>lookfor 'filter' % Tìm các hàm có liên quan đến mạch lọc
```

1.3 LẬP TRÌNH TRONG MATLAB

1.3.1 Các phát biểu điều kiện if, else, elseif

Cú pháp của if:

```
if <biểu thức điều kiện>
     <phát biểu>
end
```

Nếu `<biểu thức điều kiện>` cho kết quả đúng thì phần lệnh trong thân của if được thực hiện. Các phát biểu else và elseif cũng tương tự.

Ví dụ

```
     >>n= -10:10
     if n<0
          x(n)=0;
     eleif n>0;
          x(n)=(0.8).^n;
     else x(n)=0
     end
```

1.3.2 Switch

Cú pháp của switch như sau:

```
switch <biểu thức>
```

```
case n1
     <lệnh 1>
case n2
     <lệnh 2>
. . . . . .
case nn
     <lệnh n>
Otherwise
     <lệnh n+1>
End
```

1.3.3 While

Vòng lặp while dùng khi không biết trước số lần lặp. Cú pháp của nó như sau:

```
while <biểu thức>
     <phát biểu>
End
```

Ví dụ:

```
     >>n=1
     while n<8
          x(n)=0.5^n;
          n=n+1
     end
```

1.3.4 For

Vòng lặp for dùng khi biết trước số lần lặp. Cú pháp như sau:

```
for <chỉ số>=<giá trị đầu>:<mức tăng>:<giá trị cuối>
```

Ví dụ:

```
     >>for n=1:0.5:10
          x(n)=n^2+4*n^2
     end
```

1.3.5 Break:

Phát biểu break để kết thúc vòng lặp for hay while mà không quan tâm đến điều kiện kết thúc vòng lặp đã thoả mãn hay chưa.

1.4 MA TRẬN

1.4.1 Các thao tác trên ma trận

1.4.1.1 Nhập ma trận

Ma trận là một mảng có m hàng và n cột. Trường hợp ma trận chỉ có một phần tử (ma trận 1x1) ta có một số. Ma trận chỉ có một cột hay một hàng được gọi là một vector. Ta có thể nhập ma trận vào MATLAB bằng nhiều cách:

• Nhập một danh sách các phần tử từ bàn phím.

• Nạp ma trận từ file.

• Tạo ma trận nhờ các hàm có sẵn trong MATLAB.

• Tạo ma trận nhờ hàm tự tạo.

Khi nhập ma trận từ bàn phím ta phải tuân theo các quy định sau:

• Ngăn cách các phần tử của ma trận bằng dấu “,” hay khoảng trắng.

• Dùng dấu “;” để kết thúc một hàng.

• Bao các phần tử của ma trận bằng cặp dấu ngoặc vuông [ ].

1.4.1.2 Chỉ số

Phần tử ở hàng i cột j của ma trận có ký hiệu là A(i,j). Tuy nhiên, ta cũng có thể tham chiếu tới phần tử của mảng nhờ một chỉ số, ví dụ A(k). Trong trường hợp này, ma trận được xem là một cột dài tạo từ các cột của ma trận ban đầu.

Như vậy viết A(8) có nghĩa là tham chiếu phần tử A(4, 2) (nếu ma trận có 4 hàng). Lưu ý rằng các chỉ số của ma trận thường bắt đầu từ 1.

1.4.1.3 Toán tử “:”

Toán tử “:” là một toán tử quan trọng của MATLAB. Nó xuất hiện ở nhiều dạng khác nhau. Biểu thức 1:10 là một vector hàng chứa 10 số nguyên từ 1 đến 10.

```
>>1:10
>>100:-7:50 %tạo dãy số từ 100 đến 51, cách đều nhau 7
>>0: pi/4: pi %tạo một dãy số từ 0 đến π, cách đều nhau π/4
```

Các biểu thức chỉ số có thể tham chiếu tới một phần của ma trận. A(1:k,j) xác định k phần tử đầu tiên của cột j. Ngoài ra toán tử ":" tham chiếu tới tất cả các phần tử của một hàng hay một cột.

Ví dụ:

```
>>A(:,3)
>>A(3, :)
>>B = A(:, [1 3 2 4]) %tạo ma trận B từ ma trận A bằng cách đổi thứ tự các cột từ [1 2 3 4] thành [1 3 2 4]
```

1.4.1.4 Tạo ma trận bằng hàm có sẵn

MATLAB cung cấp một số hàm để tạo các ma trận cơ bản:

- ***zeros*** tạo ra ma trận mà các phần tử đều là 0.

>>z = zeros(2, 4)

- ***ones*** tạo ra ma trận mà các phần tử đều là 1.

>>x = ones(2, 3)

>>y = 5*ones(2, 2)

- ***rand*** tạo ra ma trận mà các phần tử ngẫu nhiên phân bố đều.

>>d = rand(4, 4)

- ***randn*** tạo ra ma trận mà các phần tử ngẫu nhiên phân bố chuẩn.

>>e = randn(3, 3)

- ***magic(n)*** tạo ra ma trận cấp n gồm các số nguyên từ 1 đến n2 với tổng các hàng bằng tổng các cột và bằng tổng các đường chéo ($n \geq 3$).

- ***pascal(n)*** tạo ra tam giác Pascal.

>>pascal(4)

- ***eye(n)*** tạo ma trận đơn vị

>>eye(3)

- ***eye(m,n)*** tạo ma trận đơn vị mở rộng

>>eye(3,4)

1.4.1.5 Nối ma trận

Ta có thể nối các ma trận có sẵn thành một ma trận mới. Ví dụ:

```
>>a = ones(3, 3);
>>b = 5*ones(3, 3);
>>c = [a+2; b];
```

1.4.1.6 Xoá hàng và cột

Ta có thể xoá hàng và cột từ ma trận bằng dùng dấu [].

```
>>b(:, 2) = [] ; %xoá cột thứ 2
>>b(1:2:5) = []; % xoá các phần tử bắt đầu từ 1 đến 5 và cách 2 (1,3,5) rồi sắp xếp lại ma trận.
```

1.4.1.7 Các lệnh xử lý ma trận

Cộng : X = A + B

Trừ : X = A - B

Nhân : X = A*B

X = A.*B nhân các phần tử tương ứng với nhau, yêu cầu 2 ma trận A và B phải có cùng kích thước.

Chia : X = A/B lúc đó `X = A * inv(B)`

X = A\B lúc đó `X = inv(A) * B`

X=A./B chia các phần tử tương ứng với nhau, 2 ma trận A và B có cùng kích thước.

Luỹ thừa : `X = A^2`

`X = A.^2`

Nghịch đảo: `X = inv(A)`

Định thức: `d = det(A)`

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)
```

1.4.2 Vector

Vector thực chất là ma trận có kích thước n x 1 hay 1 x n, nên ta có thể tạo ra vector 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) % vector 20 phần tử cách dều nhau từ 1 đến 10
>>z=rand(10,1) ; tạo 10 số ngẫu nhiên phân bố đều
```

1.4.3 Đa thức

Các đa thức trong MATLAB được mô tả bằng các vector hàng với các phần tử của vector chính là các hệ số của đa thức, xếp theo thứ tự số mũ giảm dần. Ví dụ, đa thức m = s4-s3+4s2-5s-1 được biểu diễn là:

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

Để xác định giá trị của đa thức, ta dùng hàm ***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 hàm ***roots***. Ví dụ:

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

1.5 ĐỒ HOẠ TRONG MATLAB:

1.5.1 Các lệnh vẽ:

MATLAB cung cấp một loạt hàm để vẽ biểu diễn các vector cũng như giải thích và in các đường cong này.

plot: đồ họa 2-D với số liệu 2 trục vô hướng và tuyến tính

plot3: đồ họa 3-D với số liệu 2 trục vô hướng và tuyến tính

loglog: đồ hoạ với các trục x, y ở dạng logarit

semilogx: đồ hoạ với trục x logarit và trục y tuyến tính

semilogy: đồ hoạ với trục y logarit và trục x tuyến tính

1.5.2 Tạo hình vẽ:

Hàm ***plot*** có các dạng khác nhau phụ thuộc vào các đối số đưa vào. Ví dụ nếu y là một vector thì plot(y) tạo ra một đường quan hệ giữa các giá trị của y và chỉ số của nó. Nếu ta có 2 vector x và y thì plot(x,y) tạo ra đồ thị quan hệ giữa x và y.

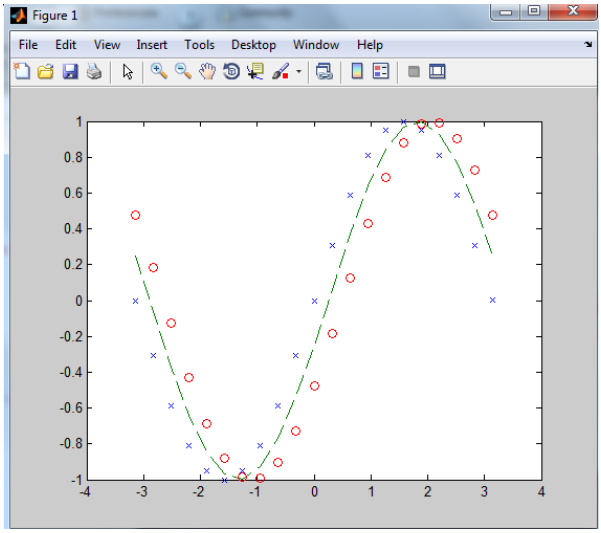
```
>>t = [0:pi/100:2*pi];
>>y = sin(t);
>>plot(t,y); % Vẽ hình sin từ 0->2π
>>grid on
```

1.5.3 Kiểu đường vẽ:

Ta có thể dùng các kiểu đường vẽ khác nhau khi vẽ hình.

Ví dụ:

```
>> y1 = sin(x);
>> y2 = sin(x-0.25);
>> y3 = sin(x-0.5);
>> plot(x,y1,'x',x,y2,'--',x,y3,'o')
```



❖ **Tham số để xác định màu và kích thước đường vẽ:**

- **LineWidth** : độ rộng đường thẳng, tính bằng số điểm
- **MarkerEdgeColor** : màu của các cạnh của khối đánh dấu
- **MarkerFaceColor** : màu của khối đánh dấu
- **MarkerSize** : kích thước của khối đánh dấu

❖ **Tham số xác định màu:**

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

❖ **Các dạng đường thẳng xác định bằng:**

- đường liền -- đường đứt nét

: đường chấm chấm **-.** đường chấm gạch

❖ **Các dạng điểm đánh dấu xác định bằng:**

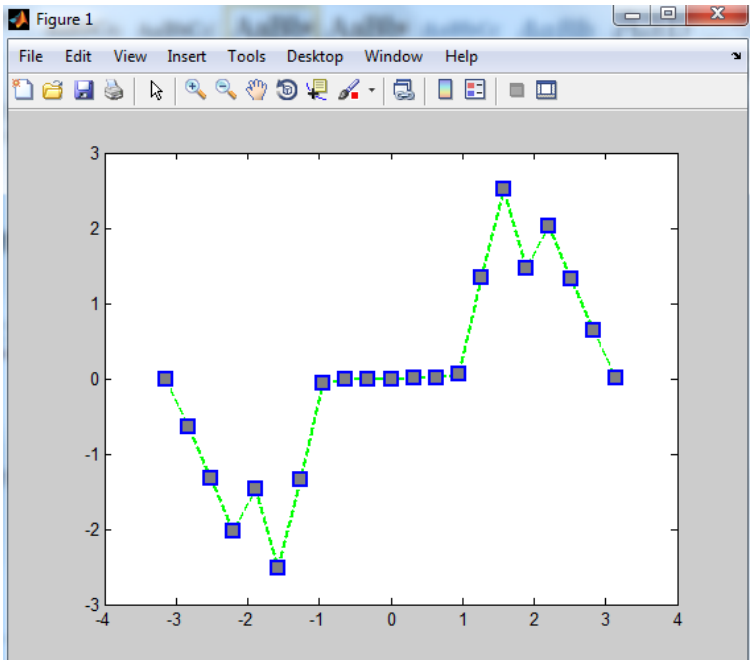
+ dấu cộng **.** điểm **o** vòng tròn **x** chữ thập ***** dấu sao

s hình vuông **h** lục giác **p** ngũ giác **d** hạt kim cương

^ tam giác hướng lên < tam giác sang trái > tam giác sang phải

v tam giác hướng xuống

```
>> x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
figure
plot(x,y,'--gs',...
     'LineWidth',2,...
     'MarkerSize',10,...
     'MarkerEdgeColor','b',...
     'MarkerFaceColor',[0.5,0.5,0.5])
```



Để vẽ hai hàm trên cùng một đồ thị, ta dùng lệnh:

```
>>hold on
```

1.5.4 Vẽ với hai trục y

Hàm ***plotyy*** cho phép tạo một đồ thị có hai trục y. Ta cũng có thể dùng plotyy để cho giá trị trên hai trục y có kiểu khác nhau nhằm tiện so sánh.

```
>>t = 0:900;
>>A = 1000;
>>b = 0.005;
>>a = 0.005;
>>z2 = sin(b*t);
>>z1 = A*exp(-a*t);
>>[haxes,hline1,hline2]=
plotyy(t,z1,t,z2,'semilogy','plot');
```

1.5.5 Vẽ đường cong 3-D

Nếu x, y, z là 3 vector có cùng độ dài thì plot3 sẽ vẽ đường cong 3D.

```
>>t = 0:pi/50:10*pi;
>>plot3(sin(t),cos(t),t);
>>axis square;
>>grid on
```

1.5.6 Vẽ nhiều trục toạ độ

Dùng hàm ***subplot*** để vẽ nhiều trục toạ độ.

```
>>subplot(2,3,5) %2,3: xác định có 2 hàng, 3 cột
% 5: chọn trục thứ 5 (đếm từ trái sang phải, trên xuống
dưới)
>>x=linspace(0,2*pi);
>>y1=sin(x);
>>y2=cos(x)
>>y3=2*exp(-x).*sin(x);
>>x1=linspace(-2*pi,2*pi);
```

```
>>y4=sinc(x1);
>>subplot(221);
>>plot(x,y1);
>>title('Ham y = sinx');
>>subplot(222);
>>plot(x,y2);
>>title('Ham y = cosx');
>>subplot(223);
>>plot(x,y3);
>>title('Ham y = 2e^{-x}sinx');
>>subplot(224);
>>plot(x1,y4);
>>title('Ham y = $${sin \pi x \over \pi
x}$$','interpreter','latex');
```

1.5.7 Đặt các thông số cho trục

Khi ta tạo một hình vẽ, MATLAB tự động chọn các giới hạn trên trục toạ độ và khoảng cách đánh dấu dựa trên dữ liệu dùng để vẽ. Tuy nhiên ta có thể mô tả lại phạm vi giá trị trên trục và khoảng cách đánh dấu theo ý riêng. Ta có thể dùng các hàm sau:

- ***axis*** đặt lại các giá trị trên trục toạ độ.
- ***axes*** tạo một trục toạ độ mới với các đặc tính được mô tả.
- ***get*** và ***set*** cho phép xác định và đặt các thuộc tính của trục toạ độ đang có.
- ***gca*** trở về trục toạ độ cũ.

1.5.7.1 Giới hạn của trục và chia vạch trên trục

MATLAB chọn các giới hạn trên trục toạ độ và khoảng cách đánh dấu dựa trên số liệu dùng để vẽ. Dùng lệnh axis có thể đặt lại giới hạn này. Cú pháp của lệnh:

```
axis[xmin , xmax , ymin , ymax]
>>x = 0:0.025:pi/2;
>>plot(x,tan(x),'-ro')
```

```
>>axis([0 pi/2 0 5])
```

MATLAB chia vạch trên trục dựa trên phạm vi dữ liệu và chia đều. Ta có thể mô tả cách chia nhờ thông số xtick và ytick bằng một vector tăng dần.

Ví dụ:

```
>>x = -pi:.1:pi;
>>y = sin(x);
>>plot(x,y)
>>set(gca,'xtick',-pi:pi/2:p);
>>set(gca,'xticklabel',{'-pi','-pi/2','0','pi/2','pi'})
```

1.5.7.2 Ghi nhãn lên các trục toạ độ

MATLAB cung cấp các lệnh ghi nhãn lên đồ hoạ gồm:

- ***title*** thêm nhãn vào đồ hoạ.
- ***xlabel*** thêm nhãn vào trục x.
- ***ylabel*** thêm nhãn vào trục y.
- ***zlabel*** thêm nhãn vào trục z.
- l***egend*** thêm chú giải vào đồ thị.
- ***text*** hiển thị chuỗi văn bản ở vị trí nhất định.
- ***gtext*** đặt văn bản lên đồ hoạ nhờ chuột.

Ví dụ:

```
>>x = -pi:.1:pi;
>>y = sin(x);
>>plot(x,y)
>>xlabel('t = 0 to 2\pi','Fontsize',16)
>>ylabel('sin(t)','Fontsize',16)
>>title('\it{Gia tri cua sin tu zero đến 2 pi}','Fontsize',16)
```

Ta có thể thêm văn bản vào bất kỳ chỗ nào trên hình vẽ nhờ hàm text.

```
>>text(3*pi/4,sin(3*pi/4),'\leftarrowsin(t)=0.707','Font Size',12)
```

Ta có thể sử dụng đối tượng văn bản để ghi chú các trục ở vị trí bất kỳ. MATLAB định vị văn bản theo đơn vị dữ liệu trên trục. Ví dụ để vẽ hàm $y = e^{\alpha t}$ với A = 0.25, t = 0 đến 900 và $\alpha = 0.005$.

```
>>t = 0:900;
>>plot(t,0.25*exp(-0.005*t))
```

1.5.8 Đồ hoạ đặc biệt

1.5.8.1 Khối và vùng

Đồ hoạ khối và vùng biểu diễn số liệu là vector hay ma trận. MATLAB cung cấp các hàm đồ hoạ khối và vùng:

- ***bar*** hiển thị các cột của ma trận m*n như là m nhóm, mỗi nhóm có n bar.
- ***barh*** hiển thị các cột của ma trận m*n như là m nhóm, mỗi nhóm có n bar nằm ngang.
- ***bar3*** hiển thị các cột của ma trận m*n như là m nhóm, mỗi nhóm có n bar dạng 3D.
- ***bar3h*** hiển thị các cột của ma trận m*n như là m nhóm, mỗi nhóm có n bar dạng 3D nằm ngang.

Mặc định, mỗi phần tử của ma trận được biểu diễn bằng một bar.

```
>>y = [5 2 1
6 7 3
8 6 3
5 5 5
1 5 8];
>>bar(y)
```

Sau đó nhập lệnh ***bar3(y)*** ta có đồ thị 3D.

1.5.8.2 Xếp chồng đồ thị

Ta có thể xếp chồng số liệu trên đồ thị thanh bằng cách tạo ra một trục khác trên cùng một vị trí và như vậy ta có một trục y độc lập với bộ số liệu khác.

```
>>TCE = [515 420 370 250 135 120 60 20];
>>nhdo = [29 23 27 25 20 23 23 27];
```

```
>>ngay = 0:5:35;
>>bar(ngay,nhdo)
>>xlabel('Ngay')
>>ylabel('Nhiet do (^{o}C)')
```

Để xếp chồng một số liệu lên một đồ thị thanh ở trên, có trục thứ 2 ở cùng vị trí như trục thứ nhất ta viết :

```
>>h1 = gca;
```

Tạo trục thứ 2 ở vị trí trục thứ nhất trước nhất vẽ bộ số liệu thứ 2:

```
>>h2 = axes('Position',get(h1,'Position'));
>>plot(days,TCE,'LineWidth',3)
```

Để trục thứ 2 không gây trở ngại cho trục thứ nhất ta viết :

```
>>set(h2,'YAxisLocation','right','Color','none','XTickLa
bel',[])
>>set(h2,'XLim',get(h1,'XLim'),'Layer','top')
```

Để ghi chú lên đồ thị ta viết:

```
>>text(11,380,'Mat do','Rotation',--55,'FontSize',16)
>>ylabel('TCE Mat do (PPM)')
>>title('Xep chong do thi','FontSize',16)
```

1.5.8.3 Đồ hoạ vùng

Hàm ***area*** hiển thị đường cong tạo từ một vector hay từ một cột của ma trận. Nó vẽ các giá trị của một cột của ma trận thành một đường cong riêng và tô đầy vùng không gian giữa các đường cong và trục x.

```
>>Y = [5 1 2
8 3 7
9 6 8
5 5 5
4 2 3];
>>area(Y)
```

1.5.9 Vẽ đồ thị

Bài 1.1. Vẽ đồ thị hàm số y1=sinx.cos2x và hàm số y2=sinx2 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 ung 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')
```

Bài 1.2. Thực hiện như trên nhưng dùng các hàm semilogx, semilogy, loglog thay thế cho plot.

Bài 1.3. Thực hiện như trên cho hàm số y = $e-x2e-x$+2

Bài 1.4. Vẽ hàm số r = sin (5π) trong toạ độ cực:

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

Bài 1.5. Vẽ hàm số r = 2sin(π) + 3cos(π)

Bài 1.6. Vẽ hàm số 2x2 + y2 = 10 ở dạng toạ độ cực.

x = rcosπ, y = rsinπ

1.6. CÁC FILE VÀ HÀM

1.6.1. Script file (file kịch bản)

Kịch bản là M-file đơn giản nhất, không có đối số. Nó dùng khi thi hành một loạt lệnh MATLAB theo một trình tự nhất định. Ta xét ví dụ tạo ra các số Fibonacci nhỏ hơn 1000.

```
f = [1 1];
i = 1;
while(f(i)+f(i+1))<1000
f(i + 2)= f(i) +f(i+1);
i = i + 1;
end
plot(f)
```

Ta lưu đoạn mã lệnh này vào một file tên là *fibo.m*. Đây chính là một script file. Để thực hiện các mã chứa trong file fibo.m từ cửa sổ lệnh ta nhập:

```
>>fibo
```

Và nhấn enter. Kết quả MATLAB sẽ vẽ ra đồ thị của chuỗi Fibonacci.

1.6.2. File hàm

Hàm là M-file có chứa các đối số. Ta có một ví dụ về hàm:

```
function y = tb(x)
%Tinh tri trung binh cua cac phan tu
[m,n] = size(x);
if m == 1
m = n;
end
y = sum(x)/m;
```

Từ ví dụ trên ta thấy một hàm M-file gồm các phần cơ bản sau :

• Một dòng định nghĩa hàm: **function y = tb(x)** gồm từ khoá **function**, đối số trả về **y**, tên hàm **tb** và đối số vào **x**.

• Dòng kế tiếp là dòng trợ giúp đầu tiên. Vì đây là dòng văn bản nên nó phải đặt sau %. Nó xuất hiện khi ta nhập lệnh help <tên hàm>. Phần văn bản này giúp người dùng hiểu tác dụng của hàm.

• Thân hàm chứa mã MATLAB.

• Các lời giải thích dùng để cho chương trình rõ ràng. Nó được đặt sau dấu %.

Cần chú ý là tên hàm phải bắt đầu bằng ký tự và ***cùng tên với file chứa hàm***. Tên hàm là ***tb*** thì tên file cũng là ***tb.m***.

Từ cửa sổ MATLAB ta gõ lệnh:

```
>>z = 1:99;
>>tb(z)
```

Các biến khai báo trong một hàm của MATLAB là biến địa phương. Các hàm khác không nhìn thấy và sử dụng được biến này. Muốn các hàm khác dùng được biến nào đó của hàm ta cần khai báo nó là ***global***.

Nếu hàm có nhiều thông số ngõ vào và ngõ ra thì khai báo như sau:

```
function [y1,y2,y3] = tb(x1,x2,x3)
```

Lưu ý rằng trong một file .m có thể có nhiều hàm nhưng hàm đầu tiên phải có tên trùng với tên file.

1.6.3. Các hàm toán học cơ bản

- **exp(x)** hàm mũ cơ số e
- **sqrt(x)** căn bậc hai của x
- **log(x)** logarit cơ số e
- **log10(x)** logarit cơ số 10
- **abs(x)** module của số phức x (giá trị tuyệt đối của số thực)
- **angle(x)** argument của số phức a
- **conj(x)** số phức liên hợp của x
- **imag(x)** phần ảo của x
- **real(x)** phần thực của x
- **sign(x)** dấu của x
- **cos(x)** tính cos
- **sin(x)** tính sin
- **tan(x)** tính tang
- **acos(x)** tính cos-1

- **asin(x)** tính sin-1
- **atan(x)** tính tang-1
- **cosh(x)** tính $ex+e-x2$
- **coth(x)** tính cosh(x)/sinh(x)
- **sinh(x)** tính $ex-e-x2$
- **tanh(x)** tính sinh(x)/cosh(x)
- **acosh(x)** tính cosh-1
- **acoth(x)** tính coth-1
- **asinh(x)** tính sinh-1
- **atanh(x)** tính tanh-1

1.6.4 Các phép toán trên hàm toán học

a. Biểu diễn hàm toán học

MATLAB biểu diễn các hàm toán học bằng cách dùng các biểu thức đặt trong **M-file**. Ví dụ để khảo sát hàm:

$$f(x) = \frac{2}{x+1} + \frac{1}{x^2+2} - 1$$

Ta tạo ra một file, đặt tên là ***ab.m*** có nội dung:

```
function y = ab(x)
y = 2./(x + 1) + 1./(x.^2 + 2) -1 ;
```

Cách thứ hai để biểu diễn một hàm toán học trên dòng lệnh là tạo ra một đối tượng inline từ một biểu thức chuỗi. Ví dụ ta có thể nhập từ dòng lệnh như sau:

```
>>f = inline('2./(x + 1) + 1./(x.^2 + 2) -1');
```

Ta có thể tính trị của hàm tại x = 2 như sau:

```
>>f(2)
```

b. Vẽ đồ thị của hàm

Hàm ***fplot*** vẽ đồ thị hàm toán học giữa các giá trị đã cho.

```
>>fplot(@(x)[tan(x),sin(x),cos(x)], 2*pi*[-1 1 -1 1])
```

c. Tìm cực tiểu của hàm

Cho một hàm toán học một biến, ta có thể dùng hàm ***fminbnd*** của MATLAB để tìm cực tiểu địa phương của hàm trong khoảng đã cho.

```
>>f=inline('1./((x-0.3).^2+0.01)+1./(x.^2+0.04)-6');
>>x = fminbnd(f,0.3,1)
```

Hàm ***fminsearch*** tương tự hàm ***fminbnd*** dùng để tìm cực tiểu địa phương của hàm nhiều biến.

Ta có hàm ***three_var.m***:

```
function b = three_var(v)
x = v(1);
y = v(2);
z = v(3);
b = x.^2 + 2.5*sin(y) -z^2*x^2*y^2;
```

Và bây giờ tìm cực tiểu đối với hàm này bắt đầu từ x = -0.6, y = -1.2; z = 0.135

```
>>v = [-0.6 -1.2 0.135];
>>a = fminsearch('three_var',v)
```

d. Tìm điểm zero

Hàm ***fzero*** dùng để tìm điểm không của hàm một biến. Ví dụ để tìm giá trị không của hàm lân cận giá trị -0.2, ta viết:

```
>>f=inline('1./((x-0.3).^2+0.01)+1./(x.^2+0.04)-6');
>>a = fzero(f,-0.2)
```

1.6.5 Thực hành trên script và function

1.6.5.1 Script

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**. 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
```

1.6.5.2 Sử dụng các tool xây dựng sẵn

MATLAB hỗ 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 neural (Neural networks), …

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

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
```

1.6.5.3 Xây dựng hàm

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).

```
function [out1,out2,…]=tenham(in1,in2,…)
% ------------------------------------
```

```
% 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 quả 1          %kết quả trả về của hàm
     out2=kết quả 2
…
% Các hàm con (nếu có)
function [subout1,subout2,…]=tenhamcon(subin1,subin2,…)
<Các câu lệnh của hàm con>
```

Bài 1.7.. Xây dựng hàm ***gptb2*** để giải phương trình bậc hai.

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 la nghiem
% a, b, c la 3 he so cua phuong trinh
if nargin<3
error('Error! Nhap 3 he so cua phuong trinh')
elseif a==0
x1=-c/b;
x2=[];
else
delta = b^2 - 4*a*c;
x1 = (-b+sqrt(delta))/(2*a);
x2 = (-b-sqrt(delta))/(2*a);
end
```

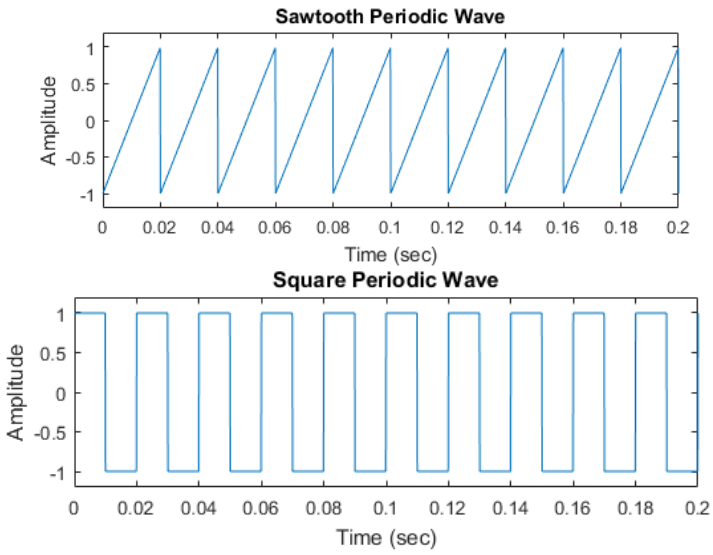
Lưu tên file là ***gptb2.m*** và kiểm tra kết quả:

```
>>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)
```

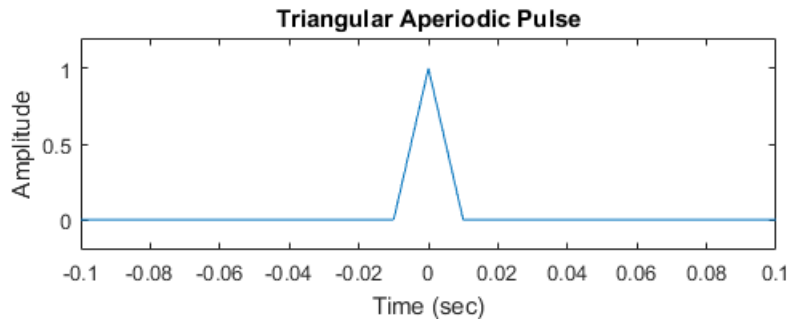
Bài 1.8: Hàm **sawtooth** tạo ra sóng răng cưa với đỉnh tại +/- 1 và hàm square tạo sóng xung vuông

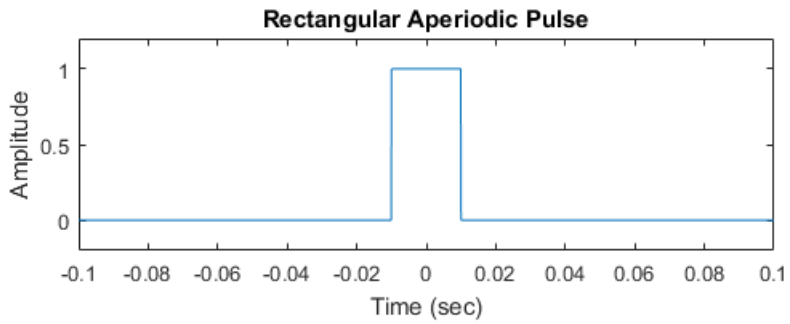
```
fs = 10000;
t = 0:1/fs:1.5;
x1 = sawtooth(2*pi*50*t);
x2 = square(2*pi*50*t);
subplot(211),plot(t,x1), axis([0 0.2 -1.2 1.2])
xlabel('Time (sec)');
ylabel('Amplitude');
title('Sawtooth Periodic Wave');
subplot(212)
plot(t,x2)
axis([0 0.2 -1.2 1.2]);
xlabel('Time (sec)');
ylabel('Amplitude');
title('Square Periodic Wave');
```



Bài 1.9: Hàm tripuls và rectpuls

```
fs = 10000;
t = -1:1/fs:1;
x1 = tripuls(t,20e-3);
x2 = rectpuls(t,20e-3);
subplot(211)
plot(t,x1)
axis([-0.1 0.1 -0.2 1.2])
xlabel('Time (sec)');
ylabel('Amplitude');
title('Triangular Aperiodic Pulse');
subplot(212)
plot(t,x2)
axis([-0.1 0.1 -0.2 1.2])
xlabel('Time (sec)');
ylabel('Amplitude');
title('Rectangular Aperiodic Pulse');
hgcf = gcf;
hgcf.Color = [1,1,1];
```



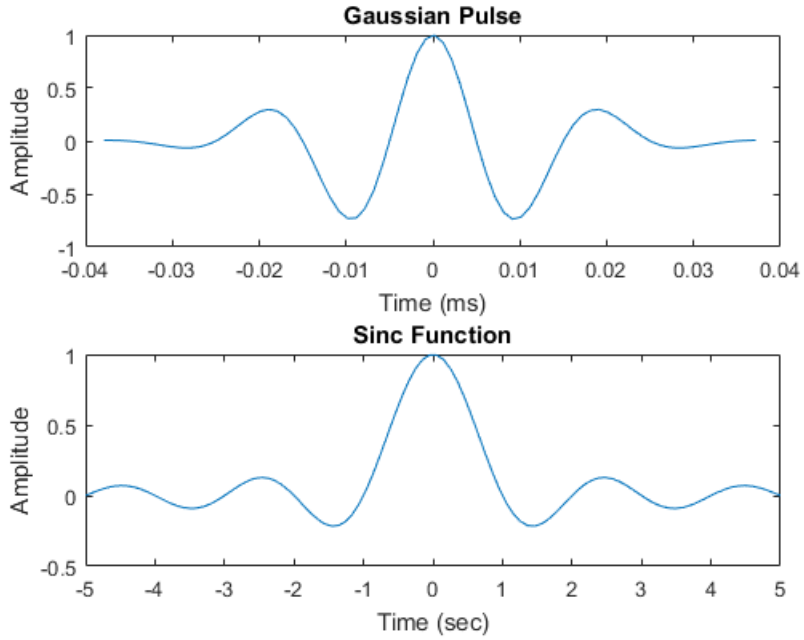


Bài 1.10: Hàm **gauspuls** và hàm **sinc**

Tạo một 50 kHz Gaussian RF xung với 60% băng thông, lấy mẫu ở tốc độ 1 MHz

Vẽ hàm Sa

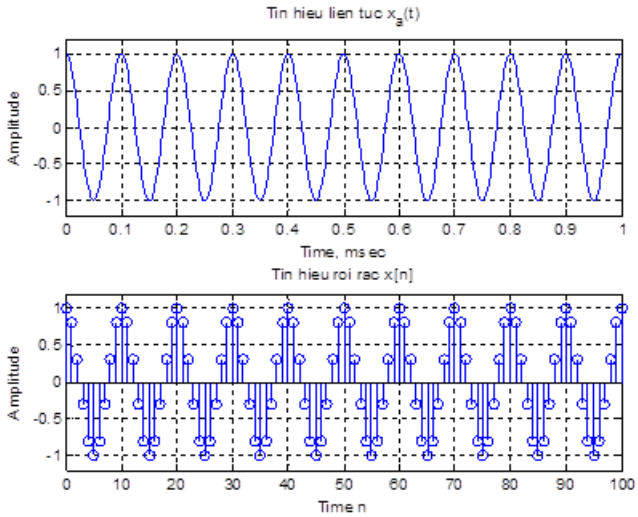
```
tc = gauspuls('cutoff',50e3,0.6,[],-40);
t1 = -tc : 1e-6 : tc;
y1 = gauspuls(t1,50e3,0.6);
t2 = linspace(-5,5);
y2 = sinc(t2);
subplot(211)
plot(t1*1e3,y1);
xlabel('Time (ms)');
ylabel('Amplitude');
title('Gaussian Pulse');
subplot(212),plot(t2,y2);
xlabel('Time (sec)');
ylabel('Amplitude');
title('Sinc Function');
hgcf = gcf;
hgcf.Color = [1,1,1];
```



Bài 1.11: Tạo tín hiệu rời rạc từ tín hiệu liên tục

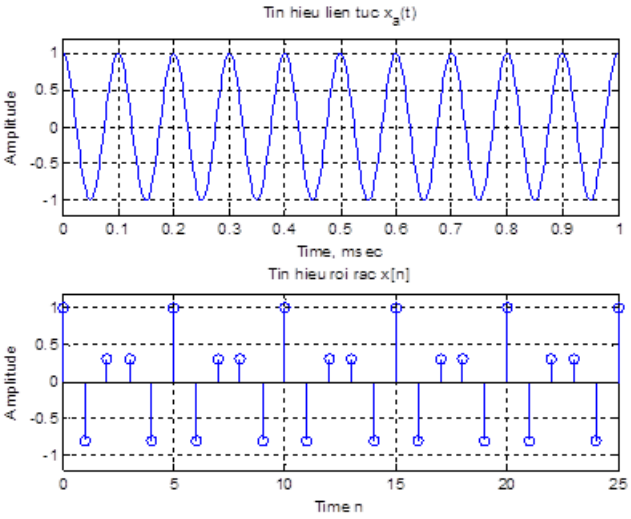
```
clf;
t = 0:0.0005:1;
f =10 ;
xa = cos(2*pi*f*t);
subplot(2,1,1)
plot(t,xa);grid
xlabel('Time, msec');ylabel('Amplitude');
title('Tin hieu lien tuc x_{a}(t)');
axis([0 1 -1.2 1.2])
subplot(2,1,2);
T = 0.01; % tan so lay mau tin hieu
n = 0:T:1;
xs = cos(2*pi*f*n);
k = 0:length(n)-1;
stem(k,xs); grid
xlabel('Time n');ylabel('Amplitude');
```

```
title('Tin hieu roi rac x[n]');
axis([0 (length(n)-1) -1.2 1.2])
```



Kết quả tạo tín hiệu rời rạc từ việc lấy mẫu tín hiệu liên tục

Thay giá trị của tần số lấy mẫu, ta có kết quả khác như sau:



1.7 BÀI TẬP

Bài tập 1: Cho tín hiệu tương tự:

$$x_a(t) = 3\cos 100\pi t$$

a. Tìm tần số lấy mẫu nhỏ nhất có thể mà không bị mất thông tin.

b. Giả sử tín hiệu được lấy mẫu ở tần số Fs = 200 Hz. Tìm tín hiệu lấy mẫu

c. Giả sử tín hiệu được lấy mẫu ở tần số Fs = 75 Hz. Tìm tín hiệu lấy mẫu

d. Tìm tần số của (0<F<Fs) tín hiệu mà cho cùng một kết quả lấy mẫu như ở câu c.

Vẽ các dạng tín hiệu của các trường hợp lấy mẫu câu a,b,c. Nhận xét.

Bài tập 2: Cho tín hiệu tương tự

$$x_a(t) = 3\cos 2000\pi t + 5\sin 6000\pi t + 10\cos 12000\pi t$$

a. Tìm tần số Nyquist của tín hiệu

b. Giả sử tín hiệu lấy mẫu có tần số là Fs=5000 Hz. Vẽ tín hiệu thu được.