PHẦN THỨ HAI

CÁC THƯ VIỆN TRỢ GIÚP VÀ VẤN ĐỀ XỬ LÝ SỐ CHO CÁC TÍN HIỆU

CHƯƠNG 6

MATLAB TRONG XỬ LÝ SỐ CHO CÁC TÍN HIỆU

6.1 TÍN HIỆU VÀ XỬ LÝ TÍN HIỆU

Tín hiệu là một khái niệm rộng. Nó được định nghĩa theo Websur (1998) như là hình thức đếm được của một đại lượng vật lý hay xung (chẳng hạn hiệu điện thế, cường độ dòng điện hoặc từ thông) với tư cách là một bản tin hay thông tin có thể truyền đi được".

Ví dụ thông tin mong muốn có thể là nhiệt độ và tín hiệu là điện thế tỉ lệ với nhiệt độ này.

Tín hiệu thay đổi như một hàm của thời gian. Một số tín hiệu có dạng liên tục, chẳng hạn, nhiệt độ không khí, sóng biển tại một điểm... Một số tín hiệu khác lại ở dạng rời rạc, ví dụ như tin tức được truyền đi theo mã Morse.

Các tín hiệu có thể rời rạc hóa bằng cách lấy mẫu thông tin không liên tục, ví dụ như nhiệt độ của khí quyển và áp suất được truyền đi theo những khoảng thời gian nhất định.

Máy tính chỉ có thể xử lý tín hiệu rời rạc. Để xử lý bằng máy tính, hầu hết các tín hiệu đều được thể hiện theo một chuỗi tổ hợp các giá trị trạng thái I / O.

Tín hiệu được sinh ra bởi cảm biến (sensor), ví dụ nhiệt điện trở, tốc độ kế ... còn gọi là các bộ phát tín hiệu. Khi lấy mẫu tin tại các khoảng không đổi ta được chuỗi số một chiều thực còn chuỗi số hai chiều được sinh ra khi thực hiện việc số hoá các hình ảnh, ví dụ các chuỗi số trong các ma trận, thể hiện theo các đường khác nhau. Chúng có thể được lọc nhiễu,

modun hoá và được xử lý để làm rõ hình ảnh, hoặc được nén trong một khoảng động. Các thao tác xử lý được thực hiện ở thời gian hoặc tần số chủ đạo. Nếu chọn được tần số chủ đạo và thuật toán xử lý đúng, thì tăng được hiệu ứng tác dụng.

Matlab được ứng dụng để giải thuật toán nhanh và chọn tần số chủ đạo.

Ở đây ta sử dụng Toolbox xử lý tín hiệu (Signal Processing). Chúng gồm hơn 70 hàm số khác nhau để phân tích số cho tín hiệu và xây dựng bộ lọc với những đặc tính cho trước.

Chương 6 chỉ xét một số ví dụ sử dụng những hàm cơ bản thông dụng. Những hàm còn lại xin bạn đọc tham khảo trong các sách hướng dẫn sử dụng.

6.2 HÀM LỌC

Một trong những hàm số hay dùng để xử lý chuỗi là hàm ***filter*** (hàm lọc). Lọc số có ở mọi nơi trong quá trình xử lý tín hiệu. Trường hợp đơn giản khi một tín hiệu x có kèm theo nhiễu (noise), ta có thể loại bỏ chúng bằng cách sử dụng lọc.

Hãy xem xét trường hợp tín hiệu hiện thị trong một thiết bị đo do người tự đọc. Việc đọc sẽ rất khó khăn vì số đo thay đổi theo thời gian, khi có nhiễu của cảm biến. Nhưng điều này có thể được cải thiện nếu ta chọn hiển thị tại một khoảng nhất định, không phải giá trị thực x mà là một tổng các trị số trung bình cũng như giá trị cuối cùng hiện thị và một đầu vào mới, có nghĩa là

$$y_n = k_1.y_{n-1} + k_2.x_n$$

trong đó y_{n-1} là giá trị hiện thị cuối, và x_n là đầu vào mới.

Ví dụ: nếu ta chọn $k_1 = 0.9$ và $k_2 = 0.1$

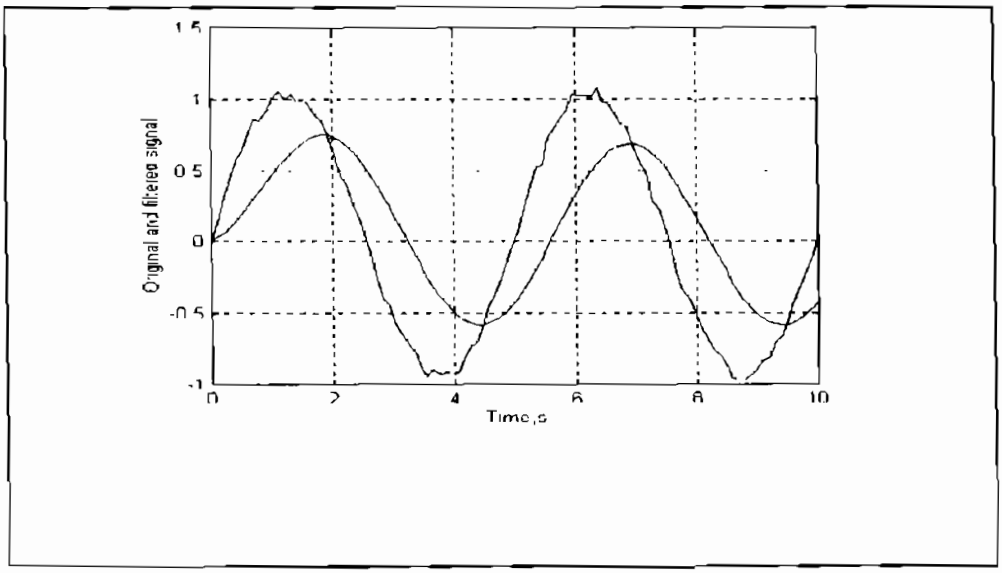
```
» t = linspace (0, 10, 100)                ; thời gian cơ bản
» s = sin (2*pi/5*t)                       ; tín hiệu
» [t,c] = size(t) ; n = 0.1 * rand(r, c)   ; nhiễu
» x = s + n                                ; đầu vào có nhiễu
» y(1) = x(1)                              ; điều kiện đầu
» for i = 2 : 100
» y(i) = 0.9* Y(i-1) + 0.1 * x(i)
```

$$y_n = 0.9.y_{n-1} + 0.1.x_n$$

Mối quan hệ này được xác định với y_n mà $n > 1$. Ta không quan tâm đến giá trị y_1 được xác định như thế nào (ta sẽ xem xét điều kiện sau) Chọn $y_1 = x_7 \rightarrow$ sử dụng Matlab

Bạn có thể so sánh x và y ***bằng lệnh chấm điểm plot***

Kết quả như hình vẽ 6.1.



Hình 6.1 ***Tuyến tính hóa tín hiệu nhiễu***

Ta có thể tính theo hai giá trị bước trước

$$y_n = 0.9.y_{n-1} + 0.05.x_n + 0.05.x_{n-1}$$

Trong trường hợp này đó là hai giá trị tự do y_1, và y_2. Khi đó y được sinh ra như sau:

```
» t = linspace (0, 10, 100)                    ; thời gian cơ bản
» s = sin (2*pi/5*t)                           ; tín hiệu
» [r,c] = size(t) ; n = 0.1 * rand(r, c)       ; nhiễu
» x = s + n                                    ; đầu vào có nhiễu
» y(1) = x(1)                                  ; điều kiện đầu
» for i = 3 : 100
```

```
» y(i) = 0.9* Y(i-1) + 0.05 * x(i) + 0.05*x(i-1);
» End
```

Cũng như trường hợp trước ta cũng làm

```
» plot(t, x, t, y)
```

Lọc số có thể được hiểu đơn giản bắt đầu từ lọc tương tự chẳng hạn lọc RC - (mạch lọc thông thấp).

Ví dụ. Cho mạch xoay chiều như sơ đồ với thông số như sau:

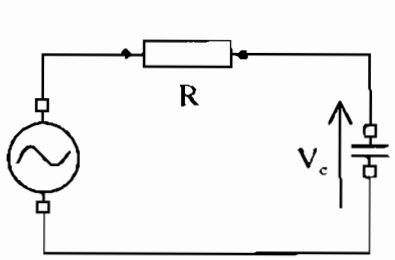
V, 10mV, 50 MHz, pha 0

R, 15Ω

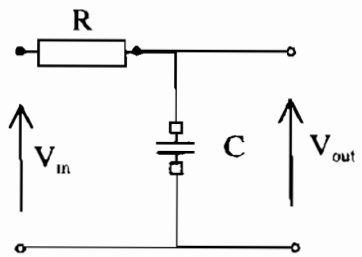
C, 100pF

Điện trở tương đương của tụ điện được tính như sau

$$Z_c = \frac{1}{j\omega\omega} = -j\frac{1}{\omega C}$$



a)



b)

Hãy vào các thông số của mạch

```
» V= 0.01;                    %V
» R= 15;                      % Om
» omega= 2*pi*50*10^6;        %rad/s
» C= 100*10^(-12);            %F
» Zc= -j/(omega*C);           %Om
```

Ta tính được tổng trở của mạch

```
» Z= R+ Zc

Z=

  16.0000 -31.8310i
```

Và dòng điện

```
» I = V/Z

I =

  1.2606e-04+ 2.5097e-04i
```

Điện áp rơi trên điện trở

```
» Vr = R*I

Vr=

  0.0018 + 0.0039i
```

Và qua tụ điện ta có được đồ thị biên pha . Sử dụng MATLAB ***subplot***

```
» subplot (1, 2, 1)
```

Các thông số 1, 2, 1 có ý nghĩa là 1 và 2 ma trận của hình vẽ sinh ra và chấm điểm theo một điểm đầu. Trong Matlab 3.5 các thông số này sẽ được viết là (121)

Chúng gồm các giá trị V, Vr, Vc như pha cuối cùng. Để định nghĩa pha, ta cần đưa ra điểm gốc của chúng

```
» VV= [0 V]; VVc = [0 Vc];  VVr = [0 Vr]
```

Tiếp đến ta chấm điểm điện thế pha

```
» plot (real(VV) , imag(VV)
```

Để quan sát được thang góc pha chính xác, biểu diễn trục thật cần tương xứng như đối với trục ảo. Có nghĩa là khung chấm điểm là hình vuông

```
» axis('square')
```

Khung của các trục sẽ được sinh ra như sau

```
» axis ([0  0.012  -0.006  0.006])
```

Phần mở rộng của cả hai trục là 0.012. Các giá trị này có thể được nhận bởi thử nghiệm và sai số, như được hiện thị ở đồ thị. Để đánh giá trục được tốt nhất ta dùng câu lệnh ***axis***. Hoặc ta có thể sử dụng một hàm khác là ***abs*** để tìm giá trị biên pha lớn nhất và điều chỉnh lại trục.

Bạn có thể dùng hàm ***hold*** để giữ cho đồ thị chồng lên hai hình khác,Vr và Vc

```
» hold on
» plot(real(VVr), imag(VVr))
» plot(real(VVc), imag(VVc))
```

Toàn bộ đồ thị sẽ được viết như sau:

```
» title('(a)')
» xlabel('Real'), ylabel('Imaginary')
```

Hoặc có thể xác định các hình bằng cách tự đánh dấu các điểm tức là dùng hàm ***gtext***

```
» text(real(V), imag(V), 'V')
» text(real(Vr), imag(Vr), 'Vr')
» text(real(Vc), imag(Vc), 'Vc')
```

Sau đó chấm điểm toàn bộ

```
» hold off
```

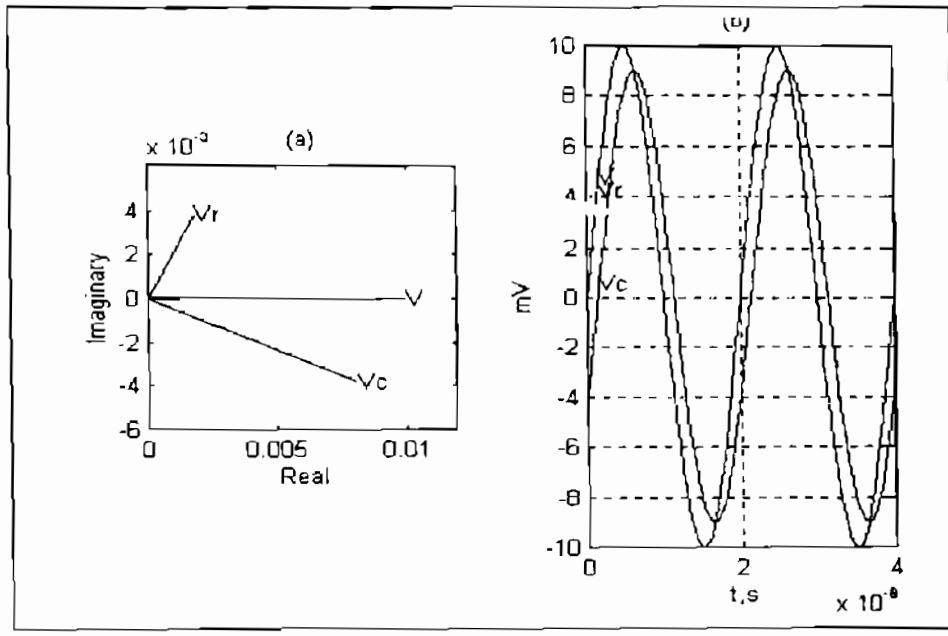
Tổng của điện áp được tính như sau (xem hình vẽ):

```
» Vr + Vc

ans =

  0.0100 - 0.0000i

» 180*(angle(Vr) - -angle(Vc))/pi

ans =

  90
```

Để có được đồ thị và tính toán điện áp ta làm theo những bước sau

```
» f= 50*10^6;                    % tần số, Hz
» T = 1/f;                       % chu kỳ, s
» omega= 2*pi*f;                 %tần số góc, rad/s
```

```
» t=0: T/50 :  2*T;                    % chuỗi các giá trị
» v = V*sin(omega*t);
» vr = abs(Vr)*sin(omega*t  + angle(Vr));
» vc = abs(Vc)*sin(omega*t + angle(Vc));
```



Hình 6.2 *Điện áp trên mạch nối tiếp RC*

Để có được điểm như phần tử thứ hai của ma trận đồ thị, hãy dùng lệnh

```
» subplot(1, 2, 2)
```

và chấm điểm theo những lệnh sau:

```
» plot (t, 1000*v, t, 1000*vr, t, 1000*vc)
» grid, title('(b)')
» xtabel('t,s'), ylabel('mV')
»text(t(5), 1000*v(5), 'V')
»text(t(20), 1000*v(20), 'Vr')
»text(t(50), 1000*v(5), 'Vc')
```

Như đã biết mạch RC là một ***mạch lọc thông thấp***. Giả sử cho đầu vào là điện áp 10mV, ta hãy xem đầu ra của tụ điện. Ký hiệu điện áp vào là V_{in} và điện áp ra V_{out} .

$$Z = R + Rc = R + \frac{1}{j\omega C}$$

$$I = \frac{V_{in}}{Z} = \frac{1}{R - j/\omega C} V_{in}$$

$$V_{out} = ZcI = \frac{1}{j\omega C} \frac{1}{R - j/\omega C} V_{in}$$

Hàm ***đáp ứng tần*** (dùng hàm biến đổi) của bộ lọc xác định bởi tỉ số

$$H(j\omega) = \frac{V_{out}}{V_{in}} = \frac{1}{1 + j\omega RC}$$

Giá trị $\omega_c = 1/RC$ gọi là ***tần số cắt*** ở dạng không thứ nguyên

$$H(j\omega) = \frac{1}{1 + j\omega / \omega_c}$$

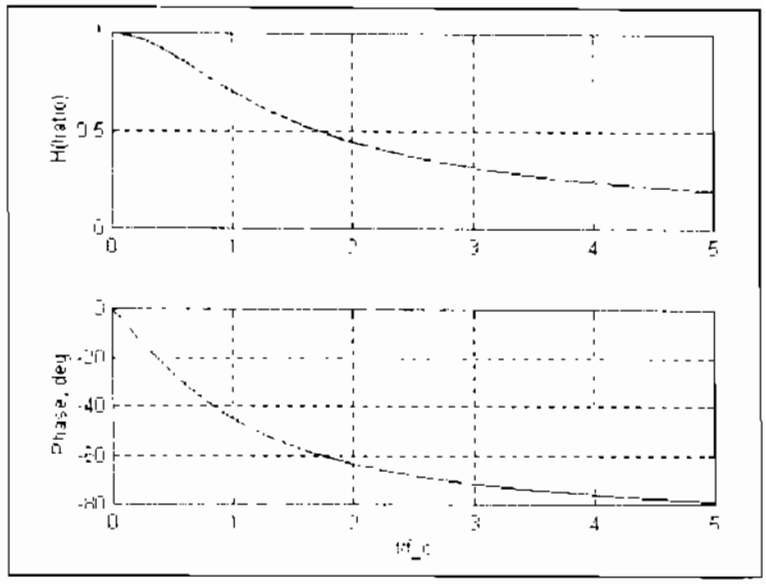
hoặc

$$\omega/\omega_c = 2\pi f/2\pi f_c = f/f_c$$

Ta nhận được dạng không thứ nguyên của hàm biến đổi

$$H(f) = \frac{1}{1 + jf/f_c}$$

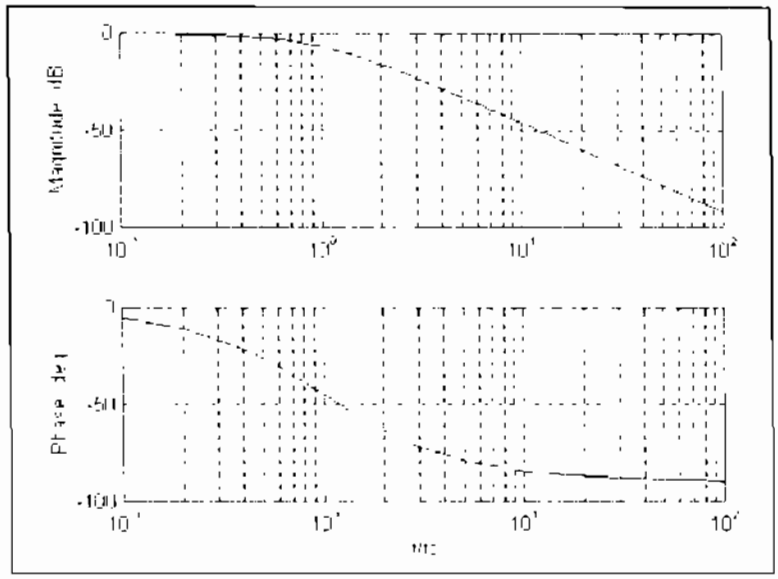
và vẽ biên và pha của hàm biến đổi theo hai đồ thị phụ thuộc tỉ số tần và dãy giá trị biến đổi



Hình 6.3 *Đồ thị biên và tần của mạch thong thap RC*

» *fratio= 0 : 0.01: 5;*

» *H = ones(size(fratio))/(1+j*fratio);*



Hình 6.4 *Mạch thong thap dự tính trước*

Vẽ đồ thị theo logarithm

Bây giờ ta xem xét bài toán theo quan điểm xử lý tín hiệu

$$\frac{L}{R}\frac{di}{dt} + i = \frac{v}{R}$$

Với $\tau = L/R$ - hằng số thời gian của mạch - $y = i$ và $x = v/R$

Đẳng thức trở thành

$$\tau\frac{dy}{dt} + y = x \qquad (6\text{-}1)$$

Giả thiết rằng y được lấy mẫu trong khoảng T_S nhỏ so với thời gian τ. Do đó có thể xấp xỉ

$$\frac{dy}{dt} = \frac{y_n - y_{n-1}}{T_S}$$

trong đó y_n - lấy được từ lần đo thứ n của giá trị y, và y_{n-1} của 1 bước trước. Đẳng thức (6-1) được viết lại như sau

$$\tau\frac{y_n - y_{n-1}}{T_S} + y_n = x_n \qquad (6\text{-}2)$$

Ta lại có $y_n = \frac{b_1}{a_1}x_n - \frac{a_2}{a_1}y_{n-1}$ (6-3)

Biểu thức này ta có được từ bộ lọc đầu tiên.

Viết cho trường hợp tổng quát

$$a_1y_n + a_2y_{n-1} + \ldots + a_{n_a}y_{n-n_a+1} = b_1x_n + b_2x_{n-1} + \ldots + b_{n_b}x_{n-n_b} \qquad (6\text{-}4)$$

hoặc

$$y_n = \frac{b_1}{a_1}x_n + \frac{b_2}{a_2}x_{n-1} - \ldots + \frac{b_{n_b}}{a_1}x_{n-n_b+1} - \frac{a_2}{a_1}y_{n-1} - \ldots - \frac{a_{n_a}}{a_1}y_{n-n_a+1} \qquad (6\text{-}5)$$

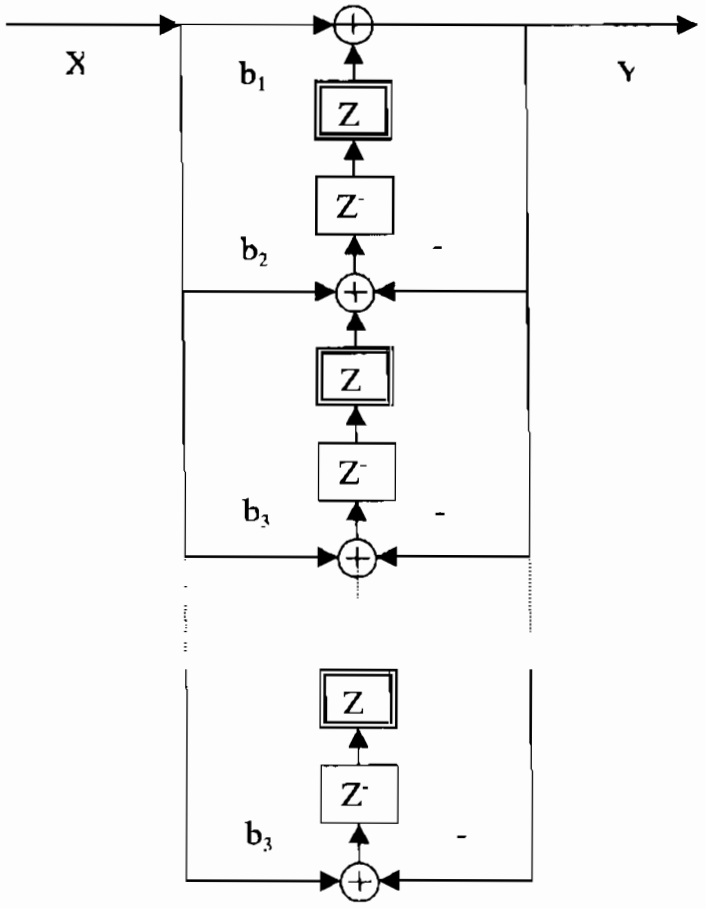
Sử dụng Matlab với bộ lọc N tầng, $N = \max(n_a, n_b) - 1$

Ta có hai chuỗi

$A = [a_1\ a_2\ , ..., a_{na}]$

$B = [b_1, b_2, ..., b_{nb}]$

Trên hình 6.5, a_i và b_i là các hệ số được làm bình thường hoá bằng cách chia cho a_1 hệ số của y_n . Qua đó thể hiện ký hiệu thao tác dịch thời gian (*unitary time shift operator*). Cho x - rời rạc của tín hiệu f(t) bị lấy mẫu theo chu kỳ T_S, do đó $x(n) = f(n.T_S)$ và $z_x(k)$. Xác định thành phần thứ k $z_x(k) = x(k+1)$. Tương tự như vậy thành phần thứ n của z^{-1} là x(k-1) z - ký hiệu thích hợp cho việc thao tác số nhanh (xin đọc thêm về biến đổi z trong các giáo trình về xử lý tín hiệu).



Hình 6.5. ***Chuyển đổi trực tiếp ở dạng II***

Việc tái hiện các bước trên hình 6-5 gọi là dạng chuyển đổi II và là mô hình tối thiểu cho việc cất giữ (tức là tiết kiệm bộ nhớ nhất). Trên thực tế, tại mỗi bước này chúng ta cần cất mỗi giá trị của trạng thái

z_1, z_2,..., z_n được thể hiện trên đồ họa bằng khung đúp ở đường trung tâm (xem hình 6-5)

Bộ lọc được thực hiện như sau:

(1) Thực hiện

(a) Vector trạng thái đầu, thường được đánh dấu bằng Z_i và được đặt vào bộ trễ Z_j không xuất hiện, tất cả các giá trị Z_j sẽ đặt = 0

(b) Giá trị của A và B được chuẩn hoá bằng cách chia cho a1

(2) Vòng lặp chính. Tại các bước này, chức năng lọc thể hiện theo những thao tác sau: (a) Sinh ra y_n từ quan hệ $y_n = b_1x_n + Z_1$

(b) Đặt vào Z_j với j từ 1 đến N-1 trong

$$Z_j = b_jx_n + Z_{j+1} - a_jy_n$$

còn đối với j = N ta tính như sau:

$$Z_N = b_Nx_n - a_Ny_n$$

Vòng lặp được tiếp tục cho đến khi chuỗi x hội tụ

(3) Bước cuối cùng. Giá trị Z_j được sao chép đến vector ra Z_f

Đẳng thức (6-5) có thể được thực hiện bởi MATLAB bằng chức năng lọc. Ở đây có 4 cách gọi hàm lọc:

y = filter (B, A, x)

y = filter (B, A, x, Z_i)

[y,Z_f] = filter (B, A, x)

[y,Z_f] = filter (B, A, x, Z_i)

Trong đó Z_i và Z_f là một chuỗi các thông số tối ưu, liên hệ với trạng thái đầu và trạng thái cuối, đã được mô tả ở trên. Hàm lọc có thể trả về giá trị cuối cùng của Z như dãy Z_f, nếu nó gọi 2 biến số bên vế trái:

»[y, Z_f] = filter (B, A, x)

Ví dụ: cho

» x = [1, 2, 1, 2, 1, 2 , 1, 2, 1, 2];

» B = [0.5];

» A = [1, -0.25, -0.25];

Các tần số được tính toán như sau

Bước	Vào	Ra	Z_1	Z_2
1	1	0.5	0.125	0.1250
2	2	1.125	0.4063	0.2813
3	1	0.9063	0.5078	0.2266
4	2	1.508	0.6035	0.3770
5	1	1.104	0.6528	0.2759
6	2	1.653	0.6891	0.4132
7	1	1.189	0.7105	0.2973
8	2	1.71	0.7249	0.4276
9	1	1.225	0.7338	0.3062
10	2	1.734	0.7397	0.4335

Trong bảng tạo bởi Matlab M-file, số được làm tròn 4 số sau dấu phẩy.

```
» y(:)
  ans =
0.5000
1.1250
0.9063
1.5078
1.1035
1.6258
1.1891
1.7338
```

Ta có thể tính lại số này theo mô tả ở hình 6.5. Cuối cùng, sau khi xác định chuỗi số x, A và B liên hệ trong hàm lọc:

```
» [Y,Zf] = filter (B, A, x, Zi) ;
```

Ta nhận được chuỗi y được tính toán bởi tệp M, và giá trị lớn nhất của

Z_1 và Z_2 trên bảng.

» Z_f =

Z_f =

0,7397

0,4335

6.3 GỌI HÀM LỌC VỚI ĐIỀU KIỆN ĐẦU

Nếu biết các điều kiện đầu để cất vào khâu trễ, bạn có thể gọi các bộ lọc.

Chúng thường được dùng nếu ta có một tín hiệu x dài không thể cất gửi vào bộ nhớ của máy tính. Bạn có thể chặt x ra thành điểm gọi là x_1, x_2, ... , x_n như trong MATLAB, và bộ lọc sẽ phân biệt được chúng một cách chính xác. Giá trị cuối cùng, Z_f của các khâu trễ tại mỗi bước được sử dụng như là các giá trị đầu, Z_i cho bước tiếp theo. Để hiểu rõ, ta dựng một chuỗi với 100 số ngẫu nhiên.

»x = [x1 ; x2 ; ... ; xn]

» x = rand(100,1) ;

Đem chặt chúng thành nhiều chuỗi con

» x1 = x(1 : 25) ;

» x2 = x(26 : 50) ;

» x3 = x(51 : 75) ;

» x4 = x(76 : 100)

Và bây giờ ta xây dựng bộ lọc liên hệ với các thành phần của các chuỗi con. Khi xây dựng bộ lọc, ta sử dụng các giá trị của trạng thái cuối cùng, Zf, sinh ra bởi lần gọi đầu tiên, như vậy một trạng thái đầu Zi, sẽ cho lần gọi thứ 2, cứ như vậy bạn có thể sử dụng toàn thể chuỗi số x

» y = filter(b, a, x) ;

và thay đổi kết quả được sinh ra bởi "partial"

» max(max(abs(y-[y1 ; y2 ; y3 ; y4]))) ;

ans = 0

Trong nhiều trường hợp, rõ ràng vector Z_i là phần chọn và bạn có thể

làm ra. Khi đó, hàm lọc sử dụng vector không chuẩn với chiều dài N là 0 (vector 0). Nếu Z xuất hiện, nó cần phải dài N, bằng với bộ lọc đã được định trước, và bằng 1 + N như ở phần trên Matlab.

Để xác định bộ lọc của N, ta cần cung cấp N trạng thái như là giá trị của vector Zi. Ta sẽ có N trạng thái ban đầu của đầu vào, viết cho vectơ y (có chiều dài bằng N và có giá trị ban đầu của y đồng thời với y_1). Nếu viết N đẳng thức đầu của thuật toán lọc, bạn nhận được mối quan hệ sau:

$$y_1 = Z_{i1} + b_1 + b_1 x_1 \qquad (6\text{-}6)$$

$$y_2 = Z_{i2} + [b_1 x_2 + b_2 x_1] - [a_2 - y_1] \qquad (6\text{-}7)$$

$$y_3 = Z_{i3} + [b_1 x_3 + b_2 x_2 + b_3 x_1] - [a_2 - y_2 + a_1 y_3] \qquad (6\text{-}8)$$

.
.
.

$$y_k = Z_{ik} \sum_{i=1}^{k} b_i x_{k+1-i} - \sum_{i=1}^{k-1} a_{i+1} y_{k-i} \qquad (6\text{-}9)$$

.
.
.

$$y_N = Z_{iN} \sum_{i=1}^{N} b_i x_{N+1-i} - \sum_{i=1}^{N-1} a_{i+1} y_{N-i} \qquad (6.10)$$

Để N giá trị đầu của bộ lọc phù hợp với y_i, ta có thể trừ y_i với y trong N đẳng thức, và tìm ra Z_i. Hàm lọc được viết để giải quyết vấn đề này. Hãy làm những bước chuẩn bị sau để gọi tệp ***filteric.m.***

Ví dụ : Cho X hiển thị 1000 điểm đầu của tín hiệu được lấy mẫu tại tần số chủ đạo 100 Hz. Cho qua tín hiệu từ 30 Hz (lọc thấp) của điểm 5, và nhận được tín hiệu y như y(i) = x(i) và i =1 đến 5

Chú ý: *Trong ví dụ này dùng tín hiệu giả.*

Đầu tiên ta xây dựng tín hiệu với

» x = rand(1000 , 1) ;

Tiếp theo xác định thông số của bộ lọc như ví dụ ở phần tổng quan của toolbox xử lý số cho tín hiệu.

» yi = x(1 : 5) ;

Các trạng thái đầu được xác định như sau

» [b, a] = butter (5 , 30/50)

```
b =
  0.1084 0.5419 0.0837 1.0837 0.5149 0.1084
a =
  1.000        0.9853 0.9738 0.3864 0.1112 0.0113
```

Và trạng thái đầu của khâu trễ

```
» Zi = filteric (b, a, x, yi) ;
```

Liên hệ với hàm lọc

```
» y = filter(b, a, x, Zi) ;
```

Ta có thể hiển thị 5 nhóm đầu của chuỗi vào, ra và thay đổi cho chúng bằng nhau

```
» [ X(1:5), Y(1:5) ]
ans=
  0.2190    0.2190
  0.0970    0.0470
  0.6789    0.6789
  0.6793    0.6793
  0.9347    0.9347
```

6.4 THIẾT KẾ CÁC BỘ LỌC SỐ

6.4.1. Các định nghĩa

Một trong những vấn đề thường gặp trong xử lý số tín hiệu là cấu trúc của bộ lọc với các đặc tính biên tại các tần số khác nhau. Một trong số công cụ trong toolbox xử lý tín hiệu là 2 hàm ***yulewalk*** và ***remez***.

Ta gọi chúng với bộ lọc số H của N điểm, đặt tần số lấy mẫu số liệu x, sinh ra tần số mới y, quan hệ với x theo đẳng thức.

$$a_1 y_n + a_2 y_{n-1} + \ldots a_{N+1}\, y_{n-N} =$$

$$\mathbf{b_1 x_n + b_2 x_{n-1} + \ldots b_{N+1}\, x_{n-N}} \qquad (6\text{-}11)$$

Các hệ số $B = [b_1, b_2, \ldots, b_{N+1}]$ và $A = [a_1, a_2, \ldots, a_{N+1}]$ đều xác định và $\neq 0$. Nhưng ta có thể giả thiết chúng chuẩn theo a_1. Hơn nữa, tại các hệ số

cuối cùng a_{N+1} hoặc b_{N+1} có thể khác 0. Trong các trường hợp khác bộ lọc cần xác định vector thu gọn A và B, và chúng cần nhỏ hơn N. Hàm trong toolbox Matlab sinh ra các hệ số của bộ lọc (***yulewalk, cheb1*** và các hàm khác ...) đó là các hệ số qui chuẩn, thành phần của hàm lọc (*filter*).

Khi sử dụng thao tác dịch thời gian, bộ lọc H được biểu diễn bằng hàm phân thức sau:

$$H(z) = \frac{B(x)}{H(z)} = \frac{b_1 + b_2 Z^{-1} + b_N Z^{-(N-1)}}{a_1 + a_2 Z^{-1} + \ldots + a_N Z^{-(N-1)}} \qquad (6\text{-}12)$$

Với $a_1 = 1$ và hệ số lớn nhất a_N và $b_N \neq 0$

Trong tất cả các version của MATLAB khi ta dùng ***help yulewalk*** không đúng, chúng sẽ hiện ra đoạn văn bản như sau

YULEWALK RECUSIVE FILTER DESIGN USING A LEAST-SQUARES METHOD.

[B,A] = yulewalk(N,F,M) finds the N-th order recursive filter coefficients B and A such that the filter:

$$\frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + \ldots + b(n)z^{-(n-1)}}{1 + a(1)z^{-1} + \ldots + a(n)z^{-(n-1)}}$$

Trong đó chỉ số của A bị sai dịch n = N + 1, ví dụ, bộ lọc được xác định bởi các vector:

$$A = [a_1, a_2, a_3, a_4, a_5]$$

$$B = [b_1, b_2, b_3, b_4, b_5]$$

Với $a_1 = 1$ và hệ số cuối b5 hoặc a5 $\neq 0$

Dạng như đã nói ở trên bị giới hạn vì lỗi trong cách sử dụng tương ứng của hàm số với bộ lọc.

Nếu các nhóm $a_2, a_3, \ldots, a_N$ đều bằng 0 thì bộ lọc sẽ gọi FIR (bộ lọc đáp ứng xung hữu hạn). ***yulewalk*** được dùng để tổng hợp bộ lọc IIR, khi hàm ***remez*** được sử dụng cho FIR.

6.4.2 Xác định đặc tính tần của bộ lọc

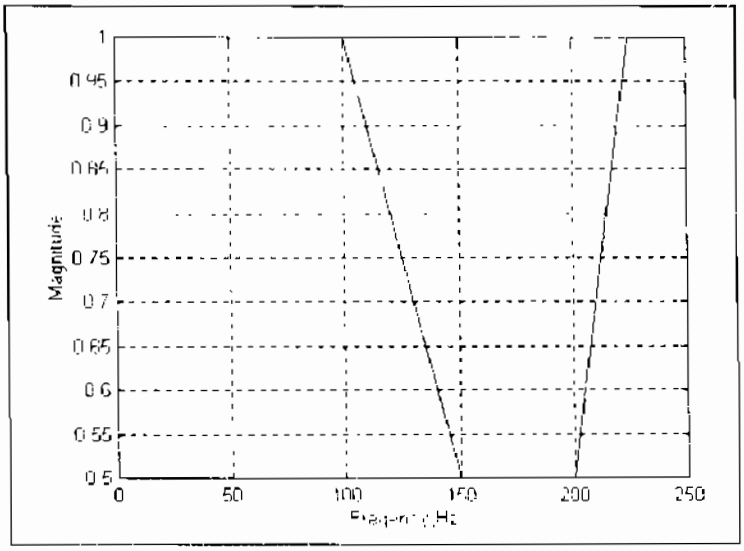
Matlab cho phép ta định nghĩa số của tần số fr_1 , fr_2 , ..., fr_i ,..., fr_k và biến tương ứng mag_1, mag_2, ... mag_i,..., mag_k cũng như mô tả bộ lọc số gần đúng (xấp xỉ) với đáp ứng của bộ lọc tương tự.

Tần số đáp ứng của bộ lọc số phụ thuộc vào tần suất lấy mẫu. Một nửa tần số đáp ứng của quá trình lấy mẫu được gọi là ***Nyquist***. Một số hàm của Matlab được nhanh chóng đẩy vào vùng tần số không thứ nguyên, có nghĩa là bằng việc định nghĩa tần số không thứ nguyên 50/(1000/2) = 0.1 và 150 Hz sẽ tương ứng với 150/(1000/2) = 0.3.

Để xác định đặc tính của bộ lọc, cần có 2 chuỗi : một là tần số không thứ nguyên, f=[f_1, f_2 ,..., f_k] và một tương ứng với biên m = [m_1, m_2, ..., m_k] Matlab đòi hỏi f_1 = 0 và f_k = 1.

Như trong ví dụ, giả thiết rằng tín hiệu x được lấy mẫu ở 500Hz ta sẽ xây dựng bộ lọc với tần số biên như sau:

Từ	Đến	Biên
0	100	1.0
100	150	giảm đến từ 1 đến 0.5
150	200	luôn là hằng 0.5
200	225	tang đều từ 0.5 đến 1
200	250	



Hình 6.6 ***Đặc tính tần của bộ lọc***

Đặc tính bộ lọc được mô tả vào MATLAB bởi

```
» fHz0 = [0 1000 150 200 225 250];
» m0 = [1.0 10 0.5 0.1 1.0 1.0];
```

Để kiểm tra ta có thể vẽ đồ thị như hình vẽ 6.7

```
» plot (fHz0 , m0) ;
```

MATLAB xây dựng bộ lọc số IIR và FIR với những đặc tính nhất định. Trong phần tổng quan về xử lý tín hiệu ta đã xem xét những vấn đề ưu nhược điểm, trong đó đã đề cập đến hàm ***yulewalk*** cho IIR và ***remez*** cho tổng hợp FIR. Để sử dụng hàm ***yulewalk***, cần qui các tần số thành không thứ nguyên

```
» fs = 500;
» f0 = fHz0 / (fs/2) ;
```

Trong đó fs là tần số lấy mẫu được xấp xỉ đúng nhất, bằng phương pháp trung bình bình phương nhỏ nhất. Hãy thử bộ lọc 6 điểm:

```
» [bIIR, aIIR] = yulewalk (6, f0, m0) ;
```

Ta có thể kiểm tra lại việc xấp xỉ bằng cách so sánh đáp ứng của bộ lọc được xác định bởi [bIIR, aIIR] với đáp ứng đã có được. Trong đó đáp ứng của bộ lọc H(z) tại tần số ω rad/s được cho bởi giá trị H(z) cho $Z = e^{i.\omega/fs}$, với fs là tần số lấy mẫu, và giả thiết cần 5 điểm đặt trước trên trục x, trong 50 điểm từ 0 đến tần số ***Nyquist***, tính theo rad/s. Với Matlab ta có được chúng theo tần số Hz :

```
» fHz1 = linspace (0, 250, 50) ;
```

và chuyển chúng thành rad/s bằng

```
» om1 = 2 * pi * fHz1 ;
```

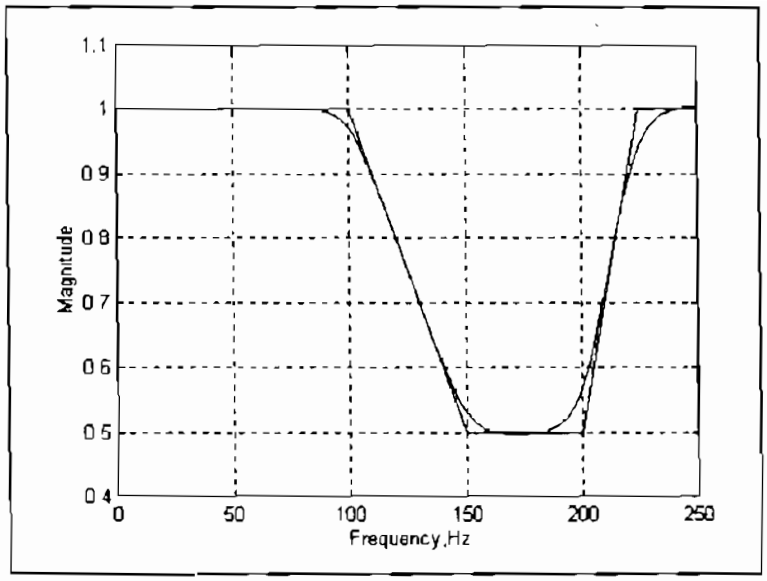
Muốn tính đáp ứng biên của bộ lọc, dùng lệnh sau

```
» z = exp (sqrt (-1) * om1 / fs) ;
» mIIR = abs (polyval (bIIR, z). / polyval (aIIR, z)) ;
```

Bây giờ ta có thể so sánh sự trùng của đáp ứng cho trước và đáp ứng thực.

```
» plot (fHz0, m0, fHz1, mIIR) ;
```

Những kết quả này được mô tả trong hình 6.4. Nếu như sự xấp xỉ không tốt như giả định thì cần tăng số điểm cho trước của bộ lọc.



Hình 6.7 *Bộ lọc IIR định nghĩa và bộ lọc thực*

Bây giờ ta hãy xem xét vấn đề sử dụng FIR. Bộ lọc FIR có thể có số điểm cho trước lớn hơn để đạt được việc so sánh, chẳng hạn dùng bộ lọc với số điểm là 20:

```
» bFIR = remez (20, f0, m0) ;
```

Hàm ***remez*** cho ta chuỗi b, tất cả bộ lọc FIR a = [1].

Bạn có thể kiểm tra lại kết quả bằng hình vẽ.

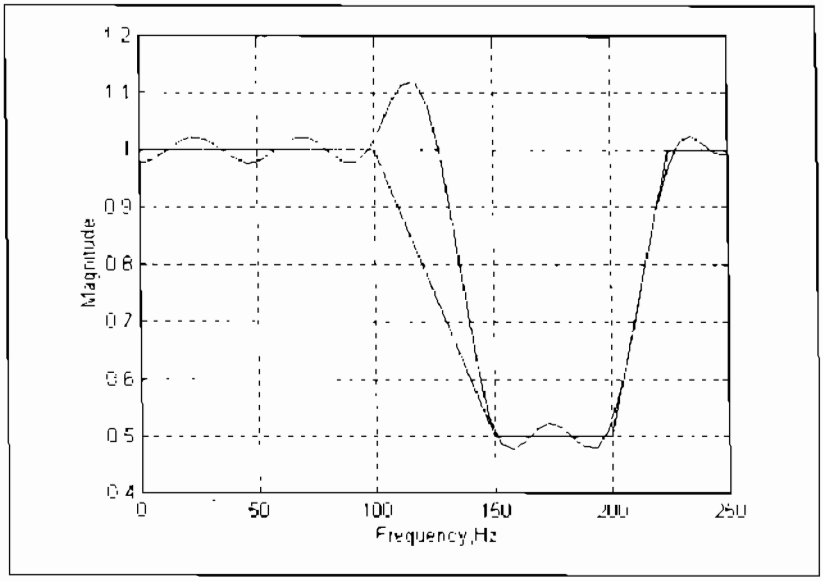
```
» mFIR = abs (polyval (bFIR, z)) ;
» plot (fHz0, m0, fHz1, mFIR) ;
```

Kết quả đồ thị như hình 6-7.

Trong toolbox xử lý số cho các tín hiệu có bổ xung thêm một số hàm để tổng hợp bộ lọc IIR: ***cheby1, cheby2, ellipt***, hàm số ***yulewalk*** đòi hỏi hai chuỗi số: 1 - tần số, 2- là đáp ứng biên. Ở đây cần đưa thêm yếu tố xác định kiểu lọc: thông cao, thông thấp, thông giữa và các trạng thái của biến đổi nhỏ (ripple). Trong trường hợp tổng hợp FIR cũng tương tự IIR dùng ***remez*** hoặc ***fir1 và fir2***.

Để tính toán đáp ứng tần số của bộ lọc số, MATLAB dùng hàm tần số ***freqz***, nhanh hơn là dùng thẳng tính toán của ***H (squtt (-1) * om / fz)***.

Để hiểu thêm quan hệ của phương pháp này, xin đọc thêm sách hướng dẫn sử dụng.



Hình 6.8 *Xác định và thực hiện của hàm lọc FIR*

Ví dụ: Tách hai sóng hình sin từ tổng của chúng.

Bộ lọc này dùng tần số để phân biệt hoặc tách thành phần cosin từ tín hiệu tổng hợp. Như trong ví dụ, ta xây dựng tín hiệu đơn từ hai sóng hình sin, một với tần số 100Hz, một là 400Hz, trong khoảng thời gian 0,1 giây. Tần số lấy mẫu là 2000 Hz.

» f_S = 2000 ;

» t = 0: (1/f_S) : 0.1 ;

» x_1 = sin(2 + pi * 100 x t) ;

» x_2 = sin(2 * pi * 400 * t) ;

» x = x_1 + x_2 ;

Tín hiệu x xuất hiện trên đường 1 ở hình 6.9. Sau đây ta sử dụng ***yulewalk*** để mô tả bộ lọc thông thấp và thông cao. Tần số cơ bản được xác định bởi

» fH20 = [0 225 275 1000] ;

Biên đặc biệt của bộ lọc thông thấp là

» m10 = [11 00] ;

Và bộ lọc thông cao là

» *mh0 = [00 11] ;*

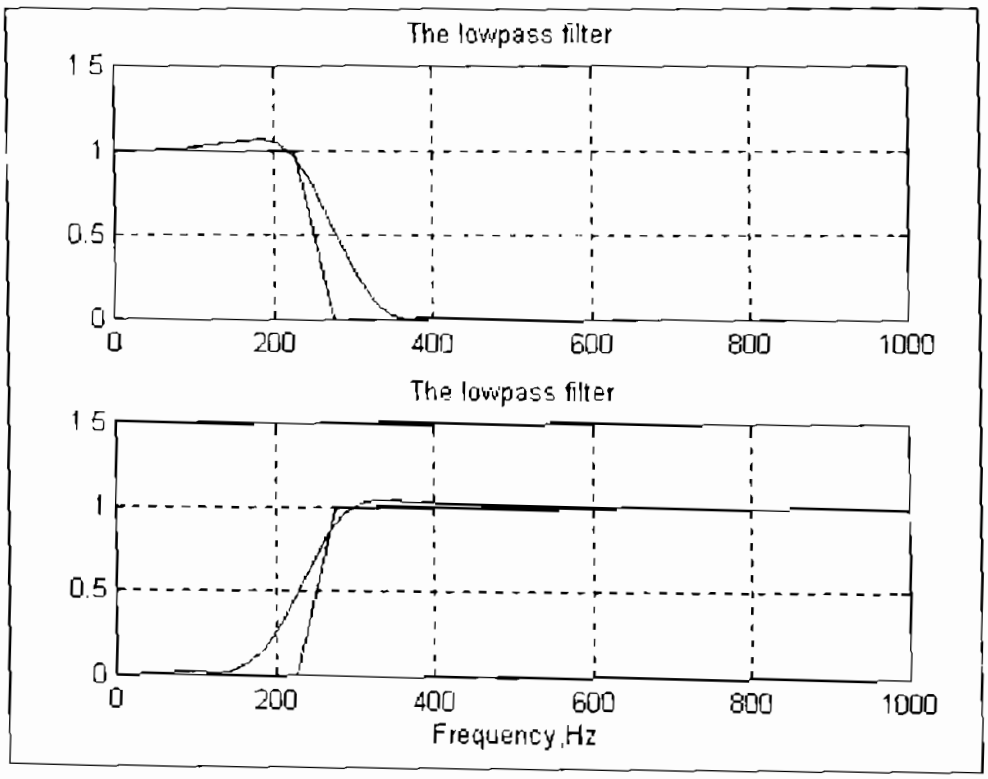
Tần số mô tả không thứ nguyên là

» *f = fH20/(f_s/2) ;*

Các thông số của bộ lọc thông thấp được tính bởi

» *[b1 , a1] = yulewalk (6,f0, mh0) ;*

Để kiểm tra lại chất lượng của bộ lọc ta tính và chấm điểm ở các tần số của chúng với lệnh sau:



Hình 6.9 *Đặc tính tần của bộ lọc*

» *fHz1 = linspace (0, f_s/2 , 50) ;*

» *om1 = 2 * pi *H21 ;*

» *Z = exp (sqrt(-1) * 0m1/f_s) ;*

» *m1 = abs(polyval(b1,z) . /polyval(a1,z)) ;*

» *mh = abs(polyval(bh, z)./polyval (ah,z)) ;*

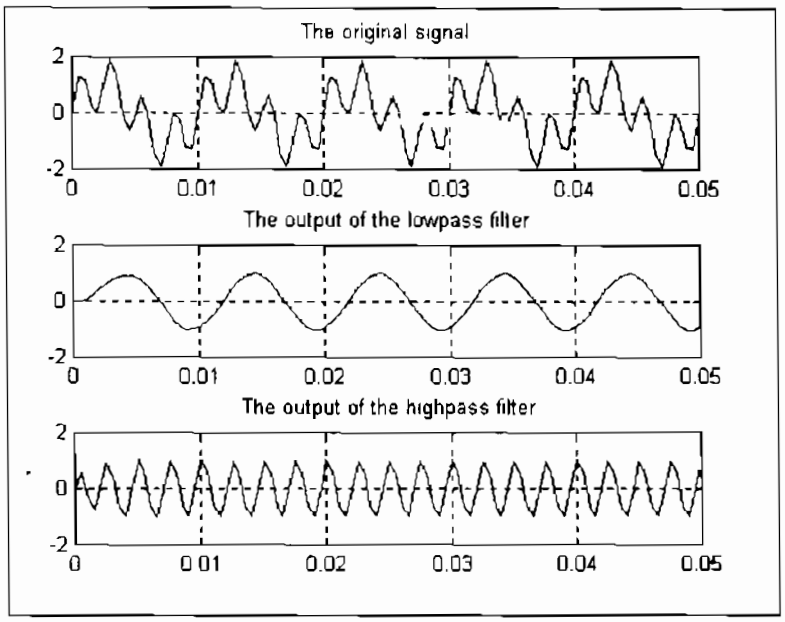
Có thể so sánh đặc tính của bộ lọc thông cao với đặc điểm sau:

```
» plot (fH20, mh0 ,fH21, ml) ;
```

Đặc tính tần của hai bộ lọc thể hiện trên hình 6.8. Sai lệch với kết quả không xa. Ta lọc được tín hiệu

```
» y1 = filter (bl, al, x) ;
```

Và chấm điểm y2 , có thể nhìn thấy thành phần 100Hz.



Hình 6.10 ***Tín hiệu gốc của x và đầu ra của hai bộ lọc y1 và y2 trong thời gian 0,05 giây.***

6.4.3 Biến đổi nửa tuyến tính Tustin

Thông thường ta có hàm biến đổi H(s) của bộ lọc tuyến tính xác định tần số chủ đạo, và muốn xấp xỉ nó với bộ lọc số $H_d(z)$. Để chuyển bộ lọc tương tự thành bộ lọc số "tương đương", thì cần đọc thêm tài liệu về bộ lọc ***Franlin Powell và Workman.***

Một trong những khả năng để đạt được s = s(z) trong các biến số z, là thực hiện xấp xỉ gần nhất .

$$H_d(z) = H(s(z)) ; \qquad (6.13)$$

Biến đổi

$$s(z) = \frac{2}{Ts}\frac{z-1}{z+1} \qquad (6.14)$$

Có gốc là biến đổi Tustin" và MATLAB dùng hàm ***bilinear***

Trong biểu thức (6-14), Ts là đoạn lấy mẫu, z là thao tác dịch thời gian

Biến đổi này gần với luật biến đổi ***tranpezoidal integration*** và chúng có tác dụng ở trong khoảng lấy xấp xỉ ***trapezoidal.*** Trong một số trường hợp nếu hàm đủ bằng phẳng trong một không gian lấy mẫu đủ ngắn, biến đổi **Tustin** sẽ cho ta phép biến đổi đại số chuyển bộ lọc tương tự về bộ lọc số.

Muốn hiểu rõ về phần này bạn nên đọc thêm lý thuyết xử lý tín hiệu.

Như trong ví dụ, ta có thể mô tả bộ lọc số với đặc tính tương tự (2 tầng), lọc thông thấp và hàm biến đổi.

$$H(s) = \frac{\omega_n^2}{S^2 + 2\xi\omega_n s + \omega_n^2} \qquad (6\text{-}15)$$

Trong đó ω_n = 30rad/s , ξ= 0,6 và ký hiệu lấy mẫu tại 100Hz. Để giải quyết vấn đề này trên MATLAB cần định nghĩa tần số lấy mẫu

```
» f1 = 100 ;
```

Các thông số của bộ lọc số là

```
» ωn = 3.0 ; z = 0.6 ;
```

Còn

```
» num = [ωn ^2] ;
```

là số lần đặt tên, để có công suất của s là

```
» den = [1 2 * z * ωn  ωn ^2] ;
```

Bạn có thể nhận được bộ lọc số tương đương

```
» [a,b] = bilinear (num, den , fs) ;
```

Để so sánh bộ lọc số, định trước [a,b], ta vẽ 2 đường đồ thị quan hệ theo tần số (đơn vị rad/s)

```
» om = linspace(0,300) ;
```

Tiếp theo ta tính tần số đáp ứng này của bộ lọc số bởi

```
» z = exp(s/fs) ;
```

» hz = polyval(b,z) ./ polyval (a,z) ;

Và so sánh biến của 2 đáp ứng này bằng

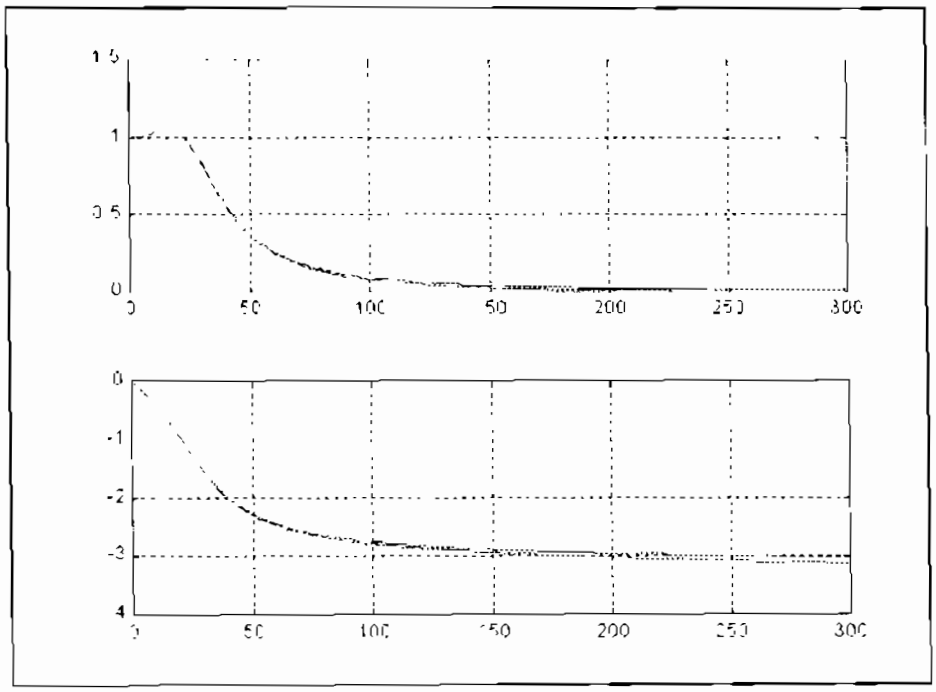
» subplot (2, 1, 1) ;

» plot (om, abs(hs), om, abc(hx))

» subplot (2 ,1, 2) ;

» plot (om, angle(hs), om, angle(hs)) ;

Kết quả thu được xem trên hình 6.11.



Hình 6.11 *Xấp xỉ giữa bộ lọc tương tự và bộ lọc IIR*

6.5 BIẾN ĐỔI FOURIER RỜI RẠC (DFT)

Tín hiệu f(t) của một biến liên tục t có chu kỳ lặp lại T nếu f(1 + T) = f(t). Nếu giá trị chu kỳ nhỏ nhất dương, thì chu kỳ đó gọi là chu kỳ cơ bản của f.

Tương tự tín hiệu số d = {..., d(-1), d(0), d(1), d(2), ...} gọi là có chu kỳ P, nếu với mỗi số nguyên dương k : d(k)= d(k+P).

Nếu ta lấy mẫu hàm số chu kỳ f của chu kỳ thực T tại các thời điểm lấy mẫu T_S, và T_S là ước số của T, ta gọi $T_S = T/N$, lần lấy mẫu d (được lấy mẫu theo vesion d) của f là một tín hiệu số lặp với chù kỳ N. Gọi là $d(k) = f(k.T_S)$

$$\begin{aligned} d(k+N) &= f((k+N) \,.\, T_S) \\ &= f((k+N) \,.\, T/N) \\ &= f(k \,.\, T/N + T) \\ &= f(k \,.\, T/N) \\ &= f(k \,.\, T_S) \\ &= d(k) \end{aligned}$$

Sự lặp lại của tín hiệu theo chu kỳ N là phân bố đều bởi số N, ví dụ đối với mỗi lần kể từ 1 đến N → trong MATLAB dùng vector x với chiều dài N. Vector x gọi là biểu diễn chủ đạo của tín hiệu d và được xác định một cách đơn giản bởi chuỗi của thành phần như sau x(h) = d(h) với h = 1 đến N.

Từ thời gian biểu diễn chủ đạo x của d, sử dụng chu kỳ lặp của tín hiệu dễ dàng xây dựng được quan hệ : d(h) = x(k), đối với k với các điều kiện sau

1. $1 \le k \le N$

2. (h - k) chia hết cho N.

Mỗi đầu vào x(k) của tín hiệu x có thể là một số thực hoặc phức. Ta có thể thấy thời điểm ghi mẫu là đặc tính phân biệt tín hiệu thực với những tín hiệu khác lấy từ việc tính toán hoặc từ quan điểm lý thuyết. Nói chung thì f(k) là số phức. Với giả định rằng tín hiệu thực trên thực tế phần biến đều bằng 0. Biến đổi Fourier rời rạc, DFT, của một chuỗi số x độ dài N là một chuỗi khác X, cũng có độ dài N. Biến đổi Fourier gọi là biến đổi ngược IFT (Inverse Fourier Transform). Ta sẽ còn quay lại hai khái niệm này ở phần sau.

IFT và DFT dùng trong MATLAB rất có giá trị vì chúng sử dụng thuật toán FFT (biến đổi Fourier nhanh). FFT là một thuật toán rất phổ biến (Coolly và Tukey 1965). Thuật toán này cần xấp xỉ Nlog(N) các phép toán để tính DFT, so với N^2 phép toán cho phép bình thường.

Biến đổi Fourier có hai tác dụng chính. Thứ nhất, nhiều thao tác trên tín hiệu nhanh hơn khi dùng trên tần số chính. Có khoảng 15 hàm số trên Toobox xử lý tín hiệu độc lập tác dụng lên các ứng dụng các hàm trực tiếp trên thời gian chủ đạo, theo cách chuyển vector (gốc) thành tần số chính, ứng dụng hàm xấp xỉ và chuyển kết quả ngược trở lại thời gian chủ đạo.

Tác dụng thứ hai là để nhận dạng các thành phần tần số của tín hiệu, như ta sẽ thấy ở mục sau.

6.6 GIỚI THIỆU TÓM TẮT BIẾN ĐỔI FOURIER RỜI RẠC (DFT)

Trong C^N (hay còn được ký hiệu R^N không gian N chiều), ví dụ chúng ta có các điểm cơ bản, ký hiệu bởi $i_1, i_2, ...,i_N$ và được xác định bởi:

$i_1 = (1, 0, 0, ..., 0)$

$i_2 = (0, 1, 0,..., 0)$

$i_3 = (0, 0, 1, ..., 0)$

.

.

.

$i_N = (0, 0, 0, ..., 1)$

C^N, hơn nữa có chấm điểm (dot product), ký hiệu bởi (!) và xác định như sau:

Nếu

$x = [x_1, x_2,..., x_N]$

$y = [y_1, y_2,..., y_N]$

Các điểm sinh ra là:

$$<x/y> = \sum_{h=1}^{N} x_h \bar{y}_h$$

Trong đó $\bar{y}_h$ là giá trị trung bình, liên hợp phức y_h là phần bù số ảo khi xác định thông thường của điểm trong R^N.

Gọi ảnh của vector x lên vector y khác không là vector

$$x_y = <x/y> = \frac{y}{< y/y >}$$

Vector này còn gọi là thành phần của x lên hướng của y.

Phần cơ bản của R^N (hoặc C^N) gọi là trực giao cùng với điểm <1>, nếu dot product của mỗi phần tử của phần cơ bản bằng 0. Ví dụ, vector

i_1, i_2, ..., i_N được xác định là trực giao cơ bản (orthogonal basic) của R^N (or C^N).

Có một nguyên tắc : Đối với các trực giao cơ bản của R^N (hoặc C^N), vector bằng tổng của các thành phần trên hướng của vector của phần cơ bản.

Trong các trường hợp khác, đặc tính này không thay đổi đối với phần cơ bản tự nhiên mà còn cho bất kỳ một trực giao cơ bản nào

Ta đã biết họ của N vector dài N $e_m = [e_m(h)]$ được xác định như sau

$$e_m(h) = \exp(2\pi i \frac{(m-1)(h-1)}{N}) \qquad 1 \le m \le N.$$

Hình dáng trực giao cơ bản đối với C^N với ánh xạ đến điểm được xác định trước. Hơn thế nữa đối với các h, $<e_h| e_h> = N$. Để chứng minh điều này bạn có thể tìm thấy trong các sách về xử lý tín hiệu. Bạn có thể thay đổi kết quả sử dụng Matlab, với N = 16.

Trong ma trận E, ta có thể xây dựng hàng thứ m biểu diễn vector e_m. Hãy đánh dòng lệnh sau:

```
» N = 16 ;
» for m = 1 : N ;
        for n = 1 : N ;
                E(m, n) = exp(2 * pi * sqrt(-1) * (m-1) * (n - 1/N) ;
        end
end
```

Cho ta biết cấu trúc của ma trận D như các (dot product) điểm sinh ra của e_h và e_k trên vị trí (h, k). Thay đổi D được cho bởi Matlab như sau

```
» D = E * E';
```

Và bằng N * eye(N, N).

Biến đổi Fourier rời rạc X, của vector x chiều dài N, được xác định như $x(h) = <x|e_h>$. Bởi X(h) được biểu diễn cho hệ số nhân, biên của x theo hướng của e_h. Thao tác của phần x từ X, gọi là biến đổi Fourier ngược, nó được dùng trong MATLAB bởi hàm ***ift***. Như tần suất của khả năng tách ly (decomposition property), ***ift*** được biểu diễn thao tác

$$x = \sum_{h=1}^{N} \frac{X(h)}{N} e_h \qquad (6\text{-}16)$$

Ví dụ: Thay đổi công thức (6-16) cho vector ngẫu nhiên của 128 điểm.

```
» N = 128 ;
» x = read (1, N) ;
» X = ift(x) ;
» t = (0 : (N - 1)/N ;
» for h = 1 : N
        yy(h, :) = X(h)/N*exp(2*pi)2*pi*sqrt(-1)*(h-1)*t) ;
  end
» y = sum(yy) ;
```

Ta tiếp tục so sánh k và y, ở đồ thị khác

```
» plot (1 : N, x, 1: N, y)
```

Hoặc gọi số

```
» max (abs(x-y))
```

Ví dụ: Tác dụng của mã (code)

Vì kích cỡ lớn của một chuỗi, việc giải quyết vấn đề trong Matlab các vùng tín hiệu và xử lý ảnh cần có kỹ năng rất cao. Có hai thao tác: một là vùng bộ nhớ động và hai là vòng lạp (for). Vùng bộ nhớ động để chỗ cho việc cất ma trận, hoặc khi chúng ta tăng kích cỡ của một ma trận thoát ra. Điều đó đòi hỏi thời gian rất lớn. Lạp For cần phải trợ giúp việc xếp lại thao tác chuỗi như +, - :, .. với khả năng nhanh hơn ở biên.

Hai thao tác này đều có tiện ích lớn. Ví dụ khi tăng ma trận E, ta tăng được hiệu quả của mã chia vùng. Lời giải nhanh hơn nhận được nhờ vùng giải ma trận và loại bỏ một vòng với vector hoá. Viết một M-file và chạy chúng

```
N = 16 ;
P1 : zeros(N, N) ;
I : [0 : (N-1)]/N ;
for m = 1 : N
        P1(m, :) = exp(2*pi*sqrt(-1)*(m-1)*I) ;
end
```

và đánh vào

» pi1 - E

Bạn sẽ nhận được ma trận 0 kích thước 16 x 16.

Tiếp tục dùng vòng lặp for để tính chuyển FFT cho một ma trận

» P2 = fft + (eye(N,N)) ;

Bạn có thể kiểm tra kết quả với

» P2 - E

Trong thời gian này, nếu có giá trị 0, ta sẽ có sai lệch rất nhỏ cho bằng số. Cũng có thể so sánh thời gian tính với giá trị N lớn hơn.

6.7 PHỔ NĂNG LƯỢNG

Cho x là tín hiệu lặp lại theo thời gian với chu kỳ T, được lấy mẫu tại khoảng $T_s = T/N$, và X là biến đổi Fourier rời rạc.

Vector cơ bản e_h được xác định ở phần trên $\exp(2\pi i \frac{h-1}{T} t)$ của tần số (h-1)/T Hz. Vector $((X(h)/N)e_h$ là chiếu của x theo hướng của e_h. Đối với h > 1, vector này thường được xem như thành phần của x của tần số (h - 1)/THz. Đầu vào của vector x, như x(h) còn gọi là thành phần thứ h của x. Tổng trung bình của x là X(1)/N và có khi được gọi là thành phần không đổi (DC) của tín hiệu x, $(x(2)/N)e_2$ và $(x(N)/N)e_N$, liên quan đến chu kỳ T, được gọi là các thành phần cơ bản của tín hiệu này.

Hình 6.12 chỉ ra quan hệ giữa các thành phần này của ***x = fft (s)*** và tần số của các thành phần theo x.

Đối với các h = 1, 2, ..., N -1, nhóm x(1 + h) và x(1+N-h) được gọi là liên hợp (xem thêm bài tập 1.1).

Chuỗi của các phần tử $|x(h)|^2/N$ gọi là phổ năng lượng của x, như là đối với bình phương của tín hiệu hay phép nhân của nó, được thể hiện là năng lượng (công suất).

MATLAB

Nếu x là rời rạc của tín hiệu có chu kỳ T-tần số lấy mẫu là fs = N/T Hz và X là biến đổi Fourier.

1. X(1)	Liên quan đến DC của tín hiệu
2. Đối với b ≤ N/2+1(X(b))	Liên quan với tần số b-1/T = b-1/N .f_s Hz
3. Với f ≤ fs/2	Thân số f Hz liên hệ với bin b = $\frac{N.f}{fs}$

Hình 6.12 ***Mối quan hệ giữa tần số của các thành phần của tín hiệu và DFT (biến đổi Fourier rời rạc)***

Bình phương biên của vector x, được xác định bằng

$$<x/x> = \sum_{h=1}^{N}(x_h)^2 \qquad (6\text{-}17)$$

Có thể tính bằng cách sử dụng biểu thức

$$<x/x> = \sum_{h=1}^{N}\frac{(X_h)^2}{N} \qquad (6\text{-}18)$$

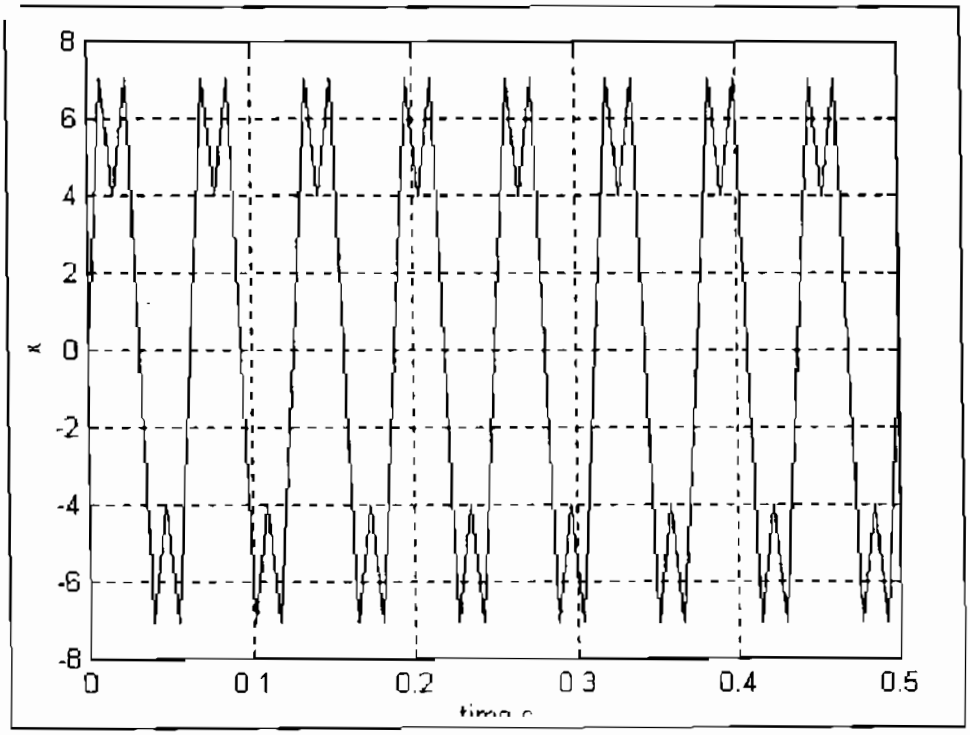
Như các thành phần hợp lại $<e_h/e_k> = \Phi$ khi $h \neq k$ nhóm $|X_h|^2/N$ thể hiện công suất của thành phần k của tần số f = (h - 1)fs/N. Do đó biểu thức (6.18) chỉ ra công suất của tín hiệu = tổng công suất của các thành phần.

Đó là một dạng của lý thuyết Parseval (1755 - 1836).

Ví dụ: Tách phổ tần số của tín hiệu

Hãy lấy tổng của hai ký hiệu tần số khác nhau và biên độ cũng khác nhau và xem xét phổ năng lượng (công suất) của nó như thế nào.

Khó khăn trong ví dụ này là giữ đúng công suất ảnh hưởng ứng với tần số sao cho biên của công suất ảnh hưởng đến biên độ $a_1 = 7$ tần số f_1 = 16Hz và tín hiệu 2 x_2 biên độ $a_2 = 3$ và tần số f_2 = 48Hz tần số lấy mẫu là 128Hz và gọi tổng của chúng.



Hình 6.13 *Tổng của hai tín hiệu hình sin*

```
» N = 512;                          % số điểm
» b = 1 : N;                        % bins
» Ts = 1/128;                       % Khoảng lấy mẫu theo giây
» fs = 1/Ts;                        % Tần số lấy mẫu theo Hz
» ts = Ts x (b - 1)                 % Khoảng lấy mẫu
» a1 = 7  ;  f1 = 16;
» x1 = a1 * sin (2 * pi * f1 * ts) ;   % tín hiệu đầu
» a2 = 3; f2 = 48
» x2 = a2 * sin (2 * pi * f2 * ts) ;   % tín hiệu thứ hai
» x = x1 + x2;
```

Vào lệnh

» plot (ts, x)

» xlabel (Time, s), y label (x)

Ta có thể xây dựng và chấm điểm của phổ công suất.

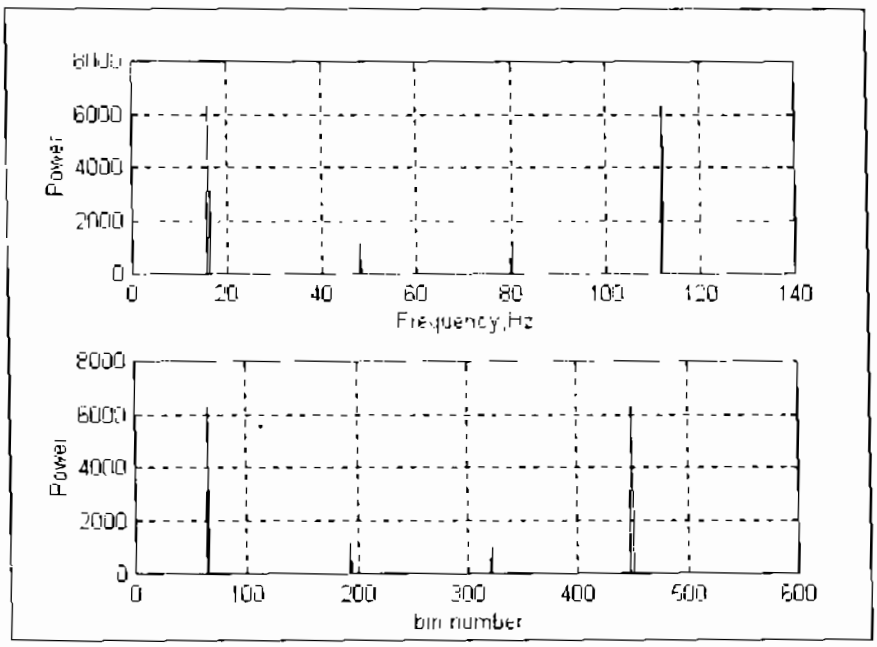
» X = fft(x); *% DFT của x*

*» pwr = x * cosj (X) / N* *% Công suất của tín hiệu*

*» frs = (b = 1), N * fs* *% Các tần số*

» Plot (frs, pwr) *% chấm điểm phổ công suất*

Kết quả chấm điểm ở trên hình 6.14. Tín hiệu x_1 đạt công suất ở điểm 65, với tần số $f_1 = 16H_z$ (65 = 1 + 512 × 16 / 128) và ở bin 449, phần chậm hơn vì liên hợp của số tín hiệu 65 = 1 + 65 được cất trong bin 449 = 1 + 512 - 64.



Hình 6.14 *Phổ năng lượng của tín hiệu x=x1+x2*

Phổ công suất được chỉ ra trên hình 6.14. Ta có thể kiểm tra **pwr (65)** $= (a_1/2)^2.N$. Tương tự như vậy đối với tín hiệu x_2. Công suất ở bin 193 và 321 và **pwr (193) = pwr (321)** $= (a_2/2)^2 N$

Ví dụ : Nhận dạng tần số và thành phần công suất chính

Trong ví dụ này ta sẽ phân tích tín hiệu tam giác của chu kỳ S = 5 giây và điểm nhảy biên độ 1 vào thành phần tần số của chúng với việc sử dụng 512 điểm lấy mẫu. Ta quan tâm đến việc tìm số phần trăm công suất tổng là thành phần tín hiệu được nhận từ gốc, bằng cách phân tích từ 4 thành phần. Ta còn muốn biết cách làm xấp xỉ tín hiệu với tín hiệu chuẩn. Trước hết, hãy xây dựng phương án rời rạc x của tín hiệu bằng cách lấy mẫu nó tại 512 điểm bằng nhau.

```
» T = S;
» N = 512;
» t = linspace (0,T, N + 1) ; t = (1 : N);
» x1 = 2 * t/T - 1/2 ;    x2 = 2*(T - t) / T - 1 / 2;
» x = min (x1, x2);              % tín hiệu tam giác và xây dựng phổ
                                  công suất của chúng;
» b = 1 : N                      % Khoảng lấy mẫu và tần số
» X = fft (x);
» Ts = T / N ; fs = N/T          % bằng (b - 1) / N * fs
» prw = X * conj (X) / N;
```

Để kiểm tra kết quả ta có thể dùng đẳng thức Parseval. Những số sau phải bằng nhau.

```
»[sum (pow) norm (x)^ 2]
ans =
      42.6680   42.6680
```

Dễ dàng nhận thấy các tần số này gồm thành phần lớn nhất của công suất, sử dụng hàm ***sort*** quay trở lại các phần tử của ***pow*** bằng cách tăng điểm:

```
» [spow, spos] = sort (pow);
```

Hãy tìm chữ số của 4 tần số thành phần công suất lớn nhất:

```
» m = 4; spos (N: -1 : (N - m + 1)
```

Ta có thể thấy các tần số này cấu thành trên 512, 2, 510 và 4. Bây giờ ta xây dựng tín hiệu xấp xỉ

```
» X4 = zesos (X);                % Vùng đổ xấp xỉ X
» h = [512  2  510  4];
```

```
» X4 (h) = X (h);                    % chóp hình cầu thành công suất cao
```

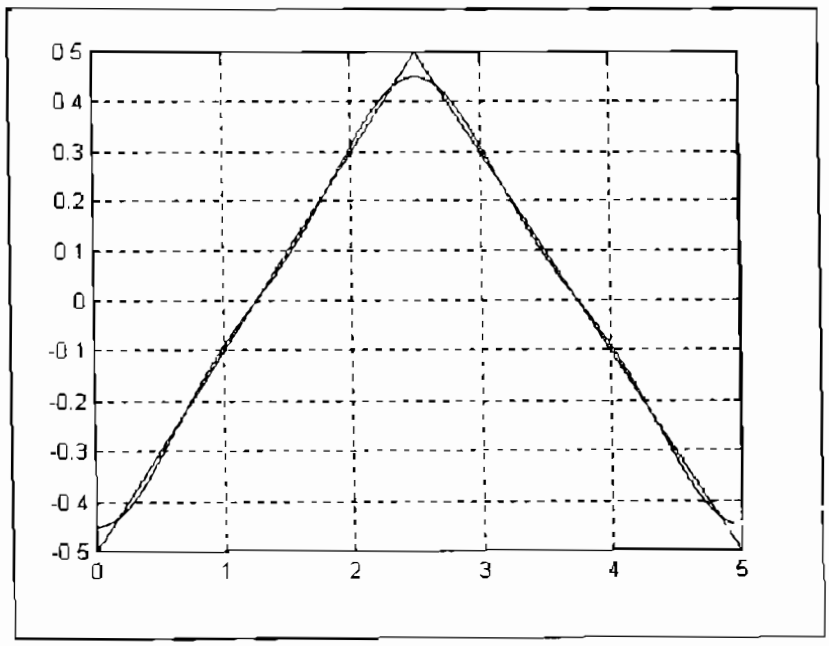
Phần trăm của công suất trên 4 thành phần chỉ đạo được đưa ra bởi

```
» pere = 100 * (norm (X4) / norm (X))^2
```

Kết luận: 99,7698 % của công suất được tạo thành trên 4 nhóm, tương ứng với tần số cơ bản 0,2 Hz; liên quan đến bin số 2, tần số truyền đạt của nó liên quan đến bin 512, giao động thứ 2, 0,6 Hz liên quan đến bin số 4, và hệ số truyền của nó, liên quan đến bin 510. Ta sẽ sử dụng kết qủa này ở mục 6.9.

Những dòng lệnh sau sẽ chỉ ra cách làm thế nào để tiến gần đến tín hiệu tam giác góc được xấp xỉ, xem hình 6.15

```
» x4 = ifft (X4);
» plot (t, [x; x4]), grid
» xlabel ( t' ), ylabel ( Tín hiệu tam giác và sự xấp xỉ chú ý)
```



Hình 6.15 *Sự xấp xỉ của tín hiệu hình tam giác*

6.8 PHẦN LƯỢNG GIÁC MỞ RỘNG CỦA TÍN HIỆU

Mục đích của phần này là làm thế nào để tạo ra version rời rạc của tín hiệu, chu kỳ T và lấy mẫu ở khoảng Ts = T/N. Có thể nhanh chóng tổ hợp tuyến tính của hình sin và cosin theo dạng sau

$$x = \sum_{h=1}^{\infty} (A_h \cos(2\pi(h-1)\frac{t}{T}) + B_h \sin(2\pi(h-1)\frac{t}{T}) \qquad (6\text{-}19)$$

Đối với t của Ts nhận thấy rằng nếu X đánh dấu chuyển đổi Fourier của x như đẳng thức (6-16)

$$x = \sum_{h=1}^{\infty} \frac{X(h)}{N} \exp(2\pi i(h-1)\frac{t}{T} \qquad (6\text{-}20)$$

Sử dụng công thức Euler, và gọi R và I tương ứng phần thực và phần ảo của X, đẳng thức (6-20) sẽ được viết lại như sau:

$$x = \sum_{h=1}^{\infty} (\frac{R_h}{N} \cos(2\pi(h-1)\frac{t}{T}) - \frac{I_h}{N} \sin(2\pi(h-1)\frac{t}{T})) + \qquad (6\text{-}21)$$

$$+ \; i \sum_{h=1}^{\infty} (\frac{R_h}{N} \sin(2\pi(h-1)\frac{t}{T}) + \frac{I_h}{N} \cos(2\pi(h-1)\frac{t}{T}))$$

và đều đúng đối với mọi x, do đó có thể đơn giản hoá khi x là số thực. Trong trường hợp đó, thành phần ảo của đẳng thức (6-21), phải triệt tiêu, dùng đồng nhất thức (6-19) cho

$A_h = R_h / N$

$B_h = -I_h / N$ và h chạy từ 1 đến N

Biểu thức (6-19) được gọi là lượng giác mở rộng của x.

Ví dụ : Biến đổi biểu thức (6.19) cho 5 giây và vector ngẫu nhiên của 128 nhóm.

```
» T = 5 ;                          % Khoảng thời gian, giây
» N = 128;                         % Chiều dài của vector
» t = linspace (0, T, N + 1);
» t = t (1 : N);                   % thời gian lấy mẫu
» x = rand (t);                    % vector ngẫu nhiên
```

```
» X = stt (x);                          % DFT của nó
»A = real (X) / N;                      % Hệ số cosine
»B = -imag (X) / N;                     % Hệ số sin
»sum cos        Zeros (N, N);
»for h = 1 : N
        sumcos (h : ) = A (h) * cos (2 * pi * (h - 1) * t/T);
        sumsin (h, ☺= - B (h) * sin (2 * pi * (h - 1) * t/N);
end
» y = sum (sumcos * sumsin);
```

Bây giờ so sánh x và y, đồ họa của chúng

```
» plot (t, x, t, y)
```

hoặc tính số

```
» Max (abs (x - y))
```

Trong version của Matlab có kết quả là 2.142e - 19.

Ví dụ : Phân tích lượng giác của tín hiệu tam giác.

Bây giờ ta hãy phân tích tín hiệu tam giác x tính trong thành phần lượng giác của nó và kiểm tra kết quả. Nếu chữ số N = 512 xuất hiện trong nhóm tiếp theo của lệnh thì cần có dung lượng rất lớn cho bộ nhớ của máy tính. Do vậy, ta sẽ giảm nó thành số nhỏ, chẳng hạn là 32

```
» T = 5;
» N = 512;
» t = linspace (0, T, N + 1); t = (1 : N);
» x1 = 2 * t / T - 1/2 ; x2 = 2 * (T - t) / T - 1/2;
» x = min (x1, x2);                      % tín hiệu tam giác
» plot (t, x)
```

Hãy tính hệ số của sines và cosine.

```
» X = fft (x);
» A = real (X) / N;                      % hệ số cosine
```

```
» B = - imag (X) / N);                          % hệ số sine
» sumcos = zeros (N, N);
» sumsin = zeros (N, N);
» for h = 1 : N
        sumcos (h, : ) = A(h) * cos (2 * pi * (h - 1) * t/T);
        sumsin (h, : ) = B (h) * sin (2 * pi * (h - 1) * t/T);
end
» y = sum (sumcos + sumsin);
```

Ta có thể kiểm tra các kết quả bằng cách so sánh x và y, đồ họa của chúng

```
» plot (t, x, t, y);
```

và số

```
»max (abs (x - y))
```

6.9 NHỮNG TÍN HIỆU TẦN SỐ CAO VÀ KÝ HIỆU

Ở hình 6.11 đã chỉ ra sự tương ứng giữa công suất của tín hiệu và biến đổi Fourier của nó đối với các tần số đến tần số Nyquist. Điều này trở nên thú vị khi xem điều gì xảy ra khi ta lấy mẫu tại khoảng thời gian Ts bằng số tín hiệu tuần hoàn liên tục của tần số cao đến tần số Nyquist $N_f = 1/(2Ts)$. Ở đây, version lấy mẫu của tín hiệu đồng nhất với tín hiệu khác tần số thấp. Hiện tượng này gọi là dấu hiệu từ C_1 "những cái khác". Để nhấn mạnh ý này ta chọn T là 5 giây, N = 16 lấy mẫu trong một chu kỳ, và hiện ra theo khoảng lấy mẫu với Ts = T/N và tần số mẫu với fs > 1/Ts.

Tín hiệu tuần hoàn với chu kỳ cơ bản của T, nối T/k với k phù hợp. Cũng như tín hiệu cho các khoảng cách sin (2πft) và cos (2πft), tần số f có thể luôn viết như sau

$$\mathbf{f = f_{app} + nfs}$$

Trong đó n và số nguyên và $0 \le |f_{app}| < N_f$. Nó dễ dàng kiểm tra tại các t bội số của Ts như sau $\mathbf{t = hTs, \sin(2\pi ft) = \sin(2\pi f_{app} t).}$

Thực tế

$$\mathbf{\sin(2\pi ft) = \sin(2\pi (f_{app} + nfs) t)}$$
$$= \sin(2\pi (f_{app} + nfs) hTs$$

$$= \sin(2\pi (f_{app} hTs + 2\pi nfshTs)$$

$$= \sin(2\pi (f_{app} hTs + 2\pi nh)$$

$$= \sin(2\pi f_{app})$$

Những tín hiệu $x = sin(2\pi ft)$, tần số f, khi lấy mẫu ở tần số fs, lại không thể phân biệt được từ tín hiệu $x = \sin(2\pi f_{app} t)$ của tần số thấp f_{app}. Matlab cho phép chúng ta giải quyết vấn đề này và biểu diễn các dấu hiệu. *Hãy dùng m tệp sau; alias.m:*

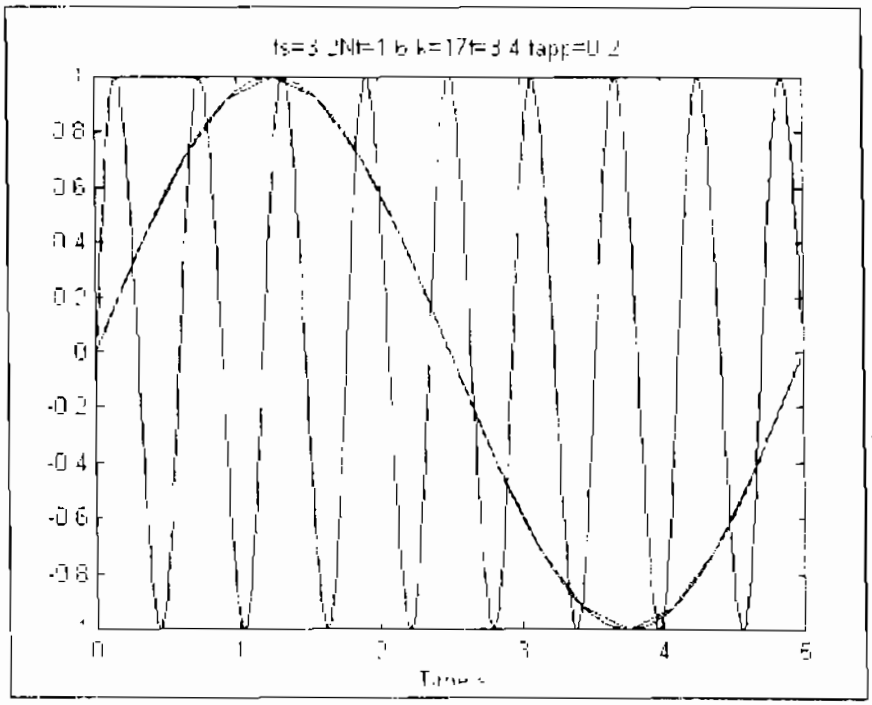
```
T = 5 ;                          % tần  số cơ bản
Np = 512;                        %số điểm để chấm
t = linspace(0,T,Np+1:
t = t(1:Np);                     % tìm  độ phân giải của thời gian
                                 %để chấm điểm
N=16;                            % số điểm lấy mẫu
Ts =T/N;                         % khoảng lấy mẫu
fs =1/Ts;                        % tần số lấy mẫu
ts = Ts*(0:(N-1));               % khoảng thời gian lấy mẫu
Nf = 1/(2*Ts);                   % Tần số Nyquist

f = k/T;                         % tần số liên tục
                                 % tín hiệu
x = sin(2*pi*f*t);               % tín hiệu, độ phân giải cao
xs = sin(2*pi*f*ts);             % tín hiệu, lấy mẫu phân giải
% tim fapp, như sau: f =n*fs+fapp
n = round(f/fs);
fapp = f-n*fn;
xa = sin(2*pi*fapp*t);
plot(t,[x;xa],ts,xs,'0');
str1 = ['fs = ', num2str(fs), 'Nf = ',num2str(Nf)];
str2 = ['k = ', num2str(k), 'f = ',num2str(f)];
str3 = [fapp=',  num2str(fapp)];
```

str = [str1, ' ',str2, ' ', str3];

title(str);

Chạy chúng với lệnh sau

» *k= 17;* *alias*



Hình 6.16 ***Tín hiệu tần số cao lấy mẫu như một tần số thấp.***

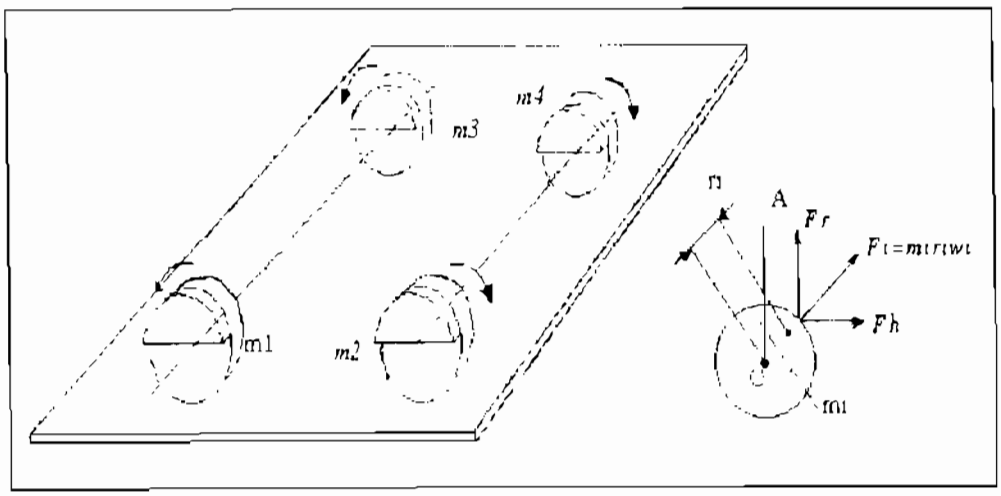
Ví dụ: Giao động của một tấm

Việc tính toán ở các ví dụ nêu trên có liên quan đến một ứng dụng kỹ thuật trong máy kiểm tra dao động. Ví dụ đơn giản là máy có dạng như hình 6.17. Các bộ phận hoạt động của máy là 4 trục quay, không có khối lượng giao động m đến m_4. Như mô tả ở trên hình 6.17(a). Khối lượng m_1 và m_3 bằng nhau, nhưng quay theo hai hướng đối ngược nhau, và cũng tương tự như đối với khối lượng m_2 và m_4. Một trong những đĩa quay được trình bày chi tiết trên hình 6.17 (b). Giả sử khoảng cách giữa trục quay qua điểm 0 và tâm của khối lượng không giao động, m là r, khối lượng quay quanh điểm 0 với tốc độ ω_1. Lực ly tâm đặt vào tâm của khối lượng không giao động bằng $F_1 = m_1 r_1 \omega_1^2$. Chuyển động quan sát theo trục thang đứng OA và hướng quay

theo chiều kim đồng hồ, sau thời gian t góc giữa OA và hướng của $F = \omega_i t$. Thành phần thẳng đứng của lực hướng tâm là $F_v = m_i r_i \omega_i^2 . cos \omega_i t$, và thành phần nằm ngang là $F_h = m_i r_i \omega_i^2 . sin\ \omega_i t$. Khối lượng bên phải bằng khối lượng có chiều quay ngược, quan sát theo trục đứng.

Lực hướng tâm được phân ra thành phần thẳng đứng F_v, và thành phần ngang $-F_h$. Thành phần nằm ngang giao động quanh điểm, khi thành phần thẳng đứng hướng lên cao, *sinh ra lực đàn hồi* $= 2\ m_i r_i \omega_i^2 . cos\ \omega_i t$. Tích $m_i r_i$ biểu diễn mômen tĩnh của khối lượng theo trục quay.

Nếu hai cặp đếm khối lượng quay sắp xếp trên cùng một bàn đàn hồi và *tỉ số giữa mòment và góc quay của chúng có thể tính (gần đúng)*, thì có thể tổng hợp được các xung đàn hồi của các hình dạng khác nhau. Hãy thử xấp xỉ dạng sóng được phân tích trong ví dụ ở mục 6.8. Ta gọi ra 4 thành phần tạo nên năng lượng chủ yếu. Đó là giao động đầu tiên với tần số ***0.2 Hz***, và liên kợp của nó, giao động thứ 3, tần số 0.6 Hz, và liên hợp của nó. Liên hợp tương ứng theo chiều ngược lại với các tần số ***0.2 Hz và 0.6 Hz***. Điều đó có nghĩa là cặp khối lượng không giao động quay theo hướng ngược lại như hình 6.17, sẽ sinh ra lực tương ứng với cặp liên hợp trong phần lượng giác mở rộng của lực. Biên độ của các thành phần tỷ lệ theo hệ số với lượng giác mở rộng. Chúng bằng 0.2026N cho tần số 0.2 Hz và -0.2Hz và 0.0225N cho tần số 0.6Hz và -0.6Hz.



Hình 6.17 ***Máy kiểm tra giao động***

Ta bắt đầu thiết kế máy đàn hồi bằng cách đưa vận tốc góc của khối lượng không giao động theo rad/s.

```
» omega 1 = 2 * pi * 0.2, omega2 = 2 * pi * 0.6
  omega 1 =
                    1.2566
  omega 2 =
                    3.7699
```

Tiếp theo hãy đưa vào biên của lực được sinh ra bởi trọng lượng không giao động

```
» F1 = 0.2026 ; F2 = 0.0225;
```

Mômen trọng lượng, m_1r_1, m_2r_2 (kgm), sinh ra những lực sau

```
» r1m1 = F1 / omega1 ^ 2
  r1m1 =
        0 . 1283
» r2m2 = F2 / omega 2 ^ 2
  r2m2 =
        0.0016
```

Giả thiết là khối lượng không giao động là 1 đĩa tròn dày 0,02m, làm bằng thép có khối lượng riêng 7850 kg/m^3. Mômen tĩnh của vùng segment (tính ra m^3) là

```
» S1 = r1m1 / (0.02 * 7850)
  S1 =
        8.1718 e - 04
»S2 = r2m2 / (0.02 * 7850)
  S2 =
        1.0084 e - 05
```

Điều này chứng tỏ mômen của vùng segmen trên đĩa phụ thuộc vào tổ hợp của t và = t^3 / 12. Dùng công thức sau để tính tổ hợp của segment trên đĩa, theo m,

```
» t1 = (12 * S1)^ (1/3)
  t1 =
        0.2140
» t2 = (12 * S2)^ (1/3)
```

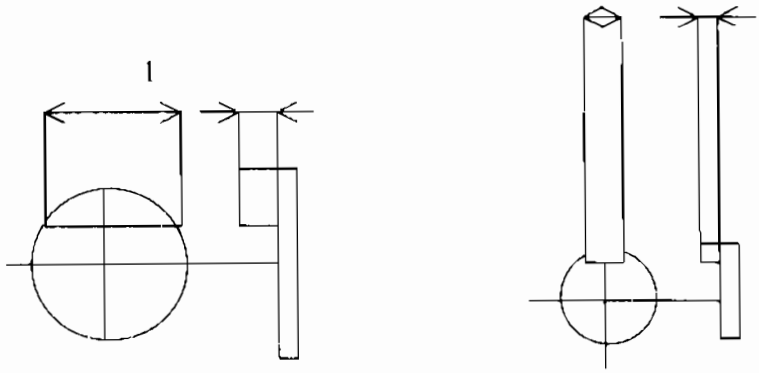
```
t2 =

        0.0495
```

Nếu giảm hợp của khối lượng m1, m2 bằng cách tăng chiều dày của chúng đến 0,03m:

```
» S1 = r1m1 / (0.03 * 7850)

  S1 =

        5.4479 e - 04

» t1 = (12 * S1)^ (1/3)

  t1 = 0.1870
```

Ta sẽ chỉ ra khối lượng giao động thiết kế sinh ra lực thẳng đứng khi đồ thị thời gian xấp xỉ hình tam giác. Bắt đầu bằng việc xác định trục thời gian.

```
» t = 0; 0.02 : 10:
```



Hình 6.18 *Kích thước của khối lượng không giao động và tiếp tục viết các điều hoà chính*

```
» f1 = 2* r1m1 * omega1 ^ 2 * cos (omega1 * t);

» f2 = 2 * r2m2 * omega2 ^ 2 * cos (omega2 * t):
```

Chấm các điểm nhận được

```
» plot (t, (f1 + f2)

» grid

» title ( Tổng hợp lực đàn hôi hình tam giác')
```

» x label (t, S)

» y label (F, N)

Hãy kiểm tra trên hình 6.18 chu kỳ của sóng tam giác là năm giây và biên độ của lực đàn hồi là 0.45N, gần với 0.5N.

6.10 PHẦN BÀI TẬP VỀ XỬ LÝ SỐ

1) Mệnh đề liên hợp

a/ Thay đổi những mệnh đề sau cho vector x xác định trong MATLAB bởi N = 128; x = rand (1, N);

Nếu x là số thực có chiều dài N và x là biến đổi Fourier rời rạc, đối với mỗi h trong khoảng [1, N - 1], x (1 + N - h) là số phức liên hợp của x (1 + h)

b/ Nếu bạn theo hướng toán học, chứng minh mệnh đề cho mỗi vector thực x.

Giả thiết là $x = x^*$ và $e_{j,k} = e_{j,k}^*$

2) Xác định đặc tính tần của bộ lọc

Một số hàm của MATLAB như ***yulewalk*** và ***remez***, xây dựng các hệ số của bộ lọc số bằng cách xấp xỉ với tần số mô tả các tính chất

a/ Xây dựng hàm ***deffiltm.m:*** cho phép người sử dụng xác định đặc tính tần của bộ lọc khi nhấy vào điểm trên biên plane tần số với chuột và quay về chuỗi của tần số không thứ nguyên (như các tần số qui chuẩn với tần số Nyquist), f_n và biên M.

b/ Kiểm tra hàm số: Xác định ý nghĩa của ***deffilt.m*** được xây dựng ở (a) của bộ lọc số này, trên tần số lấy mẫu tại 100Hz (tức là tần số Nyquist là 50Hz), có những đặc tính sau:

Biên	Tần số
1.0	0
1.0	10
0.5	20
0.5	30
1.0	40
1.0	50

3) Mô tả IIR - yulewalk

Tín hiệu được lấy mẫu tại 800Hz. Ta dùng hàm ***yulewalk*** để thiết kế bộ lọc IIR với xấp xỉ bộ lọc F, xác định bởi đặc tính tần sau:

Từ Hz	Đến Hz	Biên
0	100	0
100	150	Tăng tuyến tính từ 0 đến 2
150	180	2
180	200	Giảm đều từ 0.5
200	240	0.5
200	300	Tang đều từ 0.5 đến 1
300	400	1

a/ Viết chuỗi f_n và m_n để xác định đặc tính của bộ lọc từ yêu cầu bằng ***yulewalk***. Viết tần số là bội của tần số Nyquist.

b/ Thay đổi lời giải đúng vào (a) bằng chấm điểm m_n ***versus*** f_n.

c/ Sử dụng hàm ***yulewalk***, tìm các hệ số của bộ lọc cho 6, 8, 10 điểm bằng cách xấp xỉ bộ lọc đã đưa ra.

d/ So sánh đặc tính đó hoạ của bộ lọc nhận được với F.

4) Kiểm tra bộ lọc với đầu vào hình sin

Khi giải bài (3) ta có chuỗi

bIIR6 = [0.51 69 - 0.7337 0.6589 - 0.6989 0.4929 - 0.1354 0.1355]

aIIR6 = [10000 - 0.3217 1.2452 - 0.089 0.5872 - 0.0185 0.1643]

biểu diễn các hệ số của bộ lọc số. Nếu ta không cất chúng thì hãy đưa vào bằng tay. Muốn kiểm tra bộ lọc này có đặc tính tần yêu cầu hay không, hãy kiểm lại nó theo số điển vào hình sin, như sau

(a) Xây dựng phần rời rạc của tín hiệu $s = \sin(2\pi ft)$ lấy mẫu ở 800Hz trong thời gian 1giây, đối với f = 100Hz.

(b) Dùng hàm lọc filter qua S, qua bộ lọc xác định với hệ số của bIIR6 và aIIR6 và gọi kết quả fs. Chấm điểm fs như một hàm thời gian, đối với t trong khoảng [0.5, 0.6], sau thời gian đủ có tác động của trạng thái thì sẽ

xoá.

(c) Kiểm tra ở các bộ lọc có đặc tính tần rời rạc

(ví dụ tín hiệu 100Hz, chính xác 0,5)

(d) Thay đổi bộ lọc có đặc tính tần như (3)

(e) Lặp lại câu c cho tần số 100, 150, 180, 200, 240 và 300 Hz.

5) Thiết kế FIR - remez

Tín hiệu lấy mẫu tại 400Hz. Ta sử dụng hàm ***remez*** để thiết kế FIR lọc số cùng với hàm lọc F xấp xỉ, xác định bởi đặc tính tần như sau:

Từ Hz	Đến Hz	Biên
0	25	1
25	50	Giảm tuyến tính từ 1 đến 0
50	100	0
100	150	Tăng tuyến tính từ 0 đến 1
150	200	1

a/ Viết hai chuỗi f_o và m_o để xác định đặc tính cầu bộ lọc trên dựa vào ***remez***.

b/ Thay đổi cho đúng với lời giải (a), bằng cách chấm điểm m_o theo f_o

c/ Sử dụng hàm ***remez*** tìm hệ số của bộ lọc cho 10, 20, 30 điểm (gọi chúng tương ứng với bFIR10, bFIR20, bFIR30) và xấp xỉ bộ lọc được đưa ra.

d/ So sánh đồ hoạ đặc tính của bộ lọc nhận được với F.

6) Hàm Bilinear cùng với tính toán khoảng lấy mẫu

Hàm chuyển đổi của bộ lọc thông thấp được xác định theo mặt s như sau:

$$H(s) = \frac{1}{1 - s\ p} \cdot \frac{w^2 n}{w_n^2 + 2\xi w_n s + s^2}$$

với P = 6 rad/s; w_n = 15rad/s và ξ = 0.6

a/ Hãy viết H(s) như là tỷ số của 2 đa thức ***num*** và ***den*** tính nhanh

chúng nhờ sử dụng chuyển đổi MATLAB (chuỗi các hệ số của việc giảm năng lượng của s)

Yêu cầu : Sử dụng hàm ***cour.***

b/ Chấm các điểm biên đáp ứng của bộ lọc được xác định bởi hàm tỉ số H(s), trong khoảng 0.1 đến 100 rad/s, bảng hiển thị các tần số theo thang logarithm và biên theo dB.

c/ Chấm điểm đáp ứng pha trên cùng một khoảng, bảng hiển thị tần số theo rad/s và pha theo độ (sử dụng ***unwarp*** nếu có khả năng cài đặt vào máy của bạn).

d/ Tìm các hệ số của số tương đương của nó, lấy tổng các tần số lấy mẫu của 50Hz

Chú ý: Sử dụng hàm **bilinear.**

7) Xấp xỉ khoảng lấy mẫu cho hàm bilinear

Hàm biến đổi của bộ lọc thông thấp, xác định theo s như sau:

$$H(s) = \frac{1}{1 + s\,p^{-1} w_n^{-2} + 2\xi w_n s + s^2}$$

với P = 8rad/s, w_1 = 20rad/s và ξ = 0.65. Hãy xây dựng bộ lọc số tương đương và xác định tần số lấy mẫu với sử dụng help của MATLAB. Viết mẫu Chương trình bằng MATLAB như sau:

a/ Nhắc người sử dụng hàm số lấy mẫu Fs;

b/ Sử dụng hàm **bilinear**, cấu tạo bộ lọc số tương đương ứng với Fs1

c/ Hiển thị ra đặc tính và pha của bộ lọc tương tự và bộ lọc số tương đương.

d/ Lặp lại bước (a), (b), (c) cho đến khi người dùng thoả mãn, lúc đó thoát ra khỏi vòng lặp.

e/ Dựa trên vòng lặp trên hiện thị tần số lấy mẫu và các thông số của bộ lọc số

f/ So sánh đồ thị của đáp ứng của bộ lọc số được xây dựng ở (d) với bộ lọc tương tự đầu tiên.

8) FFT và bins

X là 256 điểm FFT, có được bằng DFT đến chuỗi x có 256 điểm, và tần số của tín hiệu lặp lại tại 64Hz. Nhìn vào phổ công suất, chúng ta thấy có nhảy ở bin thứ 33. Điều đó chỉ ra khả năng thành phần của tần số lên tín hiệu gốc (có nghĩa là dấu hiệu có thể được thực hiện)

Dùng công thức ở hình 6.11

9) Aliasing (ký tự)

Giả sử có tín hiệu được xác định ở bài 8. Nếu ký tự là hiện tại, với các tần số khác có cùng bước nhảy không?

10) Phổ công suất của tín hiệu tam giác

Tín hiệu lặp lại hình tam giác được lấy mẫu tại 256Hz theo thời gian khoảng 0.5 giây, sinh ra chuỗi x có 4 điểm của giá trị 0 với 16 điểm giá trị 1.

a/ Chấm điểm x như một hàm theo thời gian, mỗi khoảng 0.5 giây.

b/ Tính toán chấm điểm phổ tần của x

c/ Chỉ ra 5 tần số đầu tiên nó gồm những công suất lớn nhất .

Chú ý: Sử dụng hàm ***sort***

d/ Xấp xỉ x bởi 5 giao động (gọi kết quả x ppr5) và chấm điểm x và ***xapprs***.

11) Lọc và tín hiệu

Bộ lọc thông thấp F được xác định với chuỗi của các hệ số a và b có được bằng cách sử dụng lệnh MATLAB.

» *[b, a] = butter (5, 0.5)*

Tín hiệu x, được xác định như sau:

» *Ts = 1/100 ; t = Ts * (1 : 500);*

» *f = 25 : x = sin 92 * pi * f * t);*

Có sin 2πft và được lấy mẫu lại 100Hz trong đoạn [0, 5] sử dụng hàm ***filter*** mà không xác định giới hạn trạng thái zi (có nghĩa là MATLAB tự xác

định 0), tìm tín hiệu y, có được bởi cho x qua bộ lọc F, và nhận dạng hiệu quả ưu tiên bằng cách chấm điểm y như hàm của trung đoạn [0,0.2).

12) Bộ lọc với mô tả trạng thái tới hạn

Bộ lọc thông thấp F được xác định bang 1 chuỗi của các hệ số a và b có được nhờ sử dụng dòng lệnh của MATLAB

» [b, a] = butter (5, 0.5)

Tín hiệu x được xác định như sau.

*» Ts = 1/100 ; f = 25 ; x = sin (2 + pi * z * Ts * (1 : 500) ;*

Tìm vector Z; như lệnh sau

»y = filter (b,a,x,z)

Sinh ra vector y bắt đầu với 5 zeros. Kiểm tra câu trả lời bằng cách chấm điểm bắt đầu 30 nhóm của y.

*** Chú ý**: Dùng hàm ***filteric.***

6.11. CÁC HÀM THƯ VIỆN THÔNG DỤNG TRONG TOOLBOX - XỬ LÝ TÍN HIỆU SỐ DSP

6.11.1. Hàm sinh ra các dạng sóng

Chirp	Phát hàm cosin
Diric	Hàm tuần hoàn sinc
Gauspull	Phát xung Gaussian
Pulstran	Phát một dãy xung
Rectpuls	Phát hình vuông lấy mẫu không tuần hoàn
sawtooth	Hàm răng cưa
sinc	Hàm sinc hoặc sin(pi*x)/(pi*x)
square	Hàm sóng bình phương
tripuls	Máy phát hình thang lấy mẫu không tuần hoàn

6.11.2 Phân tích bộ lọc và thực hiện chúng

Abs	Giá trị tuyệt đối của số ảo
Angle	Góc pha
Conv	Quay
Fftfilt	Thực hiện bộ lọc over lap-add
Filter	Thực hiện bộ lọc
filtfilt	Bộ lọc pha không
filtic	Bộ lọc xác định điều kiện đầu
freqs	Biến đổi Laplace tần số đáp ứng
freqspace	Đạt tần số cho đáp ứng tần số
freqz	Biến đổi z tần số đáp ứng
grpdelay	Một nhóm trễ
impz	Đáp ứng xung (rời rạc)
latcfilt	Thực hiện bộ lọc Lattice
unwrap	Không bó pha
upfirdn	Bộ lọc FIR không lấy mẫu, lấy mẫu xuống
zplane	Chấm điểm cực rời rạc

6.11.3 Các biến đổi hệ tuyến tính

Convmtx	Ma trận quay(Ma trận chuyển vị, hay nghịch đảo)
latc2tf	Lưới và/hoặc lưới bậc thang để truyền hàm chuyển đổi
poly2rc	Đa phương đến hệ số biến đổi
rc2poly	Hệ số phản xạ để biến đổi đa phương
residuez	Miền mở rộng thập phân của biến đổi z
sos2ss	Chuyển đổi các vùng thứ hai để đặt trạng thái chuyển đổi
sos2tf	Chuyển đổi các vùng thứ hai để truyền hàm chuyển đổi
sos2zp	Chuyển đổi các vùng thứ hai đến trường không

ss2sos	Đặt trạng thái để đạt điểm thứ hai của vùng chuyển đổi
ss2zp	Đặt trạng thái đến chuyển đổi trường không
ss2tf	Đặt trạng thái để truyền hàm chuyển đổi
tf2latc	Truyền hàm đến lưới hoặc chuyển đổi lưới hình thang
tf2ss	Truyền hàm đến chuyển đổi trạng thái
tf2zp	Truyền hàm đến vùng chuyển đổi trường không
zp2sos	Chuyển đổi từ trường không đến vùng đạt thứ hai
zp2ss	Chuyển đổi từ trường không đến điểm trạng thái
zp2tf	Chuyển đổi từ trường không đến hàm truyền

6.11.4 Thiết kế bộ lọc số IIR

butter	Thiết kế hàm lọc đơn giản nhất
cheby1	Thiết kế bộ lọc Chebyshewl
cheby2	Thiết kế bộ lọc ChebyshewII
ellip	Thiết kế bộ lọc Elliptic (dạng Ellip)
maxflat	Thiết kế bộ lọc thong thấp được sinh ra một cách đơn giản nhất
yulewalk	Thiết kế bộ lọc Yule-Walker

6.11.5 Chọn bộ lọc cho trước IIR

Buttord	Chọn bộ lọc đơn giản Butterworth cho trước
cheb1ord	Chọn bộ lọc Chebyshew 1 cho trước
cheb2ord	Chọn bộ lọc Chebyshew 2 cho trước
ellipord	Chọn bộ lọc Ellip cho trước

6.11.6 Thiết kế bộ lọc FIR

cremez	Thiết kế bộ lọc FIR số phức và hiệu ứng (méo nhỏ) ripple pha không tuyến tính
fir1	Thiết kế cửa sổ cơ bản của bộ lọc FIR - thấp ,cao, thông giữa,dừng,tích
fir2	Thiết kế cửa sổ cơ bản của bộ lọc FIR - Đáp ứng tuỳ ý
fircls	Thiết kế bộ lọc ở điều kiện bình phương lớn nhất - Đáp ứng tuyf ys
fircls1	Thiết kế bộ lọc FIR ở điều kiện bình phương lớn nhất - thông thấp và thông cao
girrcos	Thiết kế bộ lọc FIR cosine lớn dần
firls	Thiết kế bộ lọc FIR- đáp ứng tuỳ ý cùng với vùng chuyển đổi
inflit	Thiết bộ lọc FIR nội suy
kaiserord	Chọn điểm đặt cửa sổ cơ bản của bộ lọc sử dụng cửa sổ Kaiser
remez	Thiết kế hàm lọc tối ưu FIR Parks-McChellan
remezord	Chọn hàm lọc đặt trước Parks-McChellan

6.11.7 Các chuyển đổi

czt	Biến đổi Z
dct	Biến đổi Cosine rời rạc
dftmtx	Ma trận biến đổi Fourier rời rạc
fft	Biến đổi Fourier nhanh
fftshift	Chuyển đổi veetor halves
hilbert	Biến đổi Hilbert
idct	Biến đổi cosin rời rạc ngược
ifft	Biến đổi fourier ngược nhanh

6.11.8 Xử lý tín hiệu thống kê và phân tích phổ

cohere	Hàm đánh giá chặt chẽ
corrcoef	Hệ số hiệu chỉnh (hệ số bù)
cov	Ma trận sai lệch
csd	Mật độ phổ cắt nhau
pburg	Định lượng phổ công suất theo phương pháp Burg
pmtm	Định lượng phổ công suất theo phương pháp Thomson
pmusic	Định lượng phổ công suất theo phương pháp âm nhạc
psd	Định lượng phổ công suất theo phương pháp Welch
pyulear	Định lượng phổ công suất theo phương pháp Yule-Walker
spectrum	psd, csd, đính kết và tổ hợp tfe
tfe	Đánh giá hàm truyền
xcorr	Hàm bù (hiệu chỉnh) giao nhau
xcov	Hàm sai lệch

6.11.9 Các cửa sổ tín hiệu

Bartlett	Cửa sổ Bartlett
Blackman	Cửa sổ Blackman
Boxcar	Cửa sổ Boxcar
Chebwin	Cửa sổ Chebwin
hamming	Cửa sổ hamming
hamning	Cửa sổ hamning
kaiser	Cửa sổ Kaiser
triang	Cửa sổ có dạng tam giác

1.11.10 Thông số khi mô hình hoá

invfreqs	Bộ lọc tương tự phù hợp với đáp ứng tần số
invfreqz	Bộ lọc rời rạc phù hợp với đáp ứng tần số
lpe	Các hệ số tuyến tính đoán trước sự dụng phương pháp tự bù
prony	Bộ lọc rời rạc Prony phù hợp với đáp ứng thời gian
stmcb	

6.11.11 Các thao tác đặc biệt

decimate	Lấy mẫu số liệu ở khoảng lấy mẫu thấp nhất
deconv	Quay ngược trước
demod	Mô hình hoá để chạy mô phỏng quá trình truyền tin
dpss	Rời rạc miền không gian tần số
dpssclear	Chuyển miền không gian tần số rời rạc vào miền cơ sở dữ liệu
dpssload	Nạp vào miền không gian tần số rời rạc từ miền cơ sở dữ liệu
dpsssave	Cất miền không gian tần số rời rạc vào miền cơ sở dữ liệu
interp	Lấy mẫu lại số liệu ở khoảng lấy mẫu cao hơn
interp1	Nội suy một chiều chung cho toolbox
medfilt1	Sự lọc điểm giữa một chiều
modulate	Modul hoá để mô phỏng các quá trình truyền tin
resample	Lấy mẫu tần số với khoảng lấy mẫu mới
specgram	ảnh phổ, đối với tốc độ , tín hiệu
spline	Nội suy theo hình hộp
vco	Tạo giao động điều khiển áp

6.11.12 Làm mẫu bộ lọc tương tự thông thấp

besselap	Làm mẫu bộ lọc Bessel
buttap	Làm mẫu bộ lọc Butter
cheb1ap	Làm mẫu bộ lọc Chebyshev dạng 1 (Sai nhỏ ở giữa giải thông)
cheb2ap	Làm mẫu bộ lọc Chebyshev dạng 2 (Sai nhỏ ở cuối giải thông)
ellipap	Làm mẫu bộ lọc dạng Ellip

6.11.13 Chuyển đổi tần số (Dịch tần số)

lp2bp	Biến đổi bộ lọc thông thấp thành thông theo dải
lp2bs	Biến đổi bộ lọc thông thấp thành thông đỉnh
lp2hp	Biến đổi bộ lọc thông thấp thành thông cao
lp2lp	Biến đổi bộ lọc thông thấp thành thông thấp

6.11.14 Rời rạc hoá bộ lọc

bilinear	Sự chuyển đổi nửa tuyến tính với vùng được chọn trước
impinvar	Chuyển đổi xung bất biến tương tự thành số

6.11.15 Những hàm khác

besself	Thiết kế bộ lọc tương tự Bessel
conv2	Quay hai chiều
cplxpair	Vector đặt trước vào bộ số phức liên hợp
fft2	Biến đổi Fourier nhanh hai chiều
ifft2	Chuyển đổi ngược hai chiều Fourier nhanh
polystab	Sự bền vững đa dạng
stan	Chấm điểm số liệu tần số rời rạc
strips	Chấm điểm phóng ra
xcorr2	Giao hỗ hai chiều

CHƯƠNG 7

ỨNG DỤNG PHẦN MỀM SIMULINK

7.1 KHÁI NIỆM VỀ SIMULINK

Simulink là một phần mềm dùng để mô hình hoá, mô phỏng và phân tích một hệ thống động. **Simulink** cho phép mô tả hệ thống tuyến tính, hệ phi tuyến, các mô hình trong thời gian liên tục, gián đoạn hay một hệ kết hợp cả liên tục và gián đoạn. Hệ thống cũng có thể có nhiều tốc độ khác nhau có nghĩa là các phần khác nhau lấy mẫu và cập nhật số liệu ở tốc độ khác nhau.

Để mô hình hoá **Simulink** cung cấp một giao diện đồ hoạ để xây dựng mô hình như là một sơ đồ khối sử dụng thao tác "nhấn và kéo" chuột. Với giao diện này bạn có thể xây dựng mô hình như xây dựng trên giấy. Đây là sự khác xa các phần mềm mô phỏng trước nó mà ở đó người sử dụng phải đưa vào các phương trình vi phân và các phương trình sai phân bằng một ngôn ngữ lập trình.

Việc lập trình trên **Simulink** sử dụng các đối tượng đồ họa gọi là Graphic Programming Unit. Nó được xây dựng trên cơ sở của các ngôn ngữ lập trình hướng đối tượng, tạo điều kiện hết sức thuận lợi cho việc thay đổi giá trị các thuộc tính trong những khối thành phần. Loại hình lập trình này có xu thế được sử dụng nhiều trong kỹ thuật bởi ưu điểm lớn nhất của nó là tính trực quan, dễ viết và hình dung đối với những người lập trình không chuyên nghiệp cũng như những người không muốn bỏ nhiều thời gian cho việc học thêm một ngôn ngữ lập trình mới.

Thư viện của **Simulink** cũng bao gồm toàn bộ thư viện các khối như khối nhận tín hiệu, các nguồn tín hiệu, các phần tử tuyến tính và phi tuyến, các đầu nối chuẩn. Dĩ nhiên là người sử dụng cũng có thể thay đổi hay tạo ra các khối riêng của mình. Các mô hình bài toán trong Simulink được xây dựng có thứ bậc hay còn gọi là theo mô hình phân cấp, điều đó cho phép người sử dụng có thể xây dựng mô hình theo hướng từ dưới lên hay từ trên

xuống. Người sử dụng vừa có thể quan sát hệ thống ở mức tổng quan, vừa có thể đạt được mức độ cụ thể bằng cách nháy kép vào từng khối xác định để xem xét chi tiết mô hình của từng khối. Với cách xây dựng kiểu này, người sử dụng có thể hiểu được sâu sắc tổ chức của một mô hình và những tác động qua lại của các phần trong mô hình như thế nào.

Sau khi tạo lập ra được một mô hình, người sử dụng có thể mô phỏng nó trong **Simulink** bằng cách nhập lệnh trong cửa sổ lệnh của Matlab hay sử dụng các menu có sẵn. Việc sử dụng các Menu đặc biệt thích hợp cho các công việc có sự tác động qua lại lẫn nhau, còn sử dụng dòng lệnh thường hay được dùng khi chạy một loạt các mô phỏng. Các bộ Scope và các khối hiển thị khác cho phép người sử dụng có thể xem kết quả trong khi đang chạy mô phỏng. Hơn nữa người sử dụng có thể thay đổi thông số một cách trực tiếp và nhận biết được các ảnh hưởng đến mô hình.

Cần nhấn mạnh điều quan trọng trong việc mô phỏng một quá trình là việc thành lập được mô hình. Để sử dụng tốt chương trình này bạn cần có kiến thức cơ bản về điều khiển, xây dựng mô hình toán học theo quan điểm của lý thuyết điều khiển và từ đó thành lập nên mô hình bài toán mà bạn cần giải quyết.

Kết quả mô phỏng có thể đặt vào Matlab để xử lý đưa ra máy in hay hiển thị. Công cụ phân tích mô hình bao gồm cả công cụ tuyến tính hoá và "trimming" mà ta có thể truy nhập từ dòng lệnh của Matlab, hơn nữa ta cũng có rất nhiều công cụ trong Matlab và các bộ chương trình ứng dụng của nó. Và bởi vì Matlab và **Simulink** đã được tích hợp nên ta có thể mô phỏng, phân tích và sửa chữa mô hình trong cả hai môi trường tại bất kỳ thời điểm nào.

7.2. BÀI TOÁN THỨ NHẤT

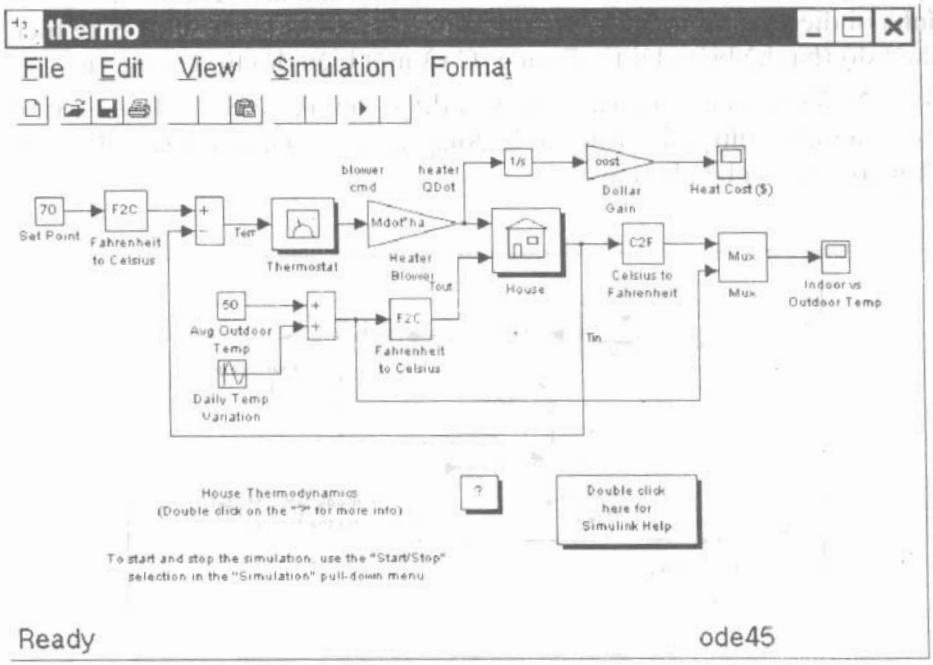
7.2.1 Đặt bài toán cho mô hình

Một ví dụ đáng chú ý của **Simulink** là mô hình nhiệt động học của một ngôi nhà.

Để chạy mô hình này ta thực hiện các bước dưới đây:

1. Chạy Matlab.

2. Để chạy mô hình ta đánh "Thermo" trong cửa sổ lệnh của Matlab. Lệnh này sẽ chạy **Simulink** và tạo ra một cửa sổ chứa mô hình sau (hình 7.1)



Hình 7.1 *Sơ đồ mô hình mô tả bằng Simulink*

Khi xem mô hình, **Simulink** sẽ đưa ra hai khối hiển thị có tên "Indoor vs Outdoor Temp" và " Heat cost".

3. Để bắt đầu mô phỏng, vào menu **Simulation** và chọn lệnh **Start** (Hoặc ấn phím **Start** trên thanh công cụ của cửa sổ **Simulink**). Khi chạy mô phỏng, nhiệt độ trong và ngoài nhà sẽ hiển thị trong khối Scope "Indoor vs Outdoor Temp" và số tiền phải trả sẽ xuất hiện trong khối Scope "Heat Cost".

4. Để dừng mô phỏng, chọn lệnh **Stop** trong menu **Simulation** (Hoặc ấn phím **Pause** trên thanh công cụ).

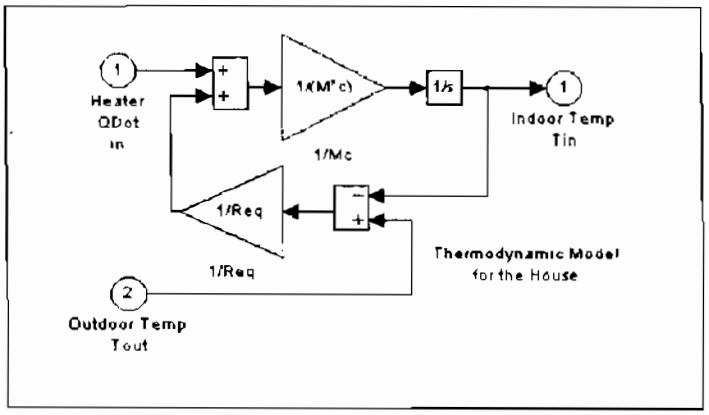
5. Khi đã kết thúc việc chạy mô hình này, đóng mô hình bằng lệnh **Close** từ Menu **File**.

7.2.2 Mô tả mô hình bài toán

Mô hình mô phỏng nhiệt động của ngôi nhà là một mô hình đơn giản. Máy điều nhiệt được đặt tại 70^0 F và bị tác động bởi nhiệt độ bên ngoài biến đổi theo luật hình sin có biên độ là 15^0F xung quanh nhiệt độ 50^0F. Đây là sự mô phỏng diễn biến thay đổi nhiệt độ hàng ngày.

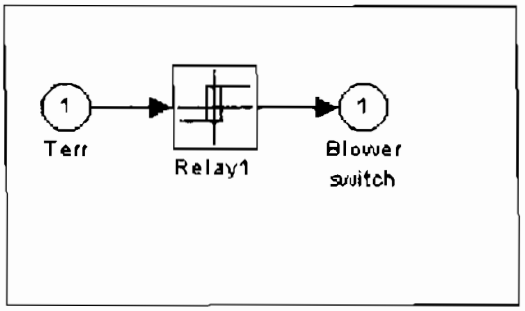
Mô hình sử dụng các hệ con để đơn giản hoá sơ đồ mô hình và tạo ra hệ thống có thể sử dụng được. Hệ con là một nhóm các khối chức năng có tính đại diện. Mô hình này có 5 hệ con: máy điều nhiệt, nhà và 3 hệ biến đổi nhiệt độ (hai hệ biến đổi từ ^{0}F sang ^{0}C và một biến đổi từ ^{0}C sang ^{0}F).

Nhiệt độ bên trong và ngoài nhà được cấp tới hệ con "House", và nó sẽ luôn cập nhật nhiệt độ trong nhà. Nhấp kép vào khối 'House" để xem các khối cơ bản của hệ phụ này.



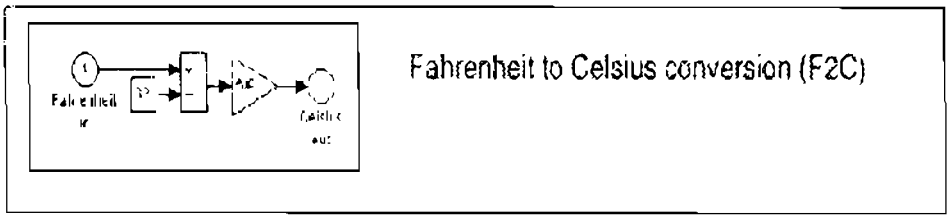
Hình 7.2 *Mô hình nhiệt động của ngôi nhà*

Mô hình hệ con ổn định nhiệt là hoạt động của máy ổn nhiệt, nó quyết định khi nào hệ thống điều nhiệt bật hay tắt. Nháy kép vào khối để xem các cụm chức năng cơ bản của hệ này (hình 7.3).



Hình 7.3 *Mô hình máy ổn nhiệt*

Cả nhiệt độ bên trong và bên ngoài nhà được biến đổi từ ^{0}F sang ^{0}C bởi một hệ con chung có tên là Fahrenheit to Celsius conversion (F2C) (hình 7.4).



Hình 7.4 *Mô hình hệ biến đổi từ độ F sang độ C*

Khi máy điều nhiệt được bật, tiền phải trả sẽ được tính toán và hiển thị trên khối "Heat Cost", nhiệt độ bên trong nhà được hiển thị trên khối "Indoor Temp".

7.2.3 Thử lại một số quá trình

Có một số quá trình mà ta cần thử lại để xem mô hình đáp ứng như thế nào đối với các thông số khác nhau.

- Một khối hiển thị bao gồm vùng hiển thị tín hiệu và điều khiển, nó cho phép ta lựa chọn khoảng tín hiệu hiển thị, phóng to từng phần tín hiệu và thực hiện các công việc khác. Trục hoành biểu diễn thời gian và trục tung biểu diễn giá trị của tín hiệu.
- Khối hằng số có tên là "Setpoint" đặt nhiệt độ yêu cầu trong nhà. Mở khối này ra và đặt giá trị tới 80° F khi đang chạy mô phỏng. Xem nhiệt độ bên trong nhà và số tiền thay đổi. Cũng như vậy ta cũng có thể thay đổi nhiệt độ bên ngoài và xem ảnh hưởng của nó đối với mô hình.
- Điều chỉnh độ biến đổi nhiệt độ hằng ngày bởi việc mở khối phát sóng sin có tên "Daily Temp Variation" và thay đổi thông số biên độ.

7.2.4 Hiệu quả của việc mô phỏng quá trình

Ví dụ này làm sáng tỏ một vài công việc đã được sử dụng để xây dựng mô hình.

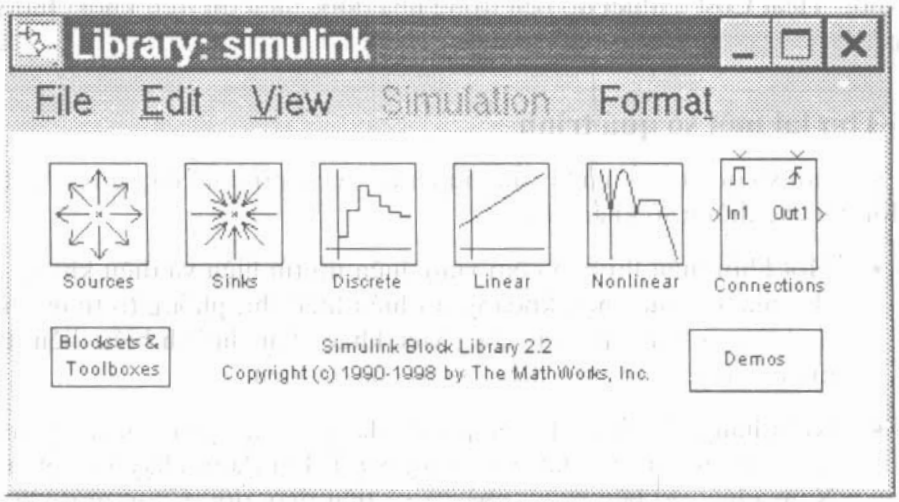
- Chạy mô phỏng bao gồm đặt các thông số và bắt đầu mô phỏng với lệnh **Start**.
- Gói gọn toàn bộ các khối có liên quan trong một khối đơn gọi là hệ con.

- Tạo ra biểu tượng riêng và thiết kế một hộp đối thoại cho một khối công việc sử dụng "masking".Trong mô hình điều nhiệt tất cả các hệ con được tạo ra biểu tượng sử dụng "Masking".
- Khối hiển thị hiển thị ra đồ hoạ như một máy hiện sóng thực sự. Khối hiển thị hiển thị tín hiệu vào của nó.

7.2.5 Các ví dụ sử dụng khác của Simulink

Bạn có thể xem các ví dụ này từ cửa sổ thư viện của Simulink.

1. Đánh "simulink" trong cửa sổ lệnh của Matlab. Cửa sổ thư viện các khối sẽ xuất hiện.

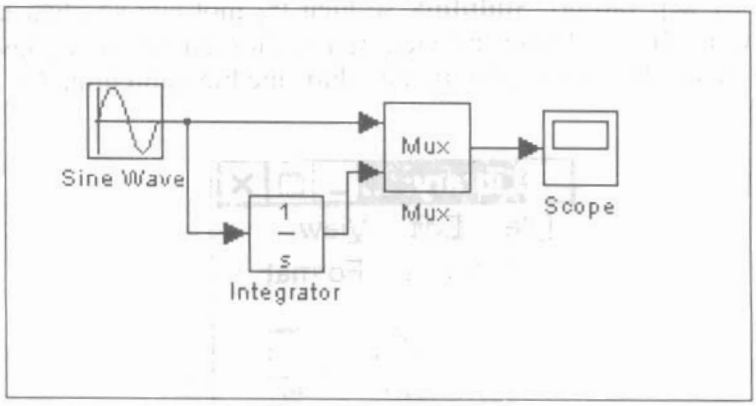


Hình 7.5 *Cửa sổ các thư viện của Simulink*

Nhấn kép vào biểu tượng "Demos". Cửa sổ "Matlab demos" sẽ xuất hiện. Cửa sổ này có một vài ví dụ đáng quan tâm mà nó làm sáng tỏ đặc điểm sử dụng của **Simulink**.

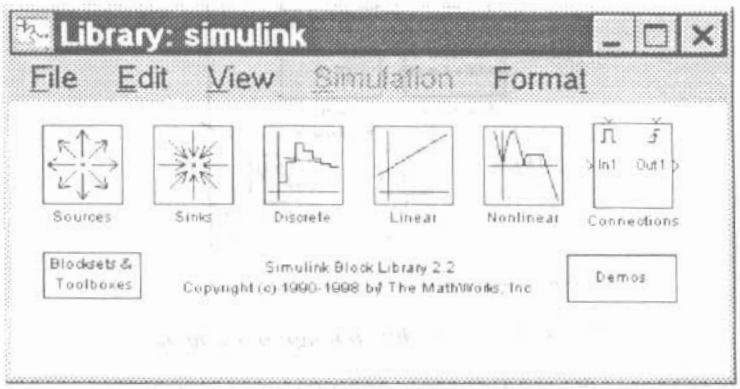
7.3 PHƯƠNG PHÁP XÂY DỰNG MÔ HÌNH

Ví dụ sau sẽ trình bầy cho ta phương pháp xây dựng một mô hình như thế nào, cách sử dụng các lệnh và các thao tác để xây dựng mô hình. Ta sẽ xây dựng mô hình tích phân sóng sin và hiển thị kết quả cùng với sóng sin. Sơ đồ khối của mô hình trình bày trên hình 7.6.



Hình 7.6 *Mô hình tích phân sóng hình sin*

Đánh lệnh "simulink" từ cửa sổ lệnh của Matlab để hiển thị cửa sổ thư viện Simulink và nếu không có cửa sổ mô hình nào được mở thì một cửa sổ mô hình mới được tạo ra. Cửa sổ thư viện của Simulink được chỉ rõ trên hình 7.7.

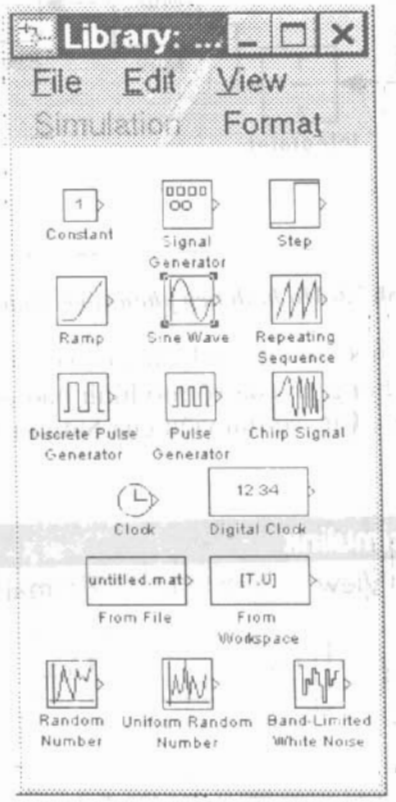


Hình 7.7 *Cửa sổ thư viện của Simulink*

Trong mô hình này bạn lấy các khối sau từ các thư viện:

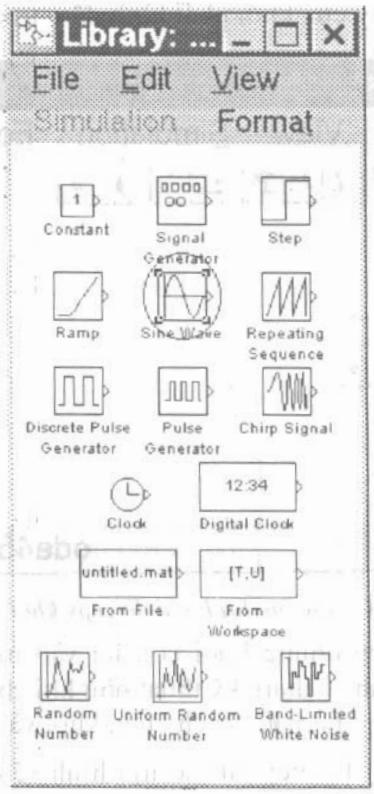
- Thư viện các nguồn tín hiệu (Khối phát sóng sin).
- Thư viện các khối nhận tín hiệu(Khối hiển thị).
- Thư viện các hàm tuyến tính (Khối tích phân).
- Thư viện các đầu nối (Khối chuyển mạch).

Mở thư viện các nguồn tín hiệu để vào khối sóng sin. Để mở một thư viện ta nháy kép vào nó. **Simulink** sẽ hiển thị một cửa sổ chứa tất cả các khối của thư viện đó. Trong thư viện nguồn tín hiệu tất cả các khối đều là nguồn tín hiệu. Thư viện nguồn tín hiệu được thể hiện như hình 7.8.



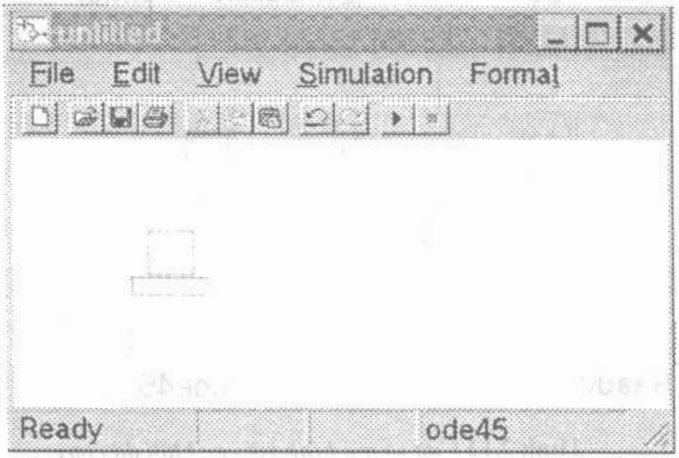
Hình 7.8 *Cửa sổ thư viện nguồn tín hiệu*

Người sử dụng thêm khối vào mô hình của mình bằng cách chép khối đó từ thư viện hay từ mô hình bất kỳ nào khác. Trong ví dụ này bạn cần chép khối phát sóng hình sin. Đặt con trỏ trên khối ấn và giữ phím chuột, kéo khối tới cửa sổ mô hình.



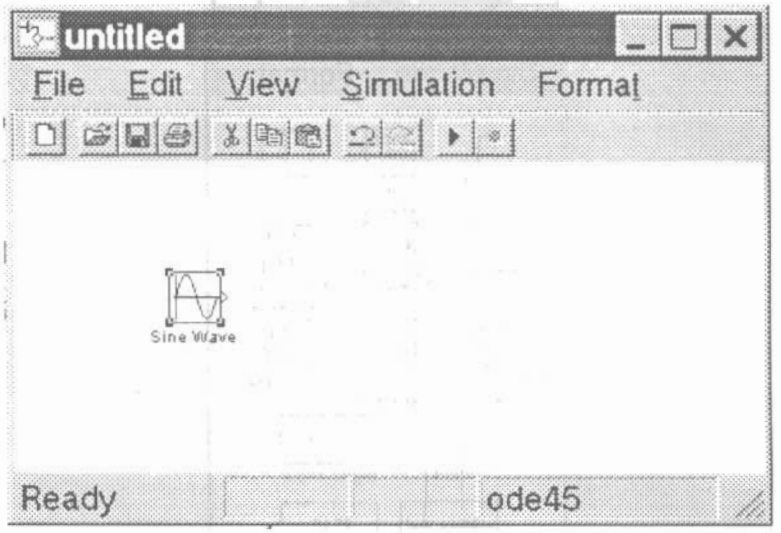
Hình 7.9 *Copy khối sin vào mô hình*

Khi di chuyển khối, bạn có thể thấy khối và tên của nó di chuyển cùng với con trỏ.



Hình 7.10 *Khối và tên khối di chuyển cùng con trỏ*

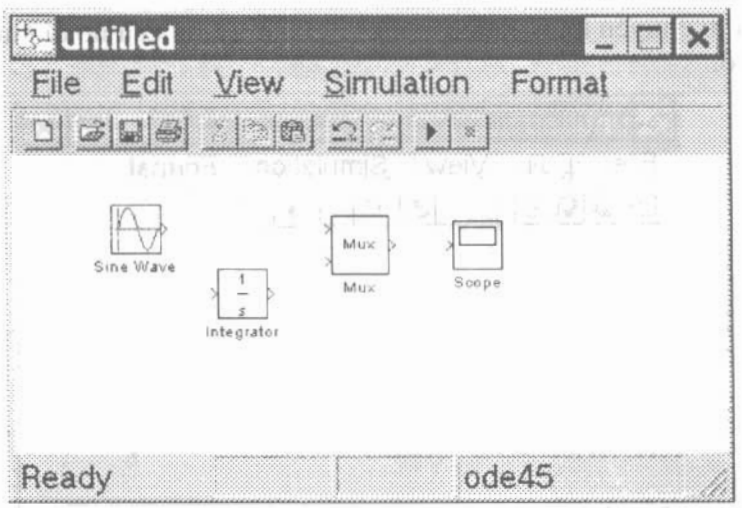
Khi con trỏ tới nơi bạn cần đặt khối trong mô hình bằng cách nhả phím chuột, một bản copy của khối phát hình đã ở trong mô hình của bạn.



Hình 7.11 *Cửa sổ mô hình khi đã copy khối sóng sin*

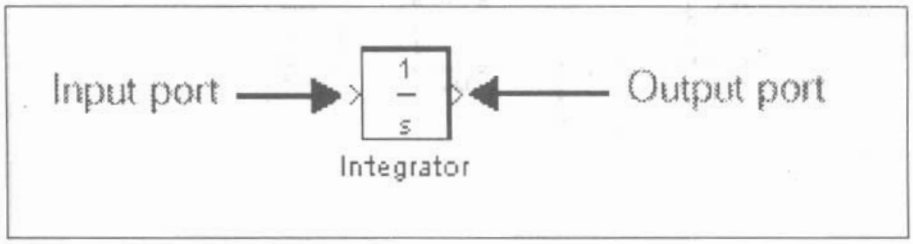
Theo cách này chép những khối còn lại vào mô hình. Bạn có thể di chuyển khối trong mô hình sử dụng kỹ thuật như khi chép khối, hoặc có thể di chuyển khối trong khoảng nhỏ bằng cách chọn khối và ấn các phím mũi tên.

Với tất cả các khối đã chép cửa sổ mô hình sẽ được hiển thị như trên hình 7.12.



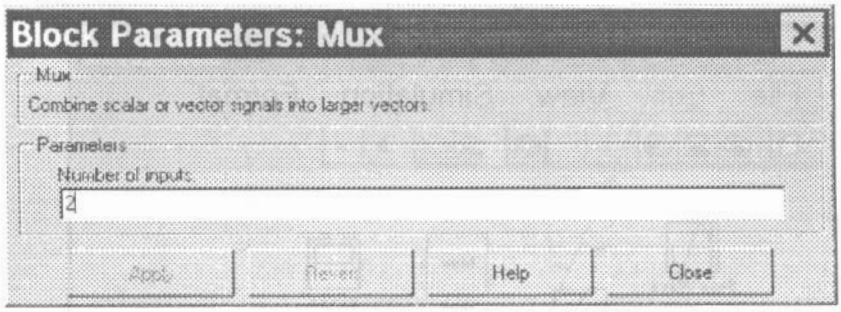
Hình 7.12 *Cửa sổ mô hình với các khối đã copy*

Nếu xem kỹ từng khối, bạn thấy dấu > ở bên phải khối sin và dấu > ở bên trái khối MUX. Dấu ở đầu ra một khối là cổng ra, ở đầu vào một khối là cổng vào. Tín hiệu đi từ đầu ra một khối tới đầu vào khối khác theo một đường nối. Khi một cổng đã được nối thì biểu tượng của cổng cũng mất đi.



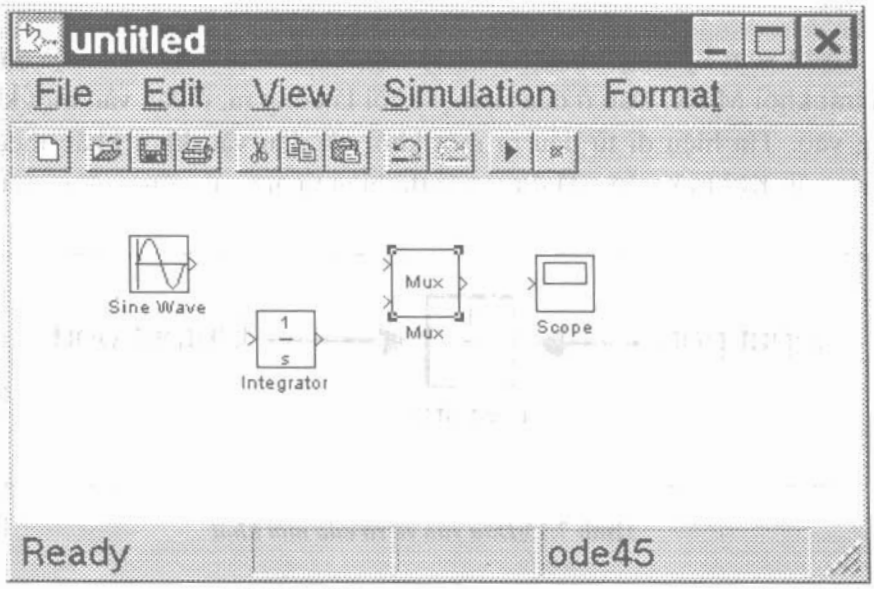
Hình 7.13 *Đầu vào và ra của một khối*

Khối MUX có ba cổng vào nhưng chỉ có 2 tín hiệu vào. Để thay đổi số cổng vào của khối bạn mở khối MUX bằng cách nháy kép trên khối và thay đổi giá trị thông số " Number of Input" bằng 2. Sau đó ấn phím **Close**, **Simulink** sẽ điều chỉnh số cổng vào theo giá trị nhập của người sử dụng.



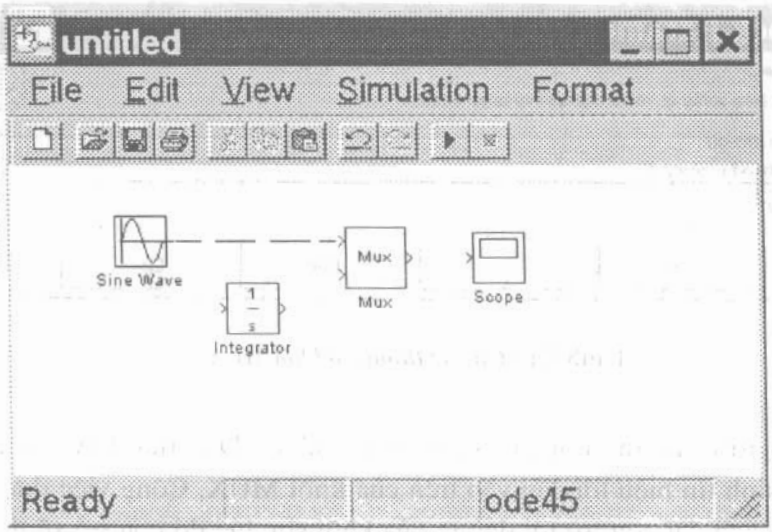
Hình 7.14 *Cửa sổ thông số khối MUX*

Bây giờ ta có thể nối các khối lại với nhau. Đầu tiên hãy nối đầu ra khối phát sinh tín hiệu tới đầu vào trên của khối MUX. Công việc được thực hiện bằng cách đặt con trỏ tới đầu ra của khối sin, lúc đó con trỏ sẽ thay đổi thành một chữ thập nhỏ.



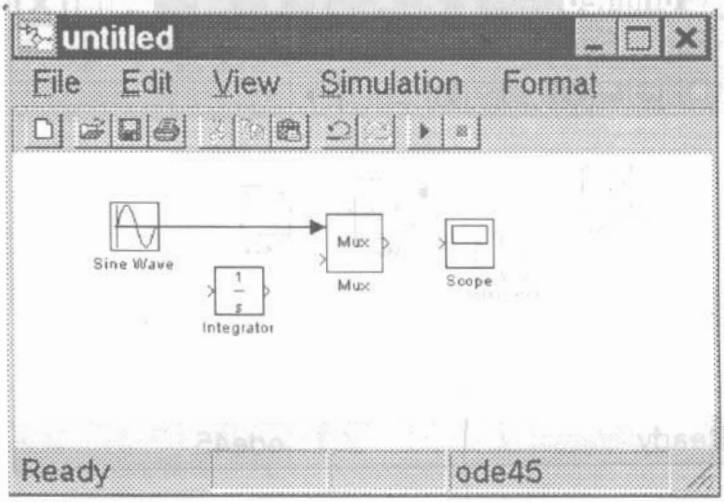
Hình 7.15 *Cửa sổ mô hình trước khi nối dây*

Giữ và kéo chuột tới đầu vào của khối MUX. Chú ý đường là nét đứt khi phím chuột vẫn giữ và con trỏ sẽ thay đổi thành chữ thập kép khi nó lại gần khối MUX.



Hình 7.16 *Cửa sổ mô hình khi đang nối dây*

Bạn có thể nối bằng cách nhả phím chuột khi con trỏ ở bên trong khối. Khi đó đường nối sẽ nối vào cổng gần vị trí con trỏ nhất.

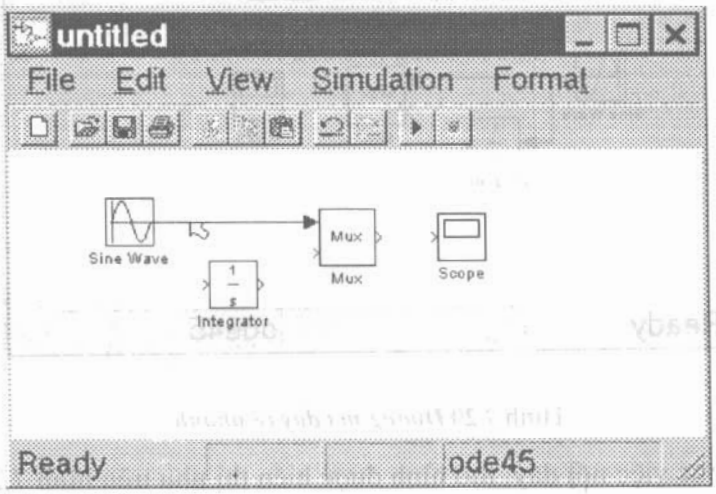


Hình 7.17 *Hình sin nối vào đầu trên khối MUX*

Phần lớn các đường nối đi từ đầu ra của khối tới đầu vào của khối khác. Có đường nối từ một đường nào đó đi tới đầu vào của một khối gọi là đường rẽ nhánh.

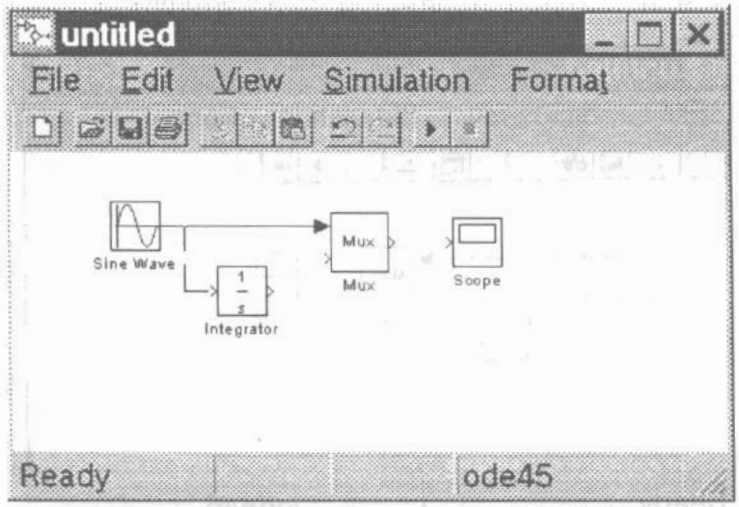
Vẽ đường rẽ nhánh có sự khác biệt nhỏ so với vẽ đường nối chính. Để nối đường đã có, hãy thực hiện theo các bước sau:

1. Đặt vị trí con trỏ ở trên đường cần rẽ nhánh.



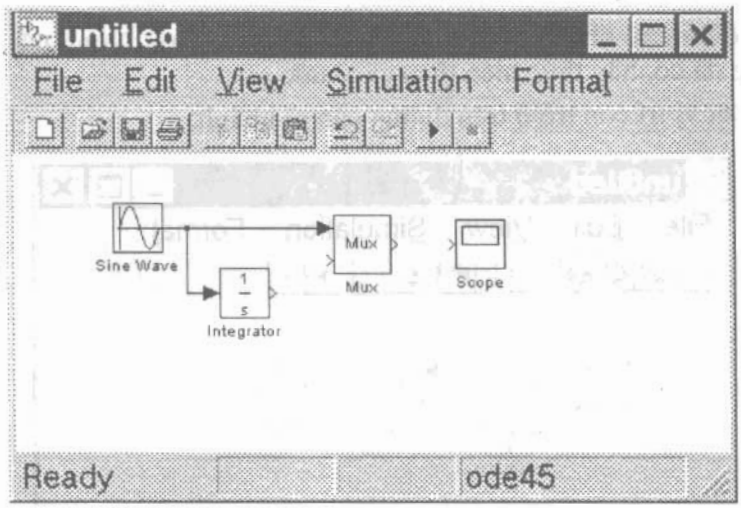
Hình 7.18 *Con trỏ đặt vào điểm cần rẽ nhánh*

2. Ấn và giữ phím **Ctrl**, ấn và giữ phím chuột kéo con trỏ tới đầu vào của khối.



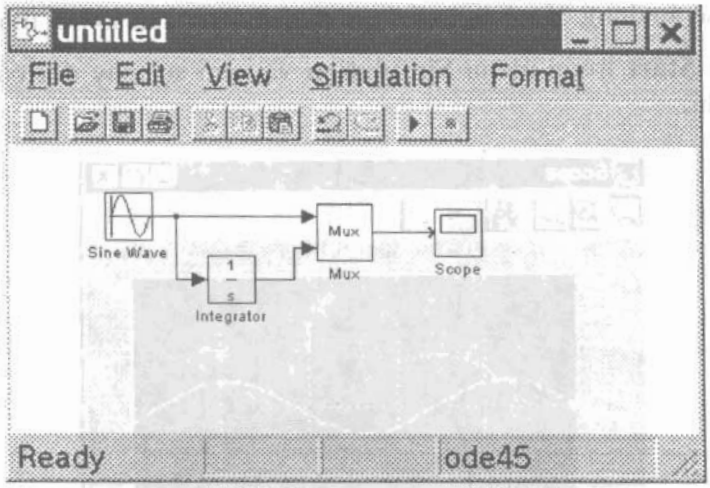
Hình 7.19 *Nối các khối*

3. Nhả phím chuột, **Simulink** sẽ vẽ một đường từ điểm bắt đầu tới cổng vào của khối.



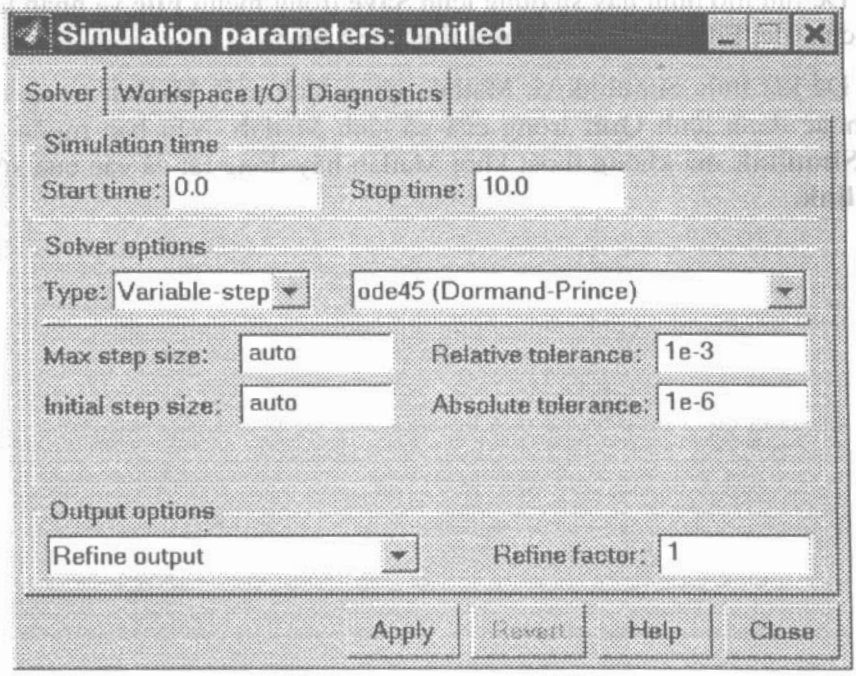
Hình 7.20 *Đường nối dây rẽ nhánh*

Kết thúc việc nối dây, mô hình được hiển thị như trên hình 7.21.



Hình 7.21 *Cửa sổ mô hình khi ta đã vẽ xong*

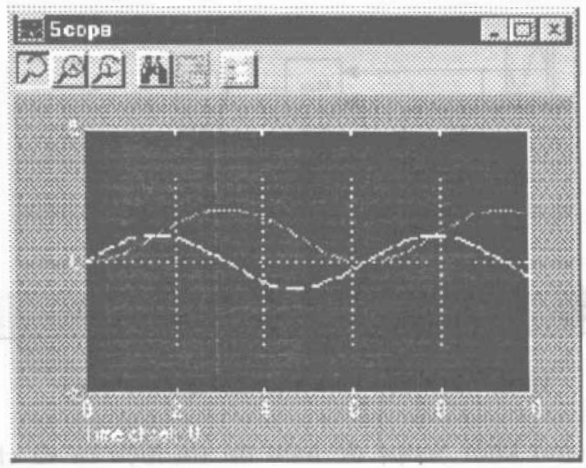
Bây giờ ta mở khối Scope để hiển thị tín hiệu ra và chạy mô phỏng trong 10s. Đầu tiên ta phải đặt thông số mô phỏng bằng lệnh **Parameter** trong menu **Simulation**, hộp hội thoại xuất hiện. Chú ý **Stoptime** đặt là 10.0s.



Hình 7.22 *Hộp đối thoại Parameter của Simulink*

Để đóng hộp đối thoại **Simulink Parameter** ta ấn phím **Close**. **Simulink** sẽ áp dụng các thông số do ta đặt và đóng cửa sổ hội thoại.

Chọn **Start** trong menu **Simulation** và xem sự thay đổi của đầu vào khối hiển thị.



Hình 7.23 *Cửa sổ hiển thị tín hiệu ra khối Scope*

Để lưu mô hình này sử dụng lệnh **Save** trong menu **File** và nhập tên và vị trí của file. File này chứa các mô tả của mô hình.

Để kết thúc Simulink và Matlab, chọn lệnh **exit Matlab** trong menu **File** hoặc đánh lệnh **Quit** trong cửa sổ lệnh Matlab. Nếu bạn muốn thoát khỏi **Simulink** mà không thoát khỏi Matlab hãy đóng tất cả các cửa sổ của **Simulink.**

TÀI LIỆU THAM KHẢO

1. Adrian Biran, Moshe Breiner
   Matlab for Engineers
   Addison-Wesley Publishing Company, 1996
2. Eva Part-Enander, Anders Sjoberg, Bo Melin, Permilla Isaksson
   The Matlab Handbook
   Addison-Wesley Publishing Company, 1996
3. Hadi Saadat
   Computational Aids in Control System Using Matlab
   Mc Graw-Hill, 1993
4. Rabiner L. and Gold B.
   Theory and Application of Digital Precessing
   Englewwood Cliffs, NJ: Prentice-Hall, 1975
5. Ramirez R.
   The FFT-Fundamental and Concepts
   Englewwood Cliffs, NJ: Prentice-Hall, 1985
6. D.M. Etter
   Engineering Problem Solving with Matlab
   Prentice-Hall, 1993
7. B.C. Kuo
   Automatic Control System
   Englewwood Cliffs, NJ: Prentice-Hall, 1987
8. Samuel D. Steams, Don R. Hush
   Digital Signal Analysis
   Englewwood Cliffs, NJ: Prentice-Hall, 1990
9. Nicolas Moreau
   Techniques de compression des signaux
   NASSon Paris, Milan, Barcelone, 1995

10. Gerald Farin, P. Bezier, W. Boehm
Cuves and Surfaces for Computer Aided Geometric Design
Academic Press, Inc. 1990
11. James D. Foley, Andreas van Dam
Computer Graphics - Principle and Practices
Addison-Wesley Publishing Company, Inc. 1996
12. Steven Harrington
Computer Graphics - Programming Approach
Mc Graw-Hill, 1986.

MỤC LỤC

***Chương 4.* Đồ hoạ hai chiều trong Matlab**

***Chương 5.* Đồ hoạ trong không gian ba chiều**

MATLAB VÀ ỨNG DỤNG

Tác giả: **Phạm Thị Ngọc Yến**
Ngô Hữu Tình
Lê Tấn Hùng
Nguyễn Thị Lan Hương

Chịu trách nhiệm xuất bản : **Pgs. Pts. TÔ ĐĂNG HẢI**
Biên tập : NGỌC KHUÊ, ĐĂNG DẬU
Trình bày và chế bản : TIỆP CẦM
Vẽ bìa :

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT
70 Trần Hưng Đạo - Hà Nội

In 500 cuốn khổ 16 x 24 cm, tại công ty in Hàng không .
Giấy phép xuất bản số : 1104 - 28 – 18/8/2004.
In xong và nộp lưu chiểu tháng 1/2005