TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA ĐIỆN TỬ VIỄN THÔNG

TÀI LIỆU THỰC HÀNH

XỬ LÝ SỐ TÍN HIỆU

Biên soạn: Th.s Nguyễn Thị Thu Hằng

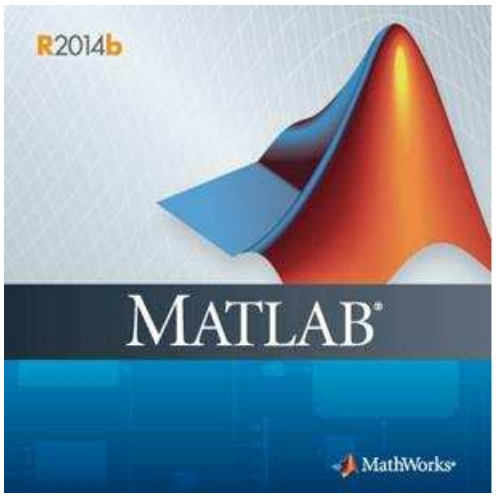
TP.HCM - 2017

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.

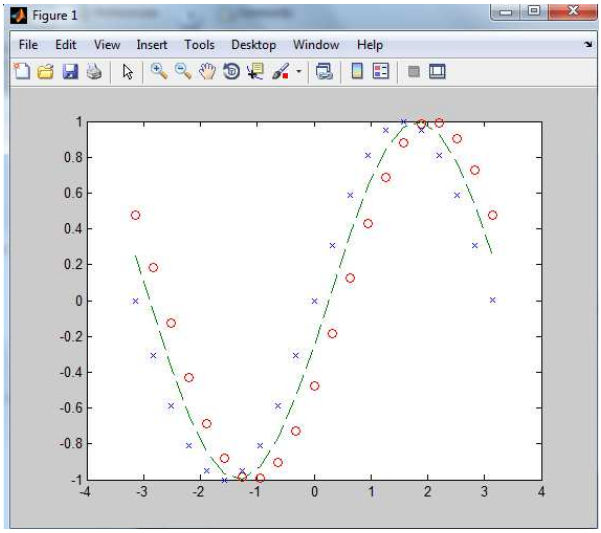
```
>>t = [0:pi/100:2*pi];
>>y = sin(t);
>>plot(t,y); % Vẽ hình sin từ 0->2π
>>grid on
```

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:**

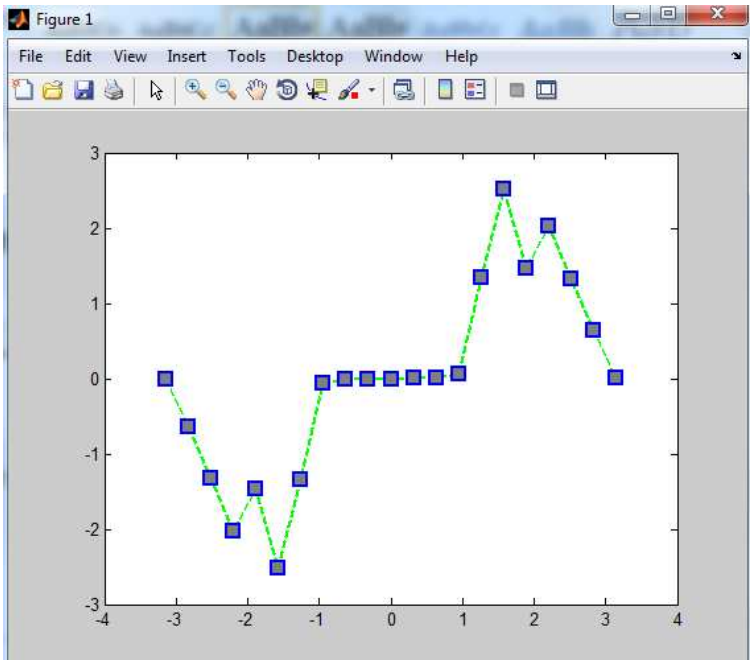
+ dấu cộng **.** điểm **o** vòng tròn **x** chữ thập ***** dấu sao

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
```

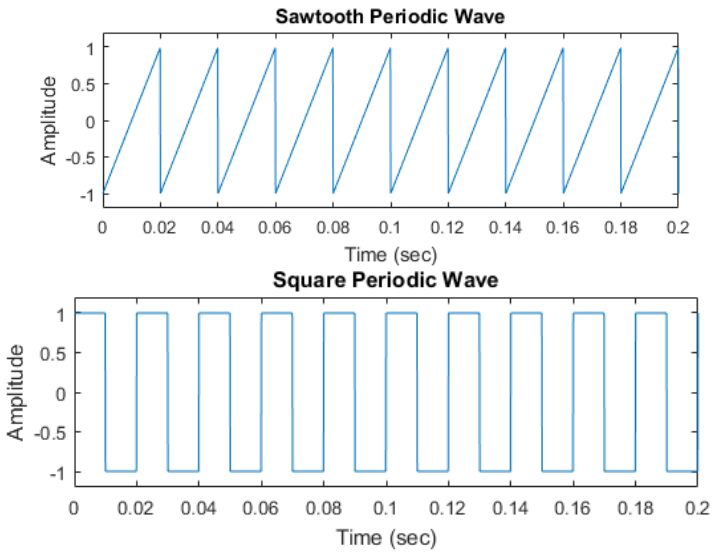
Lưu tên file là ***gptb2.m*** và kiểm tra kết quả:

```
>>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)
```

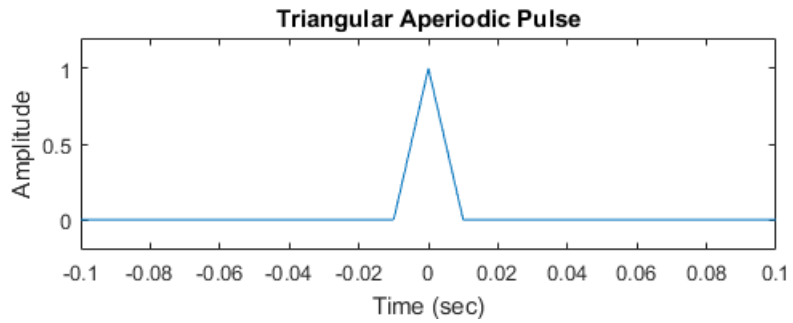
Bài 1.8: Hàm **sawtooth** tạo ra sóng răng cưa với đỉnh tại +/- 1 và hàm square tạo sóng xung vuông

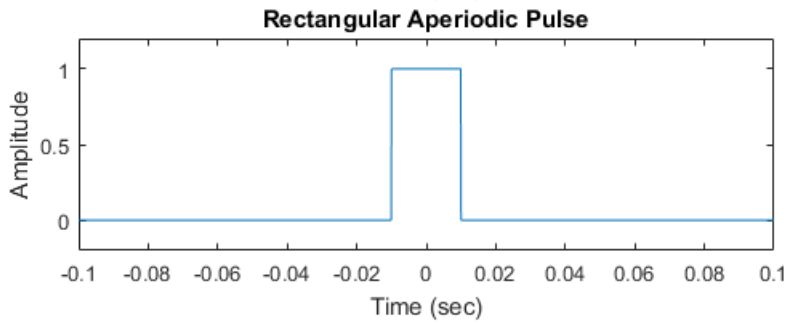
```
fs = 10000;
t = 0:1/fs:1.5;
x1 = sawtooth(2*pi*50*t);
x2 = square(2*pi*50*t);
subplot(211),plot(t,x1), axis([0 0.2 -1.2 1.2])
xlabel('Time (sec)');
ylabel('Amplitude');
title('Sawtooth Periodic Wave');
subplot(212)
plot(t,x2)
axis([0 0.2 -1.2 1.2]);
xlabel('Time (sec)');
ylabel('Amplitude');
title('Square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



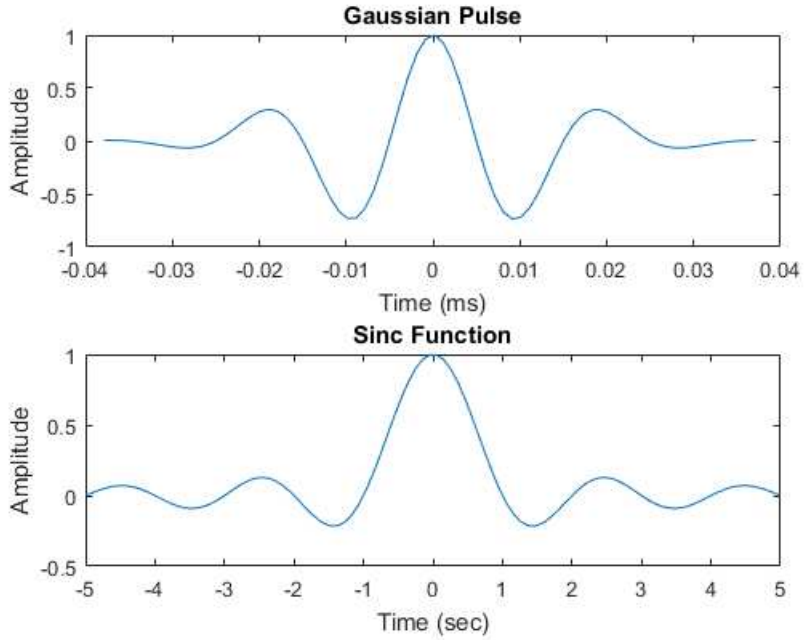


Bài 1.10: Hàm **gauspuls** và hàm **sinc**

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

Vẽ hàm Sa

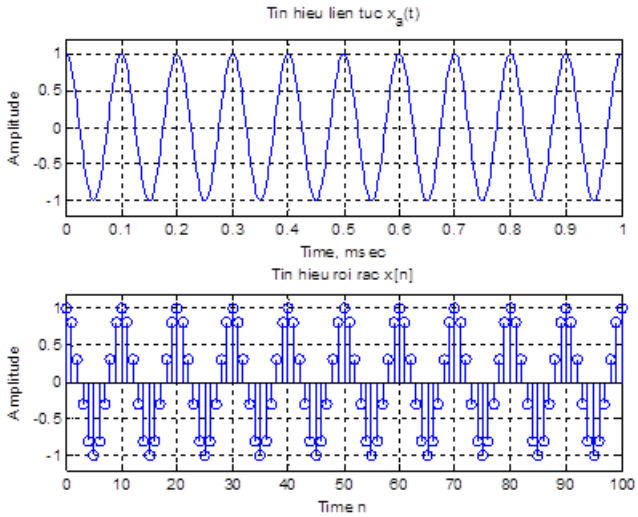
```
tc = gauspuls('cutoff',50e3,0.6,[],-40);
t1 = -tc : 1e-6 : tc;
y1 = gauspuls(t1,50e3,0.6);
t2 = linspace(-5,5);
y2 = sinc(t2);
subplot(211)
plot(t1*1e3,y1);
xlabel('Time (ms)');
ylabel('Amplitude');
title('Gaussian Pulse');
subplot(212),plot(t2,y2);
xlabel('Time (sec)');
ylabel('Amplitude');
title('Sinc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



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

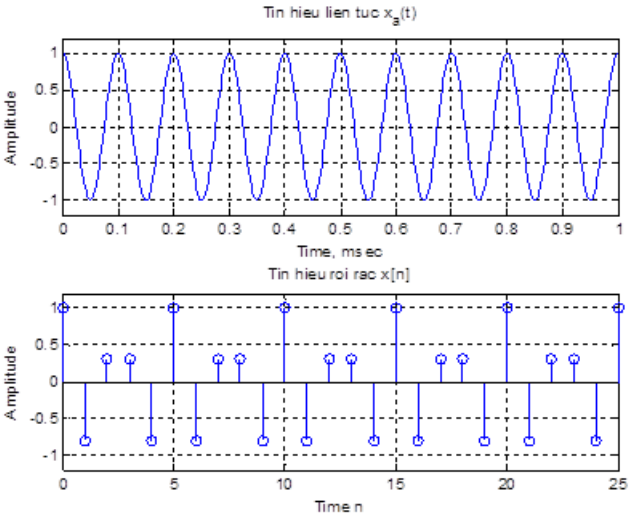
```
clf;
t = 0:0.0005:1;
f =10 ;
xa = cos(2*pi*f*t);
subplot(2,1,1)
plot(t,xa);grid
xlabel('Time, msec');ylabel('Amplitude');
title('Tin hieu lien tuc x_{a}(t)');
axis([0 1 -1.2 1.2])
subplot(2,1,2);
T = 0.01; % tan so lay mau tin hieu
n = 0:T:1;
xs = cos(2*pi*f*n);
k = 0:length(n)-1;
stem(k,xs); grid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

CHƯƠNG 2: TÍN HIỆU RỜI RẠC THEO THỜI GIAN

Nội dung chính:

- *Biểu diễn các tín hiệu rời rạc cơ bản.*
- *Thực hiện các phép toán đơn giản.*
- *Tính năng lượng của tín hiệu.*
- *Xác định các tính chất của hệ rời rạc.*

Các hàm Matlab liên quan:

♣ **stemp**: vẽ dãy dữ liệu như các que theo trục x

♣ **sum**: Xác định tổng của tất cả các phần từ của một vector

♣ **min**: Xác định phần tử nhỏ nhất của một vector

♣ **max**: Xác định phần tử nhỏ nhất của một vector

♣ **zeros**: cấp phát một vector hoặc ma trận với các phần tử 0

♣ **subplot**: Chia đồ thị ra thành nhiều phần nhỏ, mỗi phần vẽ một đồ thị khác nhau

♣ **title**: Thêm tên tiêu đề cho đồ thị

♣ **xlabel**: Viết chú thích dưới trục x trong đồ thị 2D

♣ **ylabel**: Viết chú thích dưới trục y trong đồ thị 2D

♣**Hàm impz(num, den, N+1):** Hàm xác định đáp ứng xung đơn vị của một hệ thống

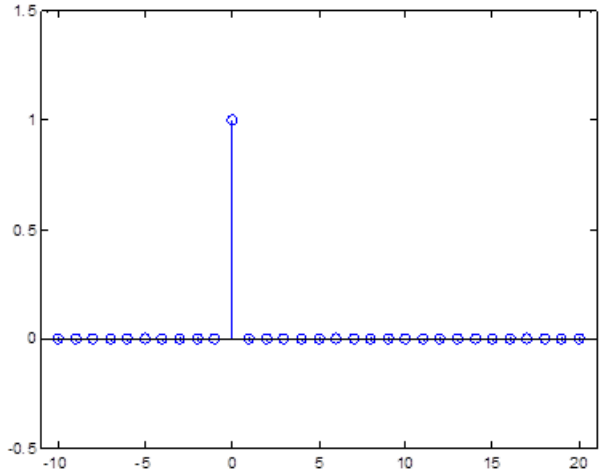
♣ **Hàm filter(num, den, x, ic):** lọc dữ liệu với mạch lọc IIR hoặc FIR

2.1 CÁC TÍN HIỆU CƠ BẢN

Bài 2.1 Hàm xung đơn vị:

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

```
n = -10:20;
delta=[zeros(1,10) 1 zeros(1,20)];
stem(n,delta);
axis([-11 21 -0.5 1.5]);
```



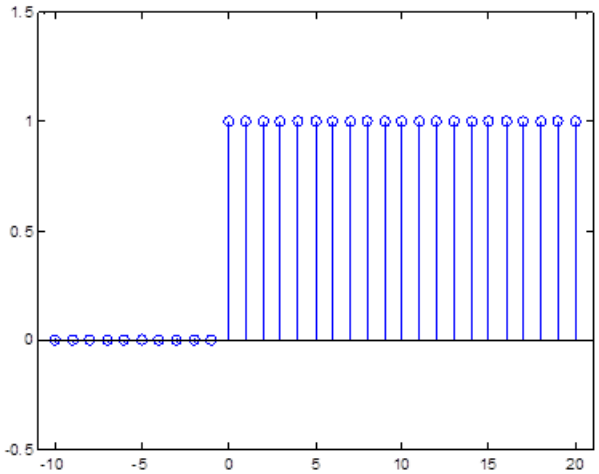
Hình 2.1 Hàm xung đơn vị

Bài 2.2. Tín hiệu hàm bước nhảy đơn vị u(n).

Hàm bước nhảy đơn vị:

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

```
n=-10:20;
u=[zeros(1,10) ones(1,21)];
stem(n,u);
axis([-11 21 -0.5 1.5]);
```

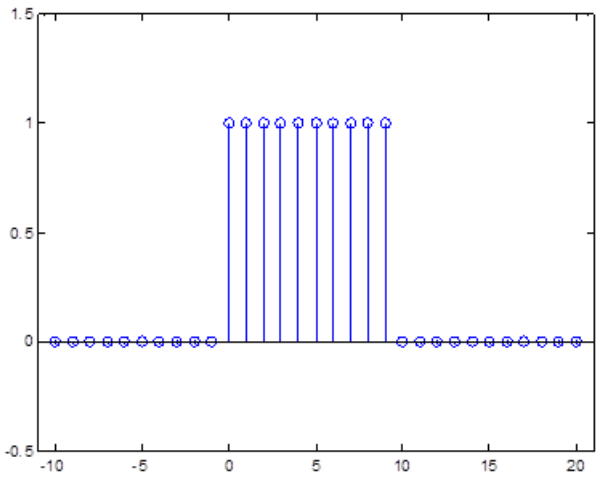


Hình 2.2 Hàm bước nhảy đơn vị

Bài 2.3: Tạo dãy xung chữ nhật

$$rect_N(n)=\begin{cases}1: \text{N-1}\geq n\geq 0\\ 0: n\end{cases}$$

```
n=-10:10;
L=10;% dãy xung có chieu dai N=10
rec=[zeros(1,10) ones(1,L) zeros(1,20-L+1)];
stem(n,rec);
axis([-11 21 -0.5 1.5]);
```

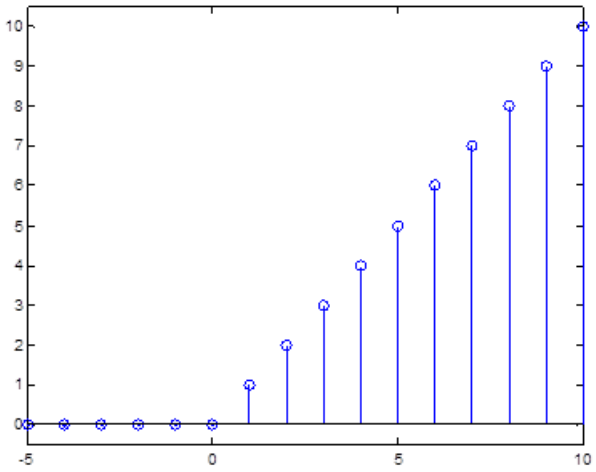


Hình 2.3 Dãy xung chữ nhật rect10(n)

Bài 2.4 : Tạo dãy dốc đơn vị

$$r(n)=\begin{cases}n: n\geq 0\\ 0: n<0\end{cases}$$

```
n=-5:10;
m=[zeros(1,5) 0:10];
u=[zeros(1,5) ones(1,11)];
r=m.*u;
stem(n,r);
axis([-5 10 -0.5 10.5]);
```



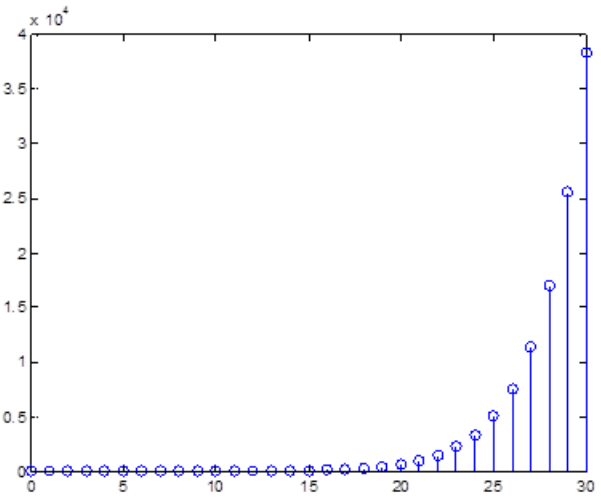
Hình 2.4 Hàm dốc đơn vị

Bài 2.5. Tín hiệu hàm mũ thực

Hàm mũ:

$$x(n) = \begin{cases} a^n & n \geq 0 \\ 0 & n < 0 \end{cases} = a^n u(n)$$

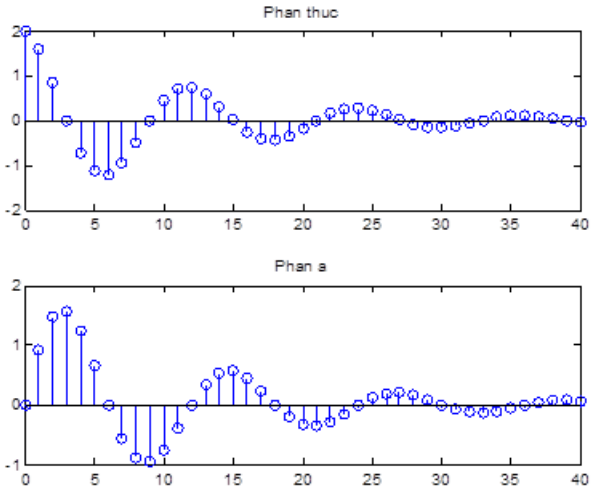
```
n=0:30;
a=1.5;
K=0.2;
x=K*a.^n; %Hàm mũ x= K.a^n
stem(n,x);
```



Hình 2.5 Hàm mũ thực

Bài 2.6. Tín hiệu hàm mũ phức

```
n=0:30;
s=-(1/12)+(pi/6)*i;
K=2;
n=0:40;
x=K*exp(s*n);
subplot(211);
stem(n,real(x));
title('Phan thuc');
subplot(212);
stem(n,imag(x));
title('Phan ao');
```



Hình 2.6 Hàm mũ phức

Bài 2.7 Tạo dãy xung vuông và dãy xung răng cưa tuần hoàn có chiều dài L, biên độ đỉnh A, chu kỳ N

```
A=input('Bien do dinh='); %A=3
L=input('Chieu dai day=');%100
N=input('Chu kỳ của day=');%15%
Fs=input('tan so lay mau mong muon=');%20kHz
DRX=input('Do rong cua xung vuong=');%60
```

```
Ts=1/Fs;
t=0:L-1;
x=A*sawtooth(2*pi*t/N);
y=A*square(2*pi*t/N,DRX);
subplot(211);
stem(t,x);
xlabel(['Thoigian',num2str(Ts),'giay']);
ylabel('Bien do');
title('Day xung rang cua');
subplot(212);
stem(t,y);
xlabel(['Thoigian',num2str(Ts),'giay']);
ylabel('Bien do');
title('Day xung vuong');
```

Kết quả mô phỏng cho các thông số của tín hiệu như sau :

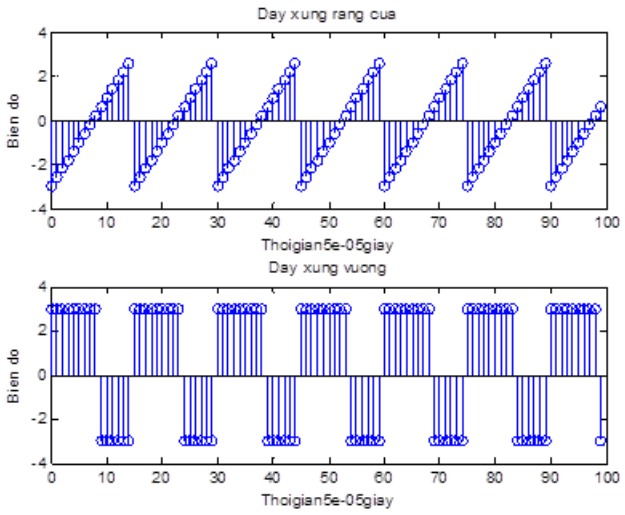
Bien do dinh=3

Chieu dai day=100

Chu ky cua day=15

tan so lay mau mong muon=20000

Do rong cua xung vuong=60



Hình 2.7 Dãy xung răng cưa và dãy xung vuông

Bài 2.8 Tạo tín hiệu hình sin $y(n) = A\cos(\omega_0 n + \varphi)$

```
A=input('Bien do dinh=');
L=input('Chieu dai day=');
omeg=input('Tan so goc=');
if ((omeg<=0)|(omeg>=pi)),error('Tan so goc khong
hop le');end;
pha=input('Goc pha=');
if ((pha<0)|(pha>(2*pi))),error('Pha khong hop
le');end;
n=0:L-1;
arg=omeg*n-pha;
x=A*cos(arg);
stem(n,x);
axis([0 50 -2.5 2.5]);
title('Day sin tuan hoan');
xlabel('Thoi gian roi rac n');
ylabel('Bien do');
```

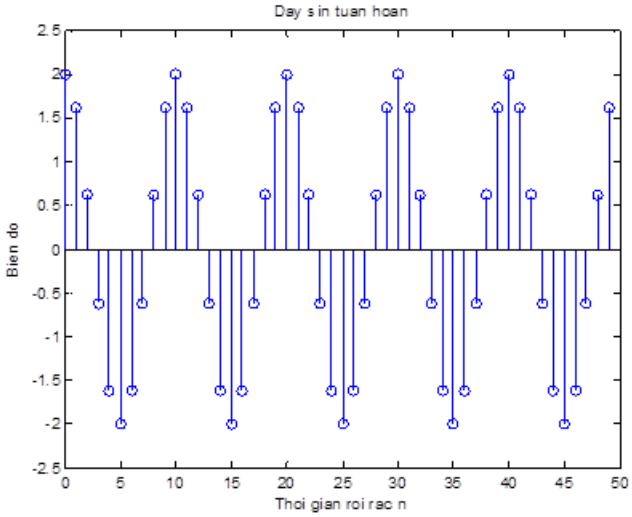
Kết quả:

Bien do dinh=2

Chieu dai day=50

Tan so goc=0.2*pi

Goc pha=0



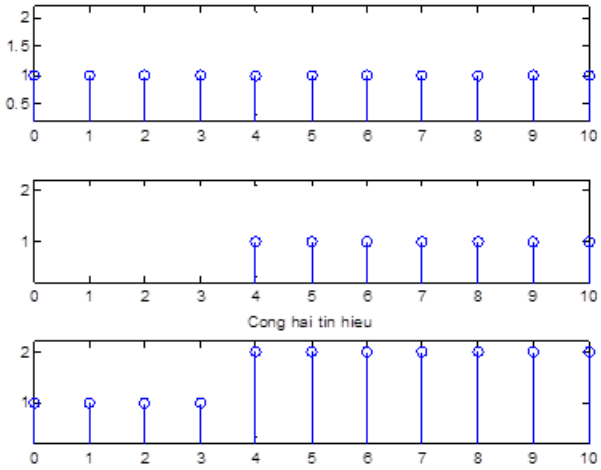
Hình 2.8 Dãy xung tín hiệu hình sin

2.2 CÁC PHÉP TOÁN TRÊN TÍN HIỆU

- Dịch: x(n) k xung: x(n – k)
- Ảnh gương: x(n) x(-n)
- Co trên miền thời gian: x(n) : x(an)
- Cộng: y(n) = x1(n) + x2(n)
- Nhân: y(n) = x1(n)x2(n)
- Co biên độ: y(n) = Ax(n)

Bài 2.9. Cộng hai tín hiệu cộng :

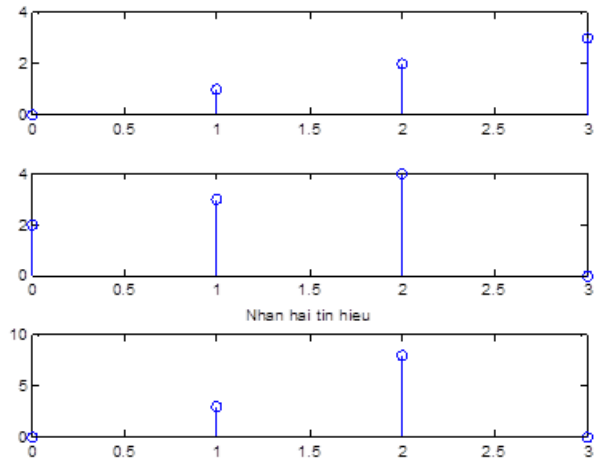
```
n=0:10;
h1=[ones(1,11)];
h2=[zeros(1,4) ones(1,7)];
h=h1+h2;
stem(n,h);
```



Hình 2.9 Cộng hai tín hiệu bước nhảy đơn vị và dãy xung chữ nhật

Bài 2.10. Nhân hai tín hiệu :

```
x1=[0 1 2 3];
x2=[2 3 4 0];
x=x1.*x2;
n=0:length(x)-1;
stem(n,x);
```

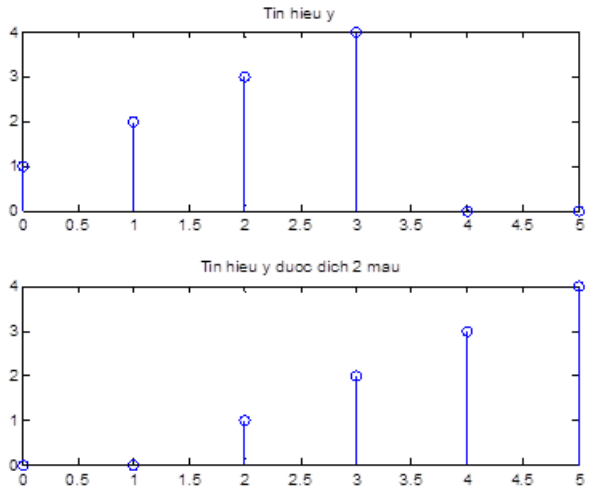


Hình 2.10 Nhân hai tín hiệu bài 2.10

Bài 2.11 Dịch tín hiệu y(n) thành y(n-d)

```
d=2;
```

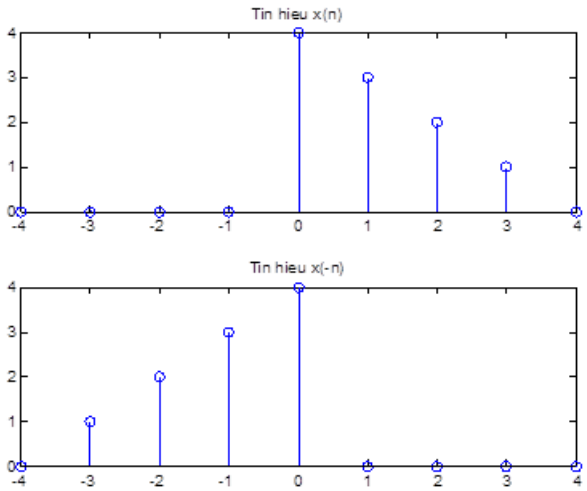
```
y=[1 2 3 4];
yd=[zeros(1,d) y];
```



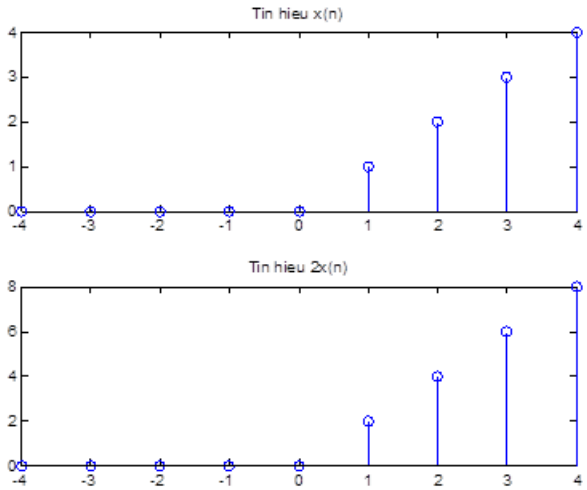
Hình 2.11 Dịch phải xung y(n) 2 đơn vị

Bài 2.12. Vẽ các tín hiệu x(-n), 2x(n)

```
x=[0 1 2 3 4];
n=0:length(x)-1;
y=fliplr(x);
m=-fliplr(n);
subplot(211); stem(n,x);
subplot(212); stem(m,y);
```



Hình 2.12 Tín hiệu x(n) và x(-n)



Hình 2.13 Tín hiệu x(n) và 2x(n)

Bài 2.13. Cộng và nhân 2 tín hiệu. Ví dụ thực hiện cho 2 tín hiệu $x(n)=\{1,-1,2,3,-2\}$ và $x(n)=\{-2,-2,1,1,-4\}$

```
x1=[1 -1 2 3 -2]
x2=[-2 -2 1 1 -4]
y1=x1+x2
y2=x1.*x2
```

Kết quả:

```
y1 = -1    -3     3     4    -6
Y2 = -2     2     2     3     8
```

2.3 KIỂM TRA TÍNH CHẤT TUYẾN TÍNH VÀ BẤT BIẾN

Hệ thống H bất biến theo thời gian nếu và chỉ nếu:

$$y(n) = H[x(n)] \rightarrow y(n-k) = H[x(n-k)]$$

Hệ thống là tuyến tính nếu và chỉ nếu:

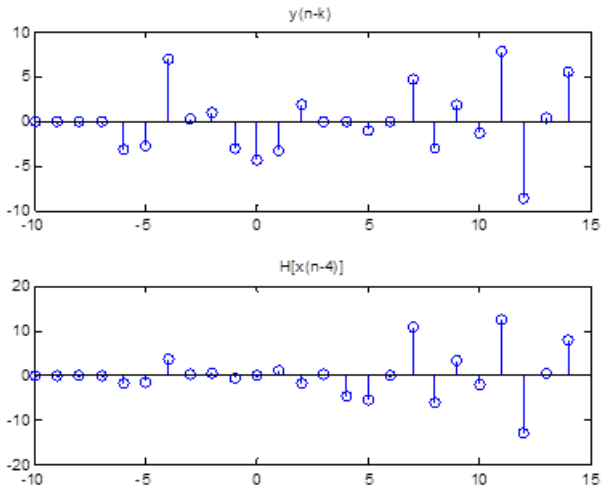
$$H[a_1x_1(n) + a_2x_2(n)] = a_1H[x_1(n)] + a_2H[x_2(n)]$$

Bài 2.14. Xét hệ thống y(n) = nx(n).

```
n = -10:10;
```

```
x = randn(size(n)); %Tín hiệu x ngẫu nhiên
y = n.*x; %y(n) = nx(n)
ynk = [0 0 0 0 y]; %Dịch phải y(n) 4 mẫu -> y(n – 4)
x1 = [0 0 0 0 x]; %Dịch phải x(n) 4 mẫu
n1 = [n 11:14];          % Bổ sung them giá trị cho n
yn = n1.*x1;    % yn = H[x(n – 4)]
subplot(211), stem(n1,ynk), title('y(n – k)');
subplot(212), stem(n1,yn), title('H[x(n – k)]');
```

Kiểm tra tính chất bất biến theo thời gian của y(n) = nx(n).



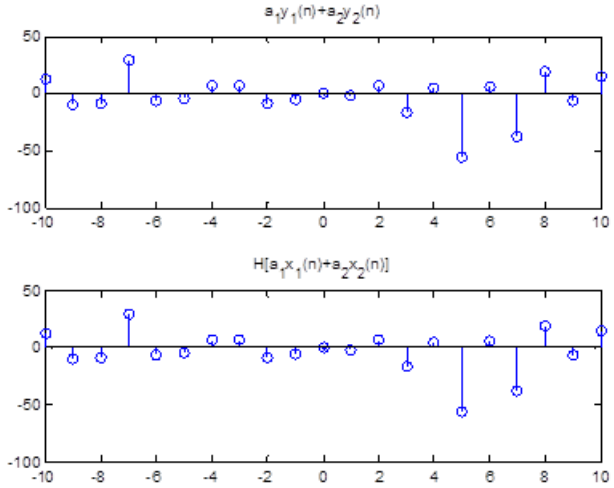
Hình 2.14 Tính chất không bất biến theo thời gian của y(n) = nx(n)

Bài 2.15. Xét hệ thống y(n) = nx(n).

```
clf
n = -10:10;
x1 = randn(size(n)); %Tín hiệu x1 ngẫu nhiên
x2 = randn(size(n)); %Tín hiệu x2 ngẫu nhiên
a1 = 3; a2 = -2; %a1, a2 tùy ý
y1 = n.*x1;
y2 = n.*x2;
y = n.*(a1*x1 + a2*x2);
subplot(211), stem(n,a1*y1+a2*y2);
```

```
title('a_1y_1(n)+a_2y_2(n)');
subplot(212), stem(n,y);
title('H[a_1x_1(n)+a_2x_2(n)]');
```

Kết luận về tính chất tuyến tính của y(n) = nx(n).



Hình 2.15 Tính chất tuyến tính của y(n) = nx(n).

2.4 Hệ LTI

Phương trình sai phân:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

Bài 2.16. Xét hệ thống có phương trình sai phân:

y(n) = 0.3x(n) + 0.2x(n – 1) – 0.3x(n – 2) -0.9y(n – 1) + 0.9y(n – 2).

Xác định đáp ứng xung đơn vị của hệ thống.

```
N = 40;
num = [0.3 0.2 -0.3];
den = [1 0.9 -0.9];
h = impz(num,den,N);% N: số lượng mẫu tính toán
stem(h);
```

Xác định ngõ ra khi biết đáp ứng xung và ngõ vào:

```
x = randn(1,10);
y = conv(x,h);
subplot(311),stem(x);
subplot(312),stem(h);
subplot(313),stem(y);
```

Bài trên sử dụng hàm **conv** để tính tích chập giữa hai tín hiệu rời rạc x và h.

Bài 2.17. Kiểm tra tính giao hoán và kết hợp:

```
h1 = [1 2 -2 -3]; Hệ thống 1
h2 = [-2 0 3 1]; Hệ thống 2
h = conv(h1,h2);
N = 30;
x = randn(1,N);
y11 = conv(x,h1);
y1 = conv(y11,h2);
y21 = conv(x,h2);
y2 = conv(y21,h1);
y = conv(x,h);
subplot(311),stem(y1);
title('y(n) = (x*h_1(n))*h_2(n)');
 subplot(312),stem(y2);
 title('y(n) = (x*h_2(n))*h_1(n)');
 subplot(313),stem(y);
 title('y(n) = x*(h_1(n)*h_2(n)');
```

Bài 2.18 Tính tương quan chéo của hai tính hiệu

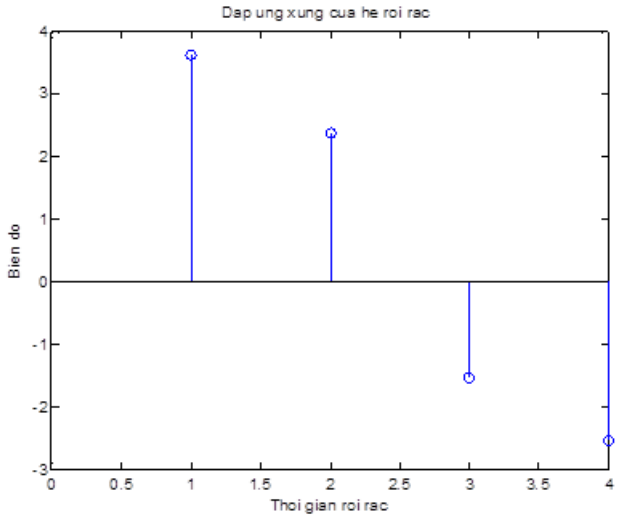
```
x=[1 2 3 2 1];
y=[1 -1 1 -1];
N1=length(y)-1;
N2=length(x)-1;
rxy=conv(x,fliplr(y));
```

```
k=(-N1):N2';
n1=0:N1;n2=0:N2;
subplot(311);
stem(n2,x);
subplot(312);
stem(n1,y);
subplot(313)
stem(k,rxy)
```

Bài này tính tương qua chéo của hai tín hiệu rời rạc dùng lệnh **conv** và **fliplr.**

Bài 2.19 Dùng hàm impz vẽ đáp ứng xung của hệ từ phương trình sai phân

```
N=input('Chieu dai dap ung xung mong muon');
p=input('Gia tri vector p=');
d=input('Gia tri vector d=');
[h,t]=impz(p,d,N;);
disp(h);
n=0:N-1;
stem(n,h);
xlabel('Thoi gian roi rac');
ylabel('Bien do');
title('Dap ung xung cua he roi rac ');
Nhập các giá trị:
N = 5
p=[2.25 2.5 2.25];
d=[1 -0.5 0.75];
```



Hình 2.16 Đáp ứng xung của hệ

Bài 2.20 Xét tính ổn định của hệ thống từ đáp ứng xung rời rạc h(n) bài 2.19

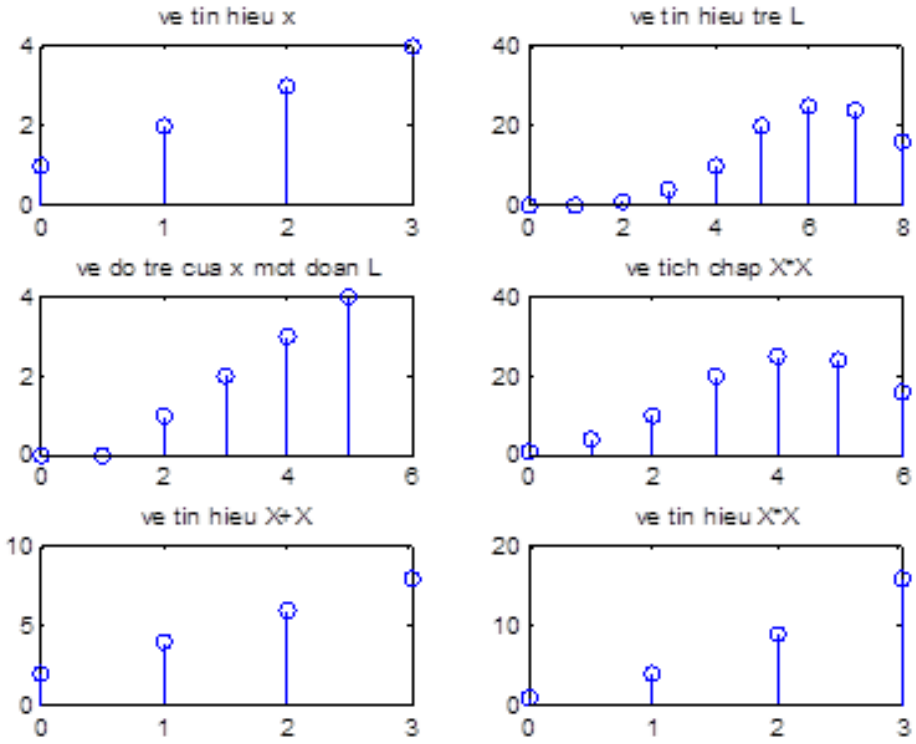
```
sum=0;
for k=1:N+1;
sum=sum+abs(h(k));
end
disp('Gia tri tong =');
disp(abs(h(k)));
```

Kết quả:

```
Gia tri tong = 2.2500
```

Suy ra hệ thống ổn định.

Bài 2.21. Chương trình Matlab tính tích chập của 2 dãy có chiều dài hữu hạn. Kết quả của tích chập được trễ đi d mẫu. Vẽ tín hiệu y(n)*h(n) và y(n-d)



Hình 2.17 Kết quả mô phỏng của bài 2.21

Bài 2.22 Cho hai hệ thống có phương trình sai phân hệ số hằng như sau:

Hệ thống 1:

$y[n] = 0.5\,x[n] + 0.27\,x[n-1] + 0.77\,x[n-2]$

Hệ thống 2:

$y[n] = 0.45\,x[n] + 0.5x[n-1] + 0.45\,x[n-2] + 0.53\,y[n-1] - 0.46\,y[n-2]$

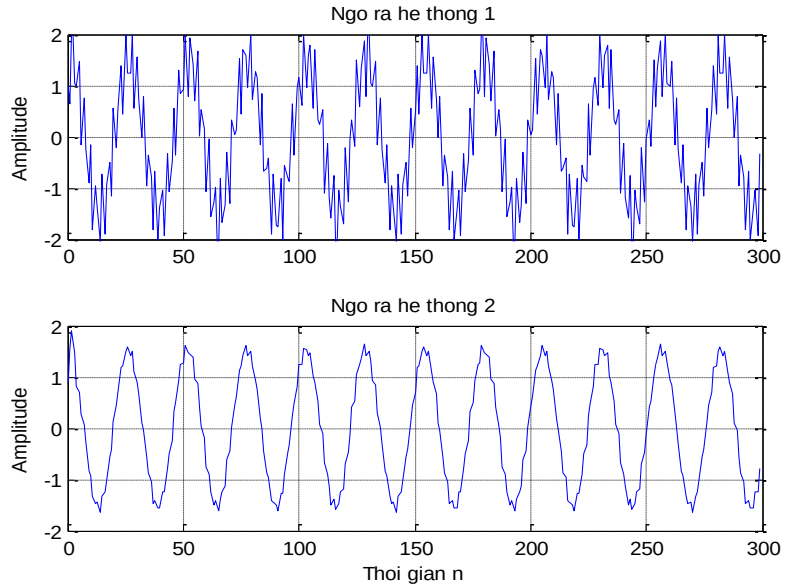
Chương trình Matlab tính đầu ra của hai hệ thống trên với đầu vào:

$$x[n] = \cos\left(\frac{20\pi n}{256}\right) + \cos\left(\frac{200\pi n}{256}\right)$$

n=0:299

```
% bai2_22
clf;
n = 0:299;
x1 = cos(2*pi*10*n/256);
x2 = cos(2*pi*100*n/256);
x = x1+x2;
```

```
% Compute the output sequences
num1 = [0.5 0.27 0.77];
y1 = filter(num1,1,x); % Output of System No. 1
den2 = [1 -0.53 0.46];
num2 = [0.45 0.5 0.45];
y2 = filter(num2,den2,x); % Output of System No. 2
% Plot the output sequences
subplot(2,1,1);
plot(n,y1);axis([0 300 -2 2]);
ylabel('Amplitude');
title('Ngo ra he thong 1');grid;
subplot(2,1,2);
plot(n,y2);axis([0 300 -2 2]);
xlabel('Thoi gian n'); ylabel('Amplitude');
title('Ngo ra he thong 2');grid;
```



Hình 2.18 Kết quả mô phỏng của bài 2.22

2.5 BÀI TẬP

Bài tập 1: Viết chương tình Matlab nhập tín hiệu x từ bàn phím, tìm và vẽ thành phần chẳn và lẽ của tín hiệu x:

$$x_{even}(n) = \frac{1}{2}\left[x(n) + x(-n)\right]$$

$$x_{odd}(n) = \frac{1}{2}\left[x(n) - x(-n)\right]$$

Bài tập 2: Sử dụng Matlab để thực hiện ghép nối hai hệ thống LTI sau

y1(n) + 0.9y1(n–1) + 0.8y1(n–2) = 0.3x(n) – 0.3x(n–1) + 0.4x(n–2)

và

y2(n) + 0.7y2(n–1) + 0.85y2(n–2) = 0.2y1(n) – 0.5y1(n–1) + 0.3y1(n–2)

Cho tín hiệu ngõ vào $x=(0.8)^n u(n)$, xác định và vẽ tín hiệu ngõ ra của hệ.

CHƯƠNG 3: BIẾN ĐỔI Z

Nội dung chính:

 Chuyển đổi tín hiệu trên miền thời gian rời rạc sang miền z.

 Biết các tính chất của biến đổi z.

 Biểu diễn hàm hệ thống của LTI có quan hệ vào – ra là phương trình sai phân hệ số hằng bằng biến đổi z hữu tỉ.

*Các hàm Matlab liên quan:

Signal Processing Toolbox

freqz	impz	residuez	tf2zp	zp2sos
zp2tf	zplane			

3.1 CÁC ĐIỂM CỰC VÀ ĐIỂM KHÔNG

Tóm tắt lý thuyết:

Biến đổi z của tín hiệu rời rạc x(n):

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$$

X(z) là hàm hữu tỉ:

$$X(z) = \frac{N(z)}{D(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}}$$

Giả sử a0 và b0 khác 0:

$$X(z) = \frac{N(z)}{D(z)} = \frac{b_0 z^{-M}}{a_0 z^{-N}} \frac{z^M + \frac{b_1}{b_0} z^{M-1} + \cdots + \frac{b_M}{b_0}}{z^N + \frac{a_1}{a_0} z^{M-1} + \cdots + \frac{a_M}{a_0}}$$

Do N(z) và D(z) là các đa thức theo z nên có thể biểu diễn như sau:

$$X(z) = G z^{N-M} \frac{\prod_{k=1}^{M} (z - z_k)}{\prod_{k=1}^{N} (z - p_k)}$$

Để biểu diễn trên đồ thị, điểm cực được đánh dấu bằng x và điểm không được đánh dấu bằng o.

Bài 3.1. Xác định điểm cực và điểm không dựa vào hàm zplane:

```
num = [1 2 3]; % Tử số
den = [2 4 7]; % Mẫu số
zplane(num,den);
```

Ta có thể vẽ các điểm cực và điểm không nếu đã biết điểm cực và điểm không bằng cách đưa thông số vào hàm **zplane** ở dạng vector cột:

```
zero = [-1 1+j*1];
pole = [j*2 -1+j];
zplane(zero',pole');
```

Để xác định điểm cực và điểm không, ta dùng hàm ***tf2zp***: [z,p,k] = tf2zp(num,den) trong đó z, p là các điểm cực và điểm không lưu dạng vector hàng, k là hệ số khuếch đại:

```
num = [1 2 3]; % Tử số
den = [2 4 7]; % Mẫu số
[z,p,k] = tf2zp(num,den)
```

Nếu đã cho điểm cực và điểm không, ta có thể xác định lại biểu thức của biến đổi z bằng hàm ***zp2tf***: [num,den] = zp2tf(z,p,k) (z, p ở dạng vector cột)

```
zero = [-1 1+j*1];
pole = [j*2 -1+j];
k = 2;
[num,den] = zp2tf(zero',pole',k)
```

Bài 3.2. Chương trình Matlab tính toán và hiển thị các điểm cực, điểm không, tăng tích; tính toán và hiển thị dạng phân số của biến đổi z sau:

$$H(z) = \frac{b_0 + b_1 z^{-1} + ... + b_m z^{-M}}{a_0 + a_1 z^{-1} + ... + a_n z^{-N}}$$

Sử dụng hàm **tf2zp để tìm** các điểm cực, điểm không, tăng tích;

Chương trình còn sử dụng hàm zp2sos cho kết quả là một ma trận Lx6, trong đó các hệ số ma trận là các hệ số của các thành phần hàm bậc hai của công thức (*)

$$\text{SOS} = \begin{bmatrix} b_{01} & b_{11} & b_{21} & 1 & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & 1 & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & 1 & a_{1L} & a_{2L} \end{bmatrix}$$

Nghĩa là:

$$H(z) = g\prod_{k=1}^{L} H_k(z) = g\prod_{k=1}^{L} \frac{b_{0k}+b_{1k}z^{-1}+b_{2k}z^{-2}}{1+a_{1k}z^{-1}+a_{2k}z^{-2}}$$

```
num=input('nhap cac he so o tu =');
den=input('nhap cac he so o mau so=');
[z,p,k]=tf2zp(num,den);grid;
m=abs(p);
disp('cac diem cuc tai');disp(z);
disp(' cac diem khong tai');disp(p);
disp(' he so tang ic'); disp(k);
disp('ban kinh diem cuc'); disp(m);
sos=zp2sos(z,p,k);
disp(' cac thanh phan bac 2:'); disp(real(sos));
zplane(num,den);
```

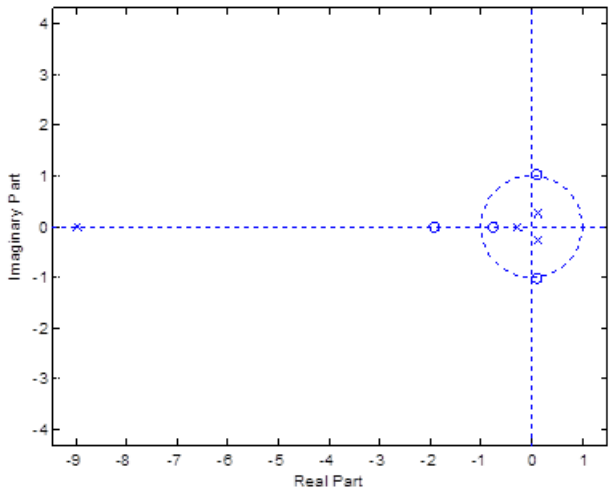
Chương trình vẽ điểm cực, điểm không của các hàm hệ thống sau:

$$H(z) = \frac{2+5z^{-1}+4z^{-2}+5z^{-3}+3z^{-4}}{5+45z^{-1}+2z^{-2}+z^{-3}+z^{-4}}$$

Cho kết quả như sau:

```
nhap cac he so o tu =[2 5 4 5 3];
nhap cac he so o mau so=[5 45 2 1 1];
cac diem cuc tai
  -1.9255 + 0.0000i
   0.0906 + 1.0112i
   0.0906 - 1.0112i
```

```
  -0.7558 + 0.0000i

 cac diem khong tai
  -8.9576 + 0.0000i
  -0.2718 + 0.0000i
   0.1147 + 0.2627i
   0.1147 - 0.2627i

 he so tang ic
    0.4000
ban kinh diem cuc
    8.9576
    0.2718
    0.2866
    0.2866

 cac thanh phan bac 2:
0.4000    1.0725    0.5821    1.0000    9.2293    2.4344
1.0000   -0.1813    1.0308    1.0000   -0.2293    0.0822
```



Hình 3.1 Biểu đồ cực-zero

Bài 3.3 Chương trình Matlab tính và hiển thị biến đổi z dưới dạng phân số hữu tỷ từ các điểm không, điểm cực và tăng ích của nó.

Sử dụng hàm **zp2tf**

```
A=input('nhap cac diem khong duoi dang vecto hang=');
B=input(' nhap cac diem cuc duoi dang veto hang=');
a=A';b=B';
k=input('nhap he so thang ichk=');
[num,den]= zp2tf(z,p,k);
disp(' cac he so cua da thuc tu so:'); disp(num);
disp('cac he so cua da thuc mau so:'); disp( den);
```

Kết quả mô phỏng với các điểm không là 0,5; 2,2 ;-0,2+j0,3;-0,2-j0,3; và các điểm cực tại 0,4; -0,75; 0,5+j0,6; 0,5-j0,6; và tăng ích k=3,7 như sau:

```
nhap cac diem khong duoi dang vecto hang=[0.5 2.2 -
0.2+0.3i -0.2-0.3i]
nhap cac diem cuc duoi dang veto hang=[0.4 -0.75
0.5+0.6i 0.5-0.6i]
nhap he so thang ichk=3.7
 cac he so cua da thuc tu so:
    3.7000    9.2500    7.4000    9.2500    5.5500

cac he so cua da thuc mau so:
    1.0000    9.0000    0.4000    0.2000    0.2000
```

3.2 PHÂN TÍCH DÙNG PHƯƠNG PHÁP THẶNG DƯ

Phân tích thành các thừa số theo phương pháp thặng dư:

$$\frac{N_1(z)}{D(z)} = \frac{A_1}{1 - p_1 z^{-1}} + \cdots + \frac{A_N}{1 - p_N z^{-1}}$$

Bài 3.4. Xác định các hệ số của biểu thức biến đổi z bằng hàm ***residuez***:

Cú pháp của hàm residuez như sau:

[r,p,k] = residuez(b,a)

[b,a] = residuez(r,p,k)

Residuez tìm thặng dư, các cực, các thành phần trực tiếp của việc khai triển biến đổi z có dạng H(z)=B(z)/A(z)

$$B(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_m z^{-m}$$

$$A(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}$$

$$\frac{B(z)}{A(z)} = \frac{r(1)}{1 - p(1)z^{-1}} + \cdots + \frac{r(n)}{1 - p(n)z^{-1}} + k(1) + k(2)z^{-1} + \cdots + k(m-n+1)z^{-(m-n)}$$

```
 num=[1  2  3]

num =

     1     2     3

 den=[2  4  7]

den =

     2     4     7

 [A,p,k] = residuez(num,den)

A =

   0.0357 - 0.0226i
   0.0357 + 0.0226i
```

```
p =

  -1.0000 + 1.5811i
  -1.0000 - 1.5811i


k =

    0.4286
```

Ta cũng có thể dùng hàm ***residuez*** để xác định lại tử số và mẫu số:

```
[num,den] = residuez(A,p,k);
num =

    1.0000    2.0000    3.0000


den =

    2.0000    4.0000    7.0000
```

Bài 3.5. Phân tích biểu thức sau dùng phương pháp thặng dư:

$$H(z) = \frac{0.05634(1+z^{-1})(1-1.0166z^{-1}+z^{-2})}{(1-0.683z^{-1})(1-1.4461z^{-1}+0.7957z^{-2})}$$

```
b0 = 0.05634;
b1 = [1  1];
b2 = [1 -1.0166 1];
a1 = [1 -0.683];
a2 = [1 -1.4461 0.7957];
b = b0*conv(b1,b2);
a = conv(a1,a2);
```

```
[r,p,k] = residuez(b,a)
```

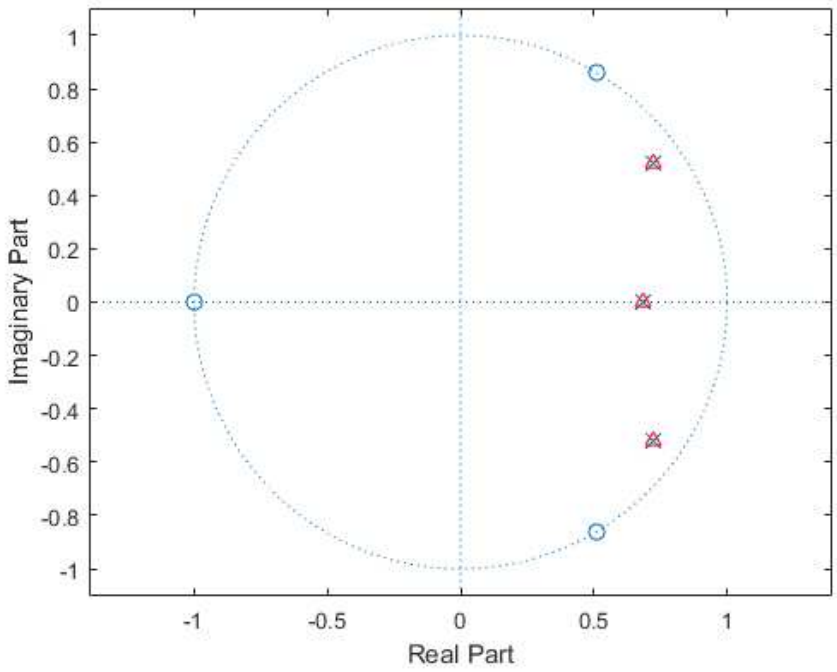
Kết quả như sau:

```
r =
  -0.1153 - 0.0182i
  -0.1153 + 0.0182i
   0.3905 + 0.0000i
p =
   0.7230 + 0.5224i
   0.7230 - 0.5224i
   0.6830 + 0.0000i
k =
   -0.1037
```

Nếu thực hiện vẽ giản đồ Z, dùng hàm zplane,

```
zplane(b,a)
hold on
plot(p,'^r')
hold off
```

Ta có kết quả như sau:



Hình 3.2 Biểu đồ cực zero

Sử dụng hàm residuez cho cú pháp:

```
[bn,an] = residuez(r,p,k)
```

Thu được kết quả như sau:

```
bn =
    0.0563   -0.0009   -0.0009    0.0563
an =
    1.0000   -2.1291    1.7834   -0.5435
```

3.3 BIẾN ĐỔI Z VÀ Z NGƯỢC

Tóm tắt lý thuyết:

Xét hệ LTI biểu diễn bằng phương trình sai phân hệ số hằng:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

Hàm hệ thống của hệ LTI biểu diễn bằng phương trình sai phân hệ số hằng:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

Biến đổi z ngược:

$$X(z) = \frac{N(z)}{D(z)} = \sum_{k=0}^{M-N} c_k z^{-k} + \frac{N_1(z)}{D(z)}$$

(Nếu bậc của tử số nhỏ hơn bậc của mẫu số)

$$\sum_{k=0}^{M-N} c_k z^{-k} \overset{z}{\leftrightarrow} \sum_{k=0}^{M-N} c_k \delta(n-k)$$

Để tính thành phần còn lại, ta phân tích thành các thừa số theo phương pháp thặng dư:

$$\frac{N_1(z)}{D(z)} = \frac{A_1}{1 - p_1 z^{-1}} + \cdots + \frac{A_N}{1 - p_N z^{-1}}$$

Nếu các giá trị pj = … = pm thì chuyển các số hạng từ Aj đến Am thành:

$$\frac{A_j}{1 - p_N z^{-1}} + \frac{A_{j+1}}{(1 - p_N z^{-1})^2} + \cdots + \frac{A_{j+m-1}}{(1 - p_N z^{-1})^m}$$

Áp dụng kết quả:

$$\frac{A}{1 - az^{-1}} \overset{z}{\leftrightarrow} Aa^n u(n), ROC: |z| > |a|$$

Bài 3.7. Dùng hàm ***ztrans*** để biến đổi z ở dạng công thức:

```
syms n x
x = 2^n;
ztrans(x)
x = (-1/2)^n;
ztrans(x)
ans =
```

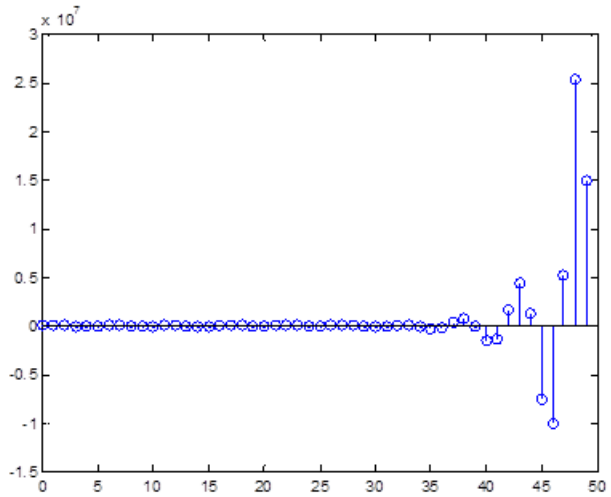
```
z/(z - 2)


ans =

z/(z + 1/2)
```

Bài 3.8. Biến đổi z ngược theo giá trị bằng hàm impz.

```
num = [1 1 2];
den = [1 -1 2];
L = 50; %Số lượng mẫu cần tính
x = impz(num,den,L) % x là biến đổi z ngược
impz(num,den,L); % Vẽ trên đồ thị
x =

     1
     2
     2
    -2
    -6
    -2
    10
    14
    -6
   -34
   -22
    46
    90
    -2
  -182
  -178
```

```
   186
   542
   170
  -914
 -1254
   574
  3082
  1934
 -4230
 -8098
   362
 16558
 15834
-17282
-48950
-14386
 83514
112286
-54742
-279314
-169830
 388798
 728458
 -49138
-1506054
-1407778
 1604330
 4419886
 1211226
-7628546
```

```
-10050998
  5206094
 25308090
 14895902
```



Hình 3.3 Đáp ứng xung của x

Bài 3.9. Xác định và vẽ 100 mẫu đầu tiên của biến đổi z ngược của hàm:

$$X(z) = \frac{0.9 + 0.7z^{-1} + 0.1z^{-2} - z^{-3} + 0.5z^{-4}}{1 + 0.5z^{-1} - 0.2z^{-3} + 2z^{-4} + z^{-5}}$$

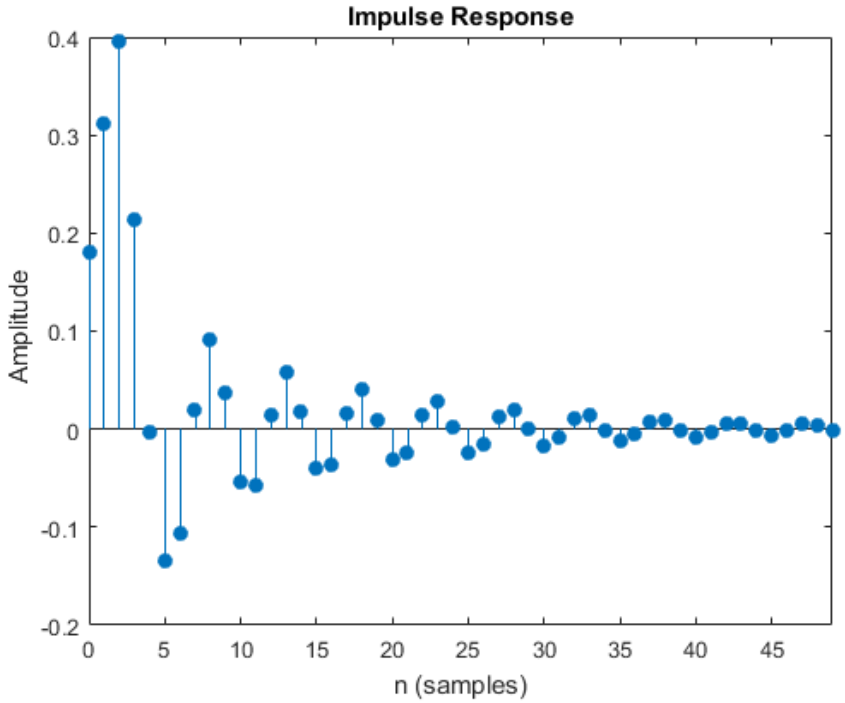
Để xác định biến đổi z ngược ở dạng công thức, ta dùng hàm ***residuez*** để phân tích thành dạng biểu thức đơn giản và dùng các kết quả biến đổi ngược để xác định.

Bài 3.10. Ta cũng có thể xác định biến đổi z ngược bằng cách dùng hàm ***iztrans.***

```
syms F z
F = 2*z^(-1)/(1-3*z^(-1));
iztrans(F)
```

Bài 3.11 Xác định đáp ứng xung của các bộ lọc eliip

```
[[b,a] = ellip(4,0.5,20,0.4);
impz(b,a,50)
```

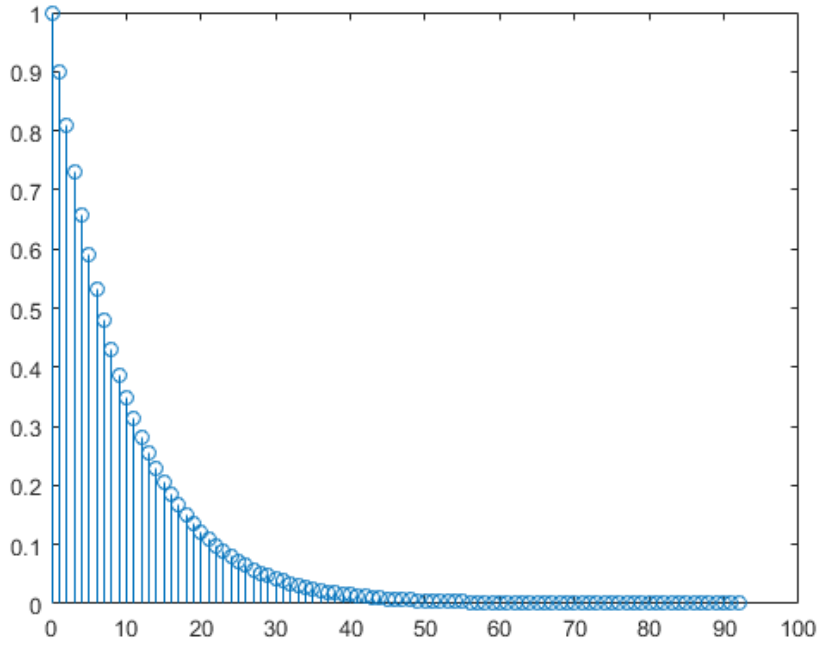


Bài 3.12: Tính chiều dài của đáp ứng xung

```
b = 1;
a = [1 -0.9];
len = impzlength(b,a)
[h,t] = impz(b,a);
stem(t,h)
h(len)
```

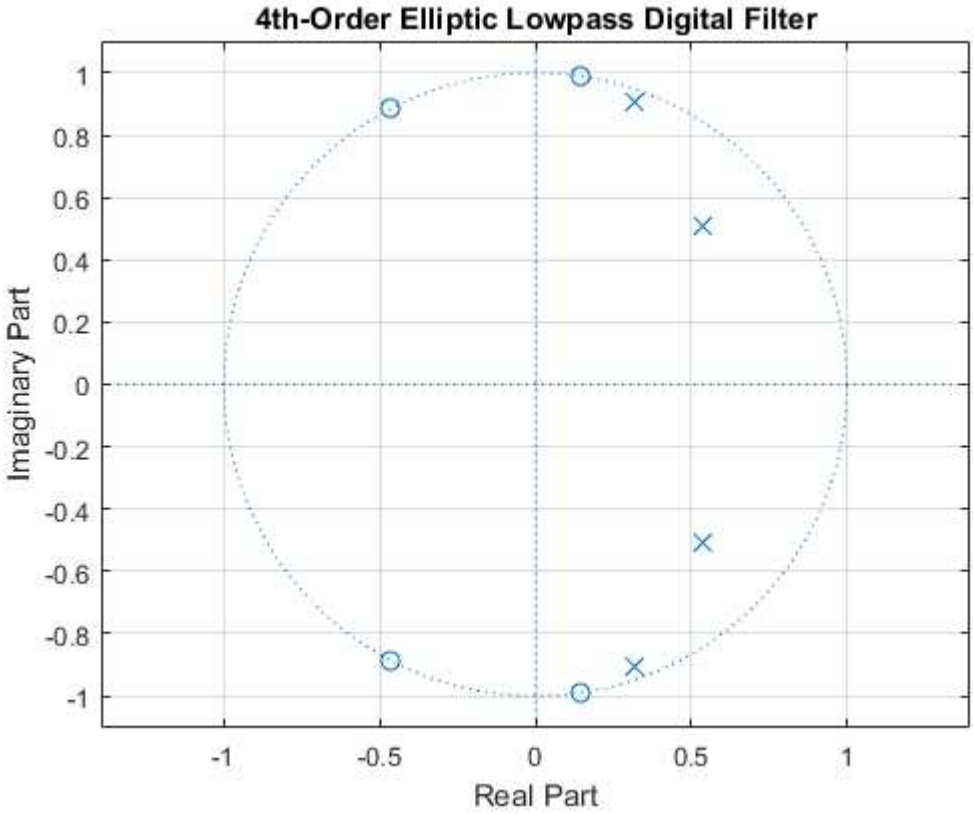
Kết quả

```
len =
    93
ans =
   6.1704e-05
```



Bài 3.13: Điểm cực và zero của bộ lọc Eliip, tần số cắt 200 Hz,dải thông 3 dB đến 30 dB.

```
[z,p,k] = ellip(4,3,30,200/500);
zplane(z,p)
grid
title('4th-Order Elliptic Lowpass Digital Filter')
```



3.4 CÁC CHƯƠNG TRÌNH MATLAB GIẢI CÁC BÀI TOÁN PHÂN TÍCH TRONG MIỀN Z

Bài 3.14: Chương trình Matlab tính toán tìm biểu thức y(n) từ hàm Y(z)

$$Y(z) = \frac{0.85 + 0.1z^{-1}}{1 - 0.4z^{-1} - 0.05z^{-2}}$$

```
num=[0.85 0.1];
den=[1 -0.4 -0.05];
[r,p,k]=residuez(num,den)
```

Kết quả như sau:

```
r=   0.8750
     -0.025
p=   0.5000
     -0.1000
k=[]
```

Từ đó suy ra:

$$Y(z) = \frac{0.85 + 0.1z^{-1}}{1 - 0.4z^{-1} - 0.05z^{-2}} = \frac{0.8750z}{z - 0.5} - \frac{0.025z}{z + 0.1}$$

Biểu thức y(n) là:

$$y(n) = \left[0.8750(0.5)^n - 0.025(-0.1)^n\right]u(n)$$

Bài 3.15 Tìm 20 giá trị của đáp ứng xung y(n) từ Y(z)

$$Y(z) = \frac{0.85 + 0.1z^{-1}}{1 - 0.4z^{-1} - 0.05z^{-2}}$$

```
num=[0.85 0.1];
den=[1 -0.4 -0.05];
[Y,T]=impz(num,den,20)
```

Ta có kết quả ngõ ra như sau:

Y =

0.8500
0.4400
0.2185
0.1094
0.0547
0.0273
0.0137
0.0068
0.0034
0.0017
0.0009
0.0004
0.0002

0.0001

0.0001

0.0000

0.0000

0.0000

0.0000

0.0000

Bài 3.16 Chương trình Matlab tính toán tìm biểu thức y(n) từ hàm Y(z)

$$Y(z)=\frac{1.5-0.2z^{-1}}{\left[1-0.4z^{-1}-0.05z^{-2}\right]\left[1-0.1z^{-1}-0.06z^{-2}\right]}$$

```
d1=[1 -0.4 -0.05];
d2=[1 -0.1 -0.06];
den2=conv(d1,d2);
num=[1.5 -0.2];
[r,p,k]=residuez(num,den2)
```

Kết quả như sau:

```
r=   1.6369
     -0.5625
     0.5714
     -0.1458
p=   0.5000
     0.3000
     -0.2000
     -0.1000
k=[]
```

Biểu thức y(n) là:

$$y(n)=\left[1.6369(0.5)^n-0.5625(0.3)^n+0.5714(-0.2)^n-0.1458(-0.1)^n\right]u(n)$$

Nếu cho n=0,1,..,9

```
n=0:9;
y=[1.6369*(0.5).^n-0.5625*(0.3).^n+0.5714*(-
0.2).^n-0.1458*(-0.1).^n]
```

Kết quả y(n) sẽ là:

```
y =

  Columns 1 through 6

    1.5000    0.5500    0.3800    0.1850    0.0986    0.0496

  Columns 7 through 10

    0.0252    0.0127    0.0064    0.0032
```

Bài 3.17 Chương trình Matlab tính toán giá trị ngõ ra y(n) của bài 3.16 khi tín hiệu ngõ vào là:

$$x(n) = \left[(-0.2)^n + 0.5(0.3)^n\right] \text{ với n=0,1,...9}$$

- **Cách 1: dùng hàm filter**

```
n=[0:9];
x=(-0.2).^n+0.5*(0.3).^n;
b=[0.85 0.1];
a=[1 -0.4 -0.05];
y=filter(b,a,x)
```

Kết quả cho ngõ ra y(n) bằng:

```
y =

  Columns 1 through 6

    1.2750      0.6175      0.3780      0.1952      0.1024
0.0520
```

```
  Columns 7 through 10

    0.0264    0.0133    0.0067    0.0033
```

- Cách 2: Dùng hàm residuez

Từ bài 3.16 ta có biểu thức y(n) là:

$$y(n) = \left[1.6369(0.5)^n - 0.5625(0.3)^n + 0.5714(-0.2)^n - 0.1458(-0.1)^n\right]u(n)$$

Nếu cho n=0,1,..,9

```
n=0:9;
y=[1.6369*(0.5).^n-0.5625*(0.3).^n+0.5714*(-
0.2).^n-0.1458*(-0.1).^n]
```

Kết quả y(n) sẽ là:

```
y =

  Columns 1 through 6

    1.5000    0.5500    0.3800    0.1850    0.0986
0.0496

  Columns 7 through 10

    0.0252    0.0127    0.0064    0.0032
```

- **Cách 3: Dùng hàm impz**

```
d1=[1 -0.4 -0.05];
d2=[1 -0.1 -0.06];
den2=conv(d1,d2);
num=[1.5 -0.2];
[y,T]=impz(num,den2)
```

Kết quả y(n) sẽ là:

```
y =

    1.5000
    0.5500
    0.3800
    0.1850
    0.0986
    0.0496
    0.0252
    0.0127
    0.0064
    0.0032
```

Nhận xét: cả 3 cách đều cho kết quả giống nhau.

Bài 3.18 Tìm đáp ứng xung h(n) với hàm truyền H(z) như sau:

$$H(z) = \frac{1}{1 - 0.4z^{-1} - 0.05z^{-2}}$$

```
b=[1];
a=[1 -0.4 -0.05];
[r,p,k]=residuez(b,a)
```

Kết quả cho:

```
r=0.8333
     0.1667
p=0.5000
-0.1000
k=[]
```

Từ đó suy ra biểu thức ngõ ra H(z) như sau:

$$H(z) = \frac{0.8333z}{z - 0.5} - \frac{0.16667z}{z + 0.1}$$

Biểu thức đáp ứng xung của hệ:

$$h(n) = \left[0.8333(0.5)^n - 0.1667(-0.1)^n\right]u(n)$$

Tìm đáp ứng xung từ hàm impz, như sau:

```
b=[1];
a=[1 -0.4 -0.05];
[h,T]=impz(b,a,20)
```

kết quả như sau:

h =

1.0000
0.4000
0.2100
0.1040
0.0521
0.0260
0.0130
0.0065
0.0033
0.0016
0.0008
0.0004
0.0002
0.0001
0.0001
0.0000
0.0000
0.0000
0.0000
0.0000

Sử dụng tín hiệu ngõ vào là xung đơn vị, dùng hàm filter, cũng cho kết quả tương tự như sau:

```
b=[1];
a=[1 -0.4 -0.05];
x=[1 zeros(1,19)];
h=filter(b,a,x)
h =

  Columns 1 through 6

    1.0000      0.4000      0.2100      0.1040      0.0521
0.0260

  Columns 7 through 12

    0.0130      0.0065      0.0033      0.0016      0.0008
0.0004

  Columns 13 through 18

    0.0002      0.0001      0.0001      0.0000      0.0000
0.0000

  Columns 19 through 20

    0.0000    0.0000
```

Bài 3.19 Phân tích hàm truyền sử dùng hàm deconv(b,a)

$$X(z) = \frac{0.1 + 0.25z^{-1}}{1 + 0.4z^{-1} + 0.5z^{-2}}$$

```
b=[0.1 0.25 0]
```

```
a=[1 0.4 0.5];
n=5;
b=[b zeros(1,n-1)];
[x,r]=deconv(b,a)
```

Kết quả như sau:

```
x =

    0.1000    0.2100   -0.1340   -0.0514    0.0876

r =

  Columns 1 through 6

         0         0         0         0         0   -
0.0093

  Column 7

   -0.0438
```

Suy ra biểu thức X(z)

$$X(z)=\frac{0.1+0.25z^{-1}}{1+0.4z^{-1}+0.5z^{-2}}=0.1+0.21z^{-1}-0.134z^{-2}-0.0514z^{-3}+\frac{0.086z^{-4}-0.0257z^{-5}}{1+0.4z^{-1}+0.5z^{-2}}$$

Bài 3.20 $X_1(z)=2+3z^{-1}+4z^{-2}$ **và** $X_2(z)=3+4z^{-1}+5z^{-2}+6z^{-3}$**.**

Tìm $X_3(z)=X_1(z).X_2(z)$

```
x1=[2,3,4];
x2=[3,4,5,6];
x3=conv(x1,x2)

x3 =
```

```
     6     17     34     43     38     24
```

Từ đó suy ra biểu thức:

$$X_3(z) = 6 + 17z^{-1} + 34z^{-2} + 43z^{-3} + 38z^{-4} + 24z^{-5}$$

Bài 3.21 Nếu $X_1(z) = z + 2 + 3z^{-1}$ **tương ứng** $x_1(n) = \{1, \underset{\uparrow}{2}, 3\}$ **và**

$X_2(z) = 2z^2 + 4z + 3 + 5z^{-1}$ **tương ứng** $x_1(n) = \{2, 4, \underset{\uparrow}{3}, 5\}$**.**

$X_3(z)=X_1(z).X_2(z)$

```
x1=[1,2,3]; n1=[-1:1];
x2=[2,4,3,5];n2=[-2:1];
[x3,n3]=conv_m(x1,n1,x3,n2)


x3=
     2     8    17   23   19   15
n3=
 -3   -2   -1   0    1    2
```

Từ đó cho kết quả $X_3(z)$ bằng:

$$X_3(z) = 2z^3 + 8z^2 + 17z + 23 + 19z^{-1} + 15z^{-2}$$

******Với hàm **conv_m** là:

```
function [y,ny] = conv_m(x,nx,h,nh)

  nyb = nx(1)+nh(1);
  nye = nx(length(x)) + nh(length(h));
  ny = [nyb:nye];
  y = conv(x,h);

end
```

3.5 BÀI TẬP

Bài tập 1. Viết chương trình Matlab cho hệ có phương trình sai phân:

$$y(n) – 0.7y(n-1) = x(n)$$

a. Tìm H(z). Vẽ đồ thị điểm cực, zero.

b. Tìm h(n). Vẽ đáp ứng xung h(n)

c. Tìm và vẽ y(n) nếu $x(n) = (0.8)^n u(n)$

Bài tập 2. Viết chương trình Matlab cho hệ có phương trình sai phân:

$y(n) – 0.5y(n-1) = x(n) + x(n-1)$

a. Tìm H(z). Vẽ đồ thị điểm cực, zero.

b. Tìm h(n). Vẽ đáp ứng xung h(n)

c. Tìm và vẽ y(n) nếu $x(n) = (0.6)^n u(n)$.

CHƯƠNG 4: BIẾN ĐỔI FOURIER RỜI RẠC

Nội dung chính:

 Chuyển tín hiệu rời rạc trên miền thời gian sang miền tần số bằng cách dùng biến đổi Fourier rời rạc.

 Tính toán biến đổi Fourier bằng một số thuật toán.

*Các hàm Matlab liên quan:

Signal Processing Toolbox

filter	filtfilt	freqz	grpdelay	impz
poly2rc	sinc	zplane		

4.1 DTFT VÀ CÁC TÍNH CHẤT

Tóm tắt lý thuyết:

Biến đổi Fourier rời rạc (DTFT) mô tả như sau:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-jn\omega} = \left|X(e^{j\omega})\right|e^{j\Phi(\omega)}$$

Bài 4.1. Tính và vẽ DTFT có dạng tổng quát như sau:

$$X(e^{j\omega}) = \frac{p_0 + p_1 e^{-j\omega} + \ldots + p_M e^{-j\omega M}}{d_0 + d_1 e^{-j\omega} + \ldots + d_N e^{-j\omega N}},$$

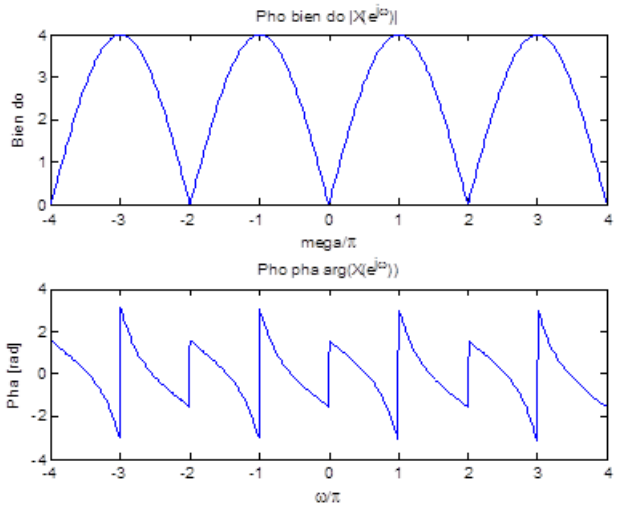
```
%Chuong trinh Matlab bai 4_1
%Tinh toan va bieu dien bien doi Fourier trong mien tan
so
w = linspace(-4*pi,4*pi,512);
% Tạo 512 giá trị từ -4pi đến 4pi
num = input('Nhap cac he so cua tu so = ');
den = input('Nhap cac he so cua mau so = ');
h = freqz(num,den,w);
subplot(211),plot(w/pi,abs(h));
```

```
xlabel('mega/\pi');
ylabel('Bien do');
title('Pho bien do |X(e^{j\omega})|');
subplot(212),plot(w/pi,angle(h));
xlabel('\omega/\pi');
ylabel('Pha [rad]');
title('Pho pha arg(X(e^{j\omega}))');
```

Áp dụng cho bài 4.1 khảo sát DTFT cho hàm X sau:

$$X(e^{-j\omega}) = \frac{1 + e^{-j\omega} - 2e^{-j2\omega}}{1 + 0.5e^{-j\omega}}$$

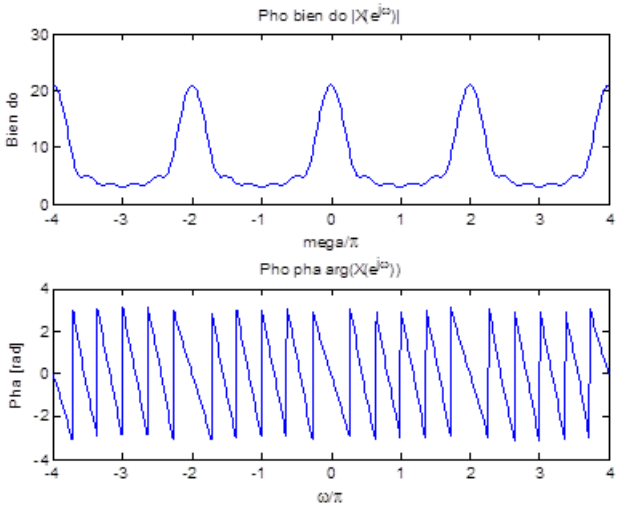
Ta có kết quả như sau:



Để khảo sát chuỗi x(n) hữu hạn, ta cho thông số thứ 2 của hàm ***freqz*** là 1.

Áp dụng cho chuỗi x(n)= `[1 2 3 4 5 6]`

```
Nhap cac he so cua tu so = [1 2 3 4 5 6]
Nhap cac he so cua mau so = 1
```



Bài 4.2 Tính chất dịch thời gian của DTFT

Tính chất dịch thời gian: Chương trình 4_2 biểu diễn phổ biên độ và phổ pha của một đáp ứng tần số có dạng:

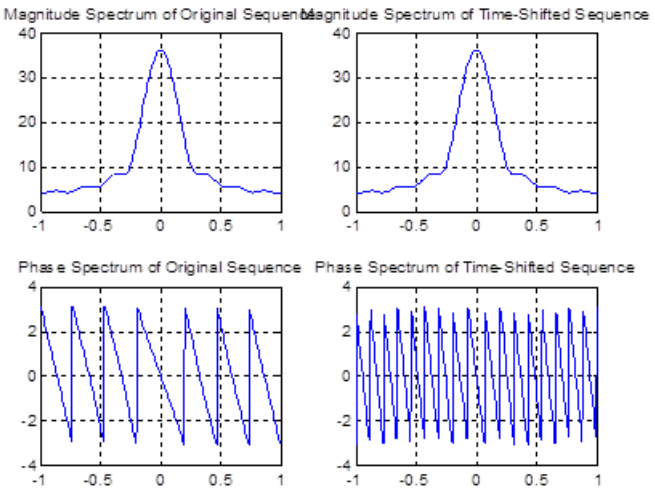
Đồng thời biểu diễn phổ biên độ và phổ pha của dãy tín hiệu trên với dịch thời gian D:

```
% Program P4_2
% Tinh chat dich thoi gian cua DTFT
clf;
w = -pi:2*pi/255:pi; wo = 0.4*pi;
D = input('Nhap thoi gian tre =' );
num=input('Nhap vector tu so = ');
h1 = freqz(num, 1, w);
h2 = freqz([zeros(1,D) num], 1, w);
subplot(2,2,1)
plot(w/pi,abs(h1));grid
title('Magnitude Spectrum of Original Sequence')
subplot(2,2,2)
plot(w/pi,abs(h2));grid
title('Magnitude Spectrum of Time-Shifted Sequence')
subplot(2,2,3)
```

```
plot(w/pi,angle(h1));grid
title('Phase Spectrum of Original Sequence')
subplot(2,2,4)
plot(w/pi,angle(h2));grid
title('Phase Spectrum of Time-Shifted Sequence')
```

Áp dụng cho trường hợp h[*n*] = [1 2 3 4 5 6 7 8] và trễ thời gian D=10, kết quả như sau:

```
Nhap thoi gian tre =10
Nhap vector tu so = [1 2 3 4 5 6 7 8]
```

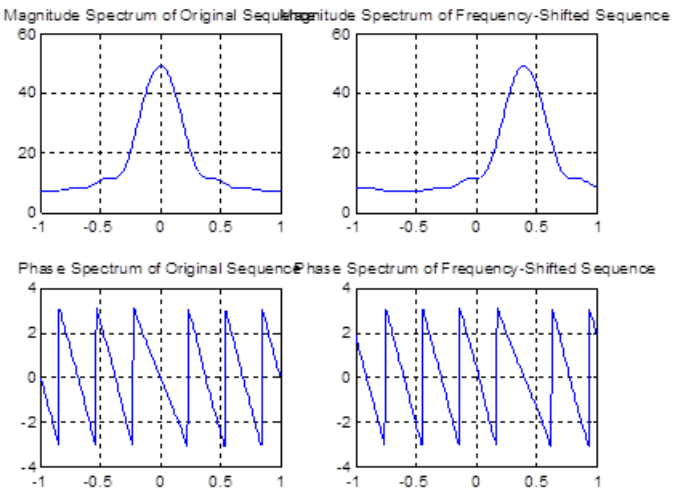


Bài 4.3 Tính chất dịch tần số của DTFT

```
%Bai 4.3
%Tinh chat dich tan so DTFT
clf;
w = -pi:2*pi/255:pi;
wo = input('Nhap do dich tan so');
num1 = input('Nhap gia tri vetor tu so= ')
L = length(num1);
h1 = freqz(num1, 1, w);
n = 0:L-1;
num2 = exp(wo*i*n).*num1;
```

```
h2 = freqz(num2, 1, w);
subplot(2,2,1)
plot(w/pi,abs(h1));grid
title('Magnitude Spectrum of Original Sequence')
subplot(2,2,2)
plot(w/pi,abs(h2));grid
title('Magnitude Spectrum of Frequency-Shifted
Sequence')
subplot(2,2,3)
plot(w/pi,angle(h1));grid
title('Phase Spectrum of Original Sequence')
subplot(2,2,4)
plot(w/pi,angle(h2));grid
title('Phase Spectrum of Frequency-Shifted Sequence')
```

Áp dụng cho h(n)=[1 3 5 7 9 11 13], trễ tần số bằng 0.4*pi ta có kết quả như sau:

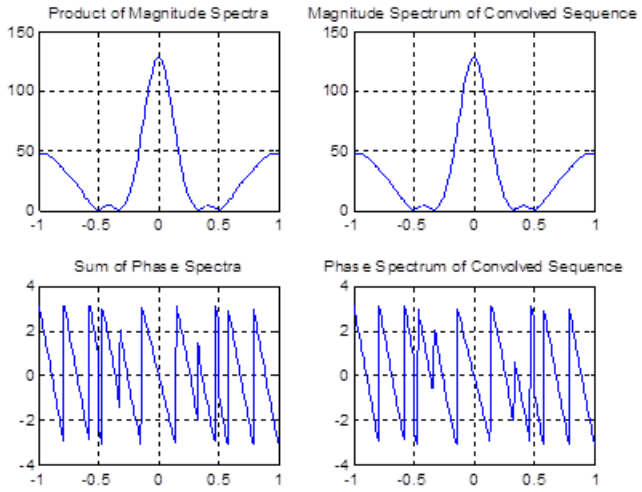


Bài 4.4 Tính chất chập của FT

```
%Bai 4 4
%Tinh chat chap cua FT
clf;
w = -pi:2*pi/255:pi;
x1 = input(' Nhap day thu nhat = ');
```

```
x2 = input('Nhap day thu hai = ')
y = conv(x1,x2);
h1 = freqz(x1, 1, w);
h2 = freqz(x2, 1, w);
hp = h1.*h2;
h3 = freqz(y,1,w);
subplot(2,2,1)
plot(w/pi,abs(hp));grid
title('Product of Magnitude Spectra')
subplot(2,2,2)
plot(w/pi,abs(h3));grid
title('Magnitude Spectrum of Convolved Sequence')
subplot(2,2,3)
plot(w/pi,angle(hp));grid
title('Sum of Phase Spectra')
subplot(2,2,4)
plot(w/pi,angle(h3));grid
title('Phase Spectrum of Convolved Sequence')
```

Áp dụng mô phỏng cho chuỗi x1(n)=[1 3 5 7 9 11 13 15] và x2(n)=[1 -1 2 -1 1]

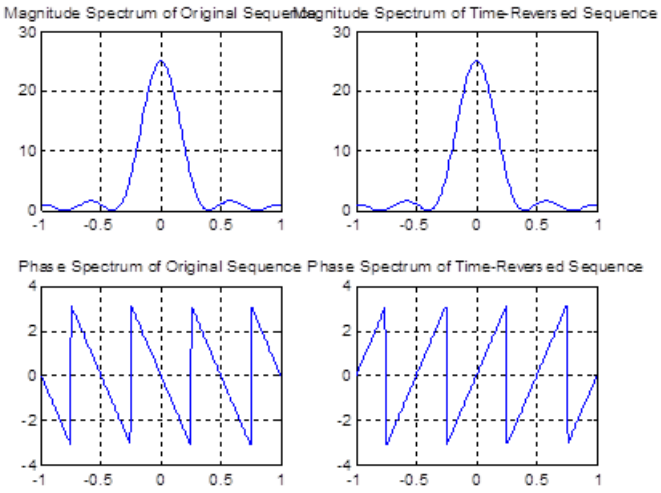


Bài 4.5 Tính chất biến số đảo của FT

Vẽ phổ biên độ và phổ pha của chuỗi x(n) và x(-n)

```
% Chuong trinh bai 4.5
% Tinh chat bien so dao
clf;
w = -pi:2*pi/255:pi;
num=input('Nhap chuoi tin hieu goc = ');
%num = [1 2 3 4];
L = length(num)-1;
h1 = freqz(num, 1, w);
h2 = freqz(fliplr(num), 1, w);
h3 = exp(w*L*i).*h2;
subplot(2,2,1)
plot(w/pi,abs(h1));grid
title('Magnitude Spectrum of Original Sequence')
subplot(2,2,2)
plot(w/pi,abs(h3));grid
title('Magnitude Spectrum of Time-Reversed Sequence')
subplot(2,2,3)
plot(w/pi,angle(h1));grid
title('Phase Spectrum of Original Sequence')
subplot(2,2,4)
plot(w/pi,angle(h3));grid
title('Phase Spectrum of Time-Reversed Sequence')
```

Áp dụng cho chuỗi x(n)=[1 2 3 4 5 4 3 2 1], kết quả như sau:

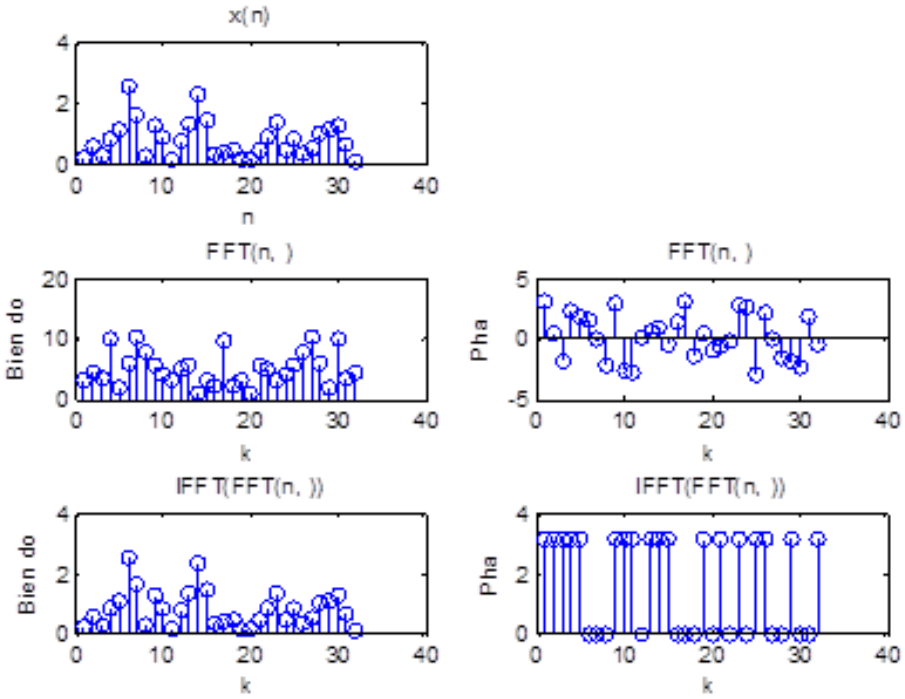


4.2 FFT VÀ CÁC TÍNH CHẤT

Bài 4.6. Dùng hàm ***fft*** và ***ifft*** để tính DFT và IDFT của x(n), với x(n) ngẫu nhiên

```
% chuong trinh bai 4.6
    clf;
    N = 32;
    x = randn(1,N);
    y = fft(x,N);
    x1 = ifft(y,N);
    subplot(321),stem(abs(x));
    title('x(n)');xlabel('n');
    subplot(323),stem(abs(y));
    title(['FFT(n,'   N   ')']);xlabel('k');ylabel('Bien
    do');
    subplot(324),stem(angle(y));
    title(['FFT(n,' N ')']); xlabel('k');ylabel('Pha');
    subplot(325),stem(abs(x1));
    title(['IFFT(FFT(n,' N '))']);
    xlabel('k');ylabel('Bien do');
    subplot(326),stem(angle(x1));
    title(['IFFT(FFT(n,' N '))']);
```

```
xlabel('k');ylabel('Pha');
```



Kết quả tính FFT và IFFT

Bài 4.7. Xác định FFT 8 điểm của x(n) cho tùy ý.

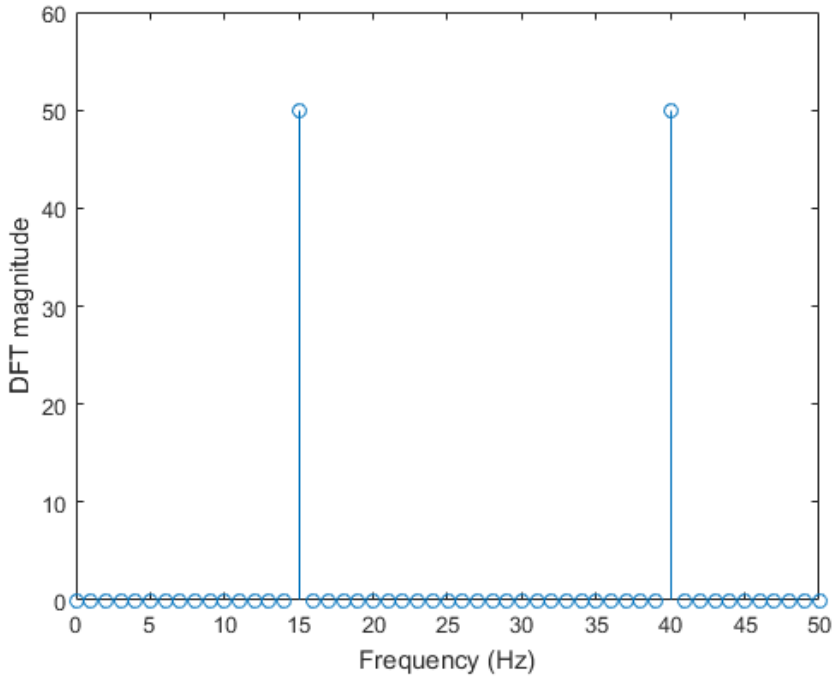
```
 x=[1 2 3 4];
 y=fft(x,8)

y =

  Columns 1 through 3

  10.0000  +  0.0000i    -0.4142  -  7.2426i    -2.0000  +
2.0000i

  Columns 4 through 6
```

```
   2.4142  -  1.2426i   -2.0000  +  0.0000i     2.4142  +
1.2426i

  Columns 7 through 8

  -2.0000 - 2.0000i  -0.4142 + 7.2426i
```

Bài 4.8: Tín hiệu sin được lấy mẫu tại 100Hz. Tính toán DFT của dãy.

```
t = (0:99)/100;                          % Time vector
  x = sin(2*pi*15*t) + sin(2*pi*40*t);   % Signal
  y = fft(x);                            % DFT of x
  m = abs(y);                            % Magnitude
  f = 0:50;                        % Frequency vector
  m = m(1:51);
  magnitudes
  stem(f,m)
  ylabel 'DFT magnitude'
  xlabel 'Frequency (Hz)'
```

Vẽ phổ biên độ



Hình kết quả mô phỏng bài 4.8

Bài 4.9. Tạo function ***circshift*** để dịch vòng một chuỗi m giá trị:

```
function out = circshift(x,m)
%Xay dung mot chuo out bang cach dich vong
%mot chuoi x co chieu dai huu han di m mau
m0 = m;
if abs(m0) > length(x)
     m0 = rem(m0,length(x));
end
while (m0<0)
     m0 = m0 + length(x);
end
out=   [x(length(x)-m0+1:length(x))   x(1:length(x)-
m0)];
```

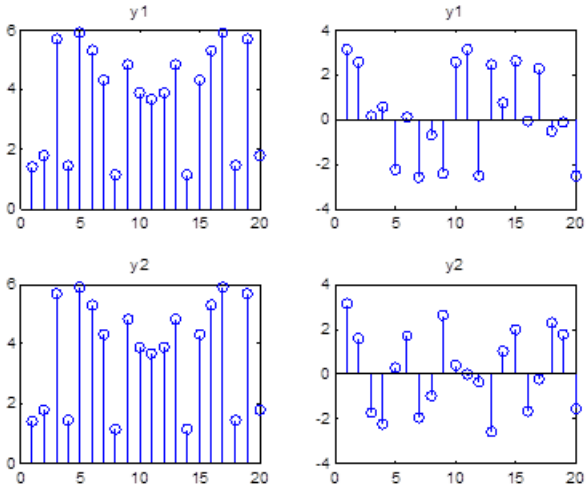
Kiểm tra chương trình như sau:

```
A = (1:10)'
A =

     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
Y = circshift(A,3)
Y =

     8
     9
    10
     1
     2
     3
     4
     5
     6
     7
```

Bài 4.10 Khảo sát tính chất dịch vòng của DFT-N điểm:

```
% chuong trinh bai 4.10
clf;
N = 20;
```

```
m = 3;
x = randn(1,N);
y = fft(x,N);
x1 = circshift(x,m);
%Có thể thay bằng hàm circshift như sau:
%x1 = circshift(x',m)';
y1 = fft(x1,N);
k = 0:N-1;
y2 = exp(-j*2*pi*k*m/N).*y;
subplot(221),stem(abs(y));title('y1');
subplot(222),stem(angle(y)); title('y1');
subplot(223),stem(abs(y2));title('y2');
subplot(224),stem(angle(y2));title('y2');
```



Kết quả mô phỏng bài 4.10

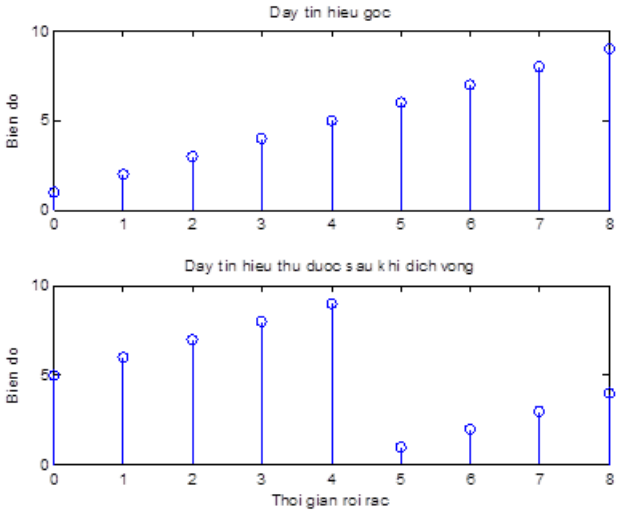
Bài 4.11. Chương trình Matlab tính tích chập vòng của một chuỗi bất kỳ nhập từ bàn phím, số mẫu cần dịch m cũng nhập từ bàn phím

```
clf;
M=input('Nhap so mau can dich=')
x=input('Nhap day tin hieu goc=')
```

```
%y=circshift(x,[1,-M]M)
y=circshift(x,[1,-M])
L=length(x)-1;
n=0:L;
subplot(211)
stem(n,x)
%axis{[0,L,min(x),max(x)]};
title('Day tin hieu goc');
ylabel('Bien do');
subplot(212)
stem(n,y)
%axis([0,L,mix(x),max(x)])
title('Day tin hieu thu duoc sau khi dich vong')
ylabel('Bien do')
xlabel('Thoi gian roi rac')
```

Áp dụng chương trình mô phỏng với M=4, x(n)=[0 1 2 3 4 5 6 7 8 9 10] có kết quả như sau:

```
Nhap so mau can dich=4

M =
     4

Nhap day tin hieu goc=[1 2 3 4 5 6 7 8 9]

x =
     1     2     3     4     5     6     7     8     9

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



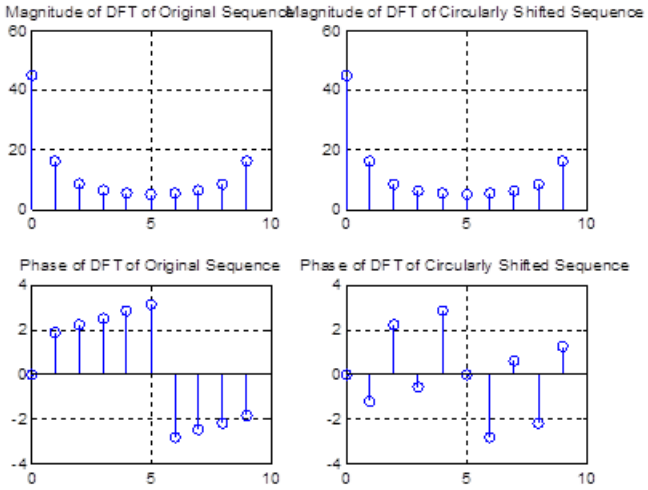
Kết quả mô phỏng bài 4.11

Bài 4.12 Vẽ phổ biên độ và phổ pha của chuỗi tín hiệu gốc và chuỗi tín hiệu sau khi dịch vòng.

```
%Chuong trinh Bai 4.11
clf;
x = input('Nhap tin hieu goc = ');
M=input('Nhap so mau can dich vong = ');
N = length(x)-1; n = 0:N;
%y = circshift1(x,5);
y = circshift(x,[1,-M]);
XF = fft(x);
YF = fft(y);
subplot(2,2,1)
stem(n,abs(XF)); grid
title('Magnitude of DFT of Original Sequence');
subplot(2,2,2)
stem(n,abs(YF)); grid
title('Magnitude of DFT of Circularly Shifted
Sequence');
subplot(2,2,3)
```

```
stem(n,angle(XF)); grid
title('Phase of DFT of Original Sequence');
subplot(2,2,4)
stem(n,angle(YF)); grid
title('Phase of DFT of Circularly Shifted Sequence');
```

Áp dụng mô phỏng cho x(n) =[0 1 2 3 4 5 6 7 8 9] và số mẫu cần dịch bằng 5, ta có kết quả sau:



Hình kết quả mô phỏng bài 4.12

4.3 BÀI TẬP

Bài tập 1: Viết chương trình Matlab tính DFT 8 điểm của chuỗi sau:

$$x(n) = \cos\frac{\pi}{2}n \qquad 0 \le n \le 15$$

Bài tập 2: Viết chương trình tính DFT của một chuỗi tín hiệu nhập từ bàn phím, độ dài mong muốn của DFT cũng được nhập từ bàn phím.

CHƯƠNG 5: BỘ LỌC SỐ FIR

Nội dung chính:

 Thiết kế một bộ lọc số FIR dựa theo các thông số cho trước.

*Các hàm Matlab liên quan:

blackman	butter	buttord	chebwin	cheb1ord
cheb2ord	cheby1	cheby2	ellip	ellipord
fir1	fir2	firpm	firpmord	freqz
hanning	hamming	kaiser		

5.1 CÁC LOẠI BỘ LỌC

Tóm tắt lý thuyết:

Điều kiện đối xứng và phản đối xứng của mạch lọc:

$$h(n) = \pm h(M - 1 - n)$$

Dựa trên tính chất đối xứng hay phản đối xứng của chuỗi đáp ứng xung và chiều dài N của chuỗi đáp ứng xung, người ta phân loại bộ lọc FIR làm 4 loại sau:

• Bộ lọc FIR loại 1: h(n) đối xứng, M lẻ, $\beta = 0$, $\alpha = (M - 1)/2$

• Bộ lọc FIR loại 2: h(n) đối xứng, M chẵn, $\beta = 0$, $\alpha = (M - 1)/2$

• Bộ lọc FIR loại 3: h(n) phản đối xứng, M lẻ, $\beta = \pi/2$, $\alpha = (M - 1)/2$

• Bộ lọc FIR loại 4: h(n) phản đối xứng, M chẵn, $\beta = \pi/2$, $\alpha = (M - 1)/2$

Đáp ứng tần số của FIR cho từng loại:

- Loại 1:

$$H(\omega) = \left[\sum_{n=0}^{\frac{M-1}{2}} a(n)\cos\omega n\right] e^{-j\frac{(M-1)}{2}\omega} \text{ với } \begin{cases} a(0) = h\left(\frac{M-1}{2}\right) \\ a(n) = 2h\left(\frac{M-1}{2} - n\right) \text{ với } 1 \le n \le \frac{M-1}{2} \end{cases}$$

- Loại 2:

$$H(\omega) = \left[\sum_{n=1}^{\frac{M}{2}} b(n)\cos\omega\left(n - \frac{1}{2}\right)\right] e^{-j\frac{(M-1)}{2}\omega} \text{ với } b(n) = 2h\left(\frac{M}{2} - n\right) \text{ với } 1 \le n \le \frac{M}{2}$$

- Loại 3:

$$H(\omega) = \left[\sum_{n=1}^{\frac{M-1}{2}} c(n)\sin\omega n\right] e^{-j\left(\frac{\pi}{2} - \frac{(M-1)}{2}\omega\right)} \text{ với } c(n) = 2h\left(\frac{M-1}{2} - n\right) \text{ với } 1 \le n \le \frac{M-1}{2}$$

- Loại 4:

$$H(\omega) = \left[\sum_{n=1}^{\frac{M}{2}} d(n)\sin\omega\left(n - \frac{1}{2}\right)\right] e^{-j\left(\frac{\pi}{2} - \frac{(M-1)}{2}\omega\right)} \text{ với } d(n) = 2h\left(\frac{M}{2} - n\right) \text{ với } 1 \le n \le \frac{M}{2}$$

Bài 5.1. Xác định đáp ứng tần số của bộ lọc FIR loại 1 từ chuỗi đáp ứng xung. Tạo function FIR_t1 như sau:

```
function [a,w,L,Hr] = FIR_t1(h)
    M = length(h);
    L = (M-1)/2;
    a = [h(L+1) 2*h(L:-1:1)];
    n = [0:1:L];
    w = linspace(0,2*pi,100)';
    Hr = cos(w*n)*a';
    stem(Hr);
```

Lưu file với tên ***FIR_t1.m***.

Thực hiện tính toán với đáp ứng xung h1 = [1.5 -2.5 3 -2.5 1.5]:

```
h1 = [1.5 -2.5 3 -2.5 1.5];
 [a,w,L,Hr]=FIR_t1(h1);
```

Bài 5.2. Cho đáp ứng xung của bộ lọc FIR như sau: h = [-1 2 1.3 -2.2 0.6 3 0.6 -2.2 1.3 2 -1]. Đáp ứng xung h(n) đối xứng với M lẻ nên đây là bộ lọc FIR loại 1.

a. Biểu diễn đáp ứng xung:

```
h = [-1 2 1.3 -2.2 0.6 3 0.6 -2.2 1.3 2 -1];
M = length(h);
n = 0:M-1;
subplot(221);
stem(n,h);
title('Dap ung xung');xlabel('n');ylabel('h(n)');
```

b. Các hệ số của bộ lọc:

```
[a,w,L,Hr] = FIR_t1(h);
subplot(222);stem(0:L,a);
```

```
title('Cac he so a(n)');
xlabel('n');ylabel('a(n)');
```

c. Đáp ứng tần số:

```
w = linspace(0,2*pi,100)';
subplot(223);plot(w,Hr);
title('Dap ung tanso');
xlabel('\omega');
ylabel ('H(\omega)');
```

d. Phân bố cực - không:

```
subplot(224); zplane(h,1);
title('Bieu do cuc – khong');
xlabel('Thuc');ylabel('Ao');
```

5.2 PHƯƠNG PHÁP CỬA SỔ

Bài 5.3. Tạo hàm ***ideal_lp*** xác định đáp ứng xung của bộ lọc thông thấp lý tưởng theo tần số cắt ω_C và chiều dài chuỗi đáp ứng xung.

```
function hd = ideal_lp(wc,M)
alpha = (M-1)/2;
n = [0:1:(M-1)];
m = n - alpha + eps;
hd = sin(wc*m)./(pi*m);
```

Bài 5.4. Tạo hàm ***freqz_m*** tính toán độ lớn và pha của đáp ứng tần số, hàm trễ nhóm.

```
function [db,mag,pha,grd,w] = freqz_m(b,a)
% db = Do lon tuong doi theo dB tren doan tu 0 den pi
% mag = Do lon tuyet doi tren doan tu 0 den pi
% pha = Dap ung pha tren doan tu 0 den pi
% grd = Tre nhom tren doan tu 0 den pi
% w = Cac mau tan so doan tu 0 den pi
% b = Cac he so da thuc tu so cua H(z) (voi FIR: b=h)
```

```
% a = Cac he so da thuc mau so cua H(z)(voi FIR: a=[1])
     [H,w] = freqz(b,a,1000,'whole');
     H = (H(1:1:501))';
     w = (w(1:1:501))';
     mag = abs(H);
     db = 20*log10((mag+eps)/max(mag));
     pha = angle(H);
     grd = grpdelay(b,a,w);
```

Bài 5.5. Thiết kế bộ lọc thông thấp theo phương pháp cửa sổ Hamming với các tham số như sau:

$$\omega_p = 0.2\pi;\ \omega_s = 0.3\pi;\ R_p = 0.25\ dB;\ A_s = 50\ dB$$

Việc thiết kế bộ lọc là quá trình tìm ra các tham số, hay chuỗi đáp ứng xung của bộ lọc, thoả mãn các yêu cầu chỉ tiêu kỹ thuật cho trước, cụ thể là một số hoặc tất cả các tham số tuyệt đối (absolute specification) sau:

- Tần số cắt dải thông ω_p
- Tần số cắt dải thông ω_s
- Bề rộng dải quá độ $\Delta\omega$
- Độ gợn sóng dải thông δ_p
- Độ gợn sóng dải chắn δ_s

Các tham số thường được cho dưới dạng đơn vị dB:

• Độ gợn sóng dải thông theo dB:

$$R_p = -20log\frac{1-\delta_p}{1+\delta_p}$$

• Độ suy giảm dải chắn theo dB:

$$A_S = -20log\frac{\delta_s}{1+\delta_p}$$

Các hàm cửa sổ trong MATLAB: ***boxcar*** *hay* ***rectwin, bartlett, hanning, hamming, blackman, Kaiser.***

```
wp = 0.2*pi; ws =0.3*pi;
```

```
tr_width = ws - wp;
M = ceil(6.6*pi/tr_width) + 1;
n = [0:1:M-1];
wc = (ws+wp)/2;
hd = ideal_lp(wc,M);
w_ham = (hamming(M))';
h = hd.*w_ham;
[db,mag,pha,grd,w] = freqz_m(h,[1]);
delta_w = 2*pi/1000;
Rp = -(min(db(1:1:wp/delta_w+1)))
As = -round(max(db(ws/delta_w+1:1:501)))
```

a. Đáp ứng xung:

```
subplot(221); stem(n,hd);
axis([0,M-1,-0.1,0.3]);
title('Dap ung xung ly tuong');xlabel('n');
ylabel('h_d(n)');
```

b. Cửa sổ Hamming:

```
subplot(222); stem(n,w_ham);
axis([0,M-1,0,1.1]);
title('Cua so Hamming');xlabel('n');
ylabel('w(n)');
```

c. Đáp ứng tần số:

```
subplot(223); stem(n,h);
axis([0,M-1,-0.1,0.3]);
title('Dap ung tan so');
xlabel('n'); ylabel('h(n)');
```

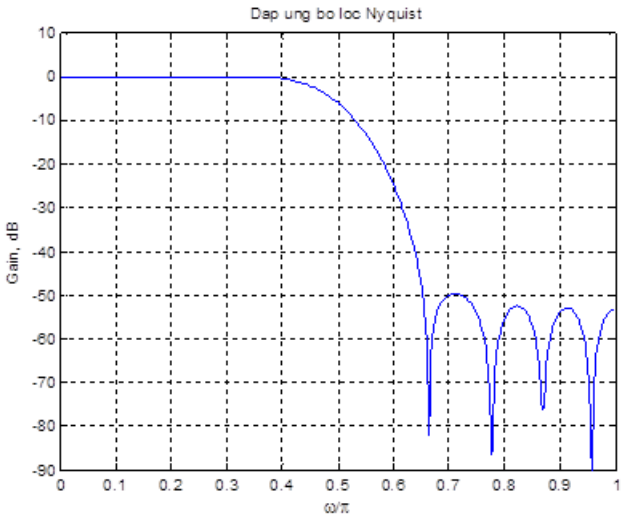
d. Đáp ứng tần số theo dB:

```
subplot(224); plot(w,db);
axis([0,1,-100,10]);
title('Dap ung tan so theo dB');
```

```
    xlabel('\omega'); ylabel('dB');
```

Bài 5.6 Thiết kế bộ lọc Nyquist dùng cửa sổ Hamming

```
%Bo loc Nyquist
n = -K:K;
b = sinc(n/2)/2;
win = hamming(23);
fil = b.*win';
c = fil/sum(fil);
[h,w] = freqz(c,1,256);
g = 20*log10(abs(h));
plot(w/pi,g);axis([0 1 -90 10]); grid
xlabel('\omega/\pi');ylabel('Gain, dB');
```



Hình 5.1 Đáp ứng bộ lọc Nyquist

5.3 PHƯƠNG PHÁP LẤY MẪU TẦN SỐ

Xét đáp ứng xung h(n) của hệ thống LTI có đáp ứng tần số là:

$$H(\omega) = \sum_{n=0}^{M-1} h(n)e^{-j\omega n}$$

Ta chỉ định một tập tần số như sau:

$$\omega_k = \frac{2\pi}{M}(k+\alpha), \begin{cases} k = 0,\ 1, \dots, \frac{M-1}{2} \text{ nếu } M \text{ lẻ} \\ k = 0,\ 1, \dots, \frac{M}{2} - 1 \text{ nếu } M \text{ chẵn} \\ \alpha = 0 \text{ hay } \frac{1}{2} \end{cases}$$

Khi đó:

$$H(k+\alpha) = H\left(\frac{2\pi}{M}(k+\alpha)\right) = \sum_{n=0}^{M-1} h(n) e^{-j\frac{2\pi}{M}(k+\alpha)n}$$

Tập hợp các giá trị {H(k +α)} gọi là các mẫu tần số của H(α). Trong trường hợp α = 0, {H(k)} tương ứng với DFT M điểm của h(n).

Bài 5.7. Thiết kế bộ lọc thông thấp theo phương pháp lấy mẫu tần số với các tham số như sau:

$\omega_p = 0.2\pi$; $\omega_s = 0.3\pi$; $R_p = 0.25$ dB; $A_s = 50$ dB

Chọn đáp ứng xung có chiều dài 60 ứng với 60 mẫu tần số trong khoảng [0,2π). Dải thông có độ rộng là 0.2π tương đương với 7 mẫu nhận giá trị 1. Giả sử quá trình tối ưu hoá chỉ ra nên chọn dải chuyển tiếp 2 mẫu nhận các giá trị T1 = 0.5925 và T2 = 0.1099. Mẫu các tần số được cho như sau:

$$H(k) = [1,1,1,1,1,1,1,T_1,T_2,0,0,\dots,0,0,T_2,T_1,1,1,1,1,1,1]$$

(43 số 0)

```
M = 60; alpha = (M-1)/2; L = 0:M-1; wl = (2*pi/M)*L;
Hk = [ones(1,7),0.5925,0.1099,zeros(1,43), 0.1099, 0.5925, ones(1,6)];
% Dap ung tan so mau ly tuong
Hdr = [1,1,0,0]; wdl = [0,0.2,0.3,1];
k1 = 0:floor((M-1)/2); k2 = floor((M-1)/2)+1:M-1;
angH = [-alpha*(2*pi)/M*k1, alpha*(2*pi)/M*(M-k2)];
H = Hk.*exp(j*angH);
h = real(ifft(H,M));
[db,mag,pha,grd,w] = freqz_m(h,1);
```

```
[a,ww,L,Hr] = FIR_t2(h);
```

a. Chuỗi mẫu tần số:

```
subplot(221);
plot(wl(1:31)/pi,Hk(1:31),'o',wdl,Hdr);
axis([0,1,-0.1,1.1]);
title('Cac mau tan so: M=60, T2 = 0.5925, T1 =
0.1099');
xlabel(f[*\pi]'); ylabel('Hr(k)');
```

b. Đáp ứng xung của bộ lọc thực tế:

```
subplot(222); stem(L,h);
axis([-1,M,-0.1,0.3]);
title('Dap ung xung');
xlabel('n'); ylabel('h(n)');
```

c. Biên độ của đáp ứng tần số:

```
subplot(223);
plot(ww/pi,Hr,wl(1:31)/pi,Hk(1:31),'o');
axis([0,1,-0.2,1.2]);
title('Bien do cua dap ung tan so');
xlabel('f[*\pi]'); ylabel('Hr(w)');
```

d. Biên độ của đáp ứng tần số theo dB:

```
subplot(224); plot(w/pi,db);
axis([0,1,-100,10]); grid
title('Bien do cua dap ung tan so ');
xlabel('f[*\pi]'); ylabel('dB');
```

5.4 PHƯƠNG PHÁP LẶP

Đáp ứng tần số của 4 loại bộ lọc FIR:

$$H(\omega) = P(\omega)Q(\omega)$$

Hàm sai số giữa bộ lọc thực tế và bộ lọc lý tưởng:

$$E(\omega) = W(\omega)[H_d(\omega) - H(\omega)] = W(\omega)Q(\omega)\left[\frac{H_d(\omega)}{Q(\omega)} - P(\omega)\right]$$

Ta định nghĩa:

$$\widehat{W}(\omega) = W(\omega)Q(\omega) \quad \widehat{H}_d(\omega) = \frac{H_d(\omega)}{Q(\omega)}$$

Khi đó:

$$E(\omega) = \widehat{W}(\omega)\left[\widehat{H}_d(\omega) - P(\omega)\right]$$

Parks và McClellan đã đưa ra giải pháp sử dụng thuật toán Remez để tìm ra đáp ứng xung của bộ lọc tối ưu, tức là gần đúng theo nghĩa Chebyshev đối với một bộ lọc lý tưởng, cho giá trị M là chiều dài của chuỗi đáp ứng xung với các điều kiện ràng buộc về độ gợn sóng ở dải thông và dải chắn như sau:

1. Xác định loại bộ lọc, tính giá trị R và xây dựng các hàm P,Q,W

Loại bộ lọc	**R**	**P(ω)**	**Q(ω)**
FIR loại 1	$\frac{M-1}{2}$	$\sum_{n=0}^{R} \bar{a}(n)cos\omega n$	1
FIR loại 2	$\frac{M}{2}-1$	$\sum_{n=0}^{R} \bar{b}(n)cos\omega n$	cos(ω/2)
FIR loại 3	$\frac{M-1}{2}-1$	$\sum_{n=0}^{R} \bar{c}(n)cos\omega n$	sinω
FIR loại 4	$\frac{M}{2}-1$	$\sum_{n=0}^{R} \bar{d}(n)cos\omega n$	sin(/2)ω

2. Xây dựng bài toán gần đúng bằng cách xác định các hàm $W(\omega)$,$Hd(\omega)$.

3. Sử dụng thuật toán trao đổi Remez để tìm ra hàm tối ưu P(ω).

 Chọn lấy R+2 điểm rời rạc, giả sử đó là các cực trị của hàm sai số.

 Trên cơ sở tại R+2 điểm rời rạc nói trên, hàm E(ω) luân phiên đổi dấu và có trị tuyệt đối bằng một giá trị δ nào đó, tính nội suy lại giá trị δ và hàm P(ω), từ đó tính ra hàm sai số E(ω), tính được cực trị thực của hàm sai số.

 Xem xét xem các giá trị rời rạc được chọn ban đầu có thực sự là các điểm mà hàm sai số E(ω) đạt cực trị và có trị tuyệt đối bằng nhau hay không. Nếu không, tìm các điểm tại đó E(ω) đạt cực trị.

 Trong các điểm cực trị của E(ω) lấy ra R+2 điểm và quay về lặp lại từ bước 2.

 Lặp lại các bước 2, 3, và 4 cho đến khi tập hợp các điểm rời rạc hội tụ.

 Từ tập các điểm rời rạc cuối cùng, tính ra hàm P(ω), từ đó tính ra các hệ số của P(ω).

4. Tính các giá trị của chuỗi đáp ứng xung h(n).

Khi chọn giá trị M càng chuẩn thì kết quả là thu được bộ lọc có hàm đáp ứng tần số càng gần với yêu cầu bài toán. Nếu như với giá trị M nào đó mà chưa thoả mãn được yêu cầu thì phải tăng giá trị M đến khi nào thoả mãn các điều kiện ràng buộc cho δp và δs (hay As và Rp). Một công thức lựa chọn ban đầu cho chiều dài M của đáp ứng xung là:

$$M_0 = \frac{-20log\sqrt{\delta_1\delta_2} - 13}{14.6\Delta f} \text{ với } \Delta f = \frac{\omega_s - \omega_p}{2\pi}$$

Trong MATLAB, tìm đáp ứng xung của bộ lọc tối ưu với giá trị M và hàm đáp ứng tần số lý tưởng cho trước được thực hiện bởi hàm ***firpm***.

Bài 5.8 Tạo script file sau để biểu diễn bộ lọc trên đồ thị:

```
wp = 0.2*pi; ws =0.3*pi; Rp = 0.25; As = 50;
delta_w = 2*pi/1000;
wsi = ws/delta_w+1;
delta1 = (10^(Rp/20)-1)/(10^(Rp/20)+1);
delta2 = (1+delta1)*(10^(-As/20));
deltaH = max(delta1,delta2);
deltaL = min(delta1,delta2);
weights = [delta2/delta1 1];
```

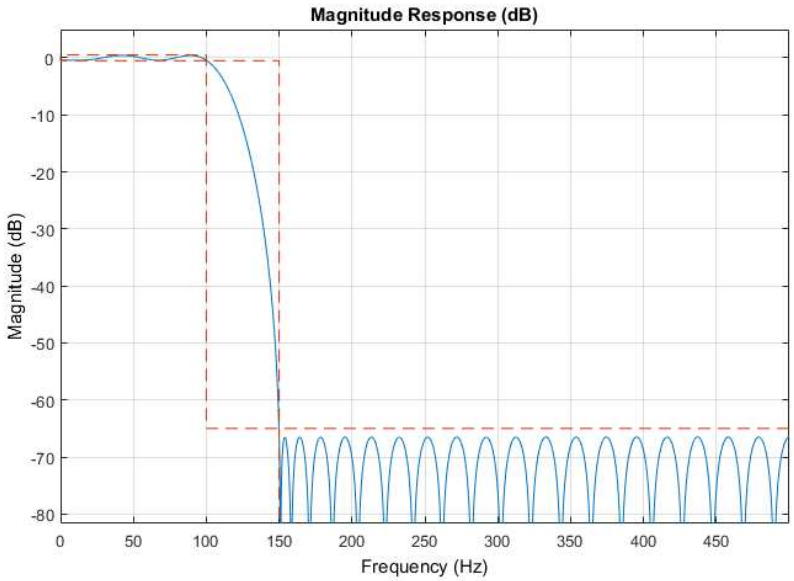
```
deltaf = (ws-wp)/(2*pi);
M=ceil((-20*log10(sqrt(delta1*delta2))-
13)/(14.6*deltaf)+1);
f = [0 wp/pi ws/pi 1];
m = [1 1 0 0];
h = firpm(M-1,f,m,weights);
[db,mag,pha,grd,w] = freqz_m(h,[1]);
Asd = -max(db(wsi:1:501))
while Asd<As
M = M+1
[h,ERR,RES] = firpm(M-1,f,m,weights);
[db,mag,pha,grd,w] = freqz_m(h,[1]);
Asd = -max(db(wsi:1:501))
end
n = [0:1:M-1];
subplot(221); stem(n,h);
axis([0,M-1,-0.1,0.3]);
title('Dap ung xung');
xlabel('n'); ylabel('h(n)');
subplot(222); plot(w/pi,db); grid;
axis([0,1,-80,10]);
title('Dap ung tan so theo dB');
xlabel('f[*\pi]'); ylabel('dB');
subplot(223); plot(w/pi,mag); grid;
axis([0,1,-0.2,1.2]);
title('Dap ung tan so');
xlabel('f[*\pi]'); ylabel('Hr(\omega)');
subplot(224); plot(RES.fgrid,RES.error); grid;
axis([0,1,-0.0150,0.0150]);
title('Dap ung tan so cua loi');
```

```
  xlabel('f[*\pi]'); ylabel('Er(\omega)');
```

Bài 5.9. Thiết kế bộ lọc thông thấp FIR có các đặc điểm kỹ thuật như sau:

Equiripple Design

```
Fpass = 100;
Fstop = 150;
Apass = 1;
Astop = 65;
Fs = 1e3;

d = designfilt('lowpassfir', ...
'PassbandFrequency',Fpass,'StopbandFrequency',Fstop, ...
'PassbandRipple',Apass,'StopbandAttenuation',Astop, ...
'DesignMethod','equiripple','SampleRate',Fs);

fvtool(d)
```



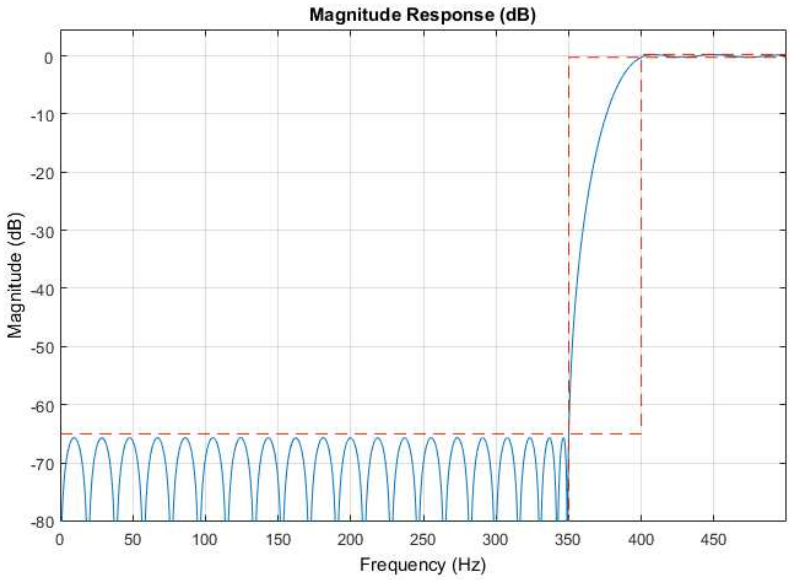
Hình 5.2 Đáp ứng bộ lọc thông thấp Lowpass Filter FIR

Bài 5.10. Thiết kế bộ lọc thông cao FIR có các đặc điểm kỹ thuật như sau:

Highpass FIR Filters

Equiripple Design

```
Fstop = 350;
Fpass = 400;
Astop = 65;
Apass = 0.5;
Fs = 1e3;

d=designfilt('highpassfir','StopbandFrequency',Fstop,...
'PassbandFrequency',Fpass,'StopbandAttenuation',Astop,..
'PassbandRipple',Apass,'SampleRate',Fs,'DesignMethod','e
quiripple');
fvtool(d)
```



Hình 5.3 Đáp ứng bộ lọc thông caoHighpass Filter FIR

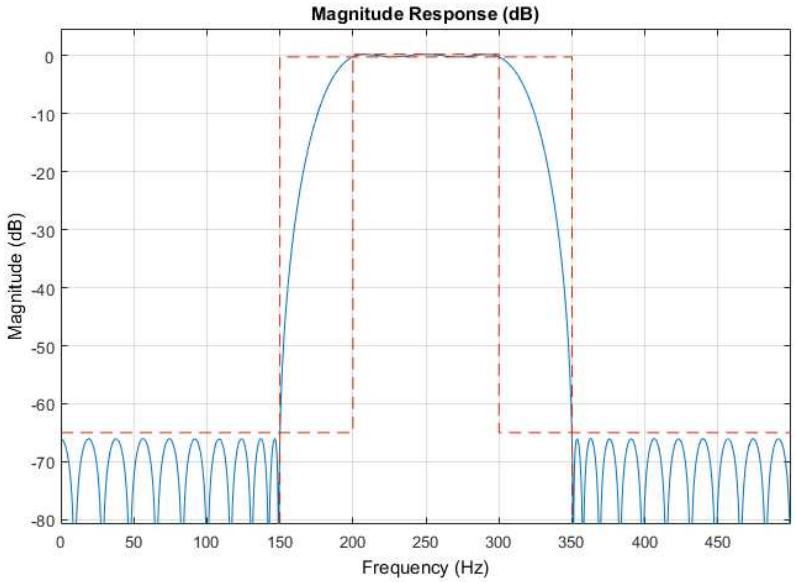
Bài 5.11. Thiết kế bộ lọc thông dãi FIR có các đặc điểm kỹ thuật như sau:

Bandpass FIR Filters

Equiripple Design

```
% Su dung phuong phap  Equiripple Design
Fstop1 = 150;
Fpass1 = 200;
Fpass2 = 300;
```

```
Fstop2 = 350;
Astop1 = 65;
Apass  = 0.5;
Astop2 = 65;
Fs = 1e3;

d = designfilt('bandpassfir', ...
'StopbandFrequency1',Fstop1,'PassbandFrequency1',Fpass1, ...
'PassbandFrequency2',Fpass2,'StopbandFrequency2',Fstop2, ...
'StopbandAttenuation1',Astop1,'PassbandRipple', Apass, ...
'StopbandAttenuation2',Astop2, ...
'DesignMethod','equiripple','SampleRate',Fs);

fvtool(d)
```



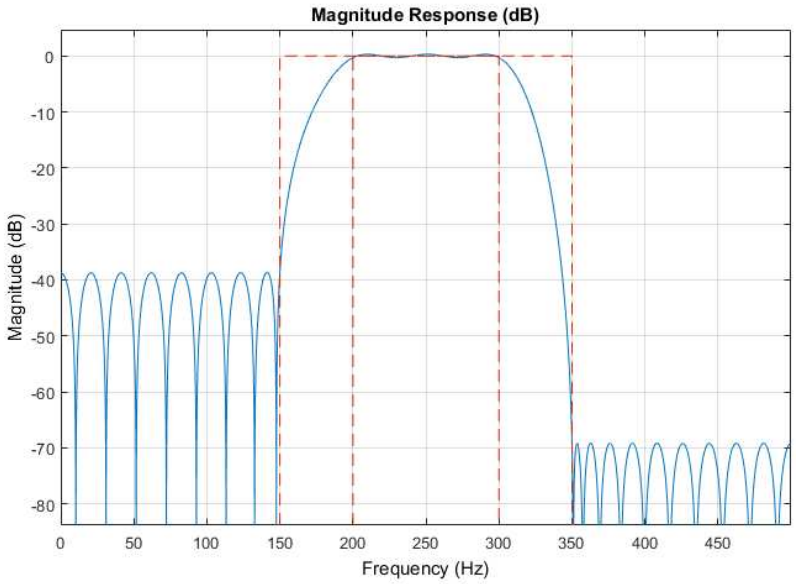
Hình 5.4 Đáp ứng bộ lọc thông dãi Bandpass Filter FIR

Bài 5.12. Thiết kế bộ lọc thông dãi FIR bậc 50 có các đặc điểm kỹ thuật như sau:

Asymmetric Band Attenuations

```
% Su dung phuong phap  Asymmetric Band Attenuations
N = 50;
```

```
Fstop1 = 150;
Fpass1 = 200;
Fpass2 = 300;
Fstop2 = 350;
Wstop1 = 3;
Wstop2 = 100;
Fs = 1e3;

d = designfilt('bandpassfir', ...
  'FilterOrder',N, ...
'StopbandFrequency1',Fstop1,'PassbandFrequency1', Fpass1,...
'PassbandFrequency2',Fpass2,'StopbandFrequency2', Fstop2,...
'StopbandWeight1',Wstop1,'StopbandWeight2',Wstop2, ...
'DesignMethod','equiripple','SampleRate',Fs);
fvtool(d)
```



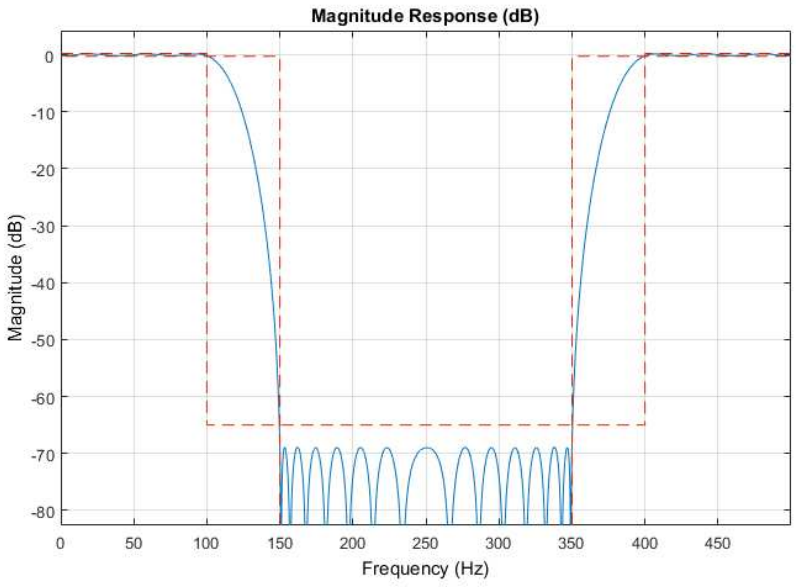
Hình 5.5 Đáp ứng bộ lọc thông dãi Bandpass Filter FIR bậc 50

Bài 5.13. Thiết kế bộ lọc chắn dãi FIR có các đặc điểm kỹ thuật như sau:

Bandstop FIR Filters

Equiripple Design

```
% Ky thuat Equiripple Design
Fpass1 = 100;
Fstop1 = 150;
Fstop2 = 350;
Fpass2 = 400;
Apass1 = 0.5;
Astop  = 65;
Apass2 = 0.5;
Fs = 1e3;

d = designfilt('bandstopfir', ...
'PassbandFrequency1',Fpass1,'StopbandFrequency1',Fstop1, ...
'StopbandFrequency2',Fstop2,'PassbandFrequency2',Fpass2, ...
'PassbandRipple1',Apass1,'StopbandAttenuation',Astop, ...
'PassbandRipple2', Apass2, ...
'DesignMethod','equiripple','SampleRate', Fs);

fvtool(d)
```



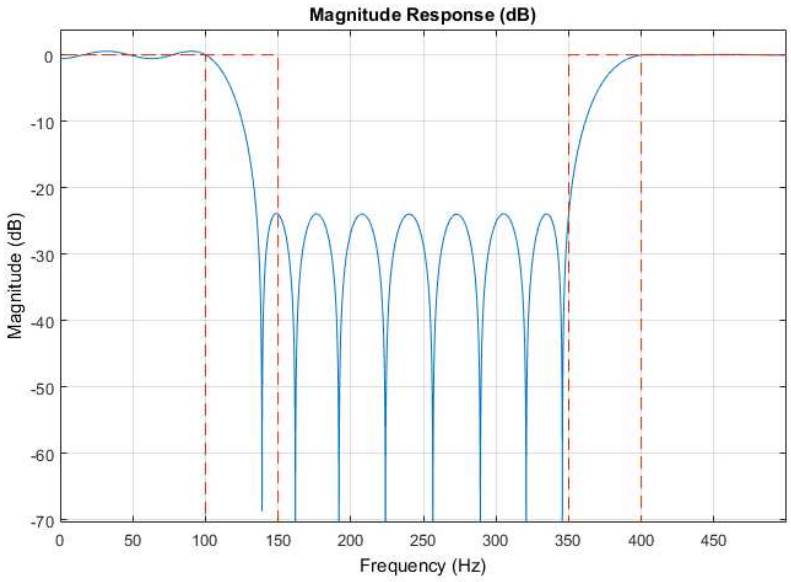
Hình 5.6. Đáp ứng bộ lọc chắn dãi Bandstop Filter FIR

Bài 5.14. Thiết kế bộ lọc chắn dãi FIR bậc 30 có các đặc điểm kỹ thuật như sau:

Bandstop FIR Filters

Asymmetric Passband Ripples

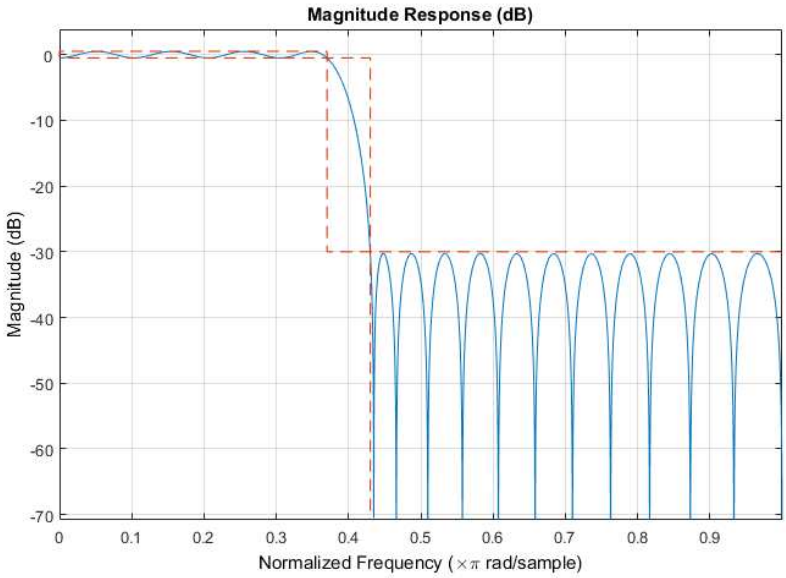
```
% ky thuat Asymmetric Passband Ripples
N = 30;
Fpass1 = 100;
Fstop1 = 150;
Fstop2 = 350;
Fpass2 = 400;
Wpass1 = 1;
Wpass2  = 10;
Fs = 1e3;

d = designfilt('bandstopfir', ...
'FilterOrder',N, ...
'PassbandFrequency1',Fpass1,'StopbandFrequency1',Fstop1, ...
'StopbandFrequency2',Fstop2,'PassbandFrequency2',Fpass2, ...
'PassbandWeight1',Wpass1,'PassbandWeight2',Wpass2, ...
'DesignMethod','equiripple','SampleRate', Fs);
fvtool(d)
```



Hình 5.7 Đáp ứng bộ lọc chắn dãi Bandstop Filter FIR bậc 30

Bài 5.15. Thiết kế bộ lọc thông thấp FIR có đặc điểm sau:

```
Fpass = 0.37;
Fstop = 0.43;
Ap = 1;
Ast = 30;

d =
designfilt('lowpassfir','PassbandFrequency',Fpass,...

'StopbandFrequency',Fstop,'PassbandRipple',Ap,'StopbandA
ttenuation',Ast);

hfvt = fvtool(d);
```



Hình 5.8 Đáp ứng bộ lọc thông thấp FIR

Bậc của bộ lọc được tính bằng hàm filtord :

```
N = filtord(d)
N =

    39
```

Kết quả : cho bậc của bộ lọc bằng 39.

Sử dụng hàm info để biết các thông số thiết kế bộ lọc:

```
info(d)
ans =

FIR Digital Filter (real)
-------------------------
Filter Length  : 40
Stable         : Yes
Linear Phase   : Yes (Type 2)

Design Method Information
```

```
Design Algorithm : Equiripple

Design Specifications
Sample Rate     : N/A (normalized frequency)
Response        : Lowpass
Specification   : Fp,Fst,Ap,Ast
Stopband Atten. : 30 dB
Passband Ripple : 1 dB
Stopband Edge   : 0.43
Passband Edge   : 0.37
```

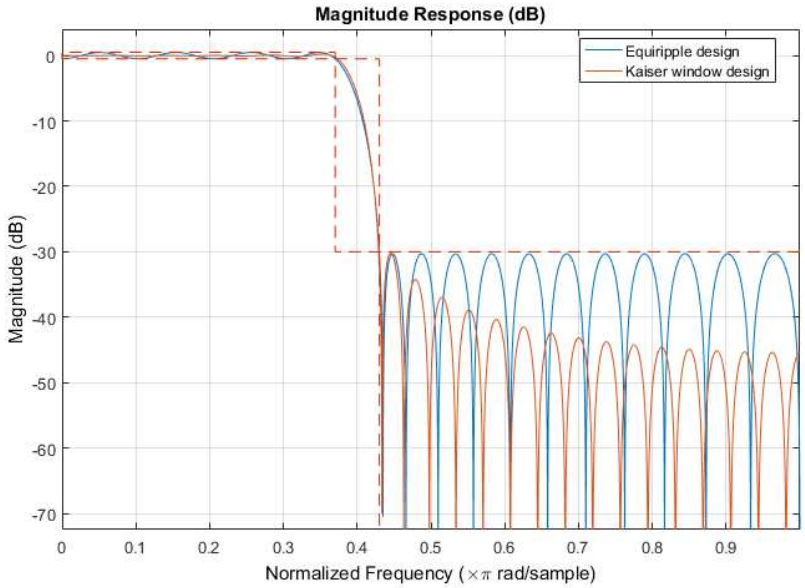
Bài 5.16. : Sử dụng hàm **designfilt** để thiết kế bộ lọc thep phương pháp cửa sổ Kaiser

```
dk =
designfilt('lowpassfir','PassbandFrequency',Fpass,...
  'StopbandFrequency',Fstop,'PassbandRipple',Ap,...
  'StopbandAttenuation',Ast, 'DesignMethod',
'kaiserwin');

addfilter(hfvt,dk);
legend(hfvt,'Equiripple design', 'Kaiser window design')

N = filtord(dk)
```

Kết quả:

```
N =

    52
```

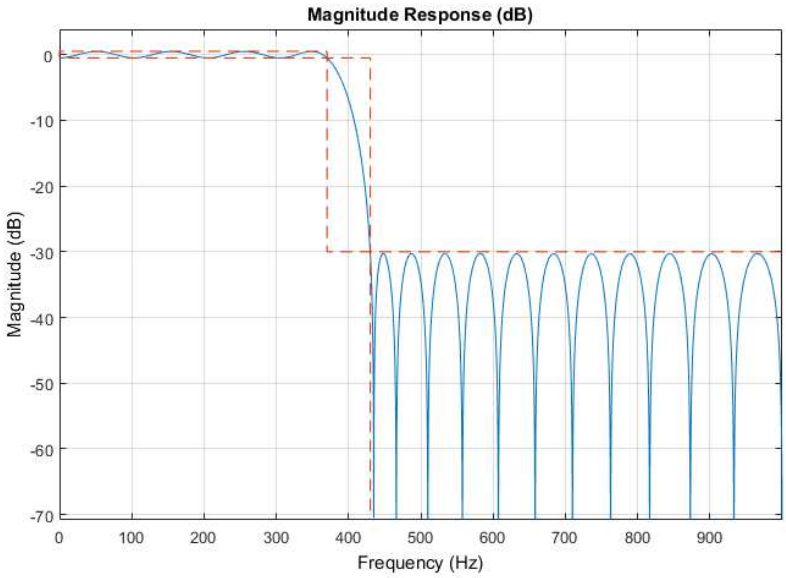
Bậc của bộ lọc bằng 52



Hình 5.9 Đáp ứng bộ lọc thông thấp FIR thiết kế bộ lọc thep phương pháp cửa sổ Kaiser

Áp dụng với tần số lấy mẫu Fs=2000 Hz

```
Fpass = 370;
Fstop = 430;
Ap = 1;
Ast = 30;
Fs = 2000;

d =
designfilt('lowpassfir','PassbandFrequency',Fpass,...
  'StopbandFrequency',Fstop,'PassbandRipple',Ap,...
  'StopbandAttenuation',Ast,'SampleRate',Fs);

hfvt = fvtool(d);
```

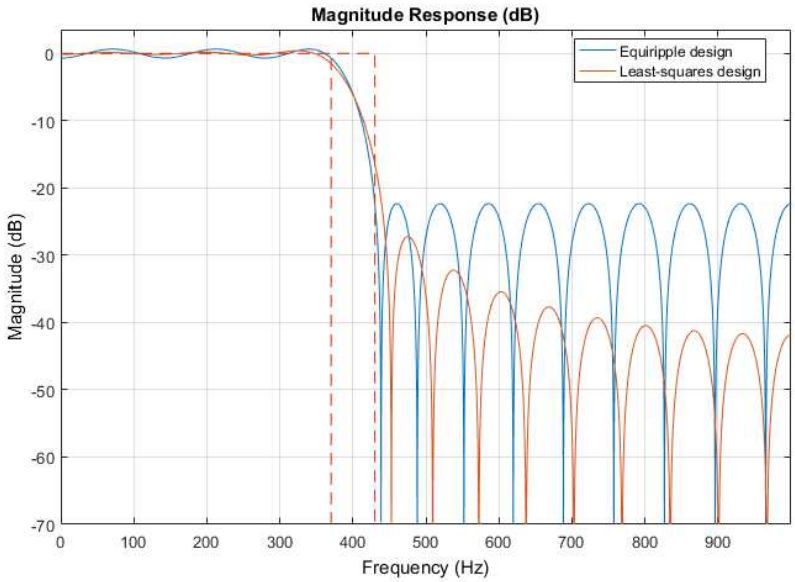


Hình 5.10 Đáp ứng bộ lọc thông thấp FIR thiết kế bộ lọc thep phương pháp cửa sổ Kaiser với tần số lấy mẫu Fs=2000 Hz

Bài 5.17: Thiết kế bộ lọc thông thấp FIR bậc 30, có dãi thông passband 370 hz, stopband 430 Hz, tần số lấy mẫu 2 kHz. Sử dụng hai phương pháp thiết kế dùng kỹ thuật **equiripple** và bình phương tối thiểu.

```
N = 30;
Fpass = 370;
Fstop = 430;
Fs = 2000;

% Design method defaults to 'equiripple' when omitted
deq =
designfilt('lowpassfir','FilterOrder',N,'PassbandFrequen
cy',Fpass,...
  'StopbandFrequency',Fstop,'SampleRate',Fs);

dls =
designfilt('lowpassfir','FilterOrder',N,'PassbandFrequen
cy',Fpass,...
```

```
'StopbandFrequency',Fstop,'SampleRate',Fs,'DesignMethod'
,'ls');

hfvt = fvtool(deq,dls);
legend(hfvt,'Equiripple design', 'Least-squares design')
```

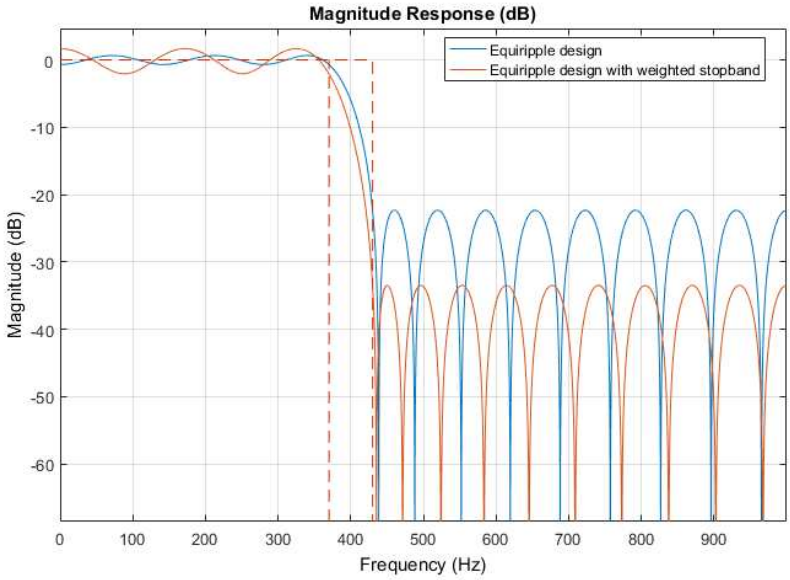


Hình 5.11 Đáp ứng bộ lọc thông thấp FIR thiết kế bộ lọc thep phương pháp equiripple và bình phương tối thiểu

Nếu muốn giảm tiết kiệm năng lượng, thiết kế tiếp theo cần chọn giá trị cho PassbandWeight và StopbandWeight như sau:

```
deqw =
designfilt('lowpassfir','FilterOrder',N,'PassbandFrequen
cy',Fpass,...
  'StopbandFrequency',Fstop,'SampleRate',Fs,...
  'PassbandWeight',1,'StopbandWeight',10);

hfvt = fvtool(deq,deqw);
```

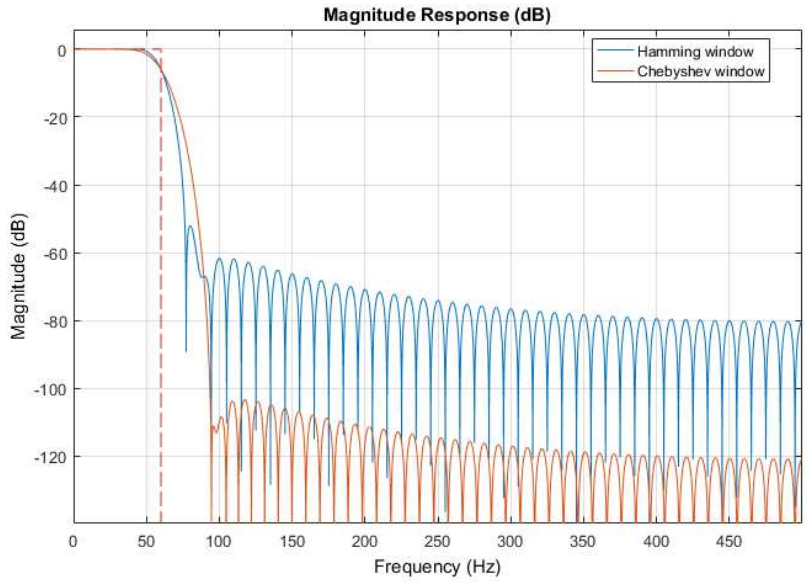
```
legend(hfvt,'Equiripple design', 'Equiripple design with
weighted stopband')
```



Hình 5.12 Đáp ứng bộ lọc thông thấp FIR thiết kế bộ lọc thep phương pháp equiripple và bình phương tối thiểu được điều chỉnh tiết kiệm năng lượng

Bài 5.18 Thiết kế bộ lọc FIR 100 bậc dùng phương pháp cửa sổ Hamming và Chebyshev , độ suy giảm 90 dB với tần số cắt 60hz, tốc độ lấy mẫu 1 kHz

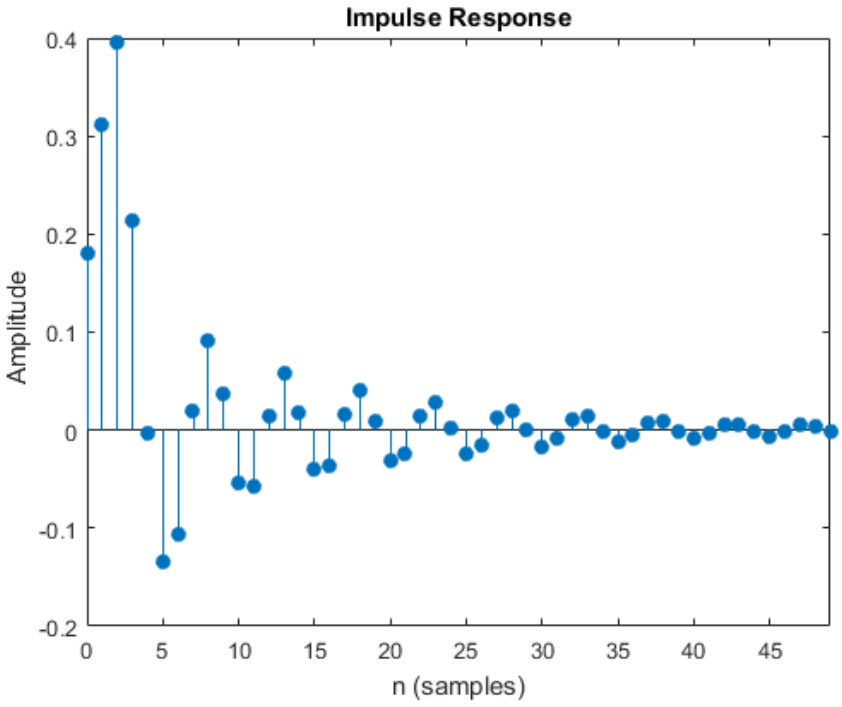
```
dhamming =
designfilt('lowpassfir','FilterOrder',100,'CutoffFrequen
cy',60,...
  'SampleRate',1000,'Window','hamming');

dchebwin =
designfilt('lowpassfir','FilterOrder',100,'CutoffFrequen
cy',60,...
  'SampleRate',1000,'Window',{'chebwin',90});

hfvt = fvtool(dhamming,dchebwin);
legend(hfvt,'Hamming window', 'Chebyshev window')
```



Hình 5.13 Đáp ứng bộ lọc thông thấp FIR thiết kế bộ lọc theo phương pháp phương pháp cửa sổ Hamming và Chebyshev

Bài 5.19 Vẽ đáp ứng xung của các bộ lọc thông thấp eliip

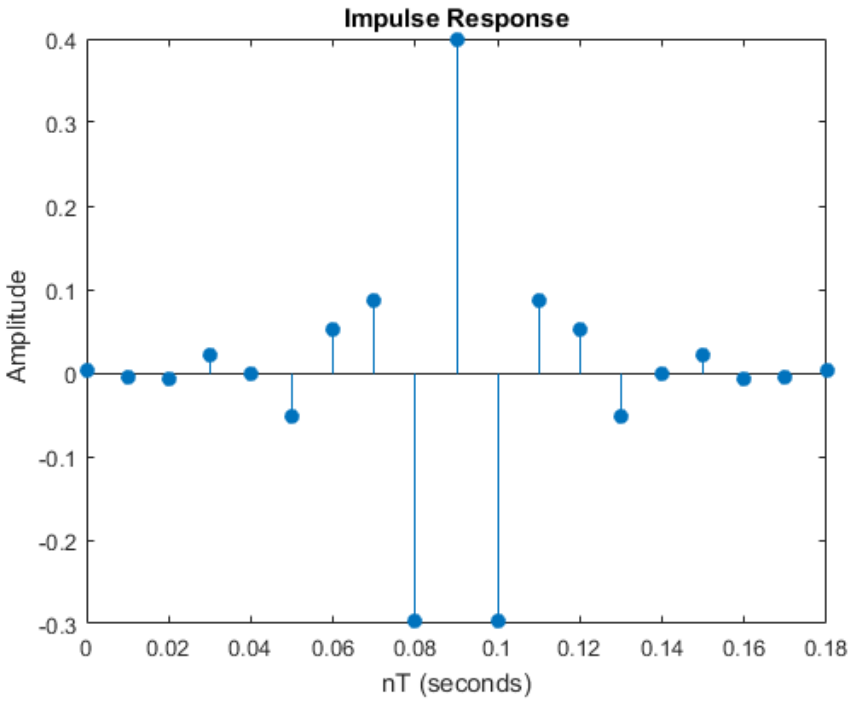
```
[[b,a] = ellip(4,0.5,20,0.4);
impz(b,a,50)
```



Hình 5.14 Đáp ứng xung bộ lọc thông thấp FIR eliip

Bài 5.20: Đáp ứng xung của bộ lọc thông cao bậc 18, dùng cửa sổ Kaiser với β=4, tốc độ lấy mẫu 100 Hz, tần số cắt 30 Hz.

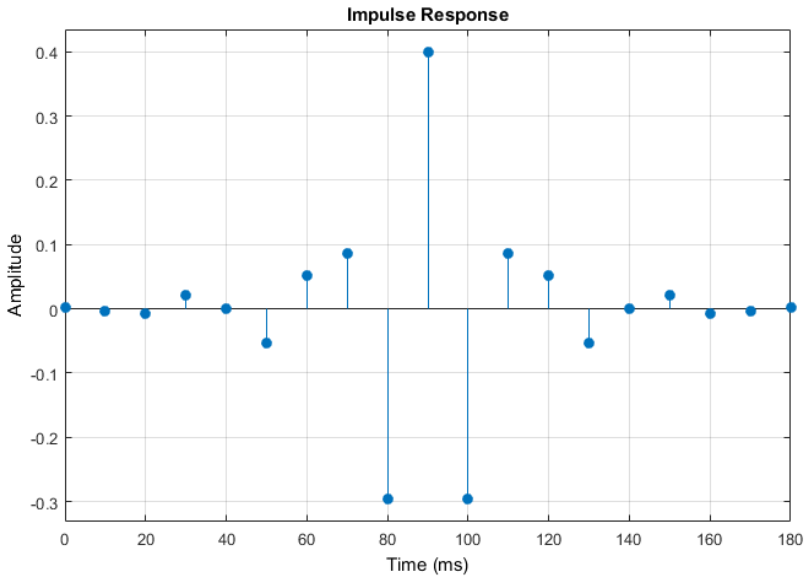
```
b = fir1(18,30/(100/2),'high',kaiser(19,4));
impz(b,1,[],100)
```



Hình 5.15 Đáp ứng xung bộ lọc thông cao bậc 18, dùng cửa sổ Kaiser

Trường hợp thiết kế bộ lọc dùng lệnh dedignfilt, chương trình vẽ đáp ứng xung như sau:

```
d =
designfilt('highpassfir','FilterOrder',18,'SampleRate',1
00, ...

'CutoffFrequency',30,'Window',{'kaiser',4});
impz(d,[],100)
```



Hình 5.16 Đáp ứng xung bộ lọc thông cao bậc 18, dùng cửa sổ Kaiser dùng hàm `designfilt`.

5.5 FDATool

Sử dụng FDATool thay thế cho các chức năng thiết kế bộ lọc bằng viết lệnh.

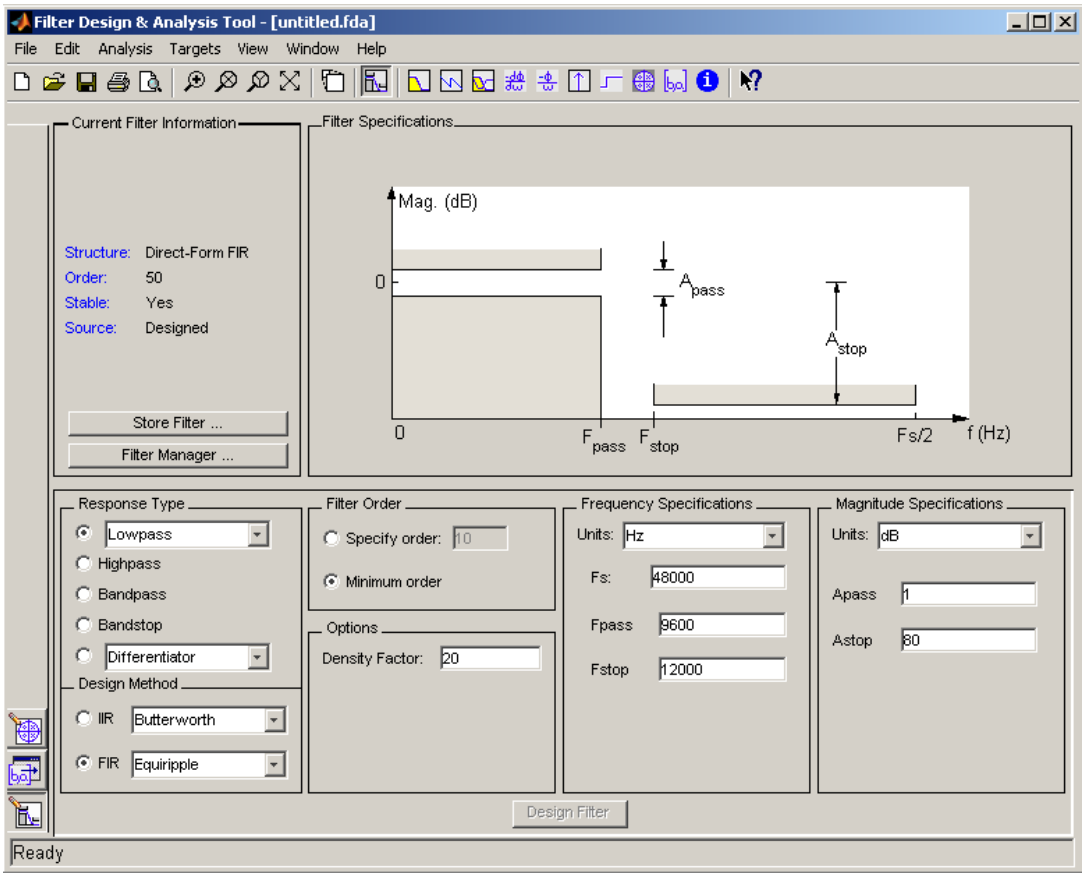
Lọc Thiết kế và Công cụ phân tích (FDATool) là một giao diện mạnh đồ họa người dùng (GUI) trong xử lý tín hiệu Toolbox ™ cho việc thiết kế và phân tích các bộ lọc.

FDATool cho phép nhanh chóng thiết kế FIR hoặc các bộ lọc IIR bằng cách thiết lập thông số kỹ thuật hiệu suất lọc, bằng cách nhập các bộ lọc từ không gian làm việc MATLAB® hoặc bằng cách thêm, di chuyển , xoá các điểm cực và điểm không . FDATool cũng cung cấp các công cụ việc phân tích bộ lọc.

Bắt đầu, gõ fdatool tại dấu nhắc lệnh MATLAB:

fdatool

Sau đó, các giao diện hiển thị bộ lọc như sau:



Hình 5.17 Giao diện thiết kế bộ lọc FDA Tool

Bài 5.21 Thiết kế một bộ lọc dùng FDA Tool

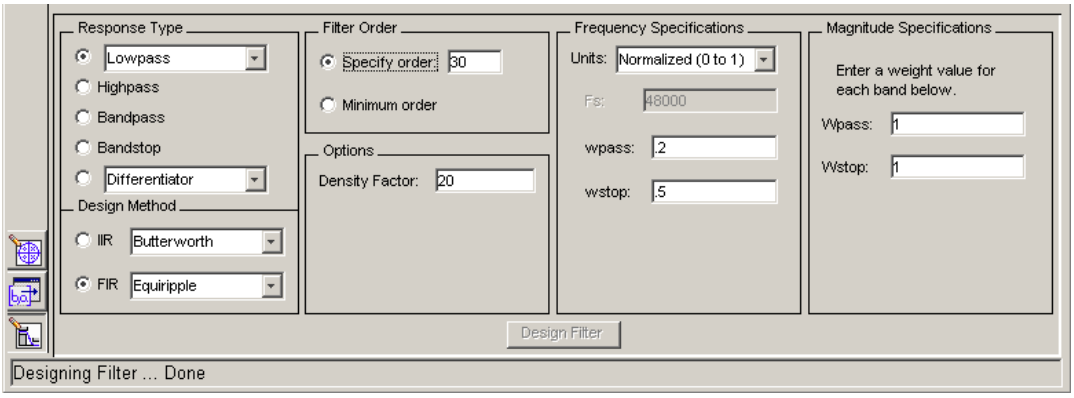
Thiết kế một bộ lọc thông thấp , tất cả các tần số nhỏ hơn hoặc bằng 20% của tần số Nyquist (một nửa tần số lấy mẫu) và tần số lớn hơn hoặc bằng 50% của tần số Nyquist. Sử dụng một bộ lọc FIR Equiripple với các thông số kỹ thuật:

Suy giảm dải thông 1 dB

Stopband suy giảm 80 dB

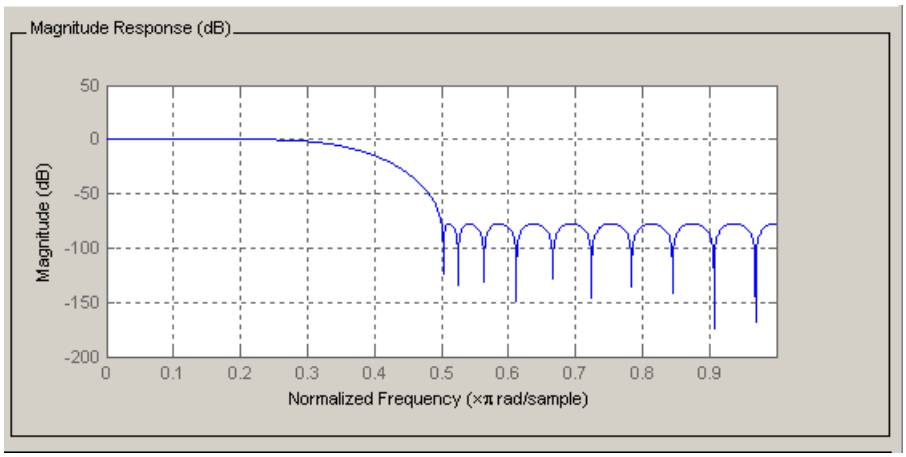
Một tần số dải thông 0,2 [Bình thường (0-1)]

Một tần số stopband 0.5 [Bình thường (0-1)]



Hình 5.18 Nhập các thông số trên giao diện thiết kế bộ lọc FDA Tool

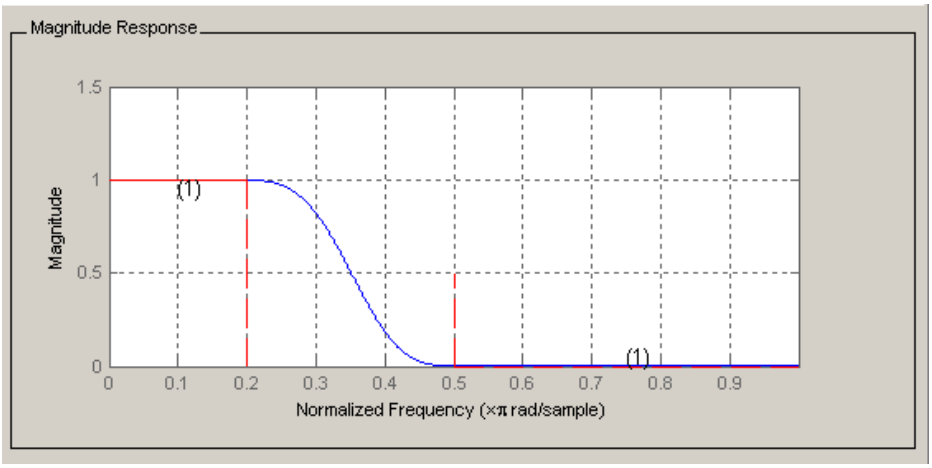
Ta có kết quả đáp ứng của bộ lọc như sau:



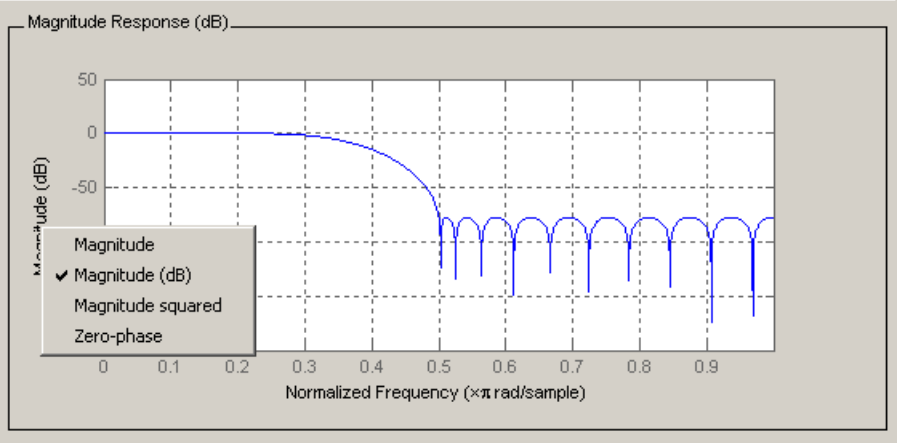
Hình 5.19 Đáp ứng của bộ lọc trên giao diện thiết kế FDA Tool

Sử dụng thanh công cụ dưới đây để xem các đáp ứng biên độ, đáp ứng pha, đáp ứng xung, điểm cực, điểm không.

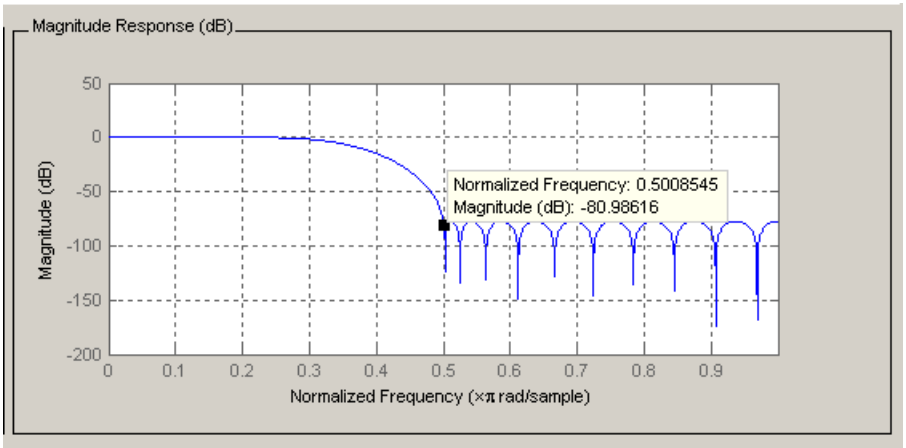
FDATool cho phép đo lường chặt chẽ thiết kế của đáp ứng các thông số kỹ thuật bộ lọc bằng cách sử dụng **Specification Mark.**



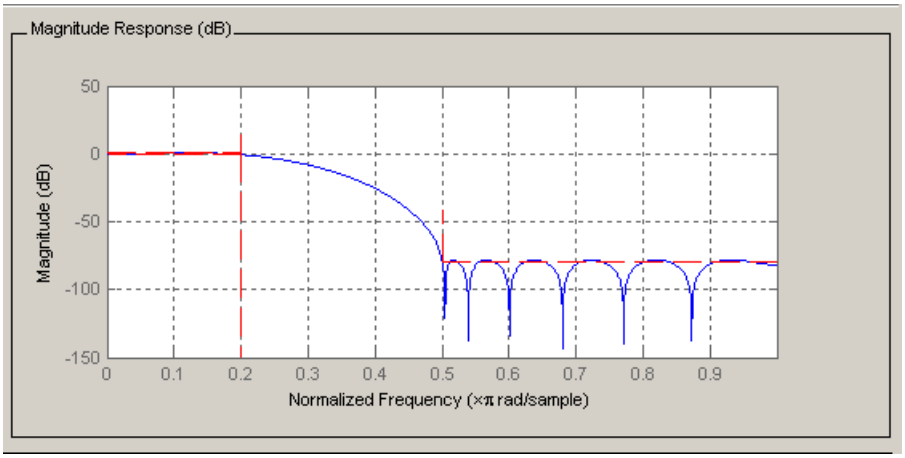
Thay đổi thang đo ở trục toạ độ



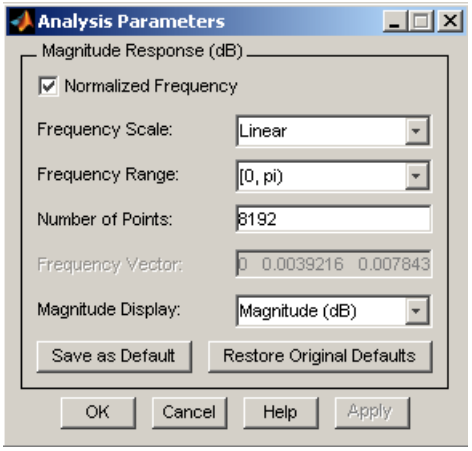
Hiển thị giá trị điểm toạ độ



Tối ưu hóa việc thiết kế: Sử dụng **Minimum Order** để tối ưu hoá bộ lọc



Thay đổi các thông số



Để lưu các thông số hiển thị nhấn Save as Default.

Để khôi phục lại các giá trị mặc định bấm Restore Original Defaults

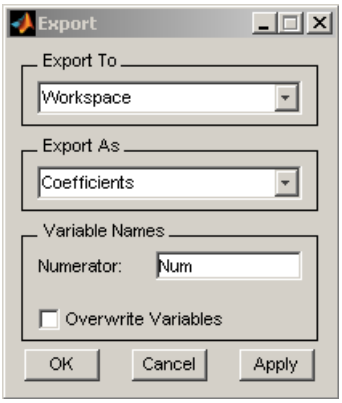
Khi hài lòng với thiết kế có thể xuất bộ lọc sang các dạng sau:

MATLAB workspace

File MAT

File Text

Chọn Export từ menu File.



Chương trình file MAT tương ứng như sau:

```
function B = minorderlowfir
%MINORDERLOWFIR Returns discrete-time filter coefficients.

% MATLAB Code
% Generated by MATLAB(R) 8.1 and the Signal Processing Toolbox 6.19.
% Generated on: 11-Oct-2012 12:19:05

% Equiripple Lowpass filter designed using the FIRPM function.

% All frequency values are normalized to 1.

Fpass = 0.2;             % Passband Frequency
Fstop = 0.5;             % Stopband Frequency
Dpass = 0.057501127785;  % Passband Ripple
Dstop = 0.0001;          % Stopband Attenuation
dens  = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass, Fstop], [1 0], [Dpass, Dstop]);

% Calculate the coefficients using the FIRPM function.
B = firpm(N, Fo, Ao, W, {dens});
```

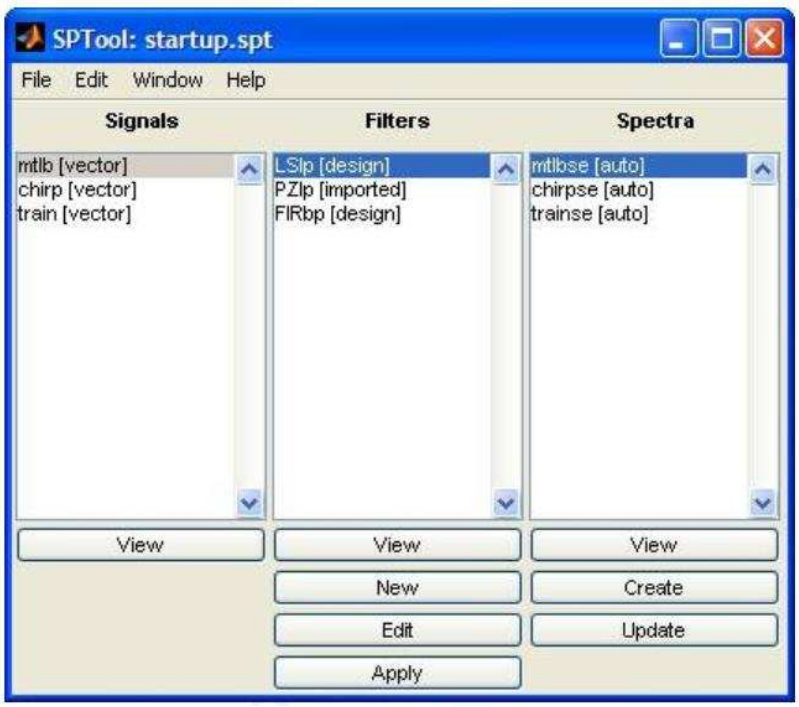
5.6. Thiết kế bộ lọc số bằng công cụ SPTool

SPTool là một công cụ có giao diện tương tác dùng cho xử lý số tín hiệu. Công cụ này có thể được sử dụng để phân tích tín hiệu, thiết kế các bộ lọc, phân tích các bộ lọc, lọc tín hiệu và phân tích phổ của tín hiệu.

Để khởi động SPTool, từ dấu nhắc lệnh của MATLAB, nhập lệnh

sptool

Khi đó, giao diện của SPTool sẽ xuất hiện như sau:



Hình 5.20 : Giao diện của SPTool

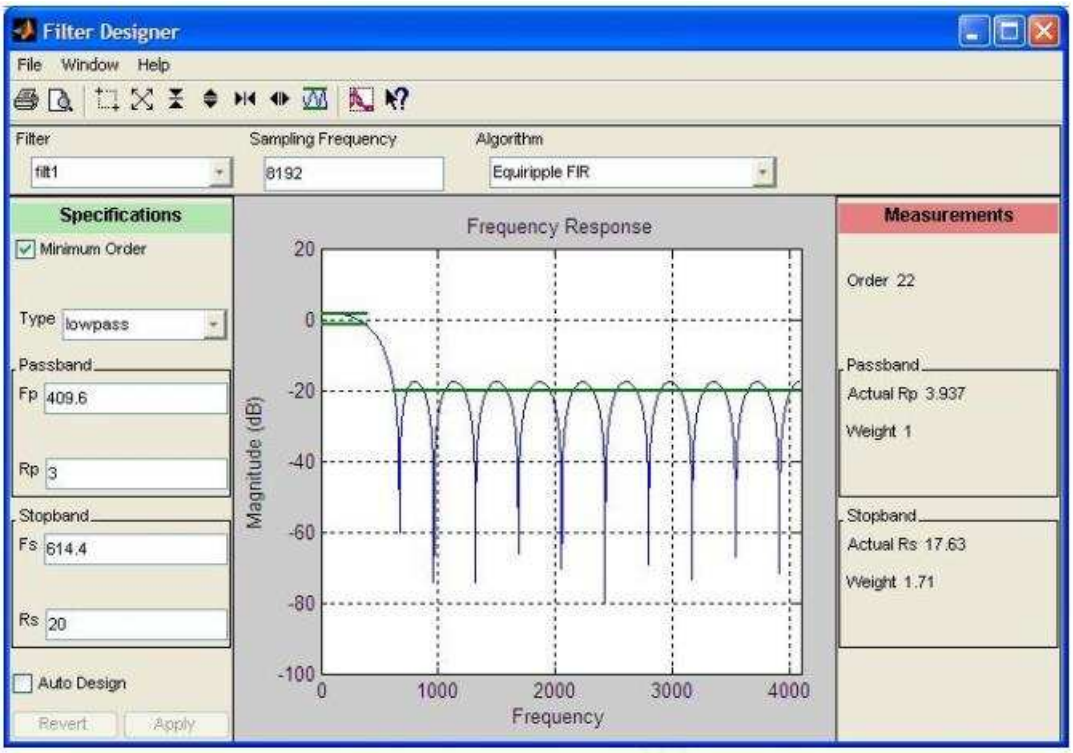
Khi mới mở SPTool, nó chứa một tập hợp các tín hiệu, bộ lọc và phổ mặc định. Trên giao diện của SPTool, có 3 cột: Signals, Filters và Spectra. Dưới mỗi cột có các nút sử dụng cho cột đó. Cột Signals hiển thị các tín hiệu, cột Filters hiển thị các bộ lọc và cột Spectra hiển thị các phổ trong workspace (vùng làm việc) của SPTool. Các tín hiệu, bộ lọc hoặc phổ trong workspace của MATLAB có thể được đưa vào SPTool bằng lệnh Import trong menu File của SPTool. Các tín hiệu, bộ lọc hoặc phổ được tạo ra hoặc được import vào SPTool tồn tại dưới dạng các cấu trúc của MATLAB. Để lưu lại các tín hiệu, bộ lọc và phổ đã tạo ra hoặc chỉnh sửa trong SPTool, sử dụng lệnh Export trong menu File, chúng cũng sẽ được lưu lại dưới dạng các cấu trúc MATLAB.

Để bắt đầu thiết kế một bộ lọc mới, nhấn vào nút New ngay dưới cột Filter. Khi đó, giao diện Filter Designer dùng để thiết kế bộ lọc như sau sẽ xuất hiện.

Filter Designer cung cấp một môi trường đồ họa tương tác để thiết kế các bộ lọc số IIR hoặc FIR dựa trên các tiêu chuẩn do người dùng xác định

- Các loại bộ lọc có thể thiết kế: Thông thấp, thông cao, thông dải, chắn dải.
- Các phương pháp thiết kế bộ lọc FIR: Equiripple, Least squares, Window

- Các phương pháp thiết kế bộ lọc IIR: Butterworth, Chebyshev loại I, Chebyshev loại II, Elliptic.



Hình 5.21: Giao diện Filter Design

Bài 5.22: Thiết kế một bộ lọc FIR chắn dải bằng SPTool

Bộ lọc, được thiết kế bằng phương pháp cửa sổ Kaiser, với các thông số sau:

Chiều dài của đáp ứng xung: N = 89 (MATLAB hiển thị bậc bộ lọc bằng 88)

Tần số trung tâm: 2700 Hz

Tần số cắt: 2500 Hz và 2900 Hz

Giá trị của $\beta = 4$

Tần số lấy mẫu 8000 Hz

Các bước thiết kế như sau:

1. Khởi động SPTool. Dưới cột Filters, nhấn nút New để mở cửa sổ Filter Designer.

2. Trong giao diện của Filter Designer:

a. Trong text box Filter: Tên bộ lọc được tự đặt (ở đây là filt1). Tên này có thể thay đổi sau này.

b. Nhập các thông số thiết kế vào:

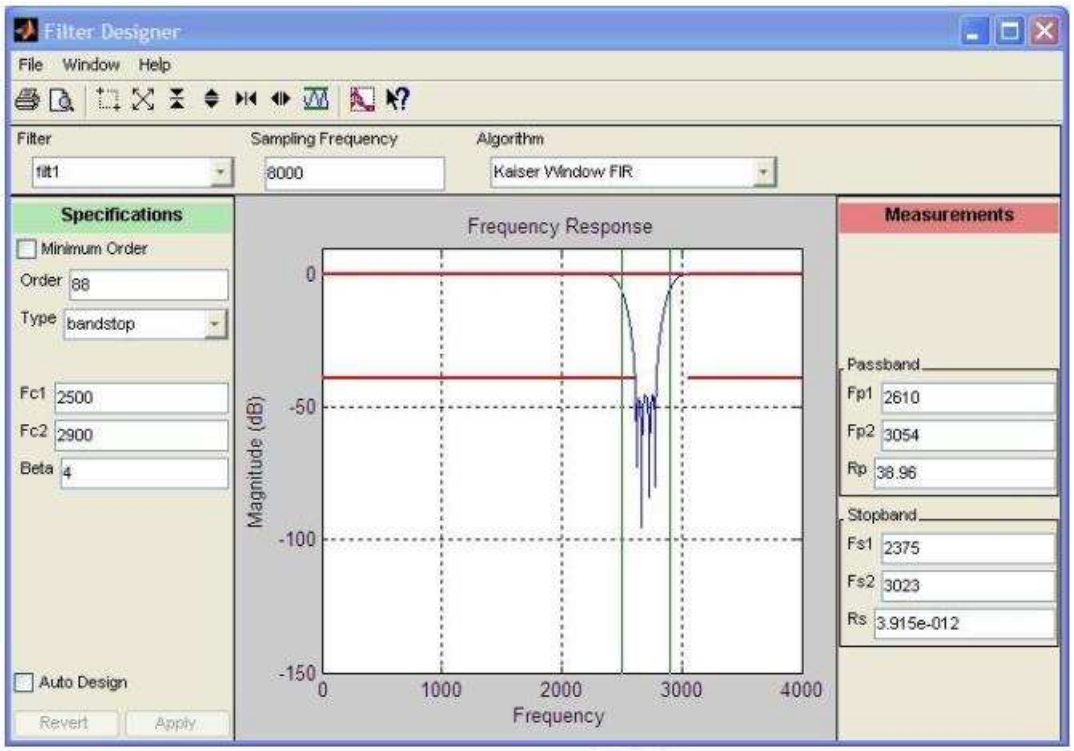
i. Sampling Frequency = 8000

ii. Algorithm: Kaiser Window FIR

iii. Bỏ chọn ở check box Minimum Order. (nếu chọn thì sẽ thiết kế bộ lọc có bậc tối thiểu).

iv. Filter Order = 88, Type = Bandstop, Fc1 = 2500, Fc2 = 2900, Beta =4

c. Nhấn Apply. Khi đó đáp ứng tần số của bộ lọc thiết kế sẽ được hiển thị.



Hình 5.22: Đáp ứng tần số của bộ lọc đã thiết kế

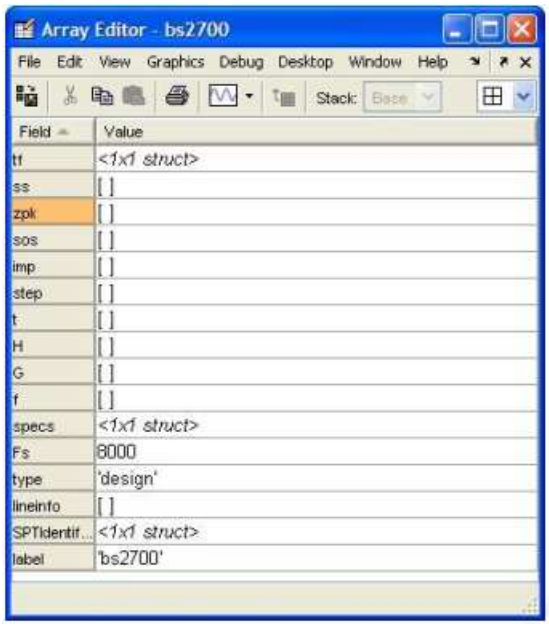
3. Trở về cửa sổ SPTool, trong cột Filters sẽ xuất hiện thêm một dòng filt1 [design]. Đây chính là bộ lọc vừa thiết kế. Sau này, nếu muốn sửa đổi thiết kế, chọn lại tên bộ lọc và nhấn nút Edit ở phía dưới. Để dễ nhớ, ta sẽ thay đổi tên bộ lọc bằng cách chọn Edit/ Name/filt1 [design]. Trong cửa sổ mới xuất hiện, có thể nhập tên mới.

Khi thiết kế một bộ lọc FIR như trên, kết quả mà ta cần nhận được sau khi thiết kế là các giá trị của vector đáp ứng xung h của bộ lọc thiết kế. Để lấy các giá trị của vector đáp ứng xung, ta thực hiện như sau:

1. Từ cửa sổ SPTool, chọn File/Export… Trong Export list xuất hiện, chọn Filter: [design] rồi nhấn nút Export to workspace

2. Đóng cửa sổ SPTool lại. Một thông báo xuất hiện hỏi có muốn lưu lại phiên làm việc hiện tại hay không. Nếu muốn lưu lại, chọn Save.

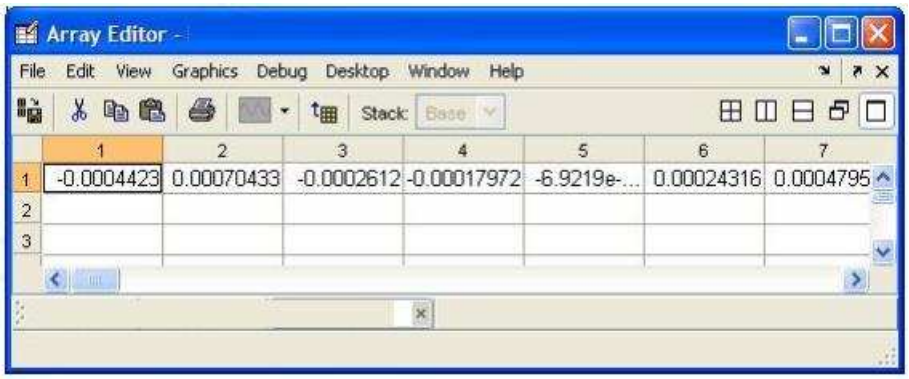
3. Mở cửa sổ Workspace của MATLAB, ta sẽ thấy trong workspace sẽ xuất hiện biến mới là tên bộ lọc. Đây chính là bộ lọc mà ta đã thiết kế trong SPTool và xuất ra workspace của MATLAB. Biến này được lưu dưới dạng một cấu trúc mô tả bộ lọc đã thiết kế. Nhấn đúp chuột vào tên biến trong workspace, ta sẽ thấy được các field của cấu trúc này như sau:



Hình 5.23: Các field của bộ lọc

4. Trong các field này, field tf thể hiện hàm truyền của bộ lọc. Field này cũng là một cấu trúc gồm 2 field: tf.num và tf.den thể hiện tương ứng các hệ số của đa thức tử số và đa thức mẫu số. Đối với bộ lọc FIR, hàm truyền chỉ có tử số và các hệ số của tử số chính là đáp ứng xung của bộ lọc. Do đó, với bộ lọc trên, các giá trị của vector đáp ứng xung được lưu trong filt1.tf.num. Trong cửa sổ Array Editor trên, lần lượt nhấn đúp vào field tf rồi nhấn đúp vào num, ta sẽ thấy các hệ số đáp ứng xung của bộ lọc. Để gán các hệ số này vào một vector h, trong MATLAB có thể dùng lệnh sau:

h=filter1.tf.num



Hình 5. 24. Vector đáp ứng xung của bộ lọc đã thiết kế

Các giá trị thu được là giá trị của đáp ứng xung của bộ lọc đã thiết kế

5.6 SO SÁNH CÁC PHƯƠNG PHÁP THIẾT KẾ:

Bài 5.23 : Thiết kế bộ lọc thông thấp FIR sử dụng hàm fir1 và designfilt, và fdatool.

Với ngõ vào là tín hiệu tín hiệu sóng sin 100 Hz với nhiễu Gaussian trắng.

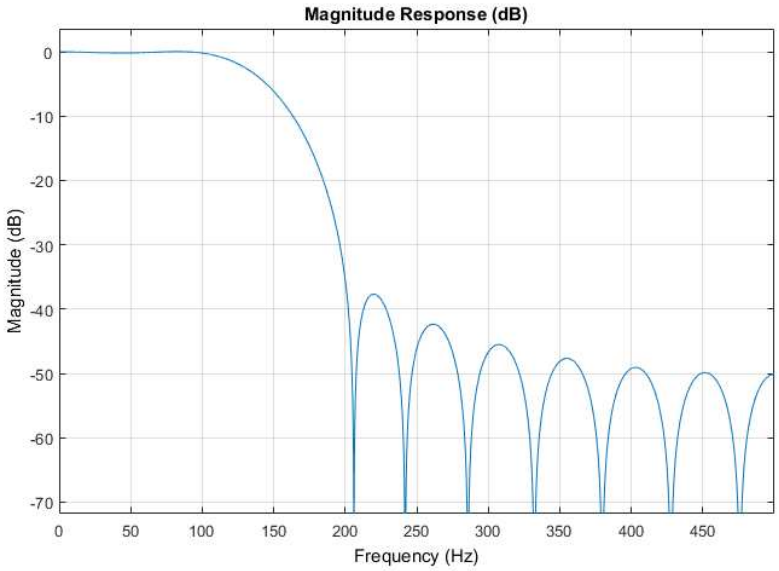
```
rng default
Fs = 1000;
t = linspace(0,1,Fs);
x = cos(2*pi*100*t)+0.5*randn(size(t));
```

Thiết kế bộ lọc FIR thông thấp dùng Phương pháp cửa sổ Kaiser với thông số sau:

Tần số cắt: 150 Hz

Bậc bộ lọc: 20

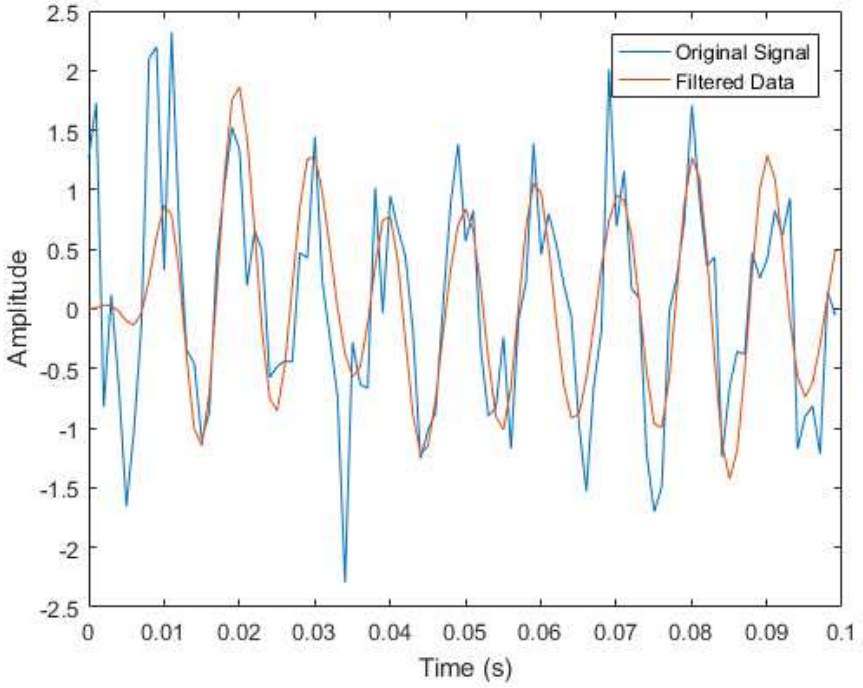
```
B =3
fc = 150;
Wn = (2/Fs)*fc;
b = fir1(20,Wn,'low',kaiser(21,3));
fvtool(b,1,'Fs',Fs)
```



Hình 5.25: Đáp ứng biên độ của bộ lọc

Ứng dụng sử dụng bộ lọc trên, cho tín hiệu x vào bộ lọc, quan sát tín hiệu ngõ ra :

```
y = filter(b,1,x);

plot(t(1:100),x(1:100))
hold on
plot(t(1:100),y(1:100))

xlabel('Time (s)')
ylabel('Amplitude')
legend('Original Signal','Filtered Data')
```



Hình 5.26: Tín hiệu gốc ngõ vào và tín hiệu ra của bộ lọc được thiết kế từ hàm fir1

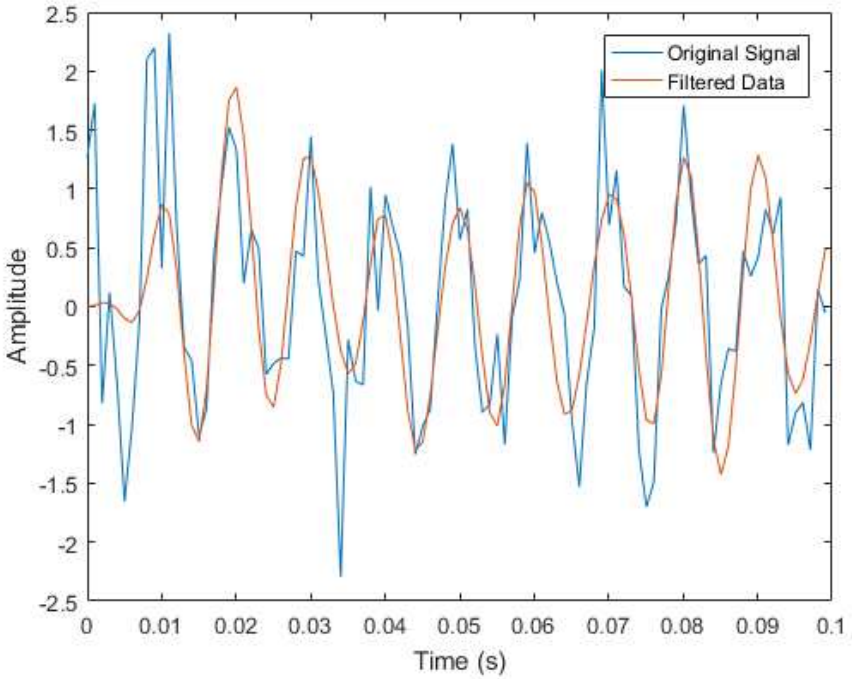
Tương tự, thực hiện với hàm designfilt như sau:

```
Fs = 1000;
Hd                                                              =
designfilt('lowpassfir','FilterOrder',20,'CutoffFrequenc
y',150, ...

'DesignMethod','window','Window',{@kaiser,3},'SampleRate
',Fs);
y1 = filter(Hd,x);
```

Khảo sát tín hiệu tại ngõ ra của bộ lọc:

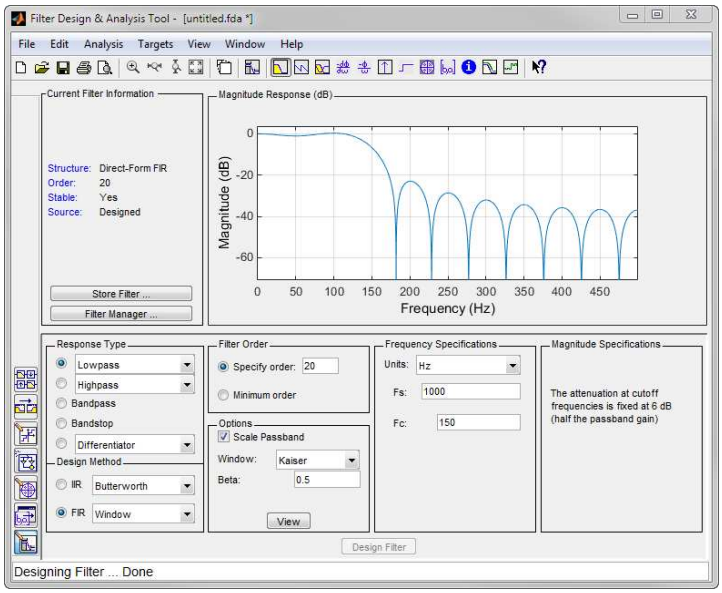
```
figure
plot(t(1:100),x(1:100))
hold on
plot(t(1:100),y1(1:100))
```

```
xlabel('Time (s)')
ylabel('Amplitude')
legend('Original Signal','Filtered Data')
```



Hình 5.27: Tín hiệu gốc ngõ vào và tín hiệu ra của bộ lọc FIR được thiết kế từ hàm designfilt

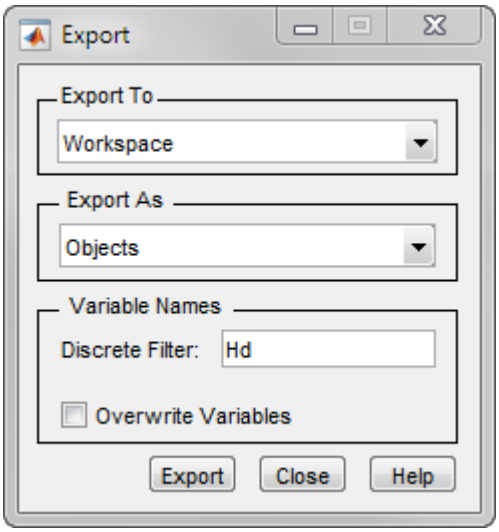
Tương tự, thiết kế bộ lọc thông thấp FIR với FDA tool



Hình 5.28: Nhập các thông số trên giao diện thiết kế bộ lọc FDATool

Chọn Design Filter.

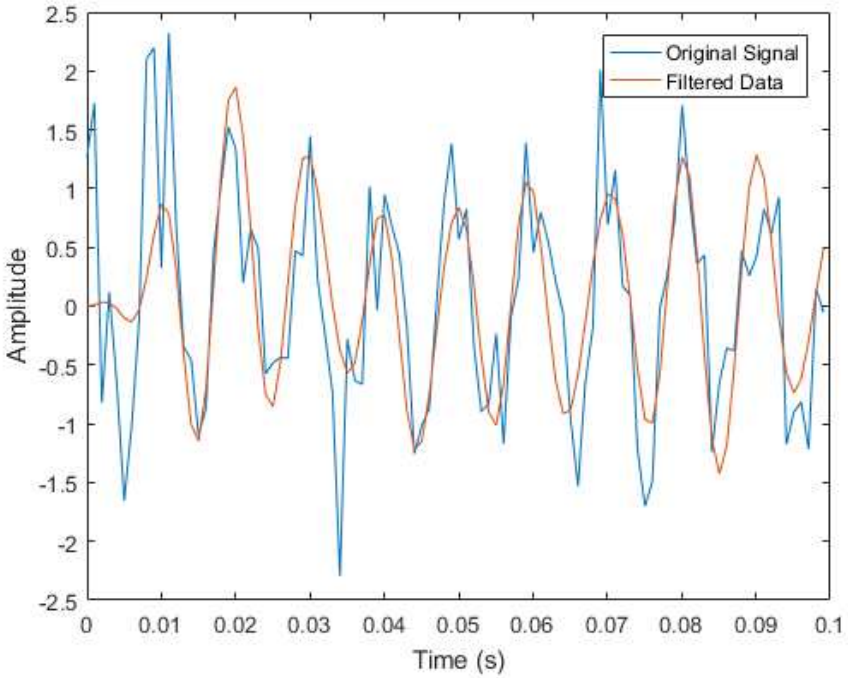
File > Export...



Chọn Export.

Cho tín hiệu x vào bộ lọc, quan sát ngõ ra như sau:

```
y2 = filter(Hd,x);

figure

plot(t(1:100),x(1:100))

hold on

plot(t(1:100),y2(1:100))

xlabel('Time (s)')

ylabel('Amplitude')

legend('Original Signal','Filtered Data')
```



Hình 5.29: Tín hiệu gốc ngõ vào và tín hiệu ra của bộ lọc
được thiết kế từ FDATool

Bài 5.24 Bậc của bộ lọc.

Thiết kế hai bộ FIR -Equiripple có thông số kỹ thuật như sau:

```
Hd1 = designfilt('bandpassfir', ...
'StopbandFrequency1',1/60,'PassbandFrequency1',1/40, ...
'PassbandFrequency2',1/4 ,'StopbandFrequency2',1/2 , ...
'StopbandAttenuation1',10,'PassbandRipple',1, ...
'StopbandAttenuation2',10,'DesignMethod','equiripple','S
ampleRate',Fs);
```

Xuất giá trị bậc của haibộ lọc:

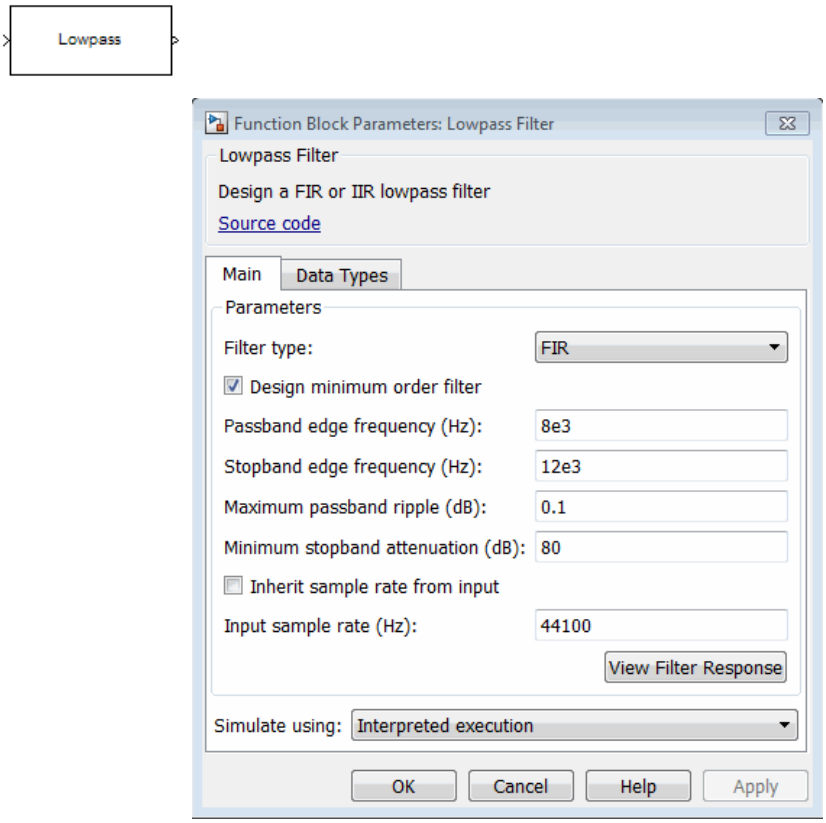
```
fprintf('Bac cua bo loc FIR la %d\n',filtord(Hd1))
```

Kết quả:

```
Bac cua bo loc FIR la 78
```

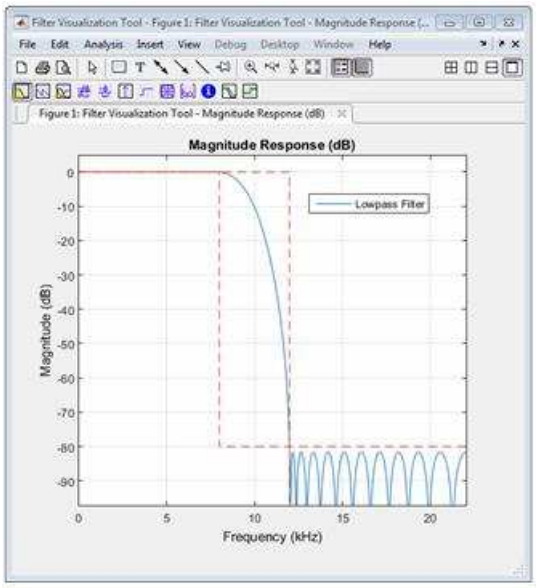
5.7 THIẾT KẾ BỘ LỌC FIR TỪ SIMULINK

Chọn khối Filter/Lowpass trong Simulink



Hình 5.30: Giao diện thiết kế bộ lọc thông thấp dùng Simulink

Mở Filter Visualization FVTool và xem phổ biên độ và pha của bộ lọc.

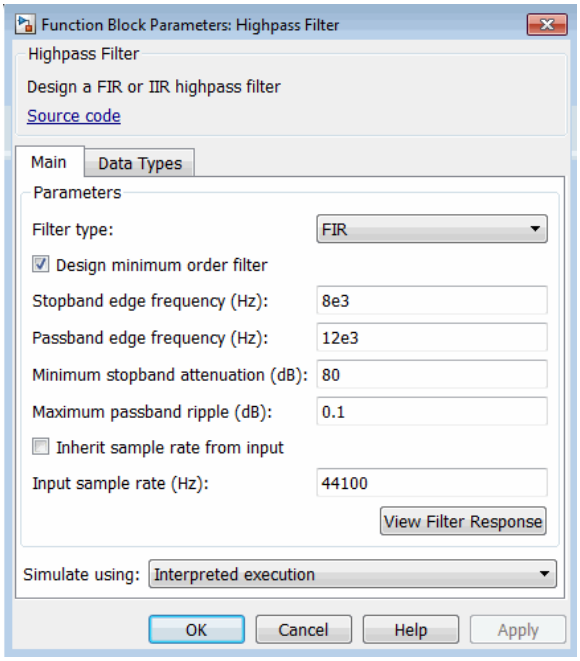


Hình 5.31 Đáp ứng của bộ lọc thông thấp

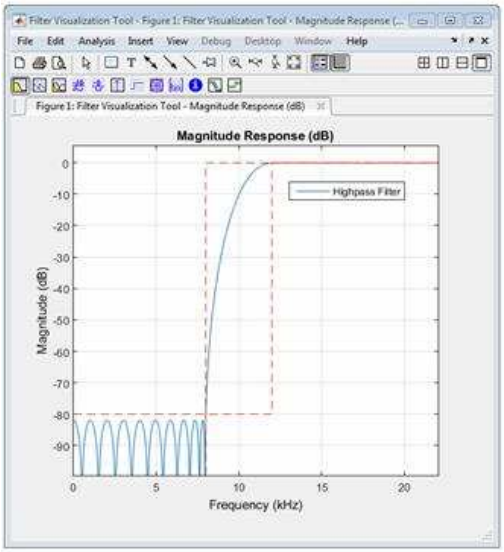
Tương tự, thiết kế bộ lọc thông cao FIR và IIR dùng simulink như sau:

Chọn khối

Nhập các thông số kỹ thuật của bộ lọc:



Hình 5.32: Nhập các thông số kỹ thuật của bộ lọc thông cao trong thiết kế dùng Simulink:



Hình 5.33 Đáp ứng của bộ lọc thông cao

5.8 BÀI TẬP

Bài tập 1: Thiết kế bộ lọc Nyquist.

Bài tập 2: Thiết kế, thực hiện và khảo sát bộ lọc FIR chắn dải bằng phương pháp cửa sổ Kaiser với các thông số sau:

- Chiều dài của đáp ứng xung: N = 63 (MATLAB hiển thị bậc bộ lọc bằng 62)
- Tần số trung tâm: 2700 Hz
- Tần số cắt: 2500 Hz và 2900 Hz
- Giá trị của $\beta = 4$
- Tần số lấy mẫu 8000 Hz

CHƯƠNG 6: BỘ LỌC SỐ IIR

Nội dung chính:

- *Thiết kế một bộ lọc số IIR dựa theo các thông số cho trước.*

*Các hàm Matlab liên quan:

Signal Processing Toolbox

butter	buttord	ellip	ellipord	filter
firpm	freqz	tf2zp	zp2sos	

6.1 CÁC HÀM THIẾT KẾ BỘ LỌC IIR:

Các yêu cầu thiết kế cho bộ lọc tương tự dựa trên điều kiện giới hạn các chỉ tiêu kỹ thuật cho bình phương biên độ của hàm đáp ứng tần số. Hàm truyền Ha(s) của một hệ thống tuyến tính bất biến tương tự là biến đổi Laplace hàm đáp ứng xung ha(t) của hệ thống và Ha(s) chính bằng tỷ số giữa biến đổi Laplace của tín hiệu đầu ra với biến đổi Laplace của tín hiệu đầu vào. Hàm đáp ứng tần số Ha(ω) của một hệ thống tuyến tính bất biến là hàm thể hiện đáp ứng của hệ thống với đầu vào là các giá trị tần số khác nhau. Do đó, hàm đáp ứng tần số là biến đổi Fourier hàm đáp ứng xung ha(t) của hệ thống và hàm Ha(ω) cũng chính là hàm hệ thống Ha(s) đánh giá trên trục ảo.

Đối với các hệ thống thực hiện được về mặt vật lý, đáp ứng xung bao giờ cũng là một hàm thực. Do đó, hàm truyền luôn đối xứng qua trục thực. Mặt khác, một hệ thống tương tự là nhân quả và ổn định nếu và chỉ nếu tất cả các điểm cực của hàm truyền đạt nằm ở nửa bên trái của mặt phẳng s hoặc nằm ở gốc toạ độ. Khi ta xét đến bình phương biên độ của hàm hệ truyền đạt, các điểm cực của hàm số này sẽ phân bố trên tất cả các góc phần tư của mặt phẳng S. Lúc này, việc xét đáp ứng tần số của hệ thống cũng thuận tiện và tất cả các điểm cực nằm bên trái mặt phẳng S của hàm |Ha(s)|2.

Yêu cầu về chỉ tiêu kỹ thuật của bộ lọc thông thấp tương tự thường được cho dưới dạng như sau:

$$\frac{1}{1+\epsilon^2} \leq |H_a(\omega)| \leq 1, |\omega| \leq \omega_p$$
$$0 \leq |H_a(\omega)| \leq \frac{1}{A^2}, |\omega| \geq \omega_s$$

với ω_p và ω_s lần lượt là các tần số cắt dải thông và tần số cắt dải chắn, ϵ là tham số gợn sóng và A là tham số suy giảm.

Quan hệ giữa ϵ và A với R_p và A_s hay δ_p và δ_s:

$$R_p = -10log\frac{1}{1+\epsilon^2} \rightarrow \epsilon = \sqrt{10^{\frac{R_p}{10}} - 1}$$

$$A_s = -10log\frac{1}{A^2} \rightarrow A = 10^{\frac{A_s}{20}}$$

$$\frac{1-\delta_p}{1+\delta_p} = \sqrt{\frac{1}{1+\epsilon^2}} \rightarrow \epsilon = \frac{2\sqrt{\delta_p}}{1-\delta_p}$$

$$\frac{\delta_s}{1+\delta_p} = \frac{1}{A} \rightarrow A = \frac{1+\delta_p}{\delta_s}$$

Có 4 định dạng cơ bản thường được vận dụng trong quá trình thiết kế bộ lọc tương tự là: **bộ lọc Butterworth, bộ lọc Chebyshev-1, bộ lọc Chebyshev-2 và bộ lọc Elliptic**.

Các **hàm MATLAB** có thể sử dụng cho bài thực hành này là:

freqs: trả về đáp ứng tần số của một hệ thống tương tự khi biết hàm truyền dưới dạng phân thức hữu tỷ.

impulse: trả về đáp ứng xung của một hệ thống tương tự khi biết hàm truyền dưới dạng phân thức hữu tỷ

buttap, cheb1ap, cheb2ap, ellipap: trả về các điểm không, điểm cực, và độ lợi trong thiết kế của một hàm truyền bộ lọc thông thấp bậc N, tần số cắt đã được chuẩn hoá bằng 1 với các định dạng lần lượt là **Butterworth, Chebyshev-I, Chebyshev-II, và Elliptic.**

impinvar, bilinear: trả về các hệ số của đa thức tử số và đa thức mẫu số hàm truyền đạt của hệ thống số xuất phát từ hệ thống tương tự qua các phương pháp chuyển đổi bất biến xung và song tuyến tính.

butter, cheby1, cheby2, ellip: trả về các hệ số của đa thức tử số và đa thức mẫu số hàm truyền của bộ lọc số dựa trên tham số đầu vào là các tần số cắt, phương pháp chuyển đổi được sử dụng trong các hàm này là phương pháp biến đổi song tuyến tính.

Hàm freqs của MATLAB trả về đáp ứng tần số của một hệ thống tương tự khi biết trước hệ số của đa thức tử số và đa thức mẫu số của hàm truyền đạt Ha(s). Trong nhiều trường hợp, để thuận tiện ta cần tìm thêm các thông số: hàm độ lớn của đáp ứng tần số, hàm pha của đáp ứng tần số, hàm trễ nhóm, thể hiện độ lớn theo thang decibels.

Filter Type	Design Function

Bessel (analog only)	`[b,a] = besself(n,Wn,options)`
	`[z,p,k] = besself(n,Wn,options)`
	`[A,B,C,D] = besself(n,Wn,options)`

Butterworth	`[b,a] = butter(n,Wn,options)`
	`[z,p,k] = butter(n,Wn,options)`
	`[A,B,C,D] = butter(n,Wn,options)`

Chebyshev Type I	`[b,a] = cheby1(n,Rp,Wn,options)`
	`[z,p,k] = cheby1(n,Rp,Wn,options)`
	`[A,B,C,D] = cheby1(n,Rp,Wn,options)`

Chebyshev Type II	`[b,a] = cheby2(n,Rs,Wn,options)`
	`[z,p,k] = cheby2(n,Rs,Wn,options)`
	`[A,B,C,D] = cheby2(n,Rs,Wn,options)`

Elliptic	`[b,a] = ellip(n,Rp,Rs,Wn,options)`
	`[z,p,k] = ellip(n,Rp,Rs,Wn,options)`
	`[A,B,C,D] = ellip(n,Rp,Rs,Wn,options)`

Bảng 6.1 Các hàm thiết kế bộ lọc số IIR

Một vài ví dụ:

```
[b,a] = butter(5,0.4);    %Lowpass Butterworth
```

```
[b,a] = cheby1(4,1,[0.4 0.7]);%Bandpass Chebyshev Type I
[b,a] = cheby2(6,60,0.8,'high');%Highpass Chebyshev Type II
[b,a] = ellip(3,1,60,[0.4 0.7],'stop');%Bandstop elliptic
[b,a] = butter(5,0.4,'s');  % Analog Butterworth filter
```

Filter Type	Order Estimation Function
Butterworth	[n,Wn] = buttord(Wp,Ws,Rp,Rs)
Chebyshev Type I	[n,Wn] = cheb1ord(Wp,Ws,Rp,Rs)
Chebyshev Type II	[n,Wn] = cheb2ord(Wp,Ws,Rp,Rs)
Elliptic	[n,Wn] = ellipord(Wp,Ws,Rp,Rs)

Bảng 6.2 Các hàm tính toán bậc bộ lọc IIR

6.2 CÁC BÀI THIẾT KẾ BỘ LỌC

Bài 6.1 Giả sử thiết kế một bộ lọc dải thông 1000-2000 Hz, stopbands 500 Hz , tần số lấy mẫu 10 kHz, suy giảm 1 dB ở dải thông, và suy giảm ít nhất là 60 dB ở cả hai phía.

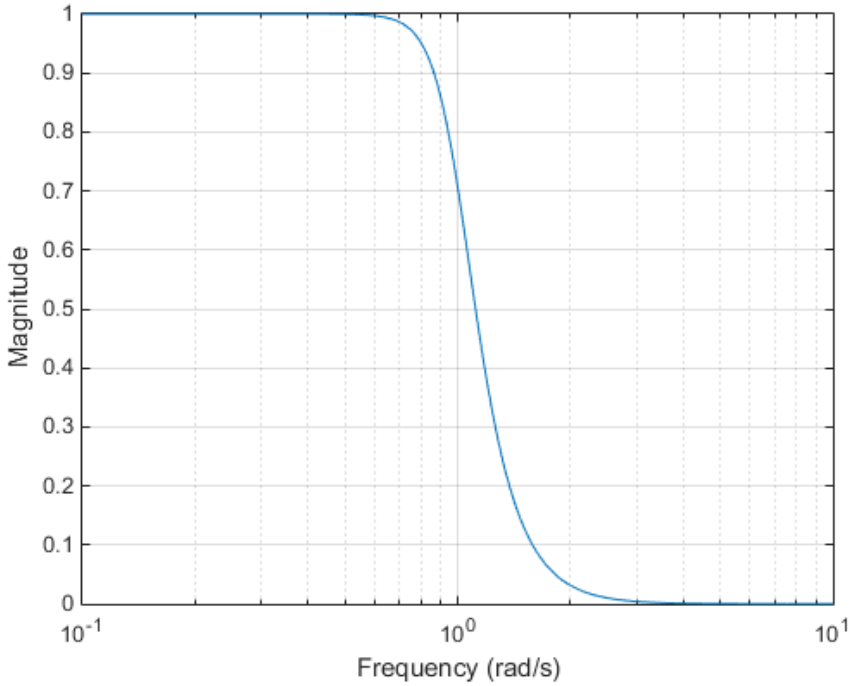
```
[n,Wn] = buttord([1000 2000]/5000,[500 2500]/5000,1,60)
[b,a] = butter(n,Wn);

n =
     12
Wn =
    0.1951    0.4080
```

Một bộ lọc elliptic đáp ứng các yêu cầu tương tự được thực hiện như sau:

```
[n,Wn] = ellipord([1000 2000]/5000,[500 2500]/5000,1,60)
[b,a] = ellip(n,1,60,Wn);
    n =
         5
    Wn =
```

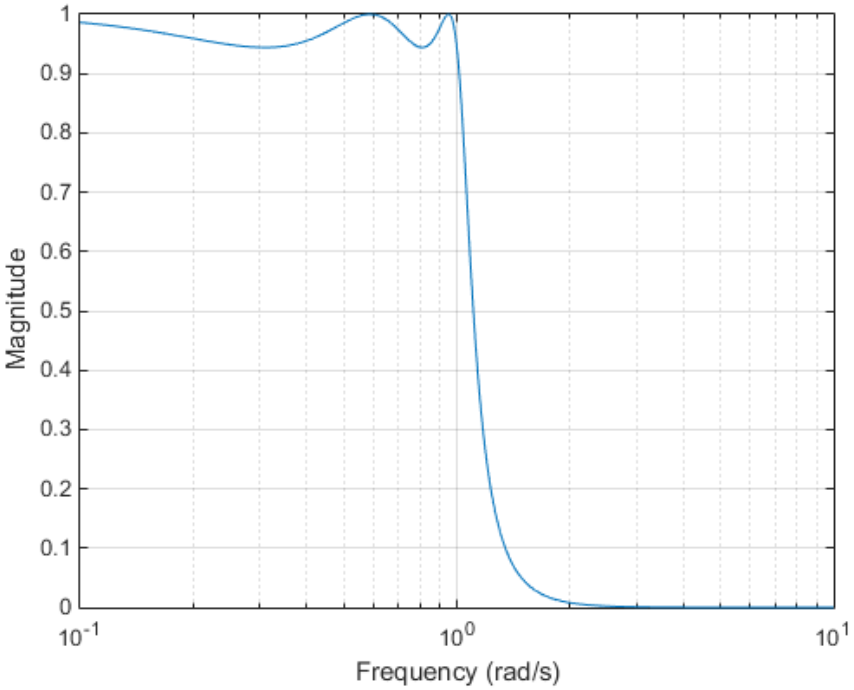
```
    0.2000    0.4000
```

Bộ lọc Butterworth



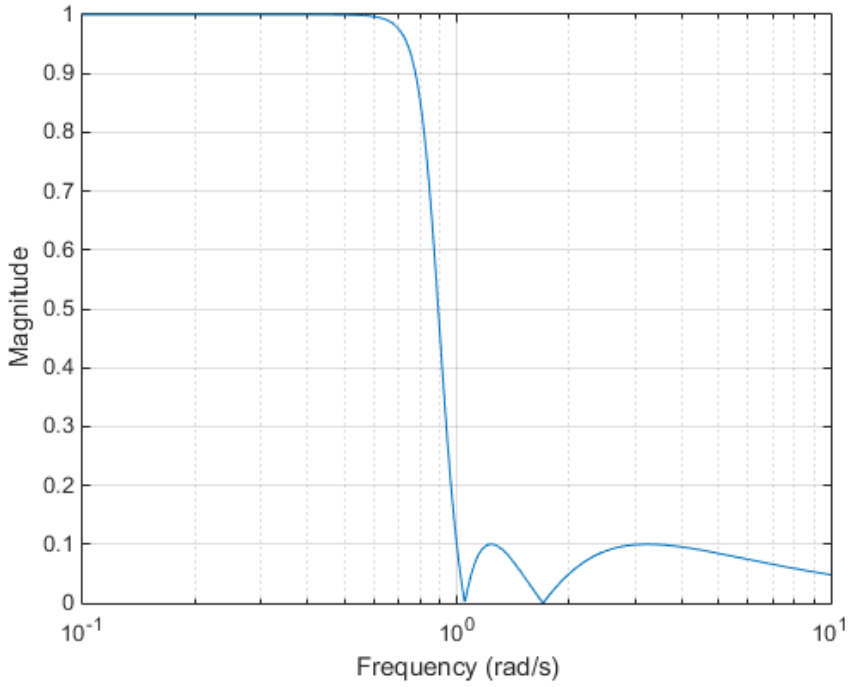
Hình 6.1 Đáp ứng biên độ Bộ lọc Butterworth

Bộ lọc Chebyshev Loại 1



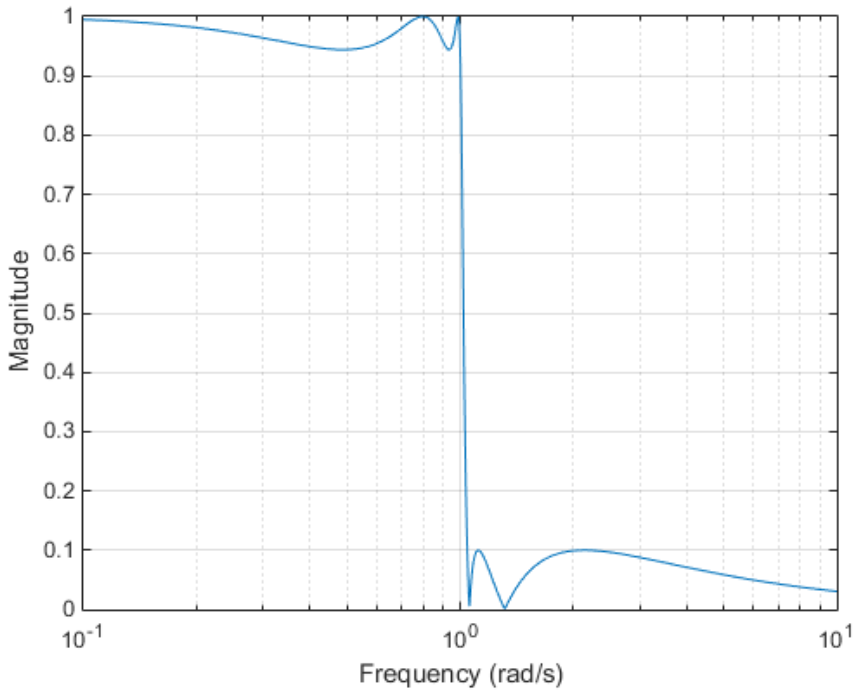
Hình 6.2: Đáp ứng biên độ Bộ lọc Chebyshev Loại 1

Bộ lọc Chebyshev Loại 2



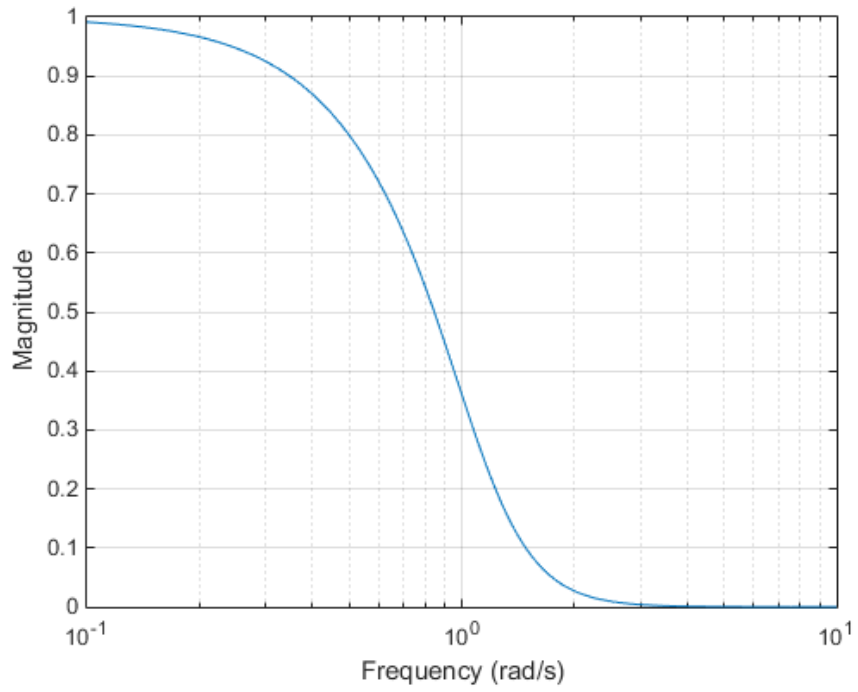
Hình 6.3: Đáp ứng biên độ Bộ lọc Chebyshev Loại 2

Bộ lọc Elliptic



Hình 6.4: Đáp ứng biên độ Bộ lọc Elliptic

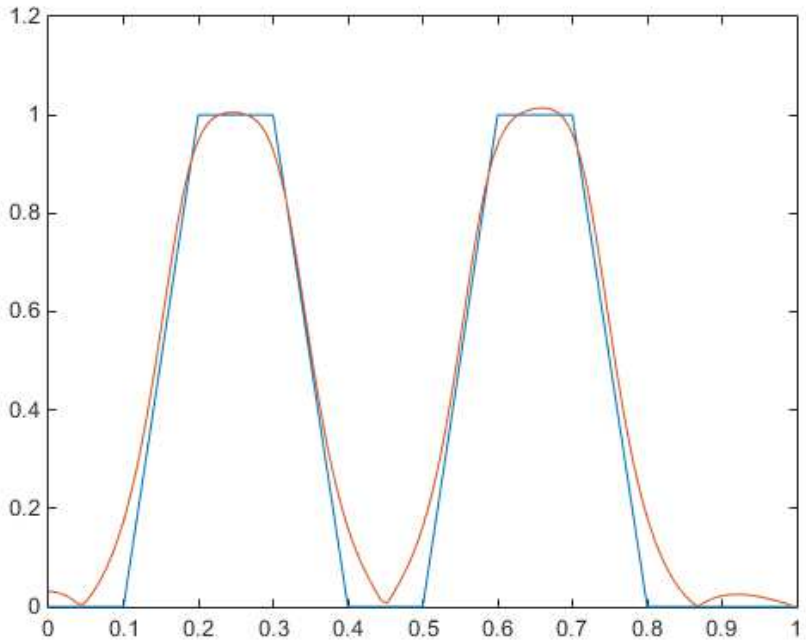
Bộ lọc Bessel



Hình 6.5: Đáp ứng biên độ Bộ lọc Bessel

Bài 6.2 Bộ lọc IIR trực tiếp

```
m = [0   0   1   1   0   0   1   1   0 0];
f = [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 1];
[b,a] = yulewalk(10,f,m);
[h,w] = freqz(b,a,128)
plot(f,m,w/pi,abs(h))
```



Hình 6.6 Đáp ứng biên độ Bộ lọc IIR trực tiếp

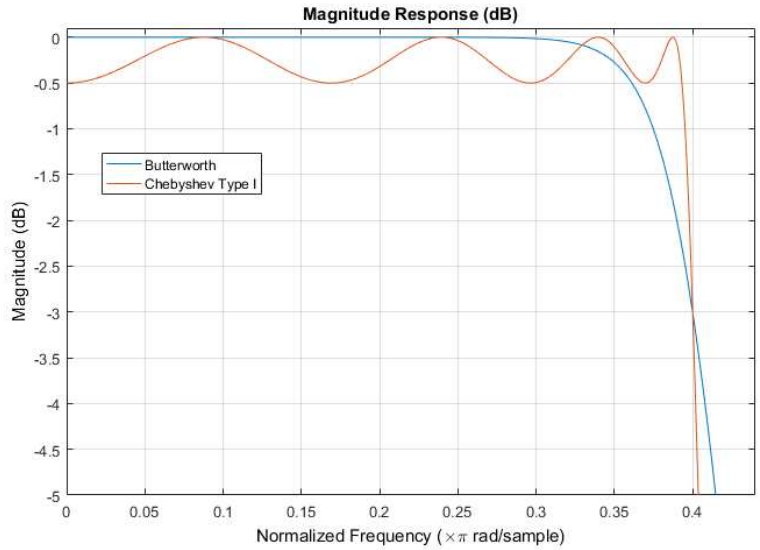
Bài 6.3 Thiết kế bộ lọc thông thấp Butterworth với các thông số sau:

```
N = 8;
F3dB = .4;
d = fdesign.lowpass('N,F3dB',N,F3dB);
Hbutter = design(d,'butter','SystemObject',true);
```

Bộ lọc Chebyshev loại 1

```
Ap = .5;
setspecs(d,'N,F3dB,Ap',N,F3dB,Ap);
Hcheby1 = design(d,'cheby1','SystemObject',true);
hfvt = fvtool(Hbutter,Hcheby1,'Color','white');
axis([0 .44 -5 .1])
```

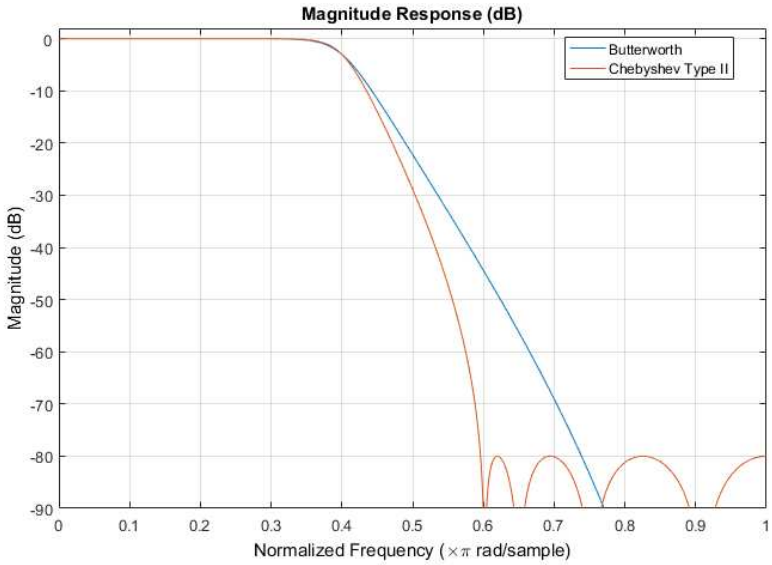
```
legend(hfvt,'Butterworth','Chebyshev Type I');
```



Hình 6.7 Đáp ứng biên độ bộ lọc thông thấp Butterworth và Chebyshev loại 1

Bộ lọc Chebyshev loại 2

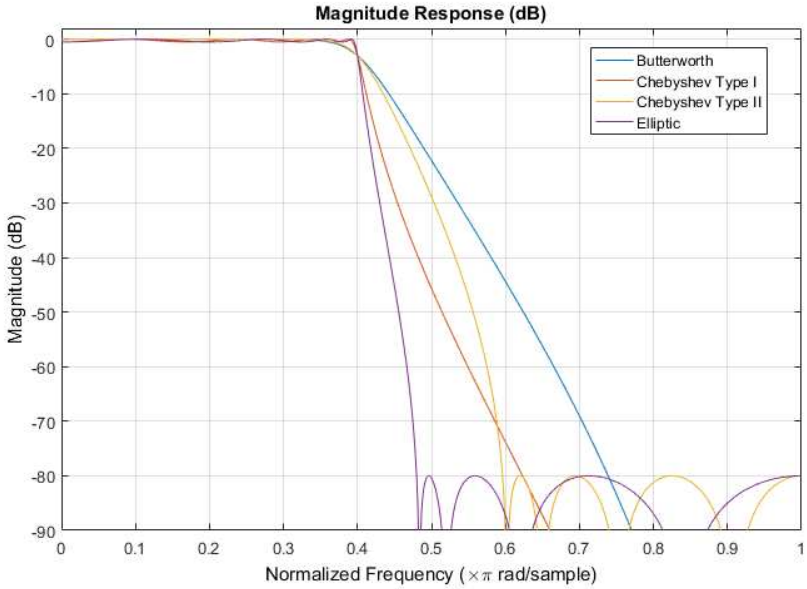
```
Ast = 80;
setspecs(d,'N,F3dB,Ast',N,F3dB,Ast);
Hcheby2 = design(d,'cheby2','SystemObject',true);
hfvt = fvtool(Hbutter,Hcheby2,'Color','white');
axis([0 1 -90 2])
legend(hfvt,'Butterworth','Chebyshev Type II');
```



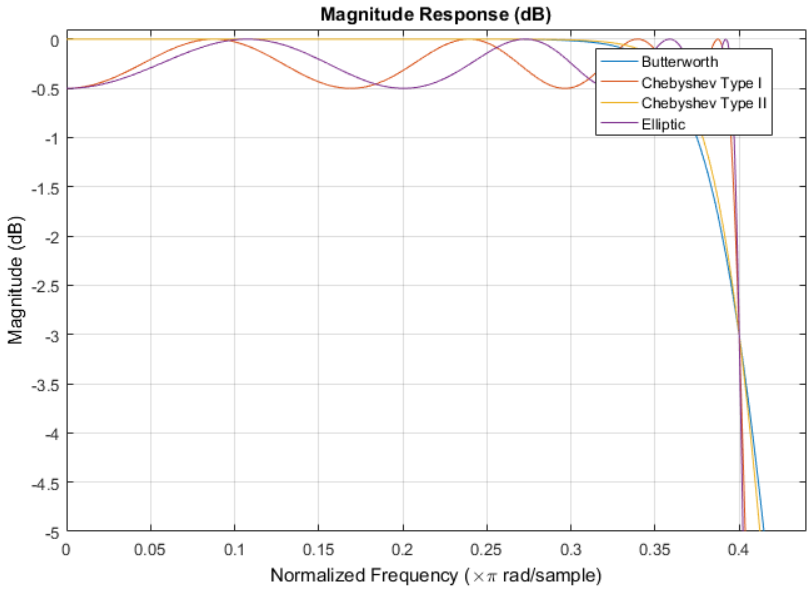
Hình 6.8 Đáp ứng biên độ bộ lọc thông thấp Butterworth và Chebyshev loại 2

Bộ lọc Elliptic

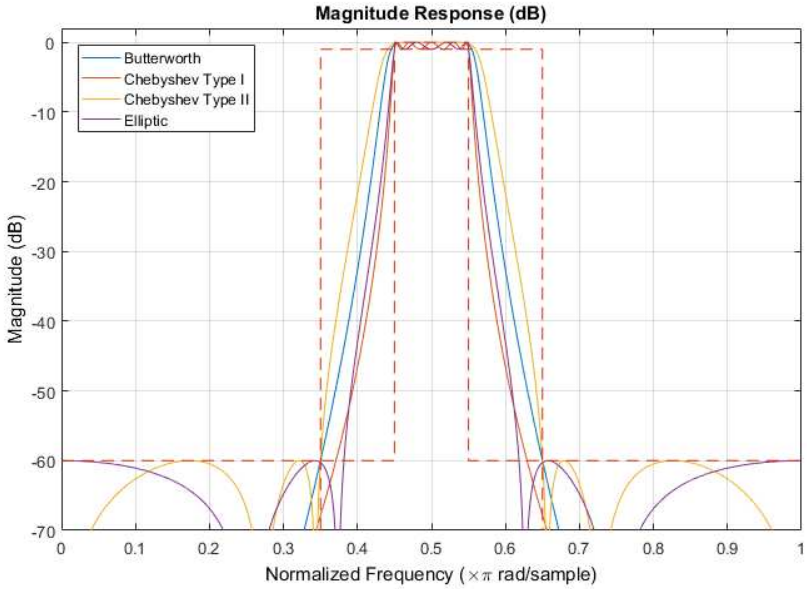
```
setspecs(d,'N,F3dB,Ap,Ast',N,F3dB,Ap,Ast);
Hellip = design(d,'ellip','SystemObject',true);
hfvt = fvtool(Hbutter,Hcheby1,Hcheby2,Hellip,'Color','white');
axis([0 1 -90 2])
legend(hfvt, ...
    'Butterworth','Chebyshev Type I','Chebyshev Type
II','Elliptic');
axis([0 .44 -5 .1])
```



Hình 6.9 Đáp ứng biên độ bộ lọc thông thấp Butterworth,Chebyshev loại 1, Chebyshev loại 2 và Elliptic



Hình 6.10 Xem rõ độ gợn sóng của đáp ứng biên độ bộ lọc thông thấp Butterworth,Chebyshev loại 1, Chebyshev loại 2 và Elliptic

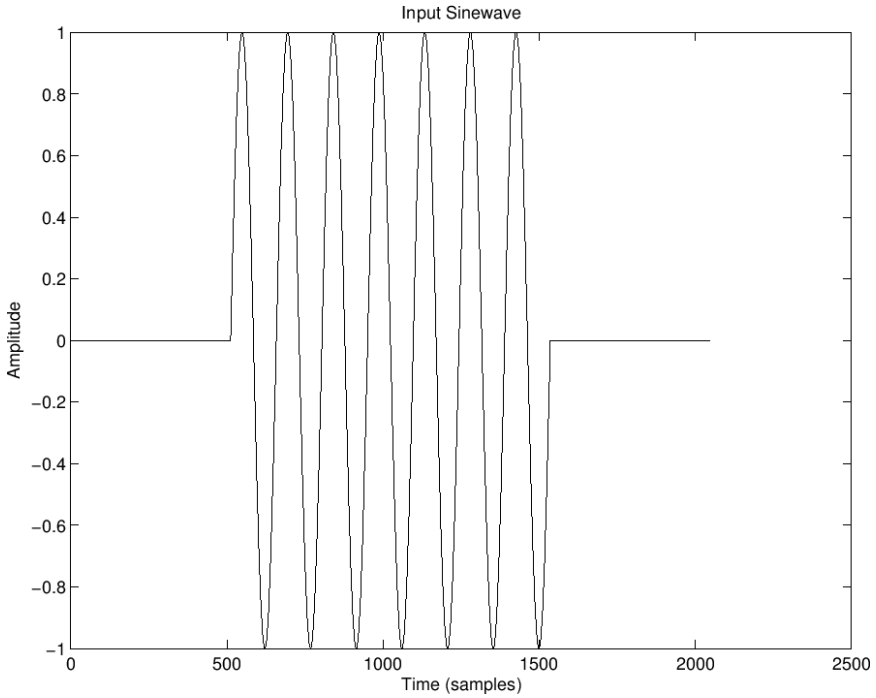


Hình 6.11 Đáp ứng biên độ bộ lọc thông chắn dãi Butterworth,Chebyshev loại 1, Chebyshev loại 2 và Elliptic

Bài 6.4 Vẽ bộ lọc IIR có phương trình sai phân:

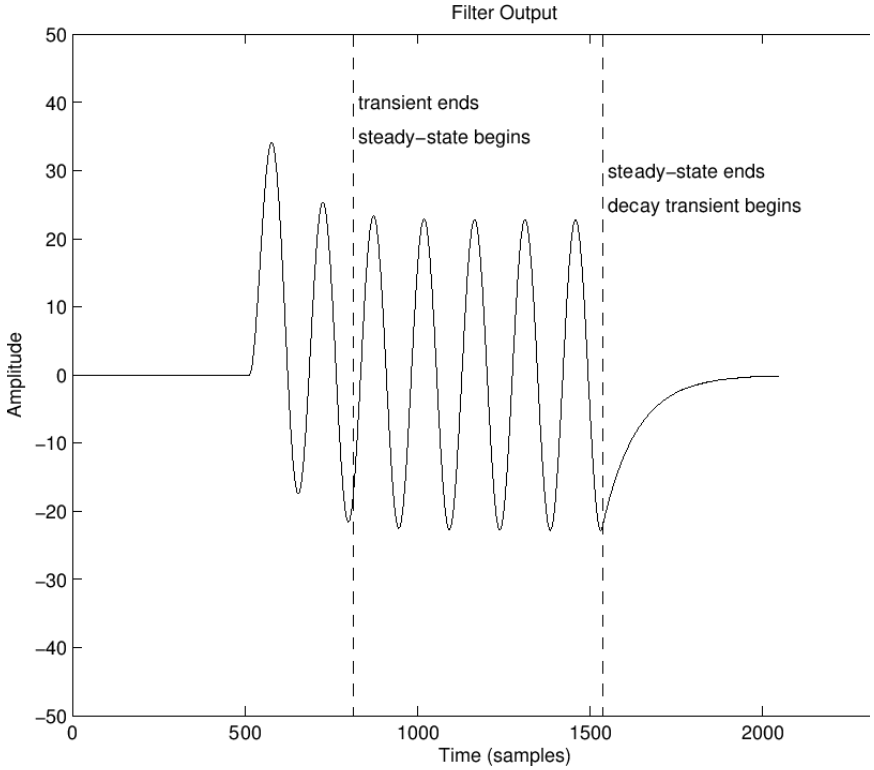
$$y(n) = x(n) + 0.9y(n-1)$$

Dạng sóng ngõ vào:



Hình 6.12: Tín hiệu ngõ vào bộ lọc

Dạng sóng ngõ ra của bộ lọc:



Hình 6.13: Tín hiệu ngõ ra của bộ lọc

Bài 6.5: Thiết kế bộ lọc Butterworth bậc tối thiểu với dãi thông 100 Hz, tần số stopband 300 Hz, độ gợn cực đại 1 dB, và suy giảm 60 dB tại stopband, tốc độ lấy mẫu là 2 kHz.

```
Fp = 100;
Fst = 300;
Ap = 1;
Ast = 60;
Fs = 2e3;

dbutter                                                    =
designfilt('lowpassiir','PassbandFrequency',Fp,...
  'StopbandFrequency',Fst,'PassbandRipple',Ap,...

'StopbandAttenuation',Ast,'SampleRate',Fs,'DesignMethod','
butter');
```

% Sử dụng Chebyshev loại 1

```
dcheby1 =
designfilt('lowpassiir','PassbandFrequency',Fp,...
  'StopbandFrequency',Fst,'PassbandRipple',Ap,...

'StopbandAttenuation',Ast,'SampleRate',Fs,'DesignMethod','
cheby1');
```

% Sử dụng Chebyshev loại 2

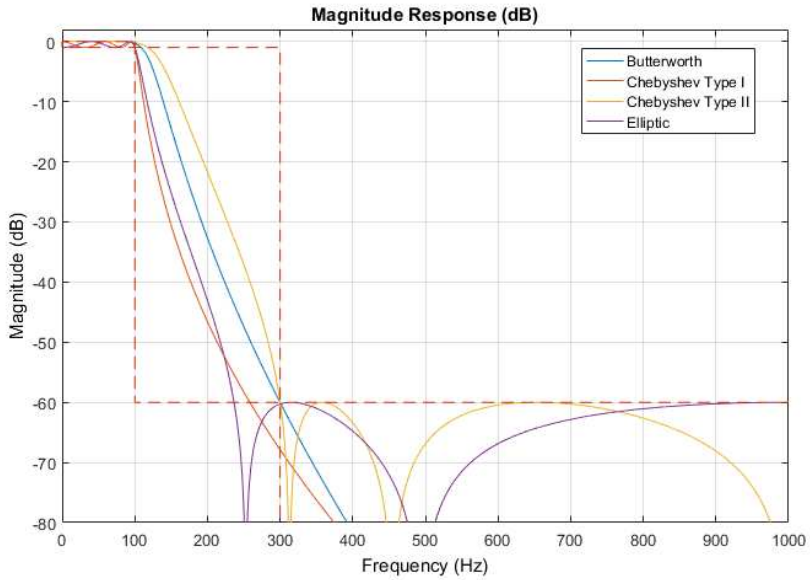
```
dcheby2 =
designfilt('lowpassiir','PassbandFrequency',Fp,...
  'StopbandFrequency',Fst,'PassbandRipple',Ap,...

'StopbandAttenuation',Ast,'SampleRate',Fs,'DesignMethod','
cheby2');
```

% Sử dụng Elliptic

```
dellip =
designfilt('lowpassiir','PassbandFrequency',Fp,...
  'StopbandFrequency',Fst,'PassbandRipple',Ap,...

'StopbandAttenuation',Ast,'SampleRate',Fs,'DesignMethod','
ellip');
```

Kết quả vẽ 4 loại bộ lọc như sau:

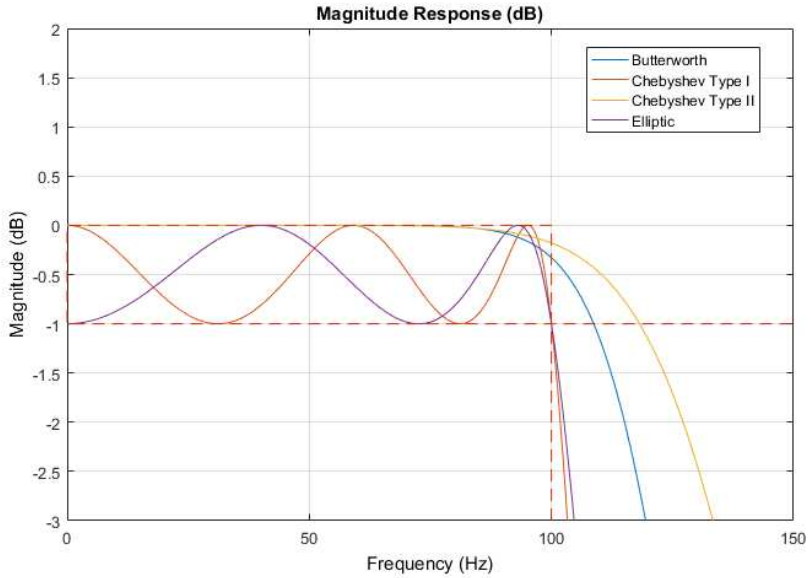
```
hfvt = fvtool(dbutter,dcheby1,dcheby2,dellip);
axis([0 1e3 -80 2]);
legend(hfvt,'Butterworth', 'Chebyshev Type I',...
  'Chebyshev Type II','Elliptic')
```



Hình 6.14 Đáp ứng biên độ bộ lọc thông chắn dãi Butterworth,Chebyshev loại 1, Chebyshev loại 2 và Elliptic

Xem dạng sóng gợn rõ hơn như sau:

```
axis([0 150 -3 2]);
```



Hình 6.15 Độ gợn sóng của Đáp ứng biên độ bộ lọc thông chắn dãi Butterworth,Chebyshev loại 1, Chebyshev loại 2 và Elliptic

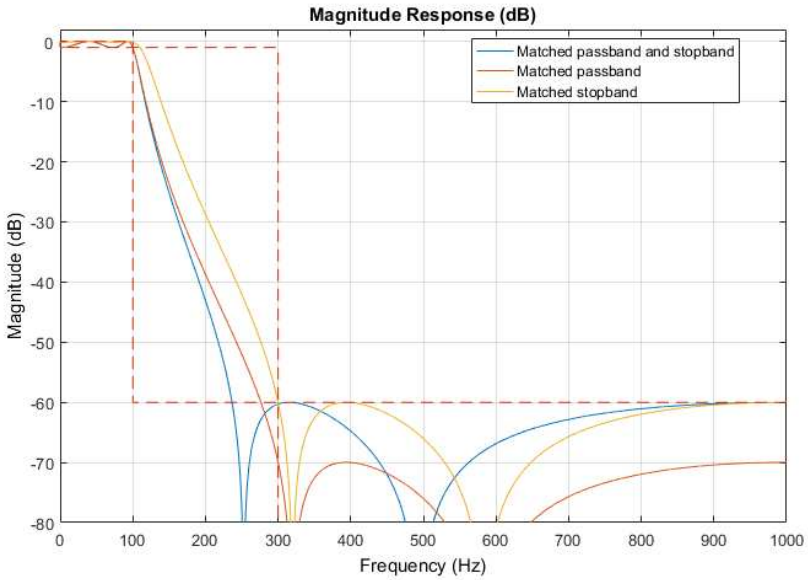
So sánh bậc của các bộ lọc

```
  FilterOrders    =    [filtord(dbutter)    filtord(dcheby1)
  filtord(dcheby2) filtord(dellip)]

  FilterOrders =
       7     5     5     4
```

Nhận xét: Với cùng đặc điểm kỹ thuật phương pháp Butterworth cho số bậc cao nhất và các phương pháp elliptic cho số bậc nhỏ nhất.

Để chính xác các thông số kỹ thuật dùng thêm thông số MatchExactly trong thiết kế

```
dellip1 = designfilt('lowpassiir','PassbandFrequency',Fp,...
  'StopbandFrequency',Fst,'PassbandRipple',Ap,...

'StopbandAttenuation',Ast,'SampleRate',Fs,'DesignMethod','el
lip',...'MatchExactly','passband');

dellip2 = designfilt('lowpassiir','PassbandFrequency',Fp,...
  'StopbandFrequency',Fst,'PassbandRipple',Ap,...

'StopbandAttenuation',Ast,'SampleRate',Fs,'DesignMethod','el
lip',...
  'MatchExactly','stopband');

hfvt = fvtool(dellip, dellip1, dellip2);
legend(hfvt,'Matched   passband   and   stopband','Matched
passband',...
  'Matched stopband');
axis([0 1e3 -80 2]);
```



Hình 6.16 Kết quả đáp ứng của bộ lọc khi dùng MatchExactly trong thiết kế

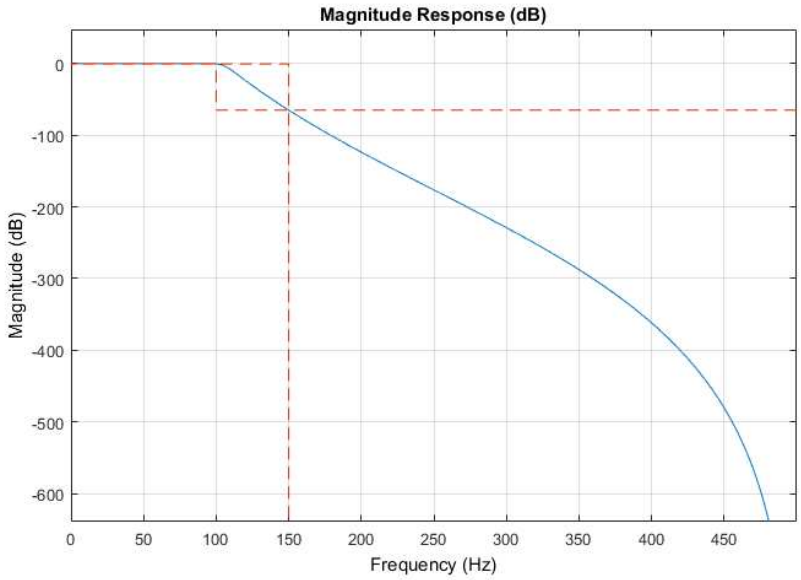
Nhận xét: kết quả thiết kế cho độ gợn sóng chính xác suy giảm 1bB tại tần số 100Hz.

Bài 6.6 Thiết kế bộ lọc thông thấp IIR với các thông số kỹ thuật sau:

Lowpass IIR Filters

Maximally Flat Design

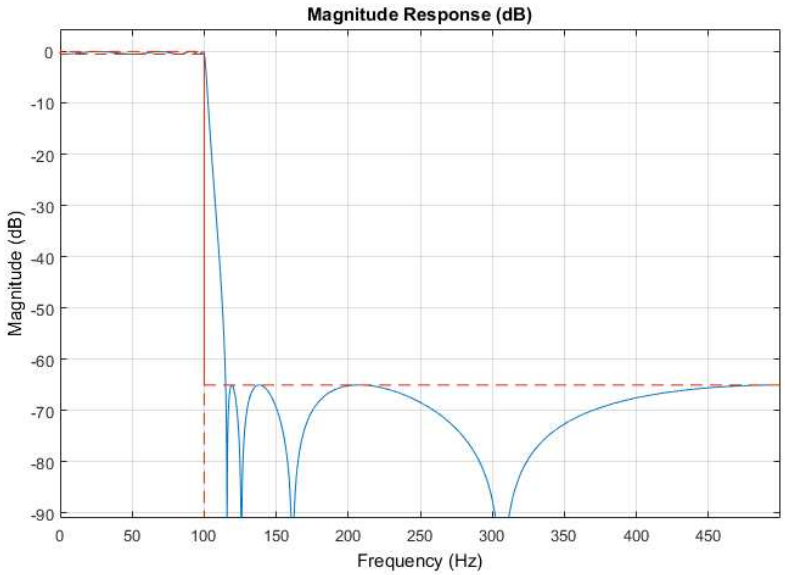
```
Fpass = 100;
Fstop = 150;
Apass = 0.5;
Astop = 65;
Fs = 1e3;

d = designfilt('lowpassiir', ...
  'PassbandFrequency',Fpass,'StopbandFrequency',Fstop, ...
  'PassbandRipple',Apass,'StopbandAttenuation',Astop, ...
  'DesignMethod','butter','SampleRate',Fs);

fvtool(d)
```



Hình 6.17 Độ gợn sóng ở dãy băng thông

của bộ lọc Lowpass IIR Filters

```
  N = 8;
  Fpass = 100;
  Apass = 0.5;
  Astop = 65;
  Fs = 1e3;

  d = designfilt('lowpassiir', ...
    'FilterOrder',N, ...
    'PassbandFrequency',Fpass, ...
    'PassbandRipple',Apass,'StopbandAttenuation',Astop, ...
    'SampleRate',Fs);

fvtool(d)
```



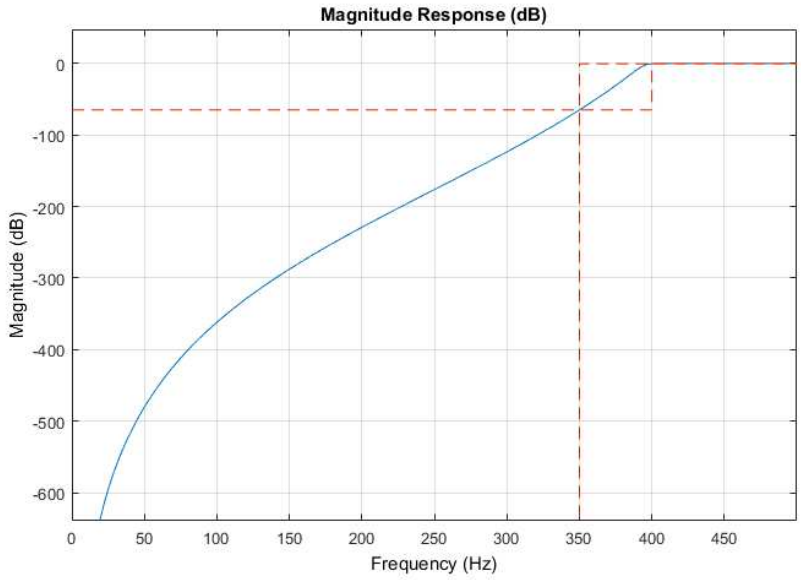
Hình 6.18 Đáp ứng của bộ lọc Lowpass IIR Filters

Bài 6.7 Thiết kế bộ lọc thông cao IIR với các thông số kỹ thuật sau:

Highpass IIR Filters

Maximally Flat Design

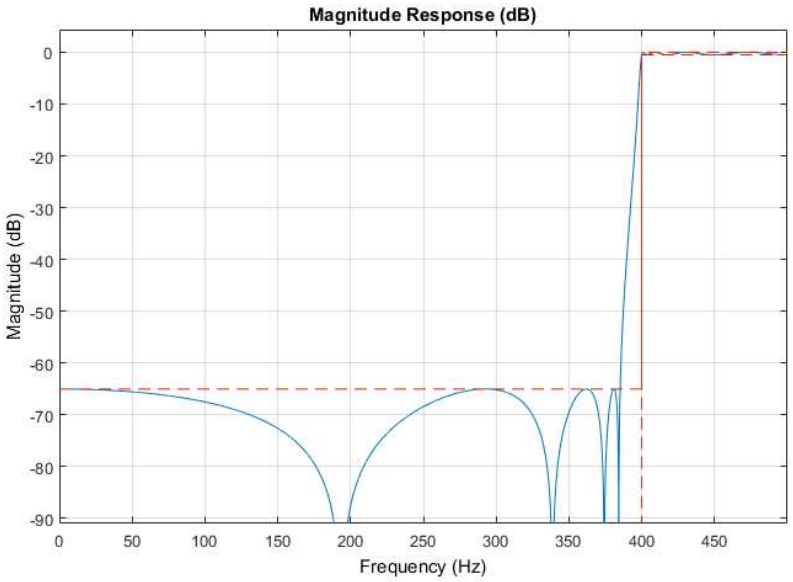
```
  Fstop = 350;
  Fpass = 400;
  Astop = 65;
  Apass = 0.5;
  Fs = 1e3;
d = designfilt('highpassiir','StopbandFrequency',Fstop ,...
  'PassbandFrequency',Fpass,'StopbandAttenuation',Astop, ...

'PassbandRipple',Apass,'SampleRate',Fs,'DesignMethod','butte
r');
fvtool(d)
```



Hình 6.19 Độ gợn sóng ở dãy băng thông

của bộ lọc Highpass IIR Filters

```
N = 8;
Fpass = 400;
Astop = 65;
Apass = 0.5;
Fs = 1e3;
d = designfilt('highpassiir', ...
  'FilterOrder',N, ...
  'PassbandFrequency',Fpass, ...
  'StopbandAttenuation',Astop,'PassbandRipple',Apass, ...
  'SampleRate',Fs);
fvtool(d)
```



Hình 6.20 Đáp ứng của bộ lọc Highpass IIR Filters

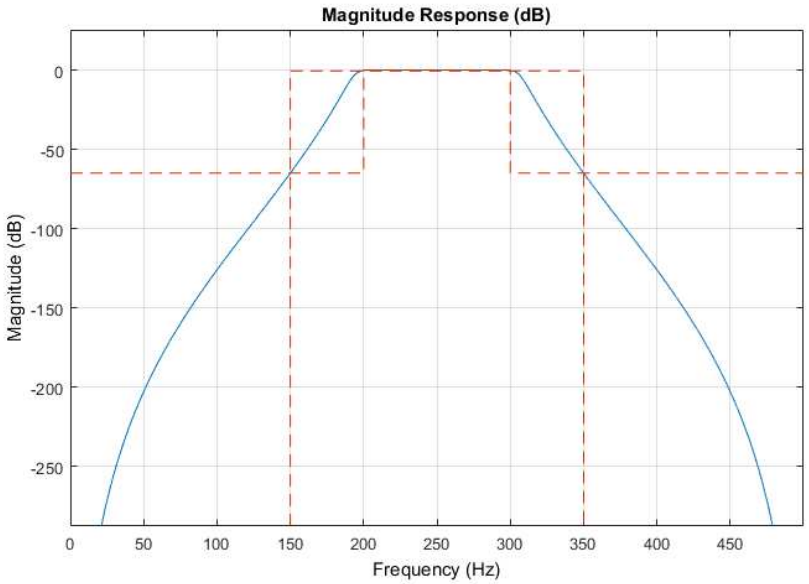
Bài 6.8 Thiết kế bộ lọc thông dãi IIR với các thông số kỹ thuật sau:

Bandpass IIR Filters

Maximally Flat Design

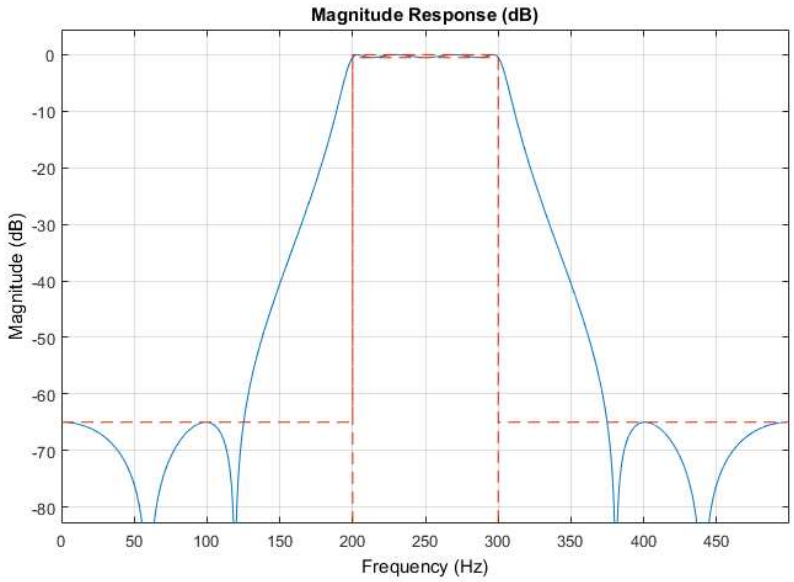
```
Fstop1 = 150;
Fpass1 = 200;
Fpass2 = 300;
Fstop2 = 350;
Astop1 = 65;
Apass  = 0.5;
Astop2 = 65;
Fs = 1e3;
d = designfilt('bandpassiir', ...
  'StopbandFrequency1',Fstop1,'PassbandFrequency1',   Fpass1,
...
  'PassbandFrequency2',Fpass2,'StopbandFrequency2', Fstop2,
  'StopbandAttenuation1',Astop1,'PassbandRipple', Apass, ...
  'StopbandAttenuation2',Astop2, ...
  'DesignMethod','butter','SampleRate', Fs);
```

```
fvtool(d)
```



Hình 6.21 Độ gợn sóng Đáp ứng của bộ lọc Bandpass IIR Filters

```
N = 8;
Fpass1 = 200;
Fpass2 = 300;
Astop1 = 65;
Apass  = 0.5;
Astop2 = 65;
Fs = 1e3;
d = designfilt('bandpassiir', ...
  'FilterOrder',N, ...
  'PassbandFrequency1', Fpass1,'PassbandFrequency2', Fpass2,
...
  'StopbandAttenuation1', Astop1, 'PassbandRipple', Apass,
  'StopbandAttenuation2', Astop2, ...
  'SampleRate', Fs);

fvtool(d)
```



Hình 6.22 Đáp ứng của bộ lọc Bandpass IIR Filters

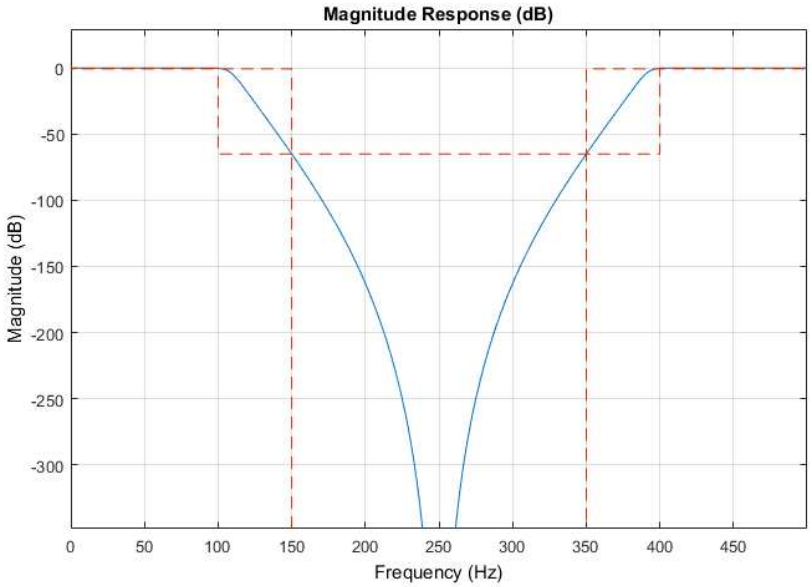
Bài 6.9 Thiết kế bộ lọc chắn dãi IIR với các thông số kỹ thuật sau:

Bandstop IIR Filters

Maximally Flat Design

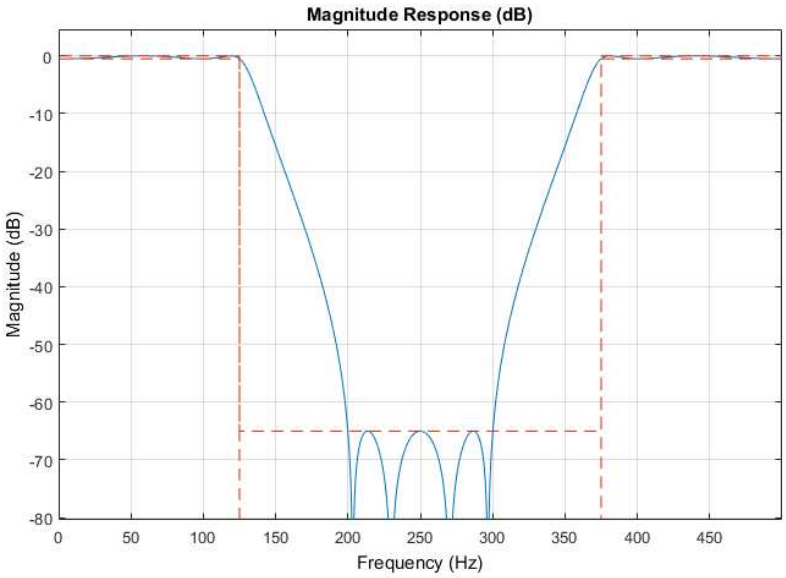
```
Fpass1 = 100;
Fstop1 = 150;
Fstop2 = 350;
Fpass2 = 400;
Apass1 = 0.5;
Astop  = 65;
Apass2 = 0.5;
Fs = 1e3;

d = designfilt('bandstopiir', ...
  'PassbandFrequency1',Fpass1,'StopbandFrequency1',Fstop1,
'StopbandFrequency2',Fstop2,'PassbandFrequency2',Fpass2, ...
  'PassbandRipple1',Apass1,'StopbandAttenuation',Astop, ...
  'PassbandRipple2', Apass2, ...
  'DesignMethod','butter','SampleRate', Fs);
```

```
fvtool(d)
```



Hình 6.23 Độ gợn sóng Đáp ứng của bộ lọc Bandstop IIR Filters

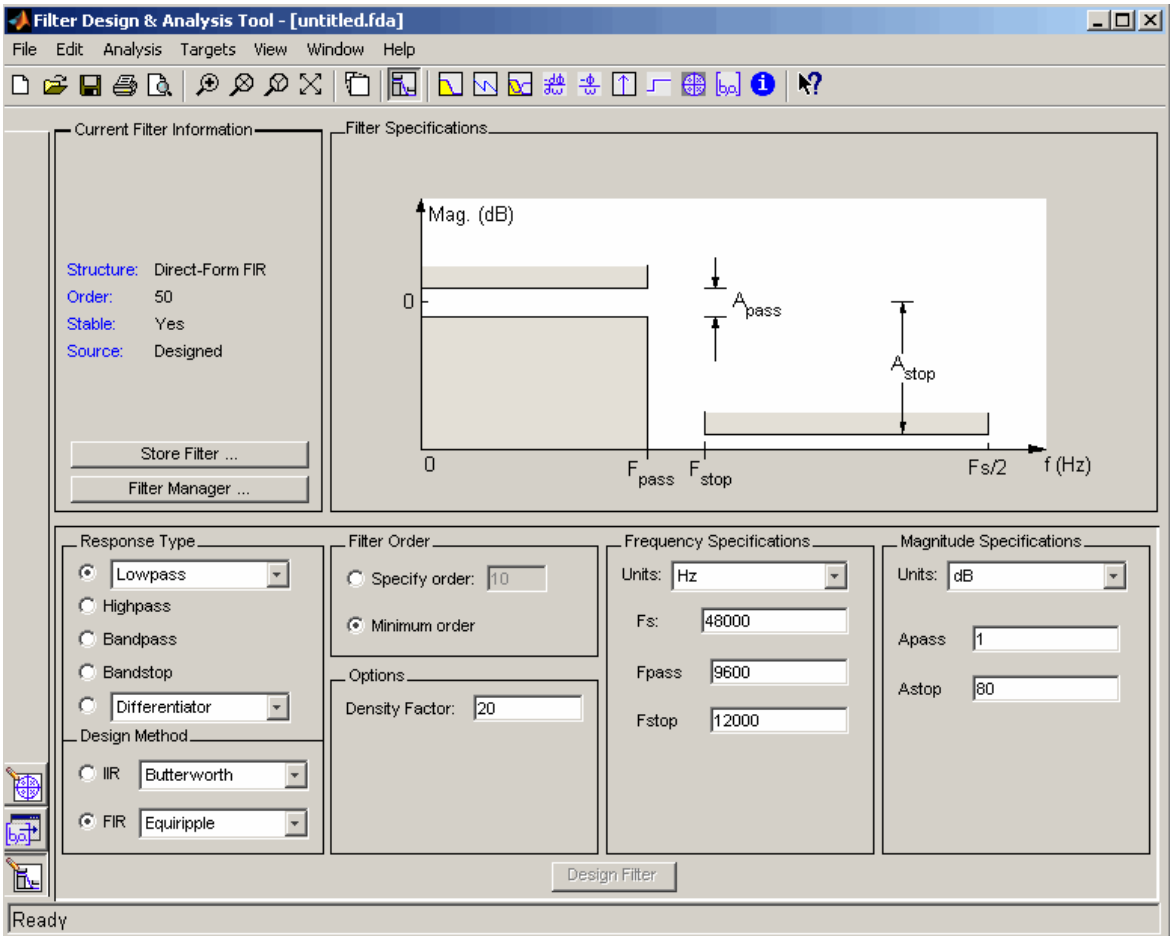
```
%Ve dap ung cua bo loc
N = 8;
Fpass1 = 125;
Fpass2 = 375;
Apass = 0.5;
Astop  = 65;
Fs = 1e3;
d = designfilt('bandstopiir', ...
  'FilterOrder',N, ...
  'PassbandFrequency1',Fpass1,'PassbandFrequency2',Fpass2,
'PassbandRipple',Apass,'StopbandAttenuation', Astop, ...
  'SampleRate',Fs);
fvtool(d)
```



Hình 6.24 Đáp ứng của bộ lọc Bandstop IIR Filters

6.3 THIẾT KẾ IIR DÙNG FDATOOL

Tương tự phần trình bày ở chương 5, sử dụng FDA Tool như sau:

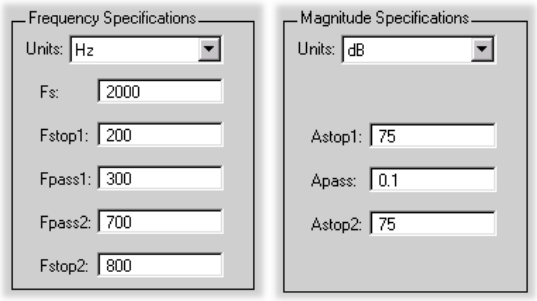


Window	Parameter
Chebyshev (`chebwin`)	Sidelobe attenuation
Gaussian (`gausswin`)	Alpha
Kaiser (`kaiser`)	Beta
Taylor (`taylorwin`)	Nbar and Sidelobe level
Tukey (`tukeywin`)	Alpha
User Defined	Function Name, Parameter

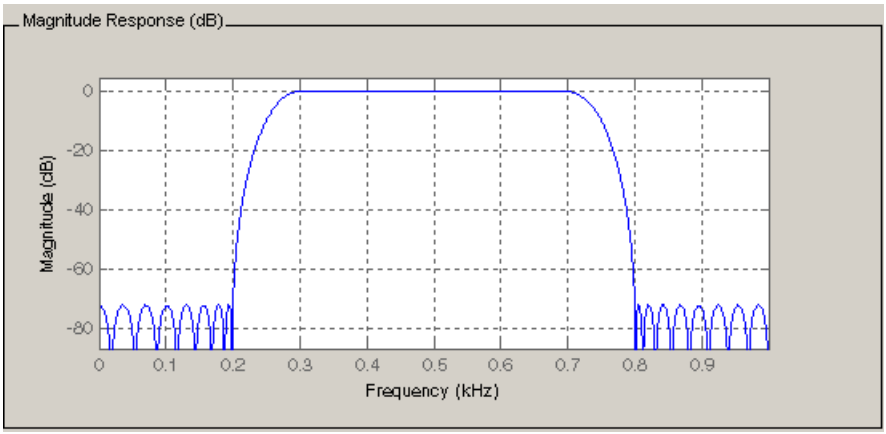
Hình 6.25 Bảng các hàm thiết kế IIR

Bài 6.10 Thiết kế bộ lọc IIR băng dãi có thông số kỹ thuật như sau:

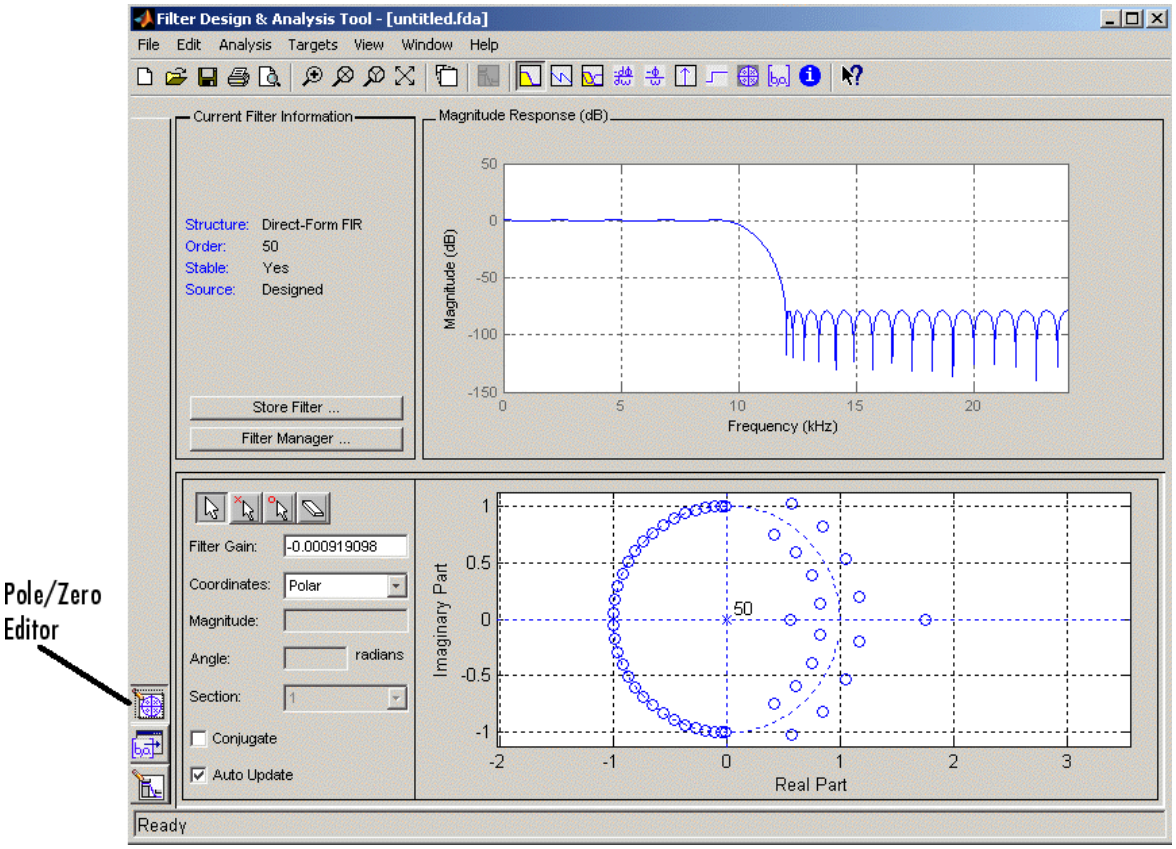
Giao diện nhập các thông số kỹ thuật



Kết quả đáp ưng bộ lọc băng dãi IIR, theo tần số chuẩn hoá



Kết quả đáp ứng và giản đồ cực-zero



6.4 Thiết kế bộ lọc IIR bằng SPTool

Tương tự hướng dẫn cách sử dụng SPTool ở chương 5, có thể thiết kế các bộ lọc IIR.

Bài 6.11: Thiết kế bộ lọc IIR chắn dải bằng SPTool

Sử dụng phương pháp Elliptic để thiết kế một bộ lọc IIR chắn dải bậc 10, tần số trung tâm 1750Hz. Chú ý rằng Matlab hiển thị bậc bộ lọc là 5, biểu diễn số phần bậc 2 của bộ lọc.

Các thông số của bộ lọc này như sau.:

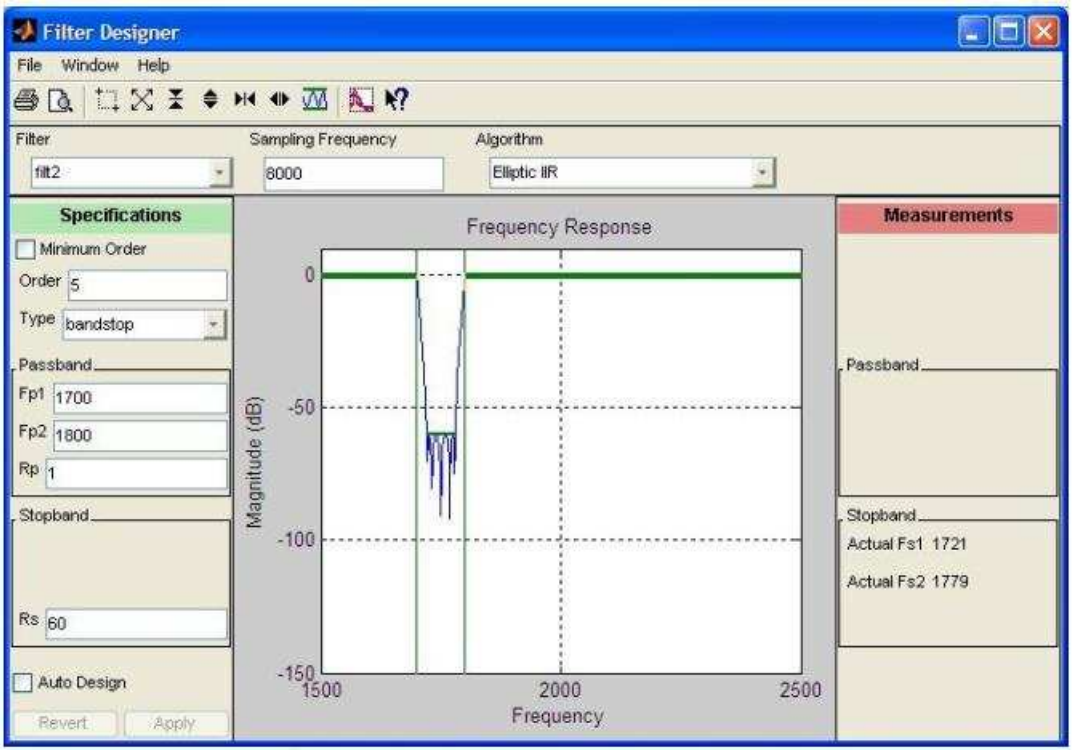
Tần số cắt: 1700 Hz và 1800 Hz

Độ gợn dải thông và dải chắn tương ứng là 1 dB và 60 dB

Tần số lấy mẫu: 8000 Hz

Lưu bộ lọc thiết kế và xuất ra workspace.

Trong workspace sẽ có một cấu trúc tên của bộ lọc. Các hệ số tử số và mẫu số của hàm truyền được lưu tương ứng trong các biến bộ lọc.tf.num và bộ lọc.tf.den.



Hình 6.25: Đáp ứng tần số của bộ lọc của bộ lọc IIR đã thiết kế

Dạng cực – zero của một hàm truyền H(z) như sau:

$$H(z) = k\frac{(z-z_1)(z-z_2)...(z-z_n)}{(z-p_1)(z-p_2)...(z-p_m)}$$

Hàm truyền trên có thể được viết lại như sau:

$$H(z) = g\prod_{k=1}^{L} H_k(z) = g\prod_{k=1}^{L}\frac{b_{0k}+b_{1k}z^{-1}+b_{2k}z^{-2}}{1+a_{1k}z^{-1}+a_{2k}z^{-2}}$$

Với L là số nguyên gần nhất lớn hơn cực đại của n/2 và m/2.

Trong MATLAB, các phần bậc 2 của H(z) được lưu trong 1 ma trận như sau:

$$sos = \begin{bmatrix} b_{01} & b_{11} & b_{21} & 1 & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & 1 & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & 1 & a_{1L} & a_{2L} \end{bmatrix}$$

Từ các hệ số tử và mẫu ở trên, chuyển thành dạng các phần bậc hai bằng các lệnh sau:

```
[z,p,k] = tf2zp(boloc.tf.num, boloc.tf.den)
```

```
    sos = zp2sos(z,p,k)
```

Các giá trị của kết quả sos có thể được sử dụng để đưa qua kit DSP.

Bài 6.12 So sánh thiết kế bộ lọc FIR và IIR

Thiết kế hai bộ FIR -Equiripple và IIR – Butterword có cùng thông số kỹ thuật như sau:

```
Hd1 = designfilt('bandpassfir', ...
    'StopbandFrequency1',1/60,'PassbandFrequency1',1/40, ...
    'PassbandFrequency2',1/4 ,'StopbandFrequency2',1/2 , ...
    'StopbandAttenuation1',10,'PassbandRipple',1, ...

'StopbandAttenuation2',10,'DesignMethod','equiripple','Sampl
eRate',Fs);
Hd2 = designfilt('bandpassiir', ...
    'StopbandFrequency1',1/60,'PassbandFrequency1',1/40, ...
    'PassbandFrequency2',1/4 ,'StopbandFrequency2',1/2 , ...
    'StopbandAttenuation1',10,'PassbandRipple',1, ...

'StopbandAttenuation2',10,'DesignMethod','butter','SampleRat
e',Fs);
```

So sánh bậc của hai bộ lọc:

```
   fprintf('Bac cua bo loc FIR la %d\n',filtord(Hd1))
   fprintf('Bac cua bo loc IIR la %d\n',filtord(Hd2))
```

Kết quả:

```
   Bac cua bo loc FIR la 78
   Bac cua bo loc IIR la  8
```

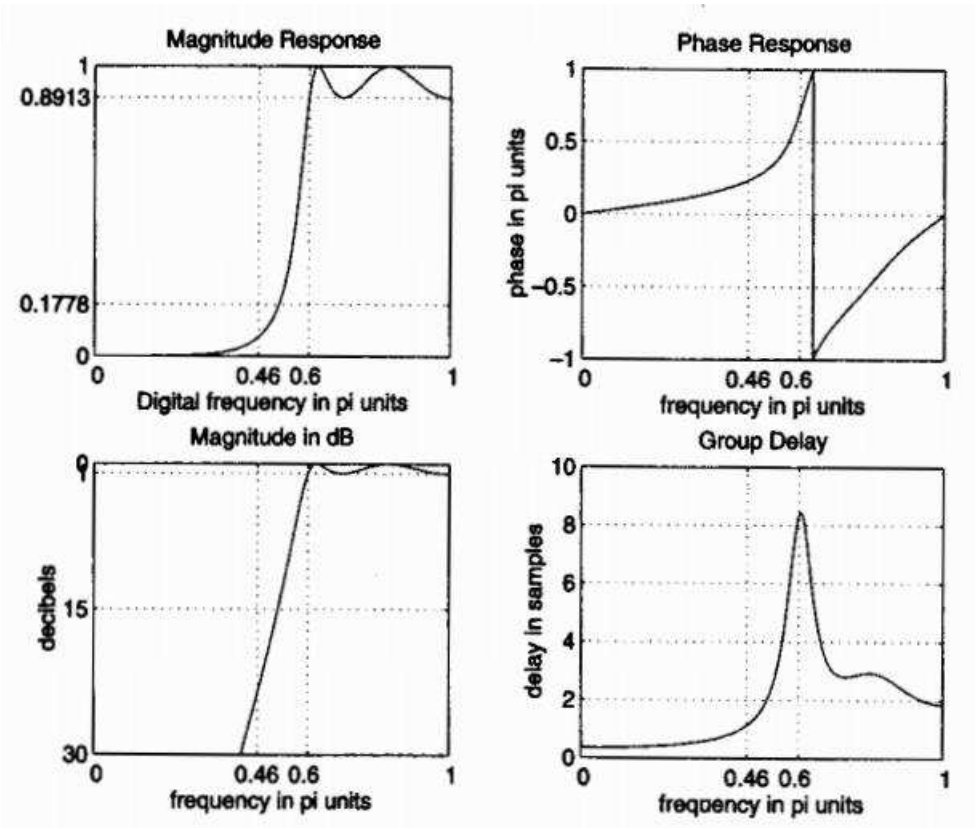
Nhận xét: Với cùng các thông số kỹ thuật bộ lọc IIR cho bậc thấp hơn bộ lọc FIR.

6.5 BÀI TẬP

Bài tập 1: Viết chương trình thiết kế bộ lọc thông thấp Chebyshev I với các thông số:

$$\Omega_p = 0.2\pi, \quad R_p = 1\,\text{dB}$$
$$\Omega_s = 0.3\pi, \quad A_s = 16\,\text{db}$$

Bài tập 2: Viết chương trình thiết kế bộ lọc thông cao có đáp ứng biên độ và pha như hình



Hình 6.26 Đáp ứng biên độ và pha của bộ lọc bài tập 2 chương 6

CHƯƠNG 7 THIẾT KẾ DÃY BỘ LỌC

Nội dung chính

Thiết kế dãy bộ lọc và xử lý tín hiệu số đa tần số.

*Các hàm Matlab liên quan:

Two-Dimensional Graphics

axis	clf	grid	plot	stem
subplot	title	xlabel	ylabel	

Signal Processing Toolbox

decimate	fir2	firpm	freqz	hamming
interp	resample	sinc		

Tóm tắt lý thuyết:

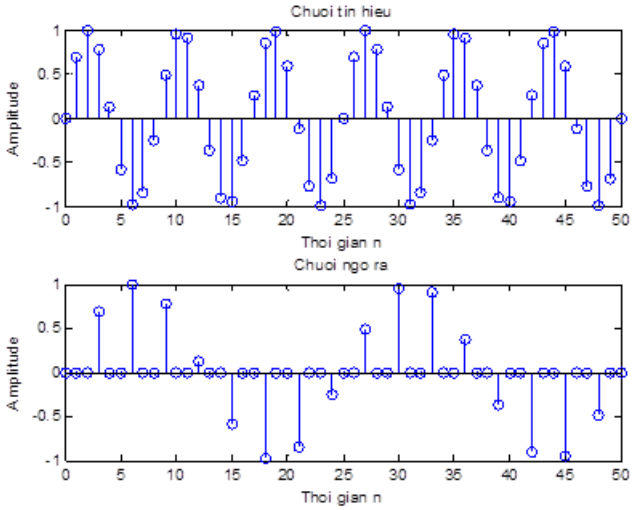
Dãy bộ lọc là sự sắp xếp kết hợp của mạch lọc thông thấp, thông dải và thông cao được sử dụng để phân tích thành phần phổ và tổng hợp tín hiệu. Dãy bộ lọc đóng vai trò quan trọng trong các ứng dụng như mã hoá âm thanh và hình ảnh. Hầu hết bank bộ lọc liên quan đến nhiều tần số lấy mẫu khác nhau, nên còn gọi là hệ thống đa tần số (multi-rate system).

7.1 TĂNG/ GIẢM TỐC ĐỘ LẤY MẪU

Bài 7.1 Mục tiêu của bài này là sử dụng MATLAB khảo sát hoạt động của việc tăng tốc độ lấy mẫu (upsampler) theo hệ số N. Tăng tốc độ lấy mẫu theo hệ số N nghĩa là thêm N-1 zero liên tiếp giữa các mẫu. Điều này giúp khôi phục lại tần số lấy mẫu gốc.

```
%Chuong trinh bai 7.1
clf;
n = 0:50;
x = sin(2*pi*0.12*n);
y = zeros(1, 3*length(x));
```

```
y([1: 3: length(y)]) = x;
subplot(2,1,1)
stem(n,x);
title('Chuoi tin hieu');
xlabel('Thoi gian n');ylabel('Amplitude');
subplot(2,1,2)
stem(n,y(1:length(x)));
title('Chuoi ngo ra');
xlabel('Thoi gian n');ylabel('Amplitude');
```

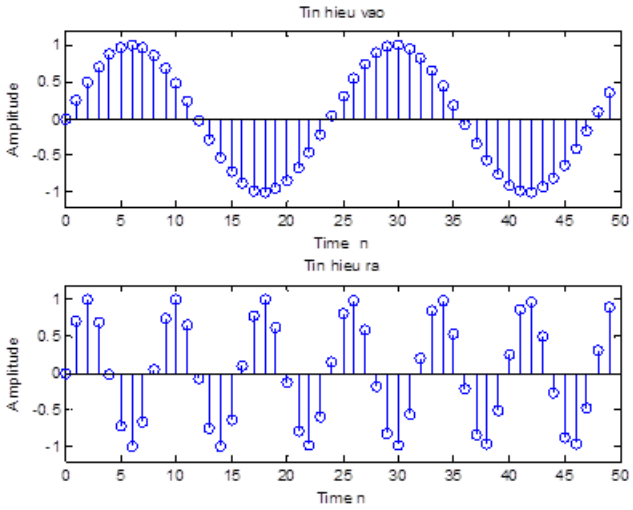


Hình 7.1 Tăng tốc độ lấy mẫu

Bài 7.2 Mục tiêu của bài này là sử dụng MATLAB khảo sát hoạt động của việc giảm tốc độ lấy mẫu (down sampler). Lấy mẫu theo hệ số N nghĩa là chỉ lấy những mẫu thứ N

```
% Lay mau down sampler
clf;
n = 0: 49;
m = 0: 50*3 - 1;
x = sin(2*pi*0.042*m);
y = x([1: 3: length(x)]);
```

```
subplot(2,1,1)
stem(n, x(1:50)); axis([0 50 -1.2 1.2]);
title('Tin hieu vao');
xlabel('Time  n');
ylabel('Amplitude');
subplot(2,1,2)
stem(n, y); axis([0 50 -1.2 1.2]);
title('Tin hieu ra');
xlabel('Time n');
ylabel('Amplitude');
```



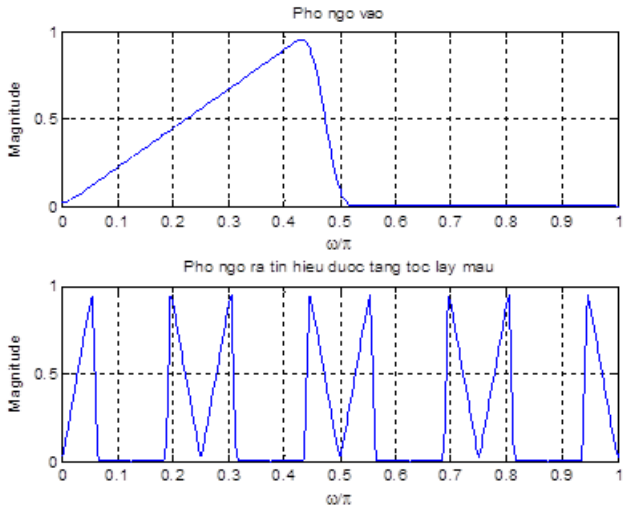
Hình 7.2 Giảm tốc độ lấy mẫu

Bài 7.3 Khảo sát phổ của tín hiệu gốc và tín hiệu được tăng tốc độ lấy mẫu:

```
clf;
freq = [0 0.45 0.5 1];
mag = [0 1 0 0];
x = fir2(99, freq, mag);
[Xz, w] = freqz(x, 1, 512, 'whole');
subplot(2,1,1);
plot(w/pi, abs(Xz)); axis([0 1 0 1]); grid
xlabel('\omega/\pi'); ylabel('Magnitude');
```

```
title('Pho ngo vao');
subplot(2,1,2);
L = input('Nhap he so tang toc do lay mau = ');
y = zeros(1, L*length(x));
y([1: L: length(y)]) = x;
[Yz, w] = freqz(y, 1, 512, 'whole');
plot(w/pi, abs(Yz)); axis([0 1 0 1]); grid
xlabel('\omega/\pi'); ylabel('Magnitude');
title('Pho ngo ra tin hieu duoc tang toc lay mau');
```

Áp dụng với L=8 ta có kết quả như sau:



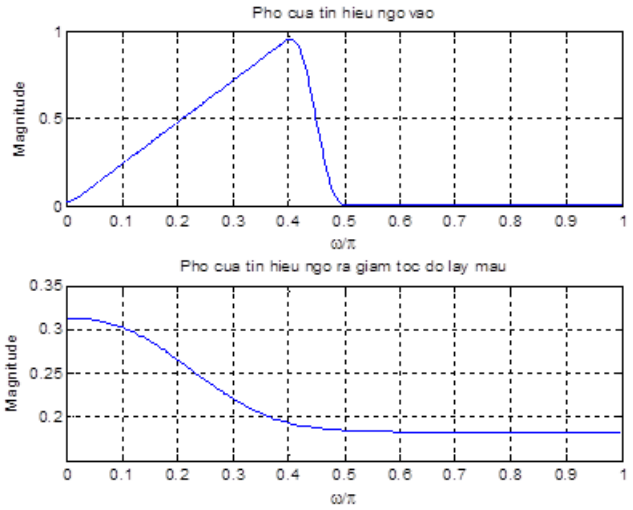
Hình 7.3 Phổ tín hiệu gốc và tín hiệu được tăng tốc độ lấy mẫu

Bài 7.4 Khảo sát phổ của tín hiệu gốc và tín hiệu được giảm tốc độ lấy mẫu:

```
clf;
freq = [0 0.42 0.48 1]; mag = [0 1 0 0];
x = fir2(101, freq, mag);
[Xz, w] = freqz(x, 1, 512);
subplot(2,1,1);
plot(w/pi, abs(Xz)); grid
xlabel('\omega/\pi'); ylabel('Magnitude');
title('Pho cua tin hieu ngo vao');
```

```
M = input('Nhap he so giam toc do lay mau =');
y = x([1: M: length(x)]);
[Yz, w] = freqz(y, 1, 512);
subplot(2,1,2);
plot(w/pi, abs(Yz)); grid
xlabel('\omega/\pi'); ylabel('Magnitude');
title('Output Spectrum');
```

Áp dụng với M=5 ta có kết quả như sau:



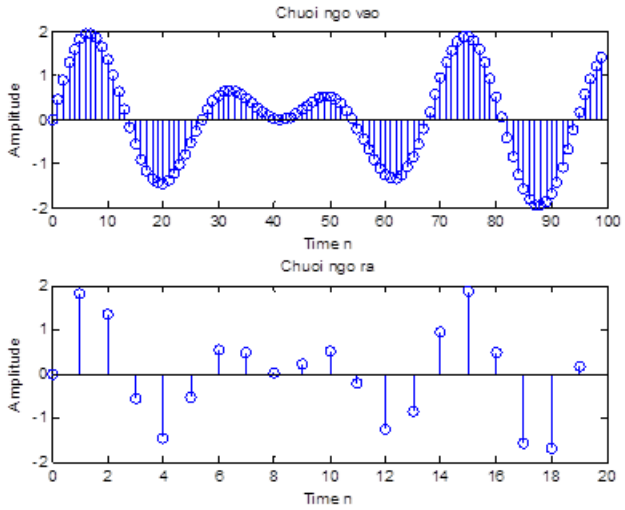
Hình 7.4:Phổ của tín hiệu gốc và tín hiệu được giảm tốc độ lấy mẫu:

Bài 7.5 Thực hiện và quan sát tín hiệu vào và ra của bộ giảm tốc độ lấy mẫu:

```
clf;
M = input('Nhap he so giam toc do lay mau = ');
n = 0:99;
x = sin(2*pi*0.043*n) + sin(2*pi*0.031*n);
y = decimate(x,M,'fir');
subplot(2,1,1);
stem(n,x(1:100));
title('Chuoi ngo vao');
xlabel('Time n');ylabel('Amplitude');
subplot(2,1,2);
```

```
m = 0:(100/M)-1;
stem(m,y(1:100/M));
title('Chuoi ngo ra');
xlabel('Time n');ylabel('Amplitude');
```

Áp dụng với M=5 ta có kết quả như sau:



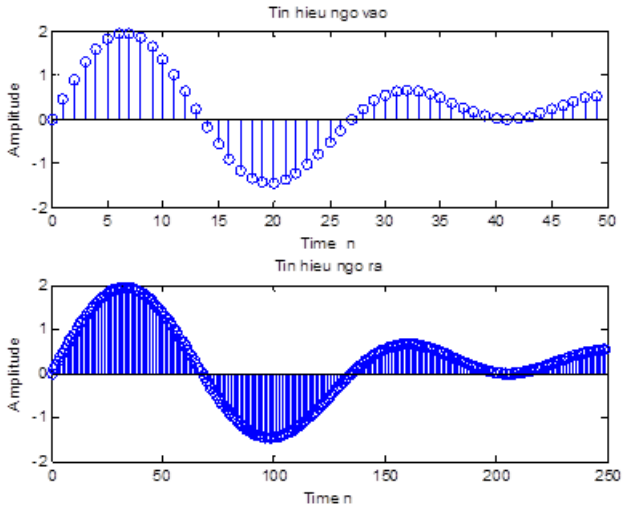
Hình 7.5 Tín hiệu vào và ra của bộ giảm tốc độ lấy mẫu:

Bài 7.6 Thực hiện và quan sát tín hiệu vào và ra của bộ tăng tốc độ lấy mẫu:

```
clf;
L = input('Nhap he so tang toc do lay mau = ');
% Generate the input sequence
n = 0:49;
x = sin(2*pi*0.043*n) + sin(2*pi*0.031*n);
% Generate the interpolated output sequence
y = interp(x,L);
% Plot the input and the output sequences
subplot(2,1,1);
stem(n,x(1:50));
title('Tin hieu ngo vao');
xlabel('Time  n'); ylabel('Amplitude');
```

```
subplot(2,1,2);
m = 0:(50*L)-1;
stem(m,y(1:50*L));
title('Tin hieu ngo ra');
xlabel('Time n'); ylabel('Amplitude');
```

Áp dụng với L=5 ta có kết quả như sau:



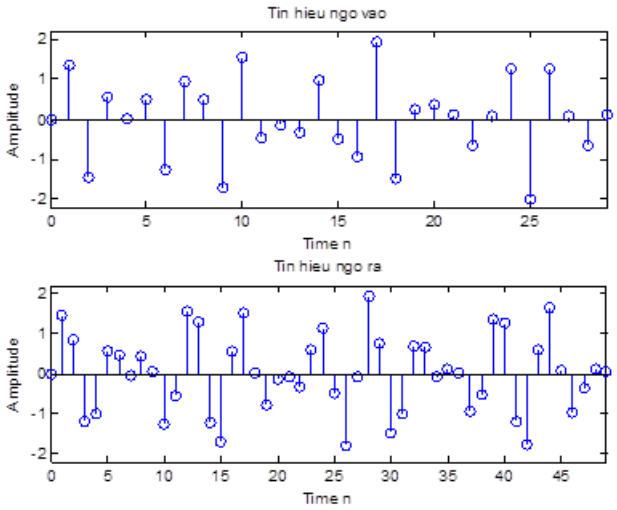
Hình 7.6 Tín hiệu vào và ra của bộ tăng tốc độ lấy mẫu:

Bài 7.7 Thực hiện và quan sát tín hiệu vào và ra của bộ thay đổi tốc độ lấy mẫu theo hệ số phân số hữu tỷ:

```
clf;
L = input('Nhap he so tang toc do lay mau = ');
M = input('Nhap he so giam toc do lay mau = ');
n = 0:29;
x = sin(2*pi*0.43*n) + sin(2*pi*0.31*n);
y = resample(x,L,M);%thay doi toc do lay mau cua tin
hieu
subplot(2,1,1);
stem(n,x(1:30));axis([0 29 -2.2 2.2]);
```

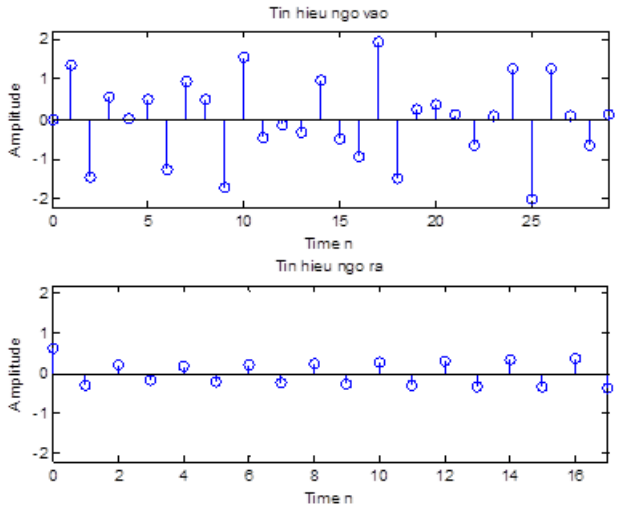
```
title('Tin hieu ngo vao');
xlabel('Time n'); ylabel('Amplitude');
subplot(2,1,2);
m = 0:(30*L/M)-1;
stem(m,y(1:30*L/M));axis([0 (30*L/M)-1 -2.2 2.2]);
title('Tin hieu ngo ra');
xlabel('Time n'); ylabel('Amplitude');
```

Áp dụng với hệ số L/M =5/3 ta có kết quả như sau:,



Hình 7.7a. Tín hiệu vào và ra của bộ thay đổi tốc độ lấy mẫu theo hệ số phân số hữu tỷ L/M=5/3

Áp dụng với hệ số L/M =3/5, ta có kết quả như sau:,



Hình 7.7b. Tín hiệu vào và ra của bộ thay đổi tốc độ lấy mẫu theo hệ số phân số hữu tỷ L/M=3/5

7.2 DÃY BỘ LỌC (FILTER BANK)

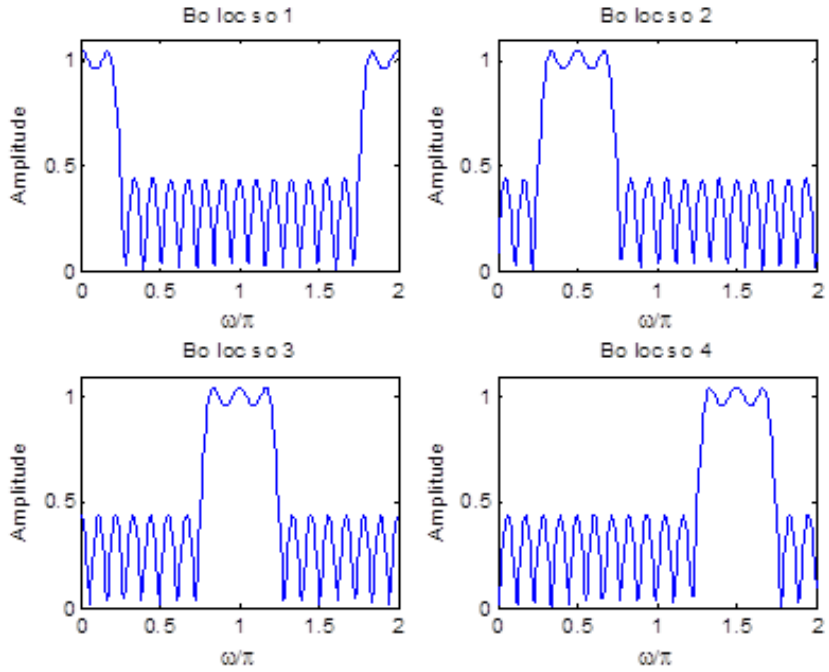
Dãy bộ lọc là sự sắp xếp kết hợp giữa mạch lọc thông thấp, thông dãi và thông cao được sử dụng để phân tích thành phần phổ và tổng hợp tính hợp. Dãy bộ lọc đóng vai trò quan trọng trong các ứng dụng như mã hoá âm thanh và hình ảnh. Vì hầu hết bank bộ lọc liên quan đến nhiều tần số lấy mẫu khác nhau, cho nên còn gọi là hệ thống đa tần số (multi-rate system).

Bài 7.8 Thiết kế mạch lọc thông thấp gồm có M kênh, vẽ đáp ứng cho mỗi bộ lọc, $\omega p < \pi/M < \omega s$

```
clf;
b = firpm(20, [0 0.2 0.25 1], [1 1 0 0], [10 1]);
w = 0:2*pi/255:2*pi; n = 0:20;
for k = 1:4;
c = exp(2*pi*(k-1)*n*i/4);
FB = b.*c;
HB(k,:) = freqz(FB,1,w);
```

```
end
subplot(2,2,1)
plot(w/pi, abs(HB(1,:)));
xlabel('\omega/\pi');ylabel('Amplitude');
title('Bo loc so 1'); axis([0 2 0 1.1]);
subplot(2,2,2)
plot(w/pi,abs(HB(2,:)));
xlabel('\omega/\pi');ylabel('Amplitude');
title('Bo loc so 2');axis([0 2 0 1.1]);
subplot(2,2,3)
plot(w/pi,abs(HB(3,:)));
xlabel('\omega/\pi');ylabel('Amplitude');
title('Bo loc so 3'); axis([0 2 0 1.1]);
subplot(2,2,4)
plot(w/pi,abs(HB(4,:)));
xlabel('\omega/\pi');ylabel('Amplitude');
title('Bo loc so 4'); axis([0 2 0 1.1]);
```

Áp dụng cho dãy bộ lọc 4 kênh



Hình 7.8 : Đáp ứng của dãy mạch lọc (filter bank) có 4 kênh

7.3 BÀI TẬP

Bài tập 1: Viết chương trình Matlab thiết kế dãy bộ lọc hai kênh, hệ số L=2, M=2.

Bài tập 2: Viết chương trình Matlab thiết kế dãy bộ lọc bốn kênh, hệ số tăng, giảm tốc độ lấy mẫu được nhập từ bàn phím. Biểu diễn tín hiệu vào và ra của dãy bộ lọc.