EESC 6360 Spring 2011
Due: April 19, 2011 <<<Note the change

Design and Implementation of FIR Filter Package:

Task: Develop a usable software package to design linear phase FIR filters (Low Pass, Band Pass, Band Stop and High Pass) using Window (Rectangular, Hamming and Kaiser) techniques.

Deliverables: An "experiential" and discussion-oriented project report with a "lessons learned/conclusion" section, including user manual needs to be submitted. You may be "randomly "selected to demonstrate the working of your software and explain the project.

Constraints & Expectations: All the filter specifications will be in terms of CT frequencies, i.e. Hz or kHz, as also all the frequency response plots in your report, and any relevant plots generated while running the software. You need not have fancy user interface, but it must guide the user. Make sure that there are error traps and warnings as appropriate. You must graphically show the filter design specification superposed on the resultant frequency response.

Design and Implementation of FIR Filter

Abstract

In this study, windowing-based Finite Impulse Response (FIR) filter design method is implemented for four regular filters (Low Pass, Band Pass, Band Stop and High Pass). Detailed steps are described. The related codes are uploaded to the website of mathworks.com for sharing. Complete documentation is provided for repetition.

1. Introduction

In reality, FIR filters are employed in filtering problem where there is a requirement for a linear-phase characteristic within the filter passband, which are based on directly approximating the desired frequency response of the discrete time system. The simplest method of FIR filter design is called window method. To obtain a casual FIR filter, a window truncating frequency response of the ideal filter is applied. According to the type of window that is selected different approximation of the filter will be obtained.

In order to design a FIR filter using MATLAB we can use "fir1" and "Filter" commands. In this study, the function "fir1" is used for simplification. To explain the result, we will also use the function of "filter", "freqz" of MATLAB.

1.1 FREQZ function:

[H,W] = FREQZ(B,A,N)

When N is an integer, [H,W] = FREQZ(B,A,N) returns the N-point frequency vector W in radians and the N-point complex frequency response vector H of the filter B/A:

$$H(e^{jw}) = \frac{B(e^{jw})}{A(e^{jw})} = \frac{b(1) + b(2)e^{-jw} + \text{....} + b(nb+1)e^{-jnbw}}{a(1) + a(2)e^{-jw} + \text{....} + a(na+1)e^{-jnaw}}$$

given numerator and denominator coefficients in vectors B and A. The frequency response is evaluated at N points equally spaced around the upper half of the unit circle. If N isn't specified, it defaults to 512.

1.2 FILTER function:

Y = FILTER(B,A,X)

It filters the data in vector X with the filter described by vectors A and B to create the filtered data Y. For FIR filters A=1 because they do not contain any poles. The filter is a "Direct Form II Transposed" implementation of the standard difference equation:

*a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)*
*- a(2)*y(n-1) - ... - a(na+1)*y(n-na)*

If a(1) is not equal to 1, FILTER normalizes the filter coefficients by a(1)

1.3 FIR1 function:

LOW PASS:
B = FIR1(N,Wn) designs an N' th order low pass FIR digital filter and returns the filter coefficients in length N+1 vector B.

HIGH PASS:
B = FIR1(N,Wn,'high') designs an N'th order high pass filter.

BAND PASS:
B = FIR1(N,Wn,'bandpass')
If Wn is a two-element vector, Wn = [W1 W2], FIR1 returns an order N band pass filter with pass band W1 < W < W2. Example: FIR1(N , [f1/(fs/2) f2/(fs/2)])

BAND STOP:
B = FIR1(N,Wn,'stop')
If Wn = [W1 W2], FIR1 will design a band stop filter. Example: FIR1(N , [f1/(fs/2) f2/(fs/2)], 'stop')

2. Data

To clearly check the filtering effect, 4 tones are produced with sampling frequency being set as 8 kHz, and the four individual freq. are set as 500, 1500,2000 and 3000Hz, respectively. Then the four tones are added together to be inputted into the filter system to check the effect. If the filtering is well designed, then it should only allow the specified frequency components pass. Say, to test the band stop filter system, we can set the two edge frequency as: 1000 and 2500 Hz so that we wish only 500Hz and 3000Hz tone can pass the system.

3. Design Method

The design of the digital filter involves the following three steps.

i The first step is the filter specification. This may include the type of filter, the desired amplitude, the sampling frequency and the word length of the input data.

ii The next step is the calculation of the filter coefficients. In this stage, we determine the transfer function, which will satisfy the specification given in (i).

iii The third step is the display of the frequency response graph. In this stage the user decides if he is satisfied with the frequency response graph or not. If he is satisfied then proceed to fourth step and if he is not go back to step i).

Please refer to the attachment 1 for flowchart.

4. Experiment

The code is implemented in a command-line prompt way. All the design process will be presented in the command line of MATLAB. User need input the filter design specification such as: filter type (Low Pass, Band Pass, Band Stop and High Pass). Windowing option (Rectangular, Hamming and Kaiser). All the necessary text prompt is provided to guide the user walk through the whole process. The code is also provided with error trapping mechanism to capture the possible wrong filtering parameter.

Here we take an example from Band-stop filtering with this code.

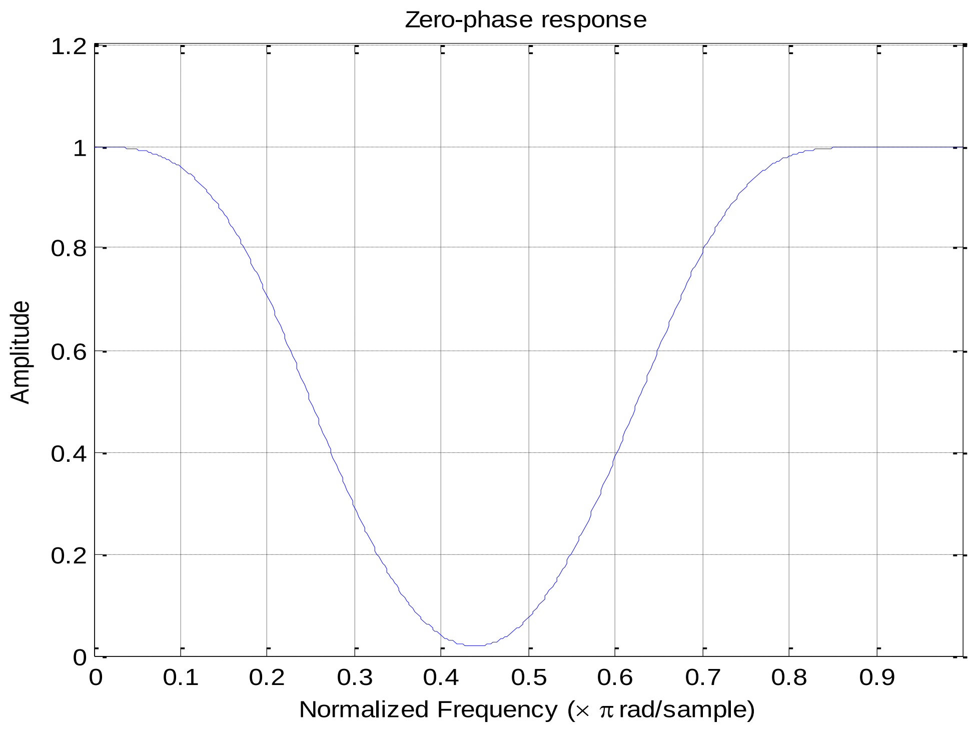


Figure.1 Zero-Phase response of the designed filter

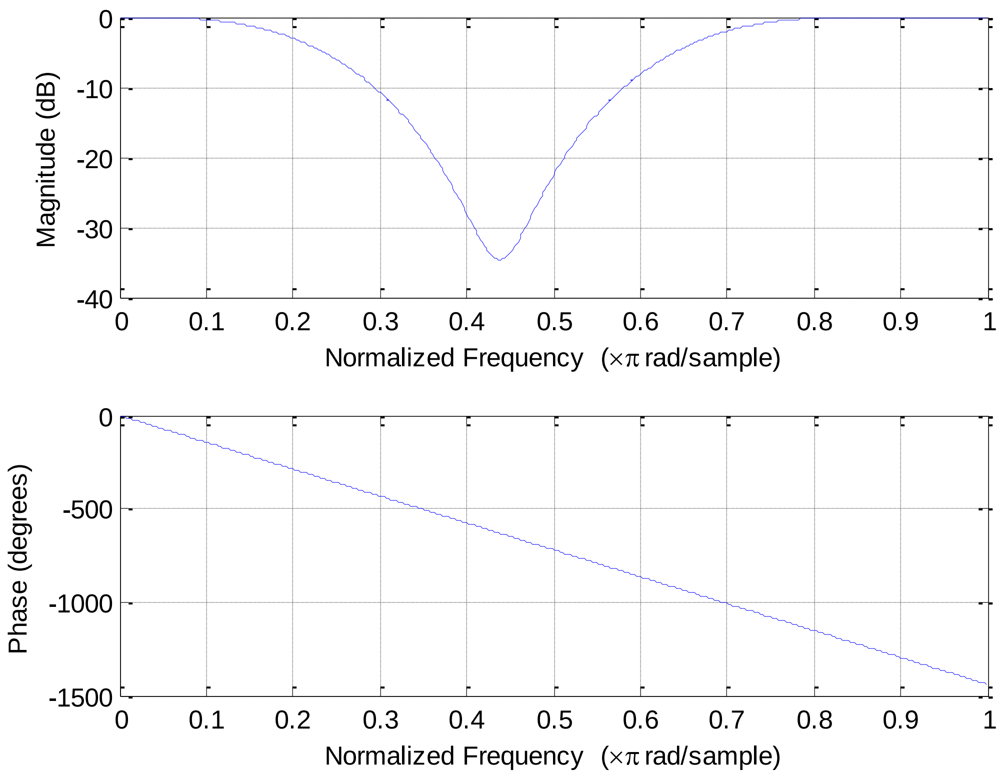


Figure.2 Magnitude and phase response of the designed filter

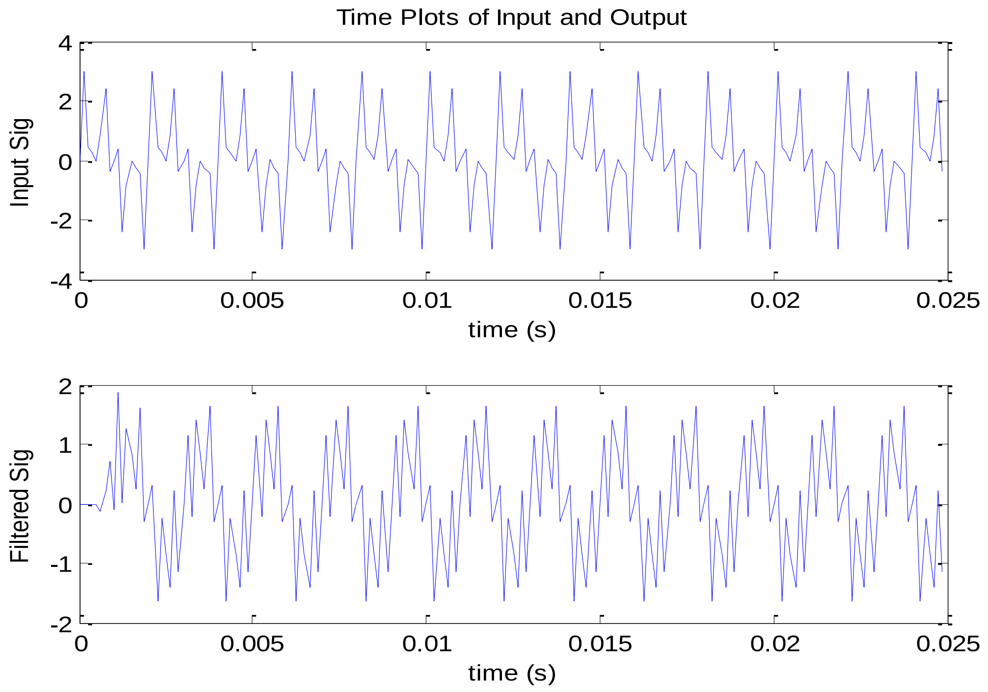


Figure.3 The filtering effect from the time domain perspective.

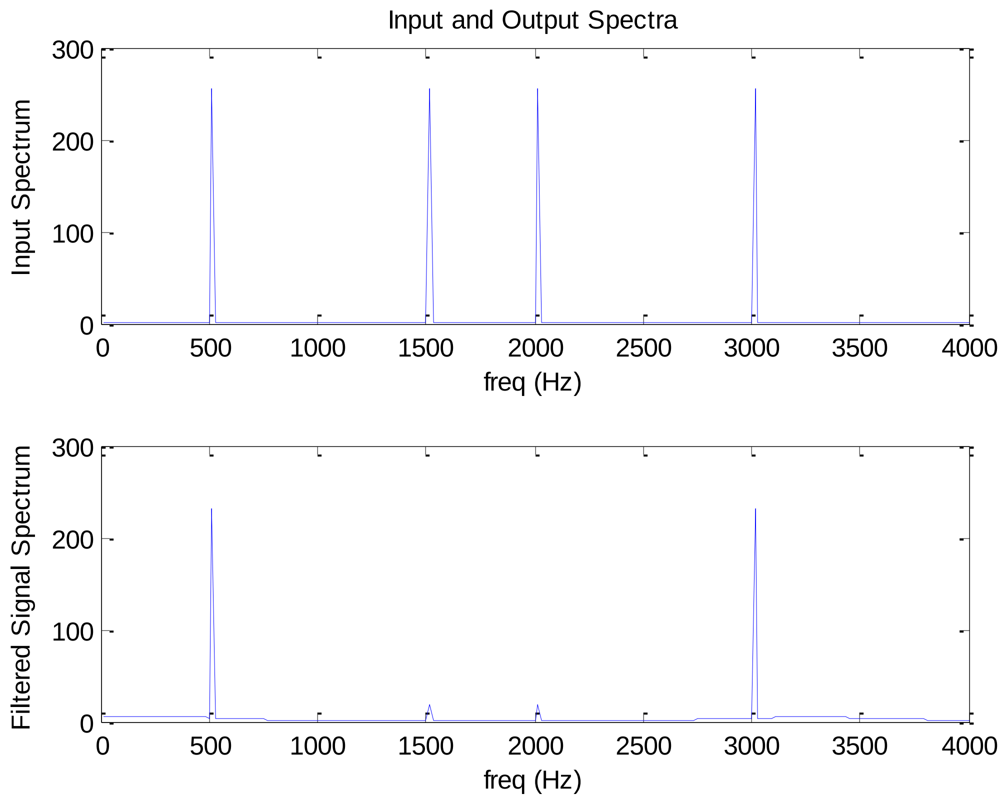


Figure.4 The filtering effect from the frequency domain perspective.

The interactive command line is recorded here:

```
>> FIR_Filter_WindowingMethod
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Please specify the filter type
Here is the filter option:
1:    Low Pass Filter
2:    High Pass Filter
3:    Band Pass Filter
4:    Band Stop Filter
You only need enter number. EX. 4 means Band Stop filter
FilterOption=4
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Enter the order of the filter(Must be a positive even number, should be less than 100
FilterOrder=16
Here is all the frequency infromation in this experiment:Fs=8000;     f1=500;   f2=1500;
f3=2000;  f4=3000;
Enter the Edge Freq. in Hz. Note: the value should be within 0~Fs
omega cutoff1=1000
omega cutoff2=2500
```

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Here is the Windowing option:
1:    hamming
2:    kaiser
3:    rectwin
WindowOption=1
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
Checking the Zero-Phase Response of the designed filter, enter any key to continue
Checking the Freq. Response of the designed filter, enter any key to continue
Checking the Filtering Effect of the designed filter in time domain, enter any key to
continue
```

5. Analysis

When we set the edge frequency as: 1000Hz and 2500 Hz, the designed filter allows all the frequency components passing except the frequency within 1000~2500Hz. So the filter meets its end. And the “linear phase” constraints are assured.

From figure 4, we note there are still some weak energy component remained after the filtering. These remains can be removed by increasing filtering order, *N* at the cost of more computation resources.

6. Conclusion

The designed FIR can successfully fulfill the filtering task in this simple case. Historically, the design method based on the use of windows to truncate the impulse response and to obtain the desired spectral shaping was the first method proposed for designing linear-phase FIR filters. It is straightforward and easy to be implemented.

The major disadvantage of the window design method is the lack of precise control of the critical frequencies, such as Wp and Ws, in the design of a lowpass FIR filter. The values of Wp and Ws, in general, depend on the type of window and the filter length M.

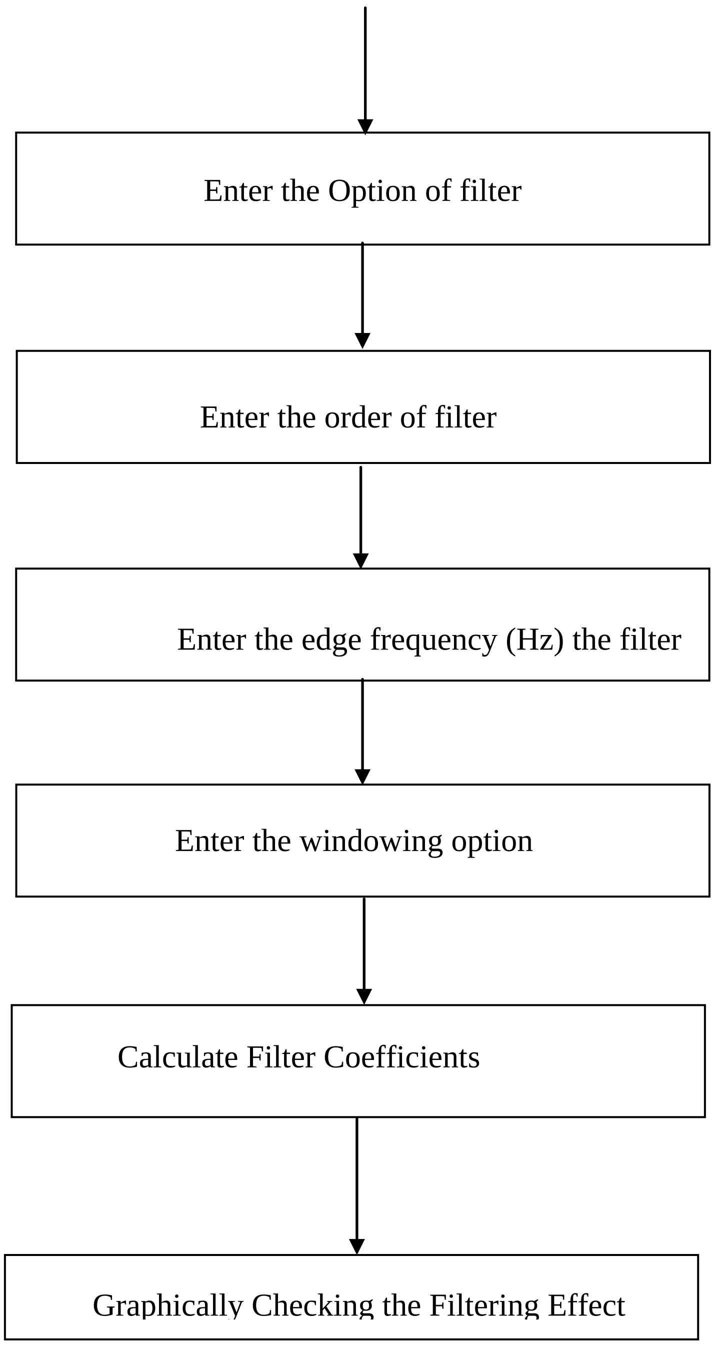
The frequency-sampling method and the Chebyshev approximation method were developed in the 1970s and have since become very popular in the design of practical linear-phase filters.

The frequency-sampling method provides an improvement over the window design method since it can exactly control the filter spectrum at the equal spaced frequency point (2pik/M or 2pi(2k+1)/M). The Chebyshev approximation method can provide total control of the filter specification, and as a consequence, it is usually preferable over the other two methods.

Attachment:

Attachment 1:	Flowchart of the Design and Implementation of FIR Filter
Attachment 2:	MATLAB Code

Attachment 1: Flowchart of the Design and Implementation of FIR Filter


Enter the Option of filter
Enter the order of filter
Enter the edge frequency (Hz) the filter
Enter the windowing option
Calculate Filter Coefficients
Graphically Checking the Filtering Effect

Attachment 2:   MATLAB Code

%%%%%%%%%%%%%%%%%%%%%%

```
function FIR_Filter_WindowingMethod

%  Design and Implementation of FIR Filter Package
% Date: 04/16/2011
%  Author: Gang LIU
% Contacts: liug "at" yahoo dot cn



%
% This code is based on the some code from MATLAB Singal Processing
% Toolbox.
%
%.  Example MATLAB M-file illustrating FIR filter design and
evaluation.

% Finite Impulse Response filter design example
% found in the MATLAB Signal Processing Toolbox
% using the MATLAB FIR1 function (M-file)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all;
Fs=8000;  %Specify Sampling Frequency
Ts=1/Fs; %Sampling period.
Ns=512;  %Nr of time samples to be plotted.

t=[0:Ts:Ts*(Ns-1)];  %Make time array that contains Ns elements
                     %t = [0, Ts, 2Ts, 3Ts,..., (Ns-1)Ts]
f1=500;
f2=1500;
f3=2000;
f4=3000;

x1=sin(2*pi*f1*t); %create sampled sinusoids at different frequencies
x2=sin(2*pi*f2*t);
x3=sin(2*pi*f3*t);
x4=sin(2*pi*f4*t);

x=x1+x2+x3+x4;  %Calculate samples for a 4-tone input signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%N=16;            %FIR1 requires filter order (N) to be EVEN
                 %when gain = 1 at Fs/2.
%W=[0.4 0.6];     %Specify Bandstop filter with stop band between
                 %0.4*(Fs/2) and 0.6*(Fs/2)

%B=FIR1(N,W,'DC-1'); %Design FIR Filter using default (Hamming window.

disp('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$');
disp('Please specify the filter type');
FilterType={'low','high','bandpass','stop'};
disp('Here is the filter option:');
disp('1:    Low Pass Filter ');
disp('2:    High Pass Filter');
disp('3:    Band Pass Filter');
disp('4:    Band Stop Filter');
disp('You only need enter number. EX. 4 means Band Stop filter')
FilterOption=input('FilterOption=');
while FilterOption ~=1 && FilterOption ~=2 && FilterOption ~=3 &&
FilterOption ~=4
    disp('You only need enter number : 1,2,3 or 4. EX. 4 means Band
Stop filter');
    FilterOption=input('FilterOption=');
end
userFilterType=FilterType{FilterOption};
disp('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$');
disp('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$');
disp('Enter the order of the filter(Must be a positive even number,
should be less than 100');

FilterOrder=input('FilterOrder=');
str=sprintf('Here is all the frequency infromation in this
experiment:Fs=%d;    f1=%d;  f2=%d;  f3=%d;  f4=%d;',Fs,f1,f2,f3,f4);
disp(str);
if FilterOption==1
        disp('Enter the Edge Freq. in Hz. Note: the value should be
within 0~Fs');W=input('omega=');
elseif FilterOption==2
        disp('Enter the Edge Freq. in Hz. Note: the value should be
within 0~Fs');W=input('omega=');
elseif FilterOption==3
        disp('Enter the Edge Freq. in Hz. Note: the value should be
within 0~Fs');W=input('omega cutoff1=');W2=input('omega cutoff2=');
W=[W W2];
elseif FilterOption==4
        disp('Enter the Edge Freq. in Hz. Note: the value should be
within 0~Fs');W=input('omega cutoff1=');W2=input('omega cutoff2=');
W=[W W2];
end

  %Fs,    W,
  W=W./(Fs/2);  % [1600 2400]
N=FilterOrder;


disp('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$');
disp('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$');
Window_Option={'hamming','kaiser','rectwin'};
disp('Here is the Windowing option:');
disp('1:    hamming');
disp('2:    kaiser');
disp('3:    rectwin');
WindowOption=input('WindowOption=');
while WindowOption ~=1 && WindowOption ~=2 && WindowOption ~=3
```

```
    disp('You only need enter number : 1,2,or 3. EX. 3 means rectwin');
    WindowOption=input('WindowOption=');
end


%WindowOption='rectwin';
switch Window_Option{WindowOption}
    case 'hamming'
        userWindow=hamming(N+1);
    case 'kaiser'
        userWindow=kaiser(N+1);
    case 'rectwin'
        userWindow=rectwin(N+1);
    otherwise
        disp('Wrong windowing option. EXIT');
        eixt
end

disp('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$');
disp('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$');

%B=FIR1(N,W,'bandpass',userWindow); %Design FIR Filter using default
(Hamming window.
B=fir1(N,W,userFilterType,userWindow); %Design FIR Filter using default
(Hamming window.
%B                %Leaving off semi-colon causes contents of
                %B (the FIR coefficients) to be displayed.
A=1;            %FIR filters have no poles, only zeros.
figure;
zerophase(B,A); % plot zero phase figure

disp('Checking the Zero-Phase Response of the designed filter, enter
any key to continue');
pause;


figure;       %Create a new figure window, so previous one isn't lost.

freqz(B,A);      %Plot frequency response - both amp and phase
response.

disp('Checking the Freq. Response of the designed filter, enter any key
to continue');
pause;
figure;

subplot(2,1,1); %Two subplots will go on this figure window.
Npts=200;
plot(t(1:Npts),x(1:Npts)) %Plot first Npts of this 4-tone input signal
title('Time Plots of Input and Output');
xlabel('time (s)');
ylabel('Input Sig');
   %Now apply this filter to our 4-tone test sequence
```

```
y = filter(B,A,x);

subplot(2,1,2);  %Now go to bottom subplot.
plot(t(1:Npts),y(1:Npts)); %Plot first Npts of filtered signal.
xlabel('time (s)');
ylabel('Filtered Sig');
disp('Checking the Filtering Effect of the designed filter in time
domain, enter any key to continue');
pause;

figure;  %Create a new figure window, so previous one isn't lost.
subplot(2,1,1);
xfftmag=(abs(fft(x,Ns)));   %Compute spectrum of input signal.
xfftmagh=xfftmag(1:length(xfftmag)/2);
   %Plot only the first half of FFT, since second half is mirror imag
   %the first half represents the useful range of frequencies from
   %0 to Fs/2, the Nyquist sampling limit.
f=[1:1:length(xfftmagh)]*Fs/Ns;   %Make freq array that varies from
                                  %0 Hz to Fs/2 Hz.
plot(f,xfftmagh);       %Plot frequency spectrum of input signal
title('Input and Output Spectra');
xlabel('freq (Hz)');
ylabel('Input Spectrum');
subplot(2,1,2);
yfftmag=(abs(fft(y,Ns)));
yfftmagh=yfftmag(1:length(yfftmag)/2);
   %Plot only the first half of FFT, since second half is mirror image
   %the first half represents the useful range of frequencies from
   %0 to Fs/2, the Nyquist sampling limit.
plot(f,yfftmagh);       %Plot frequency spectrum of input signal
xlabel('freq (Hz)');
ylabel('Filtered Signal Spectrum');
```