The University of Texas at Austin

Spring 2014 EE 445S Real-Time Digital Signal Processing Laboratory Prof. Evans

Homework #2 Solutions on Filter Analysis, Simulation and Design

2.1 Frequency Responses

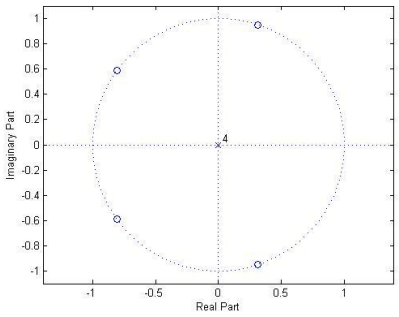
For each LTI system in problem 1.1 on homework assignment #1,
a) plot the pole-zero diagram for the transfer function.
b) is the filter bounded-input bounded-output (BIBO) stable? why or why not?
c) give a formula for the frequency response.
d) plot the magnitude response.
e) if the system is BIBO stable, pick the best one of the following choices to describe the frequency selectivity of the filter: lowpass, highpass, bandpass, or bandstop.

Solution:

(1) Causal five-tap averaging filter. Transfer function from solution to homework problem 1.1:

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

a) Pole-zero diagram (plot in MATLAB using `zplane([1/5 1/5 1/5 1/5 1/5])`



The four zeros are complex-valued roots of one and are equally spaced around the unit circle except that there is no zero at $z = 1$. There are also four "artificial" poles at the origin due to rewriting the transfer function as $(z^4 + z^3 + z^2 + z + 1) / z^4$.

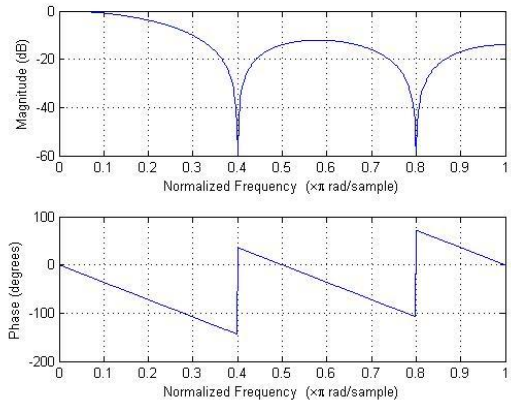
b) The causal system is bounded-input bounded-output (BIBO) stable because all poles are inside the unit circle. Alternately, since the region of convergence (ROC) $z \neq 0$ includes the unit circle, the system is BIBO stable. Moreover, FIR filters are always BIBO stable.

c) Frequency response: Since the unit circle is in the ROC, we replace z in $H(z)$ with $e^{j\omega}$:

$$H(e^{j\omega}) = \frac{1}{5}(1 + e^{-j\omega} + e^{-2j\omega} + e^{-3j\omega} + e^{-4j\omega})$$

Note the discrete-time Fourier transform is periodic in ω with period 2π.

d) Magnitude response: plot in MATLAB using `freqz([1/5 1/5 1/5 1/5 1/5])`



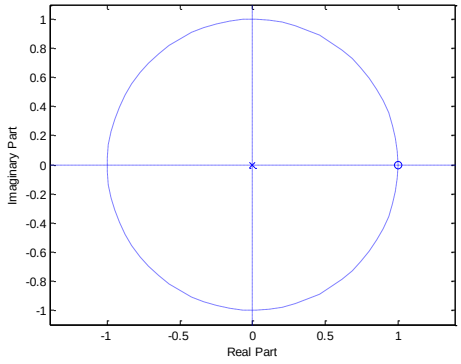
Although not asked, phase response is linear except at the two frequencies removed by filter (i.e. the two frequencies where the magnitude response goes to zero).

e) Lowpass filter. As mentioned in class, the stopband attenuation is 13.5 dB for an averaging filter regardless of number of coefficients N. The null bandwidth is $2\pi / N$. Not a great lowpass filter, but lowpass nonetheless.

(2) Causal discrete-time approximation to first-order differentiator.

$$H(z) = \frac{Y(z)}{X(z)} = 1 - z^{-1}$$

a) Pole-zero diagram: plot in MATLAB using `zplane([1 -1])`

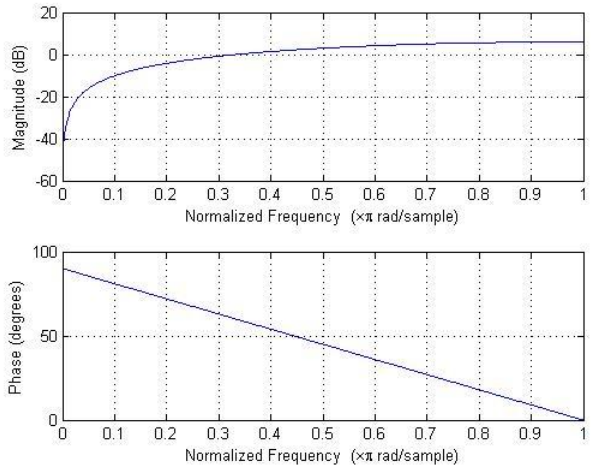


The transfer function has a zero at $z = 1$. There is an "artificial" pole at $z = 0$ which arises when rewriting the transfer function as $(z - 1) / z$.

b) Filter is BIBO stable because pole is inside unit circle. All FIR filters are BIBO stable.

c) Frequency response: Since the unit circle is in the ROC, we replace z in $H(z)$ with $e^{j\omega}$ and we obtain $H(e^{j\omega}) = 1 - e^{-j\omega}$

d) Magnitude response is plotted in MATLAB using `freqz([1 -1])`



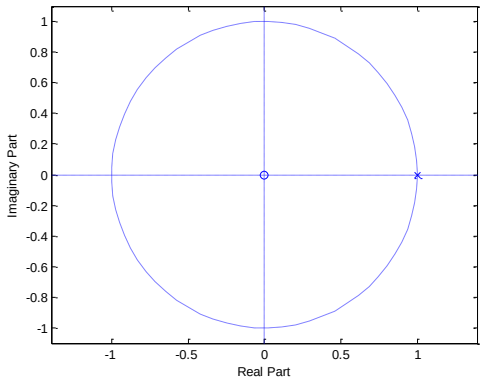
Although not asked, phase response is linear.

e) Highpass filter, or notch filter because it notches out (eliminates) zero frequency.

(3) Causal discrete-time approximation to a first-order integrator.

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - z^{-1}}$$

a) Pole-zero diagram: `zplane(1,[1 -1])`



b) Not BIBO stable, because the pole is at $z = 1$, which is on the unit circle.

c) Frequency response: Since the ROC of the transfer function does not include the unit circle, substituting $z = e^{j\omega}$ in the z-transform expression is not mathematically valid. Instead, the frequency response can be calculated by first obtaining the impulse response of the system and the transforming the impulse response to the discrete-time Fourier domain. The impulse response of a first-order integrator is given by $h[n] = u[n]$; i.e., for an input of a discrete-time impulse $\delta[n]$, the output is $h[n] = h[n\text{-}1] + \delta[n]$ with $h[-1] = 0$. The frequency response is the discrete-time Fourier transform of the unit step:

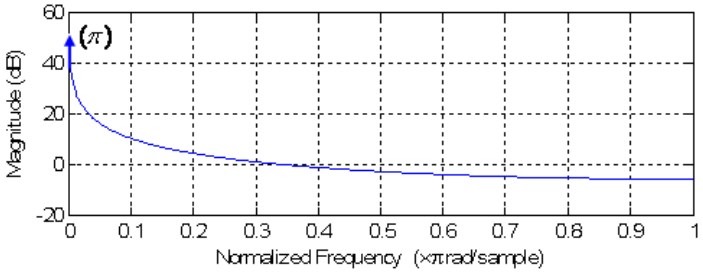
$$F\{u[n]\} = H(e^{j\omega}) = \frac{1}{1 - e^{-j\omega}} + \pi \sum_{k=-\infty}^{\infty} \delta(\omega - 2\pi k)$$

Note that the discrete-time Fourier transform is periodic in ω with period 2π.

d) Magnitude response is the magnitude of H(e j):

$$\left|H(e^{j\omega})\right| = \left|\frac{1}{1-e^{-j\omega}} + \pi\sum_{k=-\infty}^{\infty}\delta(\omega - 2\pi k)\right| \le \left|\frac{1}{1-e^{-j\omega}}\right| + \left|\pi\sum_{k=-\infty}^{\infty}\delta(\omega - 2\pi k)\right|$$

We can plot the left term by using (abusing) the Matlab command `freqz(1, [1 -1])`. This command uses the transfer function and not the discrete-time Fourier transform. We have manually added the right term, which is single Dirac delta for ω in [0, π]:



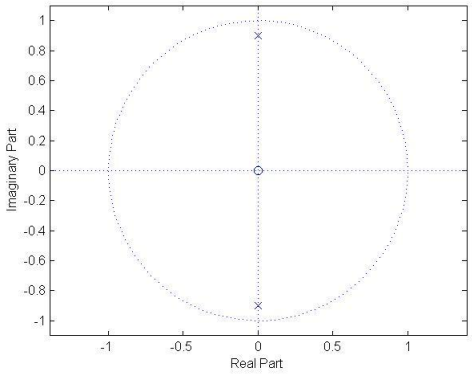
e) Type of filter: Not applicable because the system is not BIBO stable. The magnitude response grows unbounded as the frequency approaches zero. Otherwise, the magnitude response resembles a lowpass filter. If the input signal had its DC component removed, e.g. by a notch filter, then the LTI system could be used to filter other frequencies. In fact, that is commonly done in practice.

(4) Causal bandpass filter with center frequency ω_0

For plots, place poles close to but inside the unit circle, e.g. set the pole radius *r* to be 0.9 or 0.95 (from the hints) and a value of ω_0 between π/4 and 3π/4. We'll use r = 0.9 and ω_0=π/2:

$$H(z) = \frac{1 - \cos\omega_0 z^{-1}}{1 - 2(\cos\omega_0)rz^{-1} + r^2z^{-2}}$$

a) Pole-zero diagram: `zplane([1 -cos(w0)], [1 -2*r*cos(w0) r^2])`

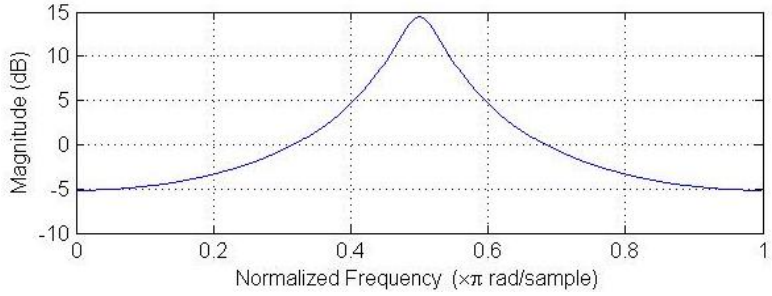


b) BIBO stable because both poles are inside unit circle due to |*r*| <1

c) Frequency response: Since |*r*| < 1, we obtain frequency response by substituting $z = e^{j\omega}$:

$$H(e^{j\omega}) = \frac{1 - \cos\omega_0 e^{-j\omega}}{1 - 2(\cos\omega_0)re^{-j\omega} + r^2z^{-2j\omega}}$$

d) Plot of magnitude responses: `freqz([1 -cos(w0)],[1 -2*r*cos(w0) r^2])`



e) Type of filter: The selectivity depends on the value of r and ω_0. For $0.9 < r < 0.95$, the filter is lowpass when $\omega_0 \approx 0$, highpass when $\omega_0 \approx \pi$ and bandpass for values of ω_0 for values in between. (If r=1, system would not be BIBO stable and hence not a filter according the definition of filter used in this class. Magnitude response would grow unbounded as frequency approaches ω_0 and $-\omega_0$.)

2.2 Finite Impulse Response Filter Design for Audio Signals

Johnson, Sethares & Klein, exercise 7.21, on page 149. Complete all parts. Please use a lowpass filter in part (a). Here is the problem from Johnson, Sethares and Klein:

"In Exercise 7.10, the program specgong.m was used to analyze the sound of an Indonesian gong. The three most prominent partials (or narrowband components) were found to be at about 520, 630, and 660 Hz.
a. Design a filter using firpm that will remove the two highest partials from this sound without affecting the lowest partial.
b. Use the filter command to process the gong.wav file with your filter.
c. Take the FFT of the resulting signal (the output of your filter) and verify that the partial at 520 remains while the others are removed.
d. If a sound card is attached to your computer, compare the sound of the raw and the filtered gong sound by using Matlab's sound command. Comment on what you hear."

Solution for parts (a), (b) and (c): The gong signal "consists primarily of three major frequencies, at about 520, 630, and 660 Hz. Physically, these represent the three largest resonant modes of the vibrating plate." (page 140)

To remove the higher frequency components, we design a lowpass filter to pass the 520 Hz component and attenuate the 630 Hz and 660 Hz components. The passband frequency could be chosen as 530 Hz. The stopband frequency could be chosen to be as high as 620 Hz.

Using the command firpm in MATLAB, we designed a lowpass finite impulse response filter with a passband frequency of 530 Hz and stopband frequency of 585 Hz. The choice of 585 Hz is arbitrary. We chose a stopband frequency that was 10% higher than the passband frequency.

In the MATLAB code below, the prolog contains specgong.m from *Software Receiver Design*. The remaining parts are (a) design the filter, (b) apply the filter, and (c) plot the results.

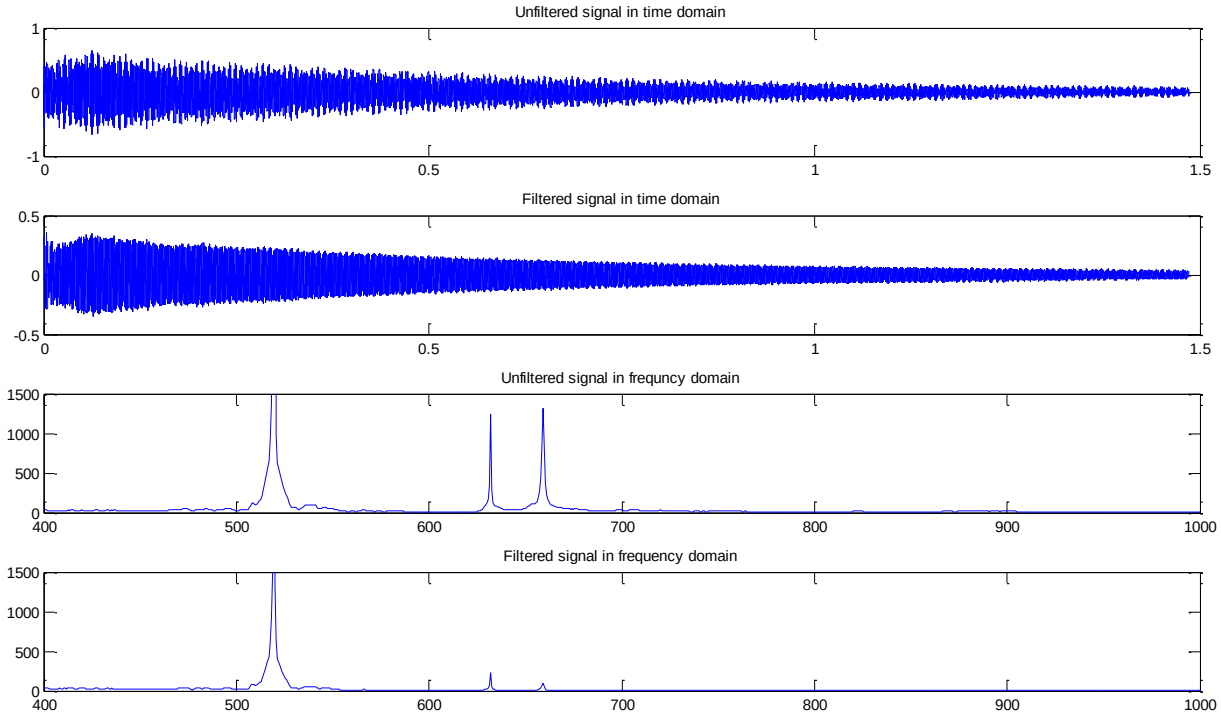
```
%% Prolog: specgong.m from Software Receiver Design text
filename='gong.wav' ;                % name of wave file goes here
[ x , sr ]=wavread( filename ) ;     % read in wavefile
Ts=1/ sr ; siz=length ( x ) ;        % sample interval and # of samples
N=2^16; x=x(1:N)';                   % length for analysis
sound(x , 1 / Ts )                   % play sound , if sound card installed
time=Ts * ( 0 : length(x)-1);        % establish time base for plotting
subplot ( 4 , 1 , 1 ) , plot ( time , x );
title ('Unfiltered signal in time domain');         % and plot top figure
magx=abs ( fft(x) ) ;                % take FFT magnitude
ssf =(0:N/2-1)/(Ts*N) ;              % establish freq base for plotting
subplot( 4 , 1 , 3 ), plot(ssf, magx ( 1:N/2));
title ('Unfiltered signal in frequncy domain');   % plot mag spectrum

%% (a) filter design
%% Design a filter using firpm that will remove the two highest partials from this
sound without affecting the lowest partial.
%% Nyquist frequency, i.e. half the sampling rate
fnyquist = sr/2;
%% Define the passband frequency in Hz
fpassband = 530;
%% Define the stopband frequency in Hz
fstopband = 585;
ctfrequencies = [0 fpassband fstopband fnyquist];
pmfrequencies = ctfrequencies / fnyquist;
%% Define the number of coefficients for the filter
filterlength = 100;
lowpassamps = [ 1 1 0 0 ];
lowpassfilter = firpm( filterlength, pmfrequencies, lowpassamps );

%% (b) filter signal
%% Use the filter command to process the gong.wav file with your filter.
y = filter(lowpassfilter, 1, x);

%% (c) plot the results
%% Take the FFT of the resulting signal and verify that the partial at 520 remains
while the others are removed.
subplot(4, 1, 2), plot (time, y);
title ('Filtered signal in time domain');

magx1 = abs( fft( y ) );
subplot( 4, 1, 4), plot (ssf, magx1 ( 1:N/2));
title ('Filtered signal in frequency domain');
sound (y, 1/Ts);
```



(d) Compare the sound of the raw and the filtered gong sound by using Matlab's sound command. Comment on what you hear.

Solution for part (d): The lowpass filtered signal sounds more like a single tone because we have reduced the strength of the highest two principal frequency components.

(e) Take the filtered gong signal from (b) and perform downsampling by 2. Downsampling by 2 keeps every other sample and discards the rest. Play the downsampled filtered gong signal at the same playback rate as the filtered gong signal. How does it differ from the filtered gong signal? Plot the magnitude spectrum of the downsampled filtered gong signal and compare it against the magnitude spectrum of the filtered going signal.

Solution for part (e): To accomplish downsampling by a factor of 2, we remove every other sample in the signal. If we play back this signal at the same sample rate as that of the original, we notice that the sound clip lasts half as long and the principal frequency has doubled. The latter is because we have fewer samples of the same signal, but we play them back at the same rate.

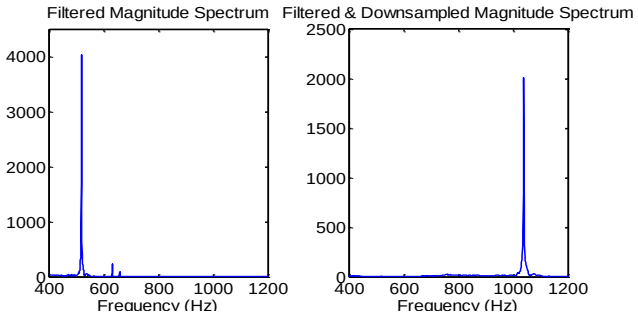
```
yDownSampled = y(1:2:length(y));
sound(yDownSampled, 1/Ts)
```

Since the downsampled signal vector is half as long as the original, the FFT of the downsampled signal vector should be half as long as well. Thus, we have half as many samples to represent frequencies from 0 Hz to fs/2 Hz. Accordingly, the frequency of each tone in the signal will double, and the spectrum will stretch in width by a factor of 2.

```
% Compute filtered spectrum and frequency vector as in specgong.m
magY = abs(fft(y));
N = length(magY);
f =(0:N/2-1)/(Ts*N);

% Compute downsampled spectrum
magYd = abs(fft(yDownSampled));
N2 = length(magYd);
f2 = (0:N2/2-1)/(Ts*N2);
```

The magnitude spectra of the filtered signal (left) and the filtered and downsampled signal (right) are plotted on the right. Note that we have zoomed into the frequency axis.



2.3 Finite Impulse Response Filter Design for an ECG Device

"There exist three types of noise that contaminate the ECG signal: the baseline wander noise (BW), electromyographic interference (EMG), and the power line interference. The BW is induced by electrodes' changes due to perspiration, movement and respiration, and is typically below 0.5 Hz. The power line interference [is] either 50 Hz or 60 Hz and its harmonics are a significant source of noise." [1] EMG noise appears in the same frequencies as the ECG signal. Our goal in designing the filter is to attenuate (reduce) the baseline wander noise and powerline interference. It would take sophisticated processing to track and cancel the EMG noise.

Here are the bandpass filter specifications for your design:

For frequencies 0 Hz to 1 Hz, the stopband attenuation should be at least 40 dB.

For frequencies 6 Hz to 40 Hz, the passband ripple should be no greater than 1 dB.

For frequencies above 45 Hz, the stopband attenuation should be at least 40 dB.

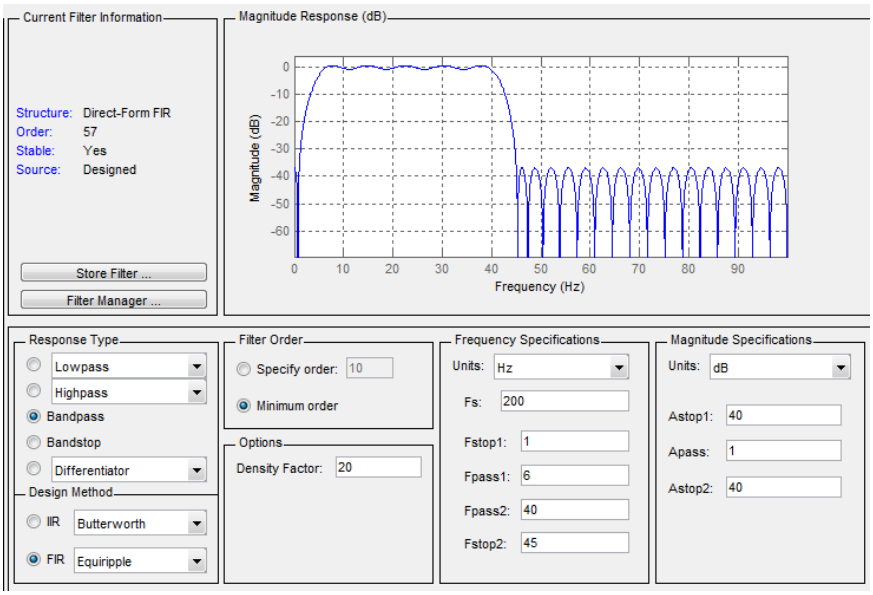
These specifications would be compatible with the monitor mode in modern ECG monitors.

Please use a sampling rate of 200 Hz. From a quick "sampling" of commercial ECG systems, I found sampling rates that vary from 100 Hz to 1000 Hz. The PTB Diagnostic ECG Database uses a sampling rate of 1000 Hz and the QT ECG Database uses a sampling rate of 250 Hz.

a) Design FIR filters *with the minimum filter order* to meet the specification by using the Parks-McClellan (Remez), FIR Least Squares (FIRLS), and Kaiser Window design methods. Turn in a plot of the magnitude and phase response for each filter.

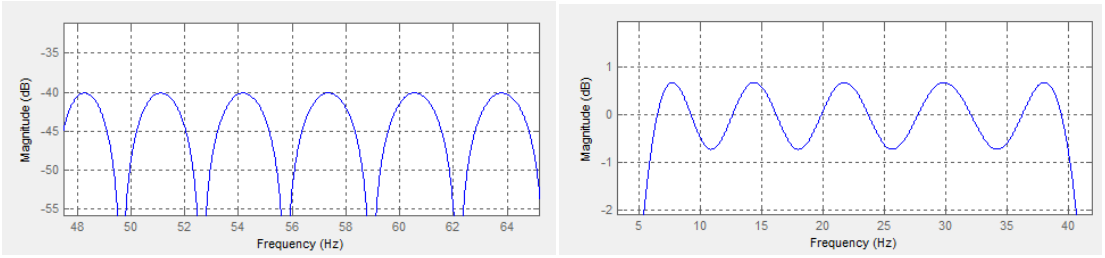
<u>Solution for part (a):</u> Equiripple (Remez) design

Parameters: f_s = 200 Hz, f_{stop1} = 1 Hz, f_{pass1} = 6 Hz, f_{pass2} = 40 Hz, f_{stop2} = 45Hz, A_{stop1} = A_{stop2} = 40dB, Apass = 1 dB. Using fdatool, we get an initial estimate of the order to be 57 to meet to the above specifications. Here is the initial look at this:



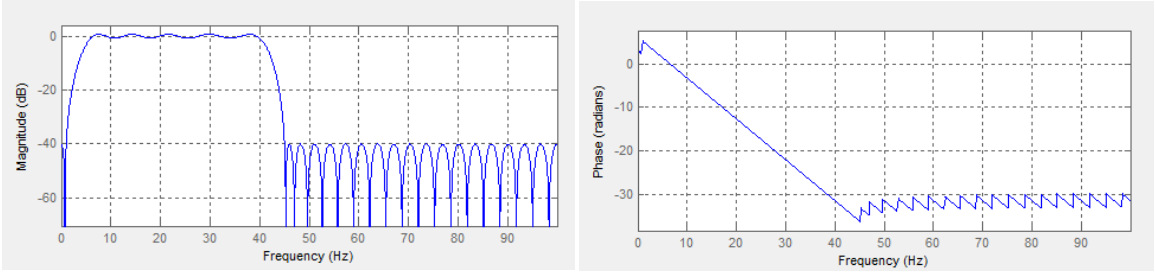
By zooming into stopband and passband, we see that the filter specification is not met. Hence, we would need to manually adjust the order until the filter meets the specifications. In fdatool, this is cumbersome. Instead, we adjust the filter parameters to make it design a filter with a tighter specification and possibly higher order. This way, the filter design will meet specification.

After some trial and error, we can use $A_{stop1} = A_{stop2} = 43$dB, Apass = 1dB. This created a filter of **order 60**. Here is the zoomed version of the stopband (left) and the passband (right).



Both stopband and passband responses shown above meet the original filter specifications.

The plots of magnitude (left) and phase (right) responses below:



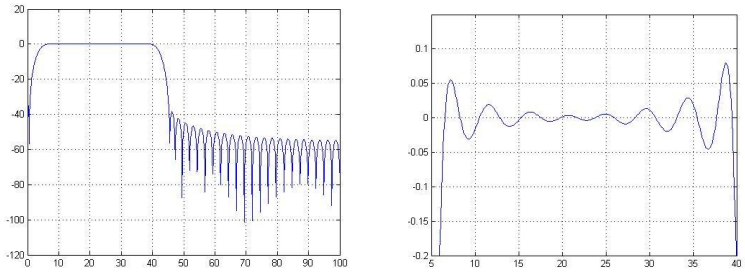
The least-squares method

This method has difficulty in meeting filter specifications at stopband frequencies. Using fdatool, which is not intuitive, a 72nd-order filter meets specifications when entering $f_s = 200$ Hz, $f_{stop1} = 1.5$ Hz, $f_{pass1} = 5.5$ Hz, $f_{pass2} = 40.5$ Hz, $f_{stop2} = 44.5$ Hz, Wstop1 = 15, Wpass = 1, Wstop2 = 15.

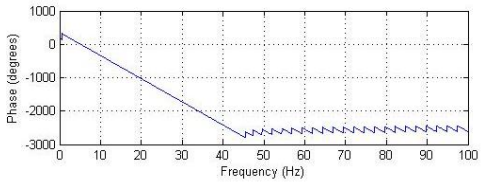
Alternately, we can use the Matlab command `firls` to search for the minimum order. The filter **order was 77**. Here is code to obtain filter coefficients and plot the magnitude response.

```
%HW2, Prob3, FIR design using least squares method, Spring 2014
clear all; close all; clc;
fn=200/2; % Nyquist frequency
h=firls(77,[0/fn 1/fn 6/fn 40/fn 45/fn 1],[0 0 1 1 0 0]);
[h1,f]=freqz(h,1,1024,fn*2);
figure; plot(f,20*log10(abs(h1)));
grid on;
```

Here is a plot of the magnitude response (left) and the zoomed-in version of the passband (right).

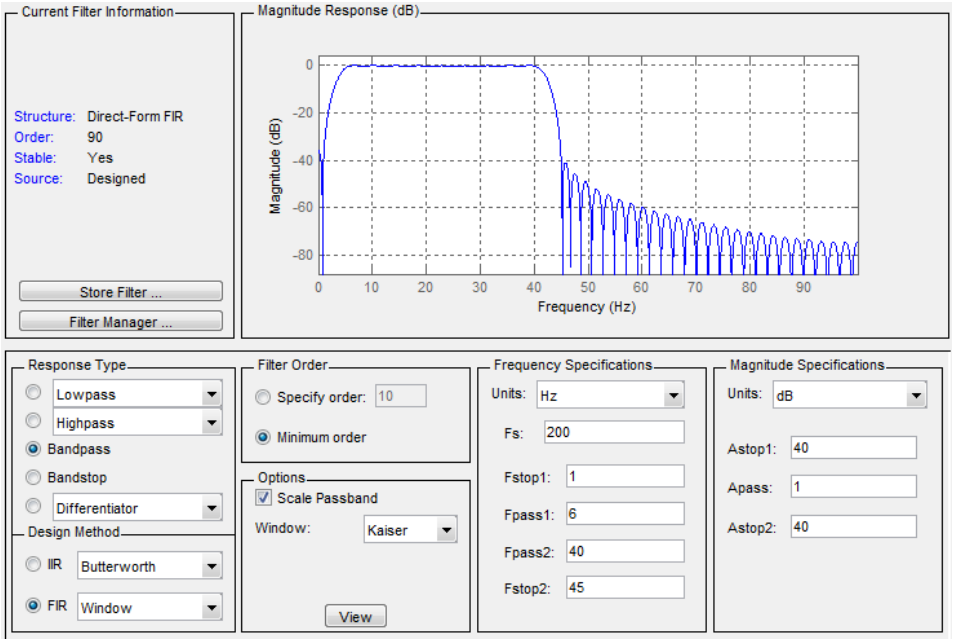


Magnitude response meets the original specifications. Here is a plot of the phase response:



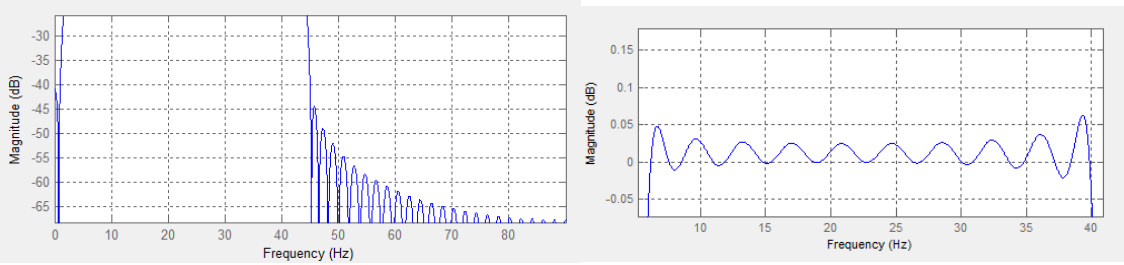
Kaiser window design

Parameters: f_s = 200 Hz, f_{stop1} = 1 Hz, f_{pass1} = 6 Hz, f_{pass2} = 40 Hz, f_{stop2}= 45Hz, A_{stop1} = A_{stop2} = 40dB, Apass = 1dB. Using fdatool, **order is 90** according to these specifications:



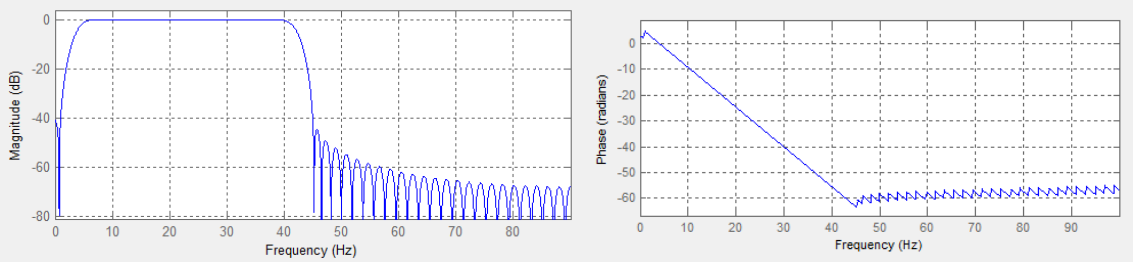
Looking at the stopband between 0 and 1 Hz, we can see that the filter does not meet the attenuation specification of 40 dB.

After some trial and error, A_{stop1} = 43 dB, A_{stop2} = 40 dB, Apass = 1 dB gave an **order of 98**.



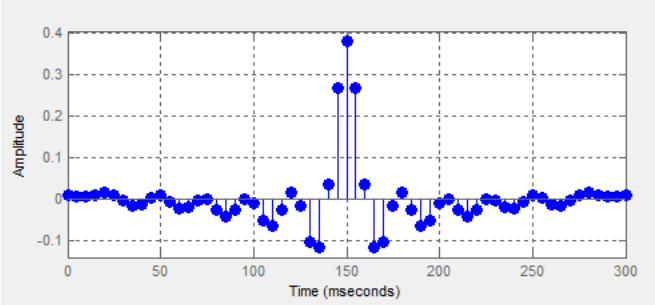
In the above plots, the stopband (left) and passband (right) responses are zoomed in. It can be seen that both the stopband and the passband meet the original specifications.

Here are the plots of magnitude (left) and phase (right) responses:



b) Plot the impulse response of the FIR filter designed by the Parks-McClellan (Remez) algorithm. What symmetry is in the impulse response?

Solution for (b): The 61 coefficients of the impulse response are plotted to the right, and the spacing between samples is the sampling period of 1/(200 Hz) or 5 ms. Impulse response has even symmetry about its mid-point. Hence, the phase response is linear, as we saw above.



c) Give the filter lengths required for filters designed for each filter design method. Which method gives the shortest filter length?

Solution for (c): The length of a filter is order of the filter plus one. This is because the order of a FIR filter is the highest negative power of *z*. Lengths of our filters are given in the table below:

Filter type	Order	Length
Remez	60	61
Least Squares	77	78
Kaiser window	98	99

For the three FIR filter designs, the coefficients are even symmetric about the midpoint and hence have linear phase. The Remez method gives the shortest FIR filter length. More generally, the Remez method gives the shortest linear phase FIR filter with real-valued coefficients.

d) Analyze the implementation complexity of each FIR filter design:
1) How many multiplication operations are needed?
2) How much memory (in words) would it take to store the FIR coefficients and the circular buffer for the current and past inputs?

Solution for (d): An FIR filter of N coefficients requires N multiplication and N -1 addition operations to compute each output sample. An FIR filter also requires a linear buffer of N words to store the impulse response (filter coefficients) and a circular buffer of N words to store the current input sample and the previous N-1 input samples. Total storage is 2N words. *Note: A "word" simply means the size of the data type needed to store a value. On the DSP board in lab, a "word" for a single-precision floating-point number (float data type in C) is 32 bits or 4 bytes.*

MATLAB Scripts from in Johnson, Sethares and Klein's *Software Receiver Design* textbook

The Matlab scripts should run "as is" in MATLAB or LabVIEW Mathscript facility.

1. Copy the .m files on your computer from the "SRD - MatlabFiles" folder on the CD ROM: http://users.ece.utexas.edu/~bevans/courses/rtdsp/homework/SRD-MatlabFiles.zip
2. Add the folder containing the .m files from the book to the search path.:
    - In MATLAB, use the `addpath` command
    - In LabVIEW, open the Mathscript window in LabVIEW by going to the Tools menu and select "Mathscript Window" (third entry), go the File menu, select "LabVIEW MathScript Properties" an d add the path.

Johnson, Sethares and Klein intentionally chose not to copyright their programs so as to enable their widespread dissemination.

Discussion of this solution set is available at http://www.youtube.com/watch?v=mnAFFNYnklU.