The University of Texas at Austin

Spring 2025 EE 445S Real-Time Digital Signal Processing Laboratory Prof. Evans

Solutions for Homework #2 on Filter Analysis, Simulation and Design

2.1 Frequency Responses

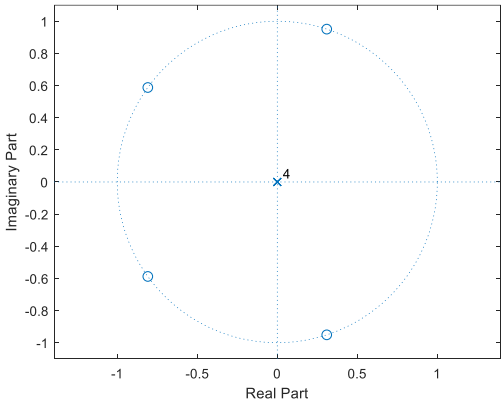
For each LTI system in problem 1.1 on homework assignment #1,

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.

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

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

a) Pole-zero plot: The zeroes occur at $e^{j2\pi\frac{m}{N}}$ where $m = 1 \ldots N\text{-}1$. In our case, the four zeros are 0.3090 + j0.9511, -0.8090 + j0.5878, and their complex conjugates. The four artificial poles are repeated.



b) BIBO Stability: This system is bounded-input bounded-output stable because the region of convergence (ROC) $z \neq 0$ includes the unit circle. FIR filters are always BIBO stable.

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

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

Discrete-time frequency domain is periodic in ω with period 2π due to the $e^{j\omega}$ terms.

Although not asked, we show that this LTI system has linear phase by first rewriting the frequency response $H(e^{j\omega})$ as $A(\omega)\, e^{j\theta}$ where $A(\omega)$ is an amplitude function. We factor out a phase shift corresponding to a delay equal to the midpoint of the impulse response:

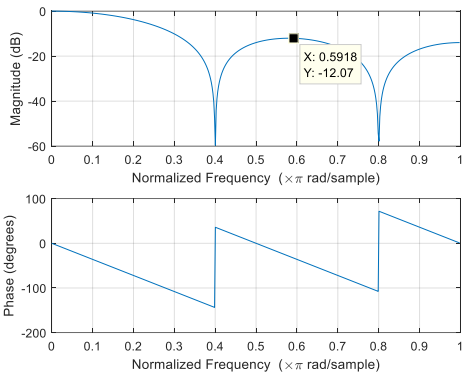
$$H(e^{j\omega}) = \frac{1}{5}(e^{2j\omega} + e^{j\omega} + 1 + e^{-j\omega} + e^{-2j\omega})\, e^{-2j\omega}$$

$$H(e^{j\omega}) = \underbrace{\frac{1}{5}(1 + 2\cos(\omega) + 2\cos(2\omega))}_{ampltude\ function\ A(\omega)} \underbrace{e^{-2j\omega}}_{phase\ is\ -2\omega}$$

Amplitude function is negative $2\pi/5 < |\omega| < 4\pi/5$ and positive otherwise $-\pi < \omega \leq \pi$. We use the fact that $-1 = e^{j\pi}$ to convert the expression into magnitude-phase form:

$$H(e^{j\omega}) = \begin{cases} -A(\omega)\, e^{j(-2\omega+\pi)} & for\ \frac{2\pi}{5} < |\omega| < \frac{4\pi}{5} \\ A(\omega)\, e^{-2j\omega} & otherwise \end{cases}$$

d) Magnitude response: The phase response is linear with constant slope of -2, except at discontinuities at the four nulls in the magnitude response (two in positive frequencies and two in negative frequencies). At nulls in magnitude response, a jump of π (or -π) occurs in phase response. Group delay is 2 samples.



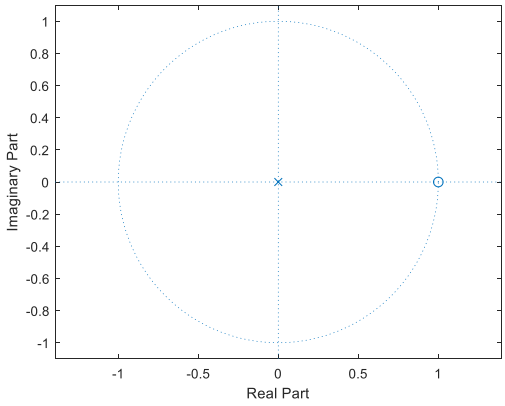
e) Lowpass filter. For the five-tap averaging filter, the first sidelobe peaks at -12 dB and the second side lobe peaks at -14 dB. As the averaging filter length N increases, the first sidelobe peaks decreases to only about -13.3 dB. Not a great lowpass filter, but lowpass nonetheless. For $N > 1$, the null bandwidth is $2\pi / N$, or 0.4π for $N = 5$. The continuous-time frequency f_0 corresponding to $2\pi / N$ can be found using $2\pi / N = 2\pi f_0 / f_s$, *i.e.* $f_0 = f_s / N$. This was used to choose lengths of the averaging filters in homework 1.3 on AM rado demodulation. Please see the handout on Designing Averaging Filters.

```
N = 5;                          % number of taps
H = (1/N)*ones(1,N);            % normalized
figure; zplane(H);              % pole-zero plot
roots(H)
figure; freqz(H);
```

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

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

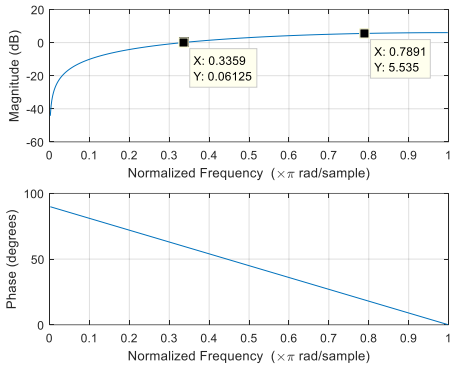
a) Pole-zero diagram: Transfer function has a zero at $z = 1$. An "artificial" pole occurs at $z = 0$. The zero on the unit circle will have a gain of zero at the 0 rad/sample in the magnitude response (see lecture slide 6-6). The zero location at $z = 1$ means that for $z = e^{j\omega} = 1$, the value of $\omega = 0$.



b) BIBO Stability: The system is bounded-input bounded-output stable because the region of convergence (ROC) $z \neq 0$ includes the unit circle. FIR filters are always BIBO stable.

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

d) Magnitude response: The gain above 0.3 rad/sample goes above 0 dB. The gain in the passband region is approximately 5.5 dB at 0.7 rad/sample. Additionally, the phase response is linear with constant slope of -½.

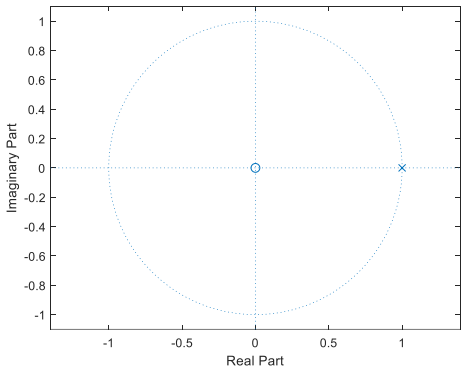


e) Highpass filter because it attenuates lower frequency components below 0.3 rad/sample. Notch filter because it notches out (eliminates) zero frequency. The in-class demonstration Cascading Two FIR Filters from *DSP First* of filtering the Mandrill (Baboon) image called it highpass.

```
close all;
figure; zplane([1 -1]);
figure; freqz([1 -1]);
```

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

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



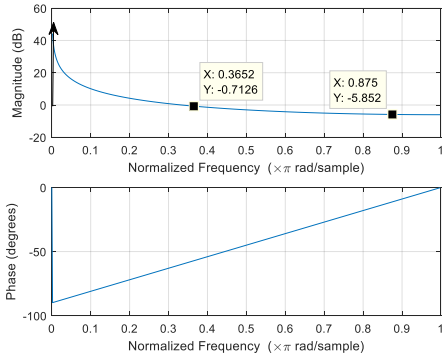
a) Pole-zero diagram: The pole is on the unit circle at $z = 1$ and the zero is at the origin.

b) BIBO Stability: Not BIBO stable, because the pole at $z = 1$ is not inside the unit circle for the causal system. Another reason is the ROC $|z| > 1$ does not include the unit circle.

c) Frequency response: Since the ROC does not include the unit circle, substituting $z = e^{j\omega}$ in the z-transform expression does not hold. Instead, the frequency response can be calculated by finding the impulse response of the system and taking its discrete-time Fourier transform. The impulse response is $h[n] = u[n]$; i.e., for an input of a discrete-time impulse $\delta[n]$, the output is $h[n] = h[n-1] + \delta[n]$ with $h[-1] = 0$. The frequency response is the discrete-time Fourier transform of the unit step:

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

Discrete-time frequency domain is periodic with period 2π.



d) Magnitude response:

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

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

```
close all;
figure; zplane(1,[1 -1]);
figure; freqz(1,[1 -1]);
```

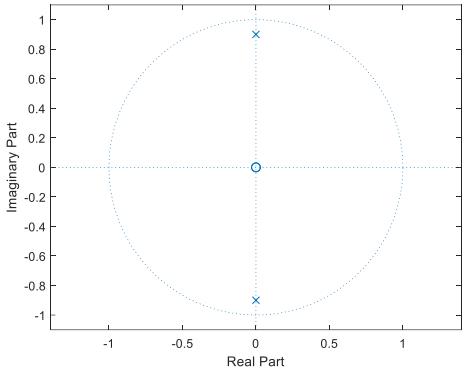
We approximate the magnitude response using the inequality in the second expression. We have manually added the right term, which is a Dirac delta at $\omega = 0$ in $[0, \pi]$.

e) Type of filter: Not applicable because system is not BIBO stable. Magnitude response becomes unbounded as frequency approaches zero. Otherwise the magnitude response resembles a lowpass filter. If the DC component were removed, e.g. by a notch filter, the LTI system could be used to filter other frequencies.

(4) Causal bandpass filter with center frequency ω_0

This is an example of a biquad (i.e. a transfer function of a ratio of two quadratics). 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 $\pi/4$ and $3\pi/4$. We'll use $r = 0.9$ and $\omega_0 = \pi/2$:

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

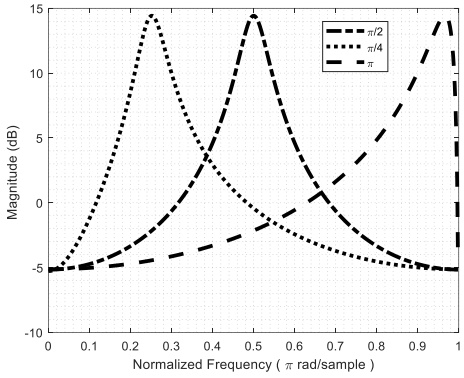


a) Pole-zero diagram: Zero at $\cos(\omega_0)$. Poles at $re^{j\omega_0}$ and its complex conjugate $re^{-j\omega_0}$.

b) BIBO Stability: Since the poles are inside unit circle, i.e. $|r| < 1$, system is BIBO stable.

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

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



d) Plot of magnitude response for $r = 0.9$ for multiple values of $\omega_0 = \frac{\pi}{2}$, $\omega_0 = \frac{\pi}{4}$, $\omega_0 = \pi$.

e) Type of filter: Bandpass $0.8 < r < 1$. Zero at $\cos(\omega_0)$ helps create the bandpass response at $\omega_0 = 0$ and $\omega_0 = \pi$. Without the zero, the response would have been a lowpass and highpass filter, respectively.

```
close all; r = 0.9;
figure;
w0 = pi/2; freqz([1 -cos(w0)],[1 -2*r*cos(w0) r^2]);
[h1,z1] = freqz([1 -cos(w0)],[1 -2*r*cos(w0) r^2]);
w0 = pi/4; [h2,z2] = freqz([1 -cos(w0)],[1 -2*r*cos(w0) r^2]);
w0 = pi; [h3,z3] = freqz([1 -cos(w0)],[1 -2*r*cos(w0) r^2]);
figure; plot(z1/pi, 20*log10(abs(h1)),'k-.', z2/pi, 20*log10(abs(h2)),
'k:', z3/pi, 20*log10(abs(h3)), 'k--', 'LineWidth', 3 );
xlabel( 'Normalized Frequency ( \pi rad/sample )' );
ylabel( 'Magnitude (dB) ');
legend('\pi/2','\pi/4','\pi'); grid minor;
```

2.2. Finite Impulse Response Filter Design for Audio Signals. 30 points.

This problem explores ways to process audio signals. Please read homework hints.

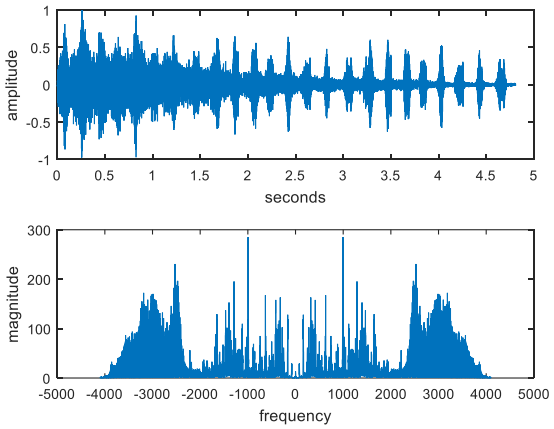
Please download the audio wave file 'twosignals.wav' from the homework Web site.

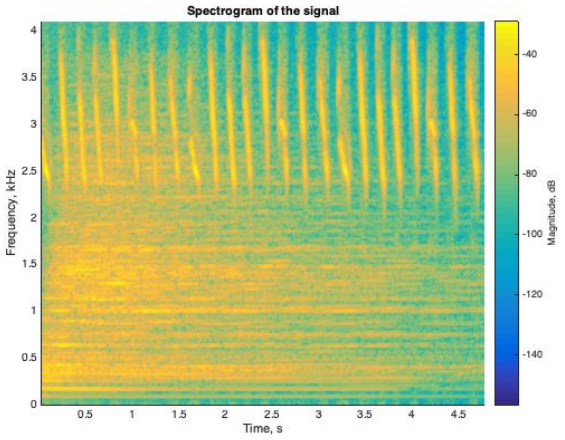
This audio file is the sum of two audio signals– a gong sound and a bird chirping. The gong sound and the bird chirping occupy different frequency bands. The gong sound is different from the gong file from the Johnson, Sethares and Klein book.

(a) Plot the spectrum of the 'twosignals' audio track using plotspec and spectrogram. Approximately what frequency band does the gong sound occupy? Approximately what frequency band does the bird chirp occupy?

```
% read in signal that is a combination of a gong and birds chirping
[twosignals, fs] = audioread('twosignals.wav');
% time and frequency plot
figure; plotspec(twosignals,1/fs);

nwin = 512;             % divide signal into blocks of nwin samples
noverlap = 511;         % number of samples in each block of signal
                        % that overlaps with the previous block
nfft = 512;             % specifies the number of frequency points used to
                        % calculate the discrete Fourier transforms.
figure; spectrogram( twosignals, nwin, noverlap, nfft, fs, 'yaxis');
h = colorbar;           % set the colorbar(dB) in y axis
ylabel(h, 'Magnitude, dB'); ylabel('Frequency, kHz');
xlabel('Time, s'); title('Spectrogram of the signal');
```





The left plot shows a signal with frequencies from 0 to ~2 kHz, and the other from ~2 to ~4 kHz, but it is unclear how much overlap there is. The spectrogram reveals the overlap. The spectrogram shows horizontal lines (principal frequencies and their harmonics) over 0 to 3 kHz for the gong sound and repeated down chirps for the birds chirping over 2 to 4 kHz. I tried several parameter choices for the spectrogram to get the right combination to see the down chirps. We'll arbitrarily assume the gong sound is in the 0 to 1.5 kHz range and bird chirps are in the 2 to 4 kHz range.

(b) Design an FIR filter using the Parks-McClellan algorithm (a.k.a. Remez Exchange algorithm and Equiripple design algorithm) to extract the gong signal from the 'twosignals' audio track. Then, apply the filter to the 'twosignals' audio track, play back the filter output to validate that the gong signal has been extracted, and plot the filter output using plotspec.

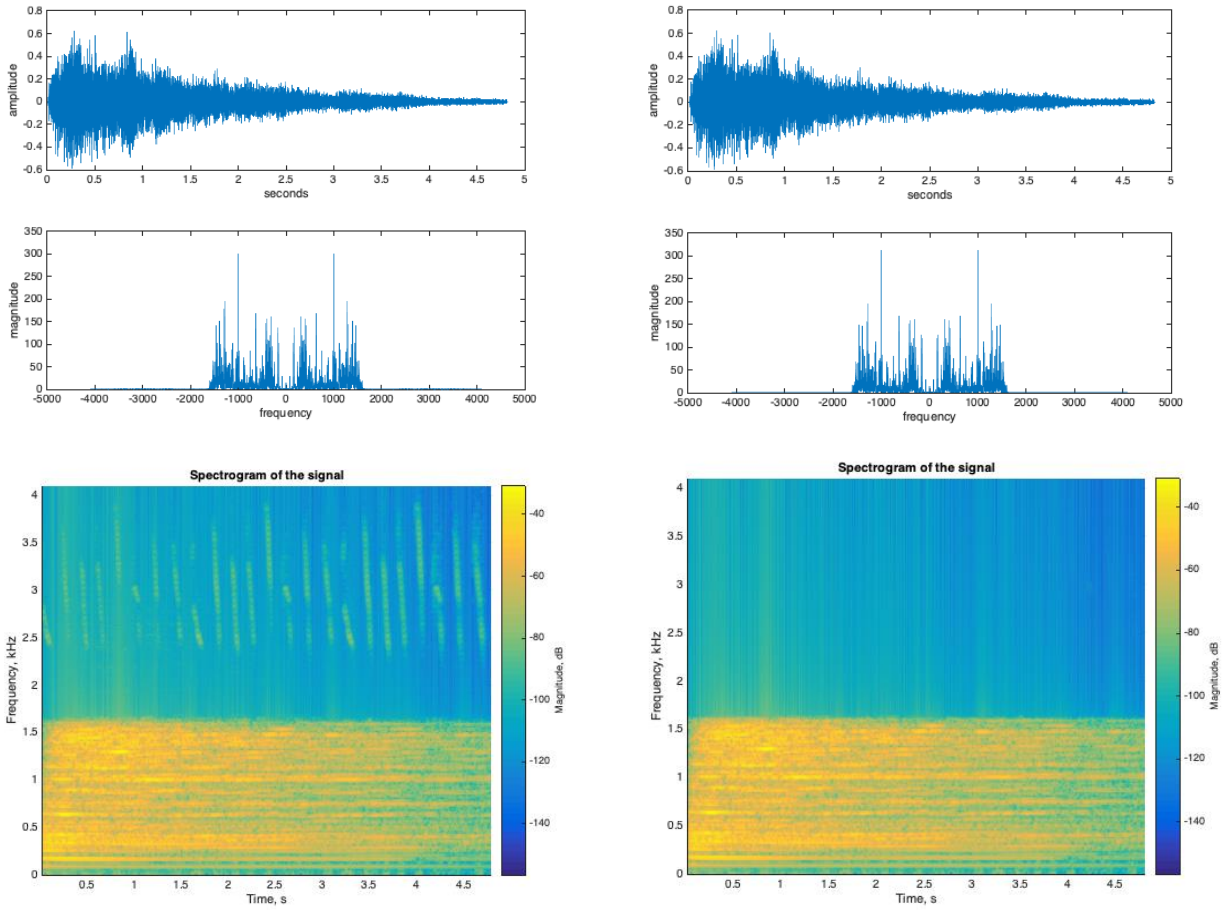
```
%% Set the Nyquist frequency to be half of the sampling rate fs.
fnyquist = fs/2;
% Define the passband frequency fpass in Hz
fpass = 1500;
% Define the stopband frequency fstop in Hz
fstop = 1500+150;
ctfrequencies = [0 fpass fstop fnyquist];
idealAmplitudes = [1 1 0 0 ];
pmfrequencies = ctfrequencies / fnyquist;
% Number of coefficients is filter order plus one
filterOrder = 130;
lowpass_flt = firpm( filterOrder, pmfrequencies, idealAmplitudes );
figure; freqz(lowpass_flt);
twosignals_lp = conv(lowpass_flt,twosignals');
sound(twosignals_lp,fs);
figure; plotspec(twosignals_lp,1/fs);
figure; spectrogram(twosignals_lp, nwin, noverlap, nfft, fs, 'yaxis');
h = colorbar; % set the colorbar(dB) in y axis
ylabel(h, 'Magnitude, dB'); ylabel('Frequency, kHz');
xlabel('Time, s'); title('Spectrogram of the signal');
```

(c) Design an FIR filter using the Parks-McClellan algorithm (a.k.a. Remez Exchange algorithm and Equiripple design algorithm) to extract the bird chirp from the 'twosignals' audio track. Then, apply the filter to the 'twosignals' audio track, play back the filter output to validate that the bird chirp signal has been extracted, and plot the filter output using plotspec.

```
% Sampling rate fs of sound card set when reading twosignals file
fnyquist = fs/2;
% Define passband and stopband frequencies in Hz
fpass = 2000;
fstop = 2000-200;
ctfrequencies = [0 fstop fpass fnyquist];
idealAmplitudes = [0 0 1 1 ];
pmfrequencies = ctfrequencies / fnyquist;
% Number of coefficients is filter order plus one
filterOrder = 114;
highpass_flt = firpm( filterOrder, pmfrequencies, idealAmplitudes );
figure; freqz(highpass_flt);
twosignals_hp = conv(highpass_flt,twosignals');
sound(twosignals_hp, fs);
figure; plotspec(twosignals_hp, 1/fs);
figure; spectrogram(twosignals_hp, nwin, noverlap, nfft, fs, 'yaxis');
h = colorbar; % set the colorbar(dB) in y axis
ylabel(h, 'Magnitude, dB'); ylabel('Frequency, kHz');
xlabel('Time, s'); title('Spectrogram of the signal');
```

·

Lowpass FIR Filter of order 130	Lowpass FIR Filter of order 260



From the spectrogram, we can eyeball that the bird chirps (mid-green) are about 40 dB weaker than those in the original spectrogram on the previous page (yellow). The gong sound is from someone striking a gong with a mallet, and its sound naturally dies out with time. At the end of the filtered signal, as the gong song naturally dies out, one can hear the birds chirping. If we double the filter order to 260, I could not hear the bird chirps at the end of the clip. One can see the dramatic reduction in the strength in the bird chirps in the spectrogram on the right.

For the spectrograms, we used blockSize = 512, overlap = 511, and FFTsize = 512.

(d) Take the extracted gong signal in part (b) and perform downsampling by 2. Downsampling by 2 keeps every other sample and discards the rest. Here's Matlab code for downsampling vector vec by 2:

```
vecDownsampledBy2 = vec(1:2:length(vec));
```

- Play the downsampled filtered gong signal at the same playback rate as the filtered gong signal. How does it differ from the gong signal extracted in part (b)?
- Plot magnitude spectrum of downsampled filtered gong signal and compare it against the magnitude spectrum of the gong extracted in part (b).

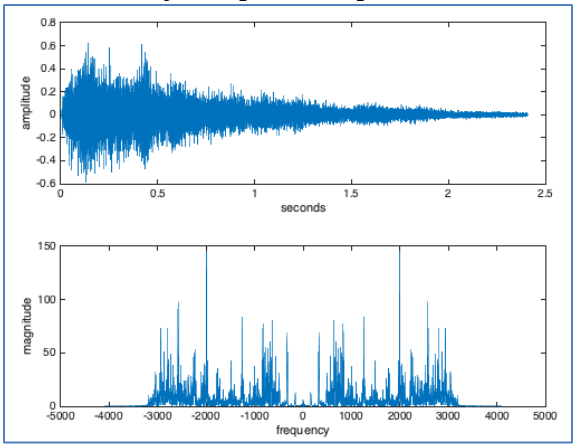
Solution: When the downsampled gong signal is played back at the same rate as the gong signal, the frequency components in the downsampled gong signal are twice those of the gong signal. The principal frequency is at 1940 Hz instead of 970 Hz. The sound clip lasts half as long. We can look at a single cosine signal $x[n]$ at frequency $\omega_0 = 2\pi f_0 / f_s$ in order to see what is happening:

$x[n] = \cos(\omega_0 n)$

$y[n] = x[2n] = \cos(\omega_0 (2n)) = \cos(2\pi (f_0 / f_s) (2n)) = \cos(2\pi ((2f_0) / f_s) n)$

If we play back $y[n]$ at the same sampling rate as that of $x[n]$, frequency components in $x[n]$ are doubled in $y[n]$. That also means the frequencies in $x[n]$ from $\pi/2$ to π will be aliased in $y[n]$.

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 ½ f_s Hz. Accordingly, the frequency of each tone in the signal will double, and the spectrum will stretch in width by a factor of 2.



We plot the downsampled gong sound extracted from the twosignals waveform using a 130th-order FIR filter.

```
reconstrGongDownsampledBy2 =
twosignals_lp(1:2:length(twosignals_lp));
sound(reconstrGongDownsampledBy2,fs);
figure;
plotspec(reconstrGongDownsampledBy2,1/fs);
```

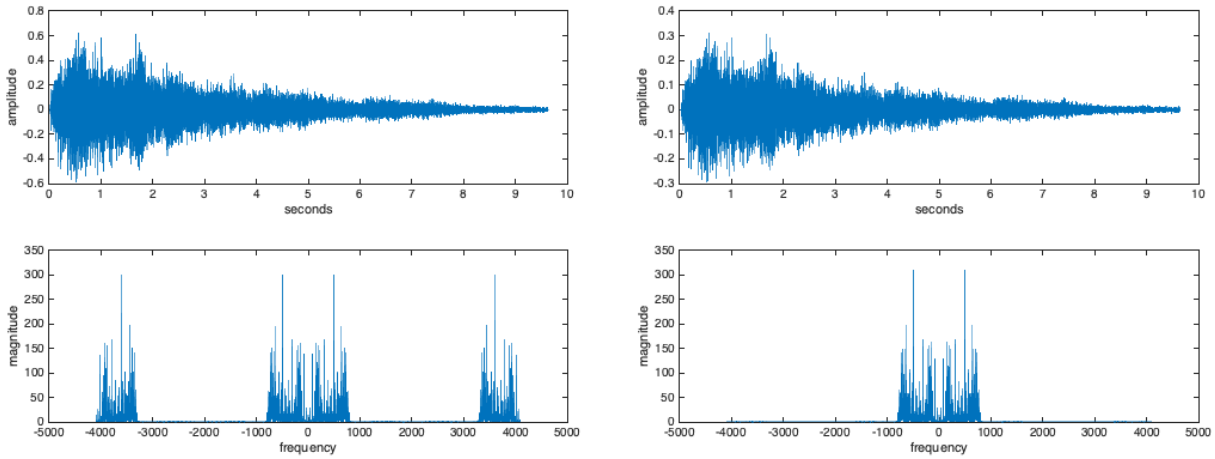
(e) Take the extracted gong signal in part (b) and perform upsampling by 2. Upsampling by 2 inserts zero after every sample. Here's Matlab code for upampling row vector vec by 2:

```
vec = cumsum( ones(1,10) );
upsampledLength = 2*length(vec);
vecUpsampledBy2 = zeros(1,upsampledLength);
vecUpsampledBy2(1:2:upsampledLength) = vec;
```

- Play the upsampled filtered gong signal at the same playback rate as the filtered gong signal *and also at twice the playback rate*. How does it differ from the gong signal extracted in part (b)?
- Plot the magnitude spectrum of the upsampled filtered gong signal and compare it against the magnitude spectrum of the gong extracted in part (b).

For the sanity of others, you might put in a pair of headphones when working this problem.

```
upsampledLength = 2*length(twosignals_lp);
reconstrGongUpsampledBy2 = zeros(1,upsampledLength);
reconstrGongUpsampledBy2(1:2:upsampledLength) = twosignals_lp;
sound(reconstrGongUpsampledBy2,fs);
figure; plotspec(reconstrGongUpsampledBy2,1/fs);
% lowpass filtering removes aliasing created by upsampling
reconstrGongUpsampledBy2_lp = conv(lowpass_flt,reconstrGongUpsampledBy2);
sound(reconstrGongUpsampledBy2_lp,fs);
figure; plotspec(reconstrGongUpsampledBy2_lp,1/fs);
```



Upsampling is a sampling operation, which causes replicas of the spectrum of the signal being sampled. When playing the upsampled gong signal at the same playback rate as the gong signal, the principal frequency at 970 Hz is halved. Playing back the left waveform at the same playback rate gives the gong signal at halved frequencies plus a high frequency ringing (which correspond to replicas of the gong signal at halved frequencies shifted by 4000 Hz to the left and right).

Epilogue: Parts (b) and (c) used LTI filters to extract the two signals by extracting the primary frequency band for each. However, some of the other signal was also included. Separating additive signals is called Source Separation. The separation of the two signals has better results when additional information is known about at least one signal, e.g. its statistical distribution. An application is to separate each instrument and the singer in a recording of a band performing.

2.3 Finite Impulse Response (FIR) Filter Design for Treatment of Tinnitus Loudness.

Tinnitus, a.k.a. “ringing of the ears”, is a symptom due to an underlying condition in the auditory system. It could have resulted from injury, infection, or other causes. People with tinnitus hear a tone, clicking, hiss, roaring or buzzing when no external sound is present [1][2]. The tinnitus sound could be at low, medium or high audible frequencies, and may occur in one ear or both ears. The tinnitus sound might be temporary or chronic. Those suffering from chronic tinnitus would hear the same sound in the same frequency range each time. The tinnitus sound has a principal frequency that can be determined through auditory testing. Hearing sound that contains the principal frequency and frequencies close to the principal frequency is particularly painful.

This problem asks you to design a discrete-time filter to alleviate the loudness of tinnitus:

> "Maladaptive auditory cortex reorganization may contribute to the generation and maintenance of tinnitus. Because cortical organization can be modified by behavioral training, we attempted to reduce tinnitus loudness by exposing chronic tinnitus patients to self-chosen, enjoyable music, which was modified (“notched”) to contain no energy in the frequency range surrounding the individual tinnitus frequency. After 12 months of regular listening, the target patient group (n = 8) showed significantly reduced subjective tinnitus loudness and

concomitantly exhibited reduced evoked activity in auditory cortex areas corresponding to the tinnitus frequency compared to patients who had received an analogous placebo notched music treatment (n = 8). These findings indicate that tinnitus loudness can be significantly diminished by an enjoyable, low-cost, custom-tailored notched music treatment, potentially via reversing maladaptive auditory cortex reorganization." [3]

The proposed treatment for tinnitus [3] alters participants' favorite music to remove an octave of frequencies around the tinnitus frequency f_c. An octave means a range of frequencies from f_1 to 2 f_1. Since f_c would be in the middle of the octave, $f_1 = (2/3)\, f_c$. After 12 months of listening to the filtered music, patients reported lessening of tinnitus loudness.

A good rule of thumb in filter design is that the transition region is about 10% of the passband width. In this case, the passband width is (2/3) f_c.

Here are the bandstop filter specifications for your design:

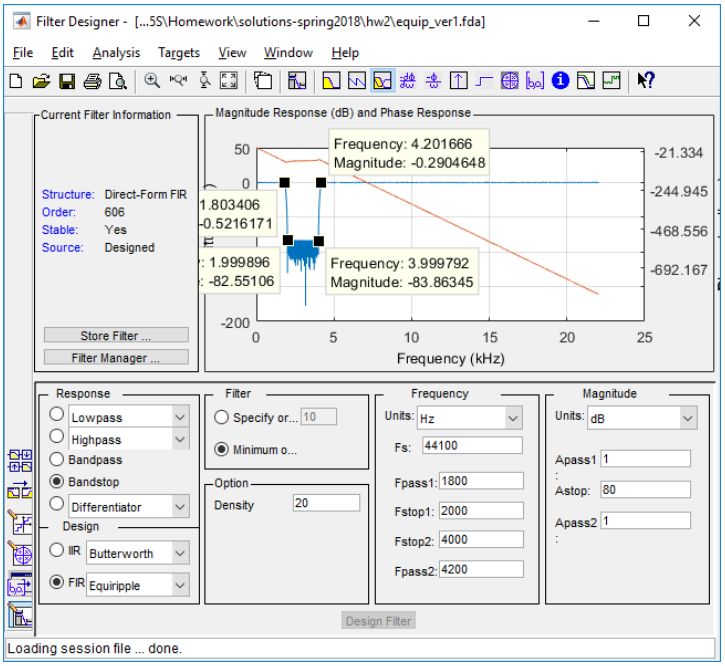
- For frequencies 0 Hz to 0.6 f_c, the passband ripple should be no greater than 1 dB.
- For frequencies (2/3) f_c to (4/3) f_c, the stopband attenuation should be at least 80 dB.
- For frequencies above 1.4 f_c, the passband ripple should be no greater than 1 dB

Please use a tinnitus frequency f_c of 3000 Hz and a sampling rate f_s of 44100 Hz.

(a) Design FIR filters *with the minimum filter order* to meet the specification by using the Equiripple, Least Squares, and Kaiser Window design methods. FIR equiripple design is also known by many other names: Parks-McClellan, Remez Exchange and Chebyshev Design. Please submit a plot of the magnitude and phase response for each filter design. Validate that each filter design meets the filter specifications.

<u>Solution:</u> Equiripple (Remez) design

Parameters: f_s = 44100 Hz, f_{pass1} = 1800 Hz, f_{stop1} = 2000 Hz, f_{stop2} = 4000 Hz, f_{pass2} = 4200 Hz, $A_{pass1} = A_{pass2}$ = 1 dB, Astop = 80 dB. Using filterDesigner, called fdatool in earlier Matlab versions, we obtain an initial estimate of the order of 606 to meet to the above specifications. Here is the initial look at this and by zooming into stopband and passband, we see the filter specifications are met. **Please see the note on next page on phase response**.



However, the filter order is high. We can manually adjust the filter specification values input into fdatool so that the filter would have a lower order for the minimum order design. By changing from "Minimum Order" to

"Specify Order" and by choosing weights Wpass1 = 1, Wstop = 550, and Wpass2 = 1, the filter order reduces to **564**, while still meeting the original specifications. We can check this in filterDesgner (fdatool) by zooming into the stopband, or by exporting the filter design to the Matlab workspace as variable Num and using `freqz(Num)` and zooming into the stopband.

Solution: Least Squares design

This method has difficulty in meeting filter specifications at stopband frequencies. Below, Wstop is the weighting of importance in meeting the stopband specification.

Solution I: Using fdatool, a **1100th-order filter** meets specs using f_s = 44100 Hz, f_{pass1} = 1800 Hz, f_{stop1} = 2000 Hz, f_{stop2} = 4000 Hz, f_{pass2} = 4200 Hz, Wpass1 = 1, Wstop = 100, Wpass2 = 1.

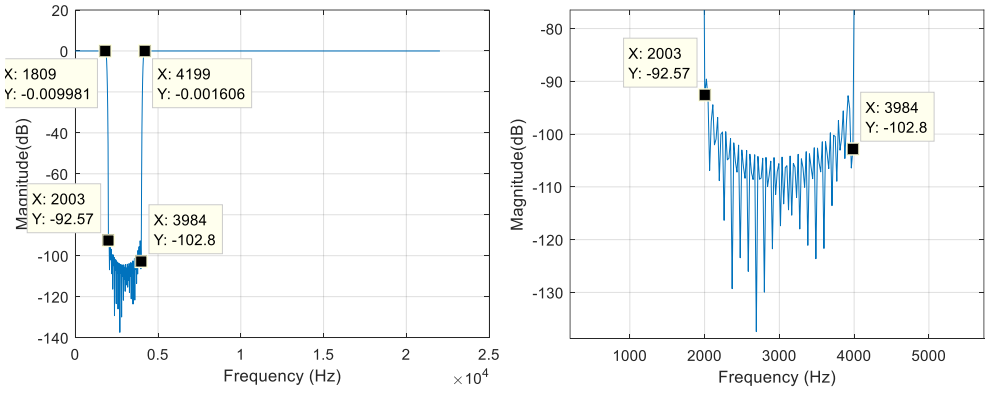
Using fdatool, a **960th-order filter** meets specs using f_s = 44100 Hz, f_{pass1} = 1800 Hz, f_{stop1} = 1980 Hz, f_{stop2} = 4020 Hz, f_{pass2} = 4200 Hz, Wpass1 = 1, Wstop = 200, Wpass2 = 1.

Using fdatool, a **778th-order filter** meets specs using f_s = 44100 Hz, f_{pass1} = 1800 Hz, f_{stop1} = 1978 Hz, f_{stop2} = 4022 Hz, f_{pass2} = 4200 Hz, Wpass1 = 1, Wstop = 10000, Wpass2 = 1.

Solution II: We can use the Matlab command `firls` to search for the minimum order. The filter **order was 1200**. Here is code to obtain filter coefficients and plot the magnitude response.

```
%HW2, Prob3, FIR design using least squares method
clear all; close all; clc;
fn=44100/2; % Nyquist frequency
h=firls(1200,[0/fn 1800/fn 2000/fn 4000/fn 4200/fn 1],[1 1 0 0 1 1]);
[h1,f]=freqz(h,1,1024,fn*2);
figure(1); plot(f,20*log10(abs(h1)));grid on;
ylabel('Magnitude(dB)'); xlabel('Frequency (Hz)')
```

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



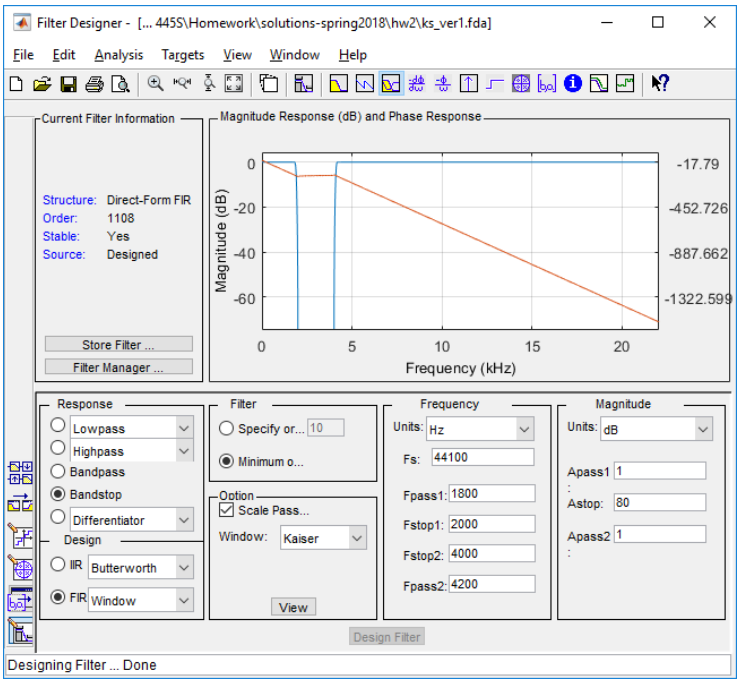
Note on Phase Response Plots: Parks-McClellan, Least Squares, and Kaiser window design methods give linear phase FIR filters. Phase should be a line of constant slope, except for jumps by π rad/sample at frequencies zeroed out in the magnitude response.

```
% h is vector of FIR filter coeffs
N = length(h);
w = -pi : ((2*pi)/(N*1000)) : pi;
Hfreq = zeros(1, length(w));;
for n = 1:N
  Hfreq = Hfreq + h(n)*exp(-j*(n-1)*w);
end
figure; plot(w, angle(Hfreq));
title('Phase');
```

fdatool and freqz unwraps the phase response to try to make it continuous, which can remove many jumps. For part (a), the phase response is flat in the stopband and a line of constant slope in the passbands without jumps. The above code will give an accurate phase plot.

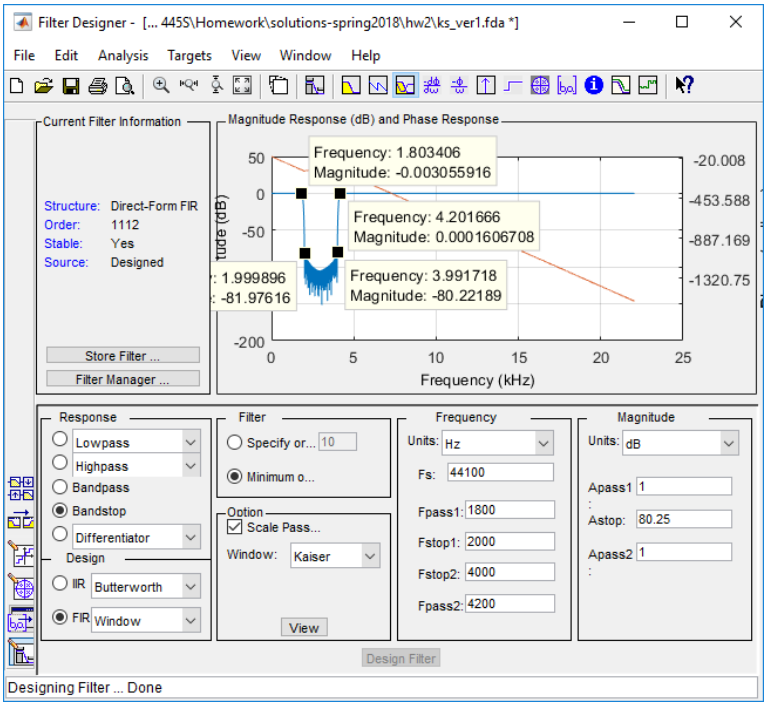
<u>Solution:</u> Kaiser window design →

<u>First Try:</u> Parameters: $f_s = 44100$ Hz, $f_{pass1} = 1800$ Hz, $f_{stop1} = 2000$ Hz, $f_{stop2} = 4000$ Hz, $f_{pass2} = 4200$ Hz, $A_{pass1} = A_{pass2} = 1$ dB, Astop = 80 dB. Using fdatool, order is 1108 based on a minimum order design:



Please see the note on previous page on phase response.

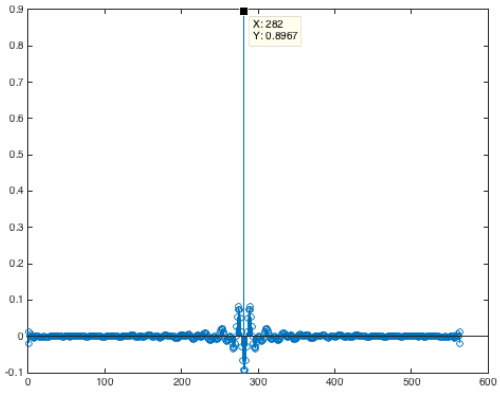
<u>Second Try</u>: By changing Astop = 80.25 dB, filter specifications are met, and the **order is 1122**. **Please see the note on previous page on phase response.**

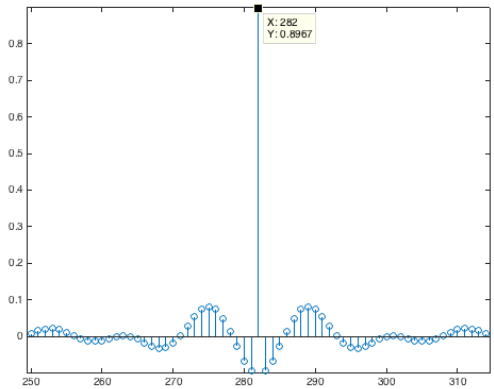


<u>Third Try:</u> The order can also be manually specified. With Fc1 = 1900, Fc2= 4100 and beta = 0.5, an order greater than 250000 will be needed to make the stopband attenuation reach 80 dB.

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

Solution: Impulse response is plotted on left. Spacing between samples is the sampling period of 1/(44100 Hz) or 0.0227 ms. Impulse response extends for 565 samples, which corresponds to 565 samples / (44100 samples/s) = 12.81 ms. Impulse response has even symmetry about its midpoint, which gives linear phase. Although not asked, group delay is 282 samples or 6.395 ms. The impulse response is zoomed into its midpoint at sample index 282 is shown on right.





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

Solution: The filter length is the filter order plus one. This is because the filter order is the highest negative power of *z*. All filter designs have linear phase. The Remez method gives the shortest linear phase FIR filter with real-valued coefficients.

Filter type	Order	Length
Remez	564	565
Least Squares	778	779
Kaiser window	1122	1123

(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: An FIR filter of *N* coefficients takes *N* multiplications and *N*-1 additions to compute an output sample. It 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 and previous *N*-1 inputs. Total storage is 2*N* words. *"Word" is an abstraction of the actual data type used (float, int, etc.).*

References

[1] R. A. Levine and Y. Oron, "Tinnitus", *Handbook of Clinical Neurology*, vol. 129, pp. 409–431, 2015. *doi*:*10.1016/B978-0-444-62630-1.00023-8*.
[2] *"Tinnitus"*. December 16, 2016. Retrieved February 17, 2017.
[3] H. Okamoto, H. Stracke, W. Stoll and C. Pantev, "Listening to tailor-made notched music reduces tinnitus loudness and tinnitus-related auditory cortex activity", *Proceedings US National Academy of Sciences*, vol. 17, no. 3, pp. 1207-1210, 2010.