Music 421A
Winter 2011-2012
Homework #4
FIR Filter Design, Resolving Spectral Peaks
Due in one week

Theory Problems

1. (5 pts) Derive the ideal impulse response corresponding to the desired amplitude response
$$H(e^{j\omega T}) = \begin{cases} 1, & |\omega| \le \omega_c \\ 0, & \omega_c \le |\omega| \le \pi/T \end{cases}.$$

2. (5 pts) Derive the ideal impulse response corresponding to the desired amplitude response
$$H(e^{j\omega T}) = \begin{cases} 1, & 0 \le \omega_1 \le |\omega| \le \omega_2 \le \pi/T \\ 0, & \text{otherwise} \end{cases}$$
[Hint: Use a Fourier theorem to make use of the answer for the ideal lowpass filter.]

Solution: An easy way to do this is to notice that the desired bandpass response can be expressed as the difference between the responses of two lowpass filters with cutoff frequencies ω_2 and ω_1, respectively. Since IDTFT is linear, we can use the result of problem 1 and show, by inspection,
$$h(n) = \frac{1}{\pi n}[\sin(\omega_2 T n) - \sin(\omega_1 T n)]$$
An alternative way, as suggested by the hint, is to think of the frequency response of the bandpass filter as the sum of a lowpass response shifted to $\pm\frac{\omega_2+\omega_1}{2}$. The cutoff frequency of that lowpass response should be $\frac{\omega_2-\omega_1}{2}$. Then, using the fact that a shift in frequency domain is a modulation in time domain, the same result can be obtained, as shown below.
$$\begin{aligned} h(n) &= \frac{2}{\pi n}\sin\left[\left(\frac{\omega_2-\omega_1}{2}\right)nT\right]\cos\left[\left(\frac{\omega_1+\omega_2}{2}\right)nT\right] \\ &= \frac{1}{\pi n}[\sin(\omega_2 T n) - \sin(\omega_1 T n)] \end{aligned}$$

3. (5 pts) When designing an FIR bandpass filter, explain the benefit of choosing the lower cut-off frequency f_1 to be equal to the difference between the Nyquist limit

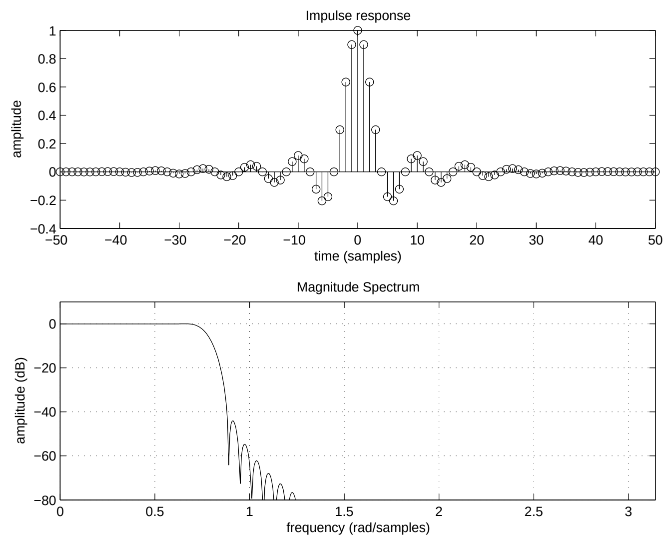

Figure 1: Impulse response of a lowpass filter and its magnitude spectrum

$f_s/2$ and the upper cut-off frequency f_2. In other words, what is the benefit of the constraint $f_1 = f_s/2 - f_2$? (Prove that the claimed benefit is obtained in general for any $f_1 \in (0, f_s/4)$.)

Solution: Forces every other sample of the impulse response to be zero. Can be proved using the stretch theorem.

4. (5 pts) Figure 1 shows the impulse response and the corresponding magnitude spectrum of a lowpass filter.

   (a) (3 pts) Without using MATLAB, sketch the magnitude spectrum of the impulse response shown in Figure 2, which was obtained from that in Fig. 1 by negating the odd-numbered samples. Explain how you obtained your answer.

   (b) (2 pts) Verify your answer by plotting the magnitude spectrum of the above impulse response using MATLAB. You can download a `.mat` file `ir.mat`[1] which contains an impulse response vector $h(n)$ shown in Fig. 2.

Solution:

   (a) (3 pts) Negating odd-numbered samples of $h(n)$ is equivalent to multiplying $h(n)$ by $(-1)^n = e^{j\pi n}$; *i.e.*, it has the same effect of shifting the spectrum $H(\omega)$ by π in the frequency domain. Therefore, the resulting spectrum will be that of a highpass filter with the cutoff frequency $\omega_c = \pi - \pi/4 = 3\pi/4$.

   (b) (2 pts) Figure 3 shows the spectrum of the impulse response $h_{hp}(n) = (-1)^n h(n)$.

[1] `https://ccrma.stanford.edu/~jos/sasp/hw/ir.mat`

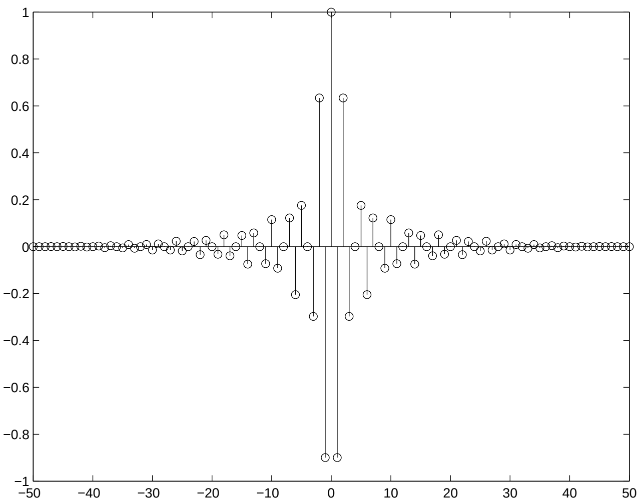

Figure 2: Impulse response

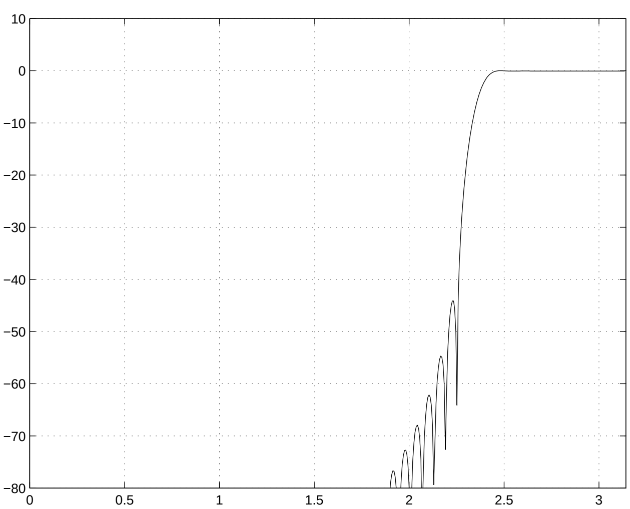

Figure 3: Magnitude spectrum of a highpass filter

5. (5 pts) What length Blackman window is required to resolve a sinusoid at 100 Hz and another one at 101 Hz? State your definition of resolution in this context, and draw a sketch showing the two sinusoids and the window transform in the frequency domain, with the window transform being centered on one of the sinusoidal frequencies.

   Solution:

   In order to resolve two sinusoids, the bandwidth of the window main lobe should be less than the frequency difference between them. That is,

   $$B_w \leq |\omega_2 - \omega_1|$$

   where $B_w = 6\frac{2\pi}{M}$ for the Blackman window. In Hz unit,

   $$6\frac{f_s}{M} \leq |f_2 - f_1| = 1$$

   Therefore,

   $$M \geq 6f_s$$

Lab Assignments

1. (5 pts) Design a real, linear-phase, FIR bandpass filter using `firpm()` in Matlab with the following specifications: Sampling rate $f_s = 100$ Hz, pass-band from 20 Hz to 30 Hz, stop-band from 0 to 10 Hz and 40 to 50 Hz, $\delta_s = 0.01$ (-40 dB) ripple in the stop-band, and $\delta_p = 0.02$ ripple in the pass-band, which is unity gain. The filter thus has transition bands from 10 to 20 Hz, and from 30 to 40 Hz. Turn in a listing of your Matlab code, and the result of its execution (e.g., using the `diary` command), which should include a print-out of the filter length, a listing of the filter coefficients, and a plot of the filter amplitude response on a dB vertical scale. [Hint: Start with '`help firpmord`' in Matlab.]

   Solution:

```
>> diary 'firpmproblem.txt'
>> [N,Fo,Ao,W] = firpmord([10 20 30 40],[0 1 0],[.01 .02 .01],100)

N =
    17

Fo =
         0
    0.2000
```

```
    0.4000
    0.6000
    0.8000
    1.0000

Ao =
     0
     0
     1
     1
     0
     0

W =
     2
     1
     2

>> h = firpm(N,Fo,Ao,W)

h =
   -0.0179   -0.0229    0.0324    0.0108    0.0295    0.0881   -0.1709
   -0.2296    0.2743    0.2743   -0.2296   -0.1709    0.0881    0.0295
    0.0108    0.0324   -0.0229   -0.0179

>> freqz(h,1,512,100);
>> print -deps 'ampresponse.eps'
```

3 extra points for recognizing that ripple specs are not met using `firpmord` output ($N = 17$), and increasing N until they are ($N = 19$).

2. (13 pts) Download the sound file `noisypeaches.wav`[2] containing speech embedded in white noise.

   (a) (5 pts) Plot the spectrogram of `noisypeaches.wav` to help you understand its spectral content but there is no need to submit it. Design a low pass filter using the window method with a Kaiser window of length 100 and β=10. The cut-off frequency of the filter should be 4 kHz. Plot its impulse response and magnitude of frequency response.

   (b) (3 pts) Apply this filter to the noisy speech signal either by the FFT method of simple filtering. Listen and describe the result compared to the original.

[2] `https://ccrma.stanford.edu/~jos/sasp/hw/noisypeaches.wav`

(c) (5 pts) Now downsample the original noisy speech signal by a factor of two by simply throwing away every other sample. Listen to the result and compare it to the original higher sampling rate. Repeat the same downsampling scheme on the lowpass filtered speech signal and again, compare with its higher sampling rate version. Why does the latter pair (lowpass filtered) sound more similar than the first pair (unfiltered)?

Solution: Possible code for this problem, which also generates the required output files, is:

```
% HW6q6.m
%[signal,fs] = wavread('noisyoboe.wav'); % read in signal
[signal,fs] = wavread('noisypeaches.wav'); % read in signal
%%%%% filter design %%%%%%%%%%%%%%%%%%%%
M = 100; % time length of window
N = 8*M; % FFT length
n = -M/2+1:M/2; % time indices
%w_c = 6000; % choose cutoff of 6kHz for oboe lowpass filter
w_c = 4000; % choose cutoff of 6kHz for speech lowpass filter
norm_rad_w_c = w_c/(fs/2)*pi; % convert to normalized rad freq
hid=sin(norm_rad_w_c*n)./(norm_rad_w_c*n+eps); % ideal response
hid(n==0) = 1; % due to sinc exception
w=kaiser(M,10)';
hw=hid.*w;  % window ideal response
Hw=fft(hw,N); % frequency response
%%%% part a/c: plots %%%%%%%%%%%%%%%%%%%%
subplot(211); plot(n,hw); xlabel('n'); ylabel('amplitude');
title...
  ('lowpass filter using Kaiser(100,10) window: impulse and frequency response')
subplot(212);
plot(-fs/2:fs/(N-1):fs/2,...
  fftshift(20*log10(max(0.000001,abs(Hw)/max(abs(Hw))))));
grid
ylabel('normalized dB magnitude'); xlabel('Freq in Hz');
%%%%%% parts b/c/d: filter signal %%%%%%%
filtered_signal = conv(signal,hw); % simple filtering
wavwrite(filtered_signal./max(filtered_signal),fs,'hw6q6out.wav');
%%%%%% part e: downsample %%%%%%%%%%%%%%%
ds_signal = signal(mod(1:length(signal),2)==0); % throw away odd samples
ds_filtered_signal = filtered_signal(mod(1:length(signal),2)==0);
ds_fs = fs/2; % downsampled sampling rate
wavwrite(ds_signal./max(filtered_signal),ds_fs,'hw6q6ds.wav');
wavwrite(ds_filtered_signal./max(filtered_signal),ds_fs,'hw6q6dsf.wav');
```

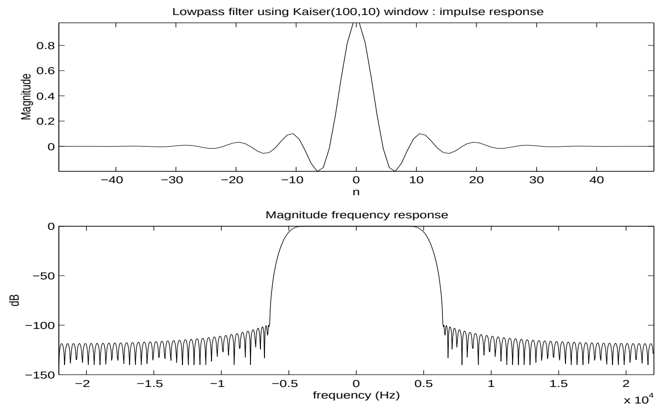

Figure 4: The impulse response and the magnitude frequency response of the designed lowpass filter with a cut-off frequency of 5 kHz.

(a) (10 pts) The spectrogram (Figure 5) shows rather obviously that most of the non-noise signal energy is concentrated the band of frequency up to about 6 kHz. We can then design a lowpass filter with a cut-off frequency here to retain most of the oboe signal while getting rid of (white) noise which spreads out through the whole spectrum. The plot of the filter's impulse response and magnitude of the frequency response are shown in Figure 4.

(b) (5 pts) By using either FFT or convolution the original signal with the filter just obtained, the output clearly has lower level of noise while the oboe sound still sounds very much the same. The spectrogram is shown in Figure 5.

(c) (10 pts) Now with the speech signal, it is not clear-cut where we should place the cut-off of the lowpass filter having sibilants like *"c"* and *"s"* (Figure 7). A compromise needs to be made but a good choice may be a cut-off at 4 kHz. The impulse and frequency response of the filter are shown in Figure 6.

(d) (5 pts) The output of the filter clearly shows that we lost some high frequency components in the speech along with the noise, especially *"c"* in *"juicy"* (Figure 7).

(e) (10 pts) If the signal is simply downsampled by throwing away every other sample, the resulting spectrum is a stretched version of the original and aliasing can occur. This is evident in listening, and even though it may sound brighter, having more high frequency content, the aliasing can be a cause of annoyance and is irrecoverable. On the other hand, if the signal is first bandlimited by lowpass filtering to make sure that, after stretching, the spectrum will not fold over the sampling frequency, the resulting downsampled sound will sound very much like the one before the conversion. The process is also reversible.

3. (10 pts)

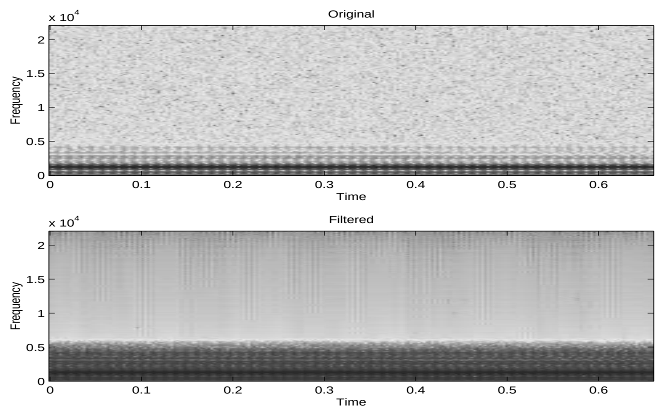

Figure 5: The spectrogram of the original noisy oboe sound and the lowpass filtered version using cut-off frequency of 5 kHz.

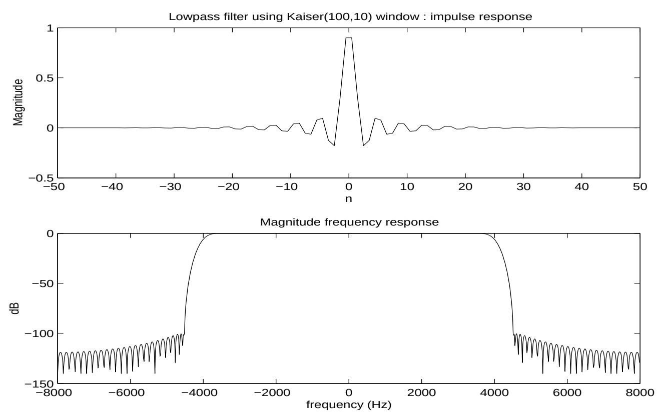

Figure 6: The impulse response and the magnitude frequency response of the designed lowpass filter with a cut-off frequency of 4 kHz.

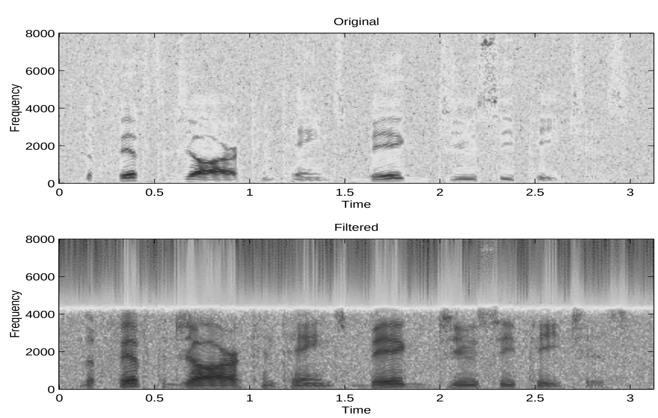

Figure 7: The spectrogram of the original noisy speech sound and the lowpass filtered version using cut-off frequency of 4 kHz.

In this problem, you will compare the ability of the Hann (a.k.a. "Raised Cosine") window to that of the Hamming window to resolve two sinusoids of significantly different amplitudes[3] and with "significantly different" frequencies[4]. To that end, write a matlab script to perform the following:

(a) (2 pts) Create a 64-sample sum of two cosines, the first with unity amplitude and normalized frequency 1/8 (cycles per sample), and the second with amplitude 0.001 and normalized frequency 3/8 (cycles per sample).

(b) (3 pts) Window this signal with a 64-sample Hann window (created using the matlab function `Hann()`)[5], and compute the resulting spectrum. For this problem, you may compute the spectrum of a signal using the matlab `freqz()` function, by setting $b = x$ (where x is your input signal vector), and $a = 1$. Finally, plot the magnitude spectrum in dB on a gridded figure with normalized frequency (in cycles per sample) along the x-axis.

(c) (3 pts) Repeat the above procedure for a 64-sample Hamming window, overlaying the spectrum with that of the Hann window.

(d) (2 pts) Which of the windows does a better job of resolving the two sinusoids? What drawback does this window have versus the other in regard to side-lobe levels?

Solution: The following code shows the required matlab operations:

```
% Part (a)
M = 64;
n = [0:M-1]';
x = cos(2*pi*(1/8)*n) + 0.001*cos(2*pi*(3/8)*n);

% Part (b)
hn = hann(M)/sum(hann(M));
xhn = x.*hn;
[Xhn,w] = freqz(xhn,1);

figure;
plot(w/(2*pi),20*log10(abs(Xhn)));
grid on;
xlabel('Frequency (cycles/sample)');
ylabel('Magnidue Spectrum (dB)');
hold on;
```

[3] Here we mean different by a few orders of magnitude.
[4] Here by "significantly different" we mean spaced by more than a few side-lobe widths.
[5] Throughout this problem, always normalize the window to read 0 dB at dc

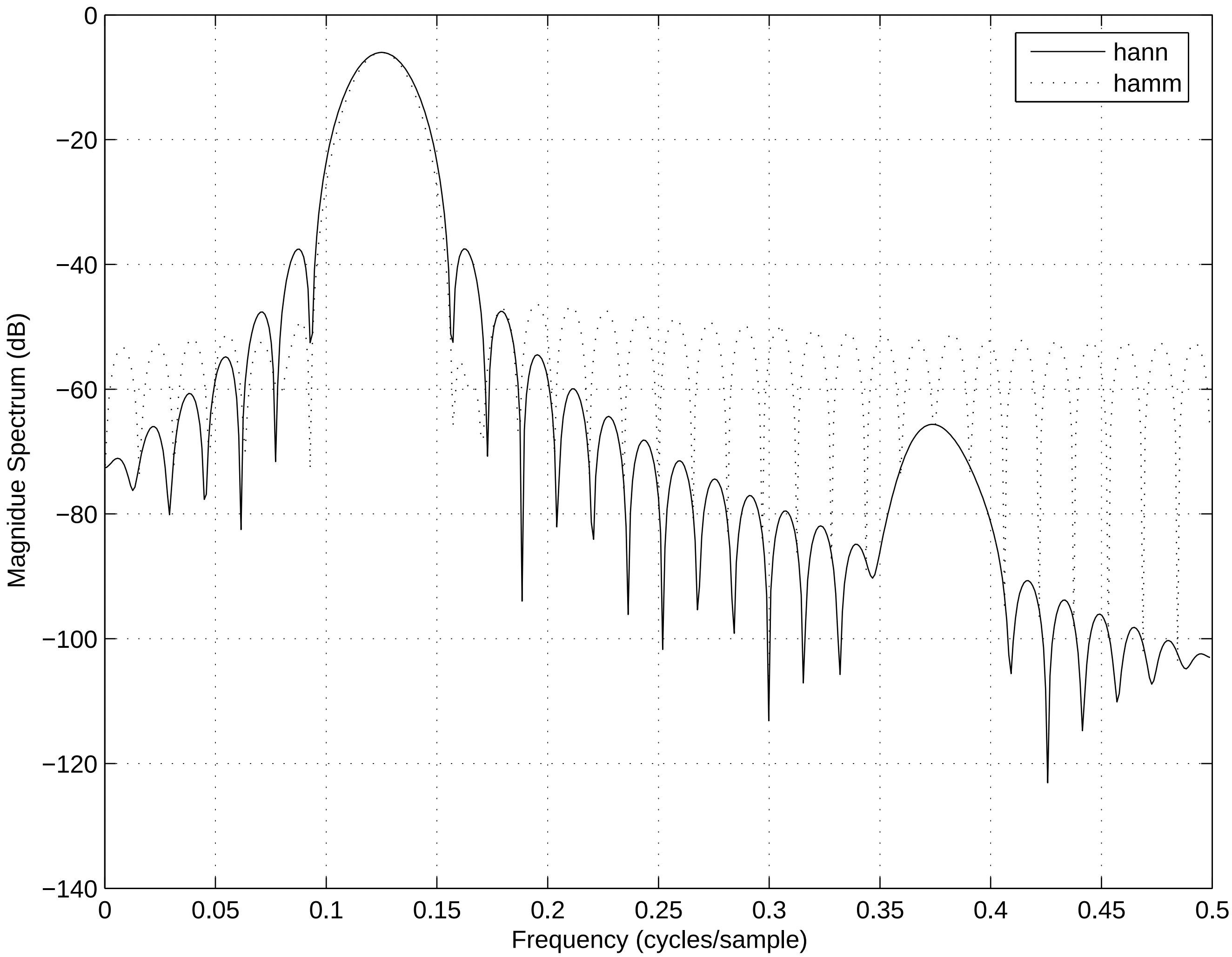

Figure 8: Magnitude spectra of two sinusoids windowed with Hann and Hamming windows.

```
% Part (c)
hm = hamming(M)/sum(hamming(M));
xhm = x.*hm;
[Xhm,w] = freqz(xhm,1);
plot(w/(2*pi),20*log10(abs(Xhm)),'b:');
legend({'hann','hamm'});

print(gcf,'-deps','../figures/comp.eps');
```

The two magnitude spectra are compared in 8.

As shown, the Hann window does a better job of resolving the two sinusoids, as its side-lobes roll off rather steeply, whereas the Hamming window side-lobes stay roughly constant in level at around -40 dB. Note, however, the the Hamming window has a lower first side-lobe level.