Music 421A
Spring 2019-2020
Homework #5
FIR Filter Design, Spectral Peak Finding
Due in one week

Theory Problem

1. (7 pts) **Least Squares Sinusoidal Parameter Estimation in the Presence of Interference**

   (a) (4 pts) Use the *orthogonality principle* to construct an optimal least-squares estimator for the amplitude A and phase ϕ of a sinusoid having known frequency ω_0 in the presence of another sinusoid at some unknown frequency. That is, the measurements consist of N samples of the following signal

   $$x(n) = \mathcal{A}e^{j\omega_0 n} + s(n)$$

   where $\mathcal{A} = A\exp(j\phi)$ and $s(n)$ is an unknown *interferer* (sinusoid at some other unknown frequency ω_i, phase ϕ_i, and amplitude A_i).

   (b) (3 pts) Under what conditions is the estimate $\hat{\mathcal{A}}$ unbiased?

Lab Problems

1. **FIR filter design with Least Squares method**

   (a) (10 pts) The MATLAB function **firls** designs an FIR filter of a given order M and certain desired filter magnitude response. In this problem you will write your own function to design an FIR lowpass filter with least squares. The impulse response of the filter should be symmetric and zero-phase. Here is a skeleton for the function you have to write :

```
function [hwin] = myfirls(M,cutoff)
% Function to design FIR lowpass filter impules response with least squares method
%   hwin = symmetric, zero-phase impulse response designed by least squares method
%   M = order of hwin (length will be M+1)
%   cutoff = cutoff frequency in rad/samp (between 0 and pi)

% Form a vector of frequencies spanning [0,pi]
% Form the desired frequency-response vector Hdes
% Form the least-squares matrix A
% Construct the symmetric impulse response hwin
end
```

   (b) (5 pts) Using your new function `myfirls`, design a lowpass filter of order 100 with cutoff frequency $fs/4$ with $fs = 44100$ Hz. Plot its magnitude response, and compare it to the one returned by MATLAB's `firls` function.

2. (20 pts) **Spectral Peak Estimation**

(a) (10 pts) Construct a program to find the frequencies (in Hz) and the magnitudes (in linear amplitude) of the positive-frequency peaks of an input DFT of a given signal, using the code skeleton shown below. Be sure to convert matlab frequency index numbers to frequencies in Hz.

```
function [peaks,freqs]=findpeaks(Xwdb,maxPeaks,fs,win,N)
% peaks = a vector containing the peak magnitude estimates (linear) using
%         parabolic interpolation in order from largest to smallest peak.
% freqs = a vector containing the frequency estimates (Hz) corresponding
%         to the peaks defined above
% Xwdb  = DFT magnitude (in dB scale) vector of a windowed signal.
%         NOTE that it may contain
%         only low-frequency (length < N/2+1), positive-frequency
%         (length = N/2+1), or all (length = N) bins of the FFT.
% maxPeaks = the number of peaks we are looking for
% fs       = sampling frequency in Hz
% win      = window used to obtain Xwdb (assumed zero phase)
% N        = NFFT, the number of points used in the FFT creating Xwdb

%-- Find all peaks (magnitudes and indices) by comparing each point of ---%
%-- magnitude spectrum with its two neighbors ---%
allPeaks = [];
for i=2:length(Xwdb)-1
    ...
    ...
end

%-- Order from largest to smallest magnitude, keep only maxPeaks of them --%
peaks = ...

%-- Do parabolic interpolation in dB magnitude to find more accurate peak --%
%-- and frequency estimates --%
for i=1:maxPeaks
    idx=find(Xwdb==peaks(i));
    %parabolic interpolation
    a=Xwdb(idx-1);
    b=Xwdb(idx);
    c=Xwdb(idx+1);
    ...
    ...
end
%-- Return linear amplitude and frequency in Hz --%
% NOTE that we must use knowledge of the window to correct the amplitude here
% if we have a TD cosine of amplitude 0.6, this output should be 0.6
peaks = ...
freqs = ...
```

(b) (4 pts) Test your `findpeaks.m` function on a length-255, 400-Hz cosine having amplitude 1 and no phase offset, i.e.,

$$x(n) = \cos(2\pi \cdot 400nT), \quad n = -127, \ldots, 127.$$

(Implicitly, a rectangular window of the same length is used here). Use your zero-phase zero-pad window function to extend the signal to length 2048 before peak detection. Let the sampling rate be $f_s = 8000$ Hz. Show your plot of the linear magnitude spectrum with the found peak clearly marked. Also plot the phase spectrum. The spectrum should be real because we zero-phase

windowed a cosine, which is even (*i.e.*, the FFT of a real, even sequence is real and even). Thus, the phase should be $\pm\pi$. (Matlab may produce some "imaginary dust" in the spectrum due to round off error.)

(c) (3 pts) Download `s1.wav`[1] and find the amplitudes and frequencies of each sinusoidal component in the signal. You need not zero-phase window the signal, nor should you zero-pad it. Just use a rectangular window whose length is the same as that of the signal. To avoid getting NaN in your spectrum due to the log of zero, use something like `Xwdb = 20*log10(abs(Xlinear)+eps);` .

(d) (3 pts) Again, with `s1.wav` but now using the Hamming window of length equal to that of $x(n)$ (255 samples) and with the same amount of zero-padding, find the frequency and amplitude estimates. Do you expect the estimates to be better in the case of Hamming window here? Why or why not?

3. (14 pts) **Simple F0 Estimation:** In this problem, you will be estimating the fundamental frequency $F0$ ("pitch") of an oboe sound, using the `findpeaks` function written previously. This problem is also concerned with the *resolvability* of the spectral peaks under different windows.

(a) Download `oboe.ff.C4B4.wav`[2] and similarly the Matlab files `oboeanal.m`[3] and `dbn.m`[4] into your directory. The posted version of `oboeanal.m` should be easy to adapt to using your `findpeaks()` function.

(b) (4 pts) Run the program `oboeanal.m`. Make sure you have your `findpeaks` function working already. Verify that the pitch estimate agrees with the plot. Explain what peak is used to estimate the pitch.

(c) (4 pts) Comment on the resolvability of peaks for the three windows.

(d) (6 pts) Modify the code on line 54, changing K to $(K-1)$ and then $(K+1)$ in order to vary the length of the signal under the windows. (NOTE: This line of code is after the window name has already been chosen, and currently contains only a comment line. Do NOT change K before this comment line. Note that some of the plots may then extend above 0 dB. Do not worry; you can zoom in on them anyway.) Comment on the resulting resolvability. Which case is more resolvable and amenable to spectral peak analysis?

[1] `http://ccrma.stanford.edu/~jos/sasp/hw/s1.wav`
[2] `http://ccrma.stanford.edu/~jos/sasp/hw/oboe.ff.C4B4.wav`
[3] `http://ccrma.stanford.edu/~jos/sasp/hw/oboeanal.m`
[4] `http://ccrma.stanford.edu/~jos/sasp/hw/dbn.m`