Comb Filter Design & Application

Assignment Due Date: Wednesday April 1, 10 a.m. via TURN-IN.

- No formal report required. Good, concise code documentation will take the place of a traditional written report.
- Turn in the following file: **yourlastname.m** where yourlastname is your last name in lower case. See web page for instructions on file submission.
- Work individually.

Data/Aim
Download two WAV files from the course web page. The file ***allplusaaa.wav*** consists of a guitar phrase (from the jazz standard *All the things you are*) additively corrupted by a second guitar playing the single note[1] A3 (twice in succession). Contained in ***aaanote.wav*** is a shorter sample of the A3 note played on the second guitar. Both files were recorded at 44.1 kHz (CD) sampling rate. Your assignment is to filter out and hence mostly remove the second guitar (harmonic interference) from the guitar phrase (desired signal).

Background
Notes played on stringed instruments at a certain pitch P Hz consist of the fundamental frequency P plus overtones or harmonics (2P, 3P, 4P, etc.). These harmonics give the note a rich quality. Your task is to estimate the frequencies of the harmonics of the A3 note for the second guitar, then design comb filters for removing the second guitar, leaving (mostly) the *All the things you are* phrase.

Methods
1. Use the Fast Fourier Transform (FFT) (instructions attached and discussed in class/lessons) to estimate the harmonic frequencies of the interference signal in *aaanote.wav*. Notice that the harmonics are not perfectly narrow lines: they are somewhat spread out. This is significant when considering the shape of the filter response that is needed to remove them.
2. Design, implement and test (on *allplusaaa.wav*) an IIR comb filter derived from the IIR-2 notch filter described in Lesson 23, pp. 2-3. The prototype filter here will be an IIR-2 notch filter. The upsampling-by-L technique for converting the prototype filter to a comb filter is described on p. 7 of the same lesson. Pay close attention to the choice of the r-parameter, which is the radius of the IIR-2 poles, and which affects the width of the notch and thereby relates to the last sentence of bullet item 1. (Trial & error and listening may be the best approach here, unless you can come up with a better way to design the value of r.)

Things to consider
- Listening to the comb filter output, you may notice a "residual signal" from each note played by the second guitar. Pay particular attention to how this residual signal sounds.
- How sensitive is the performance of the comb filter to the upsampling parameter L?
- The peaks of the harmonics in the spectrum of the *A*-note have finite width. What causes them to be spread out like this? Could it be related to the vibration of non-perfect guitar strings? (The fact that the harmonics have finite width affects the choice of the r-parameter.)
- Also, try testing your filter on the audio data in the file *aaanote.wav*. What happens? Is it what you expected? Does changing the value of r "null out" more of the energy in *aaanote.wav*? (i.e. is the optimum value of r data dependent?)

[1] Lesson 9: The fundamental pitch of A3 is 220Hz with higher and lower 'A's at octave multiples thereof, i.e. 110 Hz, 440Hz, 880Hz etc. The guitar was not perfectly in tune when used to record the data.

Matlab code: documentation

Carefully document each significant individual line of code.

- Describe what the code is doing
- If specific numerical parameters are involved, such as filter coeffs, explain how you derived them
- Some lines of code may require multiple lines of documentation. However, try to be brief - *lex parsimoniae.*

Specific things to include in your code documentation:
- Describe how you designed the comb filter from the measured harmonic interference frequencies
- Describe filter performance as a function of the comb filter parameters
- Try to explain why the "residual signal" sounds the way it does
- Issues listed under *Things to Consider* on p. 1.

Matlab code: basic requirements

Your code should be submitted as a single **m-file** yourlastname.m and I will test it by executing this command in Matlab:

[y] = yourlastname(x1, x2);

where x1 is the data array from ***allplusaaa*** and x2 is the data array from ***aaanote***. I will already have created the arrays x1 and x2 so **do NOT include any WAVREAD** commands in your m-file. All of your code should be self-contained with no additional parameters to be input at run time.

The output array y will contain the signal in *allplusaaa* after the two A-notes are filtered out, i.e. the comb filter output.

Your code must generate three separate **plots** using SUBPLOT(3,1,…). The plots should be:

1. SEMILOGY plot of the magnitude spectrum computed (using FFT) from the data in *aaanote*
2. SEMILOGY plot of the magnitude spectrum computed from the data in *allplusaaa*
3. SEMILOGY plot of the gain response of the comb filter, plus on the same axes the magnitude spectrum computed from the comb filter output data

(None of these are n-domain sequences so do not use STEM.)

Spectra plotted from F = 0 (dc) to 22.05 kHz (Fs/2) are too crowded. Hence, the x-axis of your spectra should cover the limited range 0 - 2 kHz. Sample code for doing this is:

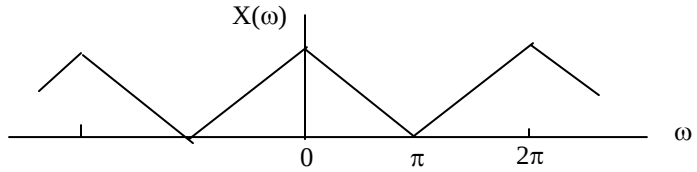
```
Y=fft(y);  %  Output audio data is in y. Spectrum returned in Y.
N=length(y); % No. data points in y
dF = Fs/N;  % Hz-interval between adjacent points in spectrum
F=0:dF:2000; % Creating a Hz-axis for plotting, from 0 to 2kHz
H=freqz(bcomb,acomb,2*pi*F/Fs); % Compute gain response of comb filter from dc to 2kHz
subplot(3,1,3),semilogy(F,abs(H), F, abs(Y(1:length(F))) )  %  Plot filter gain and spectrum of output signal on same axes
axis([0 max(F) -inf inf])  %  Scale axes
```

Take care to label and scale all axes, and don't leave any gaps at the end of the frequency axis.

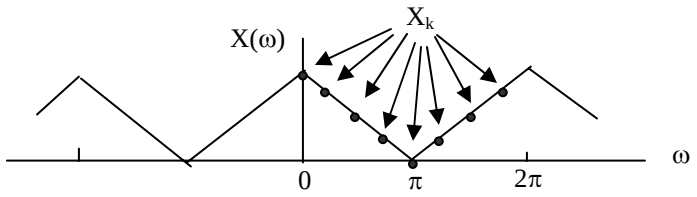
Finally, your code must play the comb filter output in array y, using: **sound(y,44100)**.

(Prior to implementing SOUND, it is a good idea to normalize your y-array so that it occupies the amplitude range -1 to +1.)

Basics of using the FFT: What does the FFT compute? Suppose your signal had a triangular spectrum, so that its DtFt looked like:



Taking the FFT of N samples of the signal x(n) yields N samples of X(ω), called DFT (Discrete Fourier Transform) coefficients X_k; $0 \le k \le N-1$. The DFT coefficients correspond to the following points on the ω-axis: 0, $\frac{2\pi}{N}, \frac{4\pi}{N}, \frac{6\pi}{N}, \frac{8\pi}{N}, ..., \frac{(N-1)2\pi}{N}$. Knowing the digital-analog frequency scaling formula $\omega = \Omega T$, you can also calculate the corresponding points on the F (Hz) or Ω (rads/s) scale. Hence, the N DFT coefficients fall at the frequencies shown below (for the very low value of N=8).



When the data x(n) are real, the X_k coefficients have the symmetry shown, so when plotting, assuming N is even, we usually plot from k=0 to k=N/2, which represents the range $\omega = 0 \rightarrow \pi$.

Armed with the above facts, you should now be able to measure the harmonic frequencies in *aaanote*, simply by observing the plot of $|X_k|$ for the given excerpt of the interference signal. Hint: use SEMILOGY to create useful spectrum plots.
