EE-3220 LABORATORY

Week 7

Notch Filters and Interference Removal

Goal – Design a multistage digital filter to remove interfering tones from a signal.

Materials

- Laptop computer with MATLAB (Octave may be substituted; minor changes are required).
- Earbuds or similar are recommended. The speakers on your laptop likely reproduce frequencies down to 300 Hz acceptably, are noticeably quiet at 250 Hz, and have virtually no audible output at 200 Hz. Many headphones and earbuds can acceptably reproduce low frequencies down to 30 Hz, so headphones are preferred for this lab.

Overview:
Design a filter consisting of cascaded notch filters to improve an audio signal that is corrupted with interfering tones. Perform a DTFT on the given signal to determine the spectral location of the interfering tones.

Download nn_filename.wav corresponding to your user name. The files can be found on the course website. The commands for loading and listening to audio in MATLAB are:

```
[x, fs] = audioread('filename.wav');
sound(x, fs)
```

Each audio file contains ***three*** interfering tones. The interfering tones are below 1000 Hz. Therefore, you only need to investigate the range of digital frequencies from DC to 1000 Hz, which is 2π(1000/fs) or about 0.045π radians/second since the sampling frequency is 44100 Hz. This can be done with the MATLAB commands:

```
f = 0:1:1000; % 1 Hz density, greater density will be slower
omega = f/fs * 2*pi; % Convert from Hz to radians/sample
M = length(omega);
```

Note that M is the number frequencies that are evaluated in this DTFT. The spectral spacing or density will be 1 Hz or 1/44100×2π radians/sample.

Recall that the DTFT can be computed and plotted with:

```
X = NaN(1,M);
n = 0:length(x)-1;
for k = 1:M, X(k) = exp(-1j*omega(k)*n)*x; end
figure, plot(f, abs(X))
```

Once you have found a tone to notch out, record the value for the digital frequency, ω_o, and construct a filter using the general form for the second order notch filter

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

where G is a gain factor used to normalize the filter and r controls the width of the notch. It is safe to ignore the effect of G. Use "fvtool" or "freqz" to see the frequency response of your filter. Set the scale to hertz (consult the documentation if needed). Notice that the passband gain of the filter is very close

to 1 (0 dB). You will need to experiment with various values of r to ensure that the notch is sufficiently narrow.

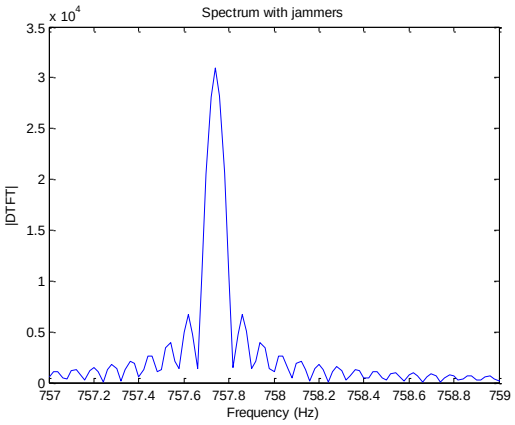
Apply the filter using:

```
x1 = filter(b,a,x);
```

Compute the DTFT of this filtered signal, x1, using a loop similar to the one above.

Compare this DTFT to the original. Continue this process to remove the other interfering tones. Note that there are some strong tones in the music that are not interferers, so the 3 highest DTFT values aren't necessarily the interferers. If you remove 3 tones but still hear an interfering tone, you may need to try changing which 3 frequencies you remove. For example, the 3 interferers might be the strongest, third strongest, and fourth strongest. Also, there is a small chance that some of your interferers are very close to each other in frequency; in some rare cases you will be able to eliminate 2 interferers with a single notch filter.

Also note that the 1 Hz frequency sampling might be insufficient. The audio is 11.96 s long, resulting in a frequency resolution of 0.0836 Hz. The main spectral lobe of a persistent tone will be twice this width, and there will be decaying sidelobes of this width. These sidelobes are quite powerful and if your 1 Hz density hits one of them you are likely to see the energy in your spectrum. But, you may need to track down some interferers using a greater spectral density (lower spectral spacing). For example, the figure at right shows a strong tone at 757.74 Hz that will be difficult to detect if sampled at integer frequencies. Note that the peak in this graph is at about 3×10^4 while the strongest component of the desired audio is at about 10% of that magnitude.



Optional alternative approach: Use MATLAB's convolve function to combine all 3 filters into a single filter to remove all tones at once (e.g., a = conv(conv(a_w0,a_w1),a_w2);).

Save the final audio with

```
audiowrite('filename_clean.wav', y, fs)
```

where y is the final output sequence signal.

Submit:

1. Provide a summary paragraph. Comment on ***both*** the spectrum and your perception of the final audio.
2. Also provide a paragraph describing the problem and your design solution.
3. Submit a magnitude plot of the DTFT for the original signal. Clearly label all plots.
4. Submit a table listing the tones (both in radians/sample and Hz) that were present in the original signal.
5. Submit the magnitude response plots for your notch filters, along with the corresponding filter coefficients used to remove these tones.
6. Submit a magnitude plot of the DTFT for the final signal.