Part A: New Filters

Since multiplying a time function by cos(2πf0n) shifts the spectrum of the signal by f0 in the frequency domain, we can shift the frequency response of a filter by multiplying its time domain representation (its unit-pulse response) by cos(2πf0n). In the case of an FIR filter, this is particularly handy, because the unit-sample response *is* the filter. So if we multiply the coefficients of an FIR filter by  cos(2πf0n) we get a new filter having the same *shape* as the old one (including *negative* frequencies) but centered about f0 rather than zero.

We saw this happen indirectly when we changed the sign of a1 in our first-order IIR filter. The frequency response went from low pass to high pass and the new pulse response contained the same values as the old, but they alternated in sign (i.e. were multiplied by {1,−1,1,−1,…}=cos(2π1/2n)=cos(nπ)=(−1)n).

Let’s try this with an FIR filter.

* Generate a length 50 FIR filter having a bandwidth of 1000 Hz.
>> b0=fir1(50,1000/5000);
* Plot its frequency response to verify it is correct.
>> freqz(b0,1)
* Create a new filter by multiplying the coefficients by cos(nπ).
>> b1 = b0 .\* cos(pi\*[0:50]);
* Once again we have a high pass filter, but this time much more nearly ideal than our simple IIR.
>> freqz(b1,1)
* Apply this filter to your speech reference signal.
>> out1=filter(b1, 1, sig1);
* Listen to the filter output and plot its spectrogram.
* What happens if we don’t shift all the way to Fs/2? Let’s try this, but with a narrower filter. Generate a length 50 FIR filter having a bandwidth of 200 Hz.
>> b=fir1(50,200/5000);
* Plot its frequency response to verify it is correct.
>> freqz(b,1)
* Shift its response up to 2 kHz (0.2\*Fs).
>> b2 = b .\* cos(0.4\*pi\*[0:50]);
>> freqz(b2,1)
* **What is the bandwidth of this bandpass filter? How does it relate to the bandwidth of the lowpass filter?**
* Apply it to your speech reference signal.
>> out2=filter(b2, 1, sig1);
* Listen to the filter output and plot its spectrogram.
	+ Hint: When a signal is so small that it is hard to hear, or so large that it distorts, use the soundsc() function instead of sound(). This scales the signal to just fill the available output range.
	>> soundsc(out2, Fs)

PART C: Better IIR Filters

So far, we’ve seen that long (in our case) FIR filters can be pretty good and that short filters of either type are pretty bad. What about long IIR filters?

It turns out that if we let our transfer function have both a numerator and a denominator (i.e., both a and b coefficients) we can get filters of “good” shape with a modest number of total coefficients. Filters of this form have been used in analog circuits (where filter order corresponds to the number of inductors and capacitors and op amps, and hence smaller is better) and the techniques for designing them carry over to digital filters. One of these is the Butterworth, which we’ve already seen. Another is the Elliptic or Cauer filter (so called because its design involves the use of the Jacobi elliptic function).

* Let’s design an 8th order Elliptic low pass filter with a cutoff frequency of Fs/8:
>> [b,a] = ellip(8, 1, 60, 1000/5000);
>> freqz(b,a)
The first argument (8) is the filter order. The second argument (1) is the *passband ripple*, the amount (in dB) that we're willing to allow the actual value of the transfer function in the passband to deviate from the desired value (1, or 0 dB). The third argument (60) is the maximum value (in dB) that we're willing to allow the transfer function to have in the stopband. Subject to these constraints, the Elliptic design gives us the shortest possible *transition band*, the region where the transfer function is changing from one to zero.
* Apply this filter to your reference signal, listen to the results, and look at the spectrogram. **Can you see the difference between this filter and the “equivalent” FIR filter (b0)? Can you hear it?**
* **Write a summary of each of the different types of filter we've looked at (boxcar, first order IIR, Butterworth, fir1, elliptic). Consider such factors as the shape of the frequency response, phase characteristics, computational complexity, etc.**
* **Is any one type of filter clearly the "best" (or the "worst")? If not, can you identify circumstances where a particular filter *would* be the best choice (or should *not* be chosen)?**