Lab 4. FIR & IIR Filters in Matlab

Filter Design

The goal of filtering is to perform frequency-dependent alteration of a signal. A simple design specification for a filter might be to remove noise above a certain cutoff frequency. A more complete specification might call for a specific amount of passband ripple (Rp, in decibels), stopband attenuation (Rs, in decibels), or transition width (Wp - Ws, in hertz). A precise specification might ask to achieve the performance goals with the minimum filter order, call for an arbitrary magnitude response, or require an FIR filter.

IIR filter design methods differ primarily in how performance is specified. For loosely specified requirements, as in the first case described previously, a Butterworth filter is often sufficient. More rigorous filter requirements can be met with Chebyshev and elliptic filters. The Signal Processing Toolbox order selection functions estimate the minimum filter order that meets a given set of requirements. To meet specifications with more rigid constraints, such as linear phase or arbitrary response, it is best to use direct IIR methods such as the Yule-Walker method or FIR methods.

Filter Configurations

First, recall that when dealing with sampled signals, we can normalize the frequencies to the Nyquist frequency, which is half the sampling frequency. All the filter design functions in the Signal Processing Toolbox operate with normalized frequencies, so that they do not require the system sampling rate as an extra input argument. The normalized frequency is always in the interval $0 \leq f \leq 1$. For example, with a 1000 Hz sampling frequency, 300 Hz is $300/500 = 0.6$. To convert normalized frequency to angular frequency around the unit circle, multiply by π. To convert normalized frequency back to Hertz, multiply by half the sample frequency.

- Lowpass filters remove high frequencies (near 1)
- Highpass filters remove low frequencies (near 0)
- Bandpass filters pass a specified range of frequencies
- Bandstop filters remove a specified range of frequencies

Calculate a normalizing factor:

```
fs = 1e4;
f = 400;
nf = 400/(fs/2)
```

Filter Specifications in Matlab

- Wp - Passband cutoff frequencies (normalized)
- Ws - Stopband cutoff frequencies (normalized)
- Rp - Passband ripple: deviation from maximum gain (dB) in the passband
- Rs - Stopband attenuation: deviation from 0 gain (dB) in the stopband

The Filter Design and Analysis Tool (FDATool) shows these specifications graphically:

```
fdatool
```

The order of the filter will increase with more stringent specifications: decreases in Rp, Rs, or the width of the transition band.

In this Lab, just do the following items 1 → 16:

1. `fdatool`
2. Design a Lowpass, FIR Eqiripple filter, Minimum Order, Fs=1000hz, Fpass=60, Fstop=200, Apass 1dB, Astop 80dB. Design Filter
3. Was the filter designed as per specifications? Confirm dB at Fpass, and at Fstop.
4. Now look at the Phase Response. It is linear up to 200 hz and then there are jumps. What size are the jumps? Are the points where the jumps occur have anything to do with the points where the amplitude frequency response changes? (Use the icon to look at the magnitude and phase response together). Explain the reasons for the linear and the nonlinear behavior. Discounting the jumps, is the phase linear with frequency?
5. Look at $\frac{d\phi}{d\omega}$ - called the group delay. What is the numerical value of this delay?
6. Look at $\frac{\phi}{\omega}$. Comment.
7. Look at the filter impulse response. Comment on symmetry and about what point is it symmetric?
8. Look at i - Filter information. Note the filter length. Now how is this related to the Group delay? Comment.
9. Look at the pole-zero plot. How many poles? How many zeros? Does this make sense? Explain.

10. Look at filter coefficients. This is what gets implemented with varying degrees of precision.

11. Export the filter parameters (Num) to your workspace. Do `freqz(Num,1)`.

12. In the command window, do `sptool`. In sptool, Import Num to sptool. Change sampling frequency from the default 1, to 1000.

13. Now under SPTool: startup spt, view the signal. Under Spectra, use "Create" and do a 1024 point fft.

14. Under Options → Magnitude → Linear. Observe the difference between observing on linear and log scales.

15. Redesign your filter with Fstop =100. Import coefficients to sptool, look at the spectrum under sptool, using linear scale. Now do you see the "equiripple" in the passband in the magnitude response? And in the log scale, do you see the "equiripple" in the attenuation band?

16. Now in fdatool, for the same specs as the first FIR filter, design an IIR Butterworth filter. (Match exactly passband). Look at the amplitude response. Look at the phase response. Linear phase? Look at the Filter information. Did it meet the specs? How many poles and zeros? Look at the impulse response. Does it have any properties dissimilar to that of the FIR filter?

SKIP all items below.

IIR Filter Design

The primary advantage of IIR filters over FIR filters is that they typically meet a given set of specifications with a much lower filter order than a corresponding FIR filter.

Although IIR filters have nonlinear phase, data processing in Matlab is commonly performed off-line, that is, the entire data sequence is available prior to filtering. This allows for a noncausal, zero-phase filtering approach (via the *filtfilt* funtion), which eliminates the nonlinear phase distortion of an IIR filter.

The classical IIR filters - Butterworth, Chebyshev Types I and II, elliptic, and Bessel - all approximate the ideal "brick wall" filter in different ways. The Signal Processing Toolbox provides functions to create all these types of IIR filters in both the analog and digital domains (except Bessel, for which only the analog case is supported), and in lowpass, high-pass, bandpass, and bandstop configurations. For most filter types, you can also find the lowest filter order that fits a given filter specification in terms of passband ripple, stopband attenuation, and the transition band widths.