Lab 10 – IIR filter design

Objectives: In this lab we will use Matlab to generate various IIR filters and apply them to filter some signals.

10.1 Low pass filters (script)

Design a digital low pass filter $H(z)$ to process analog signals as shown in the system below

$$x_a(t) \longrightarrow \boxed{\mathrm{A/D}} \longrightarrow \boxed{H(z)} \longrightarrow \boxed{\mathrm{D/A}} \longrightarrow y_a(t)$$

Build IIR filters which satisfy the following requirements:

- Sampling rate = 10 kHz
- Cut-off frequency = 2 kHz
- passband ripple = 1 dB
- stopband attenuation = 40 dB

Note that the quantities passband ripple and stopband attenuation respectively quantify (in decibels) the parameters δ_1 and δ_2 from the class notes.

(a) Design your filters using the built-in Matlab commands of `butter()`, `cheby1()`, `cheby2()`, and `ellip()`. For each of the filters, use the input parameters as specified above and design filters of order N = 6. Carefully read documentations for these commands and understand the inputs and outputs.

(b) Use `freqz()` to obtain 2001-point frequency response of the designed filters. In a 2x2 figure, plot the magnitude response (normalized and in decibels) for all the 4 filters and compare. Keep y-axis limits same for all the subplots for easy comparison. In another 2x2 figure, plot the phase response of these filters. Are any of these linear-phase?

(c) Repeat (a) and (b) for filter order N = 12. For the same order, which of the 4 designed filters would be preferable?

(d) Using `impz()` visualize the impulse response of the filters designed in (a).

(e) Vary the filter parameters and visualize how the frequency response changes (optional).

(f) Generate samples of the signal $x_a(t) = \sin(2\pi f_1 t) + \sin(2\pi f_2 t)$ at a sampling rate of $f_s = 10\ kHz$ for a duration of 1 second. Let $f_1 = 500\ Hz$ and $f_2 = 3\ kHz$. Filter this signal using the `filter()` command and each of the 4 filters designed in (a). In a 4x2 figure, plot the first 100 samples of the input signal and output signal from the filter. Verify that your filters are working as expected.

10.2 Notch filter (script)

A notch filter is a special filter which passes most frequencies virtually unchanged but has zero frequency response at $\omega = \omega_0$. We will look at two ways to do this.

(a) (FIR filter) An easy way to do this is to place two zeros on the unit circle at $e^{\pm j\omega_0}$. This ensures that there is frequency null at $\omega = \pm\omega_0$. This gives the system function

$$H(z) = b_0(1 - e^{j\omega_0}z^{-1})(1 - e^{-j\omega_0}z^{-1})$$

where b_0 is the gain factor for this filter. Note that our designed filter is causal. Select the gain factor b_0 such that $H(1) = 1$. Use `freqz()` to plot 2001-point frequency response of this filter when $\omega_0 = \frac{\pi}{4}$. Note that the inputs 'b' and 'a' to this function are coefficients of the polynomials expressed in z^{-1}.

(b) (IIR filter) Alternately, along with the two zeros on the unit circle at $e^{\pm j\omega_0}$, we can additionally place two poles at $r_0\, e^{\pm j\omega_0}$, where $r_0 < 1$ but is close to 1. This gives the system function

$$H(z) = b_0 \frac{(1 - e^{j\omega_0}z^{-1})(1 - e^{-j\omega_0}z^{-1})}{(1 - r_0 e^{j\omega_0}z^{-1})(1 - r_0 e^{-j\omega_0}z^{-1})}$$

where b_0 is the gain factor. Select the gain factor b_0 such that $H(1) = 1$. Plot the filter frequency response using `freqz()` when notch is located at $\omega_0 = \frac{\pi}{4}$ and $r_0 = 0.99$. Use the geometric evaluation to understand the frequency behaviour of this filter. Is this filter stable? Why?

(c) Compare the magnitude and phase response of the two notch filters in (a) and (b). For the second filter vary the value of r_0 and note its effect on the frequency response.

(d) Use matlab command `fvtool()` to visualize various aspects of the notch filters in (a) and (b) including their magnitude response, phase response, impulse response, pole-zero plots, etc.

(e) Load the `handel` sound file in matlab by typing `load handel`. Let this be called $x[n]$. Listen to it using the command `sound()`. To this signal add a sinusoid $\sin(2\pi f_0 t)$ of same duration and sampling rate where $f_0 = 1024\ Hz$. Listen to this modified signal. Apply the two notch filters designed in (a) and (b) to this modified signal using the `filter()` command. Listen to the two filter outputs.

If you are unable to load the `handel` sound file above, instead generate a 2 second white noise signal in Matlab as follows: $x[n] = rand(1, 2 * f_s) - 0.5$ with $f_s = 8192\ Hz$ and proceed as above.

(f) In a 2x2 figure, plot the first 100 samples of the input signal and output signal from the filter. Verify that your filters are working as expected.