Stanford University
EE 264: Digital Signal Processing
Summer, 2018

Homework #05

Date assigned: July 28, 2018 Date due: August 5, 2018

Reading: This assignment covers primarily lecture 8 - July 30 (Problem 1-2) and lecture 9 - August 1 (Problem 3-6).
Homework submission: Please submit your solutions on Gradescope. Create a single .pdf file containing all your analytical derivations, sketches, plots, and Matlab code (if any).

Problem 1: (25 points)

Consider the system of Example 5.8 of the textbook. Its system function is

$$H(z) = 0.05634 \frac{(1+z^{-1})(1-1.0166z^{-1}+z^{-2})}{(1-0.683z^{-1})(1-1.4461z^{-1}+0.7957z^{-2})} \tag{1}$$

(a) Draw the signal flow graph of an implementation of the system as a cascade of a second-order system with a first-order system. Mark the values of the coefficients on the signal flow graph.

(b) Draw a direct form II signal flow graph for implementing the system as a single third-order flow graph.

(c) Plot the pole-zero diagram of $H(z)$ for the unquantified coefficients. You may use `conv` to multiply out the numerator and denominator and obtain the coefficients vectors `a` and `b`. Then use `zplane` to plot the pole-zero diagram.

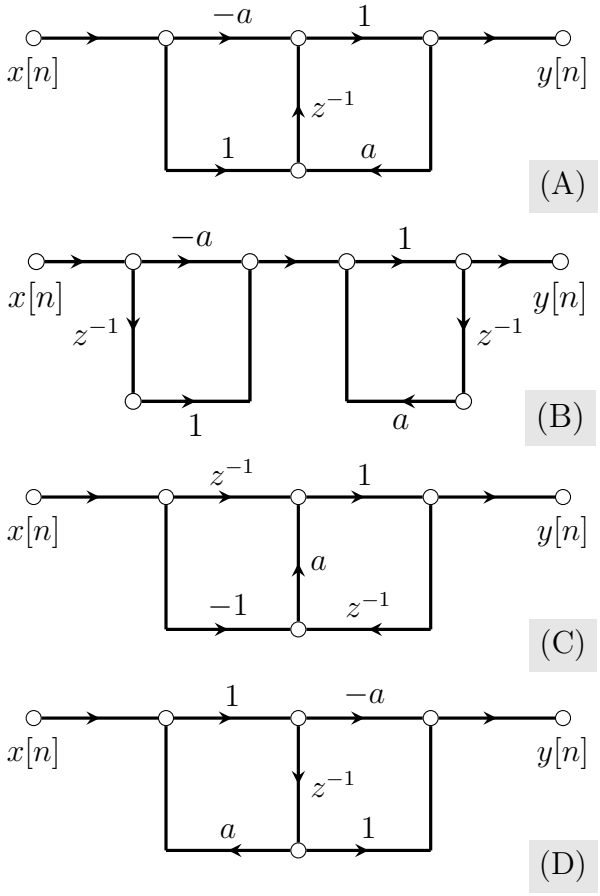
(d) Quantize the filter coefficients with 6 bits to the right of the imagined binary point. Call the Q6-quantized coefficient vectors `ah` and `bh`. Then compare the pole-zero locations using

```
plot(roots(b),'ok'); plot(roots(a),'xk'); hold on
plot(roots(bh),'or'); plot(roots(ah),'xr');
```

Also plot the log magnitude of the frequency response of both the "unquantized" and Q6-quantized coefficients on the same graph. Hand in both plots. You should observe that there is very little difference between the quantized and unquantized systems. What is your intuition about why this is so?

Problem 2: (25 points) (from the midterm of Spring-2014)

The following flow graphs represent equivalent difference equations for implementing a digital all-pass filter. We will compare these systems when implemented with 16-bit fixed-point arithmetic. In the following diagrams, the signal values and constant multipliers are all scaled as Q_{15} integers. The multiplications shown are quantized to 16 bits (15 bits plus a sign bit) directly after the multiplication and before any additions are done. Answer the questions below about the flow graphs.



(a) What is the system $H(z)$ that describes all the above systems when the difference equations are implemented without quantization?

(b) Re-draw the signal plow graphs inserting noise sources to represent quantization of the multiplications. Mark each of them with the symbol σ_{15}^2 denoting the average power of the source. Give a formula for σ_{15}^2.

(c) Assuming that $|a| < 1$, and that the output of each system is represented as $\hat{y}[n] = y[n] + f[n]$, determine the power spectrum, $\Phi_{ff}(e^{j\omega})$, of the noise component, $f[n]$, for each flow graph. Be sure to combine independent noise sources to simplify the analysis.

(d) For which system will the total noise power be the largest? Determine a closed-form formula for the largest total noise power, σ_f^2. Your equations should be a function of σ_{15}^2 and a only.

Hint: When calculating the total noise power, please avoid the frequency domain integration.

Problem 3: Classic filters (25 points)

This question will help you familiarize yourself with designing classic filters such as Butterworth, Chebyshev type I & II, and elliptic filter using Matlab.

(a) Design a Butterworth, a Chebyshev type I, a Chebyshev type II, and an elliptic lowpass filter. All filters should have order 8 and 3-dB bandwidth of approximately 4 kHz for a sampling rate of $1/T = 16$ kHz. You're free to set all other required parameters such as passband ripple and stopband attenuation. On a single graph, plot the magnitude response of all filters. On a different graph, plot the unwrapped phase response of all filters.

Note: For this question you may use the Matlab functions `butter, cheby1, cheby2`, and `ellip`.

(b) Use your filters to filter the speech signal `speech_dft.wav`, available on Canvas. This speech signal was recorded with sampling frequency of 22.05 kHz. Hence, before filtering it with your filters, you need to use your knowledge on changing the sampling rate by a non-integer factor to resample the signal to 16 kHz. Use the smallest upsampling and downsampling integer factors possible.

Hint: Use the function `decimate` after upsampling, but remember that the lowpass filter after interpolation must have gain equal to the upsampling factor L.

Note: 4 kHz is approximately the bandwidth of the telephone system. Hence, after filtering, the output signals should have roughly the same quality of a conversation over the phone. However, some of your filters may introduce enough magnitude/phase distortion to worsen the sound quality.

(c) Denote as $H_{16}(z)$ any of your filters designed above for sampling rate of $1/T = 16$ kHz. Suppose you'd like to design a filter $H_{32}(z)$ that has the same characteristics (e.g., bandwidth of 4 kHz) of $H_{16}(z)$, but that can be used on signals sampled at a rate of 32 kHz. Give an expression for $H_{32}(z)$ as a function of $H_{16}(z)$.

Hint: revisit the notes on interchanging filtering and downsampling.

Problem 4: Notch filter for ECG signals (35 points)

Notch filters are stopband filters with characteristically narrow stopband. They are used to remove specific frequency components from signals. In this problem, you will design digital notch filters to remove 60 Hz interference from an electrocardiogram (ECG) signal. 60 Hz is the frequency of alternating current (AC) in the electric power grid of most countries. The 60-Hz interference appears in many applications, and in particular, in biological signal recordings.

An analog notch filter is given by

$$H(s) = \frac{s^2 + \lambda^2}{s^2 + bs + \lambda^2}. \tag{2}$$

The magnitude plot of this filter is sketched in Figure 1

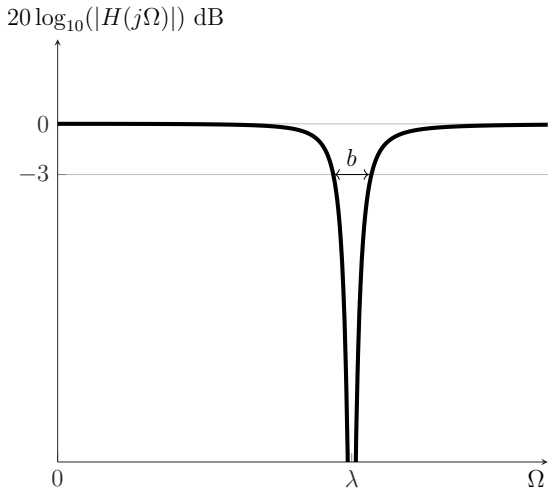


Figure 1: Typical notch filter magnitude response

You will design a discrete-time version of this filter, and use it to filter an ECG signal that has been corrupted by 60 Hz interference. An excerpt of that signal is show in Figure 2. The rapid oscillations are due to 60 Hz interference.

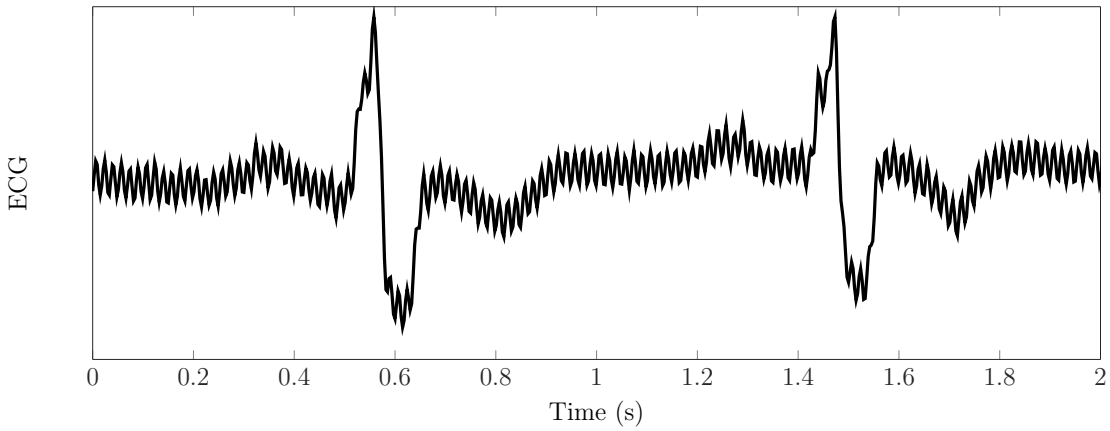


Figure 2: ECG corrupted by 60-Hz interference.

This ECG signal was sampled with sampling period $T = 3$ ms and it's available on Canvas. This file also contains a "clean" ECG signal, which you can use for comparison. Use the Matlab script `notch_filter_ecg_template.m` to load the recordings and write your code to solve the following questions.

(a) Choose appropriate values of b and λ that would make the notch filter given in (2) suitable for this application. There is more than one right answer.

(b) Use the bilinear transformation with and without frequency pre-warping to obtain discrete-time equivalents of the analog notch filter in (2). Specify what value of Ω_p you used for frequency pre-warping.

(c) On the same graph, plot the magnitude response of the digital filters obtained by the bilinear transformation. For easier comparison, the x-axis of your plot should be $\omega/(2\pi T)$ i.e., actual frequency in Hz. Discuss which filter (with or without frequency pre-warping) is more suitable for this application. For the remainder of this problem, you only need to consider the filter you think is best.

(d) For the filter you chose in part (c), plot the pole-zero diagram and discuss what characteristics of the pole-zero diagram indicate that this filter works as a notch filter.

(e) Use the filter you designed in part (c) to filter the ECG signal corrupted by 60 Hz interference. Plot the output of your filter and compare it with the clean ECG signal, and with the corrupted ECG signal.

(f) Use either the Parks-McClellan algorithm (`firpm`) or the least-squares algorithm (`firls`) to design a linear-phase FIR notch filter. Let the filter length be at most 53 (order 52).

Use the following figure to guide your design choices.

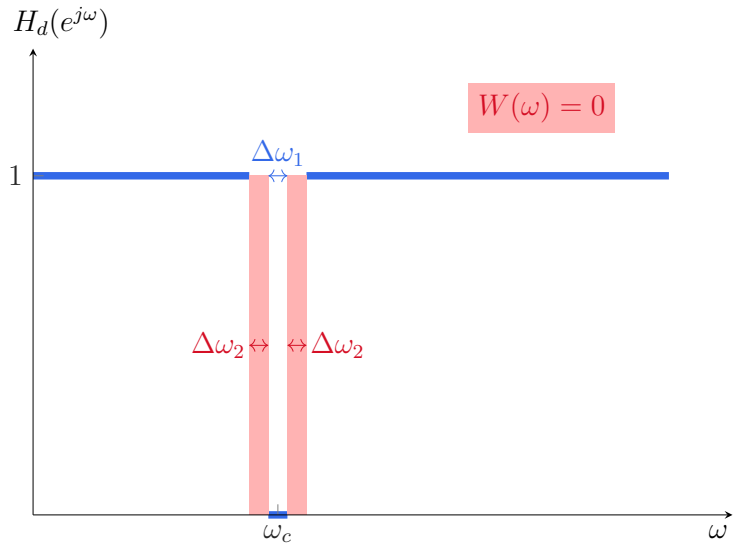


Figure 3: Design specifications for FIR notch filter.

Specify ω_c, $\Delta\omega_1$, and $\Delta\omega_2$ that you chose in your design. Clearly, this application would require $\Delta\omega_1$ and $\Delta\omega_2$ to be small as possible, but this will introduce too many undesired oscillations in the passband.

Turn in the magnitude and phase plots of your design, as well as the ECG signal filtered by your FIR filter. Note that your filter will introduce a delay of $M/2$, where M is the filter order. Hence, to compare your result with the "clean" ECG you will need to delay or advance one of the sequences.

Note: part (f) of this problem is solved on this 2004 paper using the Parks-McClellan algorithm. The figure below shows their best result for an FIR filter of order 52. To receive full credit for this question, you are not expected to do as well or close to what they achieved (though you could). You're expected, however, to make reasonable design choices.

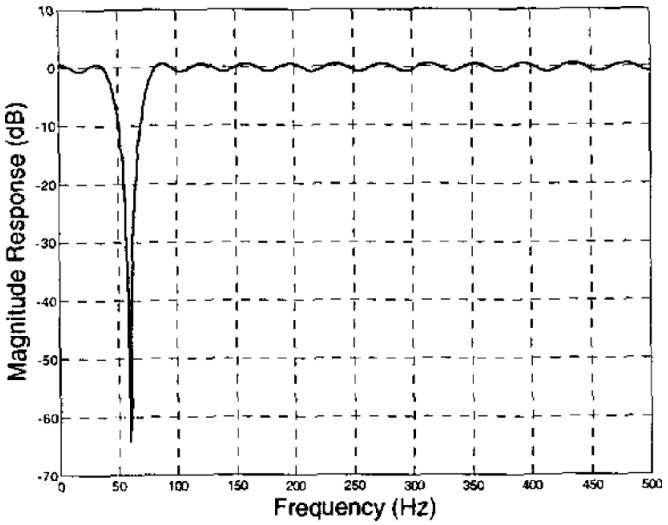


Figure 4: Result of notch filter FIR design using the Parks-McClellan algorithm. The filter order was 52.

Problem 5: Hilbert transformer (25 points)

In Homework #4 you saw some applications of the Hilbert transform. In this problem, we will focus on the Hibert transformer, which is an LTI system used to calculate the Hilbert transform of time-domain signals. More specifically, the Hilbert transformer is used to phase-shift signals by 90 degrees. Hence, the output of a Hilbert transformer to a cosine is a sine of same frequency. This operation has applications in digital communication, radar systems, and medical imaging.

As discussed in class, the Hilbert transform is defined by the convolution:

$$\mathcal{H}\{x(t)\} = \frac{1}{\pi t} * x(t) \Longleftrightarrow H_{HT}(j\Omega) = -j\text{sign}(\Omega) = \begin{cases} -j, & \Omega > 0 \\ 0, & \Omega = 0 \\ j, & \Omega < 0 \end{cases} \tag{3}$$

We will approximate this by the following discrete-time frequency response

$$H_d(e^{j\omega}) = -j\text{sign}(\omega) = \begin{cases} -j, & 0 < \omega \leq \pi \\ 0, & \omega = 0 \\ j, & -\pi < \omega < 0 \end{cases} \tag{4}$$

Note that this filter acts as an all-pass filter (except at $\omega = 0$) with 90-degree phase shift. Its impulse response is given by

$$h_d[n] = \begin{cases} \frac{2}{\pi}\frac{\sin^2(\pi n/2)}{n}, & n \neq 0 \\ 0, & n = 0 \end{cases} \tag{5}$$

The impulse response has odd symmetry about $n = 0$ i.e., $h_d[n] = -h_d[-n]$. Hence, we can design linear-phase FIR systems to approximate $h_d[n]$.

For the questions below assume the filter order is $M = 30$ (i.e., 31 coefficients).

Note: Since the FIR filters you will design are linear phase, their frequency response will be approximately

$$H(e^{j\omega}) \approx e^{-j\omega M/2}(-j\text{sign}(\omega)) = \begin{cases} e^{-j(\omega M/2+\pi/2)}, & 0 < \omega \leq \pi \\ 0, & \omega = 0 \\ e^{-j(\omega M/2-\pi/2)}, & -\pi < \omega < 0 \end{cases} \tag{6}$$

Therefore, they'll look like an all-pass except around $\omega = 0$ and $\omega = \pi$, but they still have the property of phase shifting the input signal by 90 degrees. However, due to causality and the linear phase condition, the output signal will also be delayed by $M/2$ samples.

(a) Window method. Use the window method with the window of your choice to design a linear phase FIR filter $H_{win}(z)$ that will approximate $H_d(e^{j\omega})$. The impulse response of the Hilbert transformer in discrete time is given in (5). Plot the magnitude and phase response of your design.

(b) Parks-McClellan. Use the Parks-McClellan algorithm to design a linear phase FIR filter $H_{PM}(z)$ that will approximate $H_d(e^{j\omega})$. In your design, assume that the weight function is

$$W(\omega) = \begin{cases} 0, & 0 \leq \omega < \Delta\omega \\ 1, & \Delta\omega \leq \omega \leq \pi - \Delta\omega \\ 0, & \pi - \Delta\omega < \omega \leq \pi \end{cases} \tag{7}$$

where $\Delta\omega = 0.1\pi$. This means that your Hilbert transformer will only work between 0.1π and 0.9π, since it will only have magnitude 1 in $[0.1\pi, 0.9\pi]$, Naturally, you'd like to make $\Delta\omega$ as small as possible, but that will introduce undesired ripples in the passband. Feel free to try different values of $\Delta\omega$.

Plot the magnitude and phase response of your design.

Matlab hints:

- When using the function `firpm`, note that Matlab expects the filter magnitude $|H_d(e^{j\omega})|$ (parameter `a`). However, $|H_d(e^{j\omega})| = 1$ over all frequencies, except zero. To differentiate that from an all-pass filter, you have to pass the extra parameter `'hilbert'`:

```
>> hpm = firpm(M, f, a, 'hilbert');
```

  where `f` is the normalized frequency vector and `a` is the magnitude at frequencies `f`. This way `hpm` will have odd symmetry.

- This question is already solved in the Matlab documentation of the function `firpm`.

(c) Least-squares. Use the least-squares algorithm (`firls`) to design a linear phase FIR filter $H_{ls}(z)$ that will approximate $H_d(e^{j\omega})$. Use the same assumptions and hints of part (b). Plot the magnitude and phase response of your design.

Problem 6: Linear equalization (25 points)

Figure 5 shows the block diagram of a digital communication system. In this system, a sequence of bits $x[n]$ is transmitted over a channel, which is a causal LTI system with z-transform $C(z)$. Since the channel is causal its output depends not only on the present bit, but also on previous bits transmitted over the channel. Hence, the output of the channel is the result of interference of many bits. This phenomenon is called *inter-symbol interference* (ISI). In addition to ISI, after the channel, a white and zero-mean Gaussian noise $r[n]$ with average power $\sigma^2 = 0.001$ is added to the signal. This models the noise from the receiver circuitry.

The filter $H(z)$ is an FIR linear equalizer, which should be designed to mitigate the interference introduced by the channel.

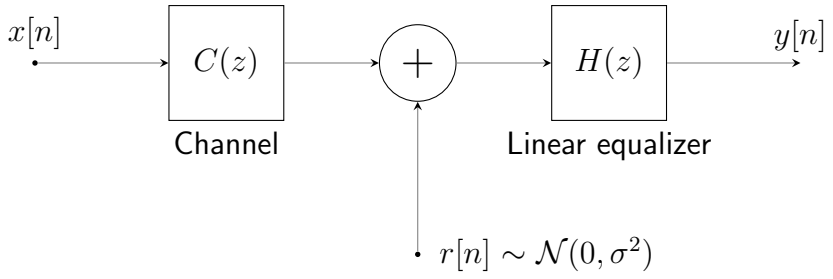


Figure 5: Diagram for linear equalization

For this problem, we will consider that the channel is the FIR filter given by

$$C(z) = 0.4032 + 0.3992z^{-1} + 0.1976z^{-2} \tag{8}$$

When noise is not significant, the optimal linear equalizer is simply the inverse of the channel:

$$H(z) = C^{-1}(z) = \frac{1}{0.4032 + 0.3992z^{-1} + 0.1976z^{-2}} \tag{9}$$

which is an IIR filter.

In the following questions you will design FIR filters to approximate $H(z)$. In all cases, assume that your FIR filter has order $M = 8$ (9 coefficients).

(a) Window method. Design an FIR filter to approximate $H(z)$ using the window method. Since $H(z)$ does not have discontinuities, you can use the rectangular window without problem. On the same graph, plot the frequency response of your FIR filter and $H(z)$. On a different graph, plot the unwrapped phase response of your FIR filter and $H(z)$. Note that they are not linear phase.

Hint: use the Matlab function `impz` to obtain the impulse response of $H(z)$.

(b) Test your linear equalizer using the function `simple_channel.m`, available on Canvas. This function takes one parameter `h`, which is the coefficients of the FIR filter you designed. The function returns the bit-error rate (BER) (number of wrongly detected bits divided by total number of bits) of this system when the receiver equalizer has coefficients `h`. Hence, `simple_channel(1)` returns the BER of the system without any equalization. Report the BER with and without equalization. For the given noise power, the BER of the equalized system should be close to 0.

(c) Non-linear phase least-squares algorithm. Now design an FIR filter using the least squares algorithm to approximate $H(z)$. Since $H(z)$ has no discontinuities, you can assume that the weight function is $W(\omega) = 1 \,\forall\, \omega$. Moreover, since this filter should not be linear phase, you cannot use Matlab's `firls` function. Instead, use the least-squares algorithm as described in class. You may use the script `predicing_bandlimited_signals.m` as a starting point.

On the same graph, plot the frequency response of your FIR filter and H(z). On a different graph, plot the unwrapped phase response of your FIR filter and H(z).

Note: the filter you designed in this part should be (ideally) identical to the filter you designed in part (a), since the rectangular window minimizes the mean-square error (lecture notes 9, slide 33)

(d) Estimate the noise power after the linear equalizer you designed.

Note: you should see that this value is much larger than $\sigma^2 = 0.001$. This happens because this equalizer tries to force ISI to zero at all costs, even if it enhances the noise. This type of equalizer is called *zero-forcing linear equalizer*. A better choice of equalizer is called the *minimum mean-square linear equalizer*, which tries to minimize ISI without excessively enhancing the noise.