Lab 4

Linear Systems and Digital Filtering

0. Preface

Any operation on a digital signal can be described as a filtering operation. In the first lab you explored the moving average. This is in fact a digital filter! In the third lab you saw that in order to properly change the sampling rate of a digital signal, with either upsampling or downsampling, and where the word "properly" signifies not only politeness but also the absence of aliasing, a digital lowpass filter is necessary. In this lab you will explore digital filters in the time and frequency domains. You will learn about linear systems, and finite and infinite impulse response filters.

Really, in constructing a lab on digital filters, it is difficult to limit the subject material. So vast is the subject that one could take several courses on it and still have more to learn. Even so, we hope the sampling provided below will interest you in at least devoting a few hours to completing this lab satisfactorily, and warm discussions around the fireplace during the break between semesters.

Your lab report should answer all questions in all sections. Good luck with that.

1. Linear Systems, Linear Filters: The Warm-up

As we all know, the eigenfunctions of linear systems are complex exponentials. This means that if we excite a linear system with the signal $x(t) = e^{jw_0 t}$, the output of the system is guaranteed to be $y(t) = H(w_0)e^{jw_0 t} = |H(w_0)|e^{j(w_0 t + f(w_0))}$. The frequency of the excitation is carried over to the output—i.e., the frequency does not change—but its amplitude and phase are changed by $|H(w_0)|$ and $f(w_0) = \text{angle}(H(w_0))$, respectively. Furthermore, by the definition of a linear system, the same principle can be applied to any linear combination of any number of these eigenfunctions. Now if you are unsure about what you just read, just smile, nod, and assume you will see the light in the next part of the lab.

Recall from the very first lab, the definition of an M-point moving average filter:

$$y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k], \tag{1}$$

where the integer M is usually odd. Indeed, this system so described is a linear system. In fact, it couldn't be otherwise because this section is titled "Linear Systems, Linear Filters." What you (should have) observed in the first lab is that the moving average acts like a lowpass filter. More formally, a lowpass filter suppresses high frequencies and passes low ones. Let's prove this by plotting the frequency response of an M-point moving averager.

1.1 Knowing that an M-point averager has an impulse response of `ones(1,M)./M`, you can find its frequency response. I know you can. For M=[5, 21, 51] plot (using `plot` and not `stem`) on the same figure (`hold on`) the magnitude response versus the *normalized frequency* domain [0:0.5]. Evaluate the FFT using 1024 points, i.e., `fft(h,1024)`. ("Normalized frequency" means that a value of 1 is the sampling rate; and a value of 0.5 is the Nyquist frequency.) Use `gtext` to label each curve. Include you figure and code. It should look something like Figure 1, but of course not nearly as amazing.

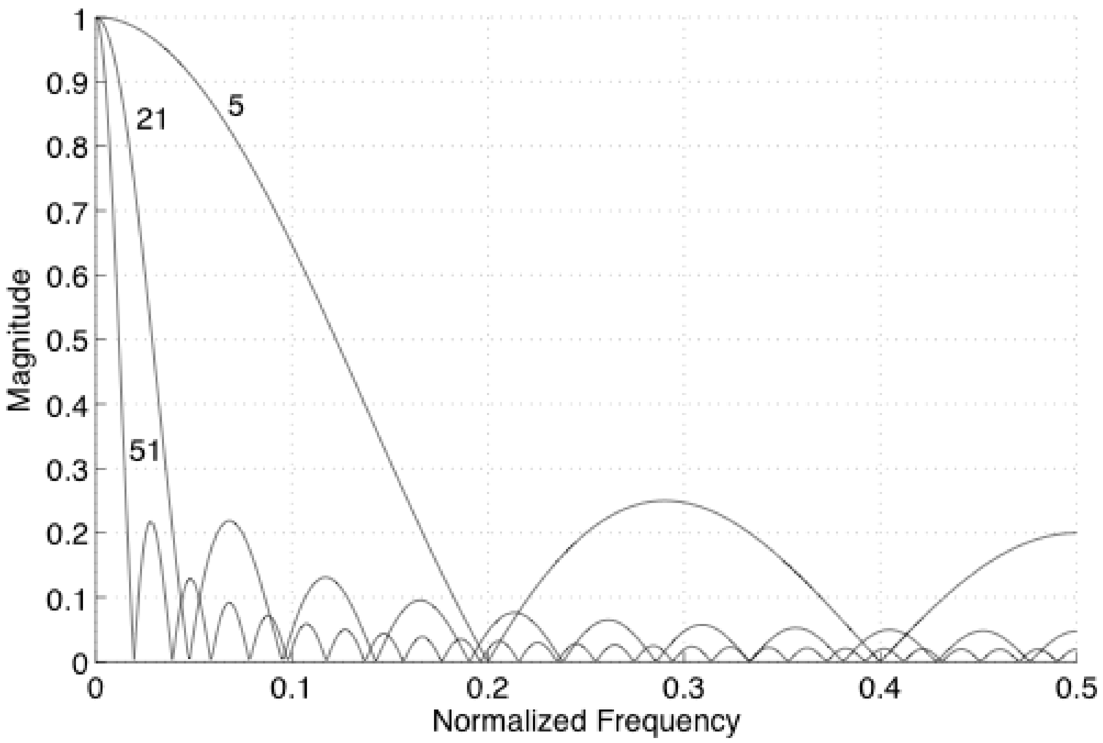

Figure 1: Magnitude responses of moving average filters of different M, labeled

1.2 What happens to the passband of the moving average filter—those frequencies that are passed with only slight attenuation—as the order M increases? What single frequency passes completely unaltered in all three filters?

1.3 For the same values of M, plot on one figure the phase responses in units of pi radians. This means using `angle` instead of `abs`, and dividing the output of `angle` by pi. Again, plot these using normalized frequencies between [0:0.5]. Include your figure, but not your code.

1.4 Can you explain why the phase response has a jagged appearance? Hint: look at your answer to 1.1, specifically the frequencies where the magnitude response becomes zero; and look to see what the jump in pi radians is in your plot. (Hint: if your phase response does not look jagged, seek help.)

A *causal* linear system, or filter, not only attenuates the components of an input—where “component” refers to an eigenfunction (see above)—it also *delays* each component. You saw this in lab #1 when you (should have) noticed that a causal moving average of the significant wave-height data caused the output to be delayed, e.g., the peaks were shifted toward future times. When you modified the moving average system to be non-causal you were able make this delay be zero. In fact, there is no such thing as a causal linear system with zero delay, unless it is a *trivial* linear system, wise guy.

Given the frequency response of a linear system, $H(w)$, the *phase delay* of the system is given by $t_p(w) = -\,\text{angle}(H(w))/w$, which is equivalent to a time-delay. This can be seen in the following example. Consider exciting a linear system by $x(t) = e^{jw_0 t}$. Then the response of the system is expressed as $y(t) = |H(w_0)|\exp[jw_0(t - \frac{-f(w_0)}{w_0})] = |H(w_0)|\exp(jw_0[t - t_p(w_0)])$. Thus phase delay is a measure of the time-shift of a particular component. The *group delay* of the system, given by $\tau_g = -\,d\phi(\omega)/d\omega$, is a measure of the change in phase with respect to frequency. Stated in a less abstract manner, the group delay describes the time-delay of the *envelope* of a signal. In many applications, for instance AM radio, it is desirable to use filters with constant group delay so that the amplitude envelope of a signal, i.e., the information, is preserved.

1.5 Plot the group delay of the moving average filter for each M above, by using the `grpdelay` routine. In this case you will say: `g = grpdelay(ones(1,M)./M, 1, 1024, 'whole')`. Plot the group delay as a function of normalized frequency, and include your plot. It should look like Figure 2.

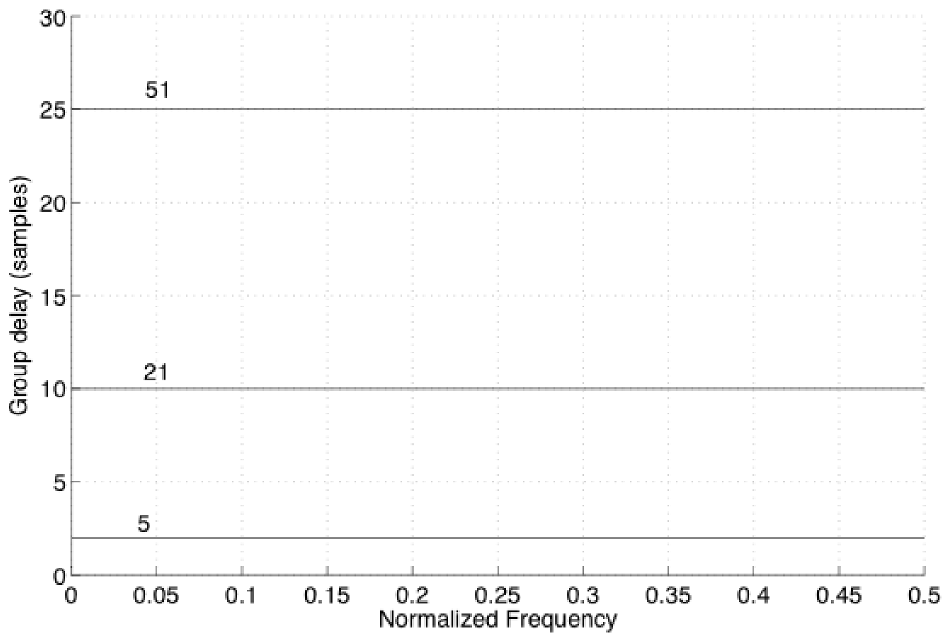

Figure 2: Group delay of moving average filters of different M

Your plot of group delay shows that for the moving average filter, its group delay is constant over all frequencies. Furthermore, the delay introduced into the data is (M-1)/2 samples. Indeed, looking back to lab #1, you observed this exact amount of delay in the envelope of the signals you averaged with the causal moving average filter.

2. Finite Impulse Response Filters

Any *linear* system that has a *finite-length non-zero response* to any *finite-length input* is a *finite impulse response* (FIR) system. (Notice how skillfully I interchange the terms "system" and "filter"; and my playful use of italics and "quotes".) The moving average system is an instance of an FIR system. The most general form for FIR system is given by:

$$y[n] = x[n] * h[n] = \sum_{k=0}^{M-1} h[k]x[n-k] = \sum_{k=0}^{M-1} x[k]h[n-k] \qquad (2)$$

where $h[n]$ is a length-M, $M < \infty$, sequence of real of complex values, which is also called the "impulse response" of the FIR filter. Why does this system have a finite impulse response? Well, you put an impulse in and you will get a finite number of non-zero output. You have already investigated one FIR system: the moving average. Now you will consider more complex FIR systems.

2.1 Consider the following impulse response of a 10-order, length-11 FIR system, as described by (2): $h[n]$={0.00506, 0, -0.04194, 0, 0.28848, 0.49679, 0.28848, 0, -0.04194, 0, 0.00506} for $0 £ n £ 10$. Plot this impulse response with `stem`. Include your figure with appropriately labeled axes.

2.2 Find and plot the frequency response (magnitude and phase), and the group delay, of this system as functions of normalized frequency (between 0 and 0.5). Evaluate the FFT using 1,024 points, i.e., `fft(h,1024)`. (This essentially zeropads the signal with enough zeros until its length is 1,024 samples, and then evaluates the FFT. Zeropadding is not always a good thing though, so get that out of your head!) Plot the magnitude response in dB. (This means something like `20*log10(abs(H)./max(abs(H)))`.) Make sure you unwrap the phase, i.e., `unwrap(angle(H))`. Include your figures and your code.

2.3 What kind of filter is this? Find the “3 dB” point, or the normalized frequency above which the attenuation is greater than 3 dB. This is also called the cut-off frequency. (Hint: it should be a number between 0 and 0.5.) Physically, an attenuation of 3 dB corresponds to a reduction of the power by two, approximately.

2.4 What is the group delay of this filter in samples? Would this filter destroy information contained in the envelope of a signal?

2.5 For the $h[n]$ given in 2.1, multiply it by a series of alternating positive and negative ones, or {1, -1, 1, -1, …}. Now $h[n]$={0.00506, 0, -0.04194, 0, 0.28848, -0.49679, 0.28848, 0, -0.04194, 0, 0.00506}. Find and plot the frequency response (magnitude and phase), and the group delay, of this system as functions of normalized frequency (between 0 and 0.5). Evaluate the FFT using 1,024 points, i.e., `fft(h,1024)`. Plot the magnitude response in dB. Make sure you unwrap the phase, i.e., `unwrap(angle(H))`. Include your figures, but not you code since it will basically be the same as 2.2.

2.6 What kind of filter is this? What is the cut-off frequency now?

You only changed the sign of one value in the impulse response of a system and this is what happens! The former lowpass filter starts attenuating the lows and passing the highs. Its magnitude and phase responses appear to be mirror images of those of the original lowpass filter, found in 2.2. Its group delay remains the same however.

3. Infinite Impulse Response Filters

Any *linear* system that has an *infinite-length non-zero response* to any *finite-length input* is an *infinite impulse response* (IIR) system. (Notice how skillfully I once again interchange the terms "system" and "filter"; not to mention my adorable use of italics and again with the "quotes"?) Essentially, an IIR filter is a linear recursive system. The most general form of an IIR filter is given by:

$$y[n] = \sum_{k=0}^{M-1} b[k]x[n-k] + \sum_{l=1}^{L-1} a[l]y[n-l] \tag{3}$$

One can immediately intuit that the inclusion of past output will make a more "responsive" filter, but could also lead to instability if the output is amplified at each step.

Consider the following IIR system:

$$\begin{aligned} y[n] = {} & 0.206572x[n] + 0.413144x[n-1] + 0.206572x[n-2] \\ & + 0.369527y[n-1] - 0.195816y[n-2] \end{aligned} \tag{4}$$

3.1 Using `filter`, find and plot the first sixteen samples of impulse response of this system. Include your plot and code.

3.2 Using the code you made in 2.2, plot the frequency response and group delay for this system. This time however, use `freqz` to find the frequency response. You may still use `grpdelay`. Evaluate the frequency response at 1,024 points around the whole unit-circle. Include your figure, but not code.

3.3 Compare these results with those found for the FIR system described in 2.2. What are the differences? For instance, how do the cut-off frequencies compare? What is different about the group delay? How does the computational complexity of this filter compare to that of the FIR?

Ironically, one of the most versatile digital filters, and probably the most employed digital filter, is an IIR filter that passes **all** frequencies without changing their magnitudes. I know. When I first heard about the *allpass* filter, a filter that passes all things, I thought, “Now that is the stupidest thing I have ever heard of.” Well, the joke was on me pal; and I can still hear people laughing.

The z-transform of a first-order allpass filter is given by:

$$A_0(z) = \frac{a + z^{-1}}{1 + az^{-1}} \tag{5}$$

3.4 Using MATLAB, confirm that the transfer function given by equation (5) is an allpass filter. Use α = -0.9, and `freqz` to calculate the frequency response. Like in 2.2 and 3.2, plot the magnitude, phase response, and group delay of this filter over the normalized frequency range $0 \le f \le 0.5$. This time do not plot the magnitude response in dB. Include these plots in your lab report, but not the code.

Perhaps you now see one use of the allpass filter. Instead of attenuating any components, it delays them. This is one application of allpass filters: equalizing the group delay of another filter to become constant in the passband. In this way the information in an amplitude modulated signal can be preserved even after distortion by some filters, for instance.

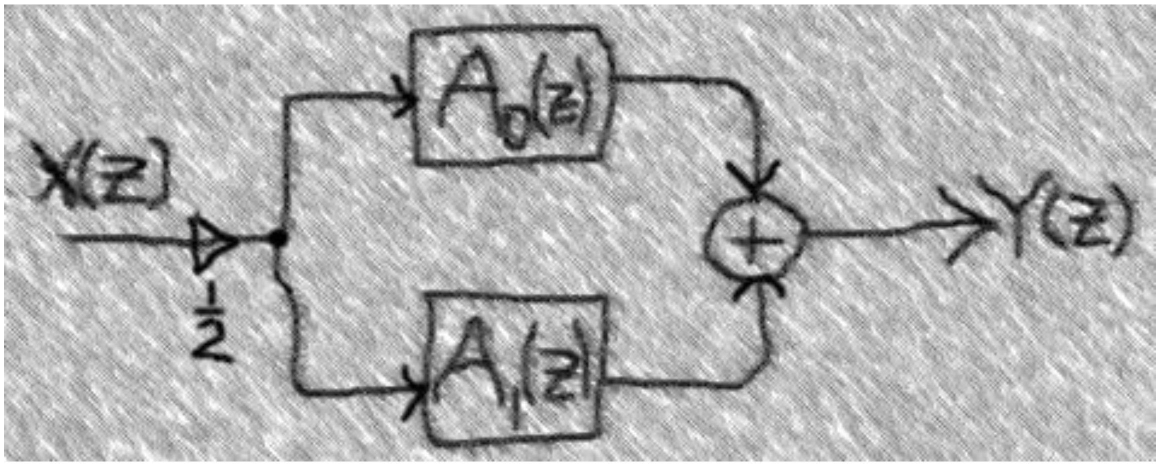

Figure 3: "Untitled #7 (two allpass filters, in parallel, in the z-domain)".

3.5 Now for our last trick, we will make something from nothing. Consider the two allpass filters in parallel artfully shown in Figure 3. With $A_0(z)$ described by (5) with α = -0.15, and $A_1(z)$ = 1, plot the frequency response and group delay of this system, just like you did in 2.2 and 3.2. Make sure your magnitude plot is in dB this time. Include your code and plot.

3.6 How does this lowpass filter compare with those created in 2.2 and 3.2? (Hint: if you are not getting a lowpass filter, again, seek professional help.) For instance, how do the cut-off frequencies compare? What is different about the group delay? How does the computational complexity of this filter compare to the other two?

And now you see another use of allpass filters: they can be combined in parallel to make filters of any kind! If you are like me though, the idea of putting two allpass filters in parallel and not getting an allpass filter as a result makes your tummy-tum hurt. The same frequency component is passed through both filters individually with no change in amplitude! How then can the sum produce a lower amplitude component?

The answer is, of course, described by that famous Brit, William Shakespeare, in his play *Much Ado about IIR*: "Verily, forgeteth not the importance of phase and group delay; for an allpasseth filter may not changeth an amplitude, but verily it changeth a phase!"

3.7 In your own words, interpret Shakespeare's line to explain why two allpass filters in parallel may not an allpass filter make.

3.8 Take a moment to reminisce with friends (Figure 4) about that crazy experience. This includes looking back at your answers while sipping tea and confirming that you understand now what was happening.

Figure 4: Two former DSP students reminisce about the crazy times when they didn't realize the significance of group delay, and all that.