Mini-Project #2: Averaging and Nulling FIR Filters

Neev Mehra & Oscar Portillo

1. Introduction

Filters are an essential part of linear systems and signals since they modify input signals to produce variable responses. Typically, the filter modifies certain components of signals and decides which waves to pass through, allowing people to extract important information while eliminating noise[1]. Specifically, we will be focusing on Averaging and Nulling filters in this project. Averaging filters are one of the most commonly used filters in practical applications and they work by averaging a set of input signals over a specified window and then producing an output signal.

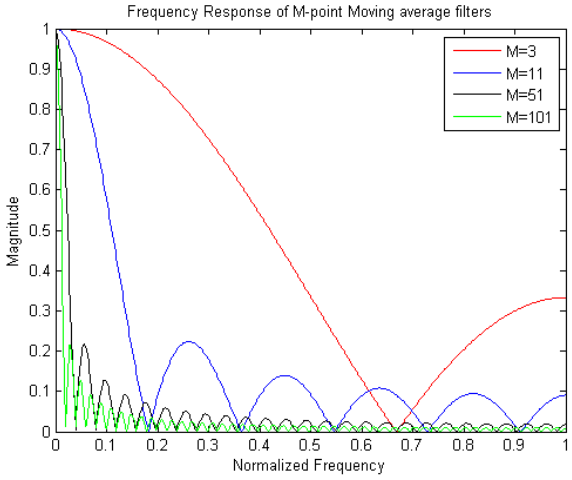


Exhibit 1.0: Low-Pass Filter Analysis

This is really good for reducing noise and they also function as low-pass filters, in that signals with lower frequencies “pass-through” while higher frequencies get attenuated. Furthermore, nulling filters are used to attenuate specific frequencies in a signal[2]. The general nulling filter

accomplishes this through the following coefficients: [1 -2cos(f0) 1]. If there is some specific interference in a signal as we're dealing with in our project, we can attenuate those signals and produce a filtered result without the unwanted noise.

1.1 Objective

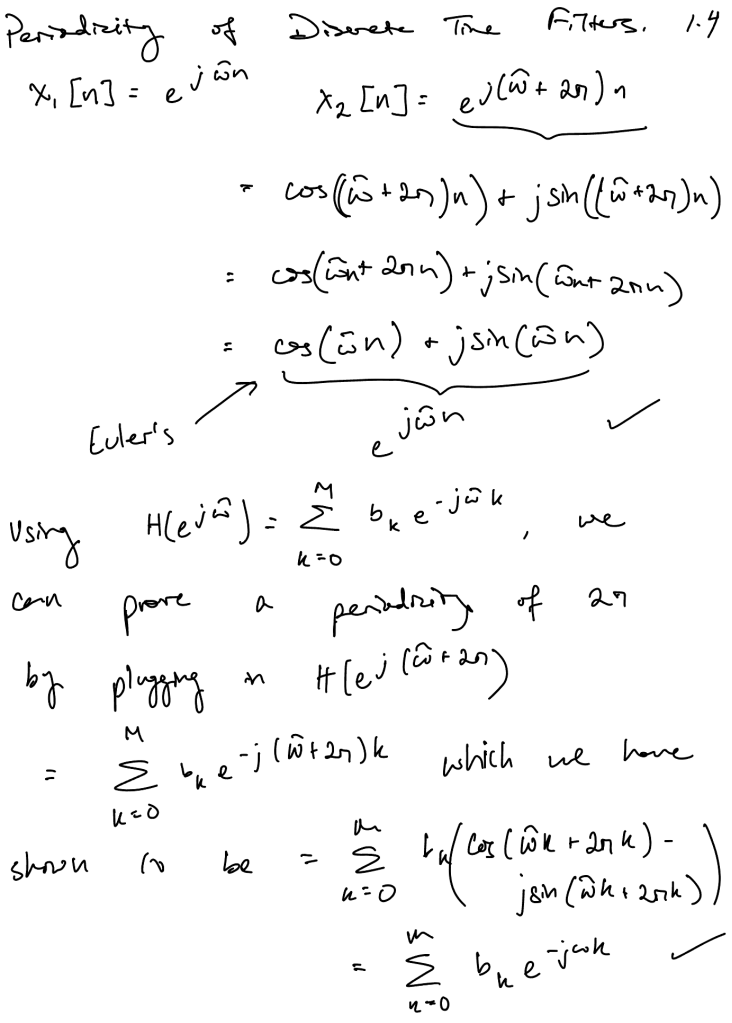
In this experiment, we heavily rely on MatLab's dltidemo GUI to evaluate the responses of FIR filters. This function illustrates the important properties of LTI systems - Sinusoid-In gives Sinusoid-Out property. The property states that when a signal goes through a filter, the output stays the same frequency as the input, but the amplitude and phase are modified. The demo provides interactive elements like a theoretical answer option that shows what the filter would result in ideally, but then also produces practical results which we heavily analyze. This makes it easier to understand the relationship between the filters and the sinusoids.

1.2 Overview

We study the frequency response in this project. We use the function freqz() to analyze the frequency response of the sinusoidal waves in the frequency domain. The responses we observe for filters are through complex exponentials and other sinusoidal functions. Through this, we will obtain an output that showcases the responses of filters. The output or response of the input e^{jwn} will be another exponential of the same frequency. The magnitude and phase will change depending on the frequency.

1.4 Periodicity of the Frequency Response

The frequency responses of discrete-time filters are always periodic with period equal to 2*pi. This is always the case as we utilize the definition of frequency response and use an input of H(e^j(w+2pi) which results in the corresponding summation of exponentials[3]. Using Euler's formula, we are then able to convert the exponential into a sum of sinusoidal waves. Applying the periodic property of sinusoidal waves, and then converting the waves back into exponentials, we notice that H(e^jw) provides an equivalent definition, therefore proving the periodicity of frequency responses in discrete-time filters.



1.5 Frequency Response of the Four-Point Average

a) a) Use Euler's formula and complex number manipulations to show that the frequency response for the 4-point running average operator can be written as:

$$H(e^{j\hat{\omega}}) =$$

$$\frac{1}{4}\left(x[n] + x[n-1] + x[n-2] + x[n-3]\right)$$

$$\frac{1}{4}\left(e^{-j\omega 0} + e^{-j\omega(1)} + e^{-j\omega(2)} + e^{-j\omega(3)}\right)$$

$$\frac{1}{4}\left(1 + e^{-j\omega} + e^{-2j\omega} + e^{-3j\omega}\right)$$

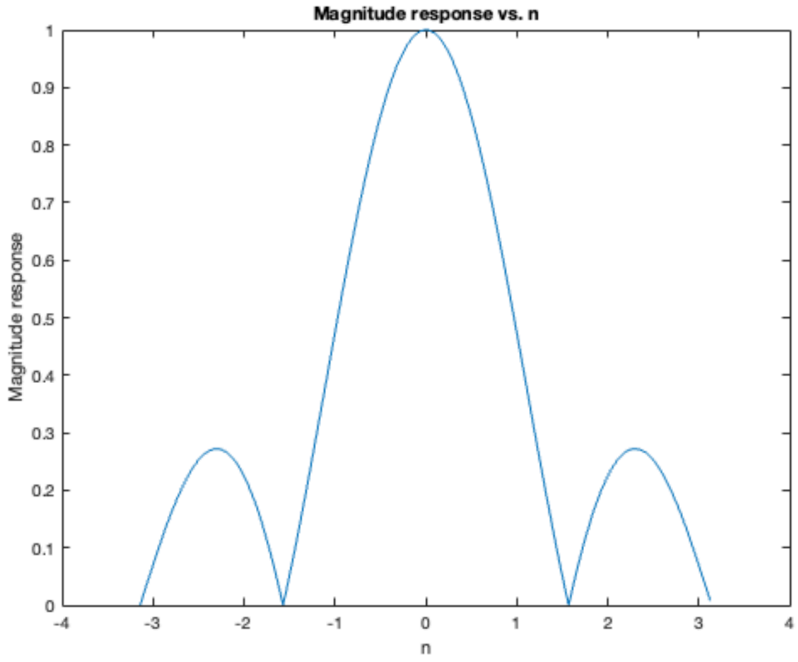
$$\frac{1}{4}e^{-j1.5}\left(e^{j1.5\omega} + e^{j0.5\omega} + e^{-j0.5\omega} + e^{-1.5j\omega}\right)$$

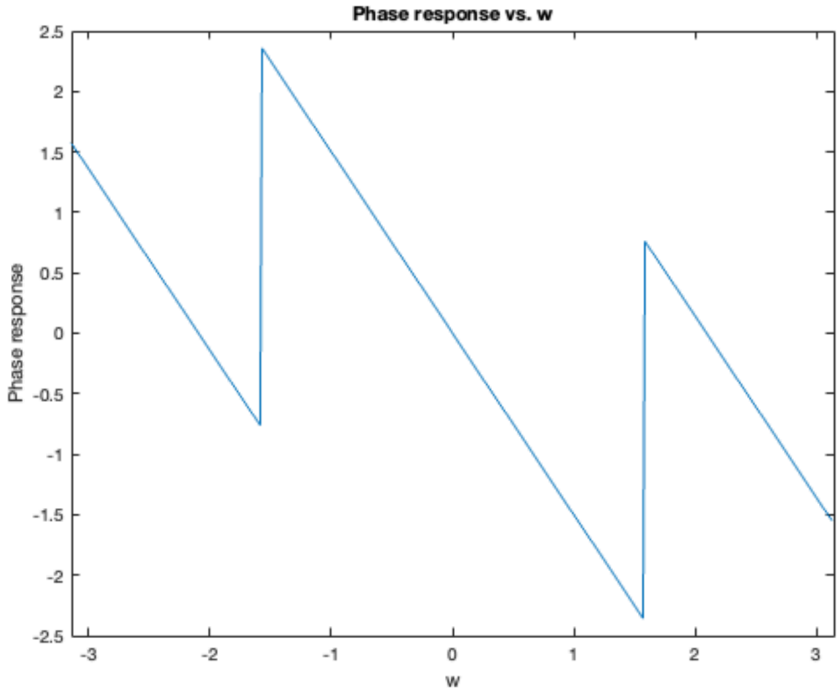
$$\left(\frac{1}{4}e^{-j1.5\omega}\left(2\cos(0.5\omega) + 2\cos(1.5\omega)\right)\right)$$ ✓

b) Implement 5) in MatLab. Use a vector that uses 400 samples. Make plots to display this.

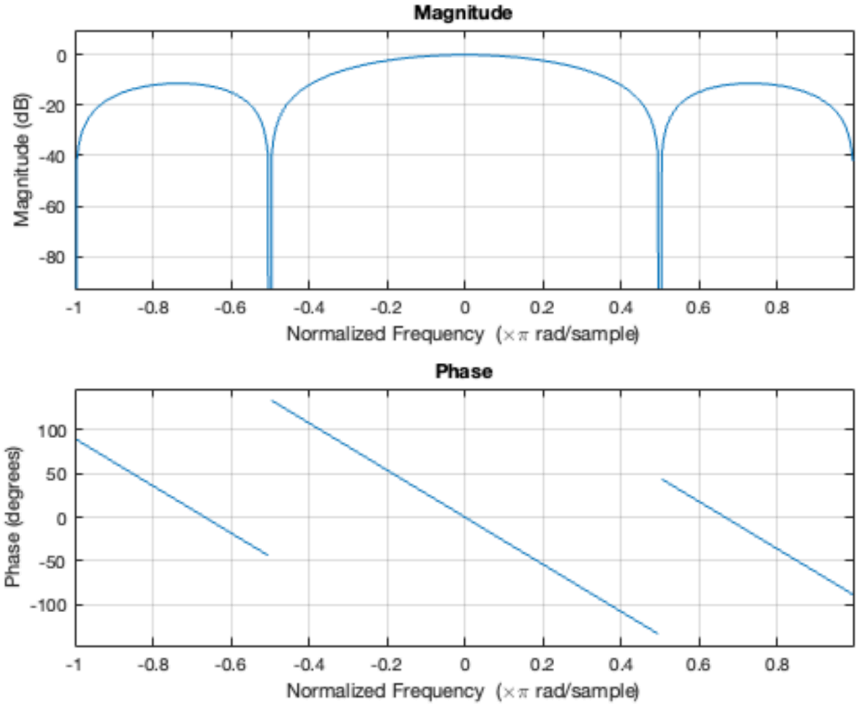
- We used a range of -pi to pi - pi/200 because that will create exactly 400 samples and avoids the MatLab property of including one extra sample in the range. Our code is as follows and we plot our signal relative to the absolute value of our since C(w) can go negative.

```
w = -pi : pi/200 : pi-pi/200;
%h1 = 1/4 * [1 1 1 1];
H = (1/4) * exp(-1.5*j*w) .* (exp(-1.5*j*w) + exp(1.5*j*w) + exp(-0.5*j*w) +
exp(0.5*j*w));plot(w, abs(H));
xlabel( 'n' );
ylabel('Magnitude response');
title('Magnitude response vs. n');

figure;
plot(w, unwrap(angle(H)));
xlabel( 'w' );
ylabel('Phase response');
title('Phase response vs. w');
xlim( [-pi, pi] );

bb = 1/4*ones(1, 4);
freqz(bb, 1, w);
```





c) Use freqz.m to compute H(e^jw) numerically and plot its magnitude. As shown in the aforementioned code, we used the filter coefficient vector of the four point averager which is defined as bb = 1/4 *ones(1, 4); which simply is a four length vector containing [¼ ¼ ¼ ¼] as defined in a basic four point averaging filter. We then plotted the phase response alongside the magnitude response. As expected, our phase was a discontinued plot caused by freqz not properly accounting for the negative values obtained in this calculation.



% These are not the same. The magnitude is different because freqz doesn't
% account for the magnitude and phase correctly, as it doesn't take absolute value. You'd
% need to convert the minus sign of negative cosine values to a phasor with phase pi radians,
% which you would then integrate into the phase plot, with resultant jumps
% of +-pi radians.

1.6 FIR Nulling Filters

Verification that the frequency response is 0 at $\omega_{null}=0.75*\pi$, compute and plot the poles and zeros for the three coefficient nulling filter in terms of a null frequency at 0.75π. As we see in our plots, when nuling a frequency at 0.75pi with the specified filter coefficients, the magnitude at 0.75pi is 0. This is shown in the magnitude response of the filter. Furthermore, we observe the same response with the input defined as 0.4 + 1.5(0.75*pi*n).

% 1.6 extra:

% Pole -> w = 0 (no point where ampltiude response is infinity (denominator

% is just 0), but at w=0, we have a pole, as this leads the denominator in the z-domain to be 0. The nulled frequencies (corresponding to zero magnitudes, as

% shown in the magnitude response plot & dltidemo) are on the unit circle

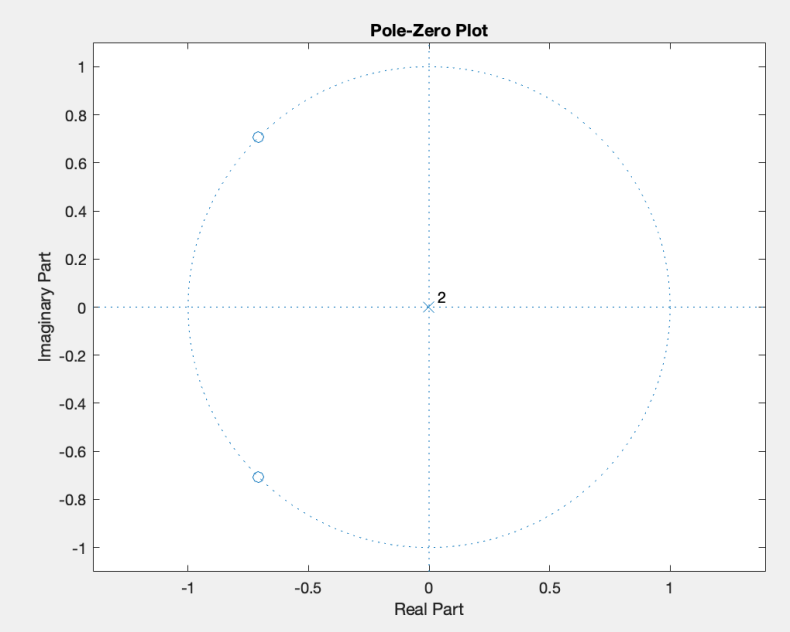
% at their respective frequencies.

% Magnitude Response Plot:

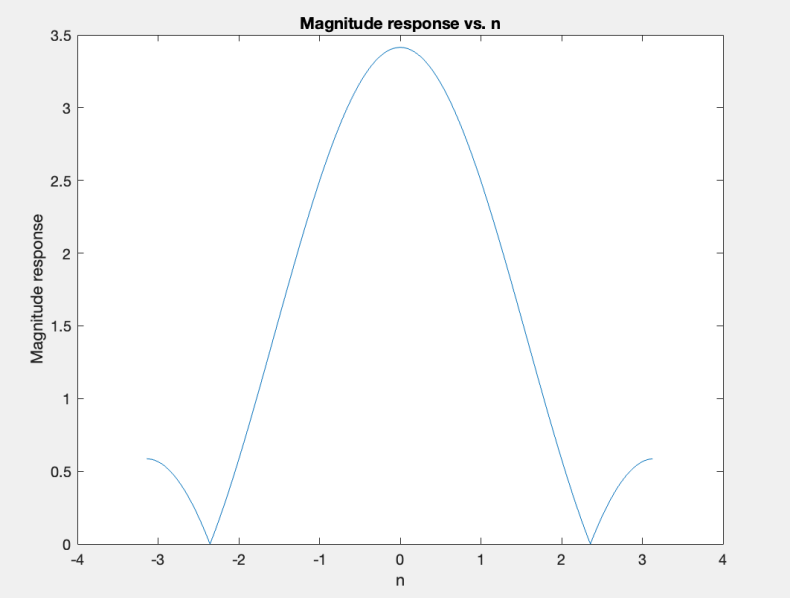
% Zeros at w=-w_null,w_null in the frequency domin, as indicated as well in

% the dltidemo. For time domain, these zeros correspond to the following:

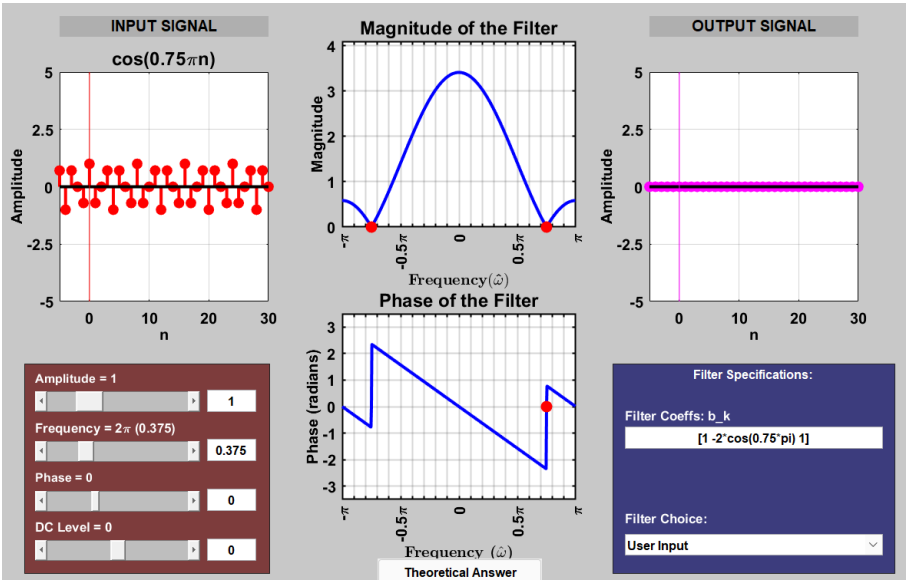
(f=-w_null/2*pi,w_null*2*pi)



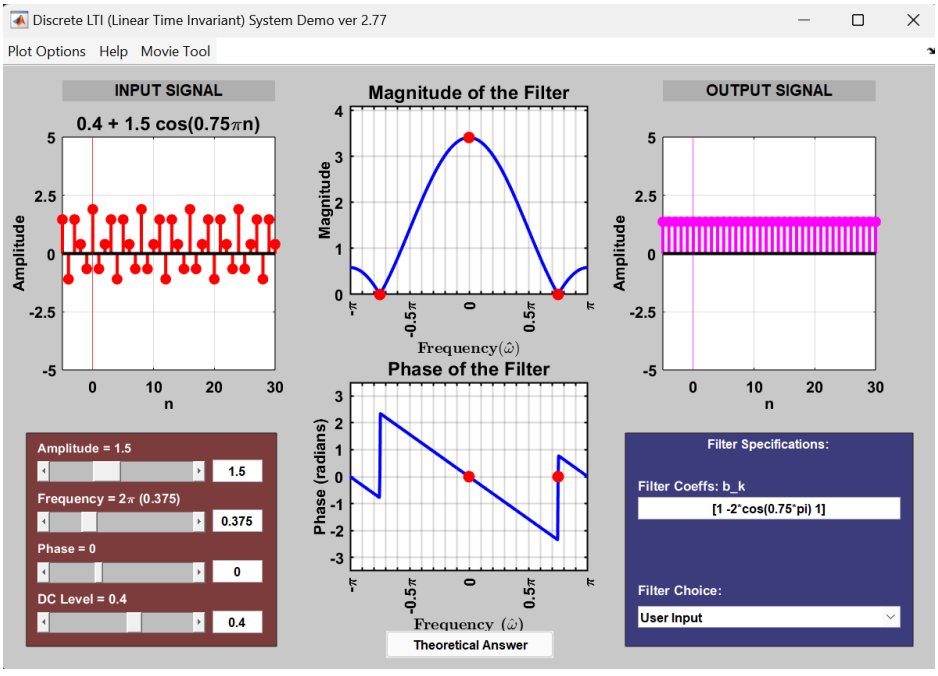
Pole Plot for the three-coefficient nulling filter (Located at -0.75)



Manually calculated magnitude plot of the nulling filter effects



Input defined as cos(0.75*pi*n)

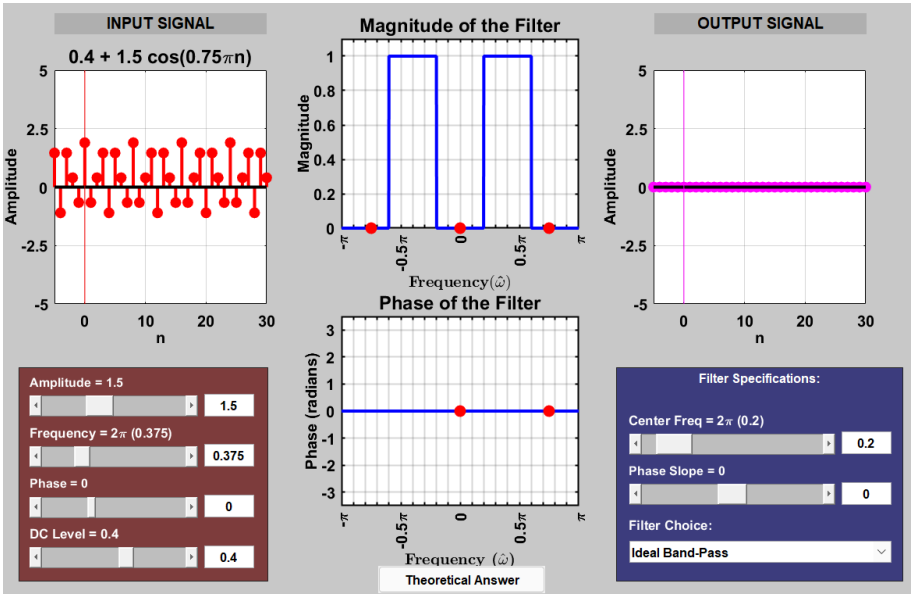


Input defined as 0.4 + 1.5cos(0.75*π*n)

1.7 Ideal Filters and Practical Filters

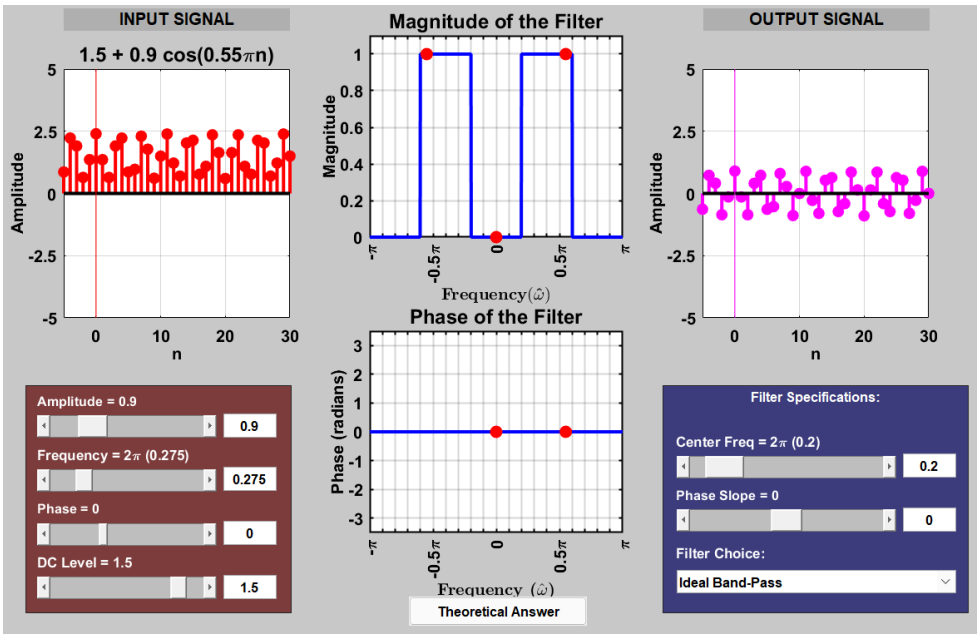
Ideal Filters

(a) Use the dltidemo to view the sinusoid-in gives sinusoid-out behavior of an ideal bandpass filter (BPF). Choose the center frequency to be 0:4. Then the two bandedge frequencies for the BPF



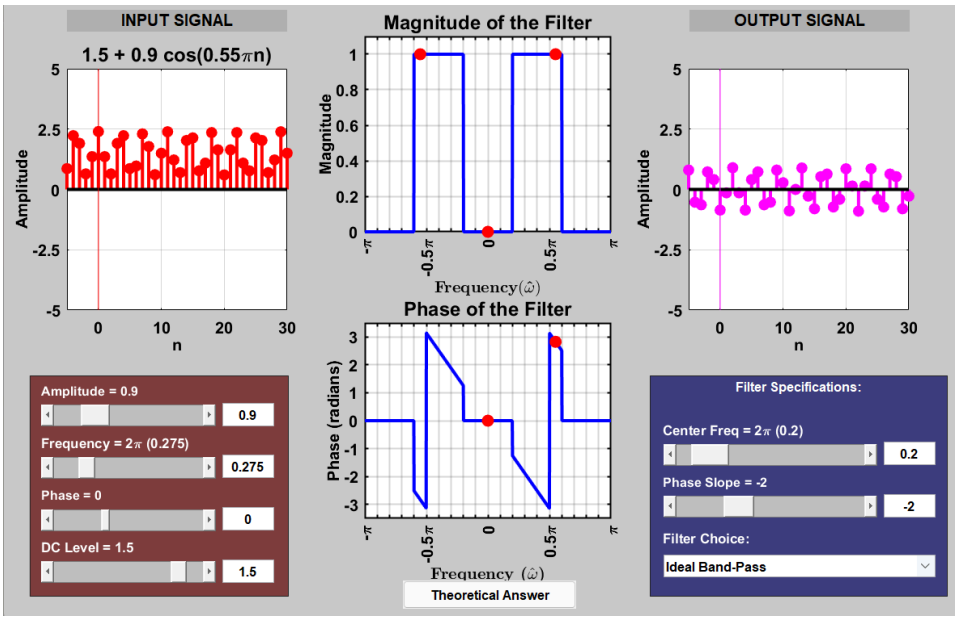
Dlti-demo showing the desired bandpass filter, entirely nulling the input signal.

(b) Set the input to be xn= 1.5 + 0.9cos(0.55*pi*n). Determine which frequency components are present in the output signal.



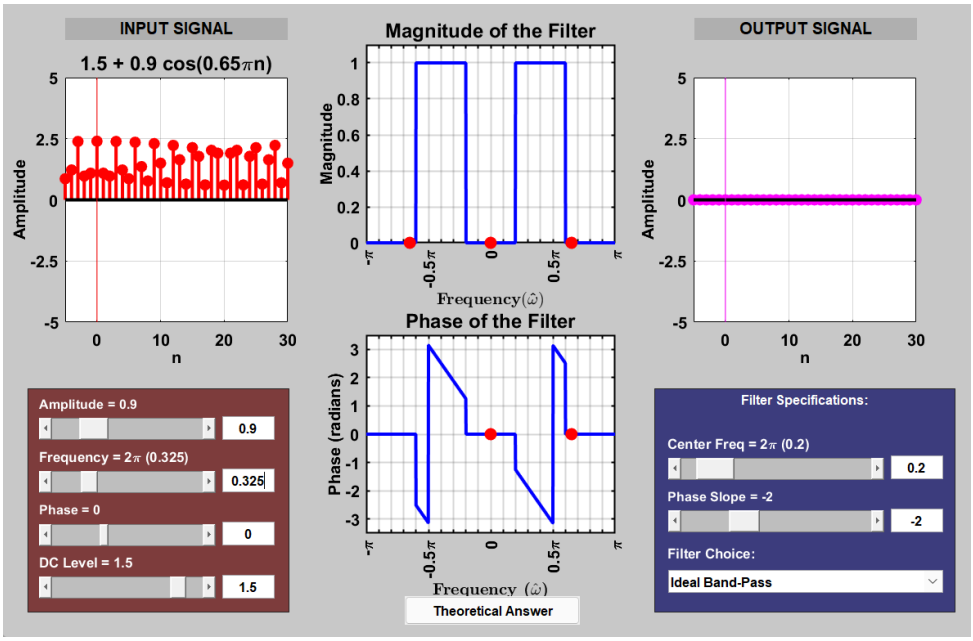
Dlti-demo response, Only frequencies between 0.2π and 0.6π pass through, so the cosine component passes through, but 1.5 doesn't pass through because its frequency is 0, which isn't within the band pass.

(c) You can also include an ideal linear phase in the frequency response, so choose the phase slope as -2. Then find the output for the same input as in the previous part. Describe how the output has changed; relate the delay in the output to the phase slope of the frequency respons



Dlti-demo with the phase slope of -2. On top of the effects of the earlier nullification of all frequencies not in the frequency pass band, this introduces a phase slope, leading to a shift two units right relative to the output signal from part b

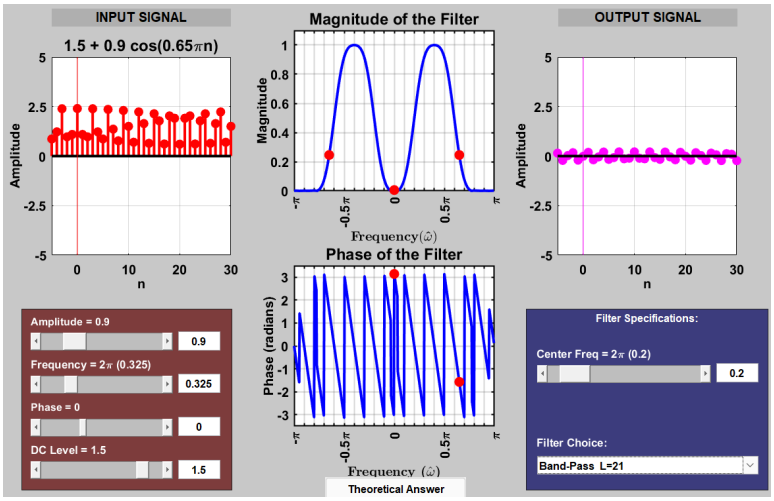
(d) When the input is xn = 1.5 + 0.9cos(0.65*pi*n), determine the output. Explain why it is zero



Dlti-demo response including a new input. The output is 0 because no frequency of the input is passed through the filter(neither the 0 frequency of 1.5 nor the 0.65π frequency of the cosine component).

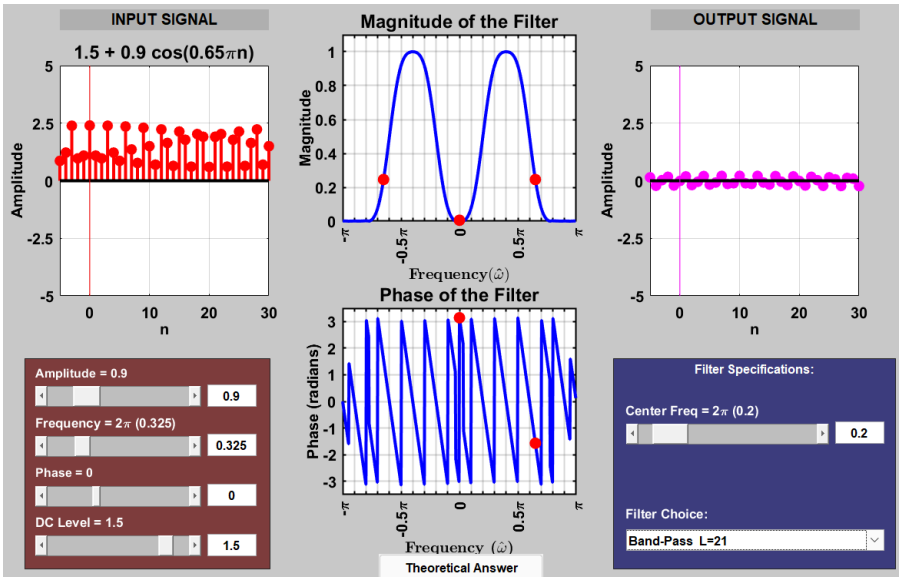
Practical Filters

(a) Use the dltidemo to view the sinusoid-in gives sinusoid-out behavior of an ideal bandpass filter (BPF).



This filter is non-ideal, resulting in the attenuation of the desired frequencies but not the entire nulling effect.

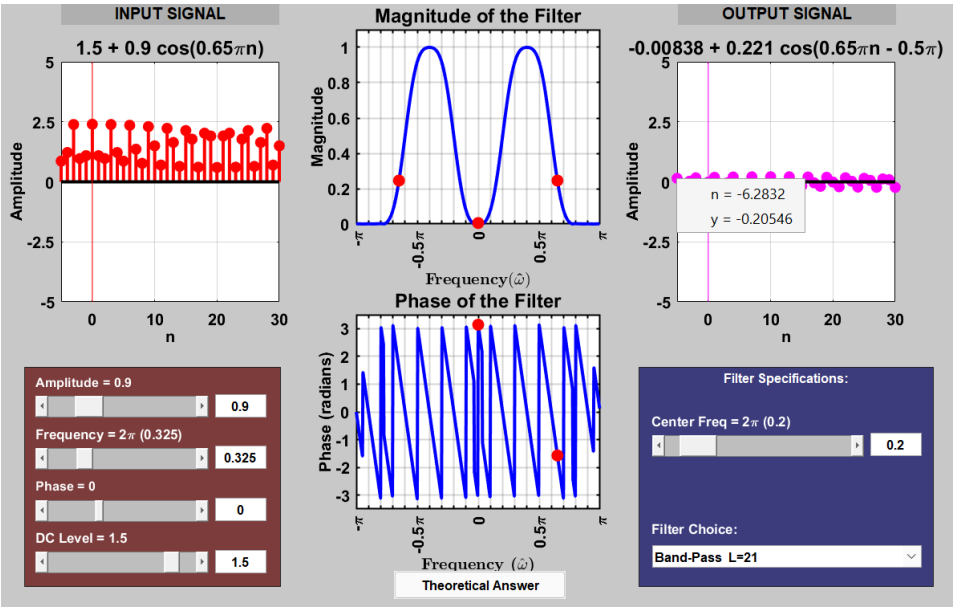
(b) Set the input to b = 1.5 + 0.9cos(0.65*pi*n). Determine the output signal and compare results to earlier.



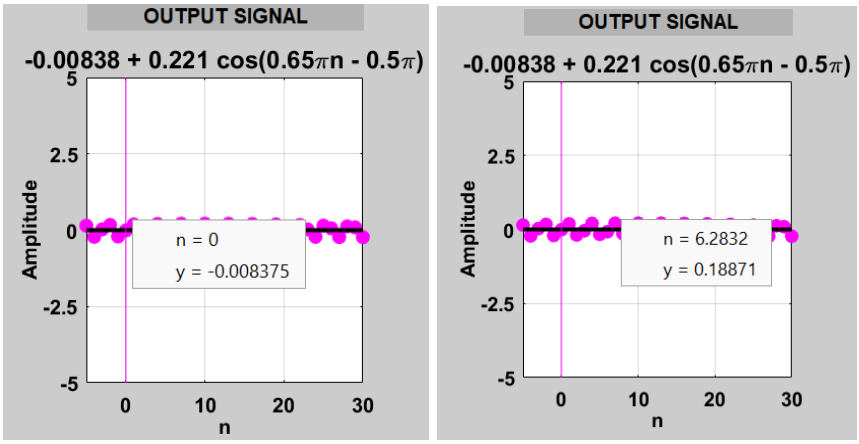
Dlti-demo with the non-ideal band pass filter.

The output is almost zero, but it isn't exactly zero like it is in part d of Ideal Filters, as the magnitude of the frequency response isn't completely ideal (rather than being a perfect square wave-like response, it's a close curved approximation $(\text{realistic})_4$, as is achieved by the band pass filter with L=21). Therefore, the output signal is almost zero but not quite zero, as 0.65π is just outside the 0.6π boundary, so it captures a small, non-insignificant bit of the non-zero tapering off from the curved, practical magnitude response component of the frequency response, as we can see in the "Magnitude of the Filter" graph. Additionally, the same applies, though to a much lesser extent, with the 0 frequency of the 1.5 DC level in the input signal.

(c) The output signal might contain both frequency components. Relate the amplitudes of the output signal's two frequency components to the non-ideal nature of the frequency response. Right-click to get values from the frequency response plot.



The output signal containing both frequency components due to non-ideal nature

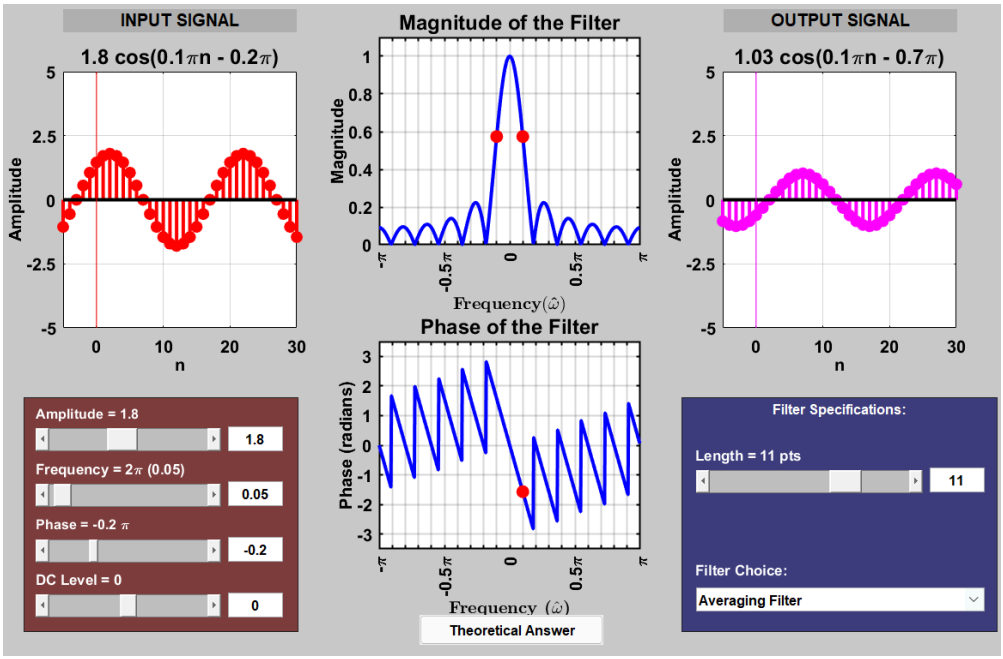


Frequency response plot values

We can see that the amplitudes of the two frequency components are non-zero, reflecting the non-ideal nature of the practical band-pass filter. Moreover, we can see that the magnitude of the 0.65π frequency response is much higher than that of the 0 frequency response, which reflects the curved, non-ideal “Magnitude of the Filter” graph presented. Moreover, we can see this reflected in the actual magnitude values shown below on the Output Signal graph images with right-click.

2.1 LTI Frequency Response Demo

(a) Set input xn = 1.8cos(0.1*pi(n-2)) with an 11 point averaging filter



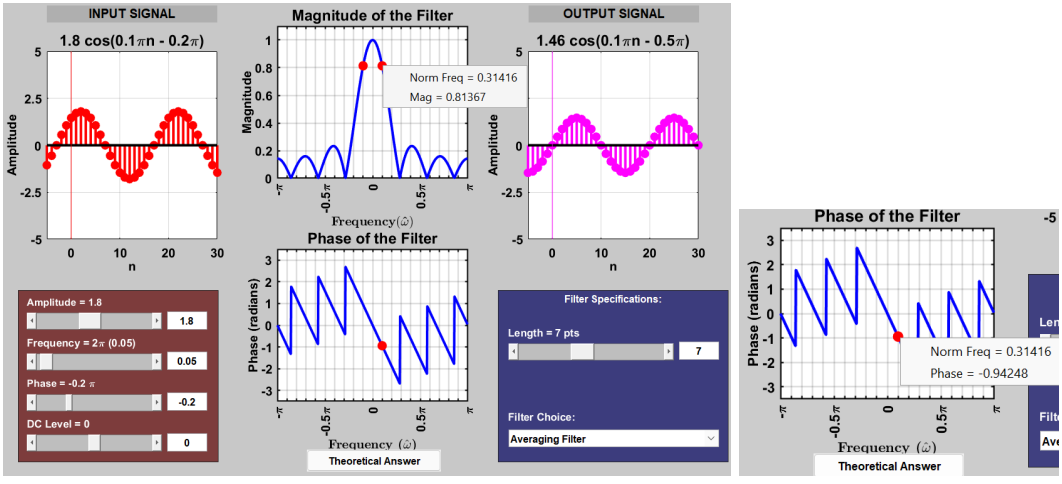
This is the DLTI-Demo of the input being run through 11 point averaging filter

b) Set the digital filter to be a 7-point averager. Using the middle panels that show the frequency response, measure the magnitude and phase of the frequency response at the input frequency. Right-click to get values from the frequency response plot

Frequency Response

Magnitude Response = 0.81367(1.8) = 1.46

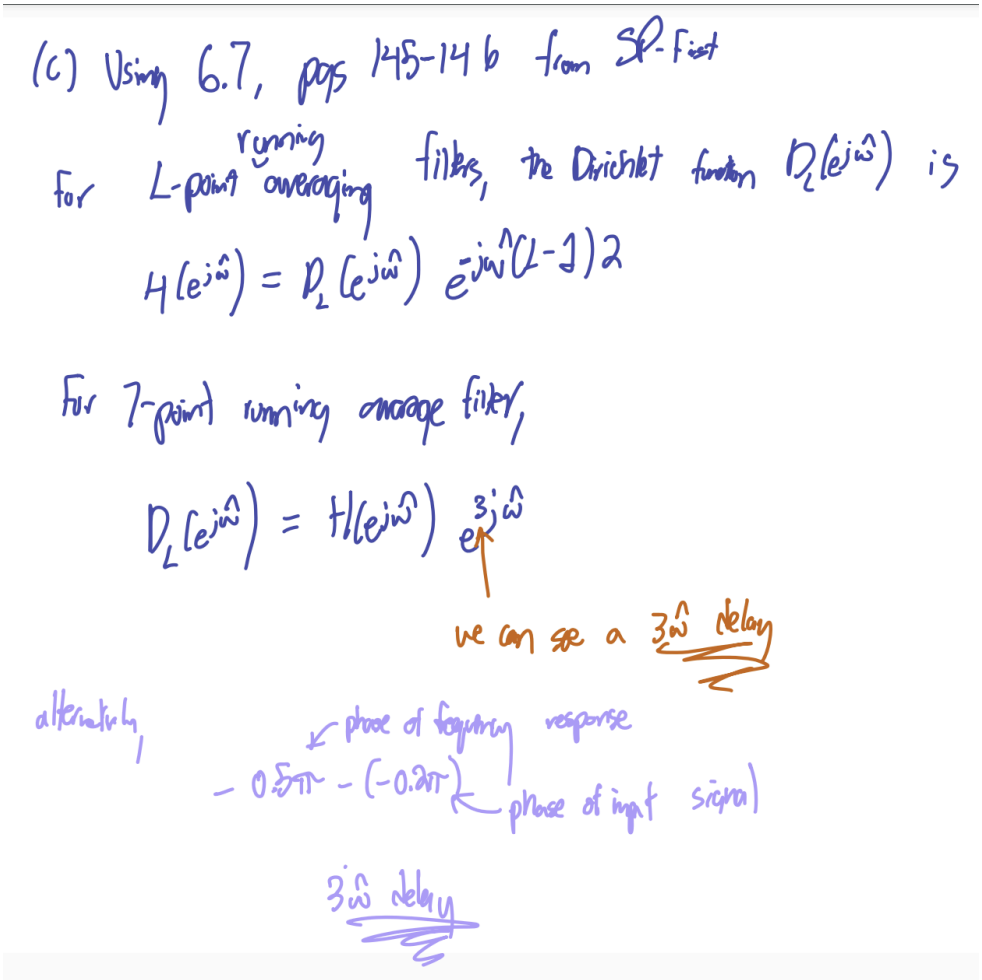
Phase Response = -0.2π -0.94248 = (-0.5π)



The same signal being ran through a 7 point averaging filter with magnitude and phase response

(c) Give an equation that explains how the time delay is related to the phase of H(e^jw) at w = 0.1*pi.

Referring to the calculations below, we see that there is a net delay of 5w (or 5 time units), but when accounting for our starting point, we have a 3w delay (or 3 time units), as indicated by the math below.



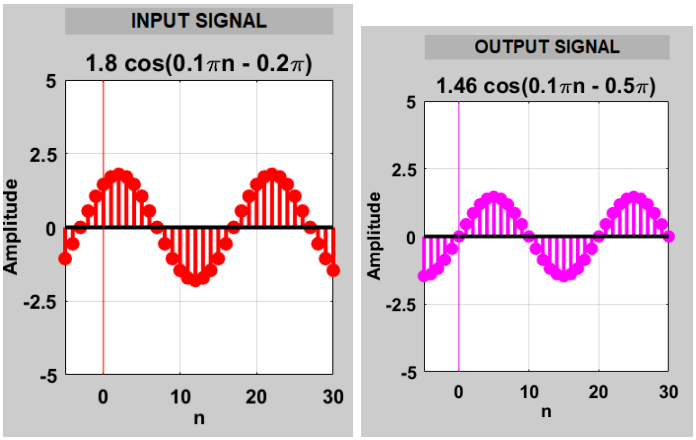
Applying the running-average filtering to our 7 point running average filter to observe a 3w delay

(d) Convert the phase to time-index delay.

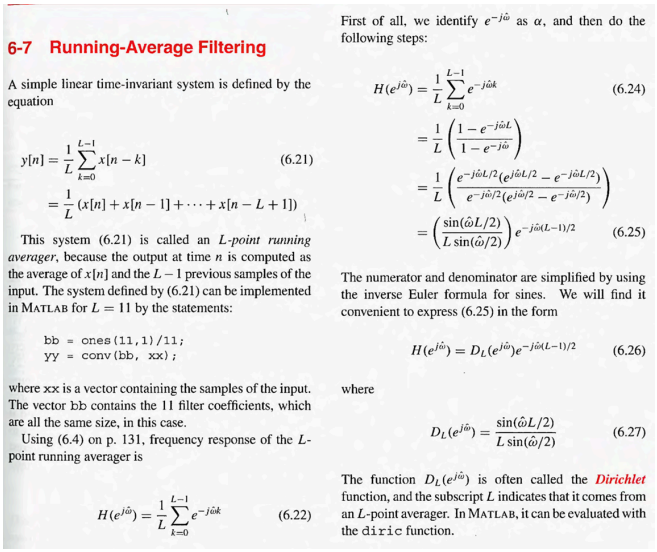
$y = 1.46\cos(0.1\pi(n - 5))$

We can easily verify that the peak has shifted by 5-2=3 units to the right since 2 is our time-index for our original signal.

→ Note that the 5 comes from the frequency response and 2 was the existing shift in the input signal



The difference between input and output signals, factoring out a 0.1*pi gets our unit delays.



6-7 Running-Average Filtering

A simple linear time-invariant system is defined by the equation

$$y[n] = \frac{1}{L}\sum_{k=0}^{L-1} x[n-k] \qquad (6.21)$$

$$= \frac{1}{L}\left(x[n] + x[n-1] + \cdots + x[n-L+1]\right)$$

This system (6.21) is called an *L-point running averager*, because the output at time n is computed as the average of $x[n]$ and the $L-1$ previous samples of the input. The system defined by (6.21) can be implemented in MATLAB for $L = 11$ by the statements:

```
bb = ones(11,1)/11;
yy = conv(bb, xx);
```

where `xx` is a vector containing the samples of the input. The vector `bb` contains the 11 filter coefficients, which are all the same size, in this case.

Using (6.4) on p. 131, frequency response of the L-point running averager is

$$H(e^{j\hat{\omega}}) = \frac{1}{L}\sum_{k=0}^{L-1} e^{-j\hat{\omega}k} \qquad (6.22)$$

First of all, we identify $e^{-j\hat{\omega}}$ as α, and then do the following steps:

$$H(e^{j\hat{\omega}}) = \frac{1}{L}\sum_{k=0}^{L-1} e^{-j\hat{\omega}k} \qquad (6.24)$$

$$= \frac{1}{L}\left(\frac{1-e^{-j\hat{\omega}L}}{1-e^{-j\hat{\omega}}}\right)$$

$$= \frac{1}{L}\left(\frac{e^{-j\hat{\omega}L/2}\left(e^{j\hat{\omega}L/2} - e^{-j\hat{\omega}L/2}\right)}{e^{-j\hat{\omega}/2}\left(e^{j\hat{\omega}/2} - e^{-j\hat{\omega}/2}\right)}\right)$$

$$= \left(\frac{\sin(\hat{\omega}L/2)}{L\sin(\hat{\omega}/2)}\right) e^{-j\hat{\omega}(L-1)/2} \qquad (6.25)$$

The numerator and denominator are simplified by using the inverse Euler formula for sines. We will find it convenient to express (6.25) in the form

$$H(e^{j\hat{\omega}}) = D_L(e^{j\hat{\omega}})e^{-j\hat{\omega}(L-1)/2} \qquad (6.26)$$

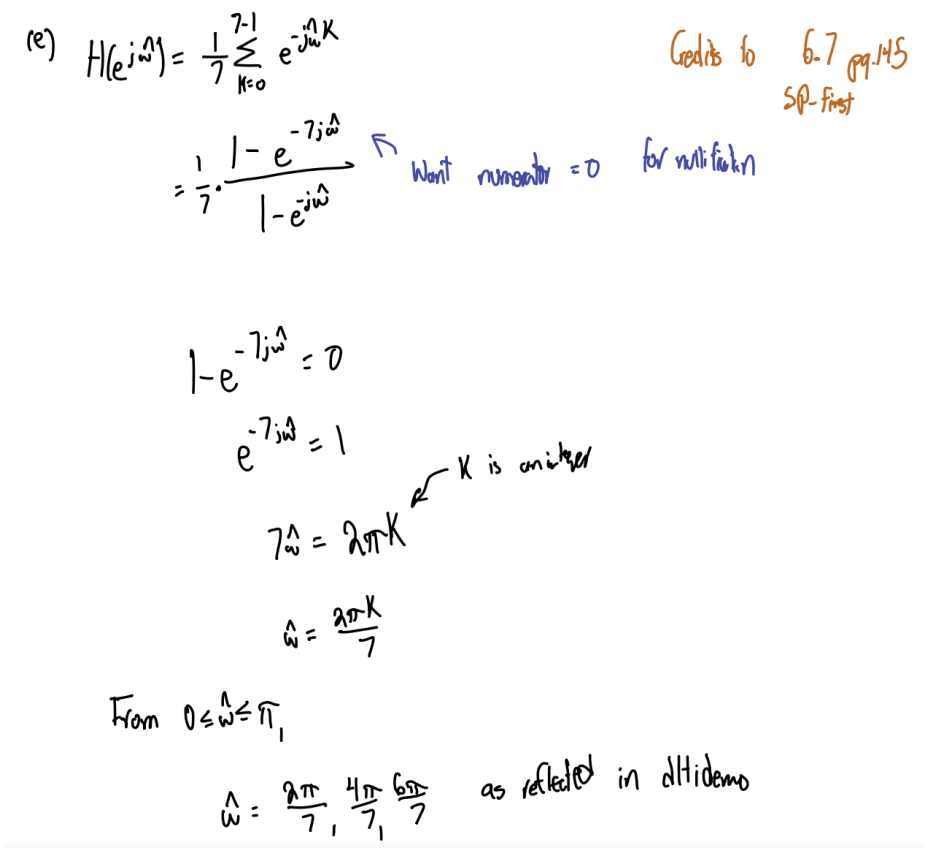
where

$$D_L(e^{j\hat{\omega}}) = \frac{\sin(\hat{\omega}L/2)}{L\sin(\hat{\omega}/2)} \qquad (6.27)$$

The function $D_L(e^{j\hat{\omega}})$ is often called the ***Dirichlet*** function, and the subscript L indicates that it comes from an L-point averager. In MATLAB, it can be evaluated with the `diric` function.

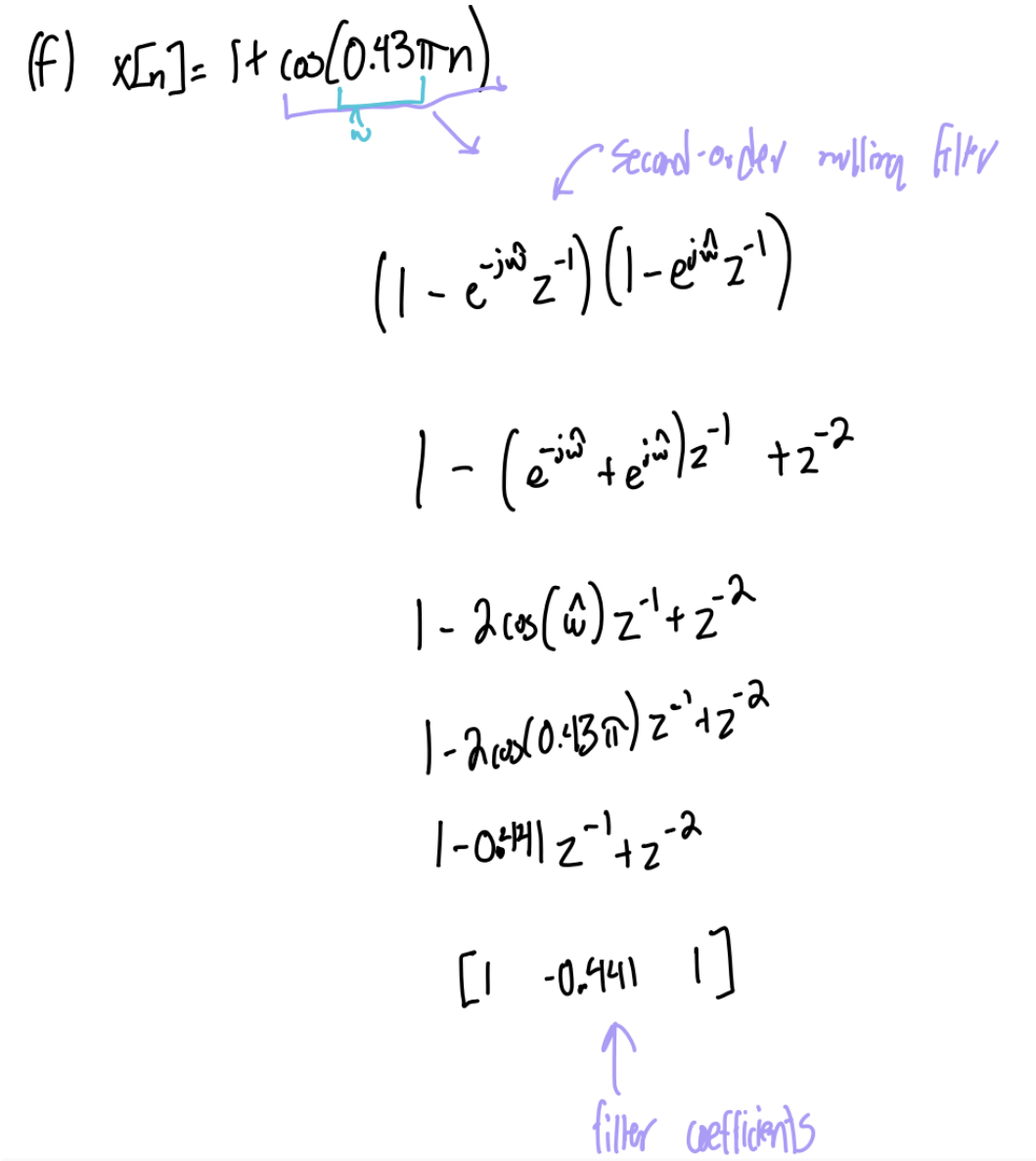
Running average filter derivation/math

(e) Now, change the frequency of the input signal so that the output is exactly zero. Sinusoidal components may also be nulled. Make a list of these frequencies.



Calculation, resulting in a list of frequencies that will be nulled

(f) When the output of an FIR filter is zero, the FIR filter acts as a nulling filter. Design a second-order nulling filter that removes the cosine component from 1 + cos(0.43*pi*n)

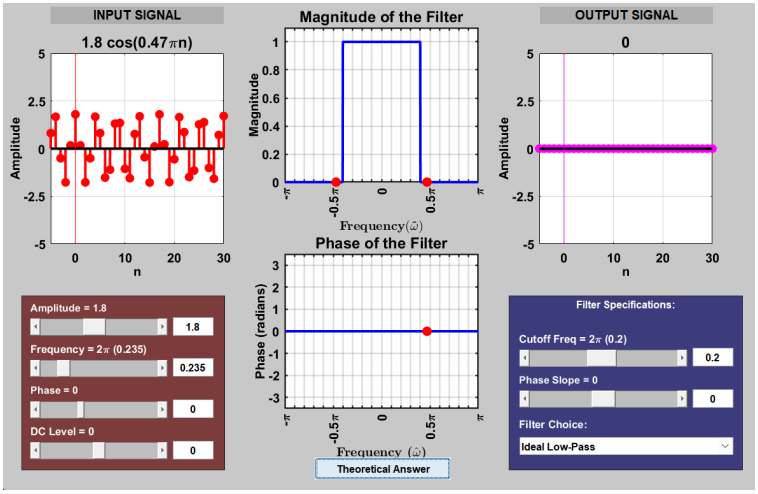


Filter coefficients for a second-order nulling filter

2.2 Ideal Filters and Practical Filters

2.2.1 Ideal Filters

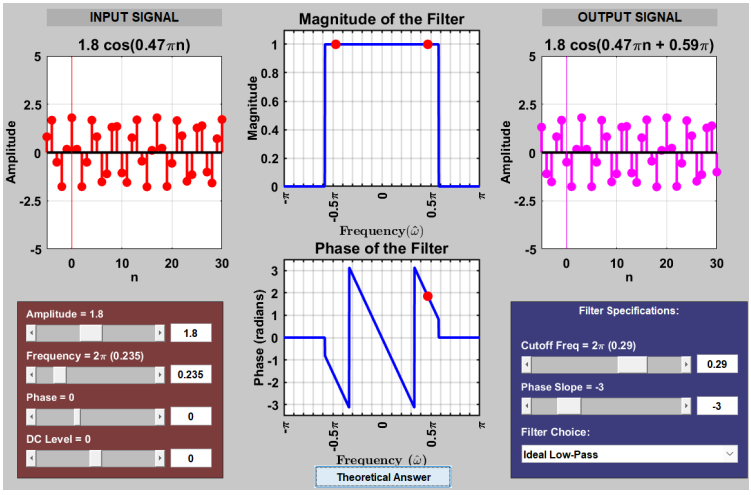
(a) Define the input to be xn = 1.8cos(0.47*pi*n)



DLTI-Demo with an input of 1.8cos(0.47 *pi * n)

(b)

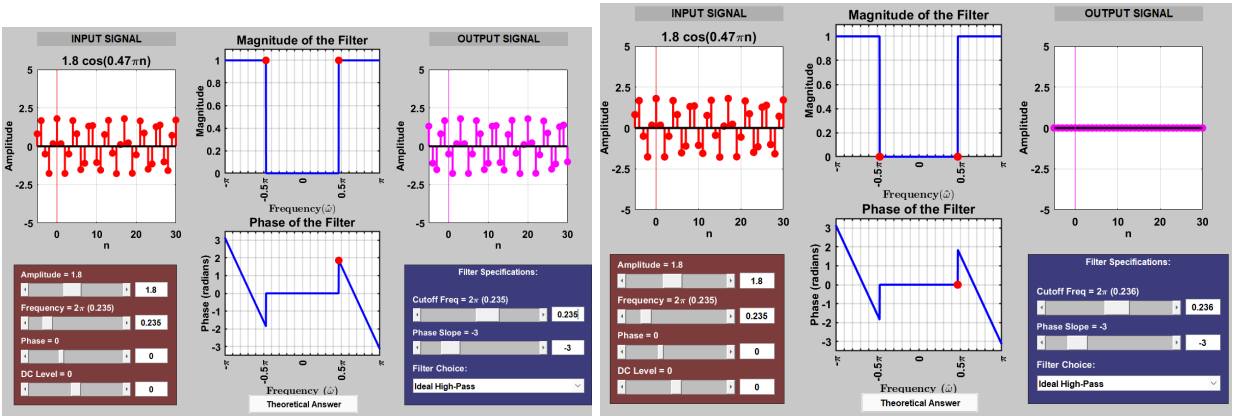
(b) LPF: Set the filter to be an ideal LPF with a cutoff frequency of $\hat{\omega}_c = 2\pi(0.29)$. Determine a value for the phase slope so that the output will be delayed by 3, i.e., $y[n] = 1.8\cos(0.47\pi(n-3))$. In addition, get the formula for the output signal from the *Theoretical Answer*. Explain why the phase of the *theoretical* output signal is not equal to -1.41π. Consider multiples of 2π in the phase which must be taken into account when relating the phase to the delay.



Phase slope of -3 to create a time delay

We utilized a phase slope of -3 to delay the output by 3. The phase of the theoretical output signal is not equal to -1.41π because 2π has been added, thus making it $0.59\pi_6$ (they're equivalent, the phase was just kept between -π and π).

(c)) HPF: Change to an ideal HPF; use the same phase slope as in the previous part. Determine the minimum value of the cutoff frequency so that the output signal is exactly zero

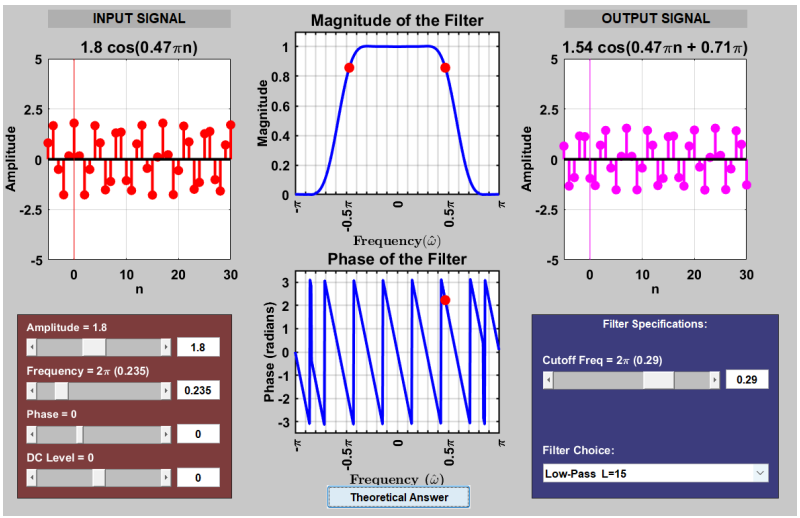


The high pass filter containing the desired cutoff frequency

The minimum value of the cutoff frequency is $\omega = 2*\pi*(0.235) = 0.47\pi$, meaning the frequency must be above this as reflected below (since this makes the cutoff frequency equal to the input frequency).

2.2.2 Practical Filters

(a)) LPF: Set the filter type to a length-15 L, cutoff frequency at 2pi*(0.29)



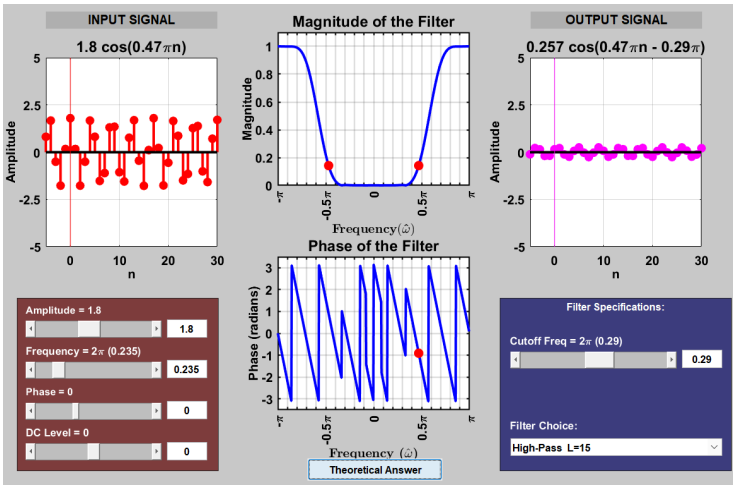
The non-ideal low pass filter, providing slight inaccuracies

The output signal passed by the practical LPF is close, though still non-insignificantly off with $1.54\cos(0.47\pi n + 0.71\pi)$ instead of $1.8\cos(0.47\pi n + 0.59\pi)$

(b)

$1.54\cos(0.47\pi n + 0.71\pi) = 1.54\cos(0.47\pi n - 3.29\pi) = 1.54\cos(0.47\pi(n - 7))$. $n_d = 7$, indicating a 7-time unit delay. $A_{practical} / A_{ideal} = 1.54/1.8 = 0.855\ldots$ This means there's an >14% degradation in signal between the practical and ideal LPFs, which is notable though the practical LPF remains nonetheless useful.

(c)

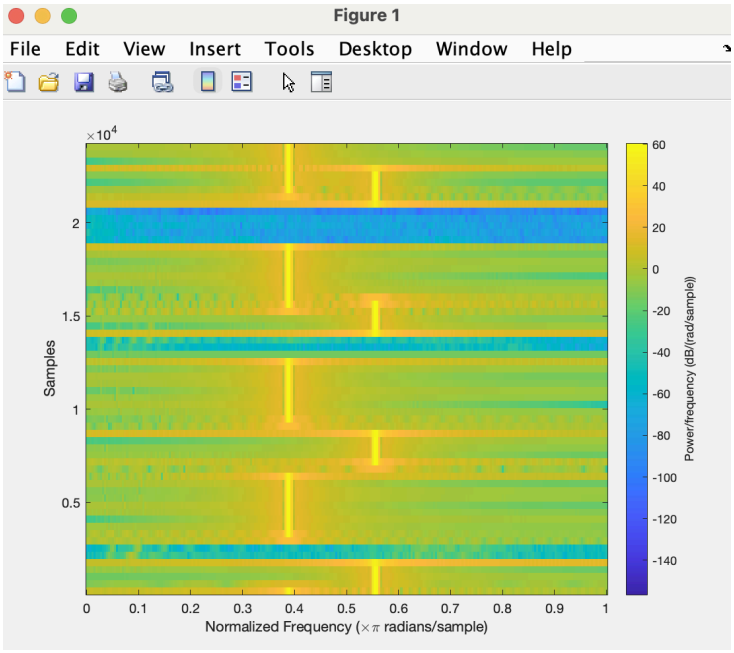


The non-ideal high-pass filter, which almost fully rejected the input signal

Output Signal = $0.257\cos(0.47\pi n - 0.29\pi)$. The amplitude is 0.257, which is pretty close to zero.

2.3 Lab-HW: Removing Interference from a Speech Signal

(a) The speech is unrecognizable due to the interferences overlapping with the normal audio



Spectrogram in decibels

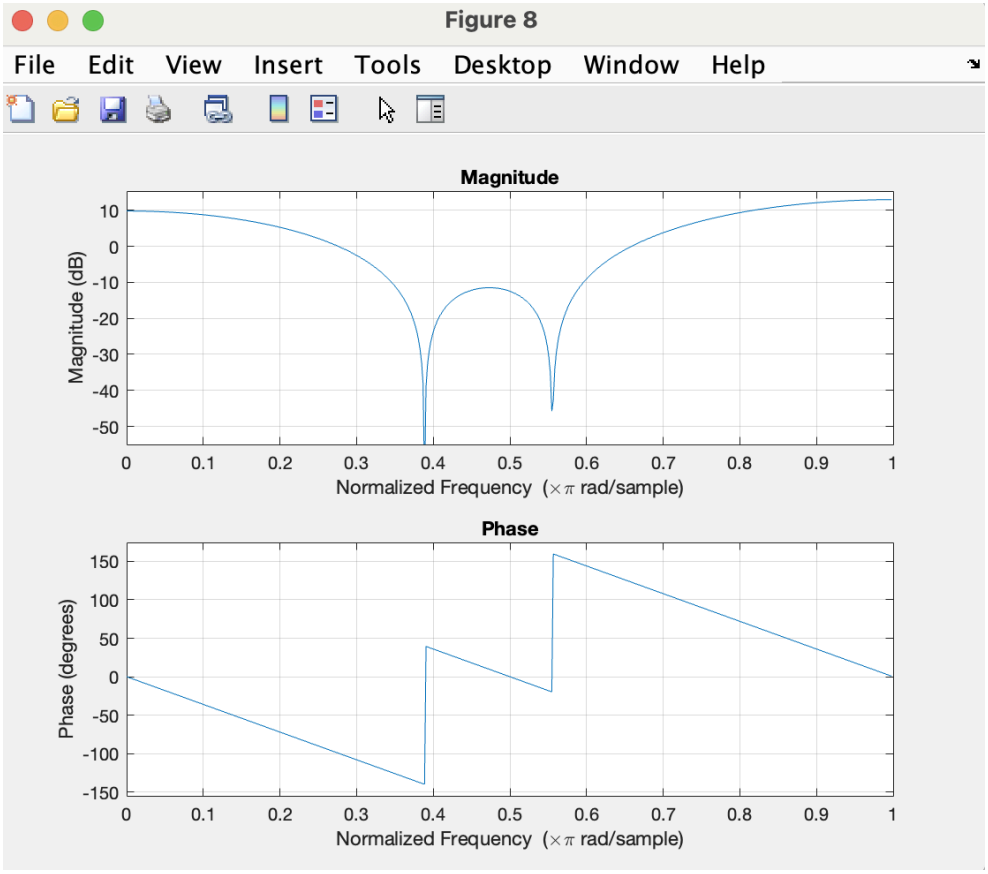
```
% 2.3a
sound(xxbad, fs);
figure;
spectrogram(xxbad, 512, 128, 512);
figure;
freqz(xxbad);
```

(b) Design a cascade of two FIR nulling filters to remove the sinusoids completely. Combine the two filters to create one.

```
% 2.3b
fir1 = [1 -2*cos(2*pi*2222/fs) 1];
fir2 = [1 -2*cos(2*pi*1555/fs) 1];
combined_filter = conv(fir1, fir2);
xxbad_clean = filter(combined_filter, 1, xxbad);
pause(5);
sound(xxbad_clean, fs);
```

The filter coefficients of the equivalent filter is calculated through the convolution of the two individual filters, resulting in: [1, -0.3379, 1.7624, -0.3379, 1].

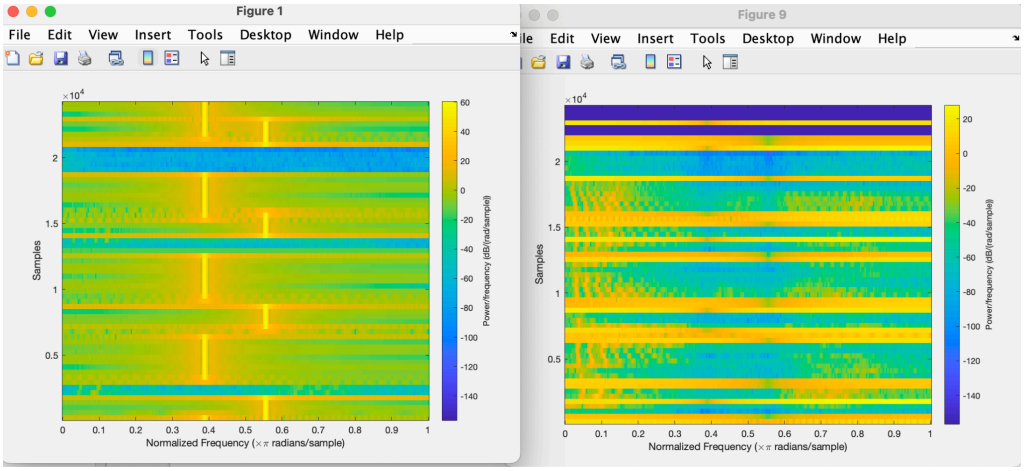
(c) Plot the frequency response of the cascaded nulling filter designed in the previous part. Indicate the frequencies where the nulls are found.



The frequency and magnitude response of the cascaded nulling filter `freqz(combined_filter);`

The frequencies where the nulls are found include approximately 0.38*pi and 0.55*pi which when denormalizing we get (0.38*pi / 2*pi) * 8000 = 1520 Hz and (0.55*pi / 2*pi) * 8000 = 2200 Hz which are the frequencies we wanted to nullify.

(d) Process the corrupted signal, xxbad, through the nulling filters. Make two spectrograms (in dB): one for the input signal and the other for the output signal$_5$. Point out features that verify that the nulling filters operated correctly.



Side-by-side comparison of the two spectrograms with the cleared-up signal (right)

The cleared-up spectrogram on the right has much clearer horizontal bands, with the background being darker which displays the reduction in noise. Furthermore, the vertical bright lines depicted at 0.38*pi and 0.55*pi (1555 and 2222 Hz) are absent in the right picture meaning they've been filtered out.

Conclusion

This project focused on analyzing the responses of averaging and nulling filters, primarily through MatLab's DLTI-Demo GUI. We first worked with a four-point averaging filter which was proven to function as a low-pass filter. We then analyzed the differences between theoretical and experimental values, observing the accuracy of our results. Then, we proved the periodicity of these functions through mathematical derivation. For nulling filters, we performed the

successful attenuation of numerous frequencies and saw the pole-zero plot for a three-point nulling filter.

We also compared the accuracy between ideal and theoretical band-pass filters, seeing how imperfect filters operate, creating a near-low/near-high response similar to what we would deal with in the industry. Again, this experiment was coupled with theoretical mathematic calculations revolving around Euler's formula in the complex exponential and sinusoidal domain.

Bibliography

1. the. “What Is the Intuition of ‘Averaging Is a Low Pass Filter’?” Signal Processing Stack Exchange, 14 Nov. 2016, dsp.stackexchange.com/questions/35539/what-is-the-intuition-of-averaging-is-a-low-pass-filter. Accessed 6 Nov. 2024.
2. “Low-Pass Filtering (Blurring).” Diffractionlimited.com, 2024, cdn.diffractionlimited.com/help/maximdl/Low-Pass_Filtering.htm. Accessed 6 Nov. 2024.
3. “Intro to Signal Smoothing Filters.” Seeq.com, 2024, support.seeq.com/kb/R58/cloud/intro-to-signal-smoothing-filters. Accessed 6 Nov. 2024.
4. “Can I Create the Same FIR Filter Using Two Different Commands?” Mathworks.com, 2023, www.mathworks.com/matlabcentral/answers/2030719-can-i-create-the-same-fir-filter-using-two-different-commands. Accessed 7 Nov. 2024.
5. “Frequency Response from Transfer Function.” Mathworks.com, 2024, www.mathworks.com/help/signal/ref/freqz.html. Accessed 7 Nov. 2024.
6. “Intro to Signal Smoothing Filters.” Seeq.com, 2024, support.seeq.com/kb/R58/cloud/intro-to-signal-smoothing-filters. Accessed 7 Nov. 2024.
7. Davis, Nick. “An Introduction to Filters.” Allaboutcircuits.com, All About Circuits, 30 Sept. 2023, www.allaboutcircuits.com/technical-articles/an-introduction-to-filters/. Accessed 7 Nov. 2024.

Addendum (Code)

1.5)

```
% 1.5b
bb = 1/4*ones(1, 4);
w = -pi : 2*pi/400 : pi-2*pi/400;
H = (1/4) * exp(-1.5*j*w) .* (exp(-1.5*j*w) + exp(1.5*j*w) + exp(-0.5*j*w) + exp(0.5*j*w));
figure;
plot(w, abs(H));
xlabel( 'n' );
ylabel('Magnitude response');
title('Magnitude response vs. n');
figure;
plot(w, angle(H));
xlabel( 'w' );
ylabel('Phase response');
title('Phase response vs. w');
xlim( [-pi, pi] );
% 1.5c
figure;
freqz(bb, 1, w);
% These are not the same. The magnitude is different because freqz doesn't
% account for the magnitude and phase correctly, as it doesn't take absolute value (you'd
% need to convert the minus sign of negative cosine values to a phasor with phase pi radians,
% which you would then integrate into the phase plot, with resultant jumps
% of +-pi radians.
```

1.6)

```
w_null = 0.75*pi;
w = -pi : 2*pi/400 : pi-2*pi/400;
bb = [1 -2*cos(w_null) 1];
H = abs(1 + -2*cos(w_null)*exp(j*w) + exp(j*2*w));
```

```
% Amplitude Response
figure;
plot(w, abs(H));
xlabel( 'n' );
ylabel('Magnitude response');
title('Magnitude response vs. n');
% Zero-Pole Response
figure;
denom = [1];
zplane(bb, denom);
% 1.6 extra
% <u>Pole-Zero Diagram Plot:</u>
% Pole -> w = 0 (no point where ampltiude response is infinity (denominator
% is just 0), but at w=0, we have a pole, as this leads the denominator in the
z-domain to be 0. The nulled frequencies (corresponding to zero magnitudes, as
% shown in the magnitude response plot & dltidemo) are on the unit circle
% at their respective frequencies.
% <u>Magnitude Response Plot:</u>
% Zeros at w=-w_null,w_null in the frequency domin, as indicated as well in
% the dltidemo. For time domain, these zeros correspond to the following:
(f=-w_null/2*pi,w_null*2*pi)
```

2.3)

```
% 2.3d is completed within 2.3a and 2.3c
% 2.3a
sound(xxbad, fs);
figure;
spectrogram(xxbad, 512, 128, 512);
figure;
freqz(xxbad);
% 2.3b
fir1 = [1 -2*cos(2*pi*2222/fs) 1];
fir2 = [1 -2*cos(2*pi*1555/fs) 1];
combined_filter = conv(fir1, fir2);
xxbad_clean = filter(combined_filter, 1, xxbad);
figure;
```

```
freqz(combined_filter);
pause(5);
sound(xxbad_clean, fs);
% 2.3c
figure;
spectrogram(xxbad_clean, 512, 128, 512);
figure;
freqz(xxbad_clean);
```