Mini-Project #2: Averaging and Nulling FIR Filters

Prof. Brian L. Evans and Mr. Gordon Ta

December 1, 2024, Version 4.0

1.0 Introduction

In speech, audio, and many other applications, filters are used to pass/amplify desired frequencies and attenuate/reject undesired frequencies. The two classes of linear time-invariant (LTI) filters are finite impulse response (FIR) filters and infinite impulse response (IIR) filters. The names refer to the duration of their impulse responses.

For an LTI system, the impulse response uniquely characterizes the system. For the impulse response, we can determine how the filter will behave in the time or frequency domains. In the time domain, the output is the convolution of the input signal and the filter's impulse response. In the frequency domain, the output is the product of the frequency response of the filter and the frequency content of the input signal. The frequency response of the filter is determined by taking the Fourier transform of the impulse response.

This multiplication in the frequency domain is the mechanism by which the filter can alter the magnitude and phase of the frequency content of the input signal. Based on the filter's magnitude response at a particular frequency, that frequency of the input signal is amplified, passed through, attenuated, or rejected if the magnitude response is greater than 1, equal to 1, between 0 and 1, or equal to 0, respectively. The filter's phase response, which causes an additive effect on the phase of the frequency content of the input signal, can be delay all frequencies in the input signal by the same amount in the time domain (linear phase) or assign a different delay to each frequency.

Based on the filter's magnitude response, we can often characterize the filter as passing low frequencies (lowpass), passing high frequencies (highpass), passing a band of frequencies (bandpass), attenuating a band of frequencies (bandstop), passing all frequencies (allpass), and eliminating a finite number of frequencies (notch/nulling). This mini-project will be cover ideal and practical, averaging and notch/nulling filters [1]. Averaging filters are lowpass filters [2].

2.0 Frequency Response

2.1 Frequency Response of FIR Filters (Section 1.3 in [1])

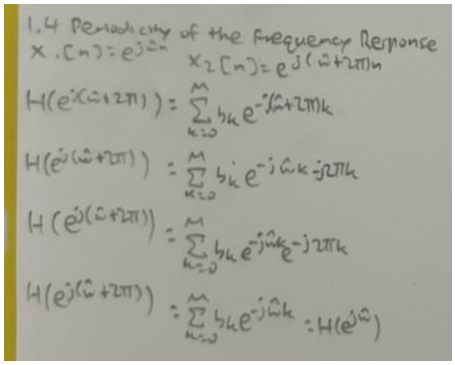
If the input to an LTI system is a complex-valued sinusoid, the output is a complex sinusoid of the same frequency scaled by the filter's frequency response evaluated at that frequency:

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]\, x[n-k] = \sum_{k=0}^{M} h[k]\, Ae^{j\phi} e^{j\hat{\omega}(n-k)} = \sum_{k=0}^{M} h[k]\, Ae^{j\phi} e^{-j\hat{\omega}k} e^{j\hat{\omega}n}$$

The output is seen to be a complex exponential with the same frequency as the input signal with a scale in magnitude and shift in phase.

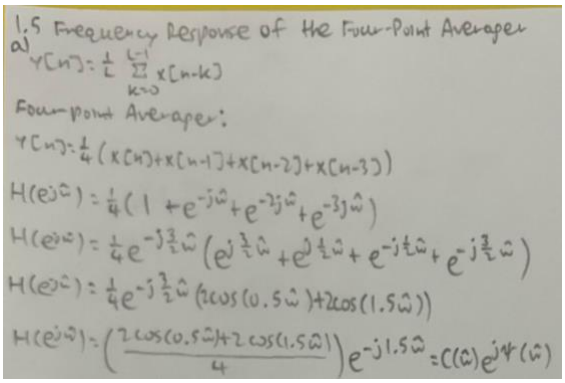
2.2 Periodicity of the Frequency Response (Section 1.4 in [1])

The frequency responses of discrete-time filters are periodic with a period equal to 2π. The derivation to show this periodicity is given on the right:



3.0 Averaging and Nulling FIR Filters

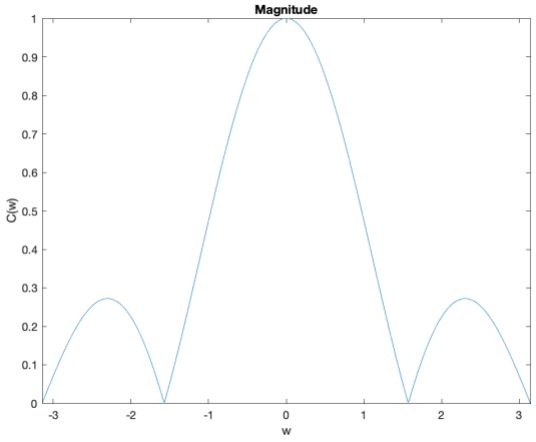
3.1 Frequency Response of the Four-Point Averager (Section 1.5 in [1])

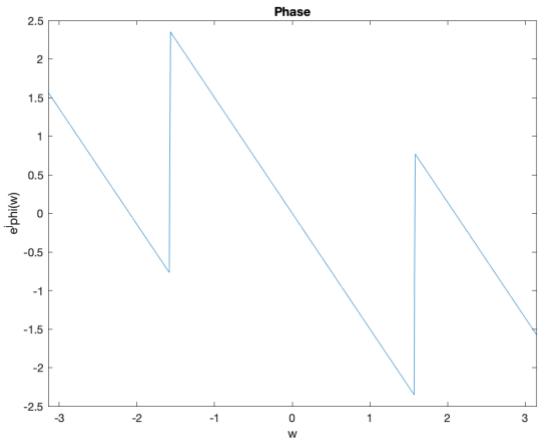


(a) We derive the frequency response of the four-point averager is on the right. An averaget is also known as a running average and an averaging filter.

(b) We compute the frequency response of the four-point averager in MATLAB. We can also plot the vectors vs. the discrete-time frequency from -π to π and covering 400 samples. The MATLAB code to perform these tasks and the resulting plots can be seen below.

```
w = linspace(-pi, pi, 400);
C_w = (2*cos(0.5*w)+2*cos(1.5*w))/4;
phi_w = exp(-j*1.5*w);
H = C_w .* phi_w;
plot(w, abs(H));
xlabel('w');
ylabel('C(w)');
title('Magnitude');
xlim( [-pi pi] );
figure;
plot(w, angle(H));
xlabel('w');
ylabel('e^jphi(w)');
title('Phase');
xlim( [-pi pi] );
```



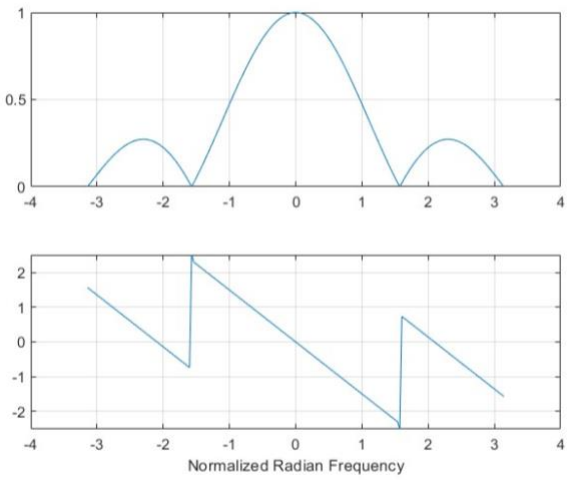


(c) An alternate way to plot the magnitude and phase response of the four-point running average is by using the freqz function. The implementation is provided in Section 1.3.1 in [1], and we change the bb variable to use the coefficients of a four-point running average ($L = 4$) given by

$$h[n] = \frac{1}{L}\sum_{k=0}^{L-1}\delta[n-k]$$

The MATLAB code and frequency response plots can be seen below.

```
% Filter Coefficients
bb = 1/4*ones(1,4);
% omega hat frequency vector
ww =-pi:(pi/100):pi;
% freekz(bb,1,ww) is an alternative
H = freqz(bb, 1, ww);
subplot(2,1,1);
plot(ww, abs(H)), grid on
subplot(2,1,2);
plot(ww, angle(H)), grid on
xlabel('Normalized Radian Frequency')
```



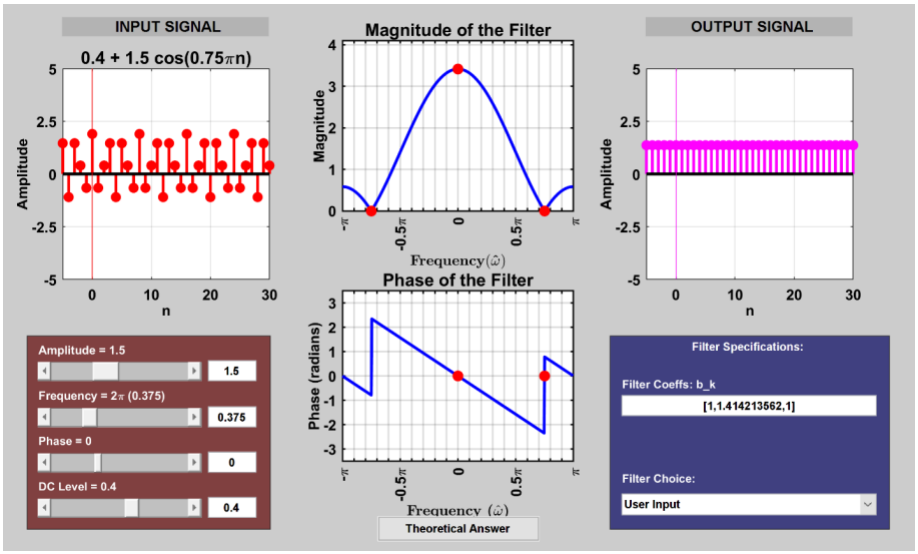
Note that the discontinuities in the phase response of sudden changes by π occur at frequencies that are zeroed out in the magnitude response. Since these frequencies do not pass through the filter, the phase is linear with a slope of -1.5, which corresponds to a group delay of 1.5 samples.

3.2 Section 1.6 FIR Nulling Filters

FIR nulling filters can be used to remove sinusoids/frequencies from the input signal. The coefficients for this filter is given by

$$b_0 = 1 \qquad b_1 = -2\cos(\hat{\omega}_{\text{NULL}}) \qquad b_2 = 1$$

We can observe the behavior of the FIR nulling filter by using the dltidemo GUI with a nulling frequency of 0.75π and input signal $x[n] = 0.4 + 1.5\cos(0.75\pi n)$.



The output signal shows the sinusoidal component of the signal has been removed, and the output signal is now constant.

We plot the magnitude and frequency response of the nulling filter in MATLAB by first initializing the coefficients to the same coefficients used in the dltidemo GUI.

Then, we compute the poles and zeroes by using the roots function. Finally, zplane is used to create a pole-zero plot. The MATLAB code to perform these actions can be seen below.

```
b = [1, 1.414213562, 1];
a = [1];
poles = roots(a)
zeroes = roots(b)
```

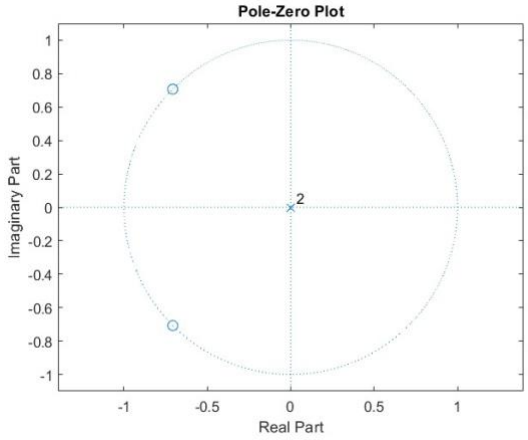
```
poles =

  0×1 empty double column vector

zeroes =

  -0.7071 + 0.7071i
  -0.7071 - 0.7071i
```

```
zplane(b,a)
```

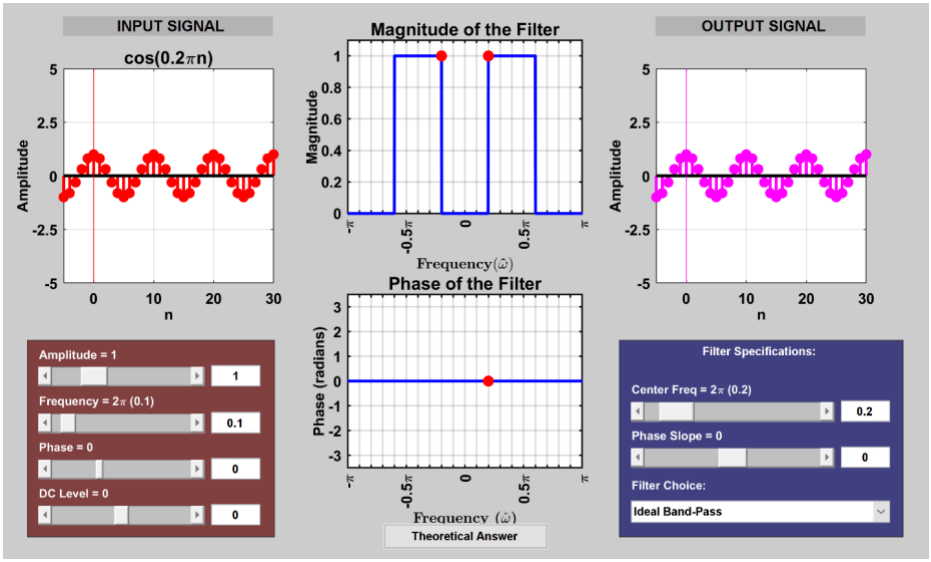


The pole-zero plot indicates zeroes at $-0.7071 \pm j0.7071$ which aligns with the magnitude response of the dltidemo GUI. Poles at the origin have no effect on the magnitude response. With the zeros at $z = e^{j\,0.75\,\pi}$ and $z = e^{-j\,0.75\,\pi}$, the magnitude response is highest at the point farthest from the zeroes, which is at $z = 1 = e^{j\,0}$ which is at 0 rad/sample as in the magnitude response of the dltidemo GUI. Further, the zero locations means that frequencies of $0.75\,\pi$ and $-0.75\,\pi$ are eliminated. On the input, every positive frequency has a negative counterpart because of the inverse Euler's formula $\cos(\omega_0\,n) = \frac{1}{2}\,e^{-j\,\omega_0\,n} + \frac{1}{2}\,e^{j\,\omega_0\,n}$ which is why we would need zeros corresponding to $-\omega_0$ and ω_0 to eliminate frequency ω_0.

4.0 Ideal and Practical Filters

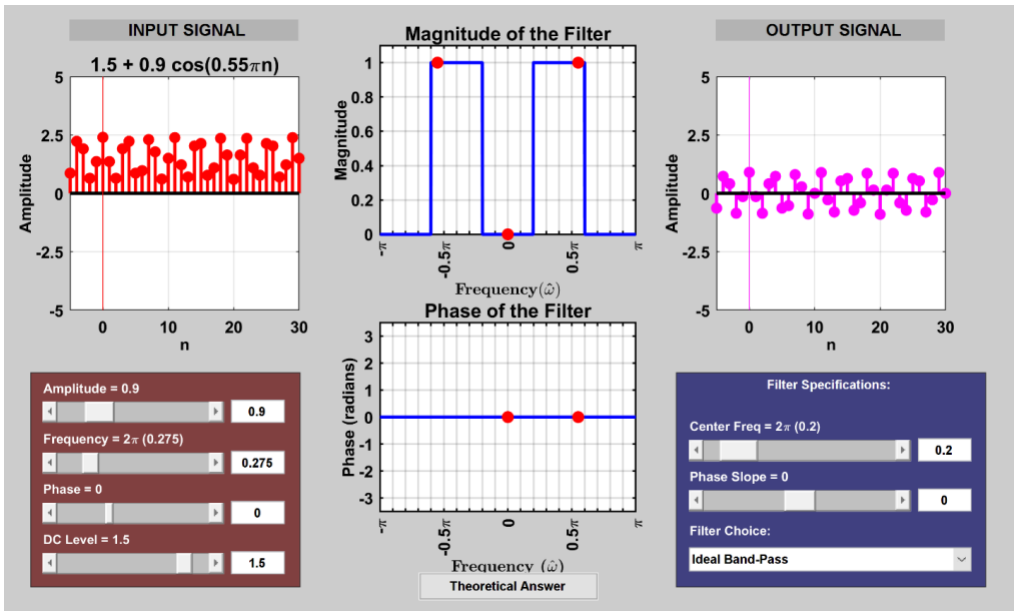
4.1 Section 1.7 Ideal Filters and Practical Filters

(a) We can use the dltidemo GUI to examine the behavior of an ideal bandpass filter.



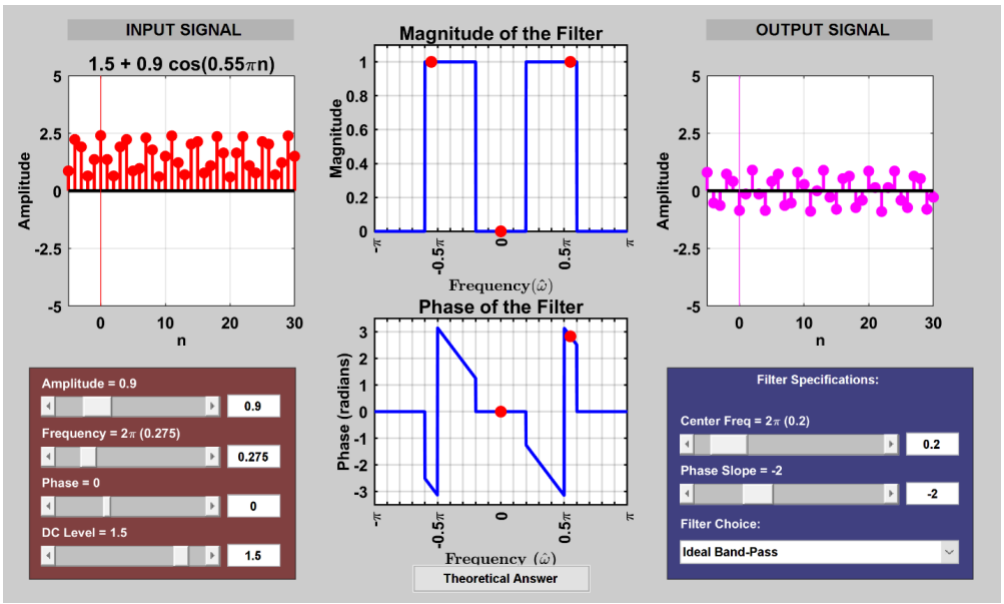
By choosing a center frequency of 0.4π and using the 0.4π bandwidth of the GUI, we can observe the outputs seen above. The 'Magnitude of the Filter' shows the frequencies that can pass through the filter, which is between 0.2π and 0.4π with the chosen filter parameters.

(b) Now, we set the input to $x[n] = 1.5 + 0.9\cos(0.55\pi n)$, and observe the changes in the GUI:



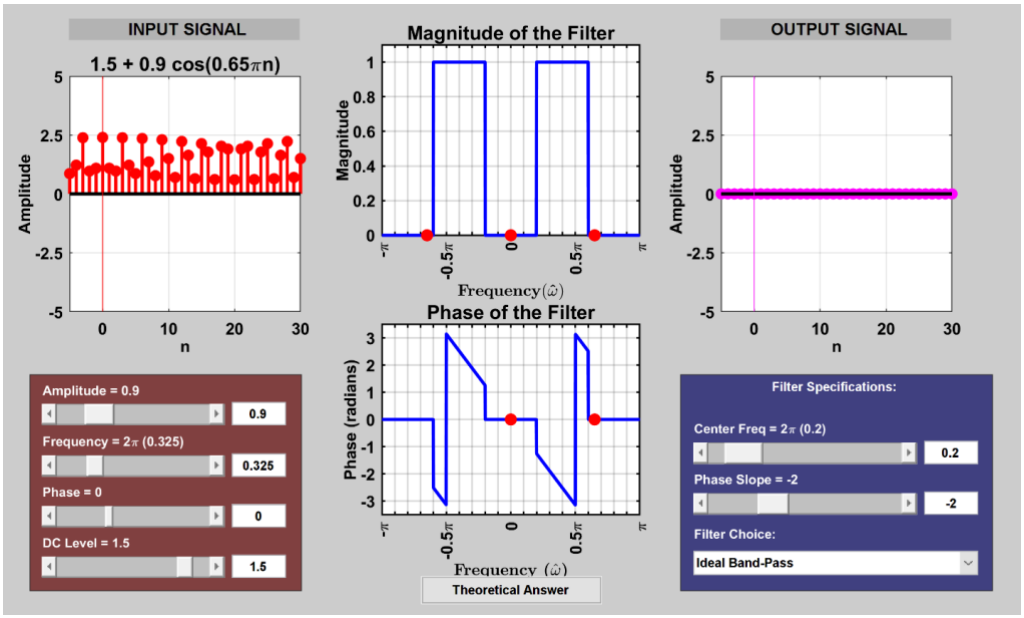
Only 0.55π is present in the output because the DC component of the input signal (1.5) has a frequency of zero, which is not in the passbands of $\pm 0.2\pi$ to 0.6π.

(c) We can set the phase slope to -2 and observe the changes to the output of the GUI.



Compared to part b, the output signal has shifted to the right. Mathematically, the phase slope caused the output to be $1.5 + 0.9\cos\big(0.55\pi(n-2)\big) = 1.5 + 0.9\cos(0.55\pi n - 1.1\pi) = 1.5 + 0.9\cos(0.55\pi n + 0.9\pi)$, which means there is a phase shift. The delay can also be found by estimating the slope of the phase slope of the frequency slope.

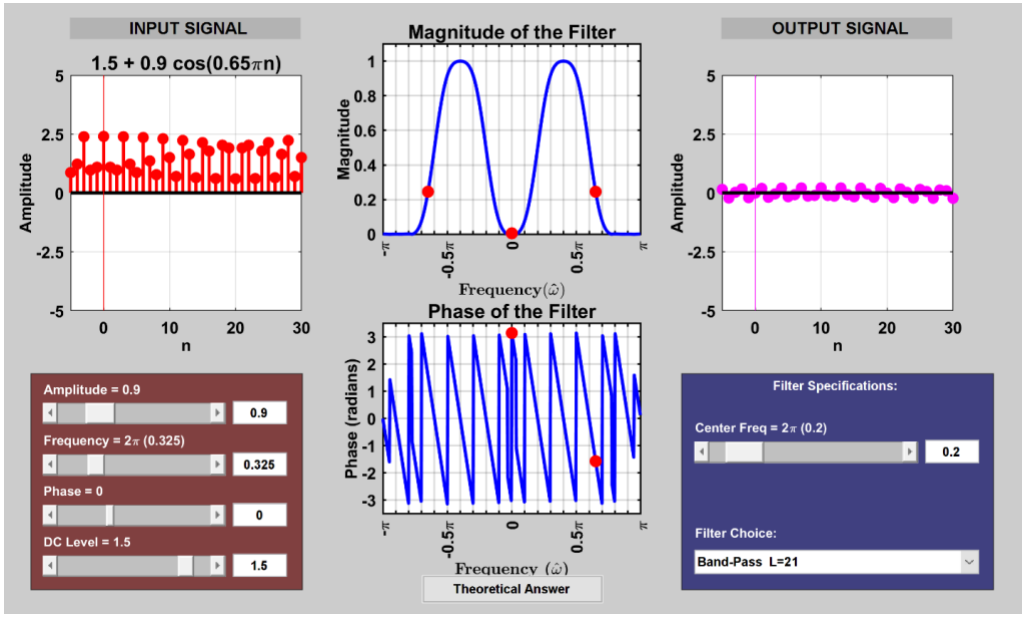
(d) Now we change the input signal to $x[n] = 1.5 + 0.9\cos(0.65\pi n)$ and observe the changes to the output of the GUI.



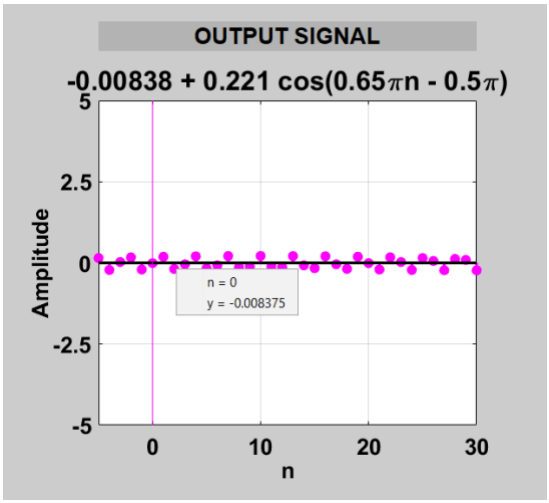
The output signal is now zero because both the DC component and the cosine component of the signal are not in the passbands of $\pm 0.2\pi$ to 0.6π.

Practical Filters

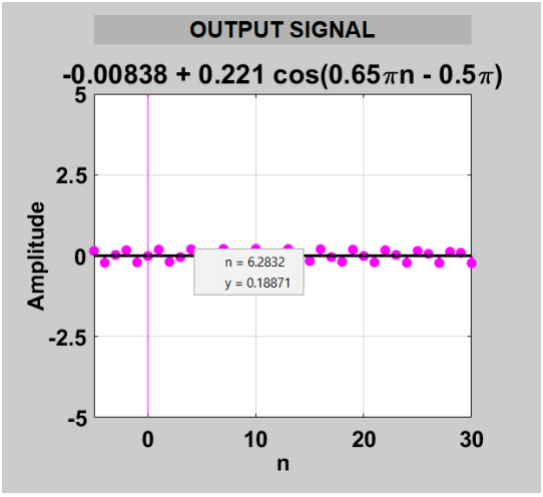
(a) To simulate the behavior of practical filters, we can change the filter choice in the dltidemo GUI to a band-pass filter with length 21. Like in part a of ideal filters, we set the center frequency to 0.4π and observe the GUI output.



(b) Looking at the GUI output in part a with the input signal $x[n] = 1.5 + 0.9\cos(0.65\pi n)$, we can see that the output is close to, but not exactly, zero. This behavior is caused by the practical filter not having sharp edges at the cutoff frequencies but rather curves allowing frequencies outside the passband to pass through to the output.



(c) Within the GUI, we can right-click the plot of the output signal to obtain the amplitude at various points which can be seen below.

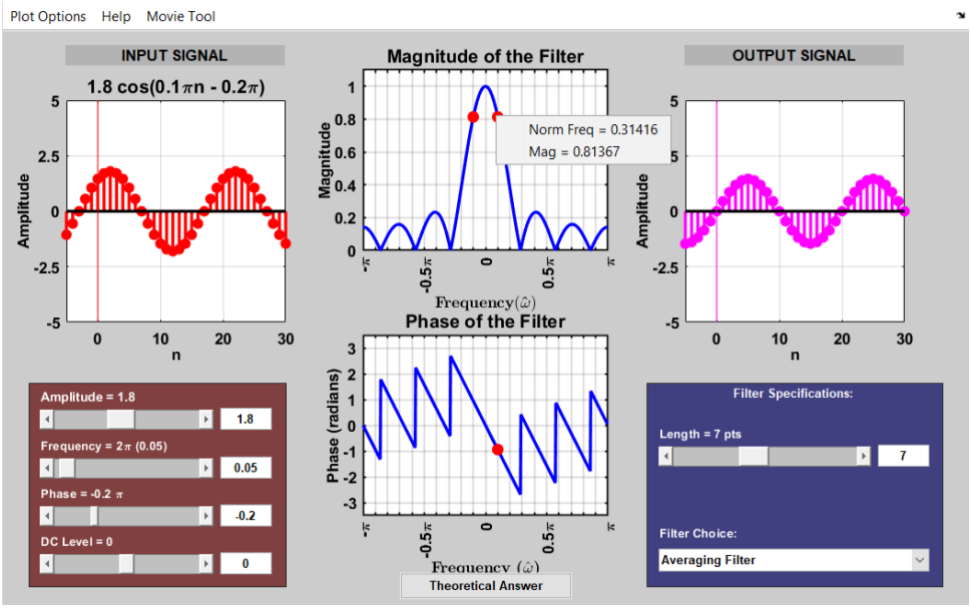


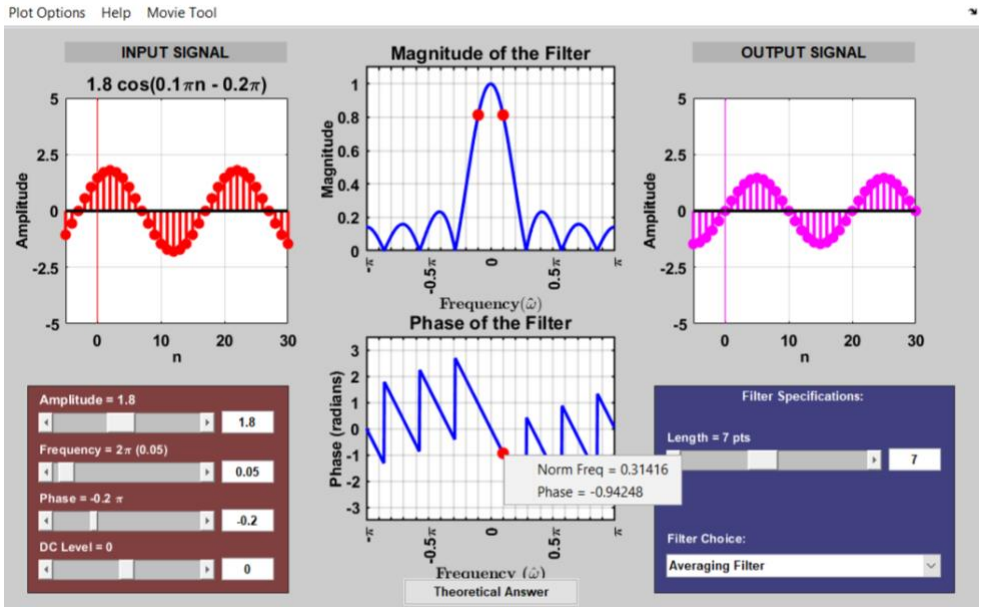
Due to the non-ideal nature of the filter, the different frequency components have been attenuated differently.

4.2 LTI Frequency Response Demo (Section 2.1 in [1])

(a) First, we set the input signal to be $x[n] = 1.8\cos\left(0.1\pi(n-2)\right)$.

(b) We use the dltidemo GUI to explore the frequency response of a 7-point averager. With a filter choice of 'Averaging Filter' and 'Length' of 7 points, the GUI produces the output below.

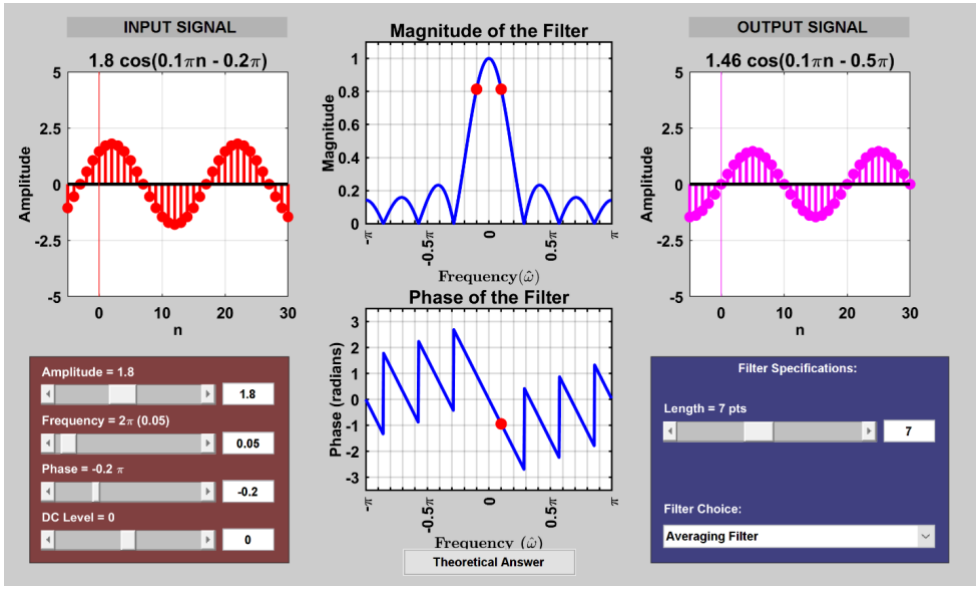




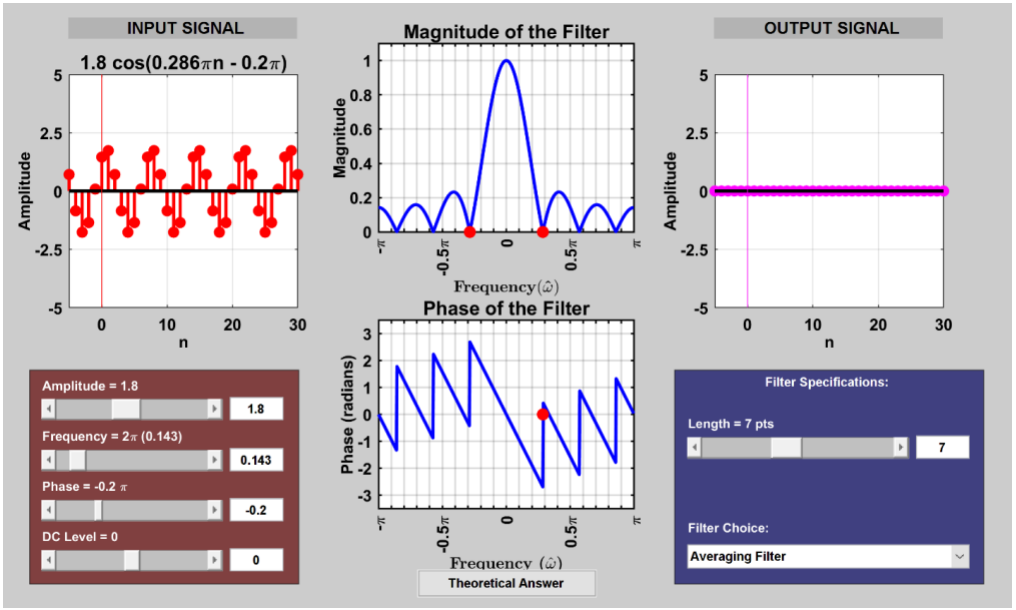
We can right-click the plots of the magnitude and phase response in the middle of the GUI to obtain the respective values at the input frequency.

(c) The group delay is the derivative of the phase response evaluated at 0.1π.

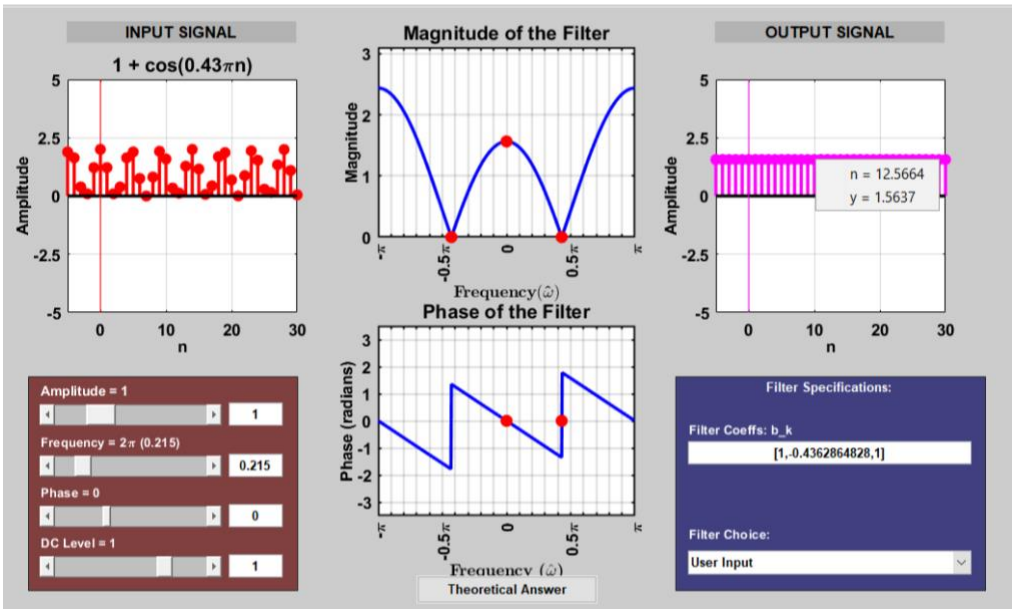
(d) After clicking 'Theoretical Answer,' we see the equation for the output signal. Writing the output signal in the form y[n] = A cos(w0 (n - n7)), we see that n7 = 5. Since the input signal had a peak at n=2, we can conclude that the peak of the cosine wave has been shifted by 3 samples.



(e) Using the handout 'Designing Averaging Filters,' [2] the null bandwidth of a seven-point averaging filter is $\frac{2\pi}{7}$ rad/sample and frequencies in the range between 0 and π is $\frac{2\pi}{7}, \frac{4\pi}{7}$, and $\frac{6\pi}{7}$ rad/sample. Below is an example of an input signal with frequency $\frac{2\pi}{7}$ rad/sample, which produced an output signal of zero.



(f) To design an FIR nulling filter for the input signal $x[n] = 1 + \cos(0.43\pi n)$, we use filter coefficients from lab Section 1.6 in [1]. The frequency we want to null is 0.43π so the filter coefficients are $[1 \ -2\cos(0.43\pi) \ 1]$ or [1 -0.4362864828 1]. After changing "Filter Choice" in the GUI and inputting these filter coefficients, we observe the output.



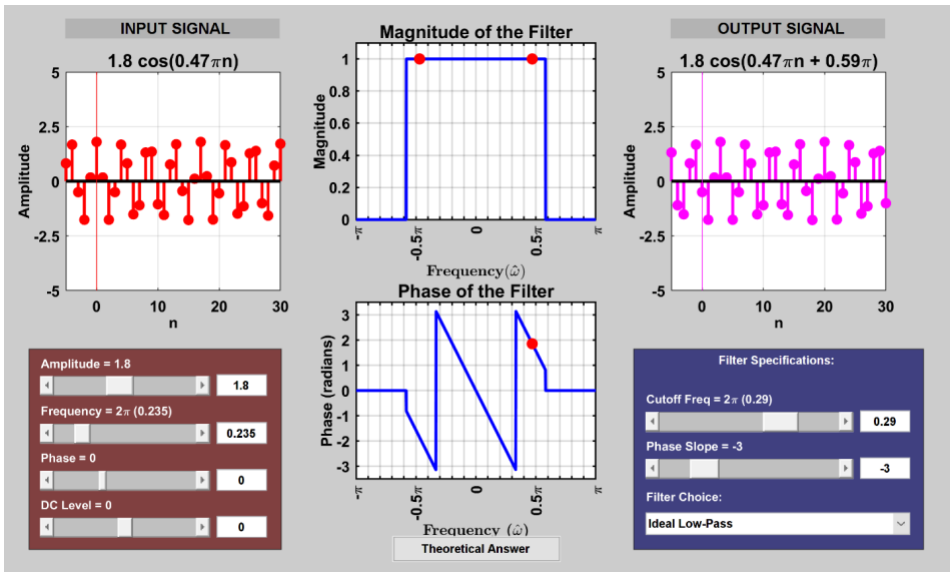
In the output signal, the DC component was not removed, and right-clicking the plot shows us the value of the signal for all n is 1.5637.

4.3 Ideal Filters and Practical Filters (Section 2.2 in [1])

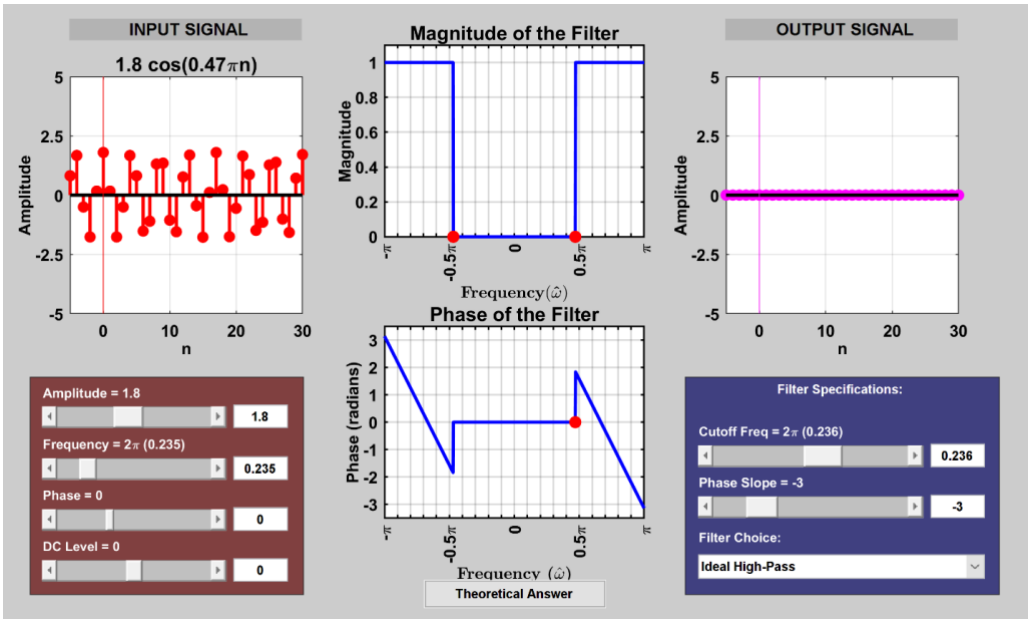
2.2.1 Ideal Filters

(a) First, we set the input signal to $x[n] = 1.8\cos(0.47\pi n)$.

(b) We use the GUI to examine the behavior of an ideal LPF with cutoff frequency of $2\pi(0.29)$. To create a delay of 3 samples, we set the phase slope to 3. After clicking 'Theoretical Answer,' the phase is 0.59π. It is not -1.41π or -3*0.47π because it is outside $-\pi$ to π and wraps to 0.59π.

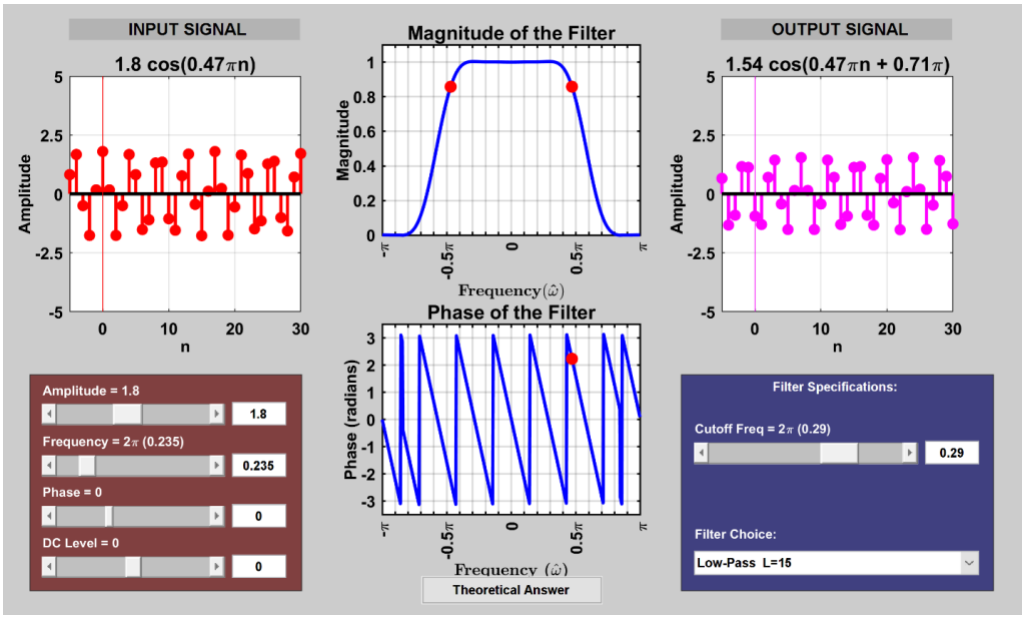


(c) We also use the GUI to examine the behavior of an ideal HPF using the same phase slope as in part b. We can determine that the minimum cutoff frequency for the HPF is 2π*0.236 because any lower would allow the input signal to pass through.



2.2.2 Practical Filters

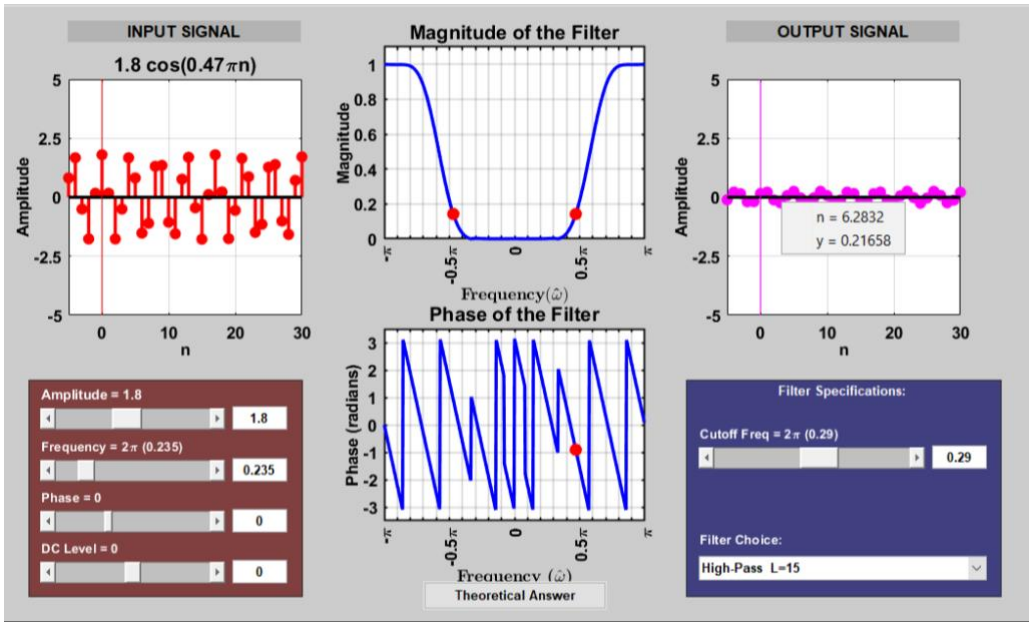
(a) Now, we use the GUI to examine the behavior of practical filters. First, we set the input signal to be $x[n] = 1.8\cos(0.47\pi n)$ like in section 2.2.1. We use a low-pass filter with length 15 and set the cutoff frequency to $2\pi(0.29)$, and we can observe the output of the GUI below.



Amplitude and phase of the output signal are different than those of the output signal obtained using the ideal filter. Additionally, the plots of the magnitude and phase response are different, so the practical filter does not perform as well when passing low frequencies of the input signal.

(b) Using the plot of the phase response, the delay can be estimated from the slope which is -3-3/(0.3), which does not match the delay in the ideal case.

(c) Now we use the GUI to examine the output of a practical HPF. We set the filter type to a HPF with length 15 and cutoff frequency of $2\pi(0.29)$. We can see the output of the GUI below.



Right-clicking the plot of the output signal shows that the high-pass filter does a decent, but not perfect, job of rejecting the input signal since the amplitude of the output signal is around 0.2 or around 10% of the original amplitude.

5.0 Removing Interference from a Speech Signal

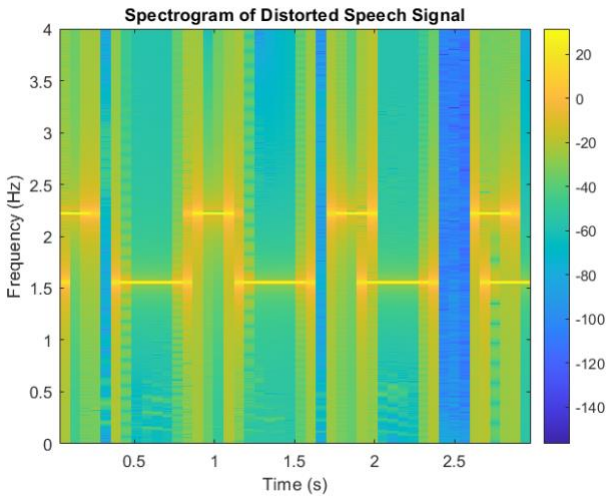
(a) We use the following MATLAB code to load the distorted speech signal from speechbad.mat. By loading the file, three variables will be defined:

xxbad

fs

f_interference

The sampling rate fs is 8000 Hz, and we can play the signal with the sound function and plot a spectrogram of the signal using the following code.




```
load speechbad.mat
sound(xxbad, fs)
figure;
spectrogram(xxbad, 1024, 512, 1024, fs,
'yaxis');
title('Spectrogram of Distorted Speech Signal');
xlabel('Time (s)');
ylabel('Frequency (Hz)');
colorbar;
```

(b) We can use the equation for discrete-time frequency from lecture slide 5-5 to find the discrete-time frequencies for the interfering frequencies of 2222 and 1555 Hz.

$$\hat{\omega} = \omega\, T_s = \frac{\omega}{f_s} = 2\pi \frac{f}{f_s}$$

The MATLAB code for obtaining the discrete-time frequencies from the f_interference vector and using the above formula can be seen below

```
w1 = 2*pi*f_interference(1)/fs
w2 = 2*pi*f_interference(2)/fs
```

```
w1 =

    1.7452


w2 =

    1.2213
```

The discrete-time frequencies are approximately 1.7452 and 1.2213. Using these frequencies, two nulling filters can be created using the filter coefficients from lab Section 1.6 in [1].

$$b_0 = 1 \qquad b_1 = -2\cos(\hat{\omega}_{\text{NULL}}) \qquad b_2 = 1$$

The MATLAB code to generate the coefficients is

```
b1 = [1 -2*cos(w1) 1]
b2 = [1 -2*cos(w2) 1]
```

```
b1 =

    1.0000    0.3470    1.0000


b2 =

    1.0000   -0.6849    1.0000
```

We can convolve the coefficients of the two nulling filters to obtain the coefficients for an equivalent filter which is the cascade of the two nulling filters.

The MATLAB code to convolve the coefficients is

```
equivalentFilterCoeff = conv(b1,b2)
```

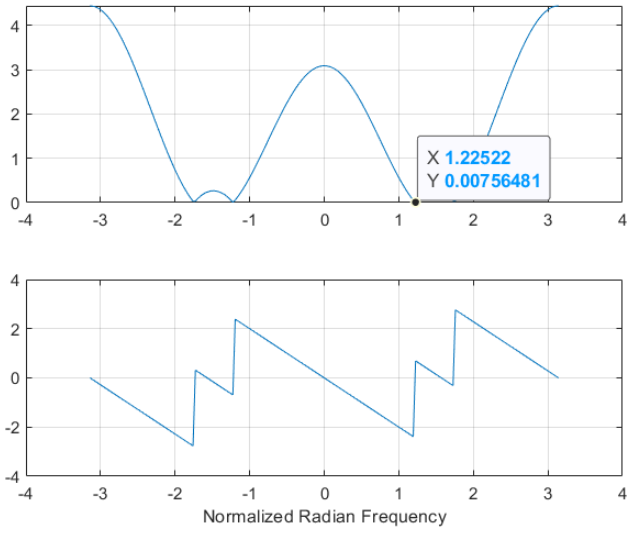
The resulting filter coefficients are

```
equivalentFilterCoeff =

    1.0000   -0.3379    1.7624   -0.3379    1.0000
```

(c) We can plot the frequency response of the cascaded nulling filter using code from lab Section 1.3.1 in [1] with the filter coefficients to null the two interfering frequencies.

```
%-- omega hat frequency vector
ww =-pi:(pi/100):pi;
H = freqz(equivalentFilterCoeff, 1, ww);
subplot(2,1,1);
plot(ww, abs(H));
grid on;
subplot(2,1,2);
plot(ww, angle(H));
grid on;
xlabel('Normalized Radian Frequency');
```



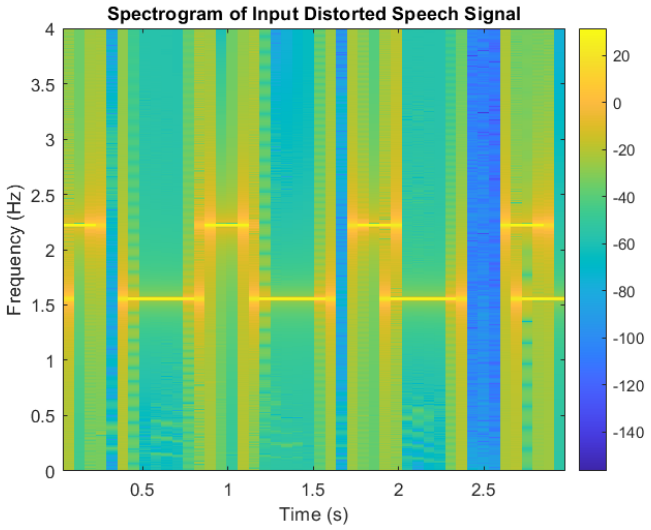
The nulls can be found by identifying the zeroes of the magnitude response. 'Data Tips' can be used to highlight these locations which are 1.2213 and 1.7452 and the corresponding negative frequencies.

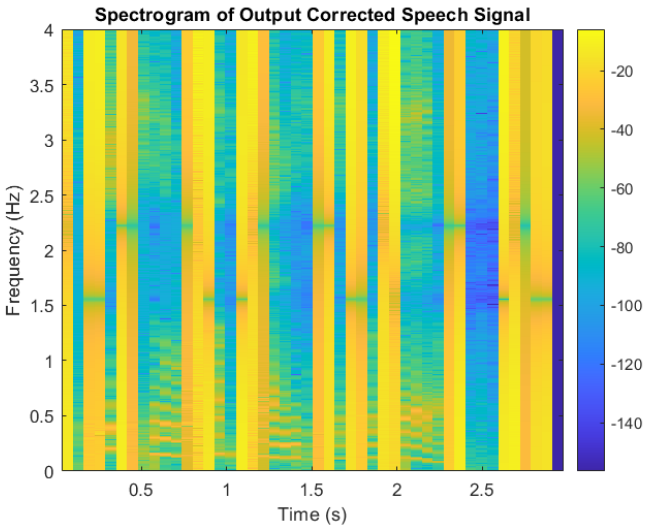
(d) Using two nulling filters in cascade, we filter the input signal xxbad and plot spectrograms of the input and output signal to see if the filter correctly removed the interfering frequencies of 2222 and 1555 Hz. The code and spectrograms can be seen below.

```
figure;
spectrogram(xxbad, 1024, 512, 1024, fs, 'yaxis');
title('Spectrogram of Input Distorted Speech Signal');
xlabel('Time (s)');
ylabel('Frequency (Hz)');
colorbar;

xx_null_filtered = filter(equivalentFilterCoeff, 1, xxbad)
figure;
spectrogram(xx_null_filtered, 1024, 512, 1024, fs, 'yaxis');
title('Spectrogram of Output Corrected Speech Signal');
xlabel('Time (s)');
ylabel('Frequency (Hz)');
colorbar;
```

The spectrogram of the output signal no longer contains the interference represented by the yellow horizontal lines at 1555 and 2222 Hz like in the spectrogram of the input signal. Thus, the cascaded nulling filter operated correctly





The original speech could not be understood due to the loud interference by the two frequencies. After applying the two nulling filters to remove the interference, the speech was intelligible as "Thieves who steal from friends deserve jail".

6.0 Conclusion

Many types of filters can be used to process signals like lowpass, highpass, bandpass, and nulling filters. The filter coefficients can be designed to meet certain frequency specifications or goals like removing or passing bands of frequencies.

Ideal filters can zero out entire frequency bands and have do not have a transition between passing and eliminating frequency bands. However, ideal filters have impulse responses that are two-sided and infinite, and hence cannot be implemented in practice.

Practical filters can only eliminate a finite number of frequencies and have a transition between passing and attenuating/eliminating frequencies. The higher the filter order, the closer the practical filter's frequency response gets to the ideal filter's frequency response.

The two classes of linear time-invariant (LTI) filters are finite impulse response (FIR) filters and infinite impulse response (IIR) filters. In this mini-project, we designed, analyzed, and simulated averaging and nulling FIR filters. Averaging filters are lowpass filters that also eliminate positive frequencies that are integer multiples of $2\pi/L$ rad/sample up to π but not including 0 as well as their negative frequency counterparts, where L is the number of filter coefficients or equivalently $L-1$ is the filter order. Nulling filters are designed by placing a pair of conjugate symmetric zeros on the unit circle at an angle equal to the frequency to be nulled or notched out. LTI filters can be placed in cascade to realize more complicated frequency responses. In a practical implementation, a necessary condition for a filter to be LTI is to have all its initial conditions be zero.

This mini-project explored three key ideas concerning linear time-invariant systems: [3]

- **Time domain**: Output signal is the convolution of the input signal and the filter's impulse response.
- **Frequency domain**: Output signal is the product of the input signal and the filter's frequency response. The filter's frequency response is the Fourier transform of the impulse response.
- **Filtering**: The magnitude of the frequency response can be designed to pass, attenuate, and amplify bands of frequencies as well as eliminate individual frequencies. The phase of the frequency response can be designed to delay all frequencies by the same amount in the time domain (linear phase) or assign a different delay to each frequency component. Practical filters have different frequency responses than ideal filters.

References

[1] James McClellan, Ronald Schafer, Mark Yoder, *DSP First*, Lab S-5: DLTI GUI and Nulling Filters.

[2] Brian Evans, Designing Averaging Filters, Handout for UT Austin ECE 313 Linear Systems & Signals, Fall 2024.

[3] Brian Evans, Mini-Project #2 Assignment, UT Austin ECE 313 Linear Systems & Signals, Fall 2024.