MMI 503/MMI 603 - Audio Signal Processing 2

Project 2 Assignment

Author: Ashay Dave

1) Guide Portion - Filter Basics

a. Filter Summary

i. What are filters and what are they used for?

A filter is a set of operations (delays) or a "function" that is used to attenuate or boost, or change the frequency content of a digital signal when the signal is passed through it. The "frequency content" of the signal is a bunch of sines and cosines that essentially tell us which frequencies make up the signal, and what their magnitude is. By tuning the "frequency content" of a signal (in the Z-domain using a Fourier Transform), a filter is able to change the characteristics of these sines and cosines which affect the input signal in whichever way the engineer deems necessary.

Filters are used in almost every signal processing application – from audio editing, to building transducer response curves, removing noise from a signal, or "filtering out" important frequency content like speech, from a noisy signal.

ii. How do you describe the characteristic of a Filter?

The characteristic of a filter describes how the filter affects the signal in the context of the frequency content of the signal.

Here's how we understand what the filter's characteristics are:

Take the Impulse Response of the filter by passing a direc delta signal through the filter, since a direc delta impulse contains all frequencies.
Analyse the response in the frequency domain using a Fourier Transform.
This shows us how the different frequencies are affected (like their magnitude and phase).
The filtered output is then brough to the Time domain using an "Inverse Fourier Transform".
When analysing a filter, we are characterising its transfer function which is basically, the Impulse Response.

iii. How to apply filters to signals

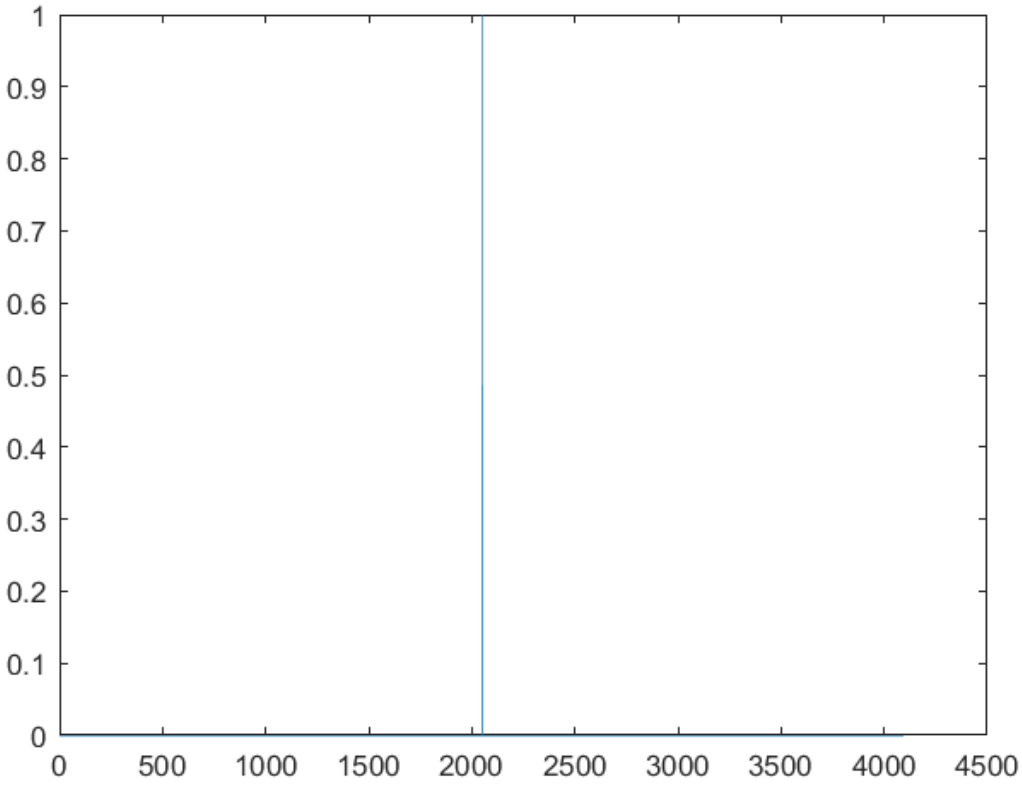
Filter can be applied to a signal in three different ways: Time-based Convolution, Frequency-based Convolution, Time-Delay Filters (FIR & IIR Filters).

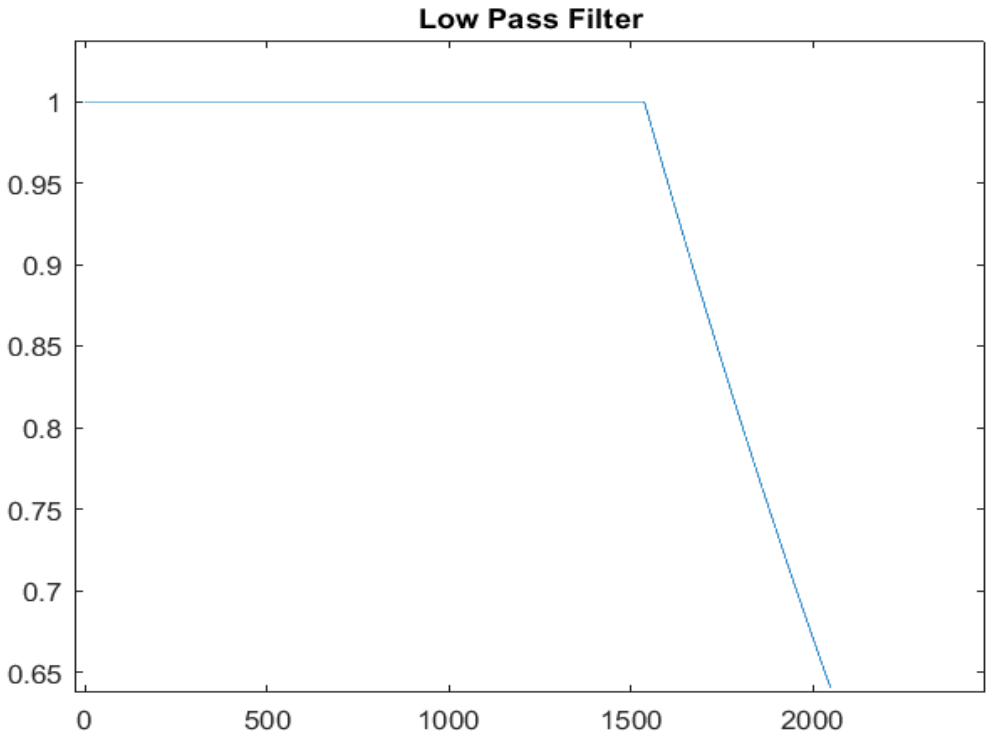
Describe Time-based Convolution

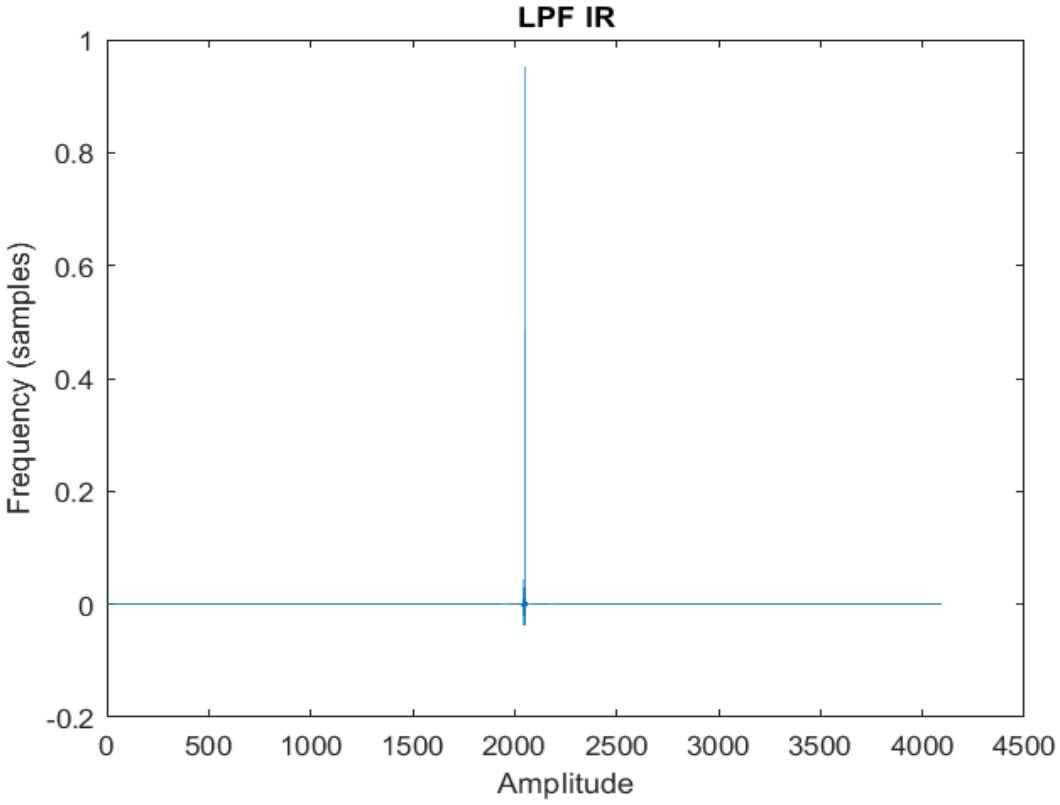
You need the impulse response of the filter in the Time Domain. There are two ways to go about it.

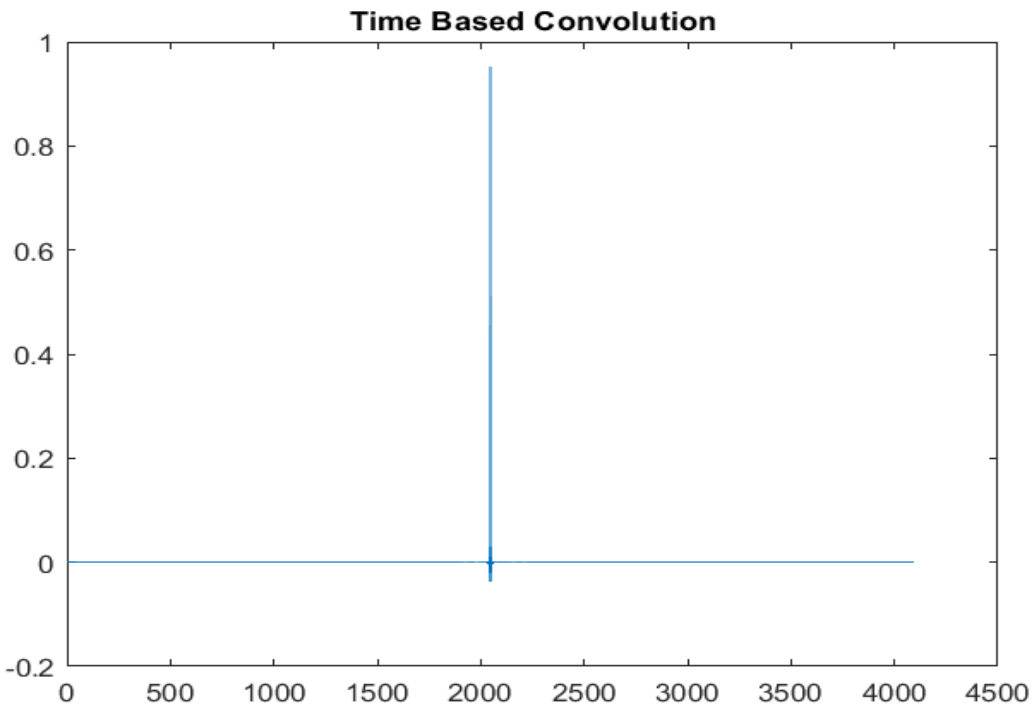
1. Using impulse responses of already existing filters, converting those IRs to the time domain to get the impulse signal.
2. Take an impulse measurement signal, convert it to the frequency domain using the Fourier Transform, adjust the frequencies to liking, then convert back to the time domain using an inverse Fourier Transform.
The length of the impulse response in the Time Domain must be the same as the length of the buffer or the length of the signal. The output of this filter should also be the same length as the buffer or signal.
Time-based convolution is a mathematical convolution of an impulse signal (h(t)) and an input signal (x(t)).

Do the MATLAB example of a Time-based Convolution and insert the plots here.




Low Pass Filter
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0
500
1000
1500
2000


LPF IR
Frequency (samples)
1
0.8
0.6
0.4
0.2
0
-0.2
0
500
1000
1500
2000
2500
3000
3500
4000
4500
Amplitude

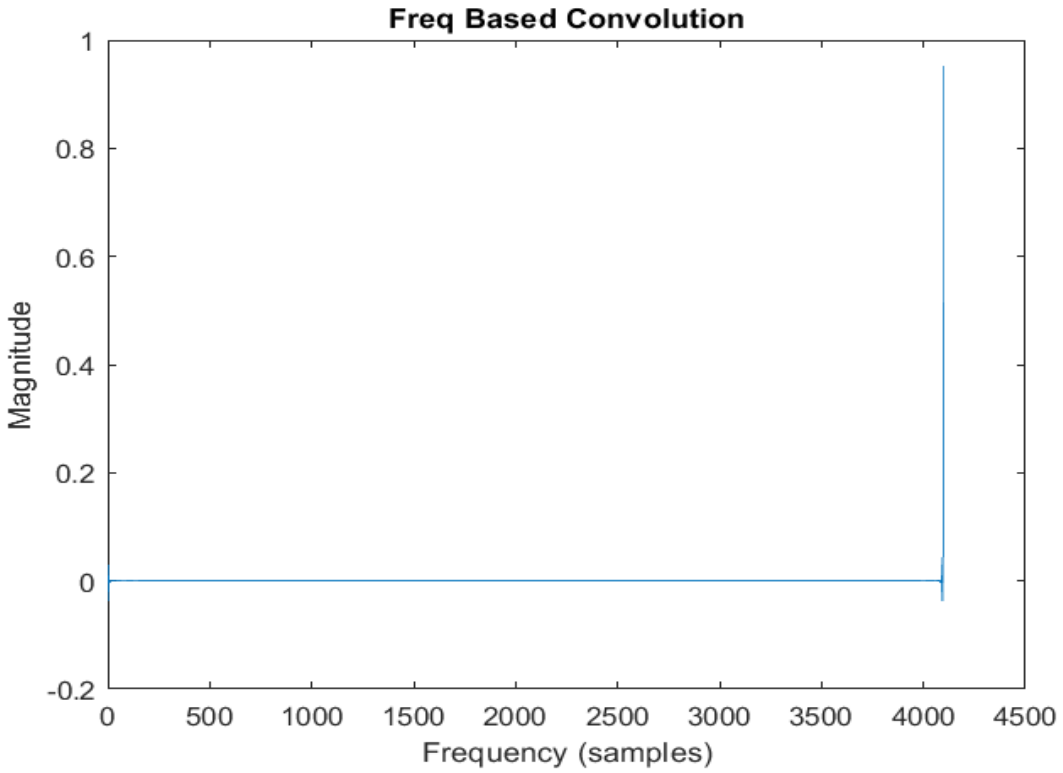


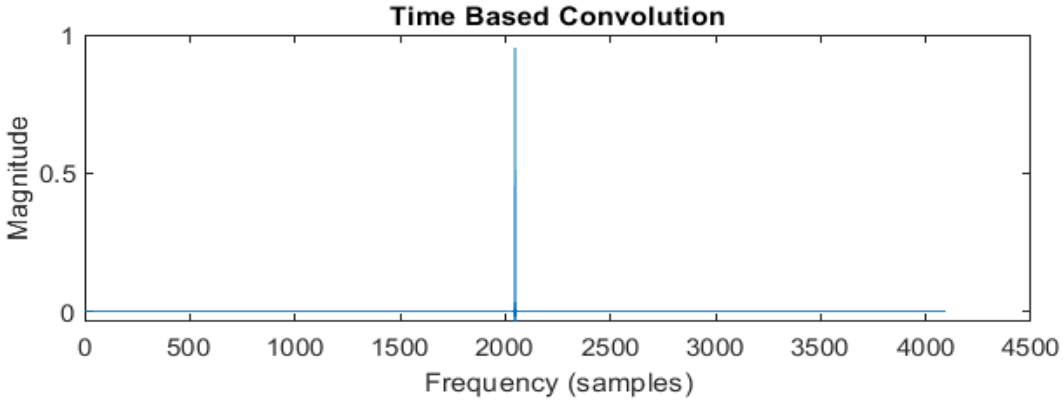
Describe Frequency-based Convolution.

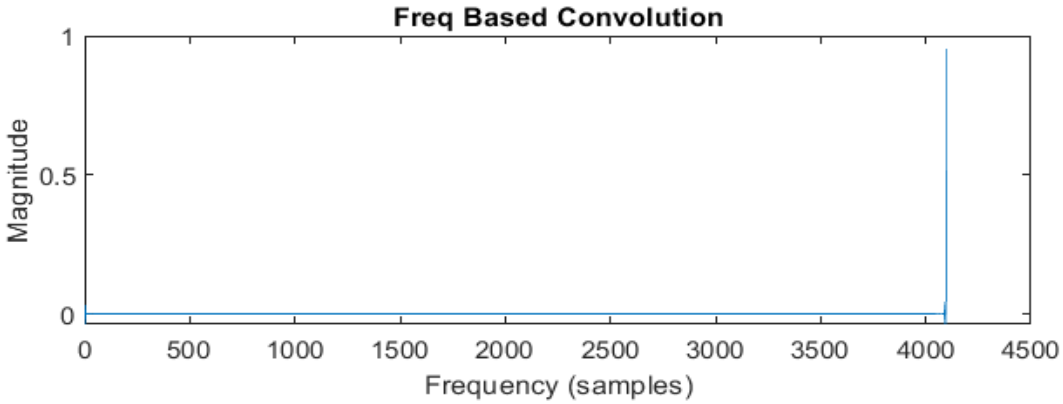
In the frequency domain, “convolution” is multiplication.

If the impulse response of a desired filter is needed in the frequency domain, apply the impulse signal to the filter by converting into the frequency domain using a Fourier Transform. Then multiply the frequency domain impulse response and the signal. After using the inverse Fourier Transform, the filtered signal is obtained.

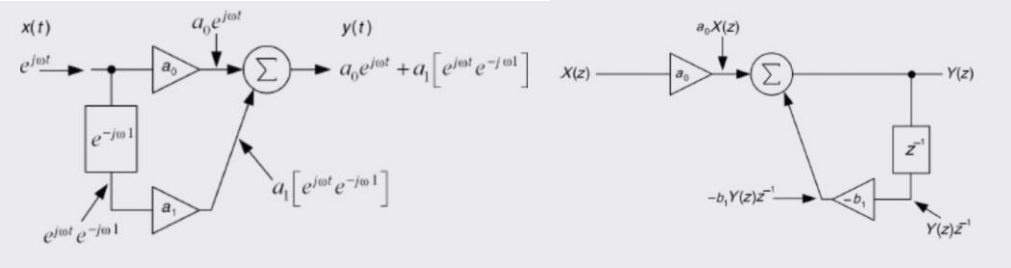
Do the MATLAB example of a Frequency-based Convolution and insert the plots here.







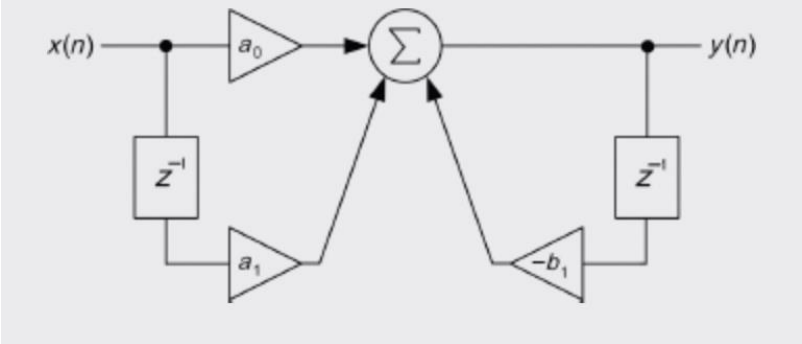
b. **Time-Delay Filters**



Time-Delay Filters used the concept that if you add delayed parts of the signals either as a feed-forward or feed-back, you can attenuate or amplify frequencies in a signal. The upside to this strategy is that it's a very fast method of applying filters. The downside is that more feed-forward or feedback lines can be needed to get the desired filter effect, which increases the algorithmic complexity. However, it can still be faster than the other approaches. When a filter structure uses only feed-forward lines, it is called a Finite Impulse Response (FIR) because the delays will eventually cause the signal to go zero once the signal ends. When the filter design includes a feedback line (even if there are feed-forward), it is called an Infinite Impulse Response (IIR) Filter because the feed-back delay can go on for infinity a proper attenuation on that feed-back line is not applied.

i. Difference Equation

The design of time delay filters allows easy formulation of the equation that is operated on the signal. This equation of the filter operation is called the difference equation and it's easy to generate it from looking at the filter block diagram.



Define the difference equation of the above block diagram of a shelving filter:

y(n) = a0*x(n) + a1*x(n-1) – b1*y(n-1)

ii. Convert Difference Equation to Z-Domain and get H(z).

You do this by moving the feed-back lines with the output and the feedforward lines with the input. Isolate y[n] and x[n] as a single term in the equation order to get y[n] * … and x[n] *… Then rearrange the terms so that you have y[n]/x[n]. Convert the terms to the z-domain and you have your H(z).

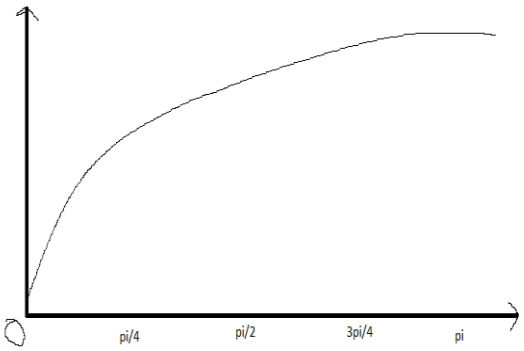
Write out the H(z) for the above difference equation.

$H(z) = (a0 + a1*z^{-1}) / (1 - b1*z^{-1})$

iii. Draw Magnitude Response of Poles and Zeros

Convert the coefficients in your in H(z) equation to Poles and Zeros. Define where the zero and pole are located in on your Unit Circle. Choose angle 0 and place the pole at 0.3 on the real axis and the zero at 0.7 on the real axis.

Draw out the expected magnitude response based on the chosen pole and zero.



iv. Compute coefficients

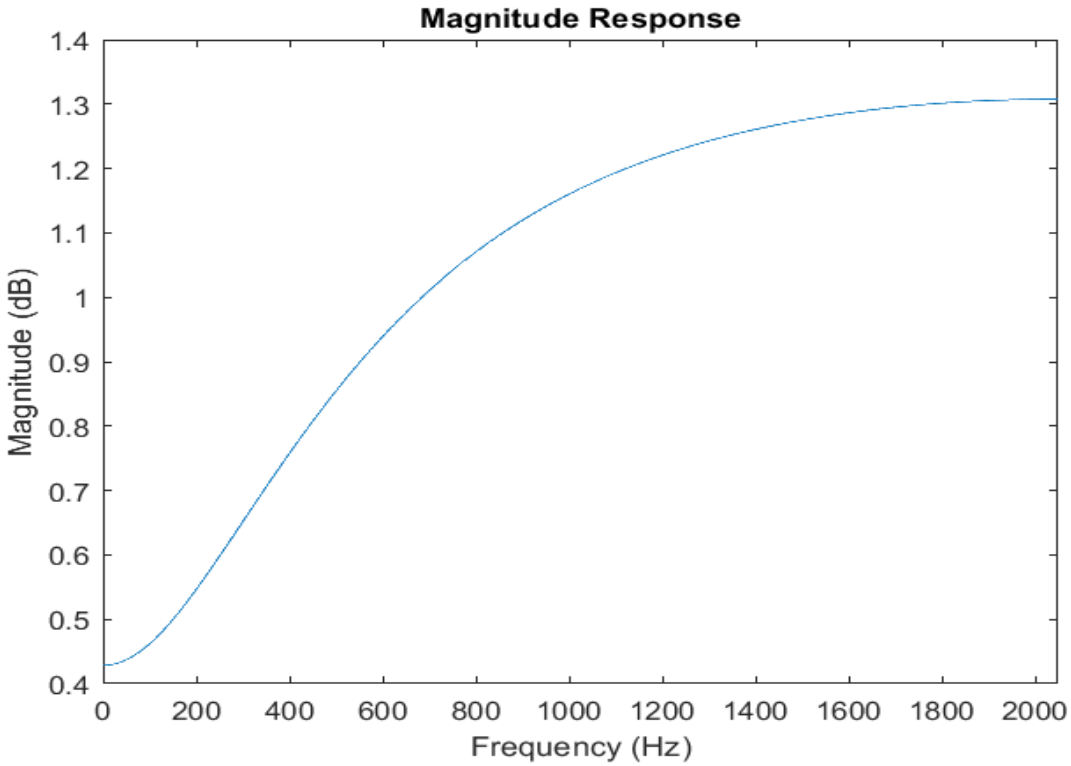
A1 = -Rz*cos(theta) = -0.7*cos(0) = -0.7
B1 = -Rp*cos(theta) = -0.3*cos(0) = -0.3

v. Apply the coefficients to the difference equation

Use the computed coefficients and apply them to the difference equation.

In MATLAB, use the dirac delta function and pass it through this difference equation.

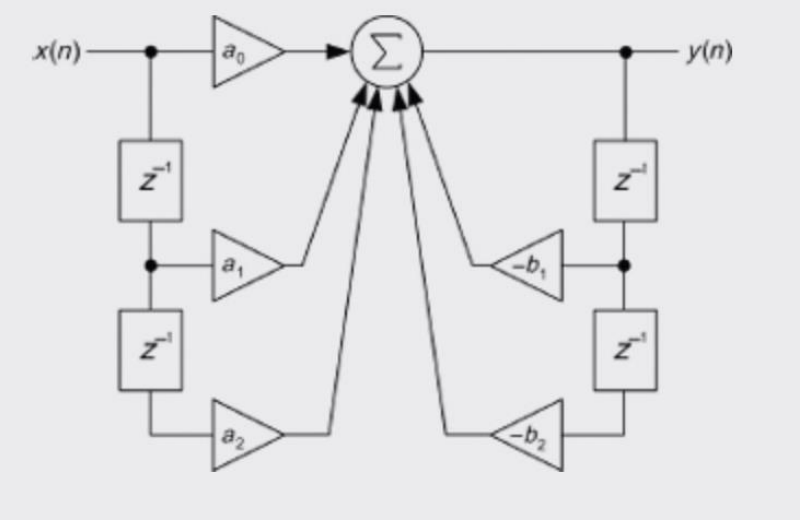
Take the FFT of the output and plot the magnitude. Compare it to your drawing that you made above.



2) **Advanced Portion**

a. Define the Direct Form 1 Biquad Filter Difference Equation

The Biquad Filter consists of two feed-forward lines and two feed-back lines allowing you to design which frequencies can be amplified and which frequencies can be attenuated in the same filter design. This allows more precision in what you can accomplish with this filter design.



Define the difference equation for this biquad filter.

y[n] = a0*x[n] + a1*x[n-1] + a2*x[n-2] – b1*y[n-1] – b2*y[n-2]

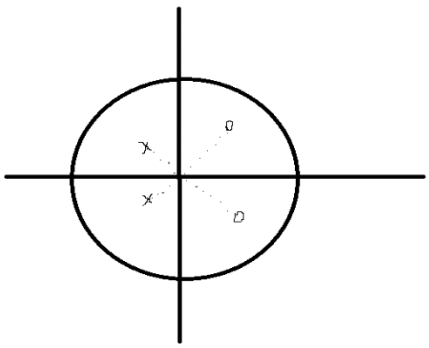
b. Convert the equation to the Z-domain and get H(z)

$H(z) = (a0 + a1*z^{-1} + a2*z^{-2}) / (1 + b1*z^{-1} + b2*z^{-2})$

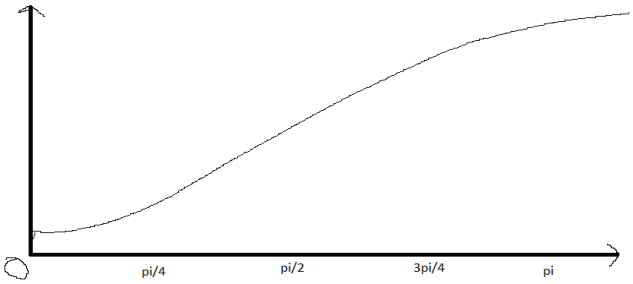
c. Draw the Magnitude Response with the Given Poles and Zeros

Remember in this biquad filter design, the poles and zeros must be conjugate pairs of each other. This allows you to remove the imaginary part out of the equation in order to compute the coefficients. Use the angles Pi/4 and –Pi/4 for the zeros and ¾*Pi and –¾*Pi for the poles. Use an R value of 0.5 for the Zeros and 0.25 for the Poles.

Draw the Unit Circle with the Poles and Zeros.



Draw out the estimated Magnitude Response with these Poles and Zeros.



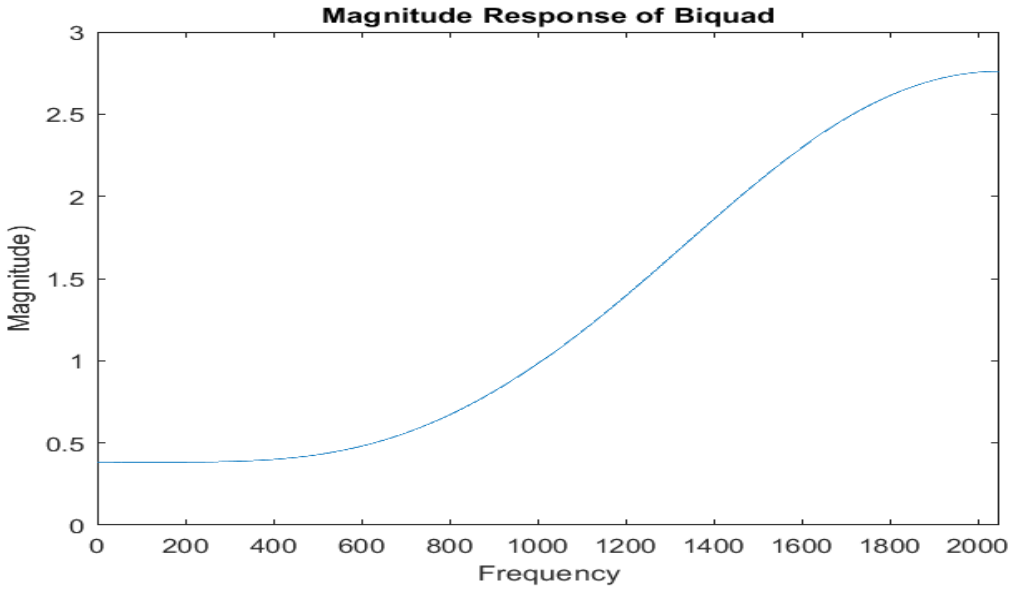
d. Compute the coefficients values for these Poles and Zeros

You can use the predefined equations we used in class to compute these values.

A1 = -2*Rz*cos(pi/4) = -2*(0.5)*cos(pi/4) = -0.7071
A2 = R^2 = 0.25
B1 = -2*Rp*cos(3pi/4) = -2*(0.25)*cos(3pi/4) = 0.3535
B2 = R^2 = 0.0625

e. Apply coefficients to the difference equation and test it.

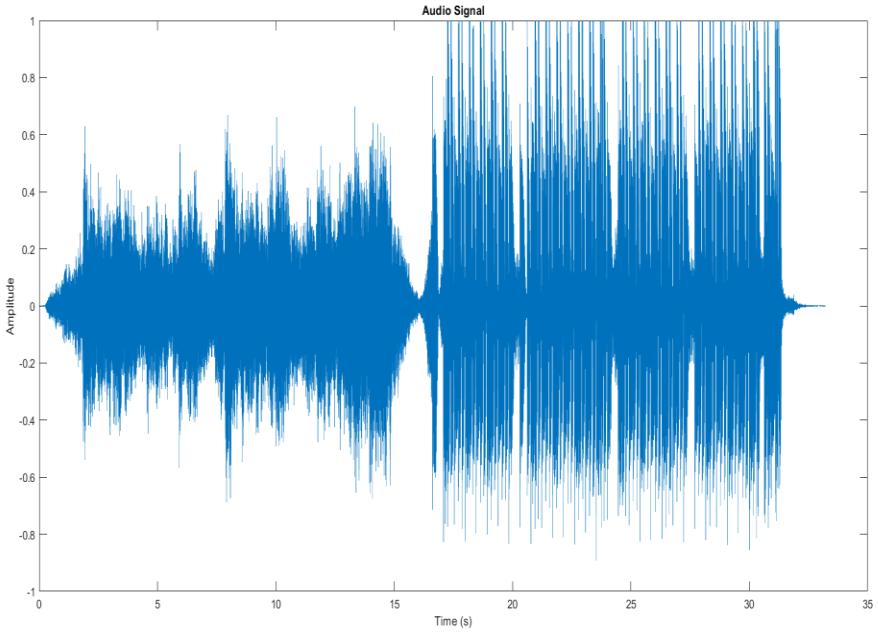
Apply the coefficients to the difference equation in MATLAB and use the dirac-delta impulse response to test the filter. Take the FFT of the output and plot the magnitude response. Compare it to the estimation you made above.

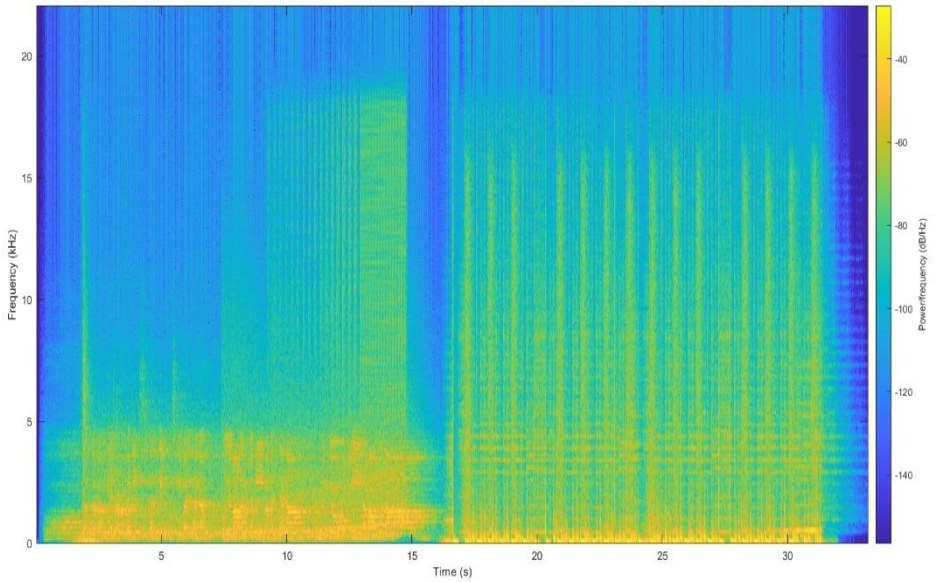


3) **Creative Portion**

a. Choose a song or a track

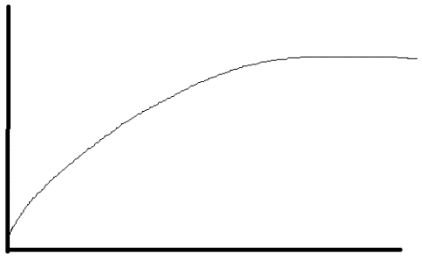
Insert the plots of the song and its spectrogram in MATLAB.





b. Choose a new set of zeros and poles for the shelving filter and pass the song through it.

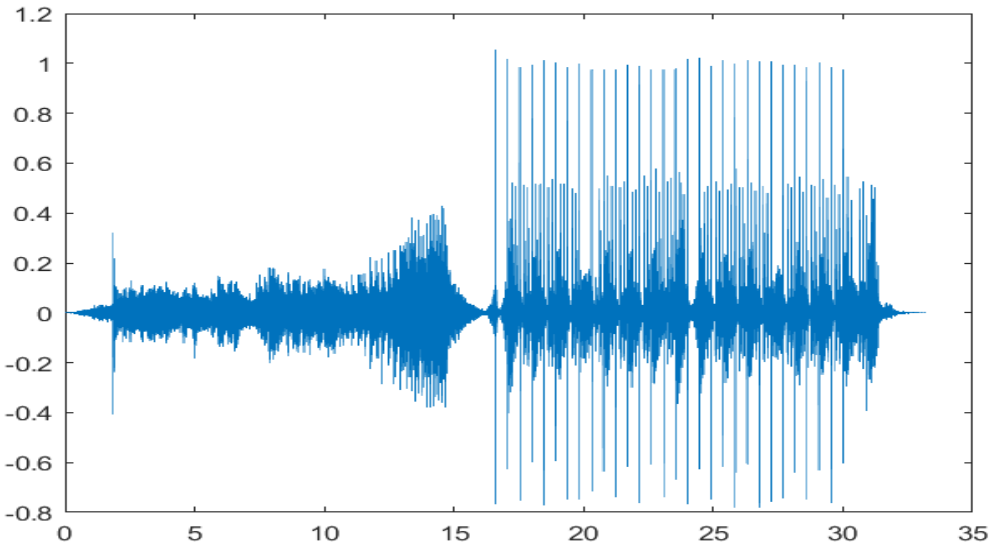
Draw the magnitude response of the new shelving filter here.

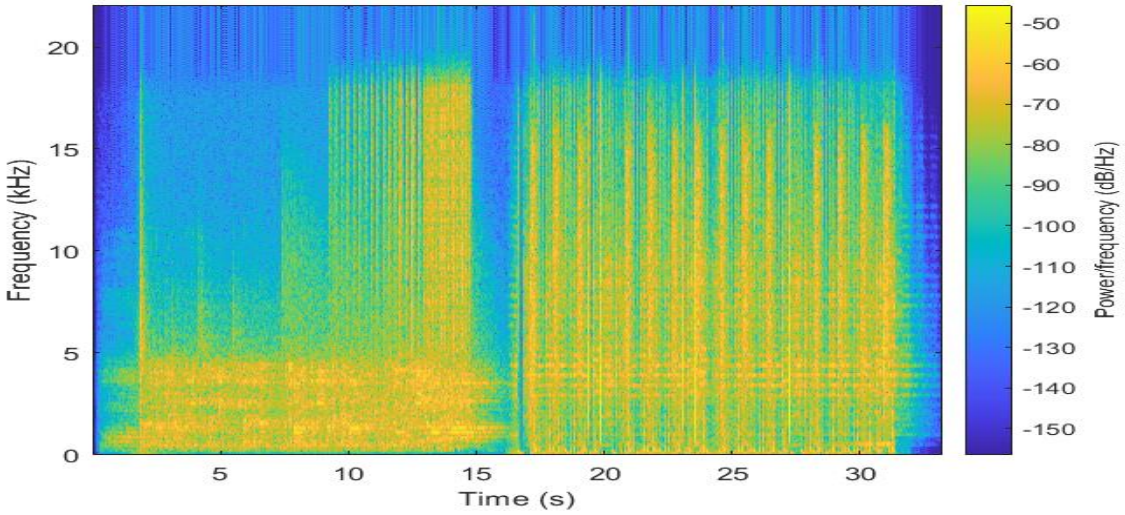


Define the new coefficients with the chosen poles.

A0 = 0.9, a1 = 0.9, b1 = 0.2

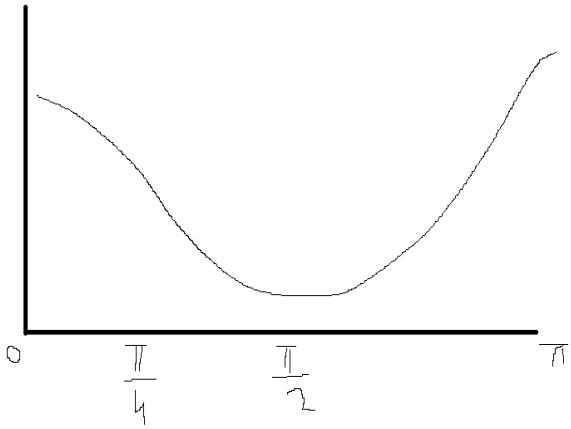
Pass the song or track in the new filter and insert the plot the output and its spectrogram in MATLAB here.





c. Choose a new set of zeros and poles for the biquad filter and pass the song through it.

Draw the magnitude response of the new biquad filter here.



Define the new coefficients with the chosen poles.

```
rb_z = 0.9

rb_p = 0.1;

theta_bz = pi/2;

theta_bp = (3*pi)/4;

a0 = 1;
a1 = -2*rb_z*cos(theta_bz);
a2 = rb_z*rb_z;
```

```
b1 = -2*rb_p*cos(theta_bp);
b2 = rb_p * rb_p;
```

Pass the song or track in the new filter and insert the plot the output and its spectrogram in MATLAB here.