Bruno E. Gracia Villalobos

EE 4643 – Digital Signal Processing

Professor Zhang

Final Project - FIR and IIR Filtering of WAV Files

December 11, 2019

Introduction

This project aims to understand how both Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) filters can remove unwanted noise due to mechanical, electrical or environmental factors. Custom IIR and FIR filters are designed to filter a signal in three distinct noise environments using MATLAB filterDesigner. Fast Fourier Transforms (FFTs) are used to compare the original signal to the filtered signal in the frequency domain. The filtered signals are exported to WAV files for reference.

Procedure

FIR and IIR Filter Design for First Environment

The WAV "df3_n0H.wav" file is analyzed first. Before utilizing MATLAB's *filterDesigner*, the signal is visualized in the frequency domain using the Digital Audio Workstation (DAW) FL Studio by Image-Line software. The plugin *Fruity Parametric EQ 2* is used to establish an intuition as to where the noise source might reside. Figure 1 shows the DAW's processing of the signal's frequency spectrum. Figure 1 shows a peculiar frequency component around the 3000 Hz range.

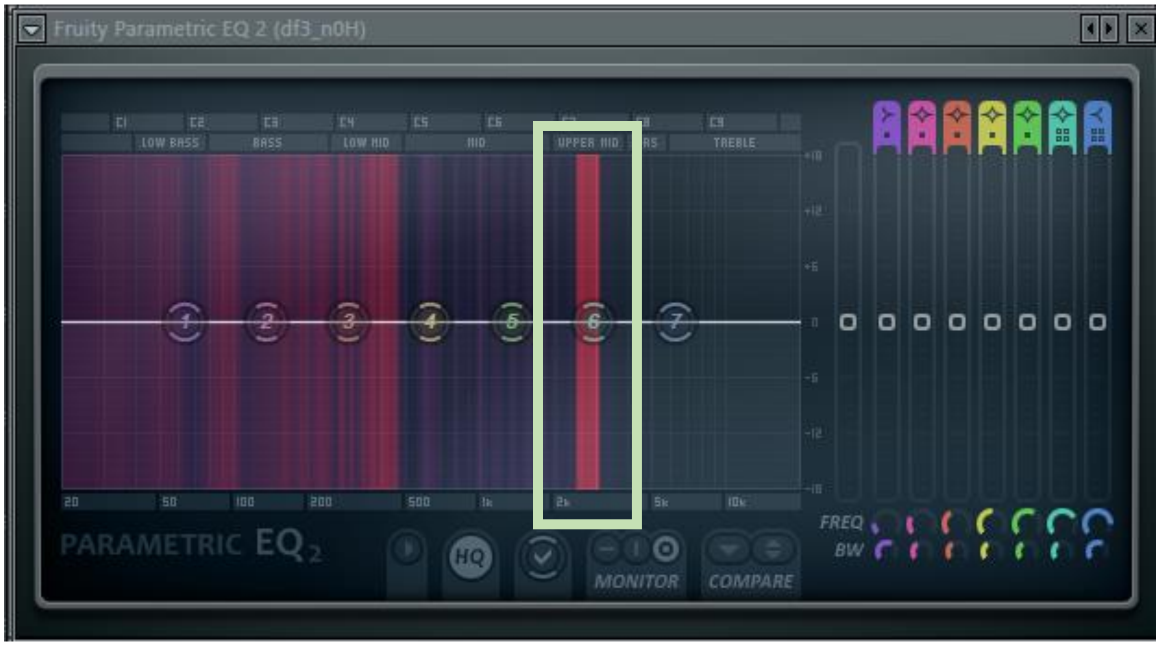
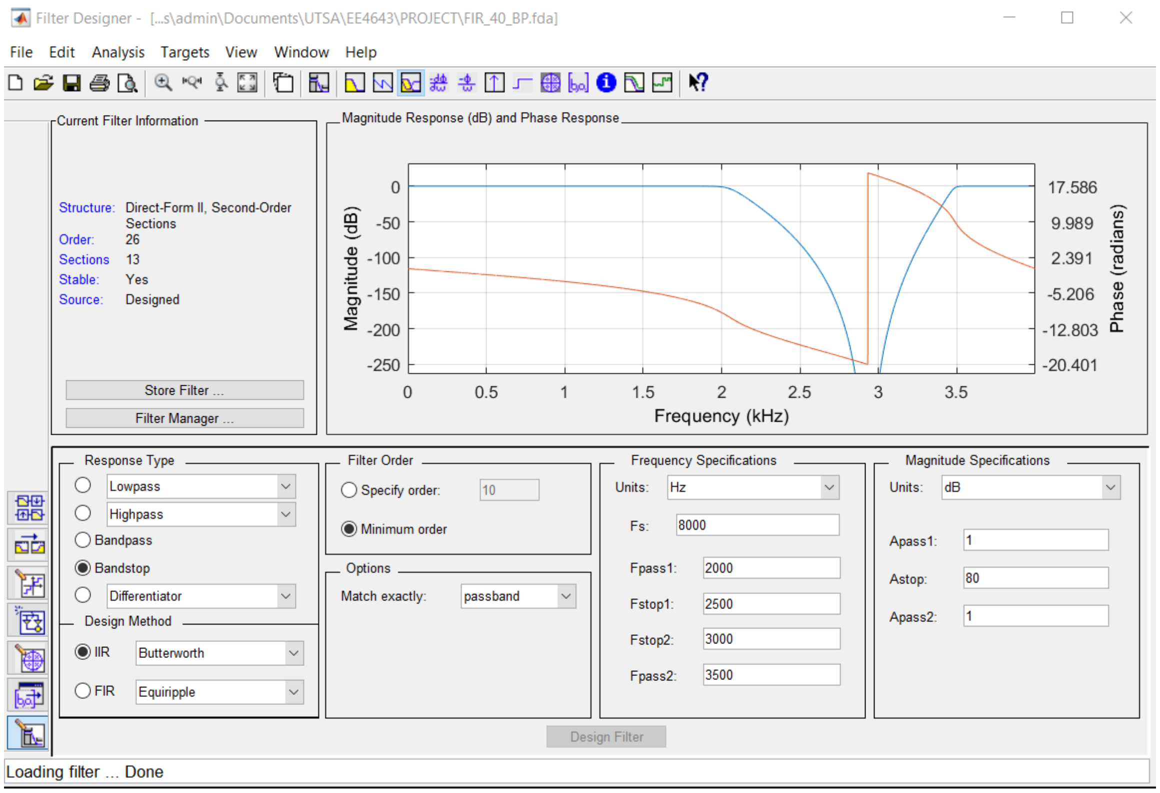


Figure 1: Frequency spectrum of df3_n0H in FL Studio. Noise outlined in green.

Now, this intuition is used to design an **IIR Bandstop Butterworth 26th Order Filter** in MATLAB *filterDesigner* to remove the unwanted noise. Figures 2-6 showcase the filter's specifications. With a bandpass between 2.5kHz and 3.5kHz, the IIR filter effectively removes the noise as shown in Figure 1.



Current Filter Information

Structure:	Direct-Form II, Second-Order Sections
Order:	26
Sections	13
Stable:	Yes
Source:	Designed

Response Type: ○ Lowpass ○ Highpass ○ Bandpass [✓ Bandstop] ○ Differentiator

Design Method: [✓ IIR] Butterworth ○ FIR Equiripple

Filter Order: ○ Specify order: 10 [✓ Minimum order]

Options: Match exactly: passband

Frequency Specifications

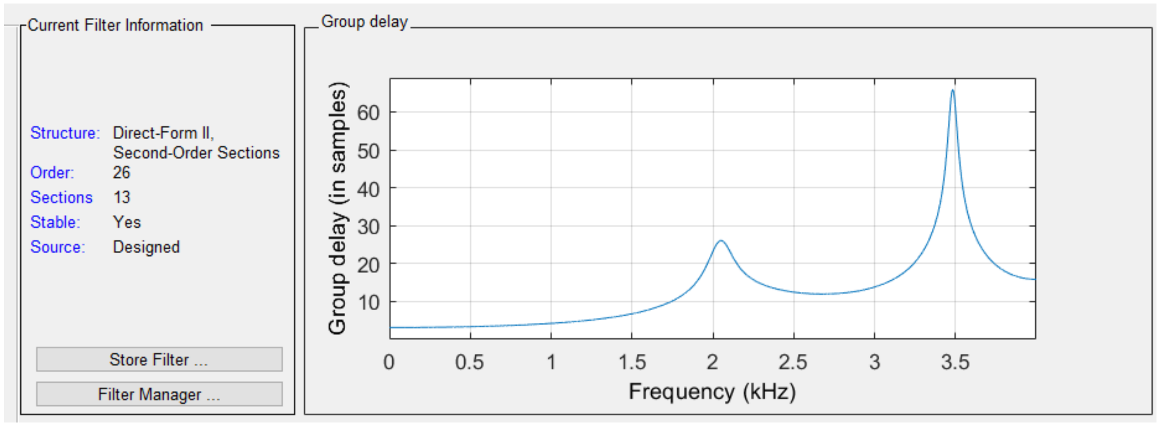
Units:	Hz
Fs:	8000
Fpass1:	2000
Fstop1:	2500
Fstop2:	3000
Fpass2:	3500

Magnitude Specifications

Units:	dB
Apass1:	1
Astop:	80
Apass2:	1

Loading filter ... Done

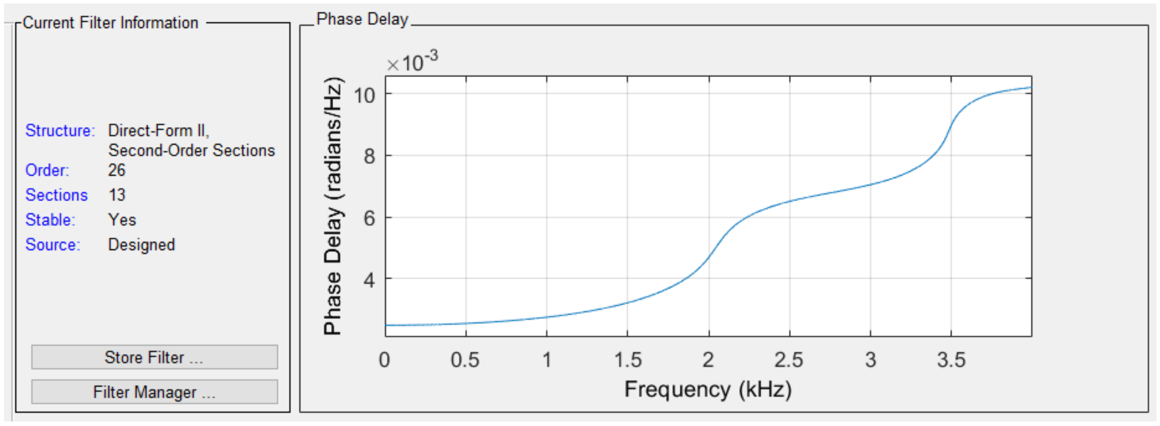
Figure 2: Magnitude and Phase IIR Bandstop Butterworth 26th Order Filter



Current Filter Information

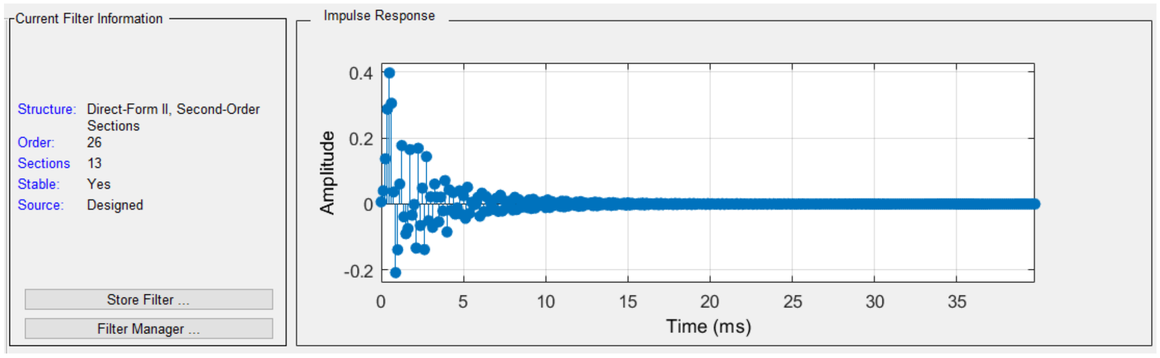
Structure:	Direct-Form II, Second-Order Sections
Order:	26
Sections	13
Stable:	Yes
Source:	Designed

Figure 3: Group delay for IIR Bandstop Butterworth 26th Order Filter


Current Filter Information

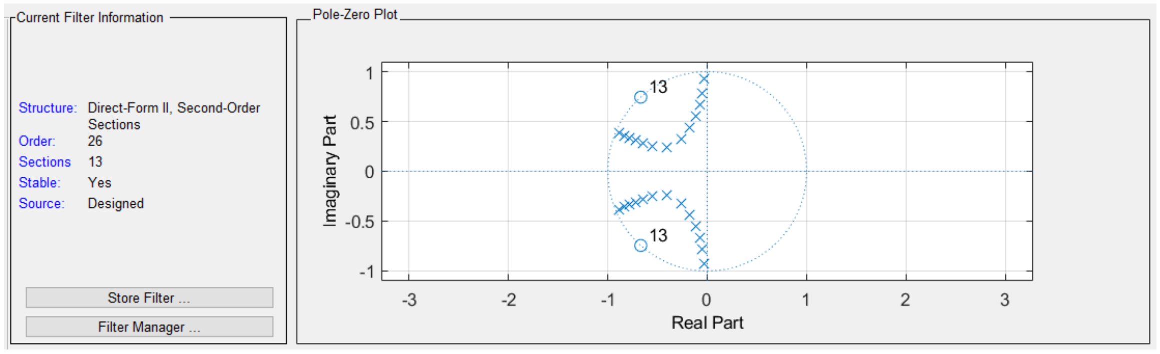
Structure: Direct-Form II, Second-Order Sections
Order: 26
Sections 13
Stable: Yes
Source: Designed

Figure 4: Phase delay for IIR Bandstop Butterworth 26th Order Filter


Current Filter Information

Structure: Direct-Form II, Second-Order Sections
Order: 26
Sections 13
Stable: Yes
Source: Designed

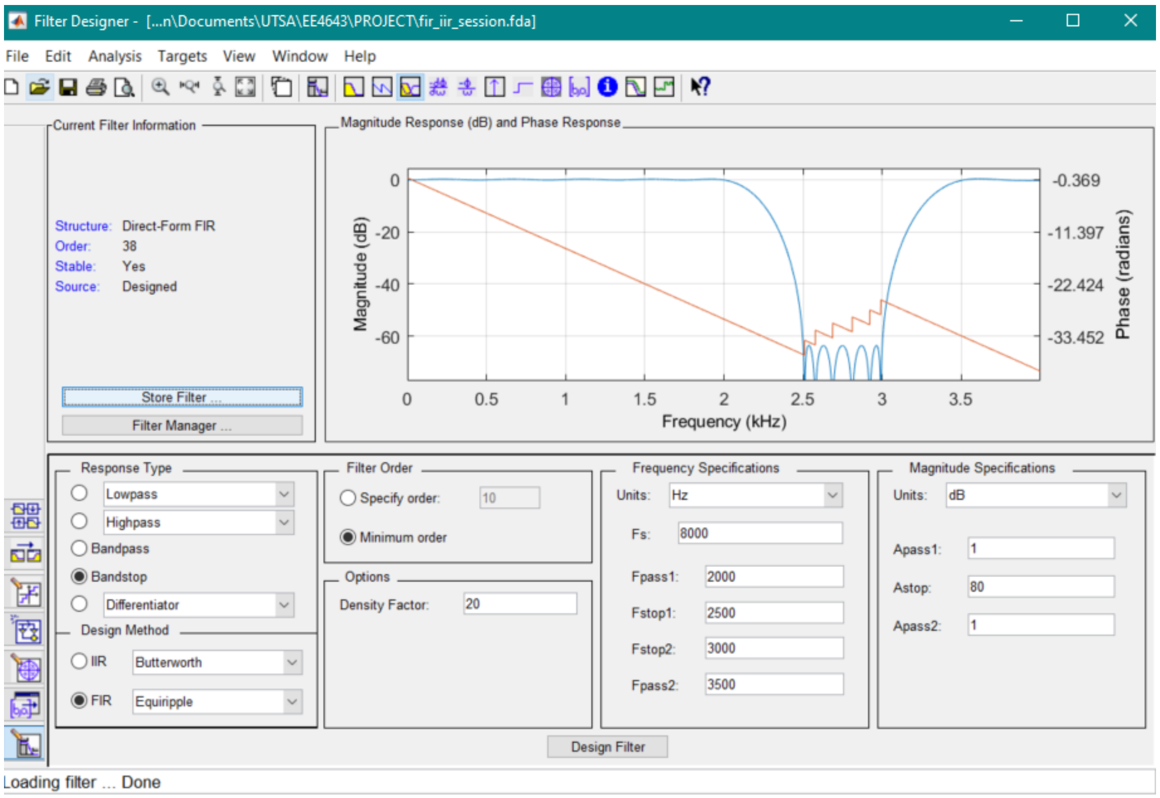
Figure 5: Impulse Response for IIR Bandstop Butterworth 26th Order Filter


Current Filter Information

Structure: Direct-Form II, Second-Order Sections
Order: 26
Sections 13
Stable: Yes
Source: Designed

Figure 6: Pole-zero plot for IIR Bandstop Butterworth 26th Order Filter

To better compare the efficacy of IIR vs FIR, the following design creates a **FIR bandstop, equiripple 38th order filter** using the *filterDesigner* with the same specs as the IIR filter—rolloff, pass and stop bands. Figures 7-11 showcase the FIR filter design specs.


Filter Designer - [...n\Documents\UTSA\EE4643\PROJECT\fir_iir_session.fda]
File Edit Analysis Targets View Window Help
Magnitude Response (dB) and Phase Response
Magnitude (dB)
0
-20
-40
-60
-0.369
-11.397
-22.424
-33.452
Phase (radians)
0 0.5 1 1.5 2 2.5 3 3.5
Frequency (kHz)
Store Filter ...
Filter Manager ...
Design Filter

Current Filter Information

Structure:	Direct-Form FIR
Order:	38
Stable:	Yes
Source:	Designed

Response Type: Lowpass, Highpass, Bandpass, [✓ Bandstop], Differentiator

Design Method: IIR (Butterworth), [✓ FIR] (Equiripple)

Filter Order: Specify order: 10, [✓ Minimum order]

Options: Density Factor: 20

Frequency Specifications

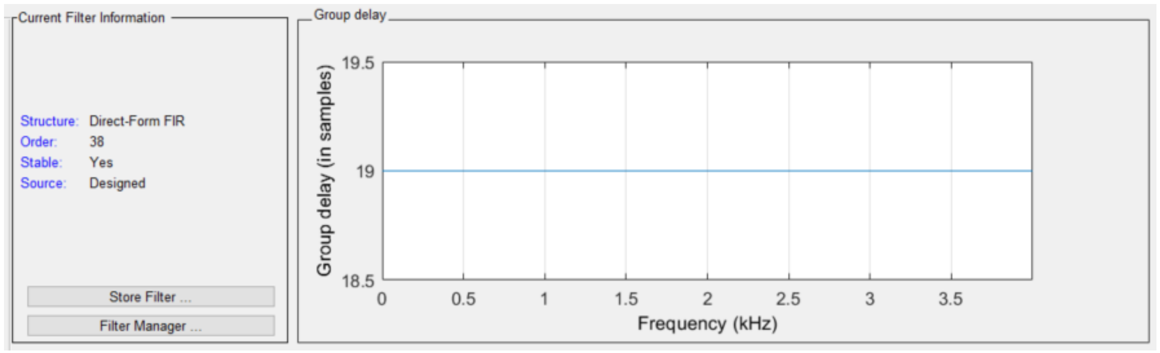
Units:	Hz
Fs:	8000
Fpass1:	2000
Fstop1:	2500
Fstop2:	3000
Fpass2:	3500

Magnitude Specifications

Units:	dB
Apass1:	1
Astop:	80
Apass2:	1

Loading filter ... Done

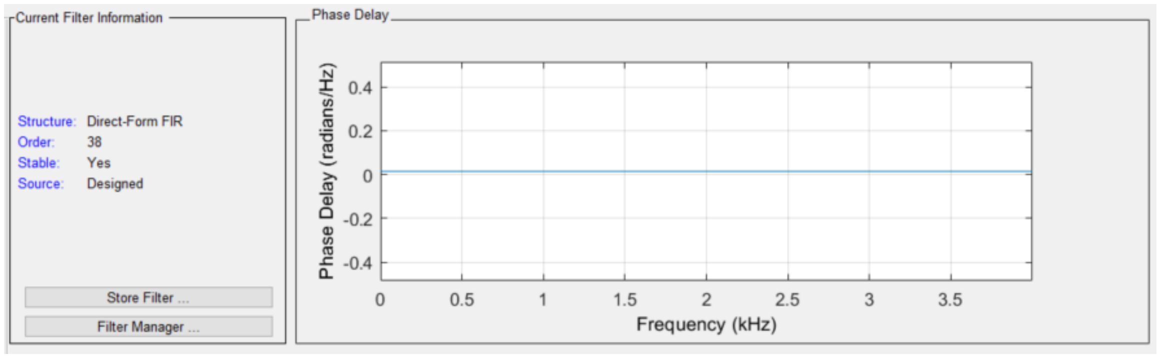
Figure 7: Magnitude and Phase response for FIR Filter



Current Filter Information

Structure:	Direct-Form FIR
Order:	38
Stable:	Yes
Source:	Designed

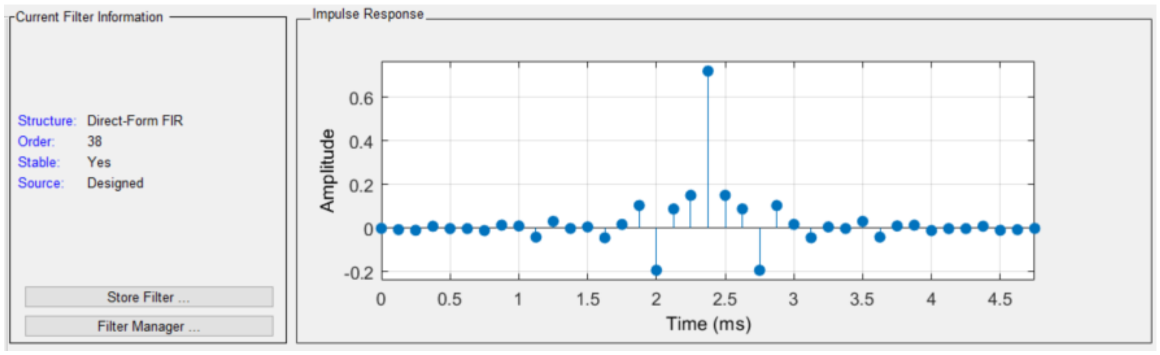
Figure 8: Group delay for FIR filter



Current Filter Information

Structure:	Direct-Form FIR
Order:	38
Stable:	Yes
Source:	Designed

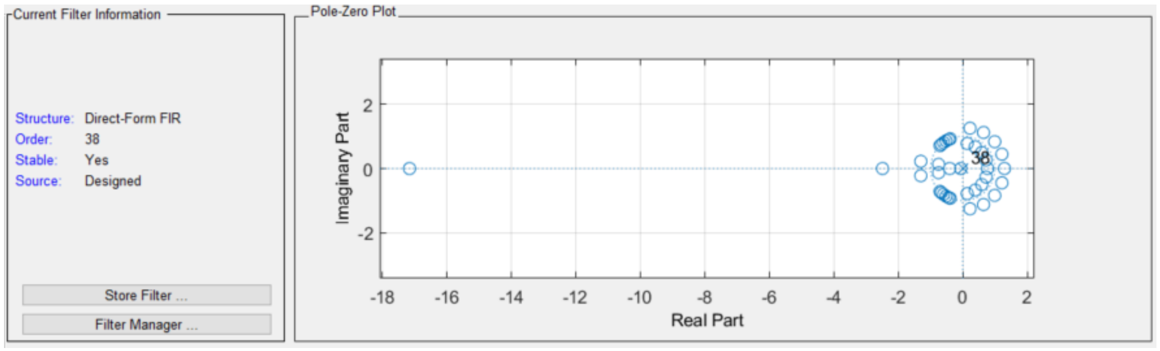
Figure 9: Phase delay for FIR filter



Current Filter Information

Structure: Direct-Form FIR
Order: 38
Stable: Yes
Source: Designed

Figure 10: Impulse response for FIR filter



Current Filter Information

Structure: Direct-Form FIR
Order: 38
Stable: Yes
Source: Designed

Figure 11: Pole-zero plot for FIR Filter

As confirmed by the FIR filter theory, this filter design does not contain any poles apart from the zero point as shown in Figure 11 due to the denominator of the transform equation being 1.

FIR and IIR Filter Design for Second Environment

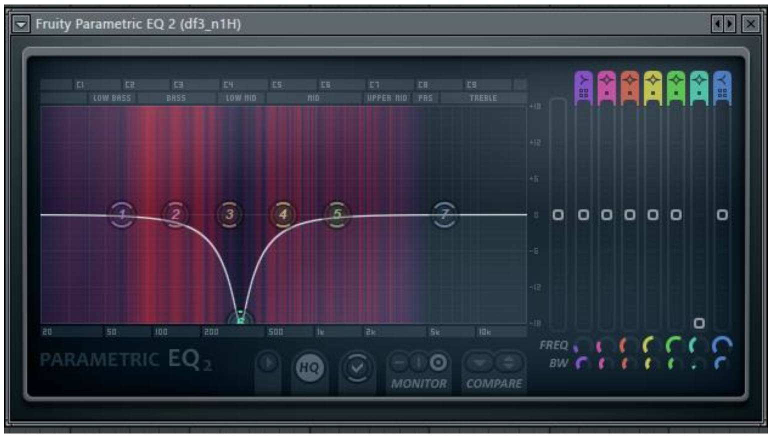
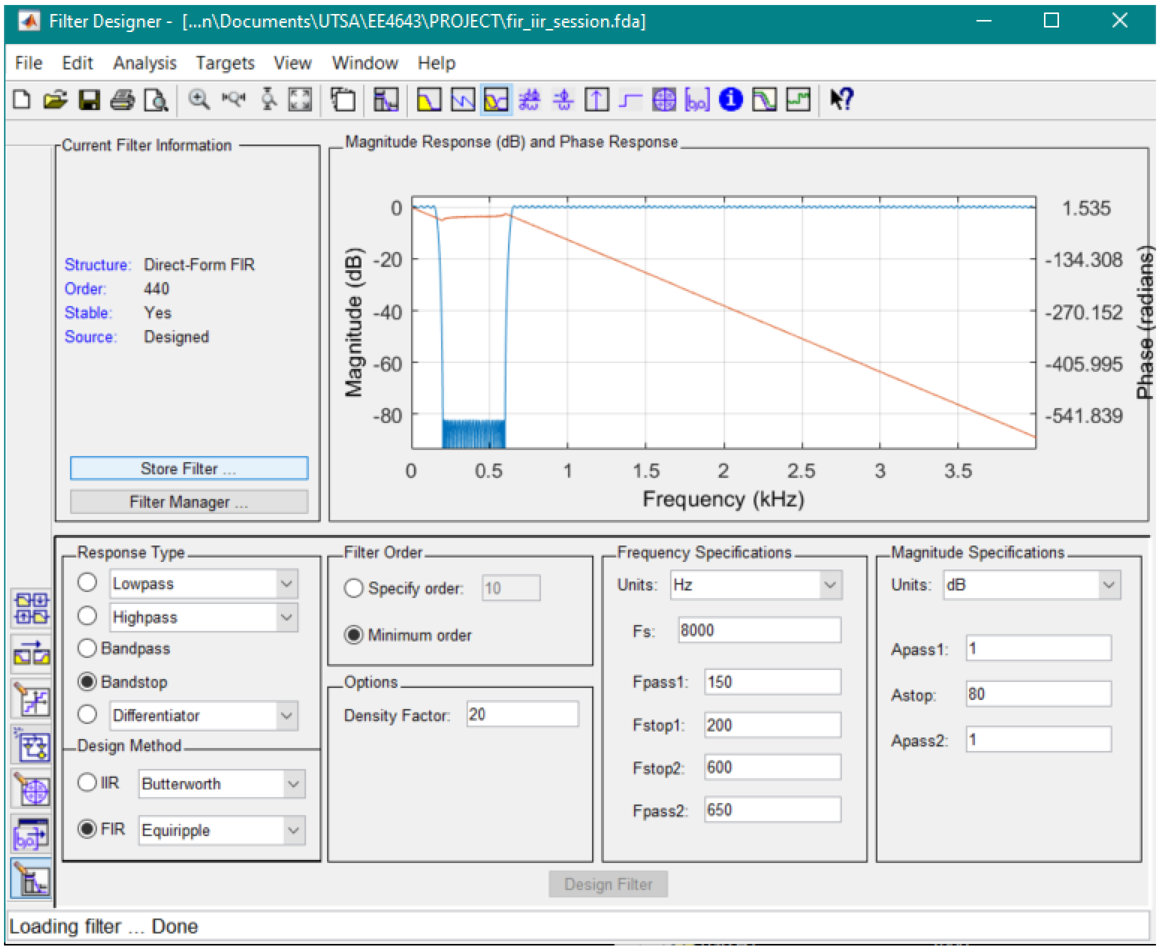


Figure 12: Looking for intuition in FL Studio

The next step is to design both FIR and IIR filters to process the noise in “df3_n1H.wav.” Once again, an intuition is created using FL Studio to better visualize and hear the signal’s noise frequency components. Although the noise is not visible as compared to the First environment, a viable choice is found between 200 and 600 Hz as shown in Figure 12 after sweeping the spectrum.

Now, a FIR filter is designed using the *filterDesigner*. The similar process is described below, and all the specifications for a **FIR bandstop, equiripple 440th order filter** are in Figures 13-18.



Current Filter Information

Structure:	Direct-Form FIR
Order:	440
Stable:	Yes
Source:	Designed

Response Type: Lowpass [✓ Bandstop] Highpass Bandpass Differentiator

Design Method: IIR Butterworth [✓ FIR] Equiripple

Filter Order: Specify order: 10 [✓ Minimum order]

Options: Density Factor: 20

Frequency Specifications

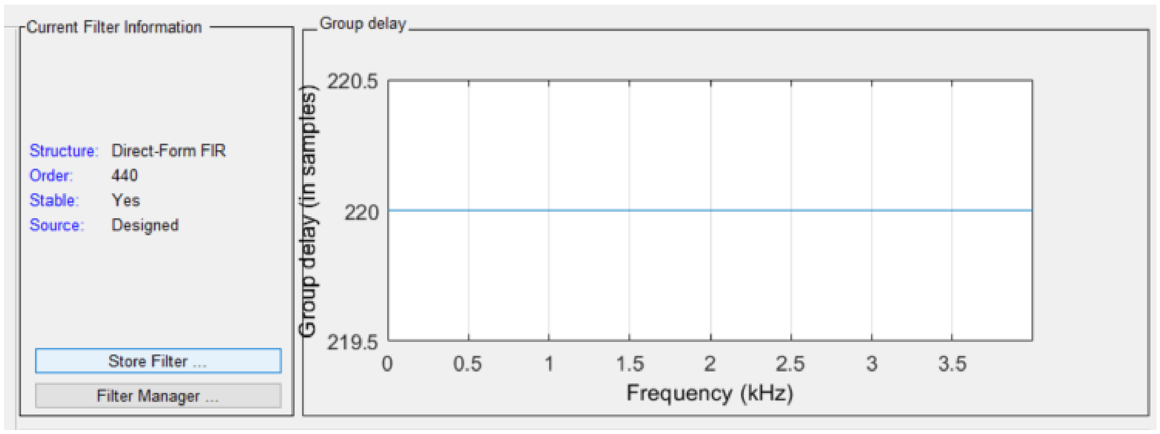
Units:	Hz
Fs:	8000
Fpass1:	150
Fstop1:	200
Fstop2:	600
Fpass2:	650

Magnitude Specifications

Units:	dB
Apass1:	1
Astop:	80
Apass2:	1

Loading filter ... Done

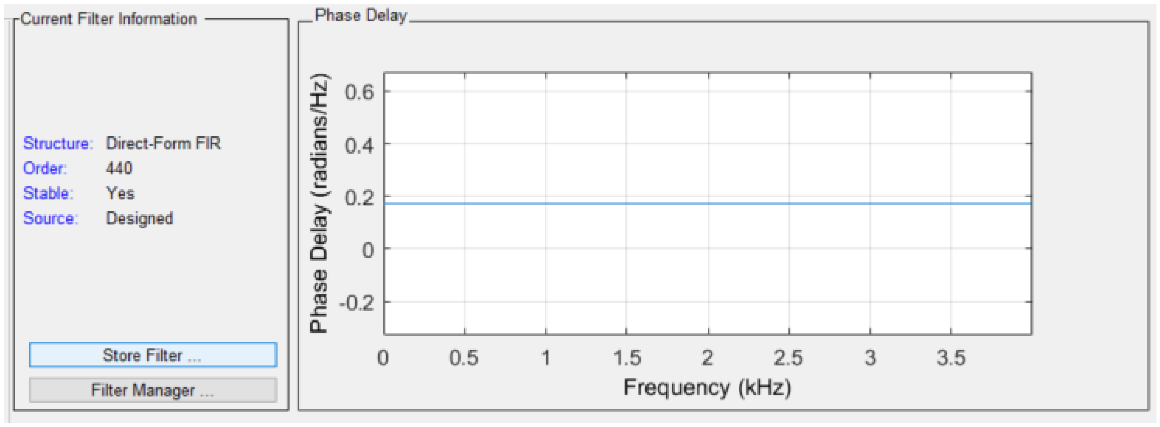
Figure 13: Magnitude and Phase response for FIR filter



Current Filter Information

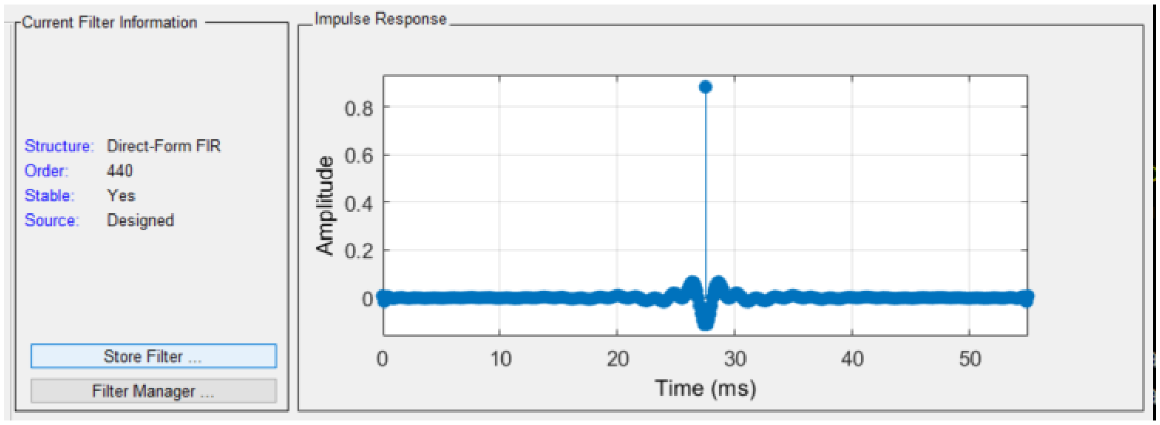
Structure:	Direct-Form FIR
Order:	440
Stable:	Yes
Source:	Designed

Figure 14: Group delay for FIR filter


Current Filter Information

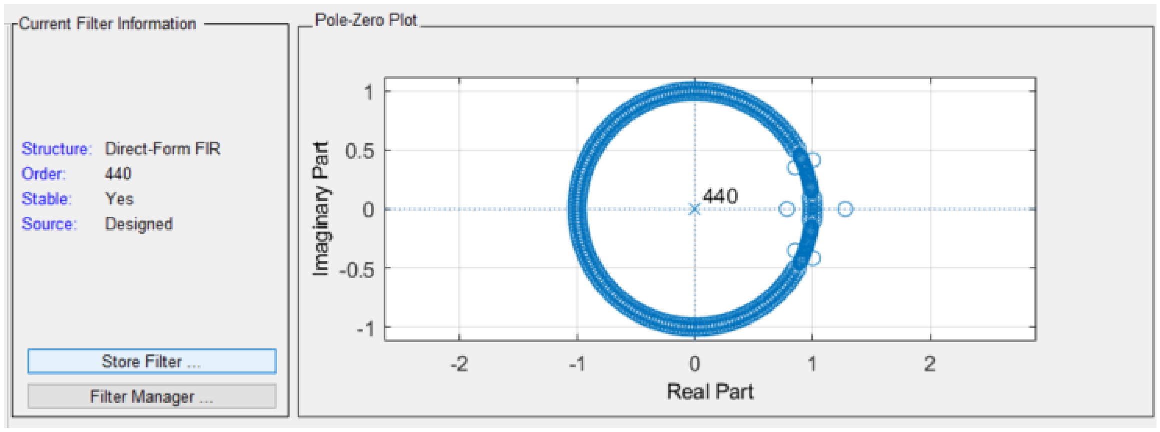
Structure: Direct-Form FIR
Order: 440
Stable: Yes
Source: Designed

Figure 15: Phase delay for FIR filter


Current Filter Information

Structure: Direct-Form FIR
Order: 440
Stable: Yes
Source: Designed

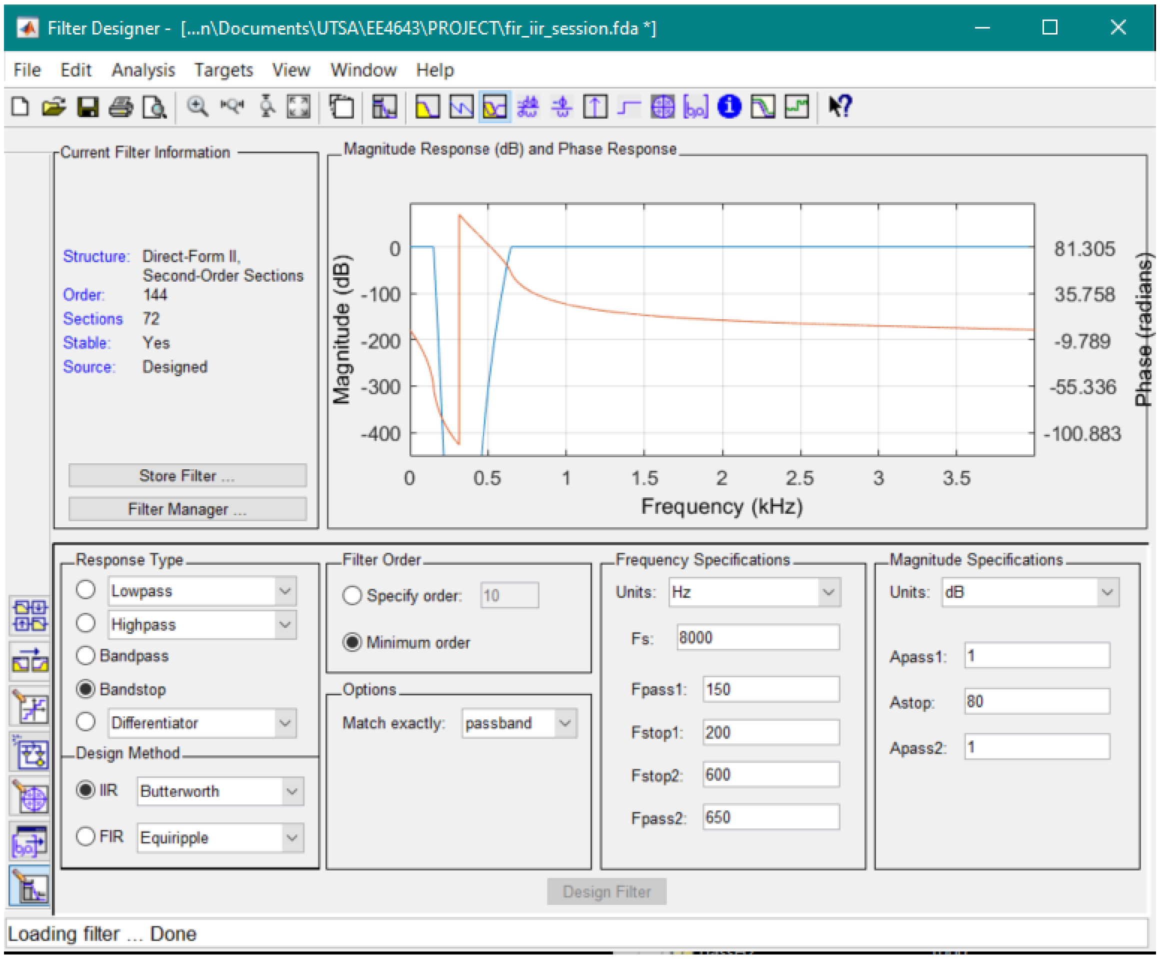
Figure 16: Impulse response for FIR filter


Current Filter Information

Structure: Direct-Form FIR
Order: 440
Stable: Yes
Source: Designed

Figure 17: Pole-zero plot for FIR filter

An **IIR, bandstop, Butterworth 144th order filter** is designed to compare the results with the FIR. Although the orders are highly disparate, it is important to mention both filters are designed using the “minimum order” specification as shown in Figures 13 and 18.



Current Filter Information

Structure:	Direct-Form II, Second-Order Sections
Order:	144
Sections	72
Stable:	Yes
Source:	Designed

Response Type: Lowpass [ ] Highpass [ ] Bandpass [ ] [✓ Bandstop] Differentiator [ ]

Design Method: [✓ IIR] Butterworth; FIR [ ] Equiripple

Filter Order: Specify order: [ ] 10; [✓ Minimum order]

Options: Match exactly: passband

Frequency Specifications

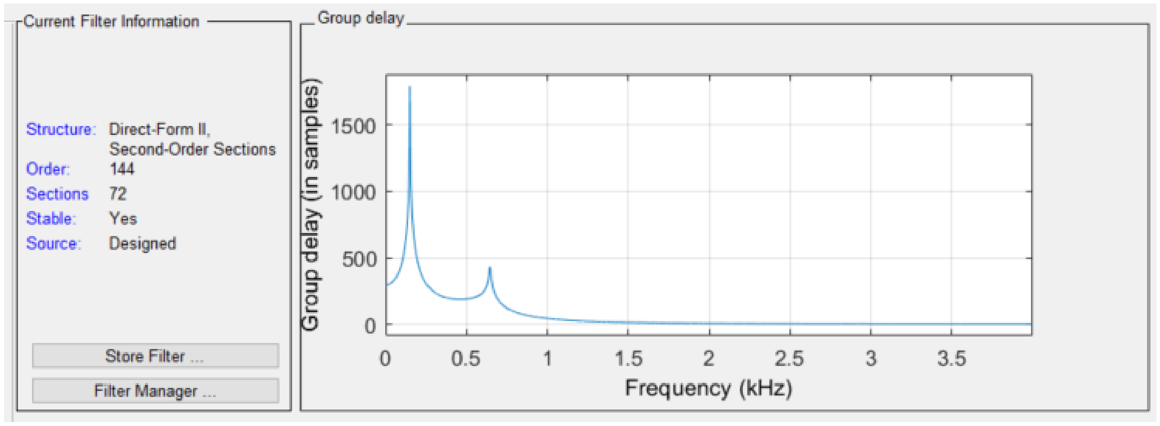
Units:	Hz
Fs:	8000
Fpass1:	150
Fstop1:	200
Fstop2:	600
Fpass2:	650

Magnitude Specifications

Units:	dB
Apass1:	1
Astop:	80
Apass2:	1

Loading filter ... Done

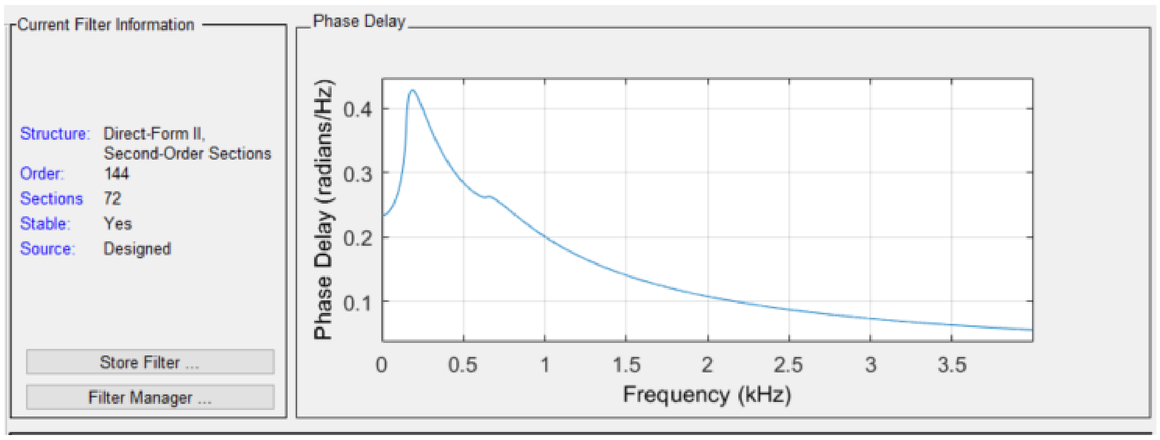
Figure 18: Magnitude and phase response for IIR filter



Current Filter Information

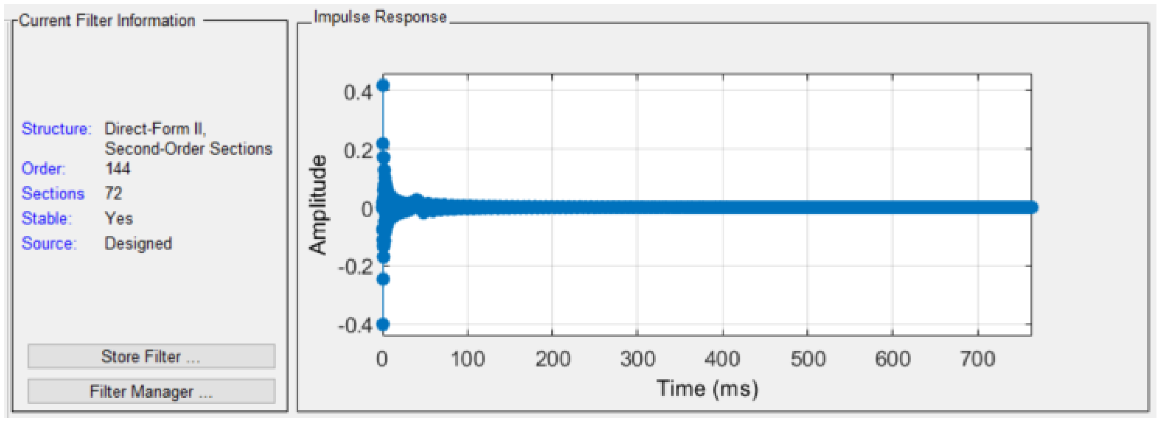
Structure:	Direct-Form II, Second-Order Sections
Order:	144
Sections	72
Stable:	Yes
Source:	Designed

Figure 19: Group delay for IIR filter


Current Filter Information

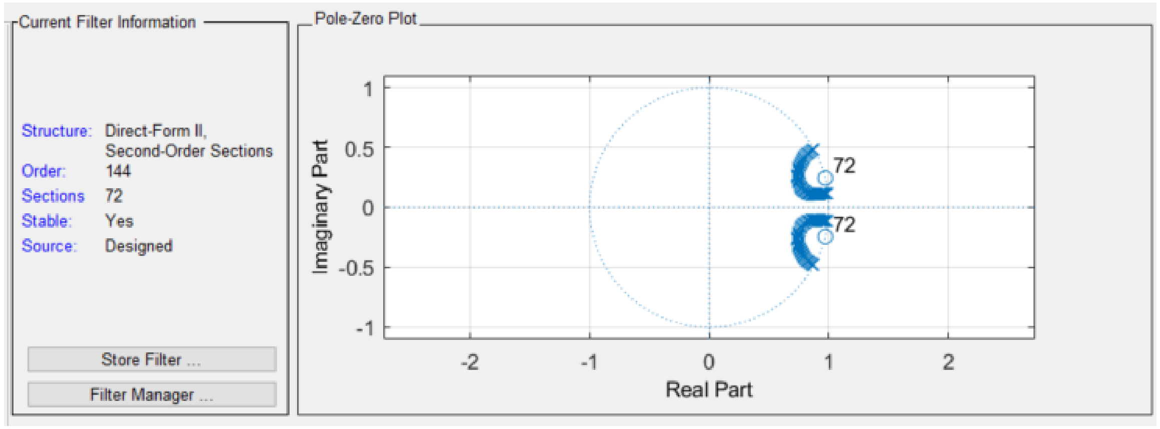
Structure: Direct-Form II, Second-Order Sections
Order: 144
Sections 72
Stable: Yes
Source: Designed

Figure 20: Phase delay for IIR filter


Current Filter Information

Structure: Direct-Form II, Second-Order Sections
Order: 144
Sections 72
Stable: Yes
Source: Designed

Figure 21: Impulse response for IIR filter


Current Filter Information

Structure: Direct-Form II, Second-Order Sections
Order: 144
Sections 72
Stable: Yes
Source: Designed

Figure 22: Pole-zero plot for IIR filter

Next, the third environment is analyzed.

FIR and IIR Filter Design for Third Environment

Lastly, the signal “df3_n2H.wav” is analyzed. Similar to the “df3_n1H.wav” signal, this one had frequency components within the spectrum of the speaker’s utterances. FL

Studio is used to gather intuition about the placement of a filter, and a bandpass architecture is chosen to better account for the low and high frequency noise present in the signal. Figure 23 demonstrates the frequency response of the signal in FL Studio.

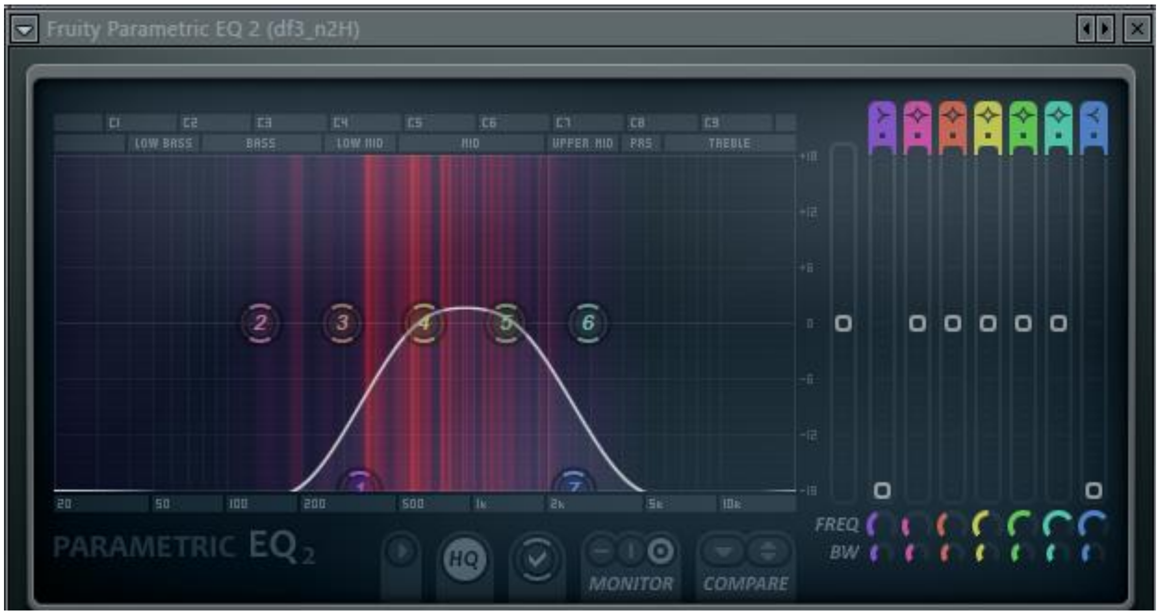
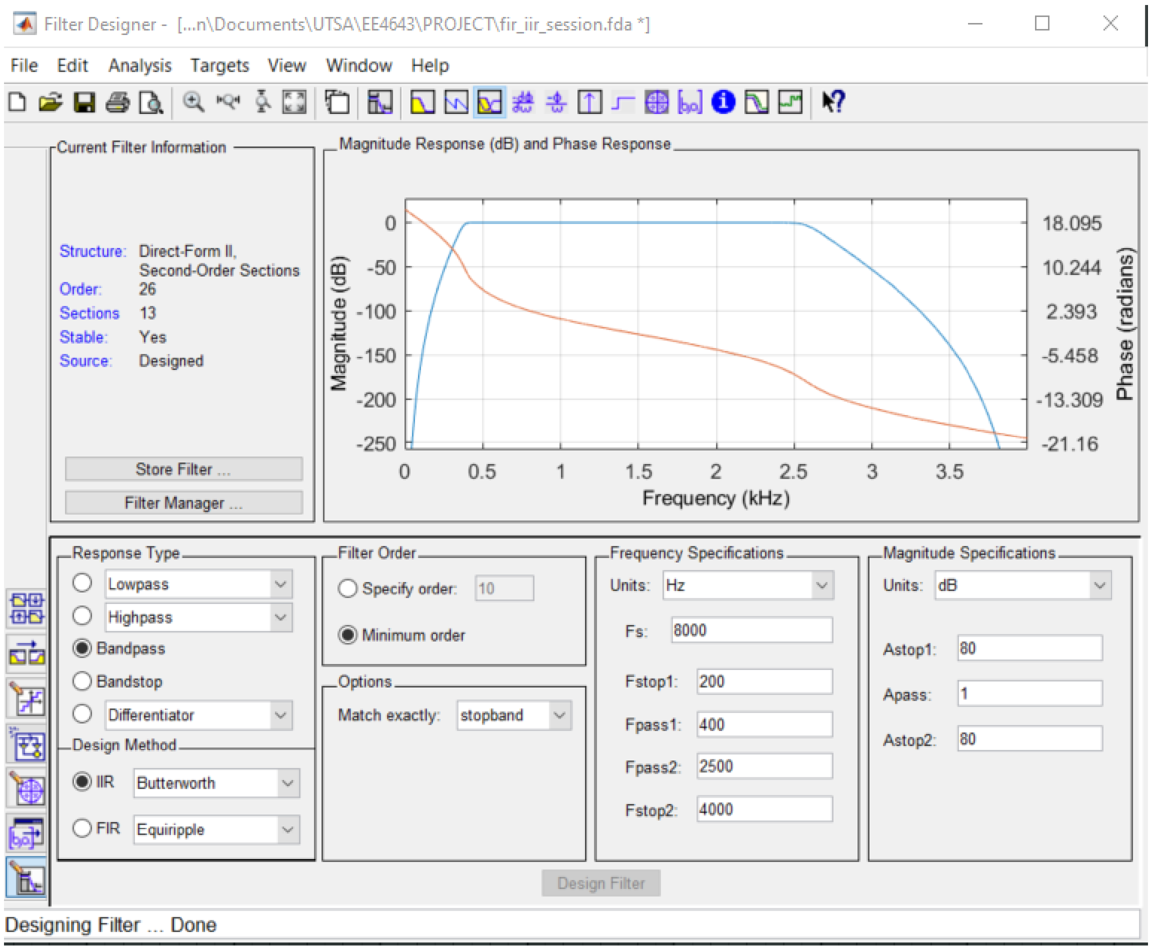


Figure 23: Looking for intuition to filter df3_n2H.wav

Because the frequency components of the noise are embedded within slight proximities to the speaker's voice, a bandpass filter with a pass band in the 400 to 2500Hz range is chosen. An **IIR bandpass, Butterworth, 26th order filter** is created using the *filterDesigner*, and its specs are outlined in Figures 24-28.


Filter Designer - [...n\Documents\UTSA\EE4643\PROJECT\fir_iir_session.fda *]
File Edit Analysis Targets View Window Help
Magnitude Response (dB) and Phase Response
Magnitude (dB)
Phase (radians)
Frequency (kHz)
Store Filter ...
Filter Manager ...
Design Filter

Current Filter Information

Structure: Direct-Form II, Second-Order Sections
Order: 26
Sections 13
Stable: Yes
Source: Designed

Response Type: Lowpass, Highpass, [✓ Bandpass], Bandstop, Differentiator

Design Method: [✓ IIR] Butterworth, FIR Equiripple

Filter Order: Specify order: 10, [✓ Minimum order]

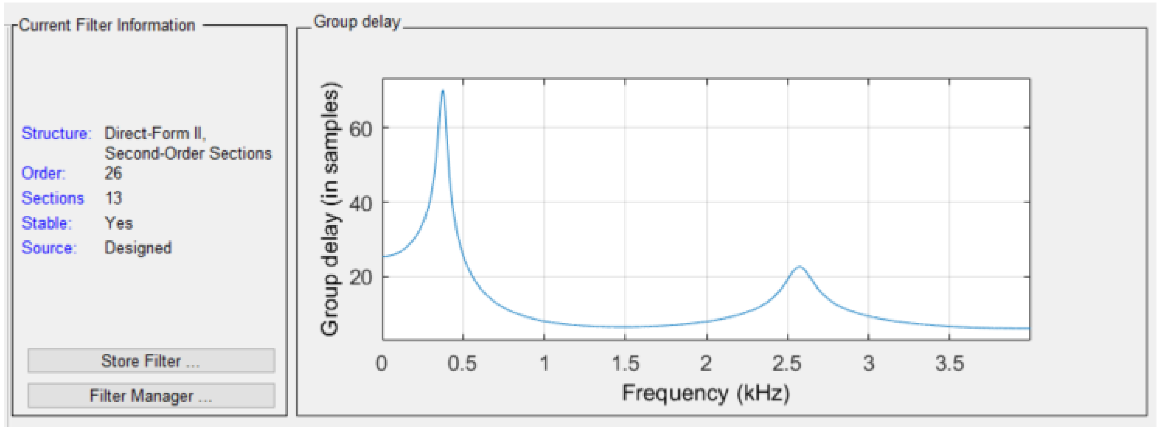
Options: Match exactly: stopband

Frequency Specifications
Units: Hz
Fs: 8000
Fstop1: 200
Fpass1: 400
Fpass2: 2500
Fstop2: 4000

Magnitude Specifications
Units: dB
Astop1: 80
Apass: 1
Astop2: 80

Designing Filter ... Done

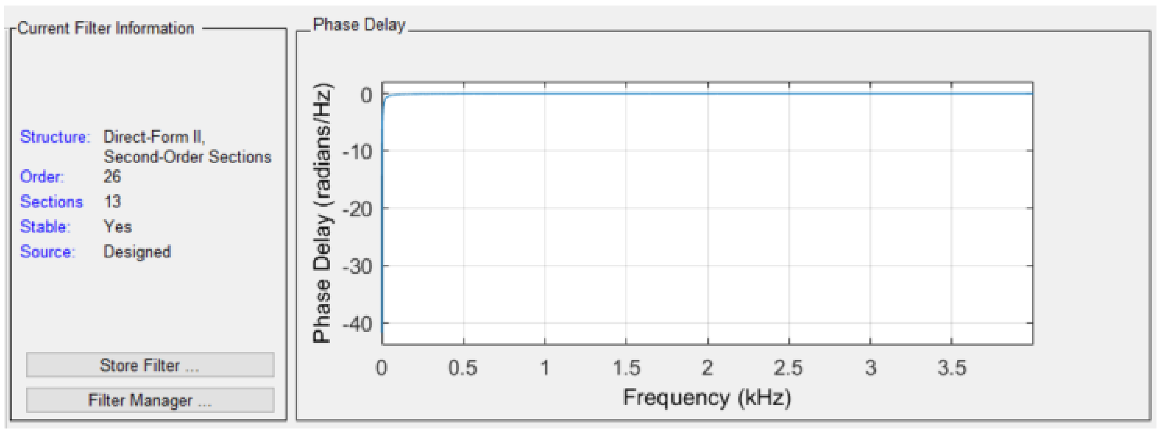
Figure 24: Magnitude and phase response of IIR filter



Current Filter Information

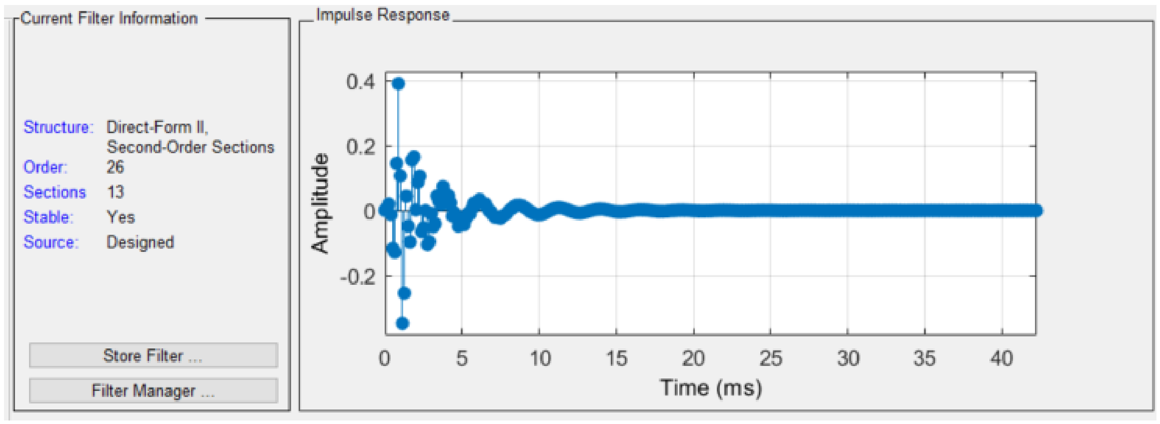
Structure: Direct-Form II, Second-Order Sections
Order: 26
Sections 13
Stable: Yes
Source: Designed

Figure 25: Group delay of IIR filter



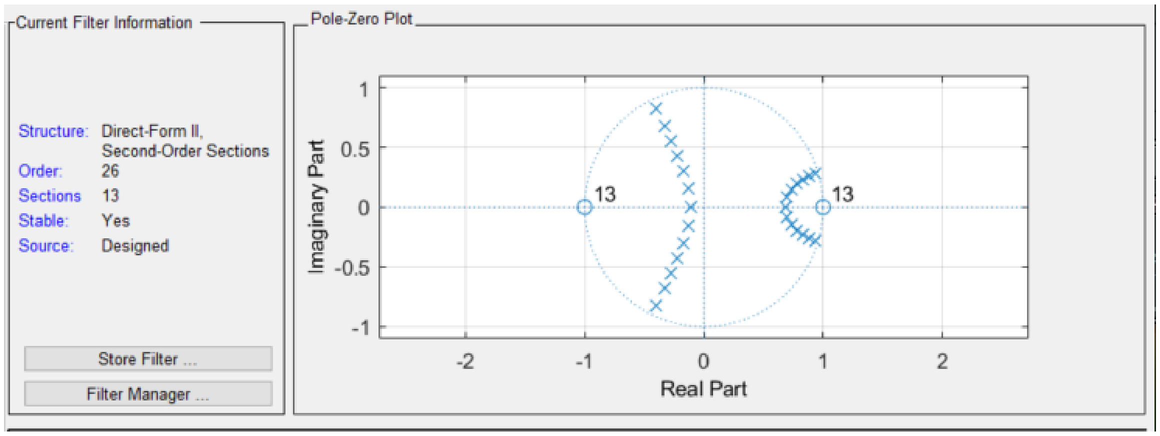
Structure: Direct-Form II, Second-Order Sections
Order: 26
Sections 13
Stable: Yes
Source: Designed

Figure 26: Phase delay of IIR filter



Structure: Direct-Form II, Second-Order Sections
Order: 26
Sections 13
Stable: Yes
Source: Designed

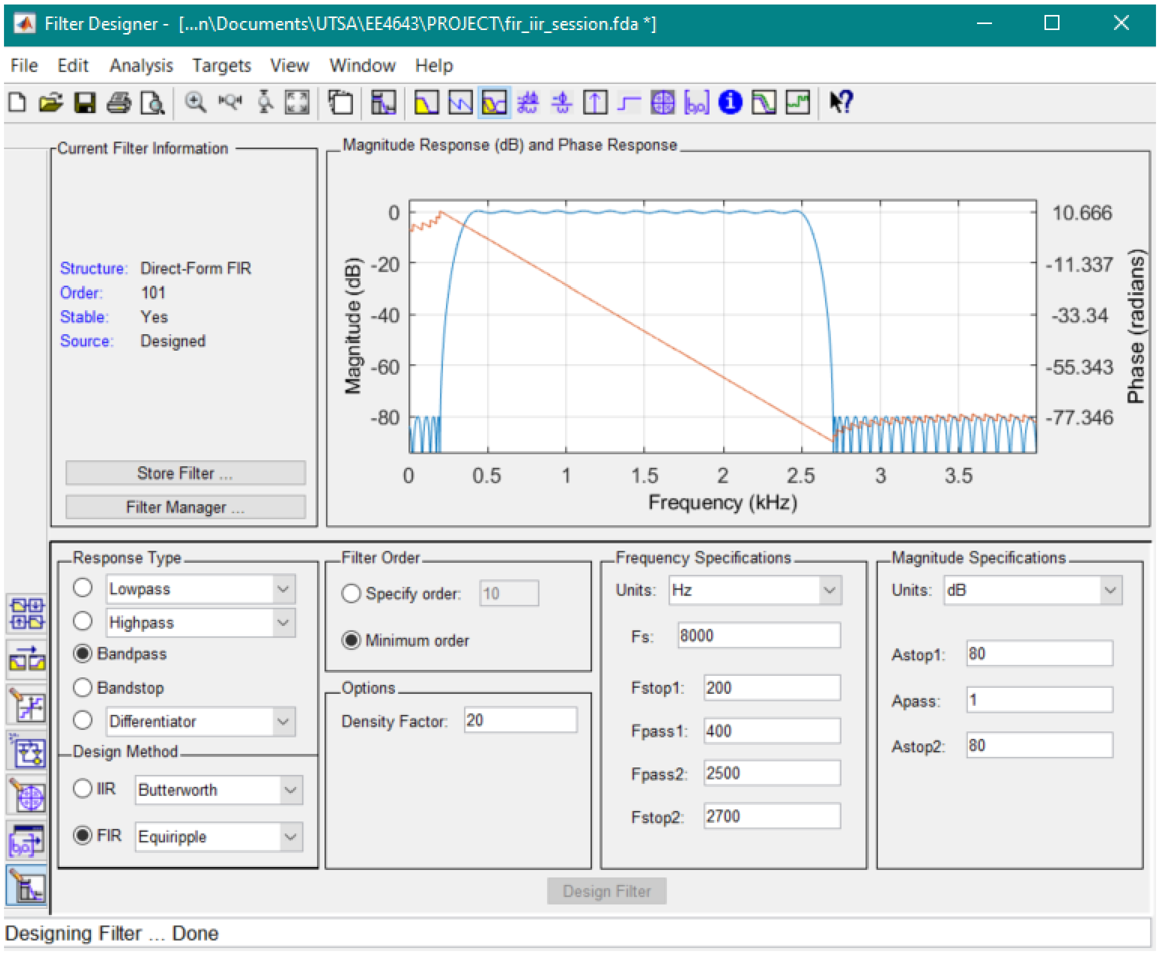
Figure 27: Impulse response of IIR filter



Structure: Direct-Form II, Second-Order Sections
Order: 26
Sections 13
Stable: Yes
Source: Designed

Figure 28: Pole-zero plot of IIR filter

Next, the FIR filter is designed with the same specifications as the IIR filter, and its specs are in Figures 29-33. A **bandpass, equiripple 101st order architecture** is designed.


Filter Designer - [...n\Documents\UTSA\EE4643\PROJECT\fir_iir_session.fda *]

Structure: Direct-Form FIR
Order: 101
Stable: Yes
Source: Designed

Response Type: Lowpass; Highpass; [✓ Bandpass]; Bandstop; Differentiator

Design Method: IIR Butterworth; [✓ FIR Equiripple]

Filter Order: Specify order: 10; [✓ Minimum order]

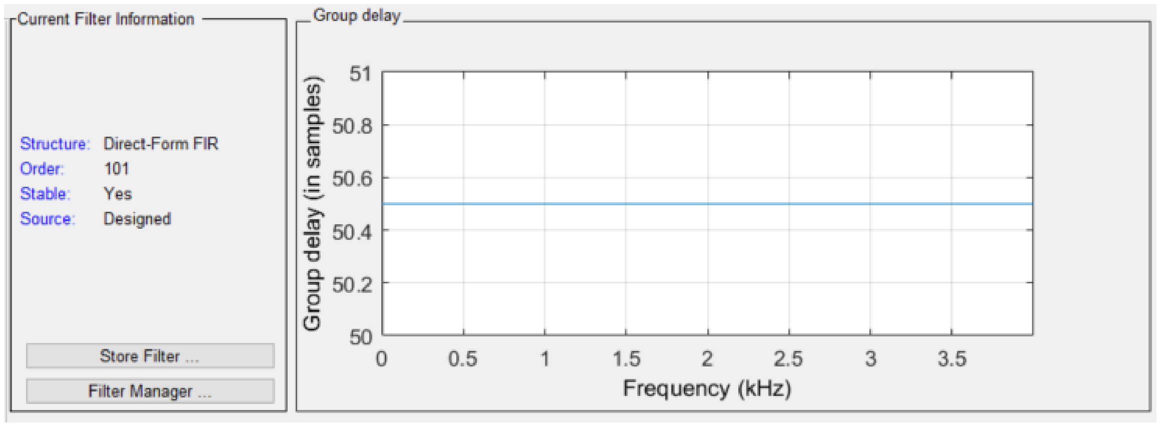
Options: Density Factor: 20

Frequency Specifications: Units: Hz; Fs: 8000; Fstop1: 200; Fpass1: 400; Fpass2: 2500; Fstop2: 2700

Magnitude Specifications: Units: dB; Astop1: 80; Apass: 1; Astop2: 80

Designing Filter ... Done

Figure 29: Magnitude and phase response of FIR filter



Structure: Direct-Form FIR
Order: 101
Stable: Yes
Source: Designed

Figure 30: Group delay of FIR filter

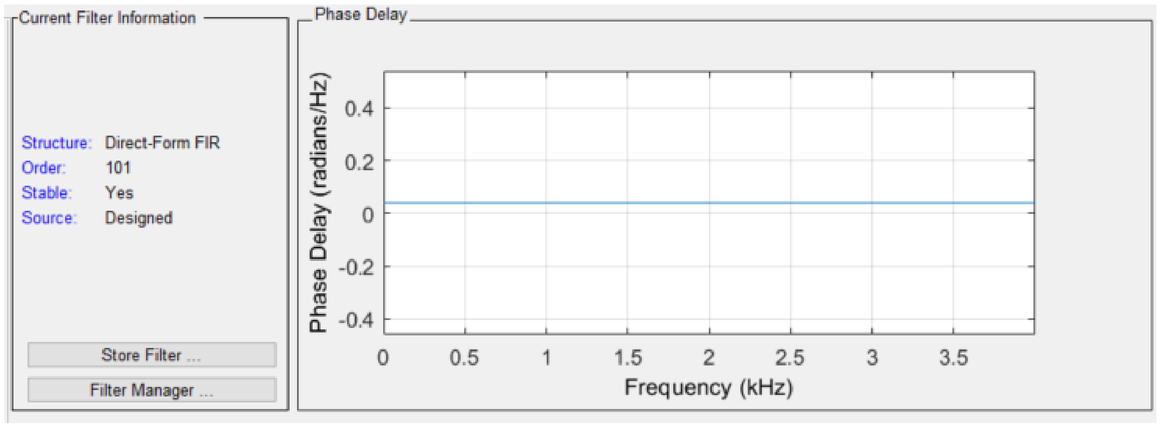


Figure 31: Phase delay of FIR filter

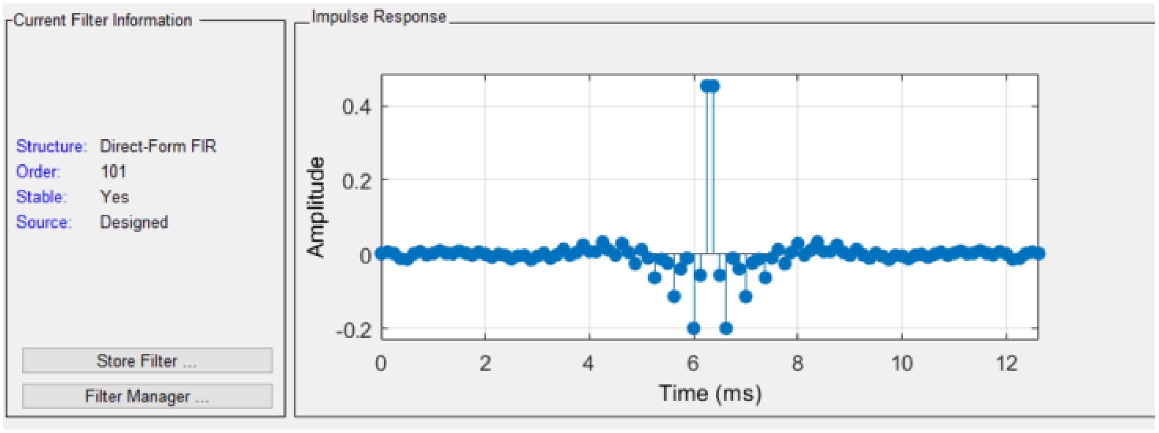


Figure 32: Impulse response of FIR filter

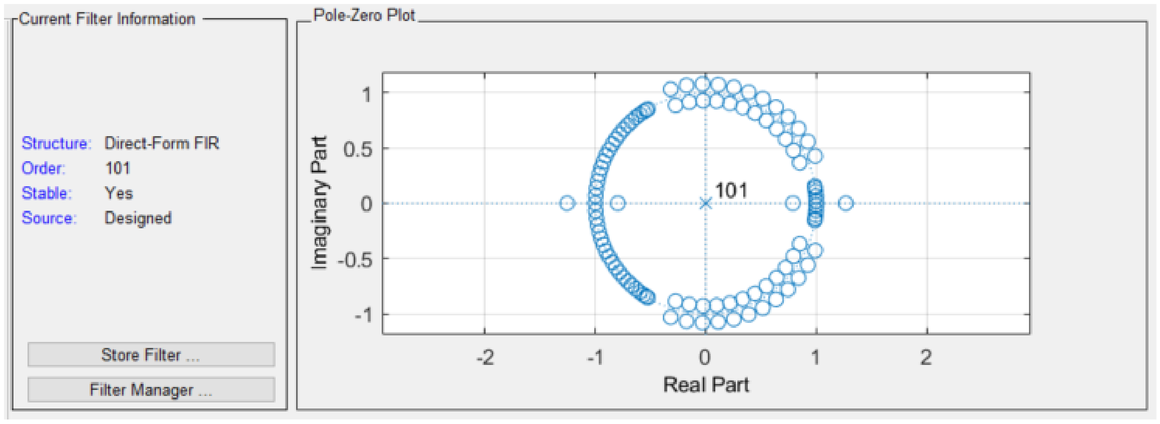


Figure 33: Pole-zero plot of FIR filter

Evident of FIR filters, all designs for the three distinct environments have linear phase delay and group delay. Their graphs are included in this section as a learning exercise and reinforcement to the author. The filtering process in MATLAB is now outlined in the following section.

Observation and Results

The FIR and IIR filters designed for each environment are exported using the *filterDesigner*'s function generator. These are written by the tool as MATLAB code and interfaced into the main code as expandable functions. Three custom functions are written to filter the signal, plot the results in the frequency spectrum, and playing the WAV file through the audio hardware device as *filterPlotSignal, plotFreqSignal, and playSignal* respectively.

All three test cases utilize the following DSP system to process the signal as shown in Figure 34. The signal is read in from the WAV file → its FFT is found → FFT of original signal is plot → original signal is filtered with designed filter → FFT of filtered signal → plot the FFT → play the filtered signal to the user.

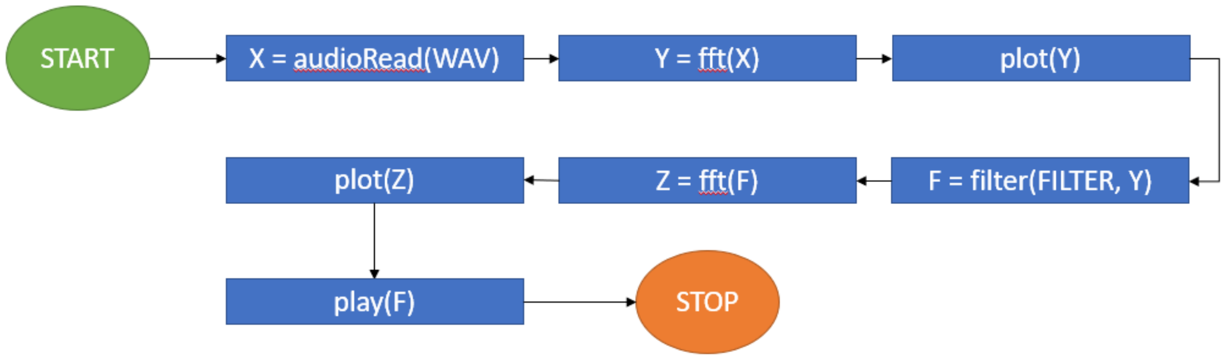


Figure 34: DSP System modeled inside MATLAB program

The filtering of the first noise environment is now observed.

Filtering the First Environment

The signal df3_n0H.wav is filtered first with the FIR filter designed in the *Procedure* section. Figure 35 displays the frequency spectrum of the original signal compared to the filtered signal.

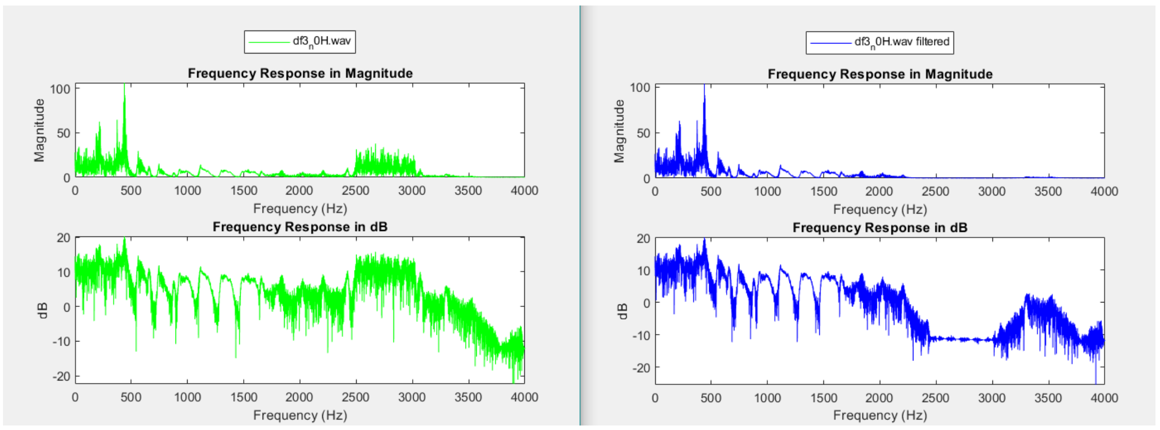


Figure 35: df3_n0H.wav filtered with a FIR, bandstop equiripple 38th order filter

Now, the signal is filtered with the IIR filter previously designed and shown in Figure 36.

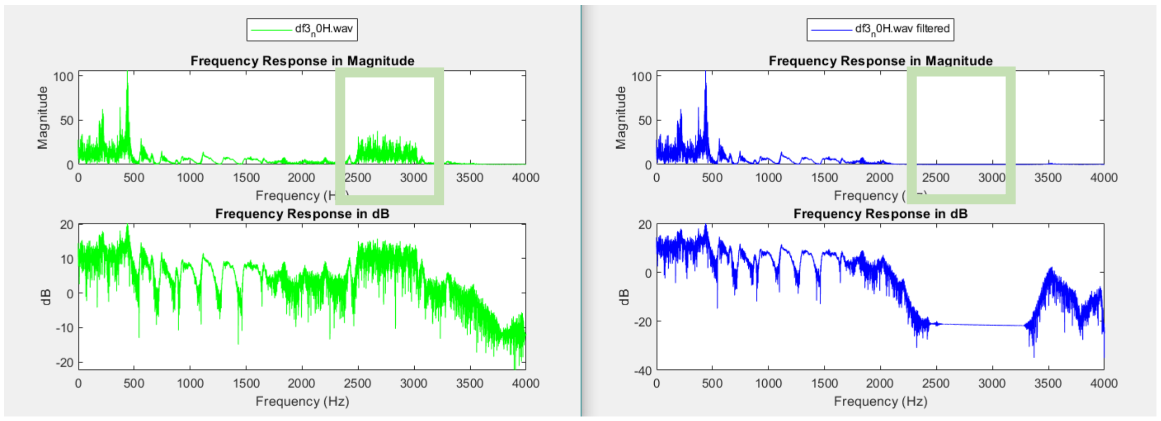


Figure 36: df3_n0H.wav filtered with an IIR, bandstop, Butterworth 26th order filter

From Figures 35, 36, it is evident the IIR filter suppressed the noise at a greater degree than the FIR as outlined in the stop band. However, apart from this observation, listening to both filtered outputs as WAV files (attached to this project) seem highly similar. Luckily for the author, the noise is present in a frequency range that was pinpoint as outlined in Figure 36, and the resulting output is almost noise free.

<u>Filtering the Second Environment</u>

The next signal to analyze is df3_n1H.wav. Compared to df3_n0H.wav, this signal has noise embedded within the speaker's frequency components, which lie approximately in the 200 and 1kHz bands. For the author's reference, the input signal consists of a series of numbers uttered, more specifically—6, 2, 2, 9. The former three numbers have frequency components in the 200 Hz band, and the latter number has frequency component in the 1kHz band approximately. The difficult part about filtering this second environment lies in the convolved noise components in these exact bands. Figure 37 shows an attempt at filtering the noise out with the custom FIR filter designed for this signal.

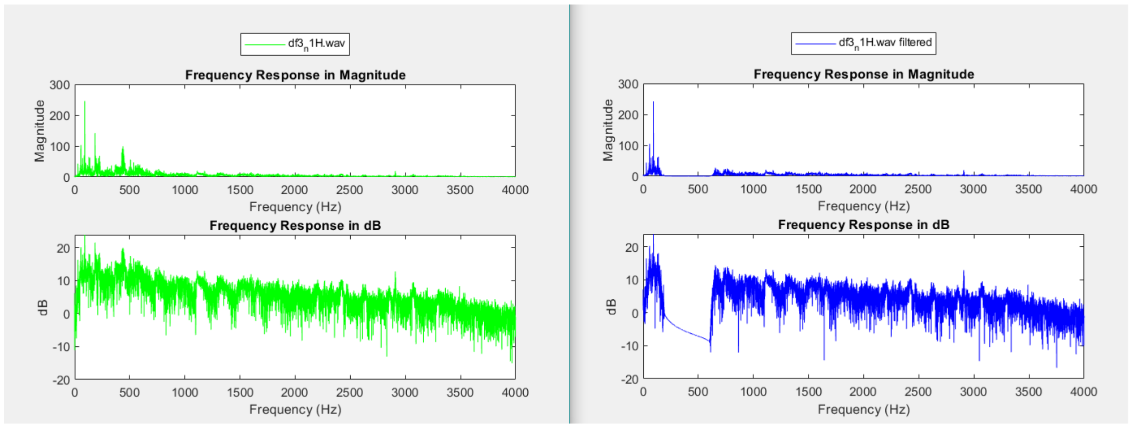


Figure 37: df3_n1H filtered with a FIR, bandstop equiripple 440th order filter

The order of this filter is exaggerated, and for practical purposes, such as implementation in an FPGA, (see author's FIR implementation at https://github.com/16oh4/FPGA-VLSI), this would be highly inefficient. Transposing the Direct Form 1 FIR filter would amount to 441 multiplications required, followed by 441 additions for the output y[n] to observe the first signal point. The IIR filter used to process the signal next is a bit more accessible, but still excessive as shown in Figure 38.

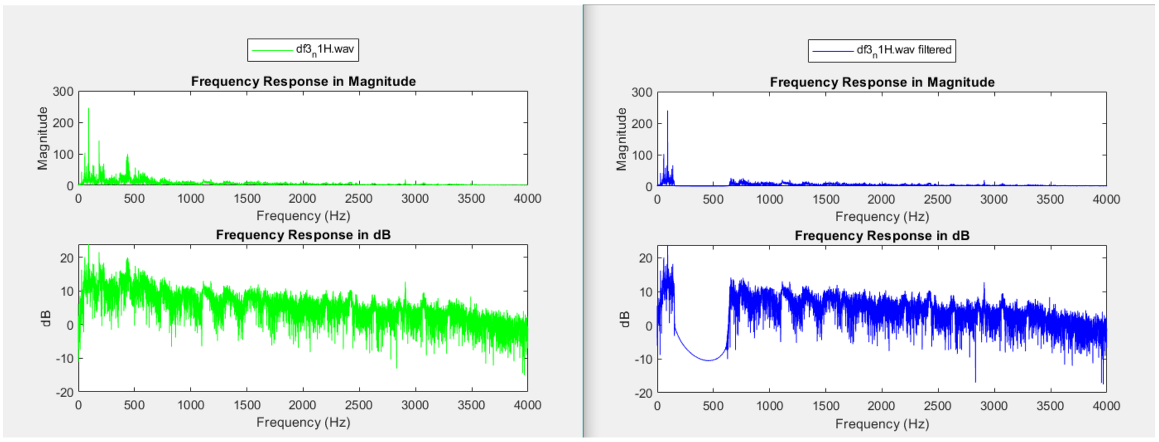


Figure 38: df3_n1H.wav filtered with an IIR, bandstop Butterworth 144th order filter

The spectral results are similar, as expected, and with a considerably lower order. Listening to the actual output in a WAV file compared to the FIR filter once again does not provide much of a difference given both filters are designed with the same specs. Next, the third environment is analyzed.

Filtering the Third Environment

The signal "df3_n2H.wav" contains background chatter compared to the other two environments. This signal is by far the most difficult to filter because of the similar spectral components of human voices. As a result, a bandpass filter is utilized to attempt to filter out noise outside of the mid-band range. The FIR filter output is presented in Figure 39.

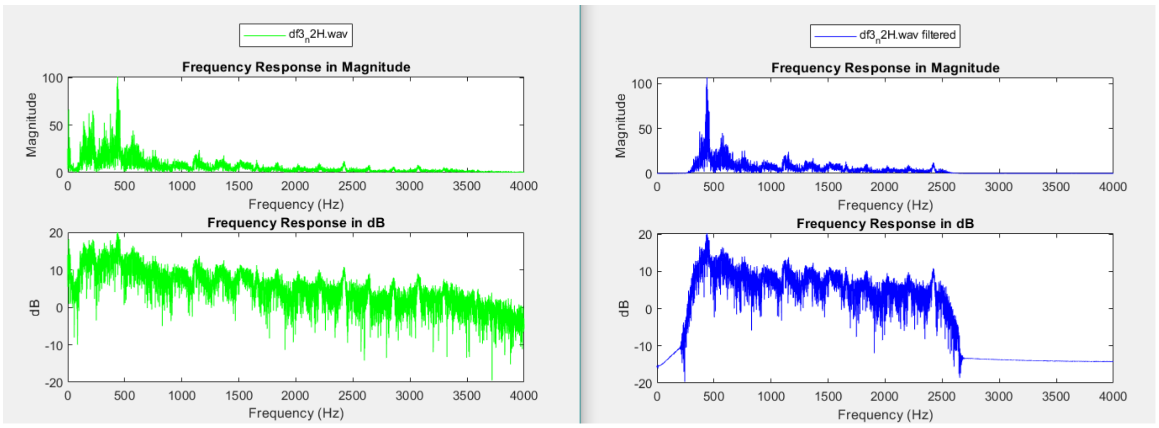


Figure 39: df3_n2H.wav filtered with a FIR, bandpass equiripple 101st order filter

The filter effectively removes noise outside the mid-band range. Now, Figure 40 shows the filtered signal using the IIR filter designed for this environment.

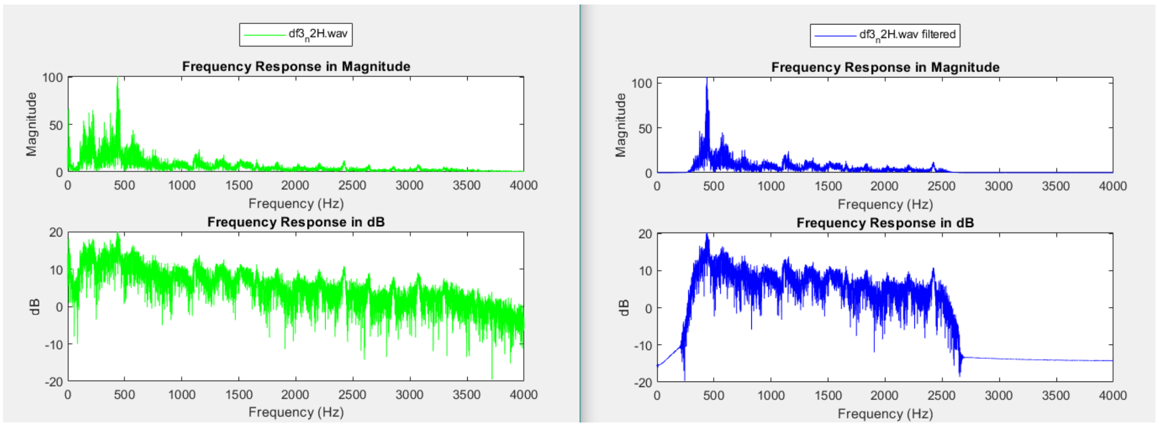


Figure 40: df3_n2H.wav filtered with an IIR, bandpass, Butterworth 26th order filter

Compared the FIR, this IIR filter is clearly the best to implement because of its substantially lower order, and most importantly due to its highly similar sound when listened to. Although the IIR filter does not have linear phase compared to the FIR, it would be more cost efficient.

Conclusion

This project allowed me to learn filter design using MATLAB's tools, and I learned the necessary intuition to pick a filter necessary to remove noise in a signal. More often than not, the FIR filter's distinct properties were insignificant for the task of reducing noise in a speaker's voice, and the IIR filter implementation seemed to be the most cost efficient. This comparison allowed me to understand the viable applications of both FIR and IIR filters.

As a musician, I've found it interesting to understand the robust theory necessary to implement both IIR and FIR filters, when DAWs like FL Studio have sliders for me to easily control multi-order filters at will. This allowed me to appreciate the DSP work being done to facilitate our lives. Thank you Professor Zhang for your instruction in EE 4643.

Notes

- Please refer to "fir_iir_session.sda" in the *filterDesigner_files* folder if you would like to see the filters I designed
- Use the command *filterDesigner* in a MATLAB interpreter, and then open the .SDA file inside the *filterDesigner* to view the designs
- MATLAB code file is appended here, but also in folder
- Filtered signals using both FIR and IIR for each signal provided are in the folder "Filtered_signals"

MATLAB Code

```
close all

%{
Bruno E. Gracia Villalobos
EE 4643
Final Project
Professor Zhang
%}

% -----------------------Setup filters and file paths----------------------%

%Store filepath of each signal
signal0Path = "df3_n0H.wav";
signal1Path = "df3_n1H.wav";
signal2Path = "df3_n2H.wav";

%Build custom filters for df3_n0H
FIR_BS0 = FIR_bandstop0();
IIR_BS0 = IIR_bandstop0();

%Build custom filters for df3_n1H
FIR_BS1 = FIR_bandstop1();
IIR_BS1 = IIR_bandstop1();

%Build custom filters for df3_n2H
FIR_BP2 = FIR_bandpass2();
IIR_BP2 = IIR_bandpass2();

% -----------------------------Filtering section ------------------------%

%{
    ----------------------Testing Instructions-----------------------------

    To change the signal and filter, change the arguments of
    filterPlotSignal:

            filterPlotSignal(signalPath, filterToUse)

    For testing df3_n0H:
    signalPath --> signal0Path
    filterToUse --> FIR_BS0 or IIR_BS0

    For testing df3_n1H:
    signalPath --> signal1Path
    filterToUse --> FIR_BS1 or IIR_BS1

    For testing df3_n2H:
    signalPath --> signal2Path
    filterToUse --> FIR_BP2 or IIR_BP2
%}

%This three line code block filters and plays the specified signal in path
[signal, signalFiltered, signalFS] = filterPlotSignal(signal2Path, IIR_BP2);
playSignal(signal, signalFS);
playSignal(signalFiltered, signalFS);

%Write results to a WAV file!
%audiowrite(strcat(signal2Path, 'IIR.wav'), signalFiltered, signalFS);

%-----Filter generator functions exported from MATLAB filterDesigner-%
function Hd = FIR_bandstop0
%FIR_BS_EQUIRIPPLE_38 Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.5 and DSP System Toolbox 9.7.
% Generated on: 11-Dec-2019 12:22:48
% Equiripple Bandstop filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 8000;  % Sampling Frequency

Fpass1 = 2000;            % First Passband Frequency
Fstop1 = 2500;            % First Stopband Frequency
Fstop2 = 3000;            % Second Stopband Frequency
Fpass2 = 3500;            % Second Passband Frequency
Dpass1 = 0.028774368332;  % First Passband Ripple
Dstop  = 0.001;           % Stopband Attenuation
Dpass2 = 0.057501127785;  % Second Passband Ripple
dens   = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass1 Fstop1 Fstop2 Fpass2]/(Fs/2), [1 0 ...
                          1], [Dpass1 Dstop Dpass2]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
end

function Hd = IIR_bandstop0
%IIR_26_BS_BUTTERWORTH Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.5 and DSP System Toolbox 9.7.
% Generated on: 11-Dec-2019 11:08:08

% Butterworth Bandstop filter designed using FDESIGN.BANDSTOP.

% All frequency values are in Hz.
Fs = 8000;  % Sampling Frequency

Fpass1 = 2000;        % First Passband Frequency
Fstop1 = 2500;        % First Stopband Frequency
Fstop2 = 3000;        % Second Stopband Frequency
Fpass2 = 3500;        % Second Passband Frequency
Apass1 = 1;           % First Passband Ripple (dB)
Astop  = 80;          % Stopband Attenuation (dB)
Apass2 = 1;           % Second Passband Ripple (dB)
match  = 'passband';  % Band to match exactly

% Construct an FDESIGN object and call its BUTTER method.
h  = fdesign.bandstop(Fpass1, Fstop1, Fstop2, Fpass2, Apass1, Astop, ...
                      Apass2, Fs);
Hd = design(h, 'butter', 'MatchExactly', match);

% [EOF]
end

function Hd = FIR_bandstop1
%FIR_BS_BUTTERWORTH_440_N1H Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.5 and DSP System Toolbox 9.7.
% Generated on: 11-Dec-2019 14:17:24

% Equiripple Bandstop filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 8000;  % Sampling Frequency

Fpass1 = 150;             % First Passband Frequency
Fstop1 = 200;             % First Stopband Frequency
```

```
Fstop2 = 600;             % Second Stopband Frequency
Fpass2 = 650;             % Second Passband Frequency
Dpass1 = 0.057501127785;  % First Passband Ripple
Dstop  = 0.0001;          % Stopband Attenuation
Dpass2 = 0.057501127785;  % Second Passband Ripple
dens   = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass1 Fstop1 Fstop2 Fpass2]/(Fs/2), [1 0 ...
                          1], [Dpass1 Dstop Dpass2]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
end

function Hd = IIR_bandstop1
%IIR_BS_BUTTERWORTH_144_N1H Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.5 and DSP System Toolbox 9.7.
% Generated on: 11-Dec-2019 14:28:59

% Butterworth Bandstop filter designed using FDESIGN.BANDSTOP.

% All frequency values are in Hz.
Fs = 8000;  % Sampling Frequency

Fpass1 = 150;         % First Passband Frequency
Fstop1 = 200;         % First Stopband Frequency
Fstop2 = 600;         % Second Stopband Frequency
Fpass2 = 650;         % Second Passband Frequency
Apass1 = 1;           % First Passband Ripple (dB)
Astop  = 80;          % Stopband Attenuation (dB)
Apass2 = 1;           % Second Passband Ripple (dB)
match  = 'passband';  % Band to match exactly

% Construct an FDESIGN object and call its BUTTER method.
h  = fdesign.bandstop(Fpass1, Fstop1, Fstop2, Fpass2, Apass1, Astop, ...
                      Apass2, Fs);
Hd = design(h, 'butter', 'MatchExactly', match);

% [EOF]
end

function Hd = FIR_bandpass2
%FIR_BP_EQUIRIPPLE_101_N2H Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.5 and DSP System Toolbox 9.7.
% Generated on: 11-Dec-2019 14:53:41

% Equiripple Bandpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 8000;  % Sampling Frequency

Fstop1 = 200;             % First Stopband Frequency
Fpass1 = 400;             % First Passband Frequency
Fpass2 = 2500;            % Second Passband Frequency
Fstop2 = 2700;            % Second Stopband Frequency
Dstop1 = 0.0001;          % First Stopband Attenuation
Dpass  = 0.057501127785;  % Passband Ripple
Dstop2 = 0.0001;          % Second Stopband Attenuation
dens   = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
```

```
[N, Fo, Ao, W] = firpmord([Fstop1 Fpass1 Fpass2 Fstop2]/(Fs/2), [0 1 ...
                          0], [Dstop1 Dpass Dstop2]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
end

function Hd = IIR_bandpass2
%IIR_BP_BUTTERWORTH_26_N2H Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.5 and DSP System Toolbox 9.7.
% Generated on: 11-Dec-2019 14:55:28

% Butterworth Bandpass filter designed using FDESIGN.BANDPASS.

% All frequency values are in Hz.
Fs = 8000;  % Sampling Frequency

Fstop1 = 200;         % First Stopband Frequency
Fpass1 = 400;         % First Passband Frequency
Fpass2 = 2500;        % Second Passband Frequency
Fstop2 = 4000;        % Second Stopband Frequency
Astop1 = 80;          % First Stopband Attenuation (dB)
Apass  = 1;           % Passband Ripple (dB)
Astop2 = 80;          % Second Stopband Attenuation (dB)
match  = 'stopband';  % Band to match exactly

% Construct an FDESIGN object and call its BUTTER method.
h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                      Astop2, Fs);
Hd = design(h, 'butter', 'MatchExactly', match);

% [EOF]
end

%------------------------Custom functions------------------------------%
function [signal, signalFiltered, fs] = filterPlotSignal(filePath, filterObject)
%Let's read in the samples of the WAV and store sample rates
[signal, fs] = audioread(filePath);

%Find the FFT of the signals
signalFFT = fft(signal);

%Plot the signals' frequency spectrums
plotFreqSignal(signalFFT, fs, 1, filePath, 'g');

%Filter the signal
signalFiltered = filter(filterObject, signal);

%Plot filtered signal frequency spectrum
signalFilteredFFT = fft(signalFiltered);
plotFreqSignal(signalFilteredFFT, fs, 1, strcat(filePath, ' filtered'), 'b');
end

function playSignal(signal, fs)
    %Play the original signal
    signalPlayer = audioplayer(signal, fs);
    playblocking(signalPlayer);
end

function plotFreqSignal(signal, fs, type, name, color)
    half = floor(length(signal)/2);

    %type ==1 means the signal is a pure FFT output
    if(type ==1)
```

```
        %remove data above nyquist limit = fs/2
        %freqAxis = linspace(0, fs/2, length(signal)/2);
        freqResolution = fs/length(signal);
        freqAxis = 0 : freqResolution : fs/2-freqResolution; %cut away nyquist limit

        %disp(half);
        figure;
        grid on;

        subplot(2,1,1);
        %plot(freqAxis, abs(signal(1:half))/ length(signal), 'b'); %plot mag and normalized
        plot(freqAxis, abs(signal(1:half)), color);
        %cut away nyquist limit @ end/2 + 1


        legend(name,'location','northoutside')

        titleGraph = strcat('Frequency Response in Magnitude','');
        title(titleGraph);

        xlabel('Frequency (Hz)');
        ylabel('Magnitude');

        subplot(2,1,2);
        plot(freqAxis, 10*log10(abs(signal(1:half))), color);

        titleGraph = strcat('Frequency Response in dB','');
        title(titleGraph);

        xlabel('Frequency (Hz)');
        ylabel('dB');

    elseif(type == 0) %means signal is from DC to NYQUIST
        figure;

        plot(linspace(0, (fs/2), length(signal)), 20*log10(signal));

        title('Mel Spectrum'), xlabel('Frequency (Hz)');
    end
end
```