**Lab 1**

**MINI PROJECT: Digital Filters and Their Applications**

**EECS3451**

**Name: Chethana Wickramasinghe**

**Student Number: 214866511**

**Professor: Peter Lian**

**13sth of April 2021**

**1. Introduction:**

This Mini Project requires to design filters using MATLAB commands, plot frequency responses of filter and last design an appropriate filter that could remove noise from a given music signal.

**2. Equipment:** MATLAB

**3. Results and discussion:**

# Part 1: FIR Filter Design

### Q1: Design a bandpass FIR filter meets the following specifications:

Stopband edge 1: 500Hz

Passband edge 1: 1500 Hz

Passband edge 2: 2000 Hz

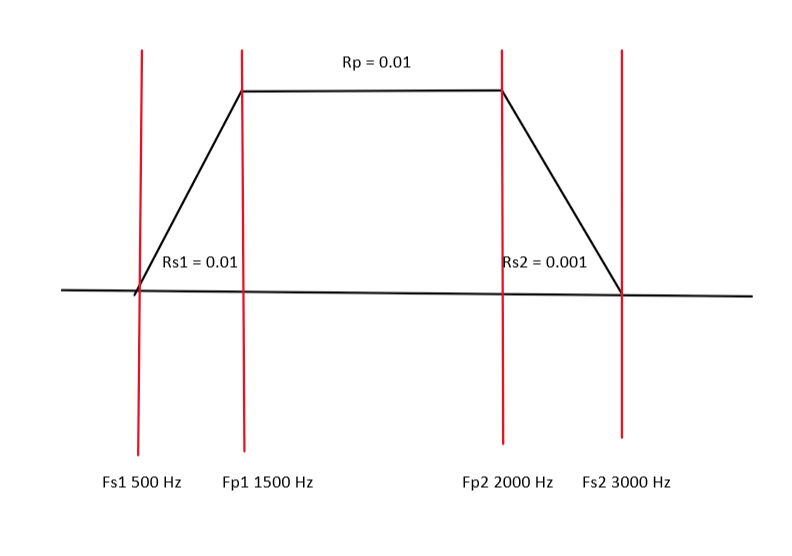
Stopband edge 2: 3000 Hz

Sampling frequency: 8000 Hz

Pass-band ripple <= 0.01

Stopband ripple from 0 to 500Hz <= 0.01

Stopband ripple from 3000 to 4000Hz <= 0.001

 **Bandpass FIR requirements**

**CODE**

F=[500 ,1500, 2000, 3000]; % band pass edge frequencies

A=[0, 1, 0]; % a stop band(0), pass band(1) and a stop band(0)

Dev = [0.01, 0.01, 0.001]; % Ripples in bands

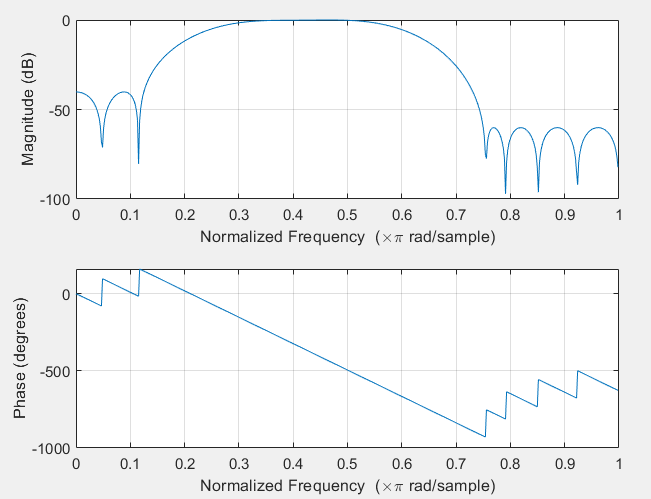
[N,Fi,Ai,W]=firpmord(F,A,Dev,8000);

% filter length / approximate order: N, normalized frequency band edges: Fi, frequency band amplitudes: Ai, and weights W that meet input specifications

B1=firpm(N,Fi,Ai,W);

% B1 gives the filter coefficients

freqz(B1,1); % displaying the magnitude and phase responses of the filter

**Magnitude and Phase responses of the filter**

**Filter length**

Low pass =

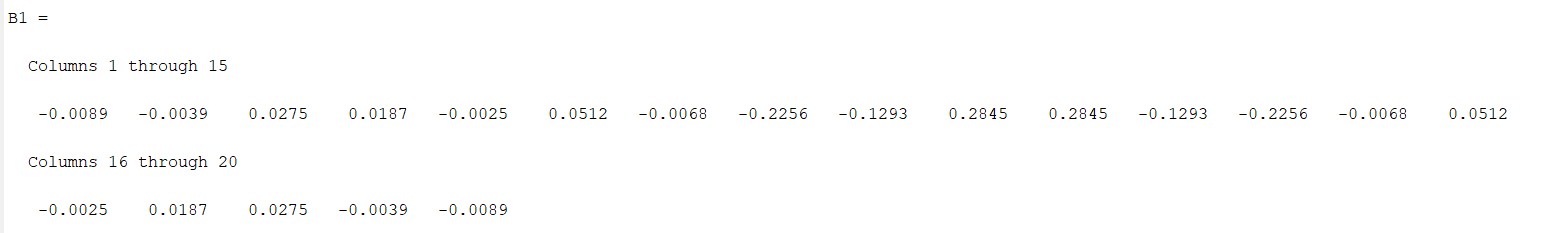
High pass =

Average of Low and High pass length =

According to the code: as well

**Filter coefficients**

Following are the coefficients of the filter generated using the ‘firpmord’ function,



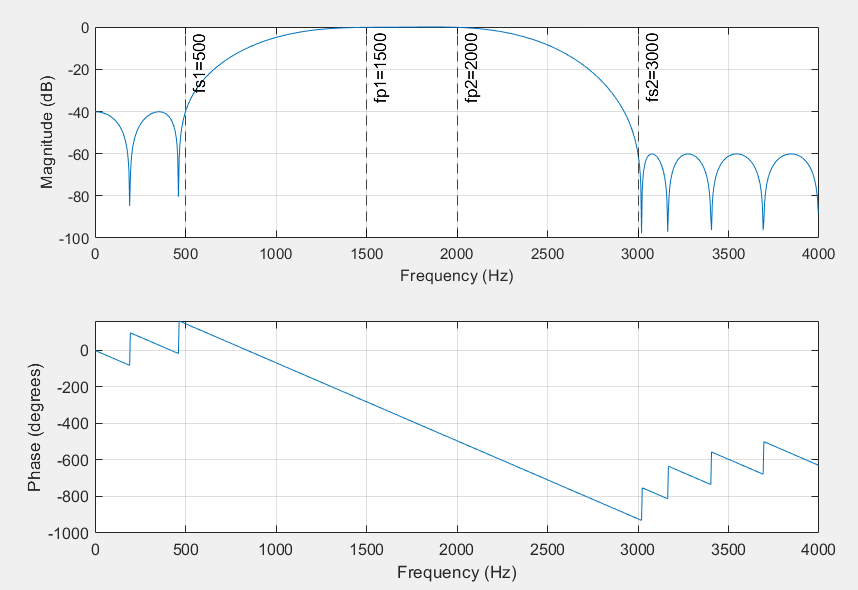
**Verify**

freqz(B1,1,1024,8000);

xline(500,'--k',sprintf('fs1=%0.0f', 500))

xline(1500,'--k',sprintf('fp1=%0.0f', 1500))

xline(2000,'--k',sprintf('fp2=%0.0f', 2000))

xline(3000,'--k',sprintf('fs2=%0.0f', 3000))

Verifying ripples

plot(abs(freqz(B1,1)))

title('Focus: Magnitude Response');

yline(0+0.01,'--k',sprintf('Rs1=%0.2f', 0+0.01))

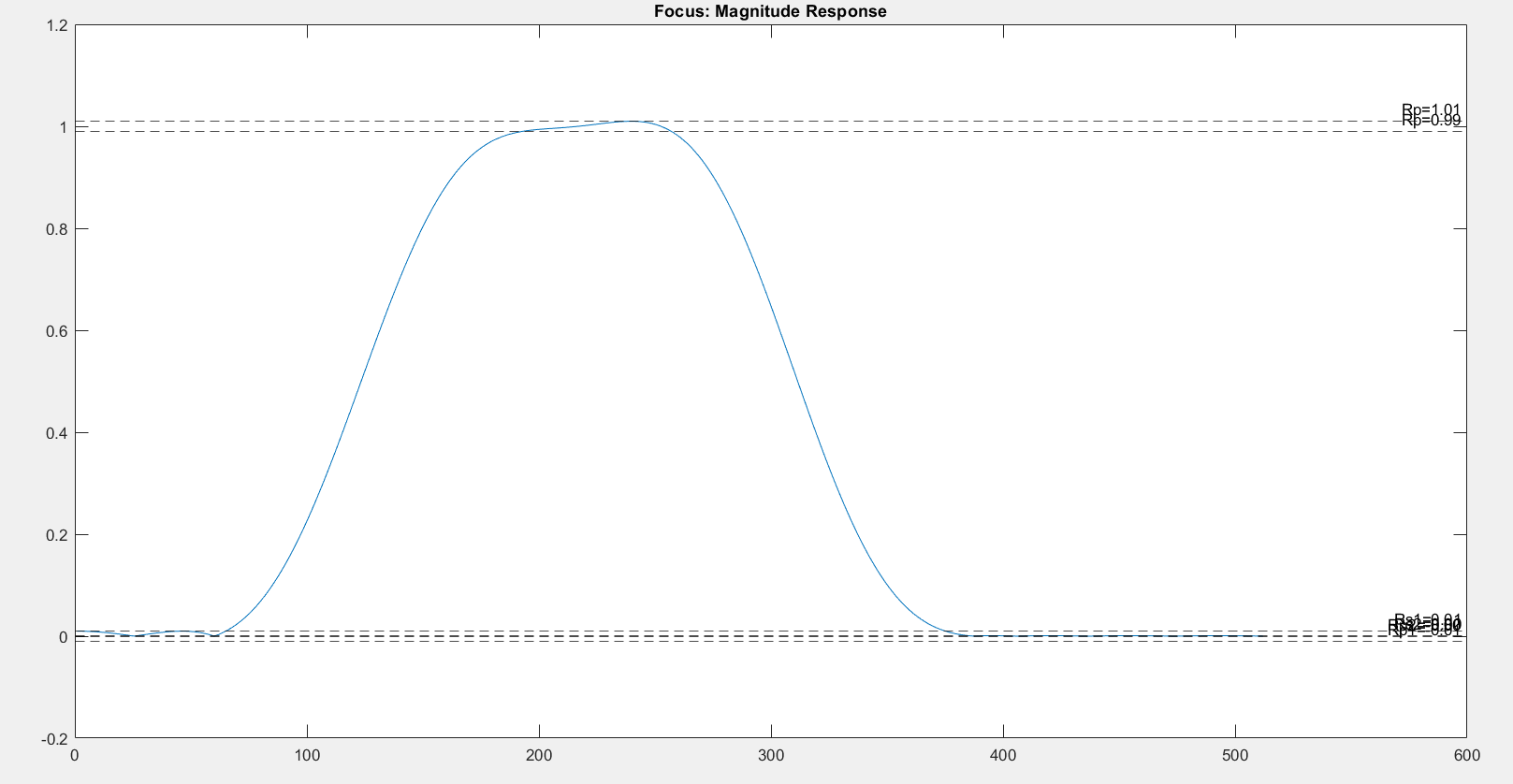
yline(0-0.01,'--k',sprintf('Rp1=%0.2f', 0-0.01))

yline(0+0.001,'--k',sprintf('Rs2=%0.2f', 0+0.001))

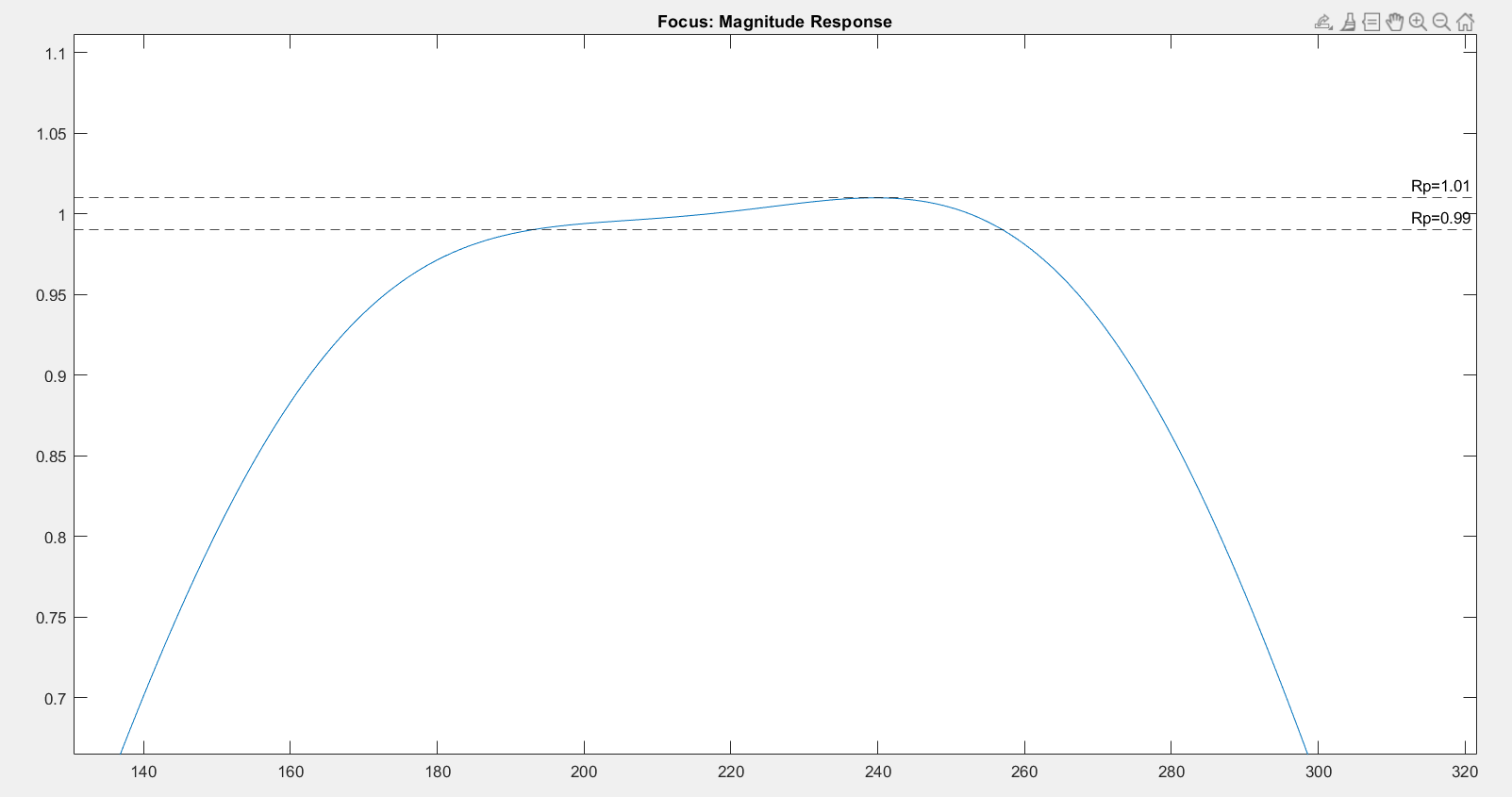
yline(0-0.001,'--k',sprintf('Rp2=%0.2f', 0-0.001))

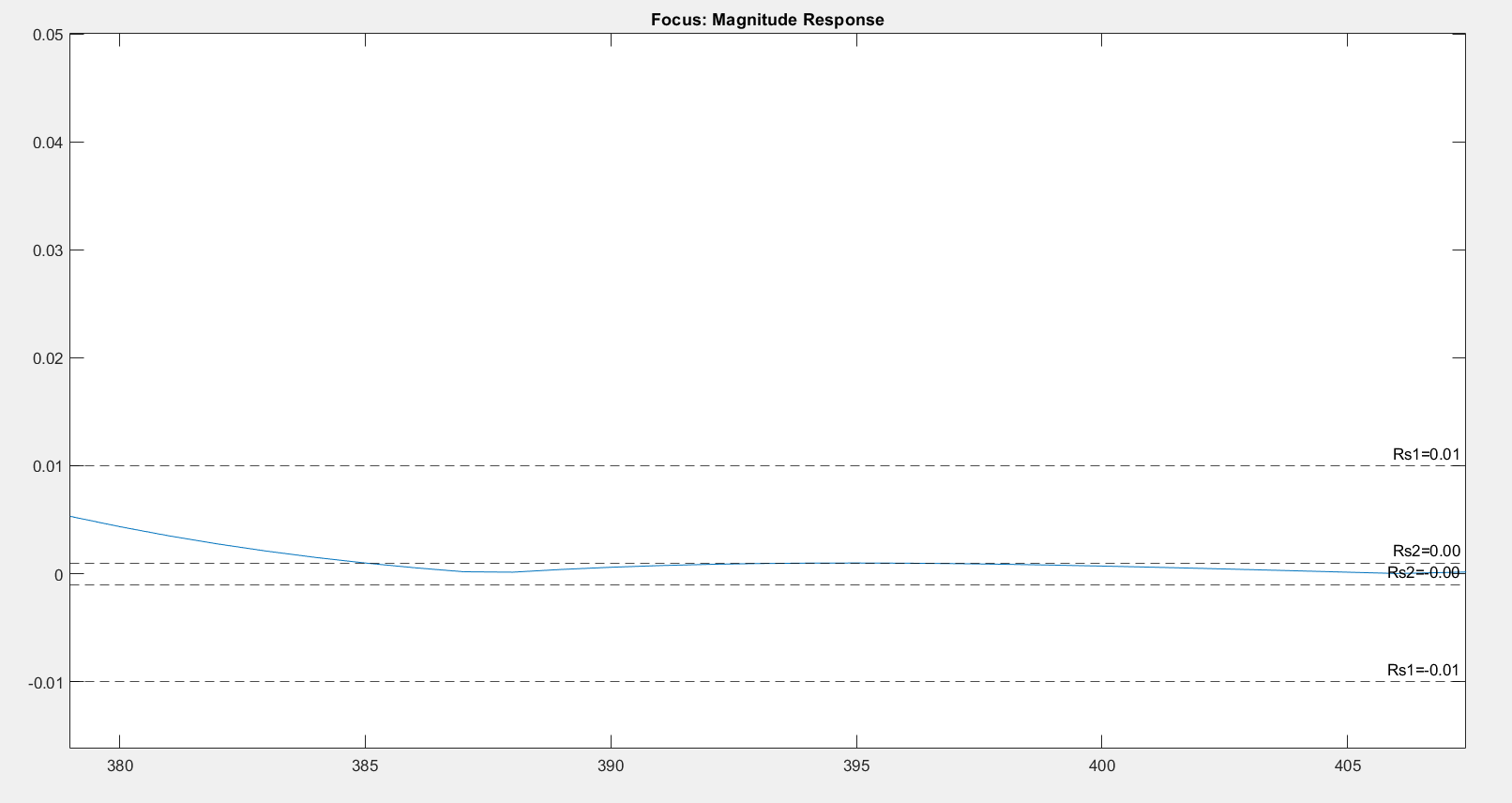
yline(1+0.01,'--k',sprintf('Rp=%0.2f', 1+0.01))

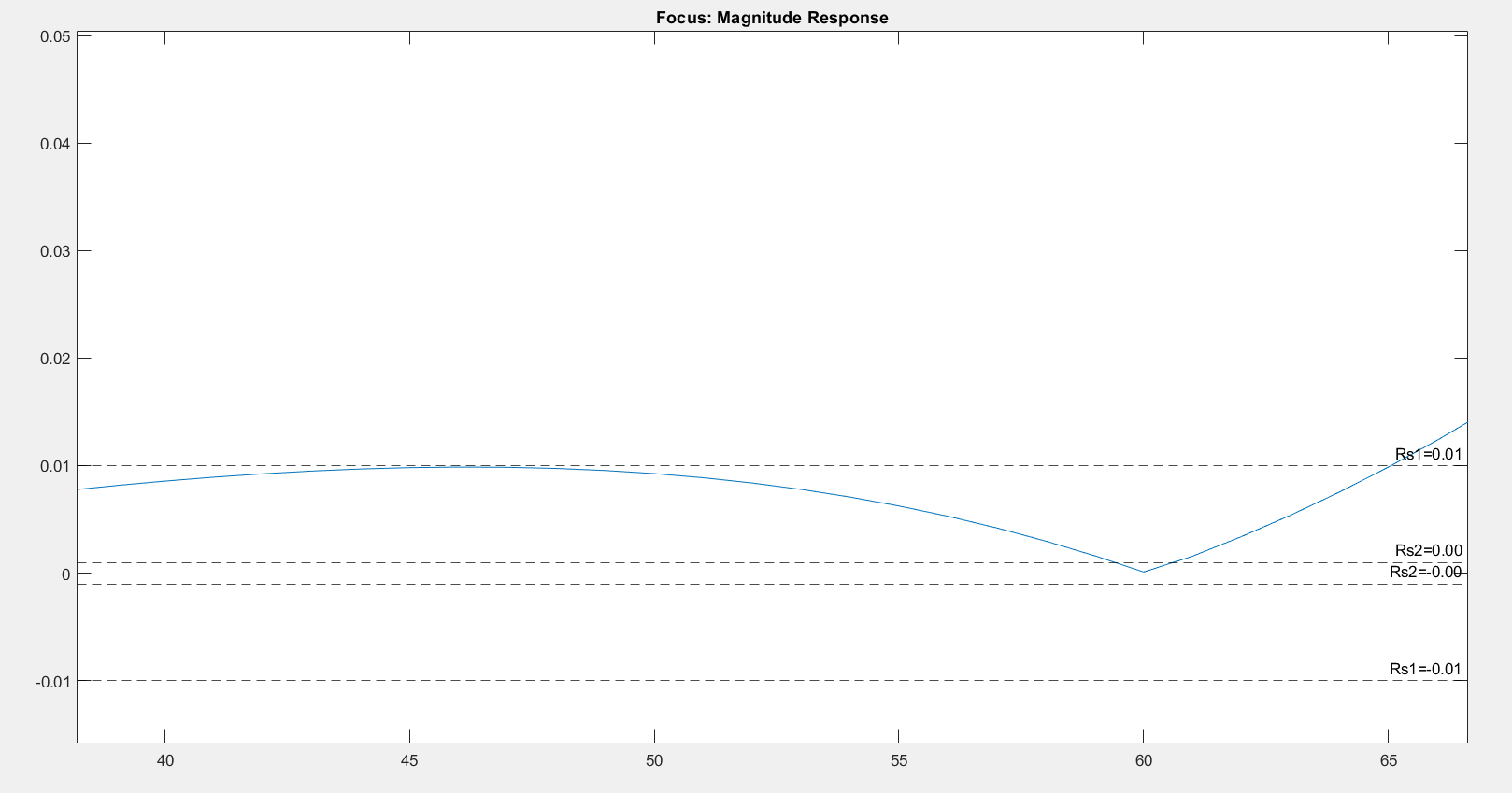
yline(1-0.01,'--k',sprintf('Rp=%0.2f', 1-0.01))



Zoomed in







# Part 2: Removing Noises in the Music Signal

### Q2: Apply the digital filter(s) to recover original music from a noisy signal.

Following are the steps taken in the process of removing the noise from the provided “[music\_noisy.wav](https://drive.google.com/file/d/1gAtWMpiYq1FV0v0BgtWDSePDaTjp44Iz/view?usp=sharing)” file that can be found in the course website.

**Identifying the noises in the noisy signal in the .wav file**

[y,fs]=audioread('music\_noisy.wav'); % reading the music with noise from the .wav file

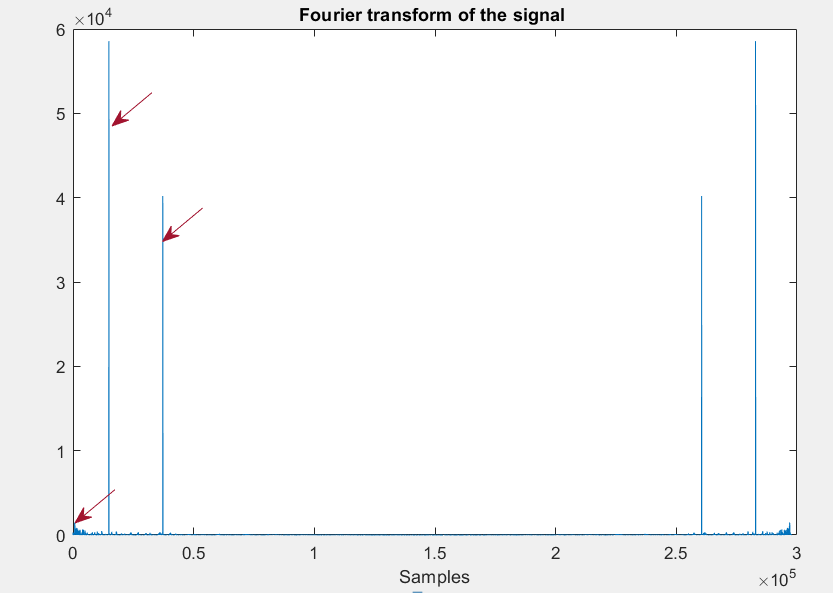
sound(y,fs); %listening to the corrupted music file

, y: input signal, fs: sample frequency of y

% fs = 22050 Hz

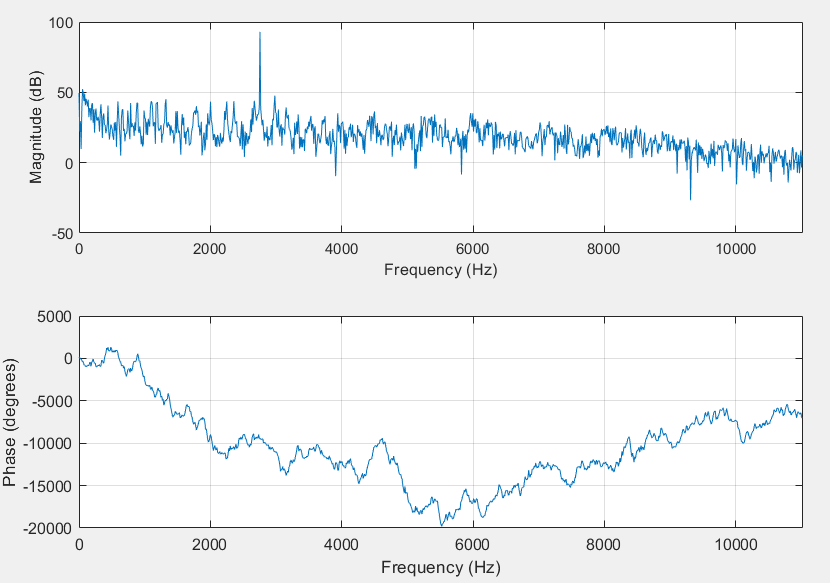
plot(abs(fft(y))); %Fourier transform of the noisy music

xlabel('Samples');

title('Fourier transform of the signal');

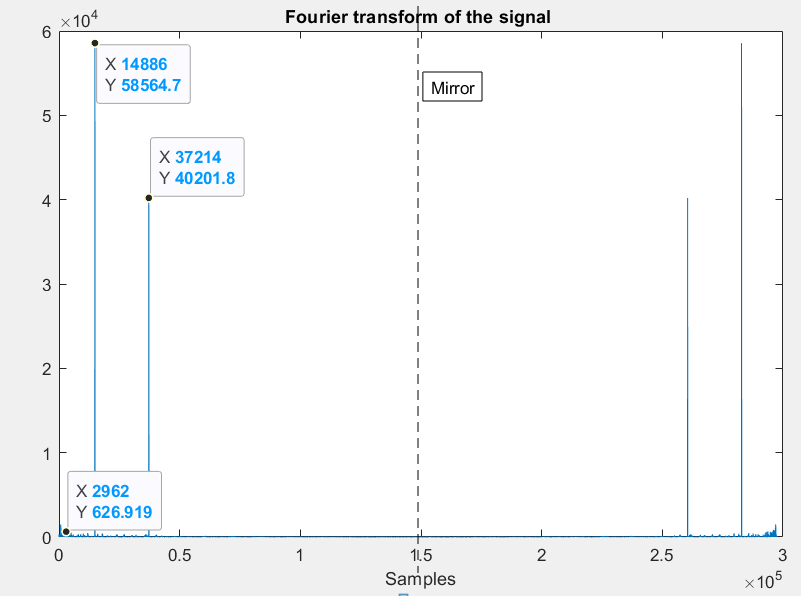
Red arrows are pointing at the unusual peaks/noise that are in this music file.

freqz(y,1,1024,fs); % displaying the magnitude and phase responses of the noisy music



**Designing a filter**

From the previous observations



The observable space is about 300 000 samples and since we are looking at a sample frequency of 22050 coming from the “music\_noisy.wav” file, ratio is around

There are multiple small peaks at the till 2962, which is about -> can use a high pass filter to remove all the small peaks before ~ 217.7 Hz.

1. **A High Pass filter with ~ 218 Hz as the cut off frequency.**

There is a big peak at about 14886, which is -> can use a bandstop filter to remove this peak.

1. **A Band Stop filter with ~ 1094 ± 300 Hz as the cut off frequency.**

There is another big peak at about 37214, which is -> can use a bandstop filter to remove this peak.

1. **A Band Stop filter with ~ 2727 Hz ± 300 Hz as the cut off frequency.**

With the found specifications following are 2 methods used in this report to remove the noise.

*1)Time Domain Method*

*2)Frequency Domain Method*

Functions used: ‘[fir1](https://www.mathworks.com/help/signal/ref/fir1.html)’

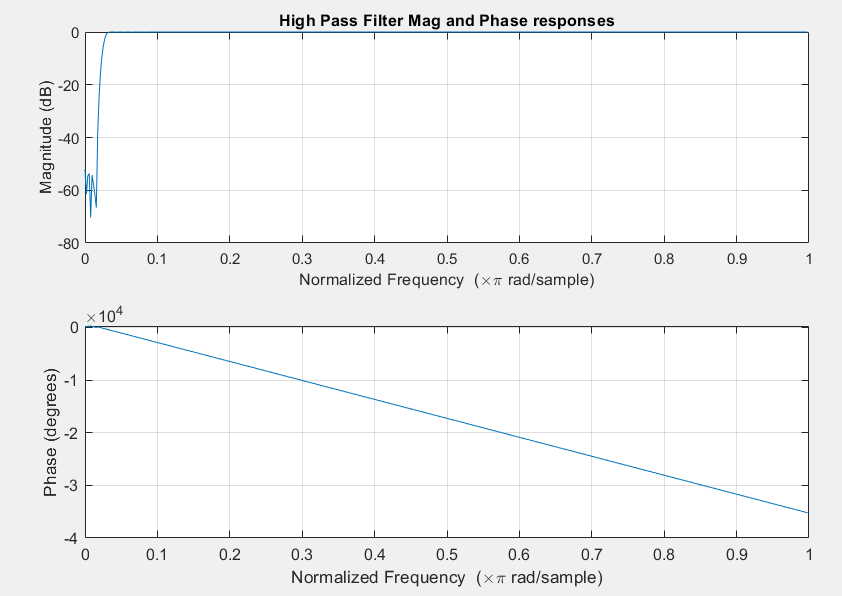
Note: fir1(n,Wn) -> decided to take a higher nth order (about 400) value since higher the number the more better results. Wn are the cut-off frequencies but will have slightly different values than decided before because following used in the code are adjusted so that a music is recovered in the best way possible.

**CODE OF THE FILTER DESIGN**

fhp1 = fir1(400, 274/(fs/2), 'high'); % high pass filter

freqz(fhp1,1);

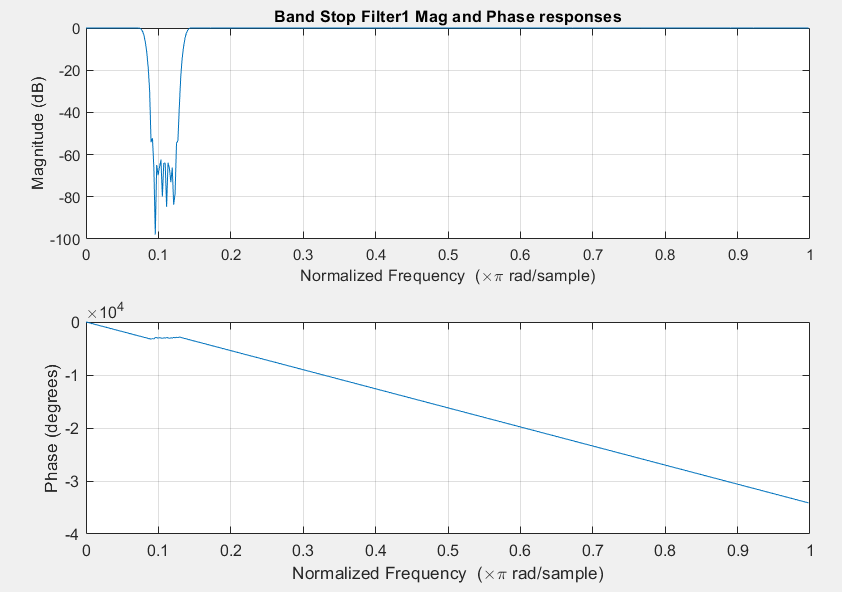
title('High Pass Filter Mag and Phase responses');



fbs2 = fir1(400, [900, 1500]/(fs/2), 'stop'); % band stop filter 1

freqz(fbs2,1);

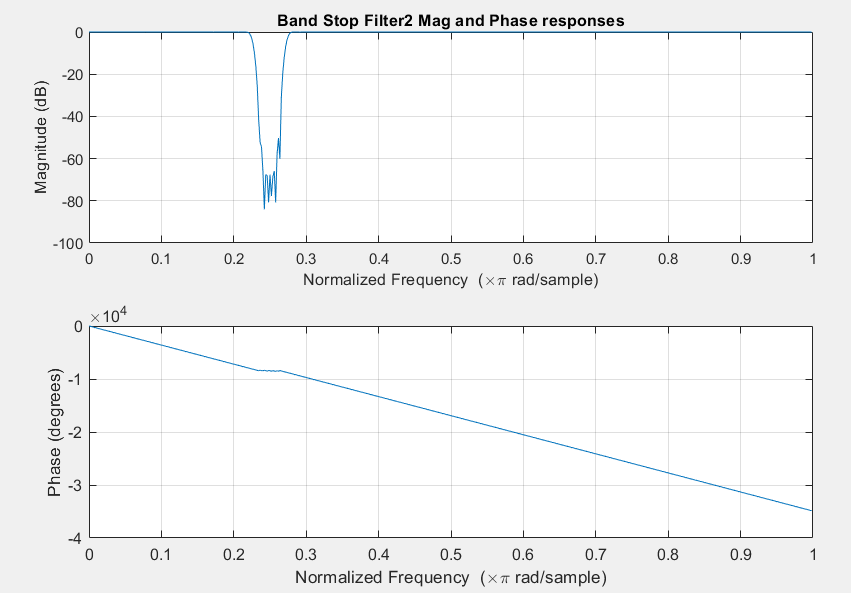
title('Band Stop Filter1 Mag and Phase responses');



fbs3 = fir1(400, [2500, 3000]/(fs/2), 'stop'); % band stop filter 2

freqz(fbs3,1);

title('Band Stop Filter2 Mag and Phase responses');



***1) Time Domain Method***

*Input signal -> convolution with the filters -> Output filtered signal*

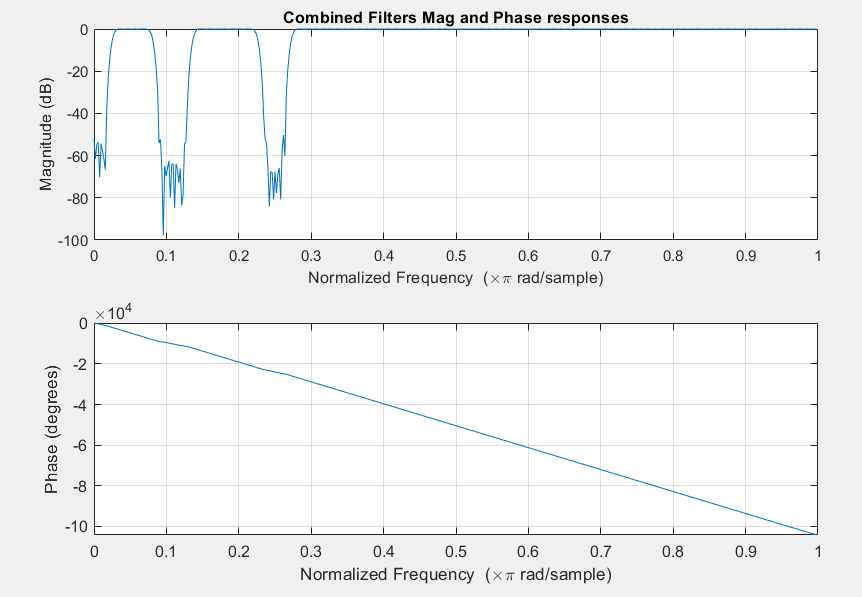
**CODE AND RESULTS**

filter\_Combine = conv(fhp1, fbs2); % convolution of 1st 2 filters

filter\_Combine = conv(filter\_Combine, fbs3); % convolution with the remaining filters

freqz(filter\_Combine,1);

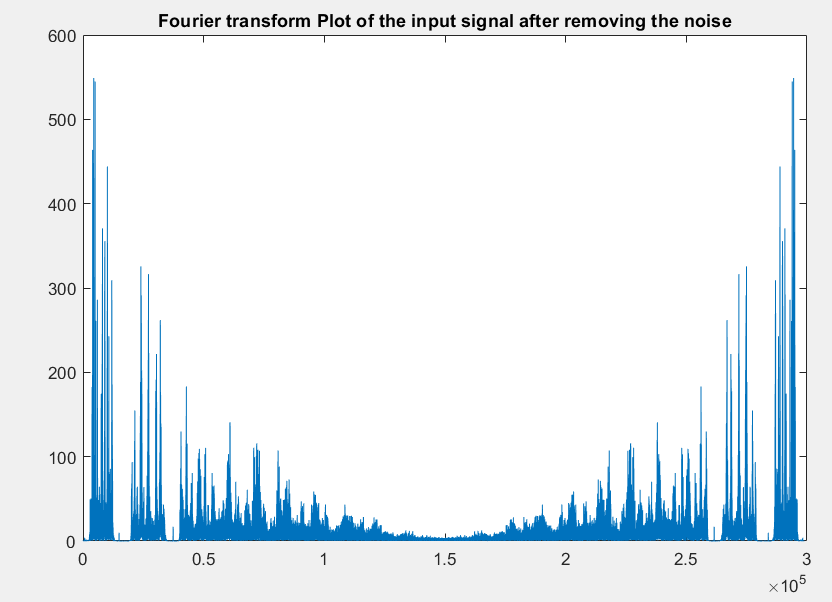
title('Combined Filters Mag and Phase responses');



filtered\_music = conv(filter\_Combine, y); % convolution of the signal with filters

plot(abs(fft(filtered\_music))); % Fourier transform of the filtered music

title('Fourier transform Plot of the input signal after removing the noise');



sound(filtered\_music,fs) % listen to verify

audiowrite('Filtered\_music\_td\_method.wav',filtered\_music,fs); % store the new filtered music

Click on this [link](https://drive.google.com/file/d/1_AiXGNxISbXtBDj0L9DaaEQmUaMBgbuM/view?usp=sharing) to listen to the filtered music.

***2) Frequency Domain Method***

*Input signal X filters -> Output filtered signal*

**CODE AND RESULTS**

fft\_y = fft(y); % Fourier transform of the input signal (music with noise)

fft\_fhp1 = fft(fhp1, length(fft\_y))'; % Fourier transform of first filter

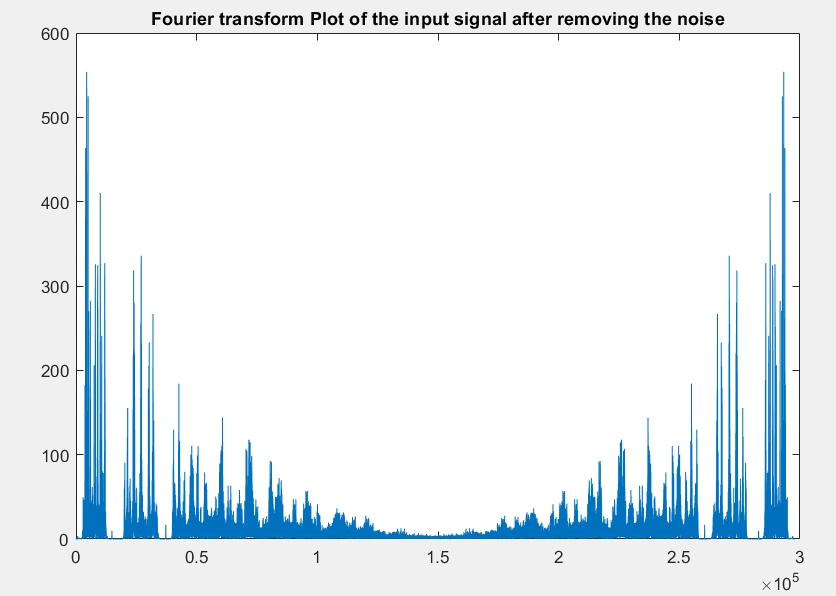
fft\_fbs2 = fft(fbs2, length(fft\_y))'; % Fourier transform of second filter

fft\_fbs3 = fft(fbs3, length(fft\_y))'; % Fourier transform of third filter

filtered\_music = fft\_y.\*fft\_fhp1.\*fft\_fbs2.\*fft\_fbs3; % multiply input signal with the filters in their frequency domains

plot(abs(filtered\_music));

title('Fourier transform Plot of the input signal after removing the noise');



filtered\_music\_time\_domain = ifft(filtered\_music); % inverse fourier transform of the final filtered signal to listen it later

sound(filtered\_music\_time\_domain,fs) % listen to verify

audiowrite('Filtered\_music\_fd\_method.wav',filtered\_music\_time\_domain,fs); % store the new filtered music

Click on this [link](https://drive.google.com/file/d/1RUKLi9QHd0SY7EUQS-9is5FCEawzbUiM/view?usp=sharing) to listen to the filtered music.

HURRAYY!!! NO MORE NOICE

**4. Conclusion: state what you learn from this lab, lab objectives you achieved, and any difficulties you met.**

Project helped to learn how to design FIR filters and use then in different methods to remove noise from a music signal using MATLAB.

It was satisfying to hear the music without the noise and make me wonder how these techniques have been applied in real work application.

Thank you for having set up this interesting mini project and other previous labs. I learned the most from these labs-based assignments.