Experiment - 1

Manomukil T - EE20B075

7th October 2022

Contents

1 Part 1:

Downsample each audio signal by 2 without any AA filtering. Play back using the new sampling rate.

1.1 Code:

```
%% Magnitude response of both the signals

music_amplitude = abs(fft(music_array));
speech_amplitude = abs(fft(speech_array));

% Generating all the frequencies
music_frequency = transpose((0:length(music_amplitude)-1)*music_fs/length(music_amplitude));
speech_frequency = t/transpose((0:length(speech_amplitude)-1)*speech_fs/length(speech_amplitude));

%% Downsample by 2
music_ds_array = downsample(music_array,2); speech_ds_array = downsample(speech_array,2);

% Fourier transform of both the signals
music_ds_amplitude = abs(fft(music_ds_array)); speech_ds_amplitude = abs(fft(speech_ds_array));
```

1.2 Plots:

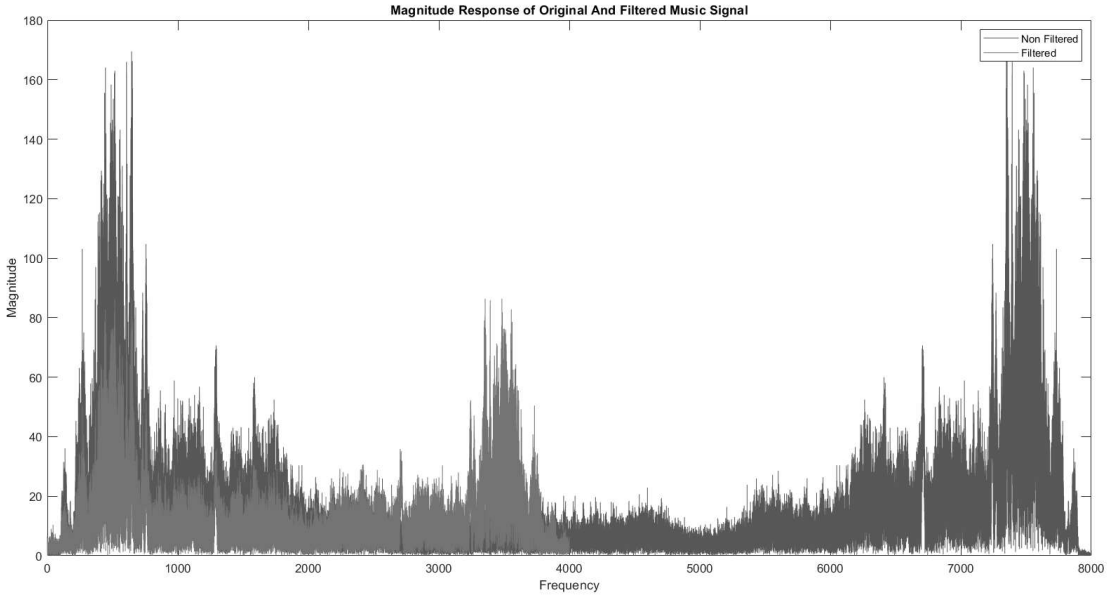

Figure 1: Original Signal vs Filtered Signal of the Music Audio File

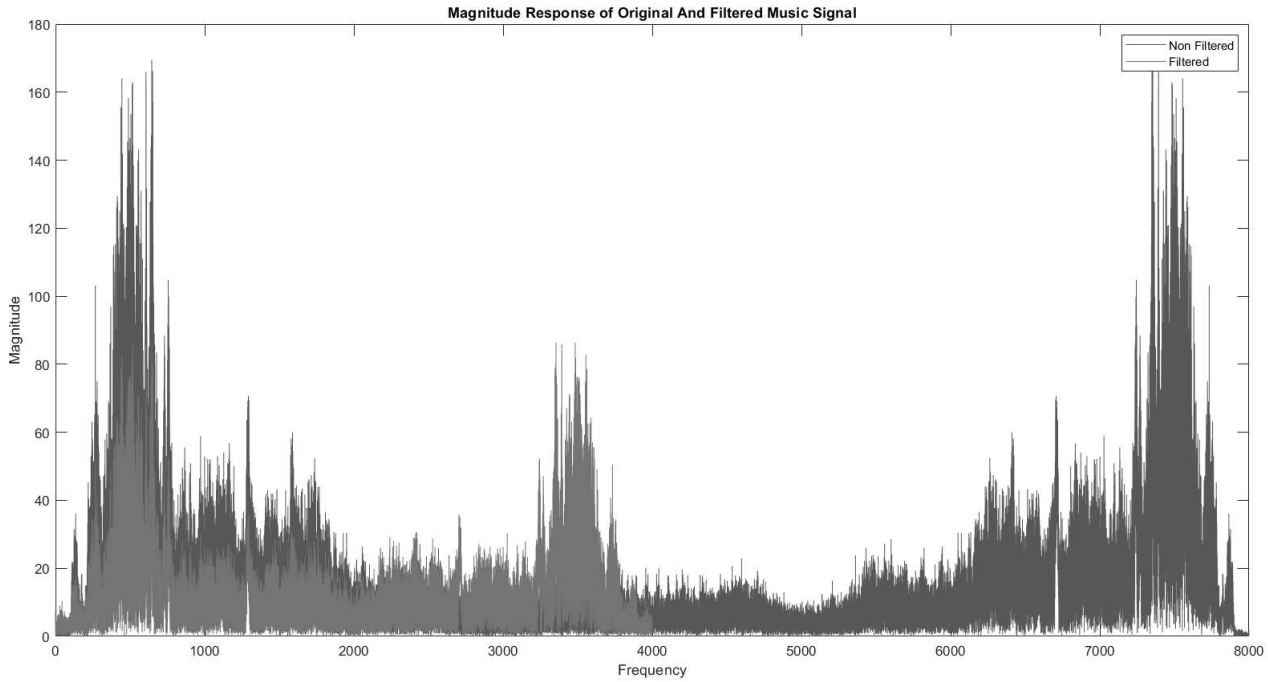

Figure 2: Original Signal vs Filtered Signal of the Speech Audio File

1.3 Result

The signal becomes worse after being passed through the downsampler. This occurs due to aliasing.

2 Part 2

Design an equiripple LPF with $\omega p = 0.45\pi$ and $\omega s = 0.55\pi$, with appropriate values for δp and δs. Using this as an AA filter, downsample each audio signal by 2. Playback using the 1 new sampling rate. Compare the two downsampled signals (with and without AA filtering) for each audio input.

2.1 Code

```
%% Design Filter
Hd = fdesign.lowpass('Fp,Fst',0.45,0.55);
d = design(Hd,'equiripple');

%Plot the Magnitude Response of the Filter
fvtool(d);
```

```
%Pass the signals through the AA Filter
music_AA_array = filter(d,music_array);
speech_AA_array = filter(d,speech_array);

%Downsample the signals
music_AA_ds_array = downsample(music_array,2);
speech_AA_dsarray = downsample(speech_array,2);

%Apply Fourier transform
music_AA_ds_amplitude = abs(fft(music_AA_ds_array)); speech_AA_ds_amplitude = abs(fft(speech_AA_dsarray));
```

2.2 Plots

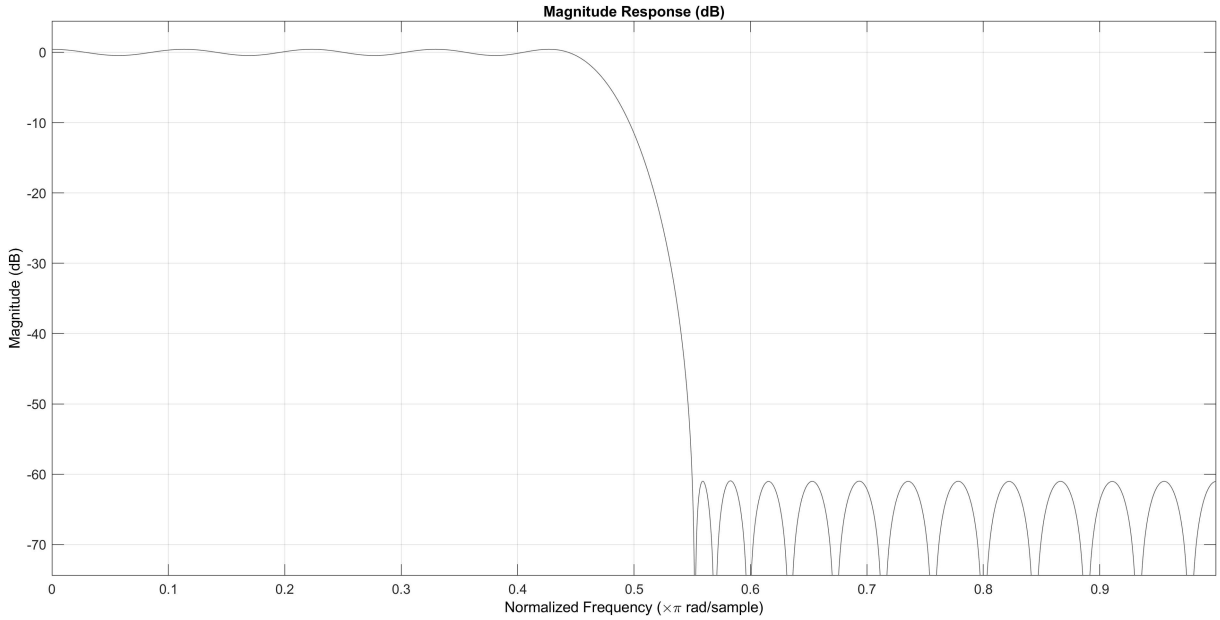

Figure 3: Magnitude Response of Equiripple LPF

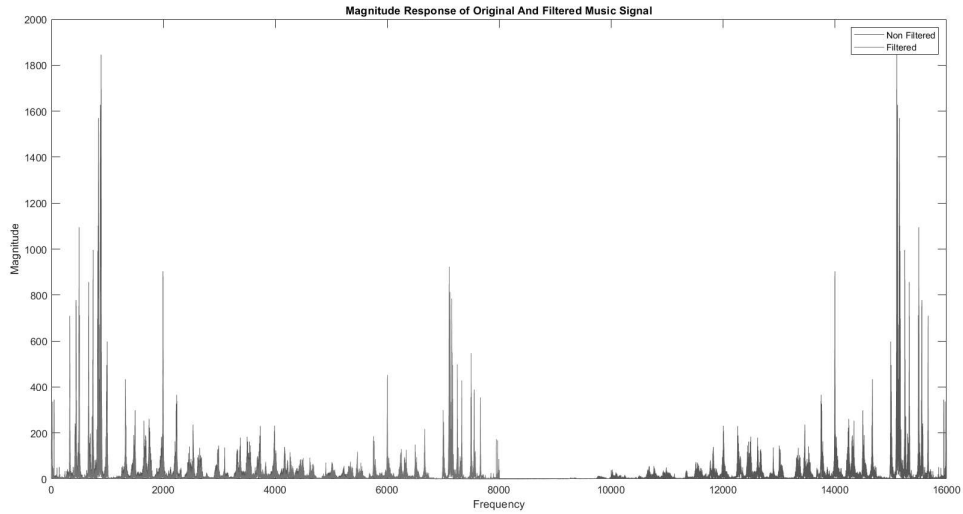

Figure 4: Original vs Filtered Response of Music Audio Signal

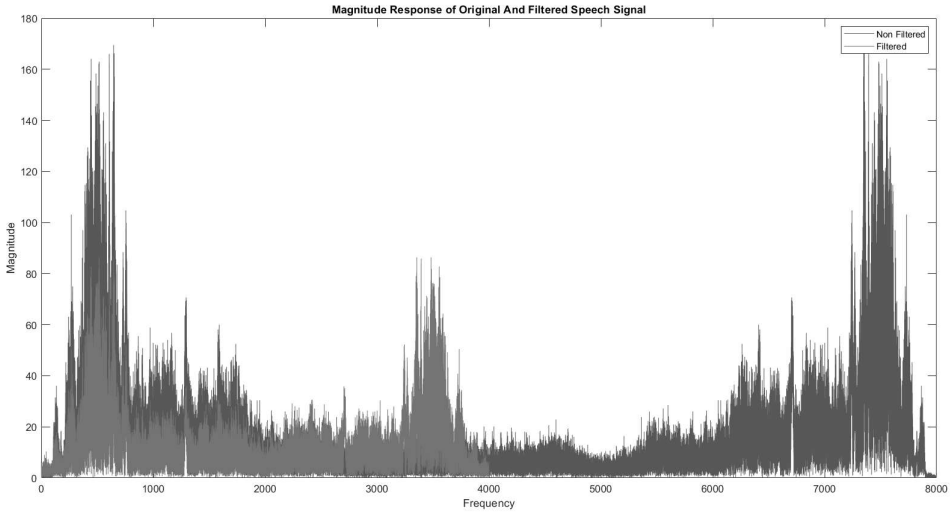

Figure 5: Original vs Filtered Response of Speech Audio Signal

2.3 Result

We have implemented the low pass filter as shown in Figure 3, such that we have suppressed the high frequency components.as shown in Figures 4 and 5.

3 Part 3

3.1 Code

```
Hd2 = fdesign.lowpass('Fp,Fst',0.22,0.28);
d2 = design(Hd2,'equiripple');
```

3.2 Plots

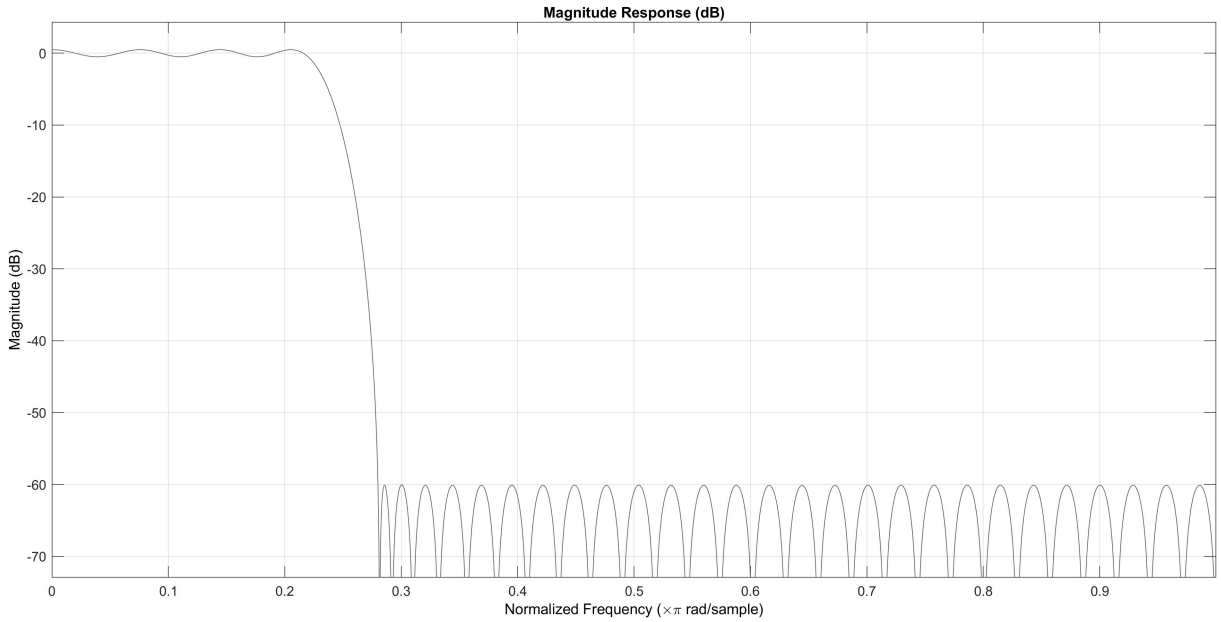

Figure 6: Magnitude Response of the Equiripple Low Pass Filter

3.3 Result

The filter has been designed for $\omega_p = 0.22\pi$ and $\omega_s = 0.28\pi$. Thus we have a filter with a cutoff frequency of $\pi/4$.

4 Part 4

First upsample each original audio signal by 3. Then apply the LPF with cutoff $\pi/4$ designed above, and downsample by 4. Play the re-sampled outputs using the new sampling rate. Compare these outputs with the originals.

4.1 Code

```
%Upsampling by 3
    music_us_array = upsample(music_array,3);
```

```
speech_us_array = upsample(speech_array,3);
music_us_amplitude = abs(fft(music_us_array));
speech_us_amplitude = abs(fft(speech_us_array));
%Passing the array through AA
filter music_us_AA_array = filter(d2,music_us_array);
speech_us_AA_array = filter(d2,speech_us_array);
%Downsampling by 4
music_us_AA_ds_array = downsample(music_us_AA_array,4);
speech_us_AA_ds_array = downsample(speech_us_AA_array,4);
%Converting to get Magnitude Response
music_us_AA_ds_amplitude = abs(fft(music_us_AA_ds_array));
speech_us_AA_ds_amplitude = abs(fft(speech_us_AA_ds_array));
music_us_fs = music_fs*3;
speech_us_fs = speech_fs*3;
music_us_ds_fs = music_us_fs/4;
speech_us_ds_fs = speech_us_fs/4;
```

4.2 Plots

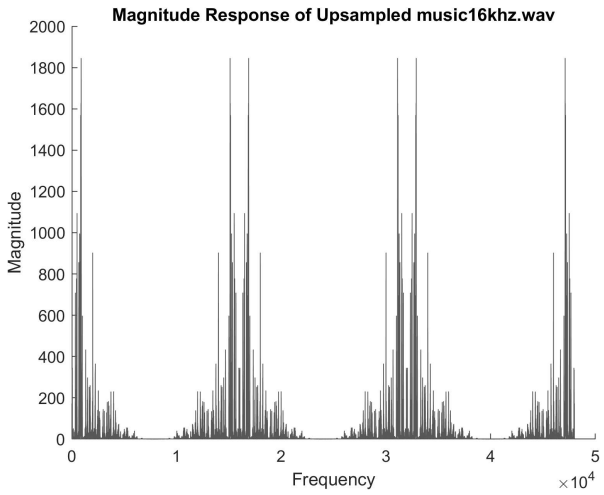

Figure 7: Upsampled Music Signal

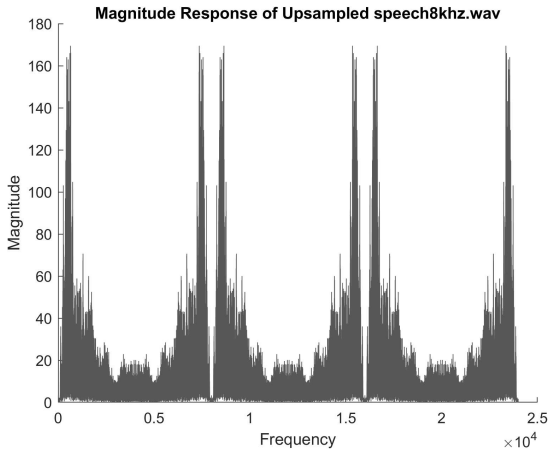

Figure 8: Upsampled Speech Signal

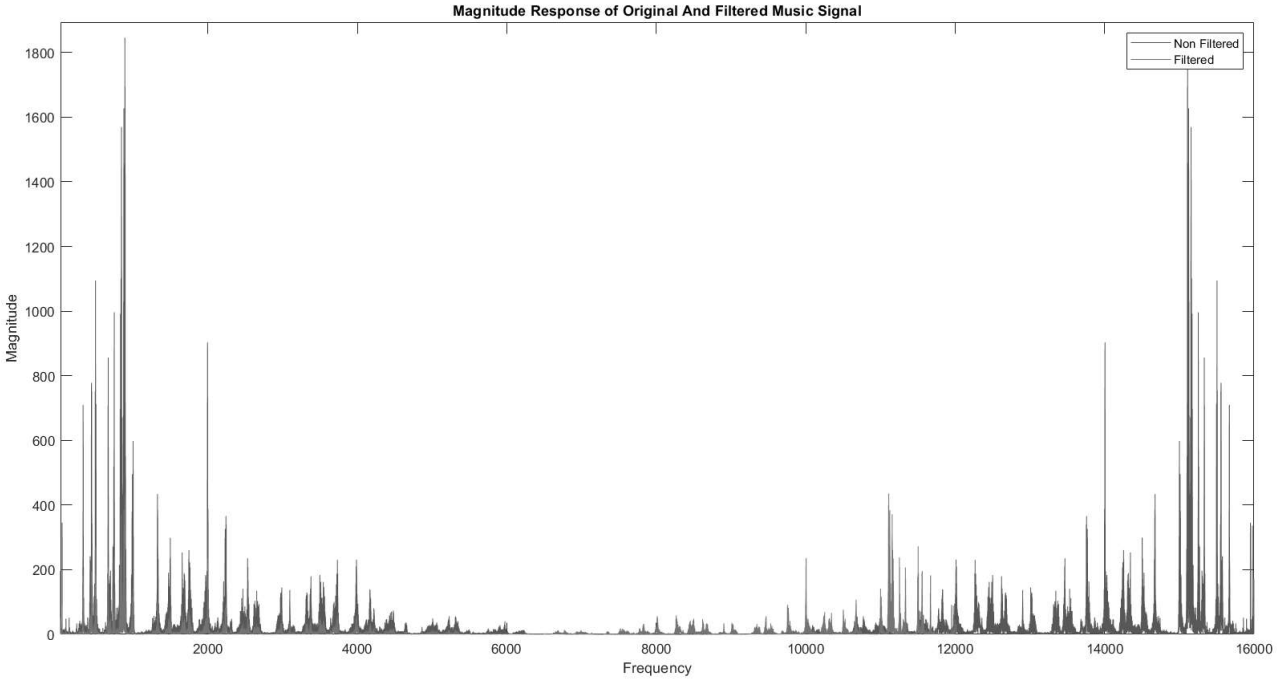

Figure 9: Original vs Filtered Music Signal

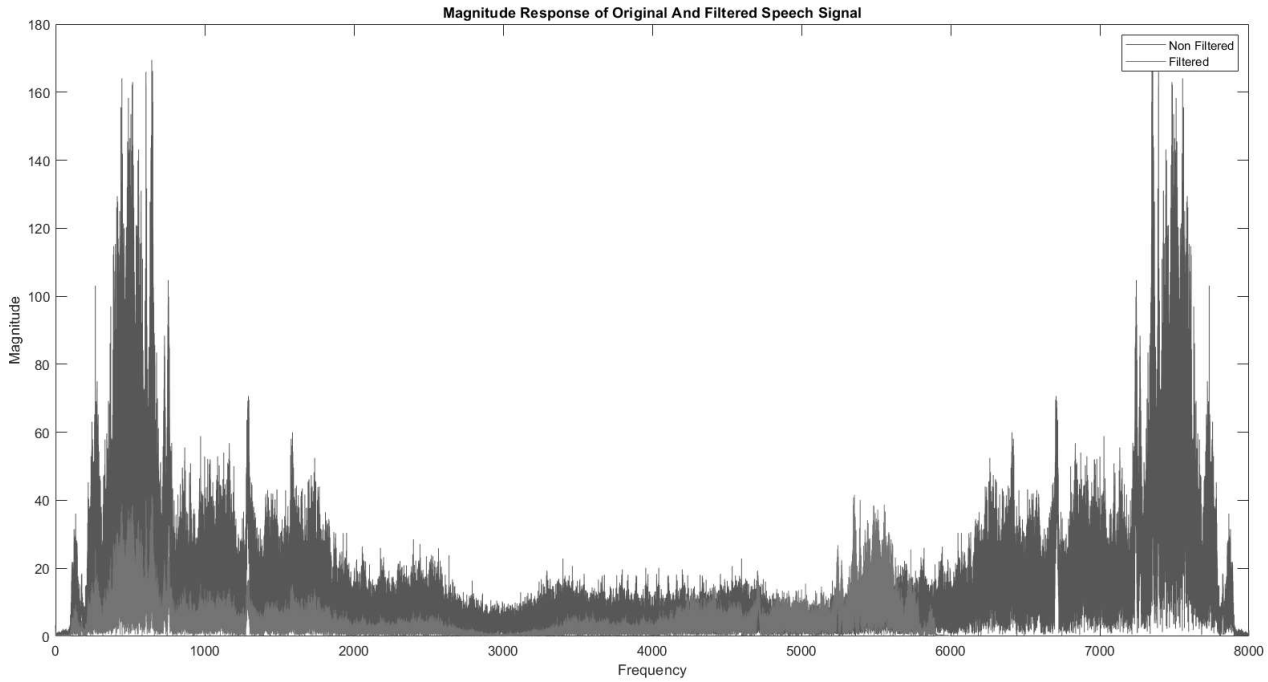

Figure 10: Original vs Filtered Speech Signal

4.3 Result

We can see 3 images and in both Figures 9 and 10 . The higher frequency components has been eliminated using the AA filter. Now the images are the cause for the deterioration in quality.

5 Part 5

First filter each audio original signal with the LPF designed above, and downsample by 4. Then upsample by 3. Finally, interpolate each upsampled signal using an interpolation filter for L = 3 with support [−8, 8] designed by any method (including windowing). (If you use "intfilt" of Matlab, set $\alpha = 1$.) Play these outputs and compare them with those of the previous part

5.1 Code

```
%Interpolation Filter Design
h1 = intfilt(3,3,1);
n_imp = linspace(-8,8,length(h1));
h1_mag = fftshift(fft(h1),17);
```

```
%Filter using the LPF music_AA_array = filter(d2,music_array);
speech_AA_array = filter(d2,speech_array);

%Downsample by 4
music_AA_ds_array = downsample(music_AA_array,4);
speech_AA_ds_array = downsample(speech_AA_array,4);

%Upsample by 4

music_AA_ds_us_array = upsample(music_AA_ds_array,3);
speech_AA_ds_us_array = upsample(speech_AA_ds_array,3);

%Filter using Interpolation Filter

music_AA_ds_us_int_array = filter(h1,1,music_AA_ds_us_array);
speech_AA_ds_us_int_array = filter(h1,1,speech_AA_ds_us_array);

%Apply Fourier Transform

music_AA_ds_us_amplitude = abs(fft(music_AA_ds_us_array));
speech_AA_ds_us_amplitude = abs(fft(speech_AA_ds_us_array));

music_AA_ds_us_int_amplitude = abs(fft(music_AA_ds_us_int_array));
speech_AA_ds_us_int_amplitude = abs(fft(speech_AA_ds_us_int_array));
```

5.2 Plots

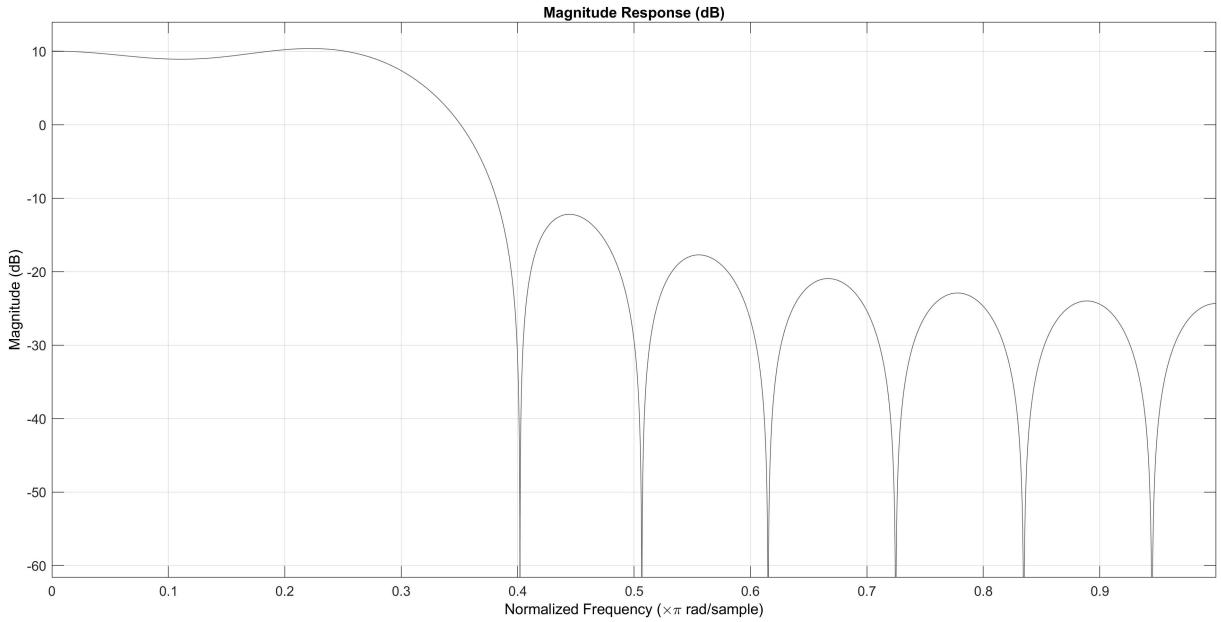

Figure 11: Magnitude Response of Interpolation Filter

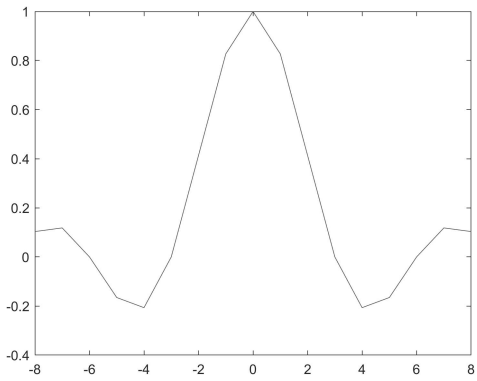

Figure 12: Impulse Response of Interpolation Filter

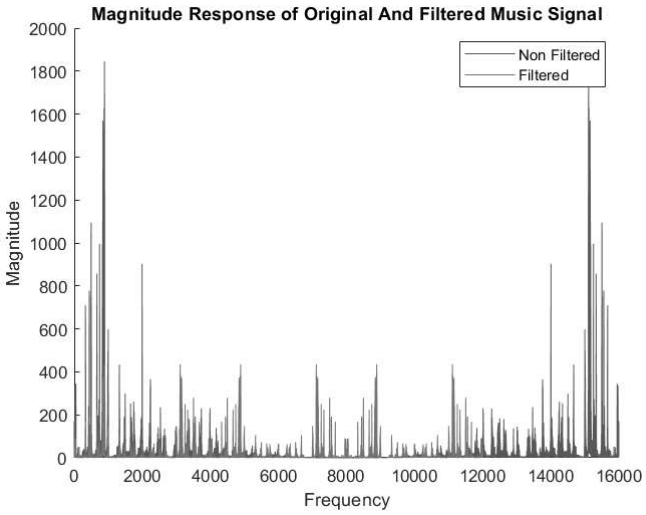

Figure 13: Original vs Filtered Music Signal

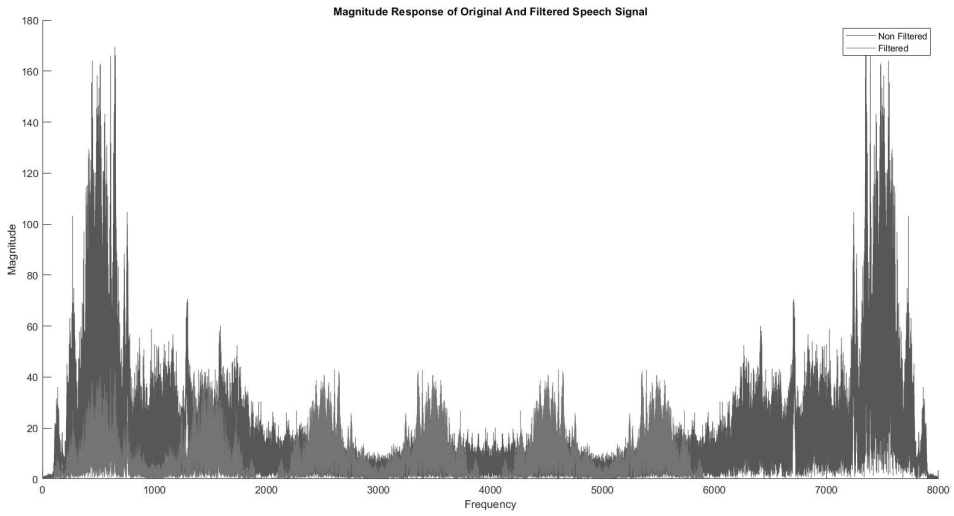

Figure 14: Original vs Filtered Speech Signal

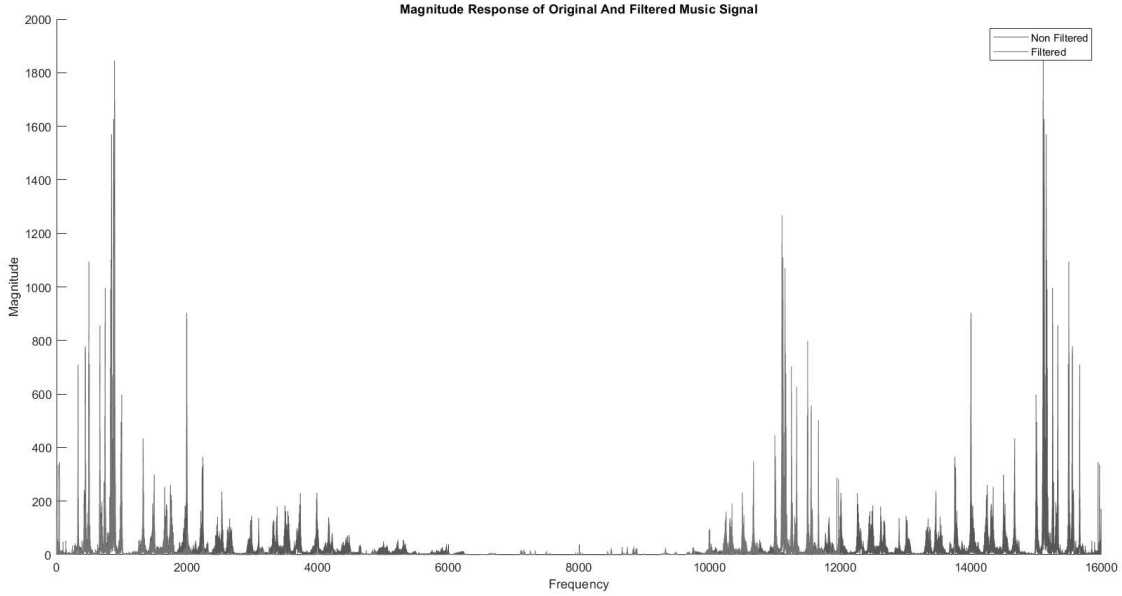

Figure 15: Original vs Filtered Music Signal With Interpolation Filter

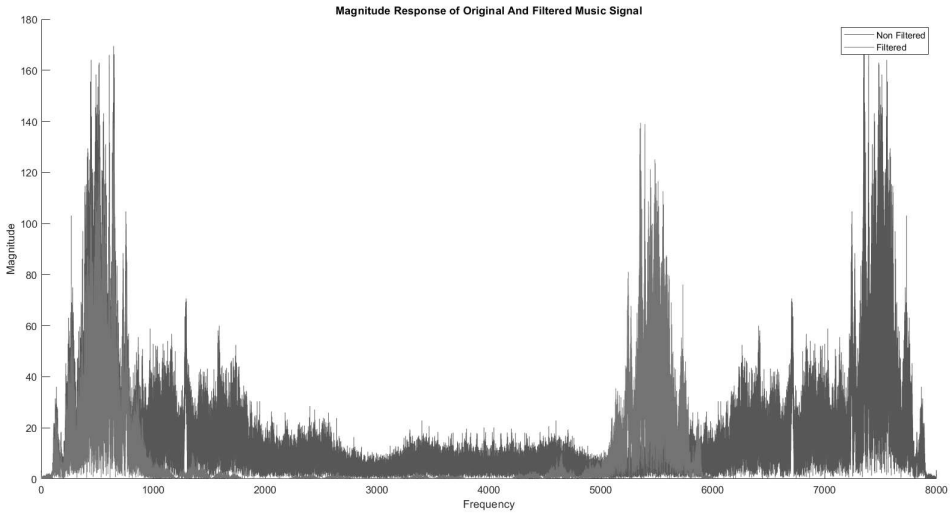

Figure 16: Original vs Filtered Music Signal With Interpolation Filter

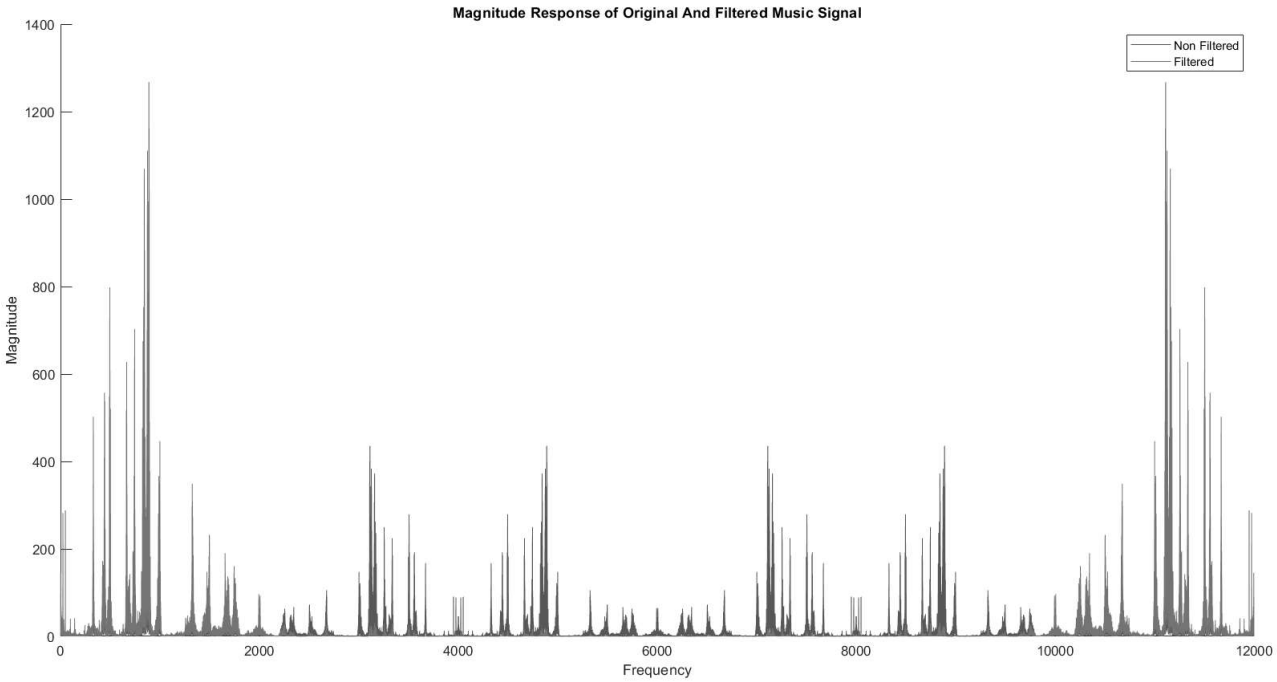

Figure 17: Comparison of Music Signal with and without Interpolation

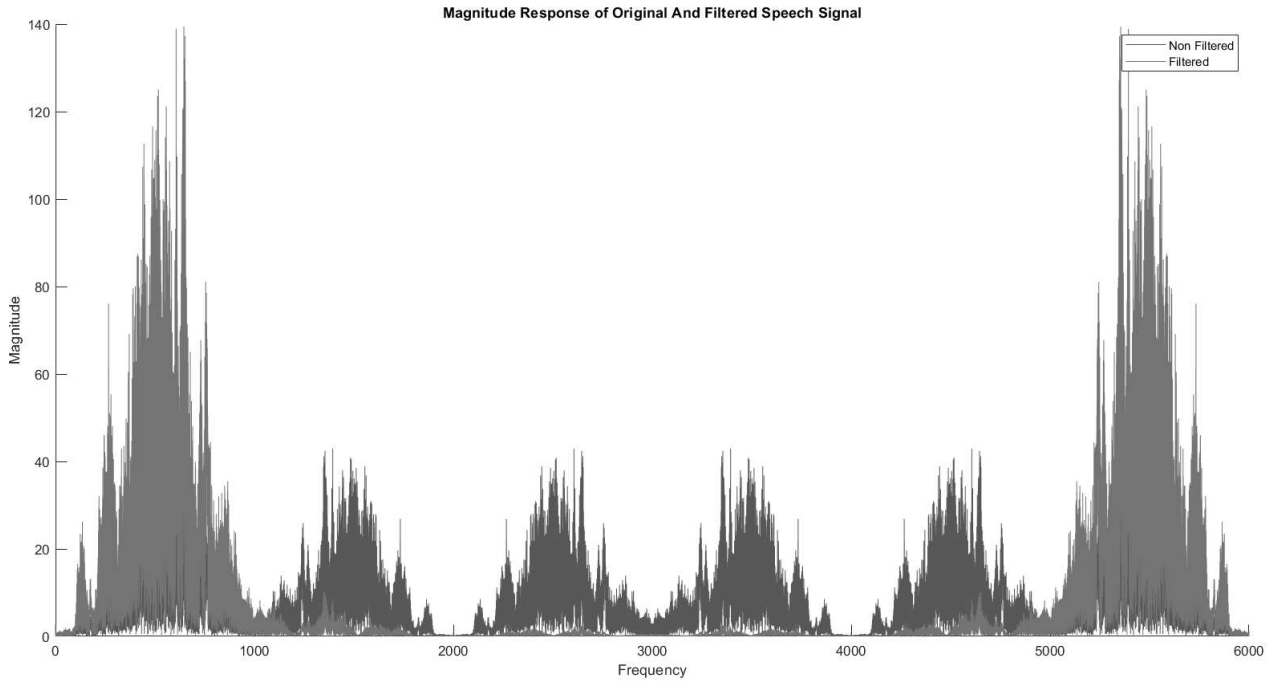

Figure 18: Comparison of Speech Signal with and without Interpolation

5.3 Result

In Figures 17 and 18, using the Interpolation filter, we eliminate the images that were seen in Figures 13 and 14 as shown , thus implying the effectiveness of the Interpolation Filter in improving the quality of the signal compared to the previous section.