Digital Signal Processing of Audio Signals

ENGN4537 - Digital Signal Processing

School of Engineering - The Australian National University

Josh Johnson - u6044123

May 30, 2019

1 Input/Output of WAV Files

Q 1.1

The sampling frequency of `Gwen.wav` is 44100 Hz.

Q 1.2

The number of bits/sample in `Gwen.wav` is 16 bits.

2 Voice Data Collection

Q 2.1

Please find `HD_x_Johnson.wav` attached.

3 Matlab Function for Frequency Analysis

Q 3.1

Please find `dft1.m` attached.

Q 3.2

Please find `dft2.m` attached.

Q 3.3

Running time of *dft1*: 3.025 seconds.
Running time of *dft2*: 0.002 seconds.

Q 3.4

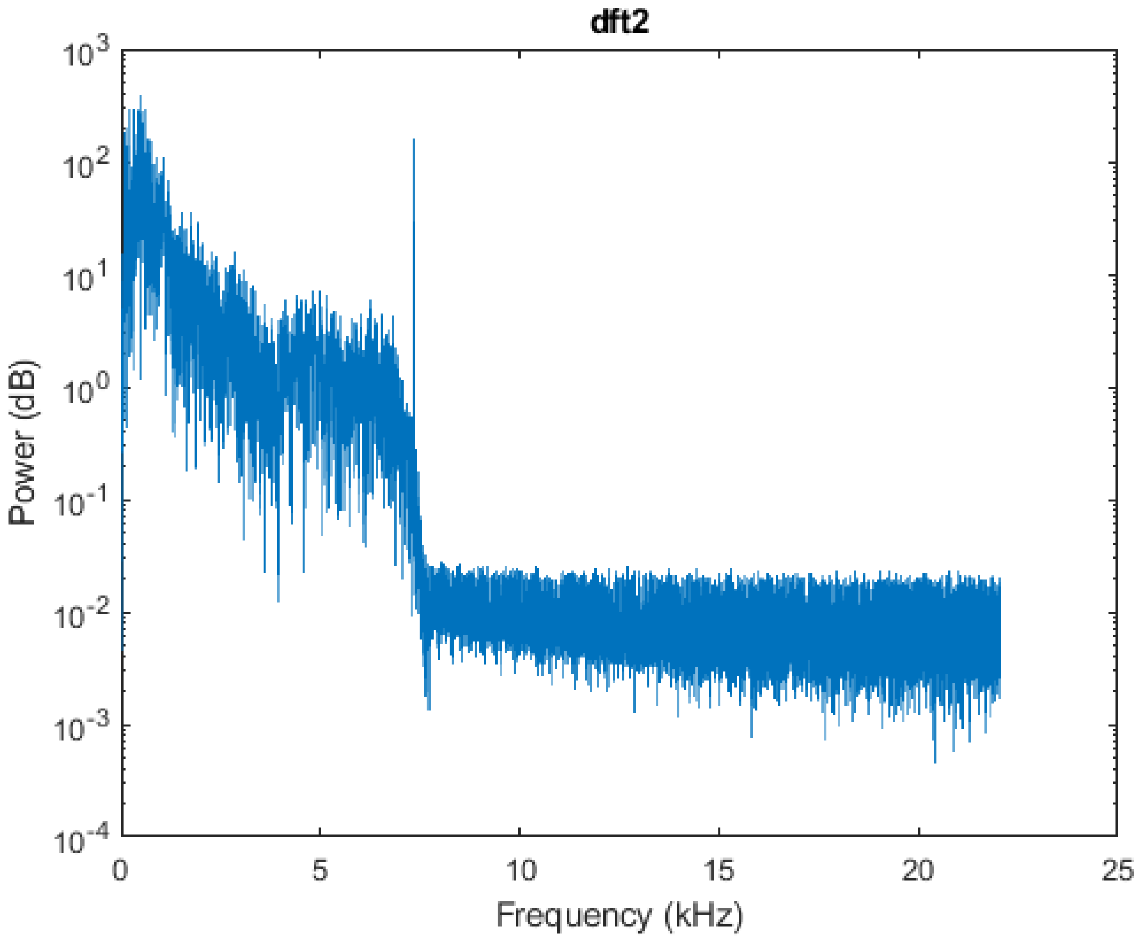

Figure 1: DFT of `HD_x_Johnson.wav`

As can be seen the above plot, there is frequency content between DC and 9 kHz, with the majority of it being below 3 kHz, and a spur at 8 kHz. The frequency content below 3 kHz is resultant from my speech, whilst the spur at 8 kHz is a result of the sine wave which was played during the recording of the audio.

Q 3.5

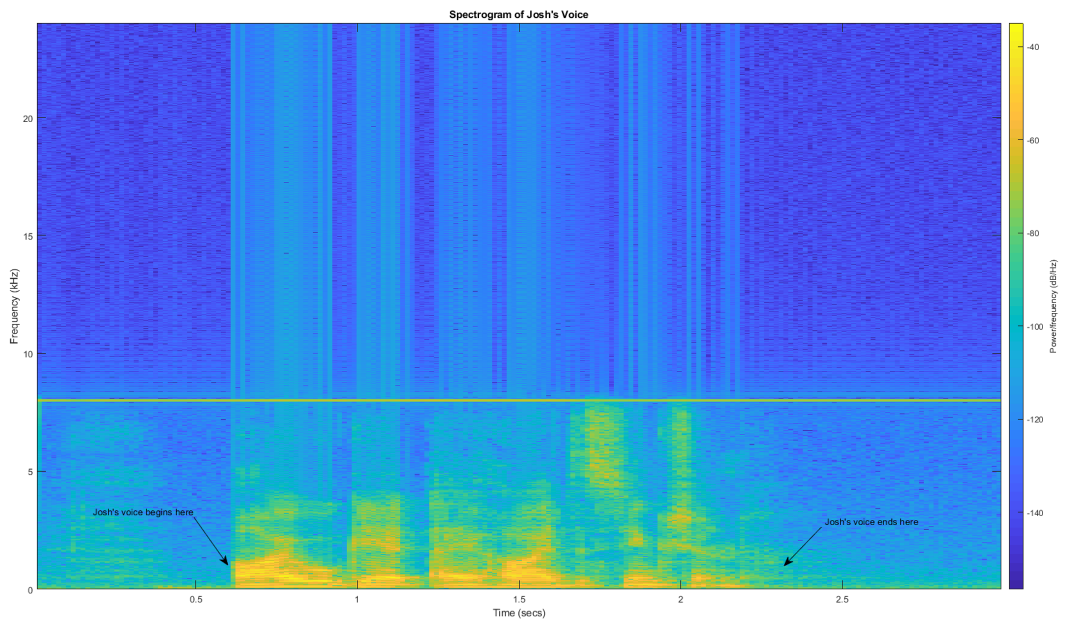

Figure 2: Spectrogram of `HD_x_Johnson.wav`

From looking at the above spectrogram we can see how the DFT of a signal changes with respect to time, instead of seeing a single DFT for the entire time period. It can be seen that the lower frequency content of the audio clip start approximately 0.6 seconds into the recording and ends at around 2.3 seconds into the clip. Given that the human voice is in the same frequency range as the lower frequency content shown above, it can be concluded that my own voice begins and ends at the above mentioned time intervals.

Q 3.6

Please find `FindSignalStart.m` attached.
My implementation works by finding the median value of the frequency content between 100 Hz and 3 kHz, and then finding when in the time domain the signal crosses this threshold. This denotes the start of my speaking, and the signal is cropped accordingly.

Q 3.7

Please find `FindSignalEnd.m` attached.
`FindSignalEnd` works similarly to `FindSignalStart`, however if finds the last time domain point where the signal is higher than the threshold, and removes audio after that point.

Q 3.8

Please find `HD_c_Johnson.wav` attached.
The implementation for Q 3.8 simply feeds the output from `FindSignalStart` into `FindSignalEnd` which results in the pre and post silence being removed, and to my surprise it works quite well.

4 The DFT

Please find code for the below sections, `fftPlot.m`, attached.

Q 4.1

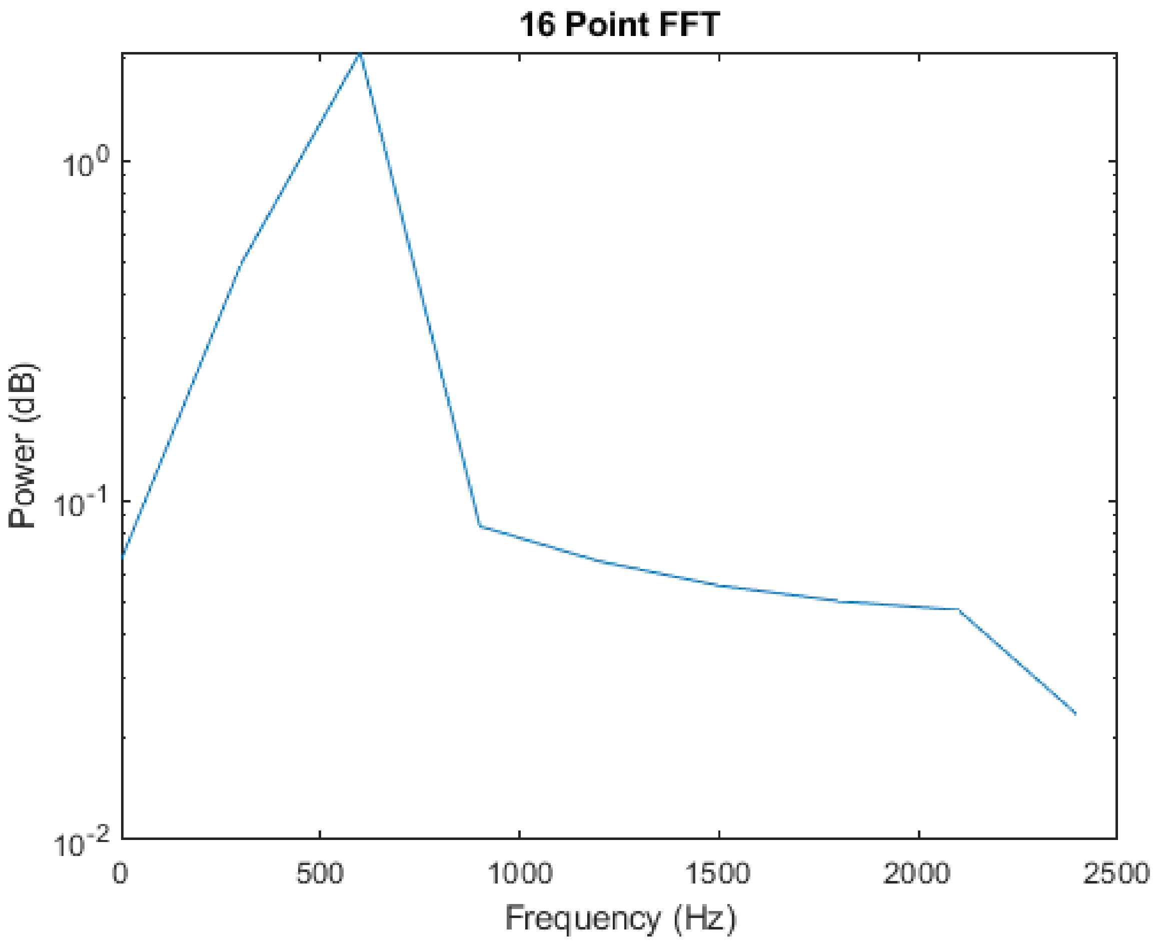

Figure 3: 16 Point DFT of *x(t)*

Due to the low sample count of a 16 point DFT, the time domain sample is unable to contain enough information regarding the frequency content for the for an accurate DFT to be produced. In saying this, it can be seen in Fig. 3 that there is a peak around 600 Hz, however given that the bin width of the DFT appears to be approx 500 Hz, the DFT is useful for indicative purposes at most.

Q 4.2

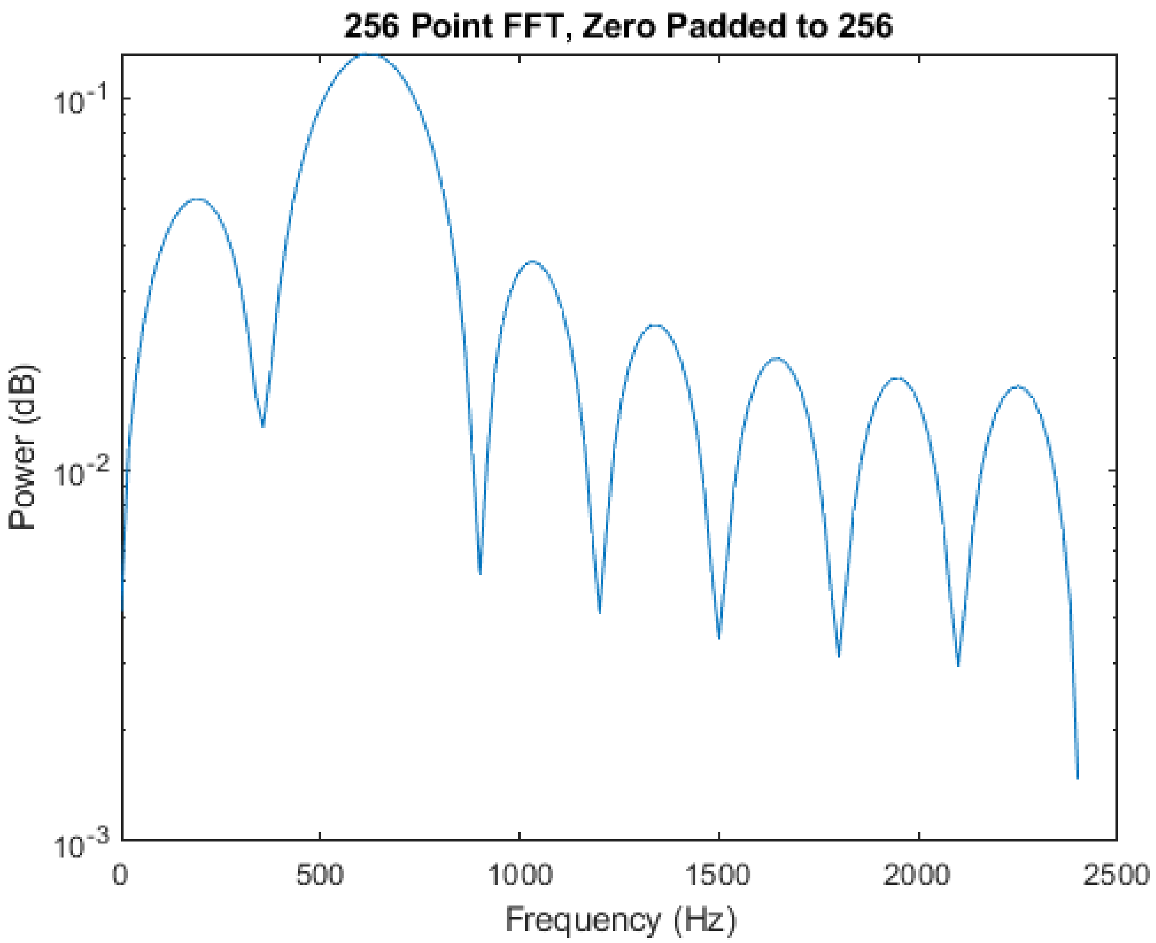

Figure 4: 16 Point DFT of *x(t)*, Zero Padded to 256 Points

Zero padding the 16 point DFT results in an increased length of the signal (increasing N) whilst not adding more of the original signal, which results in increased sampling of the existing signal. This is done as it leads to an increased frequency resolution of a signal without increasing the number of samples captured, which can be seen in the additional side lobes in the DFT.

Q 4.3

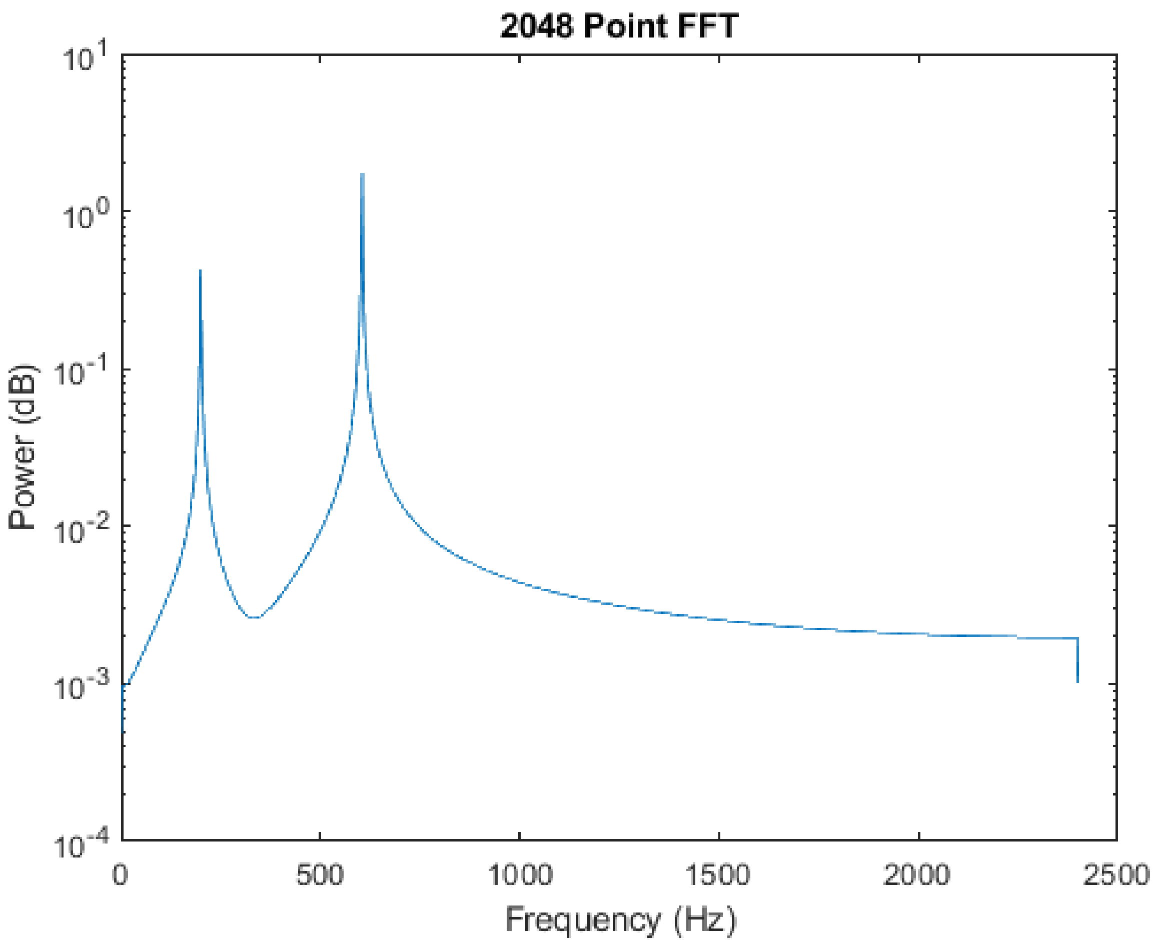

Figure 5: 2048 Point DFT of *x(t)*

As hinted to in Fig. 4, increasing the length of the signal sampled for a DFT results in a DFT which can better discriminate between the frequency content in a signal. This can be clearly seen in Fig. 5, where 2048 points were used to generated the above DFT, and the two fundamental frequencies can be seen in the resulting plot.

5 Filtering

Q 5.1

The filter does not do a good job at filtering the given noise, and as can be seen by in Fig. 6 below this is because the filter does not have a significant amount of frequency dependent attenuation which would result in reducing the amplitude of the 8 kHz tone.

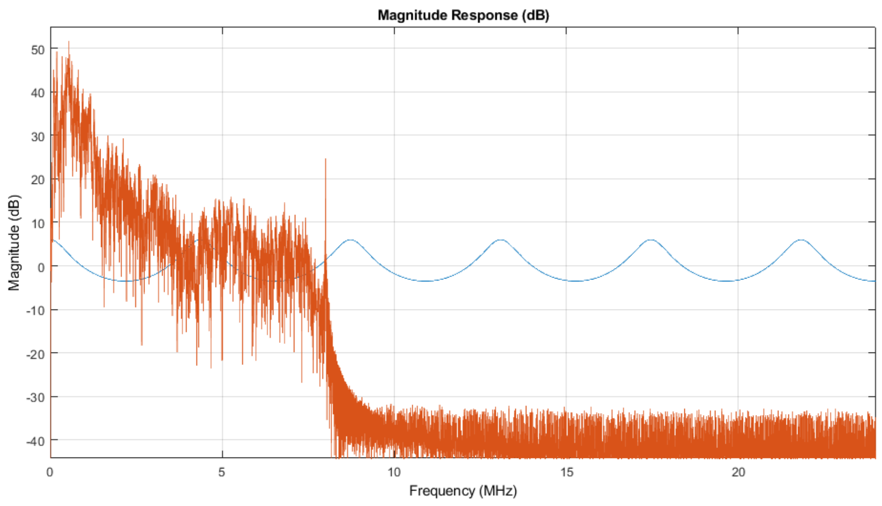

Figure 6: fvtool analysis of the given filter along with audio of my voice.

Q 5.2

The first filter (Fig. 7) has a greater than 60dB null at 8 kHz, which is the frequency we are trying to remove. It does however have 4 db of attenuation per 1 kHz on the lower side of 8 kHz, which will lead do degraded audio quality which is not ideal. In saying this, the first filter is the best option out of the provided filters to remove the unwanted tone.

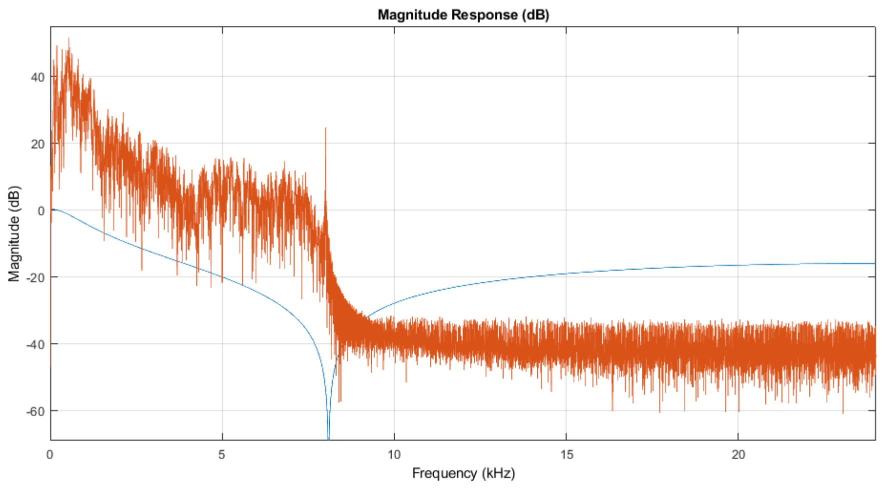

Figure 7: fvtool analysis of the first given filter.

The second filter (Fig. 8) has a much flatter frequency response either side of the null located at approximately 17 kHz, and so whilst it results in minimal distortion of my voice, it also does not provide significant attenuation to the unwanted tone and as such is not a suitable filter.

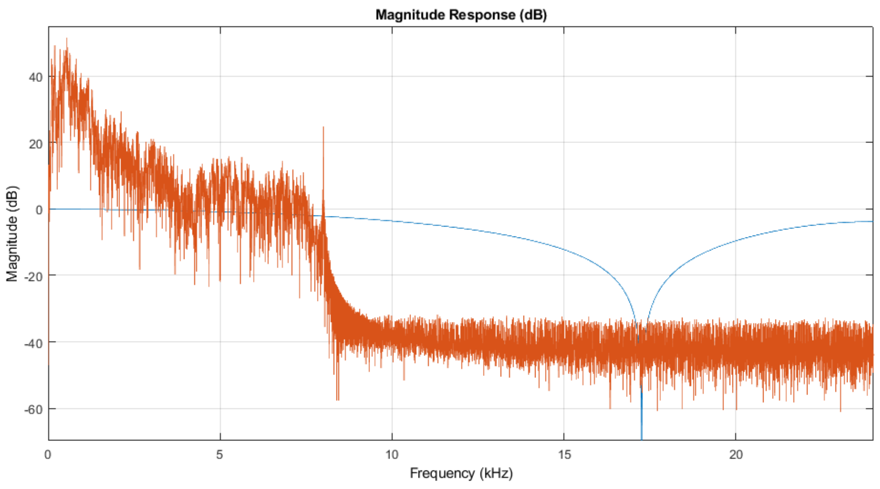

Figure 8: fvtool analysis of the second given filter.

The third filter (Fig. 9) is by far the worst for attenuating the unwanted tone, as it has a null in the middle of the voice frequency range, and a frequency response which then increases by 20 dB by 8 kHz. This will not only result in inaudible sections of voice, but with comparatively increase the sound level of the unwanted tone, making it worse than no filtering at all.

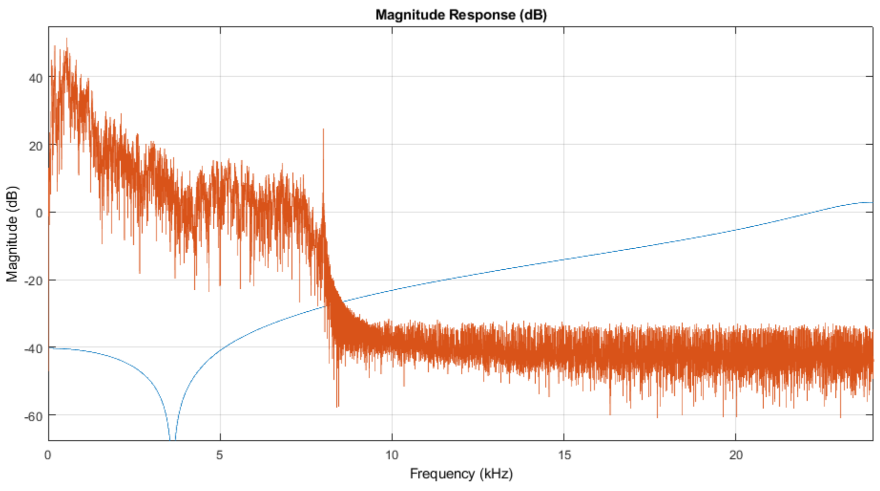

Figure 9: fvtool analysis of the third given filter.

Q 5.3

As can be seen below in Fig. 10, applying the first filter to the audio signal has significantly reduced the unwanted tone at 8 kHz, whilst minimising the attenuation seen to the lower frequencies which contain my speech. Listening to the audio, my words remain easy to understand whilst the unwanted tone is no longer audible.

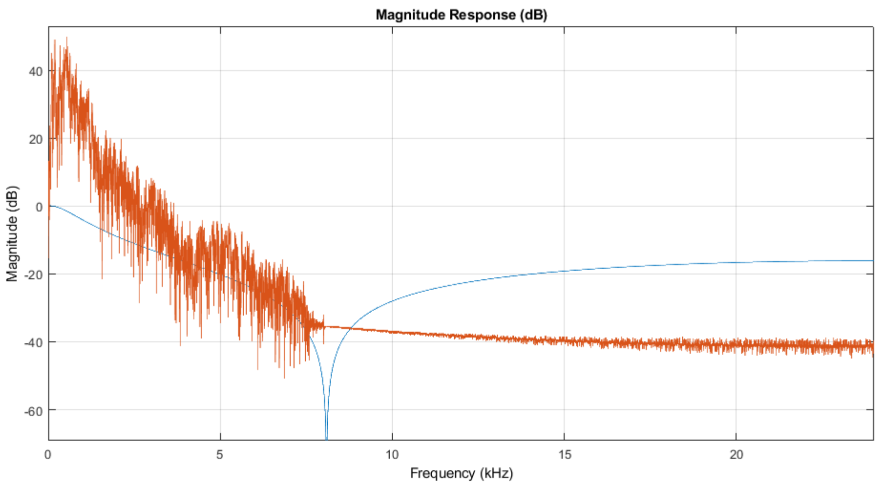

Figure 10: fvtool analysis of the first filter and filtered audio.

Q 5.4

As can be seen in the numerous plots above, my speech is within a frequency range of 0 Hz to 3 kHz, with the majority of that being in the lowest kHz. The sine wave which was played during the recording is clearly visible at approximately 8 kHz, and any frequencies above that tone are very small, with the noise floor of the system being flat from 9 kHz to 25 kHz.

6 Music Modulation and Windowed FIR LPF Design

Q 6.1

The below two figures (11 and 12) are time domain plots of the modulated signal, with Fig. 11 displaying the whole audio clip, and Fig. 12 displaying a subsection of the audio clip, to make the AM modulation visible. Figure 13 shows the DFT of the modulated audio clip, with the upper and lower side bands either side of the 9 kHz carrier clearly visible.

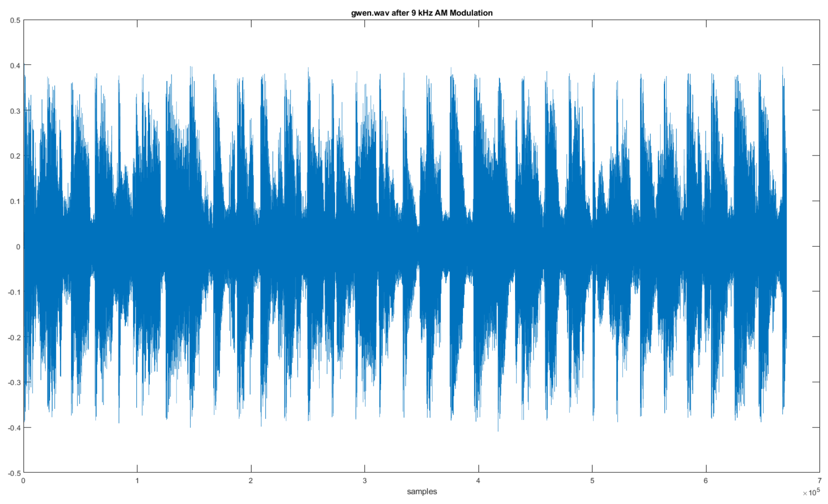

Figure 11: Time domain plot of the modulated audio.

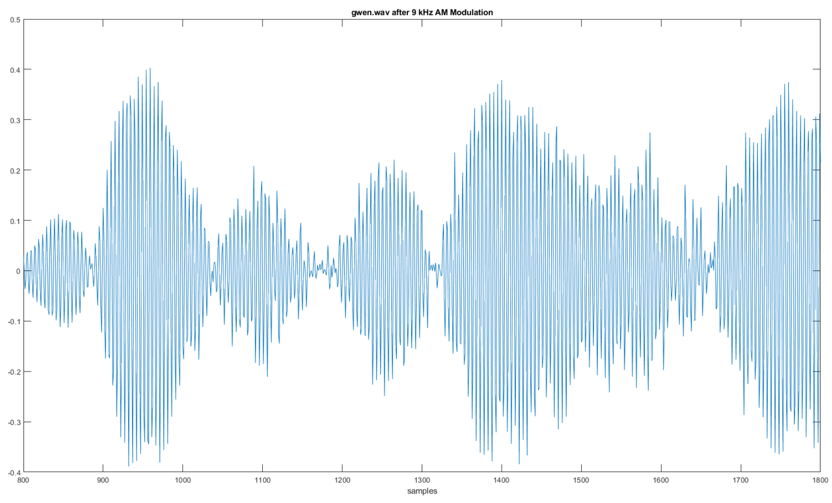

Figure 12: Time domain plot of the modulated audio, clearly showing the AM modulation of the signal.

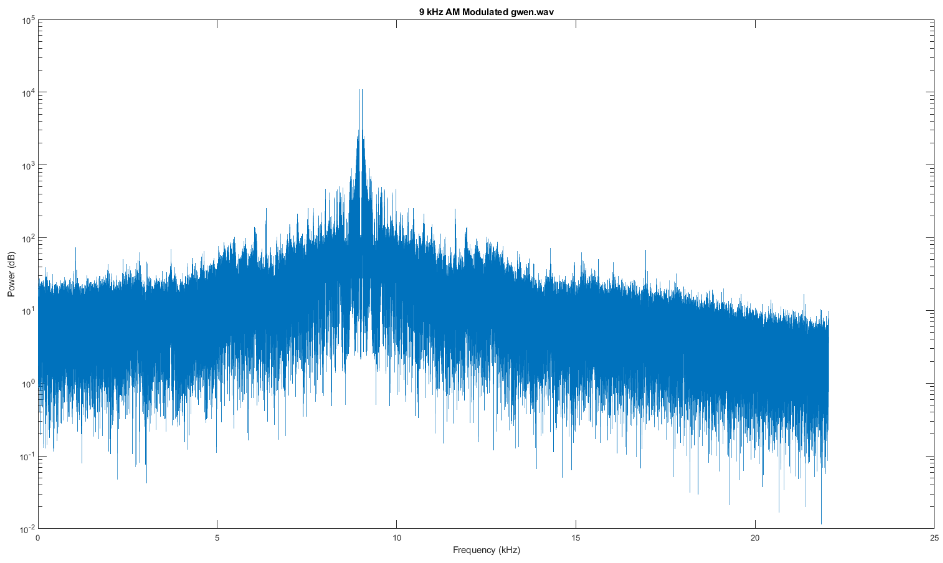

Figure 13: Frequency domain plot of the modulated audio, clearly showing the AM modulation of the signal.

Q 6.2

As can be seen below in Figure 14, after demodulation there is unwanted frequency content around $2f_c$, which in this case is 18 kHz.

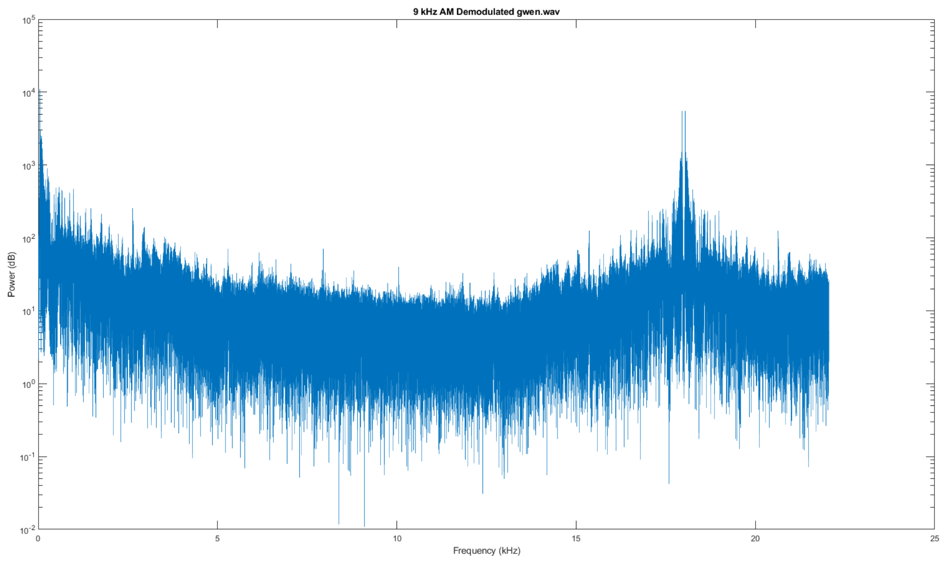

Figure 14: Frequency domain plot of the demodulated audio, with unwanted tones around 18 kHz.

A filter to remove the unwanted frequency content has been designed and will be discussed further below.

Q 6.3

The design requirements for the filter are to remove the higher frequency content whilst keeping the lower frequency content as is, along with having the transition width as short as possible, stopband attenuation greater than 50 dB, and fewer than 200 terms in the FIR filter.

Referring to the Windowed FIR LPF Design document provided, it can be seen that both Hamming and Blackman windows will meet the attenuation requirements, however the Blackman window requires twice the number of terms as Hamming and as such Hamming was chosen as it can achieve a shorter transition window with the same number of terms.

The number of terms was chosen at 199, as that is the highest odd number less than 200, and would allow the shortest transition window whilst ensuring no phase distortion to the signal. Utilising the equation which relates transition width to number of points, it can be found that for $N = 199$, $TW = 762$ Hz for a Hamming window LPF.

A Hamming filter with these properties and a 15 kHz pass band edge was then utilised in MATLAB to reduce the unwanted frequency components from the audio file, as can be seen below in §6.4.

Q 6.4

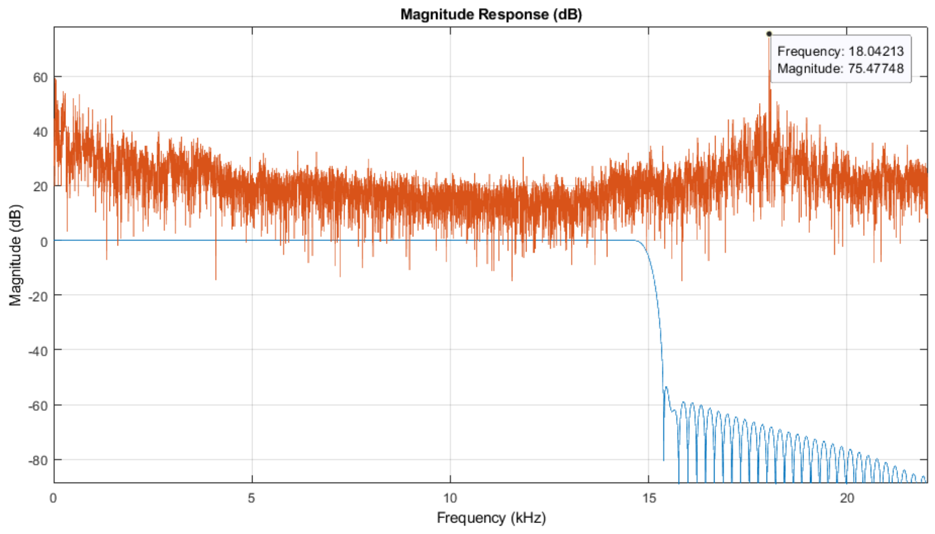

Figure 15: Frequency response of my filter, alongside the demodulated audio.

Q 6.5

Pass band edge frequency: 15 kHz
Transition width: 762 Hz
Window Type: Hamming
Number of coefficients: 199
Stop band attenuation: 55 dB

Q 6.6

Actual stop band attenuation: 62.5 dB
-3dB Bandwidth: 14.9 kHz

Q 6.7

Comparing the filtered audio in the below plot (Fig. 16) to the unfiltered audio in Fig, 15, it can be seen that the filter has reduced a significant amount of the unwanted frequency content from the signal. In particular, a 70 dB decrease in the 18 kHz tone can be seen, which makes a significant difference to the output audio, as now the unwanted high frequency tones are no longer audible.

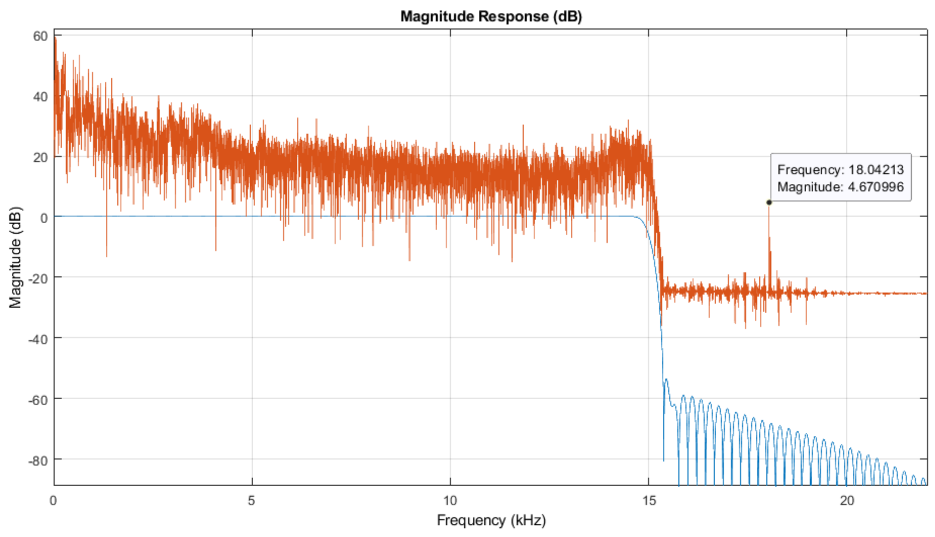

Figure 16: Frequency response of my filter, alongside the filtered audio.