CS4302 Practical 2

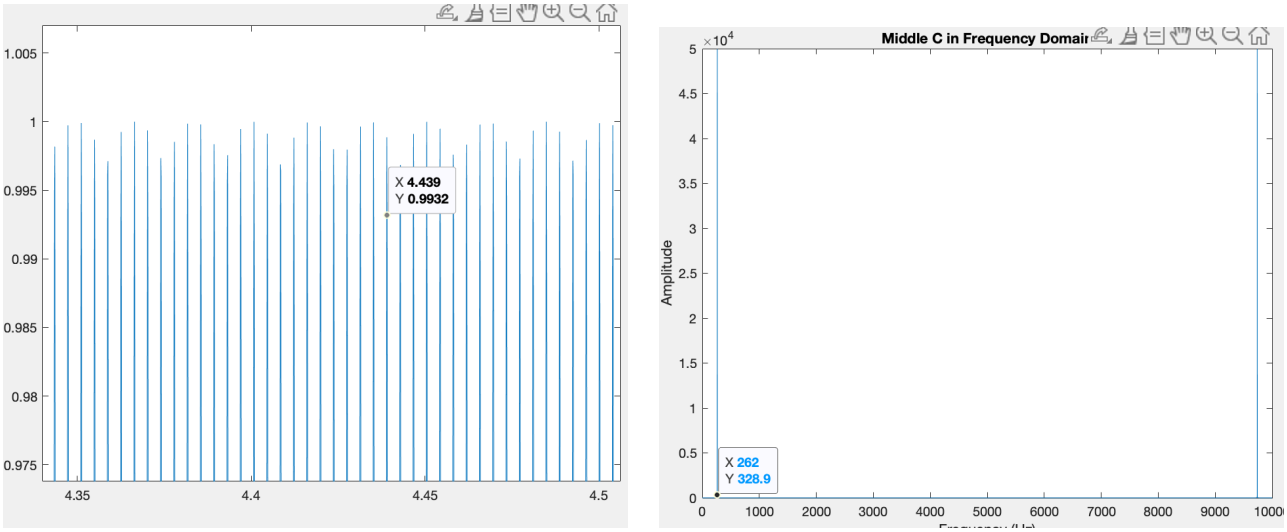
170011474

November 2020

Contents

1 Q1

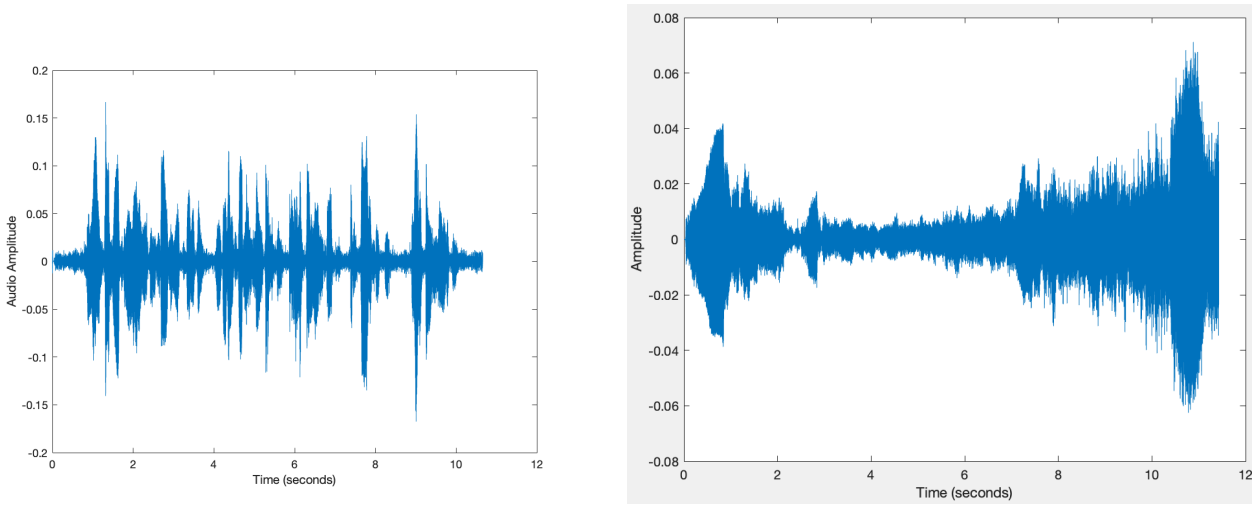
The middle C plotted in the time domain has range [1, -1], and it consists of different sinunoids (their crests and troughs are not all the same). In frequency domain, there are only 1 frequency which is about 262 Hz (same as Middle C).



(a) Time Domain (b) Frequency Domain

Figure 1: Middle C

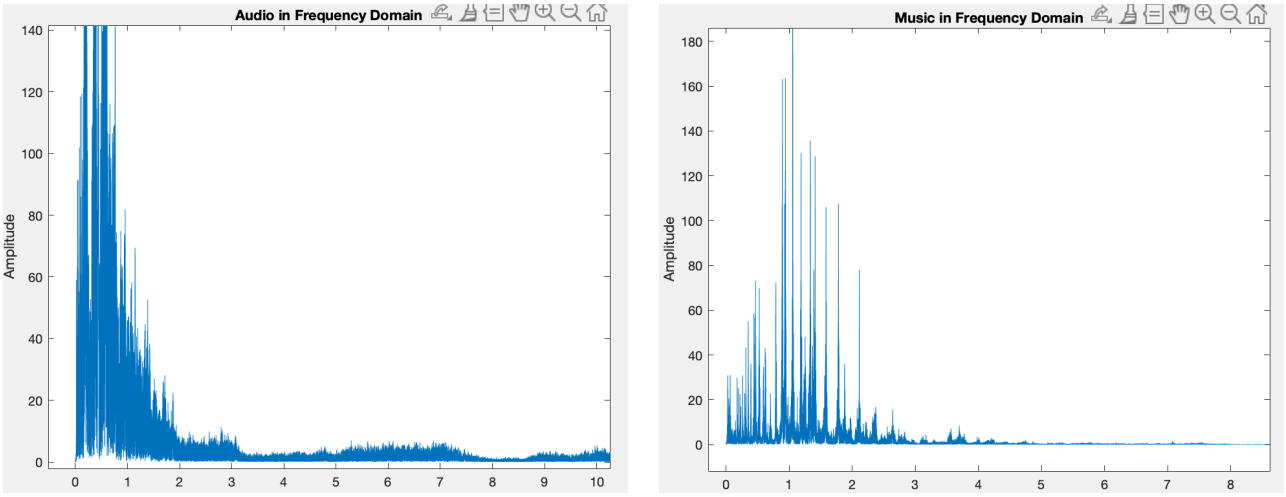
2 Q2



(a) Audio (b) Music

Figure 2: Time Domain

3 Q3

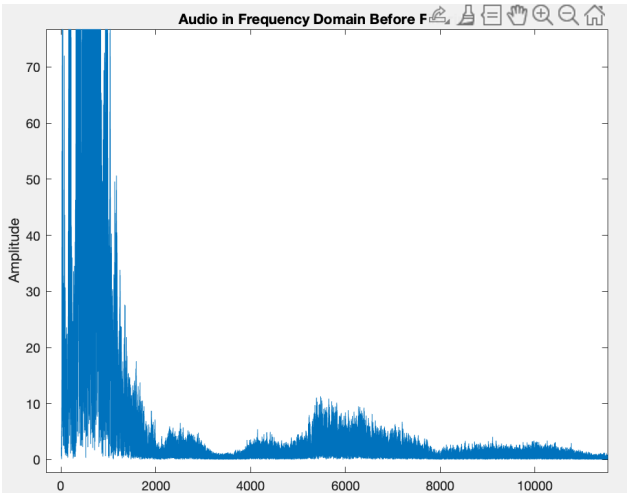


(a) Audio (b) Music

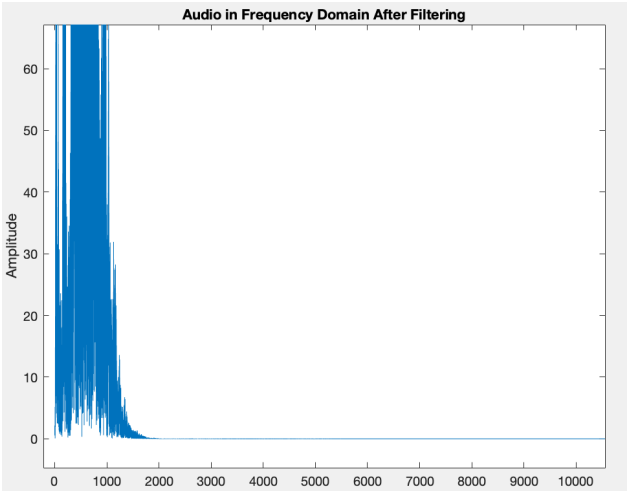
Figure 3: Frequency Domain

We can see human speech would cover more frequency bands while instrument would be more "discrete" in the frequency domain. This is probably because in `music.wav`, stringed instruments tend to have distinct and discontinuous frequencies for each string, so it would not act like human speech.

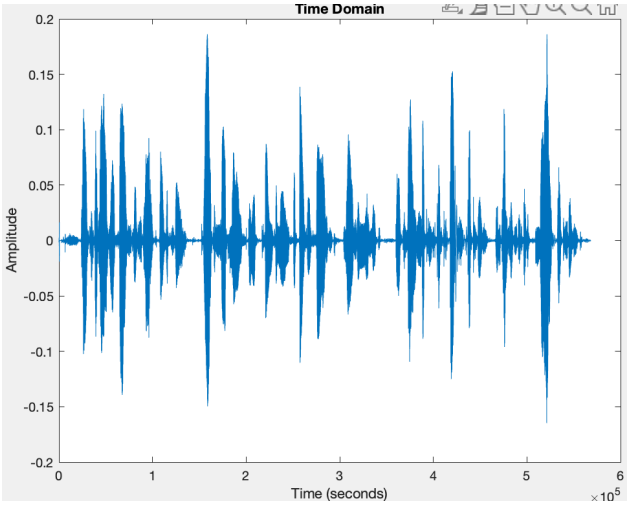
4 Q4



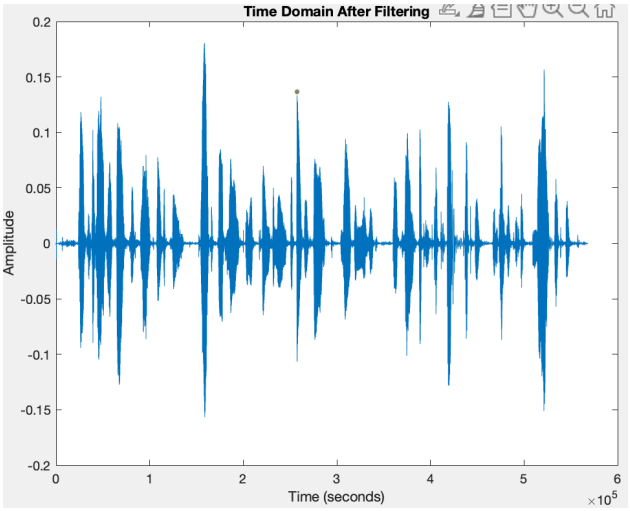
(a) Frequency Domain Before Filtering



(b) Frequency Domain After Filtering



(a) Time Domain Before Filtering

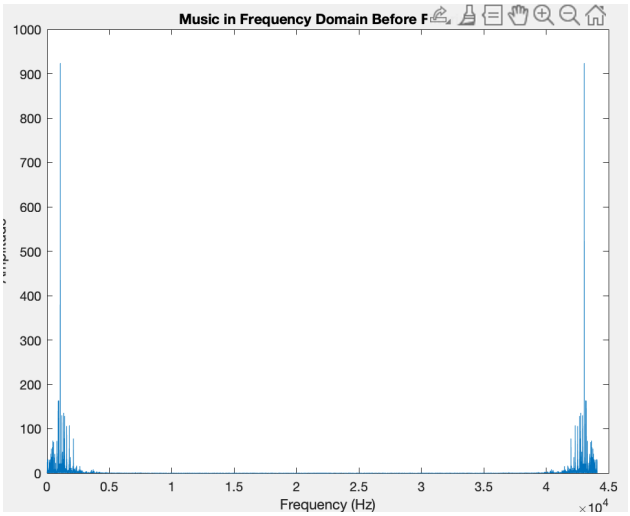


(b) Time Domain After Filtering

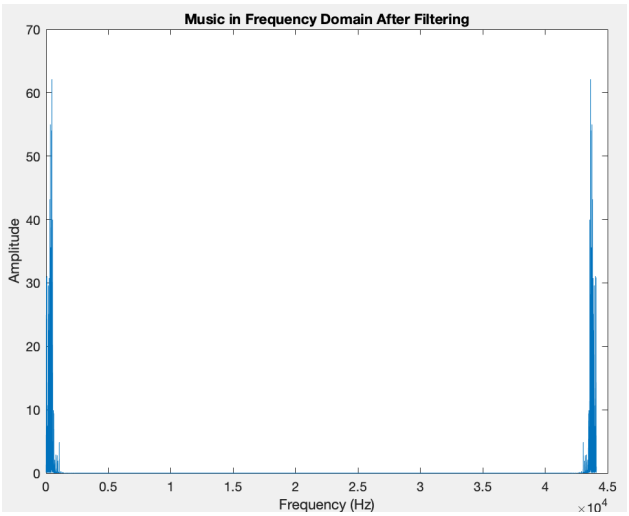
Figure 4: Audio

In `audio.wav`, I used a butterworth low pass filter with a cutoff frequency 1102.5 Hz. I also used a butterworth low pass filter with a cutoff frequency 500Hz for `music.wav`.

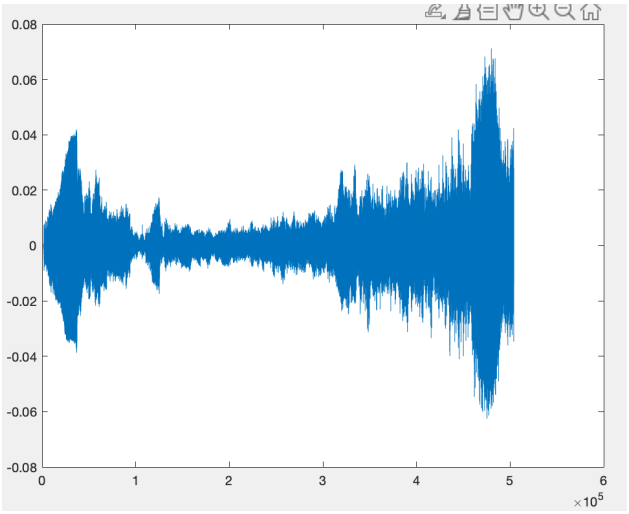
Since those filters are low-pass, so high frequency bands would be **gradually flattened** due to butterworth's properties. If the cutoff frequency is 100Hz, it does not mean that at the 100Hz frequency band, it is 0. It means the amplitude would gradually approach to 0 from 100Hz. As we can see from figures 5(a) and 5(b), its amplitude decreases from about 900 to about 60. Afterwards, higher frequency bands' values smoothly go to 0.



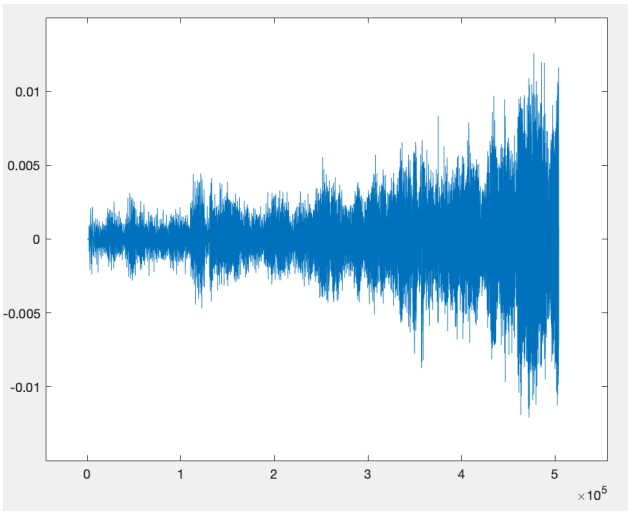
(a) Frequency Domain Before Filtering



(b) Frequency Domain After Filtering



(a) Time Domain Before Filtering

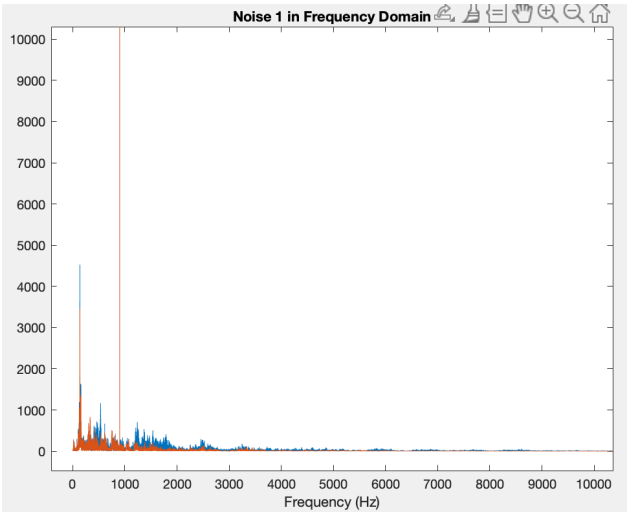


(b) Time Domain After Filtering

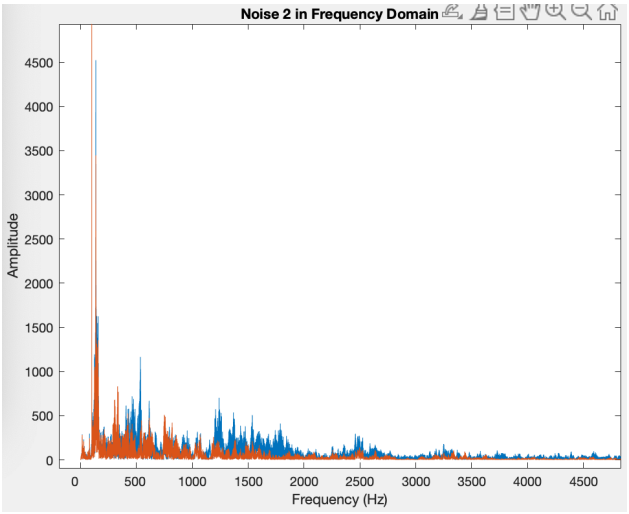
Figure 5: Music

After filtering, those audio files sound less clear and also their amplitudes are dramatically reduced.

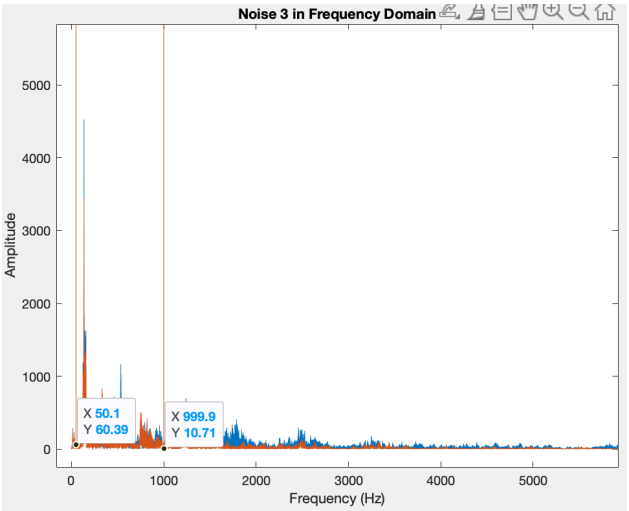
5 Q5



(a) Noise 1



(b) Noise 2

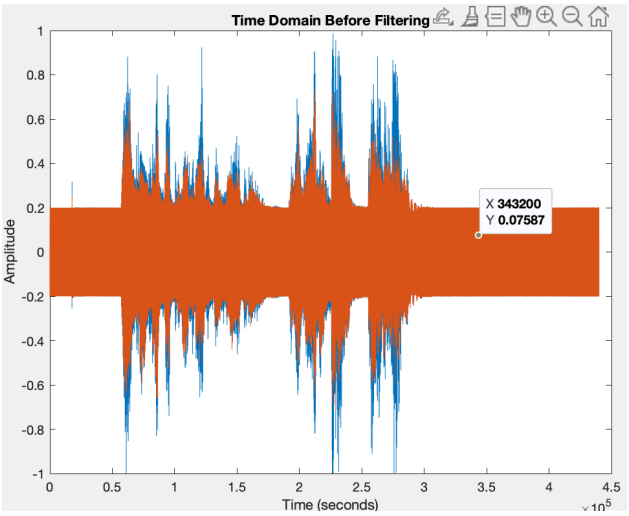


(c) Noise 3

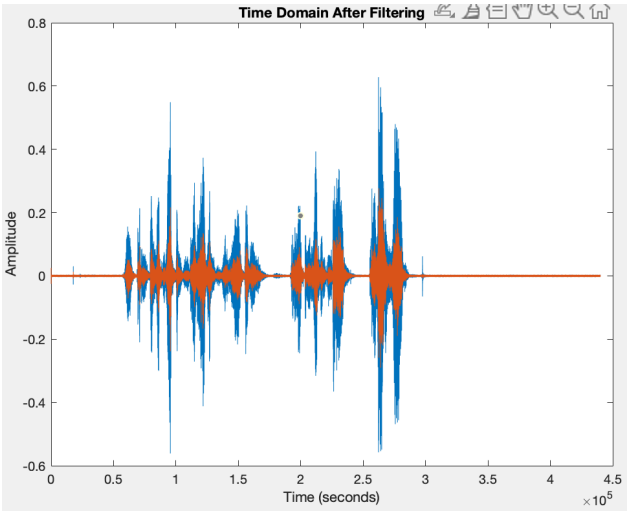
Figure 6: Frequency Domain

How to notice those noises' frequencies? Since the human speeches inside them are the same, only noise types make them different. From those figures, we can know the only variations show in 100Hz in Noise 2, 900Hz in Noise 1, and 50Hz and 1000Hz in Noise 3.

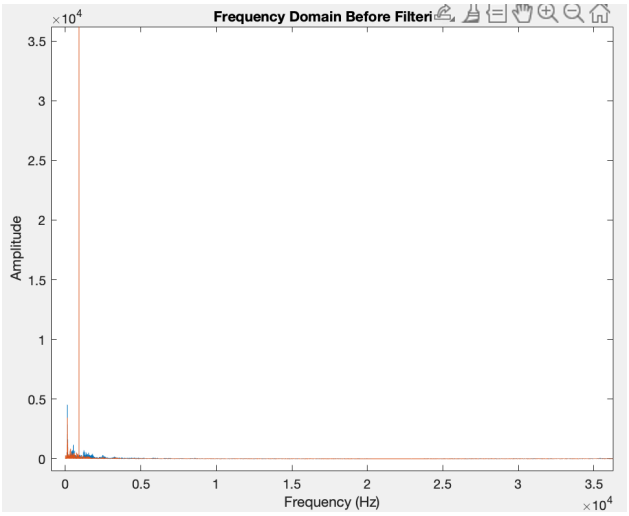
6 Q6



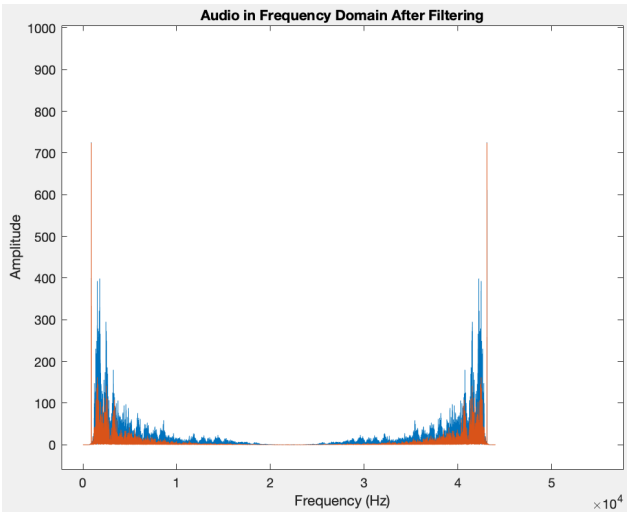
(a) Time Domain Before Filtering



(b) Time Domain After Filtering



(a) Frequency Domain Before Filtering

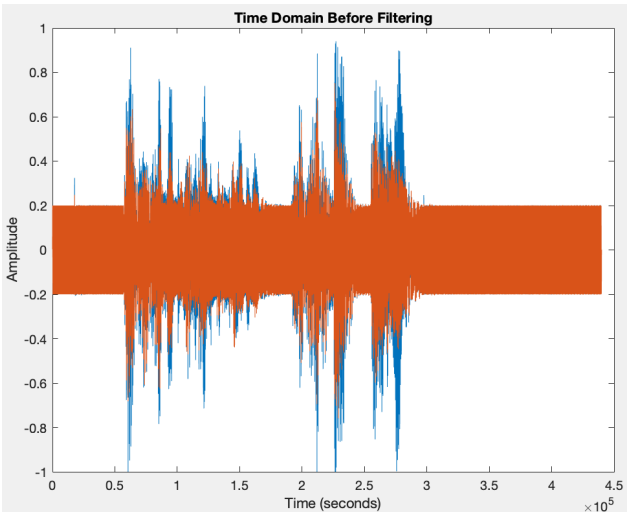


(b) Frequency Domain After Filtering

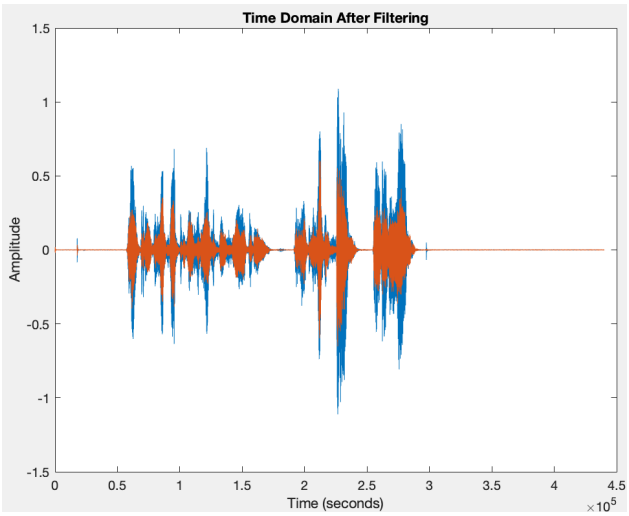
Figure 7: Noise 1

To remove noises, the process is: Firstly, for each file, read audio. Then, do `fft` to get its spectrum. Thirdly, construct a butterworth high-pass filter to remove noises. Fourthly, plot the new spectrum and then save the new audio into a new file.

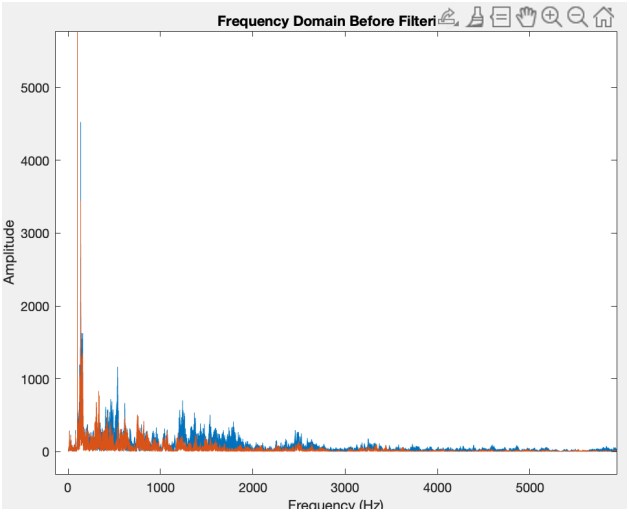
I used butterworth high-pass filters. Setting cutoff frequencies to 100Hz or 900Hz would not entirely remove those noises due to butterworth's properties (illustrated above in Q4). In noise 1, I used a cutoff frequency 2000Hz. In noise 2, I used a cutoff frequency 200Hz. In noise 3, I used 2200Hz as the cutoff frequency. Those parameters were found by making ex-



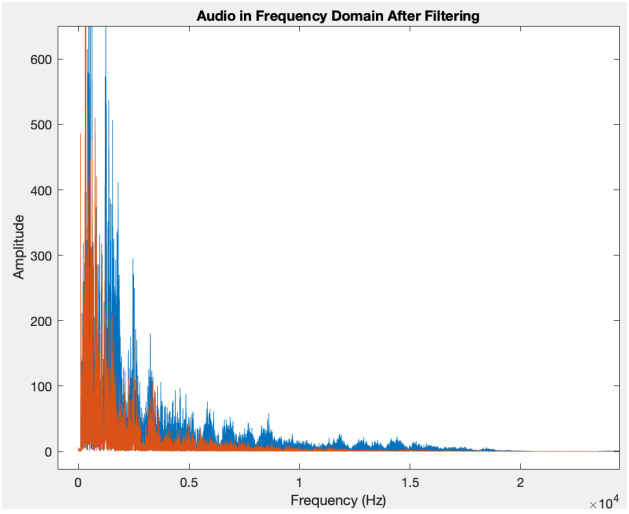
(a) Time Domain Before Filtering



(b) Time Domain After Filtering



(a) Frequency Domain Before Filtering

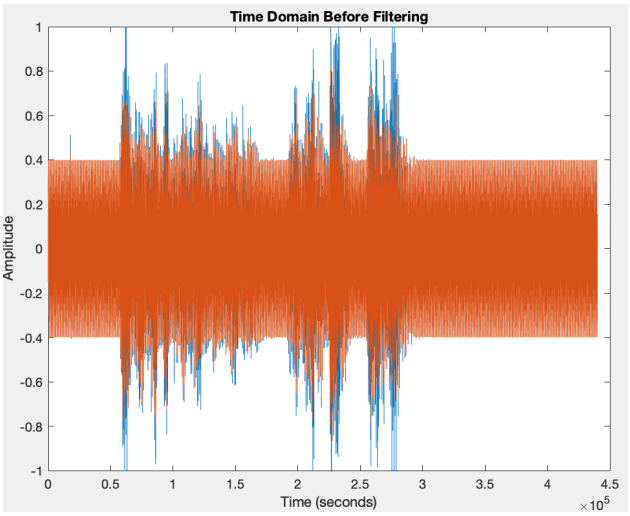


(b) Frequency Domain After Filtering

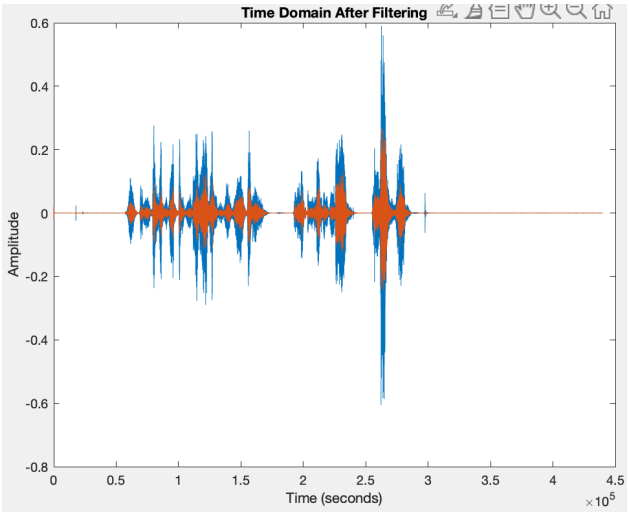
Figure 8: Noise 2

periments. Only in this way, the corresponding noise frequency would be reduced to about 0.

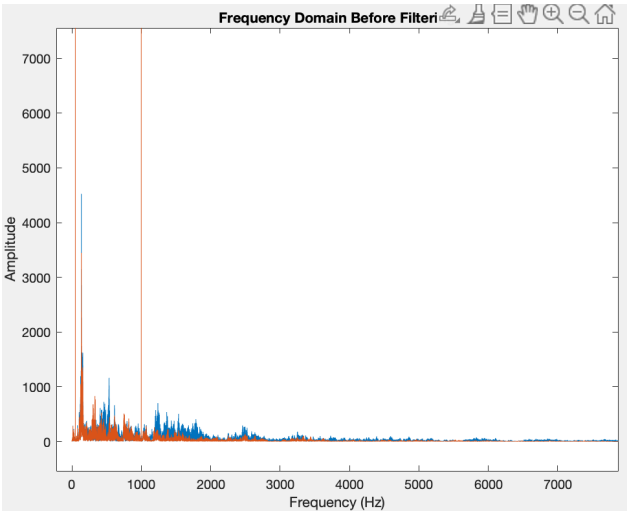
After filtering, those audio files tend to have less amplitudes and less clear sound.
In `audio_in_noise1.wav` and `audio_in_noise3.wav`, the human speech is less clear than before. In `audio_in_noise2.wav`, since the noise is in a lower frequency band, so a cut-off frequency 100Hz would not affect the original audio clip that much. Therefore, in `audio_in_noise2.wav`, the human speech is still very clear.



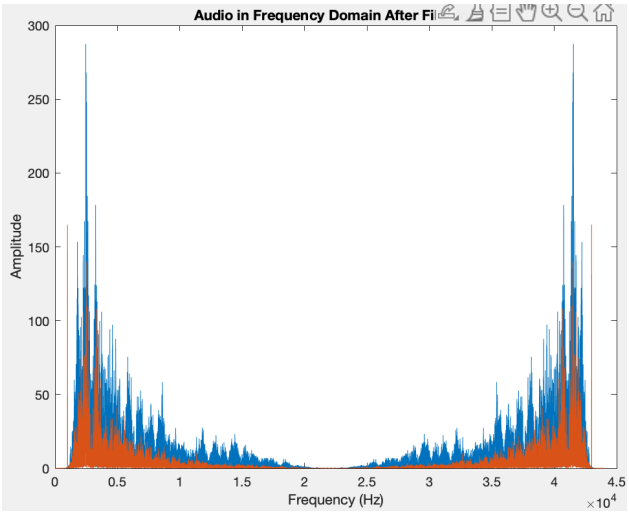
(a) Time Domain Before Filtering



(b) Time Domain After Filtering



(a) Frequency Domain Before Filtering



(b) Frequency Domain After Filtering

Figure 9: Noise 3

7 Extension

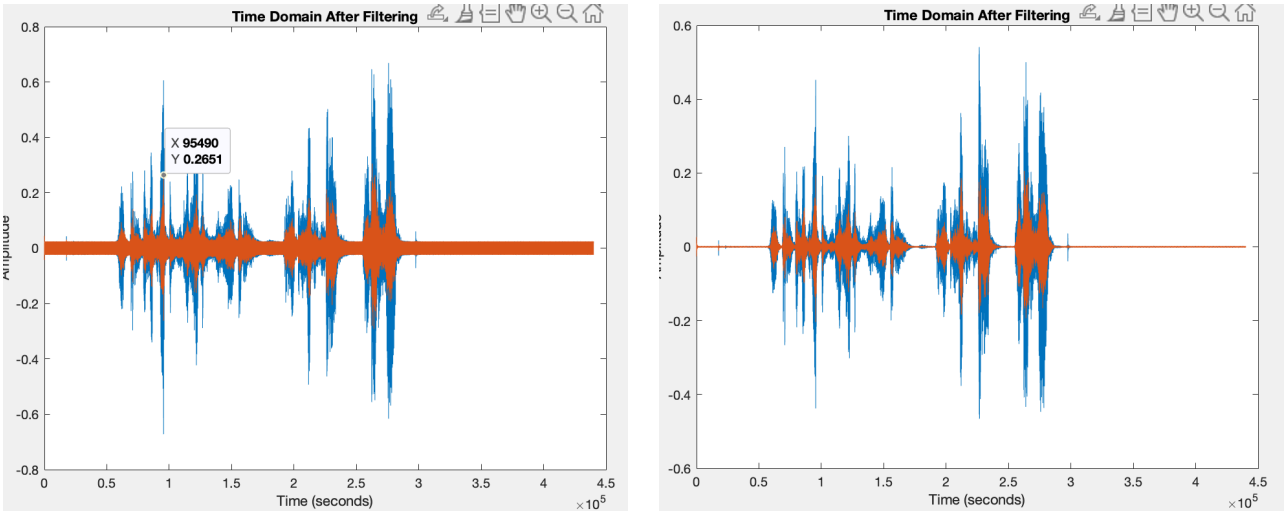
In extension, I would compare different filtering techniques. Here we use `audio_in_noise3.wav` as an example.

Cheby1

```
[b,a] = cheby1(8,5,1300/Fn,'high');
```

Cheby1 filter is Compared to `butter`, cheby1 would perform better. When the ripple is set

to 0%, the filter is called a maximally flat or Butterworth filter. As we can see, if we set

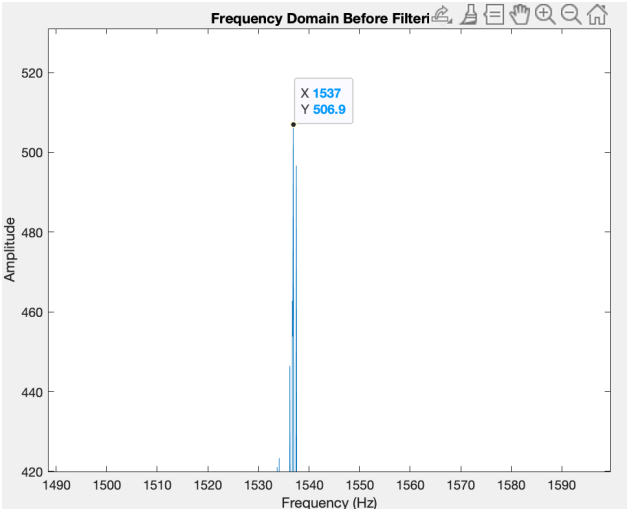


(a) butter (b) cheby1

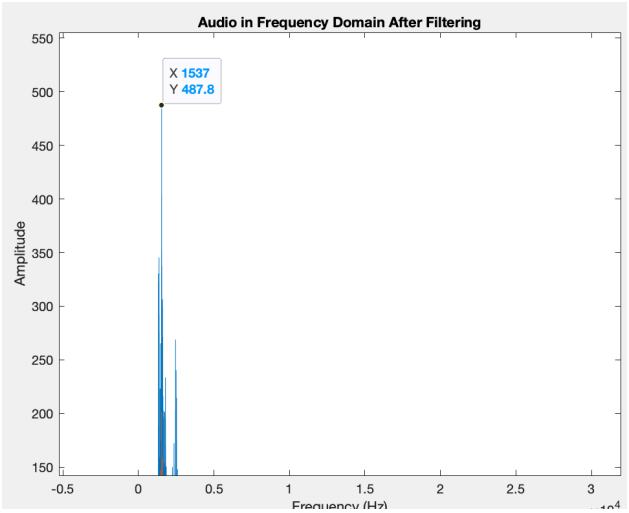
Figure 10: Cutoff Frequency = 1300Hz

the cutoff frequency as 1300Hz, then butter filter still results in a slight noise in the time domain, while cheby1 would remove almost all noise.

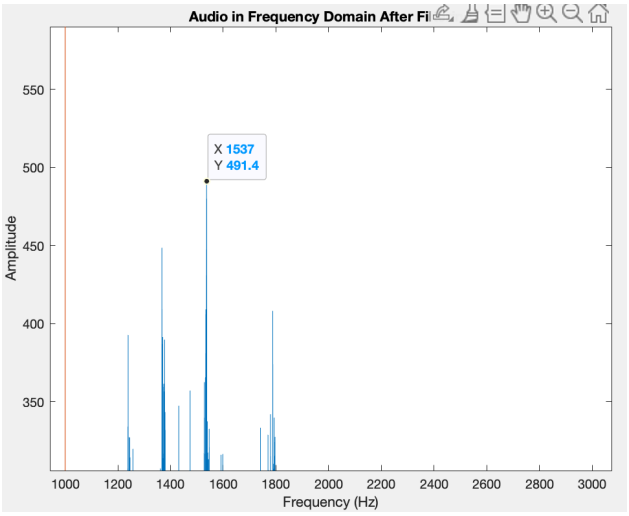
The reason for that is Cheby1 filter would bring a faster decaying in transition band compared to the original butter filter. However, it would result in amplitude variation (not that stable as butter filters) in the pass bands. For example, in following figures, when $x = 1537$ in the pass band, the expected amplitude should be 506.9, after filtering by butter, the amplitude changes to 491.4, while after filtering by cheby1, the amplitude changes to 487.8. In this case, we can roughly know that there is a trade-off. Cheby1 can be faster and more accurate to reduce noisy frequency bands, but it also can be a little more unstable.



(a) Before Filtering



(b) After Cheby1 Filtering



(c) After Butter Filtering