Filter Design Project

Armaan Nalli

May 6th 2025

1 Introduction

In this project, I designed a digital Butterworth low-pass IIR filter. The design was based on spectral analysis of the noisy audio signal, which revealed speech content primarily below 2 kHz and unwanted noise above 2.5 kHz. I derived appropriate passband and stopband specifications, determined the required filter order and analog cutoff frequency, and then converted the analog prototype to a digital filter. The filter was implemented in MATLAB using the butter() and filter() functions. The frequency response of the filtered signal confirmed that the high-frequency noise was significantly attenuated while the speech frequencies were preserved.A comparison of DFT plots before and after filtering and audio tests further validated the effectiveness of the design.

2 Design Summary

2.1 Audio Frequency Analysis

- I used MATLAB's `audioread()` function to analyze the given noisy audio file instead of my DFT implementation in Python. (see Note at the end of the document)
- This provided the audio signal array and the sampling frequency.
  - Sampling frequency(Fs) = 11025 Hz
- I computed the Discrete Fourier Transform (DFT) using the `fft()` function.
- I plotted the magnitude spectrum of the Noisy audio (in dB)
  - The plot showed speech energy in the frequency range of approximately 100 Hz–2 kHz.
  - The plot showed noise energy in the frequency range of approximately 2.5 kHz–5.5 kHz.

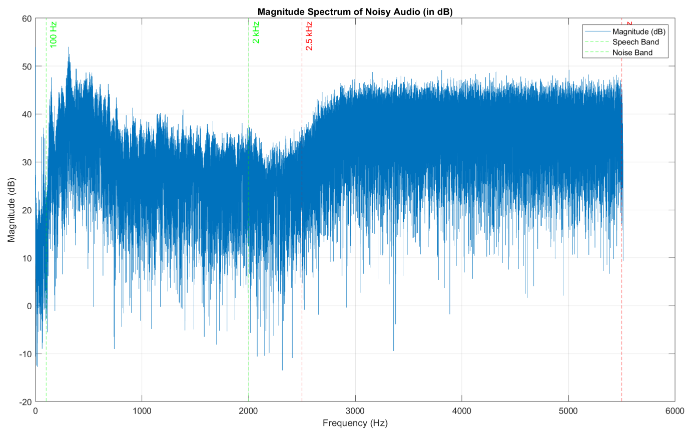

Figure 1: Magnitude Spectrum of Noisy Audio (in dB)

2.2 Filter Design

- To preserve speech and attenuate high-frequency noise, I specified a digital Butterworth low-pass filter.
  - Passband edge frequency: $f_p = 2000$ Hz
  - Stopband edge frequency: $f_s = 2500$ Hz
- Using the sampling frequency $F_s = 11025$ Hz, I calculated the digital radian frequencies:
  - $\omega_p = \dfrac{2\pi f_p}{F_s} \approx 1.1391$ rad/sample
  - $\omega_s = \dfrac{2\pi f_s}{F_s} \approx 1.4239$ rad/sample
- I selected the following attenuation specifications:
  - Passband attenuation ≤ 1 dB: $\delta_p = 10^{-1/20} \approx 0.8913$
  - Stopband attenuation ≥ 50 dB: $\delta_s = 10^{-50/20} \approx 0.0032$
- I used the impulse invariance method to map digital to analog frequencies:
  - Sampling period $T = \dfrac{1}{F_s}$
  - $\Omega_p = \dfrac{\omega_p}{T} = \omega_p \cdot F_s \approx 12551.6$ rad/s
  - $\Omega_s = \dfrac{\omega_s}{T} = \omega_s \cdot F_s \approx 15701.9$ rad/s
- I computed the Butterworth filter parameters:

- $k_p = \frac{1}{\delta_p^2} - 1 \approx 0.2589$
- $k_s = \frac{1}{\delta_s^2} - 1 \approx 97656.25$
- Filter order:

$$N = \left\lceil \frac{1}{2} \cdot \frac{\log_{10}(k_s/k_p)}{\log_{10}(\Omega_s/\Omega_p)} \right\rceil = \lceil 29 \rceil$$

- I calculated the analog cutoff frequency Ω_c and corresponding digital cutoff ω_c:
  - $\Omega_c = \frac{\Omega_p}{k_p^{1/(2N)}} \approx 12862.56$ rad/s
  - $\omega_c = \Omega_c \cdot T \approx 1.1667$ rad/sample
- I plotted the logarithmic gain of the analog Butterworth filter using the equation:

$$|H_a(j\Omega)| = -10\log_{10}\left(1 + \left(\frac{\Omega}{\Omega_c}\right)^{2N}\right)$$

Note: The MATLAB code prints the filter order N, analog cutoff frequency Ω_c, and digital cutoff frequency ω_c. Other intermediate values (e.g., ω_p, Ω_p, δ_p) are computed but not printed.

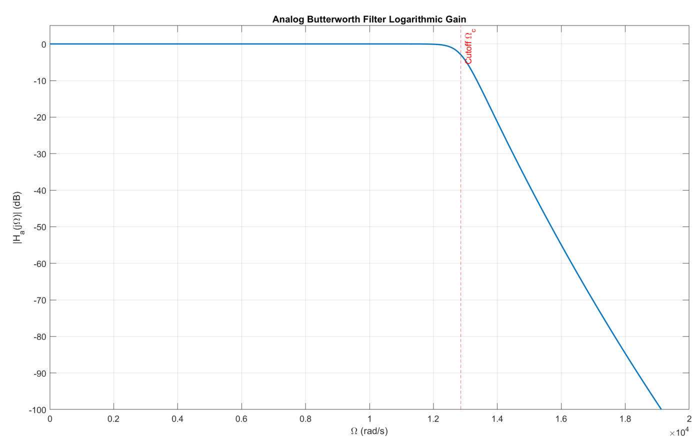

Figure 2: Analog Butterworth Filter Logarithmic Gain

2.3 Filter Implementation

- I implemented the digital Butterworth low-pass filter in MATLAB using the calculated cutoff frequency.

- The normalized cutoff frequency was computed as:

$$W_n = \frac{\omega_c}{\pi} \approx \frac{1.1667}{\pi} \approx 0.3713$$

- I used the MATLAB `butter()` function to compute the filter coefficients:
  - `[b, a] = butter(N, Wn);`
- I applied the filter using:
  - `filter_signal = filter(b, a, signal);`
- I computed and plotted the DFT of the filtered signal using the same procedure as in the initial analysis.
- I compared the magnitude spectra of the unfiltered and filtered audio to evaluate the filter performance.
  - Figure 3 shows the magnitude spectrum before filtering
  - Figure 4 shows the magnitude spectrum after filtering
  - Below 2 kHz: Speech content is preserved.
  - Above 2.5 kHz: Noise is significantly attenuated

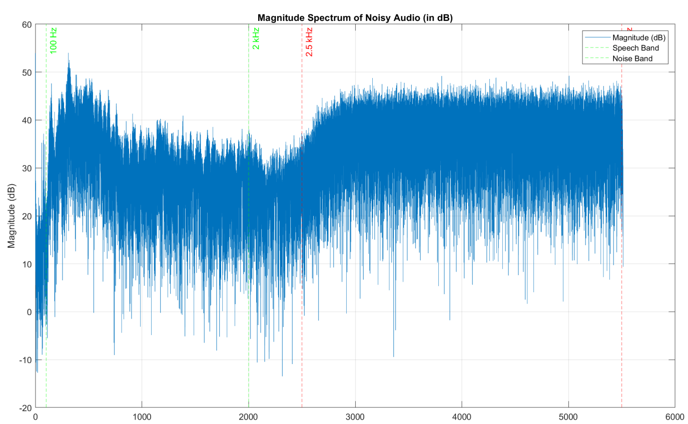

Figure 3: Magnitude Spectrum of Noisy Audio (in dB)

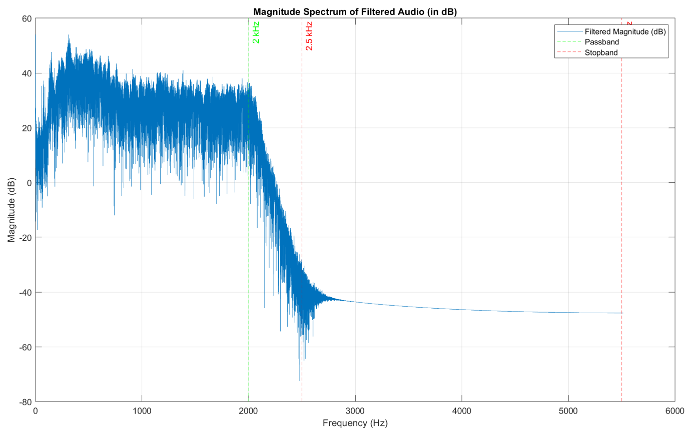

Figure 4: Magnitude Spectrum of Filtered Audio (in dB)

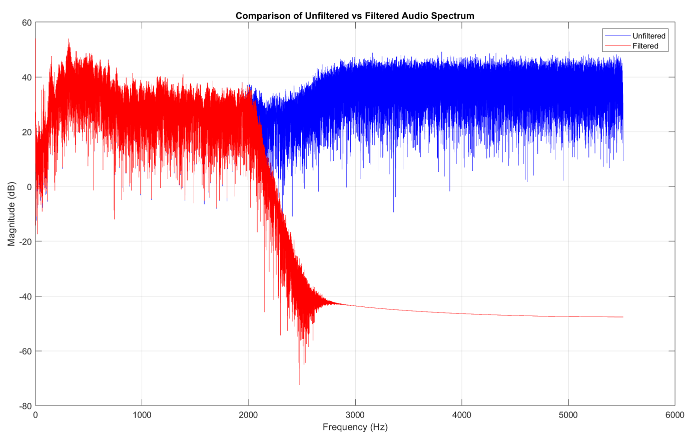

Figure 5: Comparison of Unfiltered vs Filtered Audio Spectrum

- I also listened to the original and filtered audio using MATLAB's `sound()` function:
  - `sound(signal, Fs);`
  - `pause(length(signal)/Fs + 1);`
  - `sound(filter_signal, Fs);`
- The original signal had noticeable high-frequency noise.
- The filtered signal had significantly reduced noise and clearer speech.

- I saved the final filtered output using:
  - `audiowrite('filteredaudio.wav', filter_signal, Fs);`

3 Conclusion

In this project, I successfully designed and implemented a digital Butterworth low-pass IIR filter using the impulse invariance method. The filter effectively preserved the speech content below 2 kHz while significantly attenuating high-frequency noise above 2.5 kHz. Frequency-domain analysis and audio playback confirmed the filter's performance, demonstrating the practical application of lecture concepts in real-world signal processing.

Note

I was required to use my DFT implementation to perform the analysis of the audio in step 1. However, I did my implementation in Python, and I was unable to use it in this project due to compatibility issues with the provided audio file `noisyaudio.wav`. Both the `soundfile` and `scipy.io.wavfile` libraries failed to read the file, reporting unrecognized or unsupported formats. As shown in the error messages, the file could not be interpreted as a valid RIFF or RIFX WAV format.That is why I performed all DFT computations and filtering entirely in MATLAB, which handled the audio input correctly without errors.

```
PS C:\Users\armaa\Documents\SPRING 2025\Signal Processing\Filter Design Project> python ./filter_design_dft.py
Traceback (most recent call last):
  File "./filter_design_dft.py", line 7, in <module>
    signal, Fs = sf.read('noisyaudio.wav')
  File "C:\Users\armaa\anaconda3\lib\site-packages\soundfile.py", line 282, in read
    with SoundFile(file, 'r', samplerate, channels,
  File "C:\Users\armaa\anaconda3\lib\site-packages\soundfile.py", line 655, in __init__
    self._file = self._open(file, mode_int, closefd)
  File "C:\Users\armaa\anaconda3\lib\site-packages\soundfile.py", line 1213, in _open
    raise LibsndfileError(err, prefix="Error opening {0!r}: ".format(self.name))
soundfile.LibsndfileError: Error opening 'noisyaudio.wav': Format not recognised.
PS C:\Users\armaa\Documents\SPRING 2025\Signal Processing\Filter Design Project>
```

Figure 6: Python `soundfile` error: format not recognised.

```
PS C:\Users\armaa\Documents\SPRING 2025\Signal Processing\Filter Design Project> python ./filter_design_dft.py
Traceback (most recent call last):
  File "./filter_design_dft.py", line 7, in <module>
    Fs, signal = wavfile.read('noisyaudio.wav')
  File "C:\Users\armaa\anaconda3\lib\site-packages\scipy\io\wavfile.py", line 650, in read
    file_size, is_big_endian = _read_riff_chunk(fid)
  File "C:\Users\armaa\anaconda3\lib\site-packages\scipy\io\wavfile.py", line 521, in _read_riff_chunk
    raise ValueError(f"File format {repr(str1)} not understood. Only "
ValueError: File format b'\x00\x00\x00\x1c' not understood. Only 'RIFF' and 'RIFX' supported.
```

Figure 7: Python `scipy.io.wavfile` error: unsupported file format.