FREQUENCY SPECTRUM ANALYSIS (FFT) IN THE MODULAR HEARING AID PROJECT

I. Objectives

The goal of this individual work section in the project is to analyze the frequency spectrum of the input audio and the audio processed through the WDRC compressor. Based on the spectrum, the team can assess how well the speech frequency band (300–3400 Hz) is preserved during processing.

The results of the spectral analysis will assist in making technical decisions regarding filtering, compression, and output optimization, especially when comparing with other popular compression codecs such as MP3. It also helps identify unwanted noise frequencies, thereby improving the output quality of the hearing aid.

II. Tools Used

Programming Language and Software

- **MATLAB**: Used for reading audio files, displaying waveforms and frequency spectra, performing signal processing, and evaluating metrics such as PSNR and SNR.

Why MATLAB?

- **Powerful libraries and functions**: MATLAB offers optimized signal processing functions like `fft`, `audioread`, and `spectrogram`, enabling fast and accurate analysis.

- **Excellent data visualization**: Tools like `plot`, `spectrogram`, and `imagesc` provide clear and professional visual representations of audio signals in both time and frequency domains.

- **Signal analysis-oriented**: Designed specifically for engineering applications, MATLAB is more suitable than Python or C++ for quick and precise manipulation of digital signals.

III. Processing Workflow

1. Reading the Audio File

- **Format**: The audio files use the `.wav` format, which is lossless and retains the original audio data.
- **Reason**: `.wav` ensures that the frequency spectrum accurately reflects the signal's real energy, unaffected by compression algorithms like MP3.
- Use `audioread()` to load the audio file into MATLAB.

```
[x, Fs] = audioread('filename.wav');
```

2. Displaying the Waveform

- **Method**: Plot the signal amplitude over time to visualize intensity and timing.
- **Reason**: Allows a preliminary check of the recording's length, silence intervals, or distortions.

```
t = (0:length(x)-1)/Fs;
plot(t, x);
xlabel('Time (s)');
ylabel('Amplitude');
title('Waveform');
```

3. Frequency Spectrum Analysis (FFT)

- Use the **Fast Fourier Transform (FFT)** to convert the signal from the time domain to the frequency domain.

```
N = length(x);
X = fft(x);
```

```
f = Fs*(0:N/2-1)/N;
magnitude = abs(X(1:N/2));
plot(f, magnitude);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Frequency Spectrum');
```

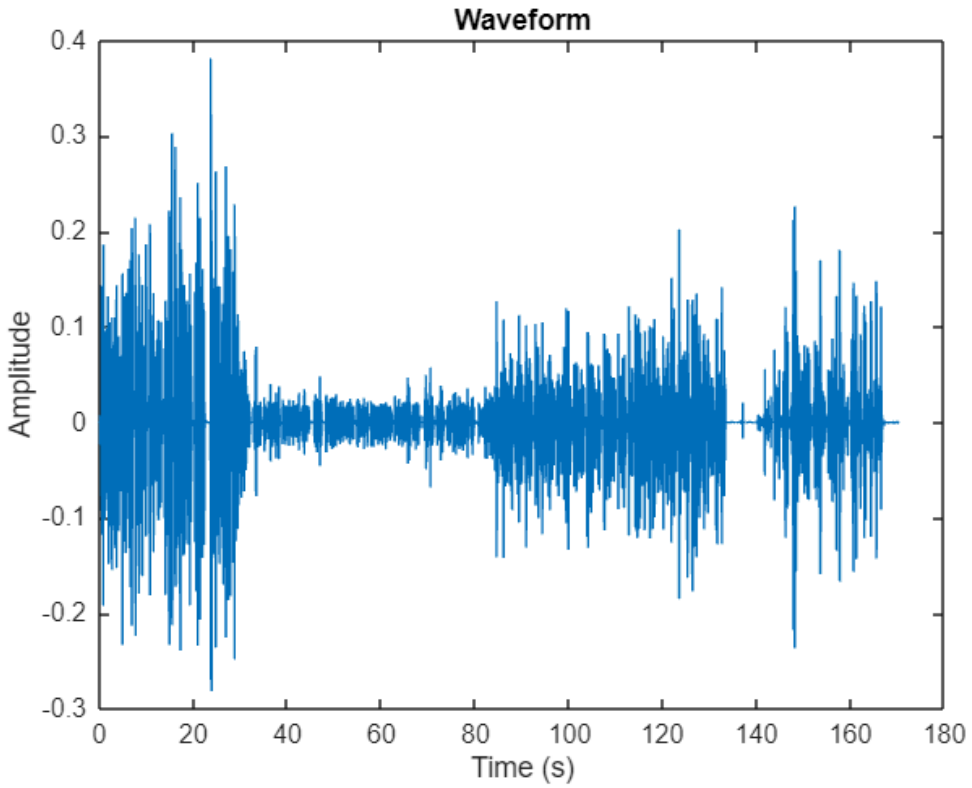
4. Extracting the Important Frequency Band

- **Reason**: Hearing aids focus on the 300–3400 Hz range to improve speech intelligibility.
- **Application**: This range is preserved during filtering and prioritized during compression.
- **How to identify**: Observe the frequency spectrum and measure energy concentration in this band.

```
speech_band = (f >= 300 & f <= 3400);
plot(f(speech_band), magnitude(speech_band));
title('Speech Band (300-3400 Hz)');
```

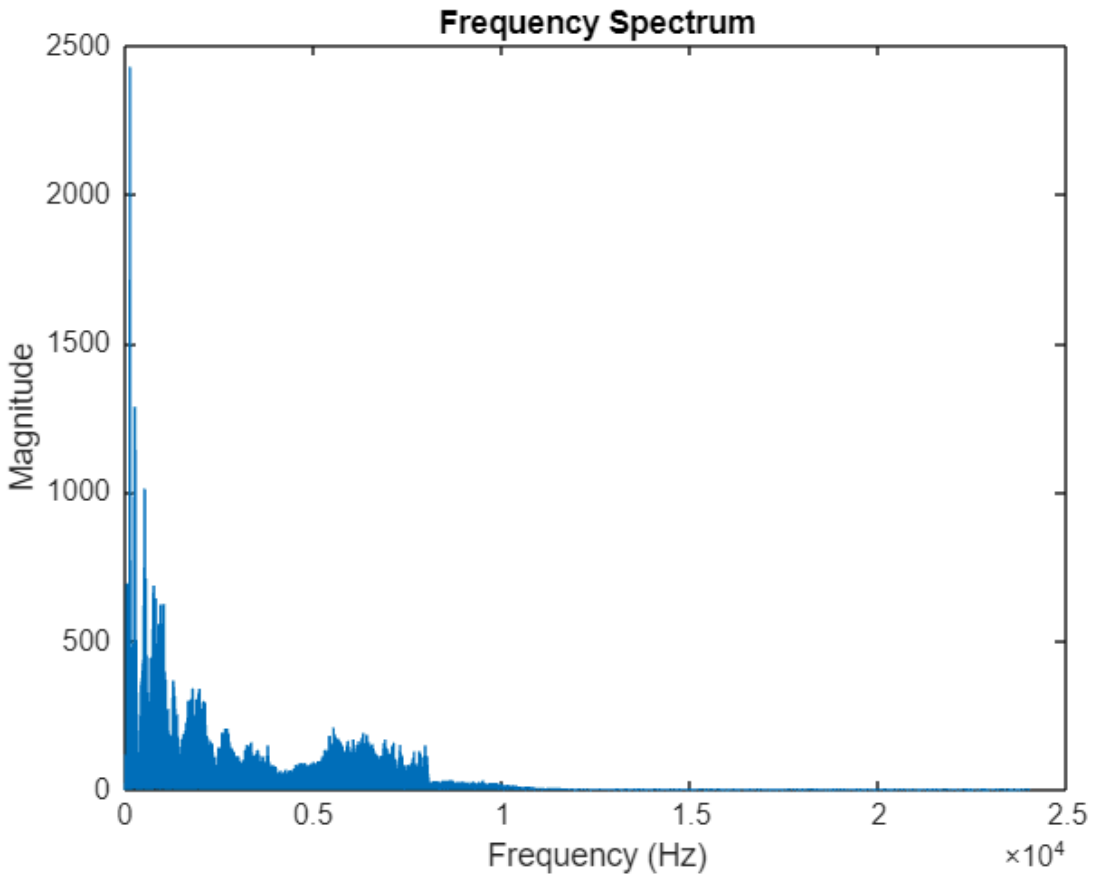
IV. Results

Original Audio File



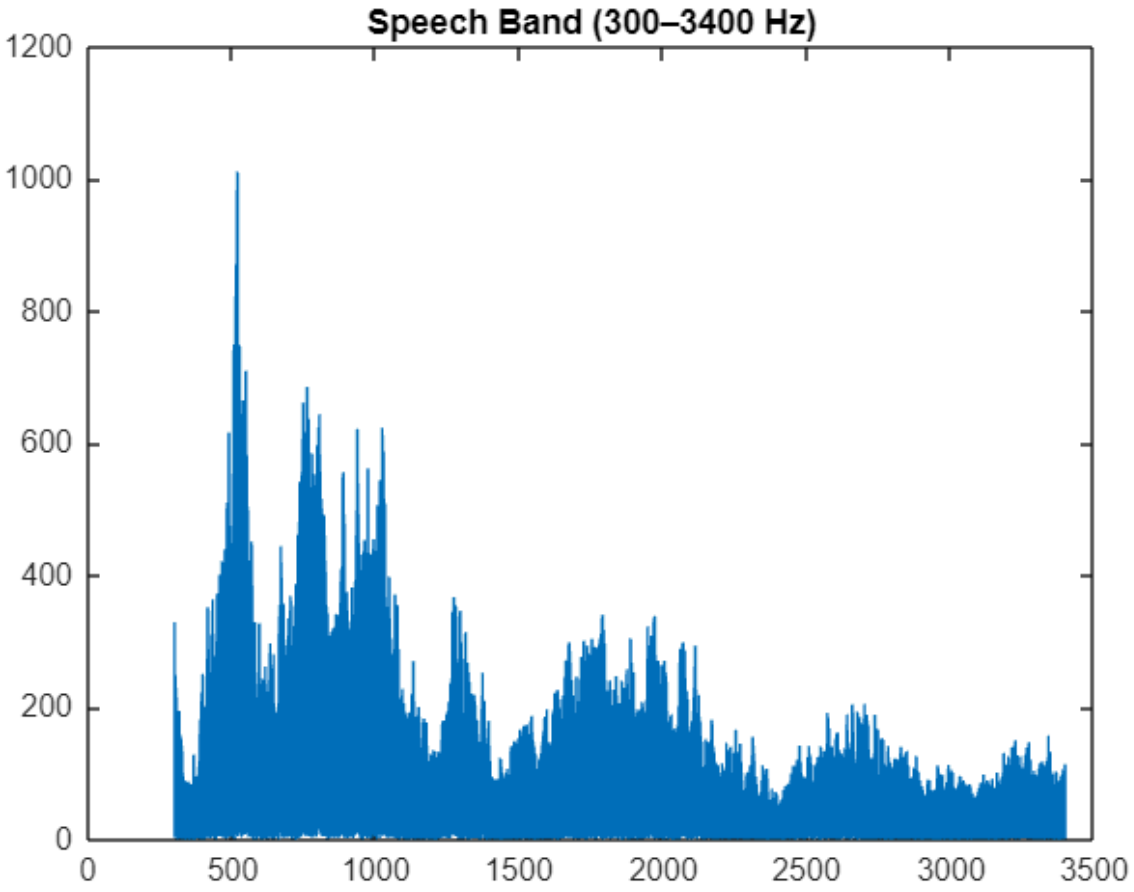
- **Waveform**: The amplitude varies clearly throughout the recording. Strongest intensities appear at the beginning (0–40 sec) and end (110–160 sec), while the middle (40–110 sec) is quieter — reflecting natural speech

patterns with pauses.



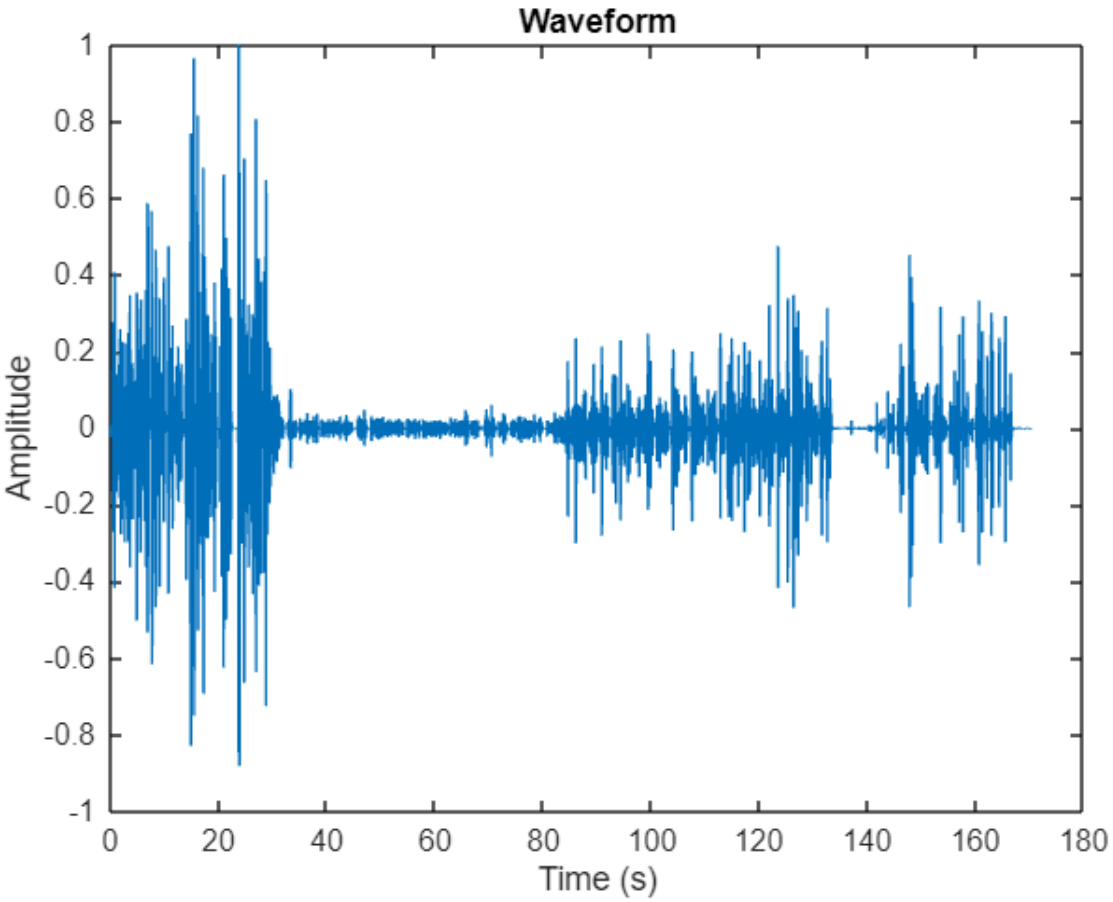
- **Full Frequency Spectrum**: Energy is concentrated below 5 kHz, especially below 1 kHz — typical of human speech, which has strong

low-frequency components.

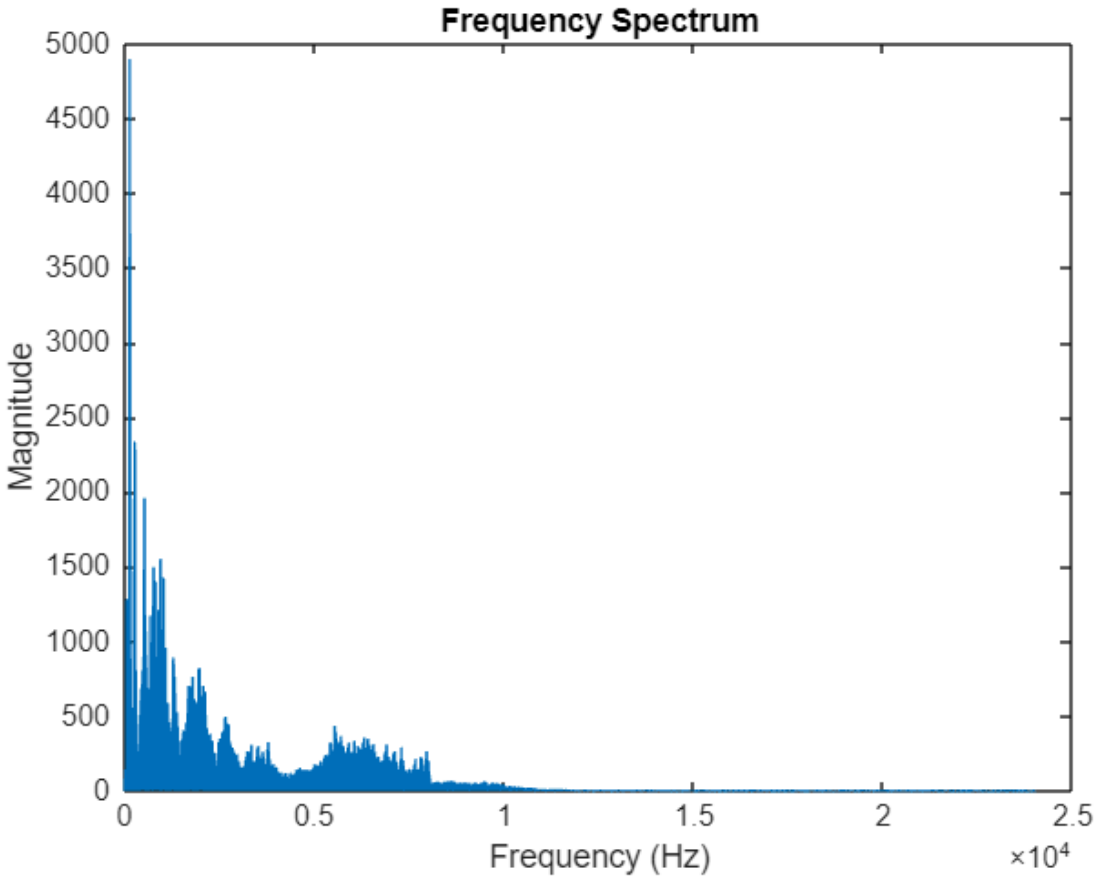


- **Speech Band Spectrum**: Even energy distribution with clear peaks around 400 Hz, 1000 Hz, and 1500 Hz — essential for speech sounds and associated with vocal tract resonances (formants). No information loss in mid frequencies, indicating high-quality recording.

Compressed Audio File

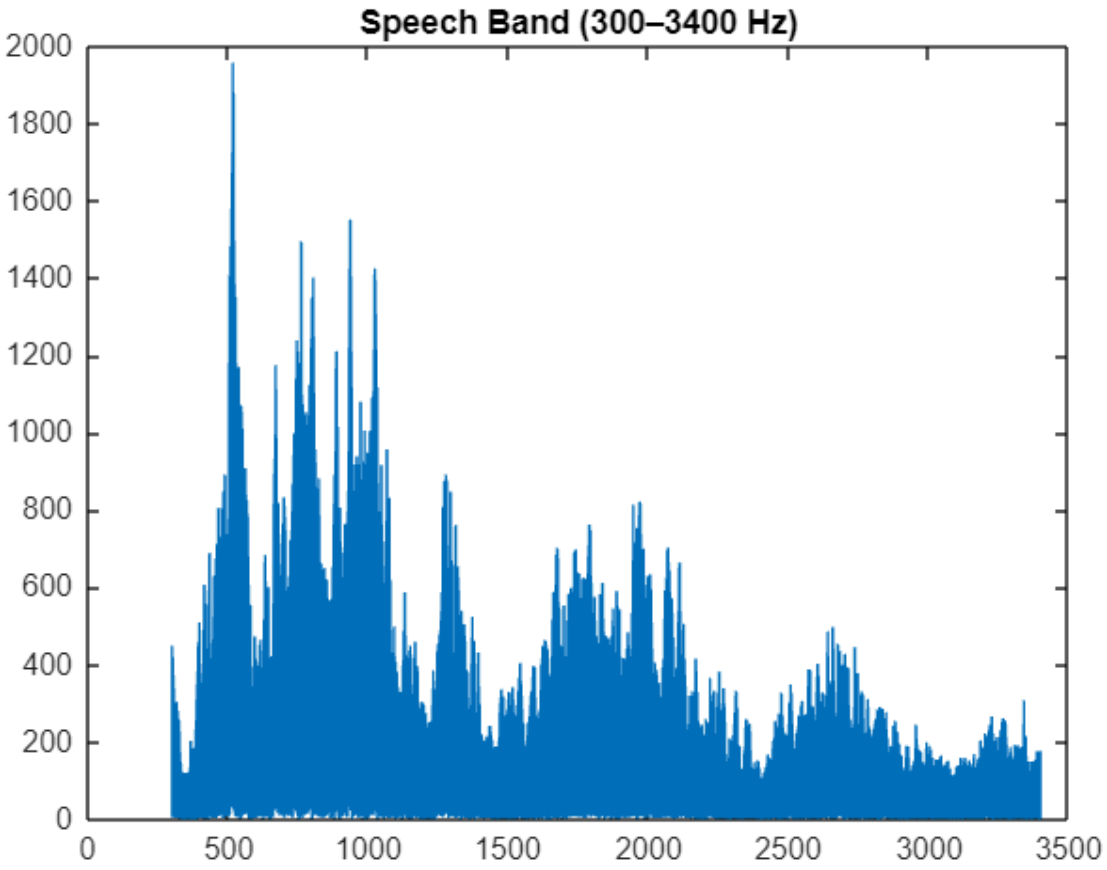

Waveform
Amplitude
Time (s)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
140
160
180

- **Waveform**: Strong amplitudes in the beginning and end, with quieter middle — indicating natural speech dynamics and pauses.



- **Full Frequency Spectrum**: Most energy lies below 5 kHz. Higher frequency components (>5 kHz) are present but weak — potentially noise

or non-speech elements.



- **Speech Band (300–3400 Hz)**: After filtering and compression, key speech components remain. Peaks at ~500 Hz, 1000 Hz, and 2000 Hz match typical speech formants. Noise outside the speech band is effectively removed, preserving intelligibility and reducing unnecessary data.

Spectrum Comparison

- **Before compression**: Energy spans up to 20,000 Hz but is mostly below 5,000 Hz. High-frequency noise outside the speech band is present.

- **After compression**: Frequency range is limited to 300–3400 Hz. Key speech components remain (especially between 500–2500 Hz), while non-speech noise is eliminated — improving clarity and reducing data size.

Compression successfully removes unneeded frequencies while preserving important speech information — ideal for hearing aid design.

V. Evaluation

5.1. Time Domain Comparison

- Post-WDRC waveform shows balanced amplitude, reducing dynamic range.
- No severe distortion — speech characteristics are retained.

5.2. Frequency Domain Comparison (FFT)

- Original spectrum has concentrated energy in the 300–3400 Hz band.
- Post-processing retains this energy, showing effective compression without losing key information.
- Slight reduction of out-of-band frequencies helps in noise suppression.

VI. Appendix: FFT

6.1. What is FFT?

FFT (Fast Fourier Transform) is an algorithm that converts a signal from the time domain to the frequency domain. It reveals which frequencies are present in the signal and their respective intensities.

6.2. Why use FFT in this project?

Audio signals are time-based, but analyzing them in the frequency domain helps understand pitch and tone.

FFT is much faster than the traditional DFT (Discrete Fourier Transform), especially for long signals like audio recordings.

- DFT complexity: $O(N^2)$
- FFT complexity: $O(N \log N)$

FFT is significantly faster, making it ideal for real-time audio processing — especially in MATLAB, where FFT functions are highly optimized.

6.3. How to use FFT in MATLAB?

- Use the fft() function — it returns a complex vector representing the frequency spectrum.
- Use abs() to get the magnitude.
- Plot only half of the spectrum (since it's symmetric for real signals).
- Normalize the frequency axis using the sample rate Fs.

6.4. Why identify the speech frequency band?

Hearing aids aim to enhance speech, so filtering and focusing on this band is essential.

Correct identification allows appropriate filtering (e.g., WDRC), removing unwanted noise.

6.5. Key considerations when using FFT in MATLAB:

- Input signal must be a vector — usually mono audio.
- Plot only the first half of the FFT (due to symmetry).
- Normalize frequency axis using sampling rate Fs to get correct Hz units.