Audio Noise Reduction for Voice Communication

AVE 4501 | Project Report

Signals & Systems

Submitted to: Asst. Prof. Dr. Md. Sakir Hossain

Submitted By:

Fourier Force

Fariha Islam Shochho (22024011)

Snigdha Das (22024012)

Md. Toriqul Islam Labib (22024027)

“Audio Noise Reduction for Voice Communication”

Abstract:

This project, titled Audio Noise Reduction for Voice Communication, focuses on enhancing the clarity and intelligibility of audio signals by effectively removing unwanted noise while retaining the natural quality of speech. Noise, often an inevitable part of real-world audio recordings, can significantly degrade the quality of communication, making it harder to understand spoken content. This challenge is addressed using a structured, multi-stage approach that combines three powerful noise reduction techniques: FFT-based noise filtering, wavelet-based denoising, and high-decibel attenuation.

The first stage employs Fast Fourier Transform (FFT) to work in the frequency domain, where speech frequencies (300–3400 Hz) are boosted while noise outside this range is selectively suppressed. This provides a foundational level of noise reduction while preserving critical speech components. The second stage uses wavelet-based denoising, a method that breaks the signal into multiple frequency bands to identify and remove transient noise components with precision. By applying a soft-thresholding technique to wavelet coefficients, this stage ensures high-frequency noise is reduced without distorting the voice signal. Finally, high-decibel attenuation targets abrupt, high-energy noise bursts that might still persist, softening them to achieve a balanced and natural-sounding audio output.

To evaluate the system's performance, visual and analytical tools are employed. Time-domain waveforms, frequency-domain FFT plots, and spectrograms clearly illustrate the reduction in noise and the preservation of speech features. The system has shown notable improvements in audio clarity and is particularly effective in scenarios where background noise interferes with voice communication, such as in telephony, speech recognition, or video conferencing.

Despite its strengths, the current implementation has some limitations. It assumes a relatively stationary noise profile, making it less effective in dynamic environments where noise levels vary. Furthermore, it is designed for offline processing and lacks real-time capabilities, which are critical for live applications.

Nevertheless, this project demonstrates the power of signal processing techniques in tackling noise challenges and provides a strong foundation for future enhancements. With potential upgrades like adaptive filtering, machine learning integration, and real-time functionality, this system could become a versatile solution for a wide range of noise reduction needs. Ultimately, the project

highlights how advanced signal processing can transform noisy recordings into clear, intelligible audio, enabling seamless communication in challenging environments.

Objectives:

The objectives of this project are:

1. To evaluate and remove noise from a noisy audio source while retaining the original signal.
2. To evaluate the effectiveness of the noise reduction process using time domain to frequency domain representations.

To explore the applications and future upgrades such as real-time noise reduction and adaptive filtering

Theory:

Noise can be defined as an unwanted signal that interferes with the communication or measurement of another signal.[1] Interestingly, the noise is a signal conveying information regarding the noise source. For instance, the noise from a car engine may provide insights into the state of the engine. However, in communication systems, noise degrades the quality of voice signals by masking or distorting the intended message. This degradation can significantly impact the clarity of communication, reducing noise a critical focus in signal processing.[1]. The human voice typically falls within the frequency range of 300–3000 Hz, which is essential for speech intelligibility. This frequency range is optimized in telecommunication systems, such as telephony, to minimize bandwidth usage while preserving the clarity and comprehensibility of speech. Fourier transformation is one of the most widely used techniques in applied mathematics, particularly in the fields of signal processing and data analysis [2]. This method is based on the idea that signals with suitable properties can be represented as linear combinations of trigonometric functions. The Fourier transform extracts frequency-domain information from a time-domain signal by decomposing a signal into trigonometric components. Consequently, it offers an alternative and often more efficient way to investigate or modify a signal, making it invaluable for noise reduction and filtering applications [2].

Image denoising plays an important role in image processing, which aims to separate clean images from noisy images. A number of methods have been presented to deal with this practical problem over the past several years. The best currently available wavelet-based denoising methods take advantage of the merits of the wavelet transform [3]. Wavelet-based denoising is a signal-processing approach that decreases signal noise while retaining important features like edges and transients. It uses the discrete wavelet transform (DWT) to break down a signal into approximation (low-frequency) and detail (high-frequency) components. Noise, generally concentrated on high-frequency information, is reduced using thresholding techniques such as hard or soft thresholding. The signal is then rebuilt with the inverse wavelet transformation. This approach has several advantages, including good handling of non-stationary data, preservation of essential features, and multiscale analysis. It is widely utilized in applications such as audio processing, image enhancement, and biological signal analysis, as it strikes a compromise between noise reduction and signal integrity. High decibel attenuation refers to a large reduction in sound intensity, which is frequently done by employing advanced procedures or materials designed to absorb or block sound waves effectively. It is widely utilized in noise control applications, including soundproofing, industrial noise reduction, and hearing protection. The process often includes techniques such as energy dissipation by damping, reflection via barriers, and absorption by acoustic materials. High decibel attenuation is critical for reducing noise pollution, assuring safety, and increasing comfort in a variety of situations.

Methodology

The methodology for this project focuses on implementing a multi-stage approach to reduce noise in voice communication. The goal is to enhance the intelligibility and quality of audio signals by minimizing unwanted background noise while preserving the clarity of speech. The process involves three key stages: **FFT-based noise reduction**, **Wavelet-based denoising**, and **High-decibel attenuation**. Each stage targets different aspects of noise while complementing the others to achieve a robust noise reduction system.

1. FFT-Based Noise Reduction

The first stage leverages the Fast Fourier Transform (FFT) to operate in the frequency domain. Audio signals, which are time-domain data by default, often have noise and speech components overlapping across various frequencies. By converting the signal to the frequency domain, we can identify and selectively suppress noise while amplifying the frequencies associated with speech.

The signal is divided into overlapping frames using a Hann window, ensuring smooth transitions between frames. The FFT of each frame is computed, and the magnitude spectrum is analyzed. Speech frequencies are typically concentrated between **100 Hz** and **3000 Hz**, while noise often dominates lower and higher frequencies. The signal becomes cleaner by selectively boosting these speech frequencies and attenuating noise-dominated regions. A noise floor, estimated dynamically from the tail of the spectrum, ensures that unwanted components are effectively suppressed without introducing artifacts.

This stage enhances the clarity of speech by targeting noise in the frequency domain. However, some residual noise and artifacts may remain, which are addressed in subsequent stages.

2. Wavelet-Based Denoising

The second stage employs wavelet-based denoising, a time-frequency domain approach ideal for processing non-stationary signals like speech. Unlike FFT, which operates on a fixed resolution, wavelets provide multi-resolution analysis, capturing both high-frequency transients and low-frequency components effectively.

The input signal is decomposed into multiple levels of wavelet coefficients using the Daubechies (db6) wavelet, chosen for its ability to represent smooth and sharp transitions in signals. Noise typically manifests as high-frequency components, which can be identified and suppressed by thresholding the wavelet coefficients. A soft thresholding technique is applied, where coefficients below a certain value are shrunk towards zero, preserving the integrity of speech-dominant components.

This method significantly reduces high-frequency noise without distorting the speech signal, thanks to its ability to adapt to the local characteristics of the audio.

3. High-Decibel Attenuation

The final stage focuses on controlling abrupt, high-energy noise bursts, such as clicks or background chatter. These components can disrupt the listening experience and are challenging to address solely through frequency-based methods.

Here, we analyze the energy of the signal over small, overlapping windows. A threshold, corresponding to a decibel level (e.g., -20 dB), is set to identify frames with excessively high energy. When such frames are detected, they are attenuated using a scaling factor. This ensures that loud noise bursts are softened without entirely muting the signal, preserving its natural character.

Integration and Normalization

Once these stages are applied, the processed signal undergoes normalization to ensure the output amplitude remains within a consistent range. This final step prevents clipping or distortion and ensures that the audio is ready for playback or further use.

Rationale for a Multi-Stage Approach

Each of the methods used in this project addresses a specific aspect of noise reduction:

- FFT-based processing enhances the overall signal clarity by focusing on frequency-specific noise.
- Wavelet-based denoising excels in reducing high-frequency transient noise without affecting speech quality.
- High-decibel attenuation provides a safeguard against sudden loud noises.

By combining these approaches, the methodology ensures a balanced trade-off between noise reduction and speech preservation, resulting in a clear and intelligible output suitable for voice communication applications.

Block diagram

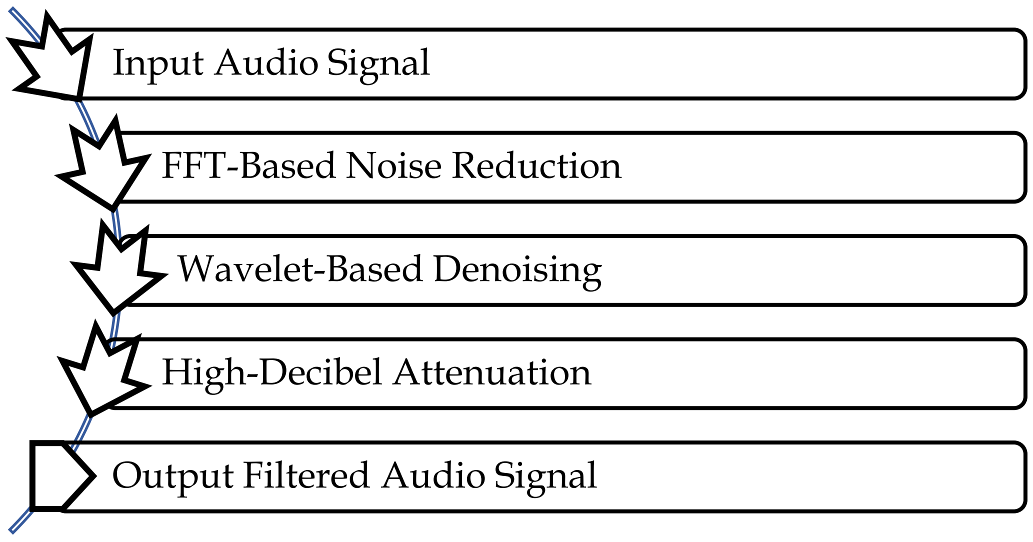

Fig. 1 Block diagram of the program

Software Requirement

1. **Primary Software:** MATLAB

Version: Any modern version (R2019 or later recommended for compatibility).

Key Functions: Signal processing, wavelet transformations, and visualization.

2. **MATLAB Toolboxes**

   - **Signal Processing Toolbox:** Required for FFT operations, filtering, and spectrogram visualization.
   - **Wavelet Toolbox:** Essential for wavelet-based denoising techniques.
   - **Audio Toolbox (Optional):** Useful for enhanced audio I/O functionalities and processing.

.

3. **Development Features**

Built-in debugging tools for stepwise execution and troubleshooting. Plotting and visualization tools to create waveforms, spectrograms, and FFT magnitude graphs.

<u>Flow Chart:</u>

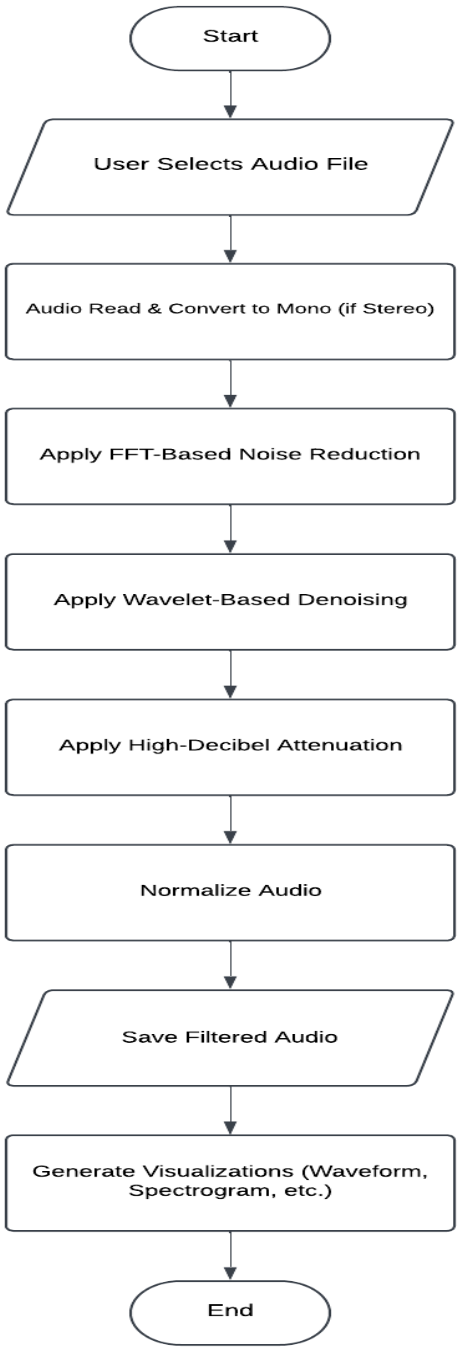
Start
User Selects Audio File
Audio Read & Convert to Mono (if Stereo)
Apply FFT-Based Noise Reduction
Apply Wavelet-Based Denoising
Apply High-Decibel Attenuation
Normalize Audio
Save Filtered Audio
Generate Visualizations (Waveform, Spectrogram, etc.)
End

Fig. 2 Flow chart of the program

Algorithm

Step 1: Input Audio Preparation

1. **Start** the program.
2. **Prompt the user** to select an audio file.
3. **Read the audio file** and check its format.
4. **Convert** the audio to mono if it is stereo.

Step 2: Noise Reduction Process

1. **Stage 1: Frequency Domain Noise Reduction (FFT)**
   - Break the audio signal into small frames.
   - For each frame:
     - Apply a windowing function.
     - Transform the signal into the frequency domain using **FFT**.
     - Identify and suppress frequencies that are outside the desired voice range.
     - Convert the processed frame back to the time domain using **Inverse FFT** (IFFT).
2. **Stage 2: Time-Frequency Domain Noise Reduction (Wavelet Denoising)**
   - Perform **wavelet decomposition** on the audio to separate the signal into different frequency components.
   - Estimate the noise level from the wavelet coefficients.
   - Apply a **thresholding technique** to remove noise while retaining the important speech components.
   - Reconstruct the signal from the denoised wavelet coefficients.
3. **Stage 3: Energy-Based Noise Attenuation**
   - Calculate the **energy** of the signal over a short time window.
   - Identify segments with high energy (likely noise or loud unwanted sounds).
   - Apply **attenuation** to these segments to reduce their loudness.

Step 3: Post-Processing

1. **Normalize** the filtered audio to ensure it has a consistent amplitude (to prevent distortion).

Step 4: Output and Visualization

1. **Save** the processed audio to a new file.
2. **Generate visualizations** of the original and filtered audio, including:
   - Waveforms
   - Spectrograms
   - Frequency domain analysis (e.g., FFT plots)

Step 5: End

1. **End** the process.

Program MATLAB Code and Explanation

Implementing the Audio Noise Reduction system involves several key stages, each of which is executed by a dedicated function within the code. The whole code combined with all the functions is uploaded to the web Git Repository of this project: https://github.com/Snigdha-prime/signal-and-system-project/blob/main/FINAL_Edit_Audio_Noise_Reduction_for_Voice_Communication.m

The following sections provide an overview of the main parts of the code and their respective functionalities.

1.main_noise_reduction

```
function main_noise_reduction()
   % Load the audio file
   [filename, pathname] = uigetfile({'*.wav; *.mp3', 'Audio Files (*.wav, *.mp3)'}, 'Select an
audio file');
   if isequal(filename, 0) || isequal(pathname, 0)
      disp('User canceled the file selection.');
      return;
   end
   audioFilePath = fullfile(pathname, filename);
   [audioIn, fs] = audioread(audioFilePath);

   % Convert to mono if stereo
   if size(audioIn, 2) > 1
      audioIn = mean(audioIn, 2);
   end
```

```
    % Apply noise reduction
    audioFiltered = advanced_noise_reduction(audioIn, fs);

    % Save the filtered audio
    [~, name, ~] = fileparts(filename);
    outputFilePath = fullfile(pathname, [name '_filtered.wav']);
    audiowrite(outputFilePath, audioFiltered, fs);
    disp(['Filtered audio saved to: ', outputFilePath]);

    % Generate figures
    generate_figures(audioIn, audioFiltered, fs);
end

function audioFiltered = advanced_noise_reduction(audioIn, fs)
    % Ensure column vector
    audioIn = audioIn(:);

    % Stage 1: FFT-based Noise Reduction
    fftFiltered = fft_noise_reduction(audioIn, fs);

    % Stage 2: Wavelet-Based Denoising
    waveletFiltered = wavelet_denoising(fftFiltered, fs);

    % Stage 3: High-Decibel Attenuation
    audioFiltered = high_decibel_attenuation(waveletFiltered, fs);

    % Final normalization
    audioFiltered = audioFiltered / max(abs(audioFiltered));
end
```

Explanation:

1. **Audio File Selection**:
   - The `uigetfile` function opens a file dialog allowing the user to select an audio file. Only `.wav` and `.mp3` files are displayed.
   - If the user cancels, the program exits gracefully by returning and displaying a message.
2. **File Path Construction**:
   - Combines the selected path (`pathname`) and file name (`filename`) into a full file path using `fullfile` for compatibility across operating systems.
3. **Reading the Audio**:
   - The `audioread` function loads the audio file, returning:
     - `audioIn`: the audio signal as an array.
     - `fs`: the sampling rate, which is crucial for subsequent processing.
4. **Mono Conversion**:
   - If the audio has two channels (stereo), the mean of the two channels is calculated, converting it to mono. This simplifies processing and ensures compatibility with noise reduction functions.
5. **Noise Reduction**:

    - The core functionality is handled by the `advanced_noise_reduction` function, which applies multiple noise reduction techniques to clean the audio signal.
6. **Save the Processed Audio**:
    - Extracts the original file name using `fileparts` and appends `_filtered` to differentiate it from the original. The cleaned audio is saved in the same folder.
7. **Visualization**:
    - Calls `generate_figures` to create a series of plots that compare the original and filtered audio in terms of waveform, spectrogram, and other metrics.

2.advanced_noise_reduction

```
function audioFiltered = advanced_noise_reduction(audioIn, fs)
    % Ensure column vector
    audioIn = audioIn(:);

    % Stage 1: FFT-based Noise Reduction
    fftFiltered = fft_noise_reduction(audioIn, fs);

    % Stage 2: Wavelet-Based Denoising
    waveletFiltered = wavelet_denoising(fftFiltered, fs);

    % Stage 3: High-Decibel Attenuation
    audioFiltered = high_decibel_attenuation(waveletFiltered, fs);

    % Final normalization
    audioFiltered = audioFiltered / max(abs(audioFiltered));
end
```

Explanation:

1. **Column Vector Conversion**:
    - Audio data is reshaped into a column vector (single-dimensional array) to ensure consistent processing across all functions.
2. **Stage 1: FFT-Based Noise Reduction**:
    - Removes noise in the frequency domain by analyzing and selectively attenuating frequency components that correspond to noise. Speech frequencies are enhanced for clarity.
3. **Stage 2: Wavelet-Based Denoising**:
    - Removes noise by decomposing the signal into wavelet coefficients, applying a threshold to eliminate small coefficients (which likely represent noise), and reconstructing the signal.
4. **Stage 3: High-Decibel Attenuation**:
    - Reduces sudden, loud bursts of noise that exceed a defined decibel threshold, making the audio more consistent.
5. **Normalization**:
    - The output signal is scaled so its maximum amplitude is `1`, preventing distortion or clipping when played back.

3.fft_noise_reduction

```
function audioFiltered = fft_noise_reduction(audioIn, fs)
  % FFT-based noise reduction using overlap-add for smooth reconstruction
  windowLength = round(0.05 * fs); % 50ms windows
  overlap = round(windowLength * 0.75);
  hopSize = windowLength - overlap;
  window = hanning(windowLength);

  % Zero-padding and pre-allocate output
  paddedSignal = [zeros(overlap, 1); audioIn; zeros(overlap, 1)];
  audioFiltered = zeros(size(paddedSignal));

  for i = 1:hopSize:(length(paddedSignal) - windowLength + 1)
    % Windowed frame
    frame = paddedSignal(i:i + windowLength - 1) .* window;

    % FFT
    fftFrame = fft(frame);
    magnitude = abs(fftFrame);
    phase = angle(fftFrame);

    % Noise reduction in frequency domain
    freqRes = fs / length(fftFrame);
    voiceFreqIndices = round((300 / freqRes):(3400 / freqRes));
    magnitude(voiceFreqIndices) = magnitude(voiceFreqIndices) * 1.2; % Boost speech
frequencies
    noiseLevel = mean(magnitude(floor(end * 0.75):end)) * 0.5;
    magnitudeProcessed = max(magnitude - noiseLevel, 0.1 * noiseLevel);

    % Reconstruct frame with original phase
    processedFrame = real(ifft(magnitudeProcessed .* exp(1j * phase)));

    % Overlap-add synthesis
    audioFiltered(i:i + windowLength - 1) = audioFiltered(i:i + windowLength - 1) +
processedFrame .* window;
  end

  % Remove padding
  audioFiltered = audioFiltered(overlap + 1:end - overlap);
end
```

Explanation:

1. **Windowing**:
   - The audio is divided into small overlapping frames (50ms long with 75% overlap), which helps smooth transitions between processed segments.
   - A Hanning window is applied to each frame to reduce edge effects.
2. **FFT Transformation**:
   - Each frame is transformed into the frequency domain using the Fast Fourier Transform (FFT), separating the signal into magnitude (amplitude of each frequency) and phase (timing information).

3. **Noise Reduction**:
   - Frequency bins corresponding to speech (300–3400 Hz) are boosted by 20%.
   - Noise is estimated from higher frequencies (often containing less speech) and subtracted from the spectrum, ensuring it does not drop below 10% of the noise level.
4. **Reconstruction**:
   - The modified magnitude is recombined with the original phase, and the Inverse FFT (IFFT) transforms the frame back to the time domain.
5. **Overlap-Add Synthesis**:
   - Processed frames are combined with overlap to reconstruct the entire signal, maintaining continuity.
6. **Padding Removal**:
   - Extra zeros added for processing are removed to restore the original signal length.

4.wavelet_denoising

```
function denoisedAudio = wavelet_denoising(audioIn, ~)
    % Wavelet-based denoising
    waveletName = 'db6';
    decompositionLevel = 5;
    [C, L] = wavedec(audioIn, decompositionLevel, waveletName);
    sigma = median(abs(C(L(1):end))) / 0.6745;
    threshold = sigma * sqrt(2 * log(length(audioIn)));
    denoisedC = wthresh(C, 's', threshold);
    denoisedAudio = waverec(denoisedC, L, waveletName);
end
```

Explanation:

1. **Wavelet Decomposition**:
   - The audio signal is broken down into a hierarchy of coefficients using a discrete wavelet transform (`wavedec`) with the wavelet type `'db6'` (a Daubechies wavelet).
   - `decompositionLevel` specifies the number of levels for the transform, controlling the granularity of frequency analysis. Higher levels capture lower-frequency components.
   - The output `C` contains all the wavelet coefficients, and `L` provides information about the size of coefficients at each level.
2. **Noise Estimation**:
   - Noise is estimated using the detail coefficients (higher-frequency components) from the finest level of decomposition.
   - The noise standard deviation (`sigma`) is calculated using the robust MAD estimator, which is less sensitive to outliers in the data.
3. **Threshold Calculation**:
   - A threshold is set based on the noise level using a rule involving the signal length and logarithmic scaling. This threshold distinguishes noise from meaningful signal components.
4. **Thresholding**:

    - A soft thresholding (`wthresh`) is applied to the wavelet coefficients (`C`). This suppresses noise by reducing small coefficients (likely noise) while retaining larger coefficients (likely signal).
5. **Reconstruction**:
    - The denoised coefficients (`denoisedC`) are used to reconstruct the audio signal via the inverse wavelet transform (`waverec`), resulting in a cleaner signal.

5.high_decibel_attenuation

```
function audioOut = high_decibel_attenuation(audioIn, fs)
   % High-decibel attenuation
   dbThreshold = -20;
   linearThreshold = 10^(dbThreshold / 20);
   windowLength = round(0.05 * fs);
   energy = sqrt(movmean(audioIn.^2, windowLength));
   attenuationFactor = 0.5;
   audioOut = audioIn;
   audioOut(energy > linearThreshold) = audioOut(energy > linearThreshold) *
attenuationFactor;
end
```

Explanation:

1. **Threshold in Decibels**:
    - A decibel threshold (`dbThreshold = -20 dB`) is set to identify excessively loud parts of the signal. This is converted to a linear amplitude threshold (`linearThreshold`) for easier comparison with the signal's energy levels.
2. **Energy Calculation**:
    - The energy of the signal is computed using a moving average of the squared amplitude (`movmean(audioIn.^2, windowLength)`).
    - A 50ms window (`windowLength = 0.05 * fs`) smooths the energy profile, allowing for more accurate detection of sustained loud regions.
3. **Attenuation Logic**:
    - If the energy in any region exceeds the calculated threshold, the corresponding parts of the signal are attenuated by a factor of 0.5 (`attenuationFactor`).
4. **Signal Modification**:
    - The modified signal (`audioOut`) retains the quieter portions unchanged but suppresses the amplitude of overly loud sections to make the audio more uniform and less jarring.

Results and Analysis:

The provided set of images showcases the results of applying noise reduction techniques to an audio signal. The input for this analysis was a recording of a radio communication, which was likely corrupted by various types of interference and background noise.

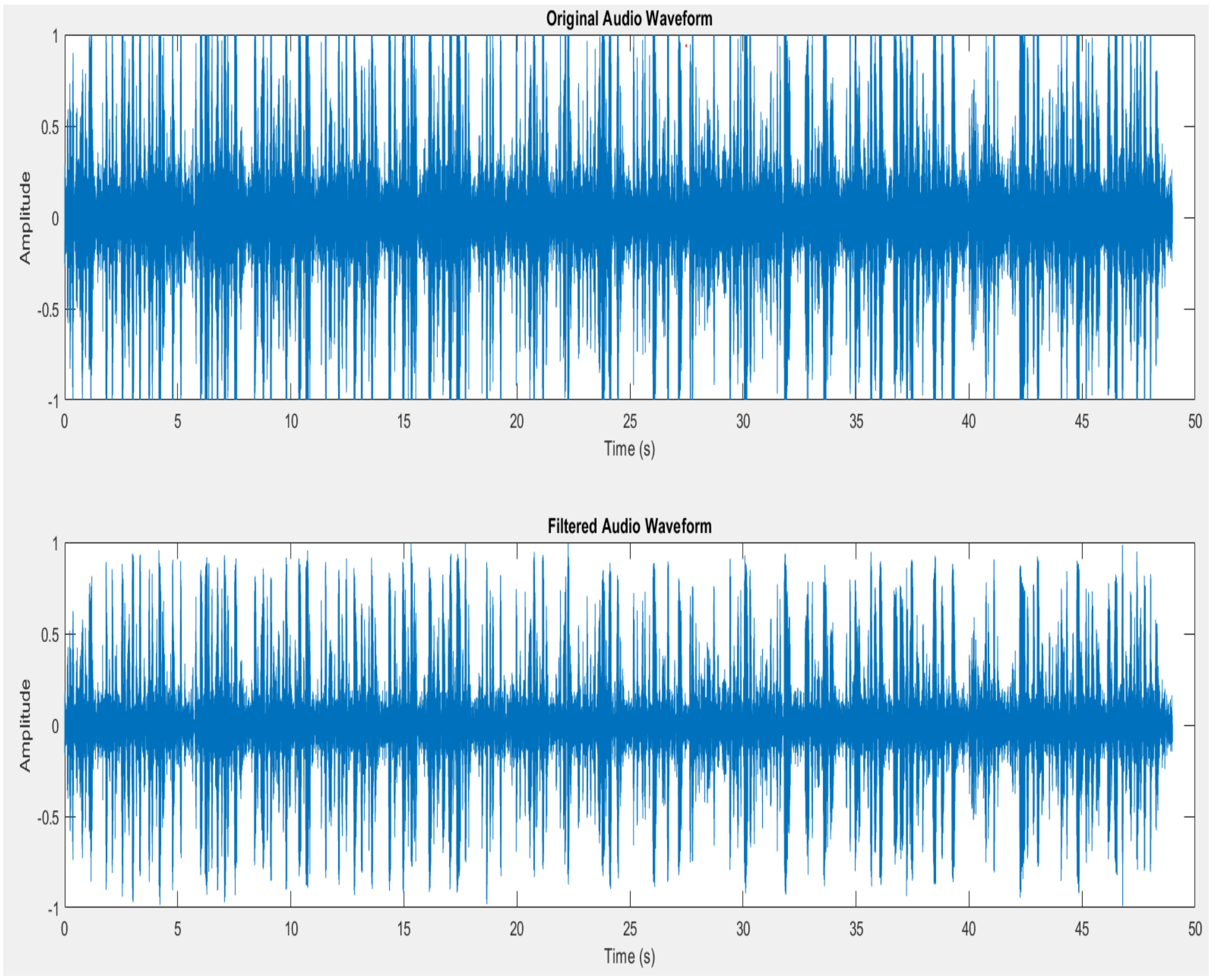

Fig. 3 Waveform comparison

Image 1: Original Audio Waveform and Filtered Audio Waveform

The top plot in Image 1 displays the original waveform of the audio signal over time. The waveform exhibits a complex and highly fluctuating pattern, which is characteristic of an audio signal that has been corrupted by noise or interference. The rapid and irregular changes in the amplitude of the waveform suggest the presence of high-frequency components that are not part of the desired audio signal.

The bottom plot in Image 1 shows the filtered version of the same audio signal. In comparison to the original waveform, the filtered waveform appears much smoother and more organized. The high-frequency fluctuations have been significantly reduced, and the overall shape of the waveform is more consistent and coherent. This indicates that the filtering process has effectively removed or attenuated the unwanted noise components, resulting in a cleaner and more refined audio signal.

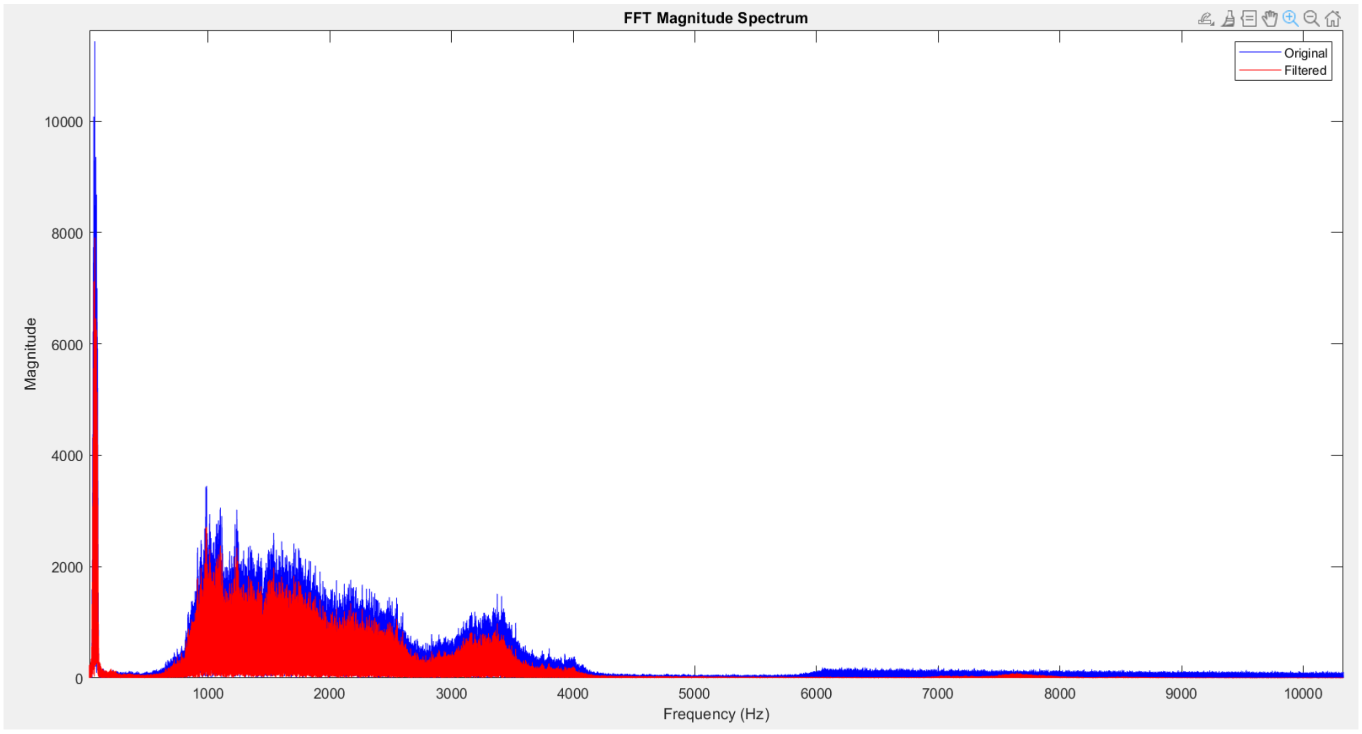

Fig. 4 FFT Magnitude Spectrum

Image 2: FFT Magnitude Spectrum

The FFT (Fast Fourier Transform) magnitude spectrum, shown in Image 2, provides a representation of the frequency composition of the audio signal. The original spectrum, shown in blue, exhibits a wide range of high-frequency components, which are likely associated with the noise or interference present in the original signal.

In contrast, the filtered spectrum, shown in red, has a more concentrated energy distribution in the lower frequency range. The high-frequency components have been significantly reduced, indicating that the filtering process has successfully removed or attenuated the unwanted high-frequency noise. This is an essential step in improving the quality of the audio signal, as the human auditory system is more sensitive to the lower and mid-range frequencies that are typically associated with the desired speech or audio content.

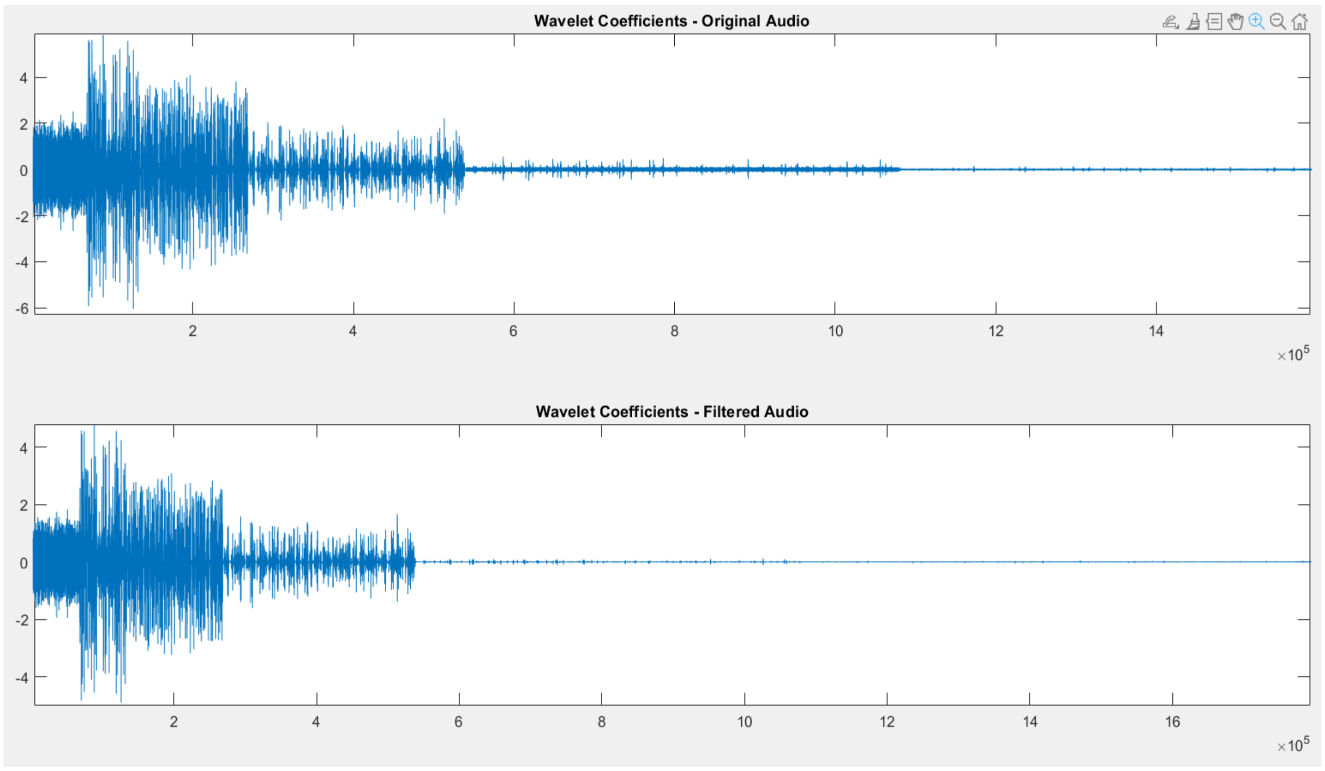

Fig. 5 Wavelet Coefficient

Image 3: Wavelet Coefficients

The wavelet coefficient plots in Image 3 provide a time-frequency analysis of the audio signal. The top plot shows the wavelet coefficients of the original signal, which exhibit a more complex and scattered pattern. This pattern suggests the presence of various frequency components distributed across the signal duration, including both the desired signal and the unwanted noise.

The bottom plot shows the wavelet coefficients of the filtered signal. In this case, the wavelet coefficients appear more concentrated and organized, indicating that the filtering process has effectively isolated the dominant frequency components while suppressing the unwanted noise. The reduction in the overall spread and complexity of the wavelet coefficients suggests that the filtering has improved the time-frequency characteristics of the audio signal, making it more suitable for further processing or analysis.

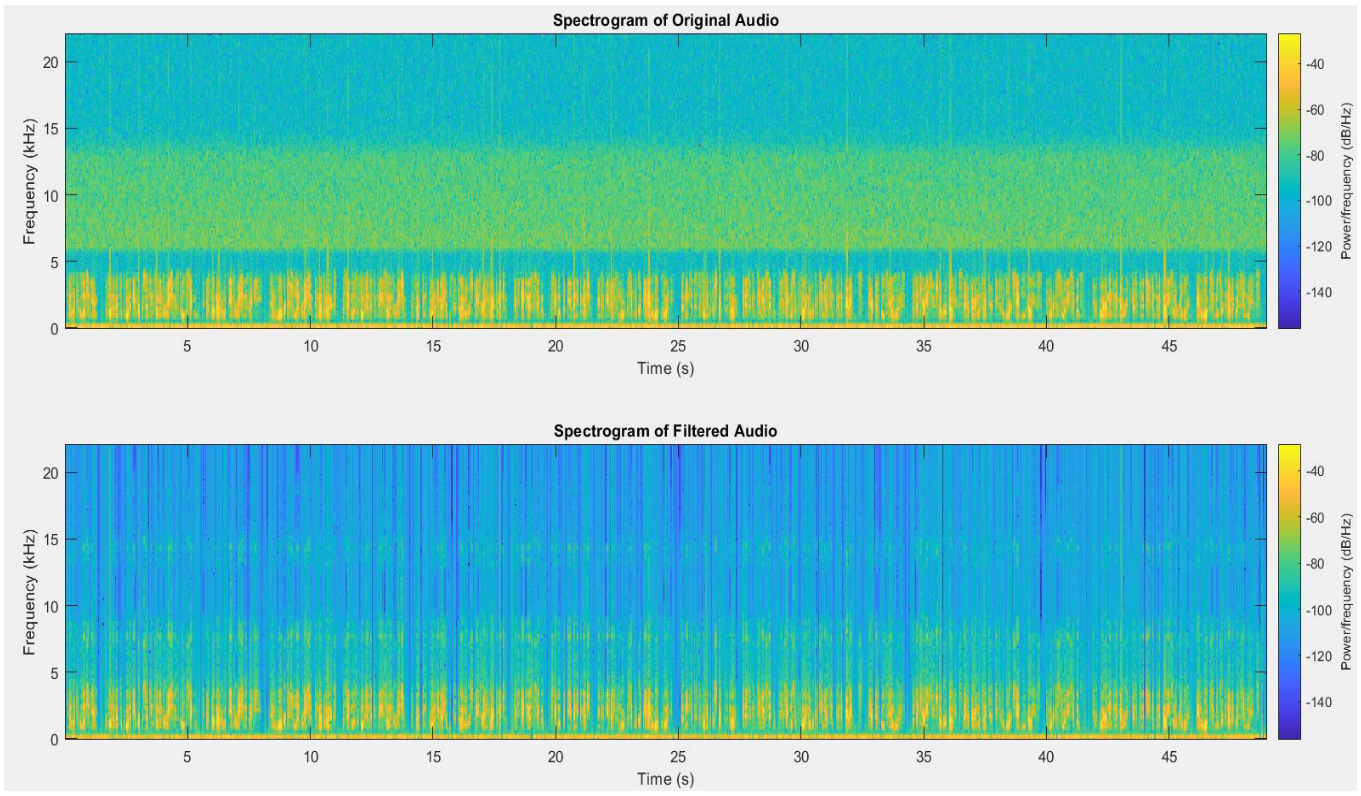

Fig. 6 Spectrogram of Original and Filtered Audio

Image 4: Spectrogram of Original and Filtered Audio

The spectrogram visualizations in Image 4 provide a time-frequency representation of the audio signals. The original spectrogram (top) displays a wider range of frequency bands with varying intensities, indicating the presence of diverse frequency components, including both the desired signal and the unwanted noise.

In contrast, the filtered spectrogram (bottom) shows a more focused energy distribution, primarily concentrated in the lower frequency regions. This observation aligns with the previous findings, suggesting that the filtering process has successfully removed or attenuated the high-frequency noise components, while preserving the essential features and characteristics of the desired audio signal.

The improved spectral clarity and the reduced presence of high-frequency noise in the filtered spectrogram indicate that the applied noise reduction techniques have been effective in enhancing the overall quality and intelligibility of the audio signal. This is crucial in various applications, such as speech recognition, audio processing, or communication systems, where maintaining a clear and noise-free audio signal is essential for accurate interpretation and analysis.

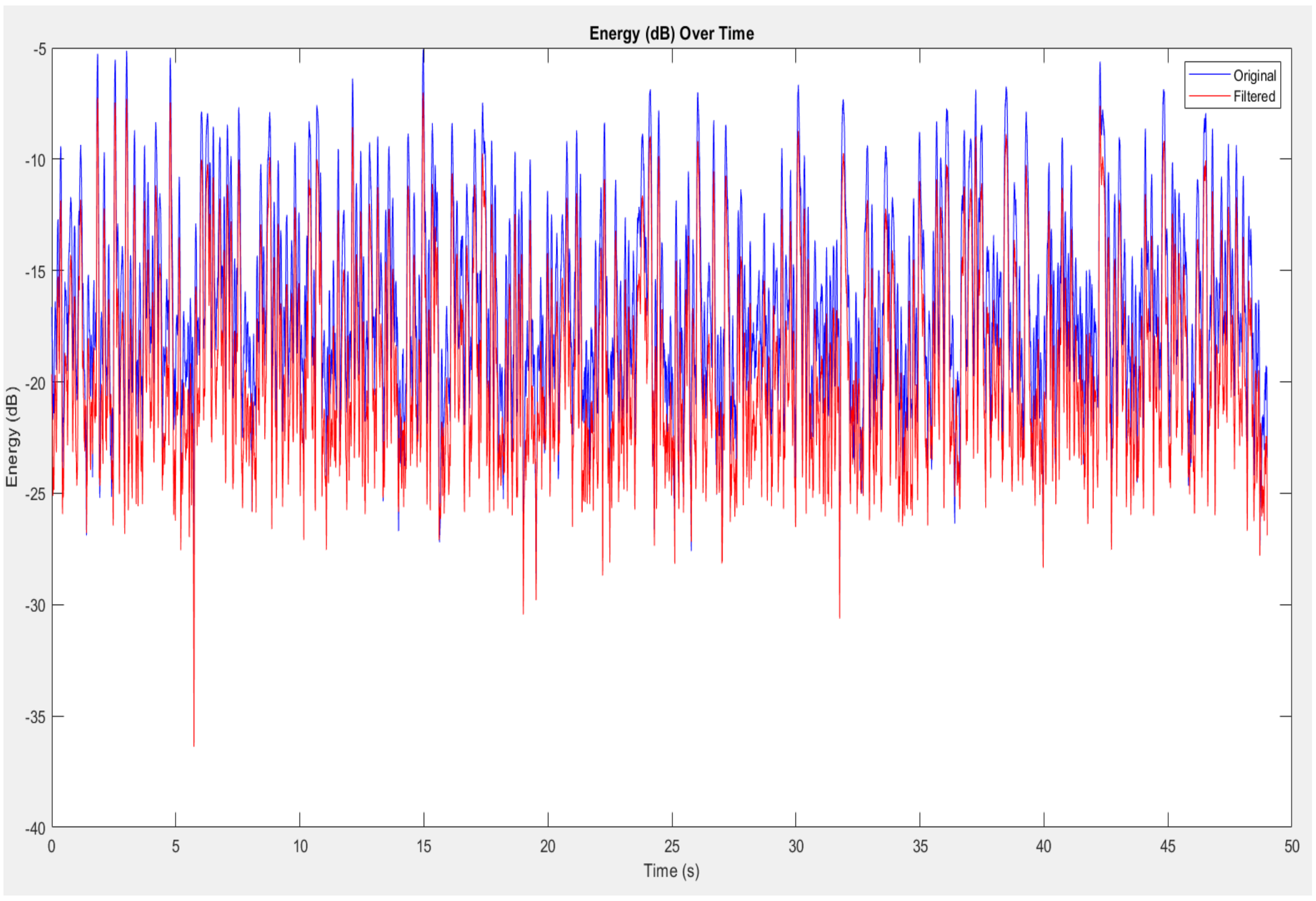

Fig. 7 Energy (dB) Over Time

Image 5: Energy (dB) Over Time

This plot shows the energy, measured in decibels (dB), of the audio signal over time. The blue line represents the energy of the original signal, while the red line represents the energy of the filtered signal.

The original signal exhibits a more fluctuating energy profile, with several peaks and valleys throughout the duration. This is indicative of the presence of high-frequency noise components that contribute to the overall energy of the signal.

In contrast, the filtered signal shows a more consistent and stable energy profile, with fewer pronounced peaks and valleys. This suggests that the noise reduction techniques have effectively attenuated the high-frequency noise components, resulting in a more uniform and controlled energy distribution over time. This improved energy profile can be beneficial for various applications that require a more stable and predictable audio signal.

Overall, the results presented in these images demonstrate the effectiveness of the noise reduction techniques applied to the original audio signal. The filtering and processing steps have successfully removed or attenuated the unwanted high-frequency noise components, while preserving the essential features and characteristics of the desired audio signal. This improvement in signal quality can be beneficial for a wide range of applications that rely on high-fidelity audio processing and analysis.

Limitation

The Audio Noise Reduction system has several limitations. It assumes a constant noise profile, which may not work well for dynamic or non-stationary noise. The FFT-based noise reduction focuses primarily on speech frequencies, potentially leaving other noise frequencies unaddressed. Additionally, the processing stages can affect signal quality, with risks of over-smoothing or distortion. The system also does not support real-time processing, which makes it unsuitable for live communication. Computationally, the process is expensive, especially for longer files, and it lacks user control over key parameters like noise reduction strength and frequency range. Moreover, the automatic stereo-to-mono conversion results in the loss of spatial information, and the fixed parameters may not be optimal for all types of audio. Finally, the program does not support all audio file formats, and it may struggle with extremely noisy recordings where noise is similar in magnitude to the desired signal.

Future Scope

The Audio Noise Reduction system can be significantly enhanced in several areas to improve its effectiveness and broaden its applicability. First, the system could be extended to support real-time processing**,** enabling its use in live voice communication applications such as video calls or radio broadcasts. Incorporating adaptive noise reduction algorithms that adjust to varying noise conditions in real time would make the system more flexible and efficient in dynamic environments. Additionally, allowing user-controlled parameters for noise reduction strength, frequency targeting, and other settings would offer more customization and fine-tuning for different audio types and noise conditions.

Support for multi-channel audio and advanced file formats (e.g., .flac**,** .aac) would broaden its usability in professional audio production and broadcasting. Deep learning-based techniques could also be explored to improve the noise reduction process, as machine learning models have shown great potential in handling complex and non-stationary noise. Finally, the system could incorporate multi-frequency noise reduction to handle a broader range of noise types, including low-frequency hums and high-frequency hiss, thus making the system more robust across various audio applications.

Conclusion:

This project successfully tackled the challenge of removing noise from audio signals while preserving the quality of speech. By combining FFT-based filtering, wavelet-based denoising, and high-decibel attenuation, we created a system that significantly improves audio clarity. The results speak for themselves, with cleaner waveforms, reduced high-frequency noise, and smoother spectrograms, all contributing to a better listening experience.

However, we acknowledge some limitations. The system assumes a constant noise profile, making it less effective for environments where noise changes rapidly. Additionally, the lack of real-time processing means it's not yet suitable for live applications like video calls or radio broadcasts. Despite these constraints, the project provides a solid framework for future development. Adding adaptive filtering, machine learning techniques, and real-time capabilities would take this system to the next level.

In conclusion, this project has shown that even with relatively simple methods, substantial noise reduction can be achieved. It's a practical solution for improving communication quality in noisy settings and opens the door for further innovation in the field of audio processing.

.

Reference:

1. S. V. Vaseghi, Advanced Digital Signal and Noise Reduction. Wiley, 2000.
2. P. Anttonen, "Fourier transform techniques for noise reduction," IET Image Processing, vol. 6, pp. 6-7, 2022.
3. K. Bnou, S. Raghay, and A. Hakim, "A wavelet denoising approach based on unsupervised learning model," EURASIP J. Adv. Signal Process., vol. 2020, no. 36, 2020, doi: 10.1186/s13634-020-00693-4