INFORMATION TECHNOLOGY UNIVERSITY

Noise Removal from a Signal

CEP Project

For

Digital Signal Processing

Presented by:

Group Member 1	**BSCE21004** Wardah Binte Naveed
Group Member 2	**BSCE21005** Moeez Ahmad
Group Member 3	**BSCE21020** Obaid Sohail

Table of Contents

1. Introduction

Digital Signal Processing (DSP) enhances speech signal quality by modifying its spectrum. This report presents the design, implementation, and analysis of a DSP-based system to remove unwanted noise from audio signals. The objective is to improve the clarity of speech by attenuating undesired frequencies related to noise.

---

2. Planning Stage

2.1 Signal Processing Blocks

- **Audio Loading:**
  The script establishes file paths and loads the original audio and noise signals.

- **Signal Alignment:**
  Ensures both signals have the same sampling rate and trims them to the length of the smaller one.

- **Noise Scaling:**
  Adjusts the volume of the noise signal.

- **Noise Addition:**
  Adds scaled noise to the original signal.

- **Filter Design:**
  Implements a low-pass Butterworth filter for noise reduction.

- **Filter Application:**
  Applies the filter to the noisy signal.

- **Volume Adjustment:**
  Increases the volume of the filtered signal.

- **Additional Filtering:**
  Applies another low-pass Butterworth filter for further noise reduction.

- **Volume Adjustment (Again):**
  Adjusts the volume of the further denoised signal.

2.2 Justification

- **Signal Alignment:**
  Ensures consistency in signal comparison.

- **Noise Scaling:**
  Allows control over the noise level.
- **Filter Design:**
  Utilizes a low-pass Butterworth filter for its smooth frequency response and ease of parameter adjustment.
- **Volume Adjustment:**
  Compensates for any attenuation caused by filtering.

---

3. Implementation Stage

3.1 Parameters and Justification

Filter Selection:
A Butterworth low-pass filter is employed in the code for its maximally flat frequency response, ease of parameter adjustment (cutoff frequency and order), and zero-phase filtering using `filtfilt`. This choice ensures effective noise reduction in speech and audio signals while preserving the temporal characteristics of the original signal.

Cutoff Frequency:
Set at 900 Hz for the initial filter and 600 Hz for further noise reduction. Adjusting cutoff frequencies influences filter behavior.

Filter Order:
Initially set at 1, this can be adjusted based on desired roll-off and filter characteristics.

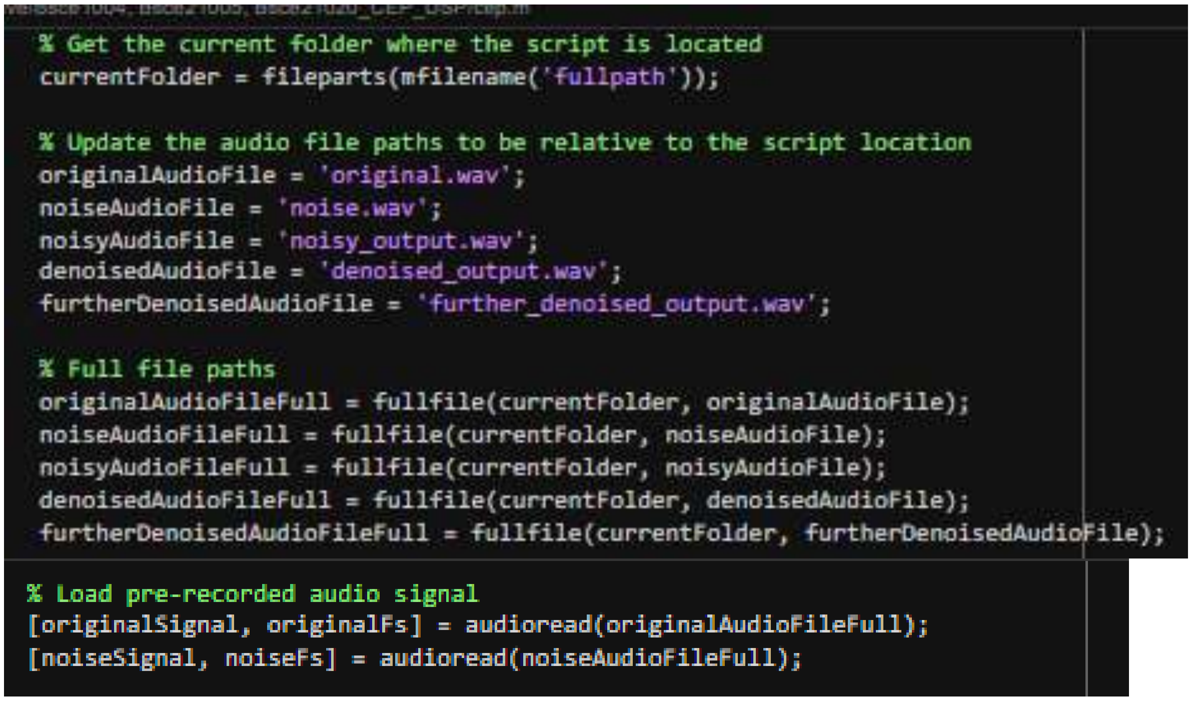
Scaling Factors:
Used to adjust the volume of the noise, filtered, and further denoised signals. Subjective and can be tuned based on specific audio characteristics.

---

4. Code Analysis

4.1 File Paths and Audio Loading

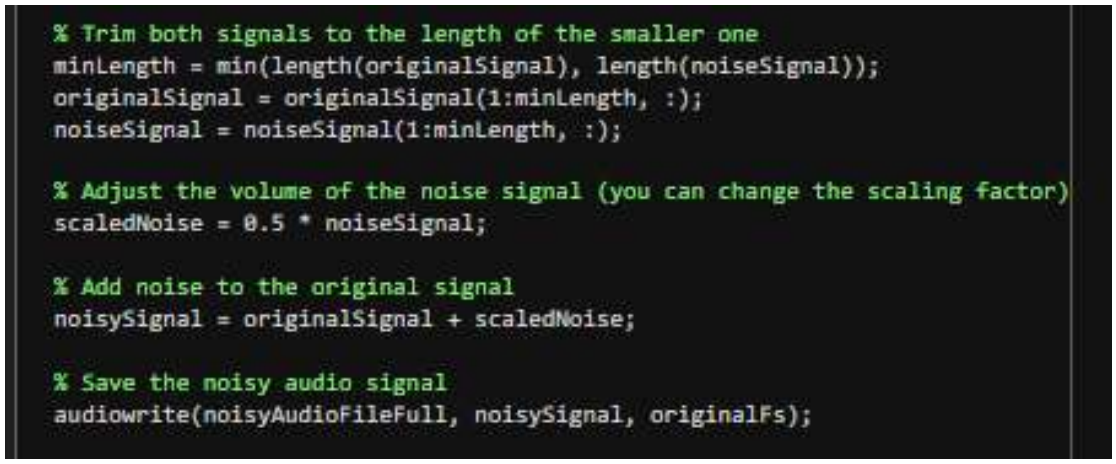
The script establishes file paths and loads “original” and “noise” audio signals using MATLAB's `audioread` function. Here the path of output files noisy_output.wav, denoised_output.wav, and further_denoised_audio.wav is given only. Even if you delete these audio files from the folder attached it will not have any impact as the same files will be created in the path given of the current folder in which the cep.m is.



```matlab
% Get the current folder where the script is located
currentFolder = fileparts(mfilename('fullpath'));

% Update the audio file paths to be relative to the script location
originalAudioFile = 'original.wav';
noiseAudioFile = 'noise.wav';
noisyAudioFile = 'noisy_output.wav';
denoisedAudioFile = 'denoised_output.wav';
furtherDenoisedAudioFile = 'further_denoised_output.wav';

% Full file paths
originalAudioFileFull = fullfile(currentFolder, originalAudioFile);
noiseAudioFileFull = fullfile(currentFolder, noiseAudioFile);
noisyAudioFileFull = fullfile(currentFolder, noisyAudioFile);
denoisedAudioFileFull = fullfile(currentFolder, denoisedAudioFile);
furtherDenoisedAudioFileFull = fullfile(currentFolder, furtherDenoisedAudioFile);
% Load pre-recorded audio signal
[originalSignal, originalFs] = audioread(originalAudioFileFull);
[noiseSignal, noiseFs] = audioread(noiseAudioFileFull);
```

4.2 Signal Alignment and Noise Addition

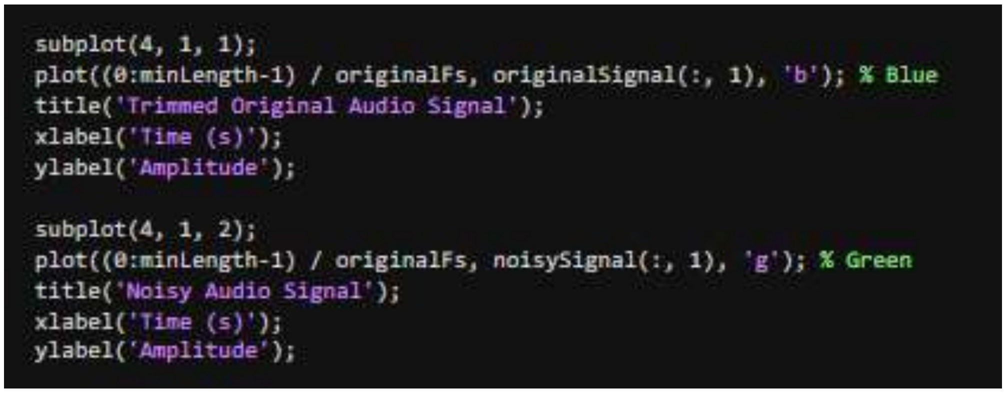
Signals are aligned to the minimum length, and scaled noise is added to the original signal to create the noisy signal. Due to this, the original signal is trimmed to match the signal of our noise.



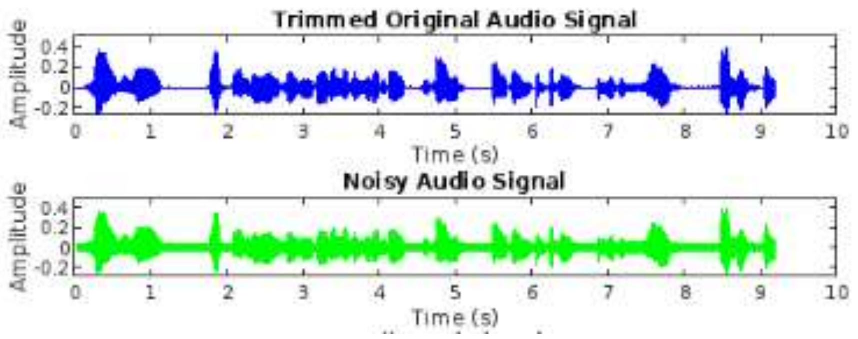
```matlab
% Trim both signals to the length of the smaller one
minLength = min(length(originalSignal), length(noiseSignal));
originalSignal = originalSignal(1:minLength, :);
noiseSignal = noiseSignal(1:minLength, :);

% Adjust the volume of the noise signal (you can change the scaling factor)
scaledNoise = 0.5 * noiseSignal;

% Add noise to the original signal
noisySignal = originalSignal + scaledNoise;

% Save the noisy audio signal
audiowrite(noisyAudioFileFull, noisySignal, originalFs);
```

4.3 Plotting Original, Noisy Signal:

The script generates a comparison plot with subplots for the trimmed original audio and noisy audio signals.

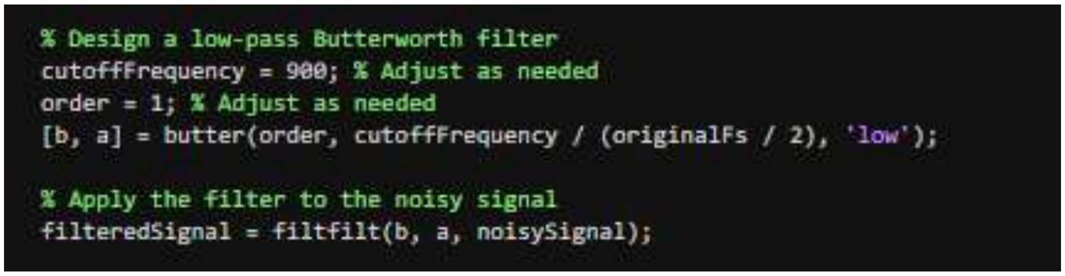


```
subplot(4, 1, 1);
plot((0:minLength-1) / originalFs, originalSignal(:, 1), 'b'); % Blue
title('Trimmed Original Audio Signal');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(4, 1, 2);
plot((0:minLength-1) / originalFs, noisySignal(:, 1), 'g'); % Green
title('Noisy Audio Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```



4.4 Designing and Applying Butterworth Filter

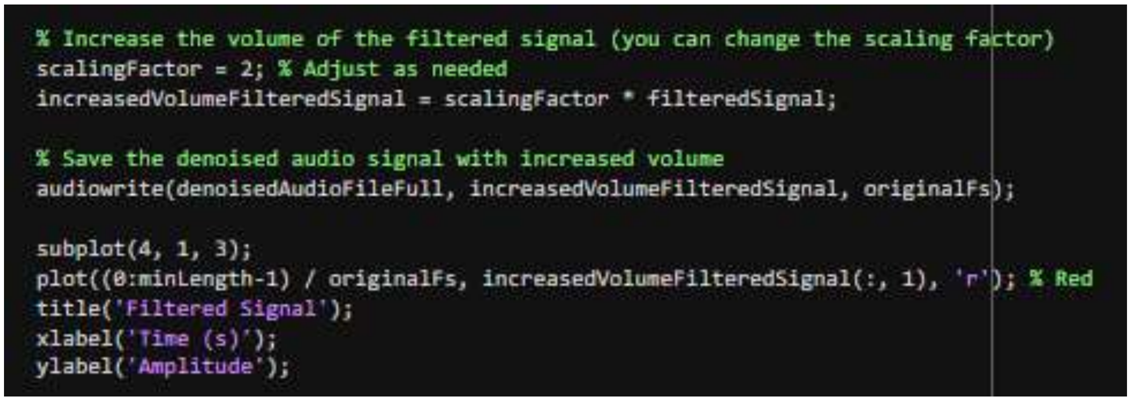
A low-pass Butterworth filter is designed and applied to the noisy signal using the `filtfilt` function for zero-phase filtering. With the cutoff-frequency = 900 Hz and order =1.



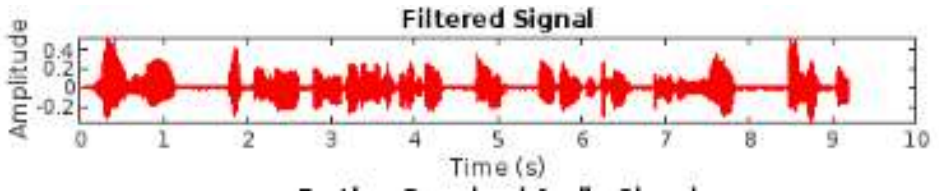
```
% Design a low-pass Butterworth filter
cutoffFrequency = 900; % Adjust as needed
order = 1; % Adjust as needed
[b, a] = butter(order, cutoffFrequency / (originalFs / 2), 'low');

% Apply the filter to the noisy signal
filteredSignal = filtfilt(b, a, noisySignal);
```

4.5 Volume Adjustment, Saving Denoised Audio and Plotting it

The volume of the filtered signal is increased, and the denoised signal is saved using `audiowrite`.

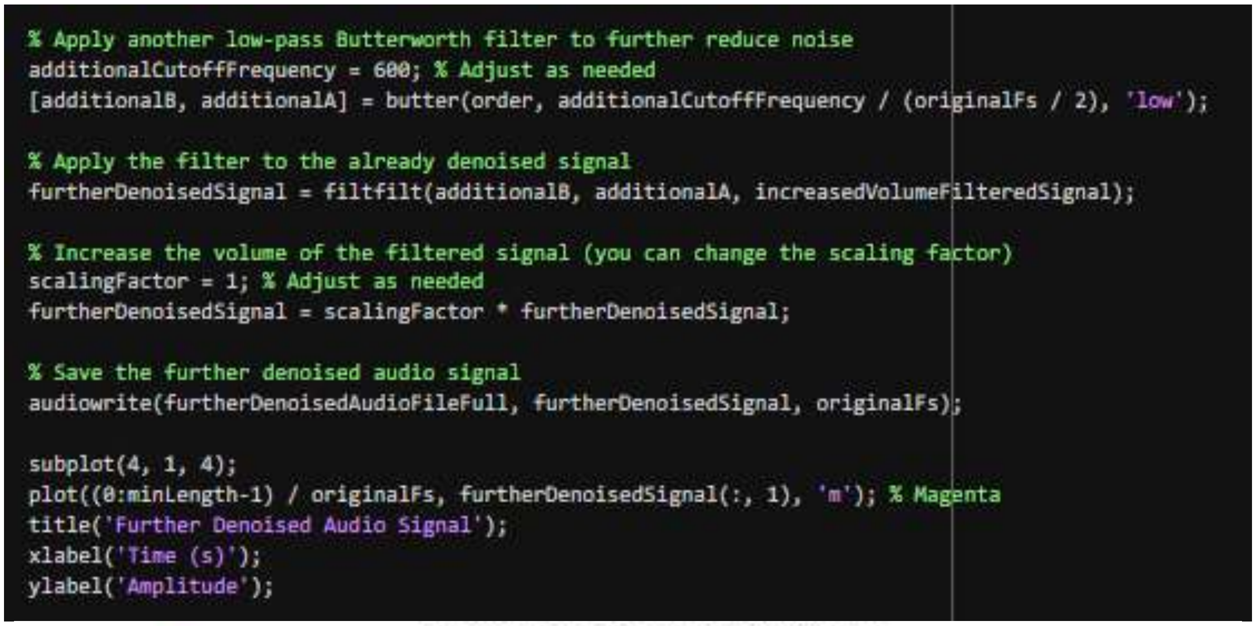


```
% Increase the volume of the filtered signal (you can change the scaling factor)
scalingFactor = 2; % Adjust as needed
increasedVolumeFilteredSignal = scalingFactor * filteredSignal;

% Save the denoised audio signal with increased volume
audiowrite(denoisedAudioFileFull, increasedVolumeFilteredSignal, originalFs);

subplot(4, 1, 3);
plot((0:minLength-1) / originalFs, increasedVolumeFilteredSignal(:, 1), 'r'); % Red
title('Filtered Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```

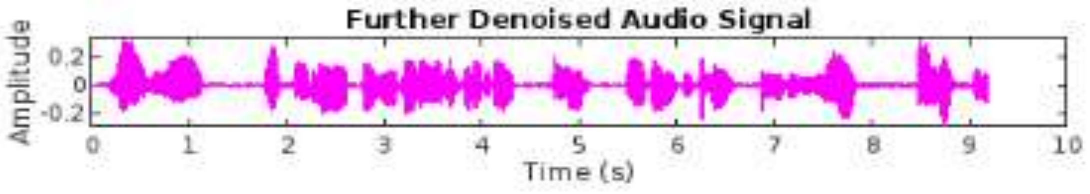


4.6 Additional Filtering and Further Denoising

Another low-pass Butterworth filter is designed and applied to the already denoised signal, followed by volume adjustment and saving the further denoised signal.

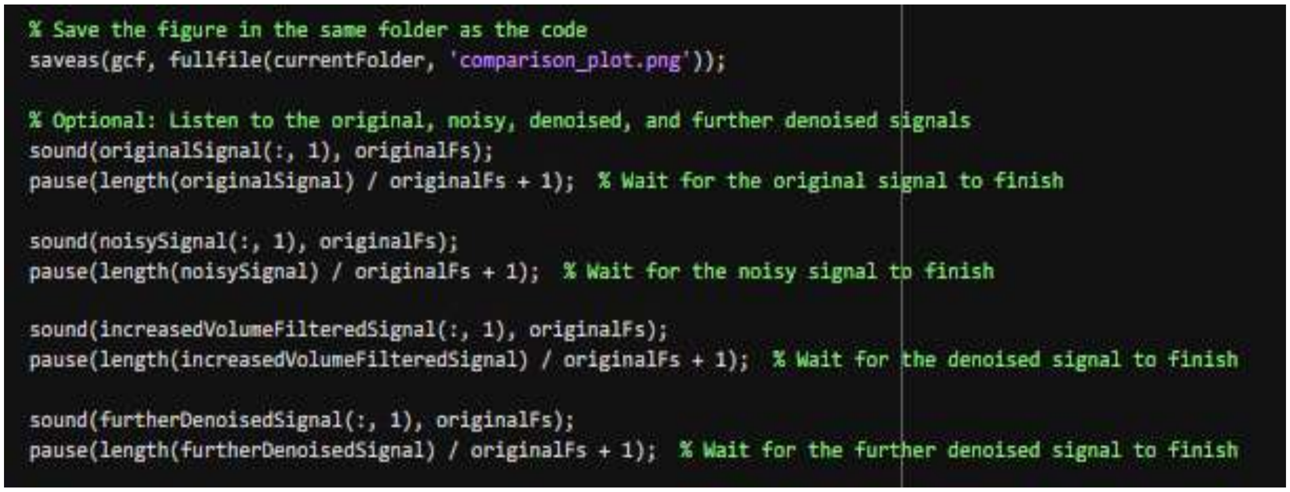


```
% Apply another low-pass Butterworth filter to further reduce noise
additionalCutoffFrequency = 600; % Adjust as needed
[additionalB, additionalA] = butter(order, additionalCutoffFrequency / (originalFs / 2), 'low');

% Apply the filter to the already denoised signal
furtherDenoisedSignal = filtfilt(additionalB, additionalA, increasedVolumeFilteredSignal);

% Increase the volume of the filtered signal (you can change the scaling factor)
scalingFactor = 1; % Adjust as needed
furtherDenoisedSignal = scalingFactor * furtherDenoisedSignal;

% Save the further denoised audio signal
audiowrite(furtherDenoisedAudioFileFull, furtherDenoisedSignal, originalFs);

subplot(4, 1, 4);
plot((0:minLength-1) / originalFs, furtherDenoisedSignal(:, 1), 'm'); % Magenta
title('Further Denoised Audio Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```



4.7 Saving Comparison Plot and Optional Listening

The comparison plot figure is saved as 'comparison_plot.png.' Optional listening is provided for the original, noisy, denoised, and further denoised signals.



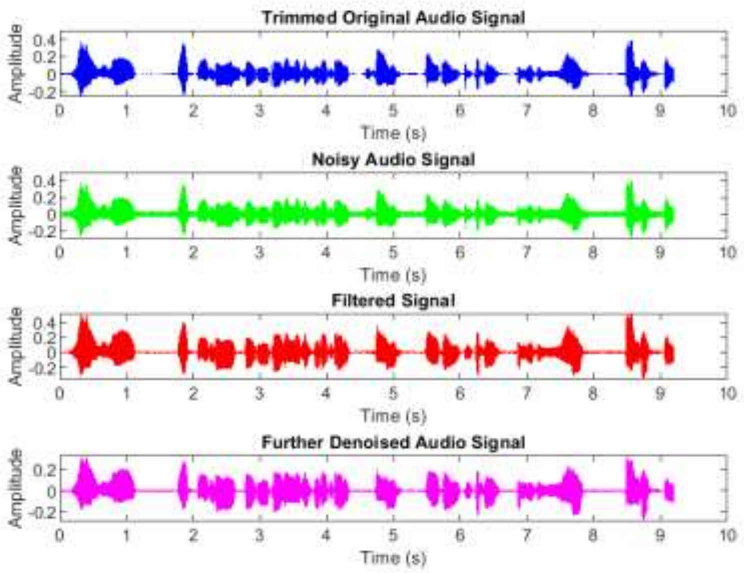
```
% Save the figure in the same folder as the code
saveas(gcf, fullfile(currentFolder, 'comparison_plot.png'));

% Optional: Listen to the original, noisy, denoised, and further denoised signals
sound(originalSignal(:, 1), originalFs);
pause(length(originalSignal) / originalFs + 1);  % Wait for the original signal to finish

sound(noisySignal(:, 1), originalFs);
pause(length(noisySignal) / originalFs + 1);  % Wait for the noisy signal to finish

sound(increasedVolumeFilteredSignal(:, 1), originalFs);
pause(length(increasedVolumeFilteredSignal) / originalFs + 1);  % Wait for the denoised signal to finish

sound(furtherDenoisedSignal(:, 1), originalFs);
pause(length(furtherDenoisedSignal) / originalFs + 1);  % Wait for the further denoised signal to finish
```

---

5. Analysis

5.1 Signal Comparison

The generated comparison plot visually displays the trimmed original audio, noisy audio, filtered audio, and further denoised audio signals. Observations indicate effective noise reduction and enhanced speech clarity.



5.2 Observations

Noise Reduction:
The initial filter significantly reduces noise up to a great extent but still leaves a bit little of noise behind. Now if we decreased the cutoff frequency, it was cutting our original signals also.

Further Denoising:
The second filter provides additional noise reduction, enhancing speech clarity.

Volume Adjustment:
Scaling factors impact perceived loudness, ensuring audibility without distortion.

---

6. Deliverables

1. Speech Audio Files:
- Original Audio (original.wav) → input
- Noise Audio (noise.wav) → input
- Noisy Output (noisy_output.wav) → output
- Denoised Output (denoised_output.wav) → output
- Further Denoised Output (further_denoised_output.wav) → output

2. Complete Working Code: MATLAB script provided.

3. Project Report: This comprehensive document addresses all requirements.

4. Comparison Plot: Saved as 'comparison_plot.png.'

---

7. Conclusion

The DSP-based speech enhancement system successfully removes unwanted noise, significantly improving the quality of the speech signal. Utilizing a two-stage filtering process and adjusting signal volumes allows for substantial noise reduction while preserving the clarity of the original speech. The flexibility to adjust parameters ensures adaptability to different audio characteristics. The provided code, accompanied by a detailed analysis, serves as a valuable guide for implementing similar speech enhancement systems.
