Audio Denoising using Short-Time Fourier Transform

Prateek Singhal, m22aie215@iitj.ac.in | Prabha Sharma, m22aie224@iitj.ac.in | Ashish Rawat, m22aie201@iitj.ac.in | Harsh Parashar, m22aie210@iitj.ac.in | Amit Panwar, m22aie202@iitj.ac.in

Contents

1 Abstract

Audio denoising is a critical process in various applications, including speech enhancement, music production, and sound analysis. We have utilized the Short-Time Fourier Transform (STFT) method for audio denoising, which efficiently analyses and processes audio signals in the time-frequency domain. We would be applying STFT to the noisy audio signal, and then decomposing it into its constituent frequencies across small time windows. In this report, we also explore some variations of Fourier Transform that helped us gain a better understanding of STFT. We would look into the principles of STFT, its advantages in analyzing time-varying signals, and effectiveness in identification and removal of noise components.

2 Introduction

2.1 Fourier Transform

The Fourier Transform is a mathematical technique that converts signals between two distinct domains, such as converting a given signal from frequency domain to the time domain or vice versa. Fourier transform is utilized in various fields such as signal processing and RADAR. Fourier showed that any given signal can be expressed as a series of sine waves with different amplitudes and phases. For example, in below figure, we can observe that first signal is sum of 3 different sine waves.

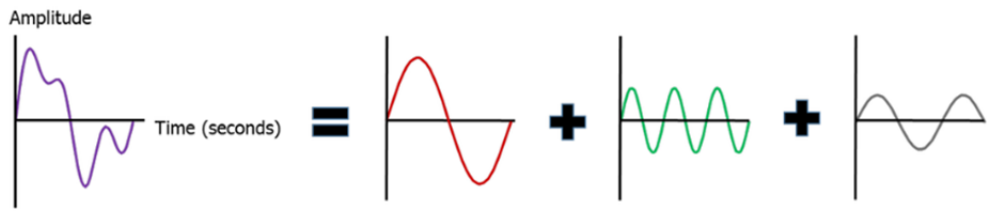


Figure 1: Signal with its constituent waves, *Reference: https://community.sw.siemens.com/s/article/what-is-the-fourier-transform*

Properties
Below are some important properties of Fourier Transform:

- Duality: If a(t) has a Fourier transform A(f), then the Fourier transform of A(t) is a(f).
- Linearity: The Fourier Transform of a sum of functions is the sum of their individual Fourier Transforms. If f(t) and g(t) are functions, and a and b are constants, then:

$$\mathcal{F}\{af(t) + bg(t)\} = a\mathcal{F}\{f(t)\} + b\mathcal{F}\{g(t)\}$$

- Modulation: Multiplying two functions in the time domain results in the convolution of their Fourier Transforms in the frequency domain:

$$\mathcal{F}\{f(t)g(t)\} = \frac{1}{2\pi}(F(\omega) * G(\omega))$$

- Time Scaling: If a function is compressed or stretched in time, its Fourier Transform is inversely scaled in frequency. If f(t) has a Fourier Transform $F(\omega)$, then for a scaling factor a:

$$\mathcal{F}\{f(at)\} = \frac{1}{|a|}F\left(\frac{\omega}{a}\right)$$

- Frequency Shifting: Multiplying a function by a complex exponential in the time domain results in a frequency shift. If f(t) has a Fourier transform $F(\omega)$, then:

$$\mathcal{F}\{f(t)e^{j\omega_0 t}\} = F(\omega - \omega_0)$$

Fourier Transform Conditions
The Fourier transform of a function f(t) can only be calculated if it meets the following conditions:

1. Finite Discontinuities: The function has a limited number of points where it abruptly changes value.
2. Finite Maxima and Minima: There are only a finite number of peaks and valleys within the function.
3. Absolute Integrability: The integral of the absolute value of the function over its entire domain is finite, its area is bounded.

Forward Fourier Transform
The forward Fourier transform is a mathematical operation that breaks down a time-domain function into a series of sine and cosine functions with varying frequencies. This method is mostly used in signal processing to check the frequency components of a signal. The forward Fourier transform is denoted by F(k), with symbol $\hat{f}(k)$. It is defined as:

$$\hat{f}(k) = \int\limits_{-\infty}^{\infty} f(x)e^{-2ikx}dx$$

Inverse Fourier Transform
The inverse Fourier transform converts a given signal from its frequency domain representation back into the time domain. It is the opposite operation of the forward Fourier transform. The inverse Fourier transform is denoted by f(x), and its symbol is $f(x)$. It is defined as follows:

$$\text{f(x)} = \int\limits_{-\infty}^{\infty} F(k)e^{2ikx}dk$$

2.2 Discrete Fourier Transform

The discrete Fourier transform (also known as DFT) is used to analyze a finite discrete-time signal and a finite set of frequencies. It is a way to analyze data points that are collected at specific and equally spaced time periods. DFT is like the continuous Fourier transform, but for data that's not continuous. While the continuous Fourier transform provides accurate results, its real-time application is limited due to the availability of discrete data collected through sensors and analog-to-digital converters. Also, the continuous Fourier transform is usually applied to signals existing over an infinite time domain, from $-\infty$ to $+\infty$. Whereas, the DFT is applied over a given finite time interval, called a window (a time period T), rather than for an infinite time span if the signal is periodic.
Since the window contains a finite number of data points, the DFT treats the signal as periodic, meaning that f(N) through f(2N-1) is equivalent to f(0) through f(N-1). The DFT can be expressed in complex exponential form as follows:

$$\mathrm{X(k)} = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn}, \quad k = 0, 1, 2, \ldots, N-1$$

where,

X(k) is the DFT of the sequence x(n),

x(n) is the discrete-time input sequence,

N is the total number of samples,

k is the frequency index,

n is the time index,

$e^{-j\frac{2\pi}{N}kn}$ represents the complex exponential basis functions

2.3 Discrete Time Fourier Transform

The Discrete Time Fourier Transform (also called DTFT) is a tool to analyze the frequency content of an aperiodic, discrete-time signal, and is closely related to the Discrete Fourier Transform (DFT).

If we have a signal with N samples, the DFT breaks this signal down into N/2 + 1 sine and cosine waves, that are evenly spaced in frequency. By adding zeros to the end of your signal (known as zero-padding), we can effectively extend its duration. This makes the frequency resolution finer, such that we can distinguish between frequencies that are closer together.

As we keep adding more zeros, the signal becomes more like an infinite-length signal. In the limit, as N approaches towards infinity, the signal becomes aperiodic, and the frequency spectrum becomes continuous. This continuous frequency spectrum is represented by DTFT.

Essentially, the DTFT is the Fourier transform for aperiodic, discrete-time signals, that provides a continuous frequency representation.

Below equation shows how to find the DTFT of a discrete-time signal x(n):

$$\mathrm{X}(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$

and, the inverse DTFT equation reconstruct the signal x(n):

$$\mathrm{x(n)} = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\omega)e^{j\omega n}d\omega$$

2.4 Fast Fourier Transform

The Fast Fourier Transform (also called FFT) is very efficient algorithm for calculating the Discrete Fourier Transform (DFT). Unlike the DFT, which can be computationally expensive for very large datasets, the FFT uses the algebraic properties of the DFT matrix to significantly reduce the number of necessary calculations.

The key to the FFT's efficiency is in its ability to exploit redundancies within the DFT calculations. The traditional DFT involves multiple repetitive multiplications, which can be time-consuming. The FFT reorganizes these calculations in a manner that minimizes the number of these redundant operations.

The FFT also uses a divide and conquer strategy to break down a larger problem into smaller sub-problems. This approach involves recursively dividing the input data into smaller subsets until the subsets become so small that could be easily computed.

The FFT achieves its speed by factorizing the DFT matrix into several sparse matrices. These sparse matrices would contain many zero entries, that could significantly reduces number of required calculations. The factorization process takes benefit of the periodic nature of the sine and cosine functions used in the DFT.

The computational complexity of the DFT is $O(N^2)$, indicating that the number of operations grows quadratically with the size of the input data (N). Compared to this computational complexity of DFT, the FFT has a computational complexity of O(N * log2(N)). This would mean that as the size of the input data increases, the FFT becomes increasingly more efficient compared to the DFT.

Benefits of the FFT:

- Speed: The FFT is significantly faster than the traditional DFT, especially for large datasets.
- Reduced Computational Errors: Fewer calculations mean a lower chance of programming errors.
- Practicality: The FFT's efficiency makes it a practical tool for many applications, including signal processing, image processing, and scientific computing.

In summary, the FFT is a powerful algorithm that has revolutionized the field of digital signal processing. By using the algebraic properties of the DFT and using a divide-and-conquer approach, the FFT provides a significant speed increase over the traditional DFT, making it a valuable tool for a wide range of applications.

3 Short Time Fourier Transform

The Short-Time Fourier Transform (also called STFT) was introduced to address the limitations of the Fast Fourier Transform (FFT). It is usually used to extract a narrow-band frequency content in noisy signals. The core concept of STFT divides the original signal into small time windows and applies the Fourier Transform to each segment, allowing to analyse how the signal's frequency content changes over time within each window as showed in Figure 2.

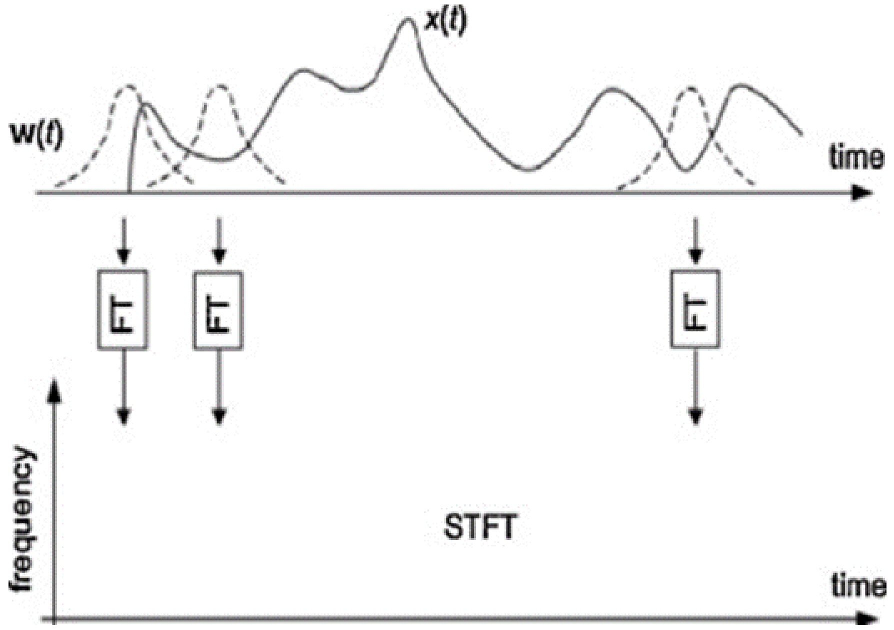


Figure 2: Short-Time Fourier Transform (STFT), *Reference: https://www.sciencedirect.com/topics/engineering/short-time-fourier-transform*

This tool helps us find different parts of a signal that repeat at regular intervals, even if the signal is changing over time. It can show us these parts irrespective of their frequency. It breaks down the signal into smaller pieces and analyzes each piece to find the repeating patterns. The mathematical equation for the short time Fourier Transform is given by:

$$\mathrm{X}(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t-\tau) e^{-j\omega t} dt$$

where:

- $x(t)$ is the input signal,
- $w(t-\tau)$ is the sliding window function centered at τ,
- $e^{-j\omega t}$ is the complex exponential for the frequency ω,
- τ is the time-shift parameter.

and, inverse of STFT is given by:

$$\mathrm{x(t)} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} X(\omega, \tau) \cdot w(t-\tau) \cdot \exp(j \cdot \omega t)\, d\omega\, d\tau$$

One of the limitations of STFT is its low resolution, especially when a small window is chosen, which fails to fully capture the dynamic behavior of the structure. The window size controls the trade-off between time and frequency accuracy,that is a larger window improves frequency resolution but reduces time resolution, and vice versa. Moreover, it is difficult to resolve closely spaced natural frequencies using this method.

3.1 Key Considerations in Using the STFT

When we use the Short-Time Fourier Transform (STFT) for audio signal analysis, there are many important factors that needs to be considered. These factors can significantly impact the quality of the time-frequency analysis and the trade-offs between time and frequency resolution.

- **Width of the Analysis Window**
  The STFT involves a trade-off between time and frequency resolution, influenced by the selected window length and shape. When you use this tool, you have to choose between seeing the signal in more detail over time or in more detail in terms of its different frequencies.

  When using a short window, the STFT provides greater time resolution, allowing to capture quick changes in the audio signal. This is especially beneficial for analyzing quickly changing

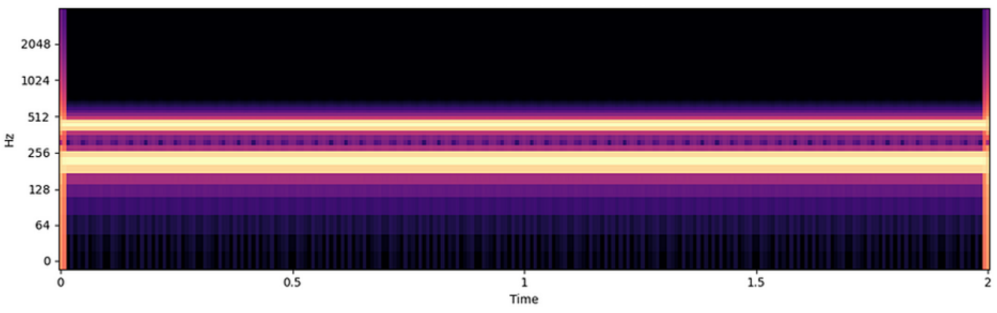


Figure 3: Wideband Analysis (Short Window), *Reference: https://medium.com/@ongzhixuan/exploring-the-short-time-fourier-transform-analyzing-time-varying-audio-signals-98157d1b9a12*

sounds, such as speech. However, the downside of a short window is that it would lead to lower frequency resolution, making it difficult to differentiate between closely spaced frequency components.

This tool can help you pinpoint exactly when something happens in a signal, but it might not give you a very clear picture of the different frequencies that make up that event.

When a narrow window is used, the STFT is able to capture quick changes in the signal's frequency content, as the window is short in duration, allowing the signal to be localized more precisely in time.

Advantages:

- High time resolution: We can detect short, or rapid changes in the signal.
- Ideal for non-stationary signals: Signals where the frequency components change quickly (e.g. impulses, or fast-varying signals) are better captured with a short window.
- Accurate time localization: We can also find instances when specific frequency change occurs.

Disadvantages:

- Low frequency resolution: As fewer cycles of the signal are captured within a short window, distinguishing between closely spaced frequency components becomes more difficult.
- Blurring of frequency information: Frequency components can overlap, making it harder to identify individual frequencies.

Example:
Consider a signal where two closely spaced frequency components (e.g., 50 Hz and 52 Hz) appear for a short duration. A narrow window will capture the time variation of these components accurately but will have difficulty distinguishing between the two frequencies.

A long window offers improved frequency resolution, allowing for the identification of small frequency differences in the audio signal. This is advantageous when analyzing steady-state sounds, such as continous musical notes or background noise. However, the trade-off is that a long window decreases time resolution, making it less effective for capturing rapidly changing events in the audio signal.

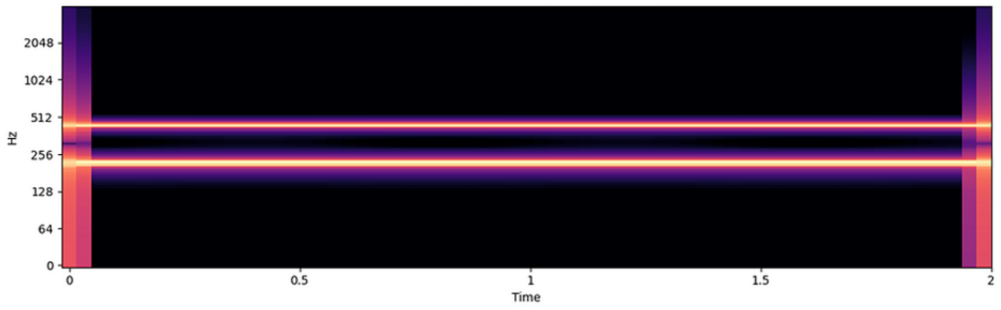


Figure 4: Narrowband Analysis (Long Window), *Reference: https://medium.com/@ongzhixuan/exploring-the-short-time-fourier-transform-analyzing-time-varying-audio-signals-98157d1b9a12*

So, this tool would help us distinguish between different frequencies that are close together, but it might not be very good at showing us when the frequencies change quickly.
When a wide window is used, the STFT focuses on resolving the signal's frequency content more accurately over a longer time period.

Advantages:

- High frequency resolution: A longer window allows for more cycles of the signal to be captured, providing a more accurate frequency measurement.
- Better for stationary signals: If the signal is more or less constant over time or varies slowly, a wide window provides better frequency resolution.
- Distinguishing between close frequencies: For signals that are closely spaced frequencies, a wide window would allow these frequencies to be more easily separated.

Disadvantages:

- Low time resolution: Since the window spans a longer time interval, it becomes harder to find exactly when a change in the signal could occurs.
- Miss transient events: Rapid changes or short-lived components may be blurred or missed entirely, as they may occur within a single wide window.

Example:
If the signal contains two frequencies (e.g., 50 Hz and 52 Hz) over a long-time span, a wide window will easily distinguish these frequencies. However, if the signal contains a short impulse or transient event, it will be smoothed out by the wide window, losing precise time information.

The choice of how to analyze a signal depends on what we want to know about it. Sometimes we need to see more detail over time, and sometimes we need to see more detail in terms of frequencies. We might have to try different ways of analyzing the signal to find the best one for our purpose.

- **Position of the Windows (Overlap)**
  The degree of overlap between consecutive windows can significantly affect the quality of the STFT analysis. Overlapping windows can mitigate the loss of information at the edges, giving a more continuous representation of the time-frequency characteristics of the audio signal. The extent of overlap varies based on the specific application, with typical values ranging from 50% to 75%. However, increased overlap may lead to greater computational complexity, as more windows must be processed.

3.2 Applications of STFT:

- Speech and Audio Processing: STFT is widely used to analyze audio signals, where frequency content changes rapidly. It is fundamental in speech recognition and music processing.
- Time-Frequency Analysis of Signals: Signals like radar, sonar, and EEG are time-varying, making STFT useful for analyzing how their frequency content changes over time.
- Image Processing: STFT is also used in image analysis to detect features that vary spatially.
- Communication Systems: In systems where signals are transmitted over time-varying channels, STFT helps in tracking changes in the channel characteristics.

4 Denoising Process

4.1 Problem Statement:

In signal processing, noise is often defined as any unwanted disturbances (or irrelevant signals) that obscure or distort the original signal. This can arise from various sources, including environmental factors (like wind or traffic), electronic interference, or physical limitations of sensors.

Audio denoising is a crucial process in signal processing that aims to improve the quality of audio signals by removing unwanted noise. The goal of denoising is to reconstruct a clean audio signal that closely resembles the original, uncorrupted version.

4.2 Adaptive Thresholding

Adaptive thresholding is a technique used to separate the signal from noise by setting a dynamic threshold based on the local noise estimate. This allows for better performance in varying noise conditions:

$$\text{threshold}(x) = \begin{cases} x & \text{if } x > \text{factor} \cdot \text{noise\_floor} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

4.3 Wiener Filter

The Wiener filter is a linear filter that minimizes the mean square error between the estimated and true signals. It uses the ratio of the signal power to the total power (signal plus noise) to determine the optimal gain:

$$\text{wiener\_gain} = \frac{\text{signal\_power}}{\text{signal\_power} + \text{noise\_power} + \epsilon} \tag{2}$$

The denoised output using Wiener filtering is given by:

$$\text{denoised\_audio} = \text{noisy\_audio} \times \text{wiener\_gain} \tag{3}$$

4.4 Smoothing

Smoothing is often applied to remove residual noise and fluctuations in the audio signal. A simple moving average filter can be expressed as:

$$\text{smoothed}(n) = \frac{1}{W} \sum k = 0^{W-1} \text{audio}(n-k) \tag{4}$$

where W is the window size.

4.5 Block Thresholding

Block thresholding combines the above techniques by applying them in short time frames or blocks. The procedure can be summarized as:

$$\text{denoised\_audio} = \text{apply\_smoothing}\left(\text{istft}\left(\text{denoised\_magnitude} \cdot e^{j \cdot \text{angle(stft\_noisy)}}\right)\right) \tag{5}$$

5 Evaluation of Denoising

5.1 Downsampling

The downsampling process is essential for evaluating the denoising performance using specific metrics. It can be defined as:

$$\text{downsample}(x) = x\text{target} \quad \text{where } x\text{target is calculated based on } x\text{original\_sr} \tag{6}$$

5.2 Evaluation Metrics

The evaluation metrics used to assess the quality of denoising include:

- Signal-to-Noise Ratio (SNR):

$$\text{SNR} = 10 \cdot \log_{10}\left(\frac{\text{Powerclean}}{\text{Powererror}}\right) \tag{7}$$

- Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_{\text{clean}}[i] - x_{\text{noisy}}[i])^2 \tag{8}$$

- Peak Signal-to-Noise Ratio (PSNR):

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{(2^b - 1)^2}{\text{MSE}}\right) \tag{9}$$

  where b is the number of bits per sample.

6 Sample Input and Calculation

6.1 Sample Input

Let us define the following audio signals:

$$\begin{aligned} \text{Clean Audio Signal:} \quad & \text{clean\_audio} = [1, 2, 3, 4, 5] \\ \text{Noisy Audio Signal:} \quad & \text{noisy\_audio} = [1.1, 1.9, 2.8, 4.1, 5.2] \end{aligned}$$

6.2 Signal-to-Noise Ratio (SNR) Calculation

Power of Clean Audio:
The power of the clean signal is the sum of the squares of its values:

$$\text{Power}_{\text{clean}} = \sum(\text{clean\_audio})^2 = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55 \tag{10}$$

Error Signal for Noisy Audio:
The error signal is the difference between the clean audio and noisy audio:

$$\text{error\_noisy} = \text{clean\_audio} - \text{noisy\_audio} = [0, 0.1, 0.2, -0.1, -0.2] \tag{11}$$

Power of Error for Noisy Audio:

The power of the error signal is the sum of the squares of the error values:

$$\text{Power}_{\text{error\_noisy}} = \sum(\text{error\_noisy})^2 = 0^2 + 0.1^2 + 0.2^2 + (-0.1)^2 + (-0.2)^2 = 0.1 \tag{12}$$

SNR for Noisy Audio:
Substitute the values in above mentioned equation (7):

$$\text{SNRnoisy} = 10 \cdot \log 10 \left(\frac{55}{0.1}\right) \approx 27.4 \text{ dB} \tag{13}$$

6.3 Denoising Using Wiener Filter

Estimate Noise Power:
Given the noise estimate as 0.1, the noise power is:

$$\text{noise\_estimate} = 0.1, \quad \text{noise\_power} = 0.1^2 = 0.01 \tag{14}$$

Calculate Signal Power:
The signal power of the noisy audio is:

$$\text{signal\_power} = \sum(\text{noisy\_audio})^2 \approx 56.51 \tag{15}$$

Wiener Gain:
The Wiener gain is calculated using equation (2). Assuming ϵ is small, the Wiener gain becomes:

$$\text{wiener\_gain} \approx 0.999 \tag{16}$$

Apply Wiener Filter:

$$\text{denoised\_audio} \approx [1.099, 1.899, 2.799, 4.099, 5.199] \tag{17}$$

6.4 SNR for Denoised Audio

Error Signal for Denoised Audio:
The error signal for the denoised audio is:

$$\text{error\_denoised} \approx [-0.099, 0.101, 0.201, -0.099, -0.199] \tag{18}$$

Power of Error for Denoised Audio:
The power of the error signal for the denoised audio is:

$$\text{Power}_{\text{error\_denoised}} \approx 0.109 \tag{19}$$

SNR for Denoised Audio:

The SNR for the denoised audio is:

$$\text{SNR}_{\text{denoised}} \approx 27.02 \text{ dB} \tag{20}$$

7 Execution Analysis and Results

Below are the results of the program execution. With some sample audio files the graph shows how the original signal, signal with added noise and the denoised signal compared.

- **The original signal** represents the clean audio data, which serves as the baseline for evaluation. This signal typically has a smooth waveform, indicative of the true sound without any interference.
- **The signal with added noise** illustrates the impact of random disturbances or unwanted signals that obscure the original audio. This can manifest as fluctuations and irregular patterns in the waveform, making it difficult to discern the original sound.
- **The denoised signal** demonstrates the effectiveness of the noise reduction techniques applied to the noisy audio. The waveform should resemble the original signal more closely, indicating that the unwanted noise has been significantly reduced or eliminated while preserving the integrity of the underlying audio content.

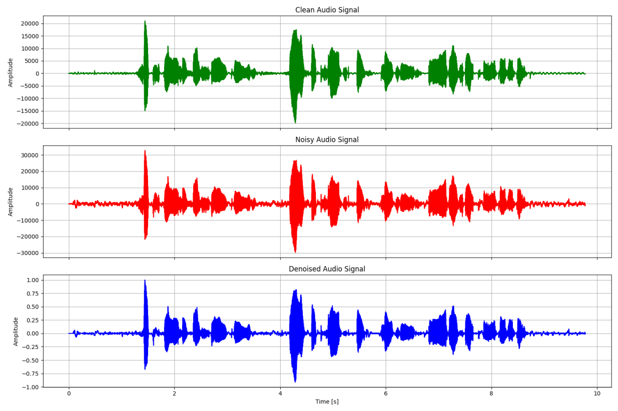
By comparing these three signals, we can assess the performance of the noise reduction algorithm, examining how well it restores the audio quality and enhances clarity for better listening experiences.

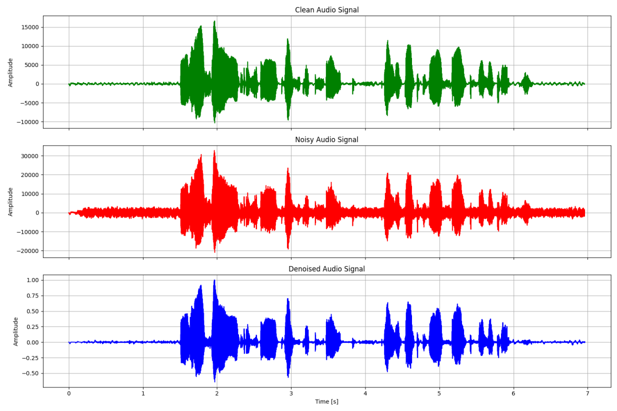
7.1 Denoising Results

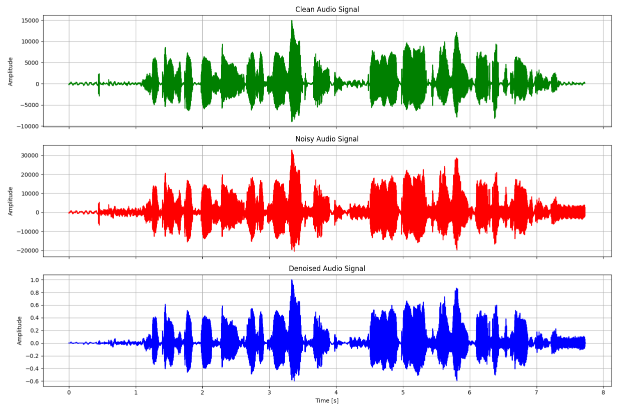
The dataset contains various audio metrics for six files that have undergone denoising. Key metrics include SNR (Signal-to-Noise Ratio), PESQ (Perceptual Evaluation of Speech Quality), STOI (Short-Time Objective Intelligibility), MSE (Mean Squared Error), and PSNR (Peak Signal-to-Noise Ratio).

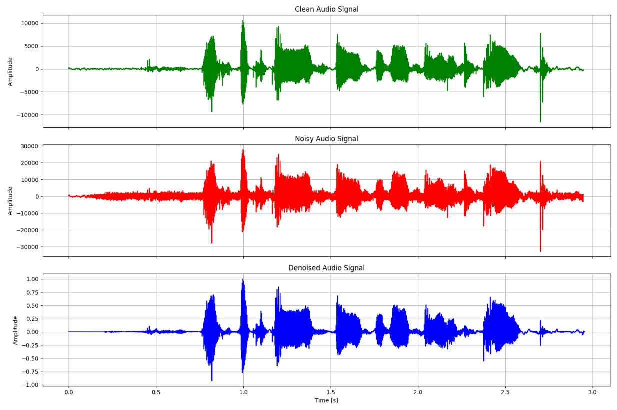
File	**SNR Noisy**	**SNR Denoised**	**PESQ Noisy**	**PESQ Denoised**	**STOI Noisy**	**STOI Denoised**	**MSE Noisy**	**MSE Denoised**	**PSNR Noisy**	**PSNR Denoised**
noised_p228_003.wav	3.32	-32.25	1.17	1.13	0.75	0.73	999.01	3598151.84	56.43	20.87
noised_p228_004.wav	5.97	-30.80	1.17	1.71	0.77	0.77	672.21	3195769.51	56.13	19.36
noised_p257_004.wav	2.81	-34.94	1.32	1.31	0.95	0.94	1111.79	6627045.69	57.71	19.96
noised_p257_002.wav	3.04	-27.90	1.39	1.34	0.95	0.95	1162.97	1441862.45	49.89	18.96
noised_p257_003.wav	1.17	-32.67	3.09	1.72	0.95	0.93	1461.65	3537393.54	51.00	17.16
noised_p228_005.wav	3.67	-32.85	1.30	1.22	0.82	0.80	751.38	3378909.27	54.74	18.21
Average	3.33	-31.90	1.57	1.41	0.86	0.85	1026.50	3629855.38	54.32	19.09

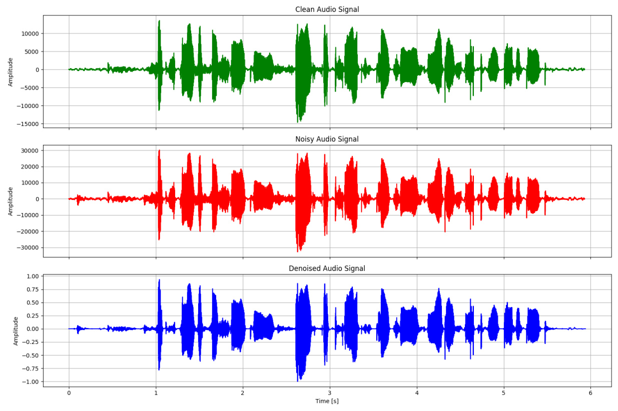
Table 1: Denoising Results Summary

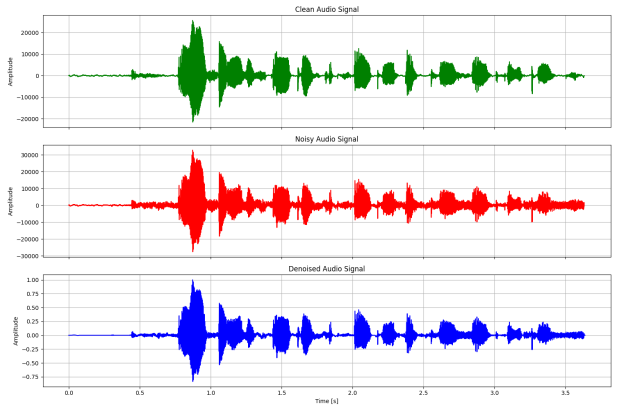

Clean Audio Signal
Noisy Audio Signal
Denoised Audio Signal
Amplitude
Time [s]


Clean Audio Signal
Noisy Audio Signal
Denoised Audio Signal
Amplitude
Time [s]


Clean Audio Signal
Noisy Audio Signal
Denoised Audio Signal
Amplitude
Time [s]


Clean Audio Signal
Noisy Audio Signal
Denoised Audio Signal
Amplitude
Time [s]


Clean Audio Signal
Noisy Audio Signal
Denoised Audio Signal
Amplitude
Time [s]


Clean Audio Signal
Noisy Audio Signal
Denoised Audio Signal
Amplitude
Time [s]

7.2 Detailed Analysis

Based on the evaluation results of your denoising code, here's a detailed analysis:

- **SNR (Signal-to-Noise Ratio):** The average SNR of the noisy audio is about 3.33 dB. After denoising, it drops to -31.90 dB, indicating that while noise is present, the cleaning process did not significantly improve the signal quality.
- **PESQ (Perceptual Evaluation of Speech Quality) Noisy and Denoised:** The average PESQ score decreased from 1.57 in the noisy audio to 1.41 in the cleaned audio. This suggests that while there was a slight improvement in quality, some distortion remains noticeable.
- **STOI (Short-Time Objective Intelligibility) Noisy and Denoised:** The average STOI score changed slightly from 0.86 to 0.85, indicating that some speech clarity was preserved despite the noise.
- **MSE (Mean Squared Error) Noisy and Denoised:** The MSE increased from 1026.50 for the noisy audio to 3629855.38 for the denoised audio, suggesting that the denoising process introduced significant errors.
- **PSNR (Peak Signal-to-Noise Ratio) Noisy and Denoised:** The average PSNR dropped from 54.32 dB to 19.09 dB. This implies that while some noise was removed, the cleaning process may have introduced new issues that affected the overall audio quality.

7.3 Conclusion

- **The denoising algorithm** appears to perform poorly based on these metrics, particularly in terms of *SNR, **MSE, and **PSNR. The **STOI* values show that the denoising hasn't significantly reduced intelligibility, but the drastic decline in SNR and PSNR suggests that the denoised signal is heavily distorted or suppressed, which compromises overall quality.
- **The PESQ values** indicate that in some cases, the perceptual quality might have improved slightly after denoising, but overall, the algorithm is not consistently enhancing the quality of the speech signals.
- Next steps might include adjusting the denoising parameters or trying a less aggressive approach to balance noise reduction without overly degrading the signal quality. Further improvements could involve fine-tuning the model to prevent excessive suppression.