Waveform Analysis and Glitch Removal

Complex Engineering Activity

From

MUHAMMAD ZEESHAN	**210711**
MUHAMMAD ABDULLAH	**210779**
UMER ZAKI	**211909**
AYRA ABID	**210757**

To

ENGR. WANIA ANOOSH

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AIR UNIVERSITY, ISLAMABAD

Abstract

This project focuses on developing a MATLAB-based solution to analyze waveform data in electrical power systems, specifically addressing the challenges of frequency glitches and noise. The primary objectives include waveform analysis, accurate detection and removal of frequency glitches, and effective noise reduction. The solution utilizes advanced signal processing techniques to ensure real-time or near-real-time processing capabilities. Validation is conducted using both simulated and real-world data to confirm the reliability and effectiveness of the methods employed. The practical applications of this solution extend to power system monitoring, transformer and motor protection, preventive maintenance, smart grid integration, industrial automation, and research and development. The outcome of this project significantly contributes to enhancing the stability, safety, and efficiency of electrical power systems.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background

In electrical power systems, maintaining a stable frequency is crucial for the safe operation of equipment and appliances. In countries like Pakistan, the standard frequency is 50Hz. Deviations from this frequency can lead to equipment malfunction and potential damage.

Sudden changes in frequency can cause motors and transformers to draw excessive currents, leading to overheating and damage. Additionally, insulation failures may occur, resulting in short circuits and equipment failures.To address these risks, it's important to analyze waveform data to identify glitches and remove them from the signal.

The task at hand involves developing a MATLAB code to analyze waveform data, identify frequency glitches, and remove them. This code will help ensure the stability and reliability of the power system, minimizing the risk of equipment damage and power disruptions.

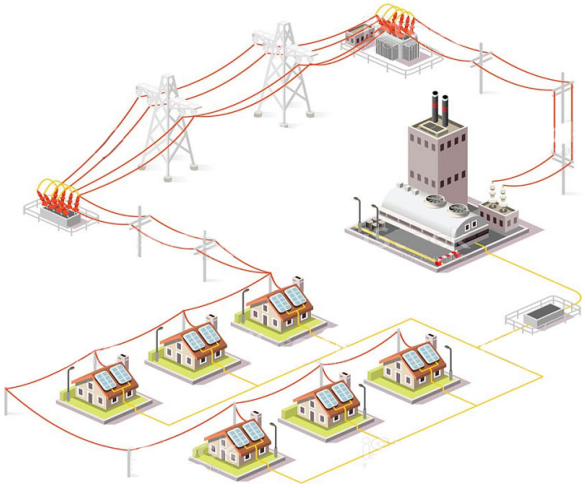

Figure 1.1: Electric-grid-station

1.2 Problem Statement

The frequency of generated voltage depends on the number of poles in an alternator and the speed of the generator's rotor. Pakistan operates on a 230V supply voltage and 50Hz frequency. All appliances are designed to be safe at 50Hz frequency. However, if the frequencies extend out of this range even for a few seconds, the effects can be catastrophic. At the power system generation level, a sudden frequency shift leads to more drastic and immediate consequences.

Motor and transformer currents vary inversely with frequency, so a sudden frequency drop could cause potentially damaging current overloading. Frequency spikes could cause sudden insulation failures, resulting in massive electrical short-circuit faults. The resulting damage could potentially cost taxpayers hundreds of millions of dollars, both in the short term and the long term, as equipment fails before its lifespan is reached.

Therefore, it is crucial to identify the sudden glitch in frequency and implement necessary actions to avoid the harmful effects discussed above. Consider any waveform file containing noise. There is a glitch in this waveform at a certain instant.

Write a MATLAB code that performs the following operations:

1. Plot the waveform with time.
2. Find the frequency of this waveform.
3. Find the precise time instant (hours, minutes, seconds) at which the glitch in frequency occurs.
4. Remove the glitch from the sinusoidal waveform.
5. Remove the noise from the sinusoidal waveform.

1.3 Objective

The primary objective of this project is to develop a MATLAB-based solution for analyzing waveform data in electrical power systems to address frequency glitches and noise. Specifically, the project aims to achieve the following objectives

1. **Waveform Analysis:** Develop MATLAB code to analyze waveform data for frequency glitches and noise.
2. **Glitch Detection:** Create algorithms to identify frequency glitches in the waveform data accurately.
3. **Glitch Removal:** Implement techniques to remove frequency glitches from the waveform data.
4. **Noise Reduction:** Apply signal processing methods to filter out noise from the waveform data.
5. **Real-time Processing:** Ensure the MATLAB code can process waveform data in real-time or near real-time.
6. **Validation:** Validate the effectiveness of the solution through testing with simulated and real-world data.

1.4 Applications

The MATLAB-based solution developed in this project has several practical applications in the field of electrical engineering and power systems:

1.4.1 Power System Monitoring

This solution enables continuous real-time monitoring of electrical grids to detect and mitigate frequency anomalies promptly. By analyzing waveform data, it ensures that

the power system operates within safe frequency ranges, thereby preventing potential disruptions and enhancing overall grid stability.

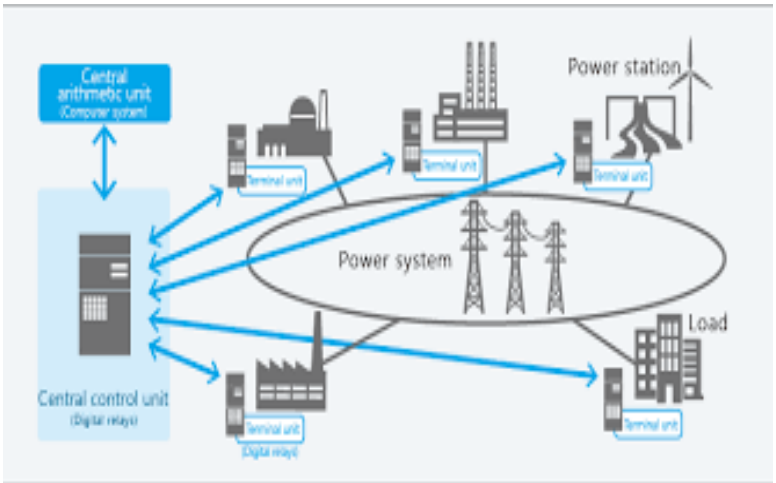

Figure 1.2: Power System Monitoring

1.4.2 Transformer and Motor Protection

Transformers and motors are sensitive to frequency fluctuations, which can cause overheating, insulation failure, and mechanical stress. The developed solution helps in protecting these critical components by identifying and correcting frequency glitches, thus extending their operational lifespan and reducing maintenance costs.

Figure 1.3: Transformer and Motor Protector

1.4.3 Smart Grid Systems

The solution can be integrated with smart grid technologies to improve grid resilience and reliability. By continuously monitoring the frequency and applying adaptive responses to changes, it ensures a stable power supply, accommodating the dynamic nature of modern electrical grids with distributed energy resources and variable loads.

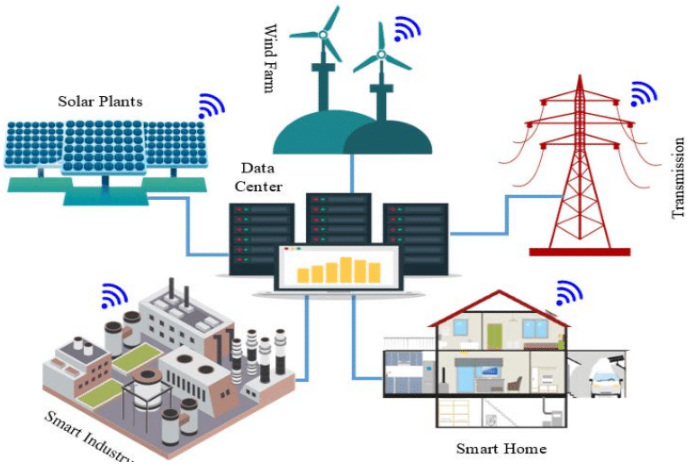

Figure 1.4: Smart Grid Systems

1.4.4 Industrial Automation

In industrial settings, maintaining a stable power supply is crucial for the smooth operation of machinery and production lines. The solution filters out noise and corrects frequency glitches in real time, preventing disruptions that can lead to costly downtime and production losses, thus enhancing productivity and efficiency.

Figure 1.5: Industrial Automation

1.4.5 Electrical Safety

Ensuring electrical safety is paramount in power systems. The solution promptly identifies frequency glitches and noise, preventing potential catastrophic failures such as insulation breakdowns and short circuits. This proactive approach reduces the risk of electrical hazards, protecting both equipment and personnel.

Figure 1.6: Safety First

Chapter 2

Methodology

2.1 Block Diagram

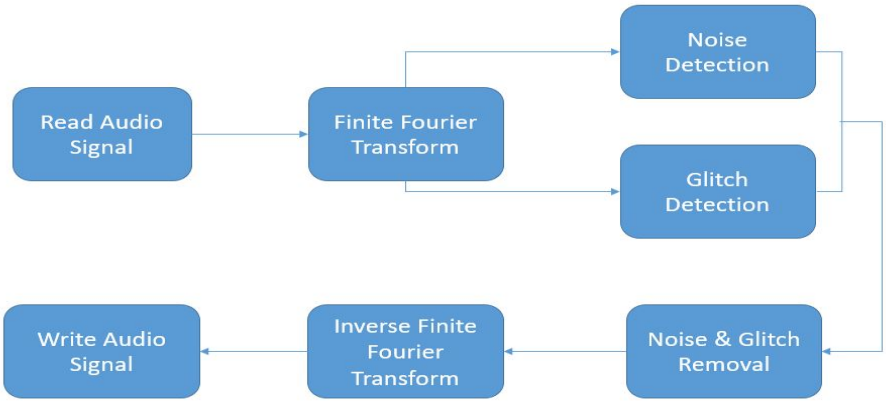

Figure 2.1: Block Diagram

2.2 Explaination

2.2.1 Reading Original Audio Signal

We use the `audioread` function in MATLAB to read the original audio signal from a specified file (`Noisy Signal.wav`).

MATLAB Code

```
[y, Fs] = audioread('U:\SEMESTER-6\DSP\PROJECT
\CODE\Noisy␣Signal.wav');
y = y(:,1);
info = audioinfo('U:\SEMESTER-6\DSP\PROJECT
\CODE\Noisy␣Signal.wav');
t = 0:(1/Fs):(info.Duration-1/Fs);
```

2.2.2 *Plotting the Original Signal*

The original audio signal is plotted in the time domain to visualize its waveform. Additionally, the number of samples is plotted to understand the length and amplitude distribution of the signal.

MATLAB Code

```
figure;
plot(t, y)
title('Audio␣Signal␣in␣Time␣Domain')
xlabel('t␣(secs)')
ylabel('y(t)')
```

The Signal is plotted in the time domain using the `plot` function. This creates a visual representation of the waveform, allowing us to observe its characteristics such as amplitude variations and overall shape.

2.2.3 *Fourier Transform*

The Fast Fourier Transform (FFT) is applied to the original signal to transform it from the time domain to the frequency domain. The single-sided amplitude spectrum is calculated and plotted to identify dominant frequencies.

MATLAB Code

```
% Fourier Transform (DFT) and Single-Sided Spectrum
Y = fft(y);
N = length(y);
P2 = abs(Y/N);
P1 = P2(1:(N/2)+1);
P1(2:end) = 2*P1(2:end);
f = Fs*(0:(N/2))/N;
[max_amp, idx] = max(P1);
```

The Fast Fourier Transform (FFT) is applied to the original signal (y) using the `fft` function. This transforms the signal from the time domain to the frequency domain.

2.2.4 Plotting the Signal in Frequency Domain

MATLAB Code

```
% Plot the single-sided amplitude spectrum
figure;
plot(f, P1)
title('Single-Sided␣Amplitude␣Spectrum␣of␣y(t)')
xlabel('freq␣(Hz)')
ylabel('Amplitude')
```

Before glitch detection, we visualize the frequency content of the original signal by plotting it in the frequency domain. This step provides insights into the spectral characteristics of the signal before further processing.

2.2.5 Glitch Detection

A glitch detection algorithm is implemented to identify sudden changes in the frequency domain. The time instant of the glitch in the audio signal is determined based on the detected changes.

MATLAB Code

```
% Glitch detection
[hours, minutes, seconds, glitch_indices] =
glitchdetection(Fs, y, t);
```

We implement a glitch detection algorithm by comparing the frequency at every index with our threshold frequency (50 Hz).

MATLAB Function

```
function [hours,minutes,seconds,glitchtime]=glitchdetection(Fs,y,t)
figure;
plot(t, y);
title('Original Signal with Glitches');
xlabel('Time (s)');
ylabel('Amplitude');
hold on;
glitchtime = [];
for i = 1:Fs:length(y)-Fs
    % Extract 1-second segment of the signal
    segment = y(i:i+Fs-1);
    Y = fft(segment);
    P2 = abs(Y/Fs);
    P1 = P2(1:Fs/2+1);
    P1(2:end) = 2*P1(2:end);
    f = Fs*(0:(Fs/2))/Fs;
    [max_amplitude, max_idx] = max(P1);
    peak_frequency = f(max_idx) ;
    % Set the threshold frequency for glitch detection
    threshold_frequency = 50; % Hz
    % Check if the peak frequency is within the threshold
    if peak_frequency < threshold_frequency
    || peak_frequency > threshold_frequency
        % Calculate time of glitch
        glitchtime = i / Fs;
        % Plot glitches
        plot(glitchtime, y(i), 'ro'); % Mark glitch with red circle
        % Display glitch time in hours, minutes, and seconds
        hours = floor(glitchtime / 3600);
        minutes = floor(mod(glitchtime, 3600) / 60);
        seconds = mod(glitchtime, 60);
        disp(time in sec)

    end
end
hold off;
end
```

When the frequency is above or below the Threshold frequency the glitch is marked. This algorithm also gives us the time in HOURS:MINUTES: and SECONDS at which the glitch is detected. now to display the glitch we only plot the signal at which time the glitches are shown.

MATLAB Code

```
figure;
A = (hours * 3600) + (minutes * 60) + seconds;
yg1 = ynewtime((A-50):(A+50), 1);
tg1 = t(1:length(yg));
plot(tg1, yg1);
title('Glitched Removed Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```

2.2.6 Noise Detection

The noisy part of the signal is analyzed to understand its characteristics.

MATLAB Code

```
yn = y(1:1800, 1);
tn = t(1:length(yn));
figure;
plot(tn, yn)
title('Noise Detection')
xlabel('t (secs)')
ylabel('y(t)')
```

To detect noise in the signal, we plot a section of the signal and visually inspect it for unwanted components. This helps us identify noisy regions in the signal that may require further processing.

2.2.7 *Masking (Filtering Out the Noise and Glitch Frequency)*

A mask is created to retain only the frequency components associated with the glitch.

MATLAB Code

```
mask = zeros(size(Y));
mask(idx) = 1;
ynew = Y .* mask;
```

This mask is applied to the frequency domain representation of the signal obtained from the FFT, effectively filtering out the glitch while preserving other signal components.

2.2.8 *Plot Filtered Signal*

After masking we obtaind the frequency spectrum that has only 50 Hz frequency.

MATLAB Code

```
P2_new = abs(ynew/N);
P1_new = P2_new(1:(N/2)+1);
P1_new(2:end) = 2*P1_new(2:end);

figure;
plot(f, P1_new)
title('Filtered␣Single-Sided␣Amplitude␣Spectrum')
xlabel('freq␣(Hz)')
ylabel('Amplitude')
```

2.2.9 *Inverse Fourier Transform (IFFT)*

The Inverse Fourier Transform (IFFT) is used to transform the filtered signal back to the time domain.

MATLAB Code

```
ynewtime = ifft(ynew, 'symmetric');
figure;
plot(t, ynewtime)
title('Filtered Signal in Time Domain')
xlabel('Time (s)')
ylabel('Amplitude')
```

This restores the signal to its original time-domain representation, allowing us to analyze the filtered signal's waveform. The filtered signal is plotted in the time domain to visualize the effect of filtering.

2.2.10 Glitch Detection After Filtering

Same as above.

MATLAB Code

```
figure;
A = (hours * 3600) + (minutes * 60) + seconds;
yg1 = ynewtime((A-50):(A+50), 1);
tg1 = t(1:length(yg));
plot(tg1, yg1);
title('Glitched Removed Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```

2.2.11 Noise Detection After Filtering

A section of the filtered signal is plotted to observe the reduction in noise or unwanted components.

MATLAB Code

```
ynew2 = ynewtime(1:1800, 1);
tnew2 = t(1:length(ynew2));
figure;
plot(tnew2, ynew2)
title('Noise Detection after Filtering')
xlabel('t (secs)')
ylabel('y(t)')
```

This step helps us evaluate the effectiveness of the filtering process in removing the glitch and improving signal quality.

2.2.12 Writing the Corrected Audio Signal

Finally, we save the filtered signal as a new audio file using the `audiowrite` function in MATLAB.

MATLAB Code

```
audiowrite('U:\SEMESTER-6\DSP\PROJECT\CODE
\Filtered Signal.wav, ynewtime, Fs);
```

This allows us to preserve the corrected signal for further analysis or use in practical applications.

Chapter 3

Results

3.1 Original Signal in Time Domain

The original audio signal is plotted in the time domain to show how the signal's amplitude changes over time. This plot provides a visual representation of the waveform, allowing us to observe any noticeable anomalies or characteristics in the signal's amplitude.

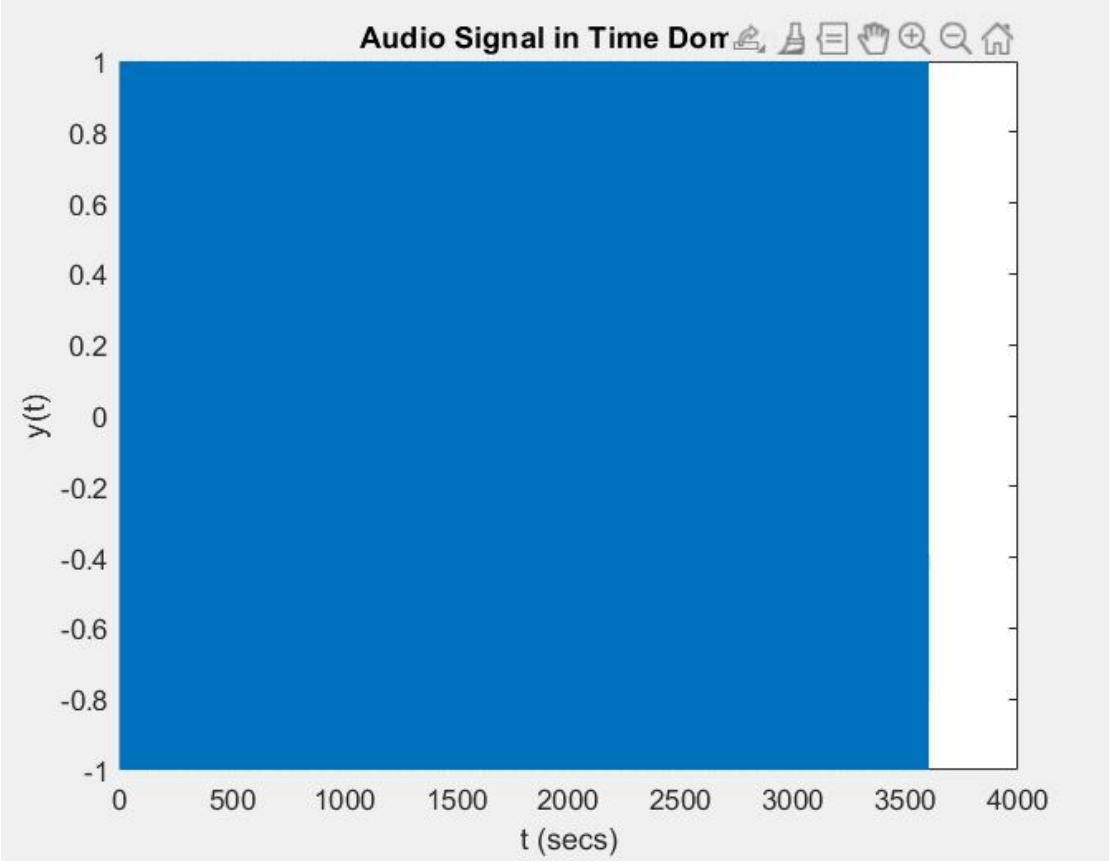

Figure 3.1: Original Audio Signal in Time Domain

A time-domain plot of the original audio signal, with the x-axis representing time (in seconds) and the y-axis representing the amplitude of the signal.

3.2 Original Signal in Frequency Domain

The frequency spectrum of the original signal is plotted after applying the Fast Fourier Transform (FFT). This plot shows the magnitude of different frequency components present in the signal. By examining this spectrum, we can identify the dominant frequencies and observe any unusual spikes that might indicate a glitch or noise.

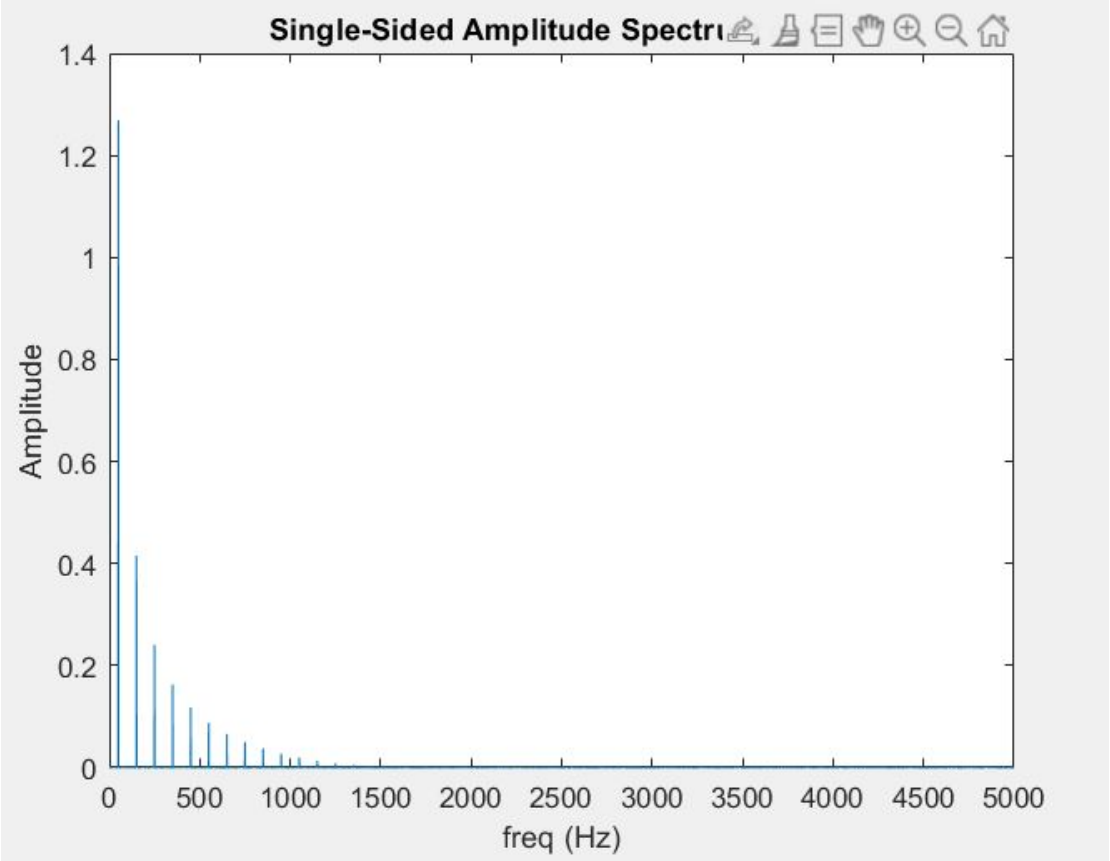

Figure 3.2: Original Audio Signal in Frequency Domain

A frequency-domain plot of the original audio signal, with the x-axis representing frequency (in Hz) and the y-axis representing the amplitude of the frequency components.

3.3 Glitch Detection

The detected glitch in the audio signal is highlighted in this subsection. The time instant at which the glitch occurs is marked on the plot. Identifying this precise moment is

crucial for understanding the nature of the anomaly and for implementing corrective measures.

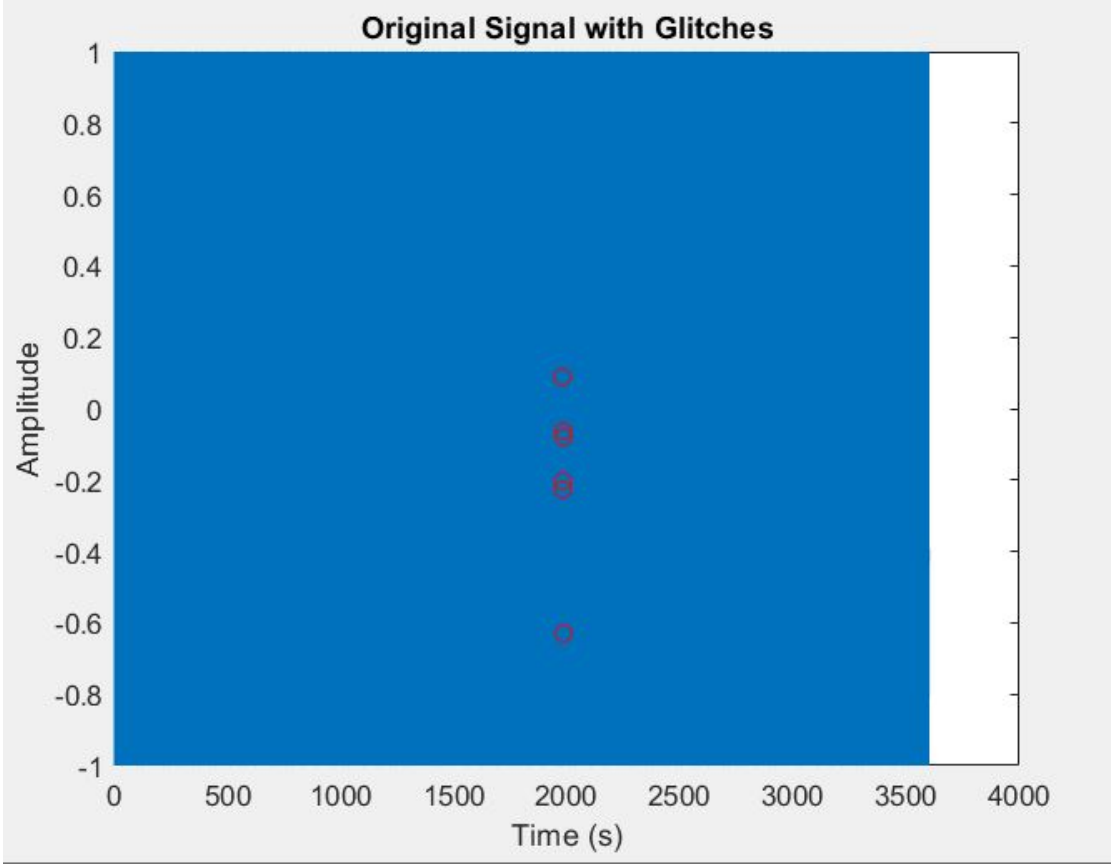

Figure 3.3: Detected Glitch in the Audio Signal

A plot indicating the time instant of the detected glitch, with annotations marking the glitch location.

```
Glitch Detected below 50 Hz at time: 0 hours, 33 minutes, 0.0001 seconds
Glitch Detected below 50 Hz at time: 0 hours, 33 minutes, 1.0001 seconds
Glitch Detected below 50 Hz at time: 0 hours, 33 minutes, 2.0001 seconds
Glitch Detected below 50 Hz at time: 0 hours, 33 minutes, 3.0001 seconds
Glitch Detected below 50 Hz at time: 0 hours, 33 minutes, 4.0001 seconds
Glitch Detected below 50 Hz at time: 0 hours, 33 minutes, 5.0001 seconds
```

Figure 3.4: Glitches Time

Now if we plot the signal for only the time at which there is glitch in the signal it will be like

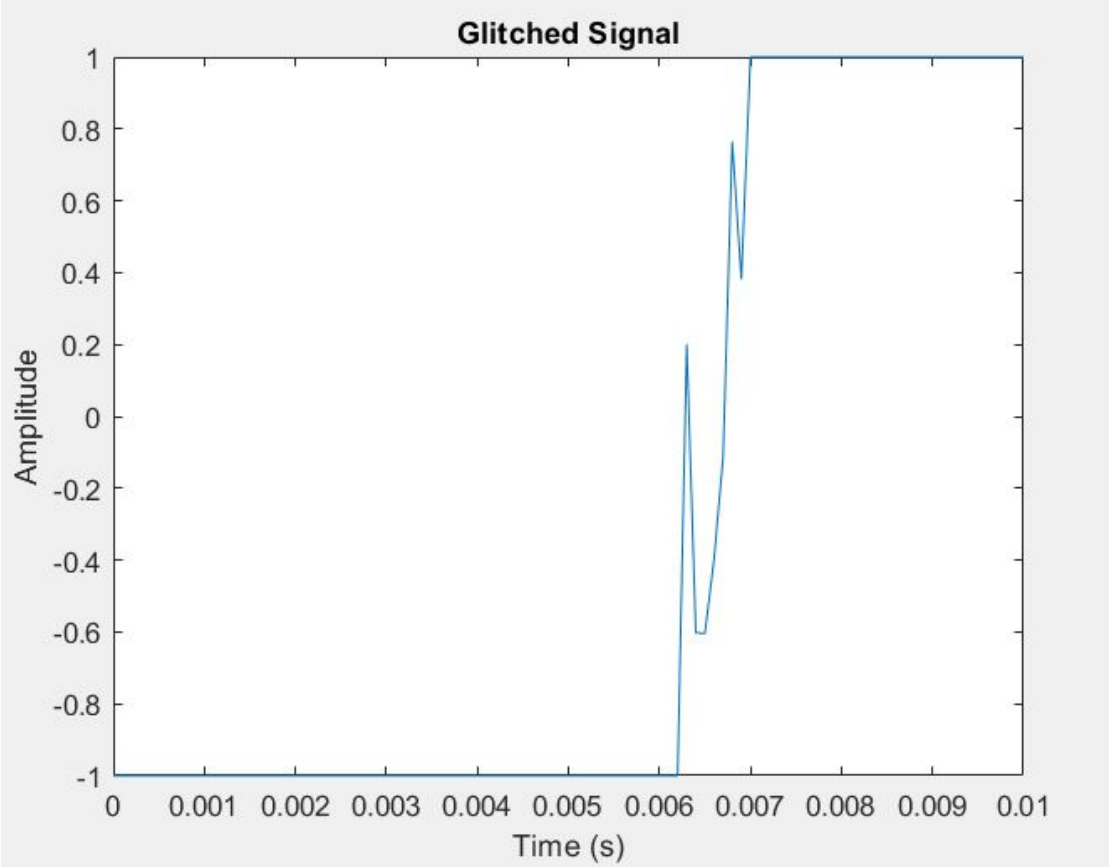

Figure 3.5: Glitches Time

This signal serves as a critical indicator of sudden frequency changes within an electrical system, offering invaluable insights into potential impacts on connected components and appliances. By detecting these abrupt frequency shifts, we gain essential foresight into the operational integrity and safety of electrical infrastructure

3.4 Noise Detection

Noise detection is performed on the original signal to identify unwanted components that may affect the signal quality. By plotting a section of the signal, we can visually inspect it for noise and determine the regions that require filtering.

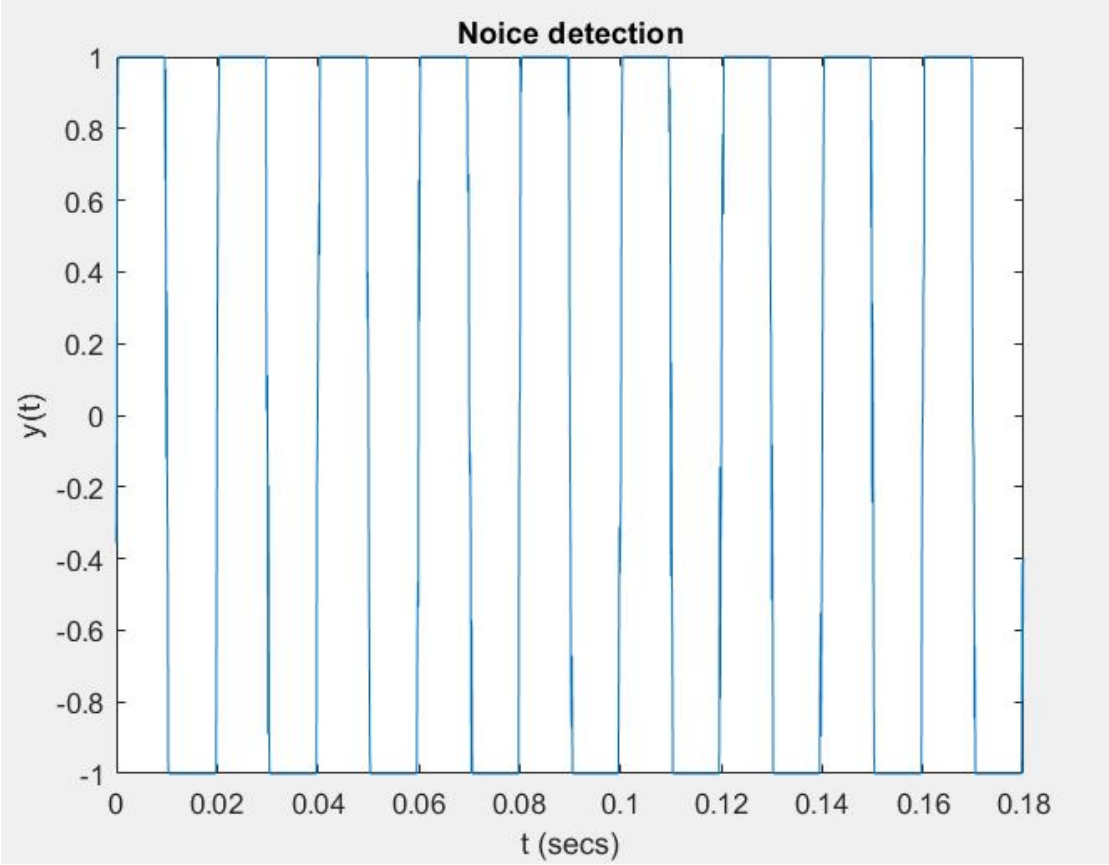

Figure 3.6: Detected Glitch in the Audio Signal

A plot showing the noise detected in the original audio signal before filtering, with the x-axis representing time (in seconds) and the y-axis representing the amplitude.

3.4.1 Masking

In the time domain, masking emphasizes the 50 Hz frequency while reducing others, refining signal clarity. This aids precision in tasks like power systems or audio processing. By isolating the desired frequency, masking allows targeted analysis, enhancing signal quality. It's a vital tool for filtering unwanted frequencies, ensuring accurate signal representation.

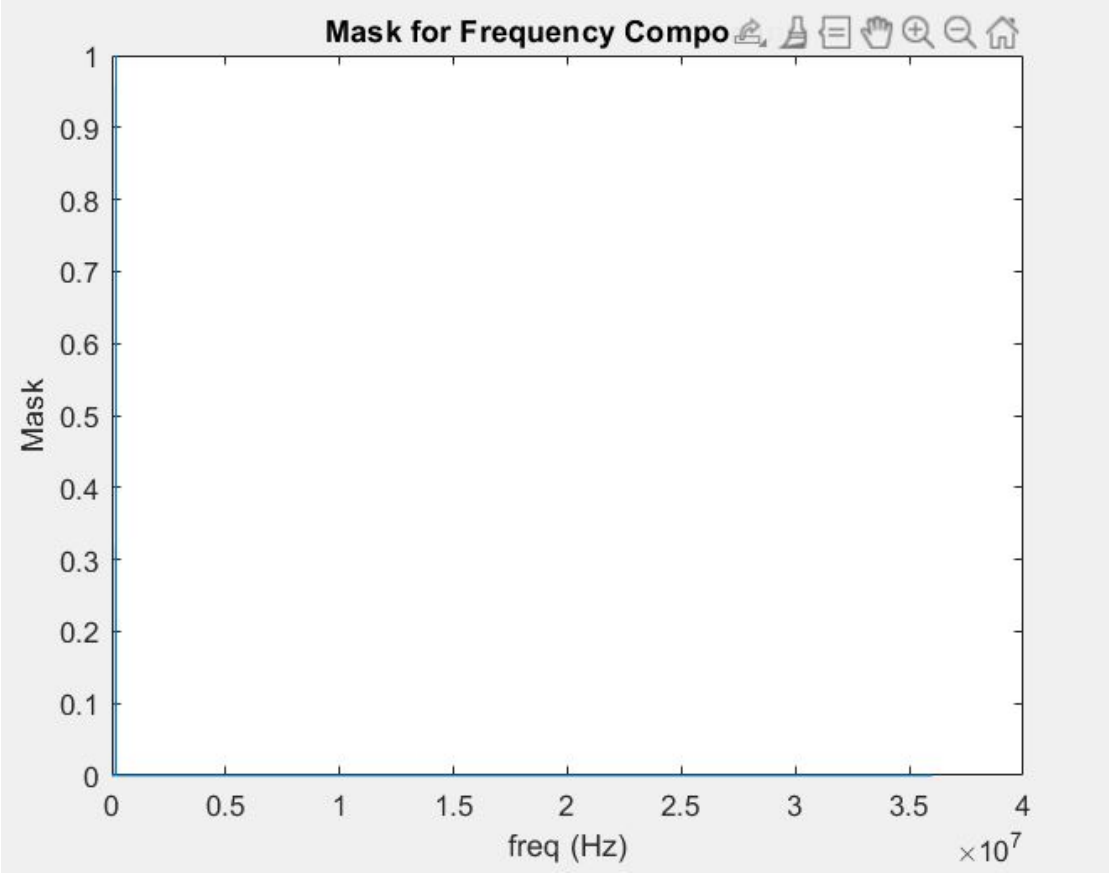

Figure 3.7: Filtered Audio Signal in Frequency Domain

3.5 Filtered Signal in Frequency Domain

The frequency spectrum of the filtered signal is plotted to show the changes after the glitch has been removed. This helps in evaluating the effectiveness of the filtering process by comparing the frequency components of the original and filtered signals. A successful filtering process should show a reduction in the unwanted frequency components associated with the glitch.

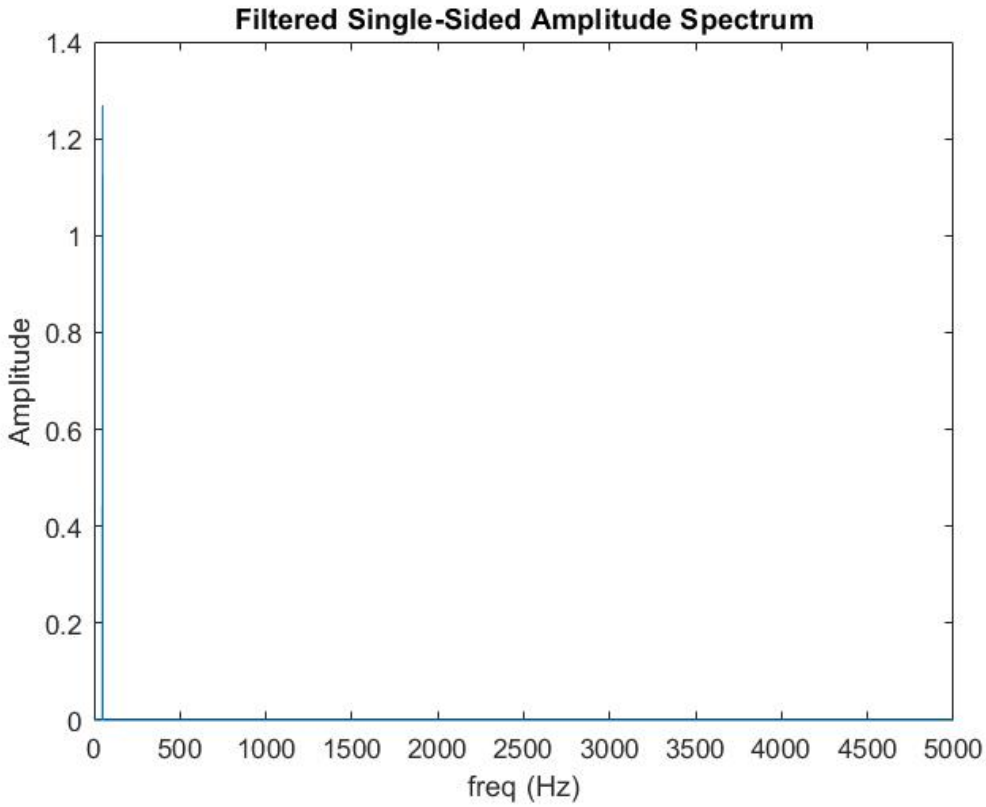

Figure 3.8: Filtered Audio Signal in Frequency Domain

A frequency-domain plot of the filtered audio signal, with the x-axis representing frequency (in Hz) and the y-axis representing the amplitude of the frequency components.

3.6 Noise and Glitch Detection After Filtering

3.6.1 Glitch Detection

After the filtering process, glitch detection is revisited to confirm the successful removal of sudden frequency deviations. This step validates the improvement in signal quality by comparing the presence of glitches before and after filtering. The plotted data visually represents the signal's glitch characteristics post-filtering, affirming the effectiveness of the glitch removal technique

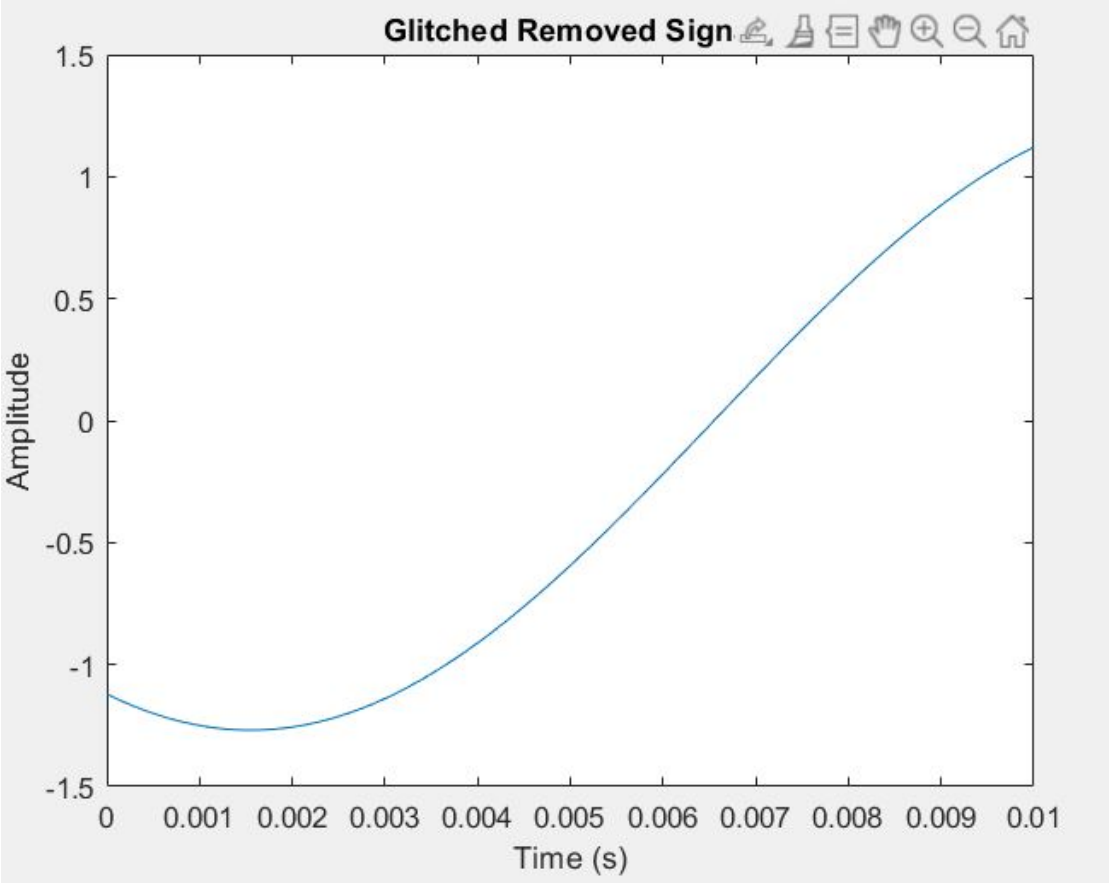

Figure 3.9: Noise Detection After Filtering

3.6.2 Noise Detection

Noise detection is revisited after the filtering process to ensure that unwanted components have been minimized. This step confirms the improvement in signal quality by comparing the noise levels before and after filtering. The plot provides a visual representation of the signal's noise characteristics post-filtering.

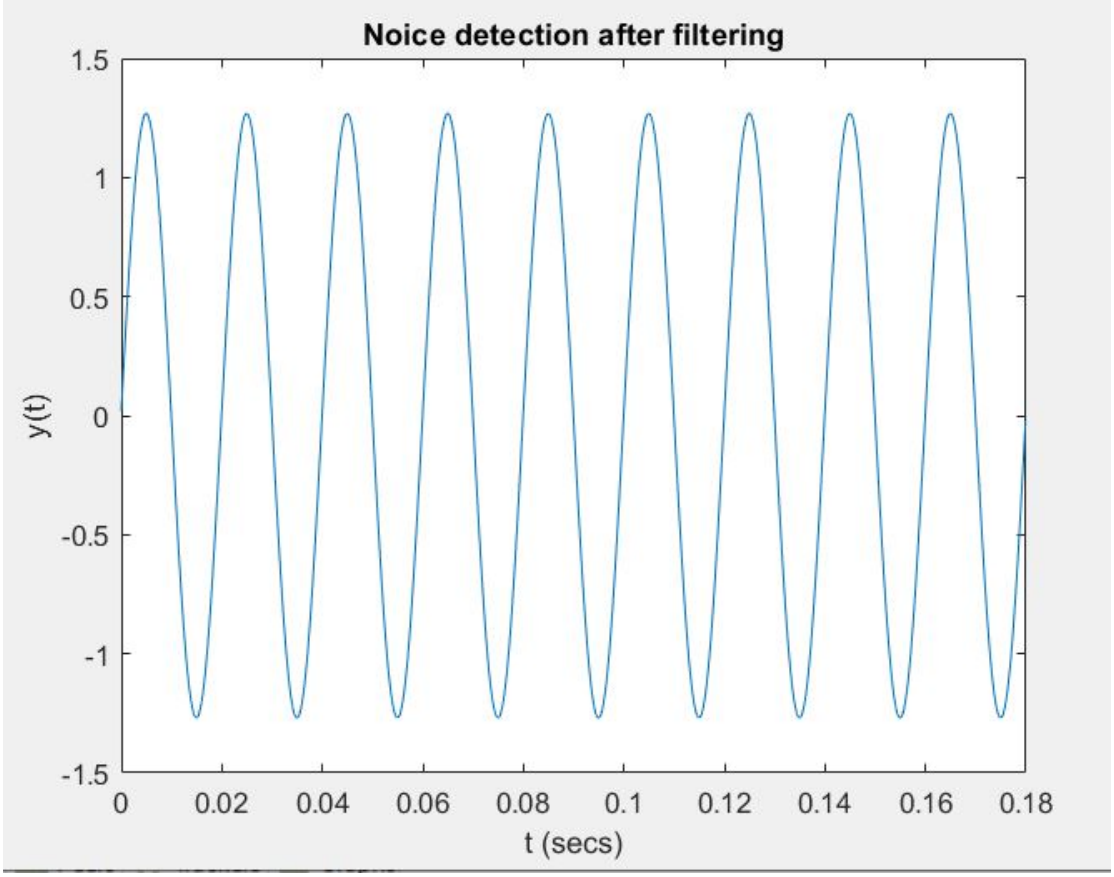

Figure 3.10: Noise Detection After Filtering

A plot showing the noise detection results after filtering, with the x-axis representing time (in seconds) and the y-axis representing the amplitude.From Figure

3.7 Filtered Signal in Time Domain

After applying the filtering process to remove the glitch, the resulting signal is plotted in the time domain. This plot shows the corrected signal with reduced anomalies. By comparing this plot to the original time-domain plot, we can assess the effectiveness of the filtering process in mitigating the detected glitch.

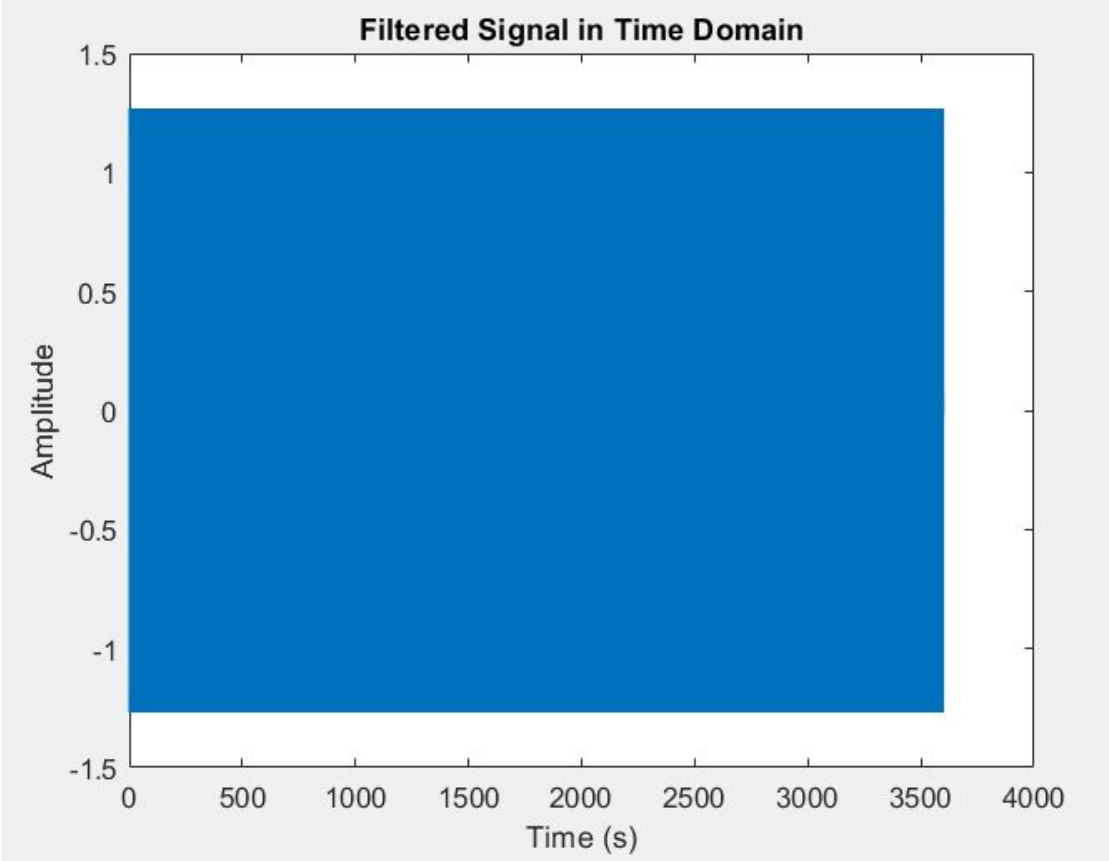

Figure 3.11: Filtered Audio Signal in Time Domain

A time-domain plot of the filtered audio signal, with the x-axis representing time (in seconds) and the y-axis representing the amplitude of the filtered signal.

Chapter 4

Conclusion

The successful development and implementation of this MATLAB-based solution significantly contribute to enhancing the stability and reliability of electrical power systems. By identifying and mitigating frequency glitches and noise, the solution helps prevent potential damage and costly equipment failures. The real-time processing capability ensures that the solution can be applied in practical scenarios, providing immediate benefits in operational power systems. The validation through testing further demonstrates the robustness and reliability of the solution, making it a valuable tool for maintaining the integrity of electrical power systems. This solution can be utilized in power system monitoring, transformer and motor protection, preventive maintenance, smart grid systems, industrial automation, and research, making it a valuable tool for ensuring safe and efficient power system operations.