Department of Electrical & Computer Engineering
AIR UNIVERSITY

PROJECT REPORT

Digital Signal Processing

Instructor Name

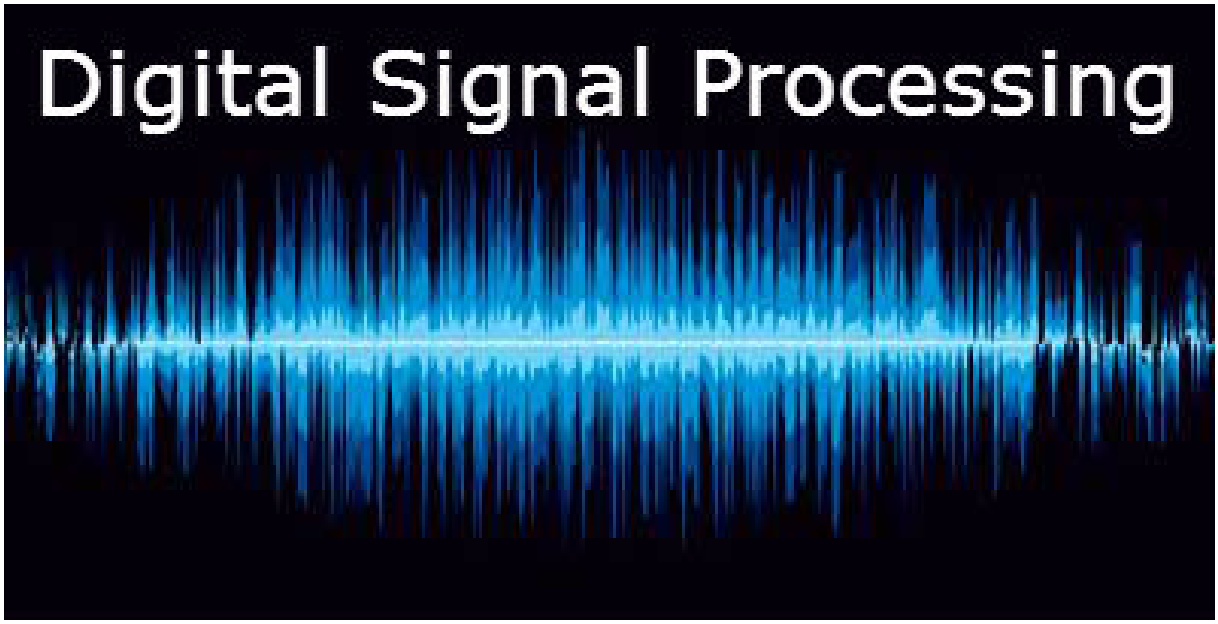
Miss Wania Anoosh

Submitted by

Muhammad Burhan Ahmed(210287)
Jawaad Ali(190549)
Muhammad Tahir Farooq(190523)

5/20/2024

Complex Engineering Activity



Signal Processing using Matlab

ABSTRACT

This project uses digital signal processing techniques to give a thorough examination of sinusoidal waveforms. Snafus and noise were successfully eliminated from the provided sinusoidal waveform by applying sophisticated DSP techniques. In order to improve the waveform's quality and accuracy, filtering techniques and adaptive algorithms were used.

The usefulness of the suggested DSP approach in analyzing and improving sinusoidal waveforms was shown by the results of experiments. The waveforms that underwent processing demonstrated enhanced clarity, decreased noise, and removed glitches, which increased their usefulness in a variety of applications, including audio processing, instrumentation, and telecommunications.

This research concludes by highlighting the importance of DSP in the analysis and enhancement of sinusoidal waveforms, thereby advancing the field of signal processing technology and enabling increased signal quality in real-world applications.

Contents

Problem Statement

The primary goal of this project is to design such a matlab program that is capable of doing Analysis and Restoration of a Noisy Sinusoidal Waveform with a Glitch. The process includes the identification of glitch and remove it along with noise, enhancing the quality of the waveform.

OBJECTIVES

- Hands-on experience in Signal Processing.
- To Identify the Instant of Glitch Occurrence.
- Familiarization with filter designing.
- To Remove Glitch and Noise from Audio Signals.

Software Used :

- Matlab 2021a

1 INTRODUCTION

1.1 Digital Signal Processing

The process of evaluating and adjusting a signal to maximize or enhance its effectiveness or performance is known as digital signal processing, or DSP. To create a signal that is better than the original, it entails using a variety of computational and mathematical methods to analog and digital signals. Its main applications are error detection and in-transit filtering and compression of analog signals. Either the frequency domain or the time domain can be used to represent a signal. One well-known example of a signal is voice. Signals also include voltage and electric current. A function of one or more independent variables can be a signal. A signal could depend on variables including distance, pressure, temperature, position, and time. In case a signal relies on a single independent.

1.2 Audio Processing

A specialized use of DSP that is dedicated to modifying and examining sound signals is called audio processing. This includes operations like audio compression, equalization, augmentation, noise reduction, and echo cancellation. Digital filters are crucial instruments for enhancing particular frequency components, eliminating undesirable noise, and changing the properties of the signal. Bandstop filters, for example, are used to remove certain frequency ranges, which is especially helpful in applications where certain frequencies result in glitches or disruptions. The application of DSP techniquesâspecifically, the use of a bandstop filterâto detect and eliminate audio signal glitches is the main topic of this paper. These bugs can seriously degrade the audio quality, which increases the risk of equipment damage and undesirable user experiences.

1.3 Background

1.3.1 Sinusoidal Signals

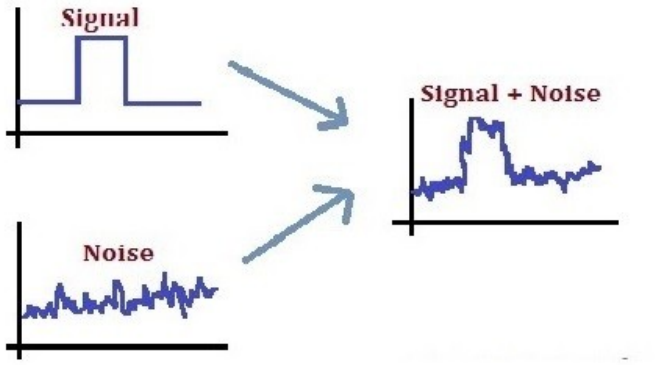
A sinusoidal wave, also known as a sin wave or sin signal is a mathematical function that describes a smooth, continuous, and periodic oscillation. It conveys information about a phenomenon and can be categorized into different types.

1.3.2 Glitch

Glitches can show up as bursts of noise, brief periods of distortion, sudden amplitude spikes or decreases, etc. To find any possible errors, it's crucial to examine the properties of the peaks that have been found as well as the surrounding audio data. The highest peaks in an audio signal may not always correspond to its glitches. A number of things can cause glitches, including electrical interference, hardware issues, software mistakes, or sudden changes in the characteristics of the signal.

1.3.3 Noise

Noise refers to random, unwanted fluctuations in the audio signal and appears as random variations in the signal amplitude. Common types of noise in audio signals include white noise (with equal power at all frequencies), background hum, hiss or clicks.



1.3.4 Filters

A digital filter is a mathematical method that is used to manipulate a signal in order to eliminate undesired information, like passing or blocking a specific frequency band. The two types of filters are listed below.

- Infinite impulse response (IIR)
- Finite impulse response (FIR)

If the impulse response of the filter drops to zero after a finite time has elapsed, it is referred to as an FIR filter (Finite Impulse Response). These filters stand as essential components in Digital Signal Processing, valued for their stability, linear phase response, and simple implementation. Following are the four different filter types:

- Highpass
- Low pass
- Bandpass
- Band stop

1.3.5 Hanning Window

In digital signal processing, the Hanning window, often referred to as the Hann window, is a commonly utilized window function that reduces spectrum leakage during Fourier transform analysis. When a signal is not periodic within the observation window, it is known as spectral leakage and results in false frequency components in the Fourier transform output. By slowly decreasing the signal to zero at the edges, the Hanning window smoothes the transitions at the windowed segment's boundaries.

2 Deliverables

Following are the functionalities provided by project

- Plot the waveform with respect to time
- Find the frequencies of the waveform
- Find the precise time instant (hr min sec) at which the glitch in frequency occurs
- Remove the glitch from the sinusoidal waveform
- Removes noise from sinusoid waveform

3 Procedure

Here are the steps taken to complete this design in Matlab:

1. **Plot the waveform over time:**
   Load the wave file using MATLAB's audioread function. The waveform represents the amplitude of the signal over time.
2. **Find the frequency of the waveform:**
   Obtain the frequency spectrum of the original waveform using Fast Fourier Transform.
3. **Find the precise time instant when the glitch occurs:**
   Detect sudden changes in the frequency domain using a short-time Fourier transform. Specific time indexes corresponding to the glitch can be found using the hamming filter.
4. **Remove the glitch from the waveform:**
   Smooth the waveform to remove the abrupt change caused by the glitch.
   Measures:
   Adjust parameters such as cutoff frequency for optimal results.

5. **Remove noise from the waveform:**
   Use low pass filters to Smooth the waveform.

4 Working Methodology

To identify a glitch in the waveform, especially in a noisy sinusoidal signal, analyzing the frequency content over time is crucial because glitches often manifest as abrupt changes in the signal's frequency characteristics.

4.1 Techniques used

- Plot the waveform wat time
    1. audioread is used to load the audio file. It returns the audio data y and the sample rate Fs.
    2. The time vector t is calculated based on the length of the audio data and the sample rate.
    3. The audio signal is then plotted against time using plot.
- Finds the frequency of this waveform
    1. Then we compute the FFT of the audio signal using fft command.
- Finds the precise time instant (hrs min secs) at which the glitch in frequency occurs
    1. We compute the STFT of the audio signal using the spectrogram function.
    2. It divides the signal into short segments using a window of size window-size and calculates the FFT of each segment.
    3. hamming window is used for this purpose
    4. starting time of the glitch is find using stft and then using this start time as reference, end time and duration of the glitch is identified.
- Removes the glitch from the sinusoidal waveform
    1. Now, this glitch is removed using low pass filter.
- Removes the noise from the sinusoidal waveform
    1. Finally, the Noise from the signal is removed by using a low-pass filter. It returns a smooth sin wave.

Reason for Method
The STFT is chosen because it provides a detailed time-frequency representation of the signal, allowing us to observe how the frequency content changes over time. This is essential for detecting sudden frequency glitches in a noisy environment, which may not be evident in a purely time-domain analysis. By analyzing the frequency deviation, we can precisely pinpoint the moment when the glitch occurs and take appropriate actions to mitigate potential damage.

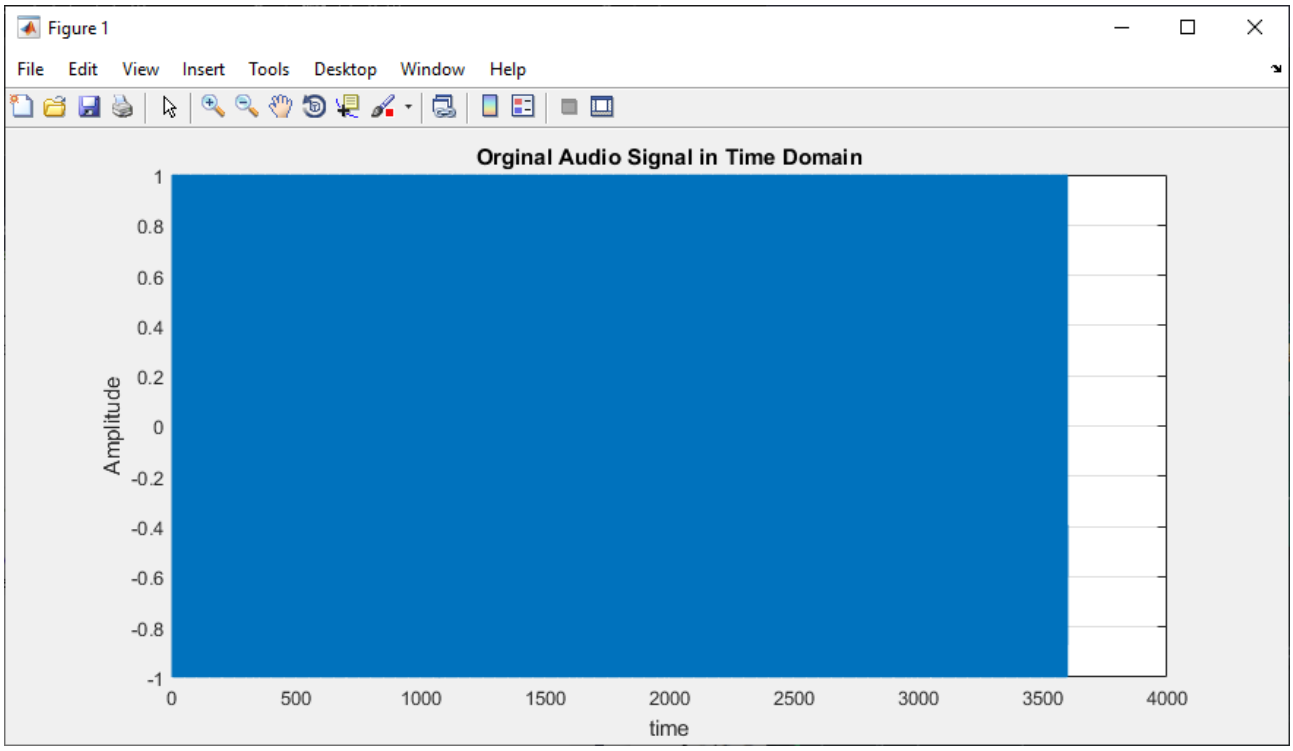
5 Source Code

To detect a glitch in frequency within a noisy waveform, we can use a time-frequency analysis technique such as the Short-Time Fourier Transform (STFT). The STFT allows us to analyze how the frequency content of the signal evolves over time, making it possible to identify sudden changes or glitches in the frequency.

6 Results & Screen Shots

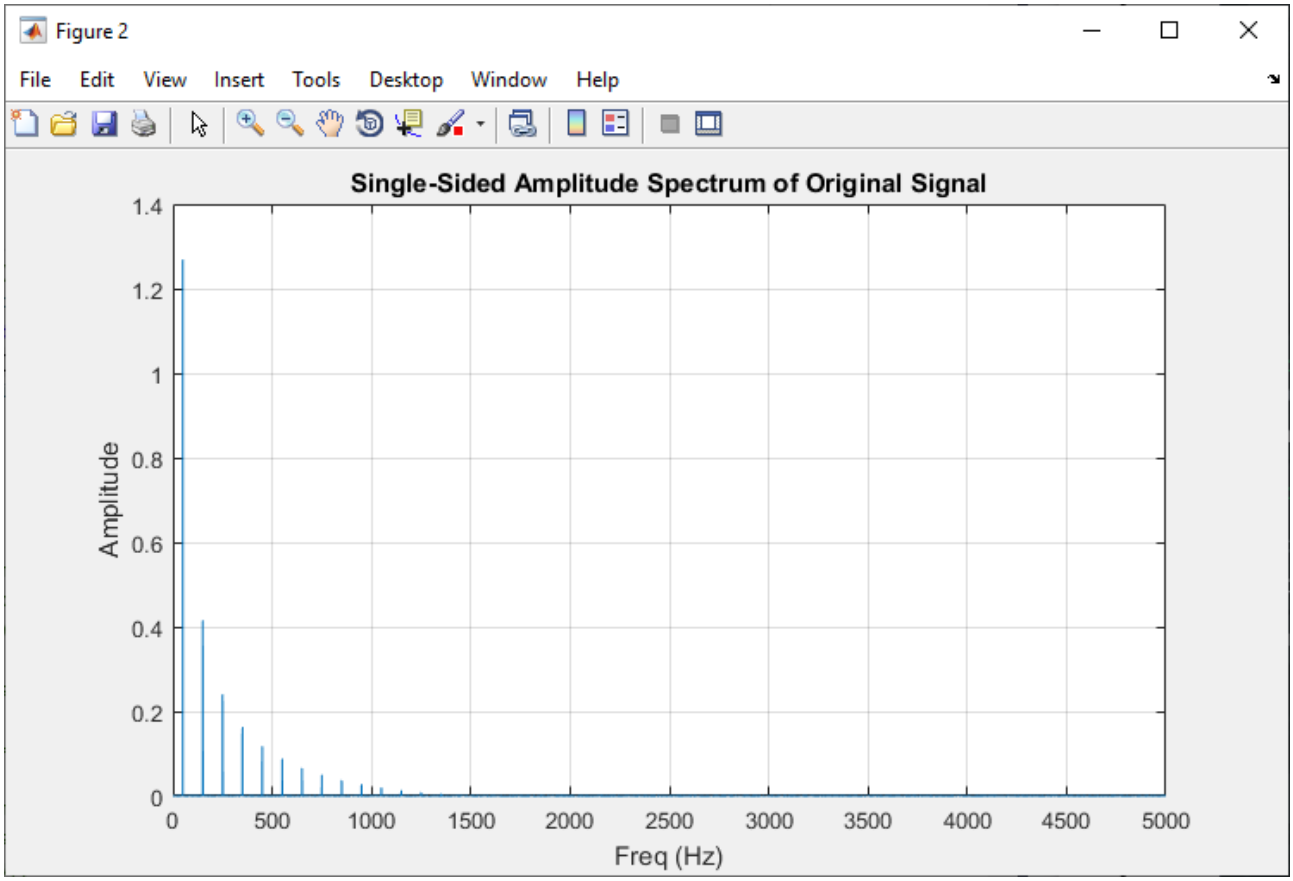
Following are the snaps for Project
Screenshot 1(Task 1)

```
clc
clear all
close all

[signal,Fs] = audioread('Noisy.wav');
signal= signal(:,1);
info = audioinfo('Noisy.wav');
t = 0:(1/Fs):(info.Duration-1/Fs);

figure;
plot(t,signal)
title('Orginal Audio Signal in Time Domain')
xlabel('time')
ylabel('Amplitude')
grid on
```



Screenshot 2 (Task 2)

```
Y = fft(signal);
N = length(signal);
P2 = abs(Y/N);
P1 = P2(1:(N/2)+1);
P1(2:end) = 2*P1(2:end);
f = Fs*(0:(N/2))/N;

figure;
plot(f,P1)
title('Single-Sided Amplitude Spectrum of Original Signal')
xlabel('Freq (Hz)')
ylabel('Amplitude')
grid on
```



Screenshot 3 (Task 3)

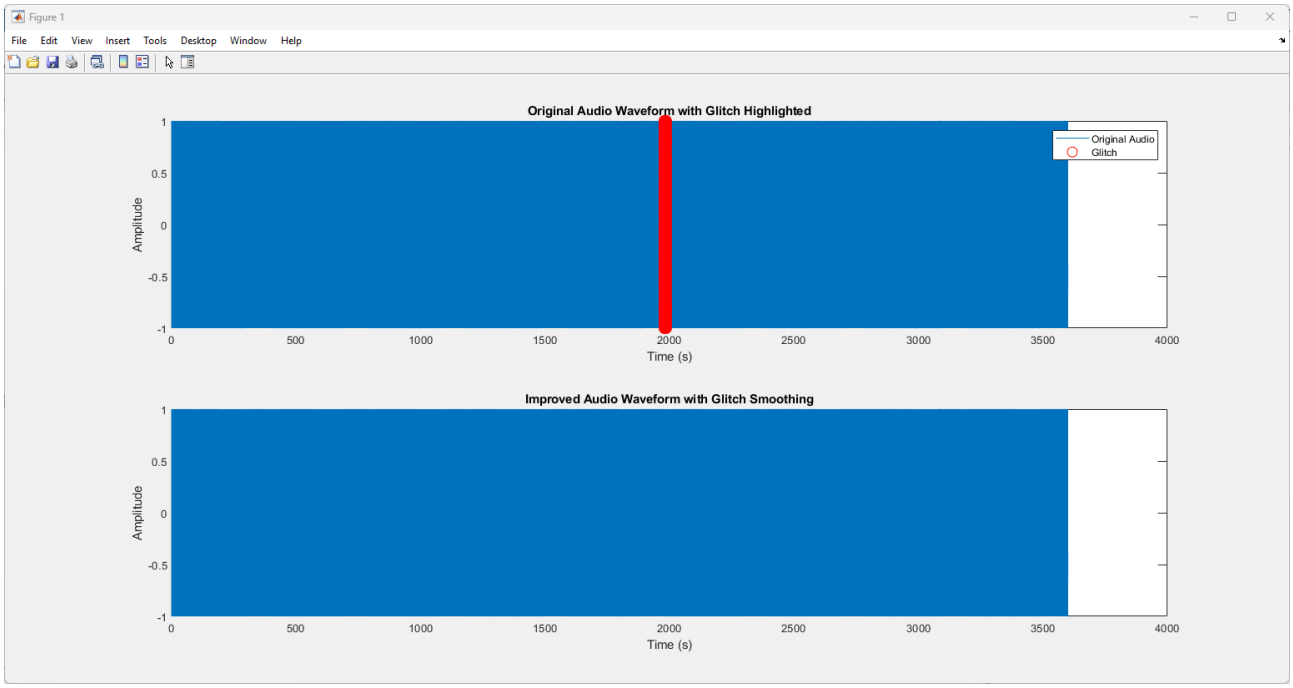
```
clc;
clear all;
close all;

[y, Fs] = audioread('Noisy.wav');
```

```
6
7  size = 1024;
8  overlap = 512;
9  length_fft = size;
10
11 [coff, f, t] = spectrogram(y, size,overlap, length_fft, Fs);
12
13 [freq, freqIndex] = max((coff));
14
15 glitch_Threshold = 1;
16 startIndex = find(abs(diff(freqIndex)) > glitch_Threshold, 1);
17
18 endIndex = startIndex;
19
20 for i = startIndex + 1 : length(freqIndex)
21     if abs(freqIndex(i) - freqIndex(startIndex - 1)) <
       glitch_Threshold
22         endIndex = i;
23         break;
24     end
25 end
26
27 StartMilli = t(startIndex);
28 EndMilli = t(endIndex);
29
30 DurationSec = EndMilli - StartMilli;
31
32 StartHour = floor(StartMilli / 3600);
33 StartMin = floor((StartMilli - StartHour * 3600) / 60);
34 StartMilli = StartMilli - StartHour * 3600 - StartMin * 60;
35
36 EndHour = floor(EndMilli / 3600);
37 EndMin = floor((EndMilli - EndHour * 3600) / 60);
38 EndMilli = EndMilli - EndHour * 3600 - EndMin * 60;
39
40 fprintf('Glitch starts at %02d:%02d:%06.3f and ends at
     %02d:%02d:%06.3f\n', ...
41     StartHour, StartMin, StartMilli, ...
42     EndHour, EndMin, EndMilli);
43 fprintf('Glitch duration: %f seconds\n', DurationSec);
```

```
Command Window
  Glitch starts at 00:32:59.955 and ends at 00:33:05.997
  Glitch duration: 6.041600 seconds
fx >>
```

Screenshot 4 (Task 4)

```
baseFreq = 45;
cutoffFreq = baseFreq / (Fs / 2);
order = 200;
fil = fir1(order, cutoffFreq, 'low');

convolved_signal = conv(y(Start:End),fil);
%convolved_signal=convolved_signal*0.9981
y(Start:End) = convolved_signal;

tSec = (0:length(y)-1) / Fs;
figure;
subplot(2,1,1);
plot(tSec, y);
hold on;
plot(tSec, y, 'ro', 'MarkerSize', 10);
xlabel('Time (s)');
ylabel('Amplitude');
title('Original Audio Waveform with Glitch Highlighted');
legend('Original Audio', 'Glitch');

subplot(2,1,2);
plot(tSec, y_corrected);
xlabel('Time (s)');
ylabel('Amplitude');
title('Improved Audio Waveform with Glitch Smoothing');

audiowrite('improved_audio.wav', y_corrected, Fs);
```



Screenshot 5 (Task 5)

```
clc
clear all;
close all;

[x, fs] =audioread('improved_audio.wav');
t = (0:length(x)-1)/fs;

Fc = 150;
[b, a] = butter(4, Fc/(fs/2), 'low');
filterSignal = filtfilt(b, a, x);

figure;
plot( t,x);
xlabel('Time (s)');
ylabel('Amplitude');
title('Orignal Waveform');

figure;
plot(t,filterSignal);
xlabel('Time (s)');
ylabel('Amplitude');
title('Filtered Waveform (Low-Pass Filter)');

audiowrite('Cleaned_Sinusoid.wav', filterSignal, fs);
```

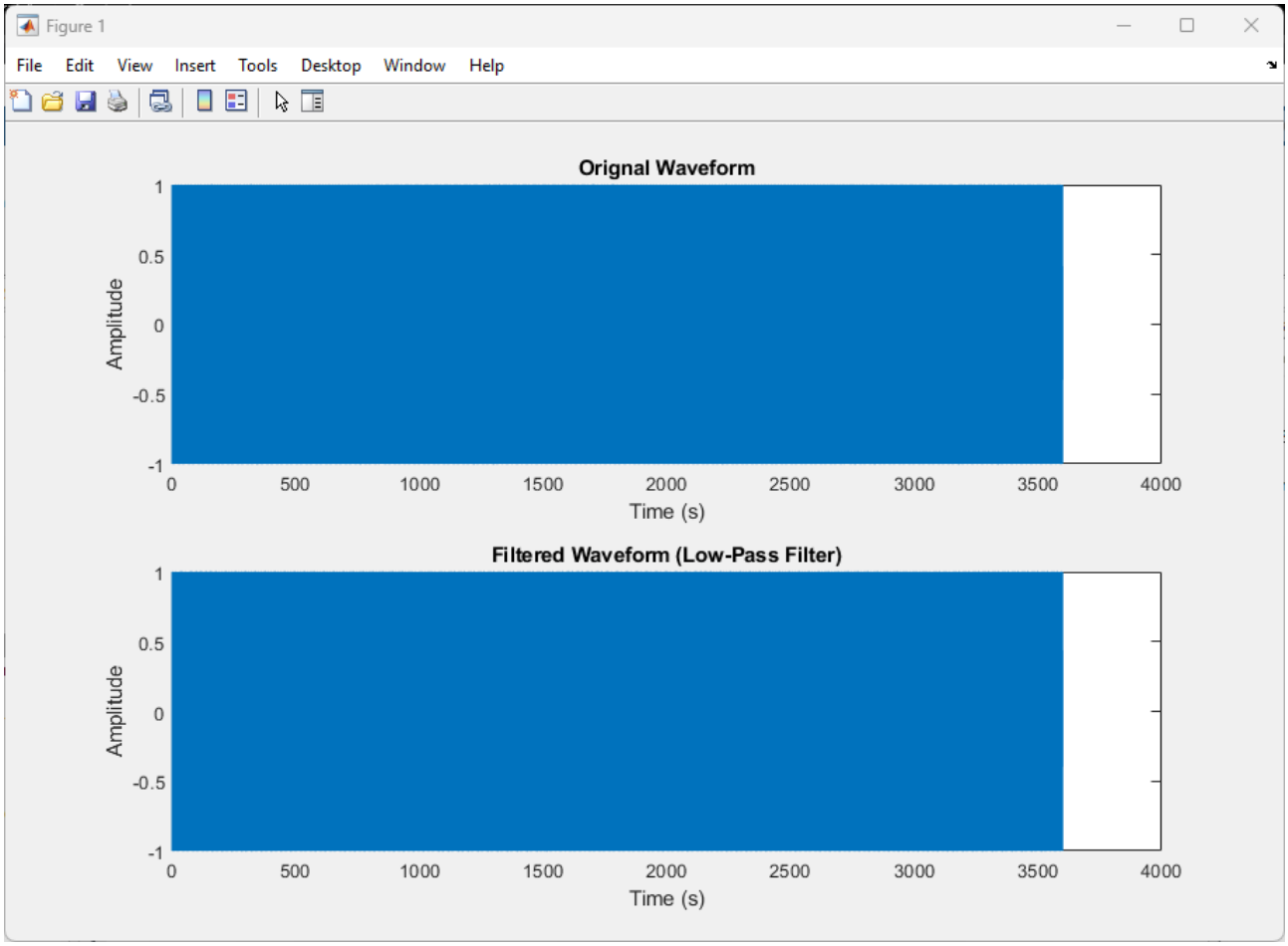


Figure 1
File Edit View Insert Tools Desktop Window Help
Orignal Waveform
Amplitude
Time (s)
Filtered Waveform (Low-Pass Filter)
Amplitude
Time (s)

7 Conclusion

In our complex Engineering Activity, we implemented signal processing methods to enhance signal quality by reducing noise and removing glitches. Through lowpass filtering and STFT we effectively attenuated high-frequency noise, improving signal clarity. We developed algorithms for glitch identification, important for identifying disturbances within the signal, and used FIR filters to remove that glitches and make signal noise free.

The technique have practical implications across various domains, such as audio processing, biomedical engineering, communication systems, industrial automation and video processing. These techniques improve data integrity, enabling clearer signals and more reliable analysis in real-world applications.Overall, our project shows the significance of signal processing.