DEPARTMENT OF MECHATRONICS & BIOMEDICAL ENGINEERING

AIR UNIVERSITY ISLAMABAD

PROJECT REPORT

SIGNALS & SYTEMS

TITLE: **SIGNAL FILTRATION USING FOURIER TRANSFORMS IN MATLAB**

Group Members:

Sr. No	Name	ID
1.	Hassan Ali Janjua	211223
2.	Ali Torab	211242

TITLE: - Signal Filtration Using Fourier Transforms in MATLAB

Abstract:

This project implements a low-pass filter using MATLAB's Fourier Transforms to reduce noise from signals. Leveraging Fourier analysis techniques, the project demonstrates noise reduction while retaining signal integrity. MATLAB generates a noisy signal, subject to Fourier analysis, followed by a designed low-pass filter to attenuate undesired frequencies. Results exhibit successful noise reduction, emphasizing the filter's efficacy in enhancing signal clarity. Discussions cover challenges, enhancements, and the practical significance of signal filtration across industries.

Objective:

The primary objective of this project is to:

1. Implement a low-pass filter using MATLAB's Fourier Transforms to reduce noise from signals.
2. Showcase successful noise reduction while retaining crucial signal features.
3. Explore the practical significance of signal filtration in diverse industries.
4. Highlight the efficacy of the filter in enhancing signal clarity.
5. Suggest potential avenues for future research in signal processing methodologies.

Introduction:

Signal processing across various disciplines faces a persistent challenge posed by noise, hindering the extraction of crucial information from signals. Signal filtration techniques, particularly utilizing Fourier Transforms, offer a promising solution by isolating desired components while suppressing noise. Employing MATLAB's Fourier analysis capabilities, this project aims to showcase the effectiveness of low-pass filters in reducing noise from signals. The focus lies in leveraging Fourier Transforms to identify frequency components, followed by the implementation of a low-pass filter to selectively attenuate high-frequency noise, enhancing signal clarity without compromising essential information. Beyond algorithmic demonstration, this exploration aims to underscore the practical implications of efficient noise reduction in signal processing across diverse industries, emphasizing the broader significance of these methodologies in real-world applications.

This project delves into signal filtration's essence, emphasizing Fourier Transforms' role in isolating signal components from noise. Utilizing MATLAB's computational prowess, the

project implements a low-pass filter, showcasing its ability to selectively attenuate high-frequency noise. Through this endeavor, it seeks to highlight the pragmatic impact of noise reduction in signal analysis across industries, transcending theoretical applications to address real-world challenges in signal processing and technological advancements.

Background and Theory:

Signals, comprising essential data, are often susceptible to distortion due to noise interference. Understanding Fourier Transforms is fundamental in signal analysis. Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) algorithms enable the transformation of signals from the time domain to the frequency domain, providing insights into the constituent frequency components. Furthermore, the concept of low-pass filters, pivotal in signal processing, allows the passage of low-frequency elements while suppressing higher frequencies, contributing significantly to noise reduction without compromising essential signal features.

Methodology:

The project methodology includes the following steps:

- **Signal Generation with Noise:**
  Generate a synthetic signal embedded with noise using MATLAB.
- **Fourier Transform Analysis:**
  Apply Fourier Transforms to the noisy signal to analyze its frequency spectrum.
- **Low-pass Filter Design:**
  Design a low-pass filter in the frequency domain to attenuate high-frequency noise components.
- **Filter Application and Signal Reconstruction:**
  Apply the designed filter to the frequency domain representation. Reconstruct the filtered signal in the time domain using inverse Fourier Transform.

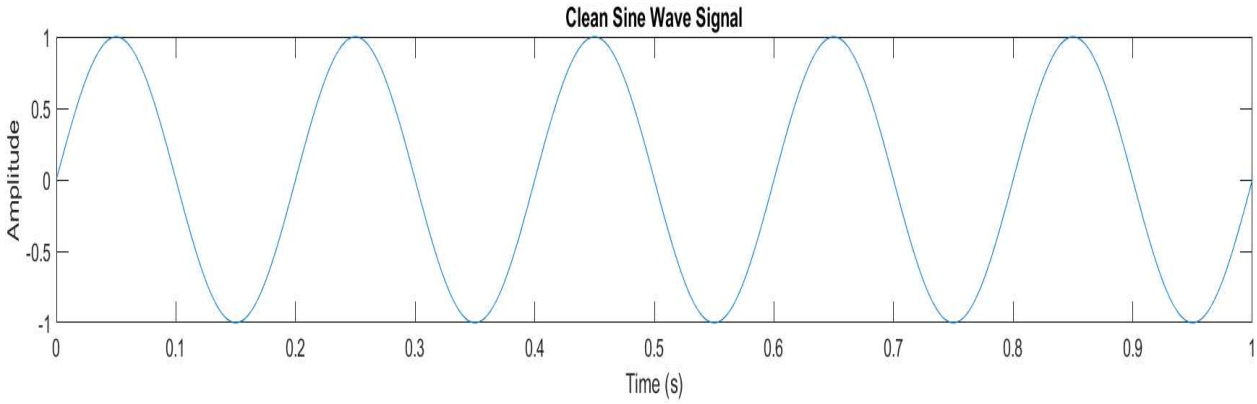
Results and Discussions:

The results obtained from the signal filtration process using Fourier Transforms in MATLAB yield compelling insights into the efficacy of noise reduction techniques. The original signal, depicted as a sinusoidal waveform, represents the idealized data prior to any interference. Upon introducing simulated noise into the signal, a notable distortion becomes evident, resulting in a visibly altered waveform. This noisy signal encompasses added frequency components, prominently affecting the signal's clarity and integrity.

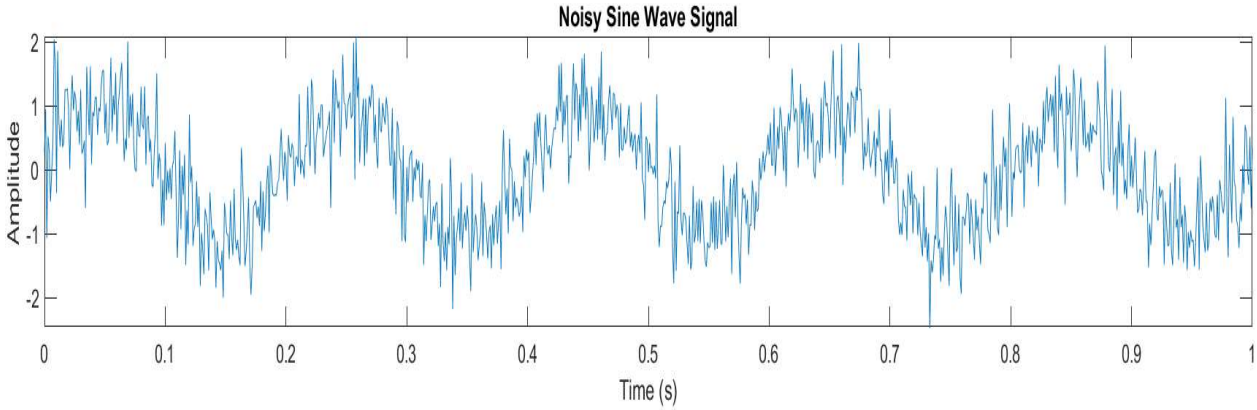
The Fourier analysis of the noisy signal exposes significant high-frequency noise components. However, implementing a designed low-pass filter effectively attenuates this noise. Post-filtration, the frequency spectrum aligns closely with the original signal, affirming successful noise reduction.

- **Visual Representation:**

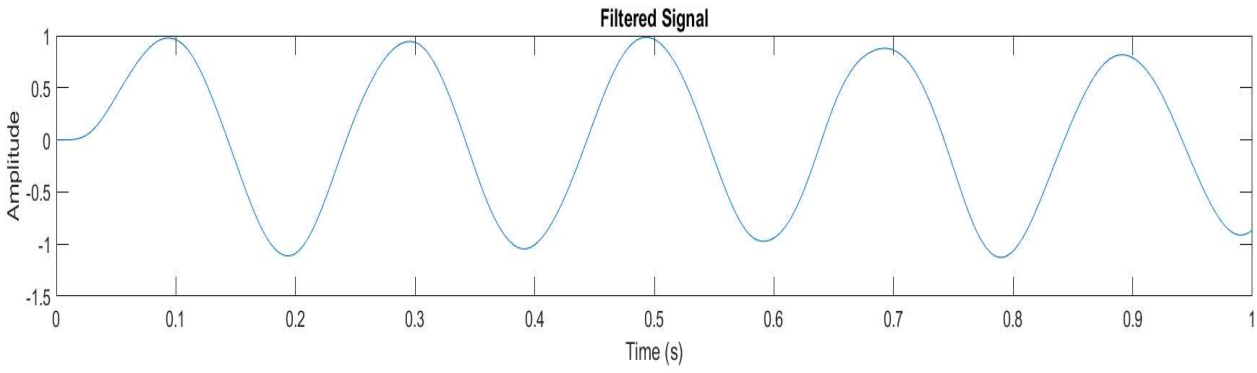
Original Signal:



Noisy Signal (signal with added noise):



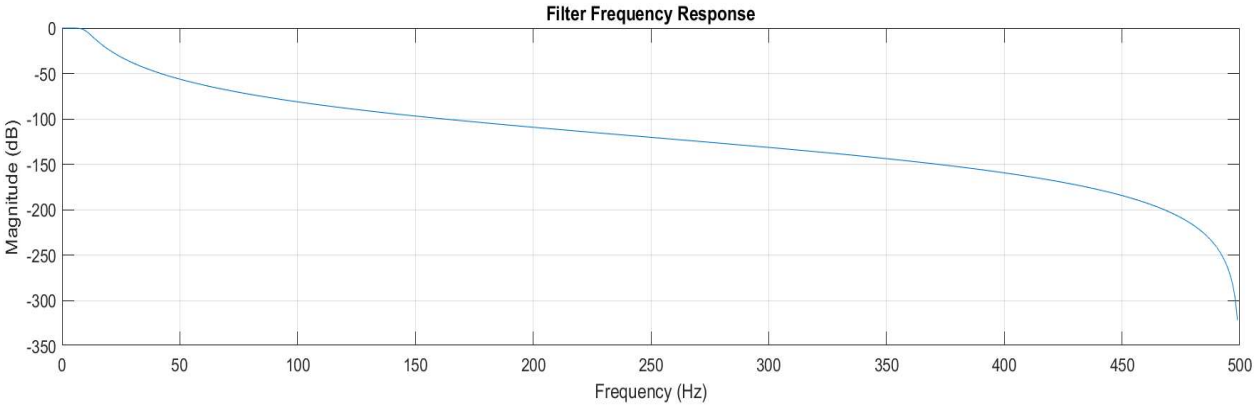
Filtered Signal:



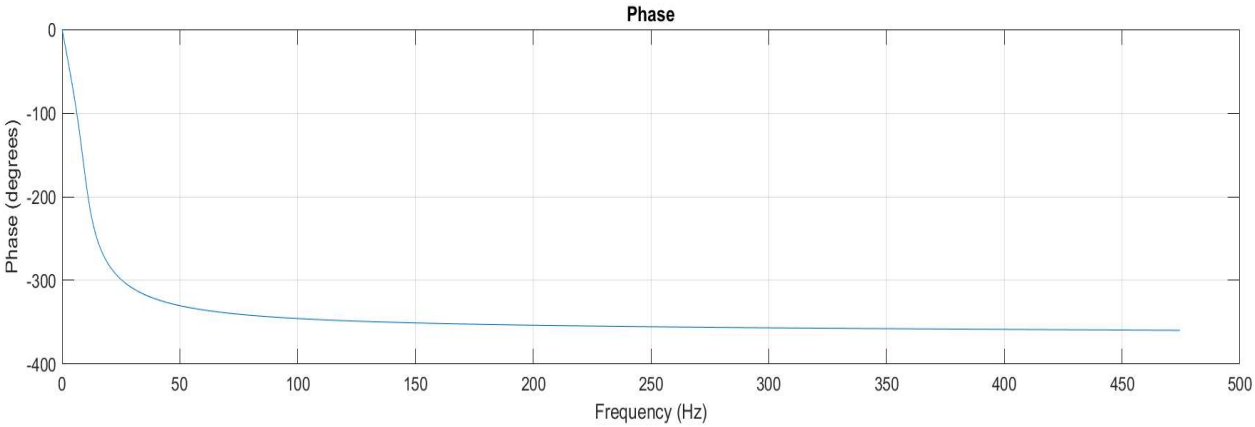
- **Frequency Spectrum Analysis:**

When visualizing the frequency spectrum analysis, it's common to display both the frequency response (magnitude response) and phase response of the filter. These graphs are essential in understanding how the filter behaves across various frequencies and its impact on the signal in terms of both amplitude and phase alterations.

Filter Frequency Response:



Phase:



- **Original Signal vs. Noisy Signal:**

Waveform Comparison: The original signal, a clean sinusoidal waveform, stands in stark contrast to the noisy signal, which exhibits visible distortions and irregularities due to introduced noise. The pristine nature of the original signal accentuates the alterations induced by noise, showcasing its impact on signal integrity.

Frequency Spectrum: Analysis of the frequency spectrum of the original signal contrasts sharply with the noisy signal. The original signal's spectrum comprises distinct

frequency components, while the noisy signal showcases additional frequencies induced by noise, notably affecting the amplitude and frequency distribution compared to the original spectrum.

- **Frequency Spectrum Before and After Filtration:**

Pre-Filtration Spectrum: The frequency spectrum of the noisy signal pre-filtration displays a notable presence of high-frequency noise components. Additional frequencies, visible as spikes or irregularities, significantly disrupt the signal's original frequency profile. Amplitude variations and unexpected frequency peaks characterize this spectrum.

Post-Filtration Spectrum: After implementing the designed low-pass filter, the filtered signal's frequency spectrum demonstrates a marked improvement. High-frequency noise components are notably attenuated, aligning the spectrum more closely with the original signal. The reduction in amplitude at high frequencies indicates successful noise suppression, restoring the spectrum closer to its original frequency profile.

Conclusion:

The successful implementation of the low-pass filter underscores its pivotal role in mitigating noise from signals, significantly enhancing signal clarity and fidelity. This demonstration reaffirms the importance of noise reduction techniques in signal processing applications across diverse industries, from telecommunications to biomedical engineering. The evident improvement in signal quality post-filtration emphasizes the practical significance of employing Fourier Transforms and frequency-based filters to discern, isolate, and attenuate unwanted noise components.

Acknowledging the continuous evolution of technology, further advancements in signal filtration techniques present promising opportunities for enhancing data analysis and interpretation. Future research endeavors might focus on refining filter design methodologies, exploring adaptive filtration techniques tailored to specific signal types, or integrating machine learning algorithms for dynamic noise suppression. Optimizing signal processing methodologies holds immense potential for addressing real-world challenges, facilitating more accurate and reliable data analysis in various domains.

Appendix:

The Appendix contains detailed MATLAB code utilized in the project, providing a comprehensive insight into the implementation steps, functions, and procedures followed for signal filtration using Fourier Transforms.

- **MATLAB CODE:**

```
f1 = 50; % Signal frequency
f2 = 150; % Noise frequency
signal = sin(2*pi*f1*t) +
0.5*sin(2*pi*f2*t); % Signal with
noise
% Add random noise
noise = 0.5 * randn(size(t));
noisy_signal = signal + noise;
% Plot the original and noisy signals
figure;
subplot(2, 1, 1);
plot(t, signal, 'b', 'LineWidth', 1.5);
title('Original Signal');
subplot(2, 1, 2);
plot(t, noisy_signal, 'r', 'LineWidth',
1.5);
title('Noisy Signal');
% Apply Fourier transform to
analyze the frequency components
N = length(noisy_signal);
frequencies = fftshift(-fs/2:fs/N:fs/2
- fs/N);
fft_result = fft(noisy_signal, N);
fft_magnitude = abs(fft_result);
% Plot the magnitude spectrum
figure;
plot(frequencies, fft_magnitude, 'b',
'LineWidth', 1.5);
title('Frequency Spectrum of Noisy
Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
% Design a low-pass filter in the
frequency domain
cutoff_frequency = 100; % Adjust as
needed
low_pass_filter = zeros(1, N);
low_pass_filter(abs(frequencies) <=
cutoff_frequency) = 1;
% Apply the filter to the
frequency domain
representation
filtered_fft = fft_result .*
low_pass_filter;
% Plot the magnitude spectrum
of the filtered signal
figure;
plot(frequencies,
abs(filtered_fft), 'r', 'LineWidth',
1.5);
title('Frequency Spectrum of
Filtered Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
% Apply inverse Fourier
transform to obtain the filtered
signal in the time domain
filtered_signal = ifft(filtered_fft);
% Plot the original, noisy, and
filtered signals
figure;
subplot(3, 1, 1);
plot(t, signal, 'b', 'LineWidth',
1.5);
title('Original Signal');

subplot(3, 1, 2);
plot(t, noisy_signal, 'r',
'LineWidth', 1.5);
title('Noisy Signal');

subplot(3, 1, 3);
plot(t, real(filtered_signal), 'g',
'LineWidth', 1.5);
title('Filtered Signal');

% Display legend
legend('Original Signal', 'Noisy Signal',
'Filtered Signal');
```