Analysis of a Square Wave Signal Using Fourier Transform

Fatemeh Amirabadi Zadeh

1 Introduction

This project focuses on analyzing a square wave signal with a frequency of 100 Hz using the Fourier transform. The goal is to compute the frequency spectrum, determine the Fourier series coefficients, reconstruct the original signal, and analyze the effect of the number of coefficients on the convergence of the reconstructed signal.

2 Square Wave Signal

A square wave signal with a frequency of $f_0 = 100\,\text{Hz}$ and amplitude $A = 1$ is considered. The signal is defined as:

$$x(t) = A \cdot \text{sign}(\sin(2\pi f_0 t))$$

where $\text{sign}(\cdot)$ is the signum function.

3 Fourier Series Coefficients

The Fourier series coefficients for a square wave are given by:

$$c_n = \frac{4A}{\pi n}, \quad \text{where } n = 1, 3, 5, \ldots$$

These coefficients represent the amplitude of the odd harmonics in the frequency domain.

4 Frequency Spectrum

The frequency spectrum of the square wave is computed and plotted. The spectrum consists of discrete frequencies at odd harmonics of the fundamental frequency f_0. The amplitude of each harmonic decreases as the harmonic number n increases.

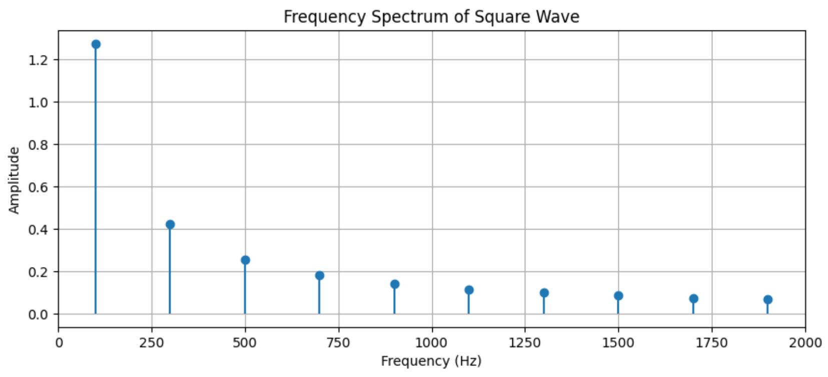

Figure 1: Frequency Spectrum of the Square Wave

5 Signal Reconstruction

The original square wave is reconstructed using the Fourier series synthesis equation:

$$x(t) = \sum_{n=1,3,5,\ldots} c_n \sin(2\pi n f_0 t)$$

The reconstructed signal is compared with the original signal in the time domain.

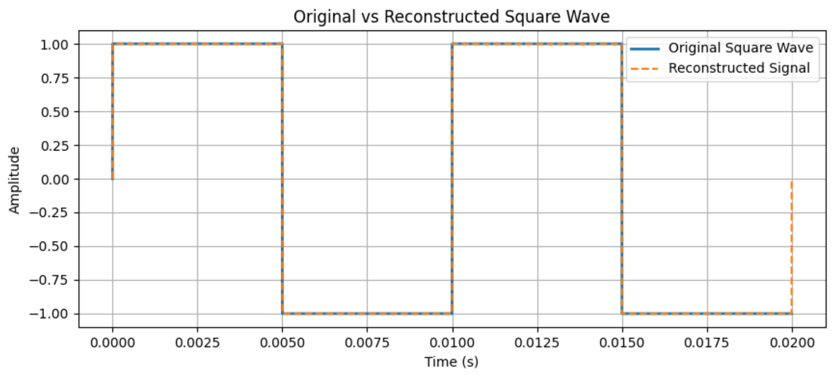

Figure 2: Original vs Reconstructed Square Wave

6 Effect of the Number of Coefficients

The number of Fourier coefficients used in the synthesis equation affects the convergence of the reconstructed signal:

- **Few Coefficients:** The reconstructed signal exhibits significant oscillations (Gibbs phenomenon) near the discontinuities.
- **Many Coefficients:** The reconstructed signal converges more closely to the original square wave, but the Gibbs phenomenon persists.
- **Theoretical Limit ($n_{\mathbf{max}} \to \infty$):** The reconstructed signal matches the original square wave perfectly, except at the discontinuities.

7 Conclusion

The project demonstrates the use of Fourier series to analyze and reconstruct a square wave signal. The frequency spectrum and Fourier series coefficients provide insights into the signal's frequency components. The reconstruction process highlights the trade-off between the number of coefficients and the accuracy of the reconstructed signal. The Gibbs phenomenon is observed near the discontinuities, which is a fundamental limitation of Fourier series for discontinuous signals.

Appendix: Python Code

The following Python code was used to perform the analysis:

```
import numpy as np
import matplotlib.pyplot as plt

# Parameters
A = 1  # Amplitude
f0 = 100  # Fundamental frequency (Hz)
T = 1 / f0  # Period
t = np.linspace(0, 2*T, 10000)  # Time vector (high resolution)
```

```
# Generate square wave
square_wave = A * np.sign(np.sin(2 * np.pi * f0 * t))

# Fourier Series coefficients
n_max = 100000  # Number of harmonics (odd only)
frequencies = np.arange(1, n_max+1, 2)  # Odd harmonics
coefficients = (4 * A) / (np.pi * frequencies)  # Fourier coefficients

# Frequency spectrum
plt.figure(figsize=(10, 4))
plt.stem(frequencies * f0, coefficients, basefmt=" ")
plt.title("Frequency Spectrum of Square Wave")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Amplitude")
plt.xlim(0, 2000)  # Limit x-axis to show relevant harmonics
plt.grid()
plt.savefig("frequency_spectrum.png")  # Save the plot
plt.show()

# Reconstruct the square wave using Fourier Series
reconstructed_signal = np.zeros_like(t)
for n, cn in zip(frequencies, coefficients):
    reconstructed_signal += cn * np.sin(2 * np.pi * n * f0 * t)

# Plot original and reconstructed signals
plt.figure(figsize=(10, 4))
plt.plot(t, square_wave, label="Original Square Wave", linewidth=2)
plt.plot(t, reconstructed_signal, label="Reconstructed Signal", linestyle="--", linewidth=1.5)
plt.title("Original vs Reconstructed Square Wave")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.legend()
plt.grid()
plt.savefig("reconstructed_signal.png")  # Save the plot
plt.show()
```