International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Fourier Series Approximation of Square and Sawtooth Waves Using Python Script in Jupyter Notebook

Dr. Vivek Parkash

Assistant Professor of Mathematics, Dyal Singh College, Karnal (Haryana), India
Email: *lethal007[at]hotmail. com*

Abstract: ***Understanding Fourier series involves investigating into a branch of mathematics known as harmonic analysis, specifically focusing on representing periodic functions as the sum of sine and cosine functions with different frequencies. The relation between Fourier series and mathematics is deep - rooted and spans across various branches of mathematics, physics, engineering, and computer science. It serves as a foundational concept for understanding the representation, analysis, and synthesis of periodic and non - periodic functions, making it indispensable in both theoretical research and practical applications. Analysing square wave and sawtooth waveforms involves understanding their mathematical representations and properties. In the ongoing paper, idea is to generate and analyse these waveforms using various mathematical and numerical libraries like NumPy and Matplotlib in Python.***

Keywords: Mathematics, Python, Square wave, Sawtooth wave, NumPy, Matplotlib, Fourier Series

1. Introduction

The Fourier series is a mathematical tool that allows periodic functions to be represented as the sum of simpler trigonometric functions—specifically, sine and cosine functions. The French mathematician Joseph Fourier developed it in the early 19th century. Python is a powerful programming language that is widely used in various fields, including mathematics. It provides numerous libraries and tools that make it particularly suited for mathematical computations, data analysis, and scientific computing. Generating specific piecewise continuous functions like square waves and sawtooth waves can be accomplished using Python libraries like NumPy, SciPy, and Matplotlib. Therefore, Python is well - suited for working with Fourier series which is a fundamental concept in signal processing, image processing, and various other fields of mathematics and engineering.

Square Wave:

1) **Description:** A square wave is a non - sinusoidal waveform where the signal switches between two levels (high and low) at regular intervals.
2) **Characteristics:**
   - The waveform alternates between two levels (e. g., +1 and - 1, or 0 and 1).
   - The transitions between the levels are abrupt, forming a square shape (hence the name).
   - The duty cycle, which is the ratio of the duration of the high level to the total period, determines the width of the high and low portions of the waveform.
   - It contains odd harmonics of the fundamental frequency.
3) **Sawtooth Wave:**
   - **Description:** A sawtooth wave is a non - sinusoidal waveform characterized by a linear rise in voltage (or signal value) followed by a sudden drop back to the starting level.
4) **Characteristics:**
   - The waveform rises linearly (or exponentially, depending on the type) from a minimum value to a maximum value (often from 0 to 1).
   - After reaching the maximum, the waveform drops suddenly back to the minimum value and starts rising again.
   - The rate of rise and fall can vary, affecting the frequency spectrum of the waveform.
   - It contains odd as well as even harmonics of the fundamental frequency.

Fourier approximation of square wave:

Analysing a square wave using Fourier series in Python involves decomposing the waveform into its sinusoidal components. A square wave alternates between two constant values over equal intervals of time. Various steps involved in this analysis are:

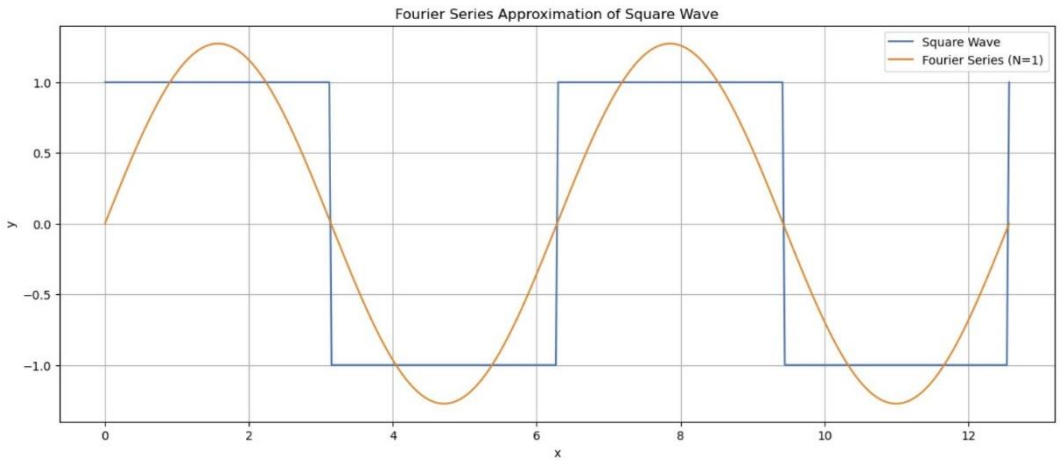
- **square_wave (x, T):** This function generates the square wave values based on the period T.
- **fourier_series_coeff (n, T):** This function calculates the Fourier coefficients a_n *and* b_n
- **fourier_series (x, T, N):** This function computes the Fourier series approximation up to N terms and returns the summed series.
- **Plotting**: The script plots the original square wave and several Fourier series approximations with increasing numbers of terms N.

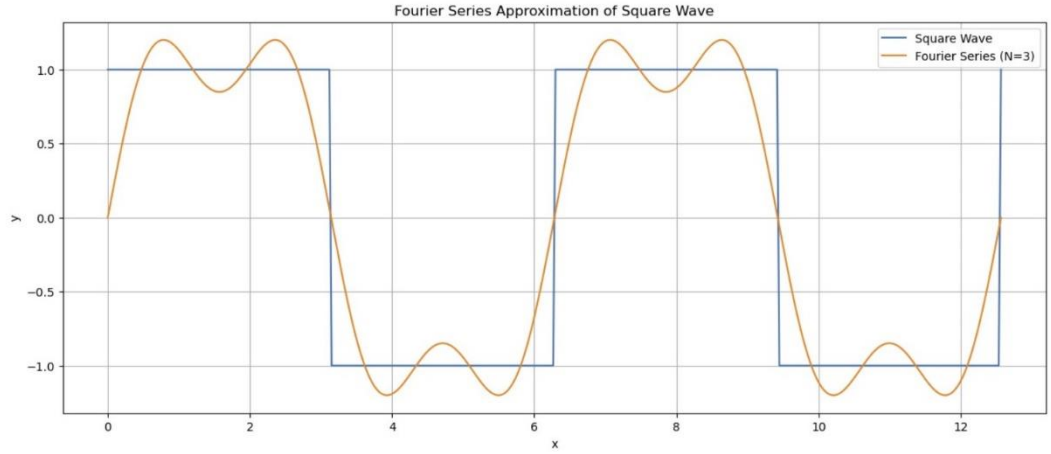
We can adjust the parameters and number of terms N as needed to see how the Fourier series improves its approximation of the square wave.

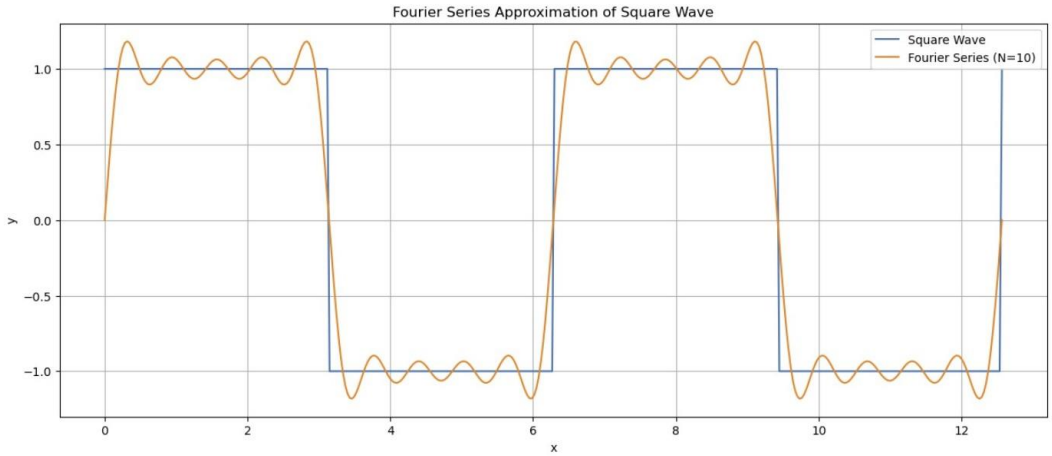
Python script for the approximation analysis of square wave is as below:

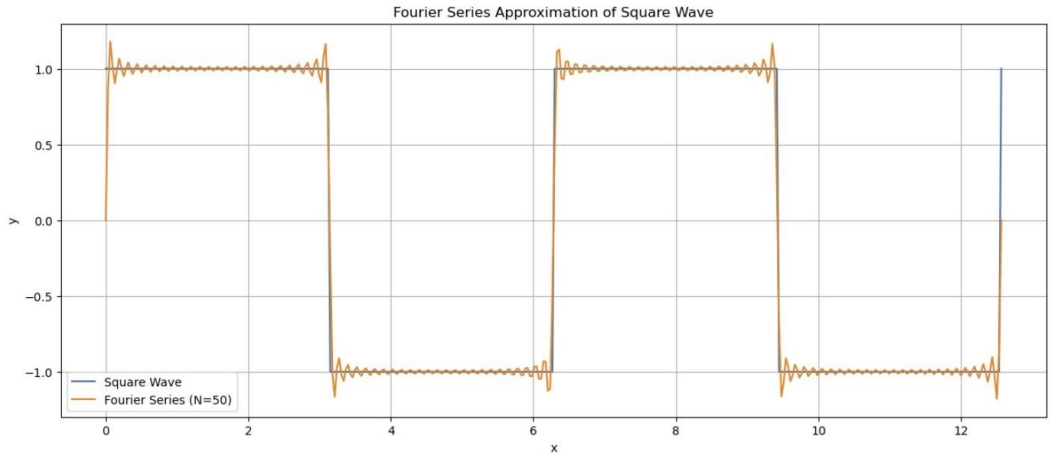
```python
import numpy as np
import matplotlib.pyplot as plt

# Function to generate a square wave
def square_wave(x, T):
    return np.where(np.mod(x, T) < T/2, 1, -1)

# Function to calculate Fourier series coefficients
def fourier_series_coeff(n, T):
    if n == 0:
        return 0
    else:
        return (2/(n*np.pi)) * (1 - (-1)**n)

# Function to calculate Fourier series approximation
def fourier_series(x, T, N):
    a0 = 0.0
    partial_sum = a0 * np.ones_like(x)
    for n in range(1, N+1):
        partial_sum += fourier_series_coeff(n, T) * np.sin(2 * np.pi * n * x / T)
    return partial_sum
```

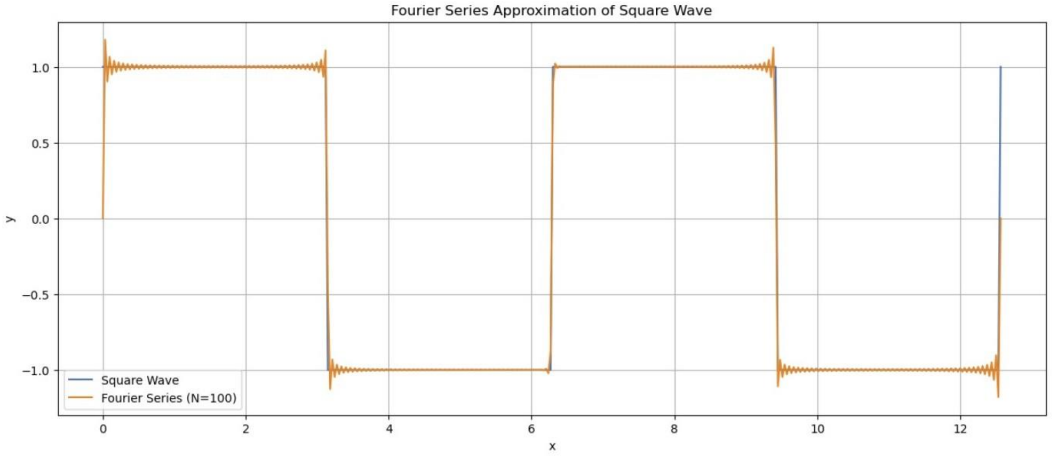
```python
# Generate x values
T = 2 * np.pi  # Period of the square wave
x = np.linspace(0, 4*np.pi, 400)  # x values from 0 to 4*pi

# Generate the square wave
square_wave_values = square_wave(x, T)

# Compute Fourier series with different number of terms
N_values = [1]
plt.figure(figsize=(10, 6))
plt.plot(x, square_wave_values, label='Square Wave')

for N in N_values:
    y = fourier_series(x, T, N)
    plt.plot(x, y, label=f'Fourier Series (N={N})')

plt.title('Fourier Series Approximation of Square Wave')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True)
plt.show()
```




Fourier Series Approximation of Square Wave
Square Wave
Fourier Series (N=3)
y
x


Fourier Series Approximation of Square Wave
Square Wave
Fourier Series (N=10)
y
x


Fourier Series Approximation of Square Wave
Square Wave
Fourier Series (N=50)
y
x



Fourier approximation of sawtooth wave:

Analysing a sawtooth wave using Fourier series in Python involves decomposing the waveform into its sinusoidal components. A sawtooth wave linearly ramps up and then sharply drops back to its starting point within a given period. Various steps involved in this analysis are:

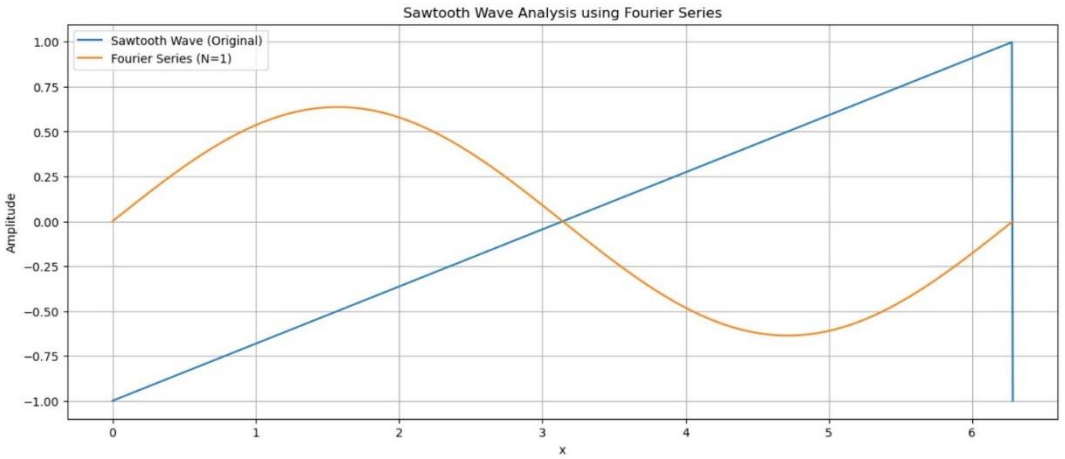
- **sawtooth_wave (x, period):** Defines the sawtooth wave function that linearly ramps up from - 1 to 1 over the specified period.
- **compute_fourier_coefficients (num_terms):** Computes the Fourier series coefficients a_n *and* b_n up to the specified number of terms.
- **fourier_series (x, period, num_terms):** Constructs the Fourier series approximation by summing the sine functions weighted by their respective coefficients.
- **Plotting**: Visualizes both the original sawtooth wave function and its Fourier series approximation using matplotlib.
- **Number of Terms**: Adjust num_terms in the fourier_series function to increase or decrease the accuracy of the Fourier series approximation.
- **Period and Range**: Modify the period parameter in both the sawtooth wave function and the plotting section to analyse different ranges and frequencies of sawtooth waves.

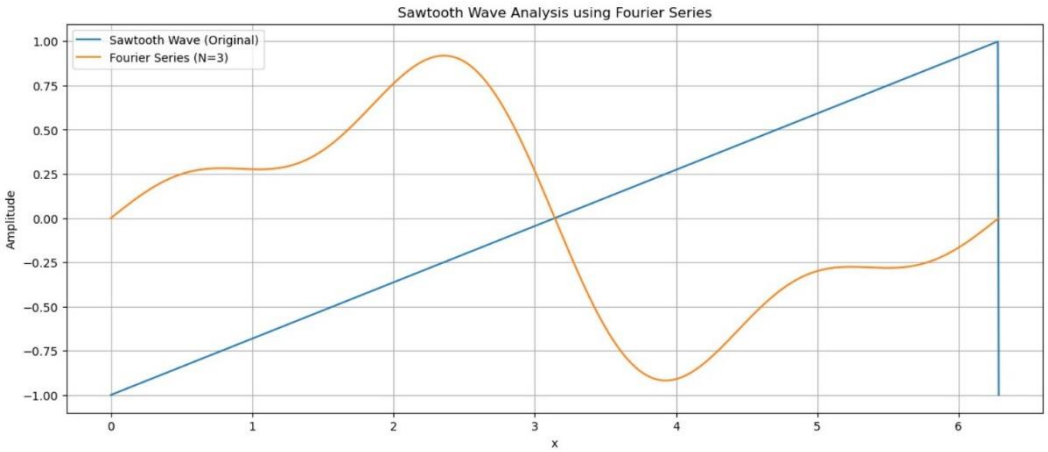
Python script for the approximation analysis of sawtooth wave is as below:

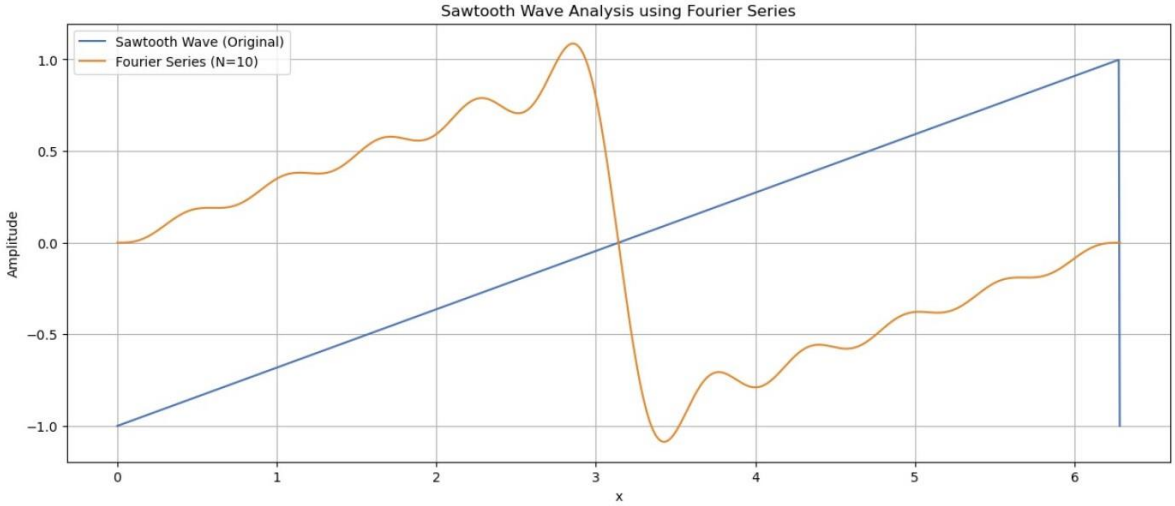
```
import numpy as np

def sawtooth_wave(x, period):
    x_norm = x % period
    return (2 * x_norm / period) - 1.0
def compute_fourier_coefficients(num_terms):
    coefficients = []
    for n in range(1, num_terms + 1):
        coefficient = (2 * (-1)**(n + 1)) / (n * np.pi)
        coefficients.append(coefficient)
    return coefficients
def fourier_series(x, period, num_terms):
    series_value = 0.0
    coefficients = compute_fourier_coefficients(num_terms)

    for n in range(num_terms):
        series_value += coefficients[n] * np.sin((n + 1) * 2 * np.pi * x / period)

    return series_value
import matplotlib.pyplot as plt
```

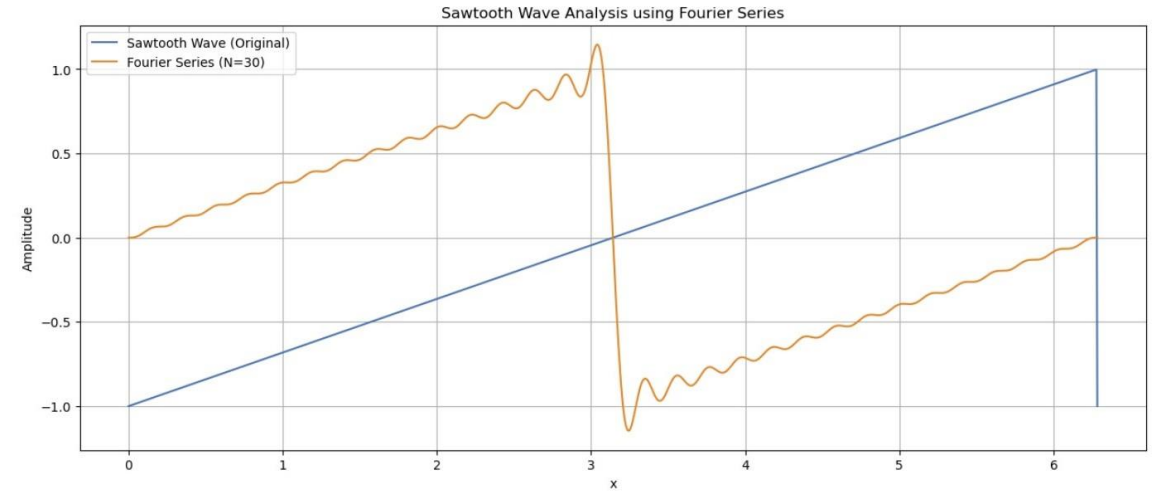
```
# Parameters
period = 2 * np.pi  # Period of the sawtooth wave
num_terms = 3  # Number of terms in the Fourier series approximation

# Generate x values
x_values = np.linspace(0, period, 1000)

# Compute sawtooth wave values
sawtooth_values = np.array([sawtooth_wave(x, period) for x in x_values])

# Compute Fourier series approximation
fourier_series_values = np.array([fourier_series(x, period, num_terms) for x in x_values])
```

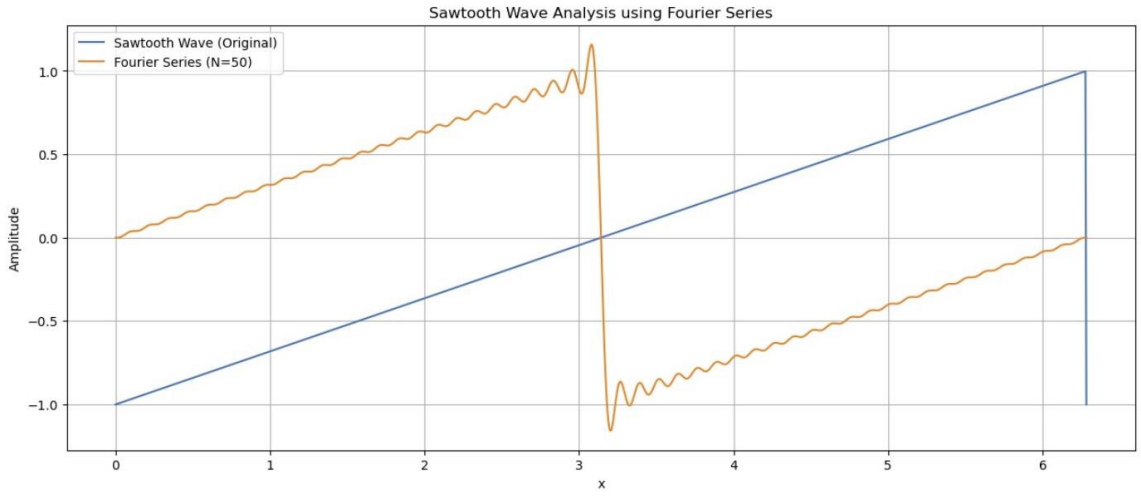
```
# Plotting
plt.figure(figsize=(15, 6))

plt.plot(x_values, sawtooth_values, label='Sawtooth Wave (Original)')
plt.plot(x_values, fourier_series_values, label=f'Fourier Series (N={num_terms})')

plt.title('Sawtooth Wave Analysis using Fourier Series')
plt.xlabel('x')
plt.ylabel('Amplitude')
plt.legend()

plt.grid(True)
plt.show()
```

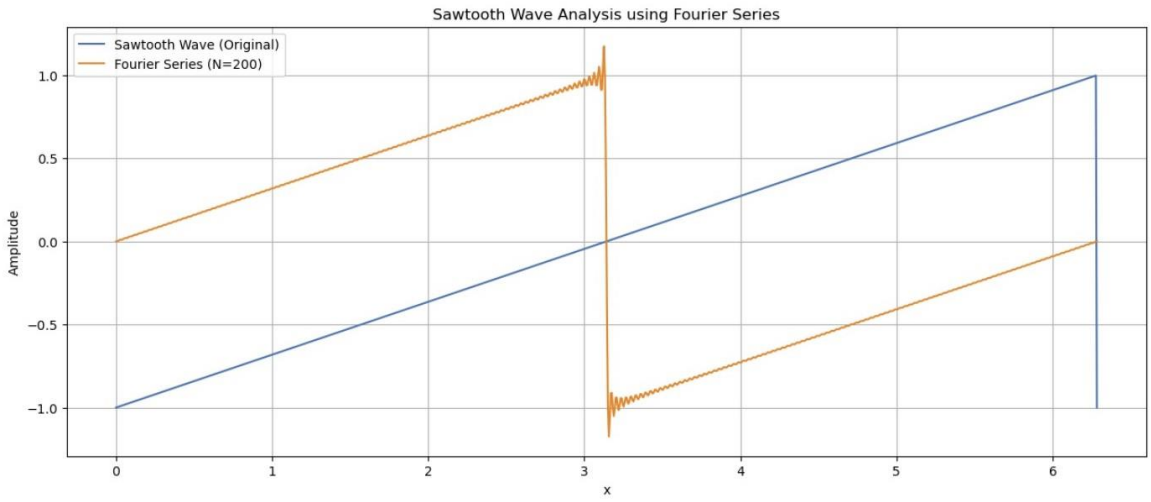












2. Conclusion

The Python script demonstrated above calculates and plots the Fourier series approximation of a square wave for the first 1 term, 3 terms, 10 terms, 50 terms, 100 terms and of sawtooth wave using the first 1 term, 3 terms, 10 terms, 30 terms, 50 terms and 200 terms respectively. Adjustments can be made to the function and coefficients depending on the specific periodic function under study and the desired precision of the approximation. As we increase the number of terms, we can see from above that the approximation improves and converges closer to the original square wave function. This paper provides a structured approach to Fourier series analysis of square wave and sawtooth wave in Python, focusing on understanding the decomposition and approximation of periodic functions using sinusoidal components. By implementing and visualizing the Fourier series, one can gain insights into the frequency content and representation quality of the original function, making it a powerful tool in both theoretical analysis and practical applications.

References

[1] https://lpsa.swarthmore.edu/Fourier/Series/DerFS.html
[2] https://firsttimeprogrammer.blogspot.com/2015/04/fourier-series-and-square-wave.html
[3] https://dynamics-and-control.readthedocs.io/en/latest/1_Dynamics/8_Frequency_domain/Fourier%20series.html
[4] https://commons.m.wikimedia.org/w/index.php?title=File:Analysis_of_Fourier_series_using_Python_Code.pdf&page=2
[5] https://www.educative.io/answers/how-to-implement-fourier-series-in-python
[6] https://commons.m.wikimedia.org/w/index.php?title=File:Analysis_of_Fourier_series_using_Python_Code.pdf&page=2