Visualizing the Gibbs Phenomenon

111-2 Data Science and Computer Programming Final Project

Hsin-Yen, Chiang

June 9, 2023

1 Abstract

This project visualize the Gibbs phenomenon in Fourier analysis using Python with `numpy` and `matplotlib` libraries.

2 Introduction

Fourier series is often used to represent periodic functions as a sum of sine and cosine functions. It allows us to express complex functions in terms of simpler harmonic components. However, when attempting to approximate a function with jump discontinuities, such as Heaviside step function

$$H(x) := \begin{cases} 0, & x < 0 \\ \frac{1}{2}, & x = 0 \\ 1, & x > 0 \end{cases}$$

or a square wave, the Gibbs phenomenon occurs: the Fourier series ”overshoots”, and the overshoot of the maxima and minima is about 9 percent of the jump in the function for each.

We consider a function $f : \mathbb{R} \to \mathbb{R}$ defined by

$$f(x) = \begin{cases} a, & -\pi \le x < 0 \\ b, & 0 \le x \le \pi \end{cases}$$

and suppose $a < b$. Let $s_n(x)$ denote the n-th partial sum of the Fourier series. Then the maximum of s_n occurs at $\pi/2n$ and the minimum at $-\pi/2n$ and

$$\lim_{n\to\infty} s_n\left(\frac{\pi}{2n}\right) = \left(\frac{b-a}{2}\right)\left(\frac{2}{\pi}\int_0^\pi \frac{\sin t}{t}\,dt + 1\right) + b \approx (b-a)(0.089) + b.$$

Similarly,

$$\lim_{n\to\infty} s_n\left(-\frac{\pi}{2n}\right) = \left(\frac{b-a}{2}\right)\left(-\frac{2}{\pi}\int_0^\pi \frac{\sin t}{t}\,dt + 1\right) + a \approx a - (b-a)(0.089).$$

We see that the overshoot of the maxima and minima are about 9 percent of the jump $b - a$.

3 Result

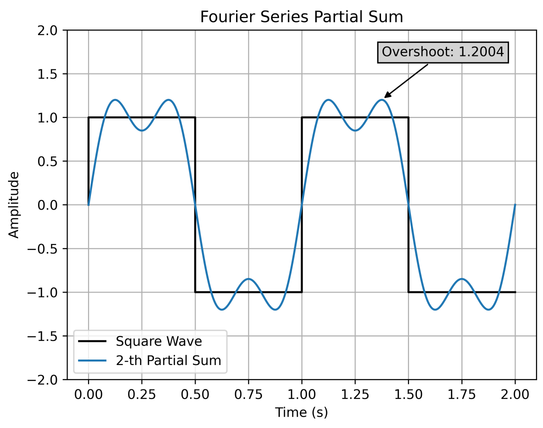
We first import the library that we needed.

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

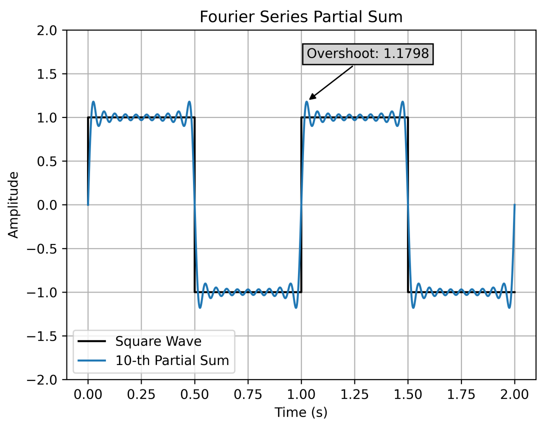
Then we a graph with the square wave function in black and the n-partial sum of its Fourier series in blue.

```python
amplitude = 1.0
frequency = 1.0
duration = 2.0
sampling_rate = 100000
n = 20

t = np.linspace(0, duration, int(sampling_rate * duration), endpoint=False)
square_wave = amplitude * np.sign(np.sin(2 * np.pi * frequency * t))

partial_sum = np.zeros_like(t)
for k in range(1, n+1):
    partial_sum += (4 / (np.pi * (2*k-1))) * np.sin((2 * np.pi * (2*k-1) * frequency * t))

plt.plot(t, square_wave, label='Square Wave', color='black')
plt.plot(t, partial_sum, label=f'{n}-th Partial Sum')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Fourier Series Partial Sum')
plt.ylim(-2, 2)
plt.legend(loc='lower left')
plt.grid(True)

# Find the maximum overshoot point and add a label
overshoot_index = np.argmax(partial_sum)  # Find the index of the maximum overshoot
overshoot_time = t[overshoot_index]  # Get the corresponding time value
overshoot_amplitude = partial_sum[overshoot_index]  # Get the overshoot amplitude
plt.annotate(f'Overshoot: {overshoot_amplitude:.4f}',
             xy=(overshoot_time, overshoot_amplitude),
             xytext=(overshoot_time, overshoot_amplitude + 0.5),
             arrowprops=dict(facecolor='black', arrowstyle='-|>'),
             bbox=dict(boxstyle='square', facecolor='lightgray'))
plt.show()
```

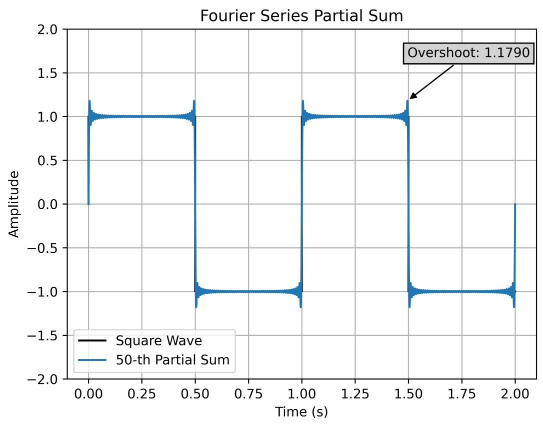
The result is the followings:



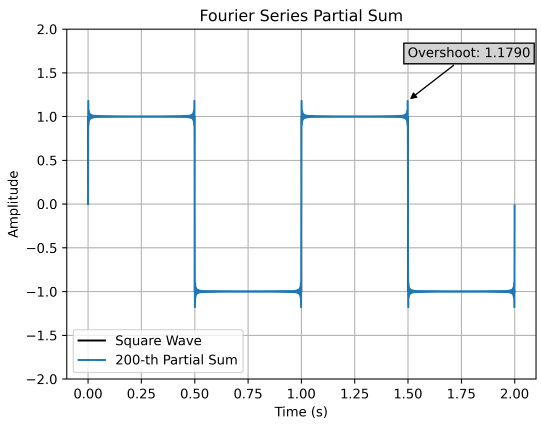
(a) 2-nd partial sum



(b) 10-th partial sum



(c) 50-th partial sum



(d) 200-th partial sum

Figure 1: Fourier Series Approaching the Square Wave

In addition, we graph the relation between the value n and the percentage of overshot.

```
n_values = np.arange(1, 50)  # Specify the range of n values to consider
overshoot_values = []

for n in n_values:
    t = np.linspace(0, duration, int(sampling_rate * duration), endpoint=False)
    partial_sum = np.zeros_like(t)
    for k in range(1, n+1):
        partial_sum += (4 / (np.pi * (2*k-1))) * np.sin((2 * np.pi * (2*k-1) * frequency * t))
    # Calculate the overshoot
    overshoot = (np.max(partial_sum) - amplitude) * 50.0
    overshoot_values.append(overshoot)

plt.plot(n_values, overshoot_values, color='black')
plt.xlabel('Number of Terms (n)')
plt.ylabel('Overshoot Percentage (%)')
```

```
    plt.title('Relationship between n and Overshoot Percentage')
    plt.grid(True)
    plt.show()
```

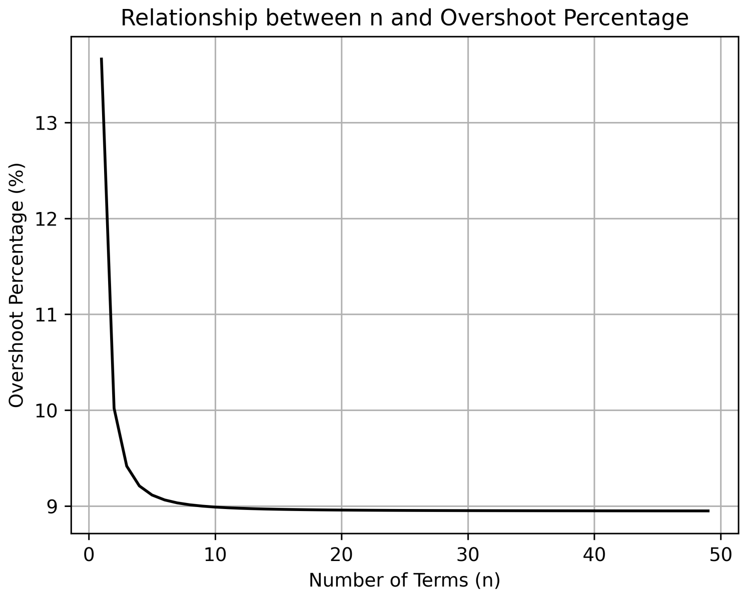


Figure 2: Relation Between n and the Overshoot Percentage

4 Conclusion

Through the implementation of the program, we successfully visualized the overshoots at the jump discontinuity of the square wave function. The figure 1 shows that the Fourier series is approaching to the function as n become larger. On the other hand, the figure 2 provides a visual representation of the approaching of the overshoot percentage to the theory gave.

Overall, this visualization may deepen the students' understanding of the Gibbs phenomenon. However, the limit of current program is that an easy way to enter various function with jump discontinuity is not provided.

References

[1] Marsden, J. E. (2008). *Elementary Classical Analysis*, 2nd edition. W. H. Freeman.