Assignment on Fourier Series Approximation

1 Overview

In this offline assignment, you will need to implement a Fourier Series approximation of periodic functions from scratch in Python. The assignment is divided into modular steps where each component of the Fourier Series calculation is isolated for easier implementation and comprehension.

2 Background

The Fourier Series is a way to represent a periodic function $f(x)$ as an infinite sum of sines and cosines:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos\left(\frac{n\pi x}{L}\right) + b_n \sin\left(\frac{n\pi x}{L}\right) \right) \tag{1}$$

where:

- L is half the period of the function.
- a_0 is the average value (DC component) of the function over one period.
- a_n and b_n are Fourier coefficients that describe the contribution of cosine and sine terms, respectively, at the n-th harmonic.

The Fourier coefficients a_0, a_n, and b_n are calculated as follows:

$$a_0 = \frac{1}{2L} \int_{-L}^{L} f(x)\, dx \tag{2}$$

$$a_n = \frac{1}{L} \int_{-L}^{L} f(x) \cos\left(\frac{n\pi x}{L}\right) dx \tag{3}$$

$$b_n = \frac{1}{L} \int_{-L}^{L} f(x) \sin\left(\frac{n\pi x}{L}\right) dx \tag{4}$$

3 Problem Description

You are tasked with implementing a Python class, `FourierSeries`, that calculates and plots the Fourier Series approximation of a given periodic function. The application must be modular and handle multiple periods of the wave.

3.1 Requirements

1. Implement `__init__` to initialize the function, half-period L, and the number of terms.
2. Implement `calculate_a0()` to compute a_0 using numerical integration (e.g., `np.trapz`).
3. Implement `calculate_an(n)` to compute a_n for the n-th cosine term.
4. Implement `calculate_bn(n)` to compute b_n for the n-th sine term.
5. Implement `approximate(x)` that sums the calculated coefficients to approximate $f(x)$.
6. Implement `plot(ax, wave_type)` to visualize the original function and its Fourier series approximation over multiple periods. The plot must dynamically adjust axis limits based on the selected wave type.

3.2 Target Functions

You must define the following periodic functions in the `target_function(x, function_type)` method:

- **Square Wave:** +1 when sin(x) > 0, -1 otherwise
- **Sawtooth Wave:** Mathematical Sawtooth: y = x for $-\pi < x < \pi$
- **Triangle Wave:** Periodic line with slope +1 and -1 alternately
- **Cubic Wave:** $f(x) = x^3$ defined on the interval $[-1, 1]$ and repeated periodically. (Implies $L = 1$).
- **Pulse Train:** A periodic spike (e.g., 1.0) occurring at intervals of 2π.

Note that the Cubic wave requires a different half-period ($L = 1$) compared to the others ($L = \pi$).

4 Implementation Guidelines

- Use numerical integration (`np.trapz`) for calculating integrals.
- Implement the missing methods in the provided Python skeleton code.
- Ensure the `get_half_period(wave_type)` helper function returns the correct L for the selected wave.

5 Helper File

An executable file named `soln.exe` is provided along with this assignment. You can run this file to view the expected output and behavior of the application. Use this to verify that your implementation matches the required visualization and functionality.

6 Submission

Submit a single Python file containing the completed class and all necessary function implementations. Rename it with your student id and then submit the python file. **Deadline: February 02, 11:59 pm**

7 Mark Distribution

Table 1: Mark Distribution for Fourier Series Assignment

Component	**Points**
Initialization & Logic	10 points
Method calculate_a0()	15 points
Method calculate_an()	15 points
Method calculate_bn()	15 points
Method approximate()	20 points
Method plot()	10 points
Target Function Implementation (All 5 types)	15 points
Total	**100 points**