
Fourier Series Representation

SGTB Khalsa College, University of Delhi
Preetpal Singh(2020PHY1140)(20068567043)
Jagjyot Singh(2020PHY1005)(20068567028)

Unique Paper Code: 32221401

Paper Title: Mathematical Physics III

Submitted on: February 8, 2022

B.Sc(H) Physics Sem IV

Submitted to: Dr. Mamta

Theory

Dirichlet Conditions

The Dirichlet conditions are sufficient conditions for a real-valued, periodic function f to be equal to the sum of its Fourier series at each point where f is continuous. Moreover, the behavior of the Fourier series at points of discontinuity is determined as well (it is the midpoint of the values of the discontinuity).

- f must be absolutely integrable over a period.
- f must be of bounded variation in any given bounded interval.
- f must have a finite number of discontinuities in any given bounded interval, and the discontinuities cannot be infinite.

Dirichlet's Theorem

If f satisfies Dirichlet conditions, then for all x , we have that the series obtained by plugging x into the Fourier series is convergent, and is given by

$$\sum_{n=-\infty}^{\infty} a_n e^{inx} = \frac{f(x^+) + f(x^-)}{2}$$

where,

$$f(x^+) = \lim_{y \rightarrow x^+} f(y)$$
$$f(x^-) = \lim_{y \rightarrow x^-} f(y)$$

Fourier Representation of a Periodic Function

A Fourier series is an expansion of a periodic function $f(x)$ in terms of an infinite sum of sines and cosines. Fourier series make use of the orthogonality relationships of the sine and cosine functions

Consider the Fourier series of the function $f(t)$. Let's take a periodic piecewise continuous function $f(t + 2L) = f(t)$.

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos n\omega_0 t + b_n \sin n\omega_0 t) \quad (1)$$

Let's take the function in range $[-\frac{T}{2}, \frac{T}{2}]$ and period T . Multiply the equation (1) by $\cos m\omega_0 t$

$$f(t) \cos m\omega_0 t = \frac{a_0}{2} \cos m\omega_0 t + \sum_{n=1}^{\infty} (a_n \cos n\omega_0 t \cos m\omega_0 t + b_n \sin n\omega_0 t \cos m\omega_0 t)$$

Integrating in the range $[-\frac{T}{2}, \frac{T}{2}]$

$$\begin{aligned} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \cos m\omega_0 t dt &= \frac{a_0}{2} \int_{-\frac{T}{2}}^{\frac{T}{2}} \cos m\omega_0 t dt + \sum_{n=1}^{\infty} a_n \int_{-\frac{T}{2}}^{\frac{T}{2}} \cos n\omega_0 t \cos m\omega_0 t dt + \sum_{n=1}^{\infty} b_n \int_{-\frac{T}{2}}^{\frac{T}{2}} \sin n\omega_0 t \cos m\omega_0 t dt \\ &= \frac{a_0}{2} \cdot 0 + \sum_{n=1}^{\infty} a_n \frac{T}{2} \delta_{mn} + \sum_{n=1}^{\infty} b_n 0 = a_m \frac{T}{2} \\ a_m &= \frac{2}{T} \int_{-T/2}^{T/2} f(t) \cos m\omega_0 t dt \end{aligned}$$

Now Multiply eq (1) by $\sin m\omega_0 t$

$$f(t) \sin m\omega_0 t = \frac{a_0}{2} \sin m\omega_0 t + \sum_{n=1}^{\infty} (a_n \cos n\omega_0 t \sin m\omega_0 t + b_n \sin n\omega_0 t \sin m\omega_0 t)$$

Integrate in the range $[-\frac{T}{2}, \frac{T}{2}]$

$$\begin{aligned} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \sin m\omega_0 t dt &= \frac{a_0}{2} \int_{-\frac{T}{2}}^{\frac{T}{2}} \sin m\omega_0 t dt + \sum_{n=1}^{\infty} a_n \int_{-\frac{T}{2}}^{\frac{T}{2}} \cos n\omega_0 t \sin m\omega_0 t dt + \sum_{n=1}^{\infty} b_n \int_{-\frac{T}{2}}^{\frac{T}{2}} \sin n\omega_0 t \sin m\omega_0 t dt \\ &= \frac{a_0}{2} \cdot 0 + \sum_{n=1}^{\infty} a_n 0 + \sum_{n=1}^{\infty} b_n \frac{T}{2} \delta_{mn} = b_m \frac{T}{2} \\ b_m &= \frac{2}{T} \int_{-T/2}^{T/2} f(t) \sin m\omega_0 t dt \end{aligned}$$

To determine a_0 , Integrate equation 1 from $[-\frac{T}{2}, \frac{T}{2}]$

$$\begin{aligned} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) dt &= \frac{a_0}{2} \int_{-\frac{T}{2}}^{\frac{T}{2}} dt + \sum_{n=1}^{\infty} a_n \left(\int_{-\frac{T}{2}}^{\frac{T}{2}} \cos n\omega_0 t dt \right) + \sum_{n=1}^{\infty} b_n \int_{-\frac{T}{2}}^{\frac{T}{2}} \sin n\omega_0 t dt \\ &= \frac{a_0}{2} \cdot T + \sum_{n=1}^{\infty} a_n 0 + \sum_{n=1}^{\infty} b_n \cdot 0 = a_0 \frac{T}{2} \\ a_0 &= \frac{2}{T} \int_{-T/2}^{T/2} f(t) dt \end{aligned}$$

Questions

Obtain the Fourier Series Representation for the following functions:

- $f(x) = \begin{cases} 0, & -1 < x < 0 \\ 1, & 0 < x < 1 \end{cases}$
and $f(x+2) = f(x)$

The period is in $[-L, L] = [-1, 1]$ as $\begin{matrix} 2L = 2 \\ L = 1 \end{matrix}$

$$\begin{aligned} a_0 &= \frac{1}{L} \int_{-L}^L f(x) dx = \frac{1}{1} \int_{-1}^1 f(x) dx = \int_0^1 dx = 1 \\ \therefore \frac{a_0}{2} &= \frac{1}{2} \\ a_n &= \frac{1}{L} \int_{-L}^L f(x) \cos \frac{n\pi x}{L} dx \\ &= \frac{1}{1} \int_{-1}^1 f(x) \cos \frac{n\pi x}{1} dx \\ &= \int_0^1 \cos n\pi x dx \\ &= \left. \frac{\sin n\pi x}{n\pi} \right|_0^1 = 0 \end{aligned}$$

so, $a_n = 0$ if $n \neq 0$

$$\begin{aligned}
b_n &= \frac{1}{L} \int_{-1}^1 f(x) \sin \frac{n\pi x}{L} dx \\
&= \int_{-1}^0 -0.5 \sin n\pi x dx + \int_0^1 0.5 \sin n\pi x dx \\
&= \frac{-1}{2} \left[\frac{-\cos n\pi x}{n\pi} \right]_{-1}^0 + \frac{1}{2} \left[-\frac{\cos n\pi x}{n\pi} \right]_0^1 \\
&= \frac{-1}{2} \left[\frac{-1}{n\pi} + \frac{\cos n\pi}{n\pi} \right] + \frac{1}{2} \left[\frac{-\cos n\pi}{n\pi} + \frac{1}{n\pi} \right] \\
&= \frac{1}{2n\pi} - \frac{\cos n\pi}{2n\pi} - \frac{\cos n\pi}{2n\pi} + \frac{1}{2n\pi} \\
&= \frac{1}{n\pi} (1 - \cos n\pi) \\
&= \begin{cases} 0, & \text{if } n \text{ is even} \\ \frac{2}{n\pi}, & \text{if } n \text{ is odd} \end{cases}
\end{aligned}$$

Here $a_n = 0$, and a_0, b_n gives some value only when n is odd, so The given function is neither odd nor even.

The fourier series is thus,

$$\begin{aligned}
f(x) &= 0 + \sum_{n=1}^{\infty} 0 \times \cos n\pi x + \sum_{n=1, \text{odd}}^{\infty} \frac{2}{n\pi} \sin \pi x n \\
&= \frac{2}{\pi} \left[\sin \pi x + \frac{1}{3} \sin 3\pi x + \dots \right]
\end{aligned}$$

$$\bullet f(x) = \begin{cases} 0, & -1 < x < -0.5 \\ 1, & -0.5 < x < 0.5 \\ 0, & 0.5 < x < 1 \end{cases} \quad \text{and } f(x+2) = f(x)$$

The period is in $[-L, L] = [-1, 1]$ as $\begin{matrix} 2L = 2 \\ L = 1 \end{matrix}$

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

$$a_0 = \frac{1}{L} \int_{-L}^L f(x) dx = \frac{1}{1} \int_{-1}^1 f(x) dx = \int_{-0.5}^{0.5} 1 dx = [x]_{-0.5}^{0.5} = 1$$

$$\therefore \frac{a_0}{2} = \frac{1}{2}$$

$$a_n = \frac{1}{L} \int_{-L}^L f(x) \cos \frac{n\pi x}{L} dx = \frac{1}{1} \int_{-1}^1 f(x) \cos n\pi x dx = \int_{-0.5}^{0.5} 1 \cdot \cos n\pi x dx = \left[\frac{\sin n\pi x}{n\pi} \right]_{-0.5}^{0.5}$$

$$= \frac{1}{n\pi} \left[\sin \frac{n\pi}{2} + \sin \frac{n\pi}{2} \right] = \frac{2}{n\pi} \sin \frac{n\pi}{2}$$

$$= \begin{cases} 0, & \text{if } n \text{ is even} \\ \frac{2}{n\pi} \sin \frac{n\pi}{2}, & \text{if } n \text{ is odd} \end{cases}$$

$$b_n = \frac{1}{L} \int_{-L}^L f(x) \sin \frac{n\pi x}{L} dx = \int_{-1}^1 f(x) \sin n\pi x dx = \int_{-0.5}^{0.5} \sin n\pi x dx = \frac{1}{n\pi} (-\cos n\pi x) \Big|_{-0.5}^{0.5} = 0$$

Here $b_n = 0$, a_n, a_0 gives some value when n is odd, So, the given function is even when n is odd.

The fourier series is thus,

$$\begin{aligned} f(x) &= \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos \frac{n\pi x}{L} + b_n \sin \frac{n\pi x}{L} \right) \\ &= \frac{1}{2} + \sum_{n=1}^{\infty} 0 \cos \frac{n\pi x}{1} + \sum_{n=1, \text{odd}}^{\infty} \frac{2}{n\pi} \sin \frac{n\pi x}{1} \\ &= \frac{1}{2} + \frac{2}{\pi} \left[1 \sin \pi x + \frac{1}{3} \sin 3\pi x + \dots \right] \end{aligned}$$

$$\bullet \quad f(x) = \begin{cases} -0.5, & -1 < x < 0 \\ 0.5, & 0 < x < 1 \end{cases}$$

$$\text{and } f(x+2) = f(x)$$

The period is in $[-L, L] = [-1, 1]$ as $\frac{2L}{L} = 2$

$$a_0 = \frac{1}{L} \int_{-1}^1 f(x) dx = \int_{-1}^{-0.5} -0.5 dx + \int_{-0.5}^0 0.5 dx = -0.5 + 0.5 = 0$$

$$\begin{aligned} a_n &= \frac{1}{L} \int_{-1}^1 f(x) \frac{\cos n\pi x}{L} dx = \int_{-1}^{-0.5} -0.5 \cos n\pi x dx + \int_{-0.5}^0 0.5 \cos n\pi x dx \\ &= \frac{-1}{2} \left[\frac{\sin n\pi x}{n\pi} \right]_{-1}^0 + \frac{1}{2} \left[\frac{\sin n\pi x}{n\pi} \right]_0^1 = 0 \end{aligned}$$

$$b_n = \frac{1}{L} \int_{-L}^L f(x) \sin \frac{n\pi x}{L} dx = \frac{1}{L} \int_{-1}^L f(x) \sin \frac{n\pi x}{1} dx = \frac{1}{1} \int_{-1}^0 -0.5 \cdot \sin n\pi x + \int_0^1 0.5 \cdot \sin n\pi x dx$$

$$\begin{aligned} &= 0.5 \left[\frac{-\cos n\pi x}{n\pi} \right]_{-1}^0 + 0.5 \left[\frac{-\cos n\pi x}{n\pi} \right]_0^1 = -\frac{0.5}{n\pi} [(1 - \cos \pi) + (1 - \cos n\pi)] = -\frac{0.5}{n\pi} [2 - 2 \cos \pi] = -\frac{1}{n\pi} [1 - 1 \cos \pi] \\ &= \begin{cases} 0, & \text{if } n \text{ is even} \\ \frac{2}{n\pi}, & \text{if } n \text{ is odd} \end{cases} \end{aligned}$$

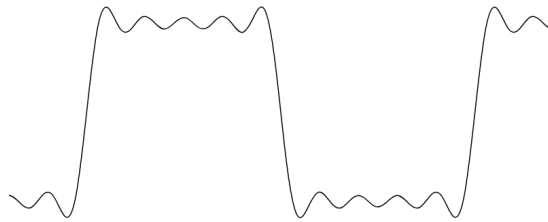
Here $a_n = 0$, and $a_0 = 0$, b_n gives some value only when n is odd, so The given function is neither odd nor even.

The fourier series is thus,

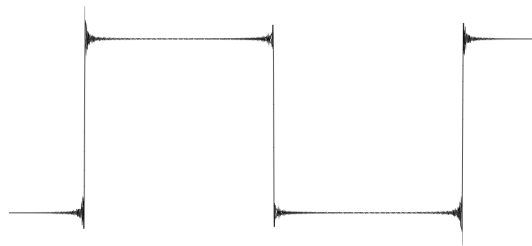
$$\begin{aligned} f(x) &= \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos \frac{n\pi x}{L} + b_n \sin \frac{n\pi x}{L} \right) = \sum_{n=1, \text{odd}}^{\infty} \frac{-2}{n\pi} \\ &= \frac{-2}{\pi} \left(1 + \frac{1}{3} + \frac{1}{5} + \dots \right) \end{aligned}$$

Explain Gibbs Phenomenon

The Gibbs phenomenon describes the fact that Fourier sums overshoot/undershoot at a jump discontinuity, and that this shoot can be reduced upto some level but can't be terminated by considering more terms to calculate the partial sum.



(a) Functional approximation of square wave using 5 harmonics



(b) Functional approximation of square wave using 125 harmonics

Figure 1: Gibbs Phenomenon

Half Range Expansion:

- How do you write the Fourier series representation for a function $f(x)$ that is defined in a finite range, say, $0 < x < L$?
- What do you mean by half range sine and cosine expansions.
- Write down the half range even and odd periodic extensions for the function defined as $f(x) = x$, $0 < x < \pi$.
- Derive cosine and sine Fourier representations for the above extensions.

If a function is defined over half the range, say 0 to L , instead of the full range from $-L$ to L , it may be expanded in a series of sine terms only or of cosine terms only. The series produced is then called a **Half Range Fourier Series**. The even and odd functions can be expanded in terms of sin and cos functions in half range. These expansions are called half range Sine and Cosine expansions.

Half range even and odd periodic extension of $f(t) = t$, $0 < t < \pi$.

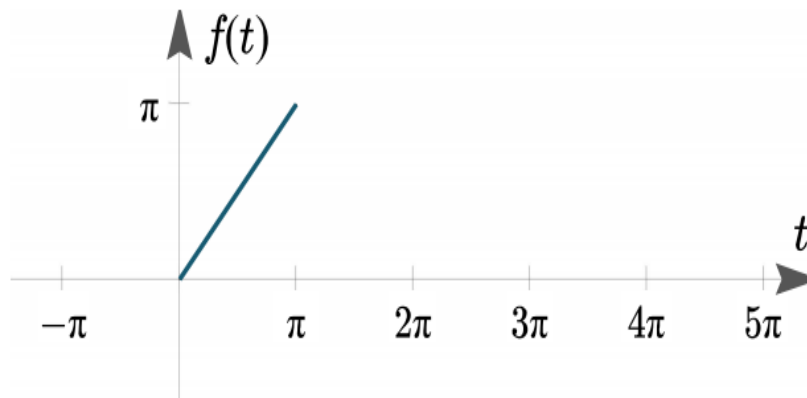
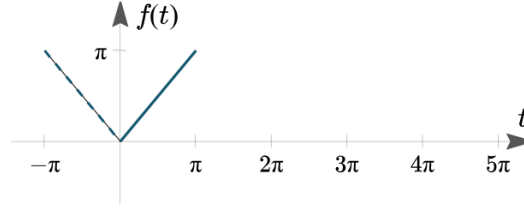
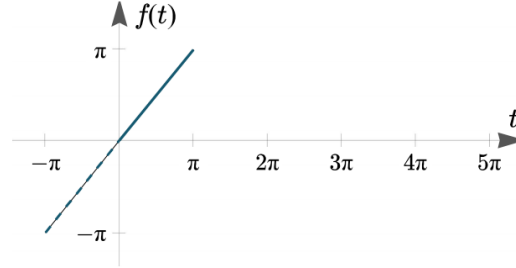


Figure 2: $f(t) = t$, $0 < t < \pi$.



(a) Even Function symmetrical about the $f(t)$ axis in $t = -\pi$ and $t = 0$



(b) Odd function symmetrical about origin between $t = -\pi$ and $t = 0$

Figure 3: Half Range Expansion

Half Range Even Expansion

$$a_0 = \frac{2}{L} \int_0^L f(t) dt = \frac{2}{\pi} \int_0^\pi t dt = \frac{2}{\pi} \left[\frac{t^2}{2} \right]_0^\pi = \frac{2}{\pi} \frac{\pi^2}{2} = \pi$$

$$a_n = \frac{2}{L} \int_0^L f(t) \cos \frac{n\pi t}{L} dt = \frac{2}{\pi} \int_0^\pi t \cos nt dt = \frac{2}{\pi} \left[\frac{1}{n^2} (\cos nt + nt \sin nt) \right]_0^\pi$$

$$= \frac{2}{\pi n^2} [(\cos n\pi + 0) - (\cos 0 + 0)] = \frac{2}{\pi n^2} [(\cos n\pi - 1)] = \frac{2}{\pi n^2} [(-1)^n - 1]$$

When n is odd, the expansion gives $-\frac{4}{\pi n^2}$.

When n is even, the expansion gives 0.

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos \frac{n\pi t}{L} = \frac{\pi}{2} - \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{\cos(2n-1)t}{(2n-1)^2} = \frac{\pi}{2} - \frac{4}{\pi} \left(\cos t + \frac{1}{9} \cos 3t + \frac{1}{25} \cos 5t + \dots \right)$$

Half Range Odd Expansion

$$f(t) = \sum_{n=1}^{\infty} b_n \frac{\sin n\pi t}{L} \quad n = 1, 2, 3, \dots$$

$$b_n = \frac{2}{L} \int_0^L f(t) \sin \frac{n\pi t}{L} dt$$

$$b_n = \frac{2}{\pi} \int_0^\pi \frac{1}{n^2} [\sin nt - nt \cos nt] = \frac{2}{\pi} \times \frac{1}{n^2} [\sin nt - nt \cos nt]_0^\pi$$

$$= \frac{2}{\pi} \times \frac{1}{n^2} [[\sin n\pi - n\pi \cos n\pi] - [\sin 0 - 0]] = \frac{2}{\pi} \times \frac{1}{n^2} [-n\pi \cos n\pi]$$

$$\cos n\pi = \begin{cases} (-1) & \text{if } n \text{ is odd} \\ 1 & \text{if } n \text{ is even} \end{cases}$$

$$= -\frac{2}{\pi} \times \frac{1}{n^2} \times (-1)^n$$

Even Function and Half Range Cosine Series

An even function can be expanded using half its range from

- 0 to L
- $-L$ to 0
- L to $2L$

That is, the range of integration is L . The Fourier series of the half range even function is given by:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos \frac{n\pi t}{L}$$

$$a_0 = \frac{2}{L} \int_0^L f(t) dt$$

$$a_n = \frac{2}{L} \int_0^L f(t) \cos \frac{n\pi t}{L} dt$$

and $b_n = 0$

Odd Function and Half Range Sine Series

An odd function can be expanded using half its range from 0 to L , i.e. the range of integration has value L . The Fourier series of the odd function is:

Since $a_0 = 0$ and $a_n = 0$, we have

$$f(t) = \sum_{n=1}^{\infty} b_n \sin \frac{n\pi t}{L} \text{ for } n = 1, 2, 3, \dots$$

$$b_n = \frac{2}{L} \int_0^L f(t) \sin \frac{n\pi t}{L} dt$$

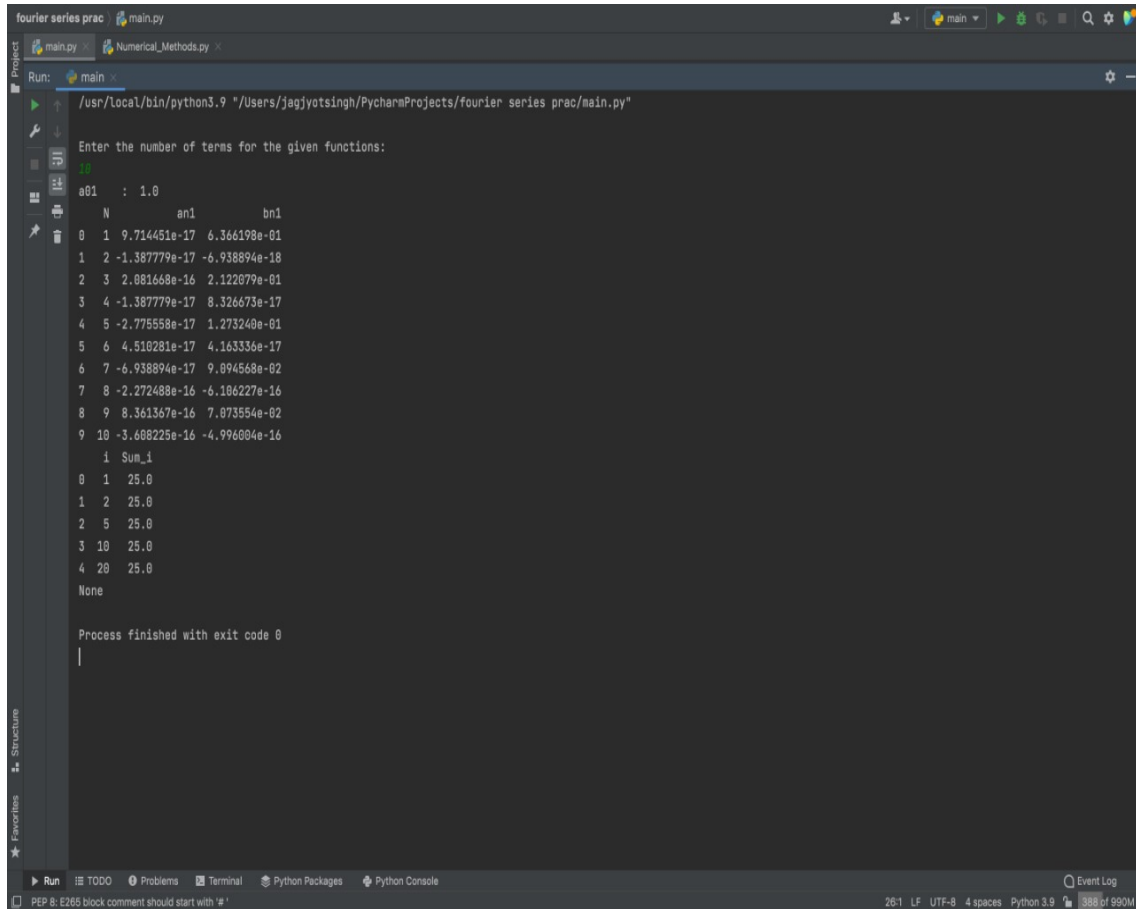
and $b_n = 0$

Algorithm

Algorithm 1 *FourierCoef*

```
1: function INPUT(method, tolerance d, function f discontinuities dis, halfrange L,  
   Number of coefficients N, even, odd or neither (var))  
2:                                     ▷ Here f is function to be integrated  
3:  
4:   For calculation of a0  
5:   for i in range(length of f) do                                     ▷ f is defined piecewise, integrated piecewise  
6:     upperlimit = discontinuity                                     ▷ discontinuity approached from left side in range of  
     function to right side  
7:     a0 += method()                                               ▷ to integrate the function between 'dis'  
8:     lowerlimit = upperlimit                                       ▷ to move towards right side in range  
9:   end for  
10:  For calculation of an and bn  
11:  A function is required to multiply cosine and sine with an and bn, so that we can use it to  
    integrate conveniently with our module  
12:  for i in range(length of f) do                                     ▷ f is defined piecewise, integrated piecewise  
13:    upperlimit = discontinuity                                     ▷ discontinuity approached from left side in range of  
    function to right side  
14:    an or bn += method(product)                                     ▷ to integrate the function between 'dis'  
15:    lowerlimit = upperlimit                                       ▷ to move towards right side in range  
16:  end for  
    return a0, an, bn
```

Function1



The screenshot shows a PyCharm IDE window with a project named "fourier series prac". The main.py file is open, and the Run console shows the output of the script. The script prompts the user to enter the number of terms, which is 10. It then displays a table of coefficients (a01, an1, bn1) for N=1 to 10. Finally, it shows the sum of the series for N=1, 2, 5, 10, and 20, all resulting in 25.0.

```
Run: main
/usr/local/bin/python3.9 "/Users/jagjyotsingh/PycharmProjects/fourier series prac/main.py"

Enter the number of terms for the given functions:
10

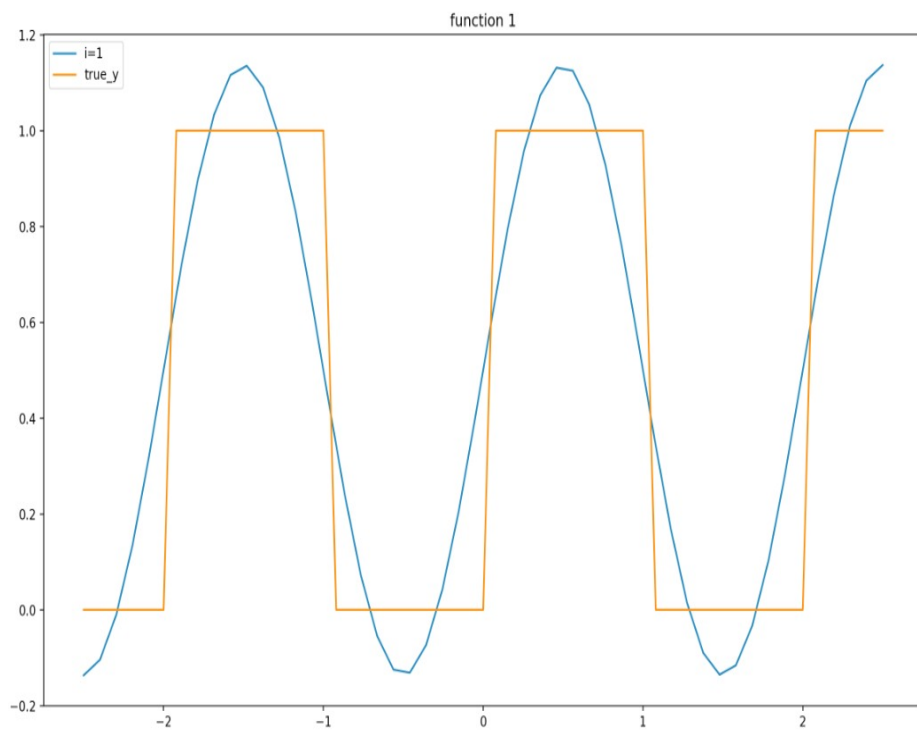
a01 : 1.0

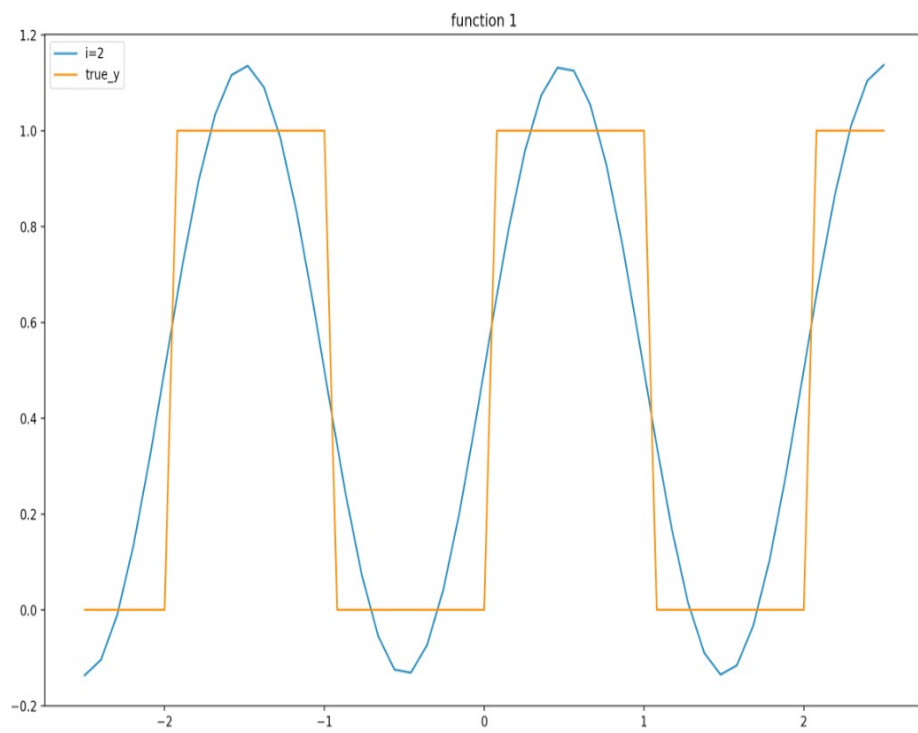
N      an1      bn1
0 1  9.714451e-17  6.366198e-01
1 2 -1.387779e-17 -6.938894e-18
2 3  2.081668e-16  2.122079e-01
3 4 -1.387779e-17  8.326673e-17
4 5 -2.775558e-17  1.273240e-01
5 6  4.510281e-17  4.163336e-17
6 7 -6.938894e-17  9.094568e-02
7 8 -2.272488e-16 -6.106227e-16
8 9  8.361367e-16  7.073554e-02
9 10 -3.688225e-16 -4.996004e-16

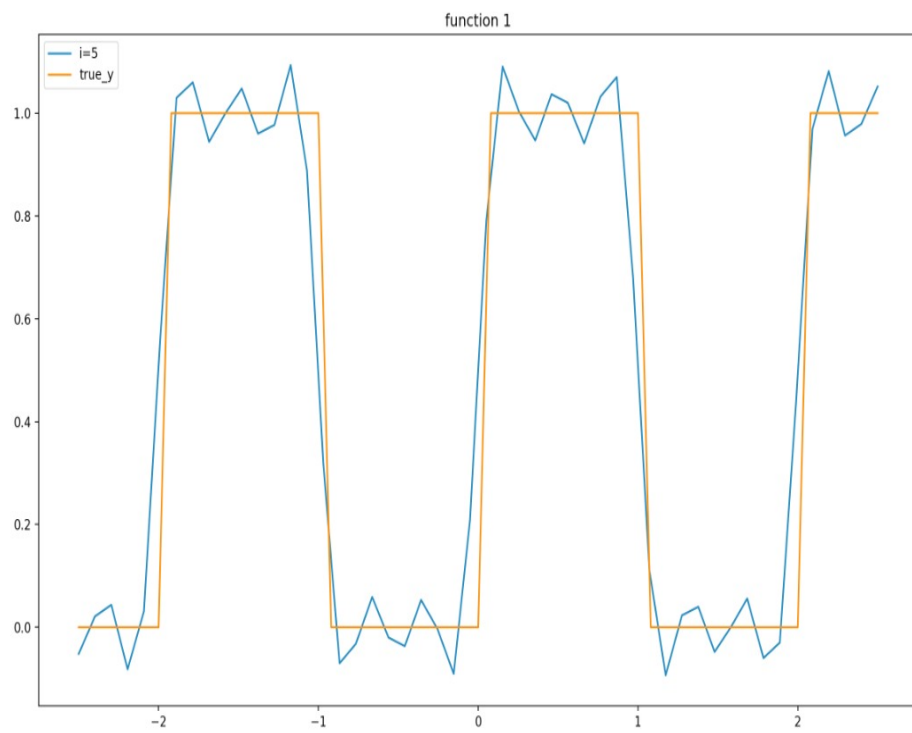
1 Sum_1
0 1  25.0
1 2  25.0
2 5  25.0
3 10 25.0
4 20 25.0

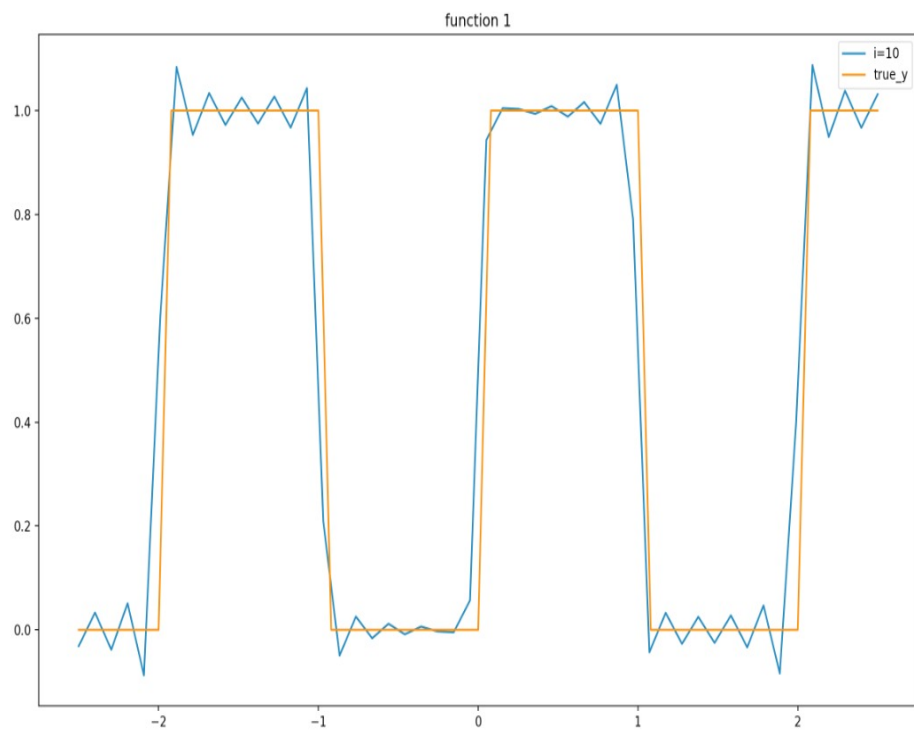
None

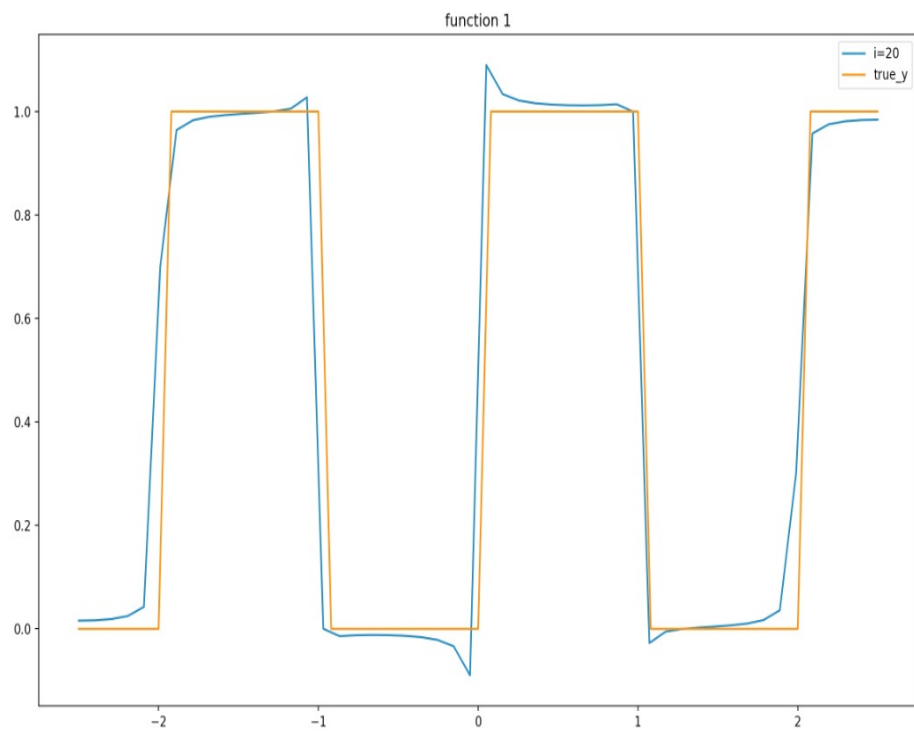
Process finished with exit code 0
```



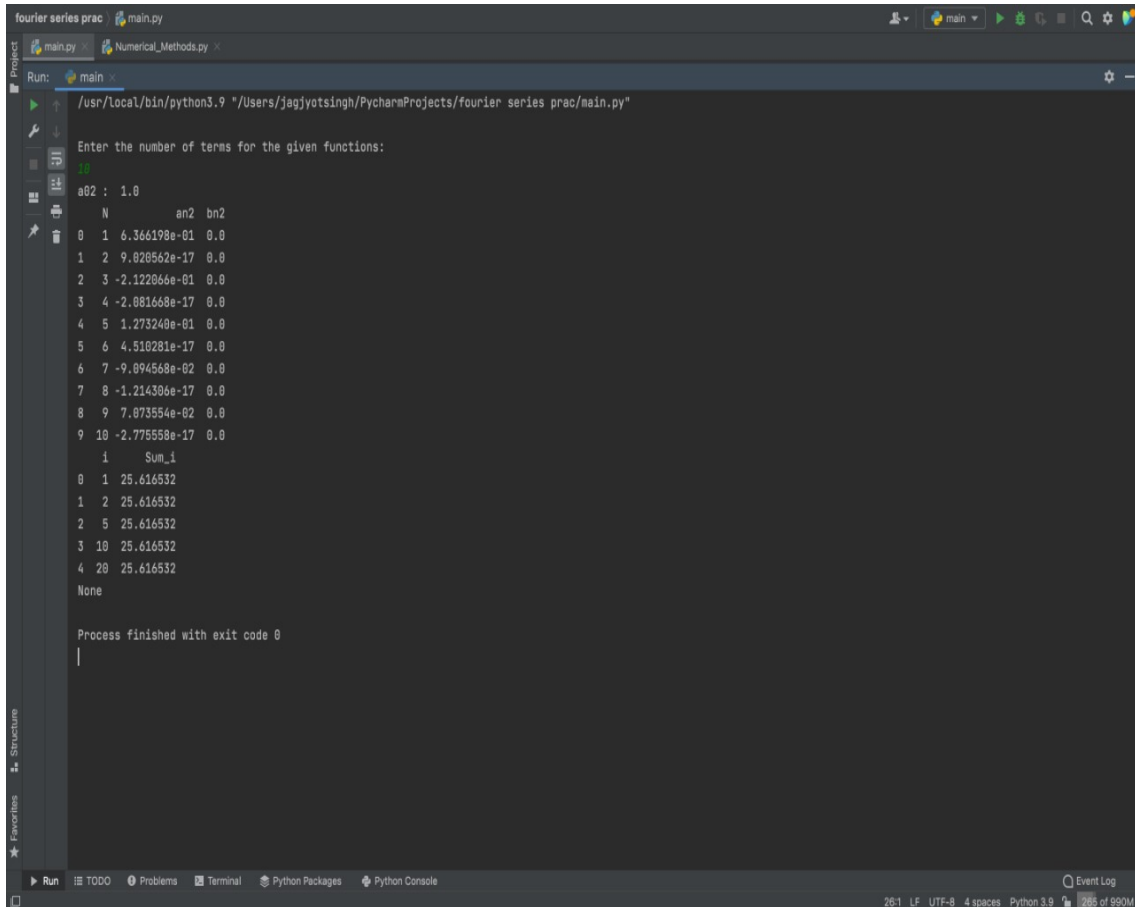








Function 2



The screenshot shows a PyCharm IDE window with a project named "fourier series prac". The main.py file is open, and the Run console is visible. The console output shows the execution of a Python script that calculates the Fourier series coefficients for a function. The user is prompted to enter the number of terms, and the output displays the coefficients for the first 10 terms.

```
Run: main
/usr/local/bin/python3.9 "/Users/jagjyotsingh/PycharmProjects/fourier series prac/main.py"

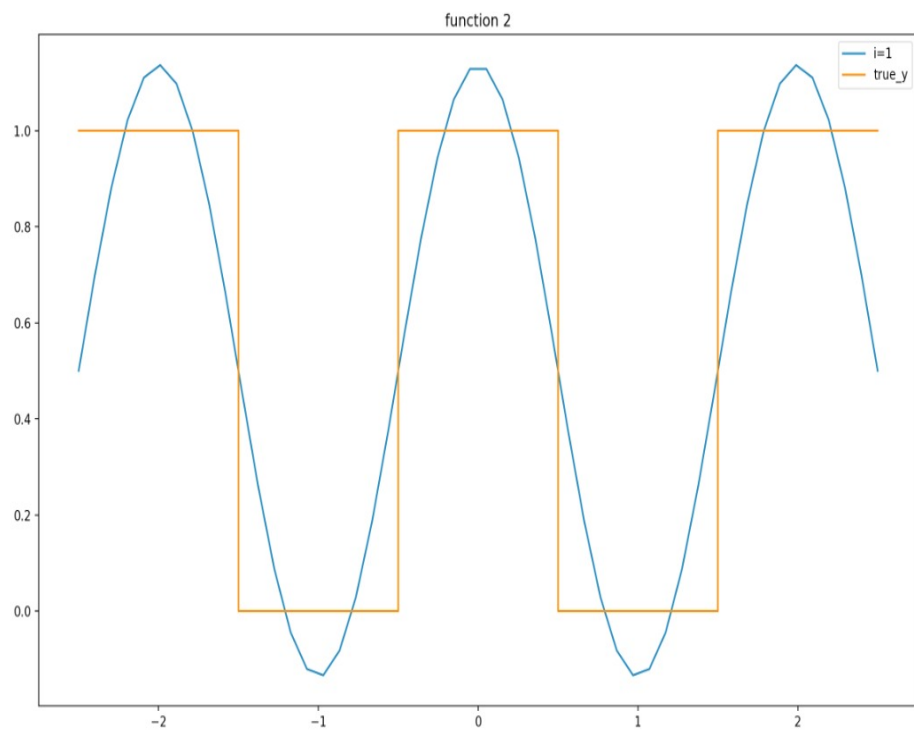
Enter the number of terms for the given functions:
10

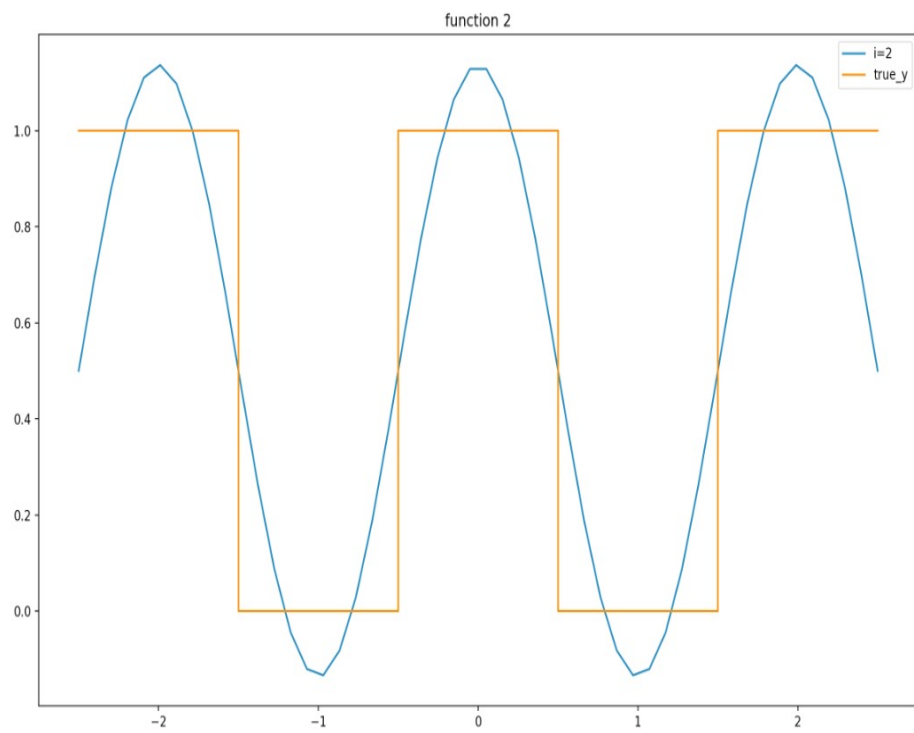
a02 : 1.0
N      an2  bn2
0 1  6.366198e-01  0.0
1 2  9.020562e-17  0.0
2 3 -2.122066e-01  0.0
3 4 -2.081668e-17  0.0
4 5  1.273240e-01  0.0
5 6  4.510281e-17  0.0
6 7 -9.094568e-02  0.0
7 8 -1.214306e-17  0.0
8 9  7.073554e-02  0.0
9 10 -2.775558e-17  0.0

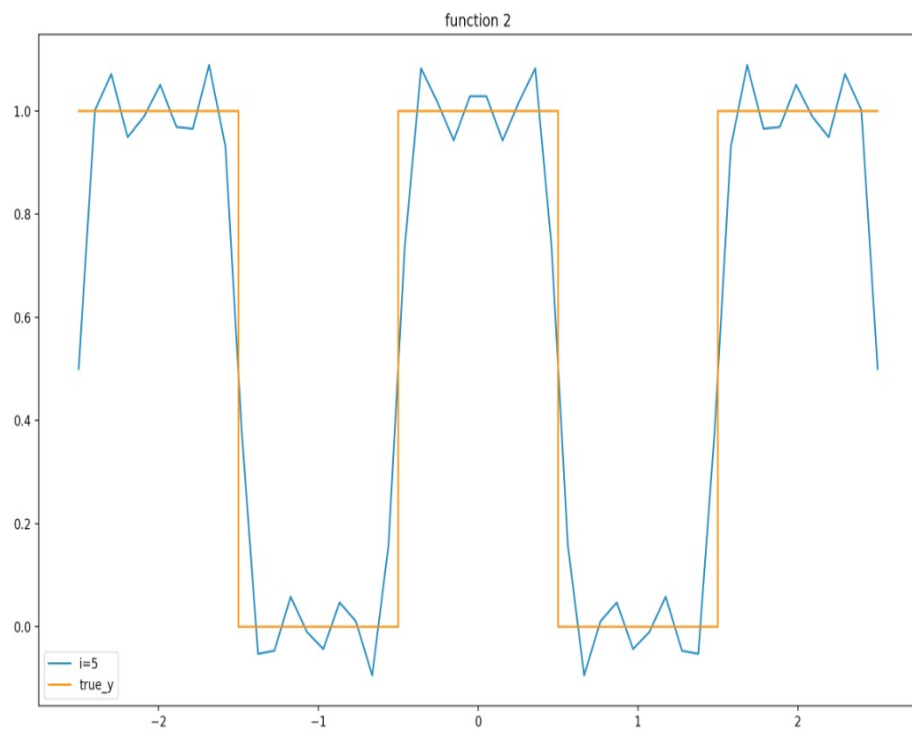
Sum_1
0 1  25.616532
1 2  25.616532
2 5  25.616532
3 10 25.616532
4 20 25.616532

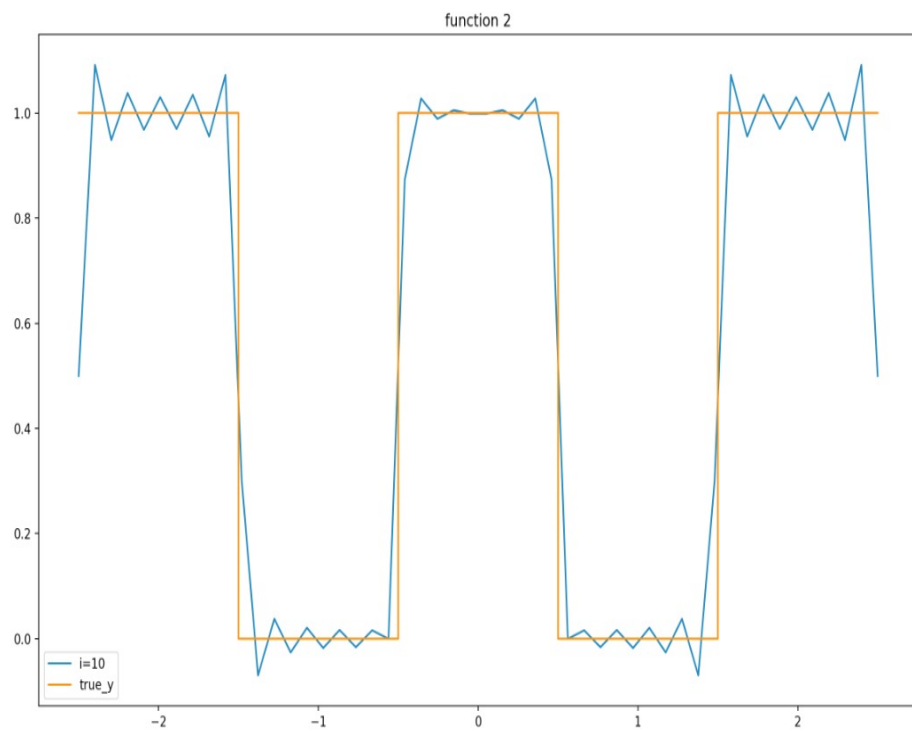
None

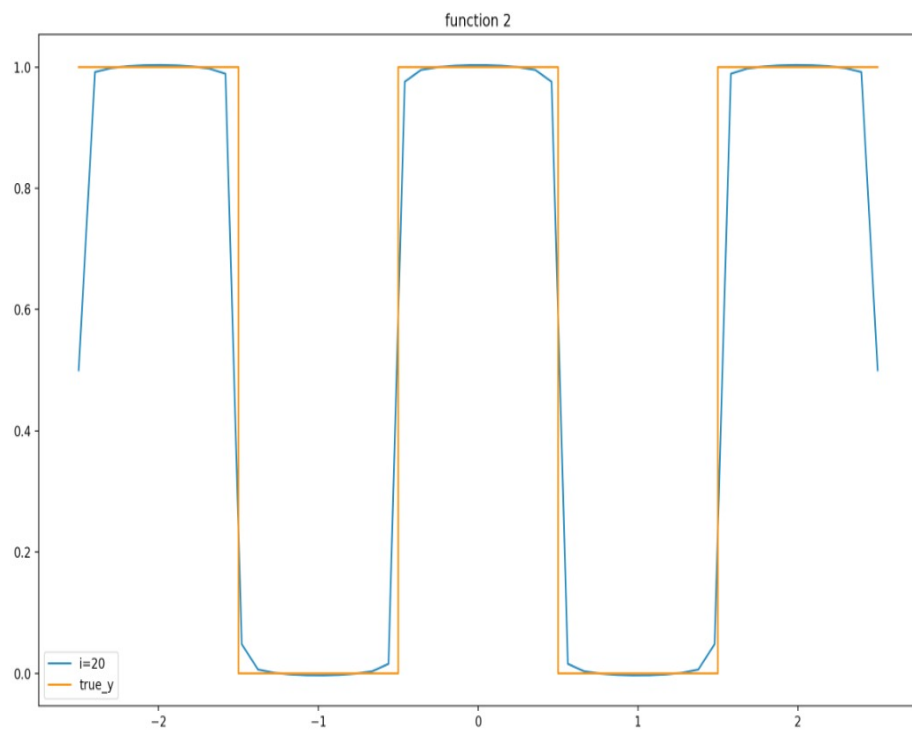
Process finished with exit code 0
```



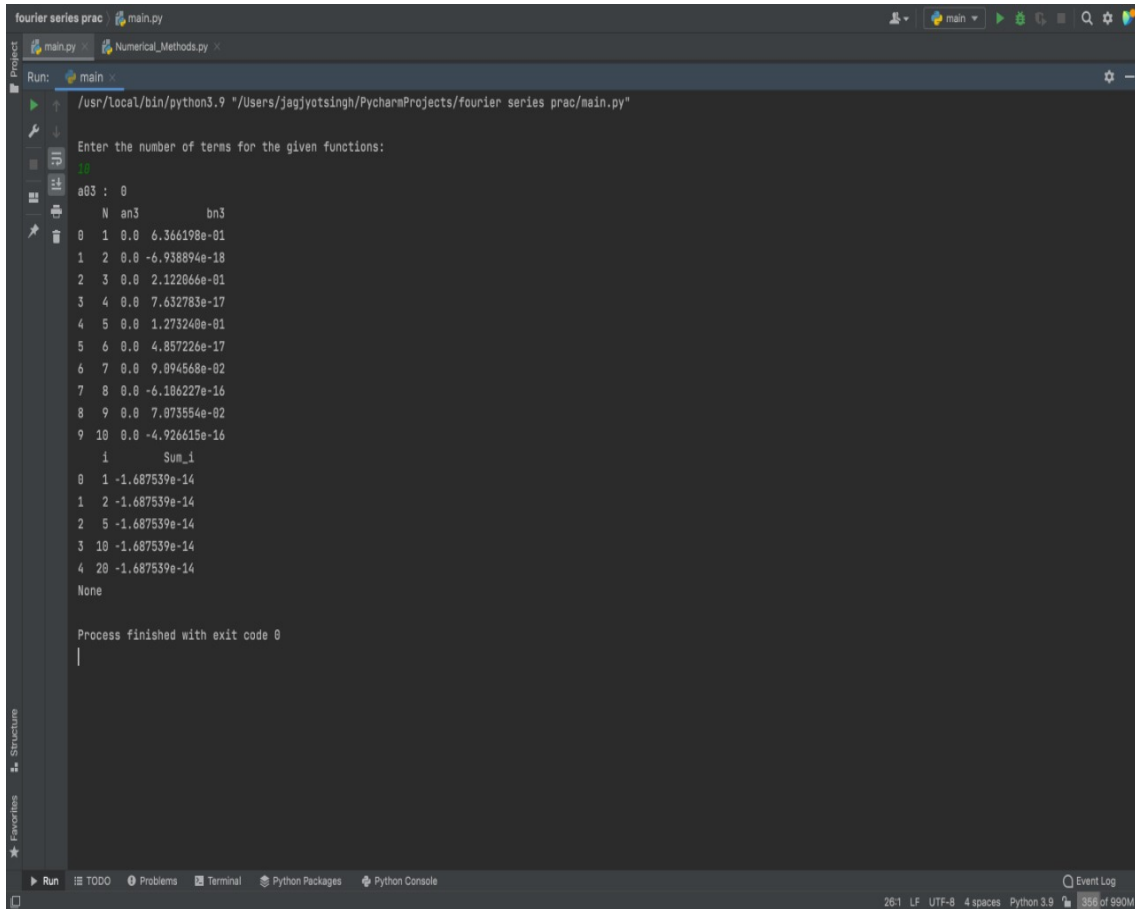




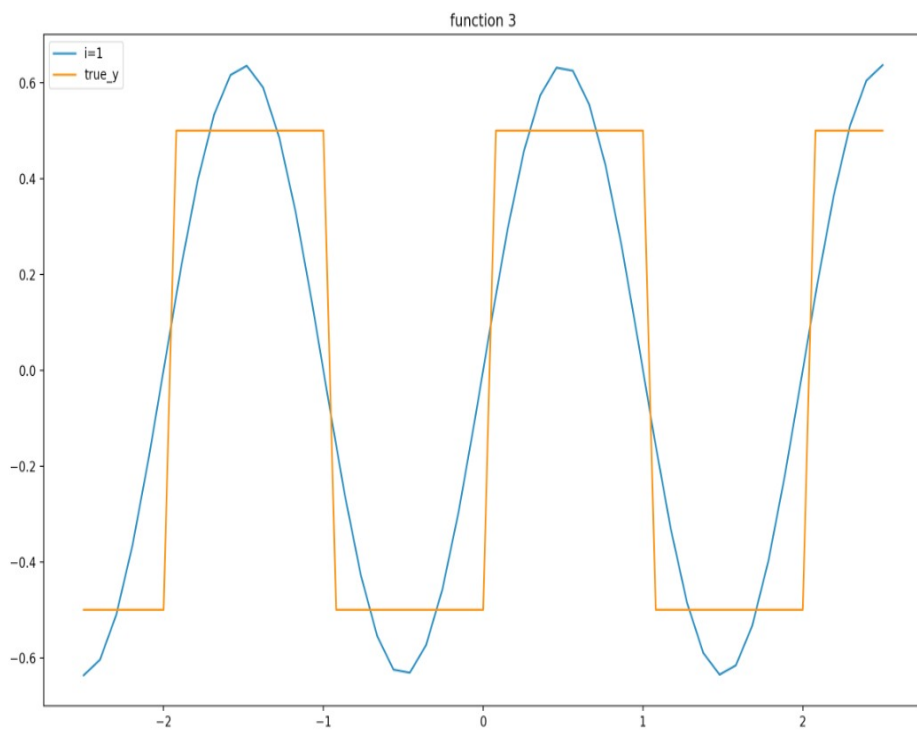


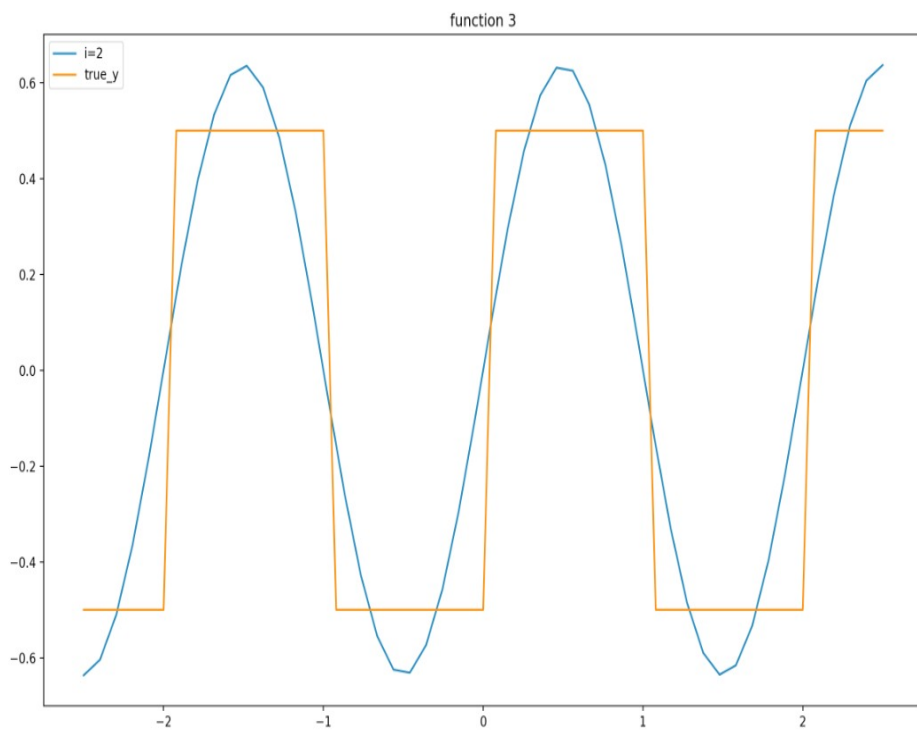


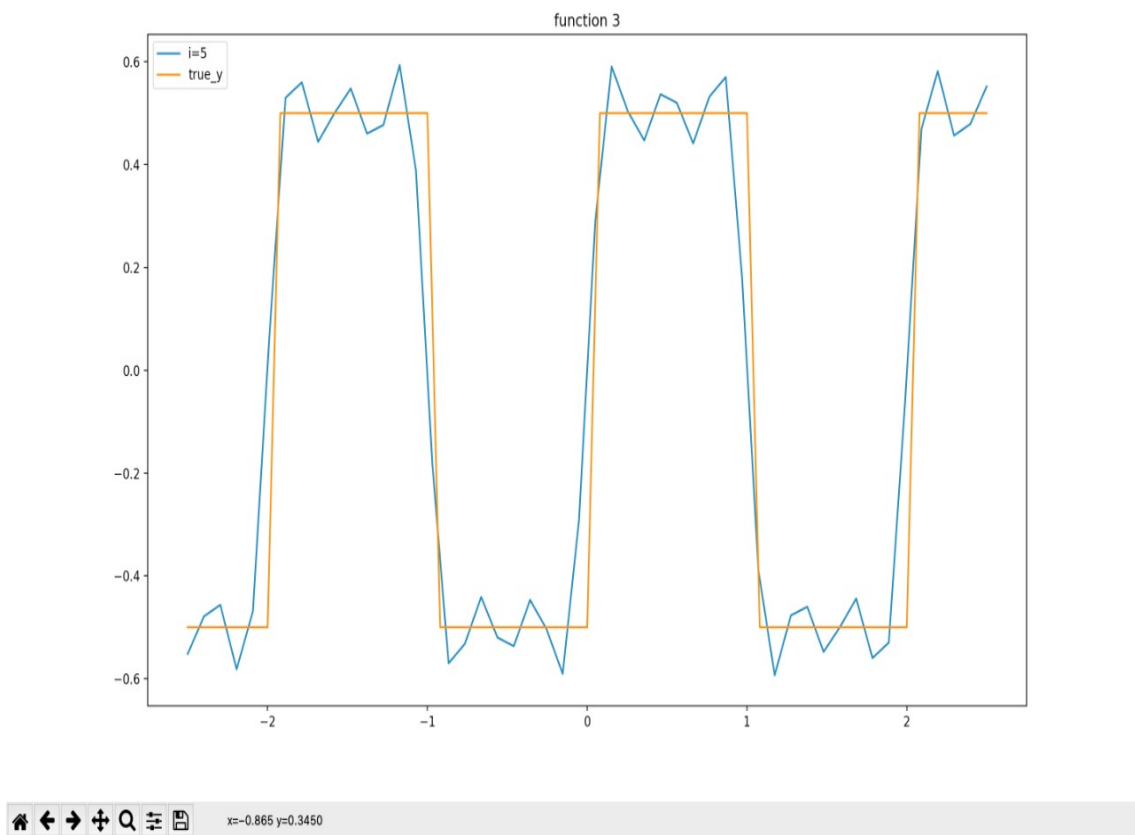
Function 3

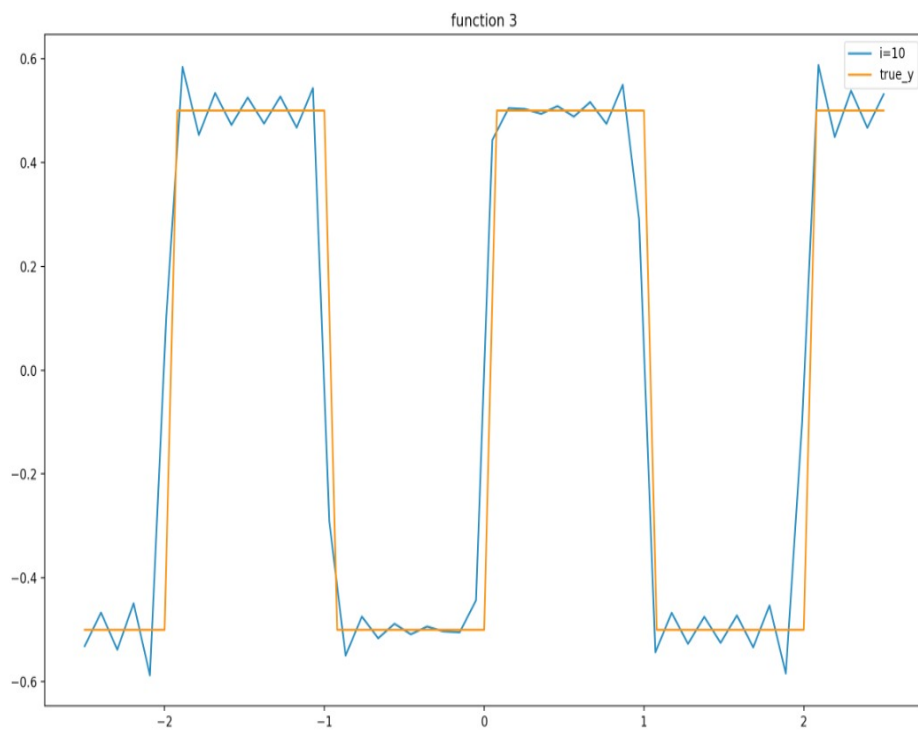


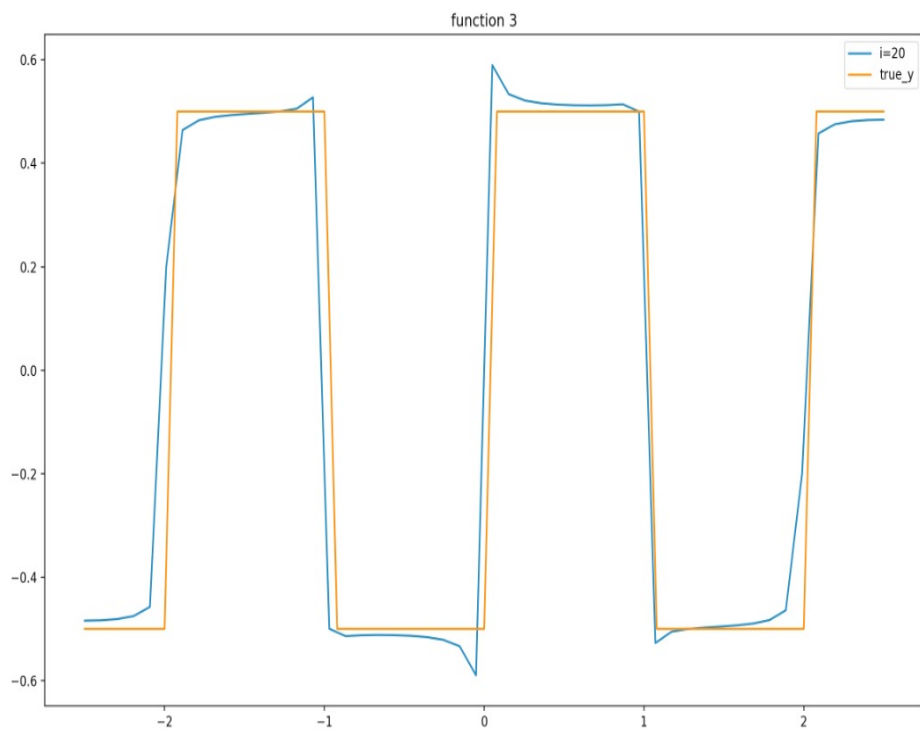
```
fourier series prac main.py
main.py Numerical_Methods.py
Run: main
/usr/local/bin/python3.9 "/Users/jagjyotsingh/PycharmProjects/fourier series prac/main.py"
Enter the number of terms for the given functions:
10
a03 : 0
N an3 bn3
0 1 0.0 6.366198e-01
1 2 0.0 -6.938894e-18
2 3 0.0 2.122066e-01
3 4 0.0 7.632783e-17
4 5 0.0 1.273240e-01
5 6 0.0 4.857226e-17
6 7 0.0 9.094568e-02
7 8 0.0 -6.106227e-16
8 9 0.0 7.073554e-02
9 10 0.0 -4.926615e-16
Sum_1
0 1 -1.687539e-14
1 2 -1.687539e-14
2 5 -1.687539e-14
3 10 -1.687539e-14
4 20 -1.687539e-14
None
Process finished with exit code 0
```











Programming

Python program

Main File

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import sympy as sym
4 from MyIntegration import MyLegQuadrature_tol_1, MyLegQuadrature, MyTrap_tol,
   MySimp_tol
5 import pandas as pd
6
7
8 x = sym.symbols('x')
9
10
11 def product(f1, f2):
12     return lambda x: f1(x) * f2(x)
13
14
15 def sumfn(f1, f2):
16     return lambda x: f1(x) + f2(x)
17
18
19 def FourierCoeff(method, d, f, dis, L, N, var):
20
21     def coef_a0():
22
23         a0 = 0
24         low = dis[0]
25
26         if method == 'quad': # if method is quad
27
28             for i in range(len(f)):
29                 # defining high index
30                 high = dis[i + 1]
31
32                 a0 += MyLegQuadrature_tol_1(f[i], low, high, n=5, d=d)
33                 low = high
34
35         elif method == 'trap':
36
37             for i in range(len(f)):
38                 # defining high index
39                 high = dis[i + 1]
40
41                 a0 += MyTrap_tol(f[i], low, high, n=20, d=d)
42                 low = high
43
44         elif method == 'simp':
45
46             for i in range(len(f)):
47                 # defining high index
48                 high = dis[i + 1]
49
50                 a0 += MySimp_tol(f[i], low, high, n=20, d=d)
51                 low = high
52
53         return a0 / L
54
55     def coef_an():
56         an = []
57         if method == 'quad':
58
59             for i in range(1, N + 1):
```

```

60         # i th coefficient of fourier series
61         ai = 0
62
63         f_for_a = lambda x: np.cos(i * np.pi * x / L)
64
65         low = dis[0]
66
67         for j in range(len(f)):
68             high = dis[j + 1]
69             ai += MyLegQuadrature_tol_1(product(f[j], f_for_a), low, high,
n=5, d=d)
70
71             low = high
72
73             an.append(ai)
74
75     elif method == 'trap':
76
77         for i in range(1, N + 1):
78             # i th coefficient of fourier series
79             ai = 0
80
81             f_for_a = lambda x: np.cos(i * np.pi * x / L)
82
83             low = dis[0]
84
85             for j in range(len(f)):
86                 high = dis[j + 1]
87                 ai += MyTrap_tol(product(f[j], f_for_a), low, high, n=20, d=d)
88
89                 low = high
90                 an.append(ai)
91
92     elif method == 'simp':
93
94         for i in range(1, N + 1):
95             # i th coefficient of fourier series
96             ai = 0
97
98             f_for_a = lambda x: np.cos(i * np.pi * x / L)
99
100            low = dis[0]
101
102            for j in range(len(f)):
103                high = dis[j + 1]
104                ai += MySimp_tol(product(f[j], f_for_a), low, high, n=20, d=d)
105
106                low = high
107                an.append(ai)
108
109        return an
110
111    def coef_bn():
112
113
114        bn = []
115        if method == 'quad':
116
117            for i in range(1, N + 1):
118                # i th coefficient of fourier series
119                bi = 0
120
121                f_for_b = lambda x: np.sin(i * np.pi * x / L)
122
123                low = dis[0]
124
125                for j in range(len(f)):

```

```

126         high = dis[j + 1]
127         bi += MyLegQuadrature_tol_1(product(f[j], f_for_b), low, high,
n=5, d=d)
128
129         low = high
130
131         bn.append(bi)
132
133     elif method == 'trap':
134
135         for i in range(1, N + 1):
136             # i th coefficient of fourier series
137             bi = 0
138
139             f_for_b = lambda x: np.sin(i * np.pi * x / L)
140
141             low = dis[0]
142
143             for j in range(len(f)):
144                 high = dis[j + 1]
145                 bi += MyTrap_tol(product(f[j], f_for_b), low, high, n=20, d=d)
146
147             low = high
148             bn.append(bi)
149
150     elif method == 'simp':
151
152         for i in range(1, N + 1):
153             # i th coefficient of fourier series
154             bi = 0
155
156             f_for_b = lambda x: np.sin(i * np.pi * x / L)
157
158             low = dis[0]
159
160             for j in range(len(f)):
161                 high = dis[j + 1]
162                 bi += MySimp_tol(product(f[j], f_for_b), low, high, n=20, d=d)
163
164             low = high
165             bn.append(bi)
166
167     return bn
168
169 if var == 0:
170     a0 = coef_a0()
171     an = coef_an()
172     bn = np.zeros(N)
173
174 elif var == 1:
175     a0 = 0
176     an = np.zeros(N)
177     bn = coef_bn()
178
179 else:
180     a0 = coef_a0()
181     an = coef_an()
182     bn = coef_bn()
183
184 return a0, an, bn
185
186
187 print('\nEnter the number of terms for the given functions:')
188 n = int(input())
189
190
191 def Q3i(n):

```

```

192 f1 = [lambda x: 0, lambda x: 1]
193 dis1 = [-1, 0, 1]
194 L = 1
195 a01, an1, bn1 = FourierCoeff("quad", 5, f1, dis1, 1, n, -1)
196 T = [t for t in range(1, n + 1)]
197
198 df = pd.DataFrame({'N': T, 'an1': an1, 'bn1': bn1})
199 print('a01 : ', a01)
200 print(df)
201 S = []
202 i = [1, 2, 5, 10, 20]
203 xi = np.linspace(-2.5, 2.5, 50)
204
205 y1 = []
206 for n in i:
207     sum_i = lambda x: a01 / 2
208
209     for j in range(len(i)):
210         f_add = lambda x: an1[j] * np.cos((j + 1) * x * np.pi / L) + bn1[j] *
np.sin((j+1) * x * np.pi / L)
211
212         sum_i = sumfn(sum_i, f_add)
213     y1.append(sum_i)
214     S.append(np.array(sum_i(xi)))
215
216 s1 = []
217 for k in range(len(S)):
218     s1.append(sum(S[k]))
219 dt = pd.DataFrame({'i': i, 'Sum_i': s1})
220 print(dt)
221
222 an_c = []
223 bn_s = []
224
225 def func(x, i):
226     a0, an, bn = FourierCoeff("quad", d=5, f=f1, dis=dis1, L=1, N=i, var=-1)
227     for j in range(0, len(an)):
228         an_c.append(an[j] * np.cos((j + 1) * np.pi * x / 1))
229         bn_s.append(bn[j] * np.sin((j + 1) * np.pi * x / 1))
230     four = (a0 / 2 + sum(an_c) + sum(bn_s))
231     return four, an_c, bn_s
232
233 #plt.plot(xi, func(xi, 1)[0], label="i=1")
234 #plt.plot(xi, func(xi, 2)[0], label="i=2")
235 #plt.plot(xi, func(xi, 5)[0], label="i=5")
236 #plt.plot(xi, func(xi, 10)[0], label="i=10")
237 plt.plot(xi, func(xi, 20)[0], label="i=20")
238
239 f = open("file1.dat", 'wb')
240 x1 = np.column_stack((an1, bn1))
241 np.savetxt(f, x1)
242 f.close()
243
244 x_true1 = [-2.5, -2, -1.92, -1, -0.92, 0, 0.08, 1, 1.08, 2, 2.08, 2.5]
245 y_true1 = [0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1]
246
247 plt.plot(x_true1, y_true1, label='true_y')
248 plt.legend()
249 plt.show()
250
251
252 #print(Q3i(n))
253
254
255 def Q3ii(n):
256     # for function (2)
257     f2 = [lambda x: 0, lambda x: 1, lambda x: 0]

```



```

258     dis2 = [-1, -0.5, 0.5, 1]
259     a02, an2, bn2 = FourierCoeff("quad", 6, f=f2, dis=dis2, L=1, N=n, var=0)
260     L = 1
261     T = [t for t in range(1, n + 1)]
262     xi = np.linspace(-2.5, 2.5, 50)
263
264     df = pd.DataFrame({'N': T, 'an2': an2, 'bn2': bn2})
265     print('a02 : ', a02)
266     print(df)
267     S = []
268     i = [1, 2, 5, 10, 20]
269
270
271     y1 = []
272     for n in i:
273         sum_i = lambda x: a02 / 2
274
275         for j in range(len(i)):
276             f_add = lambda x: an2[j] * np.cos((j + 1) * x * np.pi / L) + bn2[j] *
np.sin((j+1) * x * np.pi / L)
277
278             sum_i = sumfn(sum_i, f_add)
279             y1.append(sum_i)
280             S.append(np.array(sum_i(xi)))
281
282     s1 = []
283     for k in range(len(S)):
284         s1.append(sum(S[k]))
285     dt = pd.DataFrame({'i': i, 'Sum_i': s1})
286     print(dt)
287
288     an_c = []
289     bn_s = []
290
291     def func(x, i):
292         a0, an, bn = FourierCoeff("quad", d=5, f=f2, dis=dis2, L=1, N=i, var=-1)
293         for j in range(0, len(an)):
294             an_c.append(an[j] * np.cos((j + 1) * np.pi * xi / 1))
295             bn_s.append(bn[j] * np.sin((j + 1) * np.pi * xi / 1))
296         four = (a0 / 2 + sum(an_c) + sum(bn_s))
297         return four, an_c, bn_s
298
299     plt.plot(xi, func(xi, 1)[0], label="i=1")
300     #plt.plot(xi, func(xi, 2)[0], label="i=2")
301     #plt.plot(xi, func(xi, 5)[0], label="i=5")
302     #plt.plot(xi, func(xi, 10)[0], label="i=10")
303     #plt.plot(xi, func(xi, 100)[0], label="i=20")
304     #print(FourierCoeff("quad", d=5, f=f2, dis=dis2, L=1, N=10, var=-1))
305
306     x_true1 = [-2.499999, -2.0000001, -1.500001, -1.4999999, -1.000001, -0.999999,
-0.5000001, -0.49999999, 0.4999999, 0.5000001, 0.999999, 1.000001, 1.499999,
1.5000001, 2.0000001, 2.4999999]
307     y_true1 = [1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1]
308     f = open("file2.dat", 'wb')
309     x1 = np.column_stack((an2, bn2))
310     np.savetxt(f, x1)
311     f.close()
312
313     plt.plot(x_true1, y_true1, label='true_y')
314     plt.legend()
315     plt.show()
316     #return a02, an2, bn2
317
318
319     #print(Q3ii(n))
320
321

```

```

322 def Q3iii(n):
323     # for function(3)
324     f3 = [lambda x: -0.5, lambda x: 0.5]
325     dis3 = [-1, 0, 1]
326     a03, an3, bn3 = FourierCoeff("quad", 6, f=f3, dis=dis3, L=1, N=n, var=1)
327     L = 1
328     T = [t for t in range(1, n + 1)]
329
330     df = pd.DataFrame({'N': T, 'an3': an3, 'bn3': bn3})
331     print('a03 : ', a03)
332     print(df)
333     S = []
334     i = [1, 2, 5, 10, 20]
335     xi = np.linspace(-2.5, 2.5, 50)
336
337     y1 = []
338     for n in i:
339         sum_i = lambda x: a03 / 2
340
341         for j in range(len(i)):
342             f_add = lambda x: an3[j] * np.cos((j + 1) * x * np.pi / L) + bn3[j] *
np.sin((j + 1) * x * np.pi / L)
343
344             sum_i = sumfn(sum_i, f_add)
345             y1.append(sum_i)
346             S.append(np.array(sum_i(xi)))
347
348     s1 = []
349     for k in range(len(S)):
350         s1.append(sum(S[k]))
351     dt = pd.DataFrame({'i': i, 'Sum_i': s1})
352     print(dt)
353
354     an_c = []
355     bn_s = []
356
357     def func(x, i):
358         a0, an, bn = FourierCoeff("quad", d=5, f=f3, dis=dis3, L=1, N=i, var=-1)
359         for j in range(0, len(an)):
360             an_c.append(an[j] * np.cos((j + 1) * np.pi * xi / 1))
361             bn_s.append(bn[j] * np.sin((j + 1) * np.pi * xi / 1))
362         four = (a0 / 2 + sum(an_c) + sum(bn_s))
363         return four, an_c, bn_s
364
365     #plt.plot(xi, func(xi, 1)[0], label="i=1")
366     #plt.plot(xi, func(xi, 2)[0], label="i=2")
367     #plt.plot(xi, func(xi, 5)[0], label="i=5")
368     #plt.plot(xi, func(xi, 10)[0], label="i=10")
369     #plt.plot(xi, func(xi, 20)[0], label="i=20")
370
371     x_true1 = [-2.5, -2, -1.92, -1, -0.92, 0, 0.08, 1, 1.08, 2, 2.08, 2.5]
372     y_true1 = [-0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5, -0.5, -0.5, 0.5, 0.5]
373
374     f = open("file3.dat", 'wb')
375     x1 = np.column_stack((an3, bn3))
376     np.savetxt(f, x1)
377     f.close()
378
379     plt.plot(x_true1, y_true1, label='true_y')
380     plt.legend()
381     plt.show()
382     #return a03, an3, bn3
383
384
385 print(Q3iii(n))

```

My Integration Module

```
1 from sympy import *
2 from scipy.special.orthogonal import p_roots
3 from scipy import integrate
4 import numpy as np
5 import pandas as pd
6 import matplotlib.pyplot as plt
7 import texttable as tt
8 tab = tt.Texttable()
9
10 x = symbols('x')
11 #f = eval("lambda x:" + input("function to be integrated, f(x) = "))
12 #d = int(input("Enter the number of significant figures upto which results should
    be accurate = "))
13
14
15 def My_Trap(f_, a, b, n):
16     y = []
17     h = (b - a) / n
18     for i in range(n + 1):
19         y.append(f_(a + i * h)) # y at limit points
20     trp = h * (f_(a) + f_(b)) / 2
21     for j in range(1, len(y) - 1):
22         trp = trp + h * (y[j])
23     return (trp)
24
25
26 def My_Simp(f_, a, b, n):
27     h = (b - a) / (2 * n)
28     simp = h * (f_(a) + f_(b)) / 3
29     for i in range(1, 2 * n):
30         if (i % 2 == 0):
31             simp = simp + 2 * h * f_(a + i * h) / 3
32         elif (i % 2 == 1):
33             simp = simp + 4 * h * f_(a + i * h) / 3
34
35     return (simp)
36
37
38 def MyTrap_tol(f_, a, b, n, d):
39     i = 1
40     err = []
41     while True:
42         e = abs(My_Trap(f_, a, b, i) - My_Trap(f_, a, b, i + 1))
43         err.append(e)
44         if e > 0.5 * 10 ** -d:
45             i = i + 1
46         elif e <= 0.5 * 10 ** -d:
47             p = My_Trap(f_, a, b, i)
48             break
49     return p
50
51
52 def MySimp_tol(f_, a, b, n, d):
53     i = 1
54     err = []
55     while True:
56         e = abs(My_Simp(f_, a, b, i) - My_Simp(f_, a, b, i + 1))
57         err.append(e)
58         if e > 0.5 * 10 ** -d:
59             i = i + 1
60         elif e <= 0.5 * 10 ** -d:
61             p = My_Simp(f_, a, b, i)
62             break
63     return p
64
```

```

65
66 def MyLegQuadrature(f, a, b, n, m): # Gauss legendre Quadrature ftion
67     h = (b - a) / m
68     [leg_zer, w] = p_roots(n)
69     leg_zer.tolist()
70     w.tolist()
71     sum_ = 0
72     x_ = [a]
73     s = []
74     for k in range(0, n):
75         for i in range(1, m + 1):
76             x_.append(a + h * i)
77             sum_ += (h / 2) * w[k] * f(0.5 * h * leg_zer[k] + 0.5 * (x_[i] + x_[i -
78                 1]))
79             s.append(sum_)
80     return sum_
81
82 def MyLegQuadrature_tol(fs, a, b, n, m, d):
83     i = 1
84     err = []
85     while i <= m:
86         e = abs(MyLegQuadrature(fs, a, b, n, i) - MyLegQuadrature(fs, a, b, n, i +
87             1)) / MyLegQuadrature(fs, a, b, n,
88                 i + 1)
89         err.append(e)
90         if e > 0.5 * 10 ** -d:
91             i = i * 2
92             if i > m:
93                 print("Tolerance can't be reached for ", m, "Subintervals")
94             elif e <= 0.5 * 10 ** -d:
95                 print("tolerance is reached in", i, "subintervals")
96                 print("integration and error using n point method(composite) = ",
97                     MyLegQuadrature(fs, a, b, n, i))
98                 # print("integration using inbuilt ftion = ", integrate.quadrature(f, a
99                     , b))
100                 break
101                 return MyLegQuadrature(fs, a, b, n, i)
102
103 def MyLegQuadrature_tol_1(fs, a, b, n, d):
104     i = 1
105     p = 0
106     err = []
107     while True:
108         e = abs(MyLegQuadrature(fs, a, b, n, i) - MyLegQuadrature(fs, a, b, n, i +
109             1))
110         err.append(e)
111         if e > 0.5 * 10 ** -d:
112             i = i * 2
113             elif e <= 0.5 * 10 ** -d:
114                 p = MyLegQuadrature(fs, a, b, n, i)
115                 break
116     return p

```