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

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

where,

$$f\left(x^+\right) = \lim_{y \to x^+} f(y)$$
$$f\left(x^-\right) = \lim_{y \to x^-} f(y)$$

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 $f(t+2L) = f(t)$.

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

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

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

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

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

$$= \frac{a_0}{2}\cdot 0 + \sum_{n=1}^{\infty} a_n \frac{T}{2}\delta_{mn} + \sum_{n=1}^{\infty} b_n 0 = a_m \frac{T}{2}$$

$$a_m = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\cos m\omega_0 t dt$$

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

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

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

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

$$= \frac{a_0}{2}\cdot 0 + + \sum_{n=1}^{\infty} a_n 0 + \sum_{n=1}^{\infty} b_n \frac{T}{2}\delta_{mn} = b_m \frac{T}{2}$$

$$b_m = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\sin m\omega_0 t dt$$

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

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

$$= \frac{a_0}{2}\cdot T + \sum_{n=1}^{\infty} a_n 0 + \sum_{n=1}^{\infty} b_n \cdot 0 = a_0 \frac{T}{2}$$

$$a_0 = \frac{2}{T}\int_{-T/2}^{T/2} f(t)dt$$

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 $\begin{aligned} 2L &= 2 \\ L &= 1 \end{aligned}$

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

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

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

$$= \frac{1}{1}\int_{-1}^{1} f(x)\cos\frac{n\pi x}{1}dx$$

$$= \int_{0}^{1} \cos n\pi x dx$$

$$= \left.\frac{\sin n\pi x}{n\pi}\right|_{0}^{1} = 0$$

$$\text{so, } a_n = 0 \text{ if } n \neq 0$$

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

$$= \int_{-1}^{0} -0.5\sin n\pi x dx + \int_{0}^{1} 0.5\sin n\pi x dx$$

$$= \frac{-1}{2}\left[\frac{-\cos n\pi x}{n\pi}\right]_{-1}^{0} + \frac{1}{2}\left[-\frac{\cos n\pi x}{n\pi}\right]_{0}^{1}$$

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

$$= \frac{1}{2n\pi} - \frac{\cos n\pi}{2n\pi} - \frac{\cos n\pi}{2n\pi} + \frac{1}{2n\pi}$$

$$= \frac{1}{n\pi}(1 - \cos n\pi)$$

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

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,

$$f(x) = 0 + \sum_{n=1}^{\infty} 0 \times \cos n\pi x + \sum_{n=1,odd}^{\infty} \frac{2}{n\pi}\sin \pi x n$$

$$= \frac{2}{\pi}\left[\sin \pi x + \frac{1}{3}\sin 3\pi x + \cdots\right]$$

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

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

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

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

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

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

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

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

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

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,

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

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

  and $f(x+2) = f(x)$

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

$$a_0 = \frac{1}{L}\int_{-1}^{1} f(x)dx = \int_{-1}^{-1} -0\cdot 5dx + \int_{0}^{1} 0\cdot 5dx = -0.5 + 0.5 = 0$$

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

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

$$= 0.5\left[\frac{-\cos n\pi x}{n\pi}\right]_{-1}^{0} + 0.5\left[\frac{-\cos n\pi x}{n\pi}\right]_{1}^{0} = -\frac{0.5}{n\pi}\left[[1-\cos\pi] + [1-\cos n\pi]\right] = -\frac{0.5}{n\pi}[2-2\cos\pi] = -\frac{1}{n\pi}[1-1\cos\pi]$$

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

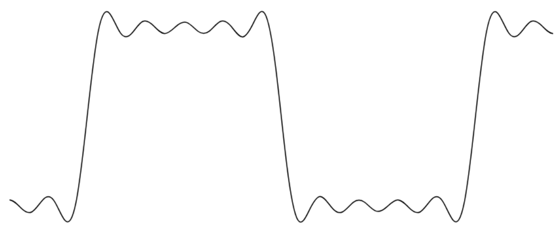
Here $a_n = 0$, and $a_0 = 0$, b_n gives some value only when n is odd, so The given function is neither odd nor even.

The fourier series is thus,

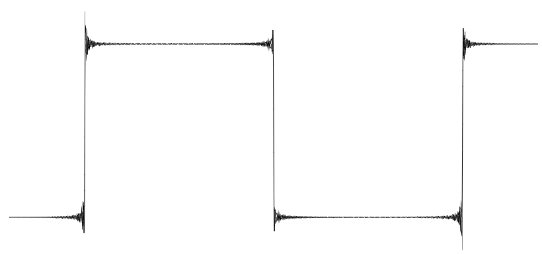
$$\begin{aligned} f(x) &= \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n\cos\frac{n\pi x}{L} + b_n\sin\frac{n\pi x}{L}\right) = \sum_{n=1,odd}^{\infty}\frac{-2}{n\pi} \\ &= \frac{-2}{\pi}\left(1 + \frac{1}{3} + \frac{1}{5} + ...\right) \end{aligned}$$

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 $f(x) = x, \quad 0 < x < \pi$.
- 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 $f(t) = t, \quad 0 < t < \pi$.

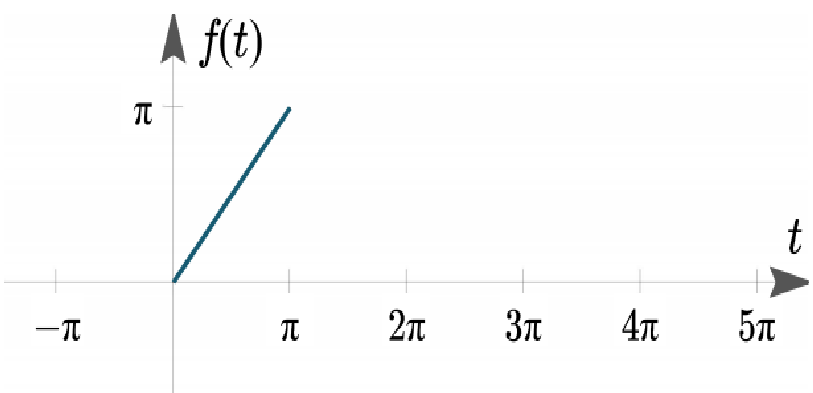
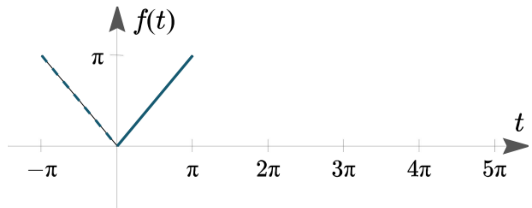
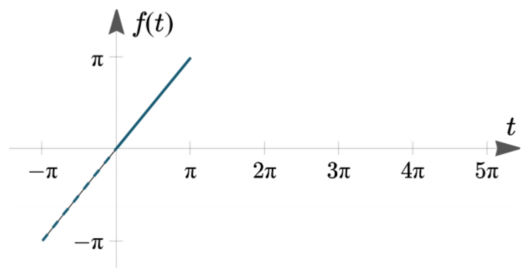


Figure 2: $f(t) = t, \quad 0 < t < \pi$.



(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

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

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

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

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

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

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

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

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

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

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

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 $a_{\text{o}} = 0$ and $a_n = 0$, we have

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

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

and $b_n = 0$

Algorithm

Algorithm 1 $FourierCoeff$

function INPUT(method, $tolerance\ d$, $function\ f\ discontinuities\ dis$, $halfrange\ L$, $Number\ of\ coefficients\ N$, $even$, odd or $neither\ (var)$)

▷ Here f is function to be integrated

For calculation of a0

for i in range(length of f) **do** ▷ f is defined piecewise, integrated piecewise)

$upperlimit = discontinuity$ ▷ discontinuity approached from left side in range of function to right side

a0 += method() ▷ to integrate the function between 'dis'

lowerlimit = upperlimit ▷ to move towards right side in range

end for

For calculation of an and bn

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

for i in range(length of f) **do** ▷ f is defined piecewise, integrated piecewise)

$upperlimit = discontinuity$ ▷ discontinuity approached from left side in range of function to right side

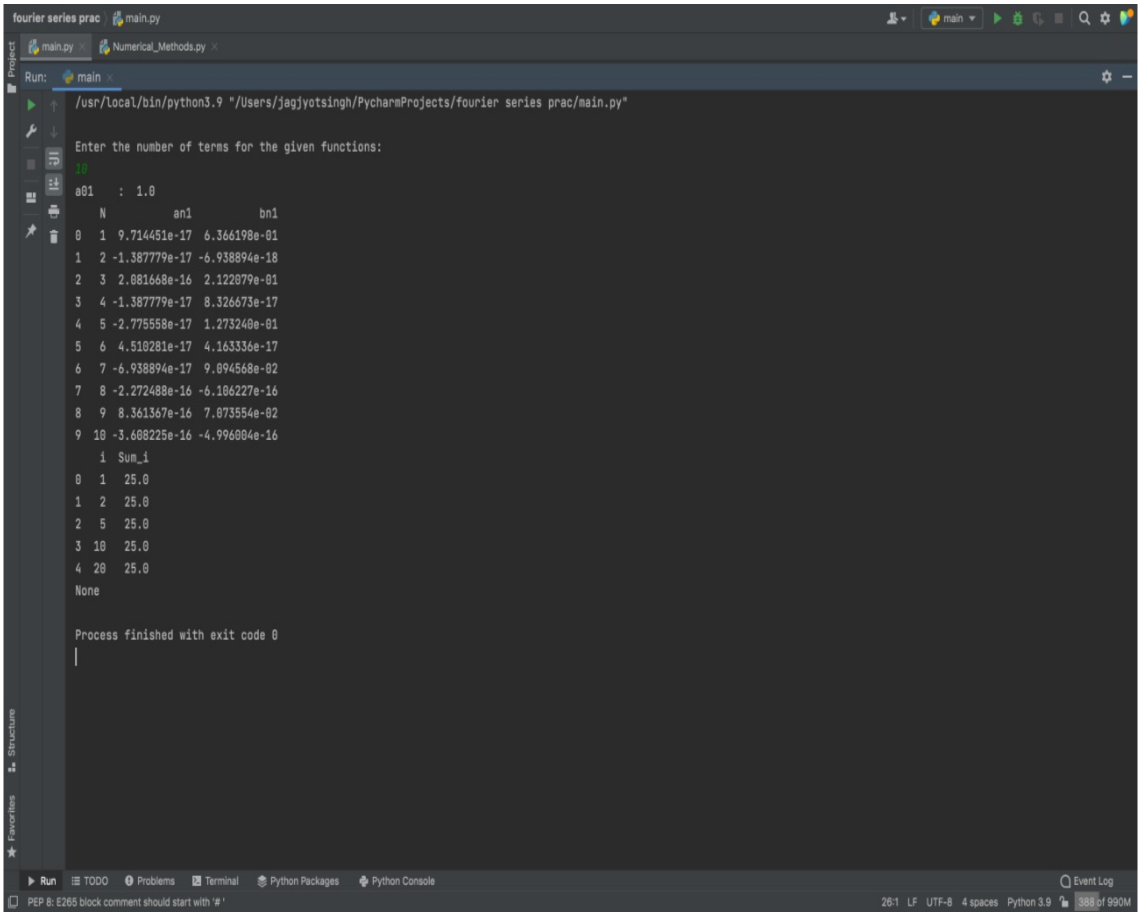
an or bn += method(product) ▷ to integrate the function between 'dis'

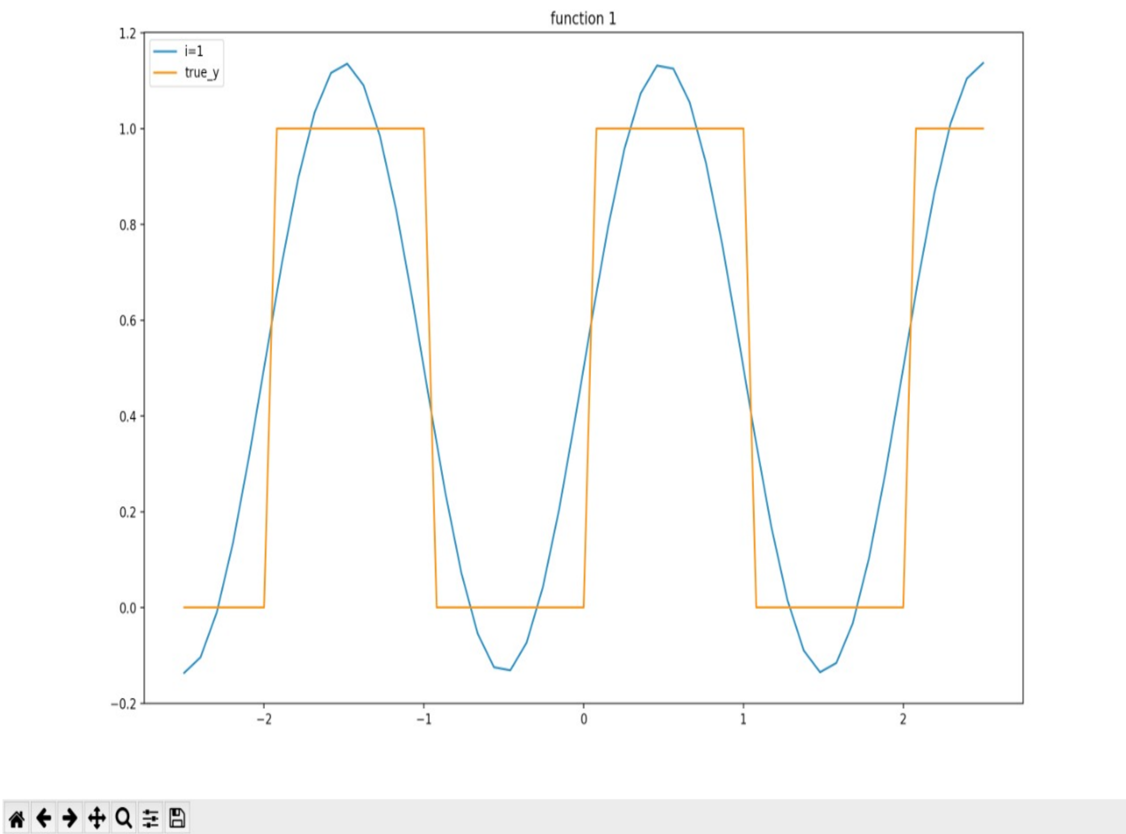
lowerlimit = upperlimit ▷ to move towards right side in range

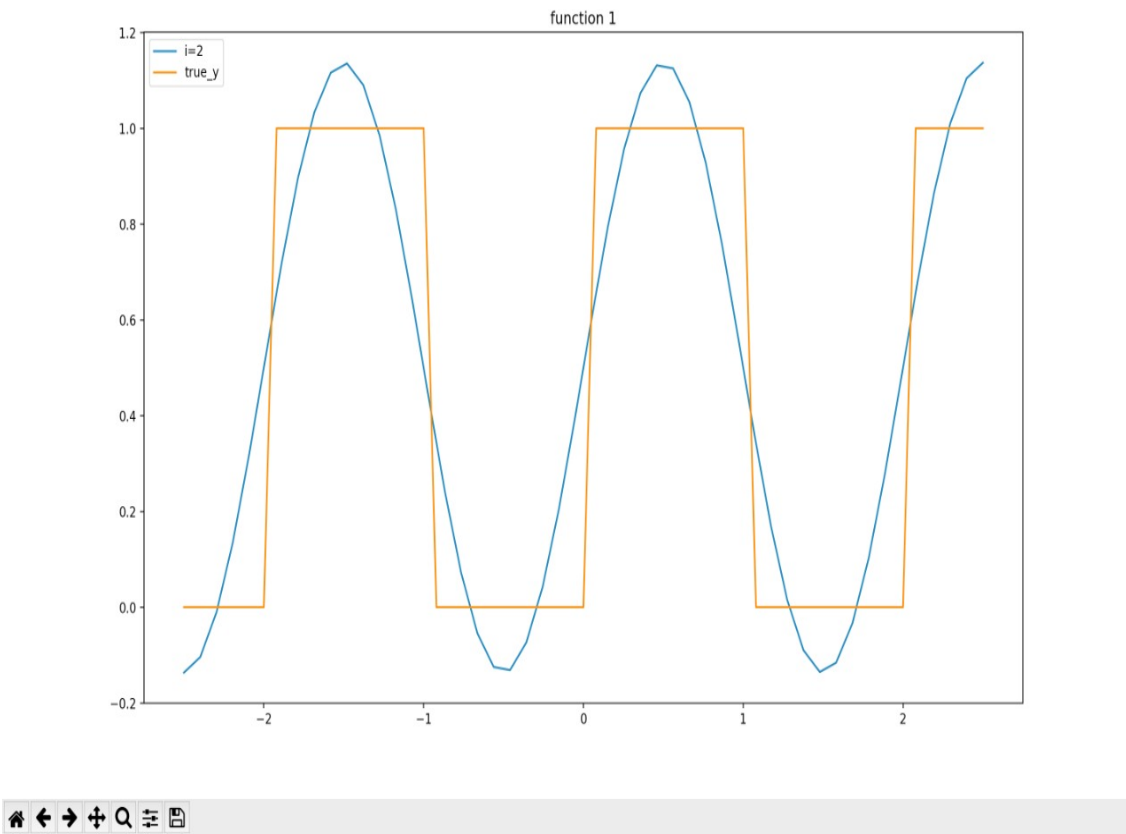
end for

return $a0, an, bn$

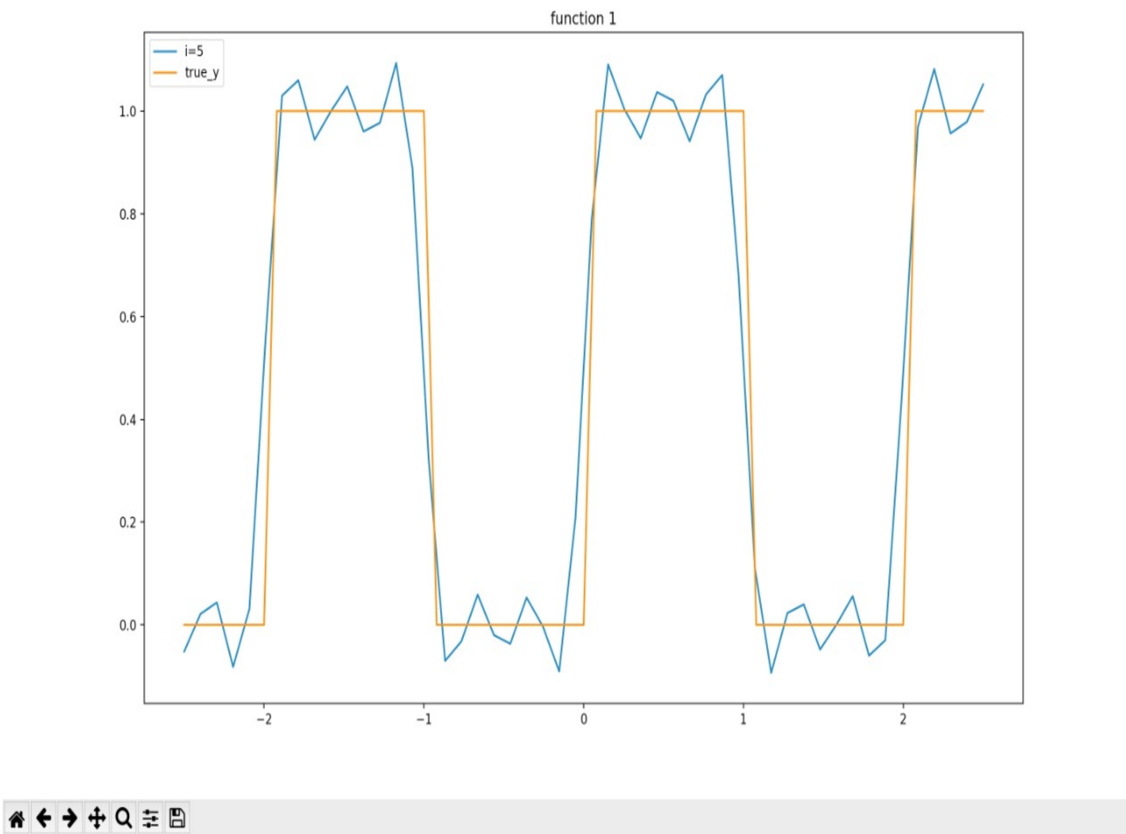
Function1

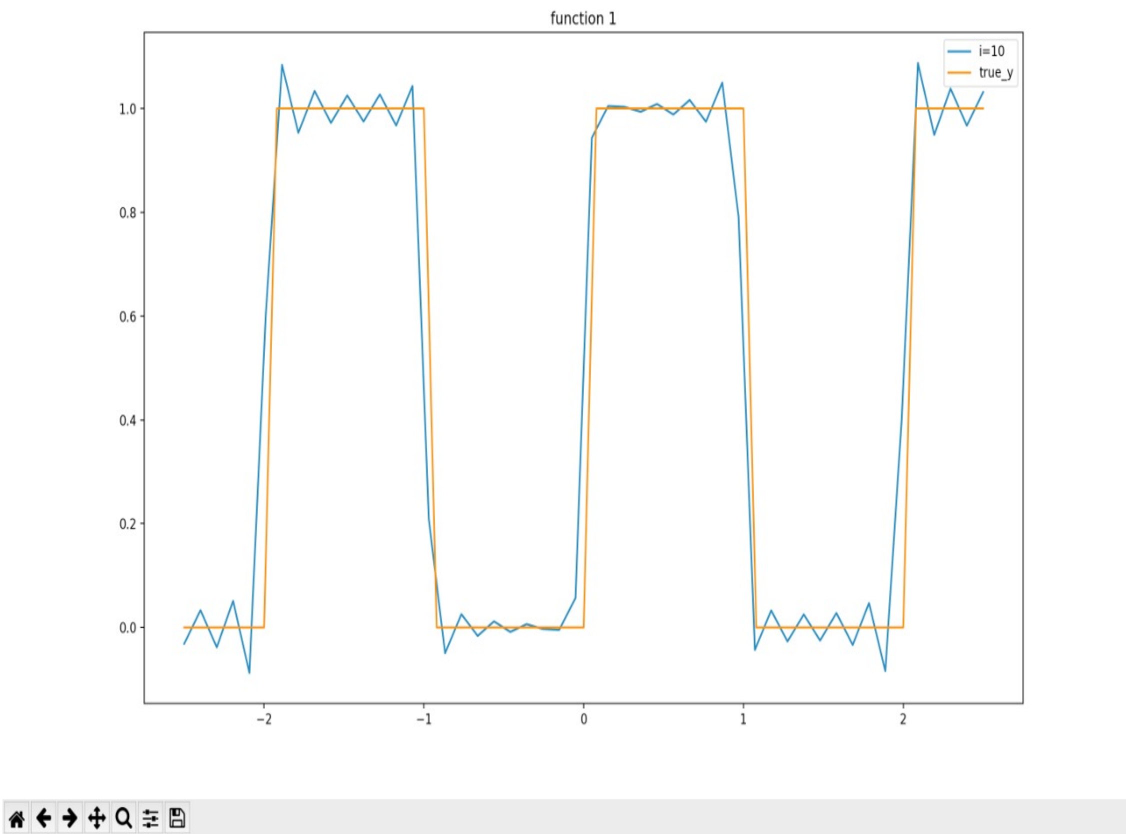

fourier series prac
main.py
main.py
Numerical_Methods.py
Run:
main
/usr/local/bin/python3.9 "/Users/jagjyotsingh/PycharmProjects/fourier series prac/main.py"
Enter the number of terms for the given functions:
10
a01   : 1.0
    N   an1   bn1
0   1 9.714451e-17 6.366198e-01
1   2 -1.387779e-17 -6.938894e-18
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 -3.608225e-16 -4.996004e-16
    i Sum_i
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
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
PEP 8: E265 block comment should start with '# '
UTF-8
4 spaces
Python 3.9


function 1
i=1
true_y
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−2
−1
0
1
2

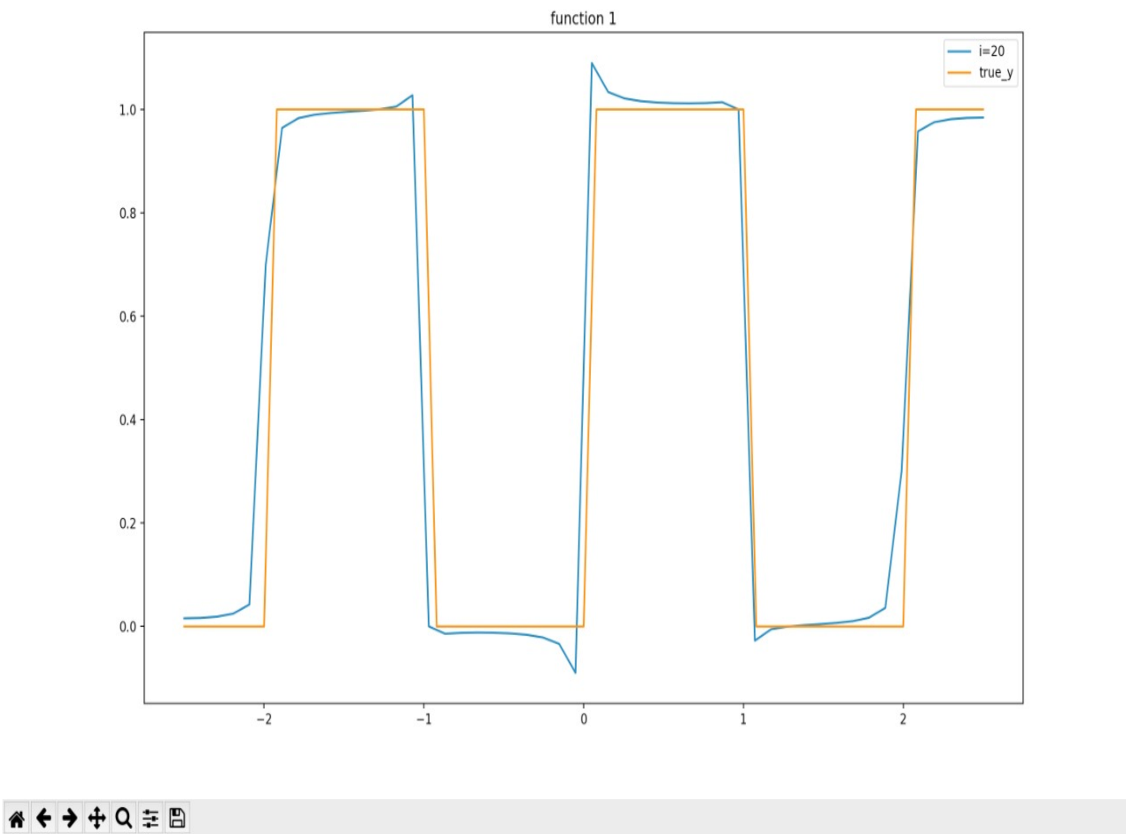


function 1
i=2
true_y
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−2
−1
0
1
2

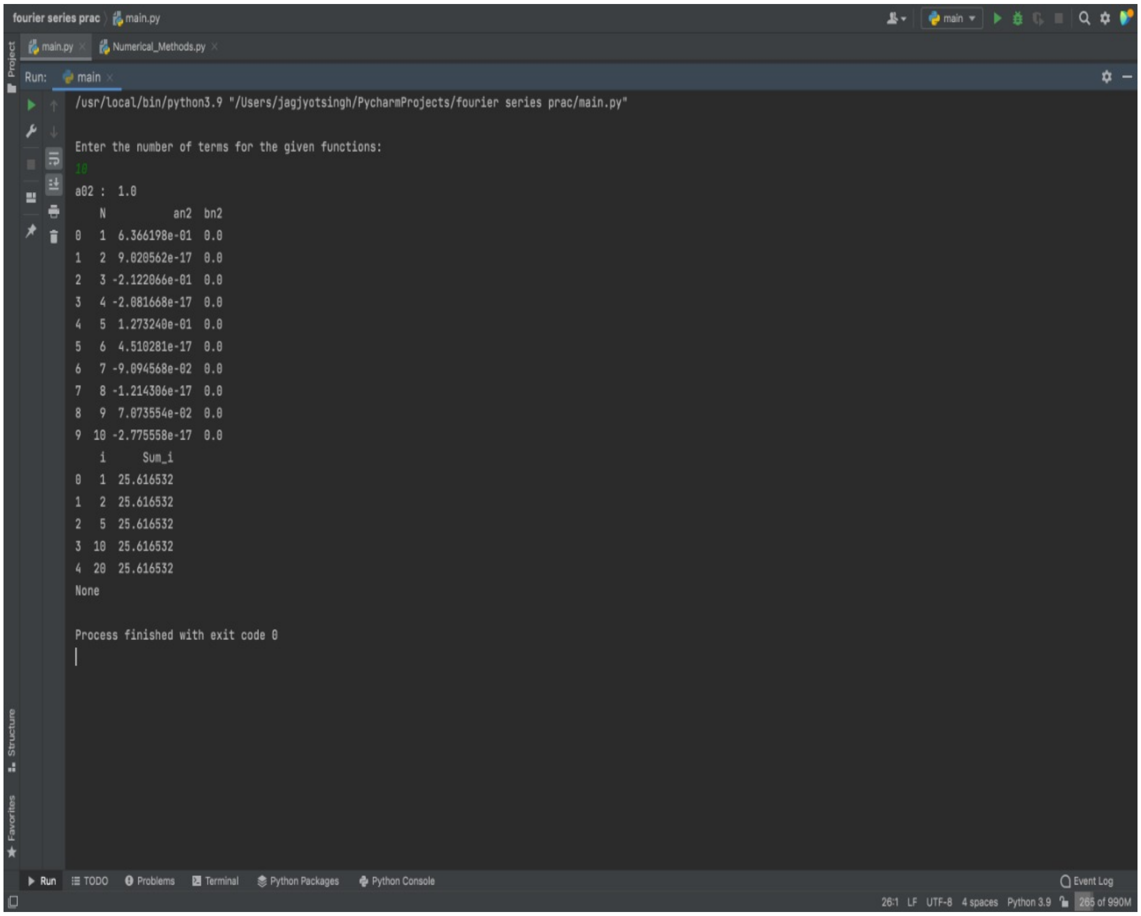

function 1
i=5
true_y
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2

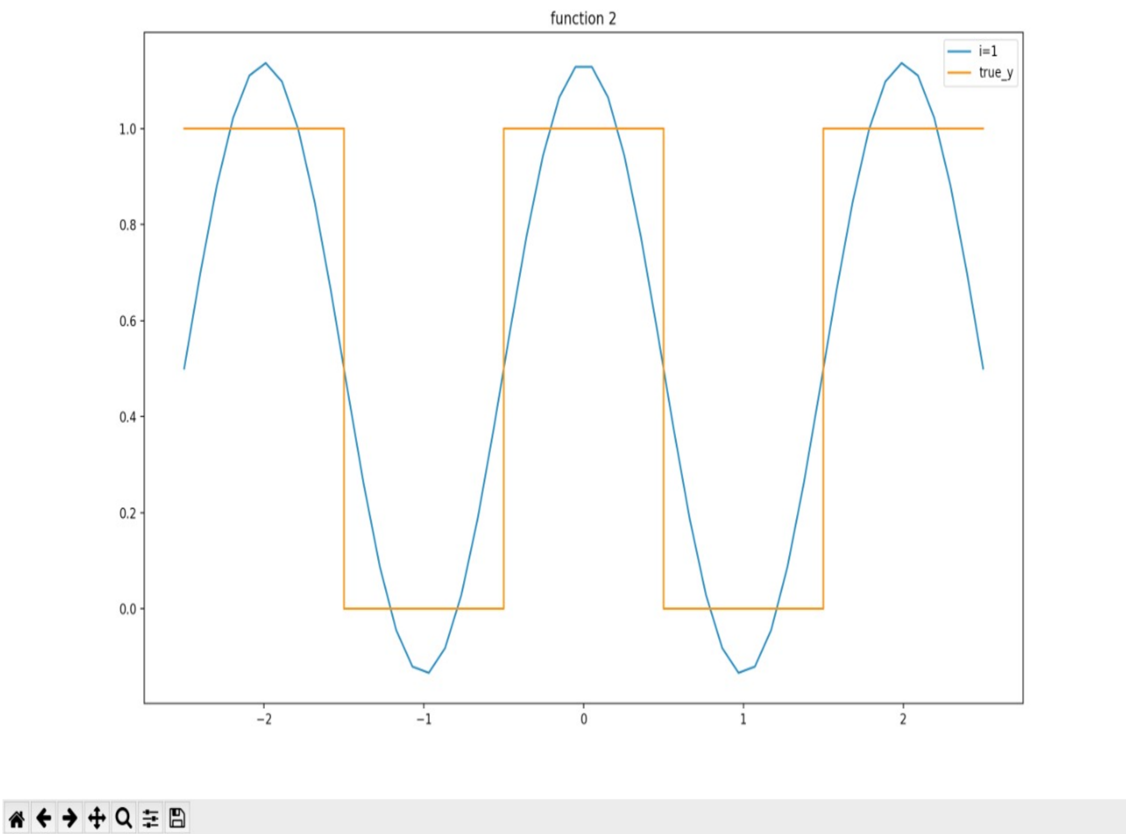


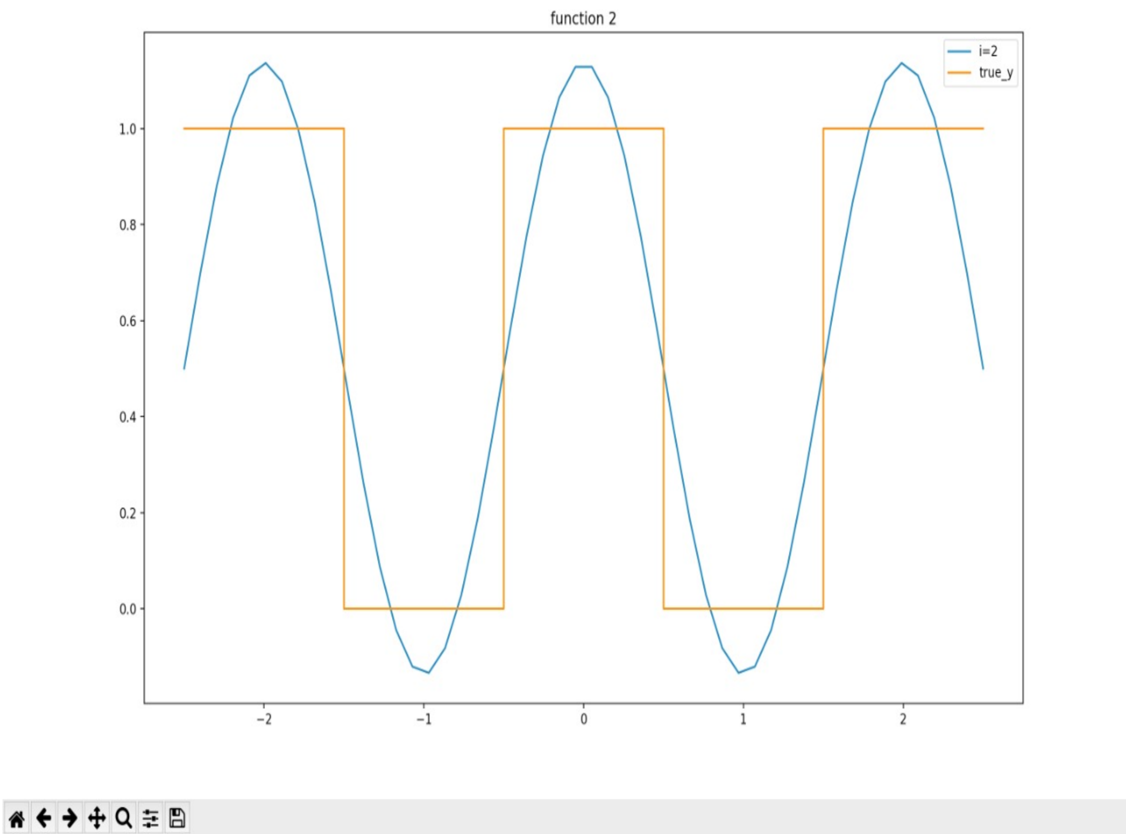
function 1
i=10
true_y
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2

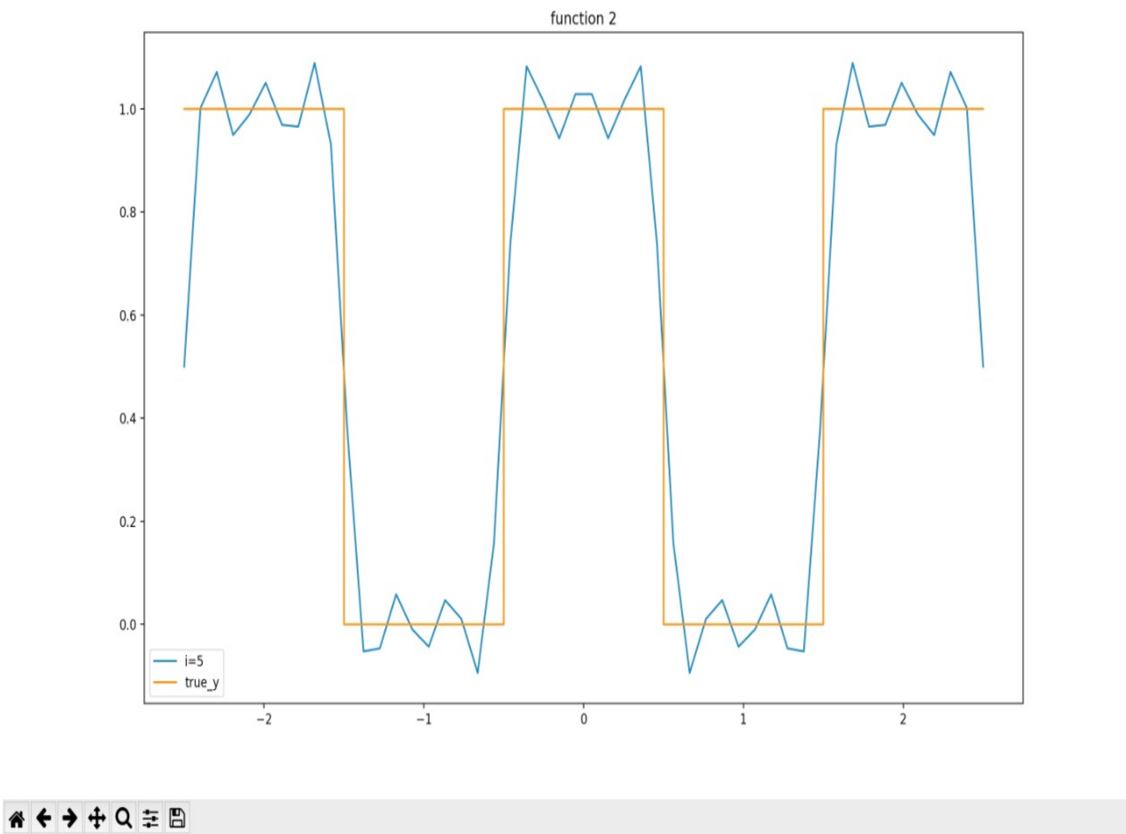

function 1
i=20
true_y
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2

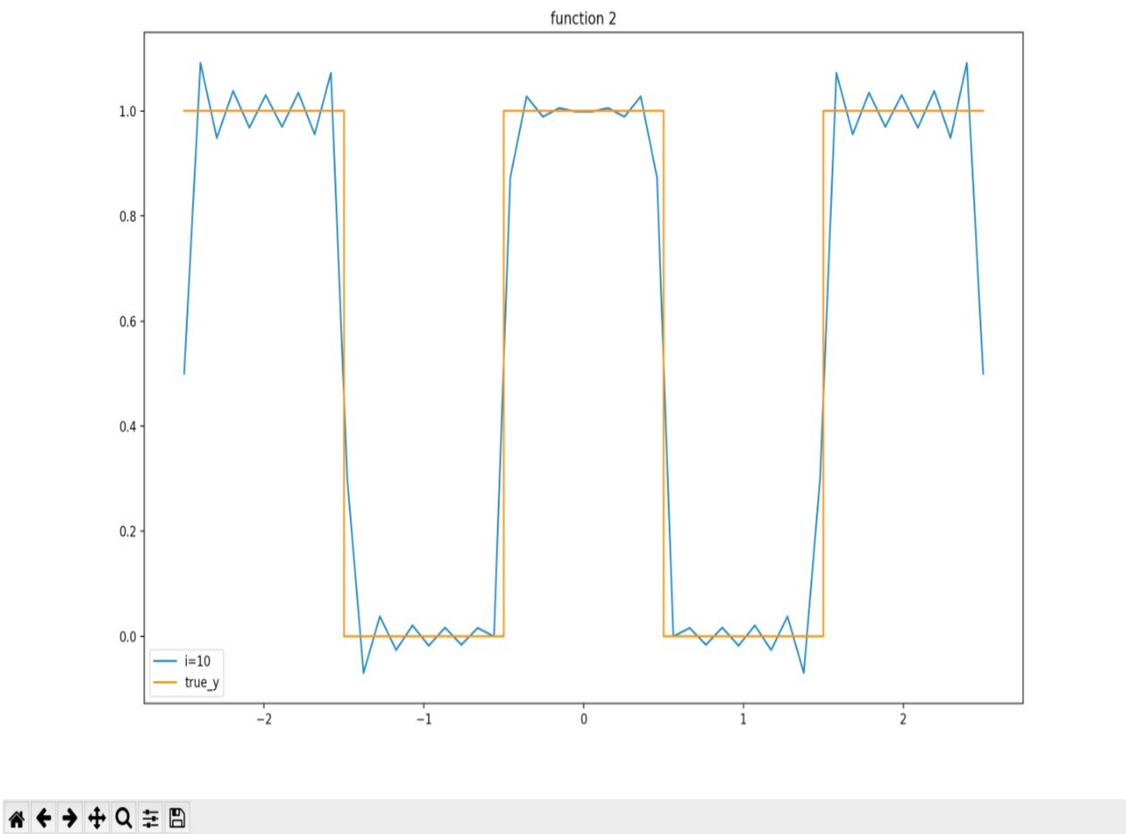
Function 2

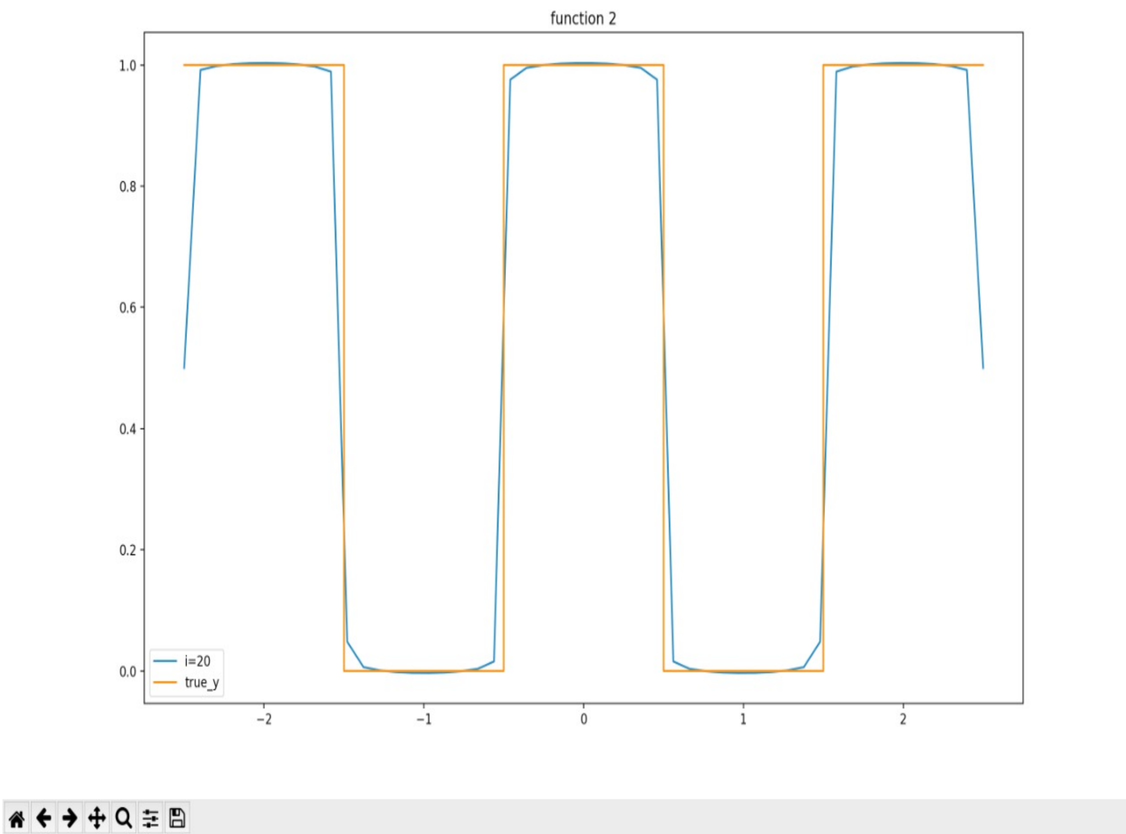

fourier series prac
main.py
Numerical_Methods.py
Run: main
/usr/local/bin/python3.9 "/Users/jagjyotsingh/PycharmProjects/fourier series prac/main.py"
Enter the number of terms for the given functions:
10
a02 : 1.0
N an2 bn2
0 1 6.366198e-01 0.0
1 2 9.020562e-17 0.0
2 3 -2.122066e-01 0.0
3 4 -2.081668e-17 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
i Sum_i
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
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log


function 2
i=1
true_y
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2

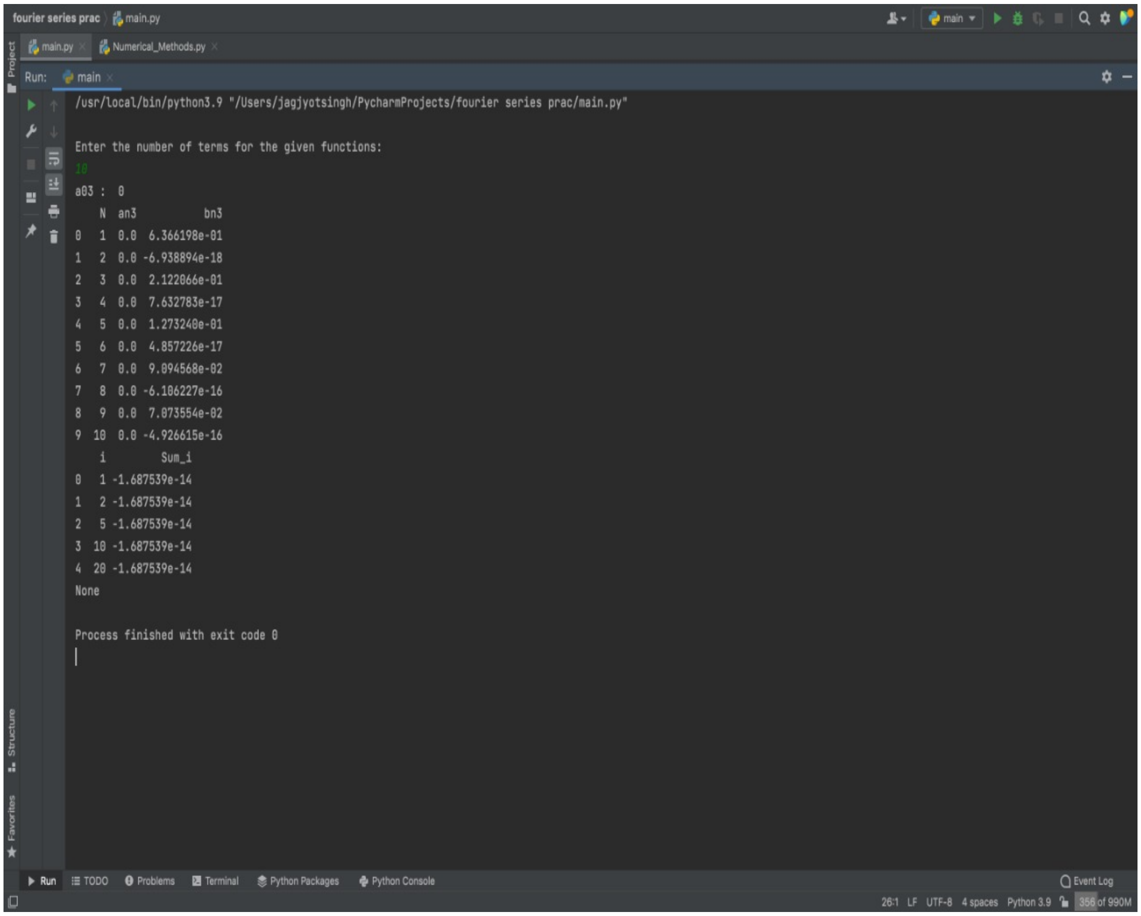

function 2
i=2
true_y
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2

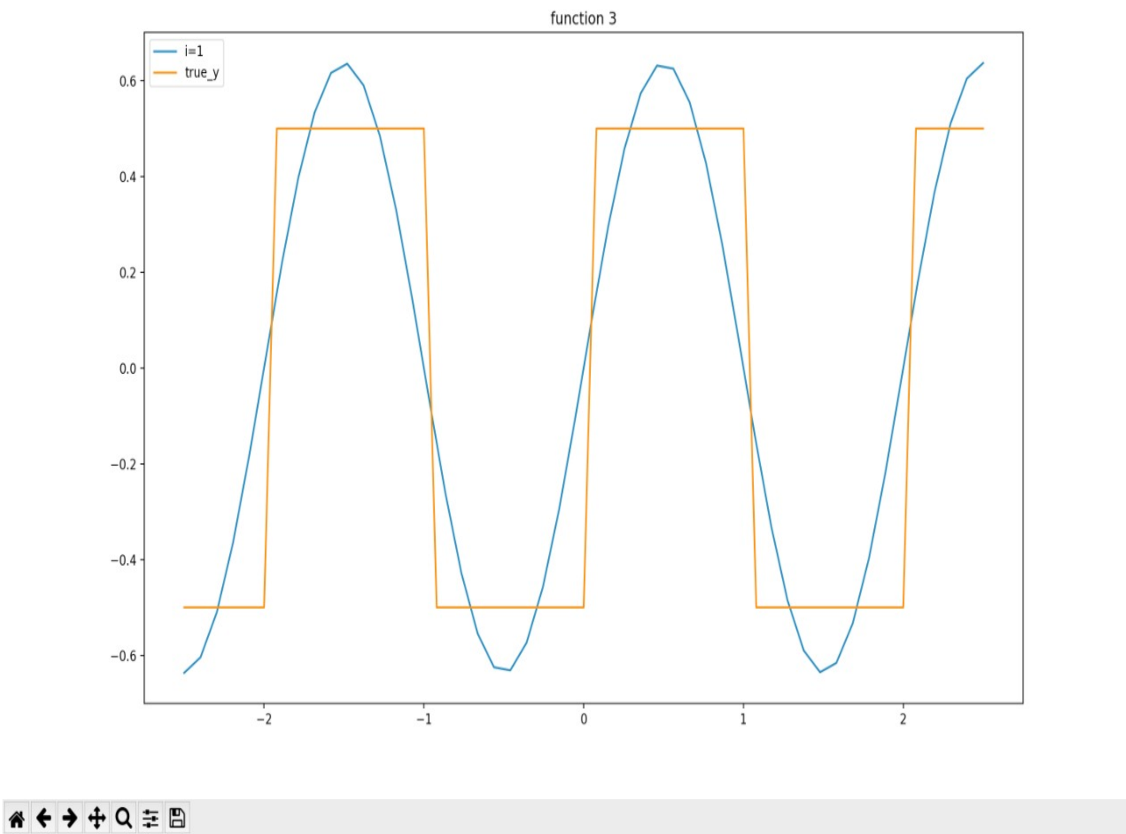

function 2
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2
i=5
true_y

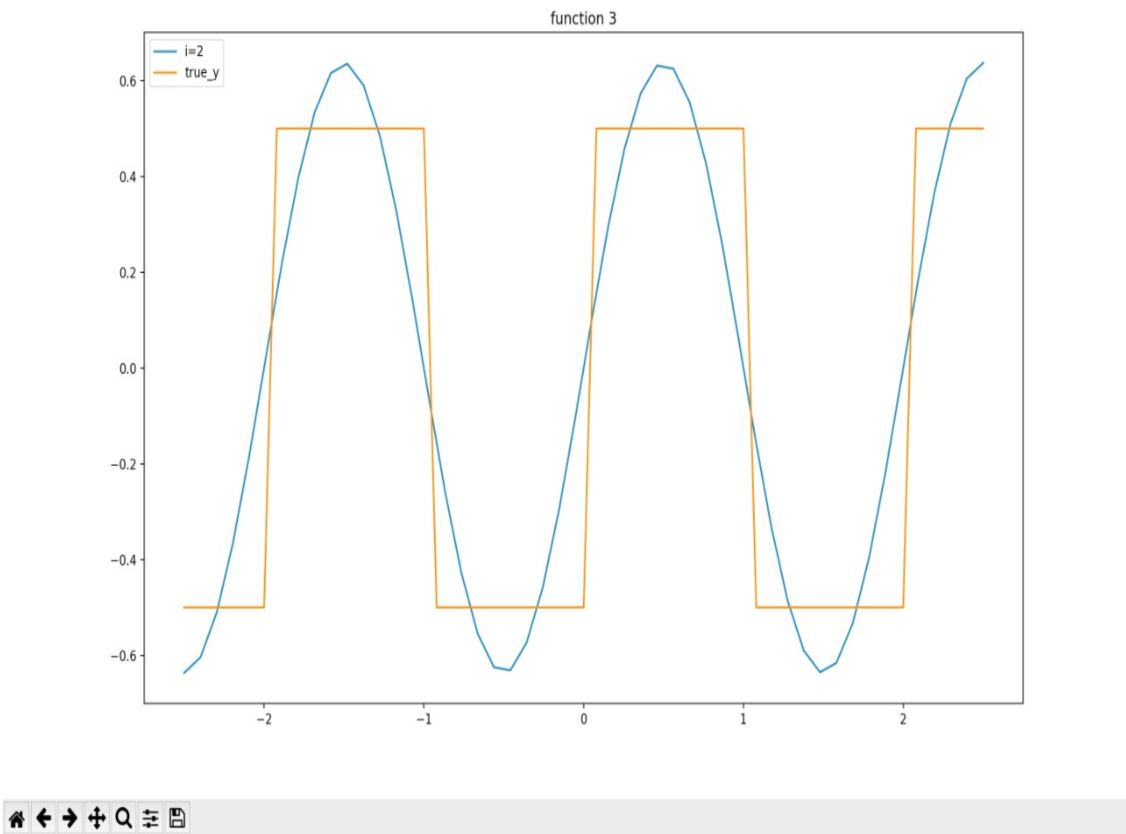

function 2
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2
i=10
true_y

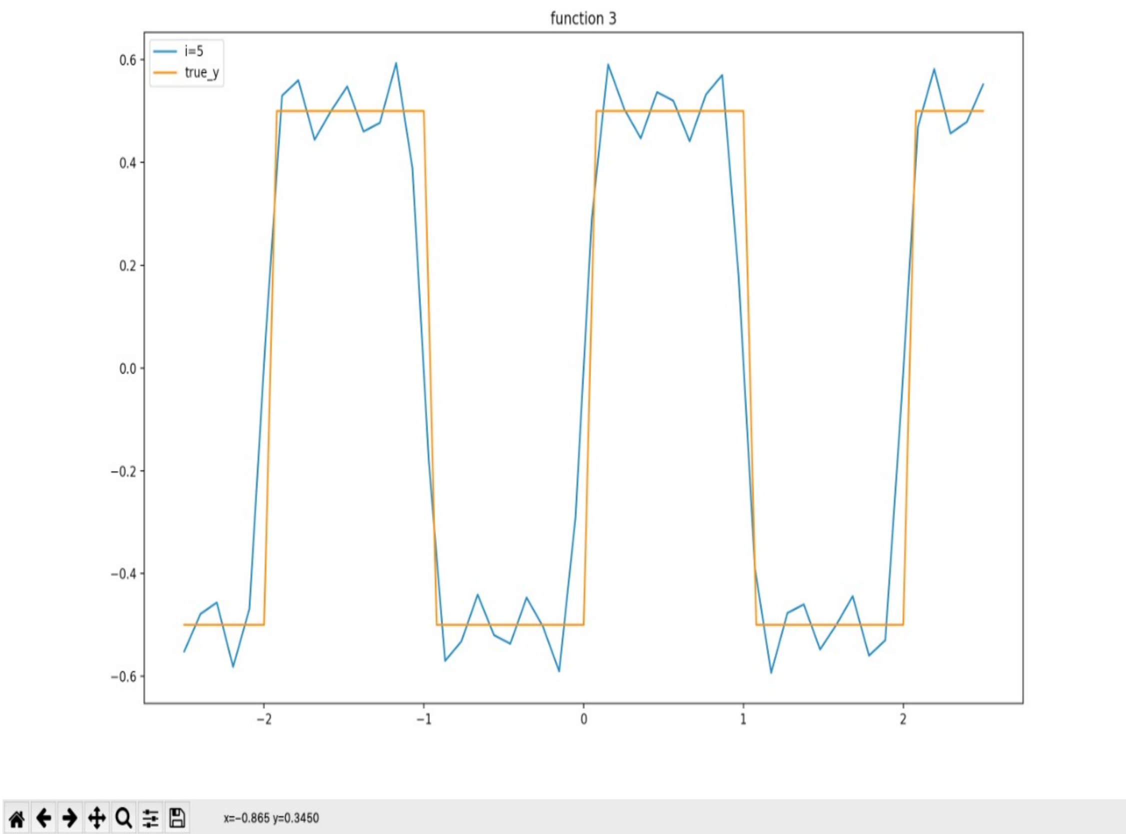

function 2
1.0
0.8
0.6
0.4
0.2
0.0
−2
−1
0
1
2
i=20
true_y

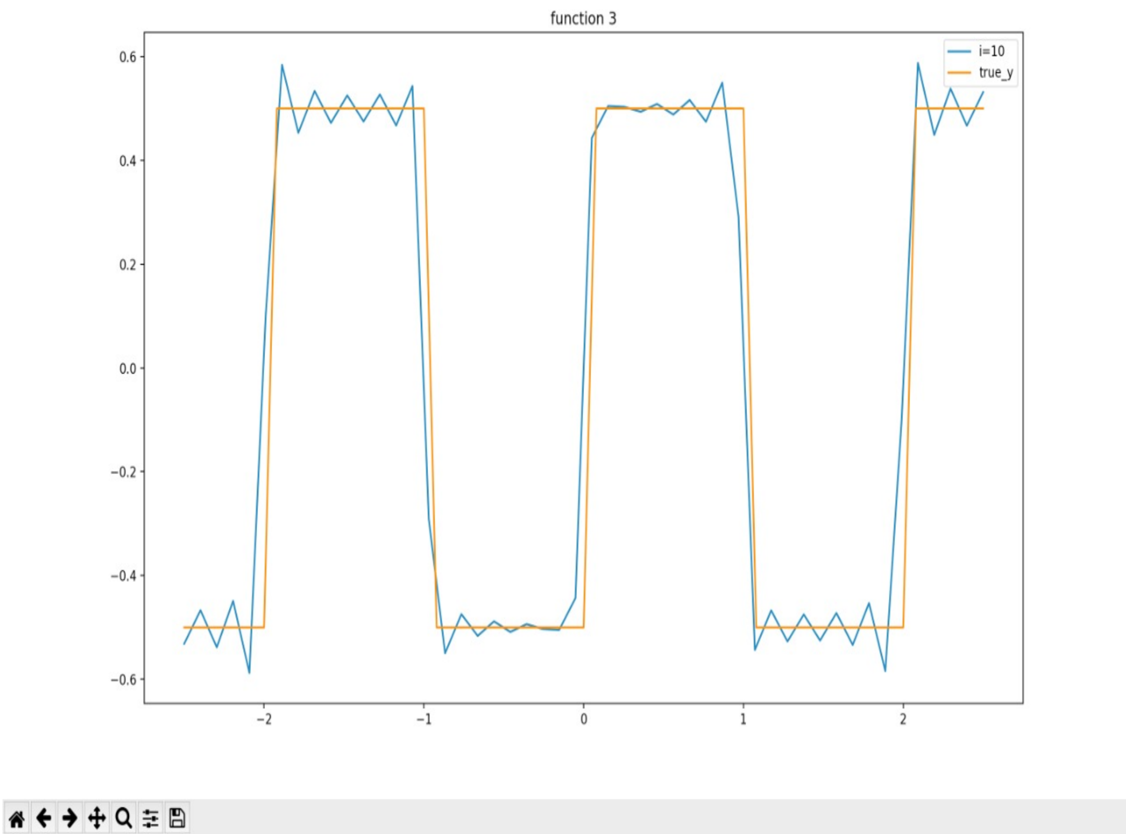
Function 3

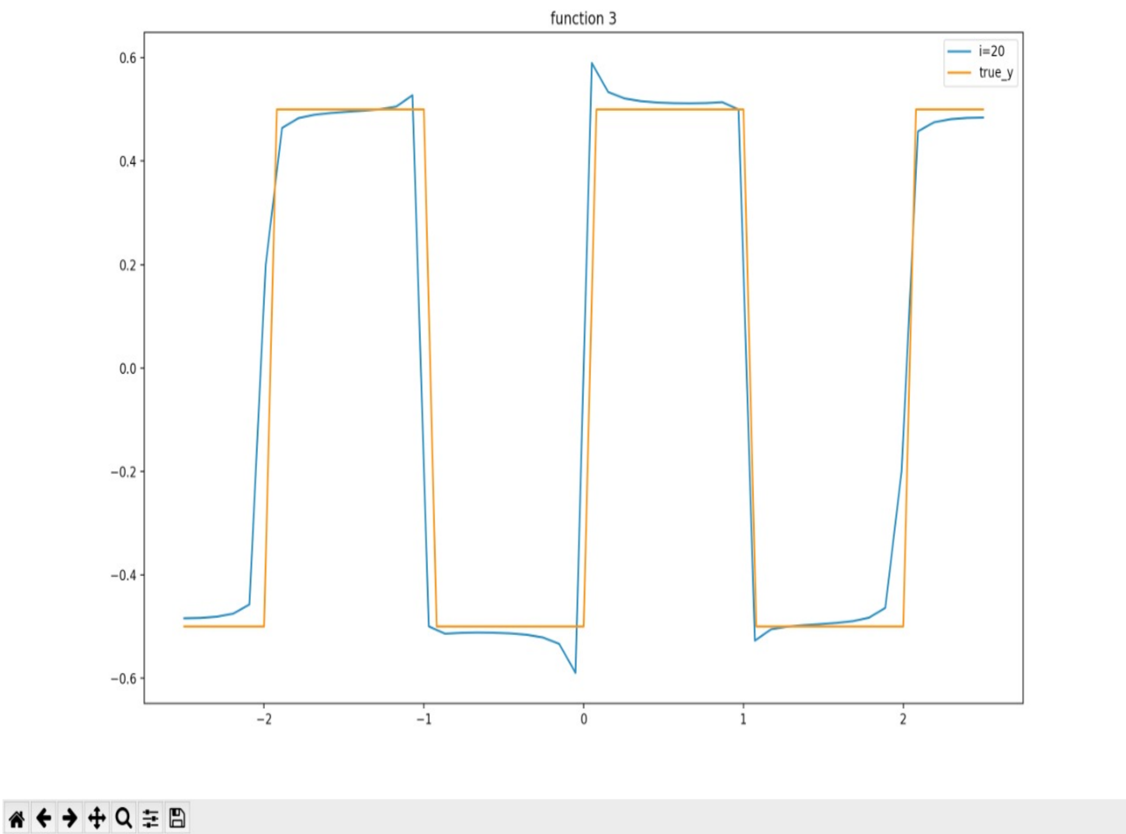

fourier series prac
main.py
main.py
Numerical_Methods.py
Run:
main
/usr/local/bin/python3.9 "/Users/jagjyotsingh/PycharmProjects/fourier series prac/main.py"
Enter the number of terms for the given functions:
10
a03 : 0
    N an3   bn3
0   1 0.0 6.366198e-01
1   2 0.0 -6.938894e-18
2   3 0.0 2.122066e-01
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
    i   Sum_i
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
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log


function 3
i=1
true_y
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−2
−1
0
1
2


function 3
i=2
true_y
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−2
−1
0
1
2


function 3
i=5
true_y
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−2
−1
0
1
2
x=-0.865 y=0.3450


function 3
i=10
true_y
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−2
−1
0
1
2


function 3
i=20
true_y
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−2
−1
0
1
2

Programming

Python program

Main File

```python
import numpy as np
import matplotlib.pyplot as plt
import sympy as sym
from MyIntegration import MyLegQuadrature_tol_1, MyLegQuadrature, MyTrap_tol,
    MySimp_tol
import pandas as pd


x = sym.symbols('x')


def product(f1, f2):
    return lambda x: f1(x) * f2(x)


def sumfn(f1, f2):
    return lambda x: f1(x) + f2(x)


def FourierCoeff(method, d, f, dis, L, N, var):

    def coef_a0():

        a0 = 0
        low = dis[0]

        if method == 'quad':  # if method is quad

            for i in range(len(f)):
                # defining high index
                high = dis[i + 1]

                a0 += MyLegQuadrature_tol_1(f[i], low, high, n=5, d=d)
                low = high

        elif method == 'trap':

            for i in range(len(f)):
                # defining high index
                high = dis[i + 1]

                a0 += MyTrap_tol(f[i], low, high, n=20, d=d)
                low = high

        elif method == 'simp':

            for i in range(len(f)):
                # defining high index
                high = dis[i + 1]

                a0 += MySimp_tol(f[i], low, high, n=20, d=d)
                low = high

        return a0 / L

    def coef_an():
        an = []
        if method == 'quad':

            for i in range(1, N + 1):
```

```python
                # i th coefficient of fourier series
                ai = 0

                f_for_a = lambda x: np.cos(i * np.pi * x / L)

                low = dis[0]

                for j in range(len(f)):
                    high = dis[j + 1]
                    ai += MyLegQuadrature_tol_1(product(f[j], f_for_a), low, high,
    n=5, d=d)

                    low = high

                an.append(ai)

        elif method == 'trap':

            for i in range(1, N + 1):
                # i th coefficient of fourier series
                ai = 0

                f_for_a = lambda x: np.cos(i * np.pi * x / L)

                low = dis[0]

                for j in range(len(f)):
                    high = dis[j + 1]
                    ai += MyTrap_tol(product(f[j], f_for_a), low, high, n=20, d=d)

                    low = high
                an.append(ai)

        elif method == 'simp':

            for i in range(1, N + 1):
                # i th coefficient of fourier series
                ai = 0

                f_for_a = lambda x: np.cos(i * np.pi * x / L)

                low = dis[0]

                for j in range(len(f)):
                    high = dis[j + 1]
                    ai += MySimp_tol(product(f[j], f_for_a), low, high, n=20, d=d)

                    low = high
                an.append(ai)

        return an

    def coef_bn():


        bn = []
        if method == 'quad':

            for i in range(1, N + 1):
                # i th coefficient of fourier series
                bi = 0

                f_for_b = lambda x: np.sin(i * np.pi * x / L)

                low = dis[0]

                for j in range(len(f)):
```

```
                    high = dis[j + 1]
                    bi += MyLegQuadrature_tol_1(product(f[j], f_for_b), low, high, n=5, d=d)

                    low = high

                bn.append(bi)

        elif method == 'trap':

            for i in range(1, N + 1):
                # i th coefficient of fourier series
                bi = 0

                f_for_b = lambda x: np.sin(i * np.pi * x / L)

                low = dis[0]

                for j in range(len(f)):
                    high = dis[j + 1]
                    bi += MyTrap_tol(product(f[j], f_for_b), low, high, n=20, d=d)

                    low = high
                bn.append(bi)

        elif method == 'simp':

            for i in range(1, N + 1):
                # i th coefficient of fourier series
                bi = 0

                f_for_b = lambda x: np.sin(i * np.pi * x / L)

                low = dis[0]

                for j in range(len(f)):
                    high = dis[j + 1]
                    bi += MySimp_tol(product(f[j], f_for_b), low, high, n=20, d=d)

                    low = high
                bn.append(bi)

        return bn

    if var == 0:
        a0 = coef_a0()
        an = coef_an()
        bn = np.zeros(N)

    elif var == 1:
        a0 = 0
        an = np.zeros(N)
        bn = coef_bn()

    else:
        a0 = coef_a0()
        an = coef_an()
        bn = coef_bn()

    return a0, an, bn


print('\nEnter the number of terms for the given functions:')
n = int(input())


def Q3i(n):
```

```
    f1 = [lambda x: 0, lambda x: 1]
    dis1 = [-1, 0, 1]
    L = 1
    a01, an1, bn1 = FourierCoeff("quad", 5, f1, dis1, 1, n, -1)
    T = [t for t in range(1, n + 1)]

    df = pd.DataFrame({'N': T, 'an1': an1, 'bn1': bn1})
    print('a01    : ', a01)
    print(df)
    S = []
    i = [1, 2, 5, 10, 20]
    xi = np.linspace(-2.5, 2.5, 50)

    y1 = []
    for n in i:
        sum_i = lambda x: a01 / 2

        for j in range(len(i)):
            f_add = lambda x: an1[j] * np.cos((j + 1) * x * np.pi / L) + bn1[j] *
    np.sin((j+1) * x * np.pi / L)

            sum_i = sumfn(sum_i, f_add)
        y1.append(sum_i)
        S.append(np.array(sum_i(xi)))

    s1 = []
    for k in range(len(S)):
        s1.append(sum(S[k]))
    dt = pd.DataFrame({'i': i, 'Sum_i': s1})
    print(dt)

    an_c = []
    bn_s = []

    def func(x, i):
        a0, an, bn = FourierCoeff("quad", d=5, f=f1, dis=dis1, L=1, N=i, var=-1)
        for j in range(0, len(an)):
            an_c.append(an[j] * np.cos((j + 1) * np.pi * x / 1))
            bn_s.append(bn[j] * np.sin((j + 1) * np.pi * x / 1))
        four = (a0 / 2 + sum(an_c) + sum(bn_s))
        return four, an_c, bn_s

    #plt.plot(xi, func(xi, 1)[0], label="i=1")
    #plt.plot(xi, func(xi, 2)[0], label="i=2")
    #plt.plot(xi, func(xi, 5)[0], label="i=5")
    #plt.plot(xi, func(xi, 10)[0], label="i=10")
    plt.plot(xi, func(xi, 20)[0], label="i=20")

    f = open("file1.dat", 'wb')
    x1 = np.column_stack((an1, bn1))
    np.savetxt(f, x1)
    f.close()

    x_true1 = [-2.5, -2, -1.92, -1, -0.92, 0, 0.08, 1, 1.08, 2, 2.08, 2.5]
    y_true1 = [0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1]

    plt.plot(x_true1, y_true1, label='true_y')
    plt.legend()
    plt.show()


#print(Q3i(n))


def Q3ii(n):
    # for function (2)
    f2 = [lambda x: 0, lambda x: 1, lambda x: 0]
```

```
    dis2 = [-1, -0.5, 0.5, 1]
    a02, an2, bn2 = FourierCoeff("quad", 6, f=f2, dis=dis2, L=1, N=n, var=0)
    L = 1
    T = [t for t in range(1, n + 1)]
    xi = np.linspace(-2.5, 2.5, 50)

    df = pd.DataFrame({'N': T, 'an2': an2, 'bn2': bn2})
    print('a02 : ', a02)
    print(df)
    S = []
    i = [1, 2, 5, 10, 20]


    y1 = []
    for n in i:
        sum_i = lambda x: a02 / 2

        for j in range(len(i)):
            f_add = lambda x: an2[j] * np.cos((j + 1) * x * np.pi / L) + bn2[j] *
    np.sin((j+1) * x * np.pi / L)

            sum_i = sumfn(sum_i, f_add)
        y1.append(sum_i)
        S.append(np.array(sum_i(xi)))

    s1 = []
    for k in range(len(S)):
        s1.append(sum(S[k]))
    dt = pd.DataFrame({'i': i, 'Sum_i': s1})
    print(dt)

    an_c = []
    bn_s = []

    def func(x, i):
        a0, an, bn = FourierCoeff("quad", d=5, f=f2, dis=dis2, L=1, N=i, var=-1)
        for j in range(0, len(an)):
            an_c.append(an[j] * np.cos((j + 1) * np.pi * xi / 1))
            bn_s.append(bn[j] * np.sin((j + 1) * np.pi * xi / 1))
        four = (a0 / 2 + sum(an_c) + sum(bn_s))
        return four, an_c, bn_s

    plt.plot(xi, func(xi, 1)[0], label="i=1")
    #plt.plot(xi, func(xi, 2)[0], label="i=2")
    #plt.plot(xi, func(xi, 5)[0], label="i=5")
    #plt.plot(xi, func(xi, 10)[0], label="i=10")
    #plt.plot(xi, func(xi, 100)[0], label="i=20")
    #print(FourierCoeff("quad", d=5, f=f2, dis=dis2, L=1, N=10, var=-1))

    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]
    y_true1 = [1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1]
    f = open("file2.dat", 'wb')
    x1 = np.column_stack((an2, bn2))
    np.savetxt(f, x1)
    f.close()

    plt.plot(x_true1, y_true1, label='true_y')
    plt.legend()
    plt.show()
    #return a02, an2, bn2


#print(Q3ii(n))
```

```python
def Q3iii(n):
    # for function(3)
    f3 = [lambda x: -0.5, lambda x: 0.5]
    dis3 = [-1, 0, 1]
    a03, an3, bn3 = FourierCoeff("quad", 6, f=f3, dis=dis3, L=1, N=n, var=1)
    L = 1
    T = [t for t in range(1, n + 1)]

    df = pd.DataFrame({'N': T, 'an3': an3, 'bn3': bn3})
    print('a03 : ', a03)
    print(df)
    S = []
    i = [1, 2, 5, 10, 20]
    xi = np.linspace(-2.5, 2.5, 50)

    y1 = []
    for n in i:
        sum_i = lambda x: a03 / 2

        for j in range(len(i)):
            f_add = lambda x: an3[j] * np.cos((j + 1) * x * np.pi / L) + bn3[j] *
    np.sin((j + 1) * x * np.pi / L)

            sum_i = sumfn(sum_i, f_add)
        y1.append(sum_i)
        S.append(np.array(sum_i(xi)))

    s1 = []
    for k in range(len(S)):
        s1.append(sum(S[k]))
    dt = pd.DataFrame({'i': i, 'Sum_i': s1})
    print(dt)

    an_c = []
    bn_s = []

    def func(x, i):
        a0, an, bn = FourierCoeff("quad", d=5, f=f3, dis=dis3, L=1, N=i, var=-1)
        for j in range(0, len(an)):
            an_c.append(an[j] * np.cos((j + 1) * np.pi * xi / 1))
            bn_s.append(bn[j] * np.sin((j + 1) * np.pi * xi / 1))
        four = (a0 / 2 + sum(an_c) + sum(bn_s))
        return four, an_c, bn_s

    #plt.plot(xi, func(xi, 1)[0], label="i=1")
    #plt.plot(xi, func(xi, 2)[0], label="i=2")
    #plt.plot(xi, func(xi, 5)[0], label="i=5")
    #plt.plot(xi, func(xi, 10)[0], label="i=10")
    #plt.plot(xi, func(xi, 20)[0], label="i=20")

    x_true1 = [-2.5, -2, -1.92, -1, -0.92, 0, 0.08, 1, 1.08, 2, 2.08, 2.5]
    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]

    f = open("file3.dat", 'wb')
    x1 = np.column_stack((an3, bn3))
    np.savetxt(f, x1)
    f.close()

    plt.plot(x_true1, y_true1, label='true_y')
    plt.legend()
    plt.show()
    #return a03, an3, bn3


print(Q3iii(n))
```

My Integration Module

```
from sympy import *
from scipy.special.orthogonal import p_roots
from scipy import integrate
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import texttable as tt
tab = tt.Texttable()

x = symbols('x')
#f = eval("lambda x:" + input("function to be integrated, f(x) = "))
#d = int(input("Enter the number of significant figures upto which results should
    be accurate = "))


def My_Trap(f_, a, b, n):
    y = []
    h = (b - a) / n
    for i in range(n + 1):
        y.append(f_(a + i * h))  # y at limit points
    trp = h * (f_(a) + f_(b)) / 2
    for j in range(1, len(y) - 1):
        trp = trp + h * (y[j])
    return (trp)


def My_Simp(f_, a, b, n):
    h = (b - a) / (2 * n)
    simp = h * (f_(a) + f_(b)) / 3
    for i in range(1, 2 * n):
        if (i % 2 == 0):
            simp = simp + 2 * h * f_(a + i * h) / 3
        elif (i % 2 == 1):
            simp = simp + 4 * h * f_(a + i * h) / 3

    return (simp)


def MyTrap_tol(f_, a, b, n, d):
    i = 1
    err = []
    while True:
        e = abs(My_Trap(f_, a, b, i) - My_Trap(f_, a, b, i + 1))
        err.append(e)
        if e > 0.5 * 10 ** -d:
            i = i + 1
        elif e <= 0.5 * 10 ** -d:
            p = My_Trap(f_, a, b, i)
            break
    return p


def MySimp_tol(f_, a, b, n, d):
    i = 1
    err = []
    while True:
        e = abs(My_Simp(f_, a, b, i) - My_Simp(f_, a, b, i + 1))
        err.append(e)
        if e > 0.5 * 10 ** -d:
            i = i + 1
        elif e <= 0.5 * 10 ** -d:
            p = My_Simp(f_, a, b, i)
            break
    return p

```

```python

def MyLegQuadrature(f, a, b, n, m):  # Gauss legendre Quadrature ftion
    h = (b - a) / m
    [leg_zer, w] = p_roots(n)
    leg_zer.tolist()
    w.tolist()
    sum_ = 0
    x_ = [a]
    s = []
    for k in range(0, n):
        for i in range(1, m + 1):
            x_.append(a + h * i)
            sum_ += (h / 2) * w[k] * f(0.5 * h * leg_zer[k] + 0.5 * (x_[i] + x_[i - 1]))
            s.append(sum_)
    return sum_


def MyLegQuadrature_tol(fs, a, b, n, m, d):
    i = 1
    err = []
    while i <= m:
        e = abs(MyLegQuadrature(fs, a, b, n, i) - MyLegQuadrature(fs, a, b, n, i + 1)) / MyLegQuadrature(fs, a, b, n,
                            i + 1)
        err.append(e)
        if e > 0.5 * 10 ** -d:
            i = i * 2
            if i > m:
                print("Tolerance can't be reached for ", m, "Subintervals")
        elif e <= 0.5 * 10 ** -d:
            print("tolerance is reached in", i, "subintervals")
            print("integration and error using n point method(composite) = ", MyLegQuadrature(fs, a, b, n, i))
            # print("integration using inbuilt ftion = ", integrate.quadrature(f, a, b))
            break
        return MyLegQuadrature(fs, a, b, n, i)


def MyLegQuadrature_tol_1(fs, a, b, n, d):
    i = 1
    p = 0
    err = []
    while True:
        e = abs(MyLegQuadrature(fs, a, b, n, i) - MyLegQuadrature(fs, a, b, n, i + 1))
        err.append(e)
        if e > 0.5 * 10 ** -d:
            i = i * 2
        elif e <= 0.5 * 10 ** -d:
            p = MyLegQuadrature(fs, a, b, n, i)
            break
    return p
```