Mathematical Physics III

Lab Assignment #3

Ishmeet Singh

College Roll No	:	2020PHY1221
University Roll No	:	20068567025
Unique Paper Code	:	32221401
Paper Title	:	Mathematical physics III Lab
Course and Semester	:	B.Sc.(Hons.) Physics, Sem IV
Due Date	:	28 Feb 2022
Date of Submission	:	28 Feb 2022
Lab Report File Name	:	A3-2020PHY1221.pdf
Submitted to	:	Dr. Mamta Dahiya

Team Members :

Name	:	Sarthak Jain
Roll Number	:	2020PHY1201
Name	:	Swarnim Gupta
Roll Number	:	2020PHY1014

Shri Guru Tegh Bahadur Khalsa College, University of Delhi
New Delhi-110007, India.

Contents

1 Theory **1**

1.1 Dirichlet Conditions for a Fourier Series 1

1.2 Fourier Representation of a Periodic Function 1

1.2.1 Derivation of Expressions for Fourier Coefficients 2

1.2.2 If $f(x)$ is even or odd function 3

1.2.3 Fourier Series Representation for some functions 4

1.3 Gibbs Phenomenon 9

1.4 Half Range Expansion 10

1.4.1 How do you write the fourier series representation for a function $f(x)$ that is defined in a finite range , say , $0 < x < L$? 10

1.4.2 What do you mean by half range sine and cosine expansion? 10

1.4.3 Odd Function and Half Range Sine Series 10

1.4.4 $f(x) = x, 0 < x < \pi$ 10

2 Programs **15**

2.1 A3a-2020PHY1221.py Module 15

2.2 A3b-2020PHY1221.py Module 24

3 Result **27**

4 Discussion **29**

1 Theory

1.1 Dirichlet Conditions for a Fourier Series

If the function $f(x)$ for the interval $(-\pi, \pi)$

1. is single-valued
2. is bounded
3. has at most a finite number of maxima and minima
4. has only a finite number of discontinuities
5. is $f(x+2\pi) = f(x)$ for values of x outside $[-\pi, \pi]$, then

$$S_p(x) = \frac{a_0}{2} + \sum_{n=1}^{P} a_n \cos nx + \sum_{n=1}^{P} b_n \sin nx$$

converges to $f(x)$ as $P \to \infty$ at values of x for which $f(x)$ is continuous and the sum of the series is equal to $\frac{1}{2}[f(x+0) + f(x-0]$ at points of discontinuity.
These conditions are sufficient.

1.2 Fourier Representation of a Periodic Function

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

where,

$$a_n = \frac{2}{b-a}\int_a^b f(x)\cos\left(\frac{2n\pi x}{b-a}\right)dx \qquad n = 1, 2, \ldots$$

$$b_n = \frac{2}{b-a}\int_a^b f(x)\sin\left(\frac{2n\pi x}{b-a}\right)dx \qquad n = 1, 2, \ldots$$

1.2.1 Derivation of Expressions for Fourier Coefficients

$$\begin{aligned} f(x) = \frac{a_0}{2} + a_1 \cos x + a_2 \cos 2x + \ldots + a_n \cos nx + \ldots + \\ b_1 \sin x + b_2 \sin 2x + \ldots + b_n \sin nx + \ldots \end{aligned} \tag{1}$$

(i) To find a_0 : Integrate both sides of (1) from $x = 0$ to $x = 2\pi$

$$\begin{aligned} \int_0^{2\pi} f(x)\,dx &= \frac{a_0}{2}\int_0^{2\pi} dx + a_1 \int_0^{2\pi} \cos x\,dx + a_2 \int_0^{2\pi} \cos 2x\,dx + \ldots + a_n \int_0^{2\pi} \cos nx\,dx + \ldots \\ &+ b_1 \int_0^{2\pi} \sin x\,dx + b_2 \int_0^{2\pi} \sin 2x\,dx + \ldots + b_n \int_0^{2\pi} \sin nx\,dx + \ldots \\ &= \frac{a_0}{2}\int_0^{2\pi} dx \\ \int_0^{2\pi} f(x)\,dx &= \frac{a_0}{2} 2\pi \\ &\implies \boxed{a_0 = \frac{1}{\pi}\int_0^{2\pi} f(x)\,dx} \end{aligned} \tag{2}$$

(ii) To find a_n : Multiply each side of (1) by $\cos nx$ and integrate from $x = 0$ to $x = 2\pi$

$$\begin{aligned} \int_0^{2\pi} f(x)\cos nx\,dx &= \frac{a_0}{2}\int_0^{2\pi} \cos nx\,dx + a_1 \int_0^{2\pi} \cos x \cos nx\,dx + \ldots + a_n \int_0^{2\pi} \cos^2 nx\,dx \ldots \\ &+ b_1 \int_0^{2\pi} \sin x \cos nx\,dx + b_2 \int_0^{2\pi} \sin 2x \cos nx\,dx + \ldots \\ &= a_n \int_0^{2\pi} \cos^2 nx\,dx \\ \int_0^{2\pi} f(x)\,dx &= a_n \pi \\ &\implies \boxed{a_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos nx\,dx} \end{aligned} \tag{3}$$

By taking $n = 1, 2 \ldots$ we can find the values of $a_1, a_2 \ldots$

(iii) To find b_n : Multiply each side of (1) by $\sin nx$ and integrate from $x = 0$ to $x = 2\pi$

$$\begin{aligned}
\int_0^{2\pi} f(x)\sin nx\,dx &= \frac{a_0}{2}\int_0^{2\pi} \sin nx\,dx + a_1\int_0^{2\pi} \cos x\sin nx\,dx + \ldots + a_n\int_0^{2\pi} \cos nx\sin nx\,dx \ldots \\
&+ b_1\int_0^{2\pi} \sin x\sin nx\,dx + \ldots + b_n\int_0^{2\pi} \sin^2 nx\,dx + \ldots \\
&= b_n\int_0^{2\pi} \sin^2 nx\,dx \\
\int_0^{2\pi} f(x)\,dx &= b_n\pi \\
&\implies \boxed{b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin nx\,dx}
\end{aligned} \tag{4}$$

1.2.2 If $f(x)$ is even or odd function

For even function:

$$a_0 = \frac{2}{\pi}\int_0^{\pi} f(x)\,dx$$

$$a_n = \frac{2}{\pi}\int_0^{\pi} f(x)\cos nx\,dx$$

$$b_n = 0$$

For odd function:

$$a_0 = 0$$

$$a_n = 0$$

$$b_n = \frac{2}{\pi}\int_0^{\pi} f(x)\sin nx\,dx$$

1.2.3 Fourier Series Representation for some functions

α)

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

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

Ans:

$$b - a = 2$$

Calculating Fourier Coefficients,

$$\begin{aligned} a_0 &= \frac{2}{b-a} \int_a^b f(x)\,dx \\ &= \frac{2}{2} \int_{-1}^{1} f(x)\,dx \\ &= \int_{-1}^{0} (0)\,dx + \int_0^1 (1)\,dx \\ &= 0 + [x]_0^1 \\ &\implies \boxed{a_0 = 1} \end{aligned}$$

$$\begin{aligned} a_n &= \frac{2}{b-a} \int_a^b f(x) \cos\left(\frac{2n\pi x}{b-a}\right) dx \\ &= \frac{2}{2} \int_{-1}^{1} f(x) \cos\left(\frac{2n\pi x}{2}\right) dx \\ &= \int_{-1}^{1} f(x) \cos(n\pi x)\,dx \\ &= \int_{-1}^{0} (0) \cos(n\pi x)\,dx + \int_0^1 (1) \cos(n\pi x)\,dx \\ &= \left[\frac{\sin(n\pi x)}{n\pi}\right]_0^1 \\ &\implies \boxed{a_n = 0} \end{aligned}$$

$$
\begin{aligned}
b_n &= \frac{2}{b-a}\int_a^b f(x)\sin\left(\frac{2n\pi x}{b-a}\right)dx \\
&= \frac{2}{2}\int_{-1}^{1} f(x)\sin\left(\frac{2n\pi x}{2}\right)dx \\
&= \int_{-1}^{0}(0)\sin(n\pi x)\,dx + \int_0^1 (1)\sin(n\pi x)\,dx \\
&= \left[\frac{-\cos(n\pi x)}{n\pi}\right]_0^1 \\
&= \frac{-\cos n\pi}{n\pi} + \frac{1}{n\pi} \\
&= \frac{1-\cos n\pi}{n\pi} \\
&\implies \boxed{b_n = \frac{1-(-1)^n}{n\pi}}
\end{aligned}
$$

So, the resulting Fourier Series,

$$
\boxed{f(x) = \frac{1}{2} + \sum_{n=1}^{\infty}(0) + \sum_{n=1}^{\infty}\left(\frac{1-(-1)^n}{n\pi}\right)\sin(n\pi x)}
$$

β)

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

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

Ans:

$$
b - a = 2
$$

Calculating Fourier Coefficients,

$$
\begin{aligned}
a_0 &= \frac{2}{b-a}\int_a^b f(x)\,dx \\
&= \frac{2}{2}\int_{-1}^{1} f(x)\,dx \\
&= \int_{-1}^{-0.5} f(x)\,dx + \int_{-0.5}^{0.5} f(x)\,dx + \int_{0.5}^{1} f(x)\,dx \\
&= 0 + [x]_{-0.5}^{0.5} + 0 \\
&\implies \boxed{a_0 = 1}
\end{aligned}
$$

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

$$
\begin{aligned}
b_n &= \frac{2}{b-a}\int_a^b f(x)\sin\left(\frac{2n\pi x}{b-a}\right)dx \\
&= \frac{2}{2}\int_{-1}^{1} f(x)\sin\left(\frac{2n\pi x}{2}\right)dx \\
&= \int_{-1}^{1} f(x)\sin n\pi x\,dx \\
&= \int_{-1}^{-0.5}(0)\sin(n\pi x)\,dx + \int_{-0.5}^{0.5}(1)\sin(n\pi x)\,dx + \int_{0.5}^{1}(0)\sin(n\pi x)\,dx \\
&= \left[\frac{-\cos(n\pi x)}{n\pi}\right]_{-0.5}^{0.5} \\
&= \frac{-\cos\frac{n\pi}{2} + \cos\frac{n\pi}{2}}{n\pi} \\
&= 0 \\
&\implies \boxed{b_n = 0}
\end{aligned}
$$

So, the resulting Fourier Series,

$$\boxed{f(x) = \frac{1}{2} + \sum_{n=1}^{\infty} \left(\frac{2}{n\pi} \sin \frac{n\pi}{2} \right) \cos(n\pi x)}$$

γ)

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

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

Ans:

$$b - a = 2$$

Calculating Fourier Coefficients,

$$\begin{aligned} a_0 &= \frac{2}{b-a} \int_a^b f(x)\, dx \\ &= \frac{2}{2} \int_{-1}^{1} f(x)\, dx \\ &= \int_{-1}^{0} (-0.5)\, dx + \int_0^1 (0.5)\, dx \\ &= \frac{-1}{2} [x]_{-1}^{0} + \frac{1}{2} [x]_0^1 \\ &= \frac{-1}{2}.1 + \frac{1}{2}.1 \\ &\implies \boxed{a_0 = 0} \end{aligned}$$

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

$$
\begin{aligned}
b_n &= \frac{2}{b-a}\int_a^b f(x)\sin\left(\frac{2n\pi x}{b-a}\right)dx \\
&= \frac{2}{2}\int_{-1}^{1} f(x)\sin\left(\frac{2n\pi x}{2}\right)dx \\
&= \int_{-1}^{1} f(x)\sin n\pi x\,dx \\
&= \int_{-1}^{0}(-0.5)\sin\left(n\pi x\right)dx + \int_{0}^{1}(0.5)\sin\left(n\pi x\right)dx \\
&= \frac{-1}{2n\pi}\left[-\cos n\pi x\right]_{-1}^{0} \\
&= \frac{1}{2n\pi}(1-\cos n\pi) + \frac{1}{2n\pi}\left[-\cos n\pi x\right]_{0}^{1} \\
&= \frac{1}{2n\pi}(1-\cos n\pi) + \frac{1}{2n\pi}(-\cos n\pi + 1) \\
&= 2.\frac{1}{2n\pi}(1-\cos n\pi) \\
&= \frac{1}{n\pi}(1-\cos n\pi) \\
&\implies \boxed{b_n = \frac{1-(-1)^n}{n\pi}}
\end{aligned}
$$

So, the resulting Fourier Series,

$$
\boxed{f(x) = 0 + 0 + \sum_{n=1}^{\infty}\left(\frac{1-(-1)^n}{n\pi}\right)\sin\left(n\pi x\right)}
$$

1.3 Gibbs Phenomenon

For a periodic signal with discontinuities, if the signal is reconstructed by adding the Fourier series, then overshoots appear around the edges. These overshoots decay outwards in a damped oscillatory manner away from the edges. This is known as **'Gibbs Phenomenon'** and is shown in the figure below.

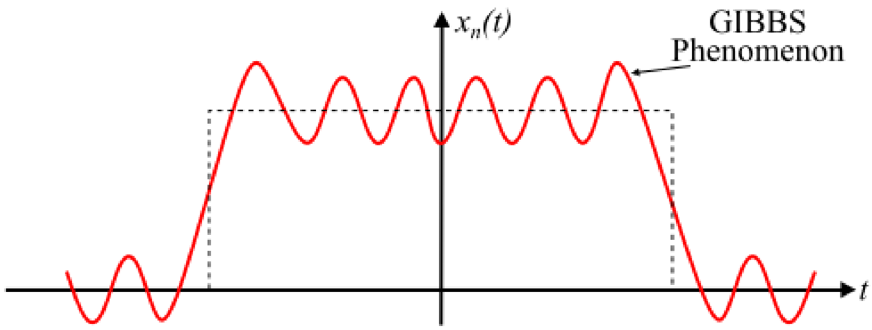

Figure 1: Gibbs Phenomenon

The amount of the overshoots at the discontinuities is proportional to the height of discontinuity and according to Gibbs, it is found to be around 9% of the height of discontinuity irrespective of the number of terms in the Fourier series. The exact proportion is given by the *Wilbraham-Gibbs Constant*.

$$\frac{1}{\pi}\int_0^{\pi}\frac{sin(t)}{t}dt-\frac{1}{2}=0.089489\ldots$$

It may also be noted that as more number of terms in the series are added, the frequency increases and the overshoots become sharper, but the amplitude of the adjoining oscillation reduces, i.e., the error between the original signal $x(t)$ and the truncated signal $x_n(t)$ reduces except at edges as the n increases. Hence, the truncated Fourier series approaches the original signal $x(t)$ as the number of terms in approximation increases.

1.4 Half Range Expansion

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.

1.4.1 How do you write the fourier series representation for a function $f(x)$ that is defined in a finite range , say , $0 < x < L$?

1.4.2 What do you mean by half range sine and cosine expansion?

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

for n=1,2,3,4 ... where

$$a_0 = \frac{2}{L}\int_0^L f(x)dx$$

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

$$b_n = 0$$

1.4.3 Odd Function and Half Range Sine Series

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

$$f(x) = \sum_{n=1}^{\infty} b_n sin(\frac{n\pi x}{L})$$

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

1.4.4 $f(x) = x, 0 < x < \pi$

Let f be a function given on the interval (0,a), we define the even/odd extension of f to be even/odd functions on the interval (-a,a), which coincides with f on the half interval (0,a).

Half range even extension

Sketching $f(x) = x$ from $x = 0$ to $x = \pi$:

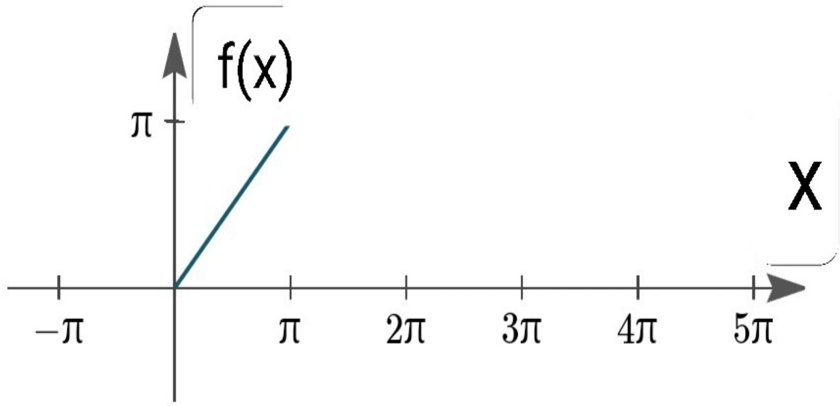

Figure 2: Graph of f(x)

An even function means that it must be symmetrical about the f(x) axis and this is shown in the following figure by the broken line between $t = -\pi$ and $t = 0$.

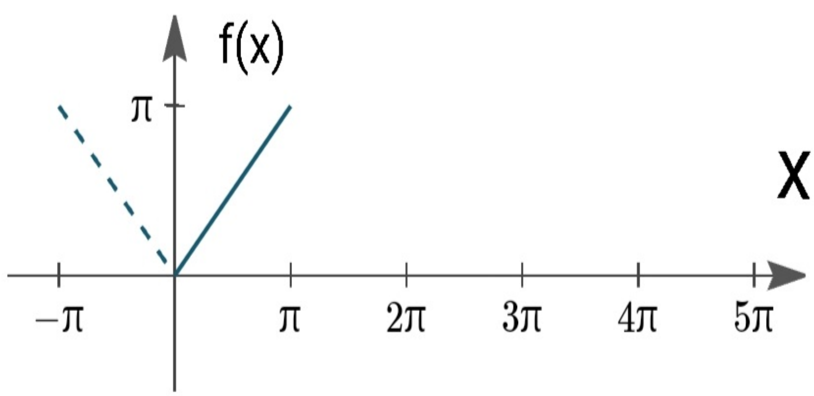

Figure 3: Graph of f(x) showing it as an even function

The "triangular wave form" produced is periodic with period 2π outside of this range as shown by the dotted lines.

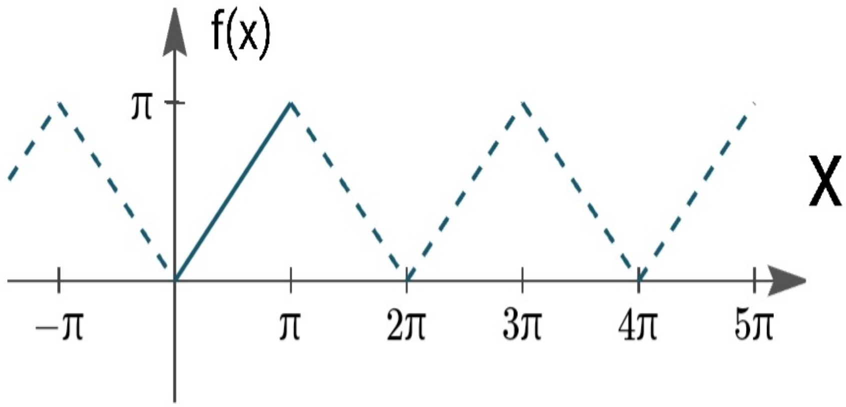

Figure 4: A periodic even function

Half range even extension:

$$f(x) = \begin{cases} -x & \text{if} \quad -\pi \leq x < 0 \\ x & \text{if} \quad 0 \leq x < \pi \end{cases}$$

f(x) is periodic with period 2π

Since the function is even:

$$b_n = 0$$

Here, $L = \pi$

$$\begin{aligned} a_0 &= \frac{2}{L}\int_0^L f(x)\,dx \\ &= \frac{2}{\pi}\int_0^\pi x\,dx \\ &= \frac{2}{\pi}\left[\frac{x^2}{2}\right]_2^\pi \\ &= \frac{2}{\pi}\left[\frac{\pi^2}{2}\right] \\ &= \pi \end{aligned}$$

$$\begin{aligned} a_n &= \frac{2}{L}\int_0^L f(x)cos\left(\frac{n\pi x}{L}\right)\,dx \\ &= \frac{2}{\pi}\int_0^\pi xcos(nx)\,dx \end{aligned}$$

We know,

$$\int xcos(nx)dx = \frac{1}{n^2}[cos(nx) + nxsin(nx)]$$
$$= \frac{2}{\pi}[\frac{1}{n^2}[cos(nx) + nxsin(nx)]_0^\pi$$
$$= \frac{2}{\pi n^2}[(cos(n\pi) + 0) - (cos0 + 0)]$$
$$= \frac{2}{\pi n^2}[(cos(n\pi) - 1]$$
$$= \frac{2}{\pi n^2}[(-1)^n - 1]$$
$$= \frac{-4}{n^2}$$

(When n is odd)

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n cos(\frac{n\pi x}{L})$$
$$= \frac{\pi}{2} - \frac{4}{\pi}\sum_{n=1}^{\infty}\frac{cos(2n-1)x}{(2n-1)^2}$$
$$f(x) = \boxed{f(x) = \frac{\pi}{2} - \frac{4}{\pi}(cosx + \frac{1}{9}cos3x + \frac{1}{25}cos5x + \ldots}$$

Half range odd extension

Sketching $f(x) = x$ from $x = 0$ to $x = \pi$:

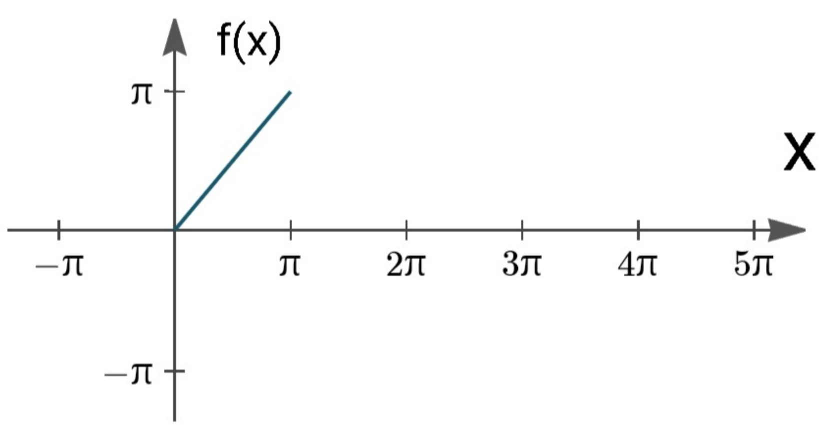

Figure 5: Graph of f(x)

An odd function means that it is symmetrical about the origin and this is shown by the broken lines between $x = -\pi$ and $t = 0$

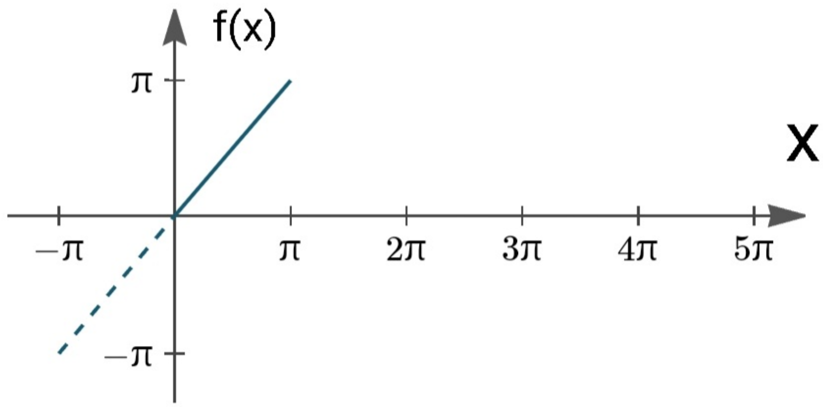

Figure 6: Graph of f(x) showing it as an odd function

2 Programs

2.1 A3a-2020PHY1221.py Module

```
import MyIntegration as mi
from sympy import *
import numpy as np
import matplotlib.pyplot as plt

def FourierCoeff(expr, N, L, method, ftype = -1, d = 1, n = 100, pw = False,n_max = 1000):
    a0 = 0
    an = []
    bn = []

    if method == "trapezoidal" or method == "Trapezoidal":
        if ftype == 0:                          # even function
            a0 = mi.MyTrap_tol(expr,L,0,n_max,d)[0]/(L)

            for i in range(1,N+1):
                bn.append(0)
                func = "(" + expr + ")" + "*" + "cos({:}*(x*pi/{:}))".format(str(i), str(L))
                a_coeff = mi.MyTrap_tol(func,L,0,n_max,d)[0]
                an.append(2*a_coeff/L)

            if (pw == True):
                an = np.array(an)/2       # /2 because we require int-0 to L and not int-(-L) to L.

            an = np.array(an)
            bn = np.array(bn)

        elif ftype == 1:                         # odd function

            for i in range(1, N+1):
                an.append(0)
                func = "(" + expr + ")" + "*" + "sin({:}*(x*pi/{:}))".format(str(i), str(L))
                b_coeff = mi.MyTrap_tol(func,L,0,n_max,d)[0]
                bn.append(2*b_coeff/L)

```

```
            if (pw == True):
                a0 = mi.MyTrap_tol(expr,L,0,n_max,d)[0]/(2*L) # a0/2
                bn = np.array(bn)/2       # /2 because we require int-0 to L and not int-(-L) to L.

            an = np.array(an)
            bn = np.array(bn)

        elif ftype == -1:    # Neither Odd nor Even Function
            a0 = mi.MyTrap_tol(expr,L,-L,n_max,d)[0]/(2*L)    # a0/2

            if (pw == True):
                a0 = a0/2

                for i in range(1, N+1):
                    func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi/(2*{:})))".format(str(i), str(L))
                    a_coeff = mi.MyTrap_tol(func_cos,L,-L,n_max,d)[0]
                    an.append(a_coeff/(2*L))

                    func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi/(2*{:})))".format(str(i), str(L))
                    b_coeff = mi.MyTrap_tol(func_sin,L,-L,n_max,d)[0]
                    bn.append(b_coeff/(2*L))

            else:
                for i in range(1, N+1):
                    func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi/{:}))".format(str(i), str(L))
                    a_coeff = mi.MyTrap_tol(func_cos,L,-L,n_max,d)[0]
                    an.append(a_coeff/L)

                    func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi/{:}))".format(str(i), str(L))
                    b_coeff = mi.MyTrap_tol(func_sin,L,-L,n_max,d)[0]
                    bn.append(b_coeff/L)

            an = np.array(an)
            bn = np.array(bn)

    elif method == "simpson" or method == "Simpson":
        if ftype == 0:                      # even function
```

```
            a0 = mi.MySimp_tol(expr,L,0,n_max,d)[0]/(L)

            for i in range(1,N+1):
                bn.append(0)
                func = "(" + expr + ")" + "*" + "cos({:}*(x*pi/{:}))".format(str(i), str(L))
                a_coeff = mi.MySimp_tol(func,L,0,n_max,d)[0]
                an.append(2*a_coeff/L)

            if (pw == True):
                an = np.array(an)/2        # /2 because we require int-0 to L and not int-(-L) to L.

            an = np.array(an)
            bn = np.array(bn)

        elif ftype == 1:                        # odd function

            for i in range(1, N+1):
                an.append(0)
                func = "(" + expr + ")" + "*" + "sin({:}*(x*pi/{:}))".format(str(i), str(L))
                b_coeff = mi.MySimp_tol(func,L,0,n_max,d)[0]
                bn.append(2*b_coeff/L)

            if (pw == True):
                a0 = mi.MySimp_tol(expr,L,0,n_max,d)[0]/(2*L) # a0/2
                bn = np.array(bn)/2        # /2 because we require int-0 to L and not int-(-L) to L.

            an = np.array(an)
            bn = np.array(bn)

        elif ftype == -1:    # Neither Odd nor Even Function
            a0 = mi.MySimp_tol(expr,L,-L,n_max,d)[0]/(2*L)    # a0/2

            if (pw == True):
                a0 = a0/2

                for i in range(1, N+1):
                    func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi/(2*{:})))".format(str(i), str(L))
```

```
                    a_coeff = mi.MySimp_tol(func_cos,L,-L,n_max,d)[0]
                    an.append(a_coeff/(2*L))

                    func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi/(2*{:})))".format(str(i), str(L))
                    b_coeff = mi.MySimp_tol(func_sin,L,-L,n_max,d)[0]
                    bn.append(b_coeff/(2*L))

            else:
                for i in range(1, N+1):
                    func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi/{:}))".format(str(i), str(L))
                    a_coeff = mi.MySimp_tol(func_cos,L,-L,n_max,d)[0]
                    an.append(a_coeff/L)

                    func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi/{:}))".format(str(i), str(L))
                    b_coeff = mi.MySimp_tol(func_sin,L,-L,n_max,d)[0]
                    bn.append(b_coeff/L)

            an = np.array(an)
            bn = np.array(bn)

    elif method == "gauss" or method == "Gauss":
        if ftype == 0:                          # even function
            a0 = mi.MyLegQuadrature_tol(0,L,expr,2,d,n_max)[0]/(L)

            for i in range(1,N+1):
                bn.append(0)
                func = "(" + expr + ")" + "*" + "cos({:}*(x*pi/{:}))".format(str(i), str(L))
                a_coeff = mi.MyLegQuadrature_tol(0,L,expr,2,d,n_max)[0]
                an.append(2*a_coeff/L)

            if (pw == True):
                an = np.array(an)/2        # /2 because we require int-0 to L and not int-(-L) to L.

            an = np.array(an)
            bn = np.array(bn)

        elif ftype == 1:                            # odd function
```

```python

        for i in range(1, N+1):
            an.append(0)
            func = "(" + expr + ")" + "*" + "sin({:}*(x*pi/{:}))".format(str(i), str(L))
            b_coeff = mi.MyLegQuadrature_tol(0,L,expr,2,d,n_max)[0]
            bn.append(2*b_coeff/L)

        if (pw == True):
            a0 = mi.MyLegQuadrature_tol(0,L,expr,2,d,n_max)[0]/(2*L) # a0/2
            bn = np.array(bn)/2      # /2 because we require int-0 to L and not int-(-L) to L.

        an = np.array(an)
        bn = np.array(bn)

    elif ftype == -1:    # Neither Odd nor Even Function
        a0 = mi.MyLegQuadrature_tol(-L,L,expr,2,d,n_max)[0]/(2*L)    # a0/2

        if (pw == True):
            a0 = a0/2

            for i in range(1, N+1):
                func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi/(2*{:})))".format(str(i), str(L))
                a_coeff = mi.MyLegQuadrature_tol(-L,L,expr,2,d,n_max)[0]
                an.append(a_coeff/(2*L))

                func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi/(2*{:})))".format(str(i), str(L))
                b_coeff = mi.MyLegQuadrature_tol(-L,L,expr,2,d,n_max)[0]
                bn.append(b_coeff/(2*L))

        else:
            for i in range(1, N+1):
                func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi/{:}))".format(str(i), str(L))
                a_coeff = mi.MyLegQuadrature_tol(-L,L,expr,2,d,n_max)
```

```
[0]
                    an.append(a_coeff/L)

                    func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi/{:}))".format(str(i), str(L))
                    b_coeff = mi.MyLegQuadrature_tol(-L,L,expr,2,d,n_max)[0]
                    bn.append(b_coeff/L)

            an = np.array(an)
            bn = np.array(bn)

    return a0,an,bn

if __name__ == "__main__":
    print("\nName: Sarthak Jain\tRoll No. : 2020PHY1201\nPartner: Swarnim Gupta\tRoll No.: 2020PHY1014\nPartner: Ishmeet Singh\tRoll No.: 2020PHY1221\n")

    n = [1, 2, 5, 10, 20]
    xx = np.linspace(-2.5, 2.5, 50)
    x = var("x")

    # # Function 1

    f1 = np.piecewise(xx, [ ((-3<xx) & (xx<-2)), ((-2<xx) & (xx<-1)), ((-1<xx) & (xx<0)), ((0<xx) & (xx<1)), ((1<xx) & (xx<2)), ((2<xx) & (xx<3))], [0, 1, 0, 1, 0, 1])
    dataf1 = []
    outf1 = []
    errf1 = []
    partialsum_f1 = []
    for i in n:
        a0, an, bn = FourierCoeff(expr = "1", N = i, L = 1,d = 4, method = 'trapezoidal', ftype = 1, pw = True)
        partialsumf1 = a0
        for j in range(len(an)):
            partialsumf1 += bn[j]*sin(int(j+1)*np.pi*x) + an[j]*cos(int(j+1)*np.pi*x)
        partialsum_f1.append(lambdify(x, partialsumf1))

    plt.figure("Piecewise Function 1")
```

```
    plt.plot(xx, f1, label = "Function 1")
    for (p, q) in zip(partialsum_f1, n):
        plt.plot(xx, p(xx), marker = '.', linewidth = 0.5, label = "N = %d"%q)
        dataf1.append(p(xx))
        outf1.append(p(-0.5))
        outf1.append(p(0))
        outf1.append(p(0.5))
        errf1.append(np.abs(p(-0.5) - np.zeros(shape = [1, 50])))
        errf1.append(np.abs(p(0) - 0.5*np.ones(shape = [1, 50])))
        errf1.append(np.abs(p(0.5) - np.ones(shape = [1, 50])))


    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.title("Swarnim Gupta , Sarthak Jain , Ishmeet Singh\nFourier Approximation of Piecewise Function (1)")
    plt.legend()
    data_f1 = np.column_stack((xx, *dataf1))
    np.savetxt(r'F:\Ishu\Fourier Series\Function1.txt', data_f1, header = "x, 1, 2, 5, 10, 20")
    data_out_f1 = np.column_stack((*outf1, *errf1))
    np.savetxt(r'F:\Ishu\Fourier Series\Output_Function_1.txt', data_out_f1, header = "n=1 fr(-0.5), fr(0), fr(0.5), n=2 fr(-0.5), fr(0), fr(0.5), n=5 fr(-0.5), fr(0), fr(0.5), n=10 fr(-0.5), fr(0), fr(0.5), n=20 fr(-0.5), fr(0), fr(0.5), n=1 err(-0.5), err(0), err(0.5), n=2 err(-0.5), err(0), err(0.5), n=5 err(-0.5), err(0), err(0.5), n=10 err(-0.5), err(0), err(0.5), n=20 err(-0.5), err(0), err(0.5)")

    ## Fuction 2

    f2 = np.piecewise(xx, [ ((-3<xx) & (xx<-2.5)), ((-2.5<xx) & (xx<-1.5)), ((-1.5<xx) & (xx<-1)), ((-1<xx) & (xx<-0.5)), ((-0.5<xx) & (xx<0.5)), ((0.5<xx) & (xx<1)), ((1<xx) & (xx<1.5)), ((1.5<xx) & (xx<2.5)), ((2.5<xx) & (xx<3))], [0, 1, 0, 0, 1, 0, 0, 1, 0])
    dataf2 = []
    outf2 = []
    errf2 = []
    partialsum_f2 = []
    for i in n:
        a0, an, bn = FourierCoeff(expr = "1", N = i, L = 0.5,d = 4, method = 'trapezoidal', ftype = -1, pw = True)
```

```python
    partialsumf2 = a0
    for j in range(len(an)):
        partialsumf2 += bn[j]*sin(int(j+1)*np.pi*x) + an[j]*cos(int(j+1)*np.pi*x)
    partialsum_f2.append(lambdify(x, partialsumf2))

plt.figure("Piecewise Function 2")
plt.plot(xx, f2, label = "Function 2")
for (p, q) in zip(partialsum_f2, n):
    plt.plot(xx, p(xx), marker = '.', linewidth = 0.5, label = "N = %d"%q)
    dataf2.append(p(xx))
    outf2.append(p(-0.5))
    outf2.append(p(0))
    outf2.append(p(0.5))
    errf2.append(np.abs(p(-0.5) - np.zeros(shape = [1, 50])))
    errf2.append(np.abs(p(0) - 0.5*np.ones(shape = [1, 50])))
    errf2.append(np.abs(p(0.5) - np.ones(shape = [1, 50])))

plt.xlabel("x")
plt.ylabel("f(x)")
plt.title("Swarnim Gupta , Sarthak Jain , Ishmeet Singh\nFourier Approximation of Piecewise Function (2)")
plt.legend(loc = "lower right")
data_f2 = np.column_stack((xx, *dataf2))
np.savetxt(r'F:\Ishu\Fourier Series\Function2.txt', data_f2, header = "x, 1, 2, 5, 10, 20")
data_out_f2 = np.column_stack((*outf2, *errf2))
np.savetxt(r'F:\Ishu\Fourier Series\Output_Function_2.txt', data_out_f2, header = "n=1 fr(-0.5), fr(0), fr(0.5), n=2 fr(-0.5), fr(0), fr(0.5), n=5 fr(-0.5), fr(0), fr(0.5), n=10 fr(-0.5), fr(0), fr(0.5), n=20 fr(-0.5), fr(0), fr(0.5), n=1 err(-0.5), err(0), err(0.5), n=2 err(-0.5), err(0), err(0.5), n=5 err(-0.5), err(0), err(0.5), n=10 err(-0.5), err(0), err(0.5), n=20 err(-0.5), err(0), err(0.5)")

# # Function 3

f3 = np.piecewise(xx, [ ((-3<xx) & (xx<-2)), ((-2<xx) & (xx<-1)), ((-1<xx) & (xx<0)), ((0<xx) & (xx<1)), ((1<xx) & (xx<2)), ((2<xx) & (xx<3))], [-0.5, 0.5, -0.5, 0.5, -0.5, 0.5])
dataf3 = []
outf3 = []
```

```python
    errf3 = []
    partialsum_f3 = []
    for i in n:
        a0, an, bn = FourierCoeff(expr = "0.5", N = i, L = 1,d = 4, method = 'trapezoidal', ftype = 1, pw = False)
        partialsumf3 = a0
        for j in range(len(an)):
            partialsumf3 += bn[j]*sin(int(j+1)*np.pi*x) + an[j]*cos(int(j+1)*np.pi*x)
        partialsum_f3.append(lambdify(x, partialsumf3))

    plt.figure("Piecewise Function 3")
    plt.plot(xx, f3, label = "Function 3")
    for (p, q) in zip(partialsum_f3, n):
        plt.plot(xx, p(xx), marker = '.', linewidth = 0.5, label = "N = %d"%q)
        dataf3.append(p(xx))
        outf3.append(p(-0.5))
        outf3.append(p(0))
        outf3.append(p(0.5))
        errf3.append(np.abs(p(-0.5) - np.zeros(shape = [1, 50])))
        errf3.append(np.abs(p(0) - 0.5*np.ones(shape = [1, 50])))
        errf3.append(np.abs(p(0.5) - np.ones(shape = [1, 50])))

    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.title("Swarnim Gupta , Sarthak Jain , Ishmeet Singh\nFourier Approximation of Piecewise Function (3)")
    plt.legend()
    data_f3 = np.column_stack((xx, *dataf3))
    np.savetxt(r'F:\Ishu\Fourier Series\Function3.txt', data_f3, header = "x, 1, 2, 5, 10, 20")
    data_out_f3 = np.column_stack((*outf3, *errf3))
    np.savetxt(r'F:\Ishu\Fourier Series\Output_Function_3.txt', data_out_f3, header = "n=1 fr(-0.5), fr(0), fr(0.5), n=2 fr(-0.5), fr(0), fr(0.5), n=5 fr(-0.5), fr(0), fr(0.5), n=10 fr(-0.5), fr(0), fr(0.5), n=20 fr(-0.5), fr(0), fr(0.5), n=1 err(-0.5), err(0), err(0.5), n=2 err(-0.5), err(0), err(0.5), n=5 err(-0.5), err(0), err(0.5), n=10 err(-0.5), err(0), err(0.5), n=20 err(-0.5), err(0), err(0.5)")

    plt.show()
```

2.2 A3b-2020PHY1221.py Module

```
from test import FourierCoeff
import matplotlib.pyplot as plt
import numpy as np
from sympy.abc import x
from sympy import sin, cos, lambdify

if __name__ == '__main__':
    print("\nName: Sarthak Jain\tRoll No. : 2020PHY1201\nPartner: Swarnim Gupta\tRoll No.: 2020PHY1014\nPartner: Ishmeet Singh\tRoll No.: 2020PHY1221\n")

    n = [1, 2, 5, 10, 20]
    xx = np.linspace(-np.pi, np.pi, 50)

    # Odd Extension

    f_odd = xx
    outodd = []
    errodd = []
    partialsum_odd = []

    for i in n:
        a0, an, bn = FourierCoeff(expr = "x", N = i, L = np.pi,d = 4, method ='simpson', ftype = 1, pw = False)
        partialsumodd = a0
        for j in range(len(an)):
            partialsumodd += bn[j]*sin(int(j+1)*np.pi*x/np.pi) + an[j]*cos(int(j+1)*np.pi*x/np.pi)
        partialsum_odd.append(lambdify(x, partialsumodd))

    plt.figure("x odd")
    plt.plot(xx, f_odd, label = "Odd Extension: f(x) = x")
    for (p, q) in zip(partialsum_odd, n):
        plt.plot(xx, p(xx), marker = '.', linewidth = 0.5, label = "N = %d "%q)
        outodd.append(p(0))
        outodd.append(p(np.pi/2))
        outodd.append(p(np.pi))
        errodd.append(np.abs(p(0) - np.zeros(shape = [1, 50])))
        errodd.append(np.abs(p(np.pi/2) - np.pi/2*np.ones(shape = [1, 50])
```

```
))
        errodd.append(np.abs(p(np.pi) - np.pi*np.ones(shape = [1, 50])))

 plt.xlabel("x")
 plt.ylabel("f(x)")
 plt.title("Swarnim Gupta , Sarthak Jain , Ishmeet Singh\nFourier Approximation of Odd Extension of f(x) = x")
 plt.legend()
 data_odd = np.column_stack((*outodd, *errodd))
 np.savetxt(r'F:\Ishu\Fourier Series\datax_odd.txt', data_odd, header = "n=1 fr(-0.5), fr(0), fr(0.5), n=2 fr(-0.5), fr(0), fr(0.5), n=5 fr(-0.5), fr(0), fr(0.5), n=10 fr(-0.5), fr(0), fr(0.5), n=20 fr(-0.5), fr(0), fr(0.5), n=1 err(-0.5), err(0), err(0.5), n=2 err(-0.5), err(0), err(0.5), n=5 err(-0.5), err(0), err(0.5), n=10 err(-0.5), err(0), err(0.5), n=20 err(-0.5), err(0), err(0.5)")

 # Even Extension

 f_even = np.abs(xx)
 outeven = []
 erreven = []
 partialsum_even = []

 for i in n:
        a0, an, bn = FourierCoeff(expr = "x", N = i, L = np.pi,d = 4, method ='trapezoidal', ftype = 0, pw = False)
        partialsumeven = a0
        for j in range(len(an)):
            partialsumeven += bn[j]*sin(int(j+1)*np.pi*x/np.pi) + an[j]*cos(int(j+1)*np.pi*x/np.pi)
        partialsum_even.append(lambdify(x, partialsumeven))

 plt.figure("x even")
 plt.plot(xx, f_even, label = "Even Extension: f(x) = x")
 for (p, q) in zip(partialsum_even, n):
        plt.plot(xx, p(xx), marker = '.', linewidth = 0.5, label = "N = %d"%q)
        outeven.append(p(0))
        outeven.append(p(np.pi/2))
        outeven.append(p(np.pi))
        erreven.append(np.abs(p(0) - np.zeros(shape = [1, 50])))
        erreven.append(np.abs(p(np.pi/2) - np.pi/2*np.ones(shape = [1,
```

```
    50])))
        erreven.append(np.abs(p(np.pi) - np.pi*np.ones(shape = [1, 50])))

    plt.xlabel("x")
    plt.ylabel("f(x)")
    plt.title("Swarnim Gupta , Sarthak Jain , Ishmeet Singh\nFourier
    Approximation of Even Extension of f(x) = x")
    plt.legend()
    data_even = np.column_stack((*outodd, *errodd))
    np.savetxt(r'F:\Ishu\Fourier Series\datax_even.txt', data_even, header
     = "n=1 fr(-0.5), fr(0), fr(0.5), n=2 fr(-0.5), fr(0), fr(0.5), n=5 fr
    (-0.5), fr(0), fr(0.5), n=10 fr(-0.5), fr(0), fr(0.5), n=20 fr(-0.5),
    fr(0), fr(0.5), n=1 err(-0.5), err(0), err(0.5), n=2 err(-0.5), err(0),
     err(0.5), n=5 err(-0.5), err(0), err(0.5), n=10 err(-0.5), err(0), err
    (0.5), n=20 err(-0.5), err(0), err(0.5)")

    plt.show()
```

3 Result

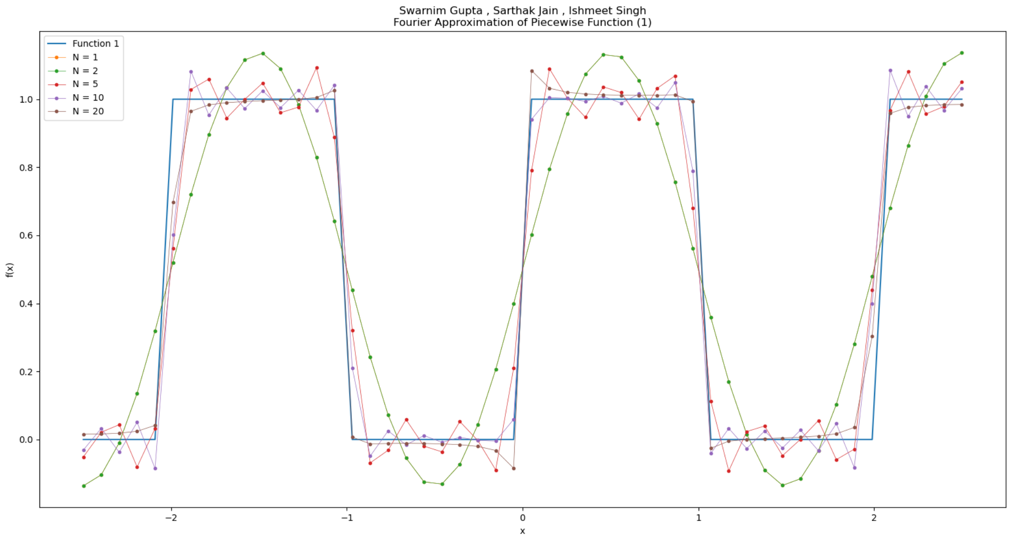

Figure 7

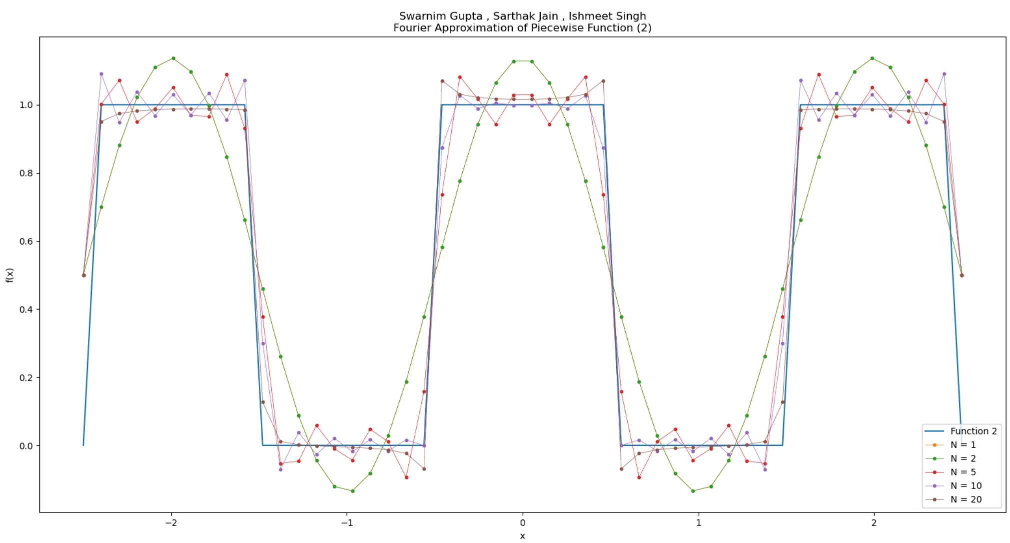

Figure 8

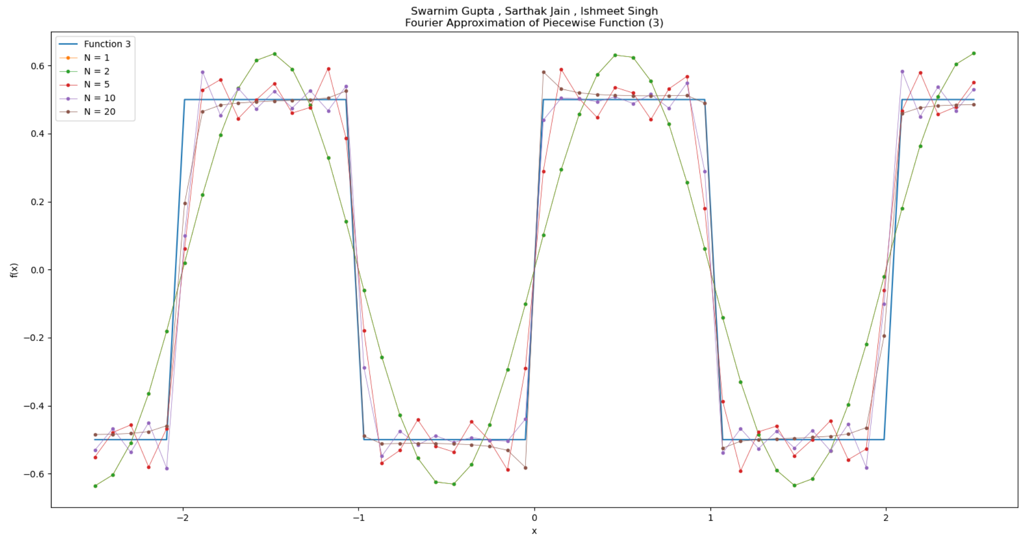

Figure 9

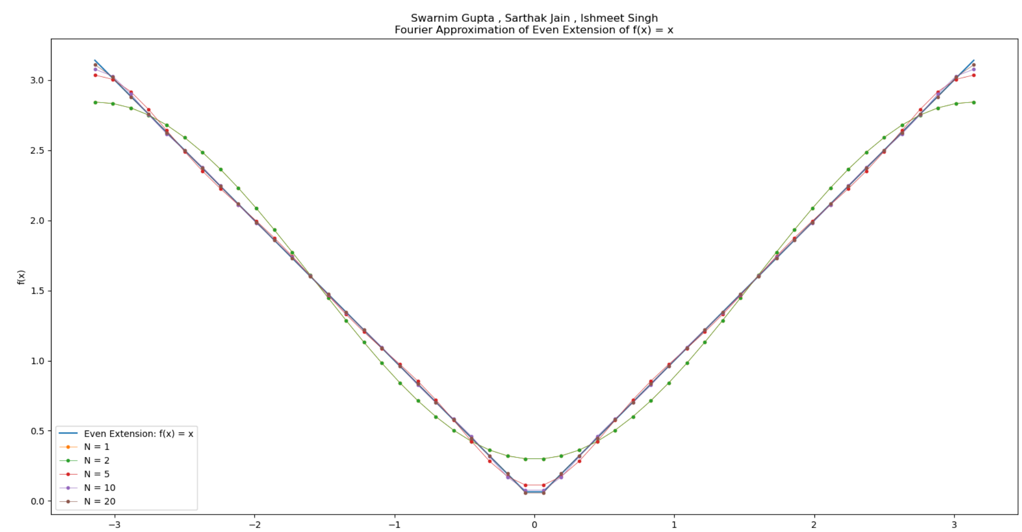

Figure 10

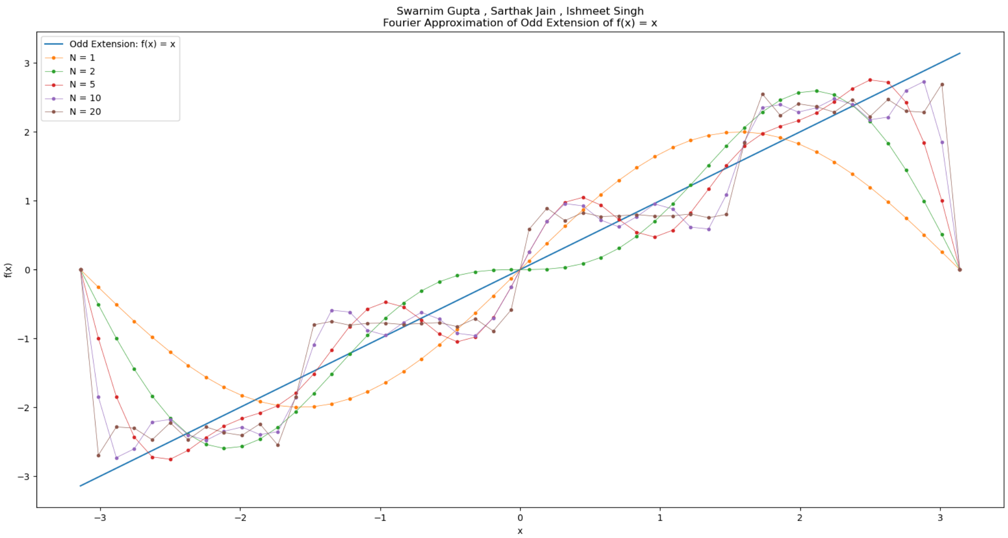

Figure 11

4 Discussion

As we can see from the graphs above the the Gibbs phenomenon gets more prominent at the points of jump / discontinuity as we add more and more terms(i.e N = 1,2,5,10...).
The Gibbs phenomenon is the overshoot that moves closer and closer to the jumps. We can clearly see that the overshoot height goes above 1 or -1 and it does not decrease with more terms of the series