

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 $P \rightarrow \infty$ 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,

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

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

1.2.1 Derivation of Expressions for Fourier Coefficients

$$f(x) = \frac{a_0}{2} + a_1 \cos x + a_2 \cos 2x + \dots + a_n \cos nx + \dots + b_1 \sin x + b_2 \sin 2x + \dots + b_n \sin nx + \dots \quad (1)$$

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

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

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

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

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

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

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

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

$\alpha)$

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

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,

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

$\beta)$

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

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

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,

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

$\gamma)$

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

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

Ans:

$$b - a = 2$$

Calculating Fourier Coefficients,

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

$$\begin{aligned}
a_n &= \frac{2}{b-a} \int_a^b f(x) \cos \left(\frac{2n\pi x}{b-a} \right) dx \\
&= \frac{2}{2} \int_{-1}^1 f(x) \cos \left(\frac{2n\pi x}{2} \right) dx \\
&= \int_{-1}^1 f(x) \cos (n\pi x) dx \\
&= \int_{-1}^0 (-0.5) \cos (n\pi x) dx + \int_0^1 (0.5) \cos (n\pi x) dx \\
&= \frac{-1}{2} \left[\frac{\sin (n\pi x)}{n\pi} \right]_{-1}^0 + \frac{1}{2} \left[\frac{\sin (n\pi x)}{n\pi} \right]_0^1 \\
&= \frac{-1}{2n\pi} [0 + \sin n\pi] + \frac{1}{2n\pi} [\sin n\pi + 0] \\
&= 0 + 0 \\
&\implies \boxed{a_n = 0}
\end{aligned}$$

$$\begin{aligned}
b_n &= \frac{2}{b-a} \int_a^b f(x) \sin \left(\frac{2n\pi x}{b-a} \right) dx \\
&= \frac{2}{2} \int_{-1}^1 f(x) \sin \left(\frac{2n\pi x}{2} \right) dx \\
&= \int_{-1}^1 f(x) \sin n\pi x dx \\
&= \int_{-1}^0 (-0.5) \sin (n\pi x) dx + \int_0^1 (0.5) \sin (n\pi x) dx \\
&= \frac{-1}{2n\pi} [-\cos n\pi x]_{-1}^0 \\
&= \frac{1}{2n\pi} (1 - \cos n\pi) + \frac{1}{2n\pi} [-\cos n\pi x]_0^1 \\
&= \frac{1}{2n\pi} (1 - \cos n\pi) + \frac{1}{2n\pi} (-\cos n\pi + 1) \\
&= 2 \cdot \frac{1}{2n\pi} (1 - \cos n\pi) \\
&= \frac{1}{n\pi} (1 - \cos n\pi) \\
&\implies \boxed{b_n = \frac{1 - (-1)^n}{n\pi}}
\end{aligned}$$

So, the resulting Fourier Series,

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

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.

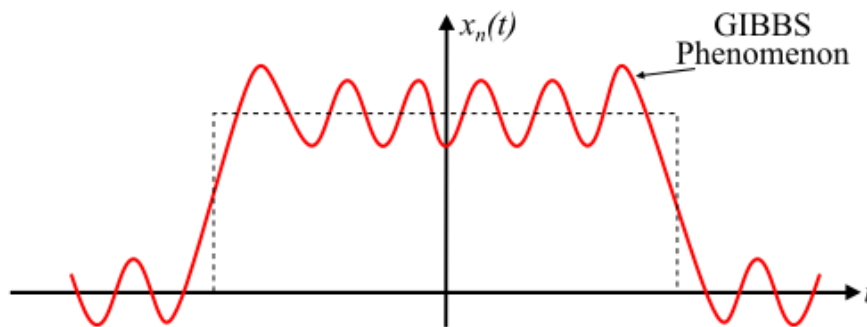


Figure 1: Gibbs Phenomenon

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*.

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

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?

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

for $n=1,2,3,4 \dots$ where

$$a_0 = \frac{2}{L} \int_0^L f(x) dx$$
$$a_n = \frac{2}{L} \int_0^L f(x) \cos\left(\frac{n\pi x}{L}\right) dx$$
$$b_n = 0$$

1.4.3 Odd Function and Half Range Sine Series

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

$$f(x) = \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi x}{L}\right)$$
$$b_n = \frac{2}{L} \int_0^L f(x) \sin\left(\frac{n\pi x}{L}\right) dx$$

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 $x = \pi$:

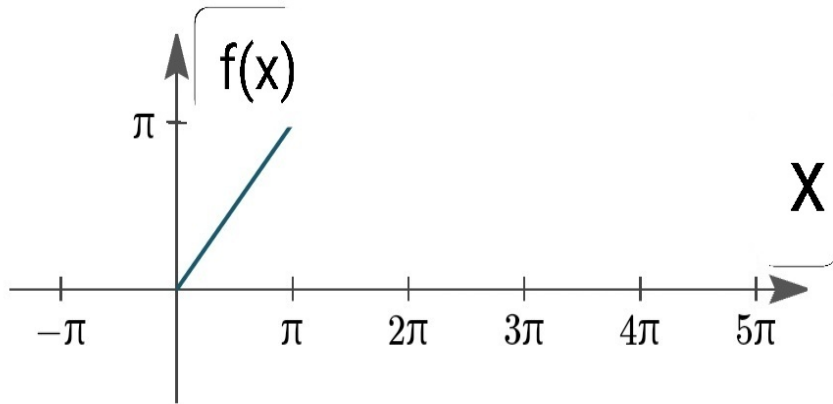


Figure 2: Graph of $f(x)$

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 $t = 0$.

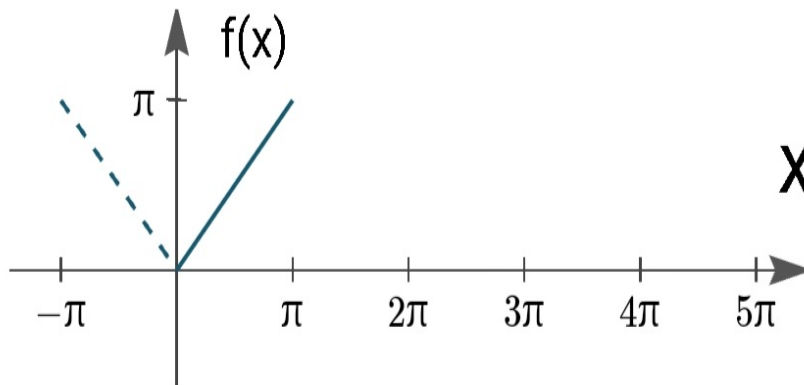


Figure 3: Graph of $f(x)$ showing it as an even function

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

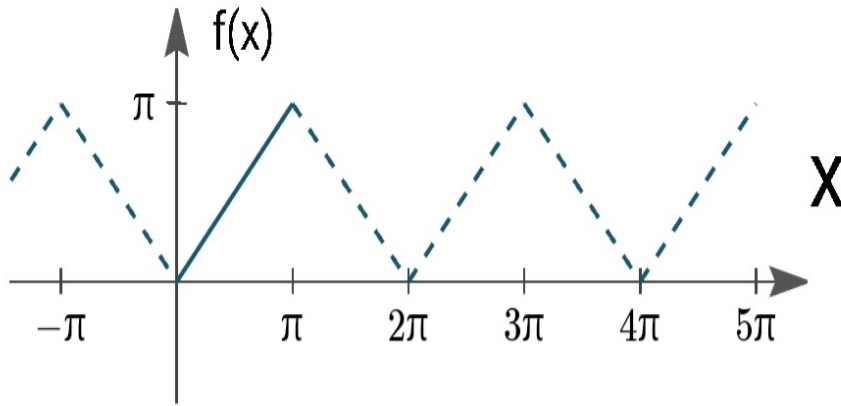


Figure 4: A periodic even function

Half range even extension:

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

$f(x)$ is periodic with period 2π

Since the function is even:

$$b_n = 0$$

Here, $L = \pi$

$$\begin{aligned} a_0 &= \frac{2}{L} \int_0^L f(x) dx \\ &= \frac{2}{\pi} \int_0^\pi x dx \\ &= \frac{2}{\pi} \left[\frac{x^2}{2} \right]_0^\pi \\ &= \frac{2}{\pi} \left[\frac{\pi^2}{2} \right] \\ &= \pi \end{aligned}$$

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

We know,

$$\begin{aligned}
 \int x \cos(nx) dx &= \frac{1}{n^2} [\cos(nx) + nx \sin(nx)] \\
 &= \frac{2}{\pi} \left[\frac{1}{n^2} [\cos(nx) + nx \sin(nx)] \right]_0^\pi \\
 &= \frac{2}{\pi n^2} [(\cos(n\pi) + 0) - (\cos 0 + 0)] \\
 &= \frac{2}{\pi n^2} [\cos(n\pi) - 1] \\
 &= \frac{2}{\pi n^2} [(-1)^n - 1] \\
 &= \frac{-4}{n^2}
 \end{aligned}$$

(When n is odd)

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

Half range odd extension

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

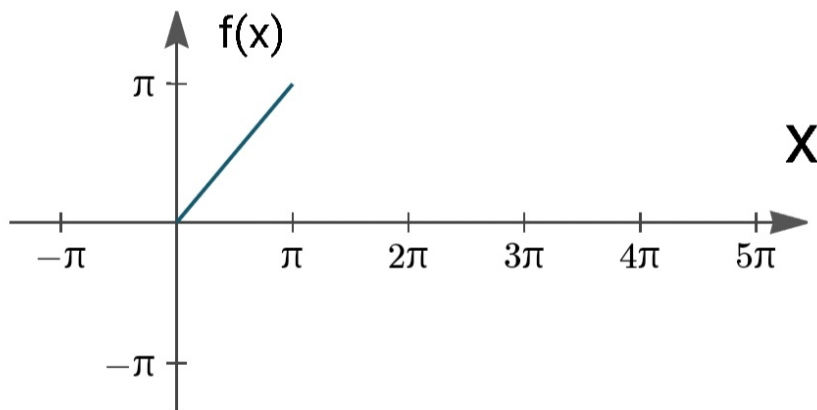


Figure 5: Graph of $f(x)$

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

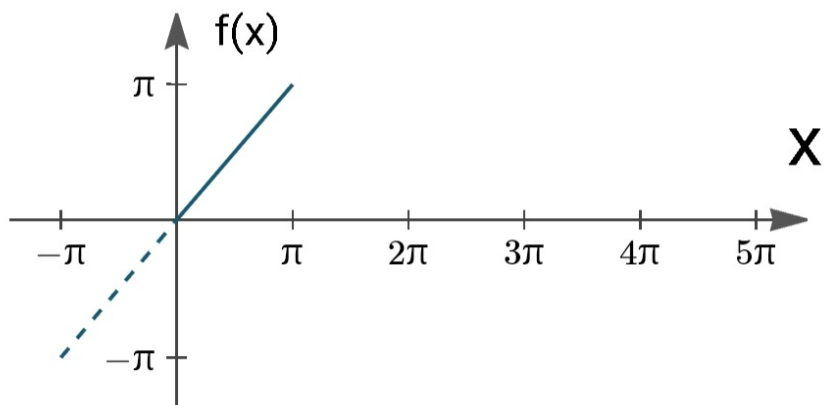


Figure 6: Graph of $f(x)$ showing it as an odd function

2 Programs

2.1 A3a-2020PHY1221.py Module

```
1 import MyIntegration as mi
2 from sympy import *
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 def FourierCoeff(expr, N, L, method, ftype = -1, d = 1, n = 100, pw =
    False, n_max = 1000):
7     a0 = 0
8     an = []
9     bn = []
10
11     if method == "trapezoidal" or method == "Trapezoidal":
12         if ftype == 0:                                # even function
13             a0 = mi.MyTrap_tol(expr, L, 0, n_max, d)[0]/(L)
14
15             for i in range(1, N+1):
16                 bn.append(0)
17                 func = "(" + expr + ")" + "*" + "cos({:}*(x*pi/{:}))".
format(str(i), str(L))
18                 a_coeff = mi.MyTrap_tol(func, L, 0, n_max, d)[0]
19                 an.append(2*a_coeff/L)
20
21             if (pw == True):
22                 an = np.array(an)/2                    # /2 because we require int-0 to
L and not int-(-L) to L.
23
24                 an = np.array(an)
25                 bn = np.array(bn)
26
27             elif ftype == 1:                            # odd function
28
29                 for i in range(1, N+1):
30                     an.append(0)
31                     func = "(" + expr + ")" + "*" + "sin({:}*(x*pi/{:}))".
format(str(i), str(L))
32                     b_coeff = mi.MyTrap_tol(func, L, 0, n_max, d)[0]
33                     bn.append(2*b_coeff/L)
34
```

```

35         if (pw == True):
36             a0 = mi.MyTrap_tol(expr,L,0,n_max,d)[0]/(2*L) # a0/2
37             bn = np.array(bn)/2 # /2 because we require int-0 to
L and not int-(-L) to L.
38
39             an = np.array(an)
40             bn = np.array(bn)
41
42         elif ftype == -1: # Neither Odd nor Even Function
43             a0 = mi.MyTrap_tol(expr,L,-L,n_max,d)[0]/(2*L) # a0/2
44
45             if (pw == True):
46                 a0 = a0/2
47
48             for i in range(1, N+1):
49                 func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi
/(2*{:})))".format(str(i), str(L))
50                 a_coeff = mi.MyTrap_tol(func_cos,L,-L,n_max,d)[0]
51                 an.append(a_coeff/(2*L))
52
53                 func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi
/(2*{:})))".format(str(i), str(L))
54                 b_coeff = mi.MyTrap_tol(func_sin,L,-L,n_max,d)[0]
55                 bn.append(b_coeff/(2*L))
56
57             else:
58                 for i in range(1, N+1):
59                     func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi
/{:})))".format(str(i), str(L))
60                     a_coeff = mi.MyTrap_tol(func_cos,L,-L,n_max,d)[0]
61                     an.append(a_coeff/L)
62
63                     func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi
/{:})))".format(str(i), str(L))
64                     b_coeff = mi.MyTrap_tol(func_sin,L,-L,n_max,d)[0]
65                     bn.append(b_coeff/L)
66
67             an = np.array(an)
68             bn = np.array(bn)
69
70         elif method == "simpson" or method == "Simpson":
71             if ftype == 0: # even function

```

```

72         a0 = mi.MySimp_tol(expr,L,0,n_max,d)[0]/(L)
73
74         for i in range(1,N+1):
75             bn.append(0)
76             func = "(" + expr + ")" + "*" + "cos({:}*(x*pi/{:}))".
format(str(i), str(L))
77             a_coeff = mi.MySimp_tol(func,L,0,n_max,d)[0]
78             an.append(2*a_coeff/L)
79
80             if (pw == True):
81                 an = np.array(an)/2           # /2 because we require int-0 to
L and not int-(-L) to L.
82
83             an = np.array(an)
84             bn = np.array(bn)
85
86         elif ftype == 1:                       # odd function
87
88             for i in range(1, N+1):
89                 an.append(0)
90                 func = "(" + expr + ")" + "*" + "sin({:}*(x*pi/{:}))".
format(str(i), str(L))
91                 b_coeff = mi.MySimp_tol(func,L,0,n_max,d)[0]
92                 bn.append(2*b_coeff/L)
93
94             if (pw == True):
95                 a0 = mi.MySimp_tol(expr,L,0,n_max,d)[0]/(2*L) # a0/2
96                 bn = np.array(bn)/2           # /2 because we require int-0 to
L and not int-(-L) to L.
97
98             an = np.array(an)
99             bn = np.array(bn)
100
101         elif ftype == -1:      # Neither Odd nor Even Function
102             a0 = mi.MySimp_tol(expr,L,-L,n_max,d)[0]/(2*L)      # a0/2
103
104             if (pw == True):
105                 a0 = a0/2
106
107             for i in range(1, N+1):
108                 func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi
/(2*{:})))".format(str(i), str(L))

```

```

109         a_coeff = mi.MySimp_tol(func_cos,L,-L,n_max,d)[0]
110         an.append(a_coeff/(2*L))
111
112         func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi
113 /({:})))".format(str(i), str(L))
114         b_coeff = mi.MySimp_tol(func_sin,L,-L,n_max,d)[0]
115         bn.append(b_coeff/(2*L))
116
117     else:
118         for i in range(1, N+1):
119             func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi
120 /{:})))".format(str(i), str(L))
121             a_coeff = mi.MySimp_tol(func_cos,L,-L,n_max,d)[0]
122             an.append(a_coeff/L)
123
124             func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi
125 /{:})))".format(str(i), str(L))
126             b_coeff = mi.MySimp_tol(func_sin,L,-L,n_max,d)[0]
127             bn.append(b_coeff/L)
128
129     an = np.array(an)
130     bn = np.array(bn)
131
132     elif method == "gauss" or method == "Gauss":
133         if ftype == 0: # even function
134             a0 = mi.MyLegQuadrature_tol(0,L,expr,2,d,n_max)[0]/(L)
135
136             for i in range(1,N+1):
137                 bn.append(0)
138                 func = "(" + expr + ")" + "*" + "cos({:}*(x*pi/{:})))".
139 format(str(i), str(L))
140                 a_coeff = mi.MyLegQuadrature_tol(0,L,expr,2,d,n_max)[0]
141                 an.append(2*a_coeff/L)
142
143                 if (pw == True):
144                     an = np.array(an)/2 # /2 because we require int=0 to
145 L and not int=(-L) to L.
146
147                 an = np.array(an)
148                 bn = np.array(bn)
149
150     elif ftype == 1: # odd function

```

```

146
147         for i in range(1, N+1):
148             an.append(0)
149             func = "(" + expr + ")" + "*" + "sin({:}*(x*pi/{:}))".
format(str(i), str(L))
150             b_coeff = mi.MyLegQuadrature_tol(0,L,expr,2,d,n_max)[0]
151             bn.append(2*b_coeff/L)
152
153         if (pw == True):
154             a0 = mi.MyLegQuadrature_tol(0,L,expr,2,d,n_max)[0]/(2*L) #
a0/2
155             bn = np.array(bn)/2          # /2 because we require int-0 to
L and not int-(-L) to L.
156
157             an = np.array(an)
158             bn = np.array(bn)
159
160         elif ftype == -1:      # Neither Odd nor Even Function
161             a0 = mi.MyLegQuadrature_tol(-L,L,expr,2,d,n_max)[0]/(2*L) #
a0/2
162
163         if (pw == True):
164             a0 = a0/2
165
166         for i in range(1, N+1):
167             func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi
/(2*{:})))".format(str(i), str(L))
168             a_coeff = mi.MyLegQuadrature_tol(-L,L,expr,2,d,n_max)
[0]
169             an.append(a_coeff/(2*L))
170
171             func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi
/(2*{:})))".format(str(i), str(L))
172             b_coeff = mi.MyLegQuadrature_tol(-L,L,expr,2,d,n_max)
[0]
173             bn.append(b_coeff/(2*L))
174
175         else:
176             for i in range(1, N+1):
177                 func_cos = "(" + expr + ")" + "*" + "cos({:}*(x*pi
/{:})))".format(str(i), str(L))
178                 a_coeff = mi.MyLegQuadrature_tol(-L,L,expr,2,d,n_max)

```

```

[0]
179         an.append(a_coeff/L)
180
181         func_sin = "(" + expr + ")" + "*" + "sin({:}*(x*pi
/{:}))".format(str(i), str(L))
182         b_coeff = mi.MyLegQuadrature_tol(-L,L,expr,2,d,n_max)
[0]
183         bn.append(b_coeff/L)
184
185         an = np.array(an)
186         bn = np.array(bn)
187
188         return a0,an,bn
189
190 if __name__ == "__main__":
191     print("\nName: Sarthak Jain\tRoll No. : 2020PHY1201\nPartner: Swarnim
Gupta\tRoll No.: 2020PHY1014\nPartner: Ishmeet Singh\tRoll No.: 2020
PHY1221\n")
192
193     n = [1, 2, 5, 10, 20]
194     xx = np.linspace(-2.5, 2.5, 50)
195     x = var("x")
196
197     # # Function 1
198
199     f1 = np.pieceswise(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])
200     dataf1 = []
201     outf1 = []
202     errf1 = []
203     partialsum_f1 = []
204     for i in n:
205         a0, an, bn = FourierCoeff(expr = "1", N = i, L = 1,d = 4, method =
'trapezoidal', ftype = 1, pw = True)
206         partialsumf1 = a0
207         for j in range(len(an)):
208             partialsumf1 += bn[j]*sin(int(j+1)*np.pi*x) + an[j]*cos(int(j
+1)*np.pi*x)
209             partialsum_f1.append(lambdify(x, partialsumf1))
210
211     plt.figure("Piecewise Function 1")

```

```

212 plt.plot(xx, f1, label = "Function 1")
213 for (p, q) in zip(partialsum_f1, n):
214     plt.plot(xx, p(xx), marker = '.', linewidth = 0.5, label = "N = %d
"%q)
215     dataf1.append(p(xx))
216     outf1.append(p(-0.5))
217     outf1.append(p(0))
218     outf1.append(p(0.5))
219     errf1.append(np.abs(p(-0.5) - np.zeros(shape = [1, 50])))
220     errf1.append(np.abs(p(0) - 0.5*np.ones(shape = [1, 50])))
221     errf1.append(np.abs(p(0.5) - np.ones(shape = [1, 50])))
222
223
224 plt.xlabel("x")
225 plt.ylabel("f(x)")
226 plt.title("Swarnim Gupta , Sarthak Jain , Ishmeet Singh\nFourier
Approximation of Piecewise Function (1)")
227 plt.legend()
228 data_f1 = np.column_stack((xx, *dataf1))
229 np.savetxt(r'F:\Ishu\Fourier Series\Function1.txt', data_f1, header =
"x, 1, 2, 5, 10, 20")
230 data_out_f1 = np.column_stack((*outf1, *errf1))
231 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)")
232
233 ## Fuction 2
234
235 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])
236 dataf2 = []
237 outf2 = []
238 errf2 = []
239 partialsum_f2 = []
240 for i in n:
241     a0, an, bn = FourierCoeff(expr = "1", N = i, L = 0.5, d = 4, method
= 'trapezoidal', ftype = -1, pw = True)

```

```

242     partialsumf2 = a0
243     for j in range(len(an)):
244         partialsumf2 += bn[j]*sin(int(j+1)*np.pi*x) + an[j]*cos(int(j
+1)*np.pi*x)
245     partialsum_f2.append(lambdify(x, partialsumf2))
246
247     plt.figure("Piecewise Function 2")
248     plt.plot(xx, f2, label = "Function 2")
249     for (p, q) in zip(partialsum_f2, n):
250         plt.plot(xx, p(xx), marker = '.', linewidth = 0.5, label = "N = %d
"%q)
251         dataf2.append(p(xx))
252         outf2.append(p(-0.5))
253         outf2.append(p(0))
254         outf2.append(p(0.5))
255         errf2.append(np.abs(p(-0.5) - np.zeros(shape = [1, 50])))
256         errf2.append(np.abs(p(0) - 0.5*np.ones(shape = [1, 50])))
257         errf2.append(np.abs(p(0.5) - np.ones(shape = [1, 50])))
258
259     plt.xlabel("x")
260     plt.ylabel("f(x)")
261     plt.title("Swarnim Gupta , Sarthak Jain , Ishmeet Singh\nFourier
Approximation of Piecewise Function (2)")
262     plt.legend(loc = "lower right")
263     data_f2 = np.column_stack((xx, *dataf2))
264     np.savetxt(r'F:\Ishu\Fourier Series\Function2.txt', data_f2, header =
"x, 1, 2, 5, 10, 20")
265     data_out_f2 = np.column_stack((*outf2, *errf2))
266     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)")
267
268     # # Function 3
269
270     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])
271     dataf3 = []
272     outf3 = []

```



```

273     errf3 = []
274     partialsum_f3 = []
275     for i in n:
276         a0, an, bn = FourierCoeff(expr = "0.5", N = i, L = 1, d = 4, method
277         = 'trapezoidal', ftype = 1, pw = False)
278         partialsumf3 = a0
279         for j in range(len(an)):
280             partialsumf3 += bn[j]*sin(int(j+1)*np.pi*x) + an[j]*cos(int(j
281             +1)*np.pi*x)
282             partialsum_f3.append(lambdify(x, partialsumf3))
283
284     plt.figure("Piecewise Function 3")
285     plt.plot(xx, f3, label = "Function 3")
286     for (p, q) in zip(partialsum_f3, n):
287         plt.plot(xx, p(xx), marker = '.', linewidth = 0.5, label = "N = %d
288         "%q)
289         dataf3.append(p(xx))
290         outf3.append(p(-0.5))
291         outf3.append(p(0))
292         outf3.append(p(0.5))
293         errf3.append(np.abs(p(-0.5) - np.zeros(shape = [1, 50])))
294         errf3.append(np.abs(p(0) - 0.5*np.ones(shape = [1, 50])))
295         errf3.append(np.abs(p(0.5) - np.ones(shape = [1, 50])))
296
297     plt.xlabel("x")
298     plt.ylabel("f(x)")
299     plt.title("Swarnim Gupta , Sarthak Jain , Ishmeet Singh\nFourier
300     Approximation of Piecewise Function (3)")
301     plt.legend()
302     data_f3 = np.column_stack((xx, *dataf3))
303     np.savetxt(r'F:\Ishu\Fourier Series\Function3.txt', data_f3, header =
304     "x, 1, 2, 5, 10, 20")
305     data_out_f3 = np.column_stack((*outf3, *errf3))
306     np.savetxt(r'F:\Ishu\Fourier Series\Output_Function_3.txt',
307     data_out_f3, header = "n=1 fr(-0.5), fr(0), fr(0.5), n=2 fr(-0.5), fr
308     (0), fr(0.5), n=5 fr(-0.5), fr(0), fr(0.5), n=10 fr(-0.5), fr(0), fr
309     (0.5), n=20 fr(-0.5), fr(0), fr(0.5), n=1 err(-0.5), err(0), err(0.5),
310     n=2 err(-0.5), err(0), err(0.5), n=5 err(-0.5), err(0), err(0.5), n=10
311     err(-0.5), err(0), err(0.5), n=20 err(-0.5), err(0), err(0.5)")
312
313     plt.show()

```

2.2 A3b-2020PHY1221.py Module

```
1 from test import FourierCoeff
2 import matplotlib.pyplot as plt
3 import numpy as np
4 from sympy.abc import x
5 from sympy import sin, cos, lambdify
6
7 if __name__ == '__main__':
8     print("\nName: Sarthak Jain\tRoll No. : 2020PHY1201\nPartner: Swarnim
9     Gupta\tRoll No.: 2020PHY1014\nPartner: Ishmeet Singh\tRoll No.: 2020
10    PHY1221\n")
11
12     n = [1, 2, 5, 10, 20]
13     xx = np.linspace(-np.pi, np.pi, 50)
14
15     # Odd Extension
16
17     f_odd = xx
18     outodd = []
19     errodd = []
20     partialsum_odd = []
21
22     for i in n:
23         a0, an, bn = FourierCoeff(expr = "x", N = i, L = np.pi, d = 4,
24         method = 'simpson', ftype = 1, pw = False)
25         partialsumodd = a0
26         for j in range(len(an)):
27             partialsumodd += bn[j]*sin(int(j+1)*np.pi*x/np.pi) + an[j]*cos
28             (int(j+1)*np.pi*x/np.pi)
29             partialsum_odd.append(lambdify(x, partialsumodd))
30
31     plt.figure("x odd")
32     plt.plot(xx, f_odd, label = "Odd Extension: f(x) = x")
33     for (p, q) in zip(partialsum_odd, n):
34         plt.plot(xx, p(xx), marker = '.', linewidth = 0.5, label = "N = %d
35         "%q)
36         outodd.append(p(0))
37         outodd.append(p(np.pi/2))
38         outodd.append(p(np.pi))
39         errodd.append(np.abs(p(0) - np.zeros(shape = [1, 50])))
40         errodd.append(np.abs(p(np.pi/2) - np.pi/2*np.ones(shape = [1, 50]))
```

```

))
36     errodd.append(np.abs(p(np.pi) - np.pi*np.ones(shape = [1, 50])))
37
38     plt.xlabel("x")
39     plt.ylabel("f(x)")
40     plt.title("Swarnim Gupta , Sarthak Jain , Ishmeet Singh\nFourier
Approximation of Odd Extension of f(x) = x")
41     plt.legend()
42     data_odd = np.column_stack((*outodd, *errodd))
43     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)")
44
45     # Even Extension
46
47     f_even = np.abs(xx)
48     outeven = []
49     erreven = []
50     partialsum_even = []
51
52     for i in n:
53         a0, an, bn = FourierCoeff(expr = "x", N = i, L = np.pi, d = 4,
method = 'trapezoidal', ftype = 0, pw = False)
54         partialsumeven = a0
55         for j in range(len(an)):
56             partialsumeven += bn[j]*sin(int(j+1)*np.pi*x/np.pi) + an[j]*
cos(int(j+1)*np.pi*x/np.pi)
57         partialsum_even.append(lambdify(x, partialsumeven))
58
59     plt.figure("x even")
60     plt.plot(xx, f_even, label = "Even Extension: f(x) = x")
61     for (p, q) in zip(partialsum_even, n):
62         plt.plot(xx, p(xx), marker = '.', linewidth = 0.5, label = "N = %d
"%q)
63         outeven.append(p(0))
64         outeven.append(p(np.pi/2))
65         outeven.append(p(np.pi))
66         erreven.append(np.abs(p(0) - np.zeros(shape = [1, 50])))
67         erreven.append(np.abs(p(np.pi/2) - np.pi/2*np.ones(shape = [1,

```

```

50]))))
68     erreven.append(np.abs(p(np.pi) - np.pi*np.ones(shape = [1, 50])))
69
70     plt.xlabel("x")
71     plt.ylabel("f(x)")
72     plt.title("Swarnim Gupta , Sarthak Jain , Ishmeet Singh\nFourier
Approximation of Even Extension of f(x) = x")
73     plt.legend()
74     data_even = np.column_stack((*outodd, *errodd))
75     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)")
76
77     plt.show()

```

3 Result

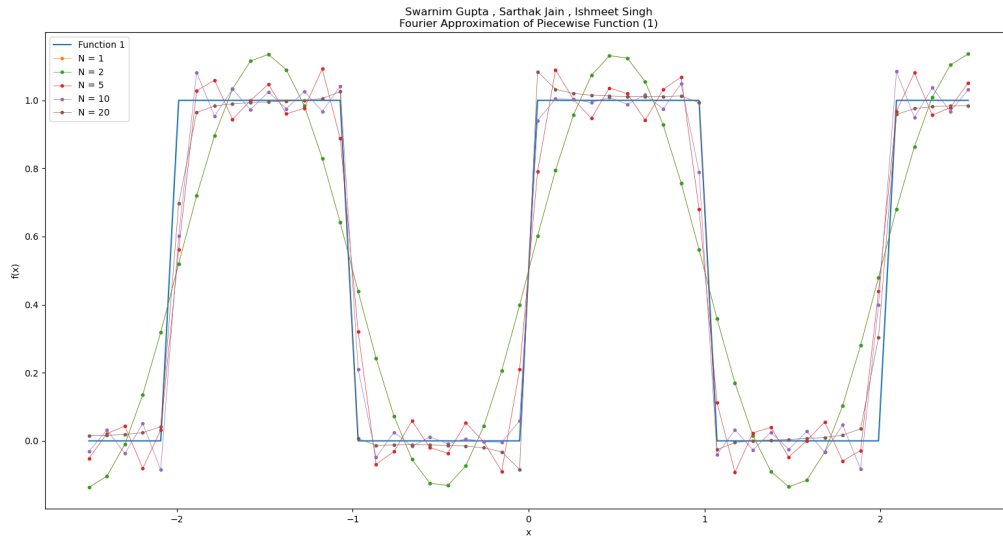


Figure 7

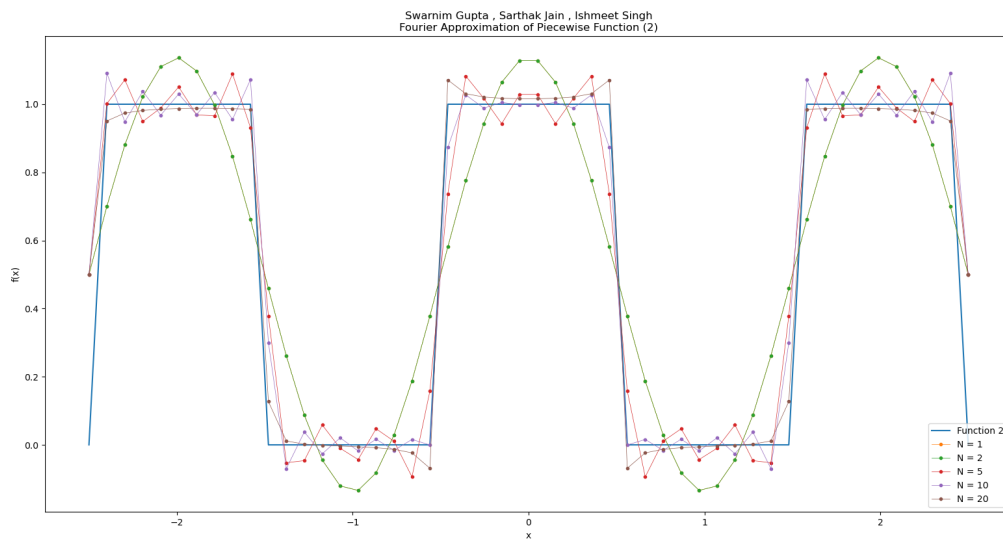


Figure 8

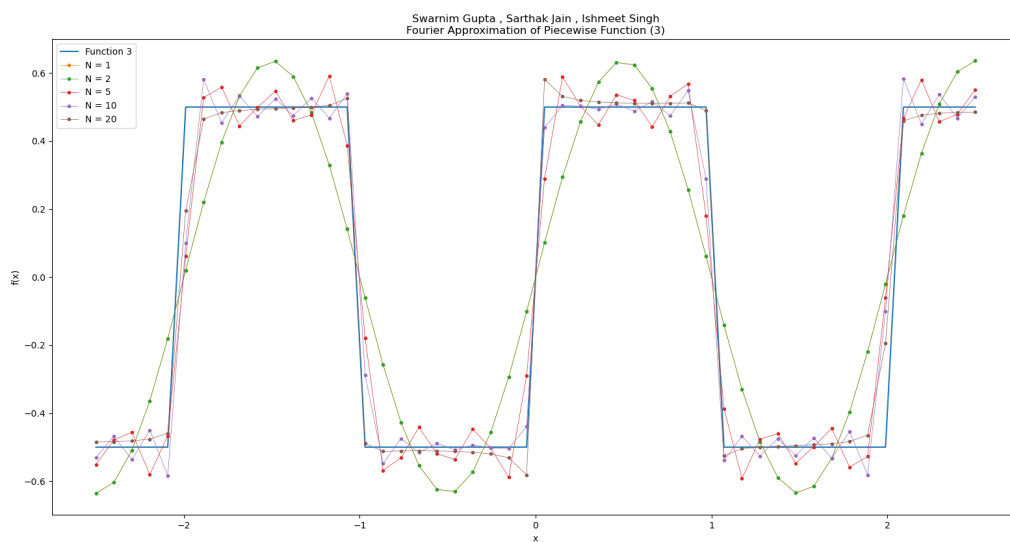


Figure 9

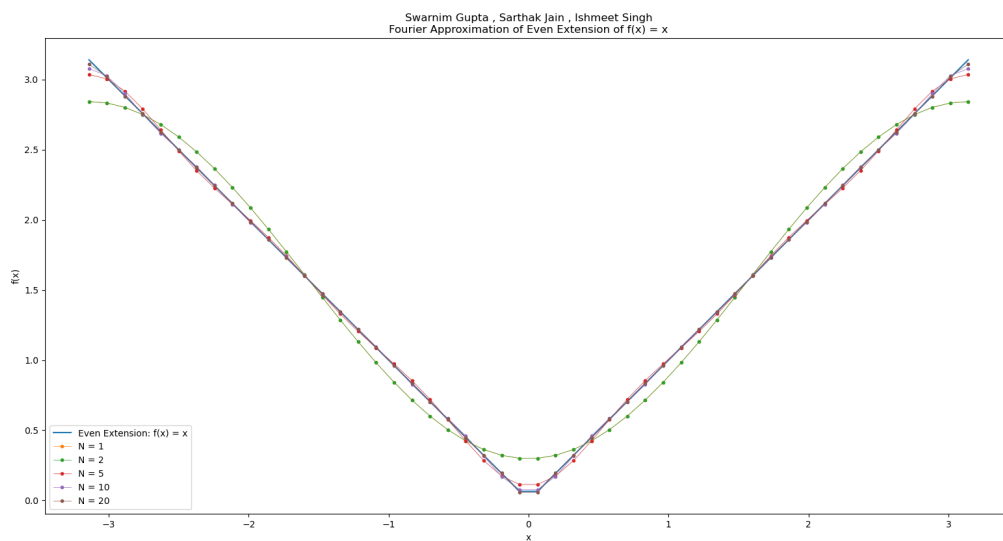


Figure 10

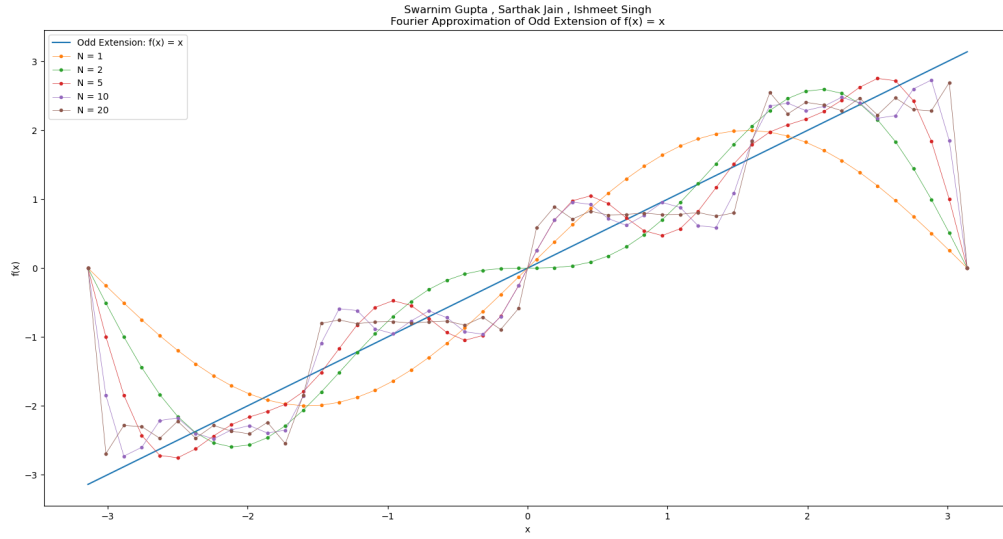


Figure 11

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 $N = 1, 2, 5, 10 \dots$).

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