

FOURIER SERIES

Pawanpreet Kaur
(2020PHY1092)
(20068567038)

Monu Chaurasiya
(2020PHY1102)
(20068567035)

Prateek Bhardwaj
(2020PHY1110)
(20068567042)

S.G.T.B. Khalsa College, University of Delhi, Delhi-110007, India.

Practical Report Submitted to

Dr. Mamta

"32221401 - Mathematical Physics III"

Table of Contents

1	Theory	1
1.1	Dirichlet Conditions	1
1.2	Fourier Representation of a Periodic Function	1
1.2.1	Derivation of Expressions for Fourier Coefficients	2
1.2.2	If the function $f(x)$ is an even or odd function	5
1.2.3	Fourier Series Representation	5
1.3	Gibbs Phenomenon	10
1.4	Half Range Expansion	11
1.4.1	Even Function and Half Range Cosine Series	11
1.4.2	Odd Function and Half Range Sine Series	11
1.4.3	$f(x) = x, 0 < x < \pi$	11
2	RESULTS AND DISCUSSION	21
A	Programs	25

1 Theory

1.1 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.

The conditions are:

- $f(x)$ should be single valued and bounded.
- There should be finite number of minima or maxima in given interval.
- $f(x)$ should be piecewise continuous with finite number of discontinuity in the given interval but the amount of discontinuity should be finite

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,

$$a_0 = \frac{2}{b-a} \int_a^b f(x) dx$$

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

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

1.2.1 Derivation of Expressions for Fourier Coefficients

(i) a_0

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

Taking $L=\pi$,

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx)] + \sum_{n=1}^{\infty} [b_n \sin(nx)]$$

$$\int_{-\pi}^{\pi} f(x) dx = \int_{-\pi}^{\pi} \frac{a_0}{2} dx + \sum_{n=1}^{\infty} \int_{-\pi}^{\pi} [a_n \cos(nx)] dx + \sum_{n=1}^{\infty} \int_{-\pi}^{\pi} [b_n \sin(nx)] dx$$

$$\int_{-\pi}^{\pi} f(x) dx = \frac{a_0}{2} [x]_{-\pi}^{\pi} + 0 + 0$$

$$\int_{-\pi}^{\pi} f(x) dx = \frac{a_0}{2} [\pi - (-\pi)]$$

$$\int_{-\pi}^{\pi} f(x) dx = \pi a_0$$

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

or

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

(ii) a_n

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

Taking $L=\pi$,

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx)] + \sum_{n=1}^{\infty} [b_n \sin(nx)]$$

$$\int_{-\pi}^{\pi} f(x) \cos(mx) dx = \frac{a_0}{2} \int_{-\pi}^{\pi} \cos(mx) dx + \sum_{n=1}^{\infty} a_n \int_{-\pi}^{\pi} [\cos(nx) \cos(mx)] dx + \sum_{n=1}^{\infty} b_n \int_{-\pi}^{\pi} [\sin(nx) \cos(mx)] dx$$

Taking $m=n$ and using standard integrals,

$$\int_{-\pi}^{\pi} f(x) \cos(nx) dx = \frac{a_0}{2} \left[\frac{\sin nx}{n} \right]_{-\pi}^{\pi} + \sum_{n=1}^{\infty} a_n \pi + \sum_{n=1}^{\infty} b_n (0)$$

$$\int_{-\pi}^{\pi} f(x) \cos(nx) dx = 0 + a_n \pi + 0 \quad n = 1, 2, 3, \dots, \infty$$

$$\int_{-\pi}^{\pi} f(x) \cos(nx) dx = a_n \pi \quad n = 1, 2, 3, \dots, \infty$$

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

or

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

(iii) b_n

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

Taking $L=\pi$,

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} [a_n \cos(nx)] + \sum_{n=1}^{\infty} [b_n \sin(nx)]$$

$$\int_{-\pi}^{\pi} f(x) \sin(mx) dx = \frac{a_0}{2} \int_{-\pi}^{\pi} \sin(mx) dx + \sum_{n=1}^{\infty} a_n \int_{-\pi}^{\pi} [\cos(nx) \sin(mx)] dx + \sum_{n=1}^{\infty} b_n \int_{-\pi}^{\pi} [\sin(nx) \sin(mx)] dx$$

Taking $m=n$ and using standard integrals,

$$\int_{-\pi}^{\pi} f(x) \sin(nx) dx = \frac{a_0}{2} \left[\frac{-\cos nx}{n} \right]_{-\pi}^{\pi} + \sum_{n=1}^{\infty} a_n(0) + \sum_{n=1}^{\infty} b_n \pi$$

$$\int_{-\pi}^{\pi} f(x) \sin(nx) dx = \frac{a_0}{2}(0) + 0 + b_n \pi \quad n = 1, 2, 3, \dots, \infty$$

$$\int_{-\pi}^{\pi} f(x) \sin(nx) dx = b_n \pi \quad n = 1, 2, 3, \dots, \infty$$

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

or

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

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

For even functions:

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

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

$$b_n = 0$$

For odd functions:

$$a_0 = 0$$

$$a_n = 0$$

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

1.2.3 Fourier Series Representation

(i)

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

Solution:

$$b - a = 2$$

We know that,

$$\begin{aligned}
 a_o &= \frac{2}{b-a} \int_a^b f(x) dx \\
 a_n &= \frac{2}{b-a} \int_a^b f(x) \cos\left(\frac{2n\pi x}{b-a}\right) dx \\
 b_n &= \frac{2}{b-a} \int_a^b f(x) \sin\left(\frac{2n\pi x}{b-a}\right) dx
 \end{aligned}$$

$$\begin{aligned}
 a_o &= \frac{2}{2} \int_{-1}^1 f(x) dx \\
 &= \int_{-1}^0 (0) dx + \int_0^1 (1) dx \\
 &= 0 + [x]_0^1 \\
 a_o &= 1
 \end{aligned}$$

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

$$\begin{aligned}
 b_n &= \frac{2}{2} \int_{-1}^1 f(x) \sin\left(\frac{2n\pi x}{2}\right) dx \\
 &= \int_{-1}^1 f(x) \sin(n\pi x) dx
 \end{aligned}$$

$$\begin{aligned}
&= \int_{-1}^0 (0) \sin(n\pi x) dx + \int_0^1 (1) \sin(n\pi x) dx \\
&= \left[\frac{-\cos(n\pi x)}{n\pi} \right]_0^1 \\
&= \frac{1 - \cos(n\pi)}{n\pi} \\
\\
b_n &= \frac{1 - (-1)^n}{n\pi}
\end{aligned}$$

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

(ii)

$$\begin{aligned}
f(x) &= \begin{cases} 0 & \text{if } -1 < x < -0.5 \\ 1 & \text{if } -0.5 < x < 0.5 \\ 0 & \text{if } 0.5 < x < 1 \end{cases} \\
f(x+2) &= f(x)
\end{aligned}$$

Solution:

$$b - a = 2$$

We know that,

$$\begin{aligned}
a_o &= \frac{2}{b-a} \int_a^b f(x) dx \\
a_n &= \frac{2}{b-a} \int_a^b f(x) \cos\left(\frac{2n\pi x}{b-a}\right) dx \\
b_n &= \frac{2}{b-a} \int_a^b f(x) \sin\left(\frac{2n\pi x}{b-a}\right) dx
\end{aligned}$$

$$\begin{aligned}
a_o &= \frac{2}{2} \int_{-1}^1 f(x) dx \\
&= \int_{-1}^{-0.5} f(x) dx + \int_{-0.5}^{0.5} f(x) dx + \int_{0.5}^1 f(x) dx
\end{aligned}$$

$$0 + \int_{-0.5}^{0.5} dx + 0$$

$$[x]_{-0.5}^{0.5}$$

$$a_o = 1$$

$$a_n = \frac{2}{2} \int_{-1}^1 f(x) \cos\left(\frac{2n\pi x}{2}\right) dx$$

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

$$= 0 + \left[\frac{\sin(n\pi x)}{n\pi} \right]_{-0.5}^{0.5} + 0$$

$$a_n = \frac{2}{n\pi} \sin\left(\frac{n\pi}{2}\right)$$

$$b_n = \frac{2}{2} \int_{-1}^1 f(x) \sin\left(\frac{2n\pi x}{2}\right) dx$$

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

As we Know that sin is an odd function, so integration above using property of integration

becomes 0.

$$= 0 + 0 + 0$$

$$b_n = 0$$

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

(iii)

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

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

Solution:

$$b - a = 2$$

We know that,

$$\begin{aligned}a_0 &= \frac{2}{b-a} \int_a^b f(x) dx \\a_n &= \frac{2}{b-a} \int_a^b f(x) \cos\left(\frac{2n\pi x}{b-a}\right) dx \\b_n &= \frac{2}{b-a} \int_a^b f(x) \sin\left(\frac{2n\pi x}{b-a}\right) dx\end{aligned}$$

$$\begin{aligned}a_0 &= \frac{2}{2} \int_{-1}^1 f(x) dx \\&= \int_{-1}^0 (-0.5) dx + \int_0^1 (0.5) dx\end{aligned}$$

$$-\frac{1}{2}[x]_{-1}^0 + \frac{1}{2}[x]_0^1$$

$$a_0 = 0$$

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

$$b_n = \frac{2}{2} \int_{-1}^1 f(x) \sin\left(\frac{2n\pi x}{2}\right)$$

$$\begin{aligned}
&= \int_{-1}^0 (-0.5) \sin(n\pi x) dx + \int_0^1 (0.5) \sin(n\pi x) dx \\
&= \frac{-1}{2n\pi} [-\cos(n\pi x)]_{-1}^0 + \frac{1}{2n\pi} [-\cos(n\pi x)]_0^1 \\
&= \frac{1}{2n\pi} (1 - \cos(n\pi)) + \frac{1}{2n\pi} (1 - \cos(n\pi)) \\
&\quad 2 * \frac{1}{2n\pi} (1 - \cos(n\pi)) \\
&= \frac{1}{n\pi} (1 - \cos(n\pi))
\end{aligned}$$

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

1.3 Gibbs Phenomenon

The Gibbs phenomenon, is the peculiar manner in which the Fourier series of a piecewise continuously differentiable periodic function behaves at a jump discontinuity. The Gibbs phenomenon involves both the fact that Fourier sums overshoot at a jump discontinuity, and that this overshoot does not die out as more terms are added to the sum. For the one-dimensional case, the simplest mathematical illustration of the Gibbs phenomenon is an approximation of a square wave function.

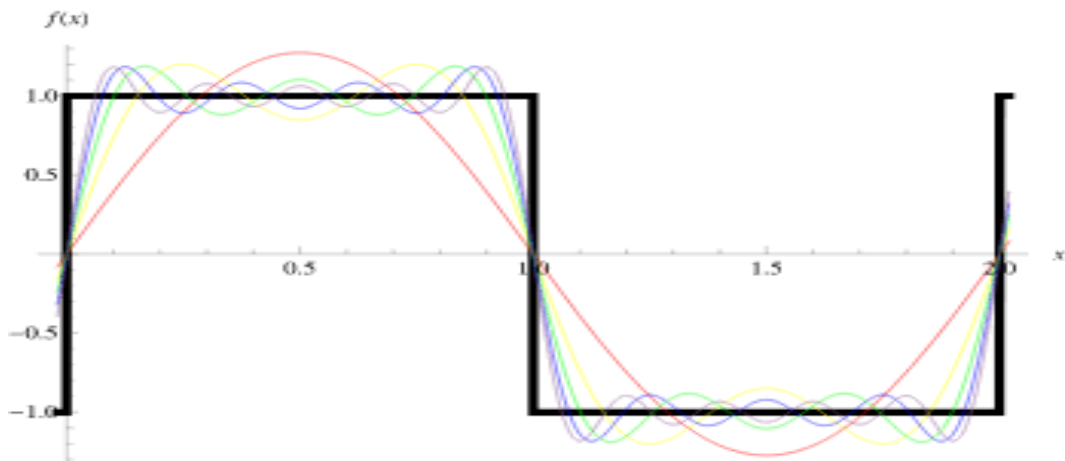


Figure 1: Gibbs Phenomenon

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 Even Function and Half Range Cosine Series

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

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

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

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

$$b_n = 0$$

1.4.2 Odd Function and Half Range Sine Series

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

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

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

1.4.3 $f(x) = x, 0 < x < \pi$

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 $x = \pi$:

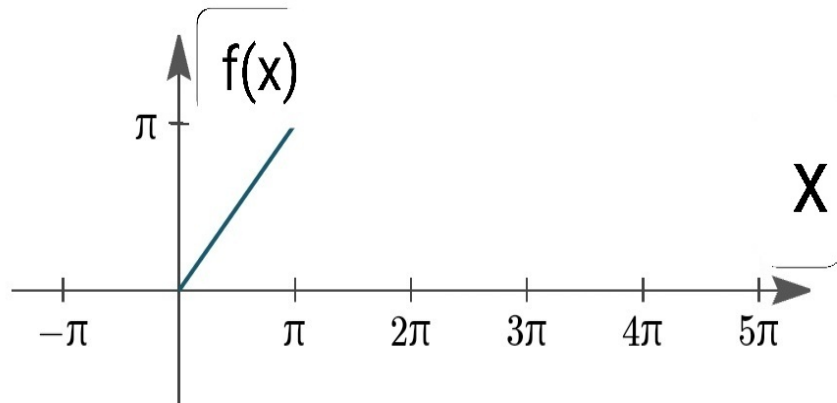


Figure 2: Graph of $f(x)$

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 $t = 0$.

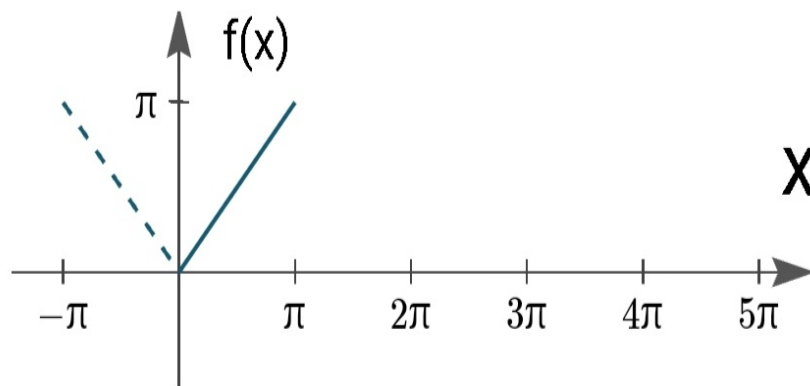


Figure 3: Graph of $f(x)$ showing it as an even function

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

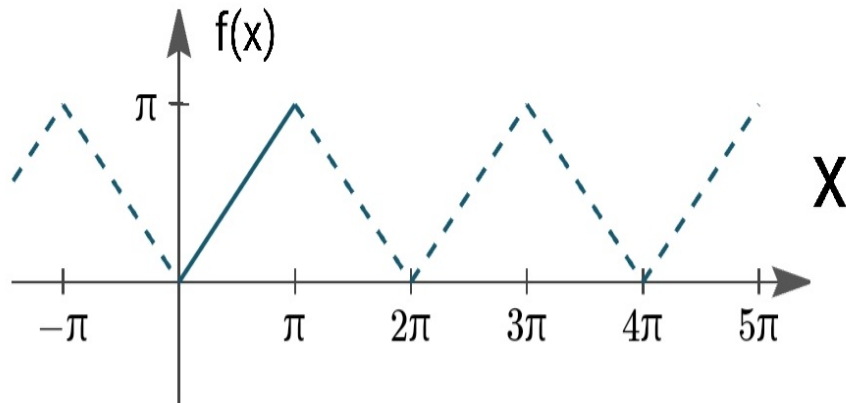


Figure 4: A periodic even function

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, $L = \pi$

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

$$= \frac{2}{\pi} \int_0^{\pi} x dx$$

$$= \frac{2}{\pi} \left[\frac{x^2}{2} \right]_0^{\pi}$$

$$= \frac{2}{\pi} \left[\frac{\pi^2}{2} \right]$$

$$= \pi$$

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

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

We know,

$$\int x \cos(nx) dx = \frac{1}{n^2} [\cos(nx) + nx \sin(nx)]$$

$$= \frac{2}{\pi} \left[\frac{1}{n^2} [\cos(nx) + nx \sin(nx)] \right]_0^\pi$$

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

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

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

$$= \frac{-4}{n^2}$$

(when n is odd)

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

$$= \frac{\pi}{2} - \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{\cos(2n-1)x}{(2n-1)^2}$$

$$f(x) = \frac{\pi}{2} - \frac{4}{\pi} \left(\cos x + \frac{1}{9} \cos 3x + \frac{1}{25} \cos 5x + \dots \right)$$

Half range odd extension

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

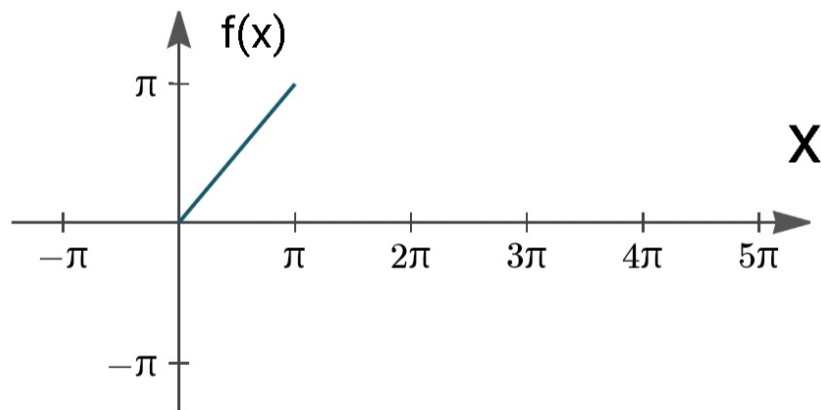


Figure 5: Graph of $f(x)$

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

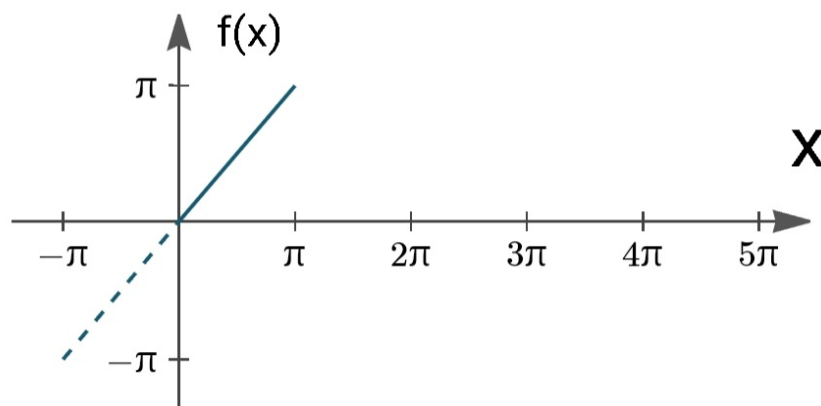


Figure 6: Graph of $f(x)$ showing it as an odd function

The waveform produced is periodic of period 2π outside of this range as shown by the dotted lines.

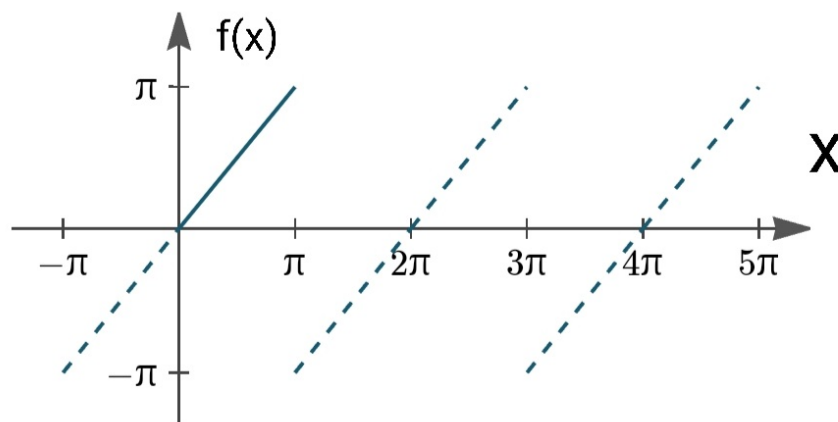


Figure 7: A periodic odd function

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 odd:

$$a_0 = 0$$

$$a_n = 0$$

Here, $L = \pi$

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

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

We know,

$$\int x \sin(nx) dx = \frac{1}{n^2} [\sin(nx) - nx \cos(nx)]$$

$$= \frac{2}{\pi} \left[\frac{1}{n^2} [\sin(nx) - nx \cos(nx)] \right]_0^\pi$$

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

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

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

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

$$= -\frac{2}{\pi} \sum_{n=1}^{\infty} (-1)^n \frac{\sin(nx)}{(n)^2}$$

$$f(x) = -\frac{2}{\pi} \left(-\sin x + \frac{1}{4} \sin 2x - \dots \right)$$

Algorithm 1 Fourier Series

function FOURIERCOEFF($f, PropKey, a, b, L, N, m_k, tole$) $\triangleright f$:function $\triangleright PropKey$: takes the value 0,1,-1 if the function is 'even','odd' or 'neither even nor odd' respectively. $\triangleright a$: lower limit , b : upper limit , $tole$: tolerance, L : half period, N : numbers of terms to be evaluated $\triangleright m_k$: method used for integration takes value 1,2,3 for trapezoidal, simpson and guass-legendre quadrature methods respectively

if PropKey==1:

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

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

if $m_k == 1$:bn=(1/L)*MyTrap(fun,a,b,1,key1=True,N_{max} = 1000,key2 = True,tol = tole)[0]elif $m_k == 2$:bn=(1/L)*MyTrap(fun,a,b,1,key1=True,N_{max} = 1000,key2 = True,tol = tole)[0]elif $m_k == 3$:bn=(1/L)*MyLegQuadrature(fun,a,b,10,1,key=True,tol=tole,m_{max} = 1000)[0]

else:

return "Wrong Method Key entered"

bn_a.append(bn)an_a.append(0) **return** a0, an_a, bn_a \triangleright For even function (that means Propkey=1) we calculate the sum of sine series. \triangleright The b_n is calculated by the suitable integration method entered by the user and append the values in the list \triangleright MyTrap is the integration module which evaluate value of the function using trapezoidal method . \triangleright MySimp is the integration module which evaluate value using simpson method . \triangleright MyLegQuadrature is the integration module which evaluate using guass-legendre quadrature.**def** f1(x): \triangleright For odd function (that means Propkey=0) we calculate the sum of cosine series.if $x \geq -np.pi$ and $x \leq 0$: **return** xelif $x > 0$ and $x \leq np.pi$: **return** x**def** pf1(x):if $x \geq -np.pi$ and $x \leq np.pi$: **return** f1(x) \triangleright The a0 and an is calculated by the suitable integration method entered by the user and append the values in the list.elif $x > np.pi$: $x_{new} = x - (np.pi - (-np.pi))$ **return** pf1(x_{new})elif $x < (-np.pi)$: $x_{new} = x + (np.pi - (-np.pi))$ **return** pf1(x_{new}) \triangleright For neither odd or even function we calculate the sum of cosine and sine series. \triangleright The a0,bn and an is calculated by the suitable integration method and append values in list.

2(b)

I think Gauss-Legendre Quadrature is the most appropriate numerical method for the above functions in previous questions. This method is optimal for all polynomials of degree less or equal to $2n-1$, while the Newton-Cotes (trapezoidal and Simpson) formulas are optimal for all polynomials of degree less than or equal to n .

But for above functions Gauss-Legendre quadrature is better as the integral is very close to true value and the absolute error is very close to zero. And this intuition has verified when we'd checked the value of integral by other numerical method.

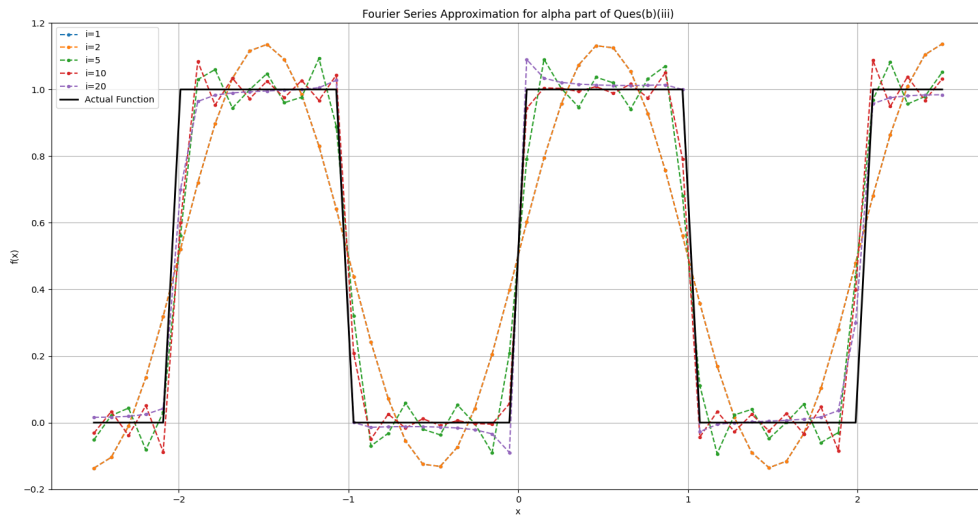


Figure 8: *Fourier Series Approximation for α using Gauss Legendre Quadrature*

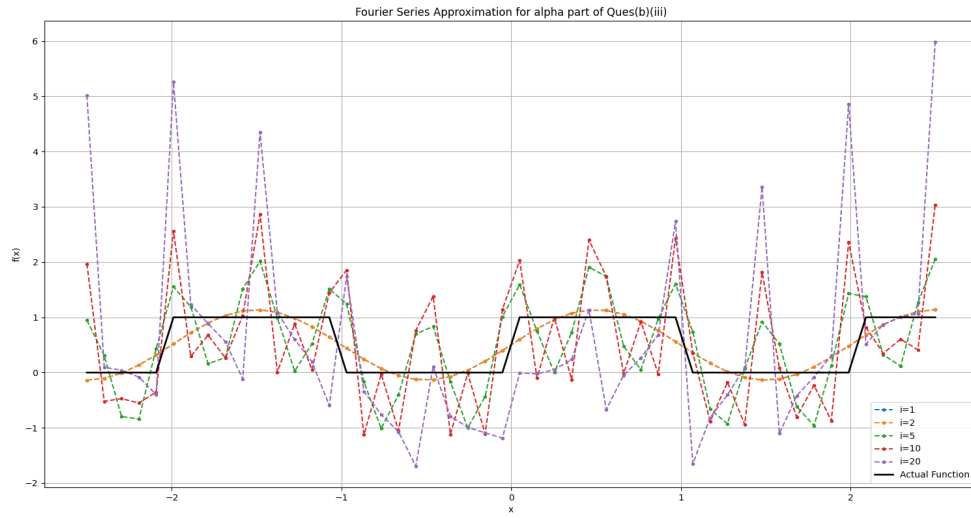


Figure 9: *Fourier Series Approximation for using Simpson Method*

If we had to approximate the output of a full wave rectifier by Fourier series, Gauss Legendre Quadrature would be most appropriate because for finding the output we have to integrate sine terms and Quadrature will serve the purpose better than simpson and trapezoidal method.

2 RESULTS AND DISCUSSION

For

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

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

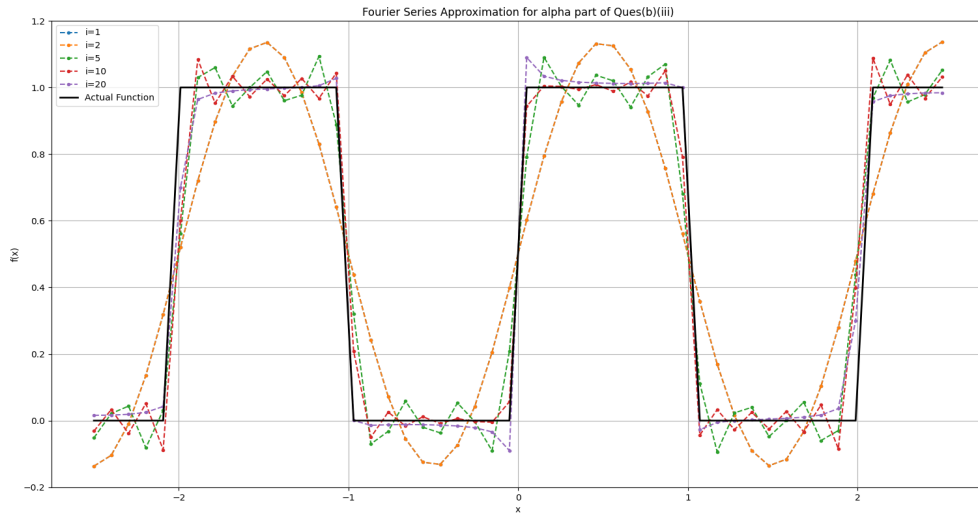


Figure 10: *Fourier Series Approximation for α*

For

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

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

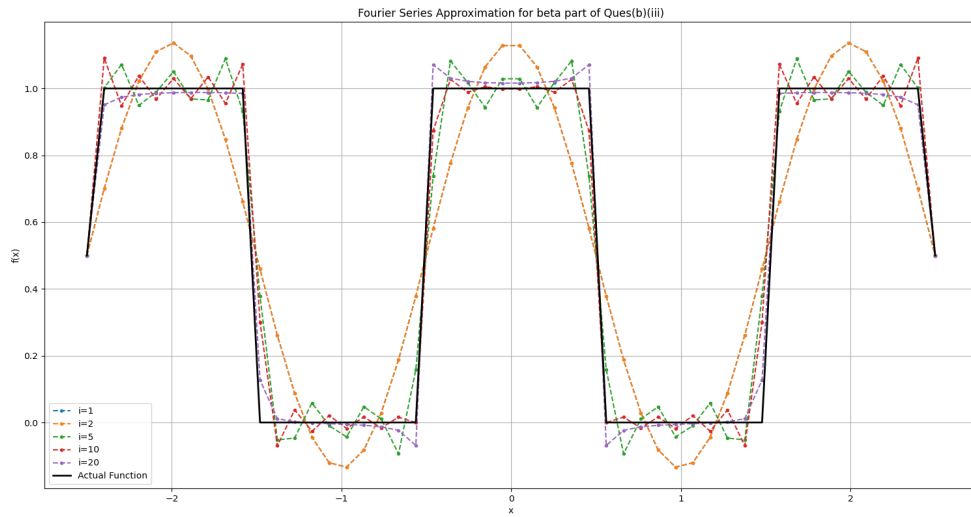


Figure 11: *Fourier Series Approximation for β*

For

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

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

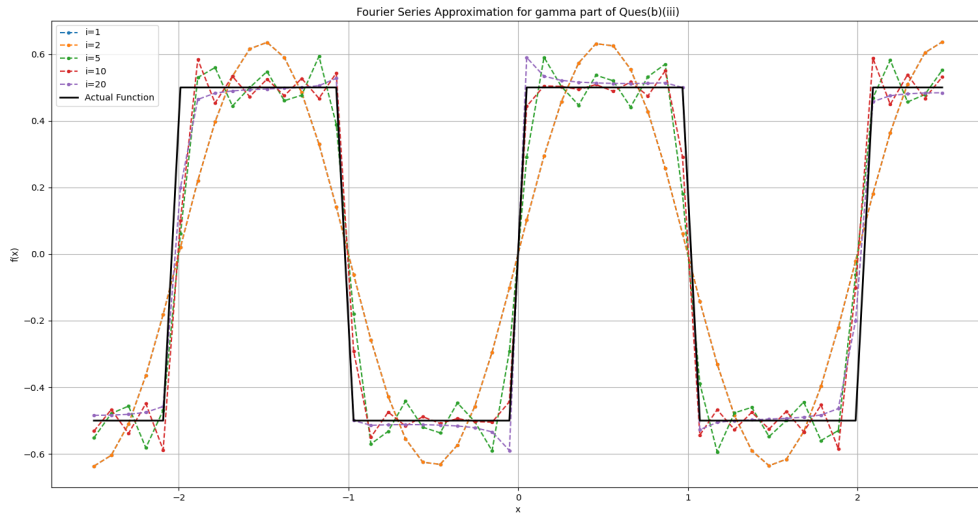


Figure 12: *Fourier Series Approximation for γ*

From the above graphs we can see the partial sums after one term, then two terms, then five terms, then ten terms, and then twenty terms. We can see the Gibbs phenomenon appearing as these "partial sums" include more terms. Away from the jumps, we safely approach $f(x)=1$ or -1 for first two graphs and $f(x)=0.5$ or -0.5 for third graph.

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!. Thus we can say that overshoot is present in all discontinuous functions.

For

$$f(x) = x \quad \text{for } 0 < x < \pi$$

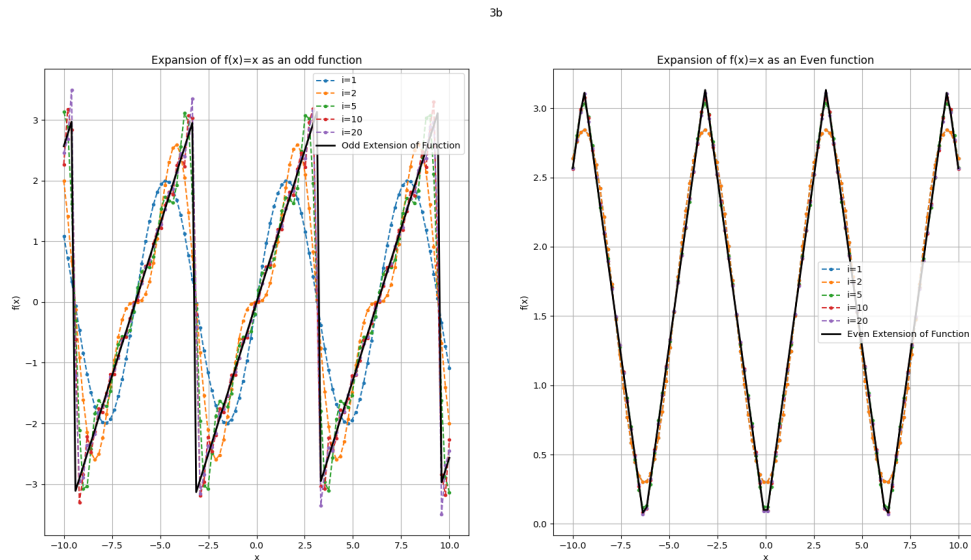


Figure 13: *Half Range Expansion of Fourier series*

Above plot shows the $f(x)$ vs x plot for half range sine and cosine expansions. Series calculated for $N=20$ is very similar to the extended function both as odd as well as even function.

A Programs

```
1 #Name= Monu Chaurasiya
2 #College Roll No. = 2020 PHY1102
3 #University Roll No. = 20068567035
4
5 '''
6 PARTNERS:
7 #Name= Pawanpreet Kaur
8 #College Roll No. = 2020 PHY1092
9 #University Roll No. = 20068567038
10
11 #Name= Prateek Bhardwaj
12 #College Roll No. = 2020 PHY1110
13 #University Roll No. = 20068567042
14 '''
15
16 from MyIntegration import MySimp
17 from MyIntegration import MyTrap
18 from MyIntegration import MyLegQuadrature
19 import pandas as pd
20 import numpy as np
21 import matplotlib.pyplot as plt
22 import math
23 from scipy import integrate
24 from sympy import *
25 from sympy import simplify
26
27 x=symbols('x')
28
29 #Function for Fourier Coefficient
30 def FourierCoeff(f,Prop_Key,a,b,L,N,m_k,tole):
31     an=0;an_a=[]
32     bn=0;bn_a=[]
33     a0=0
34     if Prop_Key==1:
35         for i in range(1,N+1):
36             fun = lambda x: f(x)*np.sin(i*np.pi*x/L)
37             if m_k==1:
38                 bn=(1/L)*MyTrap(fun,a,b,1,key1=True,N_max=1000,key2=
39                 True,tol=tole)[0]
40             elif m_k==2:
41                 bn=(1/L)*MyTrap(fun,a,b,1,key1=True,N_max=1000,key2=
42                 True,tol=tole)[0]
43             elif m_k==3:
44                 bn=(1/L)*MyLegQuadrature(fun,a,b,10,1,key=True,tol=
45                 tole,m_max=1000)[0]
46             else:
47                 return "Wrong Method Key entered"
48             bn_a.append(bn)
49             an_a.append(0)
50     return a0,an_a,bn_a
51
52 elif Prop_Key==0:
```

```

50         if m_k==1:
51             a0=(1/L)*MyTrap(f,a,b,1,key1=True,N_max=1000,key2=True,tol
=0.1e-3)[0]
52         elif m_k==2:
53             a0=(1/L)*MySimp(f,a,b,2,key1=True,N_max=1000,key2=True,tol
=tole)[0]
54         elif m_k==3:
55             a0=(1/L)*MyLegQuadrature(f,a,b,10,1,key=True,tol=tole,
m_max=1000)[0]
56         else:
57             return "Wrong Method Key entered"
58
59         for i in range(1,N+1):
60             fun = lambda x: f(x)*np.cos(i*np.pi*x/L)
61             if m_k==1:
62                 an=(1/L)*MyTrap(fun,a,b,1,key1=True,N_max=1000,key2=
True,tol=tole)[0]
63             elif m_k==2:
64                 an=(1/L)*MySimp(fun,a,b,1,key1=True,N_max=1000,key2=
True,tol=tole)[0]
65             elif m_k==3:
66                 an=(1/L)*MyLegQuadrature(fun,a,b,10,1,key=True,tol=
tole,m_max=1000)[0]
67             else:
68                 return "Wrong Method Key entered"
69             an_a.append(an)
70             bn_a.append(0)
71         return a0,an_a,bn_a
72
73     elif Prop_Key==-1:
74         if m_k==1:
75             a0=(1/L)*MyTrap(f,a,b,1,key1=True,N_max=1000,key2=True,tol
=0.1e-3)[0]
76         elif m_k==2:
77             a0=(1/L)*MySimp(f,a,b,2,key1=True,N_max=1000,key2=True,tol
=tole)[0]
78         elif m_k==3:
79             a0=(1/L)*MyLegQuadrature(f,a,b,10,1,key=True,tol=tole,
m_max=1000)[0]
80         else:
81             return "Wrong Method Key entered"
82
83         for i in range(1,N+1):
84             fun1 = lambda x: f(x)*np.cos(i*np.pi*x/L)
85             fun2 = lambda x: f(x)*np.sin(i*np.pi*x/L)
86             if m_k==1:
87                 an=(1/L)*MyTrap(fun1,a,b,1,key1=True,N_max=1000,key2=
True,tol=tole)[0]
88                 bn=(1/L)*MyTrap(fun2,a,b,1,key1=True,N_max=1000,key2=
True,tol=tole)[0]
89             elif m_k==2:
90                 an=(1/L)*MySimp(fun1,a,b,2,key1=True,N_max=1000,key2=
True,tol=tole)[0]
91                 bn=(1/L)*MySimp(fun2,a,b,2,key1=True,N_max=1000,key2=
True,tol=tole)[0]

```

```

92         elif m_k==3:
93             an=(1/L)*MyLegQuadrature(fun1,a,b,10,1,key=True,tol=
tole,m_max=1000)[0]
94             bn=(1/L)*MyLegQuadrature(fun2,a,b,10,1,key=True,tol=
tole,m_max=1000)[0]
95         else:
96             return "Wrong Method Key entered"
97         an_a.append(an)
98         bn_a.append(bn)
99         return a0,an_a,bn_a
100     else :
101         return "Wrong key entered"
102
103
104 def Task(L,f1,pf1,Prop_Key,m_k,tit,filename1,filename2,filename3,
filename4,filename5):
105     N_a=[1,2,5,10, 20]
106     x_a=np.linspace(-2.5,2.5,50)
107     ex=[]
108     f_v=[]
109     f_=[]
110     a_c=[]
111     b_c=[]
112     E1=[];E2=[];E3=[];E4=[];E5=[]
113     A_X=[-0.5,0,0.5]
114     ex2=[]
115     for x in x_a:
116         ex.append(pf1(x))
117     for x in A_X:
118         ex2.append(pf1(x))
119     for N in N_a:
120         s=FourierCoeff(pf1,Prop_Key=Prop_Key,a=-1*L,b=1*L,L=1,N=N,m_k
=m_k,tol=0.1e-8)
121         a0=s[0]
122         #print(a0)
123         e=s[1]
124         e1=s[2]
125         d_a=[]
126         d_=[]
127         for x in A_X:
128             sin=0
129             cos=0
130
131             for (i,an,bn) in zip(range(1,N+1),e,e1):
132                 sin+=bn*np.sin(i*np.pi*x/L)
133                 cos+=an*np.cos(i*np.pi*x/L)
134             d=a0/2+sin+cos
135             d_.append(d)
136         f_.append(d_)
137         d_=[]
138
139         for x in x_a:
140             sin=0
141             cos=0
142

```

```

143         for (i,an,bn) in zip(range(1,N+1),e,e1):
144             sin+=bn*np.sin(i*np.pi*x/L)
145             cos+=an*np.cos(i*np.pi*x/L)
146             d=a0/2+sin+cos
147             d_a.append(d)
148         f_v.append(d_a)
149         d_a=[]
150         a_c.append(e)
151         b_c.append(e1)
152
153     DataOut1 = np.column_stack((a_c[0],b_c[0]))
154     np.savetxt(filename1, DataOut1,delimiter=',')
155     DataOut2 = np.column_stack((a_c[1],b_c[1]))
156     np.savetxt(filename2, DataOut2,delimiter=',')
157     DataOut3 = np.column_stack((a_c[2],b_c[2]))
158     np.savetxt(filename3, DataOut3,delimiter=',')
159     DataOut4 = np.column_stack((a_c[3],b_c[3]))
160     np.savetxt(filename4, DataOut4,delimiter=',')
161     DataOut5 = np.column_stack((a_c[4],b_c[4]))
162     np.savetxt(filename5, DataOut5,delimiter=',')
163
164     plt.plot(x_a,f_v[0],marker=".",label="i=1",linestyle='dashed')
165     plt.plot(x_a,f_v[1],marker=".",label="i=2",linestyle='dashed')
166     plt.plot(x_a,f_v[2],marker=".",label="i=5",linestyle='dashed')
167     plt.plot(x_a,f_v[3],marker=".",label="i=10",linestyle='dashed')
168     plt.plot(x_a,f_v[4],marker=".",label="i=20",linestyle='dashed')
169     plt.plot(x_a,ex,linewidth=2,c="black",label="Actual Function")
170     plt.grid()
171     plt.title(tit)
172     plt.legend()
173     plt.xlabel("x")
174     plt.ylabel("f(x)")
175     plt.show()
176     print()
177     print("-----Value of f(x) calculated using Fourier
Expansion-----")
178     data={"x_a":x_a,"f(x) (i=1)":f_v[0],"f(x) (i=2)":f_v[1],"f(x) (i=5)":
f_v[2],"f(x) (i=10)":f_v[3],"f(x) (i=20)":f_v[4],"f(x) (exact)":ex}
179     print(pd.DataFrame(data))
180
181
182
183     for (r,t,y,p,j,l) in zip (f_[0],f_[1],f_[2],f_[3],f_[4],range(3)):
184         E1.append(abs(r-ex2[1]))
185         E2.append(abs(t-ex2[1]))
186         E3.append(abs(y-ex2[1]))
187         E4.append(abs(p-ex2[1]))
188         E5.append(abs(j-ex2[1]))
189
190     print()
191     print("-----Absolute error for x
=[-0.5,0,0.5]-----")
192     data={"x":A_X,"Error (i=1)":E1,"Error (i=2)":E2,"Error (i=5)":E3,"
Error (i=10)":E4,"Error (i=20)":E5}
193     print(pd.DataFrame(data))

```

```

194
195 #Function1
196 def f1(x):
197     if x>-1 and x<0:
198         return 0
199     elif x>0 and x<1:
200         return 1
201     elif x==-1 or x==0 or x==1:
202         return 1/2
203
204 def pf1(x):
205     if x>=-1 and x<=1 :
206         return f1(x)
207     elif x>1:
208         x_new=x-(1-(-1))
209         return pf1(x_new)
210     elif x<(-1):
211         x_new=x+(1-(-1))
212         return pf1(x_new)
213
214 #Function2
215 def f2(x):
216     if x>-1 and x<-0.5:
217         return 0
218     elif x>-0.5 and x<0.5:
219         return 1
220     elif x>0.5 and x<1:
221         return 0
222     elif x==-1 or x==0.5 or x==1 or x==-0.5:
223         return 1/2
224
225 def pf2(x):
226     if x>=-1 and x<=1 :
227         return f2(x)
228     elif x>1:
229         x_new=x-(1-(-1))
230         return pf2(x_new)
231     elif x<(-1):
232         x_new=x+(1-(-1))
233         return pf2(x_new)
234
235 #Function3
236 def f3(x):
237     if x>-1 and x<0:
238         return -0.5
239     elif x>0 and x<1:
240         return 0.5
241     elif x==-1 or x==0 or x==1:
242         return 0
243
244 def pf3(x):
245     if x>=-1 and x<=1 :
246         return f3(x)
247     elif x>1:
248         x_new=x-(1-(-1))

```

```

249         return pf3(x_new)
250     elif x<(-1):
251         x_new=x+(1-(-1))
252         return pf3(x_new)
253
254 print("***** (ALPHA)
255 *****")
256 Task(1,f1,pf1,-1,3,"Fourier Series Approximation for alpha part of
257 Ques(b)(iii)","alpha1.dat","alpha2.dat","alpha3.dat","alpha4.dat","
258 alpha5.dat")
259 print()
260 print("***** (BETA)
261 *****")
262 Task(1,f2,pf2,0,3,"Fourier Series Approximation for beta part of Ques(
263 b)(iii)","beta1.dat","beta2.dat","beta3.dat","beta4.dat","beta5.dat
264 ")
265 print()
266 print("***** (GAMMA)
267 *****")
268 Task(1,f3,pf3,1,3,"Fourier Series Approximation for gamma part of Ques
269 (b)(iii)","gamma1.dat","gamma2.dat","gamma3.dat","gamma4.dat","
270 gamma5.dat")

```

Source Code 1: Python Program

```

1 #Name= Monu Chaurasiya
2 #College Roll No. = 2020 PHY1102
3 #University Roll No. = 20068567035
4 '''
5 PARTNERS:
6 #Name= Pawanpreet Kaur
7 #College Roll No. = 2020 PHY1092
8 #University Roll No. = 20068567038
9
10 #Name= Prateek Bhardwaj
11 #College Roll No. = 2020 PHY1110
12 #University Roll No. = 20068567042
13 '''
14
15 from MyIntegration import MySimp
16 from MyIntegration import MyTrap
17 from MyIntegration import MyLegQuadrature
18 import pandas as pd
19 import numpy as np
20 import matplotlib.pyplot as plt
21 import math
22 from scipy import integrate
23 from sympy import *
24 from sympy import simplify
25
26 x=symbols('x')
27
28 #Function for Fourier Coefficient
29 def FourierCoeff(f,Prop_Key,a,b,L,N,m_k,tol):
30     an=0;an_a=[]
31     bn=0;bn_a=[]

```



```

32     a0=0
33     if Prop_Key==1:
34         for i in range(1,N+1):
35             fun = lambda x: f(x)*np.sin(i*np.pi*x/L)
36             if m_k==1:
37                 bn=(1/L)*MyTrap(fun,a,b,1,key1=True,N_max=1000,key2=
True,tol=tol)[0]
38             elif m_k==2:
39                 bn=(1/L)*MyTrap(fun,a,b,1,key1=True,N_max=1000,key2=
True,tol=tol)[0]
40             elif m_k==3:
41                 bn=(1/L)*MyLegQuadrature(fun,a,b,10,1,key=True,tol=
tole,m_max=1000)[0]
42             else:
43                 return "Wrong Method Key entered"
44             bn_a.append(bn)
45             an_a.append(0)
46         return a0,an_a,bn_a
47
48     elif Prop_Key==0:
49         if m_k==1:
50             a0=(1/L)*MyTrap(f,a,b,1,key1=True,N_max=1000,key2=True,tol
=0.1e-3)[0]
51         elif m_k==2:
52             a0=(1/L)*MySimp(f,a,b,2,key1=True,N_max=1000,key2=True,tol
=tol)[0]
53         elif m_k==3:
54             a0=(1/L)*MyLegQuadrature(f,a,b,10,1,key=True,tol=tol,
m_max=1000)[0]
55         else:
56             return "Wrong Method Key entered"
57
58         for i in range(1,N+1):
59             fun = lambda x: f(x)*np.cos(i*np.pi*x/L)
60             if m_k==1:
61                 an=(1/L)*MyTrap(fun,a,b,1,key1=True,N_max=1000,key2=
True,tol=tol)[0]
62             elif m_k==2:
63                 an=(1/L)*MySimp(fun,a,b,1,key1=True,N_max=1000,key2=
True,tol=tol)[0]
64             elif m_k==3:
65                 an=(1/L)*MyLegQuadrature(fun,a,b,10,1,key=True,tol=
tole,m_max=1000)[0]
66             else:
67                 return "Wrong Method Key entered"
68             an_a.append(an)
69             bn_a.append(0)
70         return a0,an_a,bn_a
71
72     elif Prop_Key==-1:
73         if m_k==1:
74             a0=(1/L)*MyTrap(f,a,b,1,key1=True,N_max=1000,key2=True,tol
=0.1e-3)[0]
75         elif m_k==2:
76             a0=(1/L)*MySimp(f,a,b,2,key1=True,N_max=1000,key2=True,tol

```

```

77         =tole)[0]
78         elif m_k==3:
79             a0=(1/L)*MyLegQuadrature(f,a,b,10,1,key=True,tol=tole,
80             m_max=1000)[0]
81         else:
82             return "Wrong Method Key entered"
83
84         for i in range(1,N+1):
85             fun1 = lambda x: f(x)*np.cos(i*np.pi*x/L)
86             fun2 = lambda x: f(x)*np.sin(i*np.pi*x/L)
87             if m_k==1:
88                 an=(1/L)*MyTrap(fun1,a,b,1,key1=True,N_max=1000,key2=
89                 True,tol=tole)[0]
90                 bn=(1/L)*MyTrap(fun2,a,b,1,key1=True,N_max=1000,key2=
91                 True,tol=tole)[0]
92             elif m_k==2:
93                 an=(1/L)*MySimp(fun1,a,b,2,key1=True,N_max=1000,key2=
94                 True,tol=tole)[0]
95                 bn=(1/L)*MySimp(fun2,a,b,2,key1=True,N_max=1000,key2=
96                 True,tol=tole)[0]
97             elif m_k==3:
98                 an=(1/L)*MyLegQuadrature(fun1,a,b,10,1,key=True,tol=
99                 tole,m_max=1000)[0]
100                 bn=(1/L)*MyLegQuadrature(fun2,a,b,10,1,key=True,tol=
101                 tole,m_max=1000)[0]
102             else:
103                 return "Wrong Method Key entered"
104             an_a.append(an)
105             bn_a.append(bn)
106         return a0,an_a,bn_a
107     else :
108         return "Wrong key entered"
109
110 def Task(L,f1,pf1,Prop_Key,m_k):
111     N_a=[1,2,5,10, 20]
112     x_a=np.linspace(-10,10,100)
113     f_v=[]
114     f_=[]
115     a_c=[]
116     b_c=[]
117     E1=[];E2=[];E3=[];E4=[];E5=[]
118     A_X=[0,np.pi/2,np.pi]
119     ex2=[]
120     for x in A_X:
121         ex2.append(pf1(x))
122     for N in N_a:
123         s=FourierCoeff(pf1,Prop_Key=Prop_Key,a=0,b=L,L=np.pi,N=N,m_k=
124         m_k,tol=0.1e-8)
125         a0=2*s[0]
126         e=np.array(s[1])
127         e=np.dot(2,e)
128         e1=np.array(s[2])
129         e1=np.dot(2,e1)
130         d_a=[]

```

```

123     d_=[]
124     ex=[]
125     for x in x_a:
126         ex.append(pf1(x))
127     for x in A_X:
128         sin=0
129         cos=0
130
131         for (i,an,bn) in zip(range(1,N+1),e,e1):
132             sin+=bn*np.sin(i*np.pi*x/L)
133             cos+=an*np.cos(i*np.pi*x/L)
134             d=a0/2+sin+cos
135             d_.append(d)
136     f_.append(d_)
137     d_=[]
138
139     for x in x_a:
140         sin=0
141         cos=0
142
143         for (i,an,bn) in zip(range(1,N+1),e,e1):
144             sin+=bn*np.sin(i*np.pi*x/L)
145             cos+=an*np.cos(i*np.pi*x/L)
146             d=a0/2+sin+cos
147             d_a.append(d)
148     f_v.append(d_a)
149     d_a=[]
150     a_c.append(e)
151     b_c.append(e1)
152
153     print()
154     print("-----Value of f(x) calculated using Fourier
Expansion-----")
155     data={"x_a":x_a,"f(x) (i=1)":f_v[0],"f(x) (i=2)":f_v[1],"f(x) (i=5)":
f_v[2],"f(x) (i=10)":f_v[3],"f(x) (i=20)":f_v[4],"f(x) (exact)":ex}
156     print(pd.DataFrame(data))
157
158
159
160     for (r,t,y,p,j,l) in zip (f_[0],f_[1],f_[2],f_[3],f_[4],range(3)):
161         E1.append(abs(r-ex2[1]))
162         E2.append(abs(t-ex2[1]))
163         E3.append(abs(y-ex2[1]))
164         E4.append(abs(p-ex2[1]))
165         E5.append(abs(j-ex2[1]))
166
167     print()
168     print("-----Absolute error for x=[0,pi/2,pi
]-----")
169     data={"x":A_X,"Error (i=1)":E1,"Error (i=2)":E2,"Error (i=5)":E3,"
Error (i=10)":E4,"Error (i=20)":E5}
170     print(pd.DataFrame(data))
171     return x_a,f_v
172
173 #odd Function

```

```

174 def f1(x):
175     if x>=-np.pi and x<=0:
176         return x
177     elif x>0 and x<=np.pi:
178         return x
179
180
181 def pf1(x):
182     if x>=-np.pi and x<=np.pi :
183         return f1(x)
184     elif x>np.pi:
185         x_new=x-(np.pi-(-np.pi))
186         return pf1(x_new)
187     elif x<(-np.pi):
188         x_new=x+(np.pi-(-np.pi))
189         return pf1(x_new)
190
191 #even function
192 def f2(x):
193     if x>=-np.pi and x<=0:
194         return -1*x
195     elif x>0 and x<=np.pi:
196         return x
197
198
199 def pf2(x):
200     if x>=-np.pi and x<=np.pi :
201         return f2(x)
202     elif x>np.pi:
203         x_new=x-(np.pi-(-np.pi))
204         return pf2(x_new)
205     elif x<(-np.pi):
206         x_new=x+(np.pi-(-np.pi))
207         return pf2(x_new)
208
209 x_a1,f_v1=Task(np.pi,f1,pf1,1,3)
210 x_a=np.linspace(-10,10,100)
211 s1=[]
212 for x in x_a:
213     s1.append(pf1(x))
214 fig, (ax1, ax2) = plt.subplots(1, 2)
215 fig.suptitle('3b')
216 ax1.plot(x_a1,f_v1[0],marker=".",label="i=1",linestyle='dashed')
217 ax1.plot(x_a1,f_v1[1],marker=".",label="i=2",linestyle='dashed')
218 ax1.plot(x_a1,f_v1[2],marker=".",label="i=5",linestyle='dashed')
219 ax1.plot(x_a1,f_v1[3],marker=".",label="i=10",linestyle='dashed')
220 ax1.plot(x_a1,f_v1[4],marker=".",label="i=20",linestyle='dashed')
221 ax1.plot(x_a1,s1,linewidth=2,c="black",label="Odd Extension of
    Function")
222 ax1.grid()
223 ax1.set(xlabel="x",ylabel="f(x)",title="Expansion of f(x)=x as an odd
    function")
224 ax1.legend()
225
226

```

```

227 x_a,f_v=Task(np.pi,f2,pf2,0,3)
228 x_a=np.linspace(-10,10,100)
229 s=[]
230 for x in x_a:
231     s.append(pf2(x))
232
233 ax2.plot(x_a,f_v[0],marker=".",label="i=1",linestyle='dashed')
234 ax2.plot(x_a,f_v[1],marker=".",label="i=2",linestyle='dashed')
235 ax2.plot(x_a,f_v[2],marker=".",label="i=5",linestyle='dashed')
236 ax2.plot(x_a,f_v[3],marker=".",label="i=10",linestyle='dashed')
237 ax2.plot(x_a,f_v[4],marker=".",label="i=20",linestyle='dashed')
238 ax2.plot(x_a,s,linewidth=2,c="black",label="Even Extension of Function
    ")
239 ax2.grid()
240 ax2.set(xlabel="x",ylabel="f(x)",title="Expansion of f(x)=x as an Even
    function")
241 ax2.legend()
242 plt.show()

```

Source Code 2: Python Program