Project Report for ECE 351

Lab 08 - Fourier Series Approximation of a Square Wave

Skyler Corrigan

October 13, 2022

ECE351 Code Repository:
$https://github.com/ElfinPeach/ECE351_Code.git$

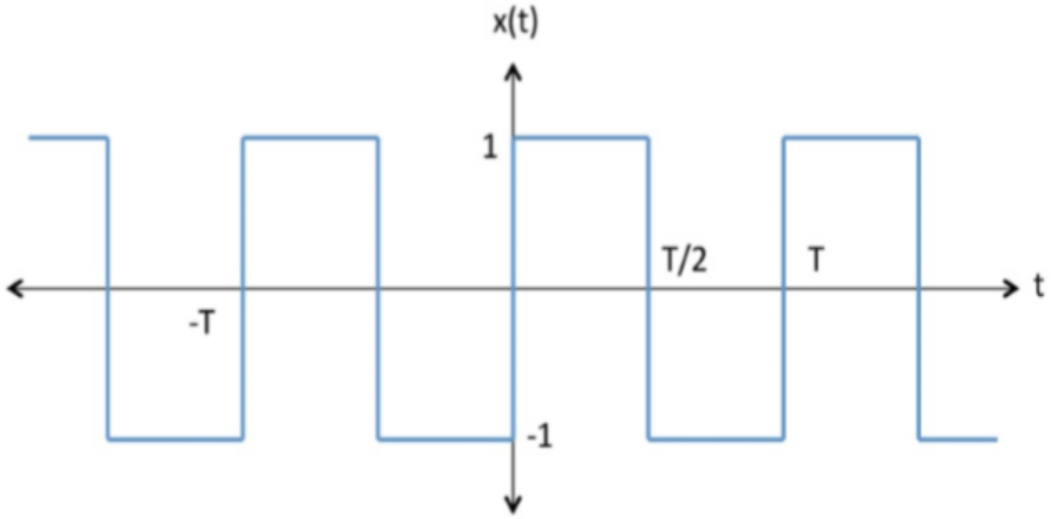
ECE351 Report Repository:
$https://github.com/ElfinPeach/ECE351_Report.git$

Contents

1 Objective

This lab is about trying to find the Fourier Series of the following Square Wave.

Figure 1



2 Equations

The following equations can be used to define the wave:

$$x(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_k \cos(k\omega_0 t)$$
$$a_k = \frac{2}{T}\int_0^T x(t)\cos(k\omega_0 t)dt$$
$$b_k = \frac{2}{T}\int_0^T x(t)\sin(k\omega_0 t)dt$$
$$\omega_0 = \frac{2\pi}{T}$$

Since this is an odd function, $a_0 = a_k = 0$.
For b_k:

$$b_k = \frac{2}{k\pi}(-cos(k\pi) + 1)$$
$$\text{if k is even, } b_k = 0$$
$$\text{if k is odd, } b_k = \frac{4}{k\pi}$$
$$\text{Therefore: } x(t) = \sum_{k=1}^{\infty} \frac{4}{k\pi}\sin(k\omega_0 t) \text{ for when k is even.}$$

Though, for the code, the original b_k function will be used in x_t to simplify the code (plus the code kept breaking when I tried to do it that way...).

3 Code

The following is the code used in the lab. *Lab Code*

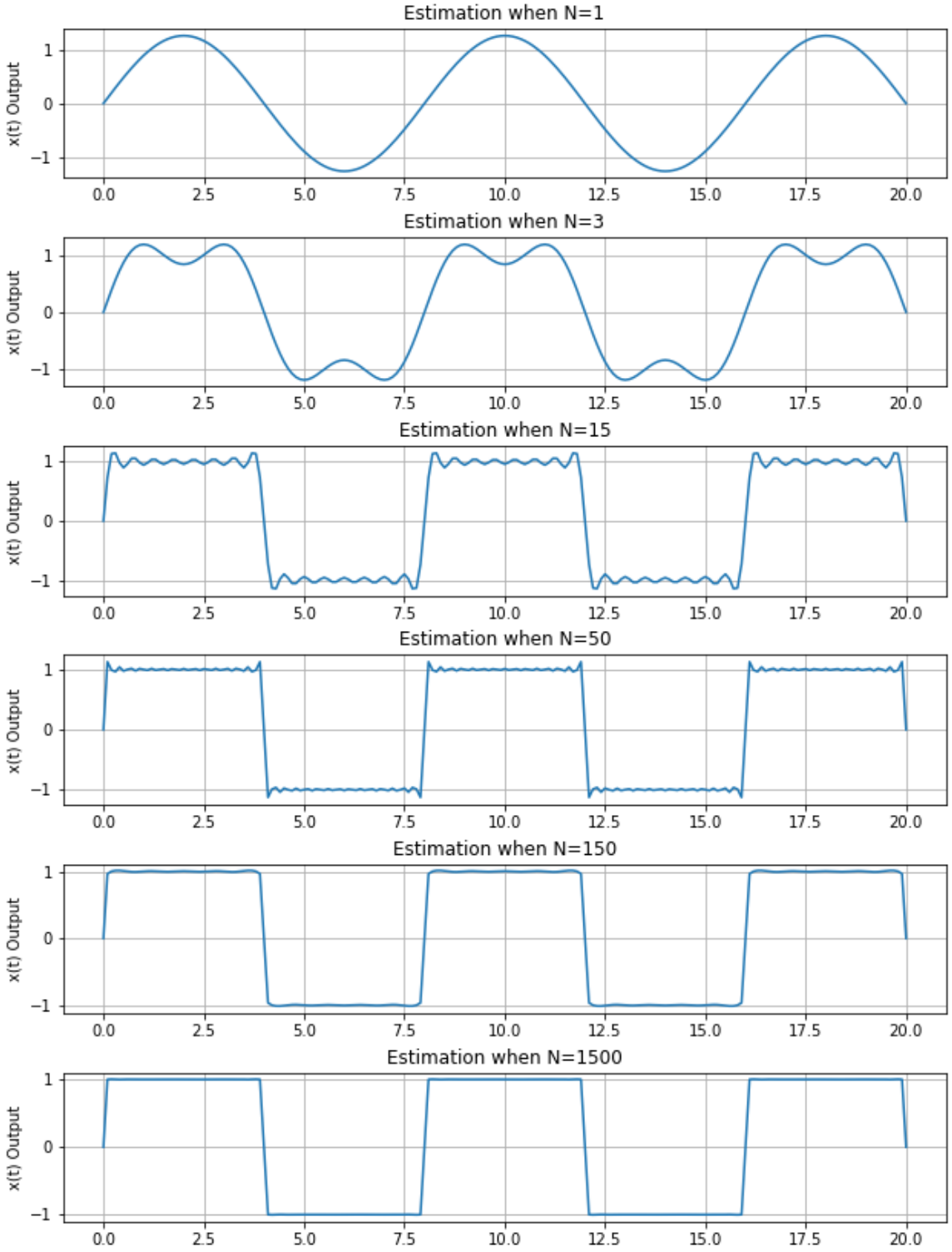
```
import numpy as np
import matplotlib.pyplot as plt

#----------PT 1----------#
def b_k(n):
    b = (2 / (n * np.pi)) * (1 - np.cos((n) * np.pi))
    return b


def W(period):
    return ((2*np.pi)/period)


def xFourier(t, period, n):
    x_t = 0 #initialization
    for i in np.arange(1, n+1):
        x_t += (b_k(i))*(np.sin(i * W(period) * t))
    return x_t

#Since this is an odd function, a0 = an = 0

#step size
steps = 1e-1

# t for part 1
start = 0
stop = 20
# Define a range of t. Start at 0 and go to 20 (+a step)
t = np.arange(start, stop + steps, steps)

#Make arrays to plot against t
x_1 = xFourier(t, 8, 1)
x_3 = xFourier(t, 8, 3)
x_15 = xFourier(t, 8, 15)
x_50 = xFourier(t, 8, 50)
x_150 = xFourier(t, 8, 150)
x_1500 = xFourier(t, 8, 1500)

#plot stuff
plt.figure(figsize=(10, 12))
#space out subplots
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9,
```

```
                    top=1.0, wspace=0.4, hspace=0.4)

#PLOT EVERYTHING!!!!
plt.subplot(6, 1, 1)
plt.plot(t, x_1)
plt.title("Estimation when N=1")
plt.ylabel("x(t) Output")
plt.grid()

plt.subplot(6, 1, 2)
plt.plot(t, x_3)
plt.title("Estimation when N=3")
plt.ylabel("x(t) Output")
plt.grid()

plt.subplot(6, 1, 3)
plt.plot(t, x_15)
plt.title("Estimation when N=15")
plt.ylabel("x(t) Output")
plt.grid()

plt.subplot(6, 1, 4)
plt.plot(t, x_50)
plt.title("Estimation when N=50")
plt.ylabel("x(t) Output")
plt.grid()

plt.subplot(6, 1, 5)
plt.plot(t, x_150)
plt.title("Estimation when N=150")
plt.ylabel("x(t) Output")
plt.grid()

plt.subplot(6, 1, 6)
plt.plot(t, x_1500)
plt.title("Estimation when N=1500")
plt.ylabel("x(t) Output")
plt.grid()

plt.show()

print("a0 and an values for a are zero. This is because the wave is odd and
print("")
print("b1 =", b_k(1))
print("")
```

```
print("b2 = ", b_k(2))
print("")
print("b3 = ", b_k(3))
```

4 Results

The following graphs are the Fourier Series Approximation of the square wave seen in *Figure 1*, iterated at $N = 1, 3, 15, 50, 150, 1500$.
Figure 2



5 Questions

5.1 Question 1

Is x(t) an even or an odd function? Explain why.

x(t) is an odd function because $x(t) = -x(-t)$

5.2 Question 2

Based on your results from Task 1, what do you expect the values of a2, a3, . . . , an to be? Why?

I expect all those value to be zero because it's an odd function!

5.3 Question 3

How does the approximation of the square wave change as the value of N increases? In what way does the Fourier series struggle to approximate the square wave?

The approximation gets better as N increases. Fourier stuggles to approximate the square wave because its lines are straight!

5.4 Question 4

What is occurring mathematically in the Fourier series summation as the value of N increases?

The Fourier Series is approaching a limit at some time value.

5.5 Question 5

Leave any feedback on the clarity of lab tasks, expectations, and deliverables.

It may be a good idea to, in the prelab anyway, throw out a hint, something like: "Hint: Look up the difference between odd and even functions when approximating a graph." I personally remembered there was a "cheat" to make life easier, but not everyone does.