ECE 351 - 52

Signals and Systems 1

Lab 8

Submitted By :
Owen Blair

Contents

1 Important Notes 2
2 Part 1 Deliverables 2
3 Results 4
3.1 Figure 1: Fourier Estimations when N=[1,3,15,50,150,1500] . . 4
4 Equations 5
4.1 Equation 1: Expression for a_k and b_k 5
5 Listings 5
5.1 Listing 1: Output for a_0, a_1, and $b_n = [1, 2, 3]$ 5
5.2 Listing 2: Code for b_n Output 5
5.3 Listing 3: Code for Fourier Estimation and Plotting 5
6 Questions 7
6.1 Question 1 7
6.2 Question 2 7
6.3 Question 3 8
6.4 Question 4 8

1 Important Notes

It is important to note that plt.show() is needed at the end of the python code to properly show the plots of each function. It was not included in every section of code that plots a function(s) because it is assumed that its included at the end the code.

The web address to the GitHub where LaTeX code is stored is here:
https://github.com/Blairis123/ECE351_Reports

The web address to the GitHub where the Python code is here:
https://github.com/Blairis123/ECE351_Code

2 Part 1 Deliverables

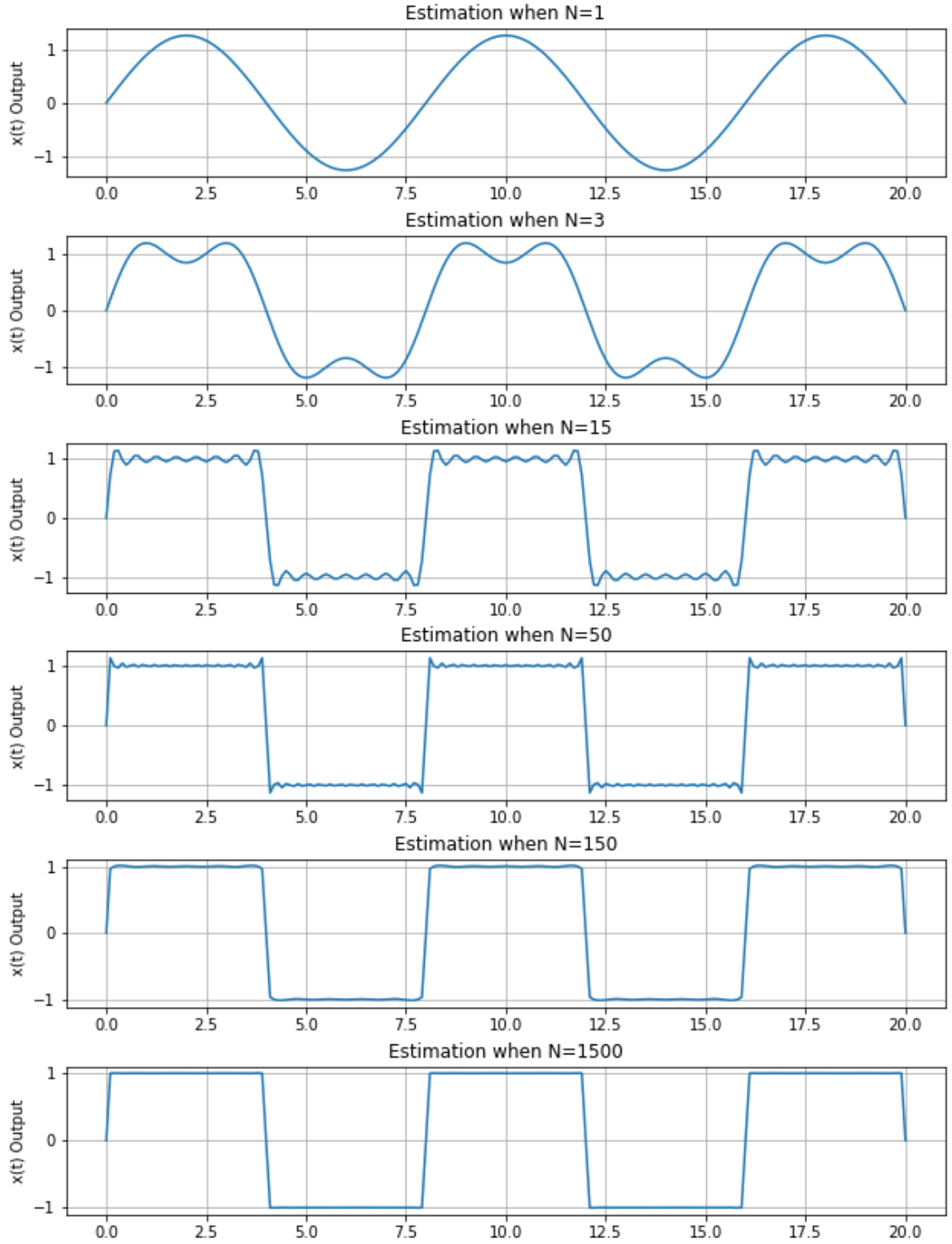
Part 1 is about using Python to model a Fourier transform. The function to model was given in the lab handout. This starts with finding the values for a_n and b_n. It turns out that the function is odd and doesn't have an offset. The equations that were used as inputs for a_n and b_n can be seen in *Equation 1*. The output for the first few a_n and b_n can be found in *Listing 1* and the code that generated the output can be found in *Listing 2*.

Within *Listing 3* there are three new user defined functions that create a Fourier transform estimation of the function in the lab handout. The function b_n() takes an input of N, the number of iterations to estimate the function, and outputs an array of values for each N value. The function W() calculates the fundamental frequency using an input of the function period and returns a float.

The third user defined function is xFourier() function. This takes inputs of t: a time step array, period: the period of the function, and n: the number of iterations in the estimation. The xFourier() uses the previously two user defined functions as helper functions. The b_n is used to find the coefficients for the Fourier transform and the W() function is used to find the fundamental frequency. The xFourier() function returns an array to plot against the t array input.

3 Results

3.1 Figure 1: Fourier Estimations when N=[1,3,15,50,150,1500]



4 Equations

4.1 Equation 1: Expression for a_k and b_k

$$a_k = 0$$

$$b_k = \frac{-2}{k\pi}(cos(k\pi) - 1)$$

5 Listings

5.1 Listing 1: Output for a_0, a_1, and $b_n = [1, 2, 3]$

```
runfile('D:/U_of_I/Fall_2021/ECE-351/Lab8/Lab8_OwenBlair.py', wdir=
    'D:/U_of_I/Fall_2021/ECE-351/Lab8')
A0 and all values for An are zero.
This is because the wave is odd and has no  offset.

B1 is
1.2732395447351628
B2 is
-0.0
B3 is
0.4244131815783876
```

5.2 Listing 2: Code for b_n Output

```
print("a0 and all values for a are zero. This is")
print("because the wave is odd and has no  offset.")
print()
for i in range(1,4):
    print("B%d is" %i)
    print(b_n(i))
```

5.3 Listing 3: Code for Fourier Estimation and Plotting

```
#---------PT 1----------------------------------------#

# Finding b_n for fourier estimation given an n
def b_n(n):
    b = (-2/((n)*np.pi)) * (np.cos((n) * np.pi) - 1)
    return b

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

```

```
def xFourier(t, period, n):
    x_t = 0
    for i in np.arange(1, n+1):
        x_t += (np.sin(i * W(period) * t) * b_n(i))

    return x_t
"""
 a_0 = 0 because function has a DC offset of 0
 a_n = 0 Because function is a reflection along the line y=-x
     This means that the function will always have non-zero
   coefficients
     in the sine terms.
"""

# Define step size
#steps = 1e-2
steps = 1e-1

# t for part 1
start = 0
stop = 20
# Define a range of t_pt1. Start @ 0 go to 20 (+a step) w/
# a stepsize of step
t = np.arange(start, stop + steps, steps)

# Calculate b_n for testing
n1 = 1
#print(b_n1)

# 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)

#Define plot size!
plt.figure(figsize=(10, 12))
# set the spacing between subplots
plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9,
                    top=1.0, wspace=0.4, hspace=0.4)

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()
```

6 Questions

6.1 Question 1

x(t) is an odd function because $x(t) = -x(-t)$. In short the function can be reflected around the line $y = -x$.

6.2 Question 2

Based on the results from the first task, the expected value for all a_n is zero. This is because the function is odd and does not have a DC offset.

6.3 Question 3

As N increases, the Fourier transform gets closer and closer to the original function. It can also be seen that the edges of the Fourier transform are slightly higher than the top of the original pulse function. The edges of the estimation are also not perfectly vertical. As N increases, the slope of the edges approaches vertical but will never actually become vertical. This means that the Fourier transform will always have trouble estimating the edges of a vertical edge.

6.4 Question 4

As N increases, the frequency of each successive cosine/sine function becomes greater.