ECE351-51: LAB 2

SEPTEMBER 16, 2019

Adriana Oliveira

INTRODUCTION

This lab covers introductory aspects of writing user defined functions to express signal operations. These functions were used to plot multiple versions of these signals. In the process, we learned how signals are effected by simple manipulation of their equations.

METHODOLOGY

Part 1

We started by importing numpy and matplotlib.pyplot packages. Next we created our first user defined function "func1" to implement $y = cos(t)$ and then plotted it with a fine resolution.

Listing 1: Code for func1

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

# User defined func1
def func1(t):
    y = np.zeros((len(t),1))

    for i in range(len(t)):
        y[i] = np.cos(t[i])
    return y
```

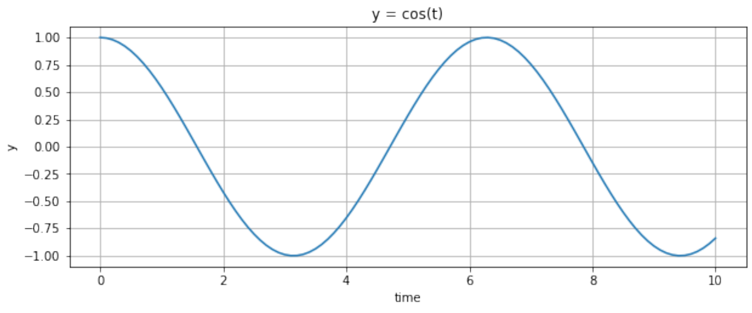


Figure 1: Plot for $y = cos(t)$

Part 2

We continued our use of user defined functions by defining and graphing an unknown function. We started by deriving the equation below based on the plotted image in the lab 2 handout.

$$y(t) = r(t) - r(t-3) + 5u(t-3) - 2u(t-6) - 2r(t-6)$$

Next, we wrote functions to define the ramp and unit step function found within the equations.

Listing 2: Code for ramp and unit step functions

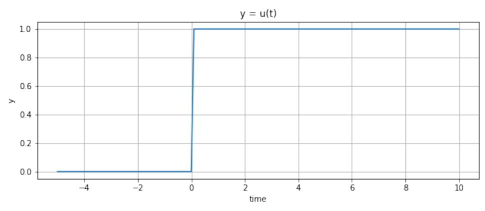
```
def u(t):
    if t < 0:
        return 0
    if t >= 0:
        return 1

def r(t):
    if t < 0:
        return 0
    if t >= 0:
        return t
```

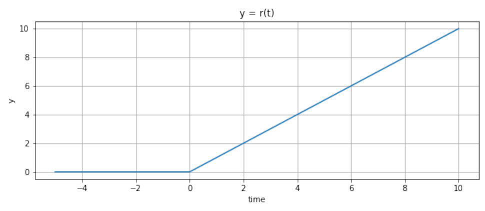
We then implemented this code by creating another function that properly enacted the functions.

Listing 3: Code for enacting ramp and step functions

```
def arrayconvert (t, func):
    y = np.zeros((len(t), 1))
    for i in range(len(t)):
        y[i] = func(t[i])
    return y
```



(a) Step function



(b) Ramp function

Figure 2: Plots for basic functions

This function allowed us to plot the step and ramp functions separately as seen above. Next, we wrote a function to implement the equation we derived and plotted it.

Listing 4: Code for enacting derived equation

```
def f(t):
    y = np.zeros((len(t), 1))
    for i in range(len(t)):
        y[i] = r(t[i])-r(t[i]-3)+5*u(t[i]-3)-2*u(t[i]-6)-2*r(t[i]-6)
    return y

t = np.arange(-5,10+steps,steps)
y = f(t)
```

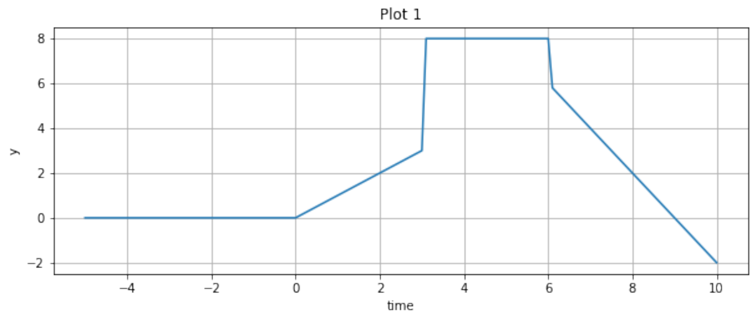
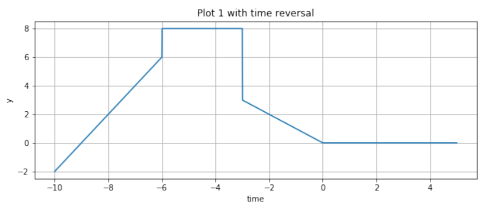


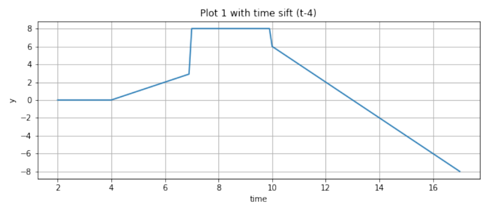
Figure 3: Plot for Derived Equation

Part 3

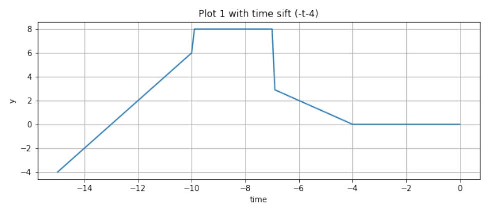
We tested our user defined function by shifting time, scaling, and negating the function.



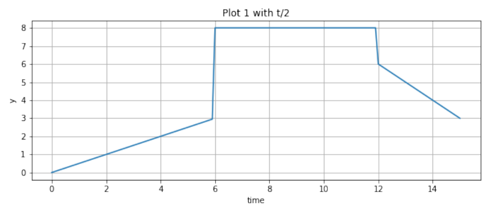
(a) Time reversal



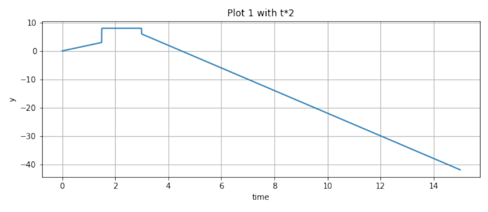
(b) Time shift (t-4)



(c) Time shift -(t-4)



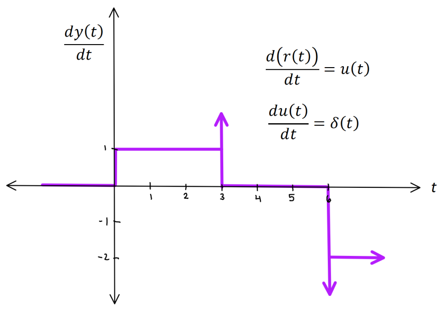
(d) t/2



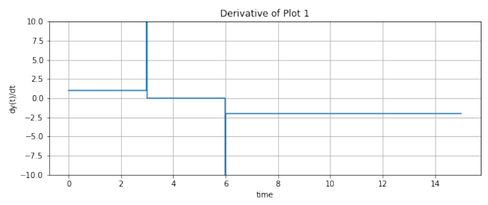
(e) 2t

Figure 4: Plots for Part 3

Next, we drew and plotted the derivative of the original equation.



(a) Hand drawn



(b) Plotted

Figure 5: Plots for Derivative

QUESTIONS

1. Are the plots from Part 3 Tasks 4 and 5 identical? Is it possible for them to match? Explain why or why not.

   They are very similar but not perfectly identical. Because the hand drawn plot is less precise and imperfectly scaled, the two will not be an exact match.

2. How does the correlation between the two plots (from Part 3 Tasks 4 and 5) change if you were to change the step size within the time variable in Task 5? Explain why this happens.

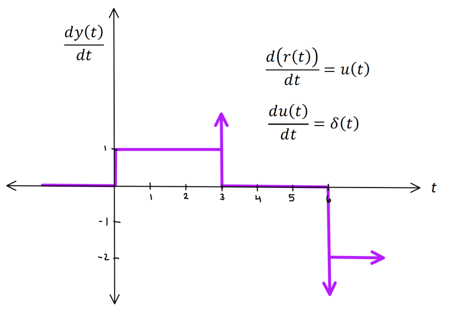
   The correlation gets much worse the larger the steps are. This is because the resolution decreases as the steps increase and a less precise image is produced.

3. What way can the expectations and tasks be more clearly explained for this lab?

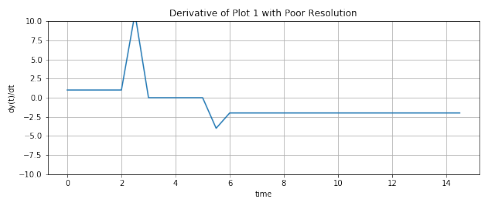
   Instructions for coding the derivative function were vague.

CONCLUSION

This lab gave valuable experience with learning user defined functions. It allowed a closer look at signals familiar from lectures and allowed us to manipulate and plot them.



(a) Hand Drawn



(b) Plotted with Poor Resolution

Figure 6: Plots for Derivative