User-Defined Functions

Cameron Williams

ECE 351-51

Lab Report 2

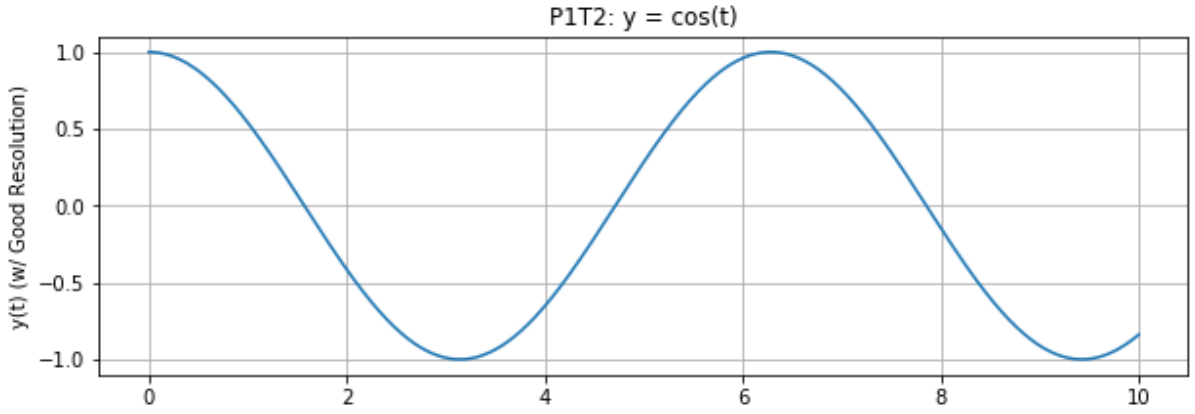
4 February 2020

1 Introduction

The objective of this lab was to write user-defined functions in Python and use them to demonstrate various signal operations such as time shifiting, time scaling, time reversal, signal addition, and discrete differentiation. The Python script used to generate the various plots for the purposes of this lab has been submitted alongside this report, but may also be found on my GitHub repository for this class at https://github.com/kwilliamscameron/ECE351_Code. All code in the associated script is divided into sections that match each part of the assignment sheet and this report.

2 Part 1

The first part of this lab consisted of following some example code of how to create a user defined function to generate a plot and creating a new function for plotting cosine functions. This was done uusing the matplotlib.pyplot package and numpy.cos. A plot generated using my function may be seen below.



3 Part 2

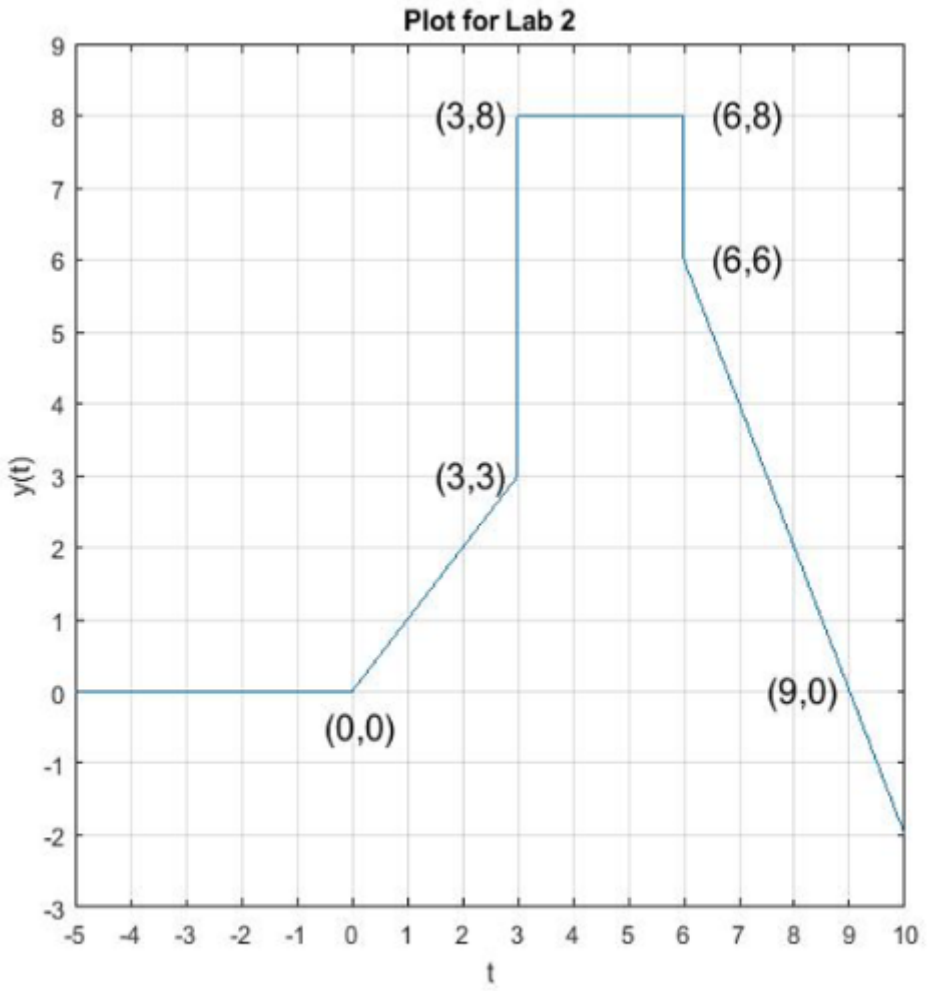
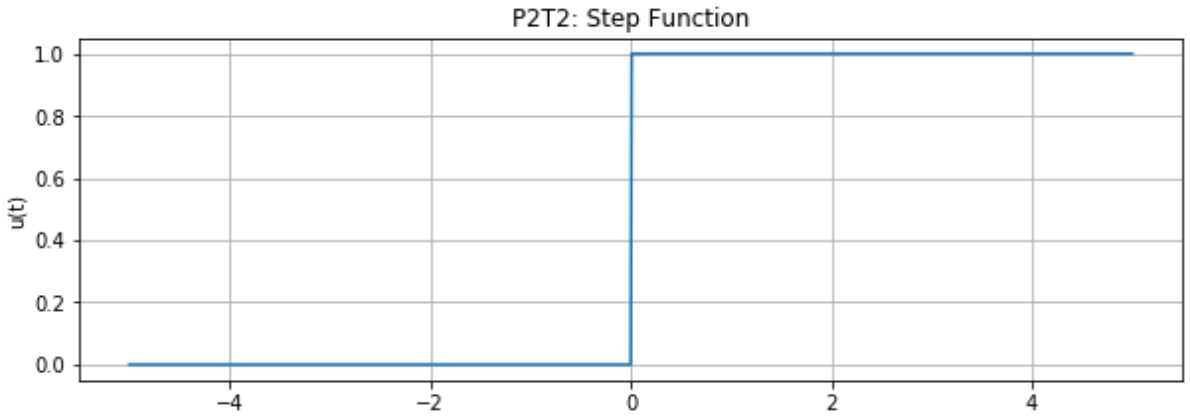


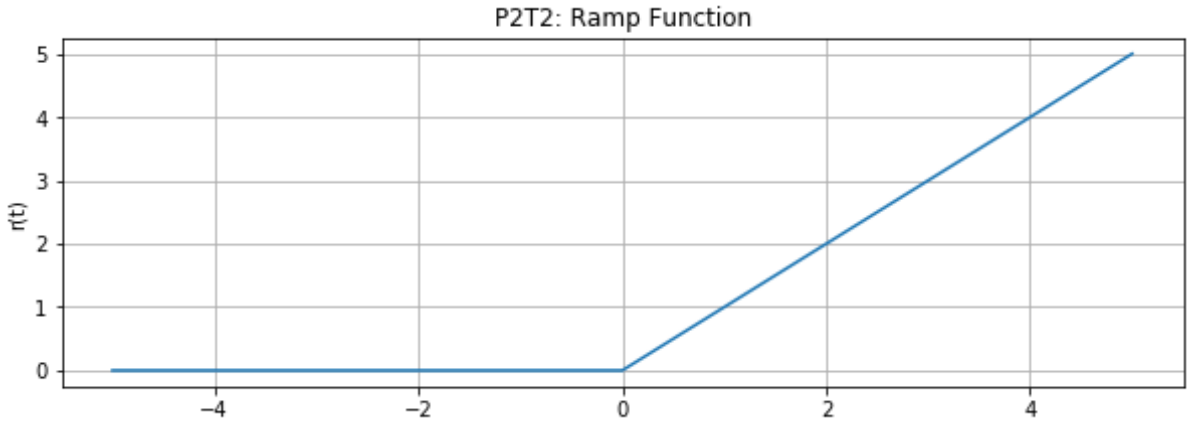
Figure 2: Function to Plot

Next, I derived an equation to model the function in Figure 2 of the lab outline. The figure may be seen above. The equation I derived for it is as follows:

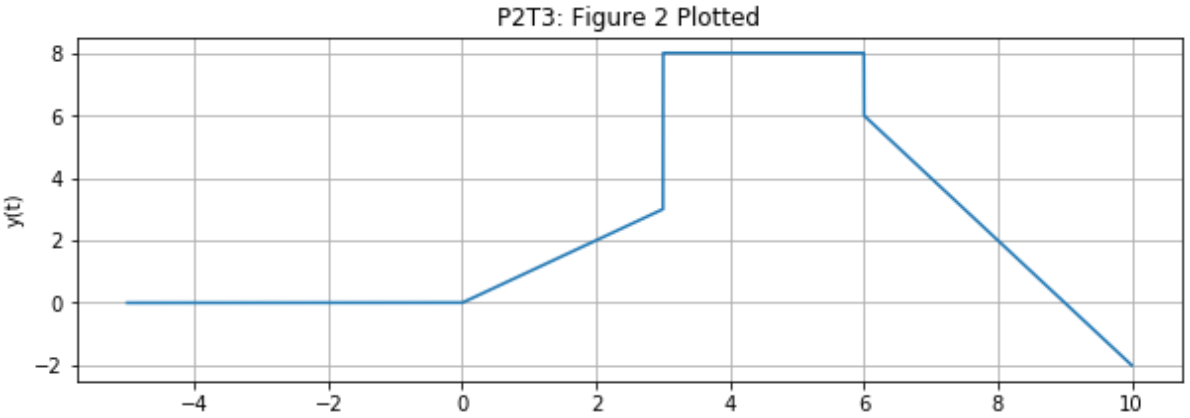
$$r(t) + 5u(t-3) - r(t-3) - 2u(t-6) - 2r(t-6)$$

After this, I created functions for step and ramp functions. Plots of u(t) and r(t) created using my functions may be seen below.



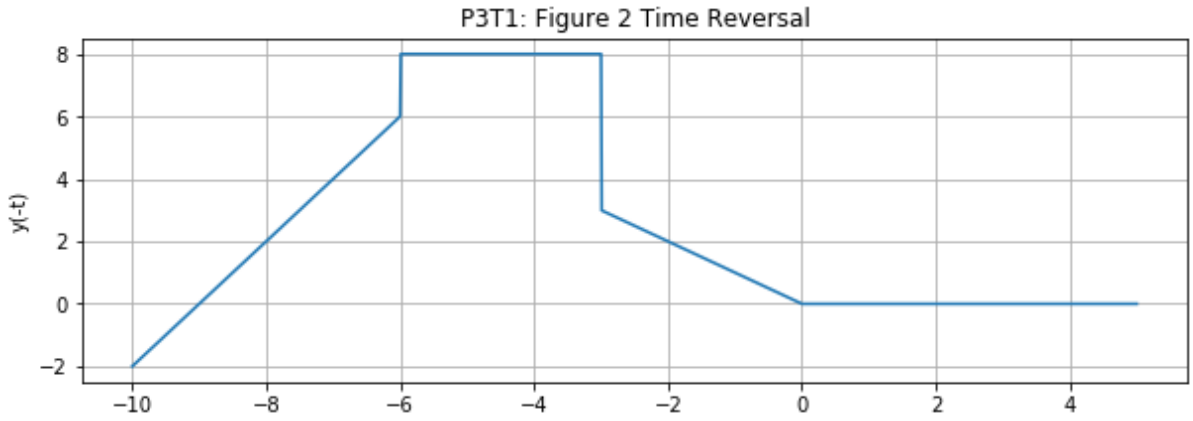


Then, I used my step and ramp functions to generate a plot of Figure 2 using my derived function. This plot may be seen below.

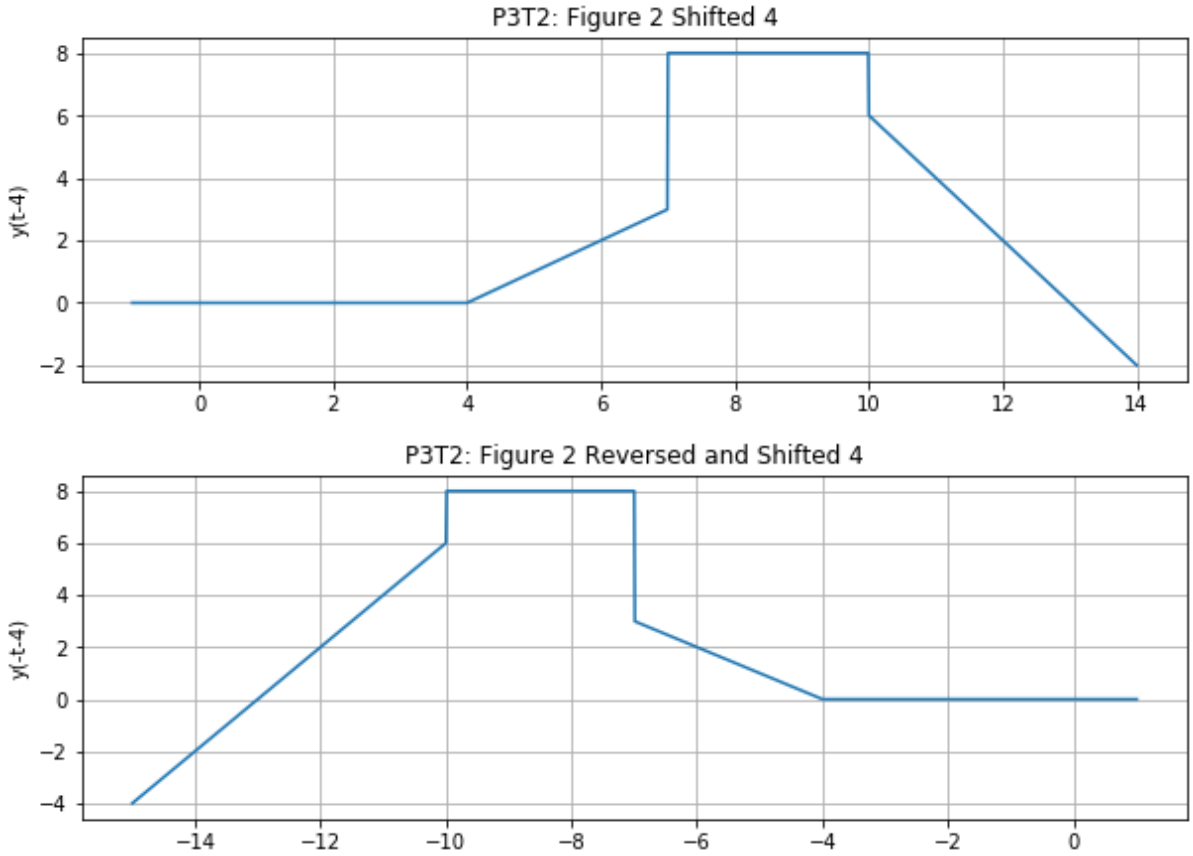


4 Part 3

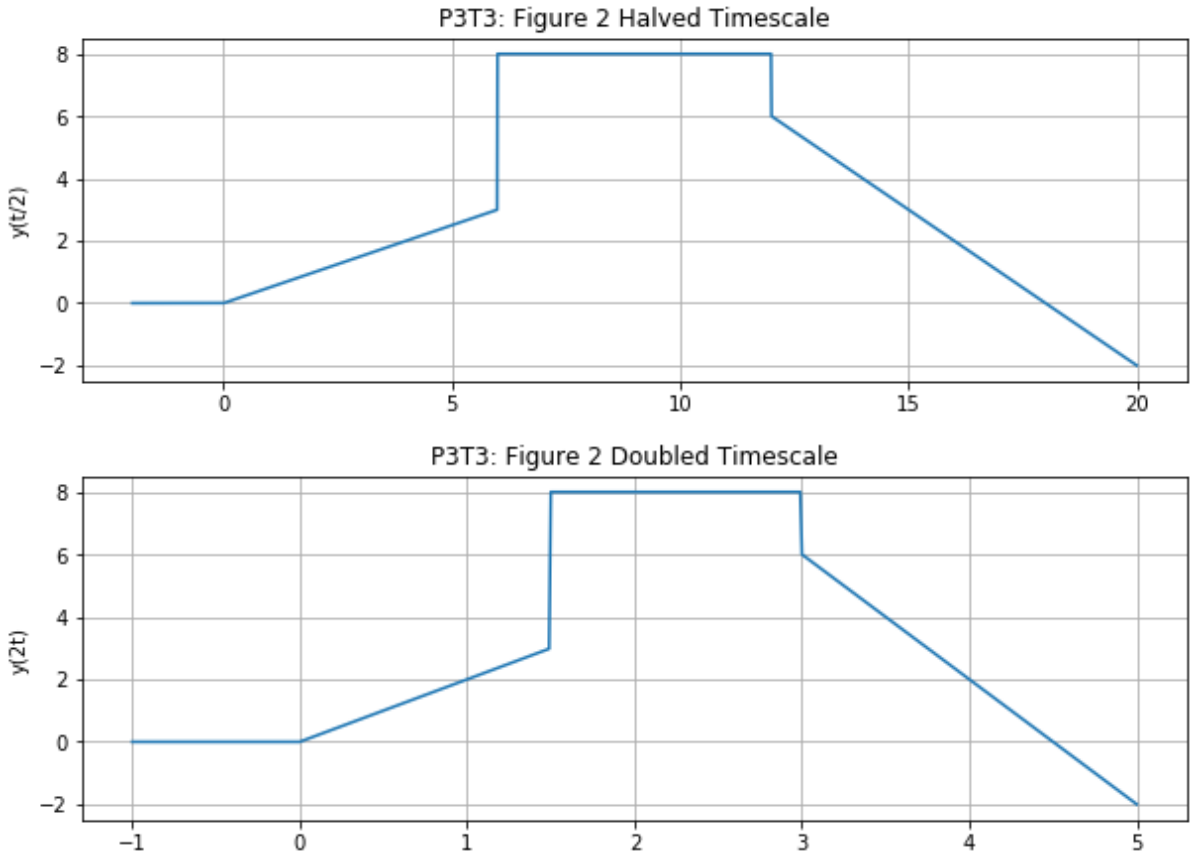
In the final portion of this lab, I further tested my implementation of the Figure 2 function by applying time reversal, time shifting, and time scaling. A plot of the function with time reversal applied may be seen below.



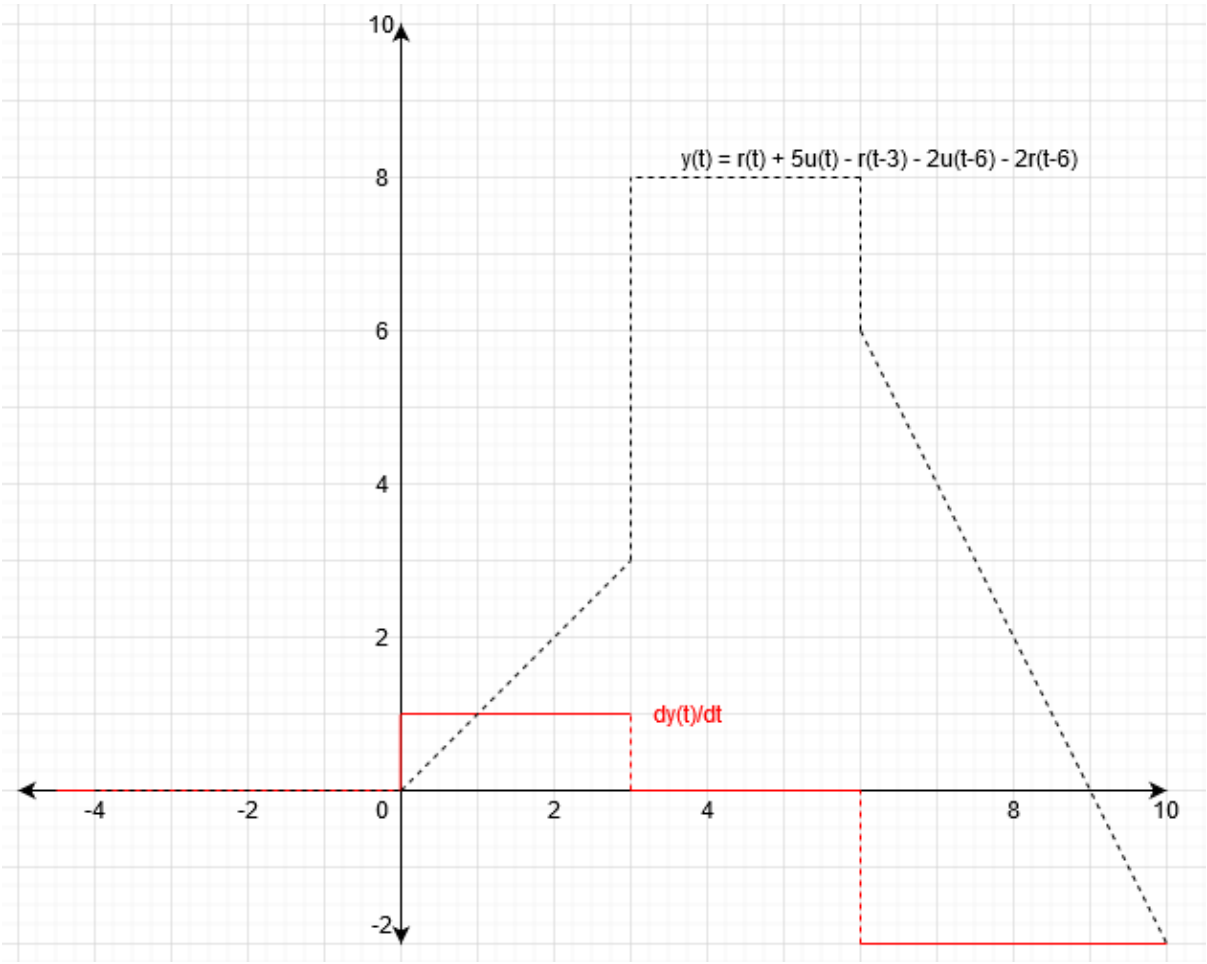
The following two plots show time shifting, f(t-4), and time shifting with time reversal, f(-t-4).



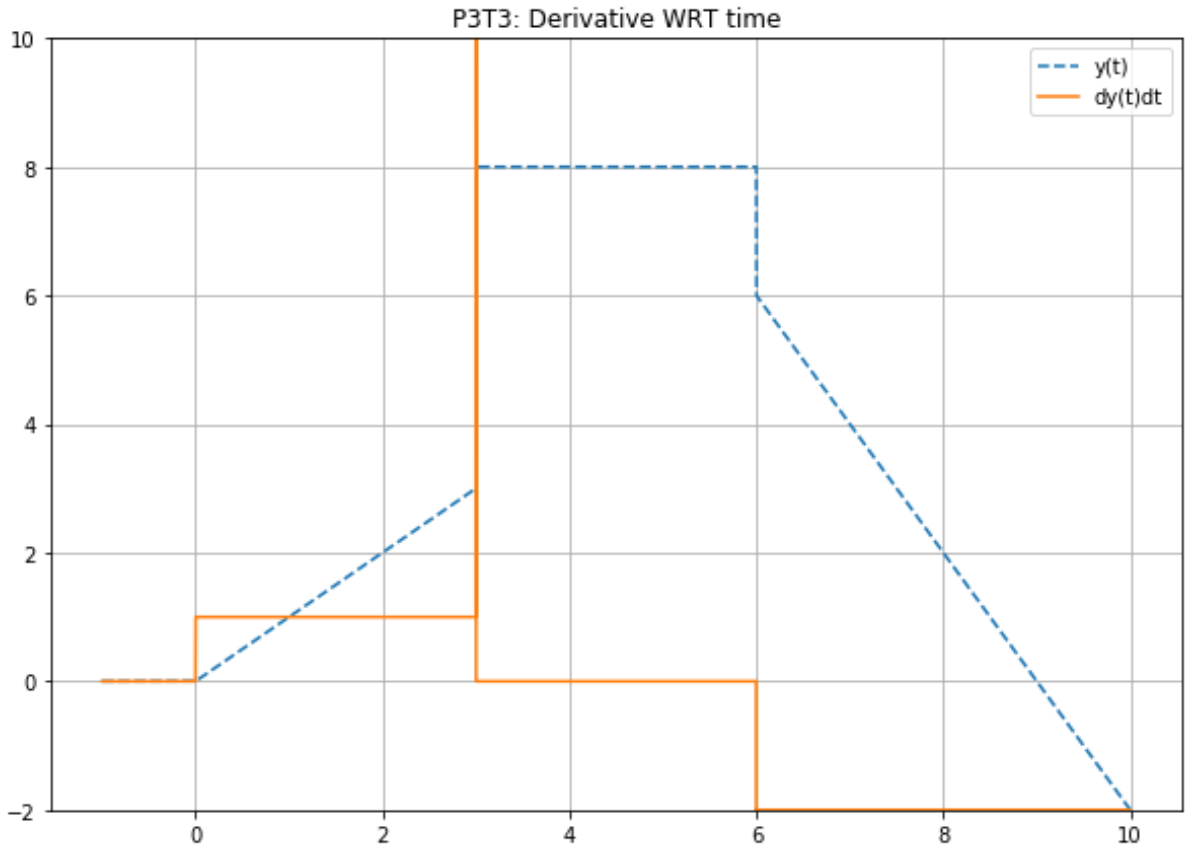
The next two plots show half time scaling, f(t/2), and double time scaling, f(2t).



The following graphic is a plot of the derivative of f(t) that I graphed using draw.io.



Finally, the following graphic is the derivative of f(t) calculated using the numpy.diff() function and plotted using matplotlib.pyplot.


P3T3: Derivative WRT time
y(t)
dy(t)dt
10
8
6
4
2
0
−2
0
2
4
6
8
10

Questions

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

The derivative plot I created and the derivative plot created via my Python script are not quite the same. Rather than draw the undefined parts as infinitely tall, I chose to represent them as discontinuities with dotted lines. While the conventional use of the numpy.diff() class will not result in this, I suppose it would be possible to create some sort of wrapper function around it to recognize undefined points and plot it as I did.

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

If the steps are assigned such that the plot is not sampled at undefined points, there will be no undefined points. For example, an undefined point occurs at t=3, so if I set the steps to .3 instead of 1e-3, the undefined point is stepped over. Though this means there are no undefined points now, the resolution is very poor and the plot of y(t) is no longer correct.

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

Though some of the tasks seemd pretty daunting at first, the hints and examples provided throughout the lab session were incredibly helpful. The tasks were laid out in a clear and concise manner.