ECE 351 Lab 2

Zachary DeLuca

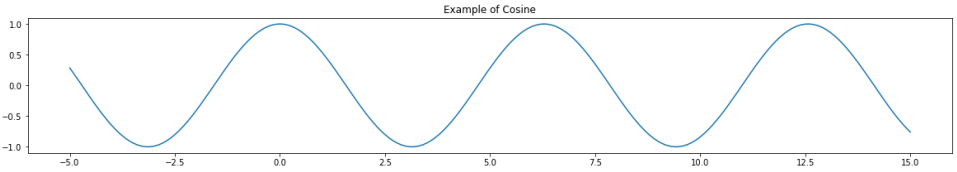
January 31st 2023

1 Introduction

In this lab we will use python to implement and graph functions that store there values into arrays. We can increase the definition of these graphs by increasing the size of the arrays that store the points. We will use step and ramp functions to build a graph based on an example, and then elaborate on that initial function. I have separated this report into two parts; the intro cosine example and all of the things we do with the given graph.

2 Cosine Example

The first part of the lab is to graph a cosine function by creating a user function then store points into parallel arrays. Once those arrays are full, we can plot them using the plt library, like so:



The code that defines this process will be attached in the .py file

3 Steps and Ramps

The majority of this lab is focused in this part, as it deals with the given graph and how to formulate, manipulate, and translate the graph. The code associated with each of the steps will be included in the .pyfile

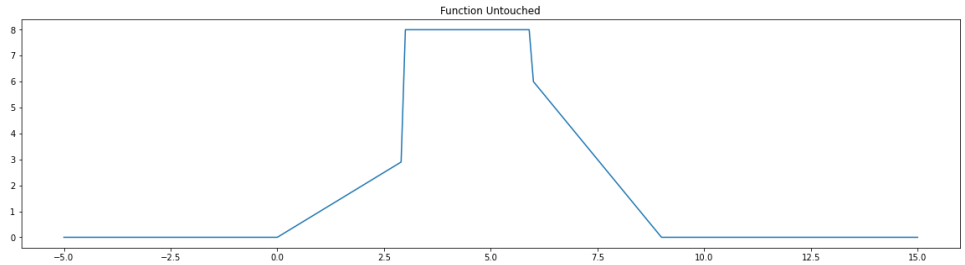
3.1 Formula

The formula for the graph can be expressed as such:

$$f(t) = r(t) - r(t-3) + 3u(t-3) - u(t-6) - 3r(t-6) + 3r(t-9)$$

3.2 Graph Replicated

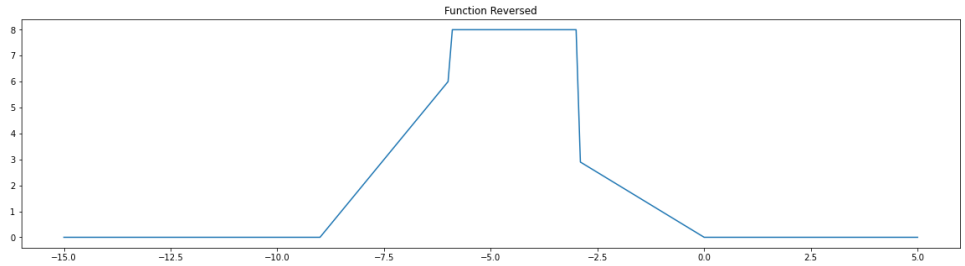
Using the formula above and the python equivalent of steps and ramp functions, the graph is replicated below:



3.3 Graph Manipulated

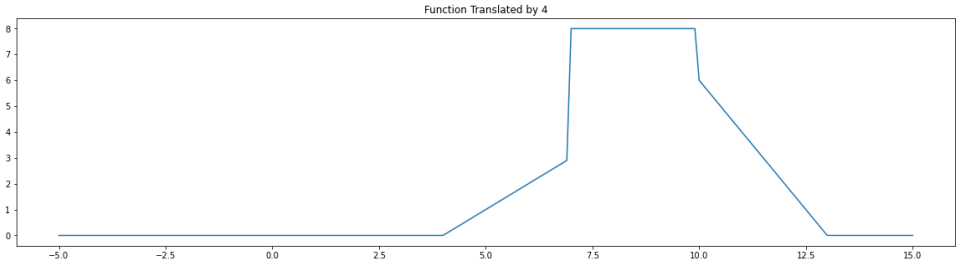
The rest of the lab asked for variations upon the given graph, so the first example is graphed with the following inversion:

$$f(-t)$$



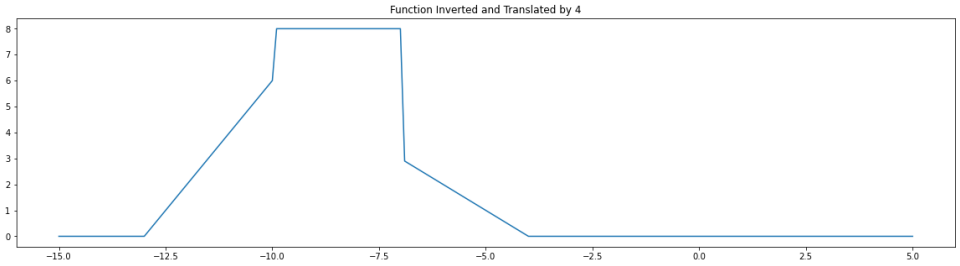
The next is the example of the graph shifted by four:

$$f(t-4)$$



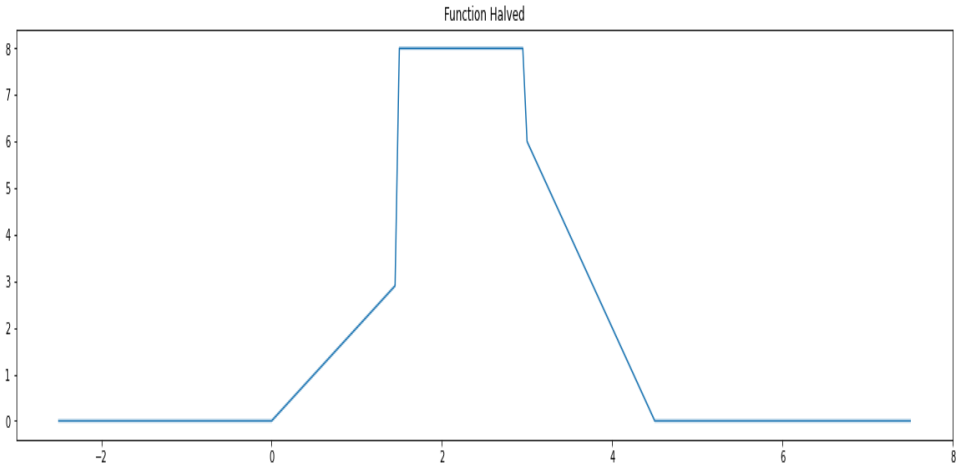
The next example is that of the function inverted and shifted 4:

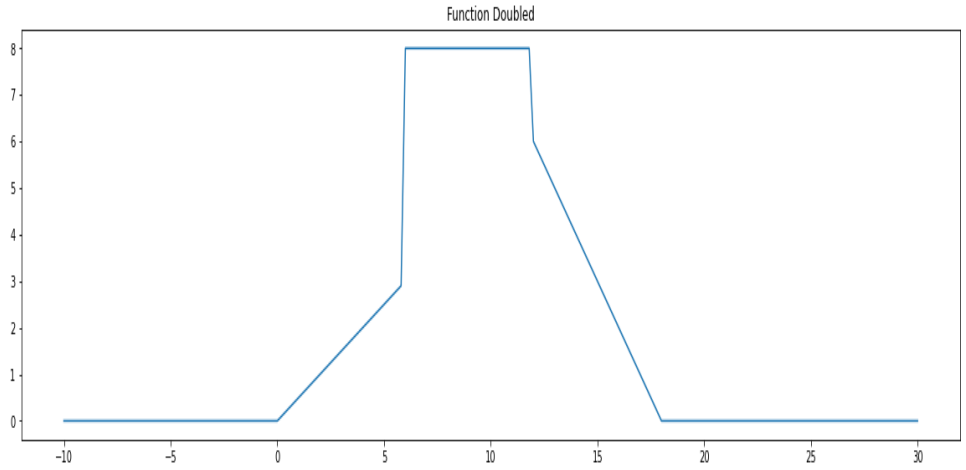
$$f(t) = (-t - 4)$$



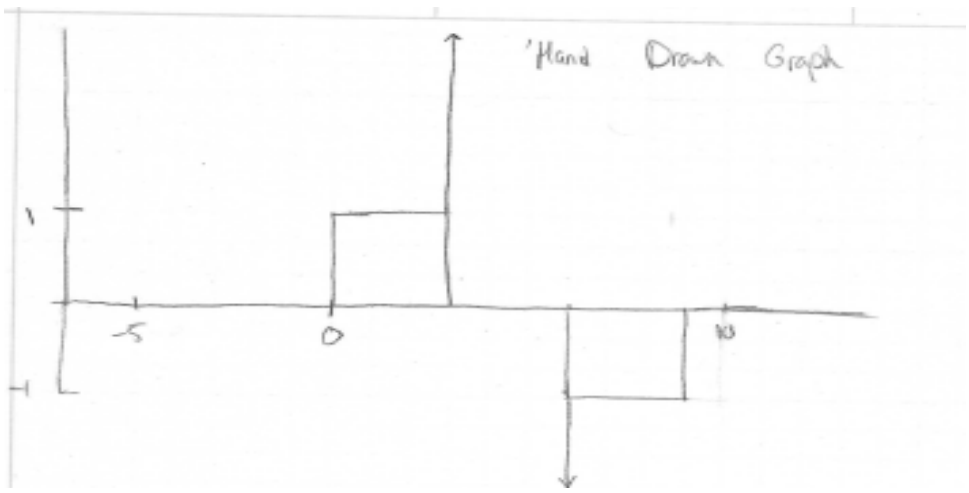
The next couple of graphs are what happens if you halve or double the function like so:

$$f(2t) and f(t/2)$$



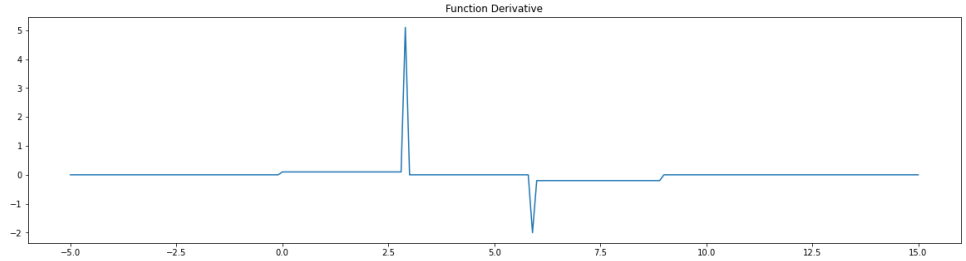


The last set of graphs are that of the derivative graphed. The first is a hand drawn graph:



The arrows indicate that the derivative at the point is technically infinite as a step function is discontinuous at the point it steps.

The next graph is the derivative of the graph using the np.diff function:



The graph gets smashed during all the interesting parts as the program tries to interpret the infinite spikes where the step functions are.

4 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.

They are identical in the sense they have the same shape, but are not in the sense that they are not in the same place or have the same orientation. The object of the graph is the same throughout just translated or mirrored.

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.

The graphs are basically identical, as they are just stretched or compressed, and as the graph auto fits the graph, they plot the same. If the step size was too small, the doubled expression would retain its shape whereas the halved would begin to loose clarity as the step size shrinks
3. Leave any feedback on the clarity of lab tasks, expectations, and deliverables.

This lab in particular was massively confusing on what each of the numbered items was referring to. I still don't know if I included what I needed to.

5 Conclusion

In this lab we were exposed to the numpy functions and the graphing functions. We were able to use arrays and user defined functions to graph a piece-wise function that would be usually a massive pain to graph otherwise. The only way to graph the function without the step and ramp functions would be a massive nested if statement that would be fraught with errors. Once the functions were functional and implemented creating the subsequent manipulated graph was not difficult. These graphical functions will be useful for the labs to come.