ECE351-51: LAB 3

SEPTEMBER 23, 2019

Submitted By:

Adriana Oliveira

INTRODUCTION

This lab expanded on our understanding of convolution properties. This was achieved by using previously created functions to define three equations. Two equations were then used as parameters in a user defined function that found their convolution.

METHODOLOGY

Part 1: Create Equations

We started by adding our peripheral libraries and pasting our previously defined step and ramp functions from Lab 2. These were nested in new functions to describe a set of equations.

Required equations:

$$f_1(t) = u(t-2) - u(t-9)$$
$$f_2(t) = e^{-t}u(t)$$
$$f_3(t) = r(t-2)[u(t-2) - u(t-3)] + r(4-t)[u(t-3) - u(t-4)]$$

Listing 1: Functions describing required equations

```
def f1(t):
    a = np.zeros((len(t), 1))
    for i in range(len(t)):
        a[i] = u(t[i]-2)-u(t[i]-9)
    return a
def f2(t):
    b = np.zeros((len(t), 1))
    for i in range(len(t)):
        b[i] = np.exp(-t[i])*u(t[i])
    return b
def f3(t):
    c = np.zeros((len(t), 1))
    for i in range(len(t)):
        c[i] = r(t[i]-2)*(u(t[i]-2)-u(t[i]-3))+r(4-t[i])*(u(t[i]-3)-u(t[i]-4))
    return c
```

These equations were plotted in subplots from t = 0s to t = 20s with steps of .1.

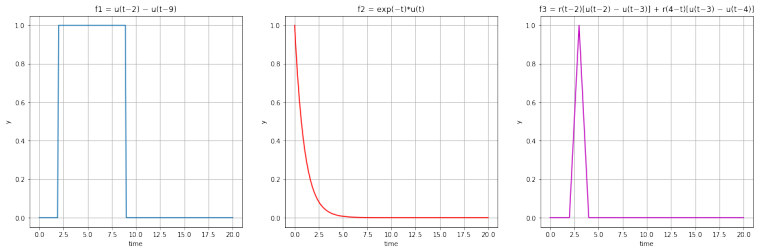

Figure 1: Plots for $f_1(t)$, $f_2(t)$, and $f_3(t)$

Part 2: Create a Convolution Function

To find a convolution, we multiply two function together over continuous time. This means multiplying the elements of both function at the same point in time and adding the results together as time continues. This integral results in a solution that is the size of the two functions added together minus one to account for them overlapping at zero.

Using our defined equations as parameters, we created our own convolution function. This was then tested using scipy.signal.convolve() from the scipy.signal library to confirm our code. Creating functioning code was a group effort and resulted in many drafts.

Listing 2: Revised convolution

```
#User defined function for convolutions
def my_convo(f1,f2):
    length_f1 = np.size(f1)  #size of first function
    length_f2 = np.size(f2)  #size of second function

    #Result with be as large as the added lengths of the functions
    #-1 because they share t = 0 on the timeline.

    result = np.zeros(length_f1 + length_f2 -1)

    for m in np.arange(length_f1):        #for any time in the f1,
        for n in np.arange(length_f2):    #take the same time in f2.
            result[m+n] = result[m+n] + f1[m]*f2[n]
            #multiply these points and add with previous results.
            #This produces an integral

    return result
```

RESULTS

The user defined convolution function was able to produce the same results as scipy.signal.convolve().

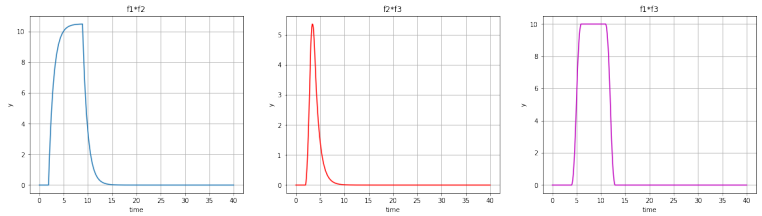

Figure 2: Plots for Convolutions using scipy.signal.convolve()

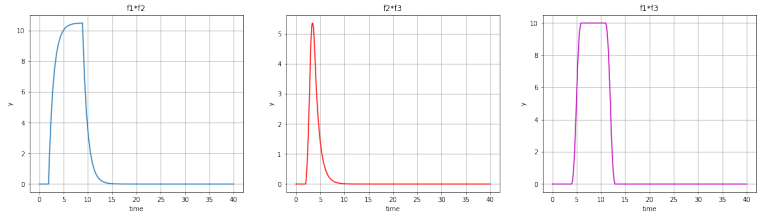

Figure 3: Plots for convolutions using user defined function

QUESTIONS

1. Did you do this lab alone or with classmates? If you collaborated to get to the solution, what did that process look like?

   I worked with classmates on the white board to hammer out how a convolution functions. This involved trying to come up with the right array size for the time and result. We also discussed what the for loops for the result needed in order to execute our integral properly.

2. What was the most difficult part of this lab for you, and what did the process of figuring it out look like?

   The most difficult part was understanding how to create an array the proper size for the result of the convolution and time. This involved a bit of trial and error as well as assistance from the TA.

3. Did you approach writing the code with analytical or graphical convolution in mind? Why did you chose this approach?

   I went with an analytical approach that looked at recreating the integral of two functions multiplied together. I chose this approach because it was the most intuitive for me and because my computer is great at arithmetic.

4. Was any part of this lab not clearly explained?

   Since we were allowed to work together, this lab gave enough instruction to produce proper results.

CONCLUSION

Using the skills built from previous labs, we were able to create a user defined convolution function that looked at two functions at the same point in time, multiplied them, and added the results while continuing in time.

In the process of creating a convolution function, we were able to clarify our understanding of convolutions as well as learn new properties in numpy such as size. This looks at how many elements are in an array despite dimension. I also learned that my for loop's complexity could be reduced if preceded by a line that describes the result as an array of zeros the length of the two function's lengths added minus one for overlap.