ECE 351-51

February 8, 2022

Lab 3

Submitted By :
Kennedy Beach

Contents

1 Introduction 2
2 Equations 2
3 Methodology 2
4 Results 4
5 Error Analysis 5
6 Questions 5
7 Conclusion 6

1 Introduction

The purpose of this lab was to gain a greater understanding of convolution by creating a user-defined function in Python that would convolve two functions.

2 Equations

The following signal equations were used to create functions in Python:

$$f1(t) = u(t-2) - u(t-9)$$

$$f2(t) = e^{-t} * u(t)$$

$$f3(t) = r(t-2)[u(t-2) - u(t-3)] + r(4-t)[u(t-3) - u(t-4)]$$

3 Methodology

The first part involved plotting the three signals given above from 0 to 20. Listing 1 provides the necessary code for the user-defined functions as well as the ramp and step functions used previously. The plots are seen in the Results section.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig

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

def f1(t):
    y = np.zeros((len(t), 1))
    for i in range(len(t)):
        y[i] = u(t[i]-2) - u(t[i]-9)
    return y

def f2(t):
    y = np.zeros((len(t), 1))
    for i in range(len(t)):
```

```
        y[i] = np.exp(-t[i])*u(t[i])
    return y

def f3(t):
    y = np.zeros((len(t), 1))
    for i in range(len(t)):
        y[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 y

steps = .01
t = np.arange(0, 20 + steps, steps)

y1 = f1(t)
y2 = f2(t)
y3 = f3(t)
```

Listing 1: Defining the functions to plot the three signals from $0 < t < 20$

The user-defined convolution function is given below in Listing 2. The specifics of how the code works is described in comments throughout the listing. It essentially takes the length of the two input functions and makes two arrays of the same length. A separate array of the same length is made to have the results put into it. A for loop then goes through the length of the combined lengths of the two functions, and combines the previous results with the product of the two entries of the functions.

```
def my_conv(f1, f2):
    Nf1 = len(f1)           #variable with the length of f1
    Nf2 = len(f2)           #variable with the length of f2

    f1Ex = np.append(f1, np.zeros((1, Nf2-1)))  #creates an array that is the same size as f1 and f2
    f2Ex = np.append(f2, np.zeros((1, Nf1-1)))

    result = np.zeros(f1Ex.shape)   #creates a zero-filled array the same size as both functions

    for i in range((Nf2+Nf1-2)):    #goes through the length of f1 and f2
        result[i] = 0
        for j in range(Nf1):        #goes through the length of f1
            if (i-j+1 > 0):         #makes sure the loop doesn't go past 0 entries
                try:
                    result[i] = result[i] + f1Ex[j]*f2Ex[i-j+1]
    #combines the previous results with the product of the new entries
                except:
                    print(i,j)
```

```
18      return result
19
20  time = len(t);
21  tEx = np.arange(0, (2*t[time-1]) + steps, steps)
22
23  a = my_conv(y1, y2)*steps
24  b = my_conv(y2, y3)*steps
25  c = my_conv(y1, y3)*steps
26
27  conv1 = sig.convolve(y1, y2)*steps
28  conv2 = sig.convolve(y2, y3)*steps
29  conv3 = sig.convolve(y1, y3)*steps
```

Listing 2: Defining the function for convolution

4 Results

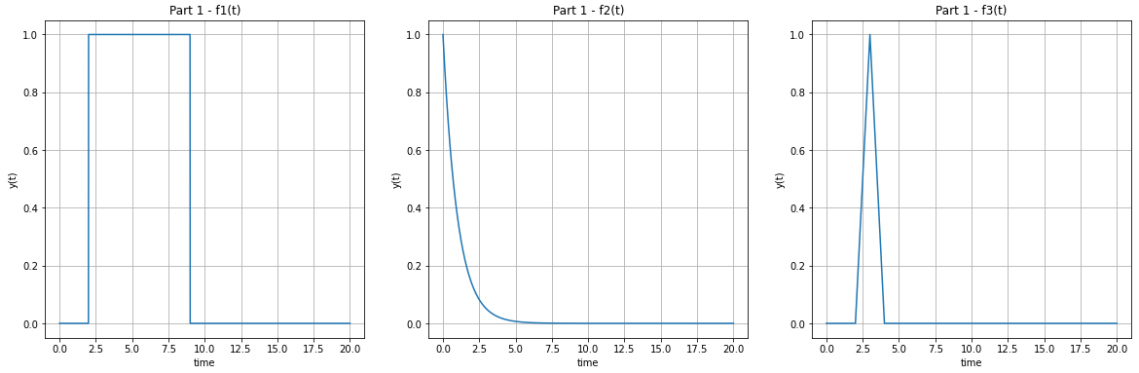

Figure 1: Plots of the three initial given sigals

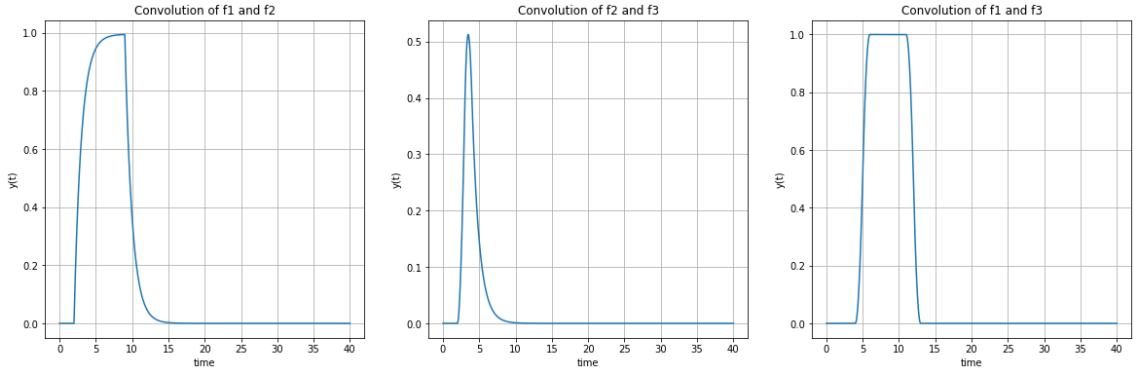

Figure 2: User-defined convolutions of the three signals

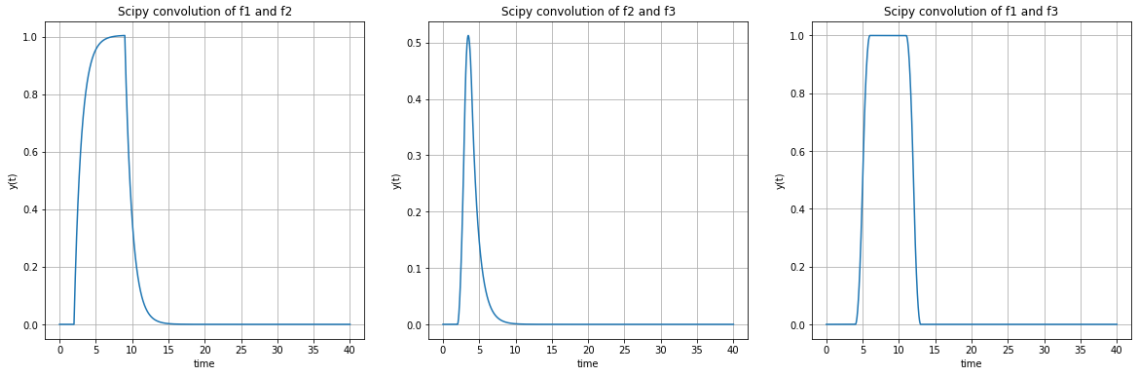

Figure 3: Convolutions using the scipy.signal library

5 Error Analysis

I had difficulty getting the time for the convolution to work. There were too many x values per y value with the initial code I came up with. After consulting the internet, I had come up with a working solution.

6 Questions

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

I mostly worked alone. There was a little collaboration when the TA went over what the user-defined function for convolution could look like, but that was mostly demonstration versus discussing possible options.

2. What was the most difficult part of this lab for you, and what did your problem-solving process look like?

The most difficult part was coming up with the convolution code initially. Once the TA explained the process and more about what convolution is actually doing, it became easier. The next hardest thing was to make sure the time for the convolutions worked properly. I struggled a bit on my own by trying different options, then I had to consult the internet for possible solutions.

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

I had chosen to approach this problem with an analytical solution in mind since

that is how I first learned of convolution.

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

The lab tasks, expectations, and deliverables were all very clear. It was how to go about some of the tasks that was difficult at times, but that was remedied by consulting the TA and other resources.

7 Conclusion

I learned more about how convolution actually works through making my own convolution function. During convolution, the two input functions are multiplied where they occur during the same time. It shows how much overlap the two functions have when one is moved over the other. There were some struggles, but these were overcome eventually. The Python and LaTeX code are seen in `https://github.com/Eniac618/ECE351_Code` and `https://github.com/Eniac618/ECE351_Reports` respectively.