ECE 351 - 52

Signals and Systems 1

Lab 3

Submitted By :
Owen Blair

Contents

1 Important Notes

It is important to note that plt.show() is needed at the end of the python code to properly show the plots of each function. It was not included in every section of code that plots a function(s) because it is assumed that its included at the end the code.

The web address to the GitHub where LaTeX code is stored is here:
https://github.com/Blairis123/ECE351_Reports

The web address to the GitHub where the Python code is here:
https://github.com/Blairis123/ECE351_Code

2 Part 1 Deliverables

The goal of part 1 is to use the step and ramp functions to create the following three functions:

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

These functions are plotted in Figure 1 located in the Results section. The code used to create these functions and plot them can be seen in Listing 1 under the Code heading.

3 Part 2 Deliverables

Part 2 was about convolving the the functions from part 1. This starts with a user defined function that convolves two arrays. The colvolve(arry1, arry2) function code can be seen in Listing 2 under the Code section. The function takes 2 inputs, the two functions to convolve and will return an array that is larger than the input arrays. The returned array will have a length of twice the length of the input arrays. This section includes plotting the convolutions of $f_1 * f_2$, $f_1 * f_3$, and $f_2 * f_3$. The plots of these convolutions using the user defined function compared to the library can be seen in Figure 2.

4 Results

Figure 1: Plots of f_1, f_2, f_3

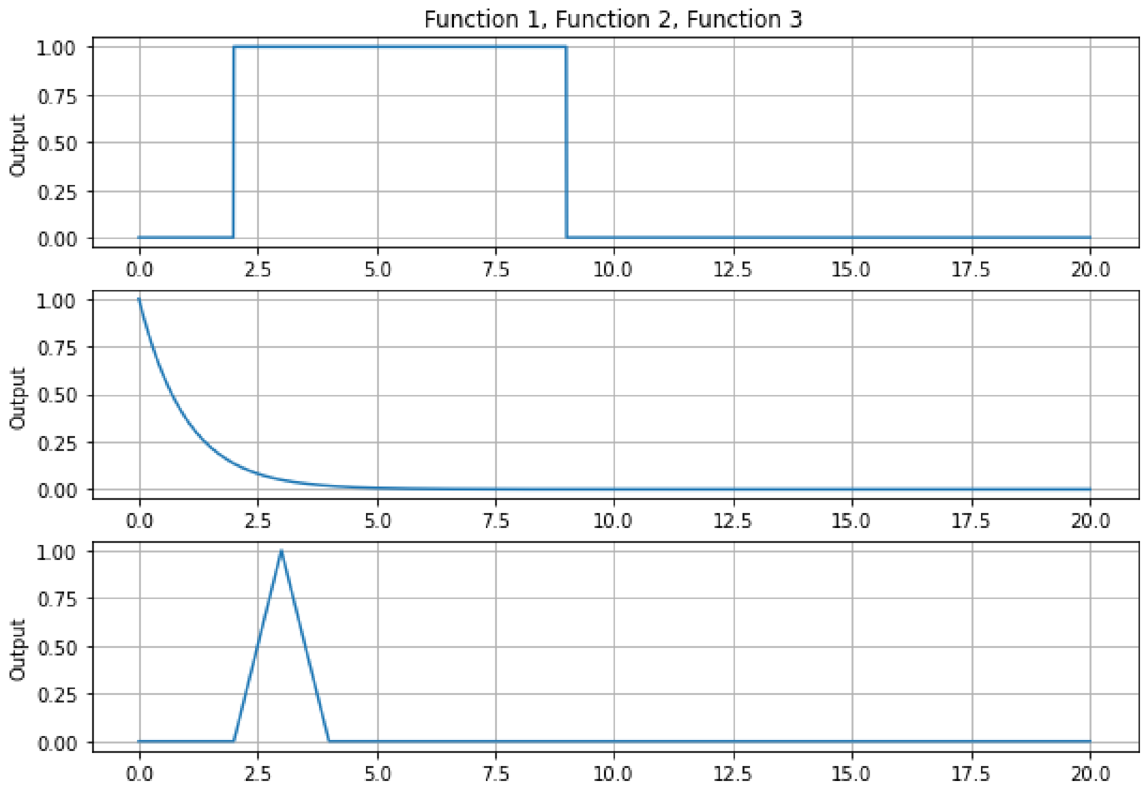
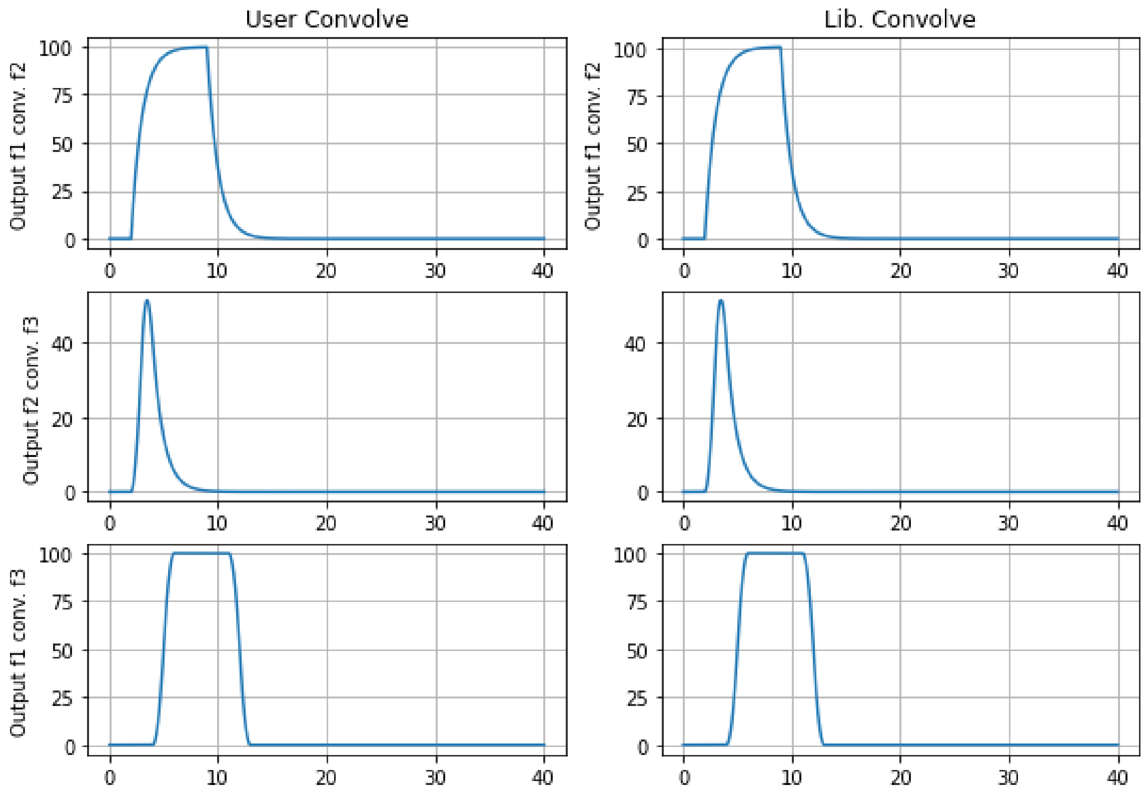


Figure 2: User Defined Convolve vs. Library Function



5 Code

Listing 1: f_1, f_2, and f_3 Creation and Plotting

```
    #Define step size
steps = 1e-2

    #Define a range of t. Start @ 0 go to 10 (+a step) w/ a
   stepsize of step
t = np.arange(0, 20+ steps, steps)

#Make stuff to plot!
func1 = stepFunc(t,2,1) - stepFunc(t, 9, 1)
func2 = eExpo(t,0,1,-1)
func3 = (rampFunc(t,2,1) * (stepFunc(t, 2, 1) - stepFunc(t, 3, 1)))
    + ((rampFunc(t, 3, -1) + 1) * (stepFunc(t, 3, 1) - stepFunc(t,
   4, 1)))

    #Make plots and then show it
```

```python
plt.figure(figsize=(10,7))
plt.subplot(3,1,1)
plt.plot(t,func1)
plt.grid()
plt.ylabel('Output')
plt.title('Function 1, Function 2, Function 3')

plt.subplot(3,1,2)
plt.plot(t,func2)
plt.grid()
plt.ylabel('Output')

plt.subplot(3,1,3)
plt.plot(t,func3)
plt.grid()
plt.ylabel('Output')
```

Listing 2: Convolve Function Code

```python
#Convolution function!!!
def convolve(f1, f2):

    #Both functions need to be the same size!
    Nf1 = len(f1)
    Nf2 = len(f2)

    #Debug statements
    #print(Nf1)
    #print(Nf2)

    f1Extend = np.append(f1,np.zeros((1,Nf2-1)))
    f2Extend = np.append(f2,np.zeros((1,Nf1-1)))

    y = np.zeros(f1Extend.shape)

    for i in range(Nf1 + Nf2 - 2):
        y[i] = 0

        for j in range(Nf1):
            if (i-j+1 >0):
                try:
                    y[i] += f1Extend[j] * f2Extend[i-j+1]

                #Where am I running out of space in my array?
                except:
                    print(i,j)
    return y
```

Listing 3: Plotting User Defined Convolutions vs. Library Functions

```
    #Define step size
steps = 1e-2

    #Define a range of t. Start @ 0 go to 10 (+a step) w/ a
   stepsize of step
t = np.arange(0, 20+ steps, steps)

#Make stuff to plot!
func1 = stepFunc(t,2,1) - stepFunc(t, 9, 1)
func2 = eExpo(t,0,1,-1)
func3 = (rampFunc(t,2,1) * (stepFunc(t, 2, 1) - stepFunc(t, 3, 1)))
     + ((rampFunc(t, 3, -1) + 1) * (stepFunc(t, 3, 1) - stepFunc(t,
   4, 1)) ) #x * y

#Make convolutions by hand!
f1Convolvef2 = convolve(func1, func2)
f2Convolvef3 = convolve(func2, func3)
f1Convolvef3 = convolve(func1, func3)

#Make convolutios using scipy.siganl.convolve!
f1Conf2Lib = sig.convolve(func1, func2)
f2Conf3Lib = sig.convolve(func2, func3)
f1Conf3Lib = sig.convolve(func1, func3)

#Make a t range to pot the convolve functions!
#This should be the same as the size for all convolutions for this
   lab
tConv = np.arange(0, len(f1Convolvef2) * steps, steps)

#New plot with subplots for convolve!
    #f1 convolve f2
plt.figure(figsize=(10,7))
plt.subplot(3,2,1)
plt.plot(tConv, f1Convolvef2)
plt.grid()
plt.ylabel("Output f1 conv. f2")
plt.title("User Convolve")

plt.subplot(3,2,2)
plt.plot(tConv, f1Conf2Lib)
plt.grid()
plt.ylabel("Output f1 conv. f2")
plt.title("Lib. Convolve")

    #f2 convolve f3
plt.subplot(3,2,3)
plt.plot(tConv, f2Convolvef3)
plt.grid()
plt.ylabel("Output f2 conv. f3")

plt.subplot(3,2,4)
plt.grid()
plt.plot(tConv, f2Conf3Lib)

    #f1 convolve f3
plt.subplot(3,2,5)
plt.plot(tConv, f1Convolvef3)
plt.grid()
plt.ylabel("Output f1 conv. f3")

plt.subplot(3,2,6)
plt.grid()
plt.plot(tConv, f1Conf3Lib)
```

6 Questions

6.1 Question 1

I worked with some of my classmates for how to make the convolve function. The process of collaboration was a discussion between three of us about how to go about making the convolve function. First, the convolve function was broken down into several smaller pieces and those were discussed first. The solving of one small problem at a time continued until the answer was given to us by the TA. The group I was working with came fairly close but wasn't all of the way to a solution.

6.2 Question 2

The most difficult part of this lab was getting my function 3 correct. A time reversal of my function was not interacting correctly with the multiplication of the $[ut-3-ut-4]$ *part to make the falling edge of the function. My problem-solving process consisted of trying the most obvious problems and working towards the least obvious problems. For example, the first thing I tried was breaking apart function 3 and trying to do it step by step and examining the variable using the variable viewer tool. This allowed me to see what was going wrong and where. Eventually I found an equivalent solution of matching a ramp function with a -1 and adding 1 to each value of the array, then multiply by* $[ut-3-ut-4]$.

6.3 Question 3

I approached writing the convolution with a graphical convolution in mind. I did this because I can better picture what is gong on when I can draw a picture. Using a graphical approach I could draw out what several steps should do on paper before trying to code them. This was really useful when comparing what I have drawn to what the output was for my function.

6.4 Question 4

This lab seemed to be quite a bit shorter than the previous lab. I also really like how previous code written for labs was used in this lab (for better or worse). The collaborative aspect of this lab was also helpful for my understanding because I needed to understand what my perspective was before I could explain it to someone else. I think labs in the future should be more collaborative in the future.