Project Report for ECE 351

Lab 03 - Discrete Convolutions

Skyler Corrigan

September 8, 2022

ECE351 Code Repository:
$https://github.com/ElfinPeach/ECE351_Code.git$

ECE351 Report Repositry:
$https://github.com/ElfinPeach/ECE351_Report.git$

Contents

1 Part 1 **1**

2 Part 2 **3**

3 Questions **7**

3.1 Question 1 7

3.2 Question 2 7

3.3 Question 3 8

3.4 Question 4 8

1 Part 1

Part 1 is creating plots for three different equations. The equations are as follows:
$f_1(t) = u(t2)u(t9)$
$f_2(t) = e^-tu(t)$
$f_3(t) = r(t2)[u(t2)u(t3)] + r(4t)[u(t3)u(t4)]$

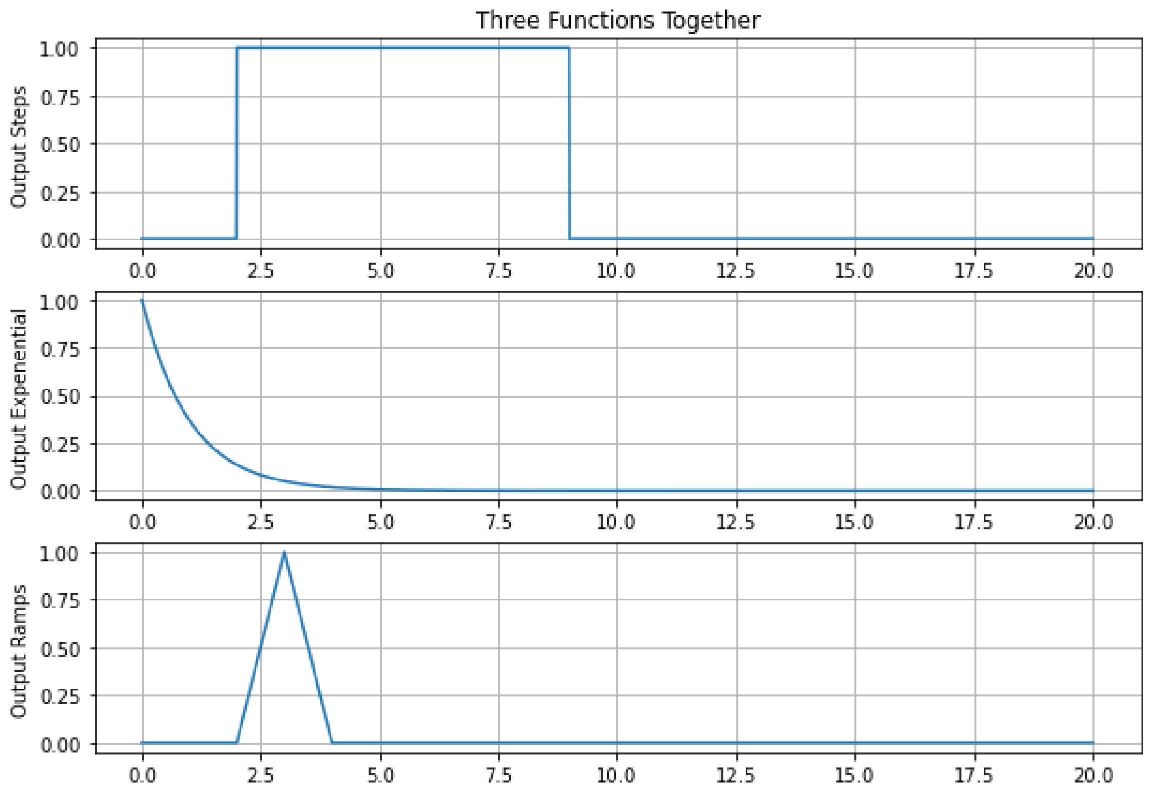
These functions were made with the following code:
Function Code

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

#----------------FUNCTIONS----------------------------------------------------#

#Make a step function using an array t, stepTime, and stepHeight
def Step(t, startTime, stepHeight):
    y = np.zeros(t.shape)
    for i in range(len(t)):
        if(t[i] >= startTime):
            y[i] = stepHeight
    return y

#Make a ramp function using an array t, startTime, and slope
def Ramp(t, startTime, slope):
    y = np.zeros(t.shape)
    for i in range(len(t)):
        if(t[i] >= startTime):
            y[i] = t[i]-startTime
    y = y * slope
    return y

#Make e^at function using array t, startTime, and alpha
def eExpo(t,startTime,amplitude,alpha):
    y = np.zeros(t.shape)
    for i in range(len(t)):
        if(t[i] >= startTime):
            y[i]=amplitude * math.exp(alpha * (t[i]-startTime))

    return y

#------------------END FUNCTIONS------------------------------------#

#------------------START DEFINITIONS--------------------------------#
```

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

#Define a range of t. Start at 0 and go to 20 w/ a StepSizeize of step
t = np.arange(0, 20+ StepSize, StepSize)

#---------------END DEFINITIONS----------------------------------#

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

#Three Figures Plot
plt.figure(figsize=(10,7))
plt.subplot(3,1,1)
plt.plot(t,func1)
plt.grid()
plt.ylabel('Output Step')
plt.title('Three Functions Together')

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

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

plt.show()
```

The following figure is the functions plotted in one figure.

Figure 1



As you can see by this plot, there is a step function corresponding to $f_1(t)$, an exponential function corresponding to $f_2(t)$, and a ramp function corresponding to $f_3(t)$.

2 Part 2

This section involves taking each of the previous functions and performing convolution them with each other. That is, performing the convolution of $f_1(t)$ with $f_2(t)$, $f_2(t)$ with $f_3(t)$, and $f_1(t)$ with $f_3(t)$.

To do so, I extended the arrays for each function to ensure that they were equal in length. Then I multiplied the first value of one array by the last value of the array. Then the second value of the array got multiplied by the second to last value of the second array, and so on so for.

After this the same convolutions were performed, but using the convolution function found inside of Python. Lastly, all six graphs were plotted onto the same figure. The entire code can be found below.

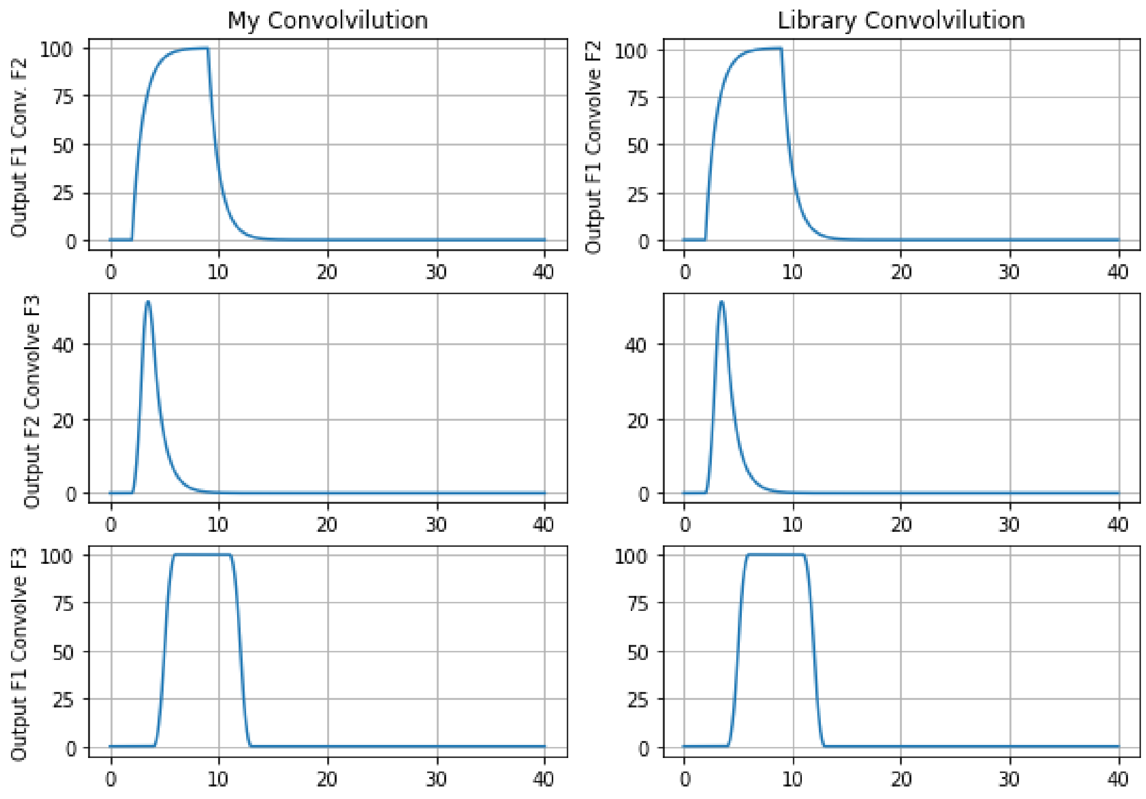
Function Code

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

#---------------FUNCTIONS------------------------------#
```

```
#Make a step function using an array t, stepTime, and stepHeight
def Step(t, startTime, stepHeight):
    y = np.zeros(t.shape)
    for i in range(len(t)):
        if(t[i] >= startTime):
            y[i] = stepHeight
    return y

#Make a ramp function using an array t, startTime, and slope
def Ramp(t, startTime, slope):
    y = np.zeros(t.shape)
    for i in range(len(t)):
        if(t[i] >= startTime):
            y[i] = t[i]-startTime
    y = y * slope
    return y

#Make e^at function using array t, startTime, and alpha
def eExpo(t,startTime,amplitude,alpha):
    y = np.zeros(t.shape)
    for i in range(len(t)):
        if(t[i] >= startTime):
            y[i]=amplitude * math.exp(alpha * (t[i]-startTime))

    return y

#Convolution function or something
def my_convolve(f1, f2):

    #Check both functions are the same size
    Nf1 = len(f1)
    Nf2 = len(f2)

    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):

        if Nf1 != Nf2: #check both funcitons same length
            print("Error")
            break
        y[i] = 0  #initialization

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

                except:
                    print(i,j)
    return y
#------------------END FUNCTIONS----------------------------------#

#------------------START DEFINITIONS------------------------------#
#Define step size
StepSize = 1e-2

#Define a range of t. Start at 0 and go to 20 w/ a StepSizeize of step
t = np.arange(0, 20+ StepSize, StepSize)

#------------------END DEFINITIONS--------------------------------#

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

#Own Convolution Function
f1Convolvef2_Mine = my_convolve(func1, func2)
f2Convolvef3_Mine = my_convolve(func2, func3)
f1Convolvef3_Mine = my_convolve(func1, func3)

#Built in Convolution Function
f1Convolvef2_Library = sig.convolve(func1, func2)
f2Convolvef3_Library = sig.convolve(func2, func3)
f1Convolvef3_Library = 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_Mine) * StepSize, StepSize)

#Convilution plots
plt.figure(figsize=(10,7))
plt.subplot(3,2,1)
plt.plot(tConv, f1Convolvef2_Mine)
plt.grid()
plt.ylabel("Output F1 Conv. F2")
plt.title("My Convolvilution")

plt.subplot(3,2,2)
plt.plot(tConv, f1Convolvef2_Library)
plt.grid()
plt.ylabel("Output F1 Convolve F2")
plt.title("Library Convolvilution")

    #f2 convolve f3
plt.subplot(3,2,3)
plt.plot(tConv, f2Convolvef3_Mine)
plt.grid()
plt.ylabel("Output F2 Convolve F3")

plt.subplot(3,2,4)
plt.grid()
plt.plot(tConv, f2Convolvef3_Library)

    #f1 convolve f3
plt.subplot(3,2,5)
plt.plot(tConv, f1Convolvef3_Mine)
plt.grid()
plt.ylabel("Output F1 Convolve F3")

plt.subplot(3,2,6)
plt.grid()
plt.plot(tConv, f1Convolvef3_Library)

plt.show()
```

The resulting figure can be seen below.

Figure 2



As shown above, my convolution is equivalent to the one that Python has.

3 Questions

3.1 Question 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 got some assistance from a friend of mine who had this class before. His advice/input was basically describing the process of multiplying the first integer in the first array by the last integer in the second array. I also worked with some people in lab mainly by describing how I did the code and helping them create there's.

3.2 Question 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 for me was trying to figure out how to extend the arrays so they were both equal. First I tried to add an array, but it wasn't quite what I was wanting. Then I Googled a bit and found the numpy.append() function, which made everything easier.
The next most difficult part is trying to get the correct part of the arrays to multiply together. To verify that I looked at the code you wrote on the board and cross checked it with mine :D.

3.3 Question 3

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

Personally I used an analytical approach. The reason why is simply: I would need to figure out some sort of algorithm to put into Python in order to actually do the lab.

3.4 Question 4

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

Aside from being difficult to wrap my head around, this lab was actually well put together in my opinion. Any clarifications for me were made in the subsequent announcement that was made with some hints on making a convolution function. Wish I looked at it before starting the lab because it would of made extending the arrays easier.