Project Report for ECE 351

Lab 3: Discrete Convolution

Isaias Munoz

September 12, 2022

University of Idaho

Contents

1 Introduction

The purpose of this lab is to graph convolutions using a user define function created and comparing it with the built in function for convolution in Python.

2 Methodology

2.1 Part 1

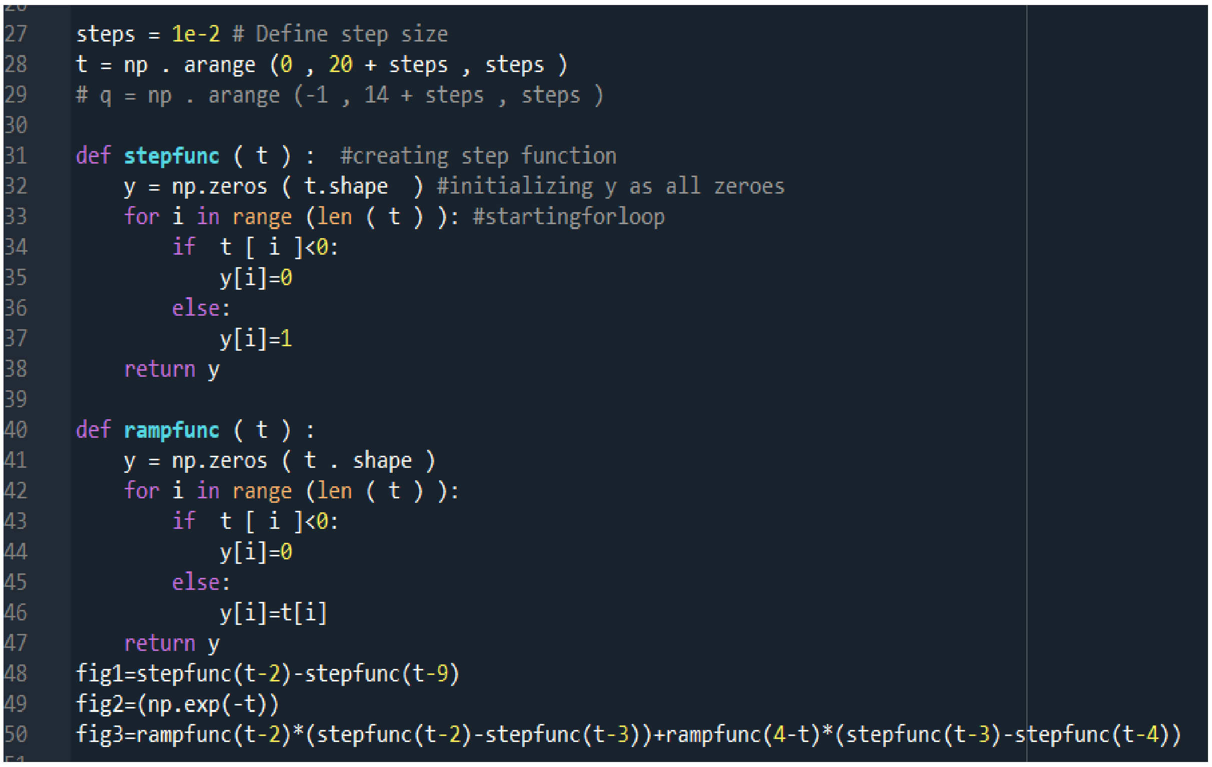
The first task that was assigned was to use Lab 2 user defined functions for a step and ramp function and write the following functions and plot them in single figure.

$$f_1(t) = u(t-2) - u(t-9)$$

$$f_2(t) = (e^-t) * u(t)$$

$$f_3(t) = r(t-2) - [u(t-2) - u(t-3) - u(t-4)]$$

Figure 1 shows the code that was used to plot the functions. It was simple enough since the user defined functions for step and ramp were already created in Lab 2 and therefore I just typed my equations and made them all as subplots to achieve one graph located in the results section. The tricky part was the exponential and how to call that in Python.

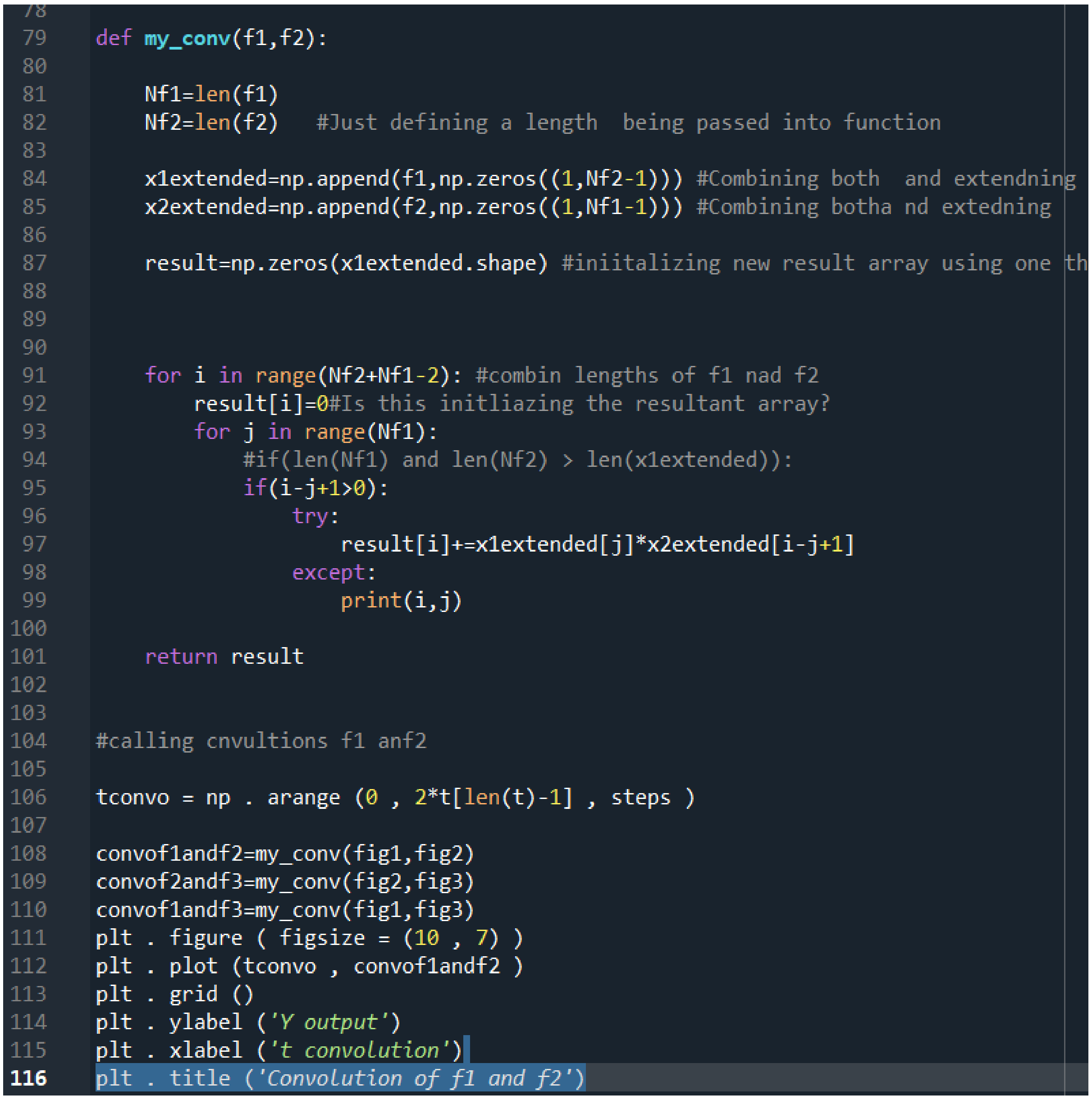


```python
steps = 1e-2 # Define step size
t = np . arange (0 , 20 + steps , steps )
# q = np . arange (-1 , 14 + steps , steps )

def stepfunc ( t ) :  #creating step function
    y = np.zeros ( t.shape  ) #initializing y as all zeroes
    for i in range (len ( t ) ): #startingforloop
        if  t [ i ]<0:
            y[i]=0
        else:
            y[i]=1
    return y

def rampfunc ( t ) :
    y = np.zeros ( t . shape )
    for i in range (len ( t ) ):
        if  t [ i ]<0:
            y[i]=0
        else:
            y[i]=t[i]
    return y
fig1=stepfunc(t-2)-stepfunc(t-9)
fig2=(np.exp(-t))
fig3=rampfunc(t-2)*(stepfunc(t-2)-stepfunc(t-3))+rampfunc(4-t)*(stepfunc(t-3)-stepfunc(t-4))
```

Figure 1: Step and Ramp user functions Code

2.2 Part 2

Part 2 was definitely the trickiest part. The task was to create a user define function in which it would perform the convolution of two functions $f_1(t)$ and $f_2(t)$. And then graph and compare with the in library convolution function. I started going the right path but in the for loops is where I got confused. Figure 2 shows the code the TA Kate provided us in achieving this convolution. The graphs are in the results section.



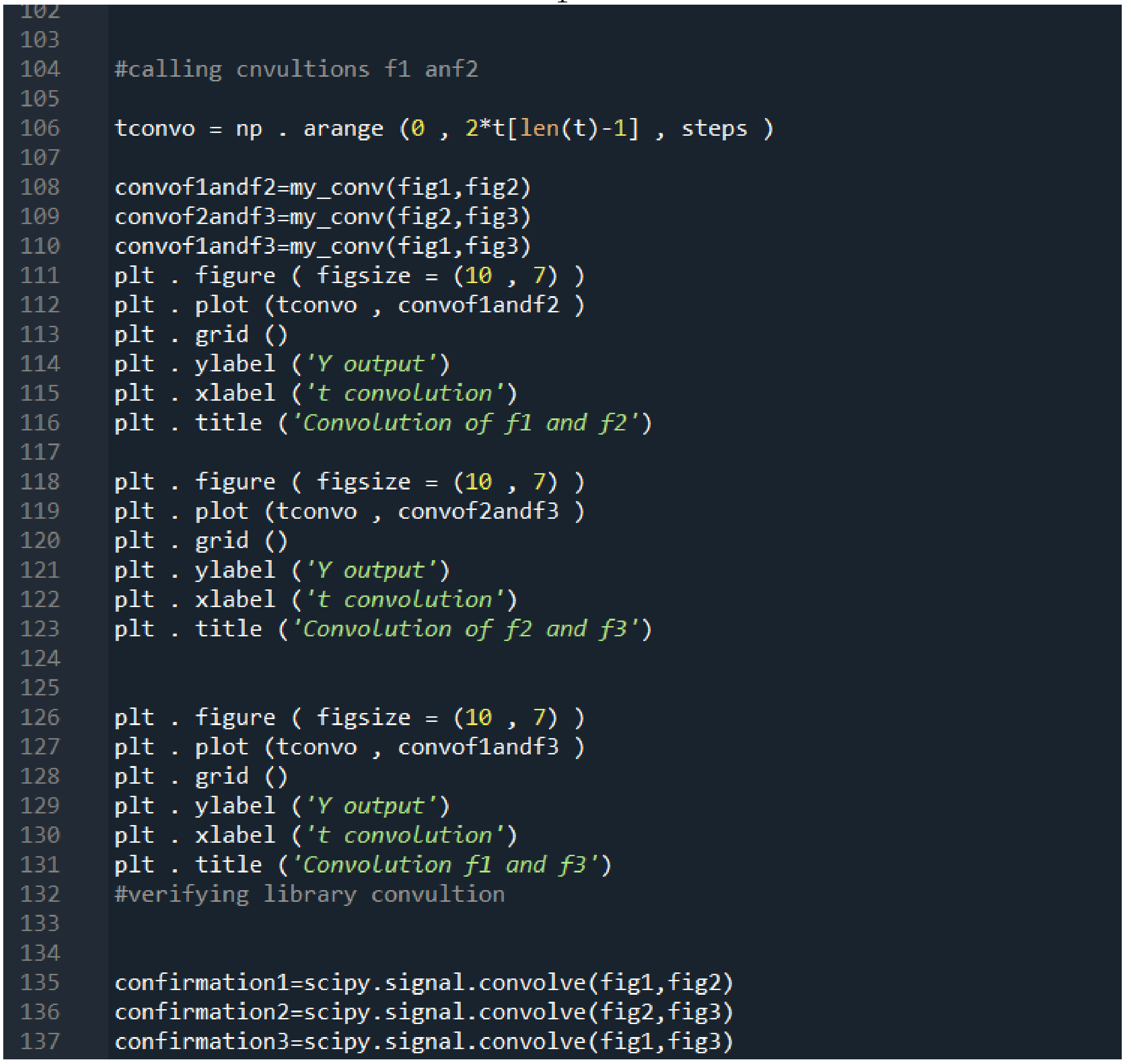
```python
def my_conv(f1,f2):

    Nf1=len(f1)
    Nf2=len(f2)   #Just defining a length  being passed into function

    x1extended=np.append(f1,np.zeros((1,Nf2-1))) #Combining both  and extendning
    x2extended=np.append(f2,np.zeros((1,Nf1-1))) #Combining botha nd extedning

    result=np.zeros(x1extended.shape) #iniitalizing new result array using one th



    for i in range(Nf2+Nf1-2): #combin lengths of f1 nad f2
        result[i]=0#Is this initliazing the resultant array?
        for j in range(Nf1):
            #if(len(Nf1) and len(Nf2) > len(x1extended)):
            if(i-j+1>0):
                try:
                    result[i]+=x1extended[j]*x2extended[i-j+1]
                except:
                    print(i,j)

    return result


#calling cnvultions f1 anf2

tconvo = np . arange (0 , 2*t[len(t)-1] , steps )

convof1andf2=my_conv(fig1,fig2)
convof2andf3=my_conv(fig2,fig3)
convof1andf3=my_conv(fig1,fig3)
plt . figure ( figsize = (10 , 7) )
plt . plot (tconvo , convof1andf2 )
plt . grid ()
plt . ylabel ('Y output')
plt . xlabel ('t convolution')
plt . title ('Convolution of f1 and f2')
```

Figure 2: Step and Ramp user functions Code

I started off with creating a function named my_conv that would accept two arrays or functions ultimately. I defined their lengths to $NF1$ and $Nf2$. Next I padded both arrays at the end with zeroes and made them both the same size. I then initialized *result* to zero this would print out the result of two for loops needed in order to pull this off.

I then set a for loop that would iterate the length of both of the functions I passed earlier and initialized $result[i]$ to zero so each time it would iterate it would be to zero. Inside that for loop was another for loop that would take into account if the value of whats in both arrays is greater then 0 then multiply their values or essentially do the calculation for a convolution and return me that. The *except* part on the code is to double check where you are screwing up and if that condition in your second for loop is not met then print out what are the values restricting that.

1



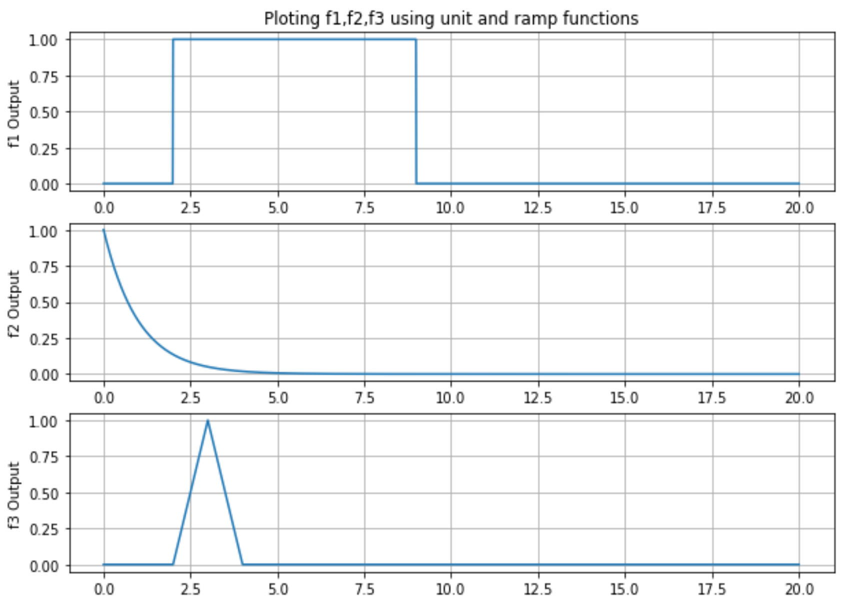
```

#calling cnvultions f1 anf2

tconvo = np . arange (0 , 2*t[len(t)-1] , steps )

convof1andf2=my_conv(fig1,fig2)
convof2andf3=my_conv(fig2,fig3)
convof1andf3=my_conv(fig1,fig3)
plt . figure ( figsize = (10 , 7) )
plt . plot (tconvo , convof1andf2 )
plt . grid ()
plt . ylabel ('Y output')
plt . xlabel ('t convolution')
plt . title ('Convolution of f1 and f2')

plt . figure ( figsize = (10 , 7) )
plt . plot (tconvo , convof2andf3 )
plt . grid ()
plt . ylabel ('Y output')
plt . xlabel ('t convolution')
plt . title ('Convolution of f2 and f3')


plt . figure ( figsize = (10 , 7) )
plt . plot (tconvo , convof1andf3 )
plt . grid ()
plt . ylabel ('Y output')
plt . xlabel ('t convolution')
plt . title ('Convolution f1 and f3')
#verifying library convultion


confirmation1=scipy.signal.convolve(fig1,fig2)
confirmation2=scipy.signal.convolve(fig2,fig3)
confirmation3=scipy.signal.convolve(fig1,fig3)
```

Figure 3: Cosine(x)

Figure 3 shows the plotting format code of the convolution after calling it and passing it the different functions mentioned earlier. What I had forgotten was that I needed a new time scale on my x-axis since that changed and that was defined as *tconvo*. This made sure I covered all of what was needed to show my convolution properly. Figure 3 also shows the convolution called using the in house function library and the results show that they both are identical.

3 Results

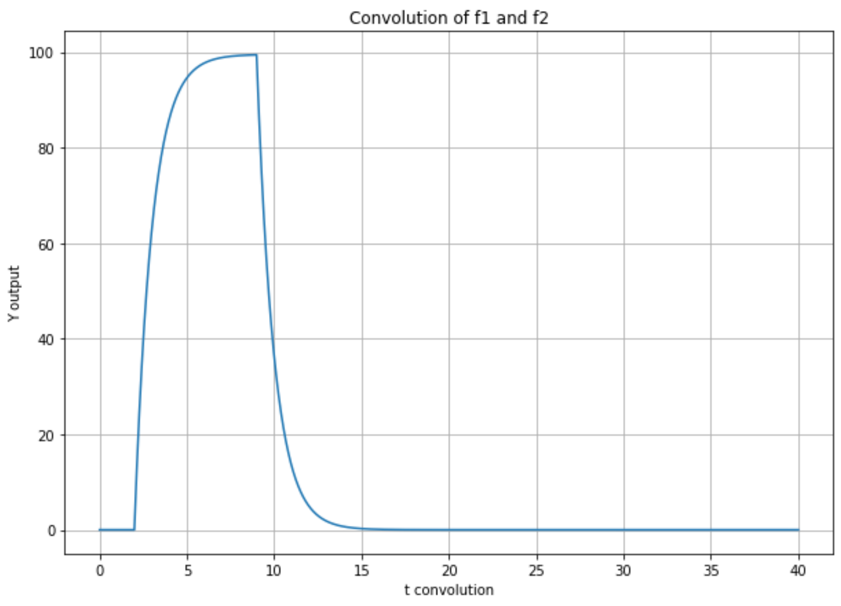
3.1 Part 1



1

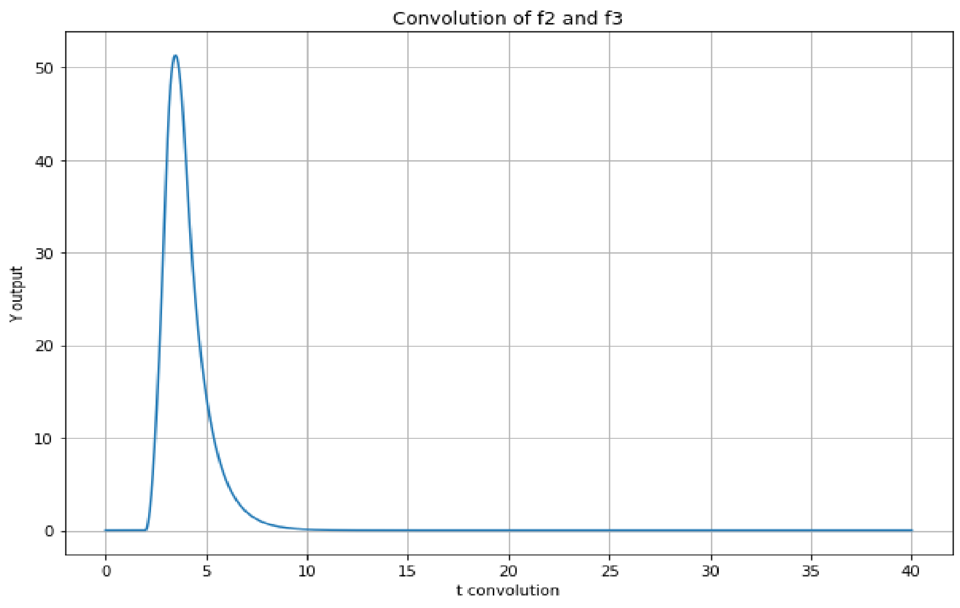
Figure 4: Combination of $f_1(t)$,$f_2(t)$,$f_3(t)$

3.2 Part 2



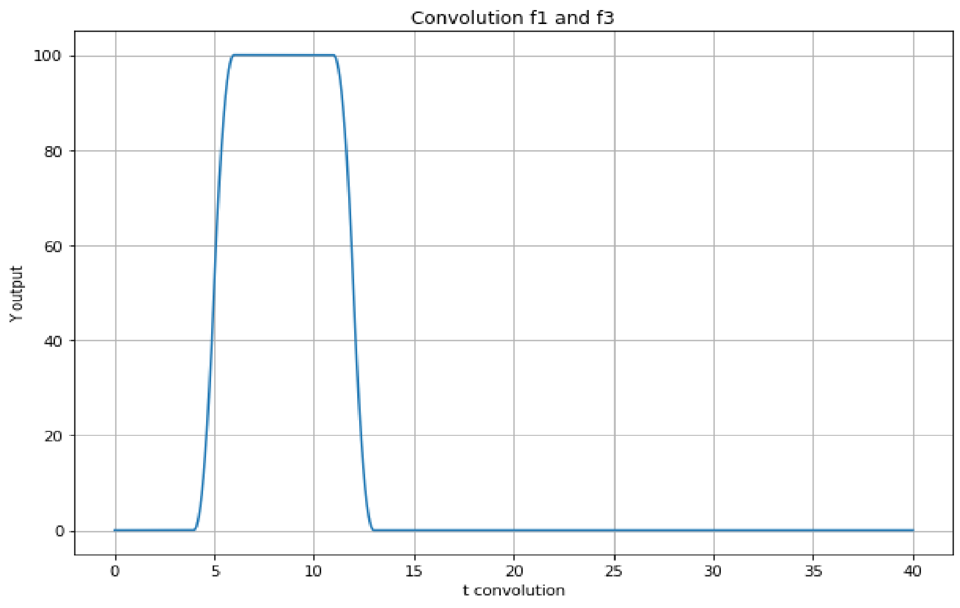
1

Figure 5: Convolution of $f_1(t)$ and $f_2(t)$



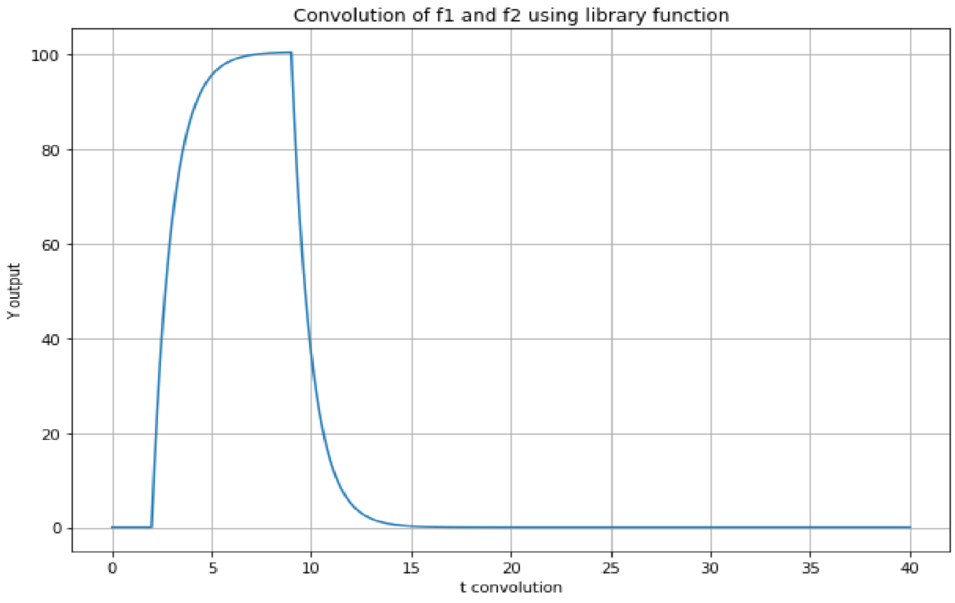
1

Figure 6: Convolution of $f_2(t)$ and $f_3(t)$



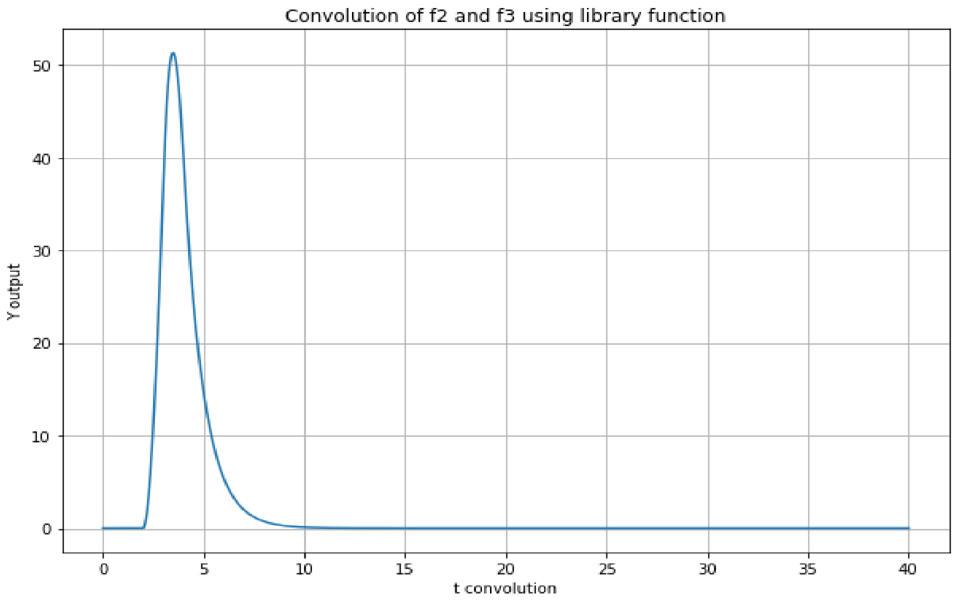
1

Figure 7: Convolution of $f_1(t)$ and $f_3(t)$



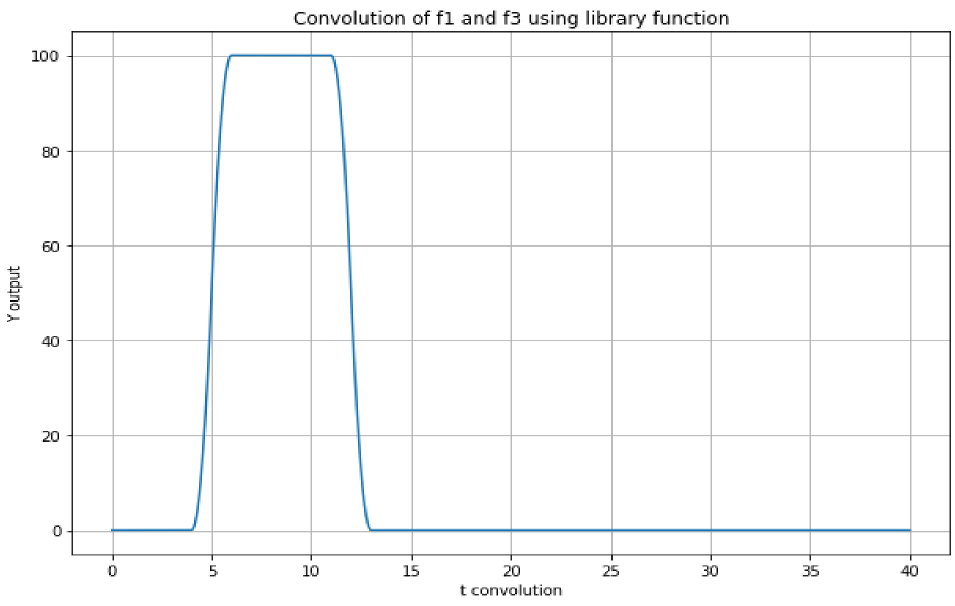
1

Figure 8: Convolution of $f_1(t)$ and $f_2(t)$ using library function



1

Figure 9: Convolution of $f_2(t)$ and $f_3(t)$ using library function



1

Figure 10: Convolution of $f_1(t)$ and $f_3(t)$ using library function

4 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 did collaborate with Skyler for guidance in setting up a new range on my time axis to graph my convolution. He pointed me in the right direction but ultimately I asked the TA for specific guidance instead. I really didn't work with anyone else after that except for the part the TA helped us with the convolution code, which it would have taken me a long time to figure out by myself.
2. What was the most difficult part of this lab for you, and what did your problem-solving process look like?
Interpreting the pseudo code, I had to look up in geeksforgeeks web-page on some of the pseudo code being given, in specific the initialization with zeroes to make the arrays the same length.
3. Did you approach writing the code with analytical or graphical convolution in mind? Why did you chose this approach?
With graphical because I am a visual learner most of the time.
4. Leave any feedback on the clarity of lab tasks, expectations, and deliverables.
It was straightforward but very hard, I really enjoyed the example given on convolution because it was graphical and that really helped me solidify with the in class portion of convolution. That was a great example, really showed what convolution means, graphically at least for me.