ECE 351-51

February 15, 2022

Lab 4

Submitted By :
Kennedy Beach

Contents

1 Introduction

The purpose of this lab was to convolve three functions using the user-defined convolution function from Lab 3 and then comparing it to hand calculations of the functions.

2 Equations

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

$$h1(t) = e^{-2t} * [u(t) - u(t-3)]$$

$$h2(t) = u(t-2) - u(t-6)$$

$$h3(t) = cos(w_0 * t) * u(t) for f_0 = 0.25Hz$$

The convolutions of the transfer functions and the step function were calculated and the results are given below:

$$y1(t) = 0.5 * [(1 - e^{-2t}) * u(t) - (1 - e^{-2(t-3)}) * u(t-3)]$$

$$y2(t) = r(t-2) * u(t-2) - r(t-6) * u(t-6)$$

$$y3(t) = (1/w) * sin(wt)u(t)$$

3 Methodology

The first part involved plotting the three signals given above from -10 to 10. 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 math

def u(t):
    if t < 0:
        return 0
    if t >= 0:
        return 1

def r(t):
    if t < 0:
        return 0
    if t >= 0:
```

```python
        return t

def h1(t):
    y = np.zeros((len(t), 1))
    for i in range(len(t)):
        y[i] = np.exp(-2*t[i])*(u(t[i])-u(t[i]-3))
    return y

def h2(t):
    y = np.zeros((len(t), 1))
    for i in range(len(t)):
        y[i] = u(t[i]-2) - u(t[i]-6)
    return y

def h3(t):
    f = 0.25
    w = f*2*np.pi
    y = np.zeros((len(t), 1))
    for i in range(len(t)):
        y[i] = math.cos(w*t[i]) * u(t[i])
    return y

steps = .01
t = np.arange(-10, 10 + steps, steps)

y1 = h1(t)
y2 = h2(t)
y3 = h3(t)
```

Listing 1: Defining the functions to plot the three signals from -10 < t < 10

The user-defined convolution function is given below in Listing 2. This was used to convolve the transer functions with the step function. Another function was defined to create a step function array.

```python
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)
    return result

#step function array
def u_array(t):
    y = np.zeros((len(t), 1))
    for i in range(len(t)):
        y[i] = u(t[i])
    return y

t = np.arange(-10, 10 + steps, steps)
time = len(t);
tEx = np.arange(-20, (2*t[time-1]) + steps, steps)

a = my_conv(y1, step)*steps
b = my_conv(y2, step)*steps
c = my_conv(y3, step)*steps
```

Listing 2: Convolution and step array functions

```
def c1(t):
    y = np.zeros((len(t),1))
    for i in range(len(t)):
        y[i] = 0.5*((1-np.exp(-2*t[i]))*u(t[i]) - ((1-np.exp(-2*(t[i]-3)))*u(t[i]-3)))
    return y

def c2(t):
    y = np.zeros((len(t),1))
    for i in range(len(t)):
        y[i] = r(t[i]-2)*u(t[i]-2) - r(t[i]-6)*u(t[i]-6)
    return y

def c3(t):
    f = 0.25
    w = 2*f*np.pi
    y = np.zeros((len(t),1))
    for i in range(len(t)):
        y[i] = (1/w)*math.sin(w*t[i])*u(t[i])
    return y
```

Listing 3: Defining functions for plotting the hand calculations

4 Results

The hand calculations foe the convolutions are shown below:

$$y1(t) = \int_0^t h1(T)u(t-T)dT$$

$$= \int_0^t e^{-2T} * [u(T) - u(T-3)]u(t-T)dT$$

$$= 0.5[(1 - e^{-2t})u(t) - (1 - e^{-2(t-3)})u(t-3)]$$

$$y2(t) = \int_0^t h2(T)u(t-T)dT$$

$$= \int_0^t [u(T-2) - u(T-6)]u(t-T)dT$$

$$= r(t-2)u(t-2) - r(t-6)u(t-6)$$

$$y3(t) = \int_0^t h3(T)u(t-T)dT$$

$$= \int_0^t cos(w_0 * T)u(T)u(t-T)dT$$

$$= (1/w_0)sin(w_0 * t)u(t)$$

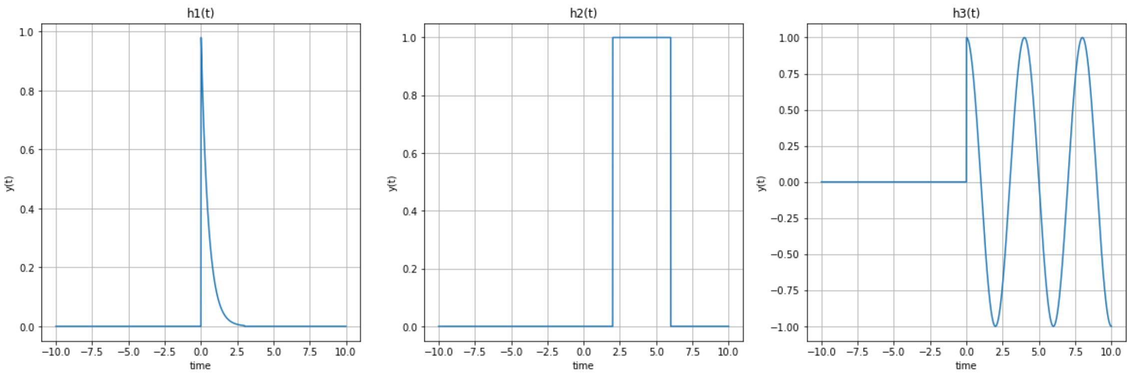

Figure 1: Plots of the three initial given signals

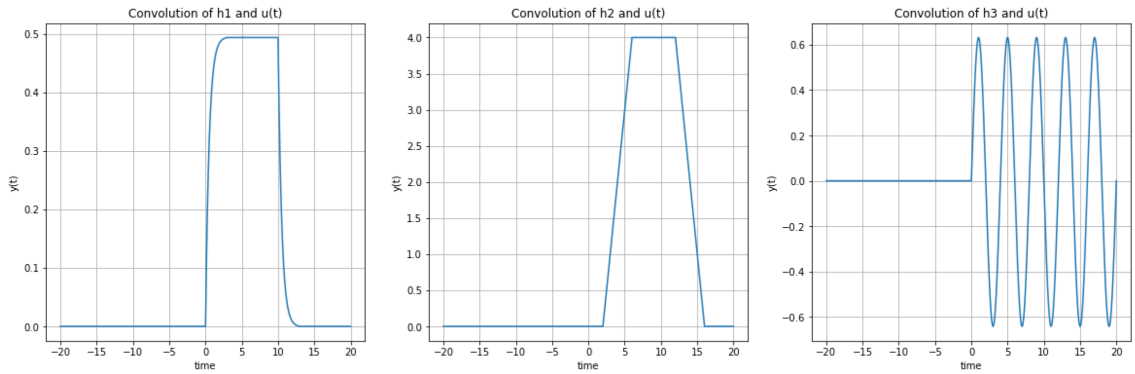

Figure 2: User-defined convolutions of the three signals

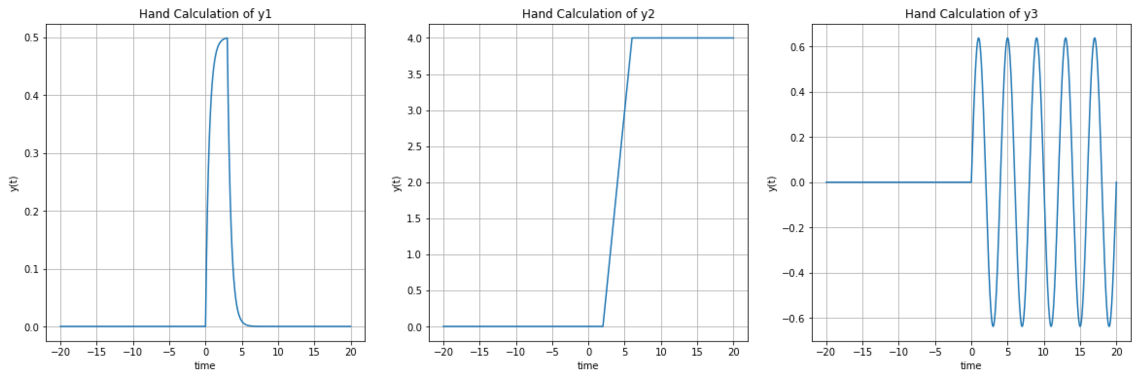

Figure 3: Convolutions using the scipy.signal library

5 Error Analysis

My hand calculation convolution for the first transfer function doesn't look like the user-defined convolution. I tried reworking my equation, but I don't know if my equation itself is wrong, or if there is an error in the Python.

6 Questions

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

Everything was clear on what needed to be done and turned in.

7 Conclusion

A comparison between hand calculated and Python convolution was explored. There was some error on the hand calculation side since the first convolution decreases at a sooner point than the user-defined convolution. There is also some scaling issues with the second hand convolution since compared to the user-defined one. The Python and LaTeX code are seen in `https://github.com/Eniac618/ECE351_Code` and `https://github.com/Eniac618/ECE351_Reports` respectively.