ECE 351

Signals and Systems

Lab 4

System Step Respone

Submitted By :
Zachary Pfaff

Contents

System Step Respone

Zachary Pfaff

February 15th, 2022

1 Introduction

Previously in lab 3, students were responsible for creating their own convolution functions and convolving three different functions. The goal of the following lab is to become more familiar with convolutions by computing a systems step response using the convolution functions created previously. In doing so, students will become more familiar with calculating step responses of transfer functions and incorporating their calculations into python code.

2 Equations

For lab 4, I will find the step response of three different user defined signals. The signals I will be working with are seen below.

$$f_1(t) = e^{-2t}[u(t) - u(t-3)]$$

$$f_2(t) = u(t-2) - u(t-6)$$

$$f_3(t) = cos(1.57t)u(t)$$

When I calculated for their respected step responses, I came up with the three solutions below with the help of our TA.

$$h_1(t) = \frac{1}{2}[(-e^{-2t} + 1)u(t)] - [-e^{-2(t-3)} + 1)u(t)]$$

$$h_2(t) = (t-2)u(t-2) - (t-6)u(t-6)$$

$$f_3(t) = \frac{1}{1.57}sin(1.57t)u(t)$$

3 Methodology

The next step was to plot the original three functions below in a single figure. The range was set from -10 to 10 in order to show off the whole plot. The plots for these functions are seen below in figure 1.

The next plot shows the step responses of the functions using the convolved function created earlier in lab 3. To set the bounds for the convolved function I had to create a new array of steps that matched the dimensions of the convolved function. The plotted step responses are seen below in figure 2.

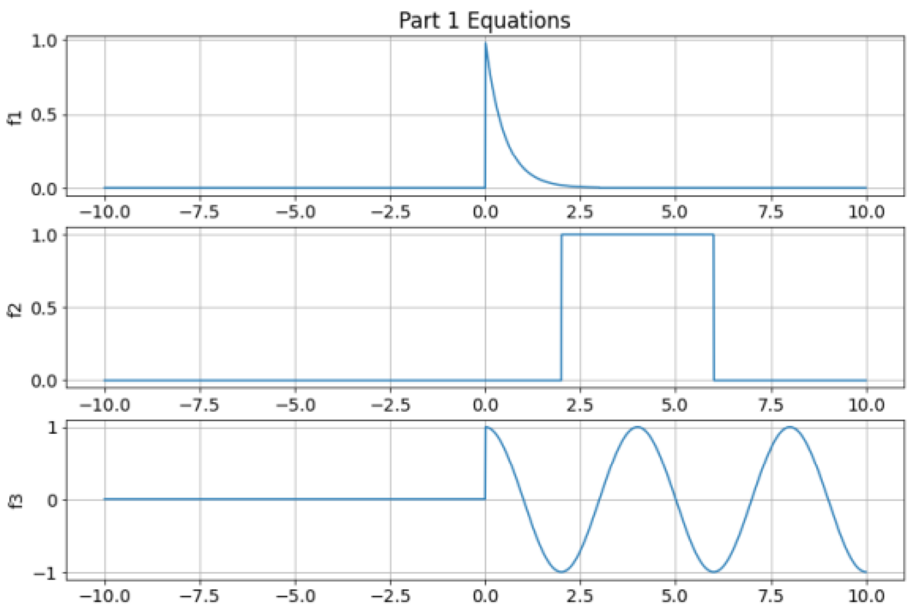

Figure 1: Plots of original equations

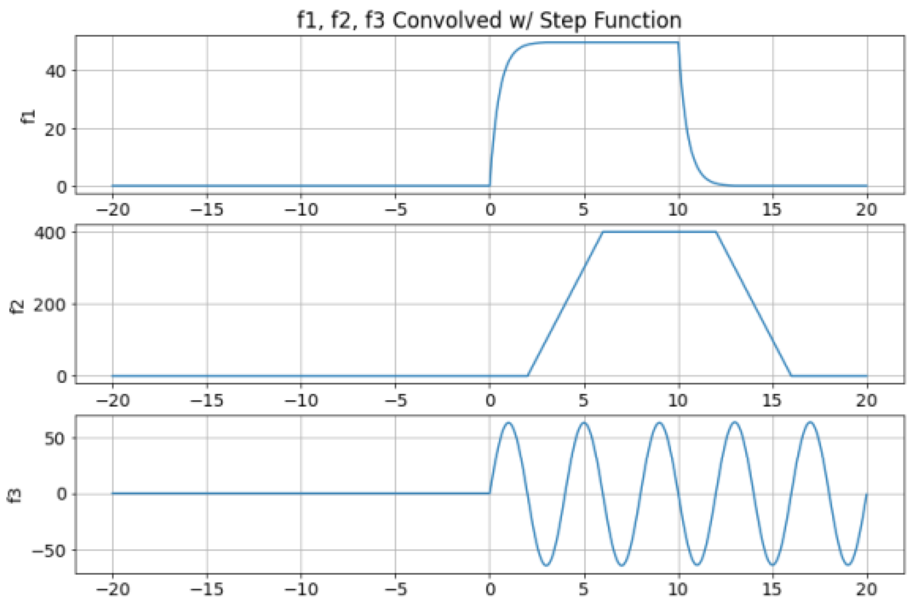

Figure 2: Step response using convolution function

The last plots are the step response functions calculated by hand and graphed to see if they match the previous sets of plots. When I attempted this I ran into a problem with f2. Function two is supposed to slope up to a value of 4 and then eventually slope back down to zero. For my calculated function, this was not the case, although it would ramp up the function would never ramp back down to zero. To try and correct this I tried re-working the problem and inputting an altered equation that uses ramp functions instead of step functions, the problem however still persisted and I was unable to perfectly match the convolved step response. Nevertheless, figure 3 below shows the plots for my hand calculated step responses.

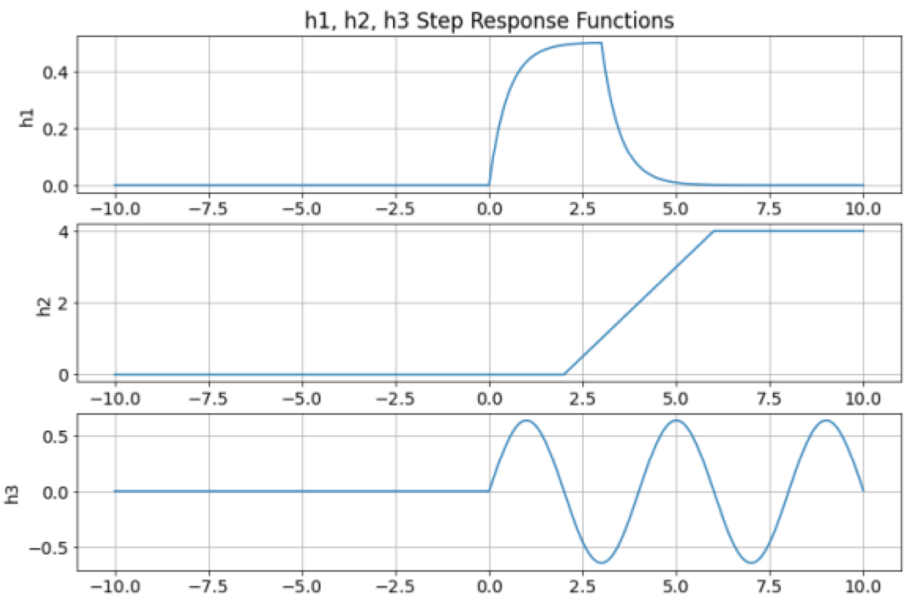

Figure 3: Step response using hand calculation

4 Listings

The following listing below shows implementation of my step, ramp, and convolution user defined functions that were used for creating my functions. All of the following code was already completed in labs 2 and 3 and then reused in this lab.

```
steps = 0.01
t = np.arange(-10, 10 + steps, steps)
tbound = np.arange(-20, 2 * t[len(t) - 1] + steps, steps)

#Code for Step Function
def step(t):
    y = np.zeros(t.shape)

    for i in range(len(t)):
        if t[i] < 0:
            y[i] = 0
        else:
            y[i] = 1

    return y

#code for Ramp Function
def ramp(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

#Code for Convolution
def my_conv(f1, f2):
   Nf1 = len(f1)    #create new arrays with same length as input
   signals
   Nf2 = len(f2)

   f1Extended = np.append(f1, np.zeros((1, Nf2 - 1)))
   f2Extended = np.append(f2, np.zeros((1, Nf1 - 1)))

   result = np.zeros(f1Extended.shape)
   for i in range(Nf2):
       for j in range(Nf1):
               try: result[i + j] += f1Extended[i] * f2Extended[j]
               except: print(i,j)

   return result
```

Listing 1: Convolution Function

The next listing shows the original functions and the step response functions being implemented into python code.

```
f1 = np.exp(-2 * t) * (step(t) - step(t - 3))
f2 = step(t - 2) - step(t - 6)
f3 = np.cos(1.57 * t) * step(t)

h1 = (1/2) * ((((-1 * np.exp(-2 * t) + 1) * step(t)) - (-1 * np.exp
   (-2 * (t - 3)) + 1) * step(t -  3)))
```

```
h2 = (t - 2) * step(t - 2) - (t - 6) * step(t - 6)
h3 = (1/1.57) * np.sin(1.57 * t) * step(t)
```

Listing 2: Convolution Function

The last listing below shows the original functions being convolved with my user defined step function. This code was used to created the graphs in figure 2.

```
x = my_conv(f1, step(t))

y = my_conv(f2, step(t))

z = my_conv(f3, step(t))
```

Listing 3: Convolution Function

5 Conclusion

By the end of this lab I was able to create the step response of the three functions using convolution. When it came to the hand calculated function however, I ran into some issues with the graphing. Graph h2 started out how it was supposed to by plotting a linear line to a height of 4 from t = 2 to t = 6, but once the graph leveled out it never sloped back down to a value of zero unlike the step response from the convolved method. A similar issue occurred with graph h1 where it resembled a similar shape as f1 convolved with the step function, however it transitions down to zero too early. Despite this problem, the rest of my plots came out as expected and I feel more confident in my understanding of step response functions.

6 Questions

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

I did not have any problems following lab procedure and expectations for lab 4.