1 Introduction

The goal of this lab is to become familiar with using convolution to compute a system's step response.

2 Equations

For this lab, we create the following signals as user-defined functions and also we had convolve this function manually:

$$h1(t) = e^{-2t}[u(t) - u(t-3)] \tag{1}$$

$$h2(t) = u(t-2) - u(t-6) \tag{2}$$

$$h3(t) = cos(w_0 t)u(t) \tag{3}$$

The step responses, respectively:

$$f1(t) = \frac{1}{2}[(1 - e^{-2t})u(t) + (e^{-2t} - e^{-6})u(t-3)] \tag{4}$$

$$f2(t) = (t-2)u(t-2) - (t-6)u(t-6) \tag{5}$$

$$f3(t) = \frac{1}{w_0} sin(w_0 t)u(t) \tag{6}$$

3 Methodology and Results

Lab 4 has 2 parts :

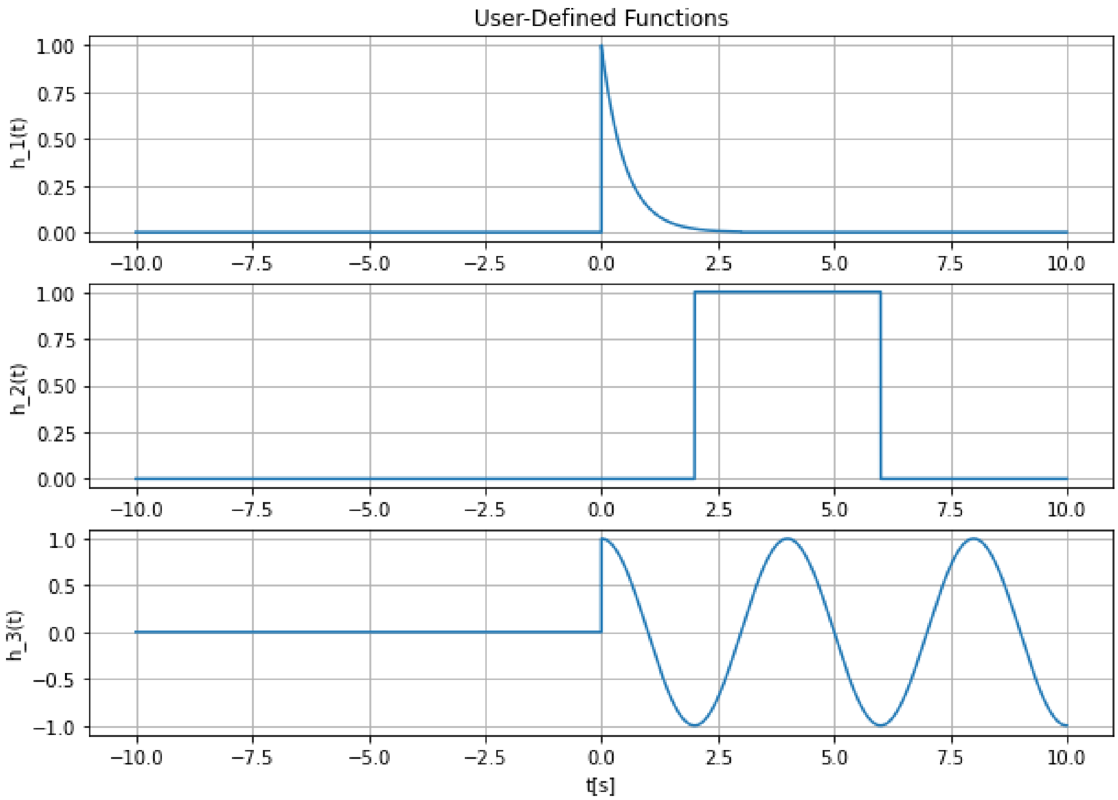
- Part 1:

  In this part, we plot the three functions in a single figure (separate subplots) from -10 to 10 with time steps small enough to achieve appropriate resolution:

```
#%% part 1 task 1 & 2


def u(t):
y = np.zeros(t.shape)
for i in range(len(t)):
if t[i] >= 0:
y[i] = 1
else:
y[i]= 0
return y

def h_1(t):
h1 = np.exp(-2*t)*(u(t)-u(t-3))
return h1

```

```
def h_2(t):
h2 = u(t-2)-u(t-6)
return h2

def h_3(t):
h3 = np.cos((2*np.pi*0.25)*t)*u(t)
return h3

steps = 1e-3
t = np.arange(-10, 10 + steps, steps)

plt.figure(figsize=(10,7))
plt.subplot(3, 1, 1)
plt.plot(t,h_1(t))
plt.ylabel ('h_1(t)')
plt.title ('User-Defined Functions')
plt.grid()
plt.subplot(3, 1, 2)
plt.plot(t,h_2(t))
plt.ylabel ('h_2(t)')
plt.grid()
plt.subplot(3, 1, 3)
plt.plot(t,h_3(t))
plt.ylabel ('h_3(t)')
plt.grid()
plt.xlabel('t[s]')
plt.show()
```

The output of this part is:

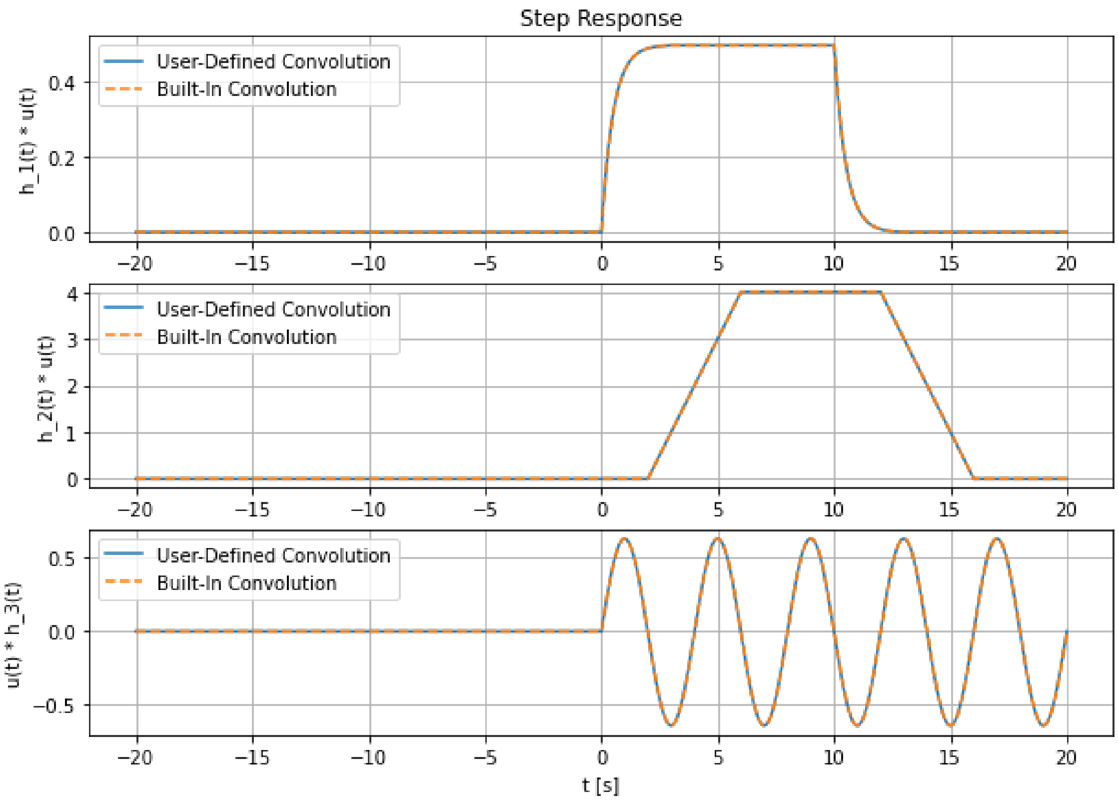


- Part 2:
  Task 1: We convolve the three transfer functions with step function to find step response using the convolution created in lab 3 and plot the response:

```
#%% part 2 task 1



def conv(h1, h2):
Nh1 = len(h1)
Nh2 = len(h2)
h1Extended = np.append(h1, np.zeros((1, Nh2 -1)))
h2Extended = np.append(h2, np.zeros((1, Nh1 -1)))
result = np.zeros(h1Extended.shape)

for i in range(Nh2 + Nh1 - 2):
result[i] = 0
for j in range(Nh1):
if(i - j + 1 > 0):
try:
result[i] = result[i] + h1Extended[j]*h2Extended[i - j + 1]
except:
print(i, j)
return result

```

```
    steps = 1e-2
    t = np.arange(-10, 10 + steps, steps)
    NN = len(t)
    tExtended = np.arange(2*t[0], 2*t[NN - 1] + steps, steps)

    h1 = h_1(t)
    h2 = h_2(t)
    h3 = h_3(t)


    conv12 = conv(h1, u(t))*steps
    conv12Check = sig.convolve(h1, u(t))*steps

    plt.figure(figsize = (10, 7))
    plt.subplot(3, 1, 1)
    plt.plot(tExtended, conv12, label = 'User-Defined Convolution')
    plt.plot(tExtended, conv12Check, '--', label = 'Built-In Convolution')
    plt.grid()
    plt.legend()
    plt.xlabel('t [s]')
    plt.ylabel('h_1(t) * u(t)')
    plt.title('Step Response')


    conv23 = conv(h2, u(t))*steps
    conv23Check = sig.convolve(h2, u(t))*steps
    plt.subplot(3, 1, 2)
    plt.plot(tExtended, conv23, label = 'User-Defined Convolution')
    plt.plot(tExtended, conv23Check, '--', label = 'Built-In Convolution')
    plt.grid()
    plt.legend()
    plt.xlabel('t [s]')
    plt.ylabel('h_2(t) * u(t)')


    conv13 = conv(u(t), h3)*steps
    conv13Check = sig.convolve(u(t), h3)*steps
    plt.subplot(3, 1, 3)
    plt.plot(tExtended, conv13, label = 'User-Defined Convolution')
    plt.plot(tExtended, conv13Check, '--', label = 'Built-In Convolution')
    plt.grid()
    plt.legend()
    plt.xlabel('t [s]')
    plt.ylabel('u(t) * h_3(t)')


```

The output of task 1 is:



Task 3: We convolved the transfer functions with step function manually and found the step response equations as follows:

$$f1(t) = \frac{1}{2}[(1 - e^{-2t})u(t) + (e^{-2t} - e^{-6})u(t-3)] \tag{7}$$

$$f2(t) = (t-2)u(t-2) - (t-6)u(t-6) \tag{8}$$

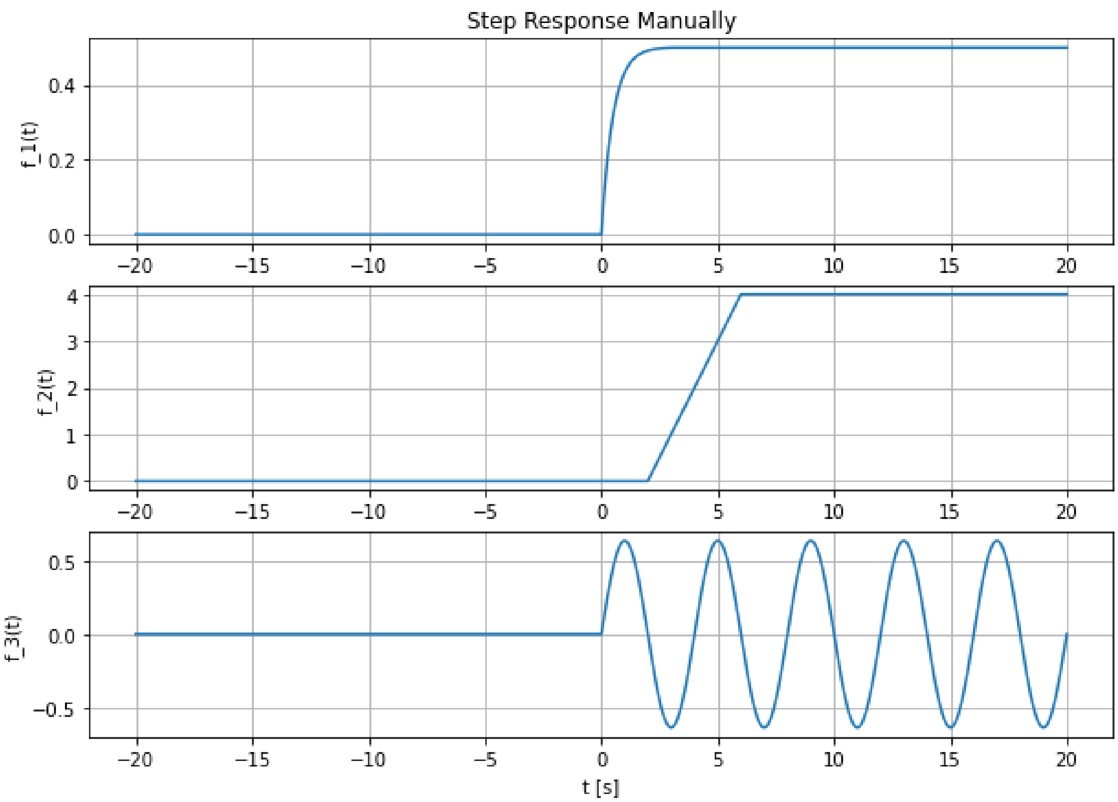
$$f3(t) = \frac{1}{w_0} sin(w_0 t)u(t) \tag{9}$$

Then, we used these equations in Python to get their plots:

```
def f_1(t):
f1 = (1/2)*(u(t)*(1-np.exp(-2*t))+u(t-3)*(np.exp(-2*t)-np.exp(-6)))
return f1
plt.figure(figsize=(10,7))
plt.subplot(3, 1, 1)
plt.plot(tExtended,f_1(tExtended))
plt.grid()
plt.ylabel ('f_1(t)')
plt.xlabel('t [s]')
plt.title ('Step Response Manually')
```

```python



def f_2(t):
f2 = (t-2)*u(t-2)-(t-6)*u(t-6)
return f2
plt.subplot(3, 1, 2)
plt.plot(tExtended,f_2(tExtended))
plt.grid()
plt.ylabel ('f_2(t)')
plt.xlabel('t [s]')



def f_3(t):
f3 = (1/(2*np.pi*0.25))*np.sin((2*np.pi*0.25)*t)*u(t)
return f3
plt.subplot(3, 1, 3)
plt.plot(tExtended,f_3(tExtended))
plt.grid()
plt.ylabel ('f_3(t)')
plt.xlabel('t [s]')


```

The output of this task is:



4 Questions

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

   The Lab is very clear.

5 Conclusion

At the end of this lab, we became more familiar with using convolution and learned how to compute a system's step response by Python.