1 Introduction

The goal of this lab is to become familiar with convolution and its properties using Python.

2 Equations

For this lab, we use three function from the previous lab to convolve them together in this lab;

$$funct1(t) = u(t-2) - u(t-9) \tag{1}$$

$$funct2(t) = e^-t \tag{2}$$

$$funct3(t) = r(t) - r(t-3) + 5u(t-3) - 2u(t-6) - 2r(t-6) \tag{3}$$

3 Methodology and Results

Lab 3 has 2 parts:

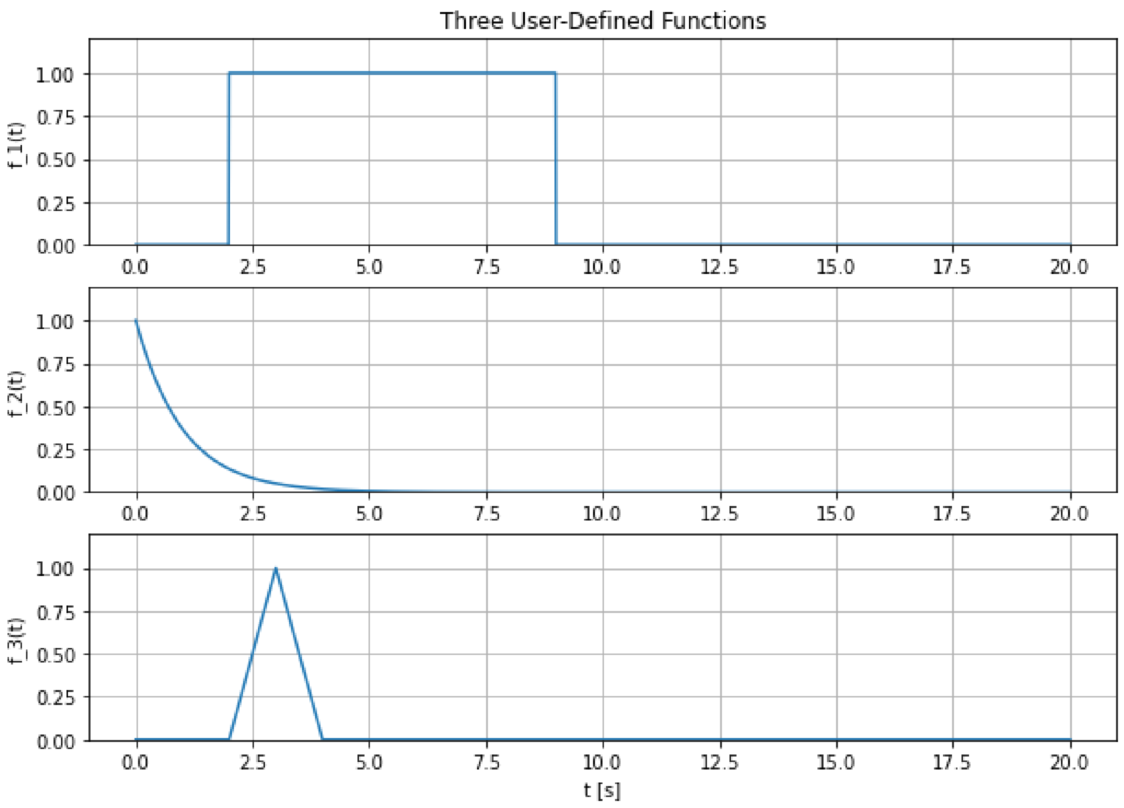
- Part 1:

  In this part, we created three functions (user defined) and plotted them in a single figure (separate subplots) from 0 to 20 with time steps small enough to achieve appropriate resolution.

```
#%% task 1

#%% Part 1 Task 1

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 r(t):
y = np.zeros(t.shape)

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

def f_1(t):
y = u(t - 2) - u(t - 9)
return y
```

```python

def f_2(t):
y = np.exp(-t)*u(t)
return y

def f_3(t):
y = r(t - 2)*(u(t - 2) - u(t -3)) + r(4 - t)*(u(t - 3) - u(t - 4))
return y

#%% Part 1 Task 2

steps = 1e-2
t = np.arange(0, 20 + steps, steps)

plt.figure(figsize = (10, 7))
plt.subplot(3, 1, 1)
plt.plot(t, f_1(t))
plt.ylabel('f_1(t)')
plt.title('Three User-Defined Functions')
plt.ylim([0, 1.2])
plt.grid()
plt.subplot(3, 1, 2)
plt.plot(t, f_2(t))
plt.ylabel('f_2(t)')
plt.ylim([0, 1.2])
plt.grid()
plt.subplot(3, 1, 3)
plt.plot(t, f_3(t))
plt.ylabel('f_3(t)')
plt.ylim([0, 1.2])
plt.grid()
plt.xlabel('t [s]')
plt.show()



```

The output of this part is:

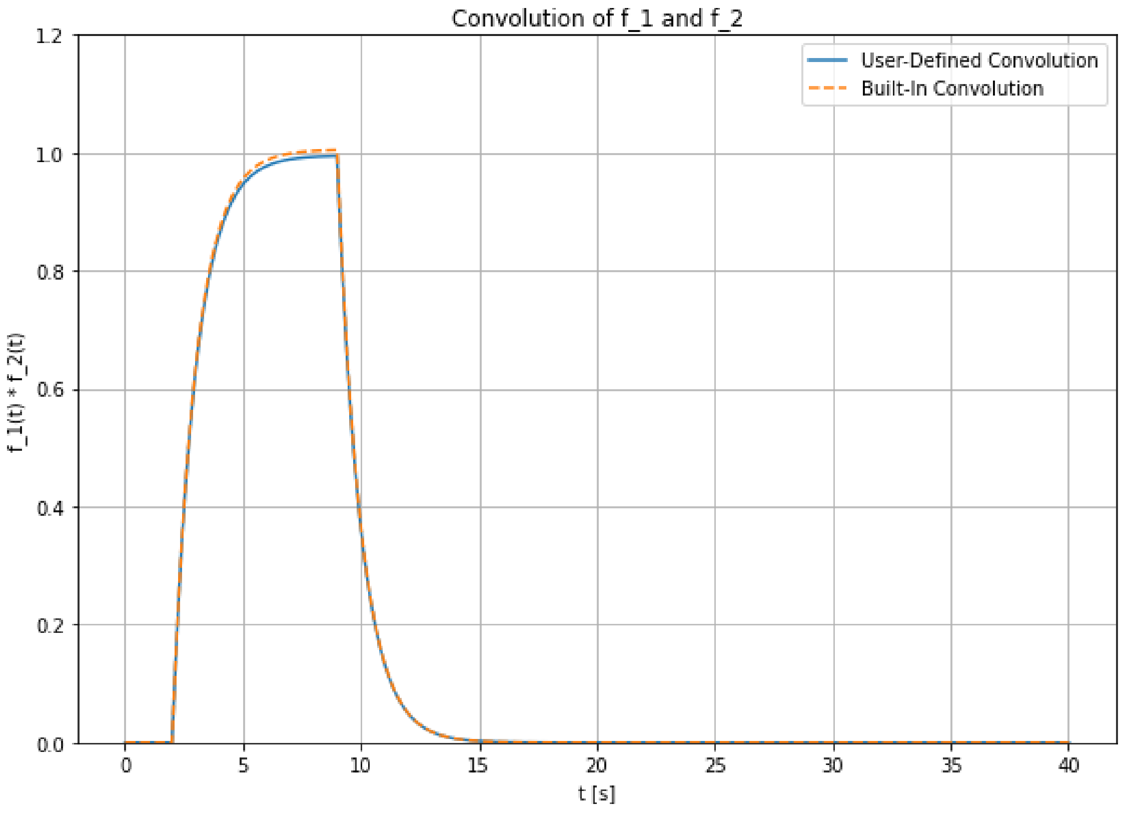


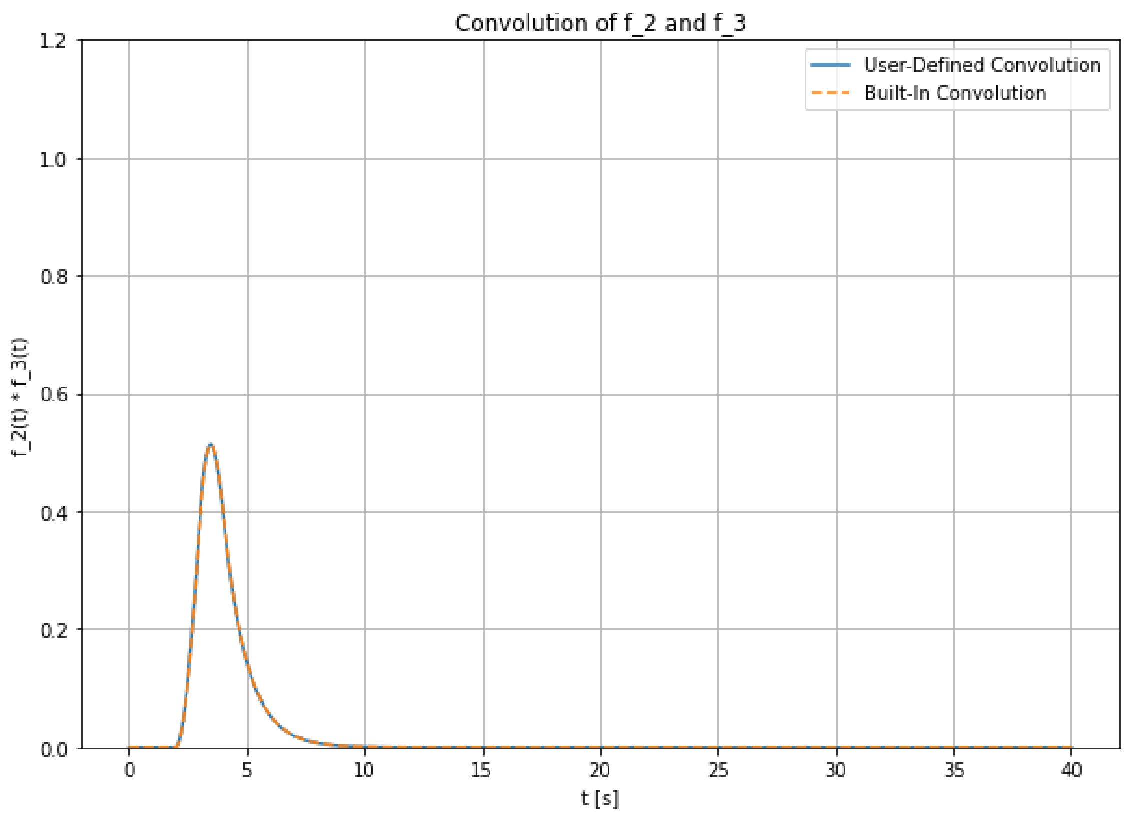
- Part 2:
  We write our own code to perform convolution. Then used the scipy.signal.convolve() function to verify our code and compared them through plot:

```python
#%% Part 2 Task 1

def conv(f1, f2):
Nf1 = len(f1)
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 + Nf1 - 2):
result[i] = 0
for j in range(Nf1):
if(i - j + 1 > 0):
try:
result[i] = result[i] + f1Extended[j]*f2Extended[i - j + 1]
except:
print(i, j)
return result

steps = 1e-2
t = np.arange(0, 20 + steps, steps)
```

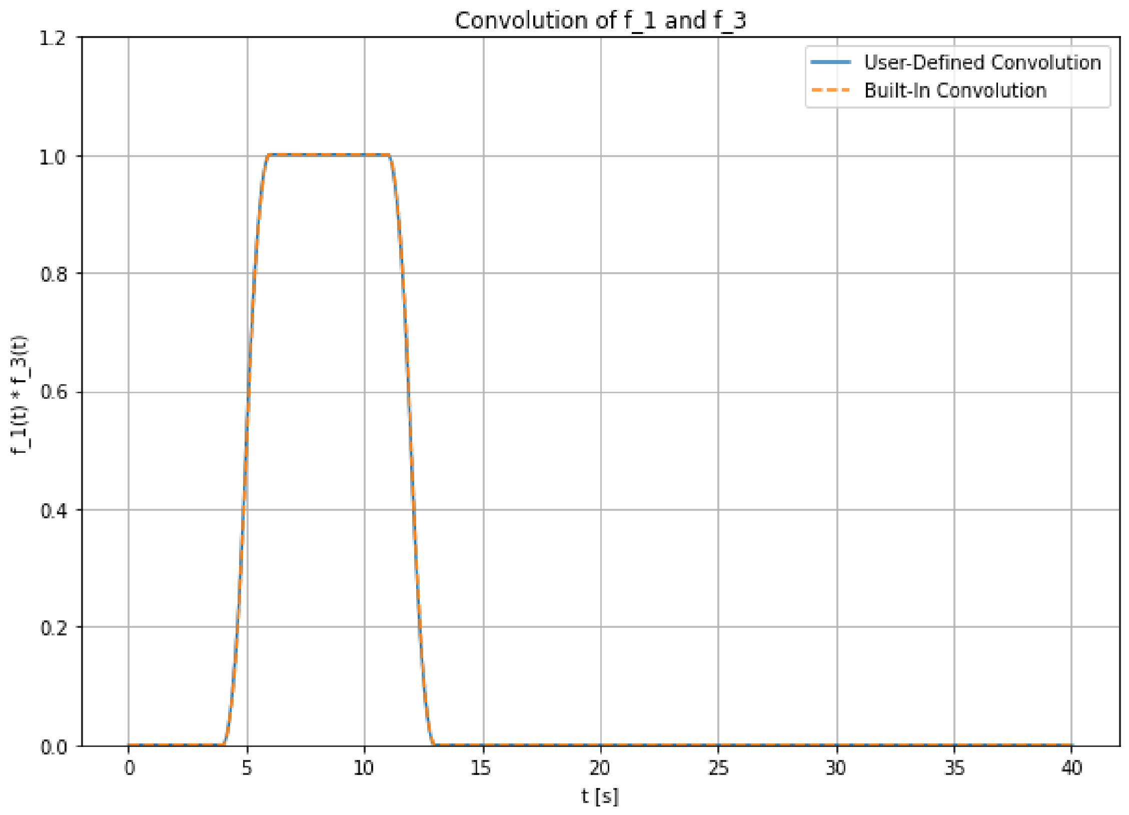
```
    tExtended = np.arange(0, 2*t[len(t) - 1], steps)

    f1 = f_1(t)
    f2 = f_2(t)
    f3 = f_3(t)

    #%% Part 2 Task 2

    conv12 = conv(f1, f2)*steps
    conv12Check = sig.convolve(f1, f2)*steps

    plt.figure(figsize = (10, 7))
    plt.plot(tExtended, conv12, label = 'User-Defined Convolution')
    plt.plot(tExtended, conv12Check, '--', label = 'Built-In Convolution')
    plt.ylim([0, 1.2])
    plt.grid()
    plt.legend()
    plt.xlabel('t [s]')
    plt.ylabel('f_1(t) * f_2(t)')
    plt.title('Convolution of f_1 and f_2')
    plt.show()

    #%% Part 2 Task 3

    conv23 = conv(f2, f3)*steps
    conv23Check = sig.convolve(f2, f3)*steps

    plt.figure(figsize = (10, 7))
    plt.plot(tExtended, conv23, label = 'User-Defined Convolution')
    plt.plot(tExtended, conv23Check, '--', label = 'Built-In Convolution')
    plt.ylim([0, 1.2])
    plt.grid()
    plt.legend()
    plt.xlabel('t [s]')
    plt.ylabel('f_2(t) * f_3(t)')
    plt.title('Convolution of f_2 and f_3')
    plt.show()

    #%% Part 2 Task 4

    conv13 = conv(f1, f3)*steps
    conv13Check = sig.convolve(f1, f3)*steps

    plt.figure(figsize = (10, 7))
    plt.plot(tExtended, conv13, label = 'User-Defined Convolution')
    plt.plot(tExtended, conv13Check, '--', label = 'Built-In Convolution')
    plt.ylim([0, 1.2])
    plt.grid()
    plt.legend()
    plt.xlabel('t [s]')
    plt.ylabel('f_1(t) * f_3(t)')
    plt.title('Convolution of f_1 and f_3')
    plt.show()


```

The output of this part is the following three graphs:





Convolution of f_2 and f_3
User-Defined Convolution
Built-In Convolution
f_2(t) * f_3(t)
t [s]



Convolution of f_1 and f_3
User-Defined Convolution
Built-In Convolution
f_1(t) * f_3(t)
t [s]

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 worked alone.

2. What was the most difficult part of this lab for you, and what did your problem-solving process look like?

   To understand the user-defined code to perform convolution.

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

   Yes, we chose this approach because we needed to perform convolution without using integral functions.

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

5 Conclusion

At the end of this lab, we learned how to convolve two functions together using Python.