# -\*- coding: utf-8 -\*-

"""

Created on Thu Sep 27 10:23:51 2021

@author: aleb9491

"""

################################################################

# Mohammad Al Ebedan #

# ECE351-52 #

# Lab 4 #

# Sep 27, 2021 #

################################################################

import numpy as np

import matplotlib.pyplot as plt

import scipy.signal as sig

#################################################################

# Part 1 #

#################################################################

#%% 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()

#################################################################

# Part 2 #

#################################################################

#%% 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)')

#%% Part 2 task 2

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')

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]')