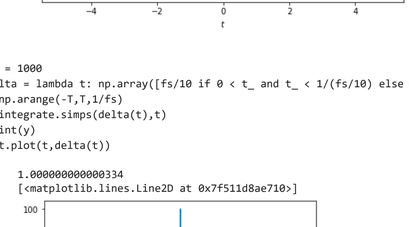
NAME : AHAMED M.B.S.A.

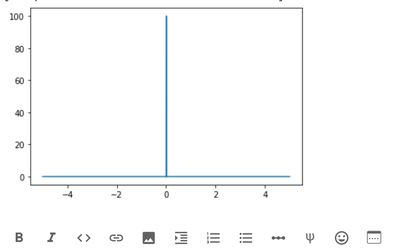
GROUP - A-01

REGISTRATION NO - 200014B

```
from scipy import integrate
import numpy as np
import matplotlib.pyplot as plt
h = lambda t: (t > 0)*1.0
x = lambda t: (t > 0) * np.exp(-2*t)
Fs = 50
T = 5
t = np.arange(-T, T, 1/Fs)
plt.figure(figsize=(8,3))
plt.plot(t, h(t), label='$h(t)$')
plt.plot(t, x(t), label='$x(t)$')
plt.xlabel(r'$t$')
plt.legend()

t_ = 1
flipped = lambda tau: h(t_ - tau)
product = lambda tau: x(tau)*h(t_ - tau)
plt.figure(figsize=(8,3))
plt.plot(t, x(t), label=r'$x(\tau)$')
plt.plot(t, flipped(t), label=r'$h(t - \tau)$')
plt.plot(t, product(t), label=r'$x(\tau)h(t -\tau)$')

y = np.zeros(len(t))
for n, t_ in enumerate(t):
  product = lambda tau: x(tau) * h(t_ - tau)
  y[n] = integrate.simps(product(t), t)

plt.plot(t, y, label=r'$x(t)\ast h(t)$')
plt.xlabel(r'$t$')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f511d9d3990>
```






```
fs = 1000
delta = lambda t: np.array([fs/10 if 0 < t_ and t_ < 1/(fs/10) else 0.0 for t_ in t])
t=np.arange(-T,T,1/fs)
y=integrate.simps(delta(t),t)
print(y)
plt.plot(t,delta(t))
```

```
1.000000000000334
[<matplotlib.lines.Line2D at 0x7f511d8ae710>]
```


TASK 3

x(t) = e^(-at).u(t), a>0

h(t) = δ(t + 2) + δ(t - 1)

y(t) = x(t) * h(t)

n = number of range

x(n).h(t-n) = e^(-at).(δ(t + 2) + δ(t - 1)) , 0 < t < n

0, otherwise

y(t) = ∫ [-∞, ∞] x(n).h(t-n)

y(t) = ∫ [-∞, ∞] e^(-at).(δ(t + 2) + δ(t - 1))

```
# This is formatted as code
```

TASK 3

x(t) = e^(-at).u(t), a>0

h(t) = δ(t + 2) + δ(t - 1)

y(t) = x(t) * h(t)

n = number of range

x(n).h(t-n) = e^(-at).(δ(t + 2) + δ(t - 1)) , 0 < t < n

0, otherwise

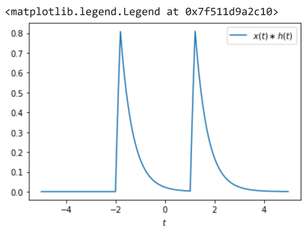
y(t) = ∫ [-∞, ∞] x(n).h(t-n)

y(t) = ∫ [-∞, ∞] e^(-at).(δ(t + 2) + δ(t - 1))

```
# This is formatted as code
```

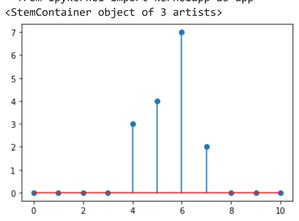
```
h = lambda t: (delta(t+2) + delta(t-1))
x = lambda t: (t > 0) * np.exp(-2*t)
fs = 50
T = 5
t = np.arange(-T, T, 1/Fs)

t_ = 1
flipped = lambda tau: h(t_ - tau)
product = lambda tau: x(tau)*h(t_ - tau)


y = np.zeros(len(t))
for n, t_ in enumerate(t):
  product = lambda tau: x(tau) * h(t_ - tau)
  y[n] = integrate.simps(product(t), t)

plt.plot(t, y, label=r'$x(t)\ast h(t)$')
plt.xlabel(r'$t$')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f511d9a2c10>
```




```
x = np.array([0, 1, 1, 2, 0])
h = np.array([0, 0, 0, 3, 1, 0, 0])
hr = np.flip(h)
xo = 2
ho = 4
y = np.zeros(len(x) + len(h) - 1)
for n in range(len(y)):
  xkmin = max(0, n - len(h) + 1)
  xkmax = min(len(x), n + 1)
  hkmin = max(0, len(h) - n -1)
  hkmax = min(len(h), len(x) + len(h) -n -1)
  y[n] = np.sum(x[xkmin:xkmax]*hr[hkmin:hkmax])
  print("y[{0}] = x[{1}:{2}]*h[{3}:{4}] = {5}".format(n, xkmin, xkmax, hkmin, hkmax, y[n]))

plt.stem([n for n in range(len(y))],y)
```

```
y[0] = x[0:1]*h[6:7] = 0.0
y[1] = x[0:2]*h[5:7] = 0.0
y[2] = x[0:3]*h[4:7] = 0.0
y[3] = x[0:4]*h[3:7] = 0.0
y[4] = x[0:5]*h[2:7] = 3.0
y[5] = x[0:5]*h[1:6] = 4.0
y[6] = x[0:5]*h[0:5] = 7.0
y[7] = x[1:5]*h[0:4] = 2.0
y[8] = x[2:5]*h[0:3] = 0.0
y[9] = x[3:5]*h[0:2] = 0.0
y[10] = x[4:5]*h[0:1] = 0.0
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as a LineCollection instead of individual lines. This significantly improves the performance
  from ipykernel import kernelapp as app
<StemContainer object of 3 artists>
```

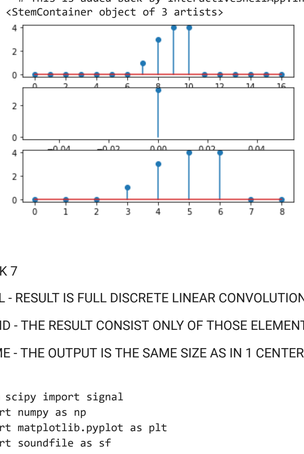


```
x = np.array([0, 0, 0, 1, 1, 2, 0, 0, 0])
h = np.array([0, 0, 0, 0, 0, 1, 2, 0, 0, 0])
hr = np.flip(h)
xo = 2
ho = 4
y = np.zeros(len(x) + len(h) - 1)
for n in range(len(y)):
    xkmin = max(0, n - len(h) + 1)
    xkmax = min(len(x), n + 1)
    hkmin = max(0, len(h) - n -1)
    hkmax = min(len(h), len(x) + len(h) -n -1)
    y[n] = np.sum(x[xkmin:xkmax]*hr[hkmin:hkmax])
    print("y[{0}] = x[{1}:{2}]*h[{3}:{4}] = {5}".format(n, xkmin, xkmax, hkmin, hkmax, y[n]))

plt.stem([n for n in range(len(y))],y)
```

```
y[0] = x[0:1]*h[8:9] = 0.0
y[1] = x[0:2]*h[7:9] = 0.0
y[2] = x[0:3]*h[6:9] = 0.0
y[3] = x[0:4]*h[5:9] = 0.0
y[4] = x[0:5]*h[4:9] = 0.0
y[5] = x[0:6]*h[3:9] = 0.0
y[6] = x[0:7]*h[2:9] = 0.0
y[7] = x[0:8]*h[1:9] = 1.0
y[8] = x[0:9]*h[0:9] = 3.0
y[9] = x[1:9]*h[0:8] = 4.0
y[10] = x[2:9]*h[0:7] = 4.0
y[11] = x[3:9]*h[0:6] = 0.0
y[12] = x[4:9]*h[0:5] = 0.0
y[13] = x[5:9]*h[0:4] = 0.0
y[14] = x[6:9]*h[0:3] = 0.0
y[15] = x[7:9]*h[0:2] = 0.0
y[16] = x[8:9]*h[0:1] = 0.0
```

```
from scipy import signal
x = np.array([0, 0, 0, 1, 1, 2, 0, 0, 0])
h = np.array([0, 0, 0, 0, 1, 2, 0, 0, 0])
N = np.arange(-4, 5, 1)
fig, ax = plt.subplots(3, 1)
y1= signal.convolve(x, h, mode="full", method="auto")
y2= signal.convolve(x, h, mode="valid", method="auto")
y3= signal.convolve(x, h, mode="same", method="auto")
ax[0].stem(y1)
ax[1].stem(y2)
ax[2].stem(y3)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:9: UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as a LineCollection instead of individual lines. This significantly improves the performance
  if __name__ == '__main__':
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as a LineCollection instead of individual lines. This significantly improves the performance
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:11: UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as a LineCollection instead of individual lines. This significantly improves the performance
  # This is added back by InteractiveShellApp.init_path()
<StemContainer object of 3 artists>
```



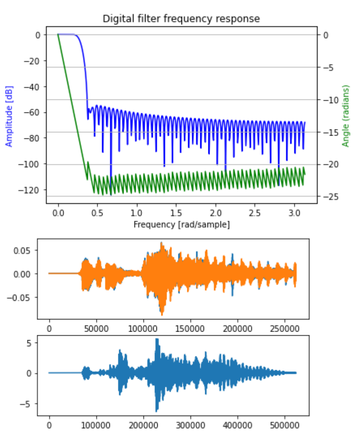
TASK 7

FULL - RESULT IS FULL DISCRETE LINEAR CONVOLUTION OF THE INPUTS

VALID - THE RESULT CONSIST ONLY OF THOSE ELEMENTS THAT NOT dEPEND ON THE ZERO PADDLING

SAME - THE OUTPUT IS THE SAME SIZE AS IN 1 CENTERED WITH RESPECT TO THE FULL OUTPUT

```
from scipy import signal
import numpy as np
import matplotlib.pyplot as plt
import soundfile as sf

data, samplerate = sf.read('power_of_love.wav')
nyquist = samplerate//2
fc = 2000/nyquist
n = 121
b = signal.firwin(n, fc, pass_zero=True)
w, h = signal.freqz(b)


import matplotlib.pyplot as plt
fig, ax1 = plt.subplots()
ax1.set_title('Digital filter frequency response')
ax1.plot(w, 20 * np.log10(abs(h)), 'b')
ax1.set_ylabel('Amplitude [dB]', color='b')
ax1.set_xlabel('Frequency [rad/sample]')
ax2 = ax1.twinx()
angles = np.unwrap(np.angle(h))
ax2.plot(w, angles, 'g')
ax2.set_ylabel('Angle (radians)', color='g')
ax2.grid()
ax2.axis('tight')
plt.show()


ch1=[]
ch2=[]

for i in range(len(data)):
  ch1.append(data[i][0])
  ch2.append(data[i][1])
y=signal.convolve(ch1,ch2)



fig,ax=plt.subplots(2,1)
ax[0].plot(data)
ax[1].plot(y)



sf.write('audio_file_filtered.wav', np.vstack((ch1, ch2)).T + data, samplerate)
```




```
xmn=np.array([0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0])
shape1=(5,5)
x=xmn.reshape(shape1)
hmn=np.array([1,2,3,4,5,6,7,8,9])
shape2=(3,3)
h=hmn.reshape(shape2)
y=signal.convolve2d(x,h)
print(y)
```

```
[[0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0]
 [0 0 1 2 3 0 0]
 [0 0 4 5 6 0 0]
 [0 0 7 8 9 0 0]
 [0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0]]
```

TASK 11

WHEN WE TAKE CONVOLUTION FOR A SPACE MATRIX WHICH ONLY CONSIST ONLY ONE CENTER ELEMENT WE GET ANOTHER SPACE MATRIX WHICH ONLY CONSIST ONLY ONE CENTER ELEMENTTHESE ELEMENTS ARE THE ELEMENTS OF THE OTHER MATRIX WE CONVOLVED WITH IT.

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
x = mpimg.imread('Allen Keys.jpg')
fig, ax = plt.subplots(1,2,figsize=(8,8))
ax[0].imshow(x, cmap='gray')
R, G, B = x[:,:,0], x[:,:,1], x[:,:,2]
imgGray = 0.2989 * R + 0.5870 * G + 0.1140 * B



hmn=np.array([-1,0,-1,-2,0,2,-1,0,1])
shape3=(3,3)
h=hmn.reshape(shape3)


y=signal.convolve2d(imgGray,h)
ax[1].imshow(y, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f511820bbd0>
```