Labsheet 2
Lab - 2 basics of signal processing

srikant nayak
(SC18M005)

22 August 2018

sub:Image and Video Processing Lab **Department**: MATHEMATICS (Machine learning)
Date of Lab sheet: August 16, 2018

Question 1:

Write a Python program to: beginenumerate[label=()]

Basic Signal Operations
Let x[n] be a signal with x[n] = [1, 2, 5, 6, 3, 2, 1, 1, 1]
Sketch the following signals:

x[n 4]

x[n]

x[2 n]

2x[n]

x[3n]

x[n/3]

Discussion

the aim of the above program is to to get familiar with shifting and scaling of given signal

algorithm

step 1: import numpy and matplotlib.pyplot librery
step 2: form an array
step 3: array the index from given range
step 4: write condition for scaling or shifting
step 5: plot it

program

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,2,5,6,3,2,1,1,1])
n=np.arange(0,len(x))
n=n+4
plt.stem(n,x)i

    import numpy as np
import matplotlib.pyplot as plt
```

```
x=np.array([1,2,5,6,3,2,1,1,1])
n=np.arange(0,len(x))
n=-n
plt.stem(n,x)
```

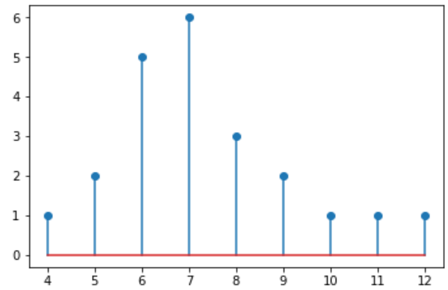
```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,2,5,6,3,2,1,1,1])
n=np.arange(0,len(x))
n=2-n
plt.stem(n,x)
```

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,2,5,6,3,2,1,1,1])
p=x*2
n=np.arange(0,len(x))
plt.stem(n,p)
```

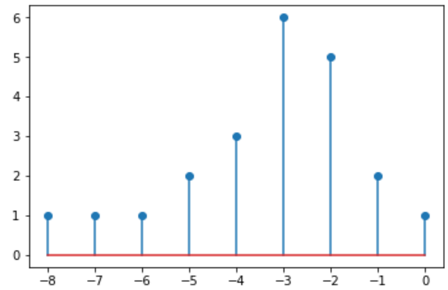
```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,2,5,6,3,2,1,1,1])
n=np.arange(0,len(x))
n=n/3
plt.stem(n,x)
```

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,2,5,6,3,2,1,1,1])
n=np.arange(0,len(x))
n=n*3
plt.stem(n,x)
```

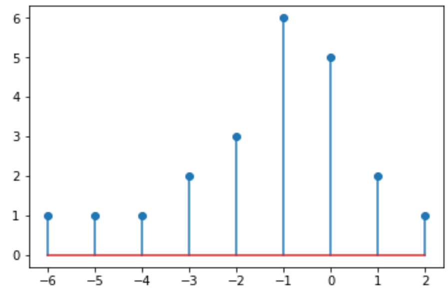
Result 1



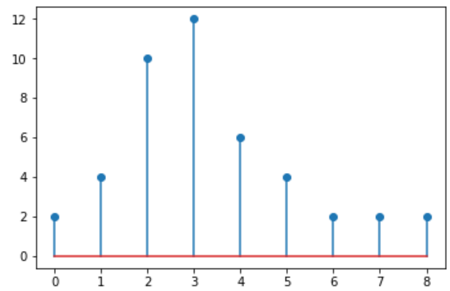
a



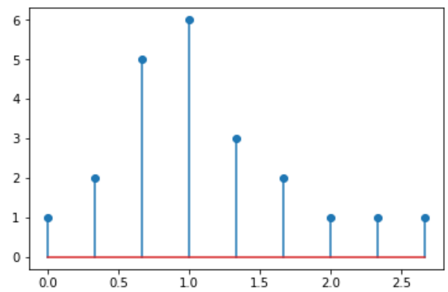
b



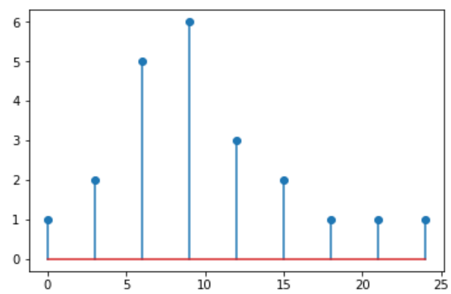
c



d



e



f

inference

the output of the image is according to the given signal

Question 2:

1-D convolution:
Perform the following 1-D convolutions between signals x[n] and h[n]:

(a) x[n] = [1, 1, 1, 1, 1] , h[n] = [0, 0, 1, 1, 1, 1, 1, 1, 0]

(b) x[n] = (0.5)nu[n 4] , h[n] = 4n[2 n]

(c) Consider the evaluation of y[n] = x1[n] x2[n] x3[n] where, x1[n] = (0.5)nu[n] ,x2[n] = u[n 3] ,x3[n] = [n] [n 1].

Discussion

when a signal is passed into a system the output from the system is the convolution of input signal x[n] and impulse response h[n] .
Convolution is a formal mathematical operation, just as multiplication, addition, and integration.

Algorithm

step 1: import librery according to operation
step 2: form 2 array , one is x[n] and other is h[n]
step 3; convolute the 2 array with the help of numpy librery
step 4: print the output

program

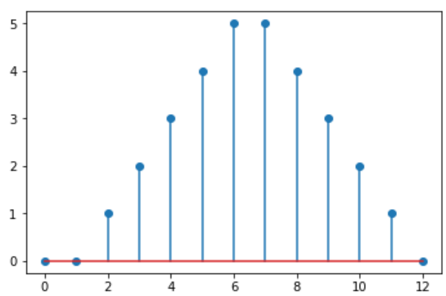
```
(a)import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,1,1,1,1])
h=np.array([0,0,1,1,1,1,1,1,0])
convolution=np.convolve(x,h)
print(convolution)
plt.stem(convolution)
```

```
(b) import numpy as np
import matplotlib.pyplot as plt
import math
n=np.array([0,1,2,3,4,5])
x=np.array([1 for i in range(0,len(n))],dtype=float)
h=np.array([1 for j in range(0,len(n))],dtype=float)
x=x+4
h=-2-h
for i in range(len(x)):
n[i]=math.pow(0.5,i)
for j in range(len(h)):
h[j]=math.pow(4,j) convolve=np.convolve(x,h)
plt.stem(convolve)
plt.show()
```

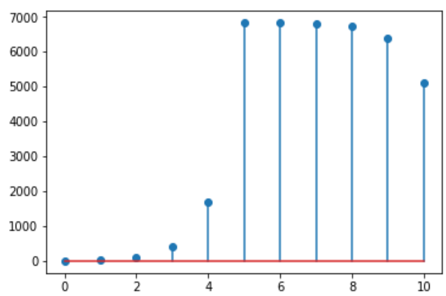
```
(c) import matplotlib.pyplot as plt
import numpy as np
p=list()
q=list()
r=list()
```

```
for i in range(0,8):
p.append((0.5)**i)
q=[0,0,0,1,1,1,1]
r=[1,-1]
plt.subplot(221)
plt.stem(p)
plt.subplot(p)
plt.subplot(222) plt.stem(q)
plt.ylabel(q)
plt.subplot(223)
plt.stem(r)
plt.ylabel(r)
z=np.convolve(p,q)
z=np.convolve(z,r)
plt.subplot(224)
plt.stem(z)
plt.ylabel(z)
```

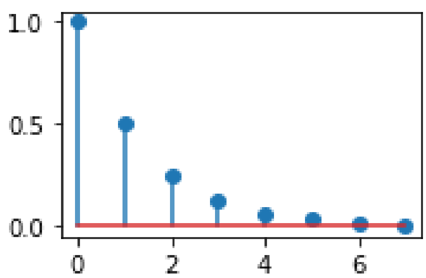
result



a



b



c

inference

the given output is the convolution of given signals

question 3:

2-D convolution:
Perform the following 2-D convolutions and comment on the kind of filtering obtained in the result

(a)x= $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 6 & 7 & 8 \end{bmatrix}$ H= $\begin{bmatrix} 1 & -2 & -1 \\ 1 & 0 & -1 \\ 1 & 2 & -1 \end{bmatrix}$

(b) use the cameraman image as input and kernel to be H=1/9 $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 6 & 7 & 8 \end{bmatrix}$

(c)Use the cameraman image as input and the kernel to be H= $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Discussion

In image processing, a kernel, convolution matrix, or mask is a small matrix. It is used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between a kernel and an image.

Algorithm

step 1: form 2 matrix step 2: convolute 2D through spicy import signal step 3 input cameraman image in the form of tif and convolute though image filter step 4 save the output image

program

(a) from scipy import signal
x=([1,2,3],[4,5,6],[6,7,8])
h=([1,-2,-1],[1,0,-1],[1,2,-1])
convolution=signal.convolve2d(x,h)
print(convolution)

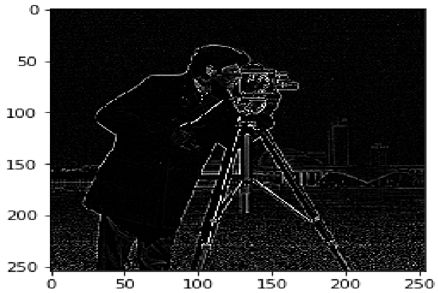
(b) from PIL import Image,ImageFilter
import numpy as np
img = Image.open('cameraman.tif')
c=ImageFilter.Kernel((3,3),np.ones(9)*1/9)
img.save('camera.png')

(c) from skimage import io , color
import matplotlib.pyplot as plt
import numpy as np
from scipy import signal
from PIL import Image
img=io.imread("cameraman.tif")
img=color.rgb2gray(img)
h=np.array([[0,-1,0], [-1,4,-1], [0,-1,0]])
y=signal.convolve2d(img,h,'valid')
y=color.gray2rgb(y)
plt.imshow(y)

result

(A) [[1 0 -2 -8 -3] [5 -1 -6 -19 -9] [11 4 -4 -24 -17] [10 20 14 0 -14] [6 19 16 9 -8]]
(b)

a



b

inference

from the above matrix we formed the blurred and edge detection through convolution 2d

question 4:

convolution as multiplication: Prove that convolution in spatial domain is multiplication in frequency domain by applying FFT.

(a) Find the FFT of the cameraman image and display the magnitude spectrum

(b) Find the FFT of the kernel in question 3(b)

(c) Multiply the two FFTs and take Inverse FFT of the product

(d) Compare it with your result in question 3(b)

Discussion

FFT convolution uses the principle that multiplication in the frequency domain corresponds to convolution in the time domain. The input signal is transformed into the frequency domain using the DFT, multiplied by the frequency response of the filter, and then transformed back into the time domain using the Inverse DFT

Algorithm

step 1: read the image in tif format
. Step 2:Convert the image from RGB to GRAY level
. Step 3:put the fft function to image
Step 4:display the magnitude spectrum

program

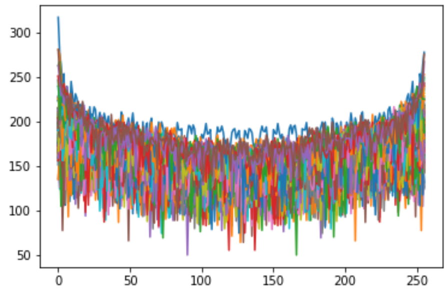
(a) import numpy as np
from skimage import io,color
import matplotlib.pyplot as plt
from scipy import fftpack
img=io.imread('cameraman.tif')
img=color.rgb2gray(img)
p=fftpack.fft2(img)
mag$_{s}pect = 20 * np.log(np.abs(p))$
$plt.plot(mag_{s}pect)$

(B) import numpy
from skimage import io,color
from PIL import Image , ImageFilter
from scipy import fftpack
a=ImageFilter.Kernel((3,3),numpy.ones(9)*1/9)
a=color.rgb2gray(img)
s=fftpack.fft2(img)
print(s)

(c) import numpy
import scipy
from skimage import io,color
import matplotlib.pyplot as plt
from PIL import Image , ImageFilter
from scipy import fftpack
img=io.imread('cameraman.tif')
a=ImageFilter.Kernel((3,3),numpy.ones(9)*1/9)
a=color.rgb2gray(img)
b=fftpack.fft2(a)
img=color.rgb2gray(img)
c=fftpack.fft2(img)
multi=b*c
print(multi)
inverse=scipy.fftpack.ifft(multi)
print(inverse)

result

(a)



a

(b) [[7780728. +0.j 479869.37901948+1167475.72134067j 382091.16744448 -344064.3500564j ... 32037.18287967 +279751.14729754j 382091.16744448 +344064.3500564j 479869.37901948-1167475.72134067j] [694675.1054863 -836574.80114987j -591204.33605153 -768229.11424111j -291668.0430127 +569278.93924113j ... -83875.04131211 -137195.71995934j 122819.1267773 +232945.05055378j -228754.75396057 +214106.94106031j] [190981.79570956 -495893.92934818j -222355.01079183 +225322.31088183j 80847.82930646 +124223.23457101j ... -104458.91757817 +128879.70980572j -51622.00056568 +80830.52066902j 181561.79918424 +101211.91328498j] ... [-167211.67596147 +95181.17949364j -104680.94479169 -53130.37855467j 55467.66444863 +132785.8778589j ... -104641.85817434 +128394.5836996j 144227.71434895 -25187.94086001j -67538.32285089 -55419.27009776j] [190981.79570956 +495893.92934818j 181561.79918424 -101211.91328498j -51622.00056568 -80830.52066902j ... 154153.34913451 +22133.05340021j 80847.82930646 -124223.23457101j -222355.01079183 -225322.31088183j] [694675.1054863 +836574.80114987j -228754.75396057 -214106.94106031j 122819.1267773 -232945.05055378j ... 174123.46893065 -80145.60887758j -291668.0430127 -569278.93924113j -591204.33605153 +768229.11424111j]]

(c) [[6.05397282e+13+0.00000000e+00j -1.13272494e+12+1.12047170e+12j 2.76133833e+10-2.62927898e+11j ... -7.72343233e+10+1.79248773e+10j 2.76133833e+10+2.62927898e+11j -1.13272494e+12-1.12047170e+12j] [-2.17283896e+11-1.16229538e+12j -2.40653405e+11+9.08360767e+11j -2.39008263e+11-3.32080948e+11j ... -1.17876430e+10+2.30145934e+10j -3.91788587e+10+5.72202154e+10j 6.48695525e+09-9.79559612e+10j] [-2.09436743e+11-1.89413426e+11j -1.32839296e+09-1.00203090e+11j -8.89504050e+09+2.00863577e+10j ... -5.69831414e+09-2.69252700e+10j -3.86874213e+09-8.34526637e+09j 2.27208355e+10+3.67524341e+10j] ... [1.89002876e+10-3.18308091e+10j 8.13526308e+09+1.11234764e+10j -1.45554276e+10+1.47306450e+10j ... -5.53525064e+09-2.68708956e+10j 2.01672012e+10-7.26559828e+09j 1.49012956e+09+7.48584911e+09j] [-2.09436743e+11+1.89413426e+11j 2.27208355e+10-3.67524341e+10j -3.86874213e+09+8.34526637e+09j ... 2.32733830e+10+6.82376862e+09j -8.89504050e+09-2.00863577e+10j -1.32839296e+09+1.00203090e+11j] [-2.17283896e+11+1.16229538e+12j 6.48695525e+09+9.79559612e+10j -3.91788587e+10-5.72202154e+10j ... 2.38956638e+10-2.79104629e+10j -2.39008263e+11+3.32080948e+11j -2.40653405e+11-9.08360767e+11j]] [[2.27130736e+11-2.78907828e-06j 2.27002625e+11-3.00048850e-06j 2.26836265e+11-2.91397331e-06j ... 2.27525939e+11-1.83307566e-06j 2.27358244e+11-2.34180906e-08j 2.27237880e+11-1.84332021e-06j] [-2.89639702e+09-2.11226030e+09j -2.92269536e+09-2.16426435e+09j -2.95836754e+09-2.23685153e+09j ... -2.74493890e+09-1.98678312e+09j -2.80543368e+09-2.01758215e+09j -2.85758976e+09-2.06356244e+09j] [-9.47712777e+08-1.08472652e+09j -9.64247605e+08-1.07955758e+09j -9.83237710e+08-1.07628115e+09j ... -8.41789942e+08-1.16763600e+09j -8.89347518e+08-1.12839466e+09j -9.25088452e+08-1.10069560e+09j] ... [-6.70209682e+07-5.87387494e+07j -9.13250562e+07+1.11119014e+07j -1.04136708e+08+8.27981895e+07j ... 1.00647614e+08-2.44858264e+08j 3.30653639e+07-1.94669607e+08j -2.37869117e+07-1.32508570e+08j] [-9.47712777e+08+1.08472652e+09j -9.64247605e+08+1.07955758e+09j -9.83237710e+08+1.07628115e+09j ... -8.41789942e+08+1.16763600e+09j -8.89347518e+08+1.12839466e+09j -9.25088452e+08+1.10069560e+09j] [-2.89639702e+09+2.11226030e+09j -2.92269536e+09+2.16426435e+09j -2.95836754e+09+2.23685153e+09j ... -2.74493890e+09+1.98678312e+09j -2.80543368e+09+2.01758215e+09j -2.85758976e+09+2.06356244e+09j]]

(D) we are getting same matrix of image that of 3(b)

inference

i have found the fft of cameraman image and its magnitude spectrum i also multiplication and addition through fft