Experiment No. 2 : Familiarisation of SciPy Library

Aim

1. To familiarise with the Python library SciPy
2. To Perform Linear Convolution as a Toeplitz matrix operation
3. To perform 2D convolution

```
# importing the scipy and numpy package
from scipy import linalg
import numpy as np


#declaring the numpy arrray

A = np.array([[1,4],[1,2]])
B = np.array([[1,3,4,5],[3,4,2,4],[1,3,4,5],[3,4,2,4]])


#passing the array to det function to the diterminant of A

x = linalg.det(A)
print("|A| : ", x)

y = linalg.det(B)
print("|B| :",y)
```

```
     |A| :  -2.0
     |B| : 0.0
```

```
C = np.array([1,2,3])
D = np.array([0,1,0.5])
z = np.convolve(C,D) # by default mode is taken as full
print(z)
```

```
     [0.  1.  2.5 4.  1.5]
```

```
#linear convolution as toeplitz matrix operation

ip = [1,2,0,-3,0.5]
N = print(len(ip))
```

```
     5
```

```
h = np.array([-1,4,-2])
M = print(len(h))
op = np.convolve(x,h) # by default mode is taken as full , ie. length of the op is N+M-1
print(op)
```

```
     3
     [ -1   2   3   4   5  12 -10   0   0   0   0   0]
```

Result : Performed Linear Operation using Toeplitz matrix.

```
from scipy.linalg import convolution_matrix  # for generating toeplitz matrix for h

H = convolution_matrix([-1,4,-2],5,mode='full')
print(H)
```

```
     [[-1  0  0  0  0]
      [ 4 -1  0  0  0]
      [-2  4 -1  0  0]
      [ 0 -2  4 -1  0]
      [ 0  0 -2  4 -1]
      [ 0  0  0 -2  4]
      [ 0  0  0  0 -2]]
```

```
Y1 = H@ip
print(Y1)
```

```
     [ -1.    2.    6.   -1.  -12.5   8.   -1. ]
```

```
import scipy as sy

# impulse resoponse

h = [1,2,3,3,2,1]

#input response

x = [1,2,3,4,5]

N1 = len(x)
N2 = len(h)
N = N1 + N2 -1
y = np.zeros(N)

#linear convolution using built-in function

y1 = np.convolve(x,h)
m = N - N1
n = N - N2

x = np.pad(x,(0,m),'constant')
h = np.pad(h,(0,n),'constant')

for n in range(N):
  for k in range(N):
    if n >= k:
      y[n] = y[n] + x[n-k]*h[k]

print('Linear Convolution using for convolution sum formula output response : ' , y)
print('Linear convolution using numpy built-in function output response : ',y1)
```

```
Linear Convolution using for convolution sum formula output response :  [ 1.  4. 10.
Linear convolution using numpy built-in function output response :  [ 1  4 10 19 30
```

Double-click (or enter) to edit

```
 # 2d covolution example from text

from scipy import signal as sg

X = [[2,1],[5,4],[3,1]]

H = [[1,1],[-1,1]]

Y2 = sg.convolve(X,H,'full')
print(Y2) # matrix is 90 degrees clock wise rotation of cartesian co-ordinates
```

```
[[ 2  3  1]
 [ 3 10  5]
 [-2  5  5]
 [-3  2  1]]
```

Result : Obtained the 2D linear convolution.

```
# circular convolution as circulant matrix operation

from scipy.linalg import circulant
import numpy as np

c = circulant([1,2,3])
c
```

```
array([[1, 3, 2],
       [2, 1, 3],
       [3, 2, 1]])
```

```
x = np.array([[1],[1],[-1]])
c = circulant([1,2,3])
y = c@x
y
```

```
array([[2],
       [0],
       [4]])
```