ASSIGNMENT - 3

Q. 1) Write the proof and various properties related to 2D DFT.

DFT: $F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$

① Linearity : If $f_1(x,y) \longleftrightarrow F_1(u,v)$

$f_2(x,y) \longleftrightarrow F_2(u,v)$

then, $a f_1(x,y) + b f_2(x,y) \longleftrightarrow a F_1(u,v) + b F_2(u,v)$

Proof: $DFT\{a f_1(x,y) + b f_2(x,y)\} = \sum_x \sum_y (a f_1(x,y) + b f_2(x,y))\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$

$= a \sum_x \sum_y f_1(x,y)\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)} + b \sum_x \sum_y f_2(x,y)\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$

$= a F_1(u,v) + b F_2(u,v)$

② Separability : Kernel is separable.

$f_1(x)\, f_2(y) \longleftrightarrow F_1(u)\, F_2(v)$

Proof.

$DFT\{f_1(x) f_2(y)\} = \sum_x \sum_y f_1(x) f_2(y)\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$

$= \sum_x f_1(x)\, e^{-j\frac{2\pi ux}{M}} \sum_y f_2(y)\, e^{-j\frac{2\pi vy}{N}} = F_1(u)\, F_2(v)$

③ Periodicity : (M,N): $F(u,v) = F(u+M, v) = F(u, v+N) = F(u+M, v+N)$

Proof: $\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{(u+M)x}{M} + \frac{(v+N)y}{N}\right)} = F(u+M, v+N)$

$= \sum_x \sum_y f(x,y)\, \underbrace{e^{-j2\pi(M)}}_{1}\, \underbrace{e^{-j2\pi N}}_{1}\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)} = F(u,v)$

④ Translation:

Spatial domain: $f(x-x_0, y-y_0) \longleftrightarrow F(u,v)\, e^{-j2\pi\left(\frac{ux_0}{M} + \frac{vy_0}{N}\right)}$

Proof: $DFT\{f(x-x_0, y-y_0)\} = \sum_x \sum_y f(x-x_0, y-y_0)\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$

$= \sum_x \sum_y f(x,y)\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}\, e^{-j2\pi\left(\frac{ux_0}{M} + \frac{vy_0}{N}\right)}$

$= F(u,v)\, e^{-j2\pi\left(\frac{ux_0}{M} + \frac{vy_0}{N}\right)}$

Freq. domain: $f(x,y)\, e^{j2\pi\left(\frac{u_0 x}{M} + \frac{v_0 y}{N}\right)} \longleftrightarrow F(u-u_0, v-v_0)$

Proof: $DFT\left\{f(x,y)\, e^{j2\pi\left(\frac{u_0 x}{M} + \frac{v_0 y}{N}\right)}\right\} = \sum_x \sum_y f(x,y)\, e^{-j2\pi\left[\left(\frac{ux}{M} + \frac{vy}{N}\right) - \left(\frac{u_0 x}{M} + \frac{v_0 y}{N}\right)\right]}$

$= \sum_x \sum_y f(x,y)\, e^{-j2\pi\left(\frac{(u-u_0)x}{M} + \frac{(v-v_0)y}{N}\right)}$

$= F(u-u_0, v-v_0)$

⑤ Symmetry: (i) $f(x,y)$ real: $F^*(u,v) = F(-u,-v)$

Proof: $F^*(u,v) = \sum_x \sum_y \underset{\underset{\text{real}}{\Downarrow}}{f^*(x,y)}\, e^{j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$

$= \sum_x \sum_y f(x,y)\, e^{-j2\pi\left(-\frac{ux}{M} - \frac{vy}{N}\right)} = F(-u,-v)$

(ii) $f(x,y)$ imaginary: $F^*(u,v) = -F(u,v)$

Proof: $F^*(u,v) = \sum_x \sum_y f^*(x,y)\, e^{j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$

$= -\sum_x \sum_y f(x,y)\, e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)} = -F(u,v)$

(6) Convolution (Circular) : $f_1(x,y) \circledast f_2(x,y) \leftrightarrow F_1(u,v)\, F_2(u,v)$

Proof: $DFT\{f_1(x,y) \circledast f_2(x,y)\} = \sum_x \sum_y \left(f_1(x,y) \circledast f_2(x,y)\right) e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$

$$= \sum_x \sum_y \sum_{x'} \sum_{y'} f_1(x',y')\, f_2\left((x-x')_M, (y-y')_N\right) e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

$$= \sum_{x'} \sum_{y'} f_1(x',y') \sum_x \sum_y f_2\left((x-x')_M, (y-y')_N\right) e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

$$= \sum_{x'} \sum_{y'} f_1(x',y') \underbrace{\sum_{x''} \sum_{y''} f_2(x'',y'')\, e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}}\, e^{-j2\pi\left(\frac{ux'}{M}+\frac{vy'}{N}\right)}$$

$$= F_2(u,v) \underbrace{\sum_{x'} \sum_{y'} f_1(x',y')\, e^{-j2\pi\left(\frac{ux'}{M}+\frac{vy'}{N}\right)}} = F_1(u,v)\, F_2(u,v)$$

(7) Average value : $\frac{1}{N^2} F(0,0) = \bar{f}(x,y)$

$$F(0,0) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)\, e^{-j2\pi\left(\frac{0x}{M}+\frac{0y}{N}\right)} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)$$

Multiply by $1/N^2$

$$\Rightarrow \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) = \frac{1}{N^2} F(0,0)$$

Q2

March 2, 2021

Q.2. (a) Write python from scratch for computing 2D DFT{X(k,l)} of the following 2D array

(i) x(m,n) = np.array([[1, 0],[2, 1]]) (ii) x(m,n) = np.array([[1,2, 3,4], [5, 6, 7, 8], [9,10,11,12], [13,14,15,16]])

(b) Verify the results of part (a) by analytical solution method (i.e. pen and paper based solution)

[27]:
```python
import numpy as np
import cmath

def DFT2D(x):
    M, N = x.shape
    dft2d = np.zeros((M,N),dtype=complex)
    for k in range(M):
        for l in range(N):
            sum_matrix = 0.0
            for m in range(M):
                for n in range(N):
                    e = cmath.exp(- 2j * np.pi * ((k * m) / M + (l * n) / N))
                    sum_matrix +=  x[m,n] * e
            dft2d[k,l] = sum_matrix
    return dft2d

x1=np.array([[1, 0],[2, 1]])
y1=np.round(DFT2D(x1),2)
print(y1)

x2=np.array([[1,2, 3,4], [5, 6, 7, 8], [9,10,11,12], [13,14,15,16]])
y2=np.round(DFT2D(x2),2)
print("\n",y2)
```

```
[[ 4.+0.j  2.-0.j]
 [-2.-0.j  0.+0.j]]

 [[136. +0.j  -8. +8.j  -8. -0.j  -8. -8.j]
 [-32.+32.j   0. +0.j   0. +0.j  -0. +0.j]
 [-32. -0.j   0. +0.j   0. +0.j  -0. +0.j]
 [-32.-32.j  -0. +0.j  -0. +0.j  -0. +0.j]]
```

Q.2 b) $F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$

(i) $x = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$.

$F(0,0) = \sum_{x=0}^{1} \sum_{y=0}^{1} f(x,y) = 1+0+2+1 = \underline{\underline{4}}$

$F(0,1) = \sum_{x=0}^{1} \sum_{y=0}^{1} f(x,y) e^{-j\frac{2\pi y}{2}} = f(0,0) + f(0,1)e^{-j\pi} + f(1,0) + f(1,1)e^{-j\pi}$

$= 1 + (-0) + (2) + (-1) = \underline{\underline{2}}$

$F(1,0) = \sum_{x=0}^{1} \sum_{y=0}^{1} f(x,y) e^{-j\pi x} = f(0,0) + f(0,1) + f(1,0)e^{-j\pi} + f(1,1)e^{-j\pi}$

$= 1+0-2-1 = \underline{\underline{-2}}$

$F(1,1) = \sum_{x=0}^{1} \sum_{y=0}^{1} f(x,y) e^{-j\pi(x+y)} = f(0,0) + f(0,1)e^{-j\pi} + f(1,0)e^{-j\pi} + f(1,1)e^{-j2\pi}$

$= 1-0-2+1 = \underline{\underline{0}}$

$\Rightarrow F(u,v) = \begin{bmatrix} 4 & 2 \\ -2 & 0 \end{bmatrix}$

(ii) $x = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$

$F(0,0) = \sum_{x=0}^{3} \sum_{y=0}^{3} f(x,y) = 1+2+3+4+\dots+16 = \underline{\underline{136}}$

$F(0,1) = \sum_{x=0}^{3} \sum_{y=0}^{3} f(x,y) e^{-j\frac{2\pi y}{4}} = f(0,0) + f(0,1)(-j) + f(0,2)(-1) + f(0,3)j$
$+ f(1,0) + f(1,1)(-j) + f(1,2)(-1) + f(1,3)j$
$+ f(2,0) + f(2,1)(-j) + f(2,2)(-1) + f(2,3)j$
$+ f(3,0) + f(3,1)(-j) + f(3,2)(-1) + f(3,3)j$

$= 28 + 32(-j) + 36(-1) + 40j = \underline{\underline{-8+8j}}$

$F(0,2) = \sum_{x=0}^{3} \sum_{y=0}^{3} f(x,y) e^{-j\frac{2\pi 2y}{4}} = f(0,0) - f(0,1) + f(0,2) - f(0,3)$
$+ f(1,0) - f(1,1) + f(1,2) - f(1,3)$
$+ f(2,0) - f(2,1) + f(2,2) - f(3,3)$
$+ f(3,0) - f(3,1) + f(3,2) - f(3,3)$

$= 28-32+36-40 = \underline{\underline{-8}}$

$$F(0,3) = \sum_{x=0}^{3}\sum_{y=0}^{3} f(x,y)\, e^{-j 2\pi \left(\frac{3y}{\cancel{4}2}\right)} = f(0,0) + f(0,1)(j) + f(0,2)(-1) + f(0,3)(-j)$$
$$+ f(1,0) + \cdots$$

$$= 28 + 32(j) + 36(-1) + 40(-j) = \underline{\underline{-8-8j}}$$

$$F(1,0) = \sum_{x=0}^{3}\sum_{y=0}^{3} f(x,y)\, e^{-j \frac{2\pi x}{\cancel{4}2}} = f(0,0) + f(0,1) + f(0,2) + f(0,3)$$
$$+ f(1,0)(-j) + f(1,1)(-j) + f(1,2)(-j) + f(1,3)(-j)$$
$$+ f(2,0)(-1) + f(2,1)(-1) + f(2,2)(-1) + f(2,3)(-1)$$
$$+ f(3,0)(j) + f(3,1)(j) + f(3,2)(j) + f(3,3)\, j$$

$$= 10 - 26j - 42 + 58j = \underline{\underline{-32+32j}}$$

$$F(1,1) = \sum_{x=0}^{3}\sum_{y=0}^{3} f(x,y)\, e^{-j 2\pi \left(\frac{x+y}{2}\right)} = f(0,0) + f(0,1)(-j) + f(0,2) + f(0,3)(-j)$$
$$= f(1,0)(-j) + f(1,1) + \cdots$$

$$= 68j - 68j = \underline{\underline{0}}$$

$$F(1,2) = \sum_{x=0}^{3}\sum_{y=0}^{3} f(x,y)\, e^{-j 2\pi \left(\frac{x}{4} + \frac{2y}{4}\right)} = \underline{\underline{0}}$$

$$F(1,3) = \sum_{x=0}^{3}\sum_{y=0}^{3} f(x,y)\, e^{-j 2\pi \left(\frac{x}{4} + \frac{3y}{4}\right)} = f(0,0) + \cdots$$

$$= \underline{\underline{0}}$$

$$F(2,0) = \sum_{x=0}^{3}\sum_{y=0}^{3} f(x,y)\, e^{-j \frac{2\pi 2x}{4}} = f(0,0) + f(0,1) + f(0,2) + f(0,3)$$
$$- f(1,0) - f(1,1) - f(1,2) - f(1,3)$$
$$+ f(2,0) + f(2,1) + f(2,2) + f(2,3)$$
$$- f(3,0) - f(3,1) - f(3,2) - f(3,3)$$

$$= 10 - 26 + 42 - 58 = \underline{\underline{-32}}$$

$$F(2,1) = \sum_{x=0}^{3}\sum_{y=0}^{3} f(x,y)\, e^{-j 2\pi \left(\frac{2x}{4} + \frac{y}{4}\right)} = f(0,0) + f(0,1)(-j) + f(0,2)(-1) + f(0,3)(j)$$
$$+ \cdots$$

$$= \underline{\underline{0}}$$

$$F(2,2) = \sum_{x=0}^{3}\sum_{y=0}^{3} f(x,y)\, e^{-j \pi (x+y)} = f(0,0) - f(0,1) + f(0,2) - f(0,3)$$

$$\begin{array}{cccc} - & + & - & + \\ + & - & + & - \\ - & + & - & + \end{array}$$

$$= 68 - 68 = \underline{\underline{0}}$$

$$F(2,3) = \sum_{x=0}^{3}\sum_{y=0}^{3} f(x,y)\, e^{-j 2\pi \left(\frac{2x}{4} + \frac{3y}{4}\right)} = f(0,0) + \cdots$$

$$= \underline{\underline{0}}$$

$$F(3,0) = \sum_{x=0}^{3} \sum_{y=0}^{3} f(x,y)\, e^{-j2\pi\left(\frac{3x}{4}\right)} = f(0,0) + f(0,1) + f(0,2) + f(0,3)$$

$$-j\left(f(1,0) + f(1,1) + f(1,2) + f(1,3)\right)$$

$$-1\left(f(2,0) + f(2,1) + f(2,2) + f(2,3)\right)$$

$$j\left(f(3,0) + f(3,1) + f(3,2) + f(3,3)\right)$$

$$= 10 - 26j - 42 + 58j = \underline{\underline{-32-32j}}$$

$$F(3,1) = \sum_{x=0}^{3} \sum_{y=0}^{3} f(x,y)\, e^{-j2\pi\left(\frac{3x}{4}+\frac{y}{4}\right)} = f(0,0) + f(0,1)(-j) + \ldots$$

$$= \underline{\underline{0}}$$

$$F(3,2) = \sum_{x=0}^{3} \sum_{y=0}^{3} f(x,y)\, e^{-j2\pi\left(\frac{3x}{4}+\frac{2y}{4}\right)} = f(0,0) + f(0,1)(-1) + \ldots$$

$$= \underline{\underline{0}}$$

$$F(3,3) = \sum_{x=0}^{3} \sum_{y=0}^{3} f(x,y)\, e^{-j2\pi\left(\frac{3x}{4}+\frac{3y}{4}\right)} = f(0,0) + f(0,1)\,j + \ldots$$

$$= \underline{\underline{0}}$$

$$F(u,v) = \begin{bmatrix} 136 & -8+8j & -8 & -8-8j \\ -32+32j & 0 & 0 & 0 \\ -32 & 0 & 0 & 0 \\ -32-32j & 0 & 0 & 0 \end{bmatrix}$$

Lab Assignment 3

March 3, 2021

Q.3. (a) Write python from scratch for 2D Circular convolution using Doubly Block Circulant matrices method between input=np.array([[1,2,3],[4,5,6],[7,8,9]])and filter=np.array([[1,2,1],[0,0,0],[-1,-2,-1]])

(b) verify the result of part (a) by analytical solution method (i.e. pen and paper-based solution)

In [2]:
```
import numpy as np
from scipy import signal

#a
i=np.array([[1,2,3],[4,5,6],[7,8,9]])
f=np.array([[1,2,1],[0,0,0],[-1,-2,-1]])

#i=np.array([[1,2,1],[1,3,-1],[0,1,0]])
#f=np.array([[1,-1,0],[1,0,0],[0,0,0]])
#f=np.array([[1,-1],[1,0]])


print("Input=\n",i,"\nFilter=\n",f)

def cir_convolv2D(i,f):
    result_shape=[max(i.shape[0],f.shape[0]),max(i.shape[1],f.shape[1])]
    print("\nResult shape: ",result_shape)

    # Padding filter with zeros based on input size
    f1=np.zeros(i.shape)
    f1[0:f.shape[0],0:f.shape[1]] = f

    #Convert input to vector
    inp_vector = i.flatten()

    h=np.zeros((f1.shape[0],f1.shape[1],i.shape[1]))
    for row in range(f1.shape[0]):
        for col in range(i.shape[1]):
            h[row,:,col]=np.roll(f1[row,:],col).transpose()

    #print("Computing")
```

```python
        H1 = H = np.hstack(np.concatenate(([h[0]],h[::-1][:i.shape[0]-1]),axis=0))
        for ctr in range(1,h.shape[0]):
            H1 = np.hstack((h[ctr],H1[:,:-i.shape[1]]))
            H = np.vstack((H,H1))
        print("Doubly Block Circulant Matrix")
        print(H)
        result = np.matmul(H, inp_vector)
        result = result.reshape(result_shape)

        return result

    result=cir_convolv2D(i,f)
    print("\nResult:\n",result)

    # Scipy method
    result_scipy=signal.convolve2d(i,f,boundary='wrap')[0:max(i.shape[0],f.shape[0]),0:max
    print("\nScipy result:\n",result_scipy)
```

```
Input=
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Filter=
 [[ 1  2  1]
 [ 0  0  0]
 [-1 -2 -1]]

Result shape:  [3, 3]
Doubly Block Circulant Matrix
[[ 1.  1.  2. -1. -1. -2.  0.  0.  0.]
 [ 2.  1.  1. -2. -1. -1.  0.  0.  0.]
 [ 1.  2.  1. -1. -2. -1.  0.  0.  0.]
 [ 0.  0.  0.  1.  1.  2. -1. -1. -2.]
 [ 0.  0.  0.  2.  1.  1. -2. -1. -1.]
 [ 0.  0.  0.  1.  2.  1. -1. -2. -1.]
 [-1. -1. -2.  0.  0.  0.  1.  1.  2.]
 [-2. -1. -1.  0.  0.  0.  2.  1.  1.]
 [-1. -2. -1.  0.  0.  0.  1.  2.  1.]]

Result:
 [[-12. -12. -12.]
 [-12. -12. -12.]
 [ 24.  24.  24.]]

Scipy result:
 [[-12 -12 -12]
 [-12 -12 -12]
 [ 24  24  24]]
```

171EC141
171EC146
171EC211

3.) b) 2-D Circular Convolution using Doubly Block Circulant matrix

$$f[m,n] = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad h[m,n] = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$\Rightarrow M_1 = 3, \quad N_1 = 3 \qquad M_2 = 3 \;\&\; N_2 = 3$

$M_1 = M_2 \;\&\; N_1 = N_2$

$$y[m,n] = f[m,n] \circledast h[m,n]$$

Length of $y[m,n] = M \times N$

$M = Max(M_1, M_2) = 3$

$N = Max(N_1, N_2) = 3$

Constructing Circulant Matrix for each row of $h[m,n]$

$$H_j = \begin{bmatrix} h[j,0] & h[j,N-1] & \cdots & h[j,1] \\ h[j,1] & h[j,0] & \cdots & h[j,2] \\ \vdots & & & \\ h[j,N-1] & h[j,N-2] & \cdots & h[j,0] \end{bmatrix}_{N \times N}$$

where $j = 0, 1, \ldots, M-1$

$$H_0 = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix} \qquad H_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad H_2 = \begin{bmatrix} -1 & -1 & -2 \\ -2 & -1 & -1 \\ -1 & -2 & -1 \end{bmatrix}$$

Doubly-Block Circulant matrix

$$H = \begin{bmatrix} H_0 & H_{M-1} & H_{M-2} & \cdots & H_1 \\ H_1 & H_0 & H_{M-1} & \cdots & H_2 \\ H_2 & H_1 & H_0 & \cdots & H_3 \\ \vdots & \vdots & \vdots & & \vdots \\ H_{M-1} & H_{M-2} & H_{M-3} & \cdots & H_0 \end{bmatrix}$$

Number of columns $= N =$ number of columns of $f[m,n]$

$$H = \begin{bmatrix} H_0 & H_2 & H_1 \\ H_1 & H_0 & H_2 \\ H_2 & H_1 & H_0 \end{bmatrix}$$

$$g = Hf = \left[\begin{array}{ccc|ccc|ccc} 1 & 1 & 2 & -1 & -1 & -2 & 0 & 0 & 0 \\ 2 & 1 & 1 & -2 & -1 & -1 & 0 & 0 & 0 \\ 1 & 2 & 1 & -1 & -2 & -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 2 & -1 & -1 & -2 \\ 0 & 0 & 0 & 2 & 1 & 1 & -2 & -1 & -1 \\ 0 & 0 & 0 & 1 & 2 & 1 & -1 & -2 & -1 \\ \hline -1 & -1 & -2 & 0 & 0 & 0 & 1 & 1 & 2 \\ -2 & -1 & -1 & 0 & 0 & 0 & 2 & 1 & 1 \\ -1 & -2 & -1 & 0 & 0 & 0 & 1 & 2 & 1 \end{array}\right] \left[\begin{array}{c} 1 \\ 2 \\ 3 \\ \hline 4 \\ 5 \\ 6 \\ \hline 7 \\ 8 \\ 9 \end{array}\right]$$

$$= \left[\begin{array}{c} -12 \\ -12 \\ -12 \\ \hline -12 \\ -12 \\ -12 \\ \hline 24 \\ 24 \\ 24 \end{array}\right]$$

$$\boxed{g[m, n] = \begin{bmatrix} -12 & -12 & -12 \\ -12 & -12 & -12 \\ 24 & 24 & 24 \end{bmatrix}_{3\times 3}}$$

Q4

March 5, 2021

Q.4. (i) Write python from scratch for 2D DCT transform on the following matrix: (a) f(m.n) = [90, 100 ; 100, 175]
(b) f(m.n) = [10, 20, 30; 40 50 60; 70 80, 90; 100,110,120; 130 140, 150] and also comment on energy compaction property of DCT. (c) f(m.n)= monalisa.tif

(ii) Write python from scratch for 2D IDCT transform and reconstruct f(m,n) from part(i)

[1]:
```
import cv2
import matplotlib.pyplot as plt
```

[2]:
```
import numpy as np
```

Helper for DCT/IDCT

[3]:
```
def A(i,Z):
    if i:
        return np.sqrt(2/Z)
    else:
        return np.sqrt(1/Z)
```

DCT

[4]:
```
def DCT(x):
    M=x.shape[0]
    N=x.shape[1]
    X=np.zeros((M,N))
    for k in range(M):
        for l in range(N):
            num=0
            for m in range(M):
                for n in range(N):
                    num+=x[m][n]*A(k,M)*A(l,N)*np.cos((2*m+1)*k*np.pi/(2*M))*np.
↪cos((2*n+1)*l*np.pi/(2*N))
            X[k][l]=num
    return X
```

IDCT

```python
[5]: def IDCT(X):
    M=X.shape[0]
    N=X.shape[1]
    x=np.zeros((M,N))
    for m in range(M):
        for n in range(N):
            num=0
            for k in range(M):
                for l in range(N):
                    num+=X[k][l]*A(k,M)*A(l,N)*np.cos((2*m+1)*k*np.pi/(2*M))*np.
 ↪cos((2*n+1)*l*np.pi/(2*N))
            x[m][n]=num
    return x
```

Matrix 1

```python
[6]: x=np.array([[90,100],[100,175]])
x
```

```
[6]: array([[ 90, 100],
            [100, 175]])
```

```python
[7]: X=DCT(x)
X
```

```
[7]: array([[232.5, -42.5],
            [-42.5,  32.5]])
```

```python
[8]: IDCT(X)
```

```
[8]: array([[ 90., 100.],
            [100., 175.]])
```

The original matrix is obtained from the IDCT.

Matrix 2

```python
[9]: x=np.array([[10, 20, 30],
             [40, 50, 60],
             [70, 80, 90],
             [100,110,120],
             [130,140,150]])
x
```

```
[9]: array([[ 10,  20,  30],
            [ 40,  50,  60],
            [ 70,  80,  90],
            [100, 110, 120],
```

```
             [130, 140, 150]])
```

[10]:
```
X=DCT(x)
np.around(X,2) #Values rounded for clarity
```

[10]:
```
array([[ 309.84,  -31.62,    0.  ],
       [-163.65,    0.  ,   -0.  ],
       [  -0.  ,    0.  ,    0.  ],
       [ -14.76,    0.  ,   -0.  ],
       [  -0.  ,    0.  ,   -0.  ]])
```

[11]:
```
IDCT(X)
```

[11]:
```
array([[ 10.,  20.,  30.],
       [ 40.,  50.,  60.],
       [ 70.,  80.,  90.],
       [100., 110., 120.],
       [130., 140., 150.]])
```

The original matrix is obtained from the IDCT.

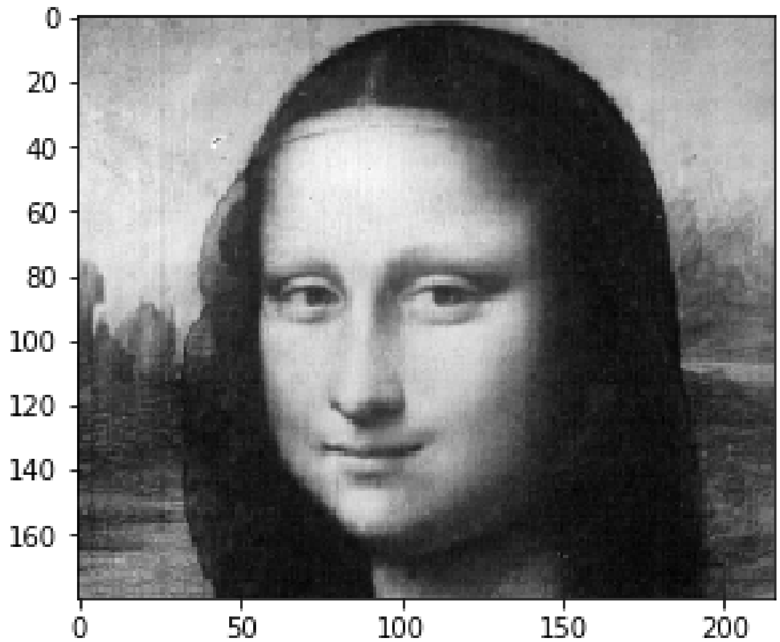
Energy Compaction

The DCT exhibits the property of energy compaction for correlated signals. The more correlated the input, the more concentrated is the energy of the output. This can be seen in the above two examples, as the DCT of the matrices tend to have larger values in the low-indexed coordinates (to the top-left). This is also true for the image given below.

Image: monalisa.tif

[12]:
```
im=cv2.imread('monalisa.tif')
plt.imshow(im)
```

[12]:
```
<matplotlib.image.AxesImage at 0x26512c78080>
```



[13]:
```
im.shape
```

[13]: (180, 216, 3)

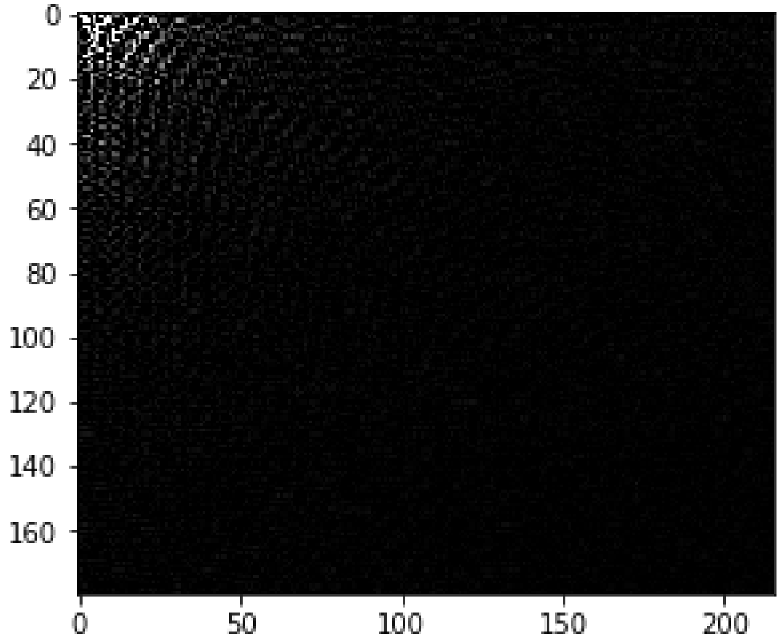
[14]:
```
img_flat = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY) #To flatten image to 2D Matrix
img_flat.shape
```

[14]: (180, 216)

[15]:
```
img_DCT=DCT(img_flat)
```

[16]:
```
plt.imshow(img_DCT, cmap='gray', vmin=0, vmax=255)
```

[16]: <matplotlib.image.AxesImage at 0x26512d18780>

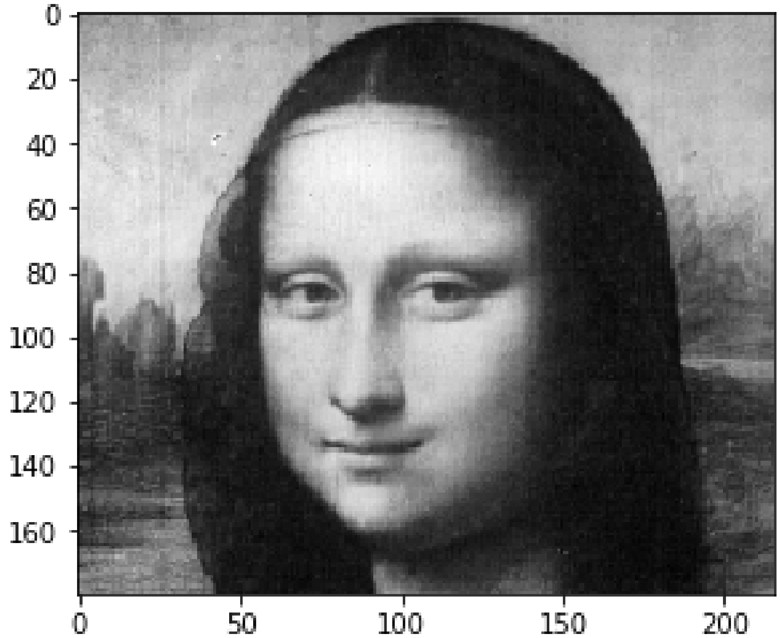


As previously noted under Energy Compaction, the higher values are concentrated at the top left of the image.

[17]:
```
img_fin=IDCT(img_DCT)
```

[18]:
```
plt.imshow(img_fin, cmap='gray')
```

[18]: <matplotlib.image.AxesImage at 0x26512d83278>



The original image is obtained from the IDCT.