

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

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

① Linearity : If $f_1(x, y) \leftrightarrow F_1(u, v)$
 $f_2(x, y) \leftrightarrow F_2(u, v)$

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

Proof: $\text{DFT}\{a f_1(x, y) + b f_2(x, y)\} = \sum_x \sum_y (a f_1(x, y) + b f_2(x, y)) e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$
 $= a \sum_x \sum_y f_1(x, y) e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)} + b \sum_x \sum_y f_2(x, y) e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$
 $= a F_1(u, v) + b F_2(u, v)$
 $=$

② Separability : Kernel is separable.

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

Proof.

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

$$= \sum_x f_1(x) e^{-j2\pi\frac{ux}{M}} \sum_y f_2(y) e^{-j2\pi\frac{vy}{N}} = F_1(u) F_2(v)$$

$$=$$

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

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

(4)

Translation:

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

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

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

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

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

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

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

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

(5)

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

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

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

(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) : $f_1(x, y) \otimes f_2(x, y) \leftrightarrow F_1(u, v) F_2(u, v)$

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

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

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

$$= \sum_{x'} \sum_{y'} f_1(x', y') \underbrace{\sum_{x''} \sum_{y''} f_2(x'', y'') e^{-j2\pi(\frac{ux''}{N} + \frac{vy''}{N})}}_{F_2(u, v)} e^{-j2\pi(\frac{ux'}{N} + \frac{vy'}{N})} = F_1(u, v) F_2(u, v)$$

⑦

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

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

Multiply by $\frac{1}{N^2}$

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

=:

Q2

March 2, 2021

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

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

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

```
[27]: 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)
$$x = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$$

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

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

$$= 1 + (-0) + (2) + (-1) = 2$$

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

$$= 1 + 0 - 2 - 1 = -2$$

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

$$= 1 - 0 - 2 + 1 = 0$$

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

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

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

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

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

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

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

$$= 28 + 32(-j) + 36(-1) + 40j = -8 + 8j$$

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

$$+ f(1,0) - f(1,1) + f(1,2) - f(1,3)$$

$$+ f(2,0) - f(2,1) + f(2,2) - f(2,3)$$

$$+ f(3,0) - f(3,1) + f(3,2) - f(3,3)$$

$$= 28 - 32 + 36 - 40 = -8$$

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

$$= 28 + 32(j) + 36(-1) + 40(-j) = -8 - 8j$$

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

$$= 10 - 26j - 42 + 58j = -32 + 32j$$

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

$$= 68j - 68j = 0$$

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

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

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

$$= 10 - 26 + 42 - 58 = -32$$

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

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

$$= 68 - 68 = 0$$

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

$$= 0$$

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

$$= 16 - 26j - 42 + 58j = -32 - 32j$$

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

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

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

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

==.

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_convolve2D(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")
```



```

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_convolve2D(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]]

```

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

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

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

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

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

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

$$M = \text{Max}(M_1, M_2) = 3$$

$$N = \text{Max}(N_1, N_2) = 3$$

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

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

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

$$H_0 = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} -1 & -1 & -2 \\ -2 & -1 & -1 \\ -1 & -2 & -1 \end{bmatrix}$$

Doubly-Block Circulant matrix

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

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

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

$$g = Hf = \begin{bmatrix} 1 & 1 & 2 & -1 & -1 & -2 & 0 & 0 & 0 \\ 2 & 1 & 1 & -2 & -1 & -1 & 0 & 0 & 0 \\ 1 & 2 & 1 & -1 & -2 & -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 1 & 2 & -1 & -1 & -2 \\ 0 & 0 & 0 & 2 & 1 & 1 & -2 & -1 & -1 \\ 0 & 0 & 0 & 1 & 2 & 1 & -1 & -2 & -1 \\ \hline -1 & -1 & -2 & 0 & 0 & 0 & 1 & 1 & 2 \\ -2 & -1 & -1 & 0 & 0 & 0 & 2 & 1 & 1 \\ -1 & -2 & -1 & 0 & 0 & 0 & 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix}$$

$$= \begin{bmatrix} -12 \\ -12 \\ -12 \\ \hline -12 \\ -12 \\ -12 \\ \hline 24 \\ 24 \\ 24 \end{bmatrix}$$

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

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) $f(m,n) = \text{monalisa.tif}$

(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

```
[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

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

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

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

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

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

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

The original matrix is obtained from the IDCT.

Matrix 2

```
[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],
            [160, 170, 180],
            [190, 200, 210],
            [220, 230, 240],
            [250, 260, 270],
            [280, 290, 300],
            [310, 320, 330],
            [340, 350, 360],
            [370, 380, 390],
            [400, 410, 420],
            [430, 440, 450],
            [460, 470, 480],
            [490, 500, 510],
            [520, 530, 540],
            [550, 560, 570],
            [580, 590, 600],
            [610, 620, 630],
            [640, 650, 660],
            [670, 680, 690],
            [700, 710, 720],
            [730, 740, 750],
            [760, 770, 780],
            [790, 800, 810],
            [820, 830, 840],
            [850, 860, 870],
            [880, 890, 900],
            [910, 920, 930],
            [940, 950, 960],
            [970, 980, 990],
            [1000, 1010, 1020],
            [1030, 1040, 1050],
            [1060, 1070, 1080],
            [1090, 1100, 1110],
            [1120, 1130, 1140],
            [1150, 1160, 1170],
            [1180, 1190, 1200],
            [1210, 1220, 1230],
            [1240, 1250, 1260],
            [1270, 1280, 1290],
            [1300, 1310, 1320],
            [1330, 1340, 1350],
            [1360, 1370, 1380],
            [1390, 1400, 1410],
            [1420, 1430, 1440],
            [1450, 1460, 1470],
            [1480, 1490, 1500],
            [1510, 1520, 1530],
            [1540, 1550, 1560],
            [1570, 1580, 1590],
            [1600, 1610, 1620],
            [1630, 1640, 1650],
            [1660, 1670, 1680],
            [1690, 1700, 1710],
            [1720, 1730, 1740],
            [1750, 1760, 1770],
            [1780, 1790, 1800],
            [1810, 1820, 1830],
            [1840, 1850, 1860],
            [1870, 1880, 1890],
            [1900, 1910, 1920],
            [1930, 1940, 1950],
            [1960, 1970, 1980],
            [1990, 2000, 2010],
            [2020, 2030, 2040],
            [2050, 2060, 2070],
            [2080, 2090, 2100],
            [2110, 2120, 2130],
            [2140, 2150, 2160],
            [2170, 2180, 2190],
            [2200, 2210, 2220],
            [2230, 2240, 2250],
            [2260, 2270, 2280],
            [2290, 2300, 2310],
            [2320, 2330, 2340],
            [2350, 2360, 2370],
            [2380, 2390, 2400],
            [2410, 2420, 2430],
            [2440, 2450, 2460],
            [2470, 2480, 2490],
            [2500, 2510, 2520],
            [2530, 2540, 2550],
            [2560, 2570, 2580],
            [2590, 2600, 2610],
            [2620, 2630, 2640],
            [2650, 2660, 2670],
            [2680, 2690, 2700],
            [2710, 2720, 2730],
            [2740, 2750, 2760],
            [2770, 2780, 2790],
            [2800, 2810, 2820],
            [2830, 2840, 2850],
            [2860, 2870, 2880],
            [2890, 2900, 2910],
            [2920, 2930, 2940],
            [2950, 2960, 2970],
            [2980, 2990, 3000],
            [3010, 3020, 3030],
            [3040, 3050, 3060],
            [3070, 3080, 3090],
            [3100, 3110, 3120],
            [3130, 3140, 3150],
            [3160, 3170, 3180],
            [3190, 3200, 3210],
            [3220, 3230, 3240],
            [3250, 3260, 3270],
            [3280, 3290, 3300],
            [3310, 3320, 3330],
            [3340, 3350, 3360],
            [3370, 3380, 3390],
            [3400, 3410, 3420],
            [3430, 3440, 3450],
            [3460, 3470, 3480],
            [3490, 3500, 3510],
            [3520, 3530, 3540],
            [3550, 3560, 3570],
            [3580, 3590, 3600],
            [3610, 3620, 3630],
            [3640, 3650, 3660],
            [3670, 3680, 3690],
            [3700, 3710, 3720],
            [3730, 3740, 3750],
            [3760, 3770, 3780],
            [3790, 3800, 3810],
            [3820, 3830, 3840],
            [3850, 3860, 3870],
            [3880, 3890, 3900],
            [3910, 3920, 3930],
            [3940, 3950, 3960],
            [3970, 3980, 3990],
            [4000, 4010, 4020],
            [4030, 4040, 4050],
            [4060, 4070, 4080],
            [4090, 4100, 4110],
            [4120, 4130, 4140],
            [4150, 4160, 4170],
            [4180, 4190, 4200],
            [4210, 4220, 4230],
            [4240, 4250, 4260],
            [4270, 4280, 4290],
            [4300, 4310, 4320],
            [4330, 4340, 4350],
            [4360, 4370, 4380],
            [4390, 4400, 4410],
            [4420, 4430, 4440],
            [4450, 4460, 4470],
            [4480, 4490, 4500],
            [4510, 4520, 4530],
            [4540, 4550, 4560],
            [4570, 4580, 4590],
            [4600, 4610, 4620],
            [4630, 4640, 4650],
            [4660, 4670, 4680],
            [4690, 4700, 4710],
            [4720, 4730, 4740],
            [4750, 4760, 4770],
            [4780, 4790, 4800],
            [4810, 4820, 4830],
            [4840, 4850, 4860],
            [4870, 4880, 4890],
            [4900, 4910, 4920],
            [4930, 4940, 4950],
            [4960, 4970, 4980],
            [4990, 5000, 5010],
            [5020, 5030, 5040],
            [5050, 5060, 5070],
            [5080, 5090, 5100],
            [5110, 5120, 5130],
            [5140, 5150, 5160],
            [5170, 5180, 5190],
            [5200, 5210, 5220],
            [5230, 5240, 5250],
            [5260, 5270, 5280],
            [5290, 5300, 5310],
            [5320, 5330, 5340],
            [5350, 5360, 5370],
            [5380, 5390, 5400],
            [5410, 5420, 5430],
            [5440, 5450, 5460],
            [5470, 5480, 5490],
            [5500, 5510, 5520],
            [5530, 5540, 5550],
            [5560, 5570, 5580],
            [5590, 5600, 5610],
            [5620, 5630, 5640],
            [5650, 5660, 5670],
            [5680, 5690, 5700],
            [5710, 5720, 5730],
            [5740, 5750, 5760],
            [5770, 5780, 5790],
            [5800, 5810, 5820],
            [5830, 5840, 5850],
            [5860, 5870, 5880],
            [5890, 5900, 5910],
            [5920, 5930, 5940],
            [5950, 5960, 5970],
            [5980, 5990, 6000],
            [6010, 6020, 6030],
            [6040, 6050, 6060],
            [6070, 6080, 6090],
            [6100, 6110, 6120],
            [6130, 6140, 6150],
            [6160, 6170, 6180],
            [6190, 6200, 6210],
            [6220, 6230, 6240],
            [6250, 6260, 6270],
            [6280, 6290, 6300],
            [6310, 6320, 6330],
            [6340, 6350, 6360],
            [6370, 6380, 6390],
            [6400, 6410, 6420],
            [6430, 6440, 6450],
            [6460, 6470, 6480],
            [6490, 6500, 6510],
            [6520, 6530, 6540],
            [6550, 6560, 6570],
            [6580, 6590, 6600],
            [6610, 6620, 6630],
            [6640, 6650, 6660],
            [6670, 6680, 6690],
            [6700, 6710, 6720],
            [6730, 6740, 6750],
            [6760, 6770, 6780],
            [6790, 6800, 6810],
            [6820, 6830, 6840],
            [6850, 6860, 6870],
            [6880, 6890, 6900],
            [6910, 6920, 6930],
            [6940, 6950, 6960],
            [6970, 6980, 6990],
            [7000, 7010, 7020],
            [7030, 7040, 7050],
            [7060, 7070, 7080],
            [7090, 7100, 7110],
            [7120, 7130, 7140],
            [7150, 7160, 7170],
            [7180, 7190, 7200],
            [7210, 7220, 7230],
            [7240, 7250, 7260],
            [7270, 7280, 7290],
            [7300, 7310, 7320],
            [7330, 7340, 7350],
            [7360, 7370, 7380],
            [7390, 7400, 7410],
            [7420, 7430, 7440],
            [7450, 7460, 7470],
            [7480, 7490, 7500],
            [7510, 7520, 7530],
            [7540, 7550, 7560],
            [7570, 7580, 7590],
            [7600, 7610, 7620],
            [7630, 7640, 7650],
            [7660, 7670, 7680],
            [7690, 7700, 7710],
            [7720, 7730, 7740],
            [7750, 7760, 7770],
            [7780, 7790, 7800],
            [7810, 7820, 7830],
            [7840, 7850, 7860],
            [7870, 7880, 7890],
            [7900, 7910, 7920],
            [7930, 7940, 7950],
            [7960, 7970, 7980],
            [7990, 8000, 8010],
            [8020, 8030, 8040],
            [8050, 8060, 8070],
            [8080, 8090, 8100],
            [8110, 8120, 8130],
            [8140, 8150, 8160],
            [8170, 8180, 8190],
            [8200, 8210, 8220],
            [8230, 8240, 8250],
            [8260, 8270, 8280],
            [8290, 8300, 8310],
            [8320, 8330, 8340],
            [8350, 8360, 8370],
            [8380, 8390, 8400],
            [8410, 8420, 8430],
            [8440, 8450, 8460],
            [8470, 8480, 8490],
            [8500, 8510, 8520],
            [8530, 8540, 8550],
            [8560, 8570, 8580],
            [8590, 8600, 8610],
            [8620, 8630, 8640],
            [8650, 8660, 8670],
            [8680, 8690, 8700],
            [8710, 8720, 8730],
            [8740, 8750, 8760],
            [8770, 8780, 8790],
            [8800, 8810, 8820],
            [8830, 8840, 8850],
            [8860, 8870, 8880],
            [8890, 8900, 8910],
            [8920, 8930, 8940],
            [8950, 8960, 8970],
            [8980, 8990, 9000],
            [9010, 9020, 9030],
            [9040, 9050, 9060],
            [9070, 9080, 9090],
            [9100, 9110, 9120],
            [9130, 9140, 9150],
            [9160, 9170, 9180],
            [9190, 9200, 9210],
            [9220, 9230, 9240],
            [9250, 9260, 9270],
            [9280, 9290, 9300],
            [9310, 9320, 9330],
            [9340, 9350, 9360],
            [9370, 9380, 9390],
            [9400, 9410, 9420],
            [9430, 9440, 9450],
            [9460, 9470, 9480],
            [9490, 9500, 9510],
            [9520, 9530, 9540],
            [9550, 9560, 9570],
            [9580, 9590, 9600],
            [9610, 9620, 9630],
            [9640, 9650, 9660],
            [9670, 9680, 9690],
            [9700, 9710, 9720],
            [9730, 9740, 9750],
            [9760, 9770, 9780],
            [9790, 9800, 9810],
            [9820, 9830, 9840],
            [9850, 9860, 9870],
            [9880, 9890, 9900],
            [9910, 9920, 9930],
            [9940, 9950, 9960],
            [9970, 9980, 9990],
            [10000, 10010, 10020],
            [10030, 10040, 10050],
            [10060, 10070, 10080],
            [10090, 10100, 10110],
            [10120, 10130, 10140],
            [10150, 10160, 10170],
            [10180, 10190, 10200],
            [10210, 10220, 10230],
            [10240, 10250, 10260],
            [10270, 10280, 10290],
            [10300, 10310, 10320],
            [10330, 10340, 10350],
            [10360, 10370, 10380],
            [10390, 10400, 10410],
            [10420, 10430, 10440],
            [10450, 10460, 10470],
            [10480, 10490, 10500],
            [10510, 10520, 10530],
            [10540, 10550, 10560],
            [10570, 10580, 10590],
            [10600, 10610, 10620],
            [10630, 10640, 10650],
            [10660, 10670, 10680],
            [10690, 10700, 10710],
            [10720, 10730, 10740],
            [10750, 10760, 10770],
            [10780, 10790, 10800],
            [10810, 10820, 10830],
            [10840, 10850, 10860],
            [10870, 10880, 10890],
            [10900, 10910, 10920],
            [10930, 10940, 10950],
            [10960, 10970, 10980],
            [10990, 11000, 11010],
            [11020, 11030, 11040],
            [11050, 11060, 11070],
            [11080, 11090, 11100],
            [11110, 11120, 11130],
            [11140, 11150, 11160],
            [11170, 11180, 11190],
            [11200, 11210, 11220],
            [11230, 11240, 11250],
            [11260, 11270, 11280],
            [11290, 11300, 11310],
            [11320, 11330, 11340],
            [11350, 11360, 11370],
            [11380, 11390, 11400],
            [11410, 11420, 11430],
            [11440, 11450, 11460],
            [11470, 11480, 11490],
            [11500, 11510, 11520],
            [11530, 11540, 11550],
            [11560, 11570, 11580],
            [11590, 11600, 11610],
            [11620, 11630, 11640],
            [11650, 11660, 11670],
            [11680, 11690, 11700],
            [11710, 11720, 11730],
            [11740, 11750, 11760],
            [11770, 11780, 11790],
            [11800, 11810, 11820],
            [11830, 11840, 11850],
            [11860, 11870, 11880],
            [11890, 11900, 11910],
            [11920, 11930, 11940],
            [11950, 11960, 11970],
            [11980, 11990, 12000],
            [12010, 12020, 12030],
            [12040, 12050, 12060],
            [12070, 12080, 12090],
            [12100, 12110, 12120],
            [12130, 12140, 12150],
            [12160, 12170, 12180],
            [12190, 12200, 12210],
            [12220, 12230, 12240],
            [12250, 12260, 12270],
            [12280, 12290, 12300],
            [12310, 12320, 12330],
            [12340, 12350, 12360],
            [12370, 12380, 12390],
            [12400, 12410, 12420],
            [12430, 12440, 12450],
            [12460, 12470, 12480],
            [12490, 12500, 12510],
            [12520, 12530, 12540],
            [12550, 12560, 12570],
            [12580, 12590, 12600],
            [12610, 12620, 12630],
            [12640, 12650, 12660],
            [12670, 12680, 12690],
            [12700, 12710, 12720],
            [12730, 12740, 12750],
            [12760, 12770, 12780],
            [12790, 12800, 12810],
            [12820, 12830, 12840],
            [12850, 12860, 12870],
            [12880, 12890, 12900],
            [12910, 12920, 12930],
            [12940, 12950, 12960],
            [12970, 12980, 12990],
            [13000, 13010, 13020],
            [13030, 13040, 13050],
            [13060, 13070, 13080],
            [13090, 13100, 13110],
            [13120, 13130, 13140],
            [13150, 13160, 13170],
            [13180, 13190, 13200],
            [13210, 13220, 13230],
            [13240, 13250, 13260],
            [13270, 13280, 13290],
            [13300, 13310, 13320],
            [13330, 13340, 13350],
            [13360, 13370, 13380],
            [13390, 13400, 13410],
            [13420, 13430, 13440],
            [13450, 13460, 13470],
            [13480, 13490, 13500],
            [13510, 13520, 13530],
            [13540, 13550, 13560],
            [13570, 13580, 13590],
            [13600, 13610, 13620],
            [13630, 13640, 13650],
            [13660, 13670, 13680],
            [13690, 13700, 13710],
            [13720, 13730, 13740],
            [13750, 13760, 13770],
            [13780, 13790, 13800],
            [13810, 13820, 13830],
            [13840, 13850, 13860],
            [13870, 13880, 13890],
            [13900, 13910, 13920],
            [13930, 13940, 13950],
            [13960, 13970, 13980],
            [13990, 14000, 14010],
            [14020, 14030, 14040],
            [14050, 14060, 14070],
            [14080, 14090, 14100],
            [14110, 14120, 14130],
            [14140, 14150, 14160],
            [14170, 14180, 14190],
            [14200, 14210, 14220],
            [14230, 14240, 14250],
            [14260, 14270, 14280],
            [14290, 14300, 14310],
            [14320, 14330, 14340],
            [14350, 14360, 14370],
            [14380, 14390, 14400],
            [14410, 14420, 14430],
            [14440, 14450, 14460],
            [14470, 14480, 14490],
            [14500, 14510, 14520],
            [14530, 14540, 14550],
            [14560, 14570, 14580],
            [14590, 14600, 14610],
            [14620, 14630, 14640],
            [14650, 14660, 14670],
            [14680, 14690, 14700],
            [14710, 14720, 14730],
            [14740, 14750, 14760],
            [14770, 14780, 14790],
            [14800, 14810, 14820],
            [14830, 14840, 14850],
            [14860, 14870, 14880],
            [14890, 14900, 14910],
            [14920, 14930, 14940],
            [14950, 14960, 14970],
            [14980, 14990, 15000],
            [15010, 15020, 15030],
            [15040, 15050, 15060],
            [15070, 15080, 15090],
            [15100, 15110, 15120],
            [15130, 15140, 15150],
            [15160, 15170, 15180],
            [15190, 15200, 15210],
            [15220, 15230, 15240],
            [15250, 15260, 15270],
            [15280, 15290, 15300],
            [15310, 15320, 15330],
            [15340, 15350, 15360],
            [15370, 15380, 15390],
            [15400, 15410, 15420],
            [15430, 15440, 15450],
            [15460, 15470, 15480],
            [15490, 15500, 15510],
            [15520, 15530, 15540],
            [15550, 15560, 15570],
            [15580, 15590, 15600],
            [15610, 15620, 15630],
            [15640, 15650, 15660],
            [15670, 15680, 15690],
            [15700, 15710, 15720],
            [15730, 15740, 15750],
            [15760, 15770, 15780],
            [15790, 15800, 15810],
            [15820, 15830, 15840],
            [15850, 15860, 15870],
            [15880, 15890, 15900],
            [15910, 15920, 15930],
            [15940, 15950, 15960],
            [15970, 15980, 15990],
            [16000, 16010, 16020],
            [16030, 16040, 16050],
            [16060, 16070, 16080],
            [16090, 16100, 16110],
            [16120, 16130, 16140],
            [16150, 16160, 16170],
            [16180, 16190, 16200],
            [16210, 16220, 16230],
            [16240, 16250, 16260],
            [16270, 16280, 16290],
            [16300, 16310, 16320],
            [16330, 16340, 16350],
            [16360, 16370, 16380],
            [16390, 16400, 16410],
            [16420, 16430, 16440],
            [16450, 16460, 16470],
            [16480, 16490, 16500],
            [16510, 16520, 16530],
            [16540, 16550, 16560],
            [16
```

```
[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.

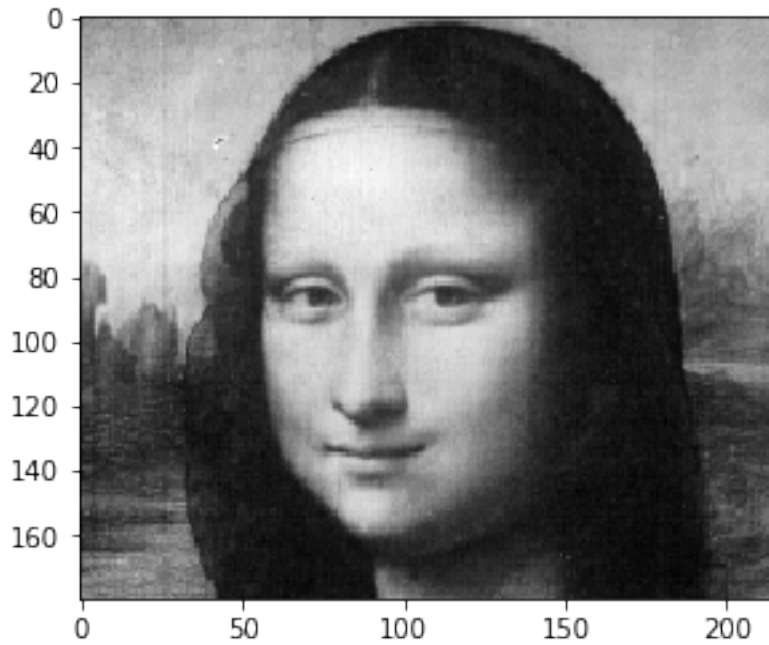
Energy Compaction

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)
```

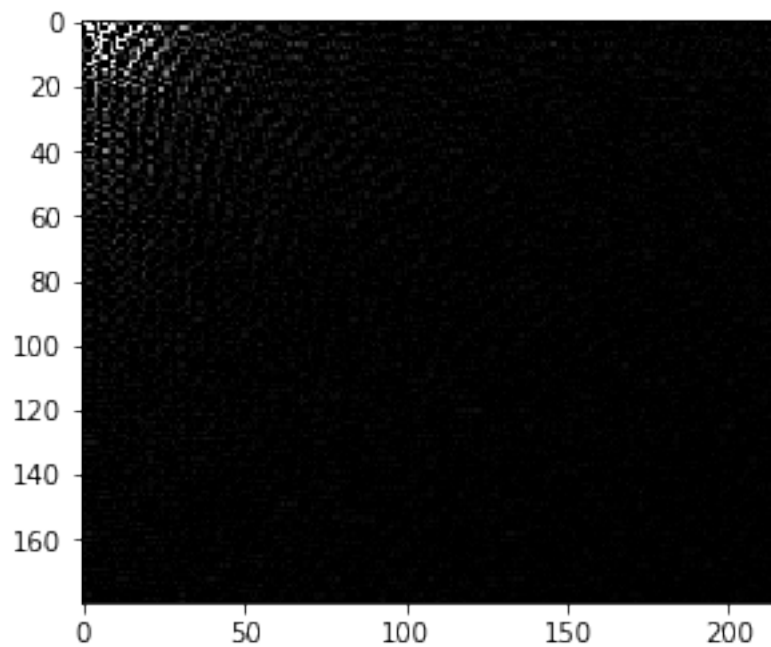
```
[14]: img_flat = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY) #To flatten image to 2D Matrix  
img_flat.shape
```

```
[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>
```

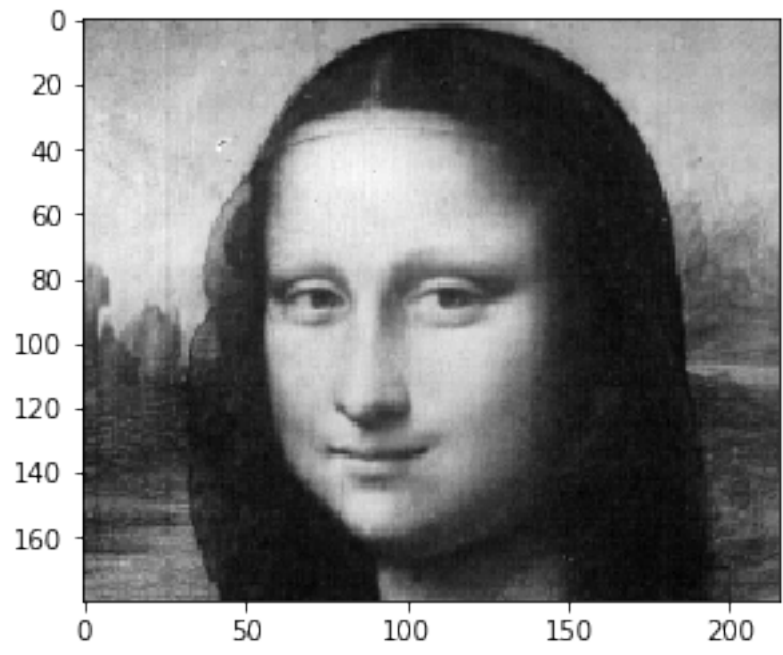


As previously noted under Energy Compaction, the higher values are concentrated at the top left of the image.

```
[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.