LAB 9

As shown below code, I had read the mat file using the h5py library, then converted it to proper NumPy array format of 10304*400. Using np.mean, row-wise mean was calculated and then subtracted from the array. The covariance matrix was calculated using np.cov, after that eigenvectors and eigenvalues were been calculated. As cov matrices are symmetric which deduces that eigenvectors are mutually orthogonal to each other. Using argsort() will sort the values, and then sort the vectors corresponding to its values. Multiplying the obtained eigenvector matrix with the original face_array will give the low dimensional representation of face_array. Now for reconstructing 1st image, considering 50 features gives 1652 norm of error, by considering 100 features gives 1360 norm of error and by taking 400(max rank possible) features into consideration gives 0 norm of error, this indicates we can regenerate the original face_array fully.

In [1]:

```python
import numpy as np
from scipy import linalg as LA
import pandas as pd
import matplotlib.pyplot as plt
import scipy.io
import h5py
#question 1
face_images = {}
with h5py.File('faceimages.mat', 'r') as f: #reading mat file using h5py library
    data = f.keys()
    for i, j in f.items():
        face_images[i] = np.array(j)
face_array = face_images['data'][:-1]# facearray will be 10304*400 matrix

#question 2
mean = np.mean(face_array, axis = 1) # finding mean row wise
mean=np.transpose([mean] * 400) # expanding the mean by repeating 400 times to make its dimension identical to face_array
face_array = face_array - mean #mean subtract
face_cov = np.cov(face_array) # calculating covariance 10304 * 10304
#question 3
print(face_cov.shape)
eigenValues, eigenVectors = LA.eig(face_cov) #calculating eigenvectors and eigenvalues
```

```
(10304, 10304)
```

In [12]:

```python
#question 4
srtdidx = eigenValues.argsort()[::-1]#sorting indexes using argsort
eigenValues = eigenValues[srtdidx]#sorting eigenvalues using sorted indexes
eigenVectors = eigenVectors[:,srtdidx]#sorting eigenvectors corresponding to its eigen value

# question 5
Y = np.matmul(eigenVectors.T, face_array) # calculating low dimensional representation of face_array
```

In [13]:
```
#question 6
rcvry1 = np.matmul(eigenVectors[:, :50], Y[:50, 0])# calculating recovery array for 1st image
rcvry2 = np.matmul(eigenVectors[:, :100], Y[:100, 0])
rcvry3 = np.matmul(eigenVectors[:, :400], Y[:400, 0])
```

In [15]:
```
#question 7
ans1= np.sqrt(np.sum((rcvry1-face_array[:,0])**2))#calculating norm of error for every case
print("By considering 50 features norm = "+str(ans1))
ans2=np.sqrt(np.sum((rcvry2-face_array[:,0])**2))
print("By considering 100 features norm = "+str(ans2))
ans3=np.sqrt(np.sum((rcvry3-face_array[:,0])**2))
print("By considering 400 features norm = "+str(ans3))
```

```
By considering 50 features norm = (1652.3993204691326+0j)
By considering 100 features norm = (1360.3627731377176+0j)
By considering 400 features norm = (1.954036861523501e-10+0j)
```