Università
degli Studi
della Campania
Luigi Vanvitelli

Numerical Methods for Analysis

By Professor Rosanna Campagna
Mathematics and Physics Department

Spring 2023

The First Assignment

Baran Booyeh

The First Assignment

Baran Booyeh

April 16, 2023

Abstract

This Assignment contains 4 questions, each having different parts to solve. The whole aim is to analyze the data by using dimensionality reduction techniques such as PCA and also clustering analysis.

Exercise 1

1. Download from Blackboard the file ModelReductionData.mat. The file contains a data matrix X which belongs to the set of all matrices with 6×4000. Each column of X represents a data vector in the six dimensional space. Visualize the raw data using scatter plots, that is, select two components of the data at a time and plot them one against the other.
2. Center the data and compute the SVD. Plot the singular values. What can you say about the dimensionality of the data? Show the scatter plots of the first few principal components. Do the plots suggest the presence of clusters in the data?

In this part of the assignment, we started by loading the ModelReductionData and created 15 scatter plots by making 2 for loops, the first for i , which is the first component, against the second component of j, to visualize the raw data by plotting each combination of two dimensions against each other. All 15 scatter plots were shown together by the use of a subplot.

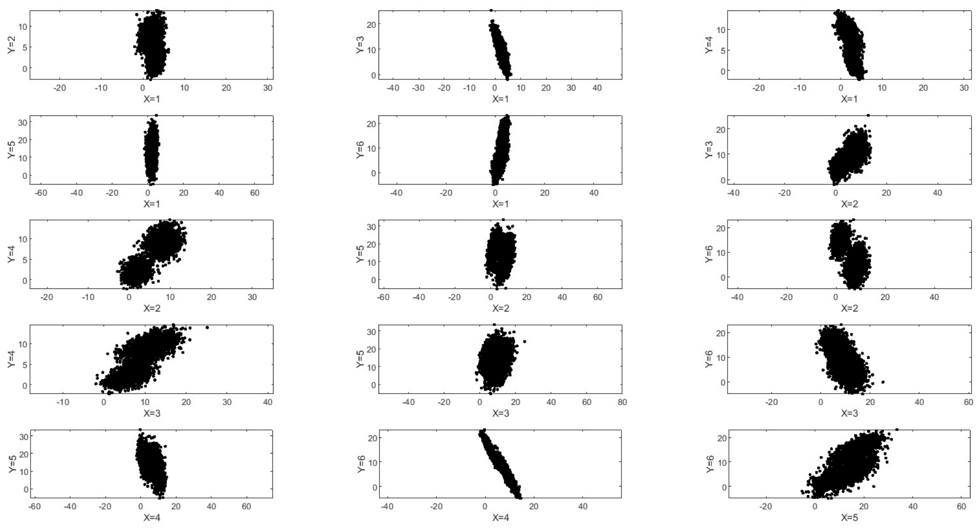
Figure 1: The scatterplots

Here, the visualization is not so obvious due to usage of the subplot. Even though, it sounds like there is clustering in the scatterplot of 7,9, and 10. It may be a better idea to represent each scatterplot individually to take a better visualization. By deleting the subplot, we get each figure for every 2 components. The interesting fact is that even though there are no obvious clusters in some figures of the subplot, like the third and eighth one, by only changing the type of visualization, some clusters may be seen. Meanwhile, there are some plots in which the correlation is stronger and more obvious, like the second and the 14th ones. In these plots, it is much easier to spot a line.

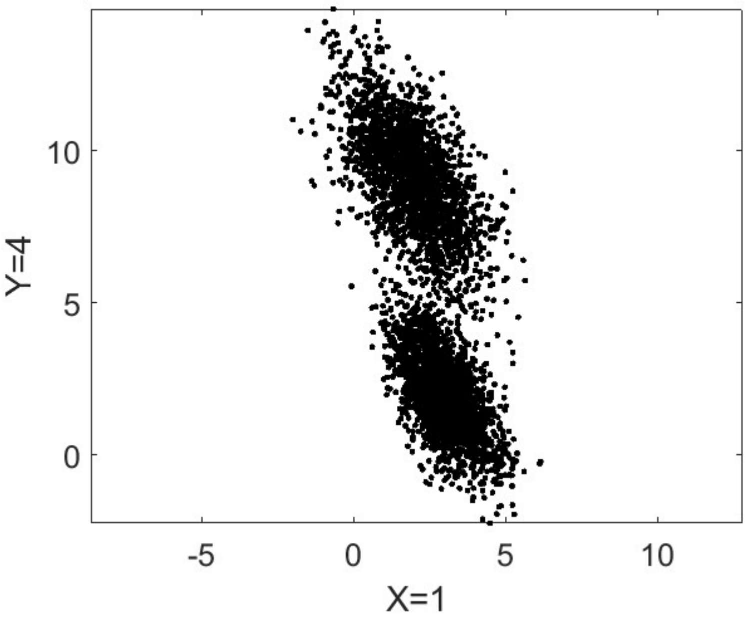

Figure 2: The same scatter plot as the third one in the subplot

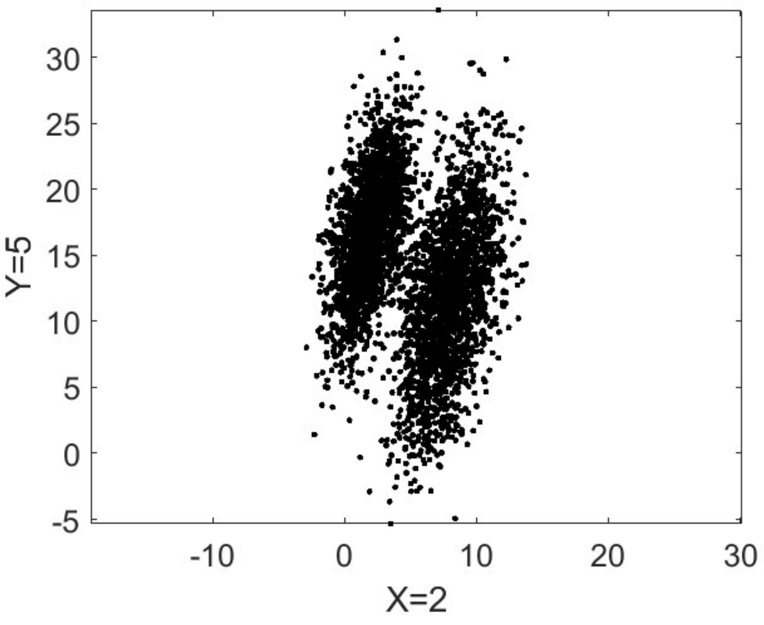

Figure 3: 8th one

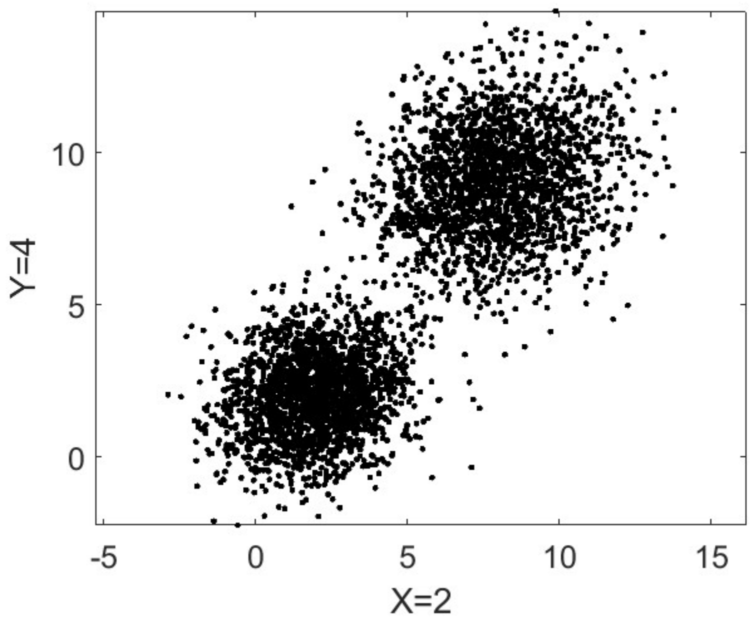

Figure 4: 7th one

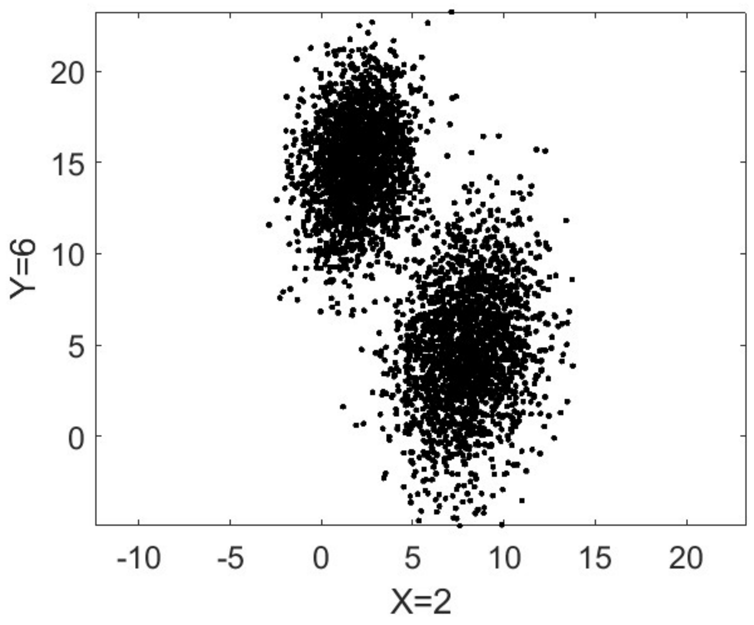

Figure 5: 9th one

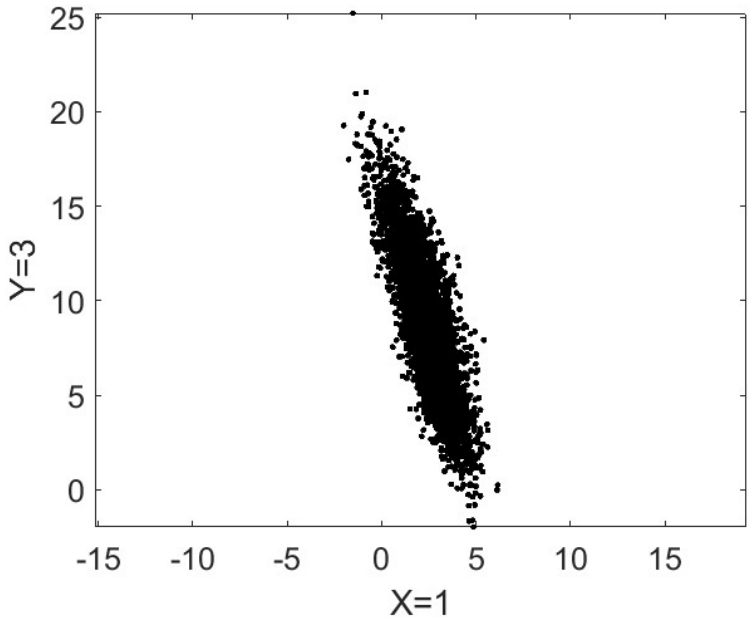

Figure 6: 2nd, sounds to have correlation

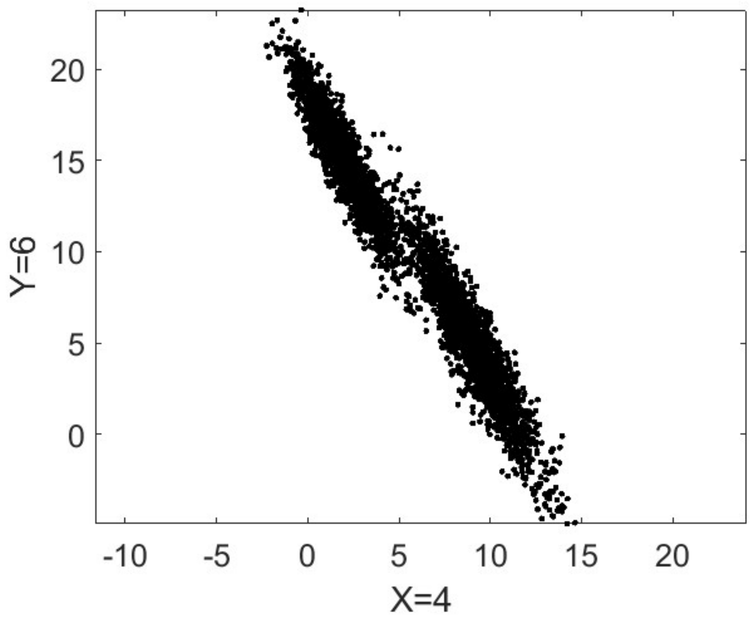

Figure 7: 14th one

```
load ModelReductionData

% Plot scatter plots for all combinations of two dimensions
figure(1)
k = 1;
for i = 1:5
    for j = i+1:6
        subplot(5, 3, k)
        plot(X(i,:), X(j,:), 'k.', 'MarkerSize', 7)
        axis('equal')
        set(gca, 'FontSize', 8)
        xlabel(['X=',num2str(i)])
        ylabel(['Y=',num2str(j)])
        k = k + 1;
    end
end
```

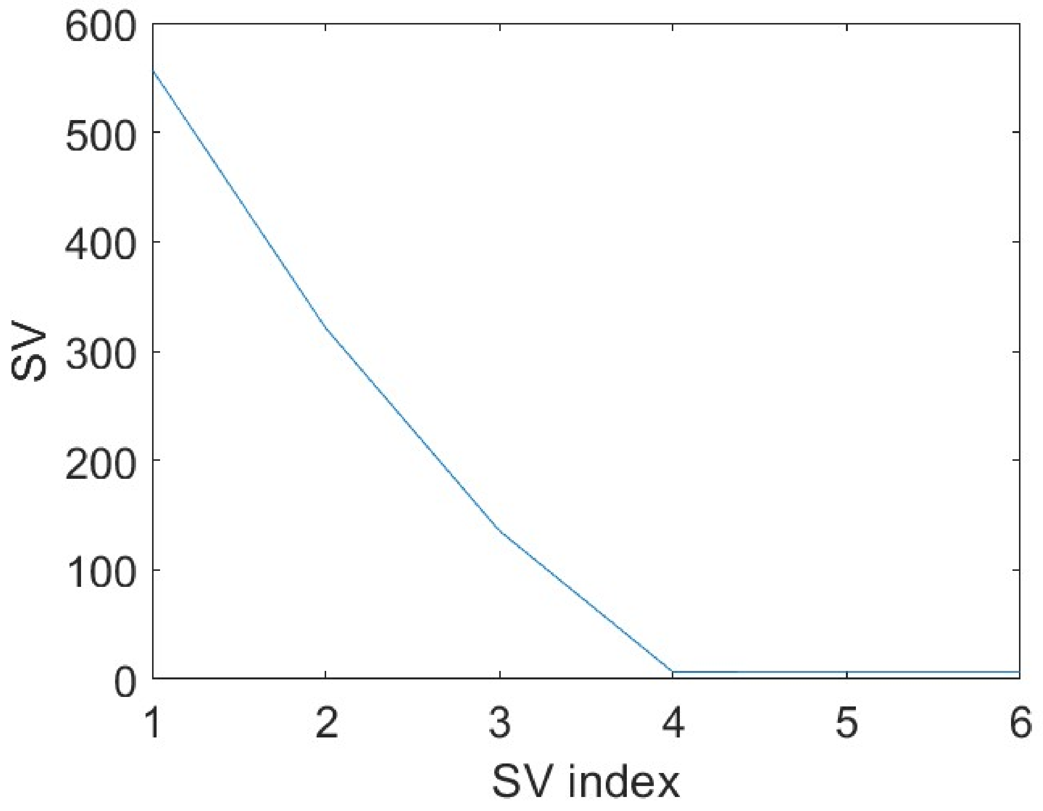

Figure 8: Singular Value Plot

For the second part of the assignment, two main operations had been done. Firstly, centering the data made by subtracting the mean of each dimension of the matrix X from the corresponding row of X. The importance of this operation is to make sure that the data is centered on zero and is not biased. Then, we compute the Singular Value Decomposition of the centered data matrix. SVD is a technique for obtaining three matrices U, D, and V. The U and V matrices contain orthonormal bases for the row and column spaces of the centered data matrix, while the D matrix contains the singular values of 'X-c'. The singular values in D represent the importance of each of the orthogonal components of the data, and can be used to reduce the dimensionality of the data while preserving the most important information. By plotting the Singular values, we realize how much each component contributes to the overall structure of the data. Lastly, to identify the clusters, and to reduce the dimensionality, the first 2 principal components of the centered data(the first two columns of the U matrix) have been plotted.

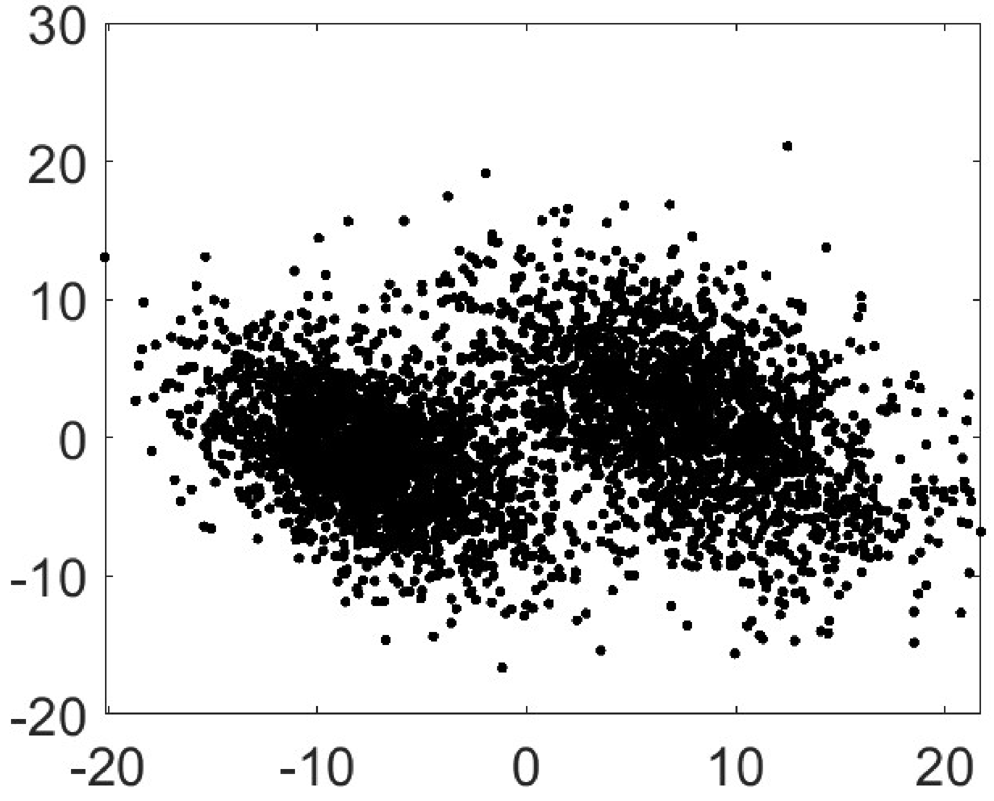

Figure 9: Scatterplot of the first 2 components

Overall, the plot shows that the singular values are arranged in decreasing order, with each singular value being greater or equal to its subsequent value. This confirms that the 'SVD' has been performed correctly and that the data has been decomposed into its most significant components, which are the first two principal components. Their scatterplot shows two clusters of points. This suggests that the data has some pattern that can be shown by the principal components.

```
load ModelReductionData.mat
% Center the data
x_c= 1/length(X)*sum(X,2) ;
X_c = X - x_c*ones(1,length(X));

% Compute the SVD
[U, D, V] = svd(X_c);
%Singualr values
sigma=diag(D);
% Plot singular values
figure()
plot(sigma)
xlabel('SV index')
ylabel('SV')
set(gca, 'FontSize', 16)

% Plot scatter plots of the first few principal components
Z=U(:,1:2)'*X_c;
figure()
plot(Z(1,:),Z(2,:),'k.','MarkerSize',10)
set(gca,'FontSize',20)
clear
```

Exercise 2

1. Download from Blackboard the data file HandwrittedDigits.mat, containing the data matrix X of size 256 × 1707 containing pixel images of the handwritten digits, and the label vector I of length 1707 containing numbers from 0 to 9, indicating the digits that the corresponding images represent.

2. Extract from X the images that correspond to numbers 0, 1, 3 and 7. From each subgroup, select 5 samples, and approximate these samples by the linear combination of the first k feature vectors, k = 5, 10, 15, 20,25. Plot the approximation as images, as well as the residual.

Firstly, we load the data file of HandWrittedDigits, which contains images of handwritten numbers, on Matlab. Then, Using a loop, it extracts the figures from the dataset that corresponds to the numbers 0, 1, 3, and 7. It centers the data for each of them by subtracting the mean of each picture from the image itself. The technique of Singular Value Decomposition (SVD) is then applied to the centred data in order to find the principal components of the pictures. This technique has been used to decompose a matrix into its constituent parts in order to understand its properties better. After that, by choosing the subset of the samples and cycles through various k values, the code computes the projection of the samples onto the first k principal components for each value of k which had been from 5 to 25 by 5 steps each time and then reconstructed the samples using these projection coefficients and the first k principal components. Lastly, for each of the numbers, the code presents the reconstructed samples for the first value of k. After that, it makes the residuals by subtracting the approximation from the original sample. For a better comparison and visualization, the original sample was also added to the figure on the first row, so in the end, we got the figures of handwriting samples from k=5 to k=25 from left to right, of the approximation and errors.

The goal of this exercise is to show how to utilize PCA to compress the data by projecting it into a lower-dimensional subspace defined by the principle components. The images of approximations demonstrate how much of the original image information can be kept when by k first components. It is obvious that as much as the k component increases, the projection becomes more accurate which makes the approximation more similar to the original sample. On the other hand, it also means that the error becomes less. So in the end, we would get a better visualization of the original data from the approximation with less error.
Therefore, we can conclude that the number of principal components used to reconstruct the images has a significant impact on the quality of the reconstruction. In general, more principal components typically result in better approximation.

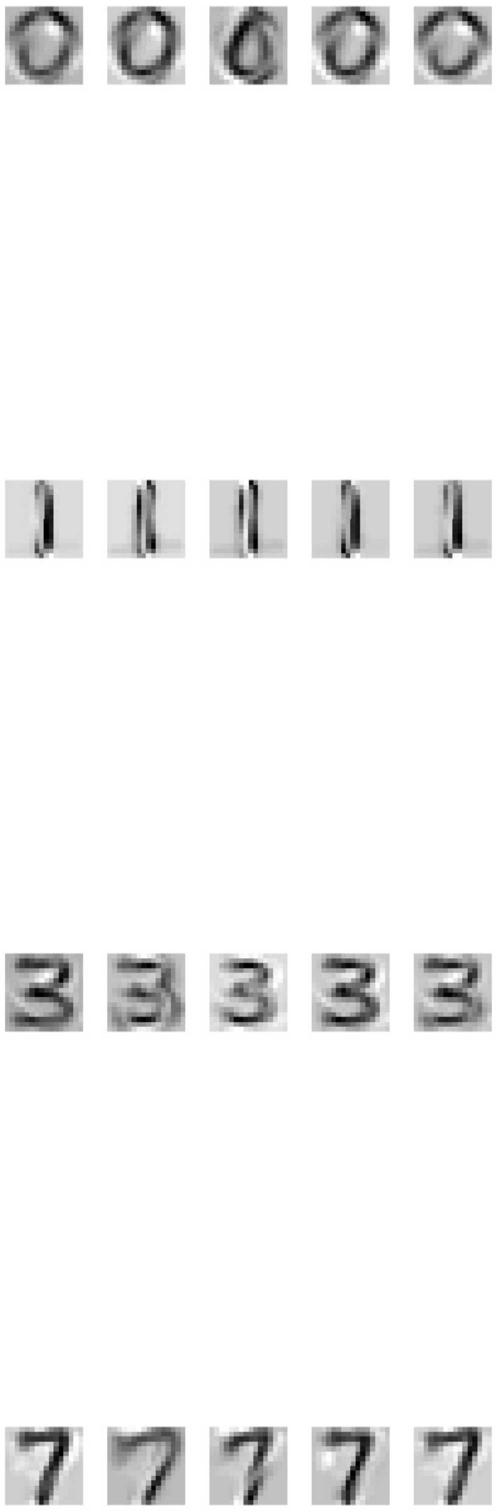

Figure 10: 5 samples for each digit

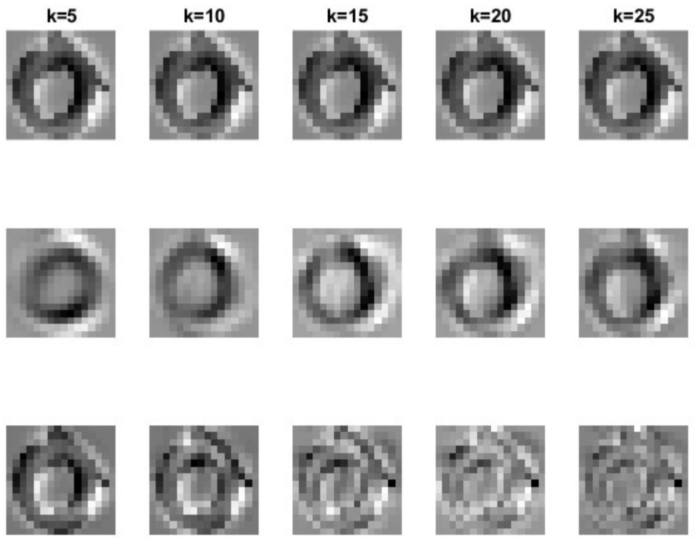

Figure 11: From the top, the Original Sample, Approximation, Residuals of 0

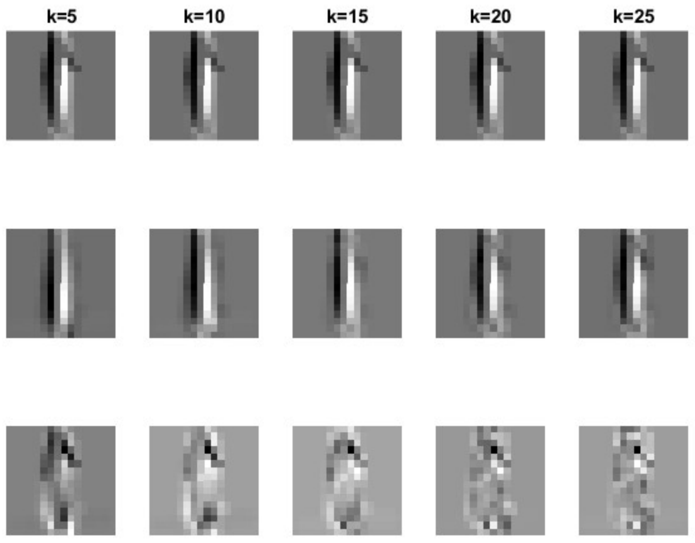

Figure 12: From the top, the Original Sample, Approximation, Residuals of 1

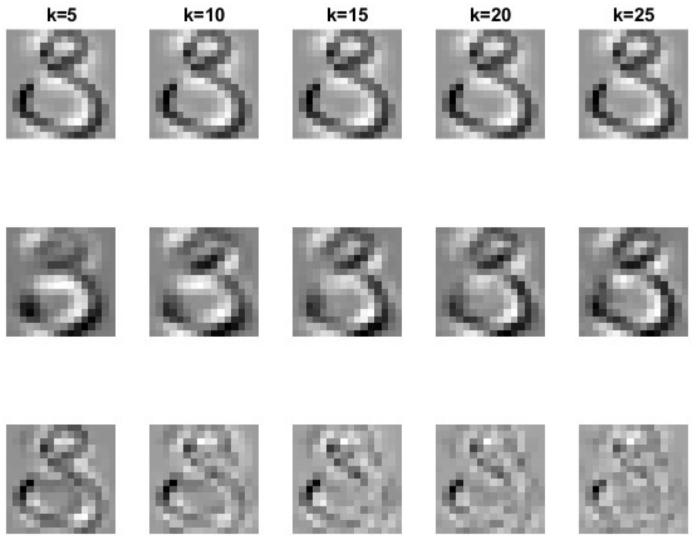

Figure 13: From the top, the Original Sample, Approximation, and Residuals of 3

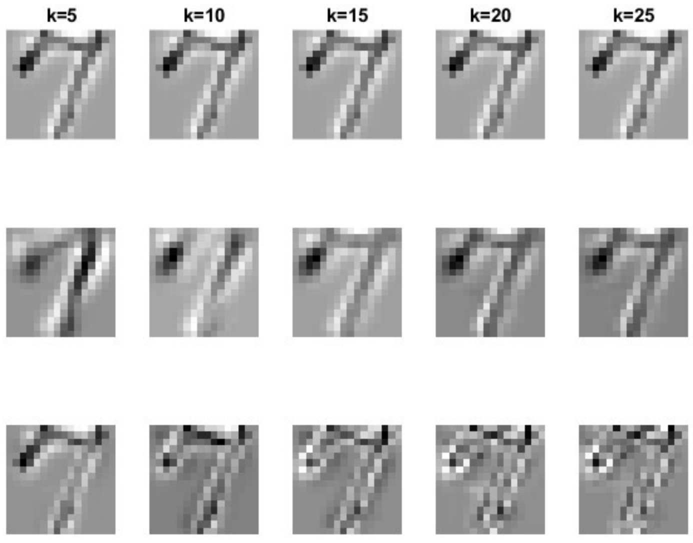

Figure 14: From the top, the Original Sample, Approximation, Residuals of 7

Plot the norms of the errors as a function of k.

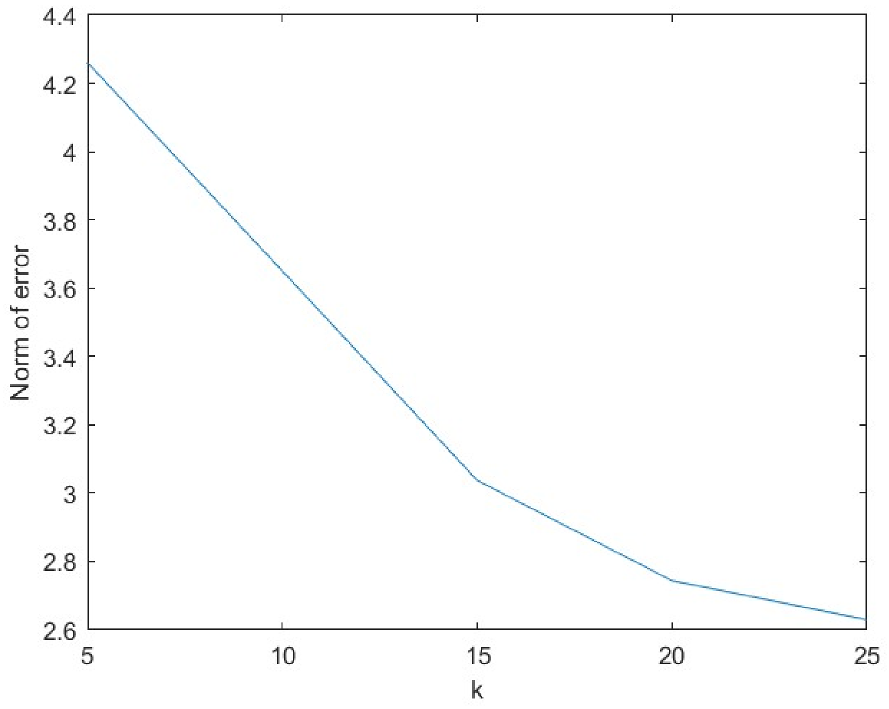

Figure 15: norm of 0

The error norms, which offer a measure of how well the reconstructed images approach the original images, are the new main goal for this part. The plots of the approximated images and residuals give visual evidence of the reconstruction's accuracy. To do this, the code expands by computing the norms of the errors for each value of k and stores them in the array called errors. Once the loop over k values is finished, the code plots the k values against the corresponding norms of the errors.
The norm is a function that measures the size or length of the vector. So here, by using the norms of the residuals as a function of k to measure the size of the error vectors, which had been gotten through the Matlab codes, the plots were extracted. By getting the norm of the errors, it is expected to visualize a decreasing linear graph by increasing the number of components. This proves that as had been seen, as much as the k goes upper, the approximation becomes more accurate while the error gets less, with regard to the formula that
the Original sample = the approximation obtained by projecting the original sample onto the Principal Component+ Residual error.
Using these techniques can help to determine the optimal number of principal components to use. By including more principal components, the norms of errors decrease. However, after a certain number of components, like 15 in the norm of 3, adding more components would not help us significantly improve the accuracy of the reconstructions. So here, we can conclude that it is so important to select the optimal and ideal number of the principal components to reduce the risk of overfitting.

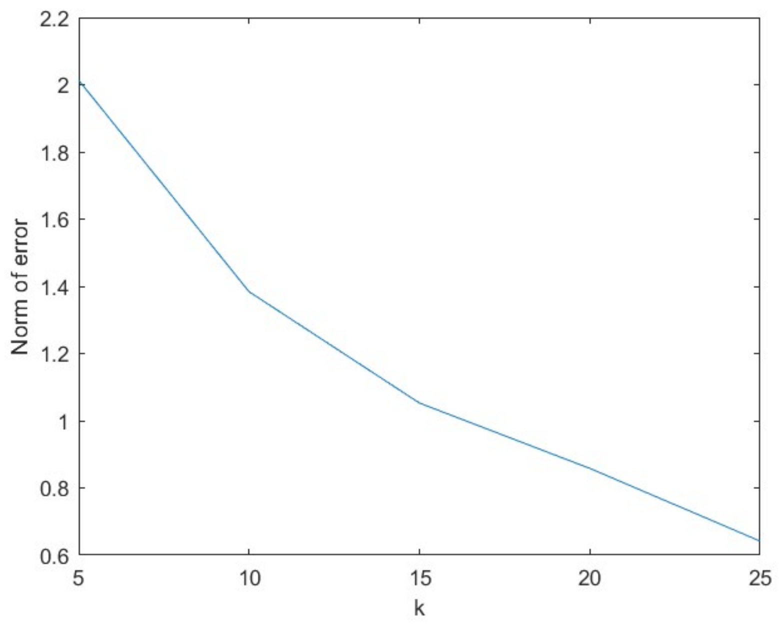

Figure 16: norm of 1

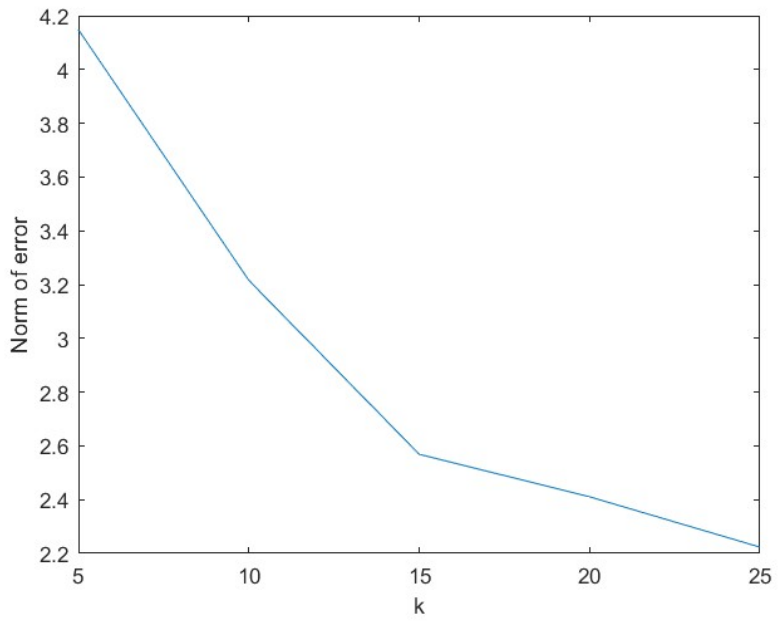

Figure 17: norm of 3

Figure 18: norm of 7

```
close all
clear

load HandwrittenDigits.mat

% Extract from X the images that correspond to numbers 0, 1, 3
   and 7
for j=[0,1,3,7]
    Ij=find(I==j);
    Y= X(:,Ij);

    %Centering the data
    y_c =1/length(Ij)*sum(Y,2);
    Y_c = Y-y_c*ones(1,length(Ij));

    %Singular value decomposition of Y_c:
    [U,D,V]= svd(Y_c);

    % Select a subset of samples
    Xj= Y_c(:,1:5);
    for l=1:size(Xj,2) %Fix the column of samples
    % Loop over the values of k
    figure;
     k=1;
     for k_values=[5, 10, 15, 20, 25]

        % Compute the projection onto the first k principal
           components
        Z = U(:, 1:k_values)'* Xj(:,l);

        % Reconstruct the samples using the projection
           coefficients and the first k principal components
        ZP  = U(:,1:k_values)*Z;

        % Compute the residual for each sample
        residual=Xj(:,l)-ZP;

        % Compute the norms of the errors for this value of k
        errors(k_values/5) = norm(residual);




                hold on
                subplot(3, 5, k)
                imagesc(reshape(Xj(:, l), 16, 16)')
                axis('square');
                axis('off');
                title(sprintf('k=%d',k_values));
                subplot(3, 5, k+5)
                imagesc(reshape(ZP , 16, 16)')
                axis('square');
                axis('off');
                subplot(3, 5, k+10)
                imagesc(reshape(residual, 16, 16)')
                colormap(1-gray);
                axis('square');
                axis('off');

                k=k+1;

            end
        end

        % Plot the norms of the errors as a function of k for this
           group of samples
        figure();
        plot([5, 10, 15, 20, 25], errors);
        xlabel('k');
        ylabel ('Norm of error');

    end
```

Exercise 3

1. Download from Blackboard the data file IrisData.mat. The matrix X consists of 150 vectors, each one having four components. The data correspond to measurements of certain dimensions in three species of flowers, Iris setosa, Iris versicolor, and Iris virginica, and the components of the data vectors have the following attributes:
   X= sepal length in cm
   sepal width in cm
   petal length in cm
   petal width in cm

2. By using the PCA, investigate if the data set suggests the presence of clusters that would make it possible to separate the three species from each other. (Later on, the flower species corresponding to each data point will be made available.)

To begin, we load the IrisDataAnnotated.mat. The same as before, we should compute the mean vector x-c of the data, which represents the average value of each column in X as we had done in the first question. Then, by computing the SVD of the centered data, we decompose the centered data matrix X-c into three matrices of UDV. These matrices can be used to compute the principal components of the data. To check the Singualr values, we got the plot of SVD which shows the Singualr indexes and values, just in case of having some assumptions. The plot helps to visualize the amount of variance (variability)in the data captured by each singular value. The index of the singular value corresponds to the principal component. so then, by plotting the singular values, we can see how many principal components we need to choose. It helps to visualize the data in a lower-dimensional space by understanding how many principal components have the most information. After, the projection of the centered data onto the first two principal components is computed by multiplying the transpose of the first two columns of U by the centered data matrix X-c. Finally, a scatter plot of the first two principal components is created. The first principal component is plotted on the x-axis and the second principal component is plotted on the y-axis.

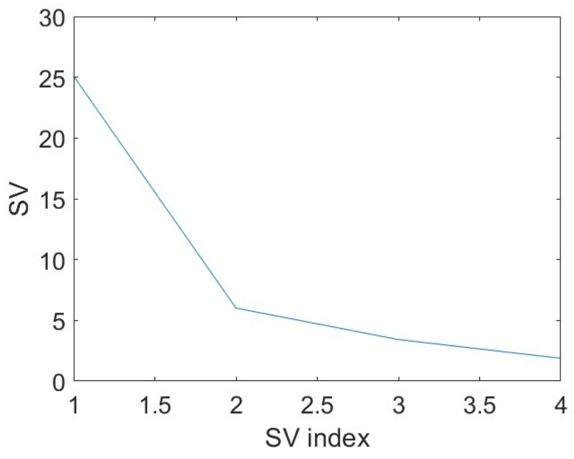

Figure 19: Singular Value Plot

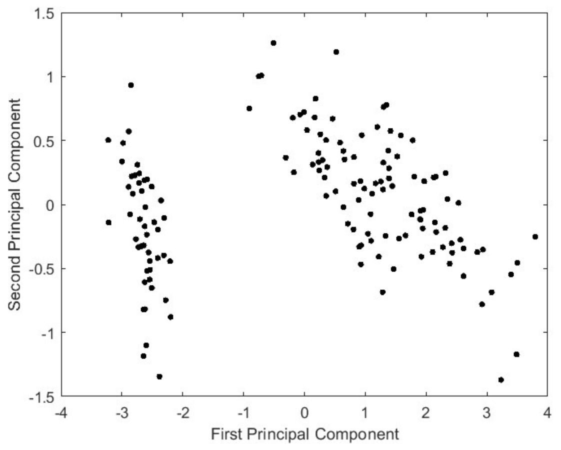

Figure 20: the first 2 Principal Components' plot

By this scatterplot, we only get 2 clusters, which suggests that the data may not be easily separable into three distinct clusters corresponding to the three different flower species (Iris setosa, Iris versicolor, and Iris virginica). Instead, the data may only have two clusters that are captured by the first two principal components.
So now, we try to check the three first components, even though the SV plot showed that the third component would not be so helpful.

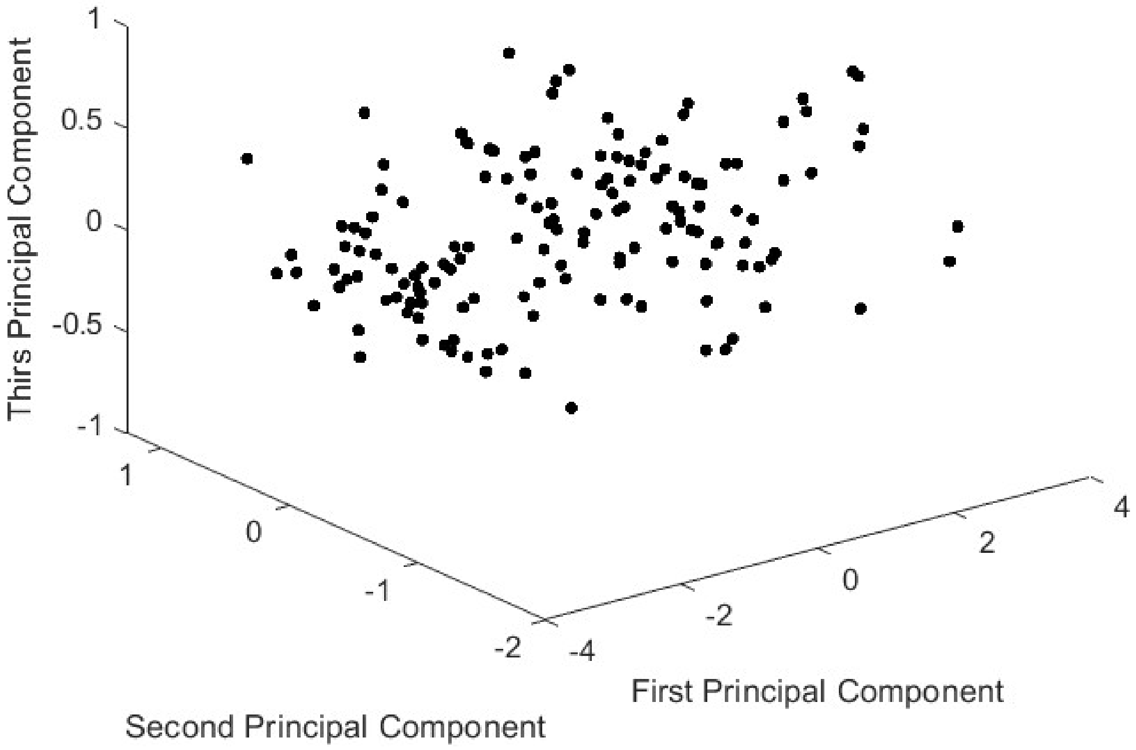

Figure 21: the first 3 Principal Components' plot

It is hard to recognize more than 2 clusters in this plot. So it may be concluded that that the sepal and petal measurements used to describe the species of the iris may not be sufficient to distinguish between the three species. This could be due to the fact that there is overlap in the measurements of the different species, making it difficult to differentiate them based on these two elements only.
At the end, we want to investigate if the data set suggests the presence of clusters that would make it possible to separate the three species. To do this, we change the code of the plot to scatter which suggests us the clusters in the graph itself.

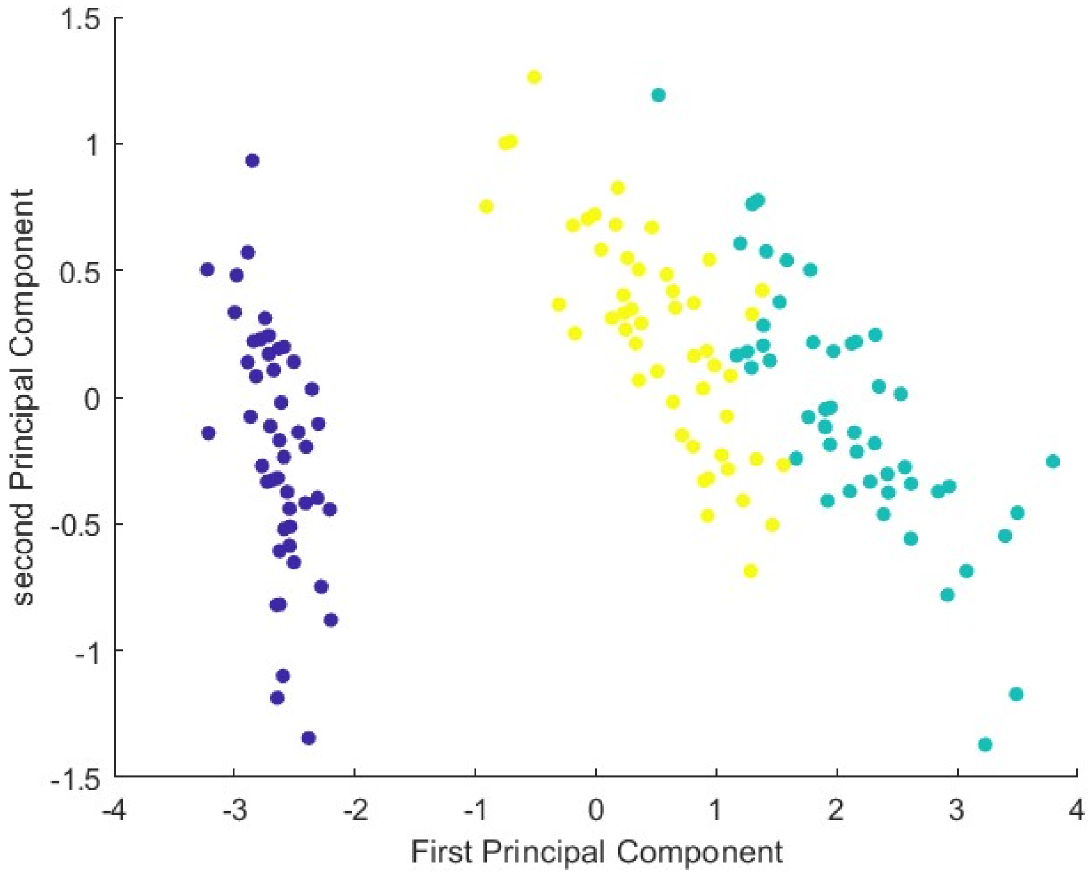

Figure 22: the first 2 Principal Components' plot

Here, by coloring, the figure suggests three clusters that look like to be only two. The coloring could be done only because we already know the titles, while in the real world, the titles are mostly unknown.
In the end, by only looking at the graph, two clusters can be seen and not three, which means that the 2 characteristics are so similar they cannot be distinguished, and it may be essential to consider additional characteristics to reliably find out the different species above all.

```
% Load the IrisData.mat file
load IrisDataAnnotated.mat

% Compute the mean vector and subtract it from the data

x_c = 1/length(X)*sum(X,2);
X_c = X - x_c*ones(1,length(X));

% Compute the SVD of the centered data

[U,D,V] = svd(X_c);
sigma=diag(D);
figure(1)
plot(sigma)
xlabel('SV index')
ylabel('SV')
set(gca, 'FontSize', 16)

% Project the centered data onto the first two principal
   components
Z = U(:,1:2)'*X_c;

% Create a scatter plot of the first two principal components

figure(2)
plot(Z(1,:),Z(2,:),'k.','MarkerSize',10);
xlabel('First Principal Component');
ylabel('Second Principal Component');

% Project the centered data onto the first three principal
   components

Z_1= U(:,1:3)'*X_c;

% Create a scatter plot of the first three principal components

figure(3)
plot3(Z_1(1,:),Z_1(2,:),Z_1(3,:),'k.','MarkerSize',10);
xlabel('First Principal Component');
ylabel('Second Principal Component');
zlabel('Thirs Principal Component');

%Coloring the three clusters
figure(4)
scatter(Z(1,:),Z(2,:), 25, I, 'filled');
xlabel('First Principal Component');
ylabel('second Principal Component');
```

Exercise 4

1. Yale Face Database is a popular facial image database, containing black-and-white photographs of 15 individuals with different facial expressions (normal, sad, happy, surprised,sleepy, wink), with or without eyeglasses, and in different lighting (center, left or right lighting), 11 photos of each one of them.
   The goal of this project is to investigate how well the low-rank approximation of the dataset reproduced the images.

2. Plot 5 different columns of the matrix Xr as grayscale images. Choose faces that you think are in some sense representative of different groups.

3. After having computed the first singular values/vectors, plot the singular values and comment on how fast (or slowly) they decrease. Notice, however, that computing the full SVD may be very slow, so use svds, increasing gradually r. It may be more informative to plot their logarithms.

4. Plot the first 5 feature vectors (columns of U) as gray scale images.

5. Approximate the 5 images corresponding to the columns that you selected by a linear combination of the first k = 4; 8; 15 feature vectors with coefficients their principal components. Each time display the approximation and difference between it and the original data in the form of a gray scale image.

This exercise is a combination of the all first exercises which helps to use the whole methods of PCA and SVD together. It investigates how well the low-rank approximation of the Yale Face Database can reproduce images.
The resulting of low-rank approximation is displayed and plotted to demonstrate its ability to preserve essential characteristics of the original images. The singular values and feature vectors, with a specified rank r1=80, are also plotted to show their importance in the approximation. Finally, the code approximates selected images using a linear combination of the first few feature vectors, and the approximation and difference between the original and the approximation (residuals)are displayed to show how well the approximation reproduces the original images and how much error is introduced as k increases.

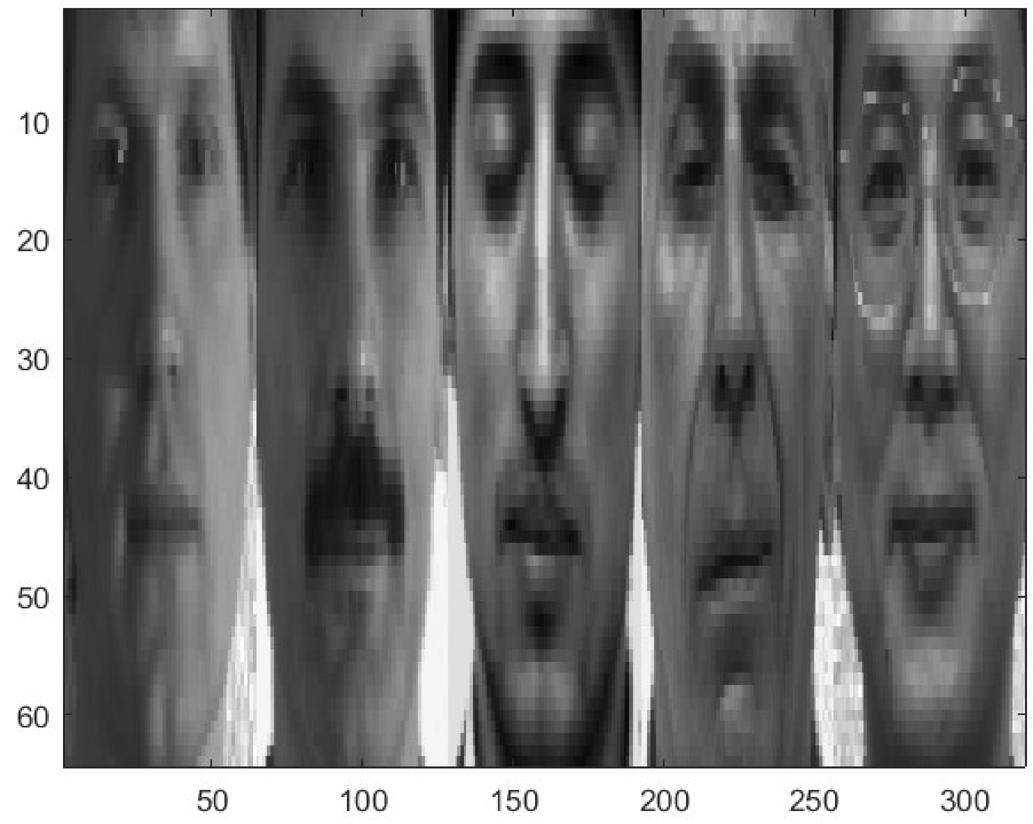

Figure 23: Selecting 5 random columns from different groups of Xr1

Here, by choosing 5 images from the reconstructed dataset of X-r1, randomly i=[9,20,33,47,59], these pictures are produced. The aim was to select 5 different columns of the matrix with a rank of 80, which represent some different groups in some sense. By choosing different numbers which do not have any order, we can say that not only the people are different, but in some sense, their facial expressions are also varied.

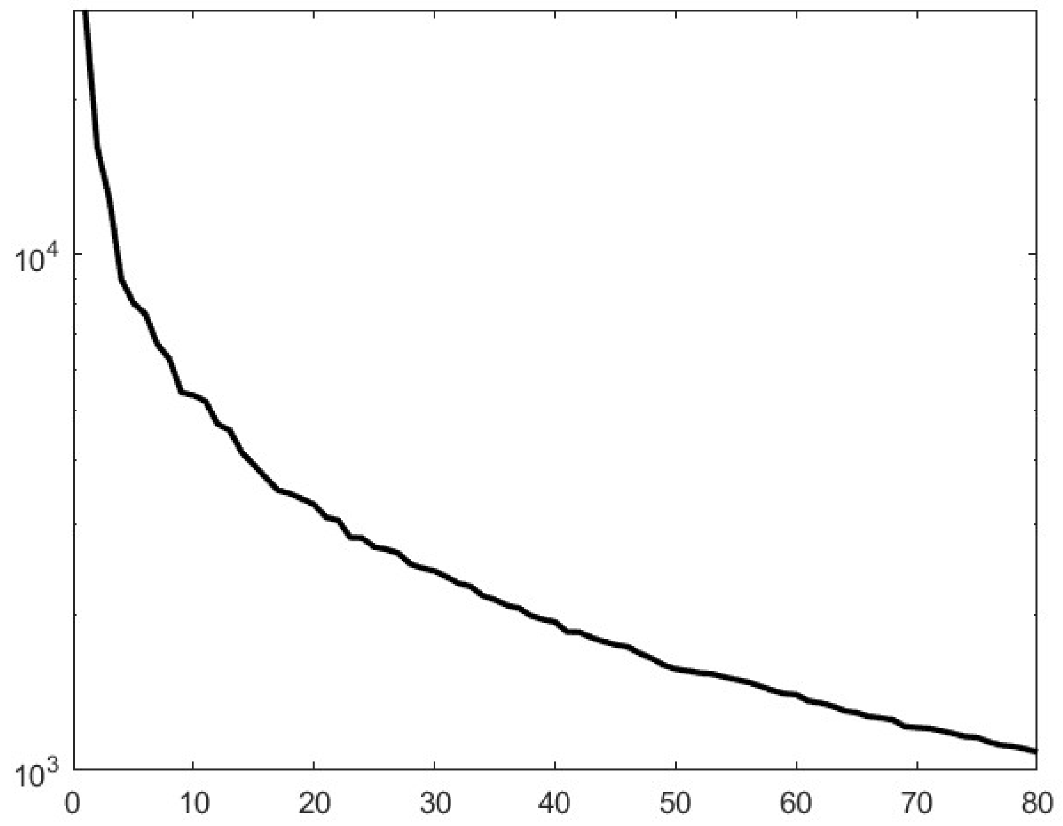

Figure 24: The plot of the logarithm of the singular values

Now, the Singular values on a logarithmic scale have been plotted. The goal is to understand how much each singular vector contains from the data set. By visualizing this plot, it can be seen that the singular values are decreasing gradually. So, we can conclude that the first singular values explain most of the data set.

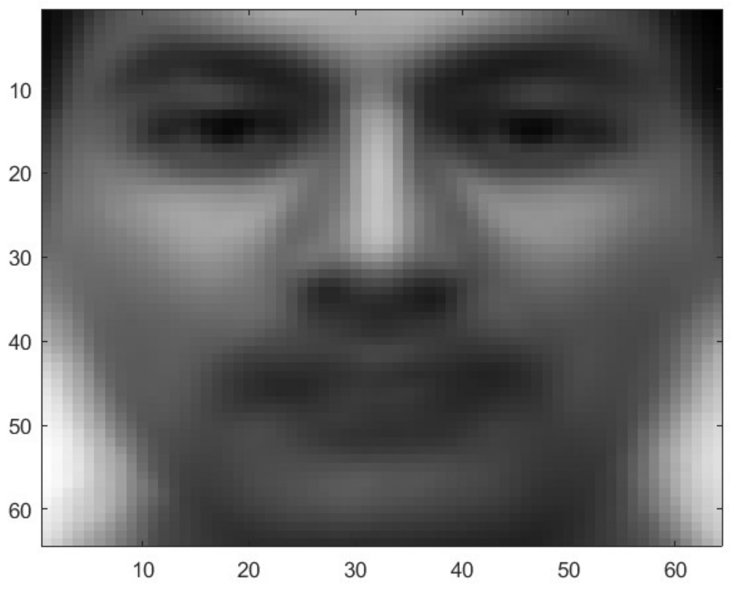

Figure 25

This part of the code plots the first 5 feature vectors, the columns of U. As had been mentioned during the class, each feature vector of U is like a layer that creates a part of the face. For example, one layer may correspond to the glasses, while another may show the mustache. So, the approximation is actually made by giving some weight to each of these feature vectors. To continue, we can say that if someone has the glasses, the weight of the feature vector of the glasses would become high, while if the same person does not have a mustache, then the corresponding feature vector would not contain a high weight. It means that the approximation of the person's figure would get the glasses, but not the mustache. To conclude, the principal components are the ones that give higher weights to the main features. In the end, it means these vectors belong to the span generated by the columns of U, so any vector is like the linear combination of all the columns of U multiplied by the weights of the components, making the composition of the approximations of the individuals.

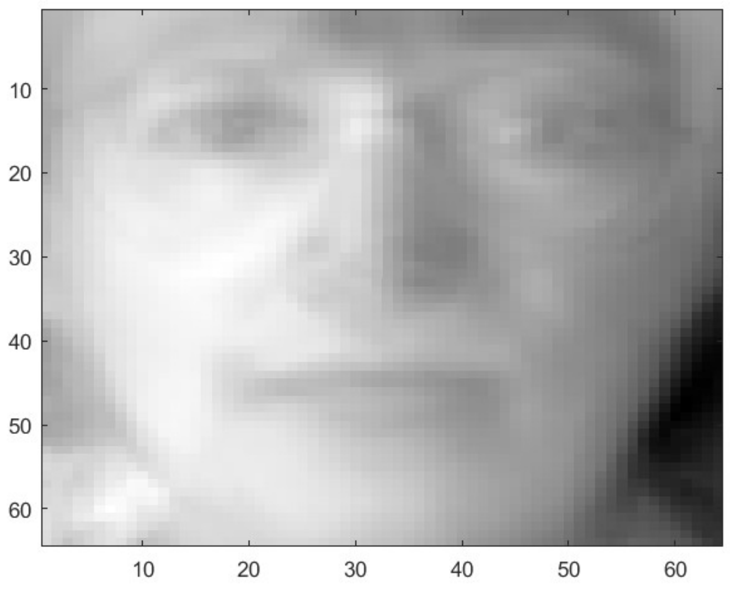

Figure 26

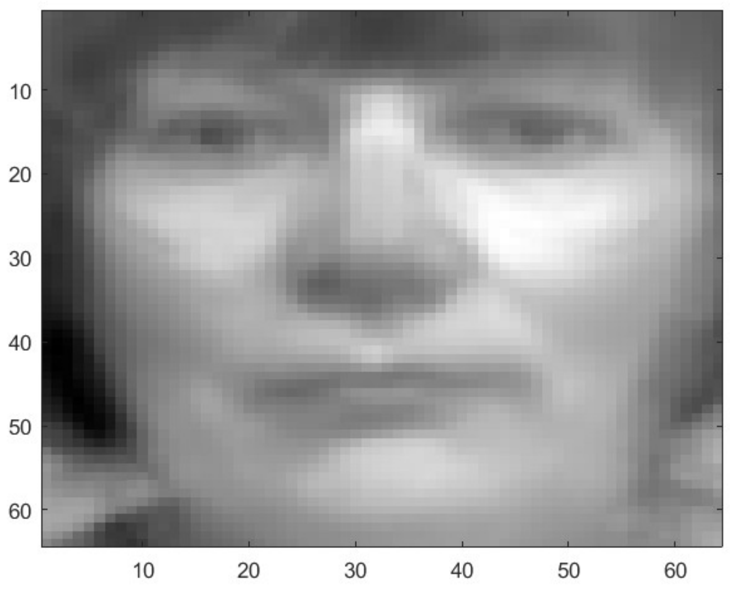

Figure 27

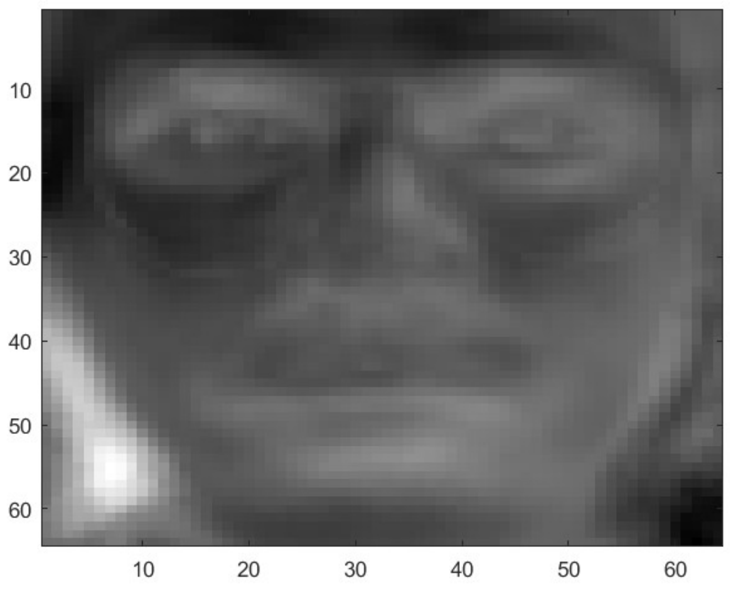

Figure 28

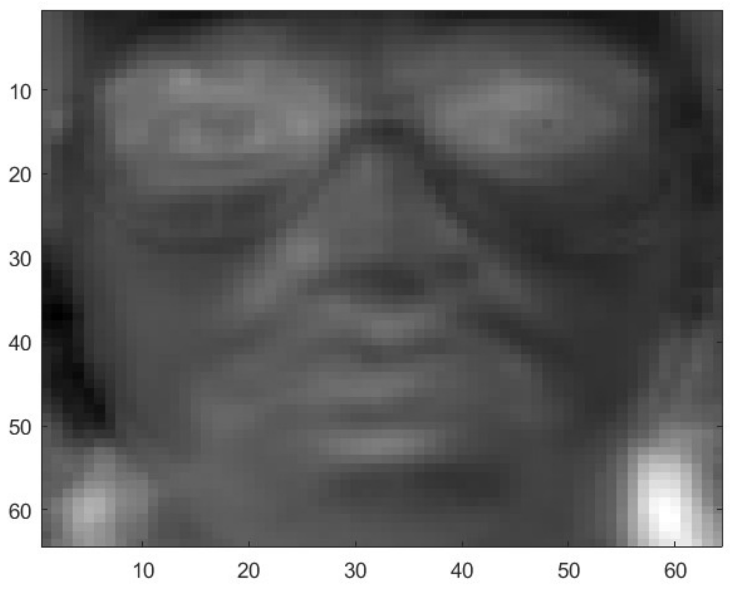

Figure 29

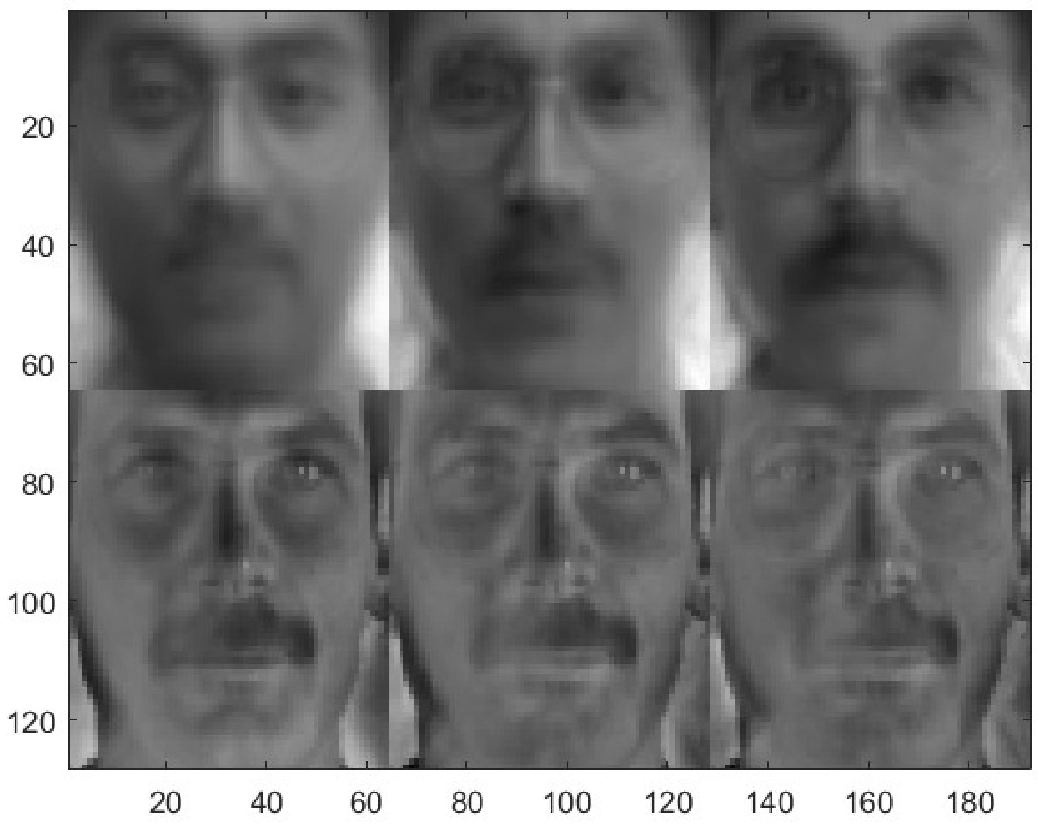

Figure 30: The approximation and residual of k components

In the end, the last question aims to approximate the 5 selected faces (9, 20, 33, 47, 59) using a low-rank approximation with different values of the rank of k= (4, 8, and 15). In each row of the grid(2*3), there are three images side by side. The first row corresponds to the approximation, while the second row shows the residuals. The low-rank approximation is obtained by projecting the face image onto the subspace spanned by the first k singular vectors (the rank's value used for the approximation). By visualizing the figures, we can realize by adding k, the approximation becomes more accurate while the error becomes less. That is the reason why the first picture on the left bottom is the most accurate, as it has the least rank of 4. So it makes sense that the residual would be much more similar to the real picture, than the approximation.
With regarding exercise c, we can also mention that by visualizing the figure34, it can be seen that the residuals all have glasses, while the approximation did not get the glasses even by the first 15 components. It may due to the fact that the layer of the glasses would add after the 15th first components, maybe.

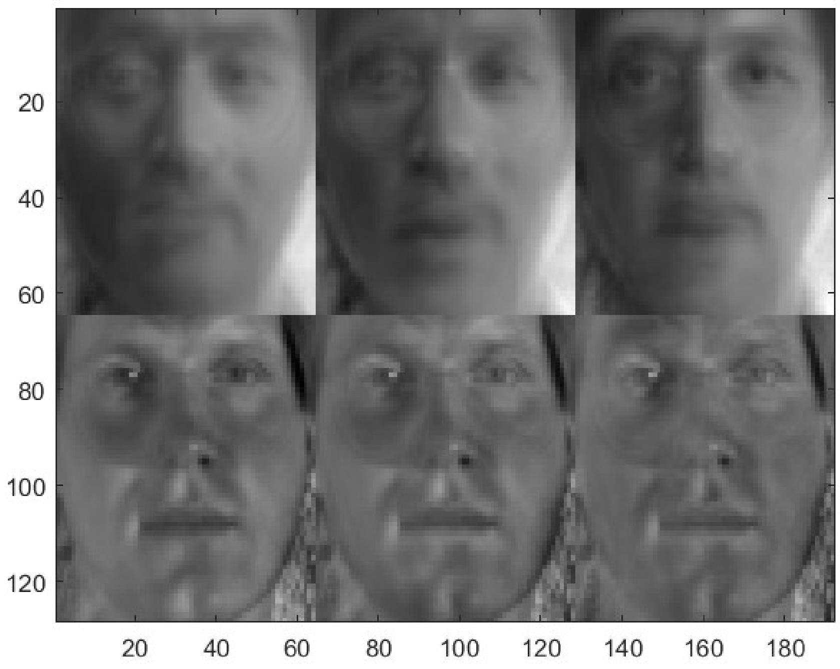

Figure 31: The approximation and residual of k components

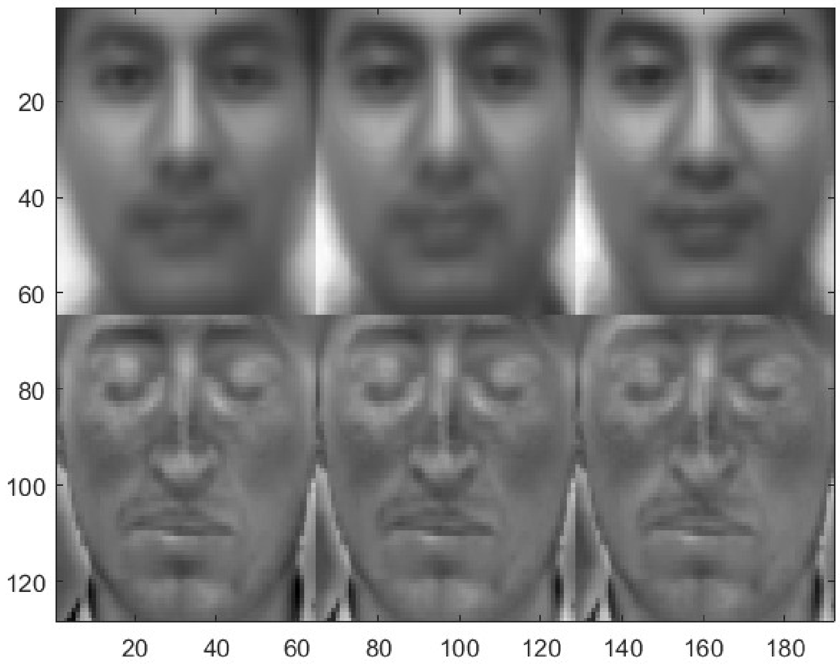

Figure 32: The approximation and residual of k components

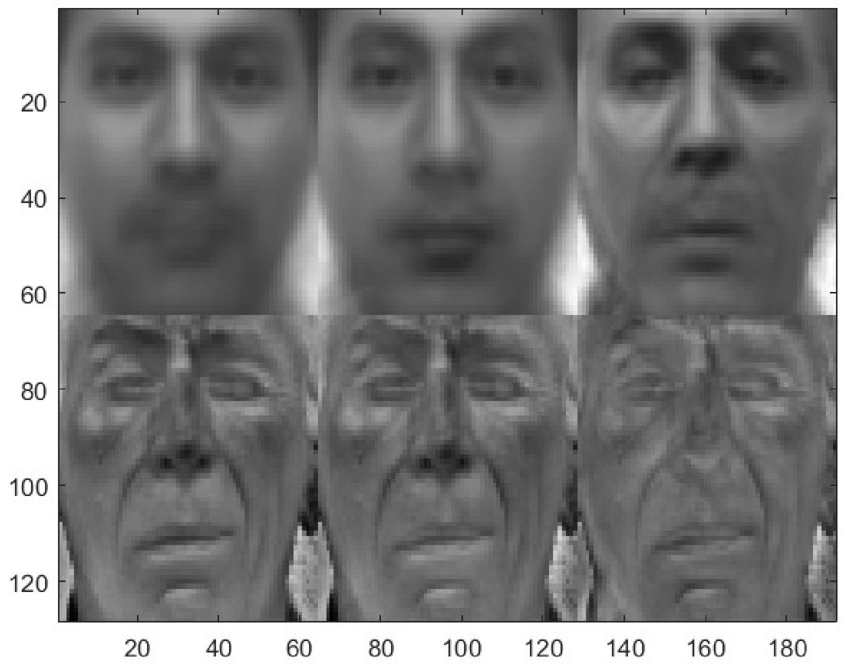

Figure 33: The approximation and residual of k components

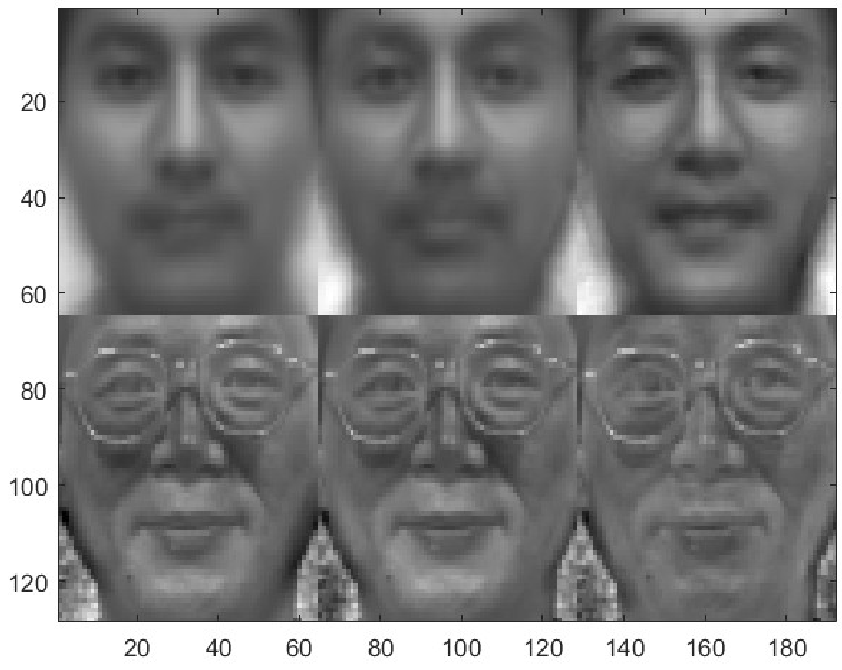

Figure 34: The approximation and residual of k components

```
close all
clear

load Yale_64x64.mat
r1=80;
faceW=64;
faceH=64;
numperLine=5;
showline=1;
Y = zeros(faceH*showline,faceW*numperLine);

% Center the data
x_c =mean(fea,2);
X_c = fea - x_c*ones(1,length(fea));
[U,D,V] = svds(X_c,r1);
X_r1 = U*D*V';

j=0;
for i=[9,20,33,47,59] %Select 5 random representative faces
    if j<6
     Y(1:64,j*faceW+1:(j+1)*faceW)= reshape(X_r1(i,:),[faceH,
        faceW]);
     j=j+1;
    end
  imagesc(Y);
  colormap("gray");
end

%Plot the log of Singular Values of X_c
sigma=diag(D);
log_sigma=log(sigma);
figure()
plot(log_sigma)

%Plot the first 5 feature vectors
for a = 1:5
    figure()
    Z = U(:,a)'*X_c;
    imagesc(reshape(Z(1,:),64,64));
    colormap("gray")
end

%approximate the selected images using first k feature vectors
numperLine2=3;
showLine2=2;
Y=zeros(faceH*showLine2,faceW*numperLine2);
for i=[9,20,33,47,59]
  n=0;
  figure()
  for k= [4,8,15]
     k_c =mean(fea,2);
     K_c = fea - k_c*ones(1,length(fea));
     [U,D,V] = svds(K_c,k);
     X_k = U*D*V';
     Y(1:64,n*faceW+1:(n+1)*faceW)=reshape(X_k(i,:),[faceW,faceH
        ]);
     Y(65:128,n*faceW+1:(n+1)*faceW)=reshape(K_c(i,:)-X_k(i,:),[
        faceW,faceH]);
     n=n+1;
     imagesc(Y);
     colormap("gray")
  end
end
```