ECE 311 Lab6 report

Name: Yang Shi, NetID: yangshi5

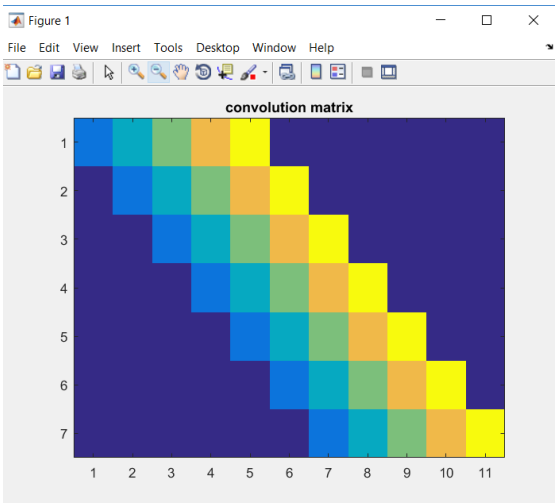
Report Item 1:

Code part:

```
%report_item_1
x = [1, 4, -4, -3,2, 5, -6];
n = length(x);
h = [1,2,3,4,5];
C = convmtx(h, n);
y1 = x*C;
y2 = conv(x, h);
figure(1);
imagesc(C);
title('convolution matrix');
figure(2);
subplot(2, 1, 1);
stem(y1);
title('result using matrix');
subplot(2, 1, 2);
stem(y2);
title('result using conv');
```

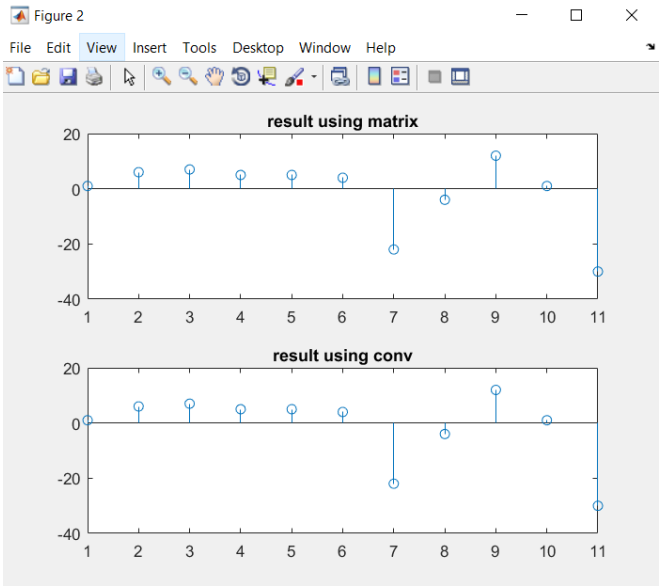
Explanation:

Figure 1 below is the resulting convolution matrix using convmtx:



The row of the matrix contains the h(n) and moves it from left to right row by row.

Figure 2 below is the result of the convolution using the matrix method and conv function.



The two results match with each other perfectly.

Report Item 2:

$A^HA = V\sum^H U^H U\sum V^H = V\sum^H\sum V^H$

$A^HAV = V\sum^H\sum$, therefore, V is the eigen vector of A^HA and since $\sum$ and Λ are both diagonal, $\sum$ is simply the square root of the elements of Λ.

Report Item 3:

Code part:

```
%report_item_3
A = [1, 4, -2; 3, 11, 5; 7, 7, 7];
[V1, D1] = eig(A*A');
[V2, D2] = eig(A'*A);
[U,S,V] = svd(A);
eigenvalue = S*S'
test1 = A*A'*U - U*eigenvalue
test2 = A'*A*V - V*eigenvalue
```

Explanation part:

Using the svd command, we can get the diagonal matrix of eigenvalue is:

290.3648	0	0
0	27.0128	0
0	0	5.6224

And A*A'*U – U*eigenvalue and A'*A*V – V*eigenvalue are both almost 0 which means that U and V are separately the eigenvector of A*A' and A'*A. The result is not completely zero because of the floating-point computation error.

When using the eig function to obtain the eigenvector and eigenvalue, we have the diagonal matrix of eigenvalue to be:

5.6224	0	0
0	27.0128	0
0	0	290.3648

It is simply changing the positon of the three eigenvalues. Therefore, it also makes the eigenvector a little bit different from the one obtained using svd command.

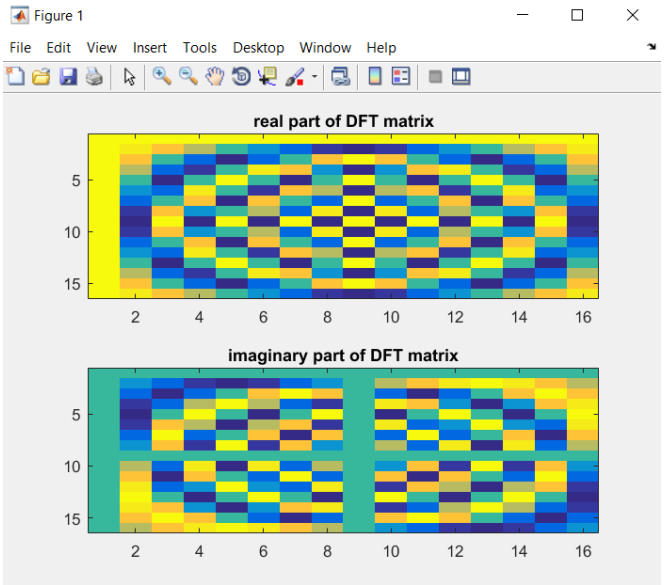
Report Item 4:

Code part:

```
%report_item_4
x = [1,1,4,-4,-3,2,5,-6,3,2,4,-2,5,9,-8,4];
n = length(x);
dm = dftmtx(n);
figure(1);
subplot(2,1,1);
imagesc(real(dm));
title('real part of DFT matrix');
```

Explanation part:

The real and imaginary part of the matrix are both in the plot below. According to the image, in each row of the real part plot, it is like a cosine wave waveform and in each row of the imaginary part plot, it is like a negative sine wave waveform. And with the increase of the row, the frequency increases. The dot product of x and each column of the DFT matrix will be each element of y(n).



Report Item 5:

Code part:

```
%report_item_5
x = [1,1,4,-4,-3,2,5,-6,3,2,4,-2,5,9,-8,4];
n = length(x);
dm = dftmtx(n); %W matrix
y = dm*x'; %result, y = W*x'
inverse = dm'./n;%inverse matrix
X_b = inverse*y %x = W'/N * y;
A = inverse'*inverse;
```

Explanation part:

If the DFT matrix is W, then the inverse DFT matrix = W'/N.

According to the data in matrix A, the dot product of a vector with itself will give us 0.0625 while the dot product of a vector with another discrete vector will give us 0. Therefore, the columns in this matrix are orthogonal with each other.

Report Item 6:

Code part:

Main program:

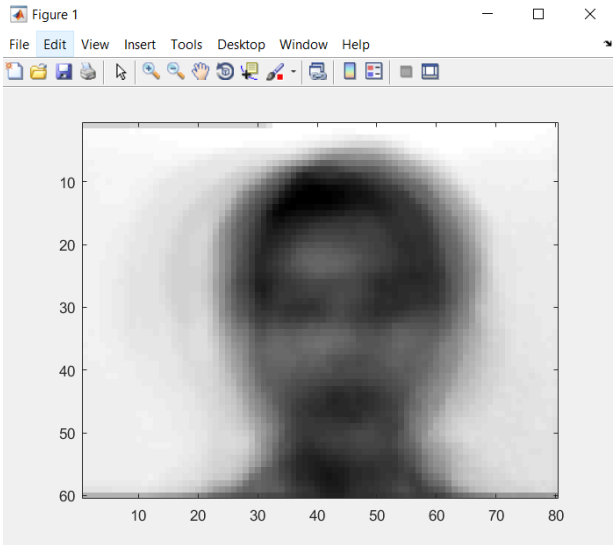
```
%report_item_6
X = loadImages('./yalefaces');
mean = compMeanVec(X);
[row, column] = size(X);
mean_reshape = reshape(mean, [60, 80]);
figure(1);
imagesc(mean_reshape);
colormap(gray);
```

compMeanVec.m:

```
function [ mean ] = compMeanVec(X)
[M, N] = size(X);% M = 165, N = 4800 in this case
mean = zeros(1, N);
for i = 1 : M
    mean = mean + X(i, :)/M;
end
end
```

Explanation part:

The resulting 2D image in a gray color map is like plot below:



It is like the ‘average’ face of all the faces in the database.

Report Item 7:

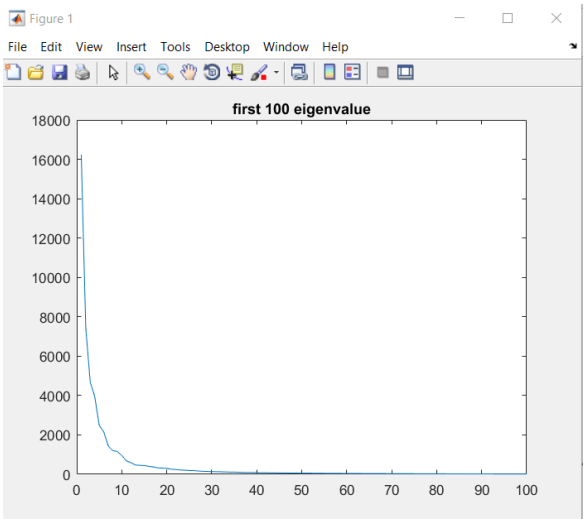
Code part:

```
%report_item_7
X = loadImages('./yalefaces');
mean = compMeanVec(X);
[row, column] = size(X);
Xmean = zeros(row, column);
for i = 1 : row
    Xmean(i, :) = X(i, :) - mean;
end
R = Xmean'*Xmean;% covariance matrix
[U,S,V] = svd(R);%first 100 column of U (same as V) and first
                 %100 column of S
eigenvalue = zeros(1, 100);
for i = 1 : 100
    eigenvalue(i) = S(i, i); %first 100 eigenvalue
end
%reshapre the first 4 eigenvector and the 50th and the 100th
eigen1 = reshape(U(:, 1), [60, 80]);
eigen2 = reshape(U(:, 2), [60, 80]);
eigen3 = reshape(U(:, 3), [60, 80]);
eigen4 = reshape(U(:, 4), [60, 80]);
eigen50 = reshape(U(:, 50), [60, 80]);
eigen100 = reshape(U(:, 100), [60, 80]);
figure(1);
plot(eigenvalue);
title('first 100 eigenvalue');
figure(2);
imagesc(eigen1);
colormap(gray);
title('the first eigenvector image');
figure(3);
imagesc(eigen2);
colormap(gray);
title('the second eigenvector image');
figure(4);
imagesc(eigen3);
colormap(gray);
title('the third eigenvector image');
figure(5);
imagesc(eigen4);
colormap(gray);
title('the fourth eigenvector image');
figure(6); %the 50th image
imagesc(eigen50);
colormap(gray);
```

```
title('the fifth eigenvector image');
figure(7);%the 100th image
imagesc(eigen100);
colormap(gray);
title('the 100th eigenvector image');
```

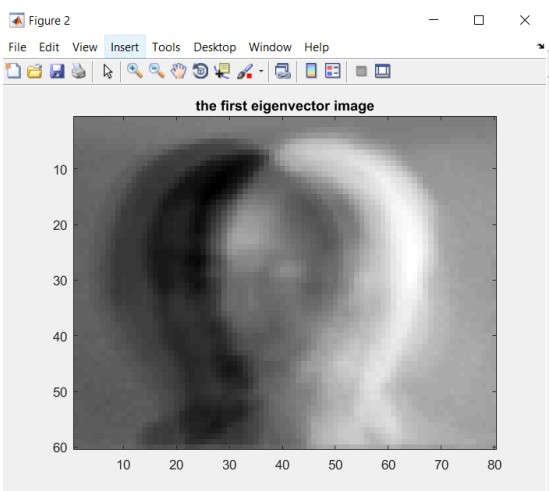
Explanation part:

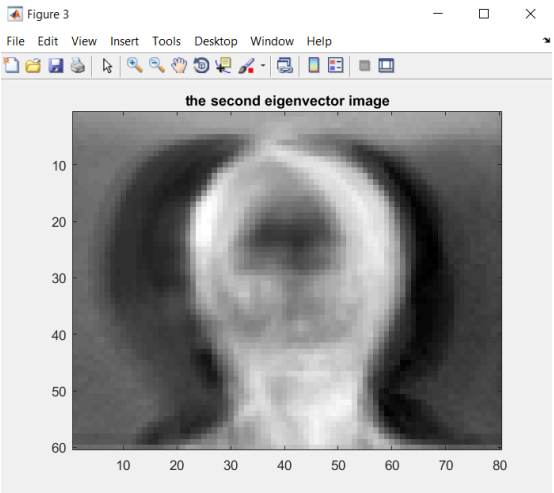
The graph of the first 100 eigenvalues is plotted below:

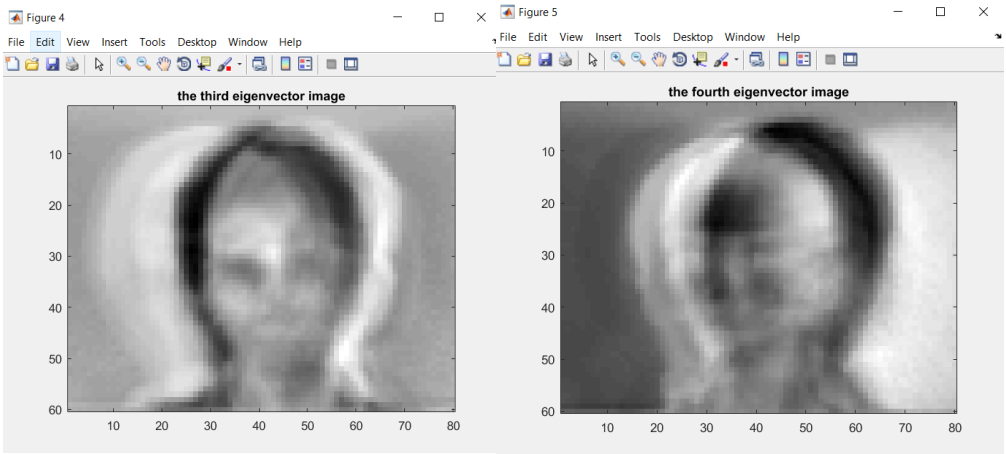


According to the plot, the first 20 eigenvalues is much larger than the rest eigenvalues. The whole image like an exponential decay plot.

The first 4 eigenvectors have been displayed below:

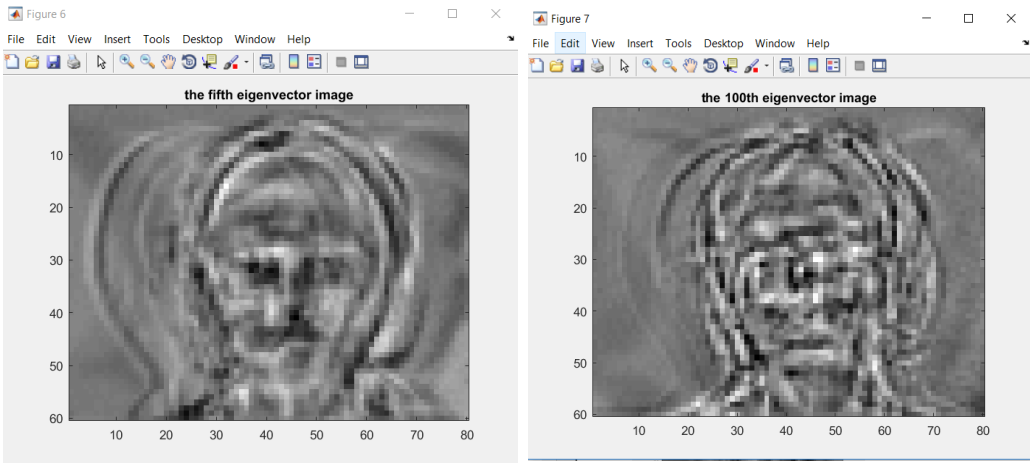






It seems like there is a darker average face coming from left to right and a brighter average face coming from right to left and they overlap with each other at the second and third image.

The 50th and 100th eigenvectors have been displayed below:



I can see more face outlines overlapping with each other on the plots. I think the eigenvectors v_n for n >= 50, will be like an increasing number of face outlines putting together with the increase of n.

Report Item 8:

Code part:

```
function [ pca ] = PCAtransform(mean, V, x)
xnew = x - mean;
pca = V'*xnew';
pca = pca';
end
function [ orig ] = invPCAtransform(mean, V, xpca)
orig = V*xpca' + mean';
```

```
orig = orig';
end
```

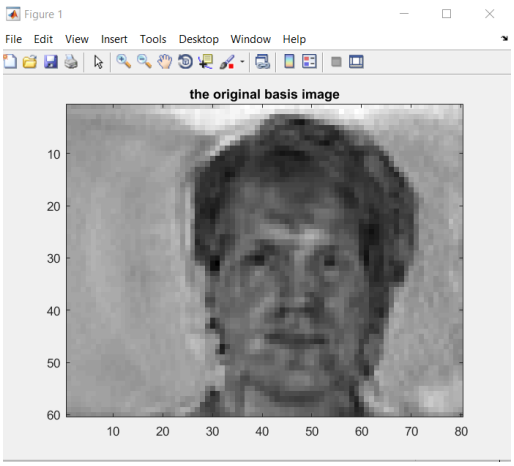
Report Item 9:

Code part:

```
%report_item_9
X = loadImages('./yalefaces');
mean = compMeanVec(X);
[row, column] = size(X);
Xmean = zeros(row, column);
for i = 1 : row
    Xmean(i, :) = X(i, :) - mean;
end
R = Xmean'*Xmean;% covariance matrix
[U,S,V] = svd(R);%first 100 column of U (same as V) and first
                 %100 column of S
Vtest = V(:, 1:100);
A = imread('noisy_face.tiff');
Atest = double(A)./255;
[M, N] = size(Atest);
sizet = M*N;
Aorig = reshape(Atest, [1, sizet]);
Apca = PCAtransform(mean, Vtest, Aorig);
Areorig = invPCAtransform(mean, Vtest, Apca).*255;
Areconst = reshape(Areorig, [M, N]);
figure(1);
imagesc(Areconst);
colormap(gray);
title('the original basis image');
```

Explanation part:

The reconstruct image is plot below:



Most of the noise in the original image has been cancelled out.