Hw 3/4

Problem 1:

A.

```
function [S_k, V_k avgface] = eigenfaces2(filenames, k, s)

row = 64;
col = 64;
image_vector  = @(Bitmap) double(reshape(Bitmap,row*col,1));
vector_image  = @(Vec) reshape( uint8( min(max(Vec,0),255) ), row, col);
vector_render = @(Vec) imshow(vector_image(Vec));

sz = max(size(filenames));
if s > sz || isempty(s),
    s = sz;
end;

for i=1:s
    Face = imread(filenames{i});
    F(i,:) = image_vector(Face);
end;

avgface = mean(F);
vector_render(avgface');

M = ones(s,1) * avgface;

X = F - M;
[U_k,S_k,V_k] = svds( cov(X), k );
```

B.

```
face_descriptions;

n = size(face_features, 1);
o = size(image_to_omit, 1);

goodfaces = [];
evilfaces = [];
for i=1:n
    omit = 0;
    for j=1:o
        if strcmp(face_features(i,1), image_to_omit(j,1))
            omit = 1;
        end;
    end;
    if omit == 0
        if strcmp(face_features(i,4), 'Good')
            goodfaces = [goodfaces; face_features(i, 1)];
        else
            evilfaces = [evilfaces; face_features(i, 1)];
        end;
    end
```

```
end;

[S, V_good, avggood] = eigenfaces2(goodfaces, 30, max(size(goodfaces)));
singvals = diag(S)
singvals =

   1.0e+05 *

  Columns 1 through 12

    5.5912    3.7065    1.9792    1.7895    1.4300    1.2548    0.8248
0.7859    0.6071    0.5792    0.4835    0.4211

  Columns 13 through 24

    0.4057    0.3641    0.3450    0.3140    0.2895    0.2881    0.2513
0.2321    0.2115    0.2085    0.1875    0.1836

  Columns 25 through 30

    0.1769    0.1601    0.1589    0.1510    0.1390    0.1304C.

[S, V_evil, avgevil] = eigenfaces2(evilfaces, 30, max(size(evilfaces)));
singvals = diag(S)'
singvals =

   1.0e+05 *

  Columns 1 through 12

    7.9775    4.0438    3.1472    2.6505    1.6465    1.5754    1.1839
1.0298    0.7799    0.6742    0.6203    0.5543

  Columns 13 through 24

    0.5311    0.4805    0.4391    0.4091    0.3602    0.3466    0.3033
0.2939    0.2814    0.2665    0.2513    0.2458

  Columns 25 through 30

    0.2302    0.2063    0.1925    0.1874    0.1772    0.1750
```

C.

Average goodfaces, produced by the above eigenfaces2 function with vector_render(avgface'):

Average evilfaces:

D.

```
vector_image  = @(Vec) reshape( uint8( min(max(Vec,0),255) ), 64, 64);
vector_render = @(Vec) imshow(vector_image(Vec));
diff = abs(avggood-avgevil);
vector_render(diff)
```

E.

Check for similarity by looking for faces with similar coefficients.

I added this line to the end of my eigenfaces function:

```
C = X*V_k
```

Then I compared coefficients looping through the rows of C_good and C_evil, the coefficients produces via X*V_k for the good eigenfaces and evil eigenfaces:

```
duplicates = [];
s = size(C_good, 1);
for i=1:s
    for j=i+1:s
        if C_good(i,:) == C_good(j,:)
            duplicates = [duplicates; i j];
        end;
```

```
    end;
end;
duplicates =

    74    75
goodfaces{74}
ans =
face102.bmp
```

Result:

I found one duplicate pair (face102.bmp and face103.bmp) using the eigenfaces produced by the goodfaces.

Then I repeated the process for the evil faces and found no duplicates there.

Here are the duplicates displayed:

```
imshow(goodfaces{74})
imshow(goodfaces{75})
```

F.

```
predictions = zeros(1, 47);
for i=1:47
    predictions = [predictions classify(evilfaces{i})];
end;

predictions

%%function------------------------
function result = classify(filename)

face_descriptions;

n = size(face_features, 1);
o = size(image_to_omit, 1);

goodfaces = [];
evilfaces = [];
for i=1:n
    omit = 0;
    for j=1:o
        if strcmp(face_features(i,1), image_to_omit(j,1))
            omit = 1;
        end;
```

```
    end;
    if omit == 0
        if strcmp(face_features(i,4), 'Good')
            goodfaces = [goodfaces; face_features(i, 1)];
        else
            evilfaces = [evilfaces; face_features(i, 1)];
        end;
    end
end;

[S, V_good avggood, C_good] = eigenfaces2(goodfaces, 30,
max(size(goodfaces)));

[S, V_evil avgevil, C_evil] = eigenfaces2(evilfaces, 30,
max(size(evilfaces)));

row = 64;
col = 64;
image_vector  = @(Bitmap) double(reshape(Bitmap,row*col,1));
vector_image  = @(Vec) reshape( uint8( min(max(Vec,0),255) ), row, col);
vector_render = @(Vec) imshow(vector_image(Vec));

Face = imread(filename);
F = image_vector(Face)' ;

dist_good = (F - avggood)*(F - avggood)' ;
dist_evil = (F-avgevil)*(F-avgevil)' ;

if dist_evil < dist_good
    result = 1;
else
    result = 0;
end;
```

Result:

```
predictions =

  Columns 1 through 20

     0     1     1     1     1     0     1     1     1     0     0     0
1     1     0     1     1     0     0     1

  Columns 21 through 40

     1     0     1     0     1     1     1     1     1     1     1     1
1     0     0     1     1     0     1     0

  Columns 41 through 47

     1     0     0     1     0     1     0
sum(predictions)
```

The classifier identified 29 of the 47 evilfaces as evil.

Problem 2:

A.

```
audiofile = 'tune.wav';
[tune, Fs] = audioread(audiofile);
y = tune(:,1); % first track of the stereo clip
Fs =
       44100
```

CD quality sound is 44100, which matches tune.wav.

B.

```
n = size(y, 1);
factor(n)
n =
     1411200

ans =

     2     2     2     2     2     2     2     3     3     5     5     7
7
```

C.

I got an error when I tried to run profile on fft(y) in Matlab because profile doesn't work for built in functions. Instead, I used cputime.

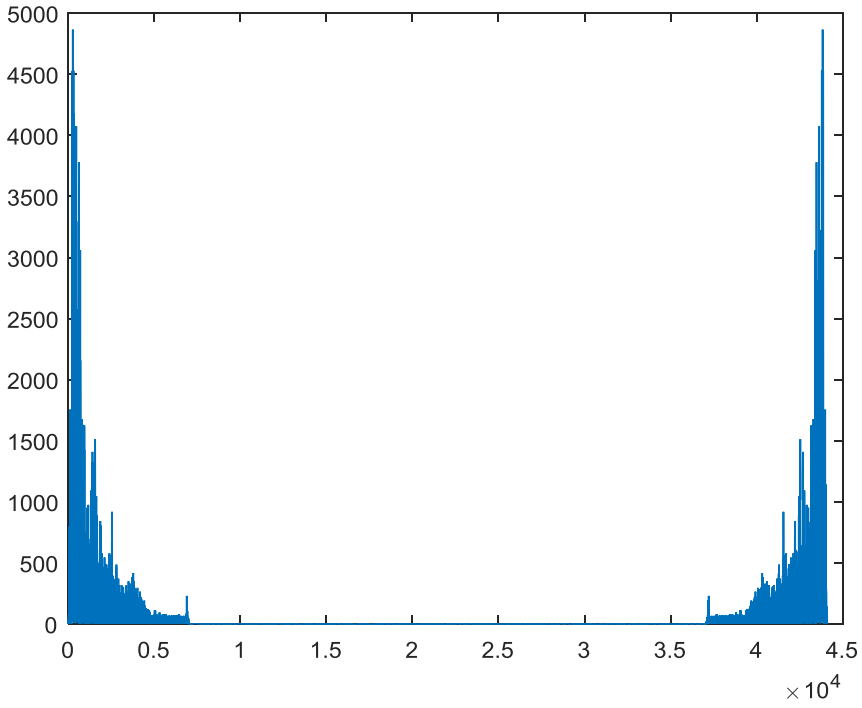
```
t = cputime;
fft(y);
elapsed = cputime - t
```

elapsed =

0.0780

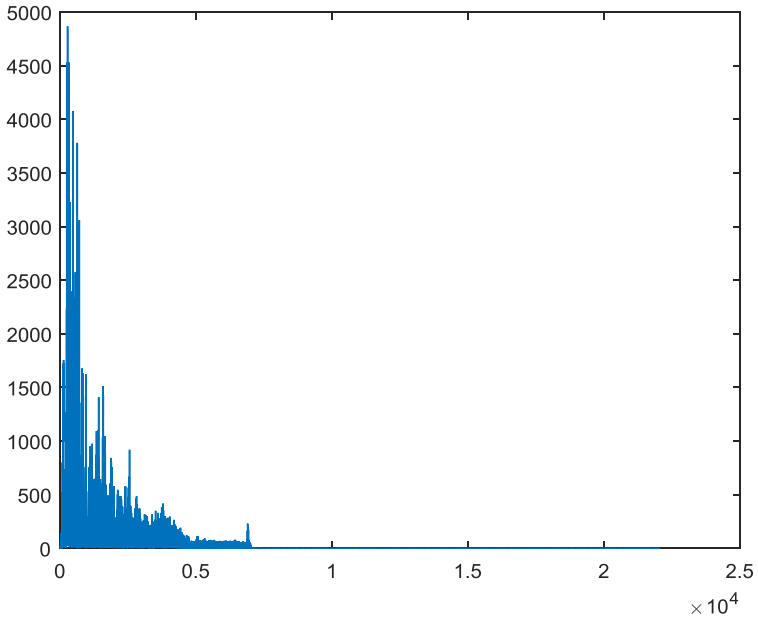
D.

```
power = abs(fft(y));
freqspectrum = linspace(1, Fs, n);
plot(freqspectrum, power);
```



E.

```
n = floor(n/2);
power = power(1:n);
halfspectrum = linspace(1, Fs/2, n);
plot(halfspectrum, power);
```



Nyquist frequency = Fs/2 = 22050.

F.

I chose my frequency ranges by looking at a closeup of the graph produced above. The highest spikes seem to be below 700, so I looked at approximate ranges for those spikes I saw.

```
i1 = 244:290;
i2 = 290:330;
i3 = 200:250;
i4 = 430:500;

[f1 p1] = top_spike(i1, power)
[f2 p2] = top_spike(i2, power)
[f3 p3] = top_spike(i3, power)
[f4 p4] = top_spike(i4, power)

f1 =
   266
p1 =
   3.4649e+03
f2 =
   312
p2 =
   2.0219e+03
f3 =
   235
p3 =
   2.3723e+03
f4 =
   469
p4 =
   1.8121e+03
```

Conclusion: the four frequencies with the highest spikes are approximately 266, 312, 235, 469.

```
%function------------------------------
function [spike_freq,spike_power] = top_spike(frequency_values,power_values)

%ignore spikes from frequencies below 100Hz:
frequency_values = frequency_values(frequency_values >= 100);

%convert from frequency to index
ratio = 22050/705600;
%frequency = 1 + ratio*(index-1);
indices = (frequency_values-1)./ratio + 1;

[spike_power, i] = max(power_values(indices));
spike_freq = frequency_values(i);
```

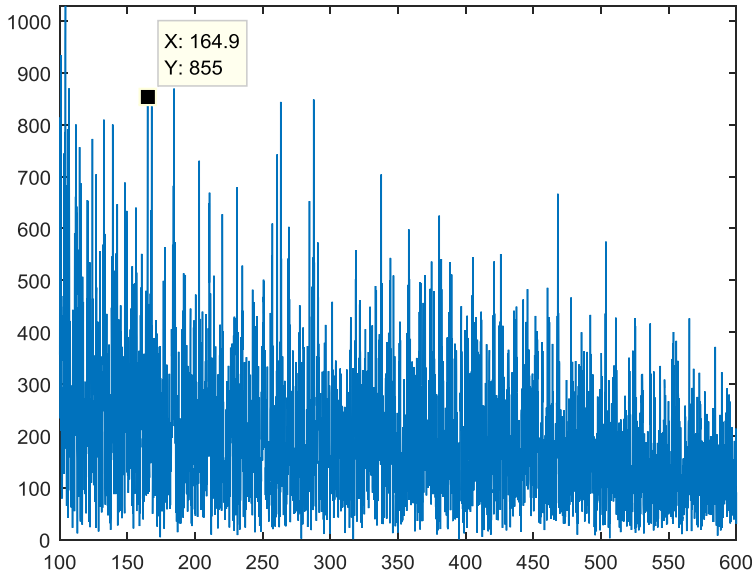
G.

The top spike probably corresponds to the note C#. The song seems to have harmony. It sounds like there is harmony because the spikes within the range of the music scale given in the hw pdf seem to correspond to C#, D#, and A#.

H.

```
audiofile = 'untune.wav';
[untune, Fs] = audioread(audiofile);
y = untune(:,1); % first track of the stereo clip

n = size(y, 1)

power = abs(fft(y));
freqspectrum = linspace(1, Fs, n);
plot(freqspectrum, power);

n = floor(n/2);
power = power(1:n);
halfspectrum = linspace(1, Fs/2, n);
plot(halfspectrum, power);
axis([100 600 0 inf]);
```

Looking at a closeup of the graph, there is a spike at around 104. There are also spikes at 184, 287, 263, 165. It is hard to tell if the song is in harmony, because the frequencies are not very distinct. There are many spikes. But it seems like the song is not in harmony, because the ratios between the spike frequencies are too small.
Here is a closeup of a small range of the graph:



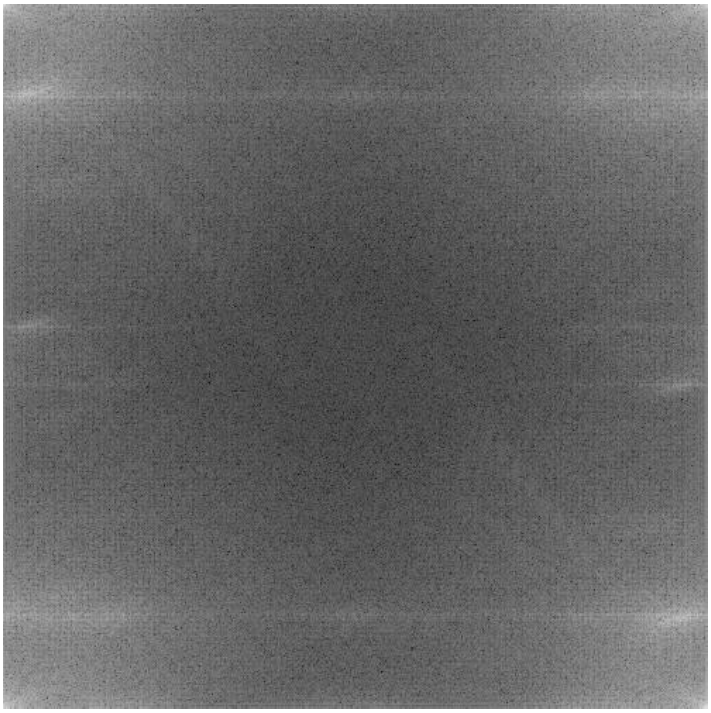
Problem 3:

A.

```
Img = imread('taeyeon.bmp');
```

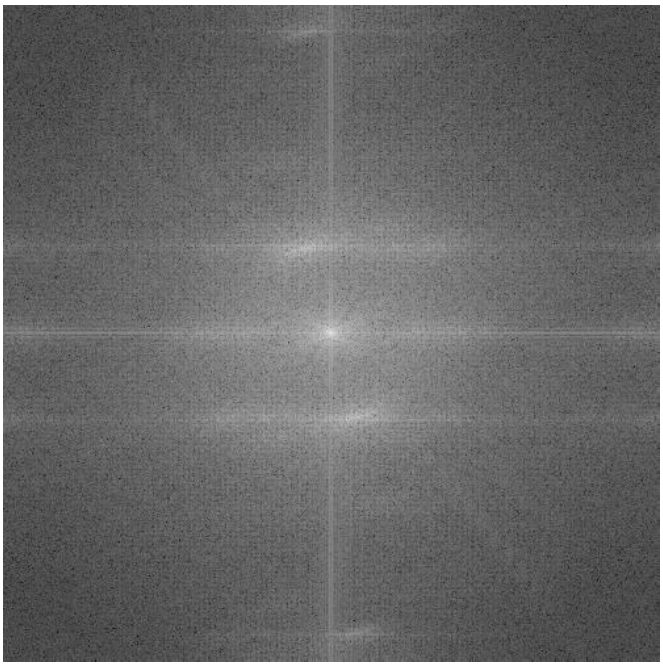
B.

```
Z = fft2(Img);
P = log(abs(Z));
%normalize:
minval = min(min(P));
maxval = max(max(P));
P = (P - minval)/(maxval-minval);
imshow(P);
```



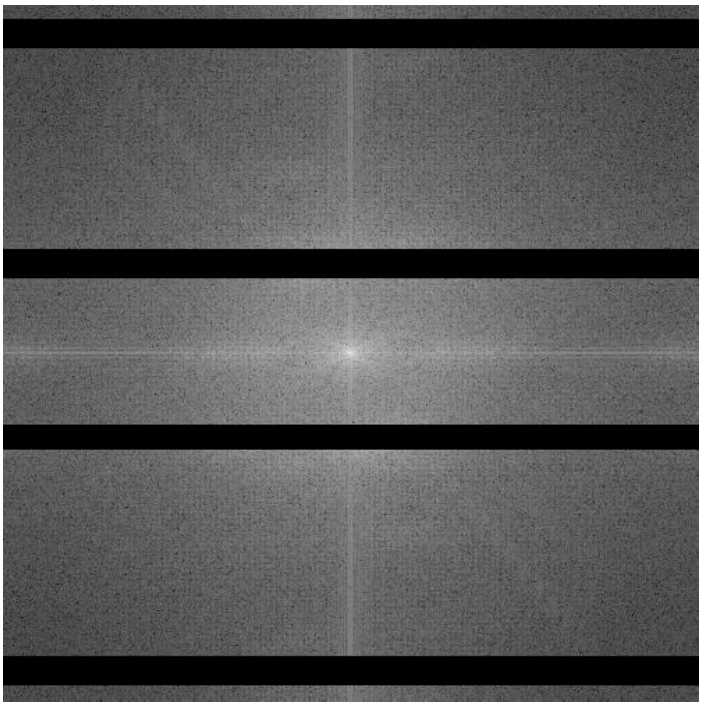
C.

```
Zc = fftshift(Z);
Pc = fftshift(P);
imshow(Pc);
```



D.

```
%top band: row 187-10 to 187+10
%bottom center = 250 + (250-187) = 313
%bottom band: row 313-10 to 313+10
for i=177:197
    Zc(i,:) = 0;
    Pc(i,:) = 0;
end;

for i=303:320
    Zc(i,:) = 0;
    Pc(i,:) = 0;
end;
for i=12:32
    Zc(i,:) = 0;
    Pc(i,:) = 0;
end;

for i=469:489
    Zc(i,:) = 0;
    Pc(i,:) = 0;
end;
imshow(Pc);
```



E.

```
Znew = ifftshift(Zc);
I = ifft2(Znew);
minval = min(min(I));
maxval = max(max(I));
I = (I - minval)/(maxval-minval);
imshow(I);
```

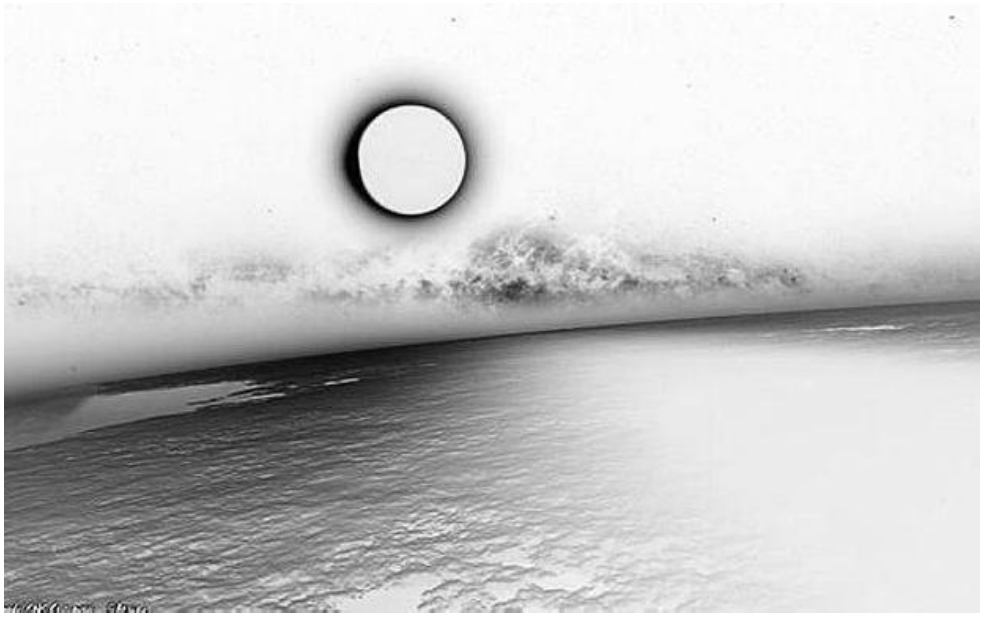
```
imshow(Img); %original taeyeon.bmp
```

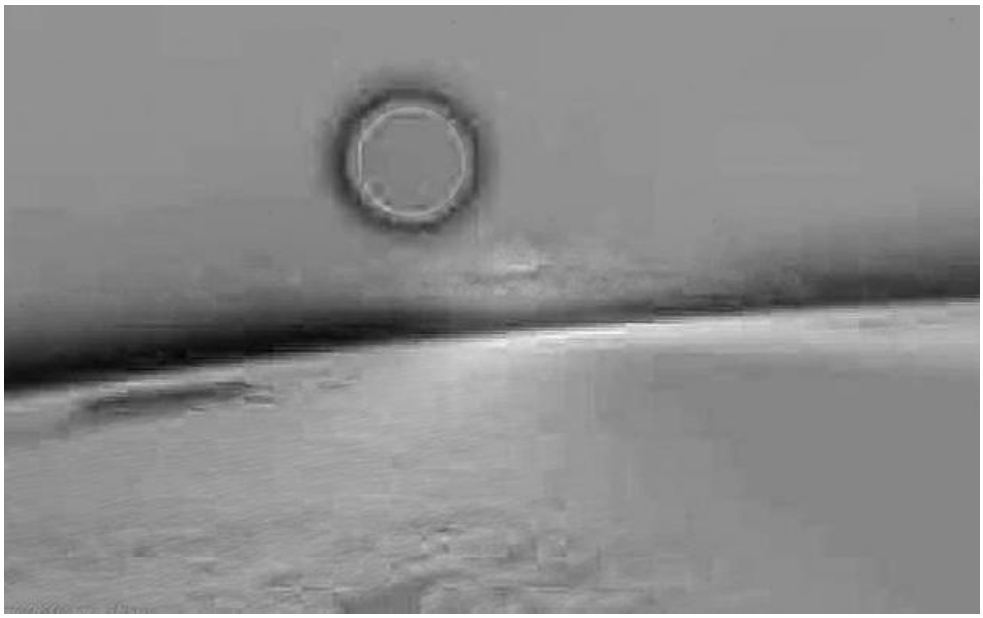
There is some improvement compared to the original. The image is clearer.

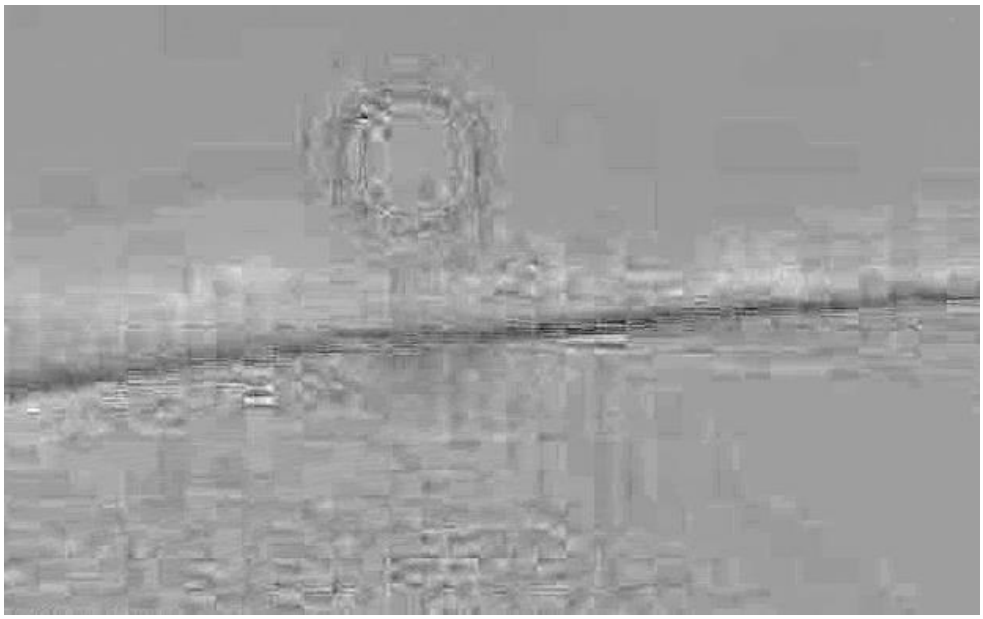
Problem4:

A.

```
function analyzeimage(filename)

Img = double(imread(filename));
m = size(Img, 1);
n = size(Img, 2);

I = reshape(Img, [m*n 3]);
C = cov(I);
[U S V] = svd(C);

I1 = I*V(:,1);
I2 = I*V(:,2);
I3 = I*V(:,3);

Img1 = reshape(I1,m,n);
Img2 = reshape(I2,m,n);
Img3 = reshape(I3,m,n);

imshow(mat2gray(Img1));
figure, imshow(mat2gray(Img2));
figure, imshow(mat2gray(Img3));
```





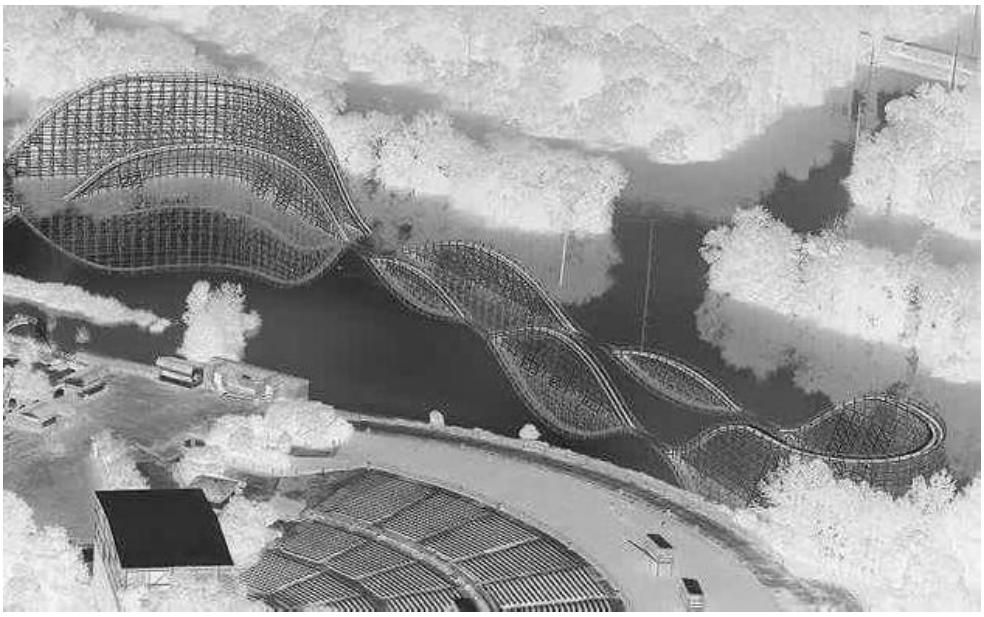


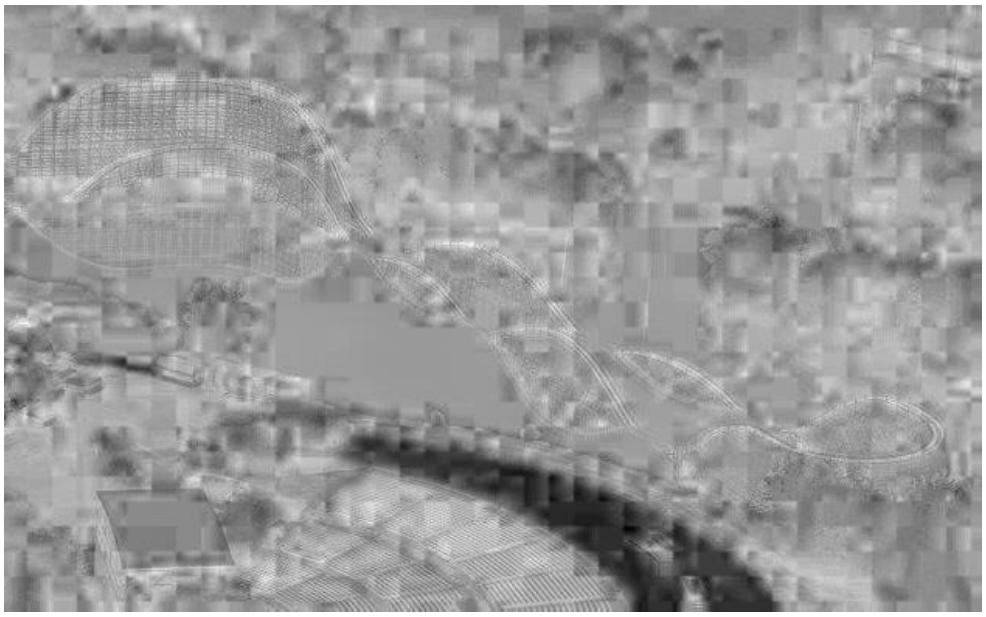
I think the image might be fake because there are very sharp bright spots in the first two projected PC images.

B.

I think this image is also fake, because of the bright and dark contrast of the first PC image.

C.





Conclusion: I think the first and third PC can be a better indication than the second PC, based on

the images above. The first PC shows high contrast for the two fake images, and the third PC shows bright and dark spots for the racoon image. The road and the rollercoaster also stand out in the third PC of the flood image.

Problem6:

A.

```
function result = randgenerator(n)

v1 = rand(n);
v2 = rand(n);

result = v1 + v2 + 2;
```

Estimate variance:

```
n = 10000;
values = randgenerator(n);
avg = mean(values)
variance = var(values)
avg =
    3.0047
variance =
    0.1661
```

B.

```
syms x;
f = (x-2)*heaviside(x-2)*heaviside(3-x) + (4-x)*heaviside(x-3)*heaviside(4-
x);
cdf = int(f, x)
cdf =
2*heaviside(x - 2) - 9*heaviside(x - 3) + 8*heaviside(x - 4) + (x*heaviside(x
- 2)*heaviside(3 - x)*(x - 4))/2 - (x*heaviside(x - 3)*heaviside(4 - x)*(x -
8))/2
```

Interpretation:

F(x) = 0 for x < 2
(x^2)/2 -2x +2 for 2 < x < 3
(x^2)/2 -4x – 7 for 3 < x < 4
1 for x > 4

C.

```
function result = randgenerator(n)

random_percents = rand(1, n);
random_percentiles = icdf('Triangular', random_percents, 2, 3, 4);
result = random_percentiles;
```

Calculate mean:

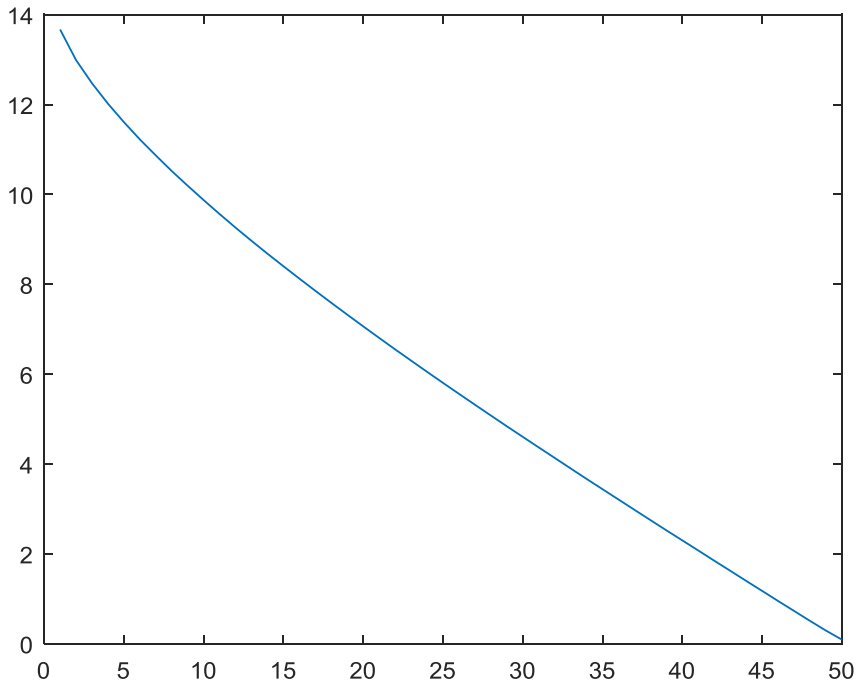
```
result = randgenerator(10000);
avg = mean(result)
avg =
    2.9966
```

D.

```
function result = randgenerator(n)

random_percents = rand(1, n);

random_percentiles = icdf('Lognormal', random_percents, 0, 1);
result = random_percentiles;
```

E.

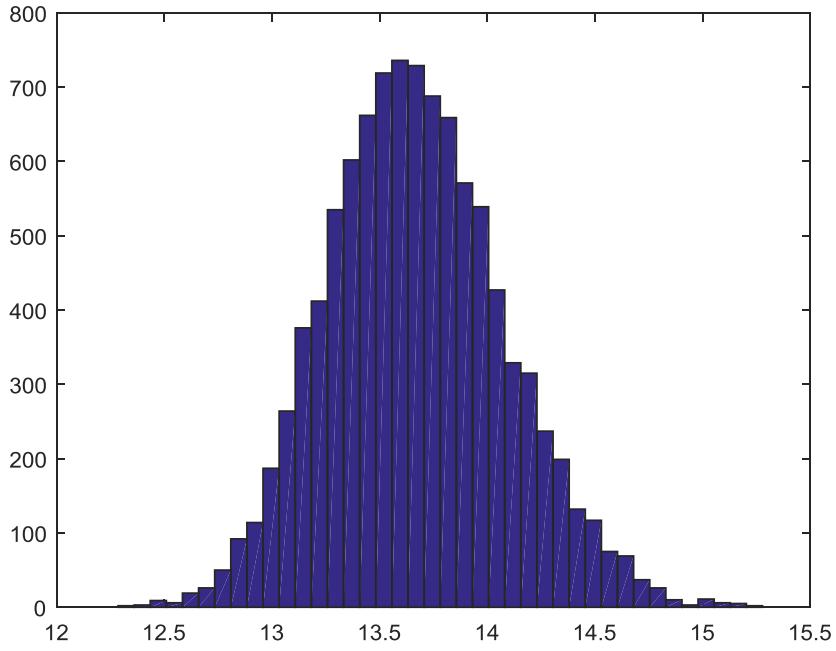
```
X = randn(50, 50, 10000);
S = zeros(50, 50, 10000);

for i=1:10000
    [~, S(:, :, i), ~] = svd(X(:, :, i));
end;

avgSVs = mean(S, 3);
plot(1:50, diag(avgSVs));
```



The scree plot looks mostly linear. The y axis (avg singular values) seems to decrease linearly as x axis (j-th index of singular value) increases.

F.

```
hist(reshape(S(1, 1, :), 1, 10000));
```



The distribution looks slightly lognormal because it is not quite symmetric. To check, I took the log of the distribution, and it looks a bit more normal:

```
hist(log(reshape(S(1, 1, :), 1, 10000)), 40);
```