Student: Elad Sofer - 312124662

HomeWork 2

Reading the image as RGB and convert it into gray-scale since

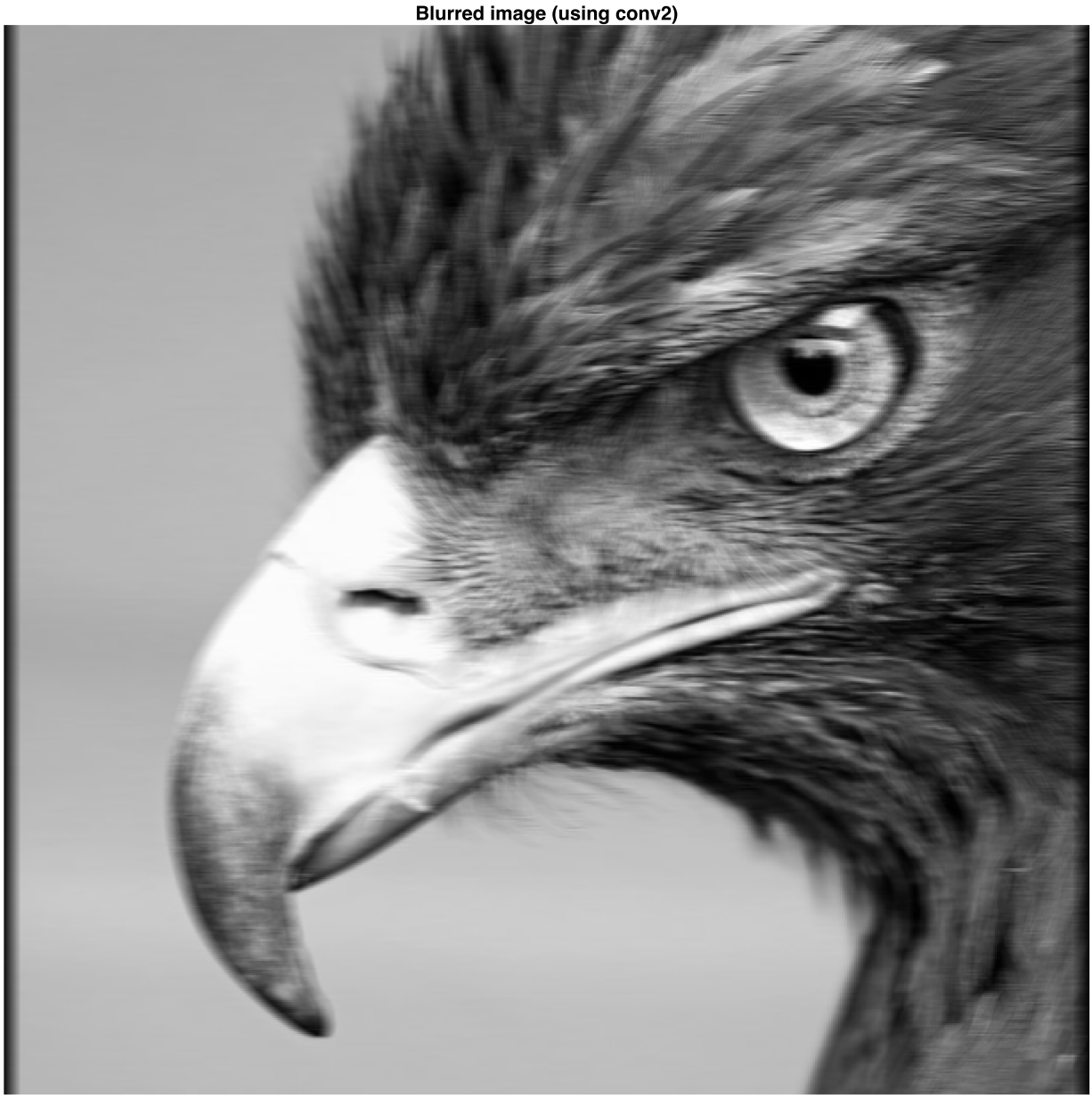
```
% the image
RGB_x = imread('bw.jpeg');

% Turn to gray-scale
x = rgb2gray(RGB_x);

% Turn the picture to NxN for convinience
N = min(size(x));
x = x(1:N, 1:N);

colormap('gray')
imagesc(x);
```

Question 1 section A - Since I perform convolution, with an h_t kernel of dimension Kx1, I get an image BlurX=ht*x in dimension of (N+k-1,N)

Thats why I have black lines in the edges (right & left)

```
% 10 pixel blurring filter
```

```
h = ones(1,10)./10;

% Performing 2D convolution - ht*x
blur_x = conv2(h, x);

% Since blur_x=ht*x it's dimension is (N+k,N+k), while k=10 the following
% appeared
figure(); imshow(blur_x, []); title('Blurred image (using conv2)')
```



To avoid these edges I can perform blurring in the frequency domain with multiplication instead of convolution.

```
Hw = fft2(h, N, N);
Xw = fft2(x);
```

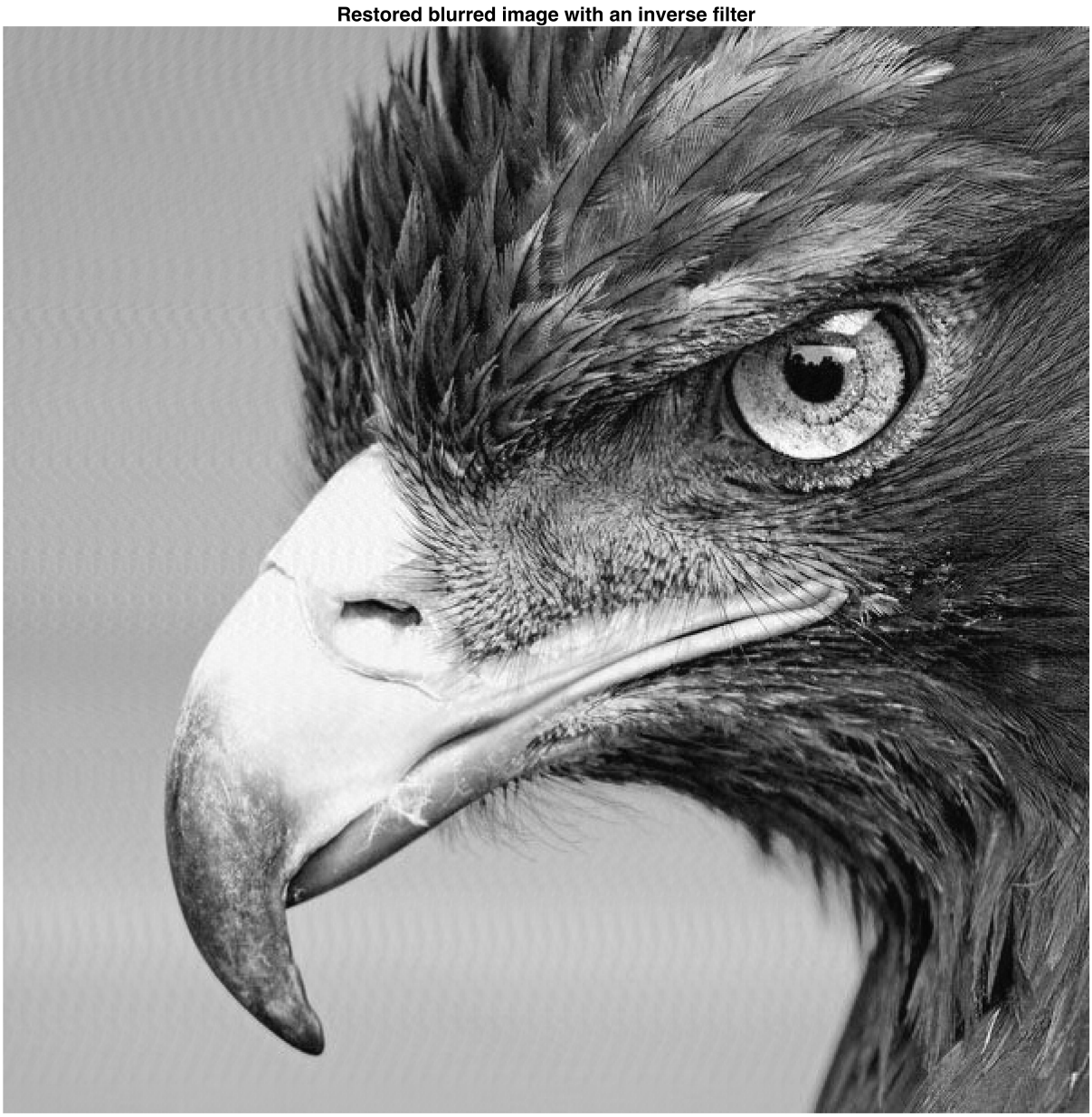
```
blurXw = Hw.*Xw;
blur_x = ifft2(blurXw);

figure(); imshow(blur_x, []); title('Blurred image (freuqncy domain)')
```

Question 1 section B - Since Hw is a sinc function in the frequency domain, some of it's value are equalled to 0.

to create an Inverse filter I need to change these values to a very small value. since 1/0 isn't allowed numerically.

than I reconstructed the image. I can see the restored image is satisfactory using the inverse filter.

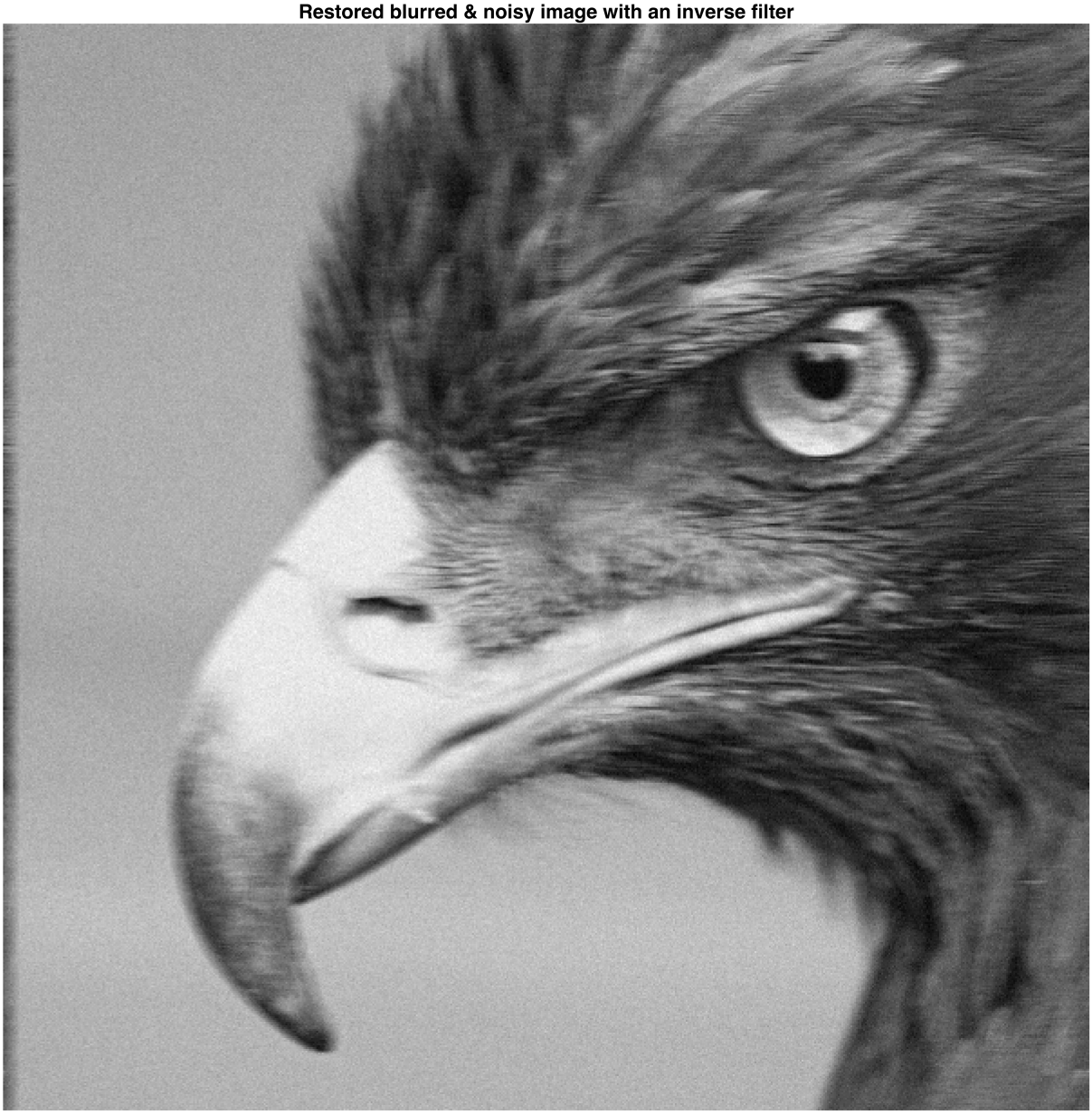
```
% Creating inverse filter
Hw( find(Hw==0) ) = 10^-10;
HwInverse = 1./Hw;

% Reconstruct the image
rec_blurXw = HwInverse.*blurXw;
rec_x = ifft2(rec_blurXw);
figure(); imshow(rec_x, []); title('Restored blurred image with an inverse
filter')
```



Question 1 - Section C - it's easy to see that the noise impaired the restoration of the inverse filter as taught in class.

```
% noise creation
SNR = 20;
img_var = var(double(x(:)));
noise_var = img_var/( 10^(0.1*SNR));
noise =  sqrt(noise_var)*randn(size(x));
% add the noise to the image
blur_noise_x = blur_x + noise;
blur_noise_Xw = fft2(blur_noise_x);
figure(); imshow(blur_noise_x, []); title('Restored blurred & noisy image
with an inverse filter')
```

Restored blurred & noisy image with an inverse filter

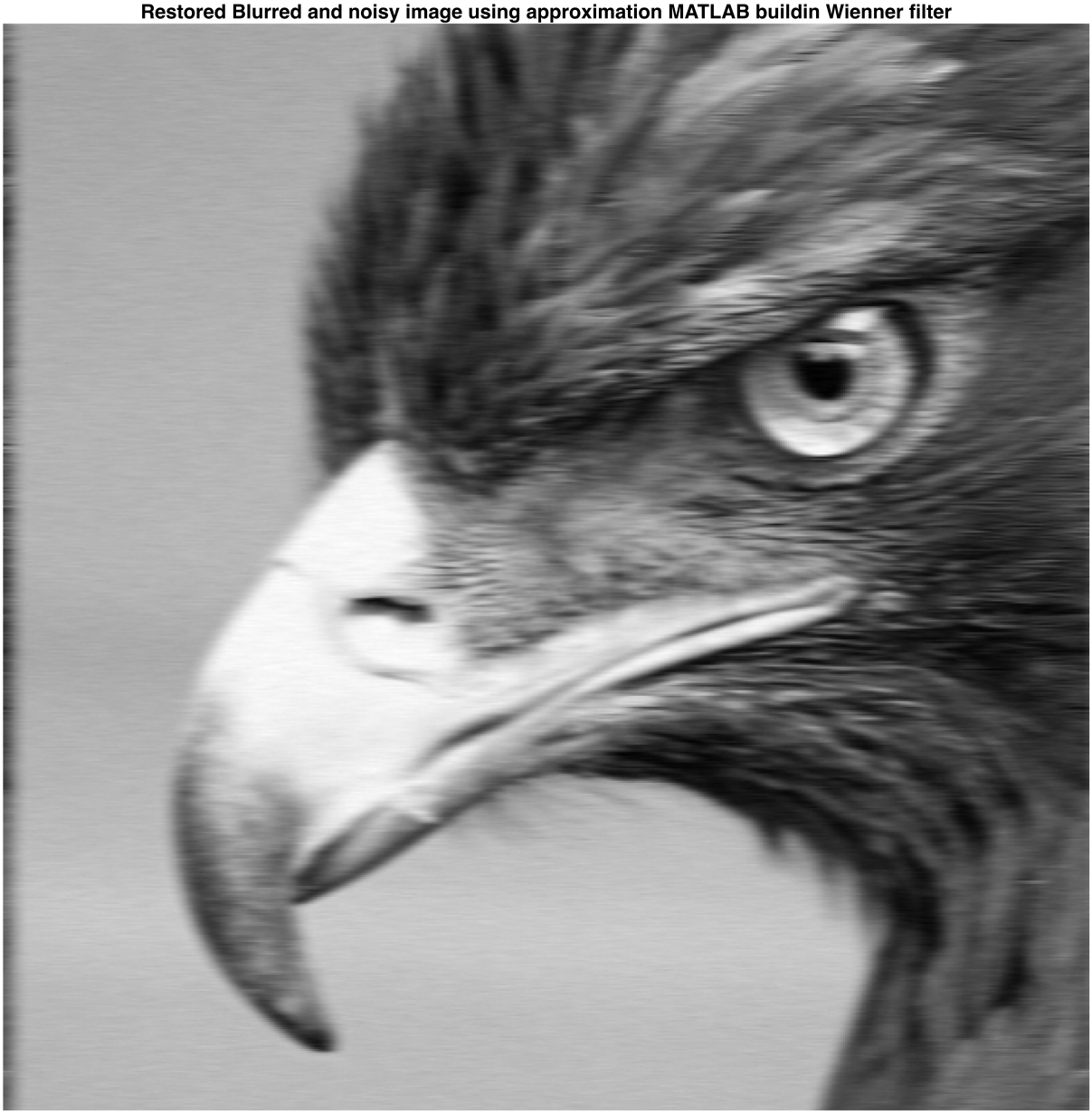


Question 1 - Section D

Firstly I tried to perform the buildin MATLAB wienner filter (which approximates the noise) in order to benchmark my result.

```
wnr1 = deconvwnr(blur_noise_x, h, 20);
figure(); imshow(wnr1, []); title('Restored Blurred and noisy image using
approximation MATLAB buildin Wienner filter')
```

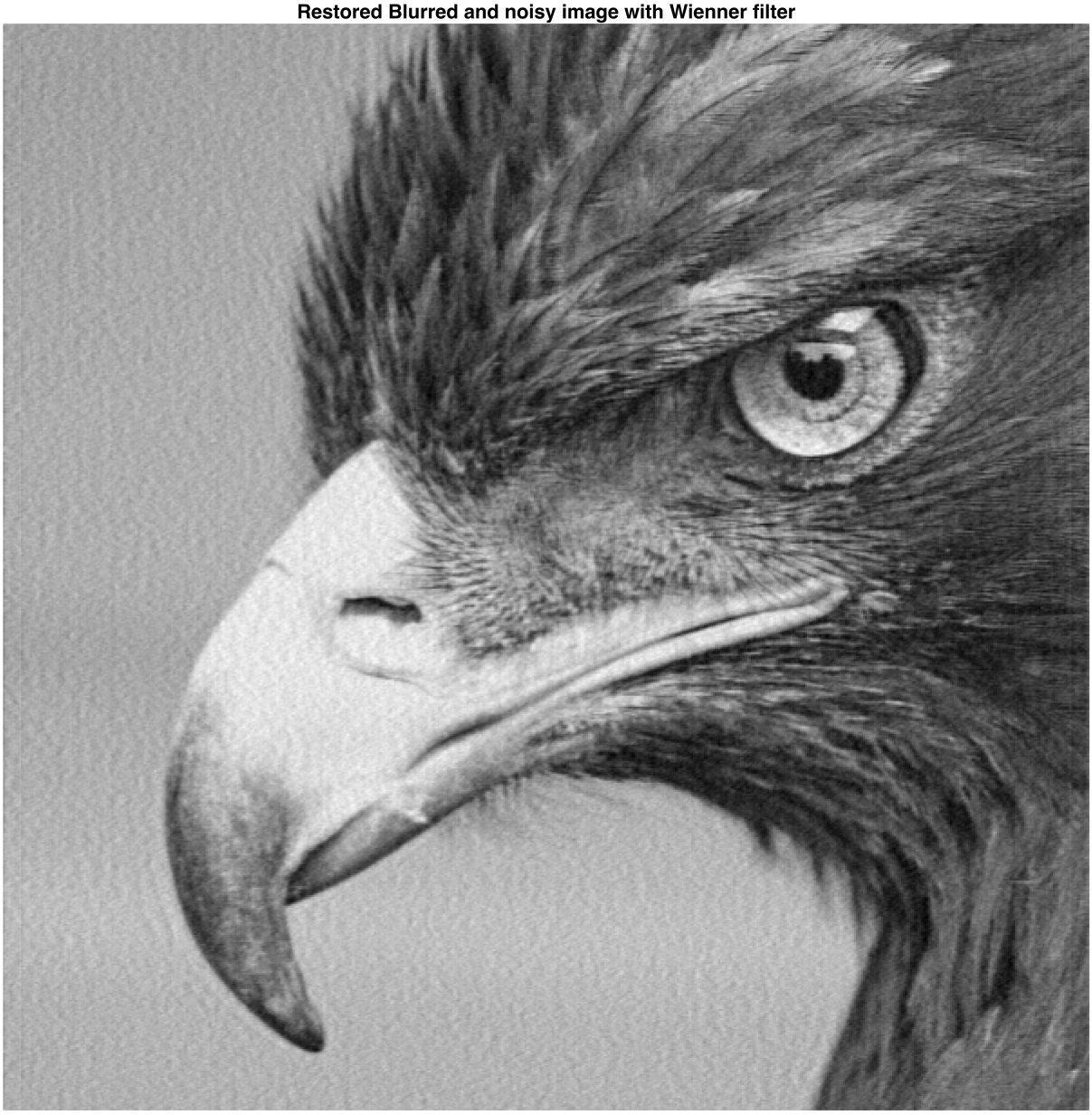
Restored Blurred and noisy image using approximation MATLAB buildin Wienner filter



Here I construct the weinner filter by formula

```
% calculate wiener filter
Su = abs(Xw).^2;
conjH = conj(Hw);
HwAbsSquare = (abs(Hw).^2);
Sn = abs(fft2(noise)).^2;

nom = (conjH.*Su);
den = HwAbsSquare.*Su+Sn;
Gw = nom./den;
Wienner_Xw = blur_noise_Xw.*Gw;
```

```
z = ifft2(Wienner_Xw);

figure(); imshow(z, []); title('Restored Blurred and noisy image with
Wienner filter')
```



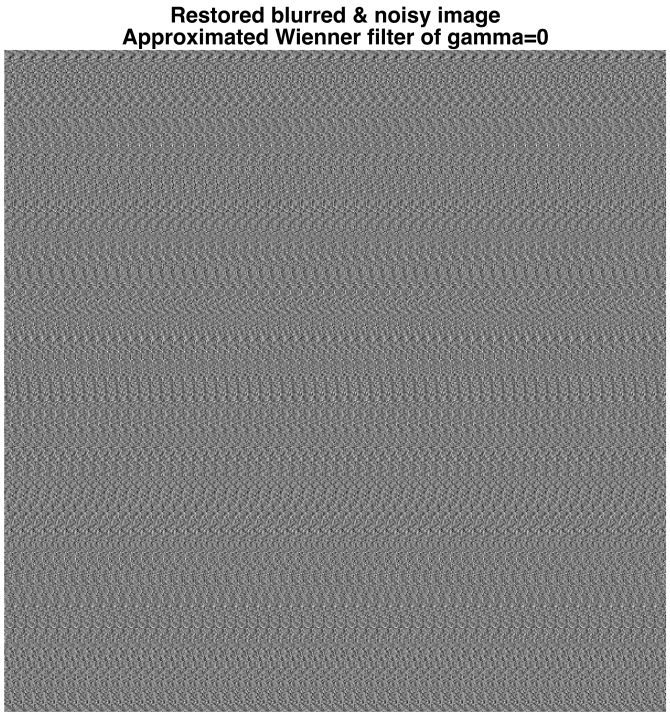
I have performed the reconstruction using various gamma values, and it is evident that the resulting restorations differ depending on the gamma value chosen.

For gamma=0, the restoration is carried out with a filter of 1/H(w1,w2), which disregards the presence of noise entirely. Consequently, the restoration outcome is not satisfactory.

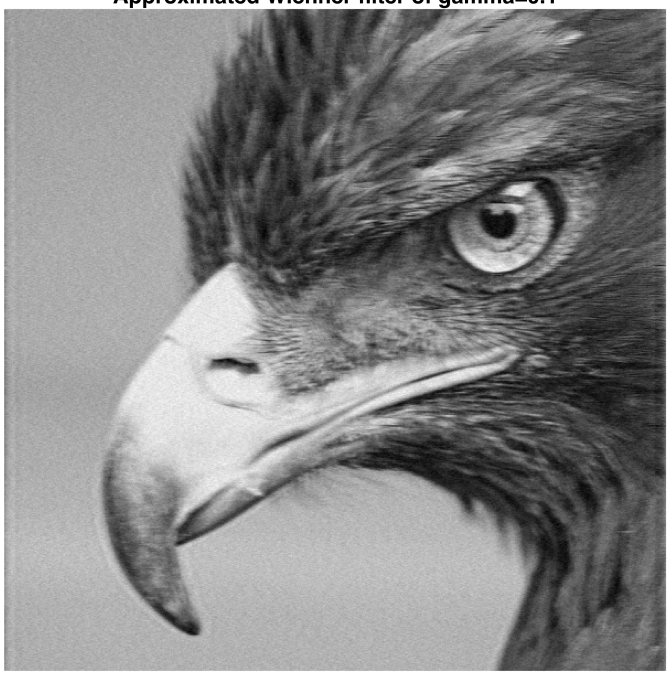
On the other hand, for higher gamma values, the noise is taken into consideration during the restoration process. Specifically, for gamma=10, I attempt to approximate the Wiener filter for higher noise levels than the noise actually present. As a result, the restoration is not as successful.

However, when using gamma=0.1, the restoration quality improves. This is because the noise level Sn is approximately matched to the actual noise amplitude, leading to a better restoration outcome.

```
for gamma = [0, 0.1, 1, 10]
    G = conjH ./ (HwAbsSquare + gamma);
    recWiennerF = blur_noise_Xw.*G;
    recWienner = ifft2(recWiennerF);

    figure(); imshow(recWienner, []); set(gcf, 'Position',  [100, 100, 400,
400])
    title({'Restored blurred & noisy image';['Approximated Wienner filter
of gamma=', num2str(gamma)]})
end
```



Restored blurred & noisy image
Approximated Wienner filter of gamma=0.1



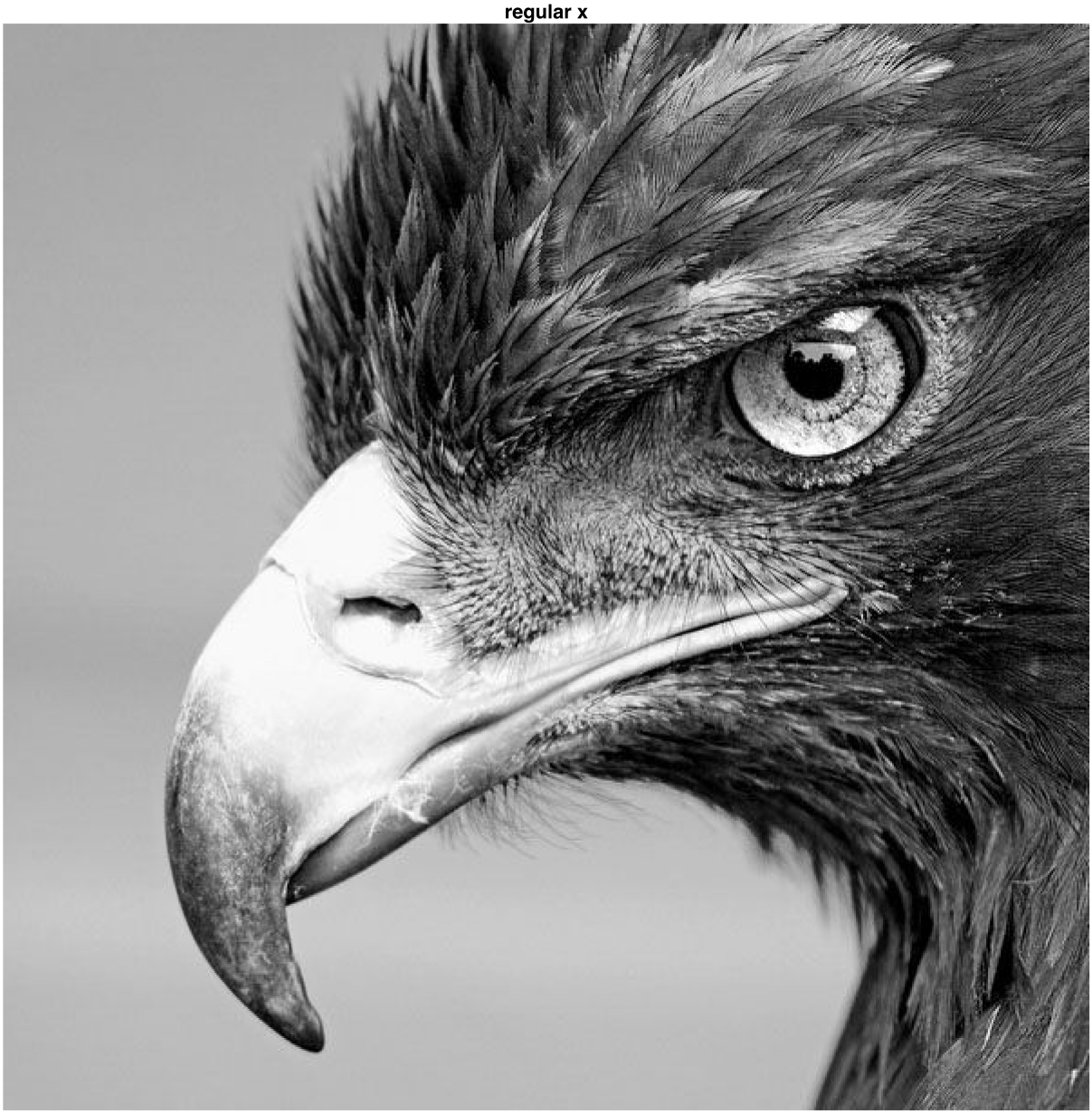
Restored blurred & noisy image
Approximated Wienner filter of gamma=1

Restored blurred & noisy image
Approximated Wienner filter of gamma=10

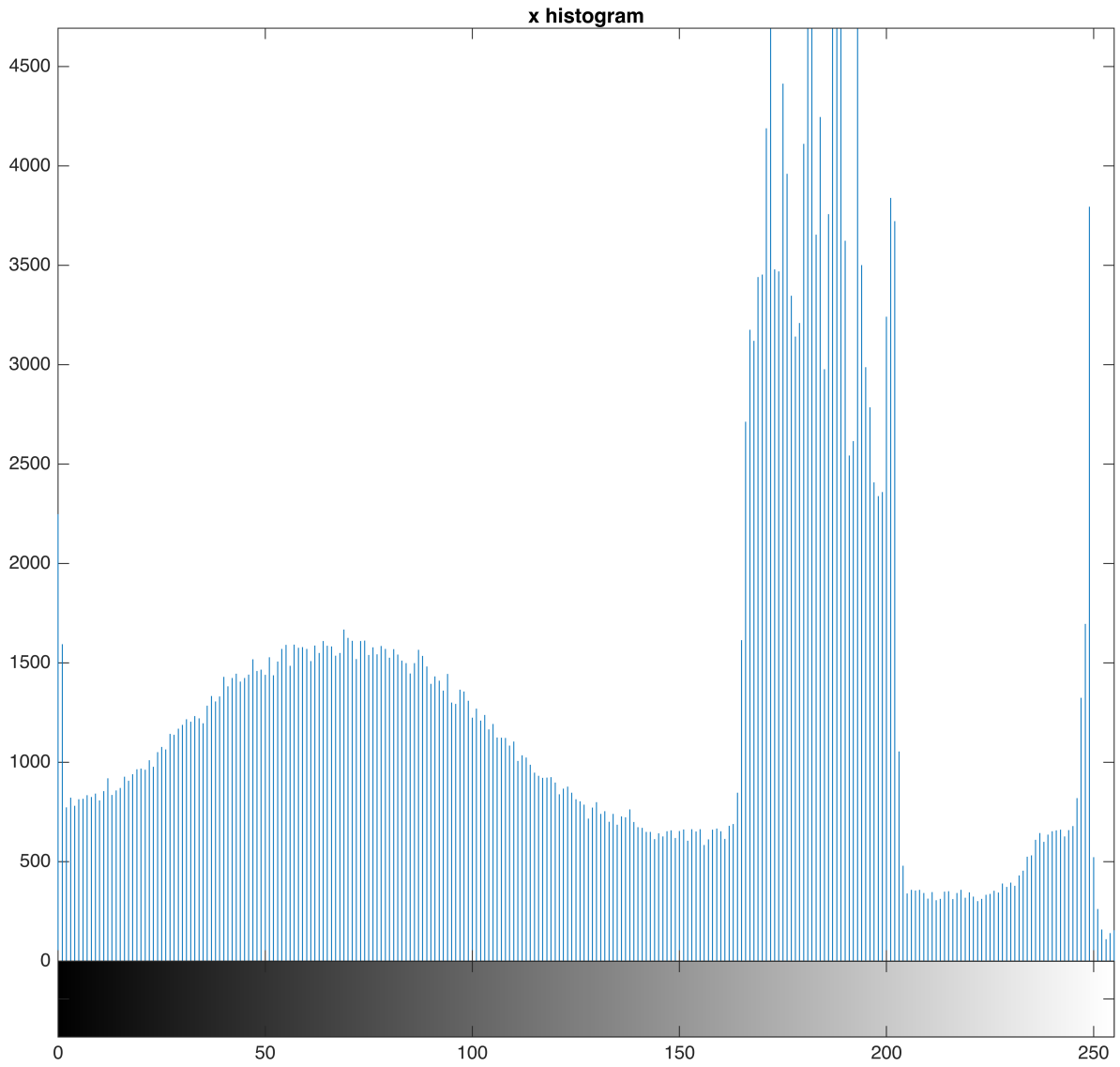
Question 2

```
figure
eqX = histeq(x);
figure(); imshow(x, []); title('regular x')
```

regular x



```
imhist(x); title('x histogram');
```



In this picture I can see certain levels which was pretty much "black" let's say color range 0-50 was pretty much black, but after the equalization

I can see much more levels in these areas. one evidence is the eagle's mouth that was black and now it has more constrast. Another example is the background alternation.

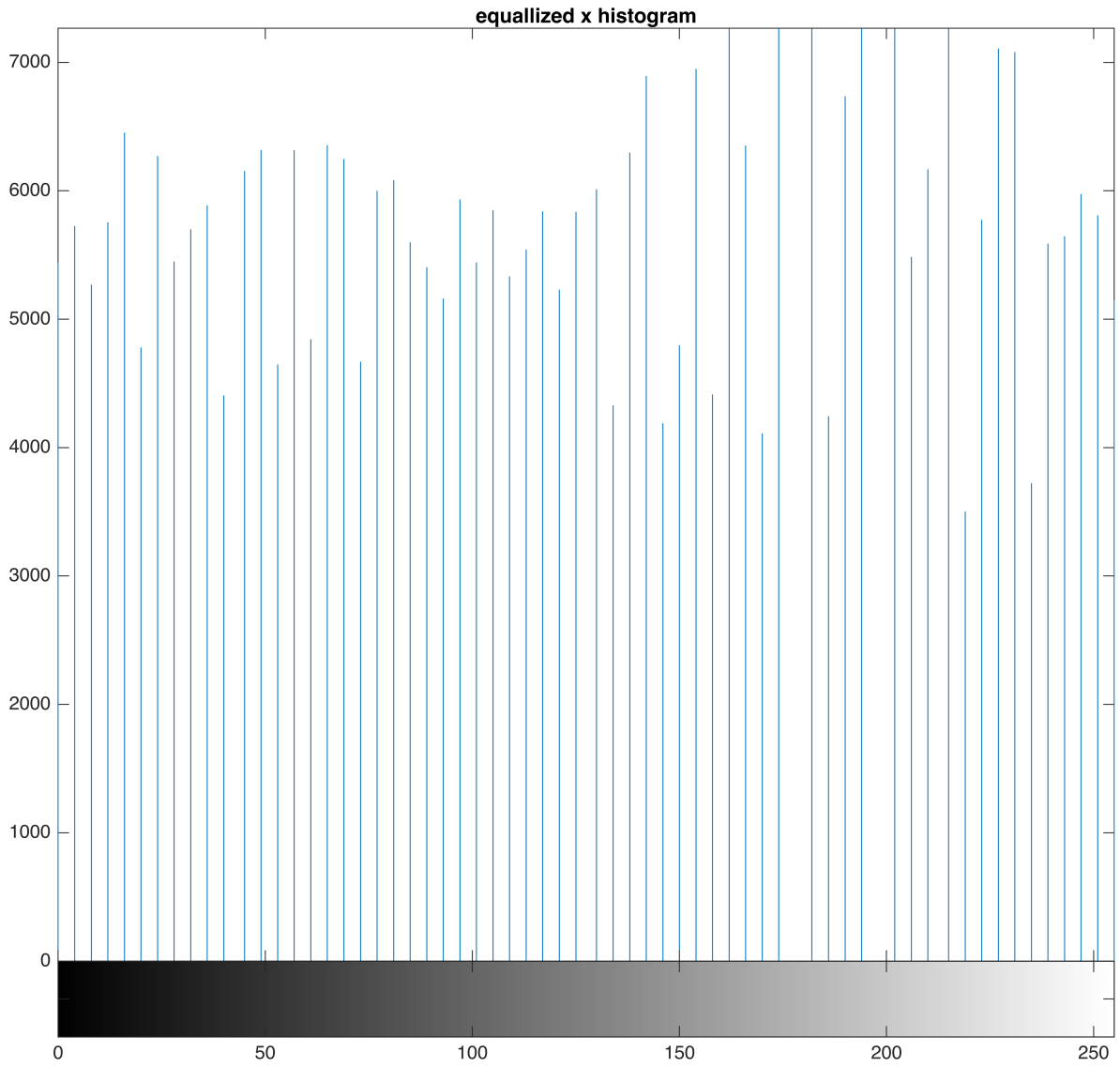
I can see more clearly patterns in the background in the equallized image comparing to the original image.

Overall, I can see the histogram after the equllization is much more balanced.

```
figure(); imshow(eqX, []); title('equallized x')
```

equallized x

```
imhist(eqX); title('equallized x histogram');
```

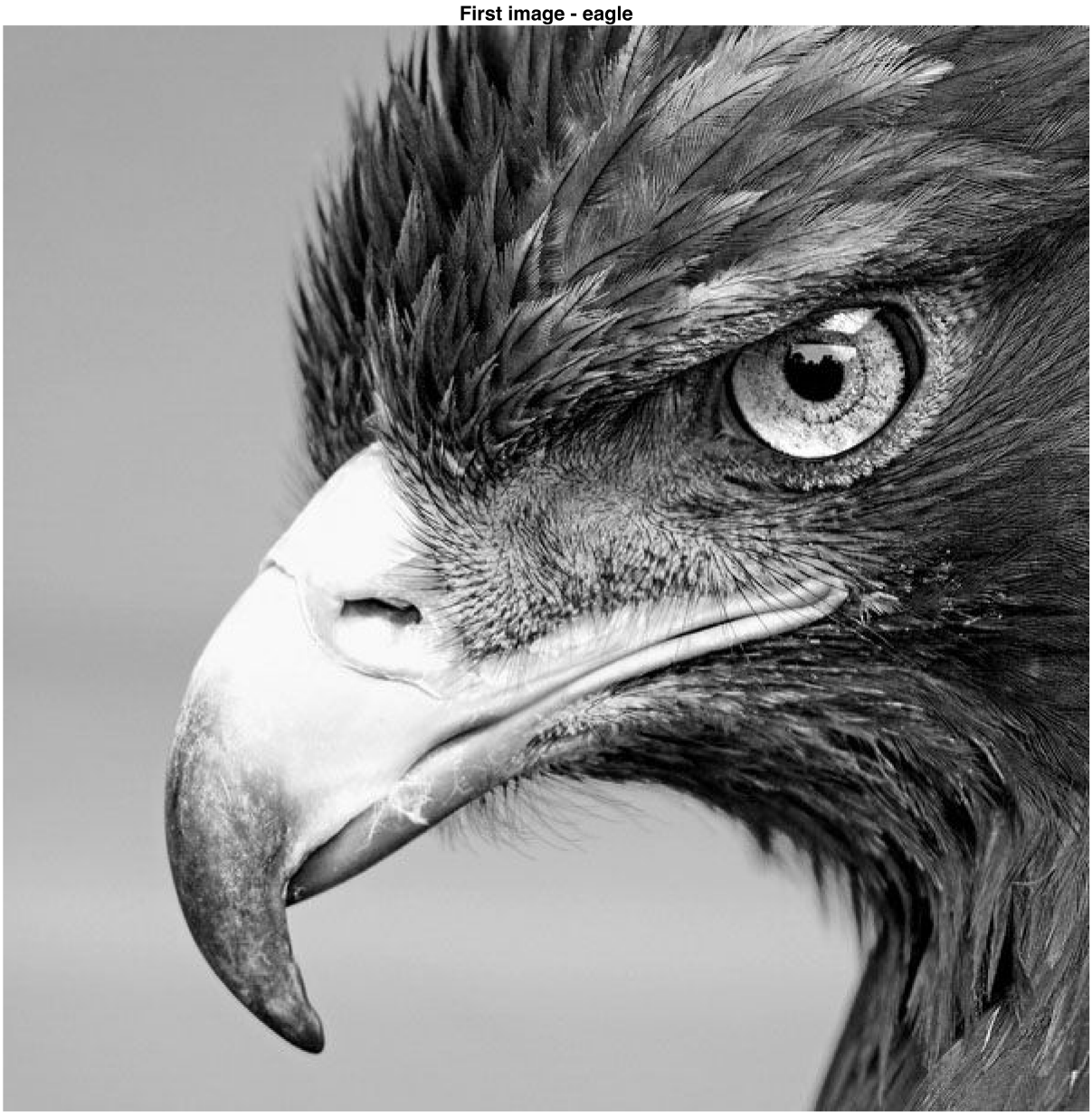


Question 3 - loading firstly the second image -

Reading the image as RGB and convert it into gray-scale since

```
% Load the second image
RGB_y = imread('bw2.jpeg');

% Turn to gray-scale
y = rgb2gray(RGB_y);

% Turn the picture to NxN for convinience
```

```
M = min(size(y));
y = y(1:M, 1:M);

figure(); imshow(x); title('First image - eagle')
```




```
figure(); imshow(y); title('Second image - women')
```

Second image - women

It appears that the phase component of the image preserved the structural details such as lines and edges, while the amplitude component retained information about the color intensity. For instance, in areas where the eagle's mouth appears black, the corresponding image regions are also black, and where the eagle image is whiter, the image regions are also whiter. However, it is worth noting that all the information regarding edges and lines is captured primarily in the phase of the women's image.

Explanation:

The phase of an image contains valuable information because it encodes the spatial frequency content of the image.

Spatial frequencies in an image refer to the rate at which intensity values change across different regions. Higher spatial frequencies correspond to rapid intensity changes, like edges or fine details, while lower spatial frequencies represent smoother areas or gradual transitions.

```
XwAmp = abs(fft2(x));
YwPhase = angle(fft2(y));

newImageF = XwAmp.*exp(1i*YwPhase); %construct the new image

newImageX = ifft2(newImageF);
figure(); imshow(newImageX); title('Amplitude: Eagle, phase: Women')
```

```
Warning: Displaying real part of complex input.
```