MÜHENDİSLİK FAKÜLTESİ
FACULTY OF ENGINEERING
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ
DEPARTMENT OF ELECTRICAL AND
ELECTRONICS ENGINEERING

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 23 02
F: +90 312 210 23 04
ee@metu.edu.tr
www.eee.metu.edu.tr

EXPERIMENT 7. IMAGE PROCESSING, 2D FFT, FILTERING, EDGE DETECTION
PART 1
LABORATORY REPORT

Student 1: Abdullah Canbolat, 2304244

Student 2: Güray Özgür, 2167054

Student 3: Yunus Bilge Kurt, 2232395

MATLAB Programming Tasks

a) Write a ***"expriment7.m"*** file to load two images,'***cameraman.tif'*** and ***'lena.png'***. Use '***imshow'*** to display the images.
Attach the plots to your report.

Figure 1: The "cameraman" image

Figure 2: The ”lena” image

b) Take the 2D FFT of the cameraman in order to visualize the low and high frequency components. Use ***'fftshift'*** to center the DC component.

- First, use **“mesh”** command to observe 2D FFT magnitude transform. Note that DC component is extremely high. In order to plot magnitude transform using **“imshow”** as a grayscale image, you need to scale the magnitude transform.

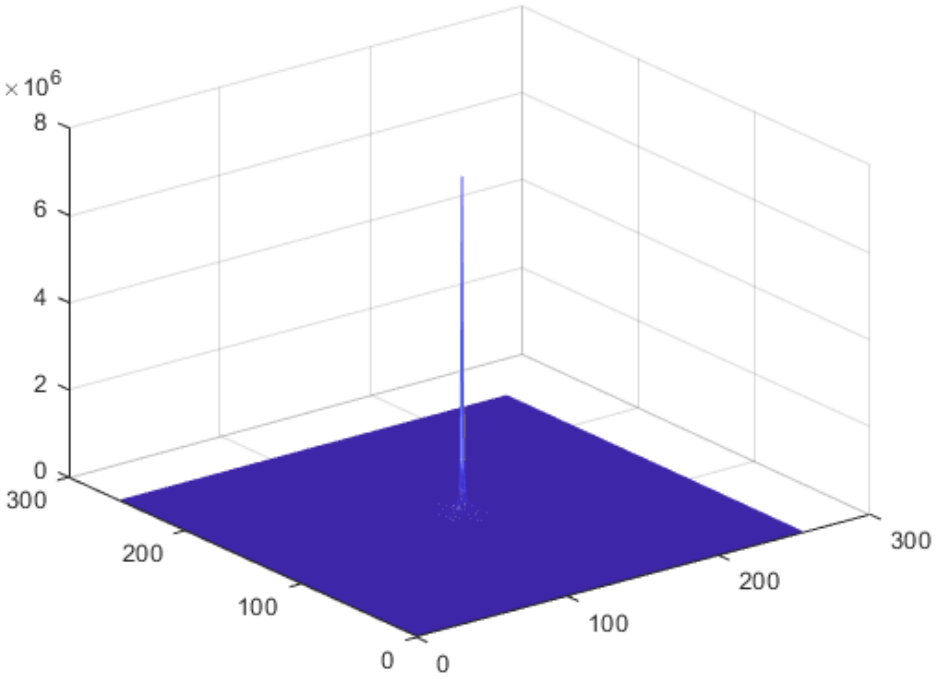

Figure 3a: 2D FFT of cameraman

- Divide all the components of 2D FFT magnitude transform by the maximum element to normalize the magnitude transform. Then, plot the magnitude transform using **"imshow". Attach the magnitude of 2D transform to your report.**

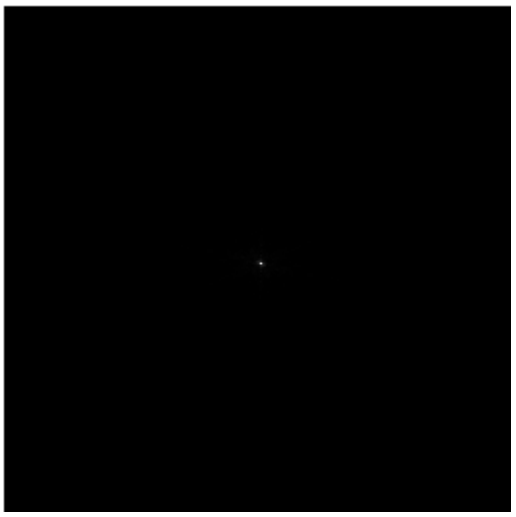

Figure 3b: Magnitude of 2D FFT of cameraman

- Observe that scaling by the maximum element is not an effective approach in order to observe frequency distribution of the cameraman image. Now, scale the 2D FFT magnitude transform by 1/10000 and plot the magnitude transform using **"imshow". Attach the magnitude of 2D transform to your report.**

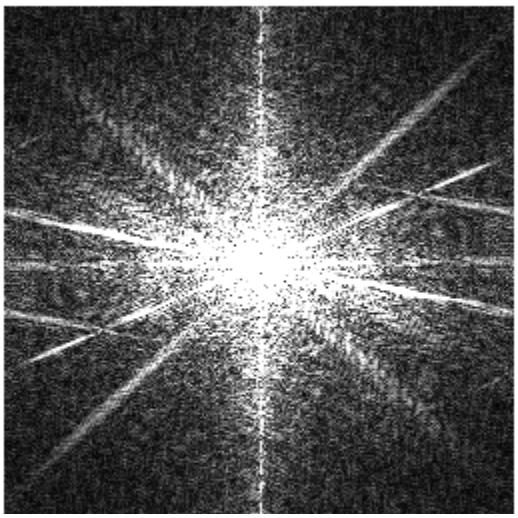

Figure 4: Magnitude of 2D FFT scaled by 1/10000 of cameraman

c) Show that 2D fft, ***'fft2'***, is equivalent to applying ***'fft'*** to rows and columns respectively.

- For this purpose, consider the following matrix,

```
X= [  1    2    3    4;
      5    6    7    8;
      9   10   11   12];
```

- Take 2D FFT of X and denote it by Y. **Write the elements of Y in matrix form.**

Y =

```
[78.0000 + 0.0000i  -6.0000 + 6.0000i  -6.0000 + 0.0000i  -6.0000 - 6.0000i;
-24.0000 +13.8564i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i;
-24.0000 -13.8564i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i]
```

- Now, apply fft to the rows of X and denote the result by Z. **Write the elements of Z in matrix form.**

Z =

```
[10.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i;
 26.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i;
 42.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i]
```

- Now, apply fft to the columns of Z and denote the result by T. **Write the elements of T in matrix form.**

T =

```
[ 78.0000 + 0.0000i  -6.0000 + 6.0000i  -6.0000 + 0.0000i  -6.0000 - 6.0000i;
-24.0000 +13.8564i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i;
-24.0000 -13.8564i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i]
```

d) Obtain the phase components of 2D fft of ***cameraman***. Assume that the magnitude components are **one** and use ***ifft2*** to reconstruct the image by using only the phase information as shown in Fig. 5. Selecting magnitudes **randomly** may result in a **complex image after ifft2**. In this case, show the **real part of the image** in order to preserve the phase components of the original cameraman image. **Attach the reconstructed image**. ***Comment on*** your findings.

When we look at Figures 5-7, we see that the shape of the image is decoded by phase characteristics. Magnitude response of the image helps to add color details of the image, i.e. power of the image on a certain frequency is given by the magnitude response.

- In order to bring the resultant image into the range 0-1, you can use ***"mat2gray"*** command before **"imshow",** i.e., **imshow(mat2gray()).**

Figure 5: Reconstruction of cameraman by using only the phase where the magnitude components are one

- Try alternatives for the magnitude terms during the reconstruction. One alternative is to use random numbers, i.e. ***rand()***. Another alternative is to use the 2D FFT magnitude of ***lena*** image. **Attach the reconstructed image for each case**. Which one gives ***better reconstruction***? **Comment on** the results.

We have tried different alternatives other than ones, which are random values and values taken from the magnitude of the "lena" image. Since "lena" is also an image, it gave a natural look to the "cameraman" image, while random values did not contribute to the texture of the image. Thus, it can be said that Figure 7 is a better construction of the "cameraman" image than Figure 6. It can also be seen that high frequency components are having a larger amplitude in the images constructed with ones and rand, which is not the case in our original image.

Figure 6: Reconstruction of cameraman by using only the phase where the magnitude components are random

- Note that size of ***lena*** is **512x512**. Hence, you should resize it according to the ***cameraman***. You can use **"imresize"** function for this purpose.

Figure 7 :Reconstruction of cameraman by using only the phase where the magnitude components are of "lena"

e) Consider the following ***9x9 nonlinear phase lowpass kernel***.

```
h1=1/4248*[92   35   75   96   22   39   70   30   81;
           82   40   38   34  100   49   36    3    3;
           16   66   60   71   57   88   80    8   21;
           82   37   15   93   81   90   98   17   91;
           82    9    4   41   52   92   68   34   82;
           49   37   56   54   84    6   62   98   50;
           82    9   28   88    9   36   73   83   40;
            2   58    2   40   48   38   54   79   19;
           80   77    6   54   58   50   35   60   84;];
```

Design a ***9x9 separable lowpass linear phase filter***, h2=(ha'*hb)/sum(sum(ha'*hb)). Let

```
ha=[1 2 3 4 5 4 3 2 1];
hb=[1 1 2 2 3 2 2 1 1];
```

Plot the ***magnitude*** and ***phase*** characteristics of these filters as shown in Fig. 6 using ***"mesh"*** function. **Attach the plots to your report**. Filter the ***cameraman*** and ***lena*** images by these filters and ***comment on*** the phase distortions introduced by the nonlinear phase filter. Use **"double"** command before **filter2** command in order to convert the uint8 data type of the loaded images to double.

Note that you may need appropriate scaling for better visualization (You can use ***"mat2gray"*** command).

Attach all the filtered images to your report.

When we compare Figure 16 with 17 and Figure 18 with 19 we see that, the image filtered with linear phase filter is more clear whereas the image with nonlinear phase filter is distorted and is more blurry. This is due to phase characteristics of the filters shown in Figures 8-15, when we look at the nonlinear phase filter we see that the phase information will be lost in the resulting image because linearity in phase results in a shift in spatial dimension, however, if we look at the Figure 10, the filter does not provide equal shifts for each frequency rather it shifts each frequency differently. Therefore, we observe distorted and blurry images if we filter with nonlinear phase filters. Also, from Part d, we know that image shape is determined by phase characteristics, therefore we expect that shapes are distorted in nonlinear phase filters.

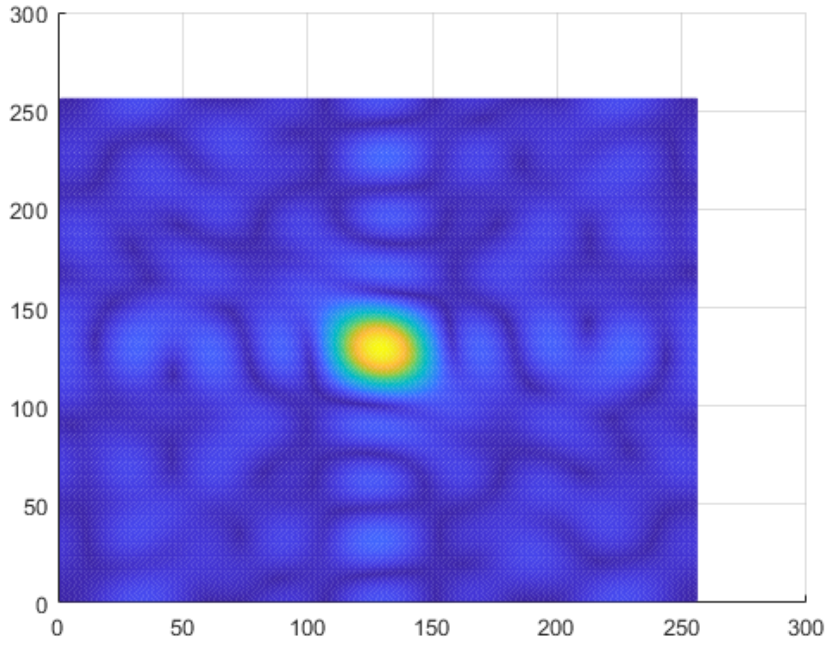

Figure 8: Magnitude characteristics in 2D of the filter "h1"

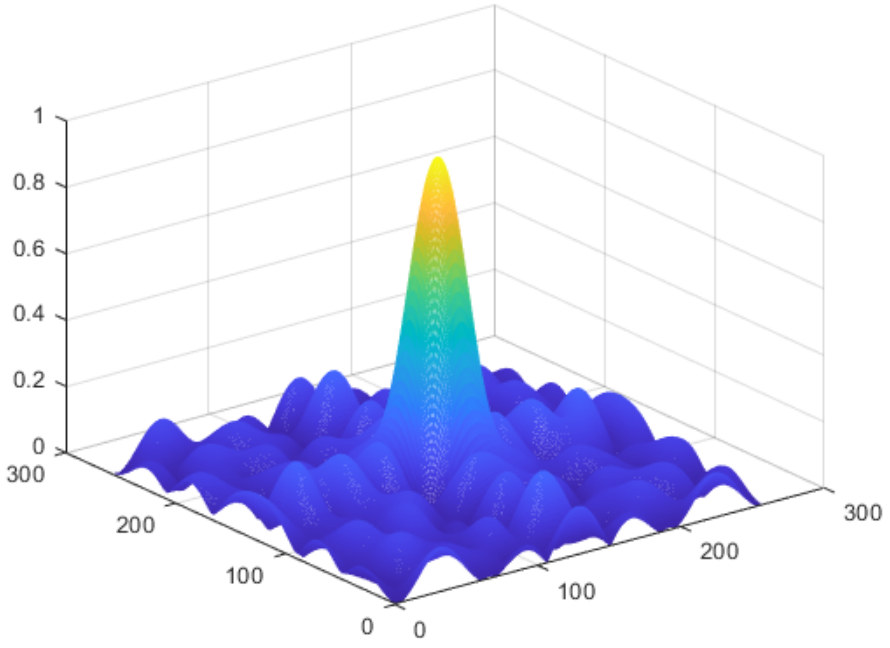

Figure 9: Magnitude characteristics in 3D of the filter "h1"

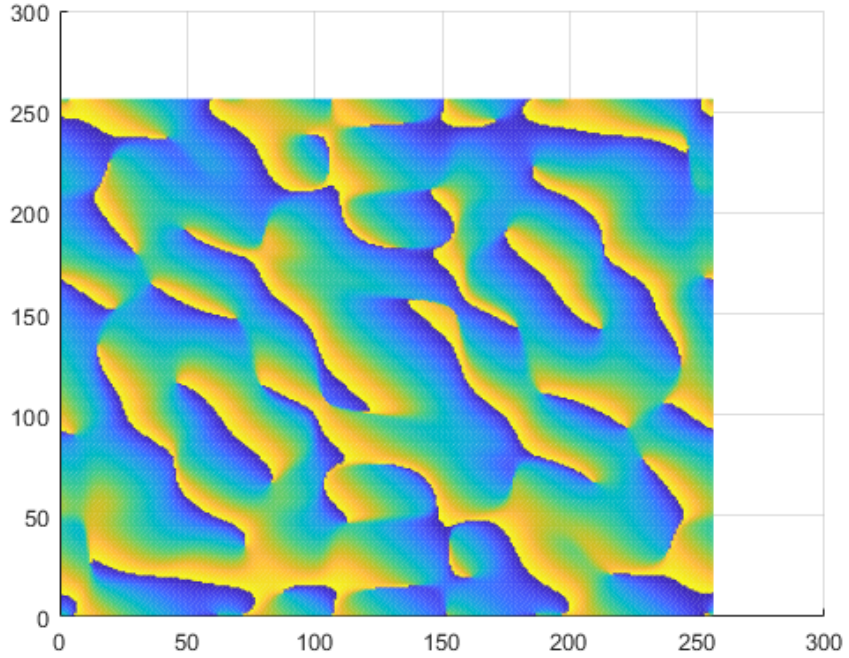

Figure 10: Phase characteristics in 2D of the filter "h1"

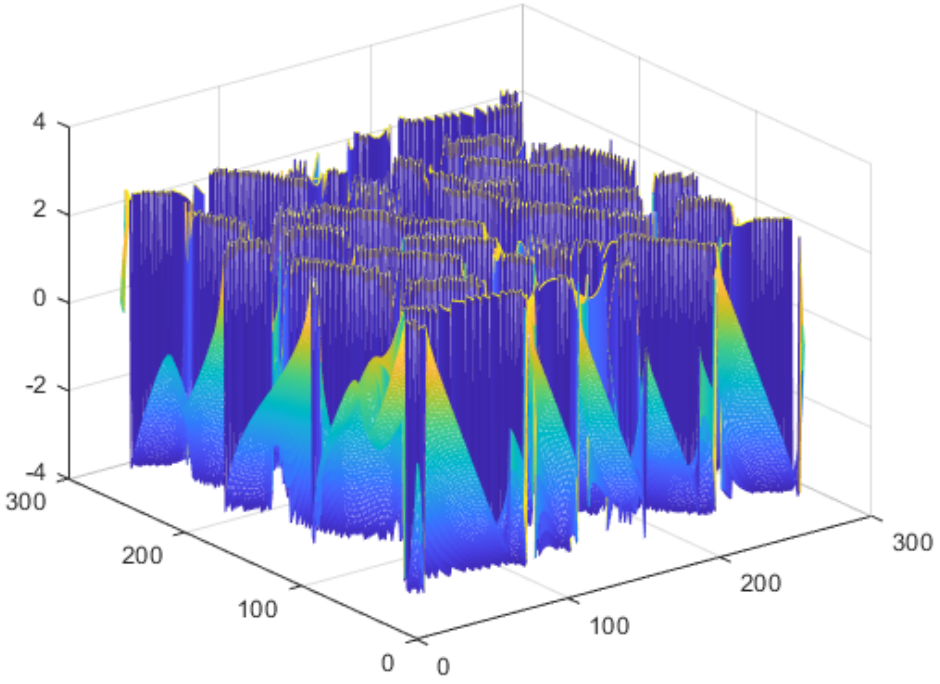

Figure 11: Phase characteristics in 3D of the filter "h1"

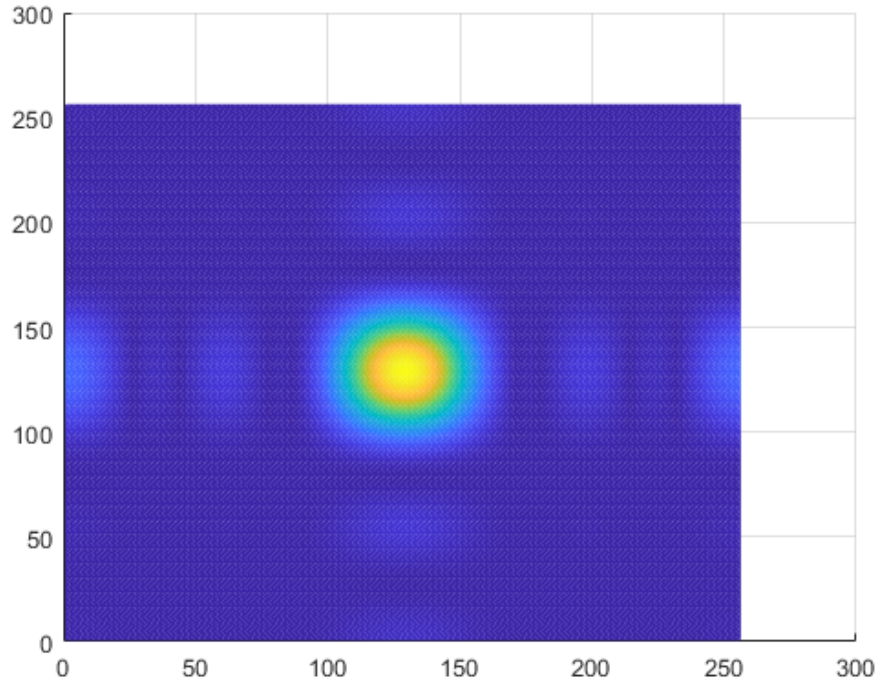

Figure 12: Magnitude characteristics in 2D of the filter "h2"

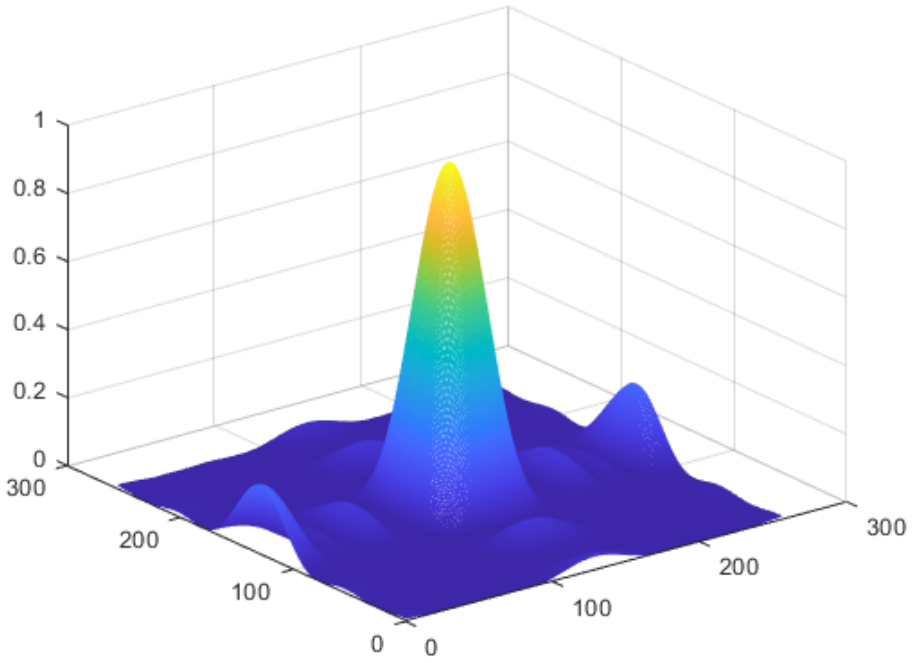

Figure 13: Magnitude characteristics in 3D of the filter "h2"

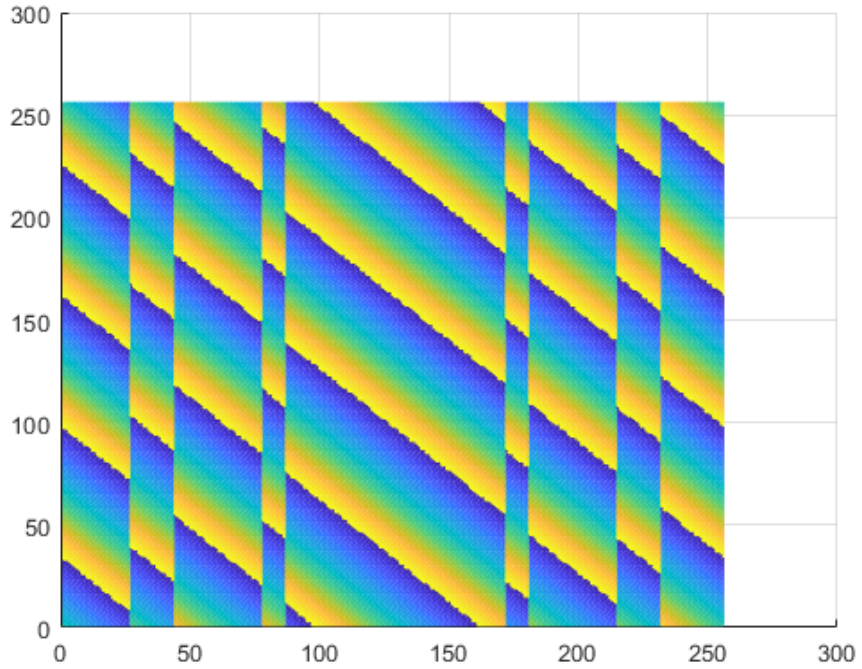

Figure 14: Phase characteristics in 2D of the filter "h2"

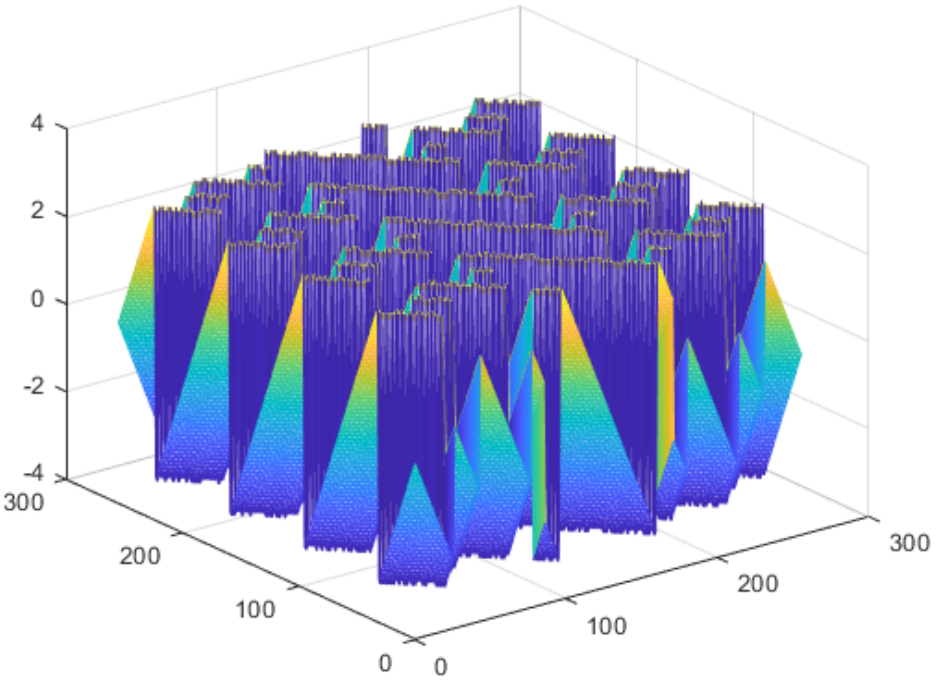

Figure 15: Phase characteristics in 3D of the filter "h2"

Figure 16: Filtered “cameraman” by the filter “h1”

Figure 17: Filtered “cameraman” by the filter “h2”

Figure 18: Filtered “lena” by the filter “h1”

Figure 19: Filtered “lena” by the filter “h2”

f) Consider the filter, ***h2***, in ***e)***. Filter the ***lena*** image with this filter and obtain the output ***y(n_1,n_2)*** as shown in Fig. 7 by adding Gaussian random noise with a known variance. Take the **standard deviation as 1**. Display the resulting image with ***imshow*** command. Design the inverse filter as it is given in equation (7). You can take the **1/SNR as constant at 1**. Filter the output, y(n_1,n_2) with the inverse filter. Display the result of the inverse filtering and ***comment on*** the deconvolution performance. **Attach both noise added and reconstructed images.** ***Determine and write*** the ***least squares error*** in reconstruction. Use **mat2gray** while calculating LSE.

- Try different noise levels (take **standard deviation as 10 and 100**) and **comment** on the reconstruction performance. You can change 1/SNR constant for better reconstruction for different noise levels. ***Determine and write*** the ***least squares error*** in reconstruction for each noise level.

When we compare the LSE results from Table 1 and Table 2, we see that the 2nd filter gives lower LSE values, thus inverse filtering quality is better for the 2nd filter. As to be explained from pole-zero plots, the zeros on the unit circle of the 1st filter eliminates some frequency components, which leads to information loss, and not a good reconstruction.
Furthermore, in some situations, LSE does not give meaningful results, thus, Structural similarity (SSIM) index is used measuring image quality.

In addition, it is clearly seen that when the noise increases, LSE also increases as expected in both situations.

Table 1: Least Squares Error in reconstruction for each noise level

Standard Deviation	**1**	**10**	**100**
LSE	**0.0128**	**0.0132**	**0.0193**

Figure 20: Filtered noisy “lena” image with the filter “h2”, where std=1

Figure 21: Reconstruction of Figure 20

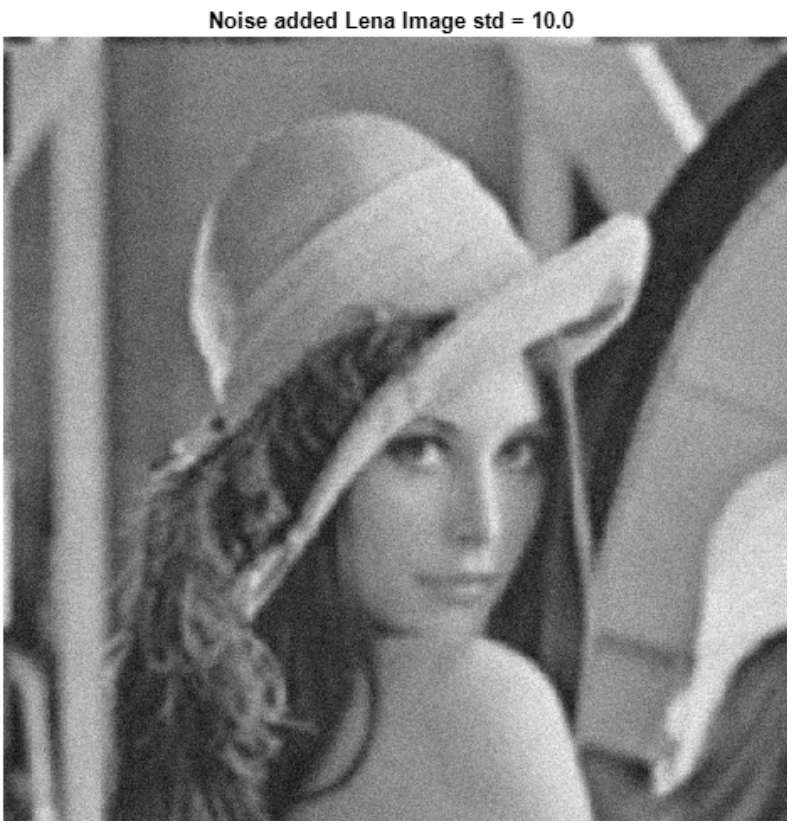

Figure 22: Filtered noisy "lena" image with the filter "h2", where std=10

Figure 23: Reconstruction of Figure 22

Figure 24: Filtered noisy "lena" image with the filter "h2", where std=100

Figure 25: Reconstruction of Figure 24

- **Change the h2 filter (by changing ha and hb)** as follows,

```
ha_new=[1.0000  7.9139  27.5038  54.7921  68.4044  54.7921  27.5038  7.9139
1.0000];
hb_new=ha_new;
h2_new=(ha_new'*hb_new)/sum(sum(ha_new'*hb_new)).
```

- **Repeat the previous parts** (take noise standard deviation as 1, 10 and 100) and ***comment on*** the inverse filtering quality. Don't **forget to attach the images and LSE values for each noise level.**

Table 2: Least Squares Error in reconstruction for each noise level

Standard Deviation	1	10	100
LSE	0.0156	0.0161	0.0272

Figure 26: Reconstruction of Figure 20 with new the filter

Figure 27: Reconstruction of Figure 22 with new the filter

Figure 28: Reconstruction of Figure 24 with new the filter

- Find the zeros of ha and hb using the **“roots”** command. Find also zeros of ha_new. **What is the effect of zeros on the reconstruction quality based on your observations? Comment on** the results.

The zeros of the filters "ha" and "hb" are on the unit circle. For the filter "ha_new", the number of zeros on the unit circle is only two and they are at the same frequency. The zeros on the unit circle result in loss in that frequency. So, we have lost information from 12 frequencies in the first case and only 1 frequency in the second case. Thus, the filter using "ha_new" is better at reconstruction.

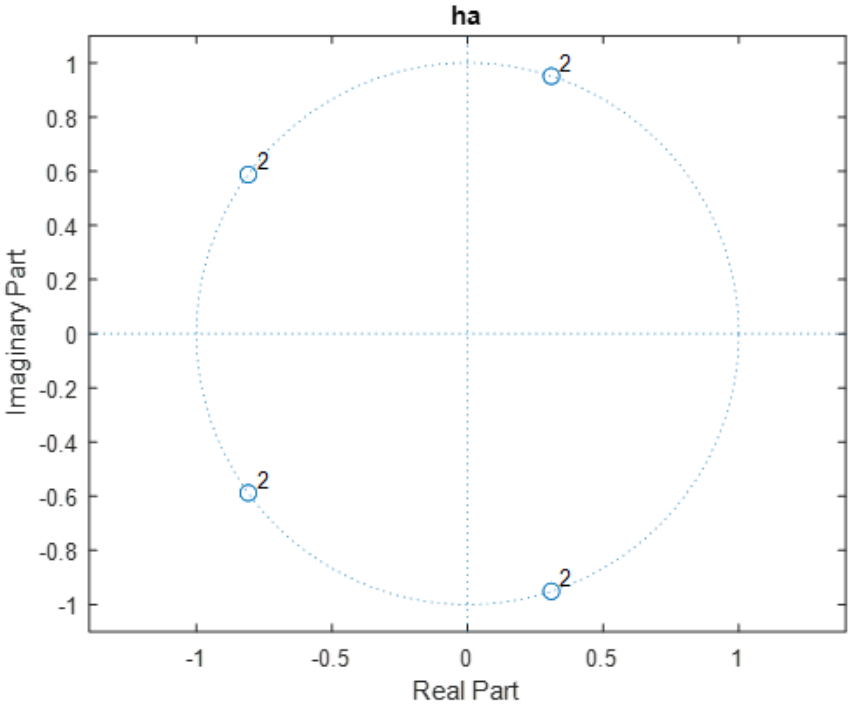

Figure 29: Roots of the filter “ha”

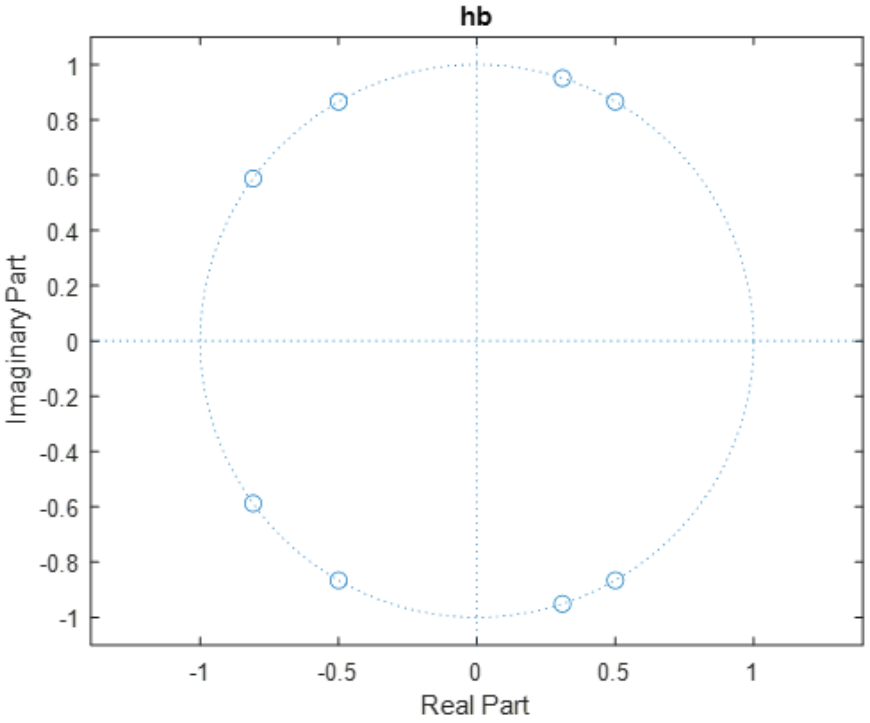

Figure 30: Roots of the filter "hb"

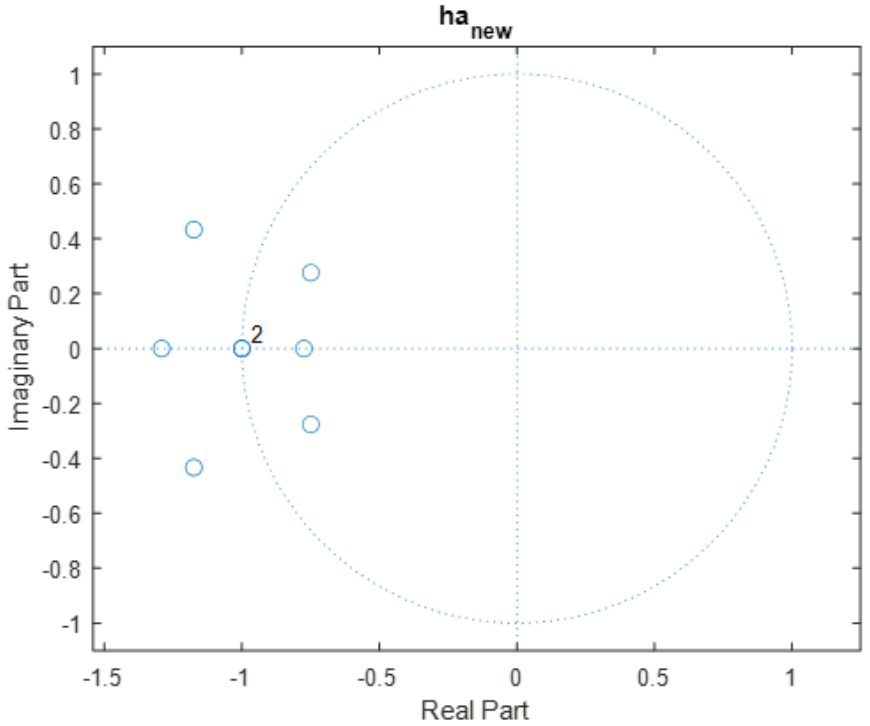

Figure 31: Roots of the filter "ha_new" and "hb_new"