Experimental Assignment 1
Medical Imaging Techniques
(EECS4640/5640)

Professor Sadeghi-Naini

SeyedMostafa Ahmadi
(#219138924)
February 2023

Consideration

After getting professor Sadeghi's approval, I programmed this assignment using Python (3.6) instead of Matlab.
You can find all my codes in the directory "*codes*" for each question in the following format: q1.py for the first question, q2.py for the second question, etc.
To run the code for each item in a specific question, please uncomment it under the line :

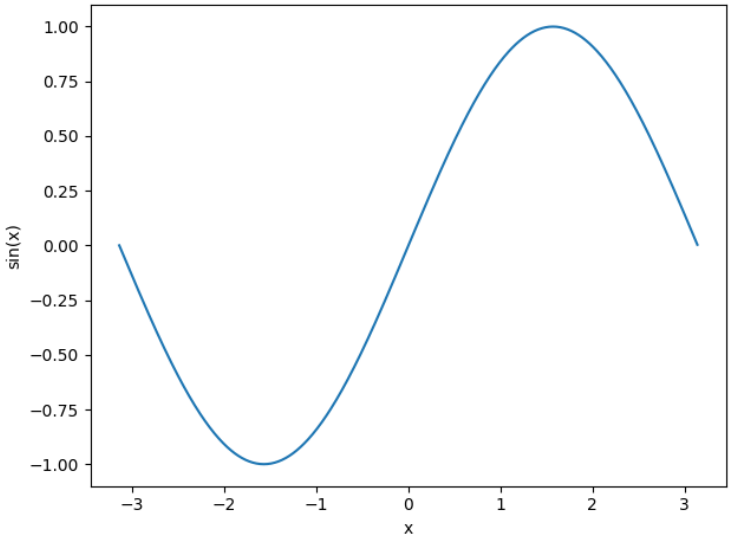
```
if __name__ == "__main__":
```

in the corresponding file. For example, for running the code for ***question 2 item b***, i.e. ***2-b)***, please uncomment the following line in file ***q2.py***:

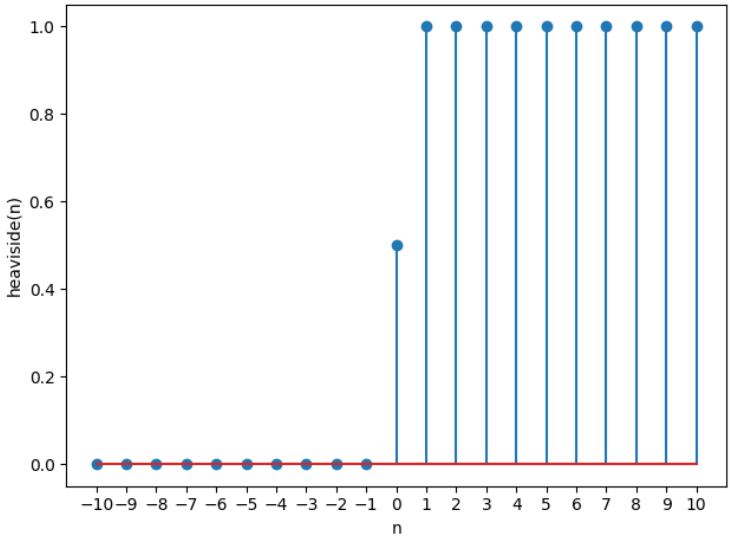
```
# item_b()
```

Question 1

a)

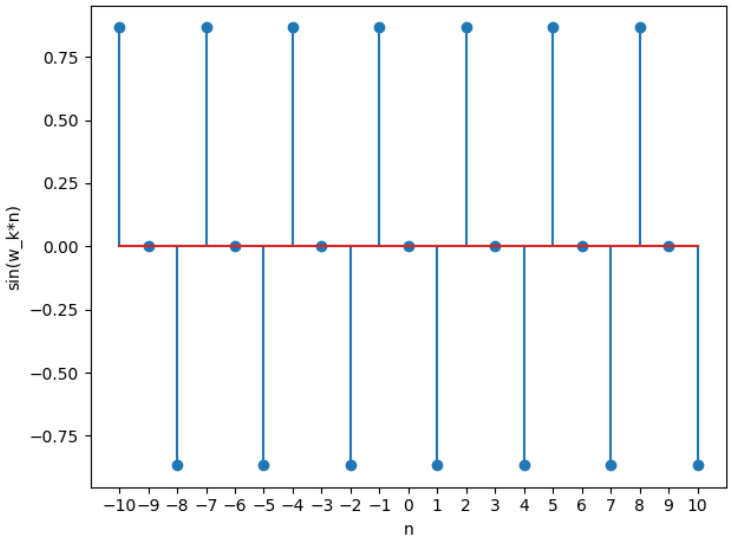


b)



c)

W_k means W_k in the vertical axis, and obviously k = 2.



d)

According to the definition of the signal:

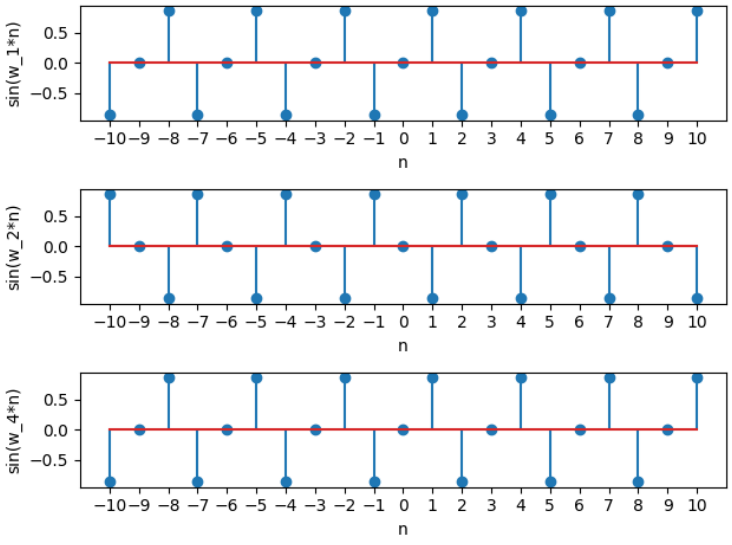
$$f[n] = sin(n\frac{2\Pi k}{3})$$

For k=1 and k=4 we have:

$$k = 1 \Rightarrow f[n] = sin(n\frac{2\Pi}{3})$$

$$k = 4 \Rightarrow f[n] = sin(n\frac{8\Pi}{3}) = sin(n(\frac{2\Pi}{3} + 2\Pi)) = sin(n\frac{2\Pi}{3})$$

And the last equation is true, since $n \in \mathbb{Z}$. So, the signals when k=1 and k=4 are equal. With a similar calculation, we can show that when k=2, the signal is the negative of the other two.



Question 2

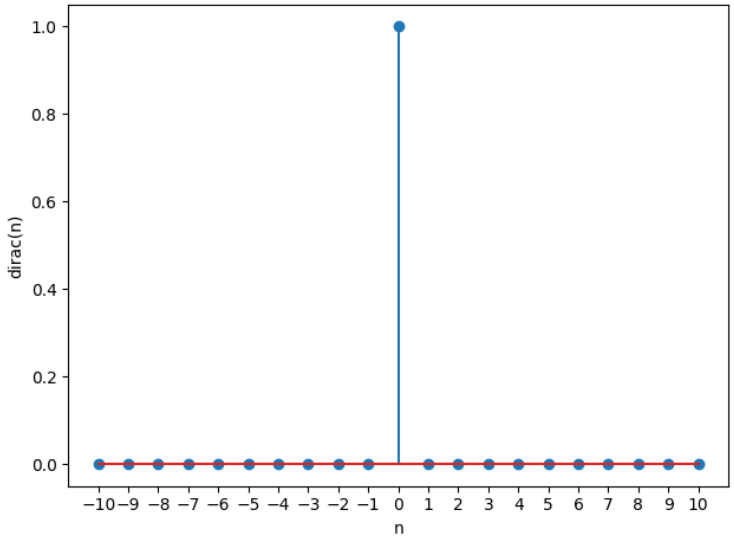
a)

infinity

Note:
I couldn't find a similar function in numpy, scipy, scikit-image, etc. So, I ran this item in Matlab and then implemented a Python function generating the same results. ("dirac" function in q2.py file)

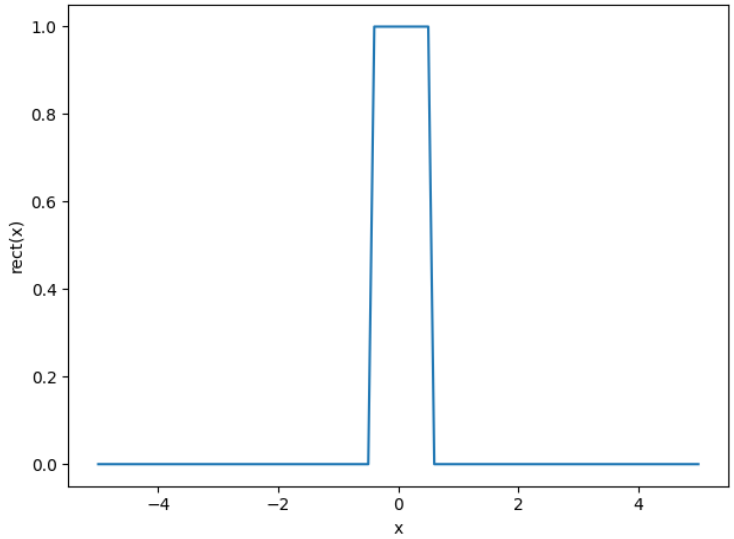
b)

Plotted the function for [-10, 10]. For any other frames, it would be similar.

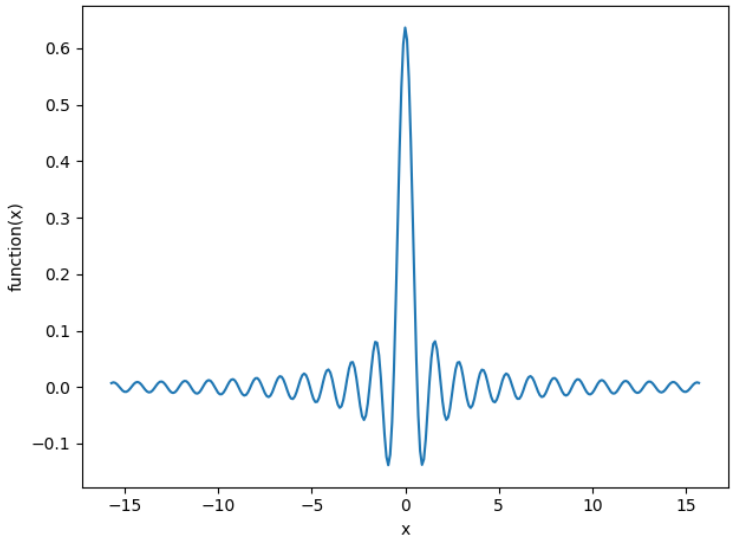


c)

The same happened with rect function. I couldn't find it in python. So, I implemented a function "rect" for this item.

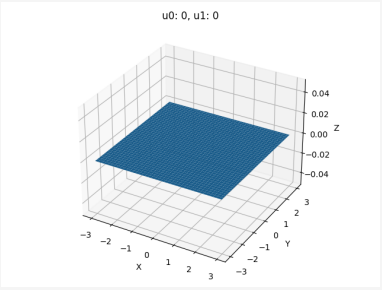
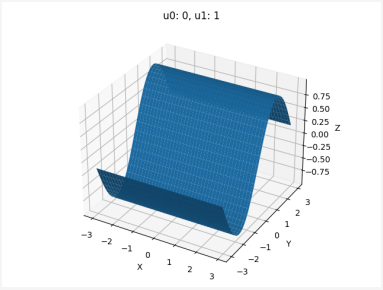
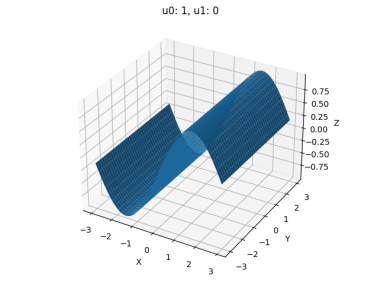
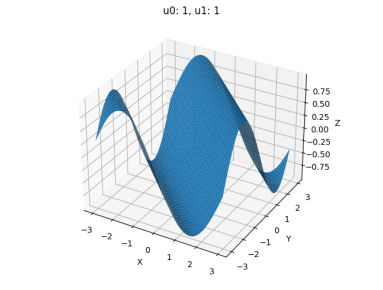


d)



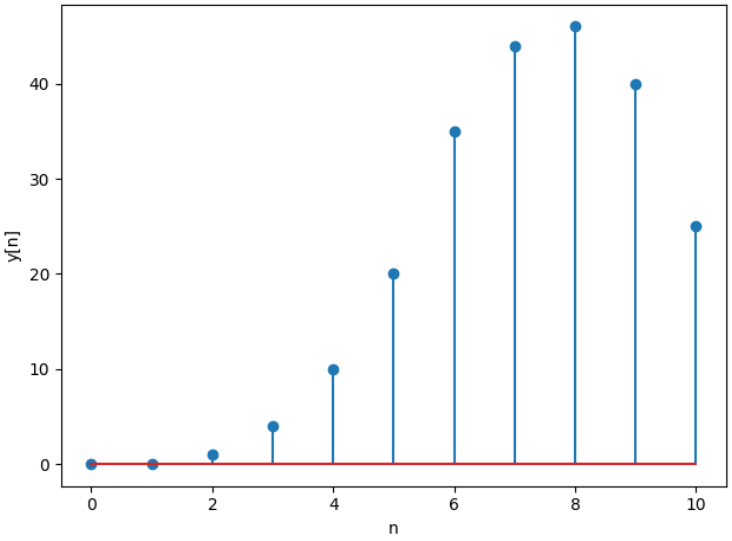
e)

According to the table below, looking at the column where u1=0 (it is constant), u0 corresponds to the period in the x-axis. Similarly, looking at the row where u0=0 (it is constant), u1 corresponds to the period of the signal in the y-axis.
Meaning that the bigger the u0, the more fluctuations in the x-axis. And we can say a similar statement for u1.

	u1 = 0	u1 = 1
u0 = 0		
u0 = 1		

Question 3

a)

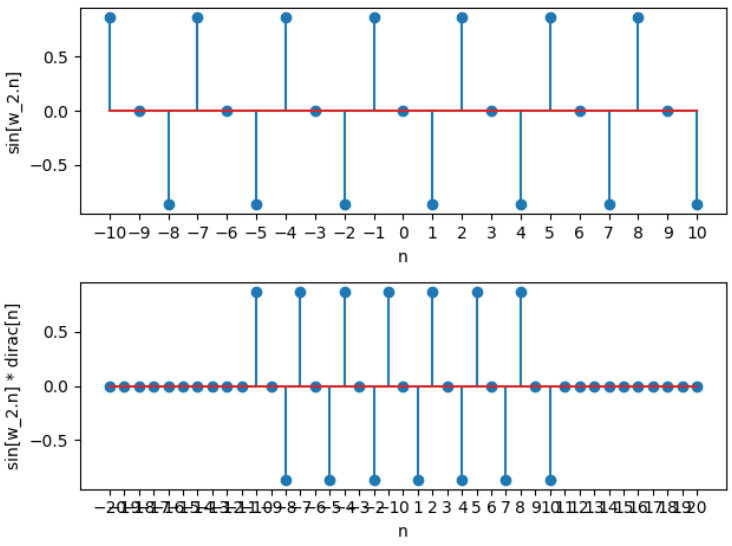


b)

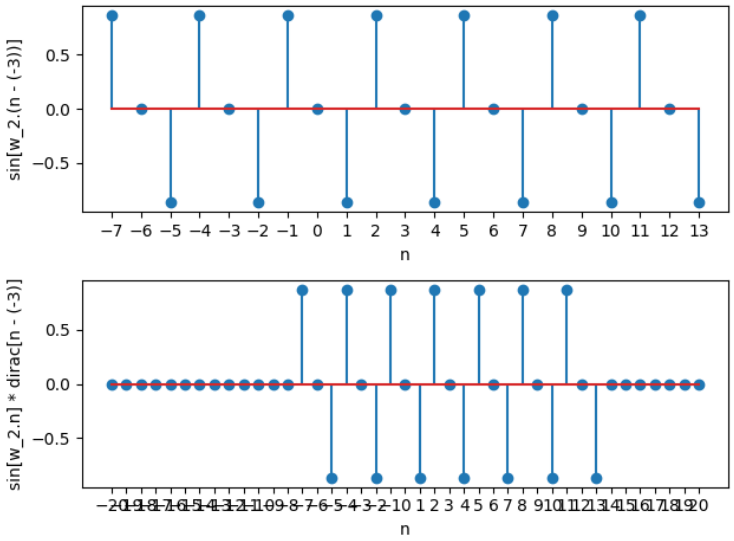
Note: since the convolution in Python or Matlab for discrete-time signals only works with the values in arrays, we took the liberty to use the shifting property for better illustrations in diagrams for negative values (please check ***item_b_1*** and ***item_b_2*** functions in ***q3.py***).

To show $f[n] * \delta[n] = f[n]$:

Note: the below diagrams are the same. The difference between the upper and the lower one is the frame. The upper one is shown in [-10, 10], and the lower one is shown in [-20, 20], and its n-axis is a little bit messed up because there is small space and lots of numbers to display.

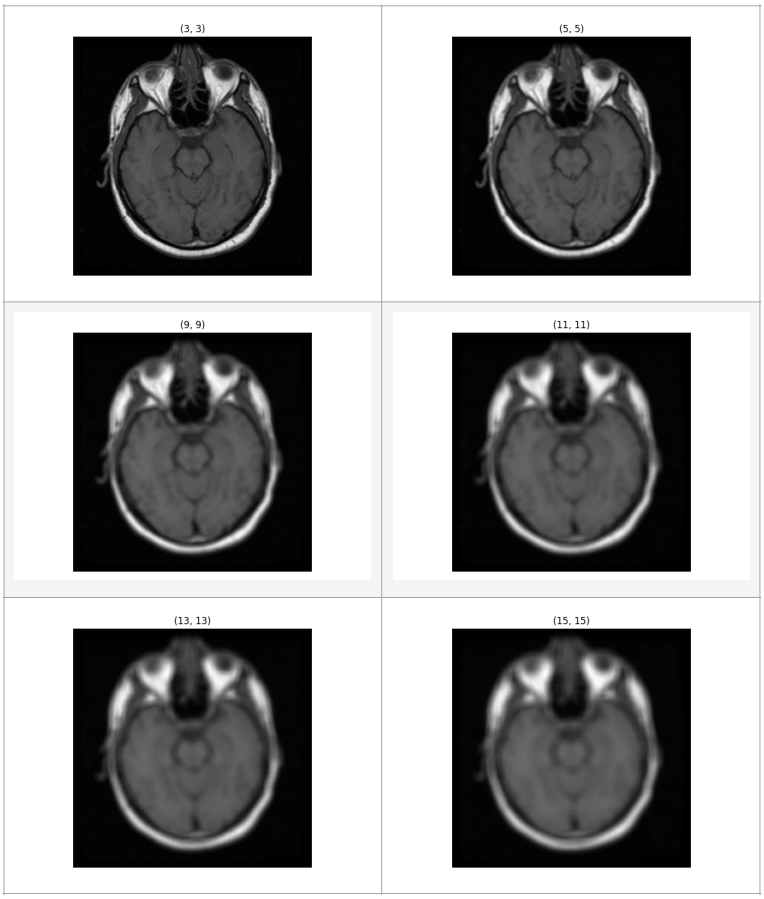


And to show $f[n] * \delta[n-m] = f[n-m]$, please check the below diagrams.



c)

As these filters have an averaging effect based on the surrounding pixels of an image, they are considered low pass filters, where they preserve low frequencies in the Fourier transform but remove the high frequency (or details) of the image. As the filters get larger, the averaging happens with farther neighbours, which results in removing more details (high frequencies). So the resulting image will seem blurrier. The table below shows the result of convolving the image with different sizes of averaging filters.



d)

Using the Sobel filter, we can easily detect the sudden changes in the vertical or horizontal axis and illustrate them.
For this item, we can use the filter below:

```
[1, 0, -1]
[2, 0, -2]
[1, 0, -1]
```

And after using it, we can just keep the absolute value of the resulting image since we are interested to see the absolute value of the sudden changes in the neighbouring pixels.
Vertical edges do not have sudden changes in the pixel value in the vertical direction, but they have it in the horizontal direction. **(I have included the images in the table in the next item for compactness)**

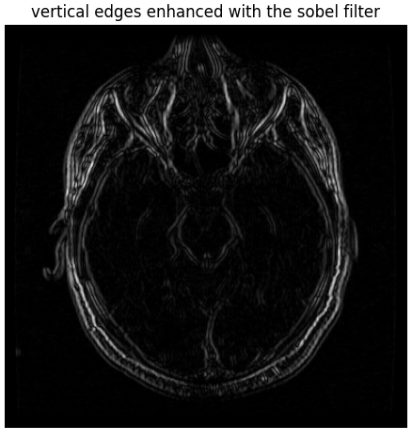
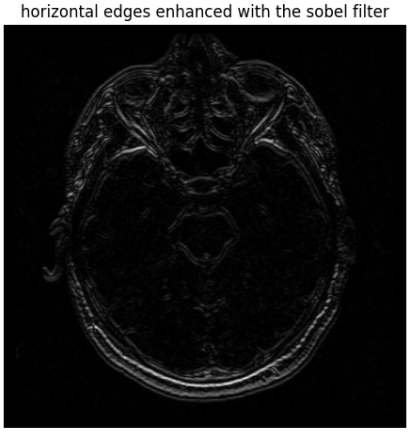
e)

Here, we have a similar scenario for the horizontal edges.

The Sobel filter will be:

```
[ 1,  2,  1]
[ 0,  0,  0]
[-1, -2, -1]
```

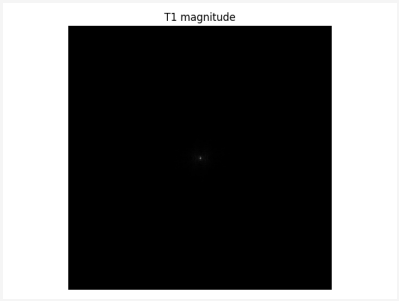
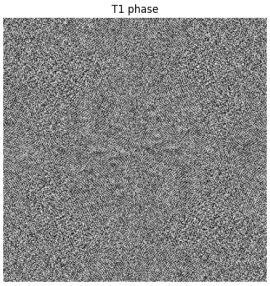
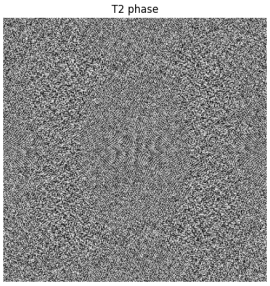
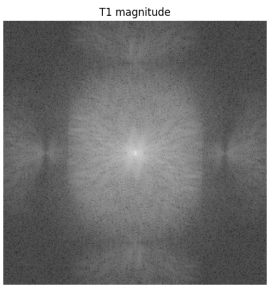
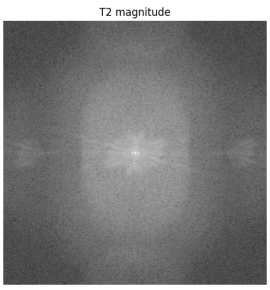
And the resulting images are:

vertical edges enhanced with the sobel filter 	horizontal edges enhanced with the sobel filter 

Question 4

a)

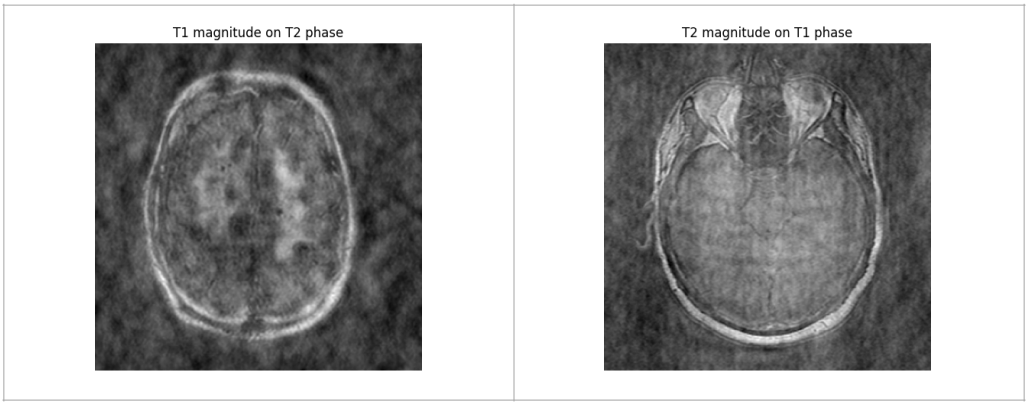
When I tried to display the magnitude, since the intensity in some small circles (DC in the middle of the image) was huge it prevented showing the intensity changes in the other parts. So I took the Log(Magnitude) of the Fourier transform and visualized it as well:

	T1.bmp	T2.bmp
Magnitude		
Phase		
Log(Magnitude)		

b)

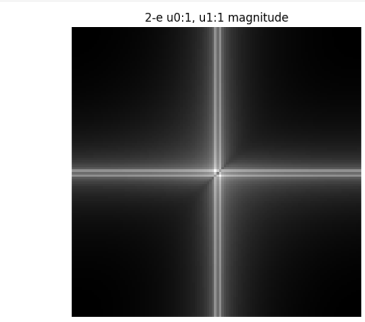
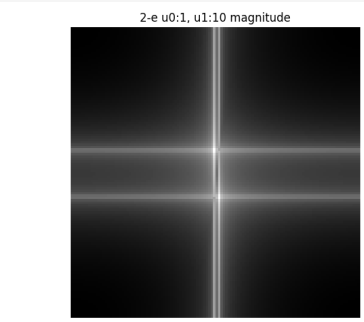
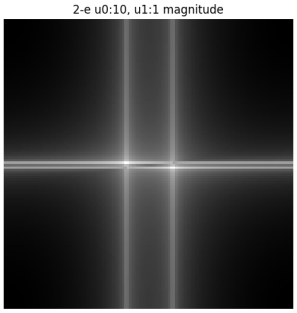
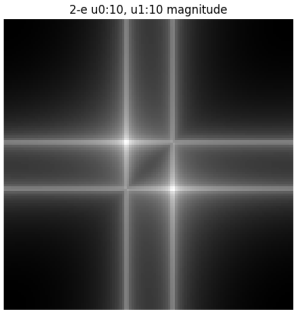
This is a very good exercise, since in the previous images, phases seemed to have no worthy information when we visualized them, and it seemed like magnitudes contained more information since they were a bit more clear. But here in this exercise, we see that those seemingly uninterpretable images contain more information. As you can see, in the table below, the resulting image with T1 magnitude and T2 phase is more like T2. And the similar for the other result. This is the result of the fact that the magnitude tells "how much" of a certain frequency component is present and the phase tells "where" the frequency component is in the image. With this description, at the location of

edges and lines, most of the frequency components have the same phase. Meaning that **the phase contains information about the locations of features (like edges)**.



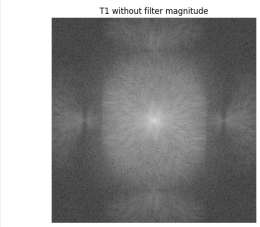
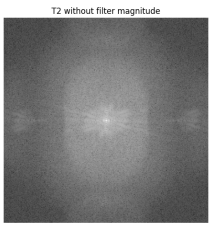
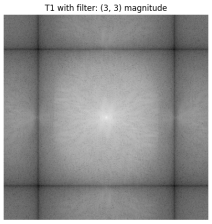
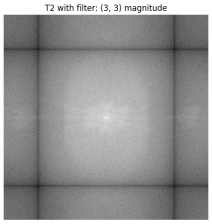
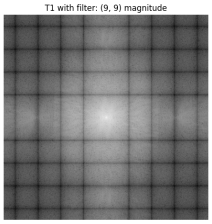
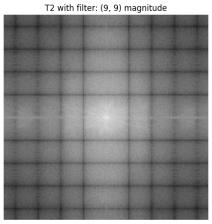
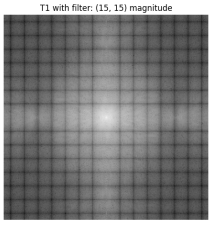
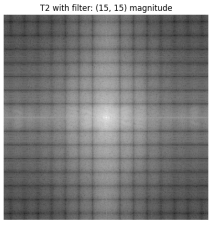
c)

Looking at the table below, even without looking at the signal function equation, we can easily tell which variable has more effect in horizontal and vertical frequency. Each pixel in the Fourier transform has a coordinate (u0, u1) representing the contribution of the sine wave with *x*-frequency u0, and *y*-frequency u1 in the Fourier transform. The center point represents the (0,0) wave – a flat plane with no ripples – and its intensity (its brightness in colour in the grey scale) is the average value of the pixels in the image. The points to the left and right of the center, represent the sine waves that vary along the *x*-axis, (ie u1=*0*). The brightness of these points represents the intensity of the sine wave with that frequency in the Fourier transform (the intensity is the amplitude of the sine wave, squared). Those vertically above and below the center point represent those sine waves that vary in *y* but remain constant in *x* (ie *u0=0*). And the other points in the Fourier transform represent the contributions of the diagonal waves.

	u1 =1	u1 = 10
u0 = 1		
u0 = 10		

d)

What we did was perform a low pass filter. It involves the elimination of the high-frequency components in the image. It results in blurring of the image (and thus a reduction in sharp transitions associated with noise). And in the frequency domain, we can see obviously the higher frequencies have been filtered out using this action, but still lower frequencies remain (along with the x and y axes of the diagrams). As the filters get larger, the averaging happens with farther neighbours, which results in removing more details (high frequencies). The bigger the filter, the higher frequencies get eliminated.

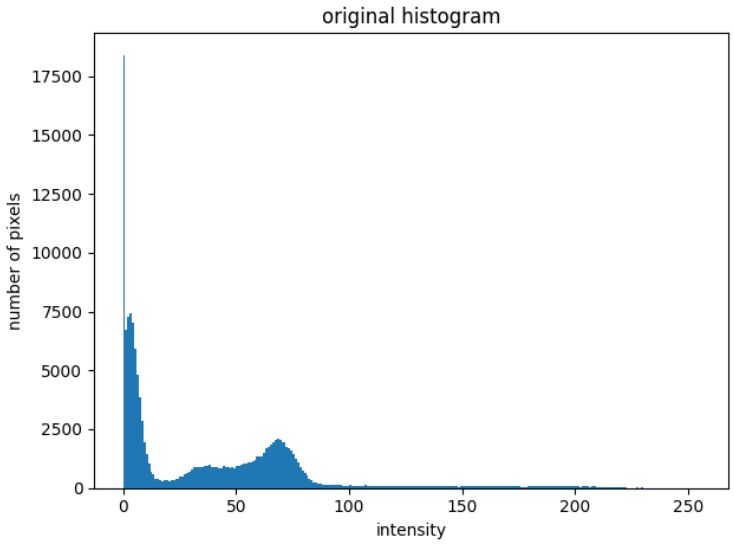
	T1.bmp	T2.bmp
Original		
Filtered with 3×3 kernel		
Filtered with 9×9 kernel		
Filtered with 15×15 kernel		

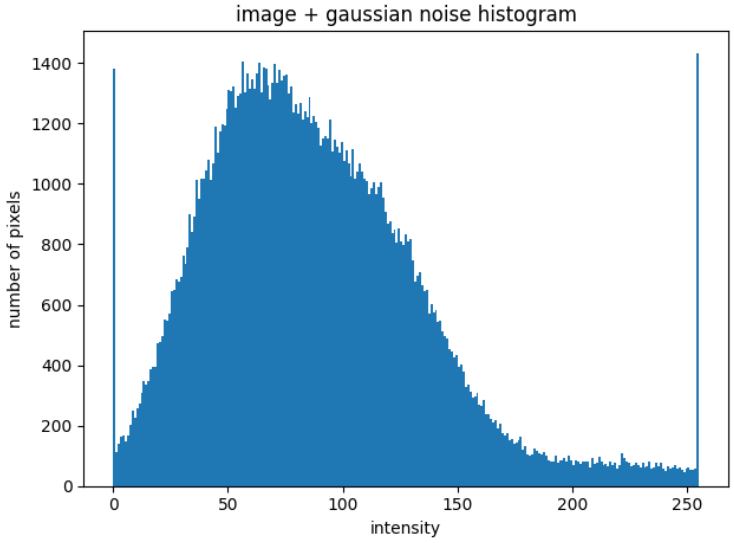
If the kernel gets large enough, only pixels in the x and y axes of the diagrams will be bright.

Question 5

a)

In the original image, most of the pixel intensity values are zero, and the rest of them are mainly below 100.
But after adding the gaussian noise, we see a peak between 50 and 100, which is because of the fact that the gaussian mean is 0.2 (0.2 * 255 = 51). We see a bell-like histogram (without a long tail since the intensities are limited to between 0 and 255).

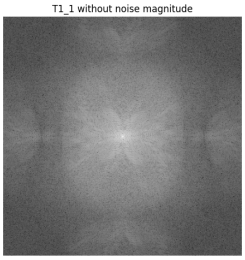
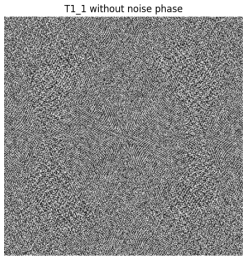
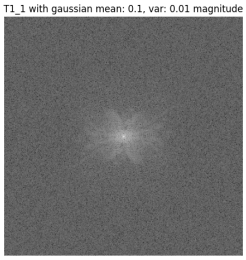
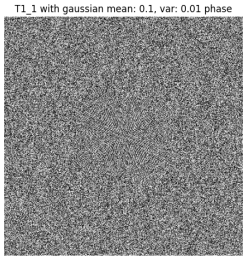
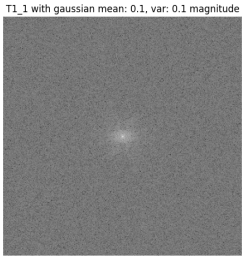
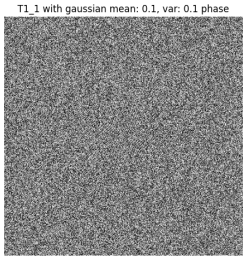
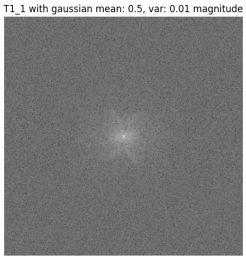
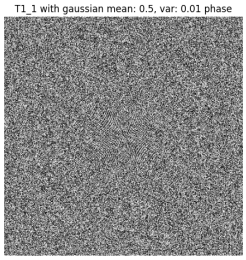




b)

- General effect of increasing mean and variance:

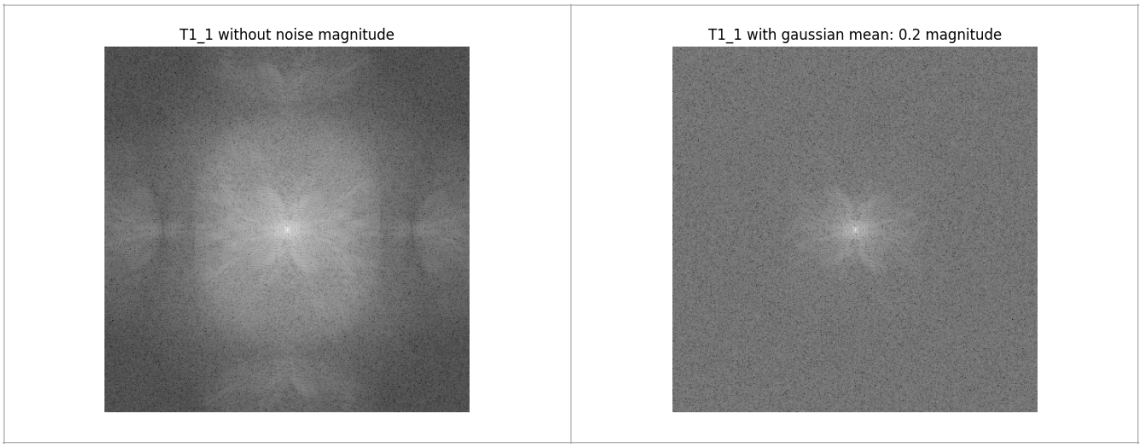
A general pattern can be seen in the magnitude diagrams with the increase of mean and variance, the focus is removed from the center of the image, and it goes more toward uniformity. But, this is a bit biased observation since the gaussian distribution is not a real gaussian being limited to be in [0, 255]. But speaking of an ideal case, in my opinion, increasing the variance should increase the intensity of the high frequencies (farther from the center of the magnitude diagram). And increasing the mean, the magnitude diagram goes toward uniformity. And phase loses its ordered pattern in both cases.

Magnitude	Phase
T1_1 without noise magnitude 	T1_1 without noise phase 
T1_1 with gaussian mean: 0.1, var: 0.01 magnitude 	T1_1 with gaussian mean: 0.1, var: 0.01 phase 
T1_1 with gaussian mean: 0.1, var: 0.1 magnitude 	T1_1 with gaussian mean: 0.1, var: 0.1 phase 
T1_1 with gaussian mean: 0.5, var: 0.01 magnitude 	T1_1 with gaussian mean: 0.5, var: 0.01 phase 

- **Comparison** of the two **magnitudes**:

In the left diagram, we see higher intensities in a circle-like shape around the middle of the image. Meaning that more of the focus of the changes is on the lower frequencies in the original image. And **there are not many sudden changes in intensity values in the original MRI image**.

But, after adding the gaussian noise to the MRI image, there is a much lower focus on the lower frequencies in the middle of the Fourier magnitudes diagram. This means that **the intensity is changing in all the frequencies (approximately) uniformly** in comparison with the original image.

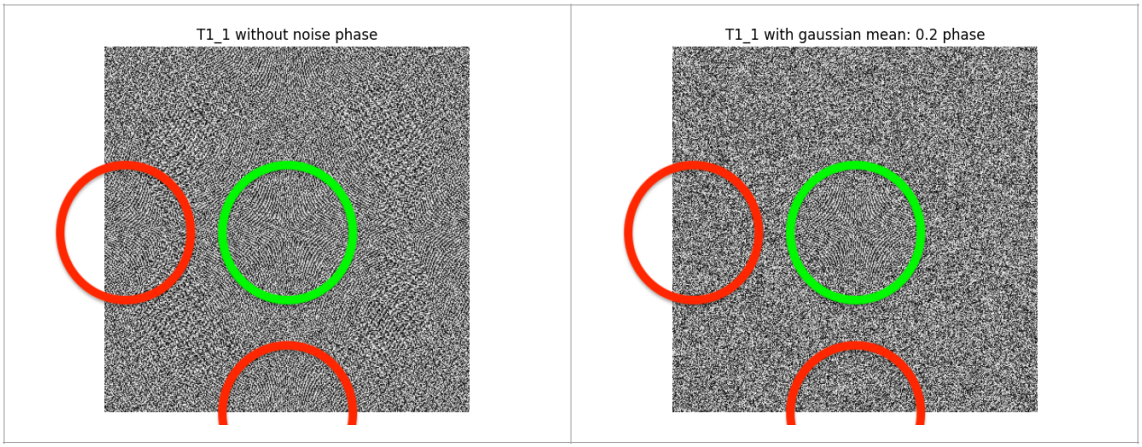


- **Comparison** of the two **phases**:

Note: I have added the circles for the comparison, and obviously, they are not actually in the phase of the Fourier transform of the image!

We see a similar pattern in both diagrams inside the green circles (which correspond to the low-frequency components). Meaning that the **gaussian noise has a lower effect on the phase of the low-frequency components**.

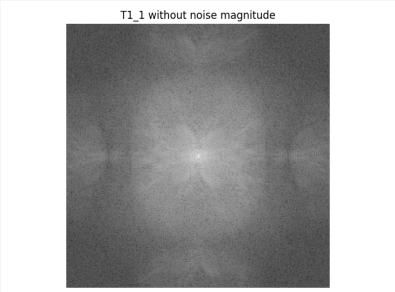
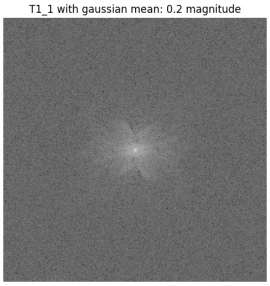
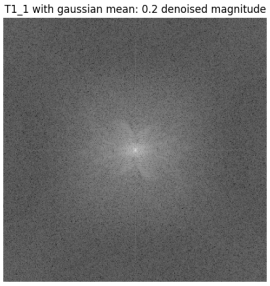
But, in the red circles that correspond to the phase of the high-frequency components, we see the ordered pattern of the original image has changed a lot. So, we can conclude that the **gaussian noise can affect the high-frequency components more**.



c)

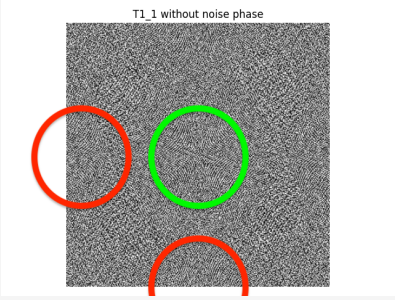
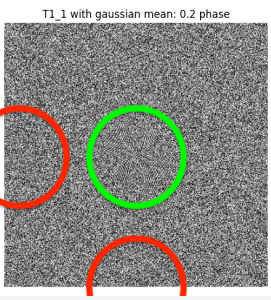
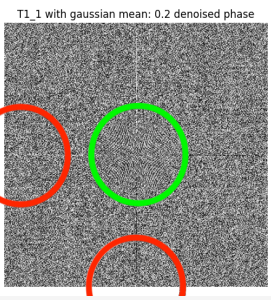
- Comparison of the two **magnitudes**:

We can see a brighter circle-shaped aura in the middle of the denoised diagram. It is more like the original one. Going from the center of the diagram to the sides, the aura gets disappeared. This shows that more of the focus of the changes is on the lower frequencies in the denoised image, similar to the original one. The denoising algorithm can partially revert the image.

Original	Noisy	Denoised
		

- Comparison of the two **phases**:

Unlike the magnitudes, the denoising algorithm is not that successful in removing the noise effect on the phases, and we cannot see the ordered pattern in the red circles. But, the patterns in the green circles still look alike.

Original	Noisy	Denoised
		

Question 6

a)

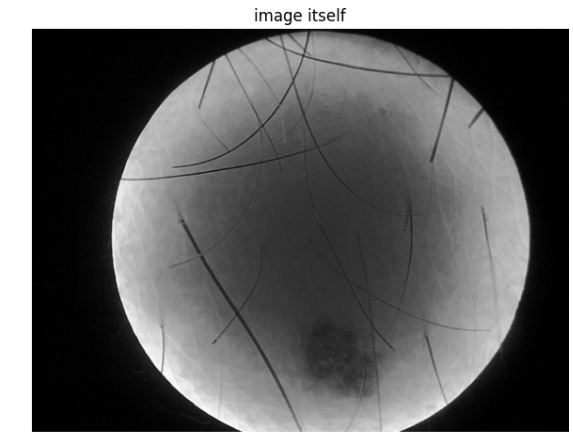
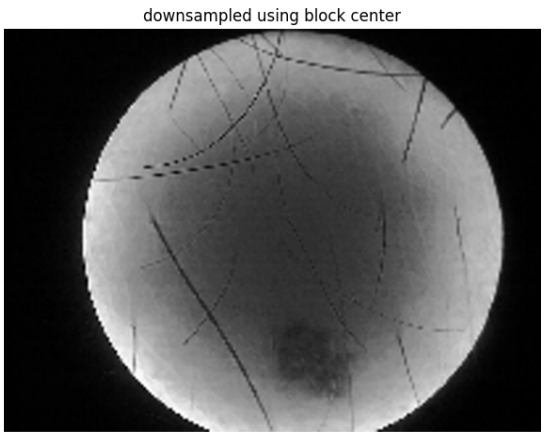
please check the ***_block_center*** function in file ***q6.py***.

b)

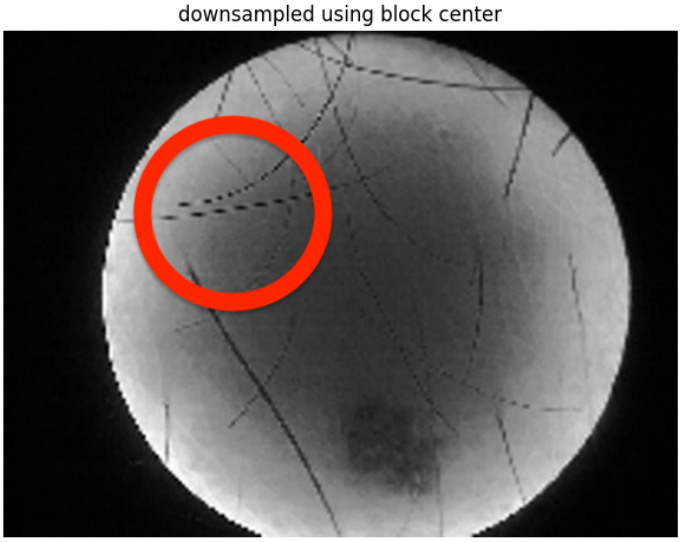
Loss of detail or aliasing effect: Since only the center of the kernel is taken, any details that are present in the surrounding pixels will be lost.

Additionally, we have a blurring effect in the resulting image. Since only a small portion of the image is taken, the resulting image may appear blurred or smudged.

Also in color images, there can be a loss of color information. Depending on the kernel size, some color information may be lost due to averaging or discarding of pixels.

Original	Down-sampled
	

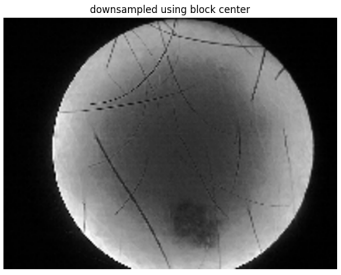
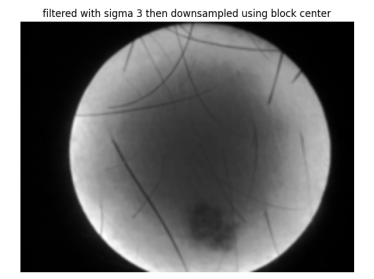
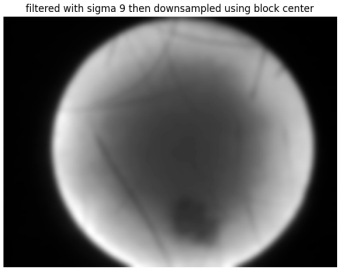
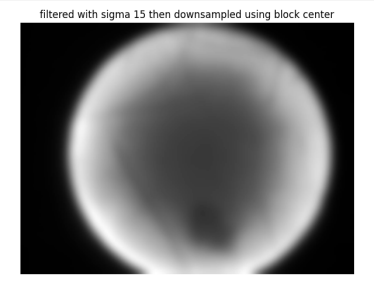
The aliasing effect can be seen in the image below inside the red circle. There is hair, but it seems that there are broken lines in the downsampled image.



c)

- Comparing with the result in item b:

Using the gaussian filter helps to see the hairs connected, and we are not seeing a single hair being disconnected. But, it has a downside and it is that the image looks blurry.

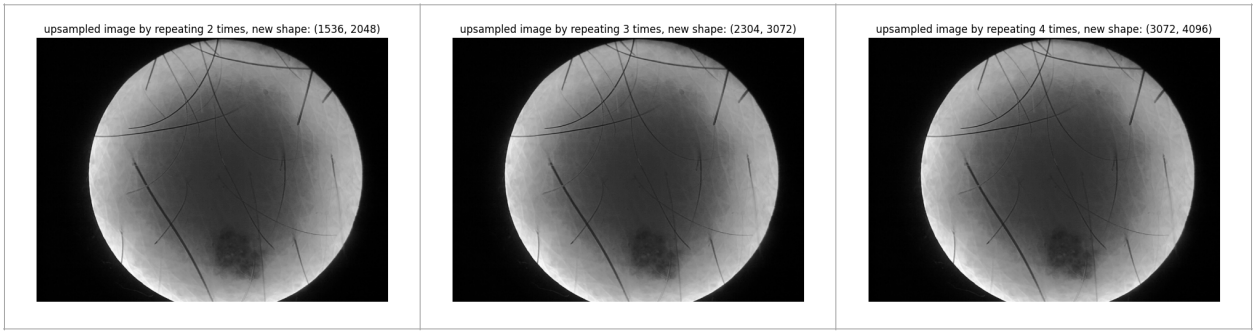
Result of item b	
Filtered with sigma=3 before downsampling	
Filtered with sigma=9 before downsampling	
Filtered with sigma=15 before downsampling	

- Discussion on the result of down-sampling on the image resolution:

Down-sampling involves reducing the number of pixels in the image, which results in a lower resolution image with fewer details and less clarity. Mathematically, this is done

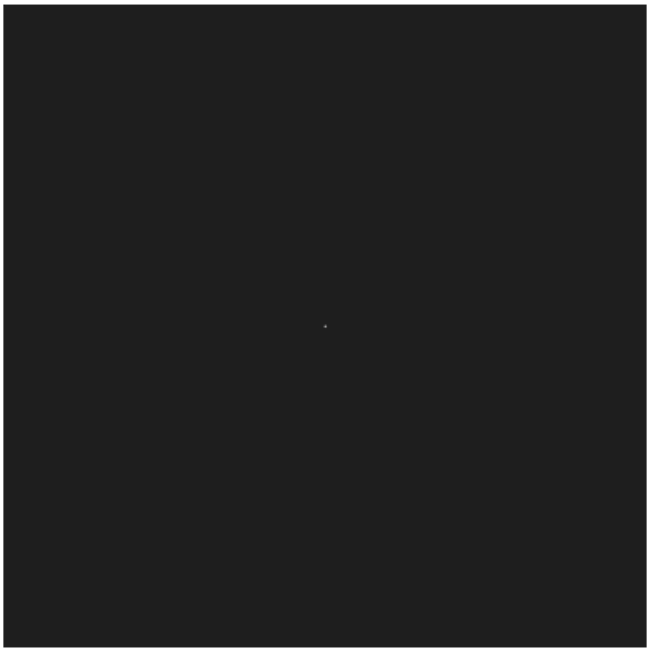
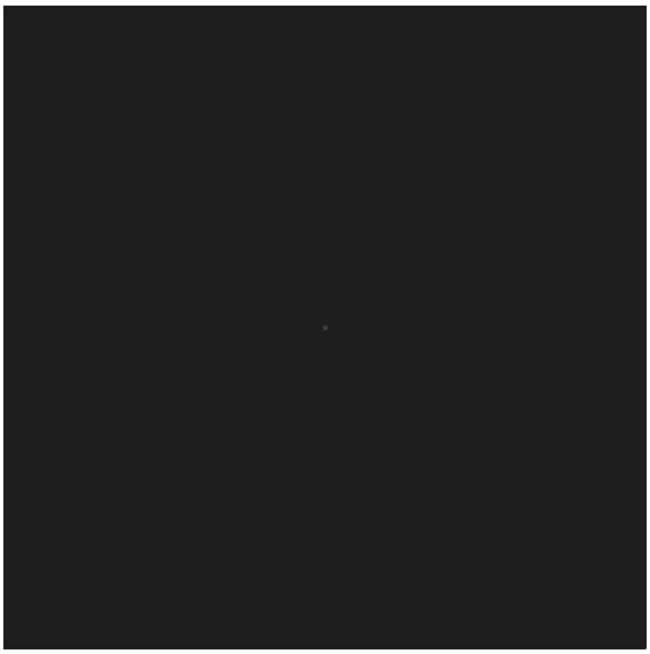
by dividing the width and height of an image by a factor, resulting in a decrease in the total number of pixels. The more downsampling that is done, the more detail will be lost.

d)



Please read the image titles to see the difference between the images. They do not differ in our eyes. As you can see, up-sampling here is a process of increasing the number of pixels in an image without adding any new information. Some other methods can make an image appear smoother, but they do not actually increase the physical resolution of the image and add any “detail” to it, just like when we cannot describe behind the wall when we cannot see it! So, Considering these descriptions, the answer is no, it does not change the physical resolution!

Question 7
a)

The small circle	
The small circle after filter	

I calculated m_f and m_g using this formula:

$$modulation = \frac{max - min}{max + min}$$

For the circles we have:

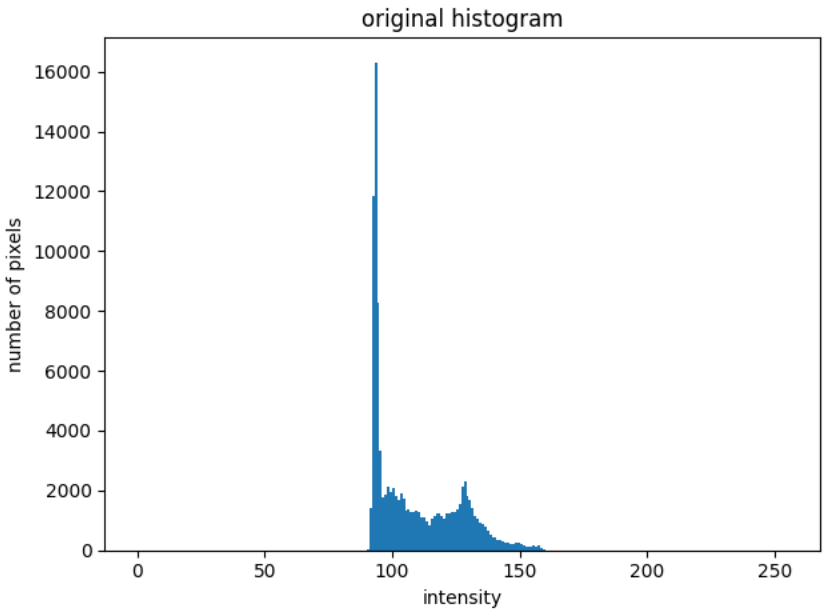
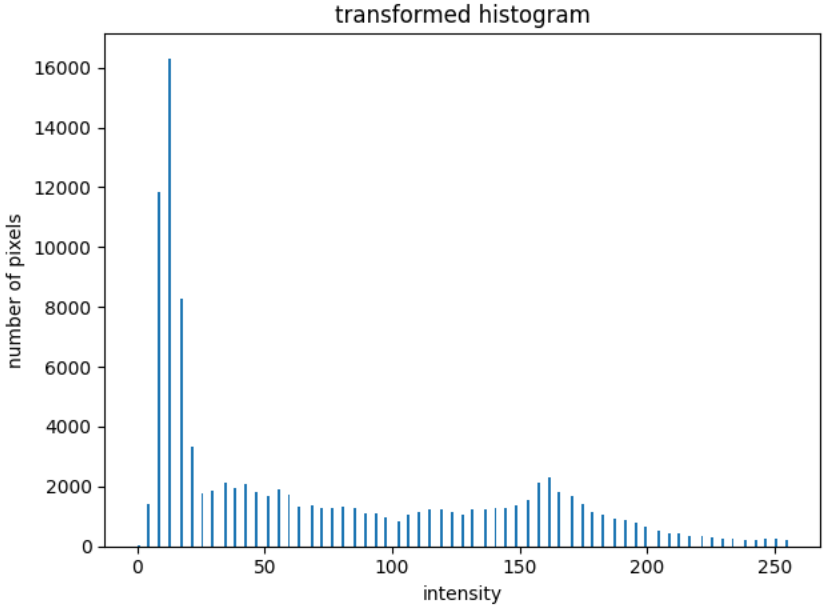
$$m_f = 0.7894736842105263$$

$$m_g = 0.7482517482517483$$

The modulation ratio is:

$$ratio = \frac{m_g}{m_f} = 0.9477855477855479$$

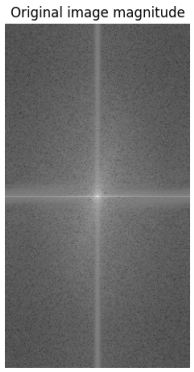
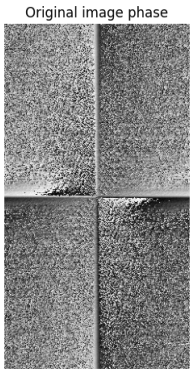
b)

The original image histogram	
After histogram stretching	

c)

According to the figures below, the magnitudes change a bit: after the histogram stretching the whole image gets brighter (or, better to say the DC gets darker and the others appear to be brighter). This is because the DC value in the middle of the image

gets changed (since the DC value is the sum of all the intensities!). And the remaining relative values stay the same. Actually, histogram stretching only changes the intensity value and preserves the relative intensity. And, this causes an incredibly amazing thing to happen, the phases before and after the transformation (histogram stretching) are equal.

	Magnitude	Phase
The original		
After histogram stretching