Digital Image Processing
Assignment 3
Meher Shashwat Nigam (20171062)

Assignment 3

QUESTION 1 (q1.m)

Part 1

Works on arrays of length that are powers of 2.

Input array:

```
x = [1 ;2 ;3 ;4 ;5 ;6 ;7 ;8];
```

Inbuilt function output:

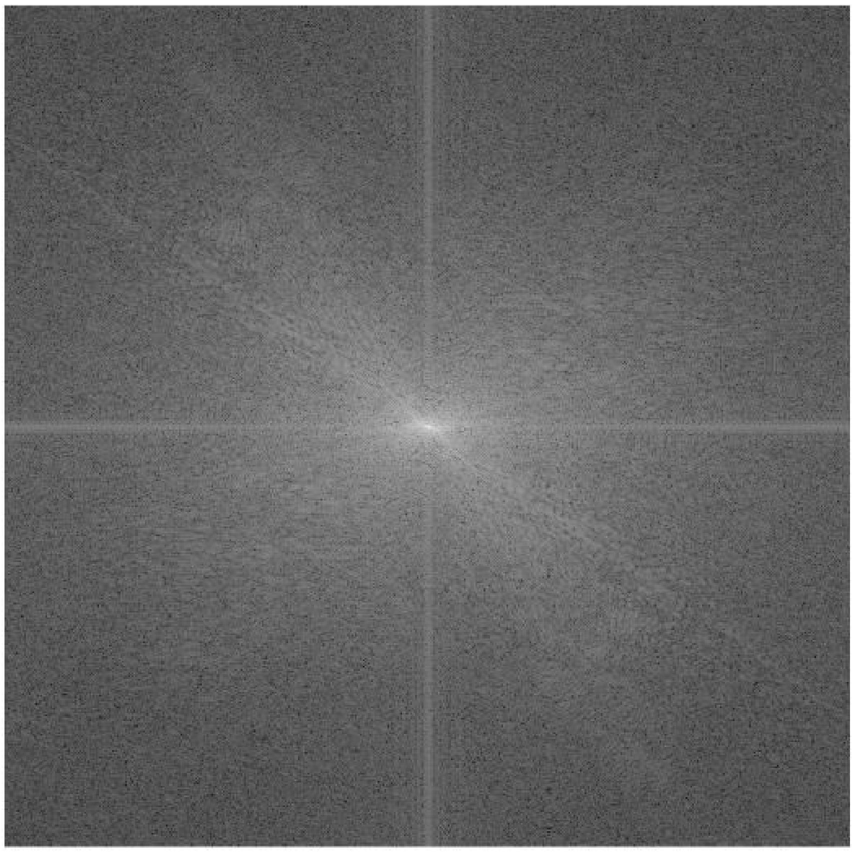
```
fftx =

  36.0000 + 0.0000i

  -4.0000 + 9.6569i

  -4.0000 + 4.0000i

  -4.0000 + 1.6569i

  -4.0000 + 0.0000i

  -4.0000 - 1.6569i

  -4.0000 - 4.0000i

  -4.0000 - 9.6569i
```

My function output:

```
fft_x =
36.0000 + 0.0000i

-4.0000 + 9.6569i

-4.0000 + 4.0000i

-4.0000 + 1.6569i

-4.0000 + 0.0000i

-4.0000 - 1.6569i

-4.0000 - 4.0000i

-4.0000 - 9.6569i
```

Part 2

Image used: lena.jpg



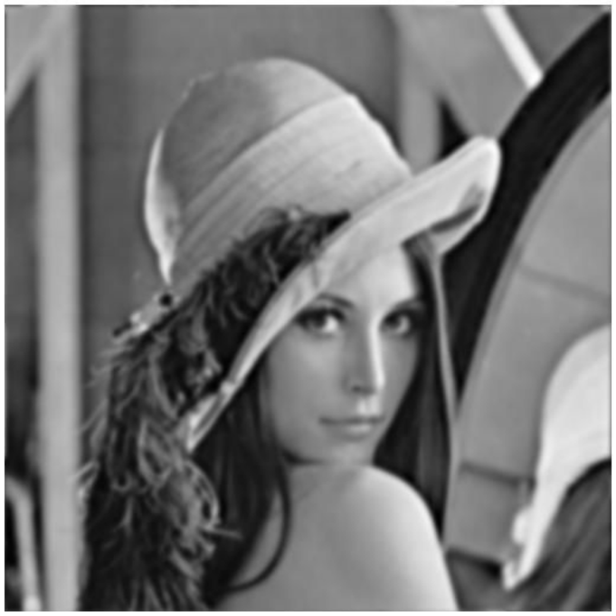
Output on extending to 2d dft from running fft on all rows and then running fft on all columns on the resulting matrix, using the concept of linear separability of 2d dft.

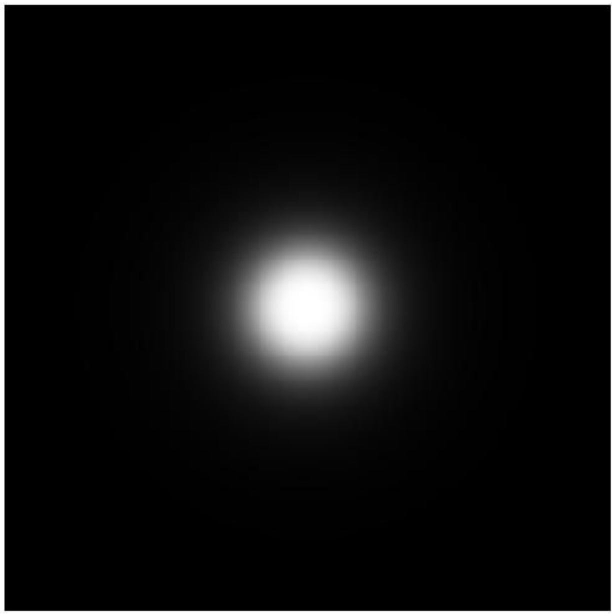
QUESTION 2 (q2.m)

Part 1

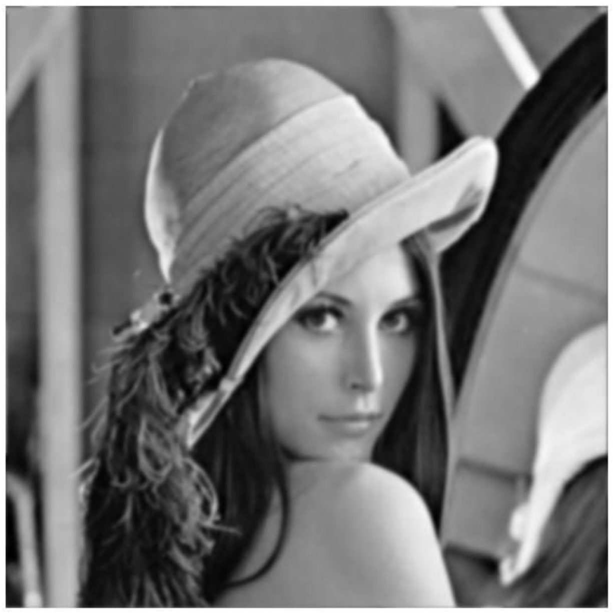
<- Input image

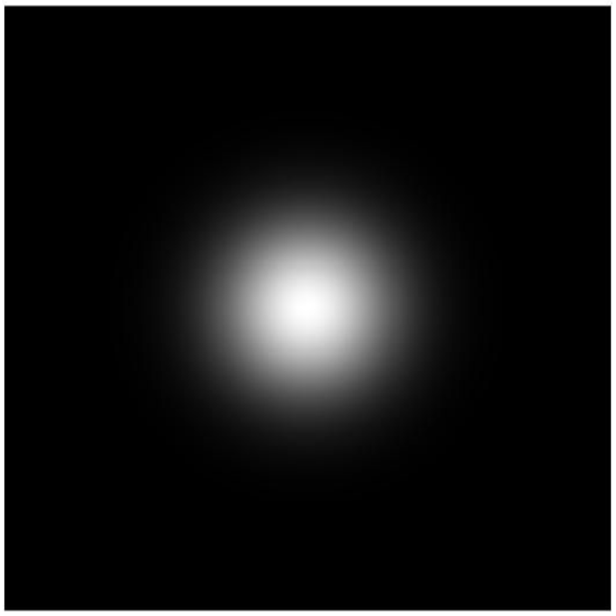
Output from butterworth filter (d0 =50, n=2):





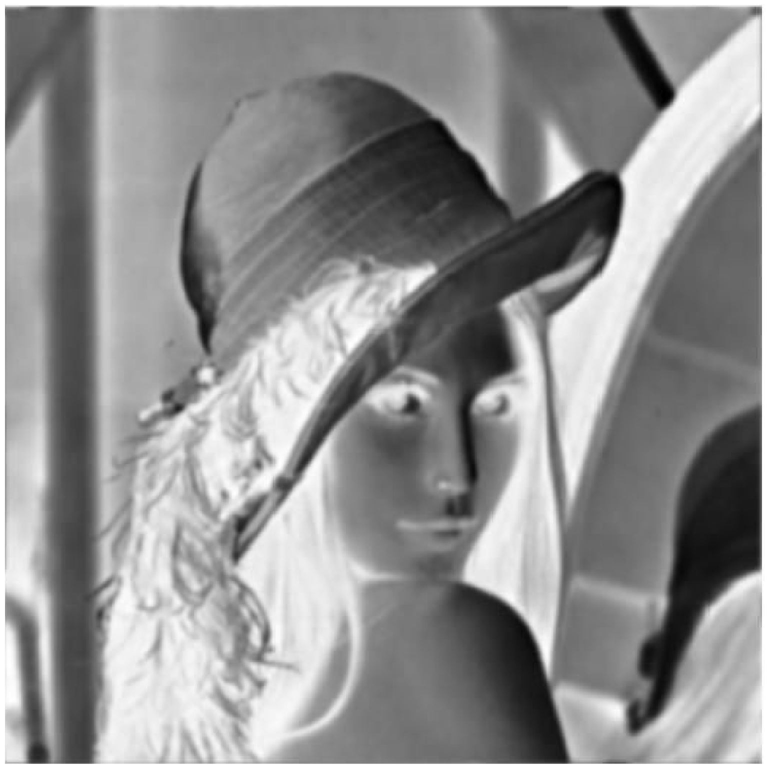
Output from Gaussian filter (sigma =50) :





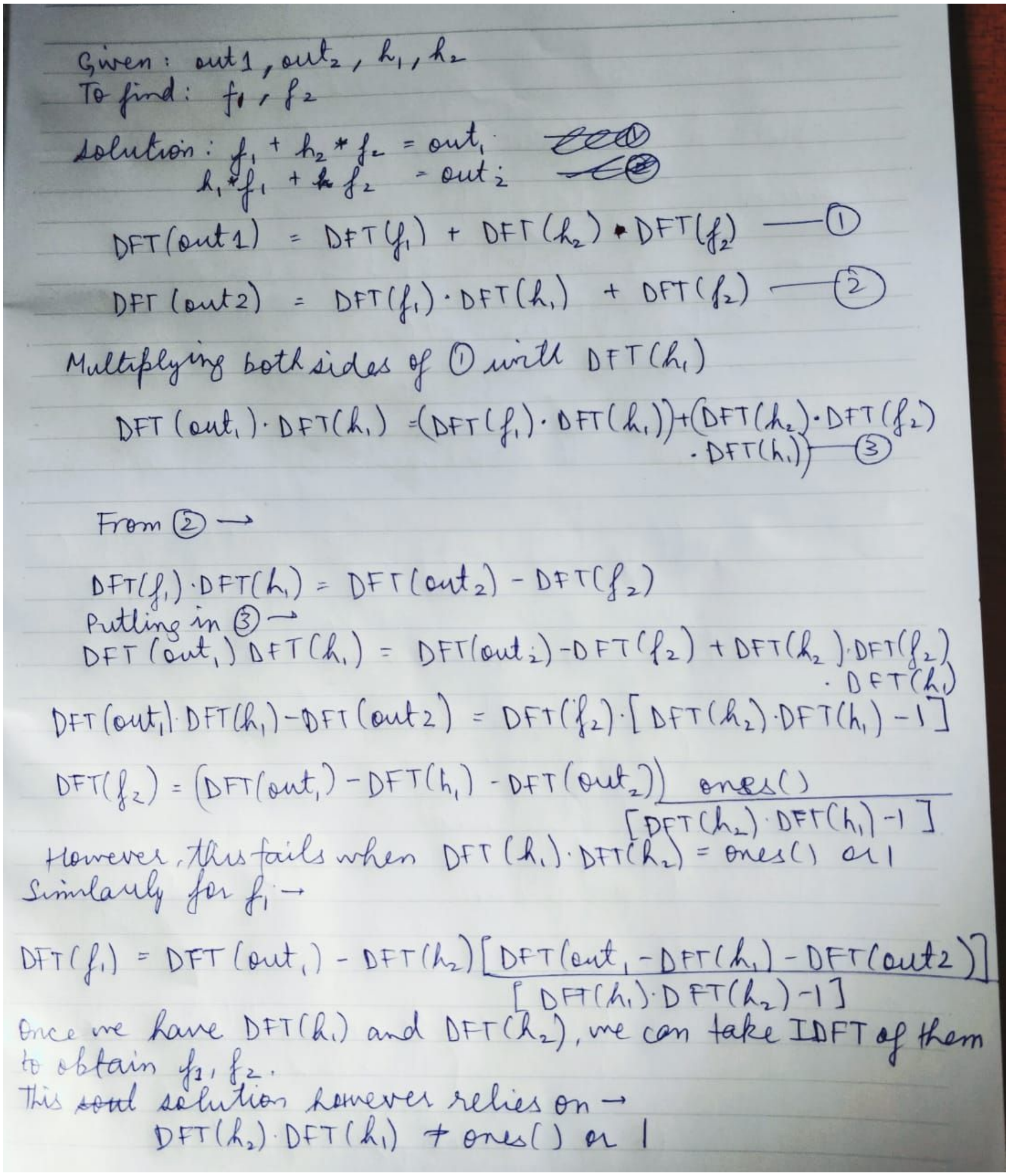
Part 2

Difference between gaussian filtered images (sigma = 50 and 100).



It acts like a bandpass filter. Allowing only certain frequencies to pass.

QUESTION 3 (q3.m)



Given : out1 , out_2 , h_1 , h_2

To find : f_1 , f_2

Solution : $f_1 + h_2 * f_2 = out_1$

$h_1 * f_1 + \cancel{h}\, f_2 = out_2$

$$DFT(out1) = DFT(f_1) + DFT(h_2) \cdot DFT(f_2) \text{ —— } ①$$

$$DFT(out2) = DFT(f_1) \cdot DFT(h_1) + DFT(f_2) \text{ —— } ②$$

Multiplying both sides of ① with $DFT(h_1)$

$$DFT(out_1) \cdot DFT(h_1) = (DFT(f_1) \cdot DFT(h_1)) + (DFT(h_2) \cdot DFT(f_2) \cdot DFT(h_1)) \text{ —— } ③$$

From ② →

$$DFT(f_1) \cdot DFT(h_1) = DFT(out_2) - DFT(f_2)$$

Putting in ③ →

$$DFT(out_1)\, DFT(h_1) = DFT(out_2) - DFT(f_2) + DFT(h_2) \cdot DFT(f_2) \cdot DFT(h_1)$$

$$DFT(out_1) \cdot DFT(h_1) - DFT(out2) = DFT(f_2) \cdot [DFT(h_2) \cdot DFT(h_1) - 1]$$

$$DFT(f_2) = \frac{(DFT(out_1) - DFT(h_1) - DFT(out_2)) \quad ones()}{[DFT(h_2) \cdot DFT(h_1) - 1]}$$

However, this fails when $DFT(h_1) \cdot DFT(h_2) = ones()$ or 1

Similarly for f_1 →

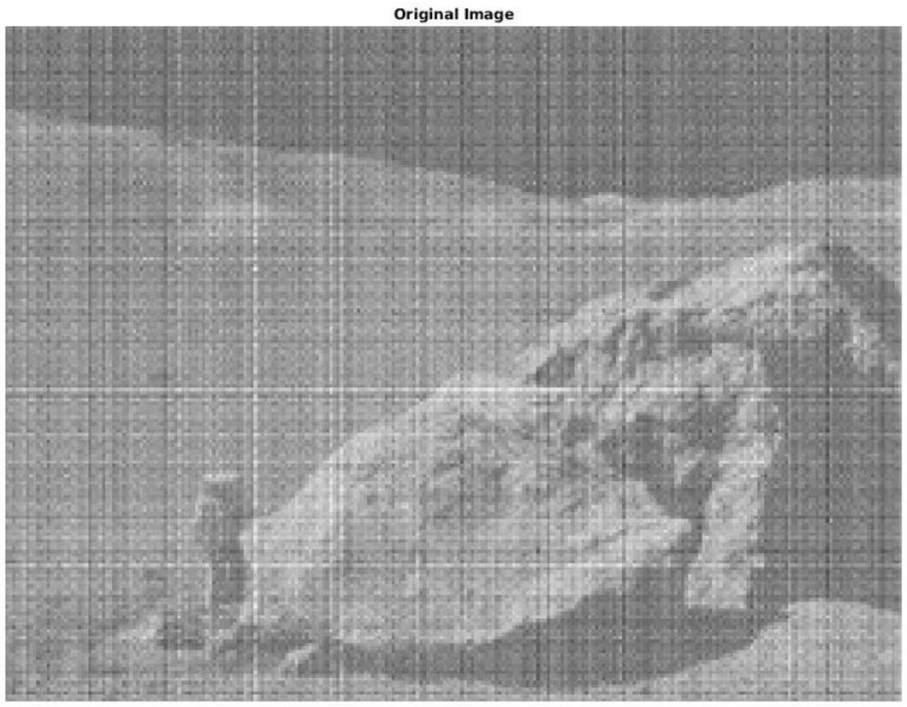
$$DFT(f_1) = DFT(out_1) - DFT(h_2) \frac{[DFT(out_1 - DFT(h_1) - DFT(out2))]}{[DFT(h_1) \cdot DFT(h_2) - 1]}$$

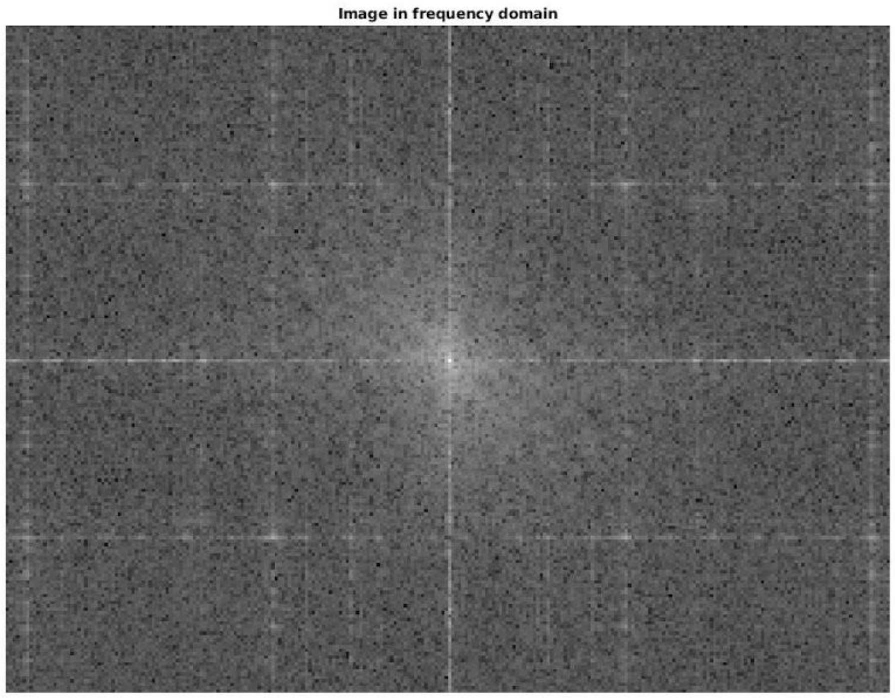
Once we have $DFT(h_1)$ and $DFT(h_2)$, we can take IDFT of them to obtain f_1, f_2.

This ~~soul~~ solution however relies on →

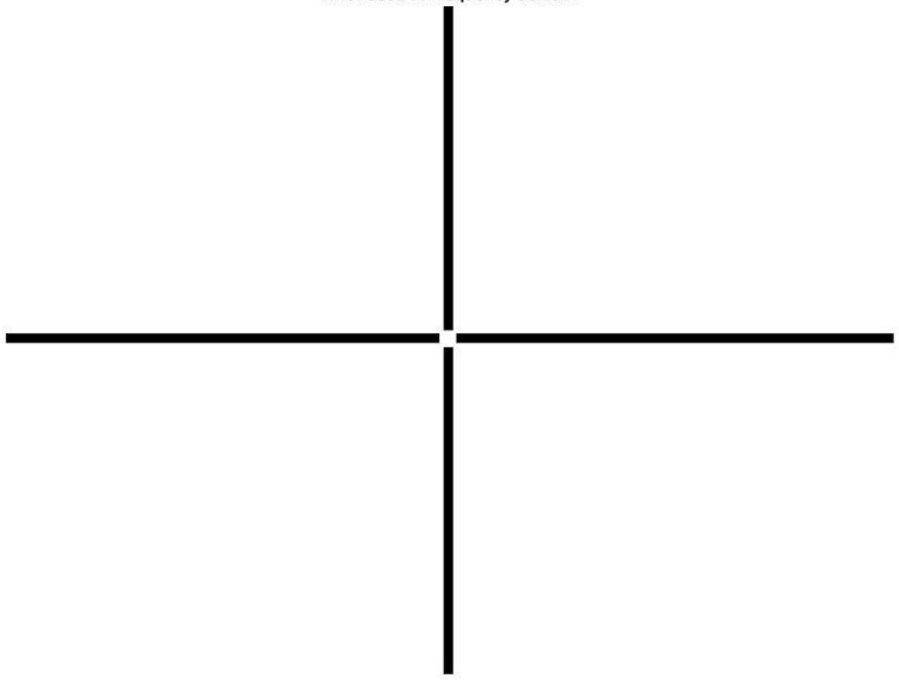
$$DFT(h_2) \cdot DFT(h_1) \neq ones() \text{ or } 1$$

QUESTION 4 (q4.m)

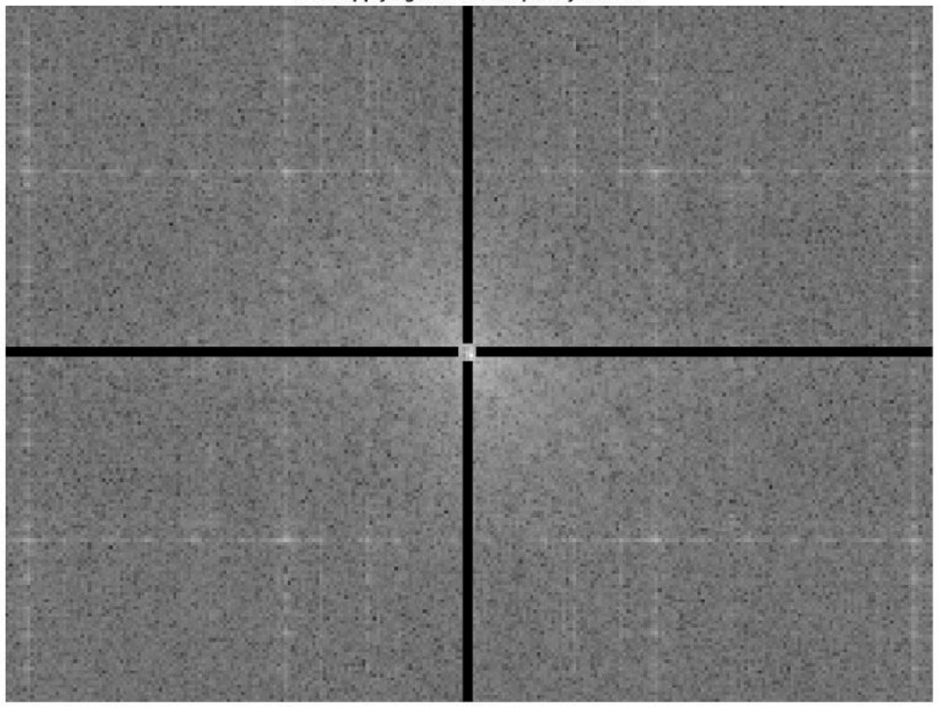




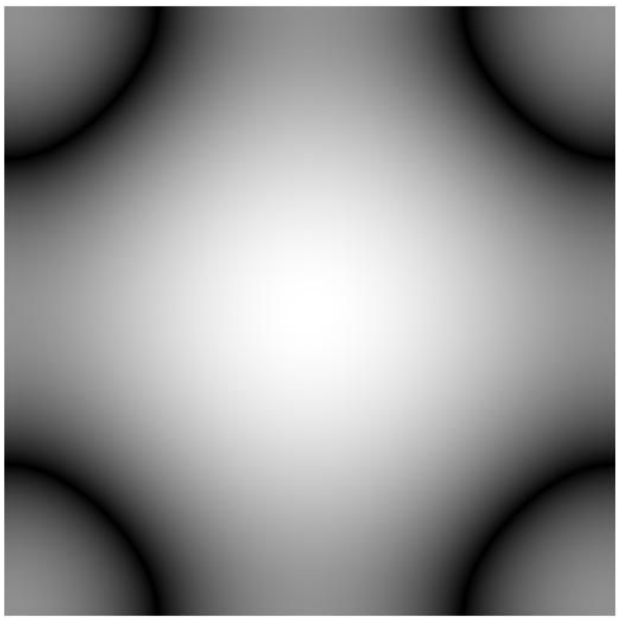
filter used in frequency domain



After applying mask in frequency domain



QUESTION 5 (q5.m)



The output image and the filter in frequency domain.

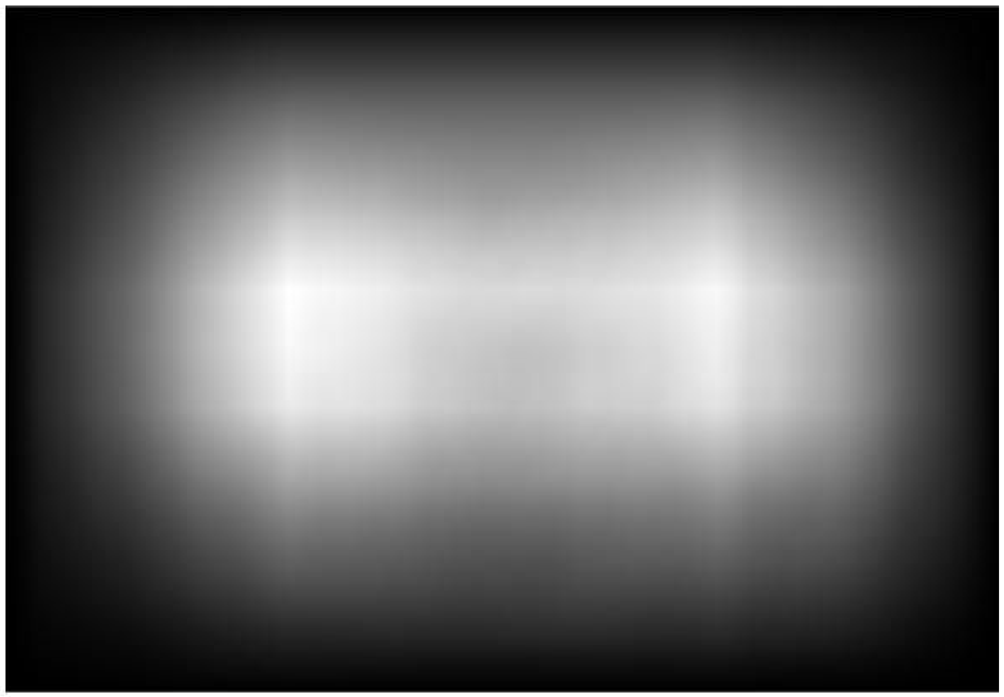
The output image was obtained from taking its fft after appropriate padding, doing pointwise multiplication with fft of filter and then taking ifft.

The filter acts like a band pass filter in frequency domain as can be seen from its fft image. It doesn't allow frequencies in the black regions to pass.

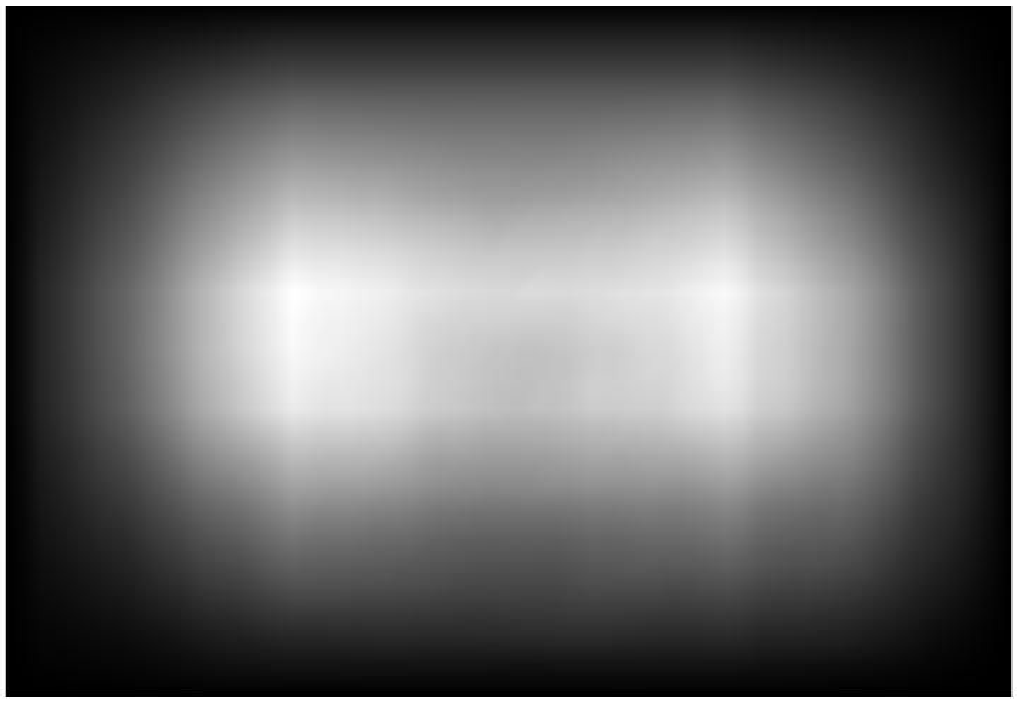
QUESTION 6 (q6.m)

Part 1

Output from pointwise multiplication in frequency domain.



Output from taking fft of image convolved with filter using conv2..



Part 2

When the dimensions of the images being convolved are small, then conv2 works much faster than taking the process through fourier domain. When of larger dimensions, the fft domain filtering outperforms conv2 by a large margin.

Case 1 (small dimensions):

Img1 = (4 x 3), Img2 = (3 x 8)

```
fourier_time =

   0.0172

conv_time =

  1.0000e-04
```

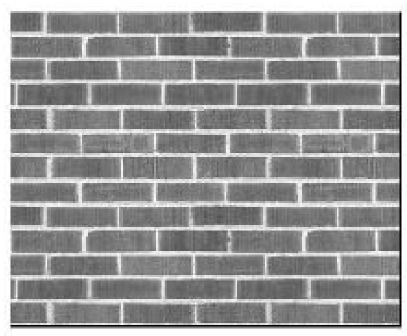
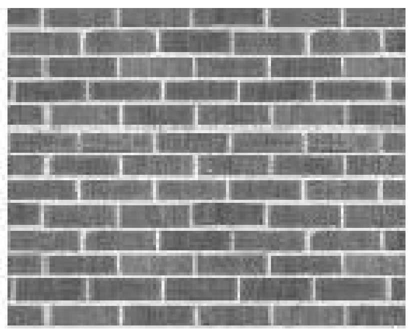
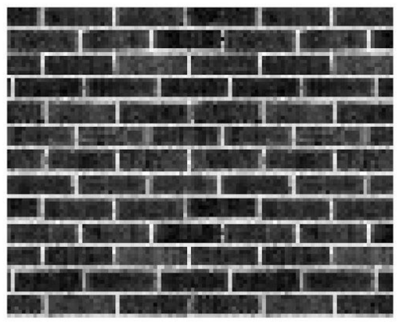
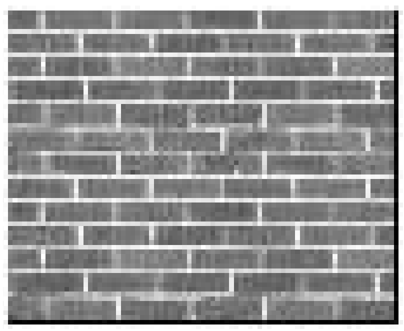
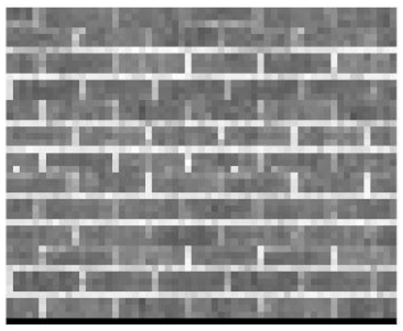
Case 2(larger dimensions):

Img1 = (512 x 512), Img2 = (720 x 1280)

```
fourier_time =

   0.2050

conv_time =

   8.6422
```

QUESTION 7 (q7 .m)

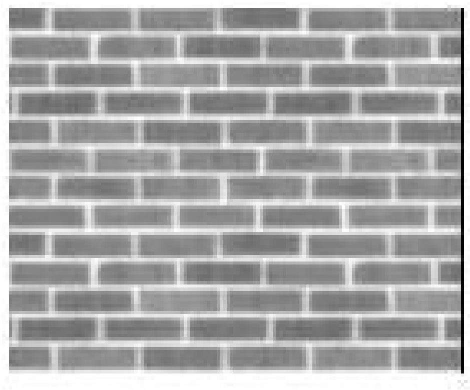
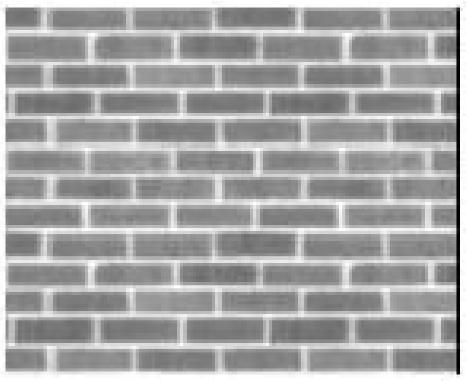
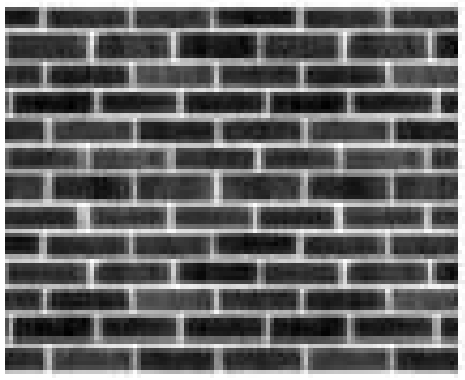
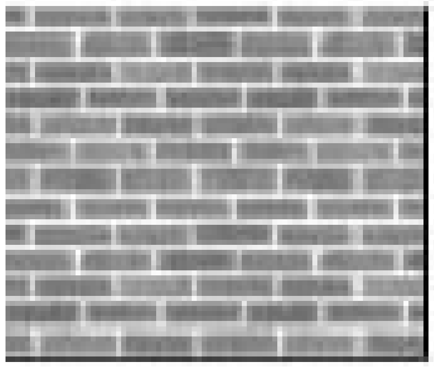
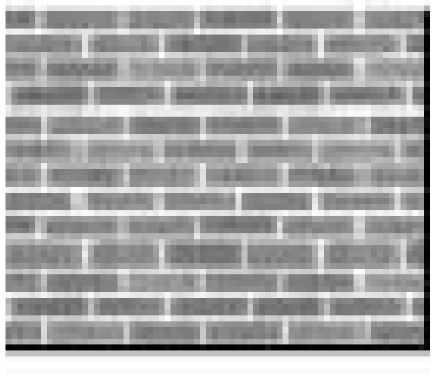
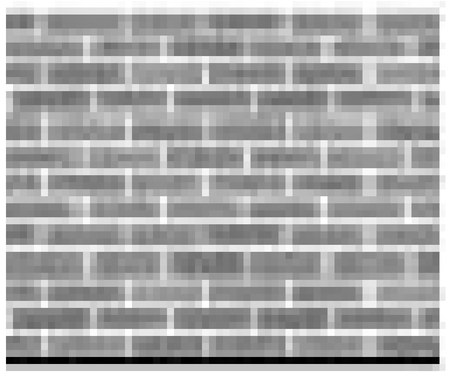
Part 1

Sampling at nx,ny = 2,3,4,5,7.

We observe that after 4, the image starts aliasing. So nx,ny = 4 and nyquist rate =0.25.

Part 2

For blurring I used a gaussian filter of sigma 2.

Above is the output for nx,ny = 2,3,4,5,6,7. It is observed that after nx,ny = 6 it starts aliasing.

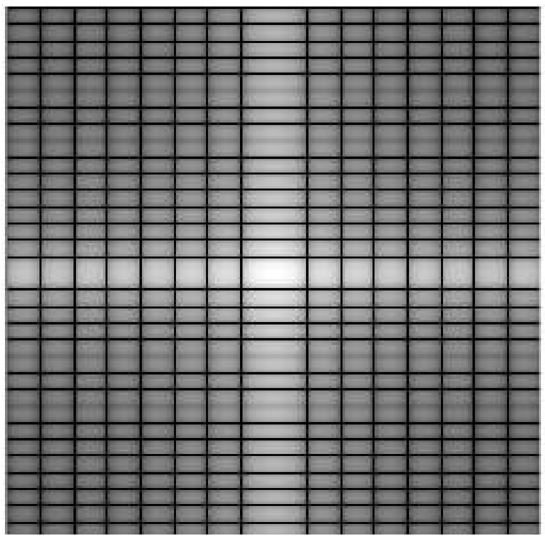
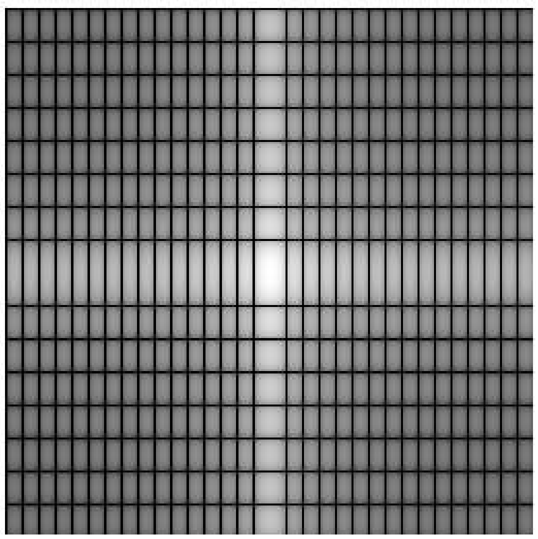
So the rate is 0.1667.

Blurring has the effect of increasing nx,ny hence it does some antialiasing as a conclusion.

QUESTION 8 (q8.m)

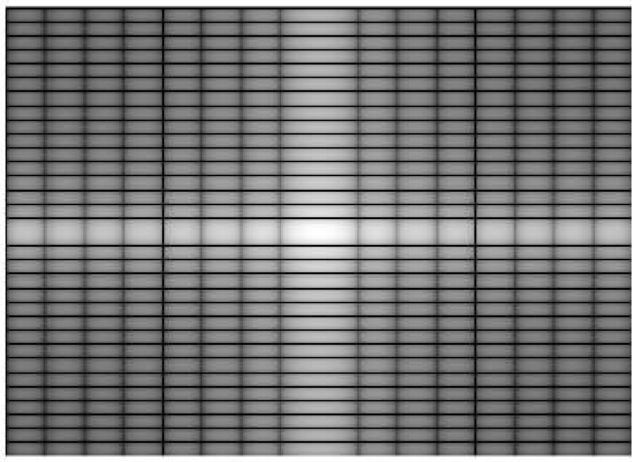
Part 1

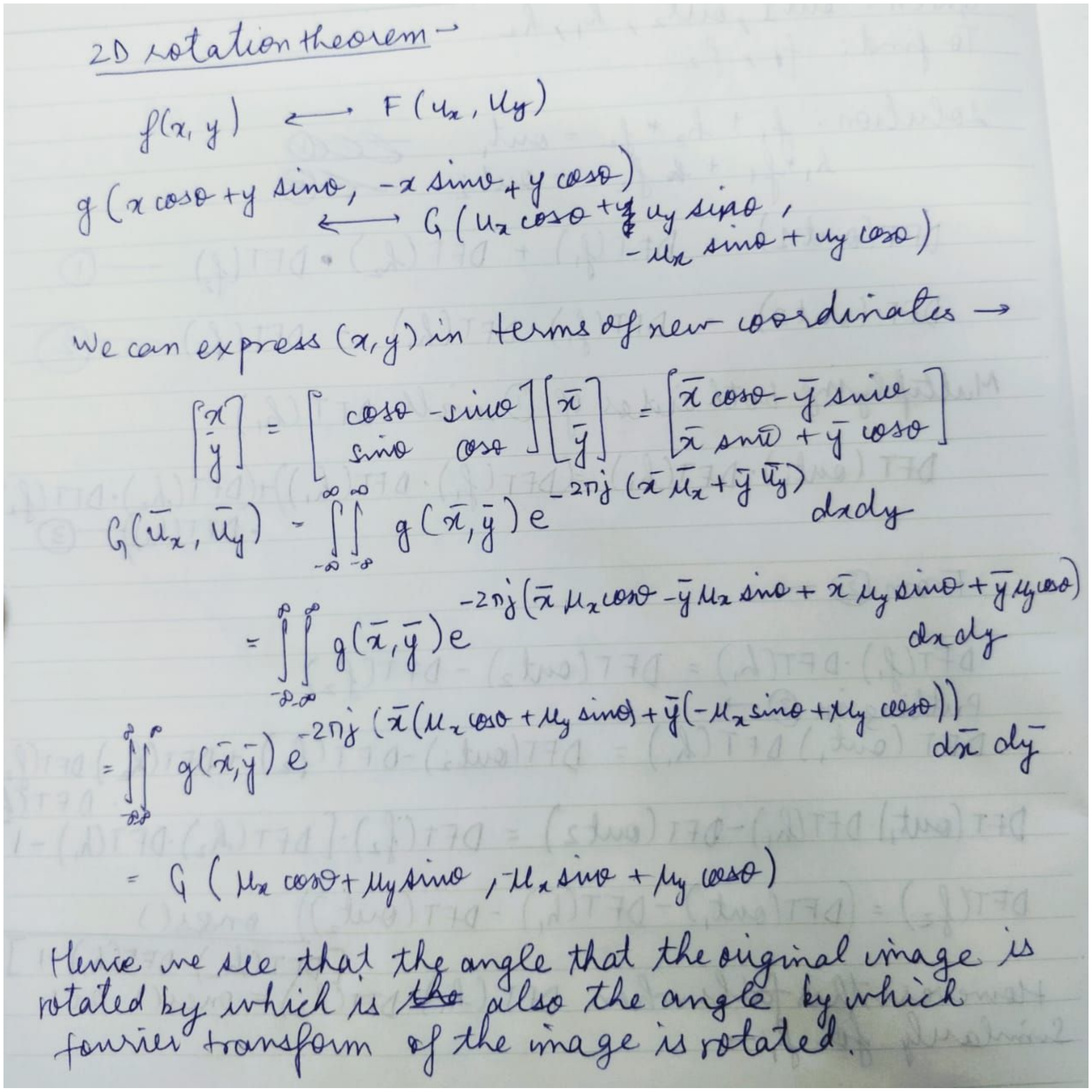
Fourier transform of the original image and the rotated image:

Part 2

Fourier transform of the image after translating (padding on left) by 100 pixels:





2D rotation theorem →

$$f(x, y) \longleftrightarrow F(u_x, u_y)$$

$$g(x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta) \longleftrightarrow G(u_x\cos\theta + \cancel{y}\, u_y\sin\theta, -u_x\sin\theta + u_y\cos\theta)$$

We can express (x, y) in terms of new coordinates →

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} = \begin{bmatrix} \bar{x}\cos\theta - \bar{y}\sin\theta \\ \bar{x}\sin\theta + \bar{y}\cos\theta \end{bmatrix}$$

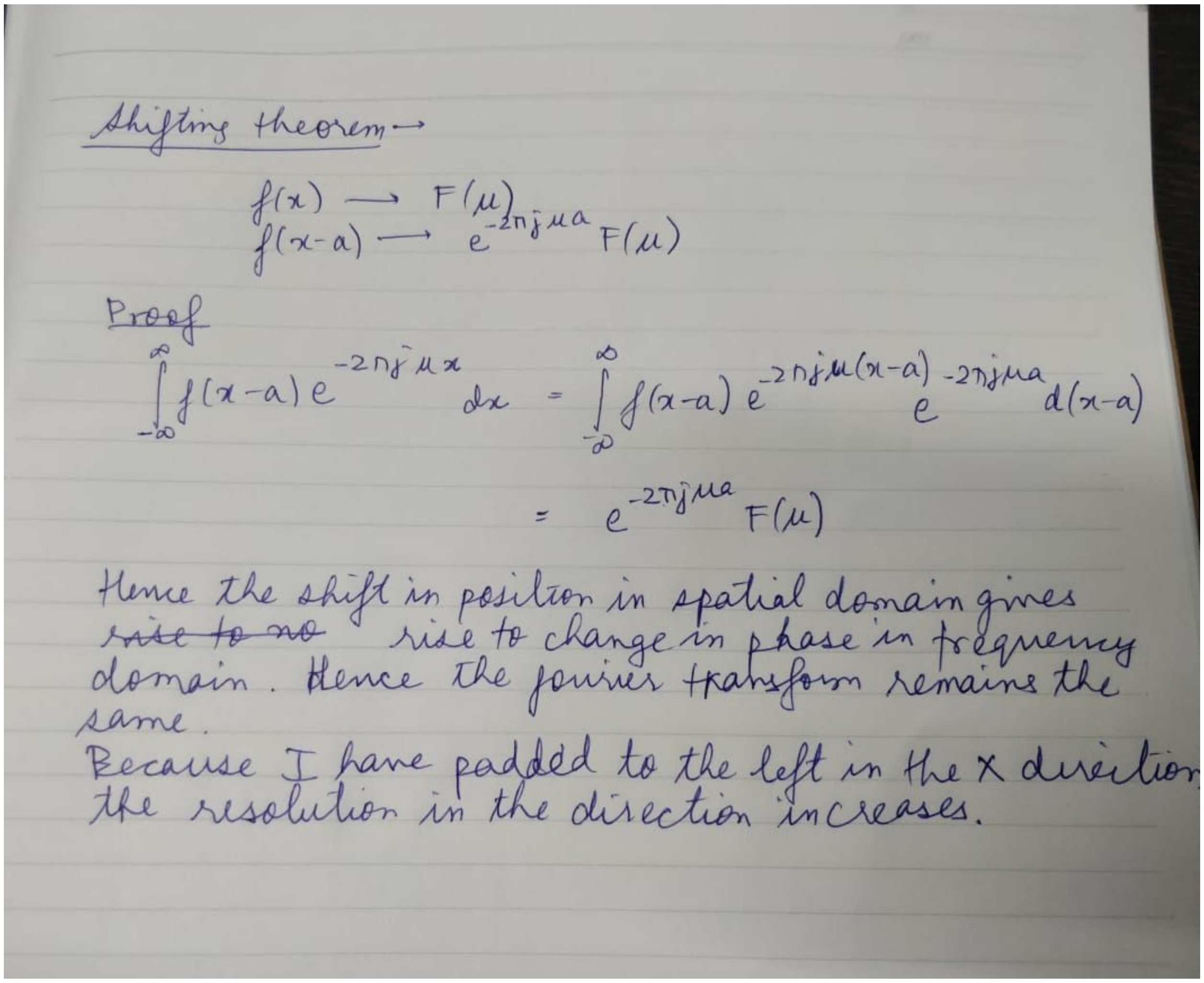
$$G(\bar{u}_x, \bar{u}_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(\bar{x}, \bar{y})\, e^{-2\pi j(\bar{x}\bar{u}_x + \bar{y}\bar{u}_y)}\, dx\, dy$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(\bar{x}, \bar{y})\, e^{-2\pi j(\bar{x}u_x\cos\theta - \bar{y}u_x\sin\theta + \bar{x}u_y\sin\theta + \bar{y}u_y\cos\theta)}\, dx\, dy$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(\bar{x}, \bar{y})\, e^{-2\pi j(\bar{x}(u_x\cos\theta + u_y\sin\theta) + \bar{y}(-u_x\sin\theta + u_y\cos\theta))}\, d\bar{x}\, d\bar{y}$$

$$= G(u_x\cos\theta + u_y\sin\theta, -u_x\sin\theta + u_y\cos\theta)$$

Hence we see that the angle that the original image is rotated by which is ~~the~~ also the angle by which fourier transform of the image is rotated.



Shifting theorem →

$$f(x) \longrightarrow F(\mu)$$
$$f(x-a) \longrightarrow e^{-2\pi j \mu a} F(\mu)$$

Proof

$$\int_{-\infty}^{\infty} f(x-a) e^{-2\pi j \mu x} dx = \int_{-\infty}^{\infty} f(x-a) e^{-2\pi j \mu (x-a)} e^{-2\pi j \mu a} d(x-a)$$

$$= e^{-2\pi j \mu a} F(\mu)$$

Hence the shift in position in spatial domain gives ~~rise to no~~ rise to change in phase in frequency domain. Hence the fourier transform remains the same.

Because I have padded to the left in the x directio the resolution in the direction increases.