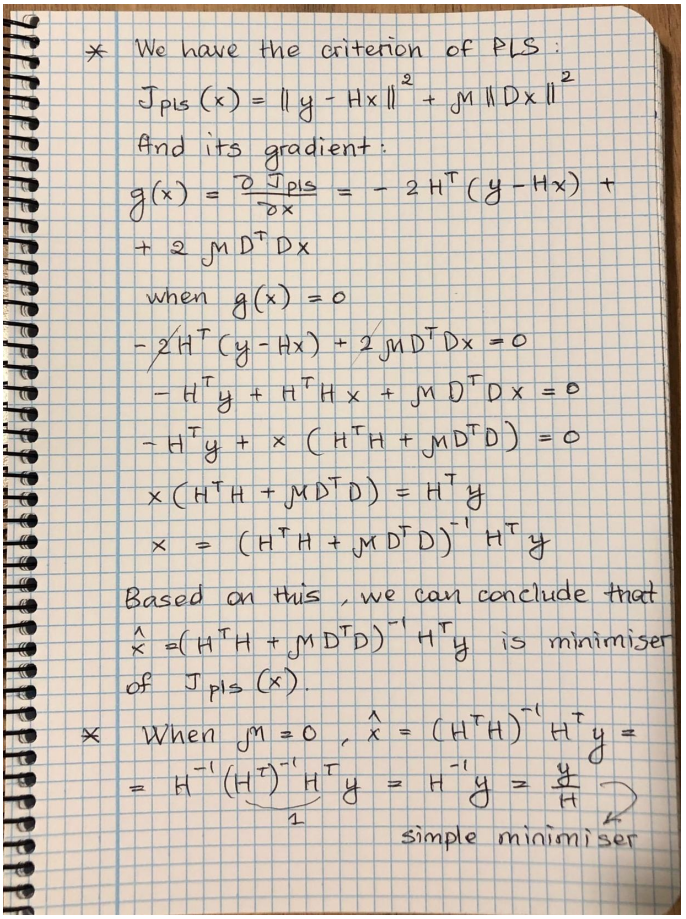
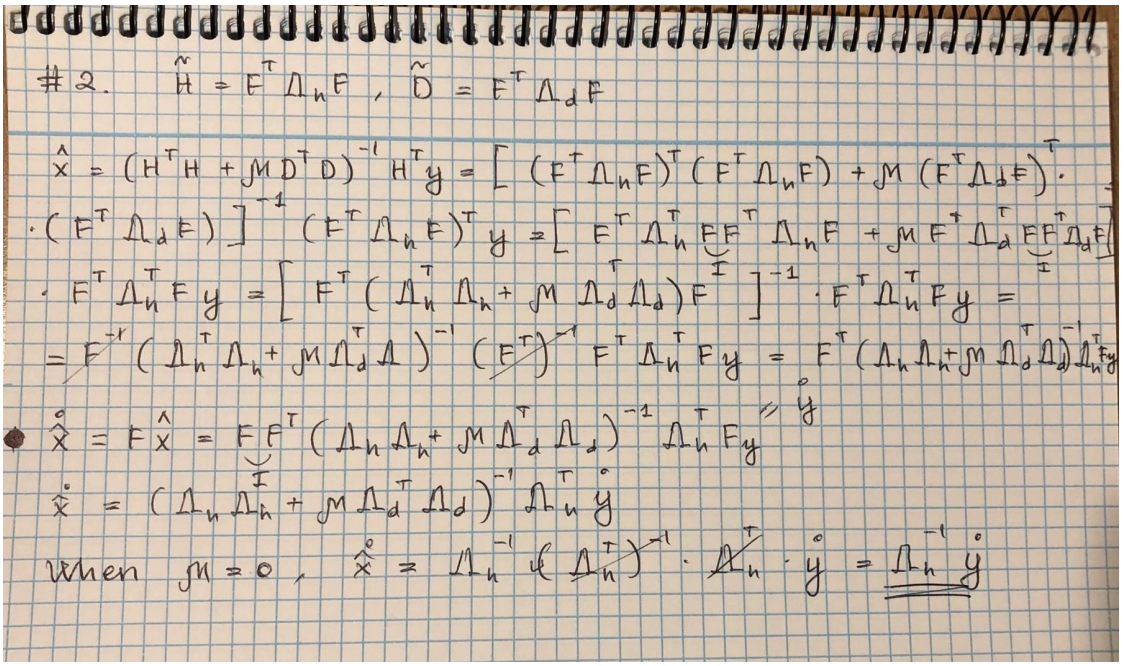
Image deconvolution. Wiener-Hunt Method

Baglan Aitu (IPCV)

Exercise 1.



Exercise 2.



Exercise 3.

We have 2 datasets: *Data 1* (Fig 1) and *Data 2* (Fig 2). They contain the following images: *true image*, *impulse response*, and *observed image*. The last image is the result of convolution of a true image with a given impulse response.

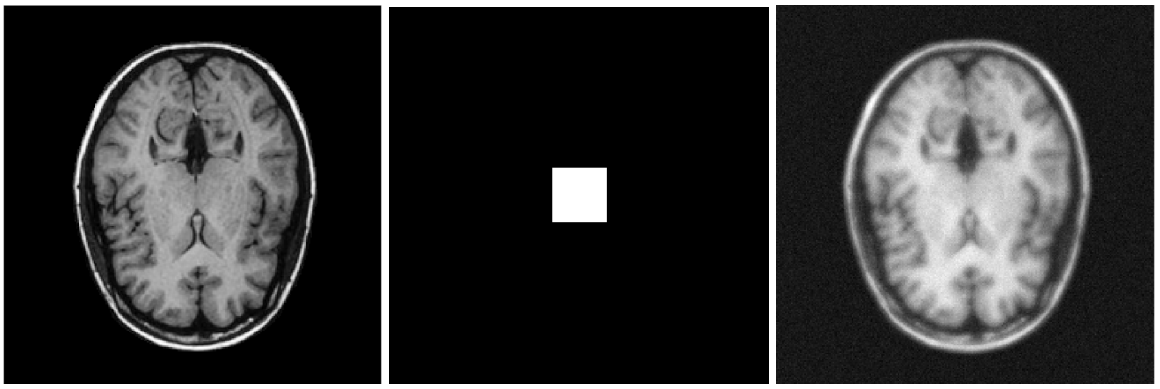


Fig 1. Data 1

Fig 2. Data 2

We can notice in both datas, the observed images don't have clear edges which shows us attenuation of high frequencies. Hence, we can conclude that given impulse responses act as a low-pass filter.

If we look closely, we can see that data 2 is a bit more blurred than data 1. Let's see it in more detail. For that we analyze the images in the frequency domain.

a) Linear scale

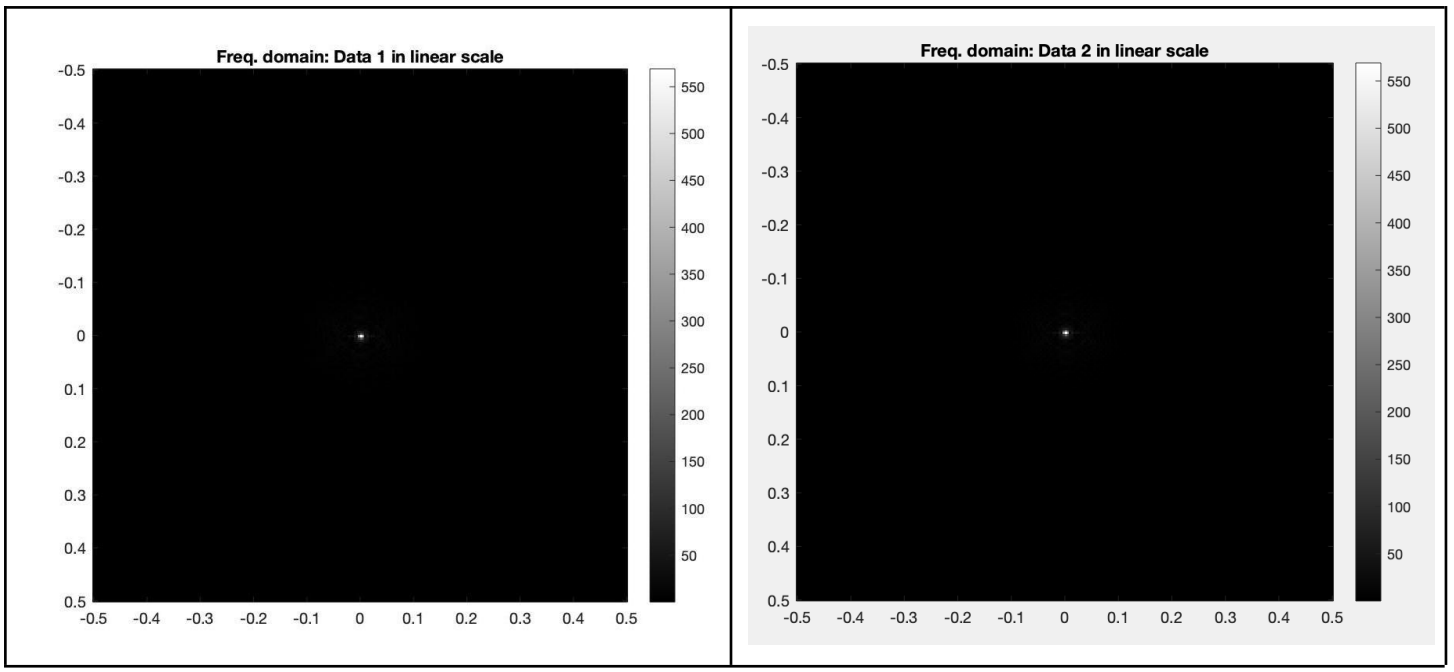


Fig 3. Datas in “linear” frequency scale

According to Fig 3, both datas have the same linear scale which shows us the same linear frequency changes. It is not accurate for estimation.

b) Logarithmic scale

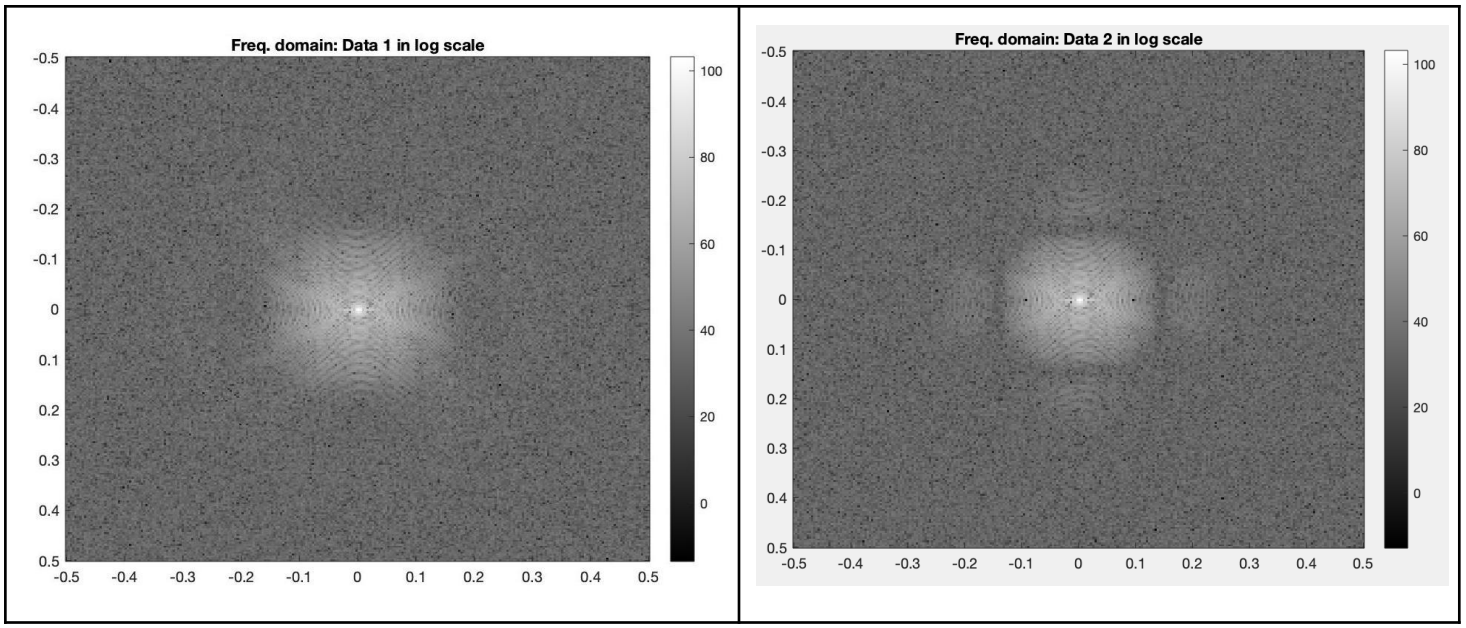


Fig 4. Datas in “logarithmic” frequency scale

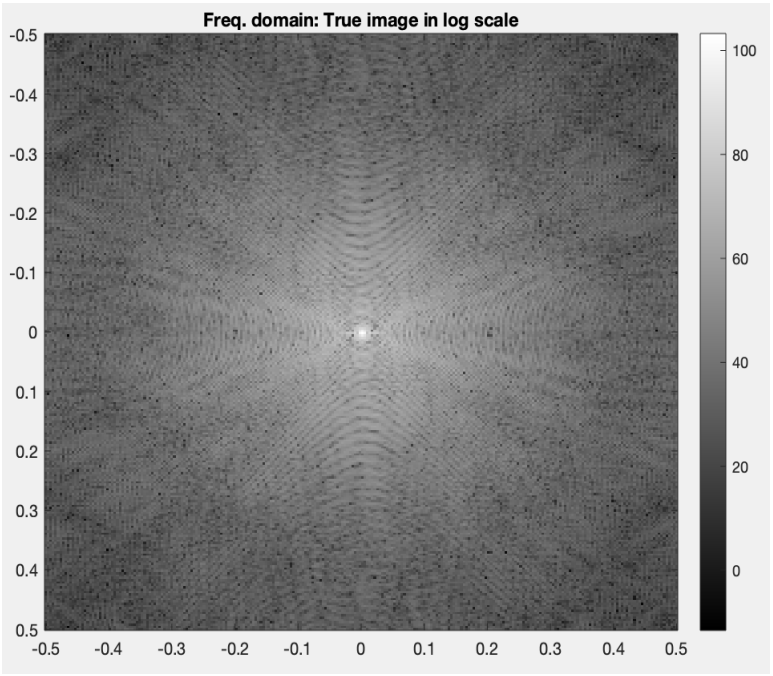
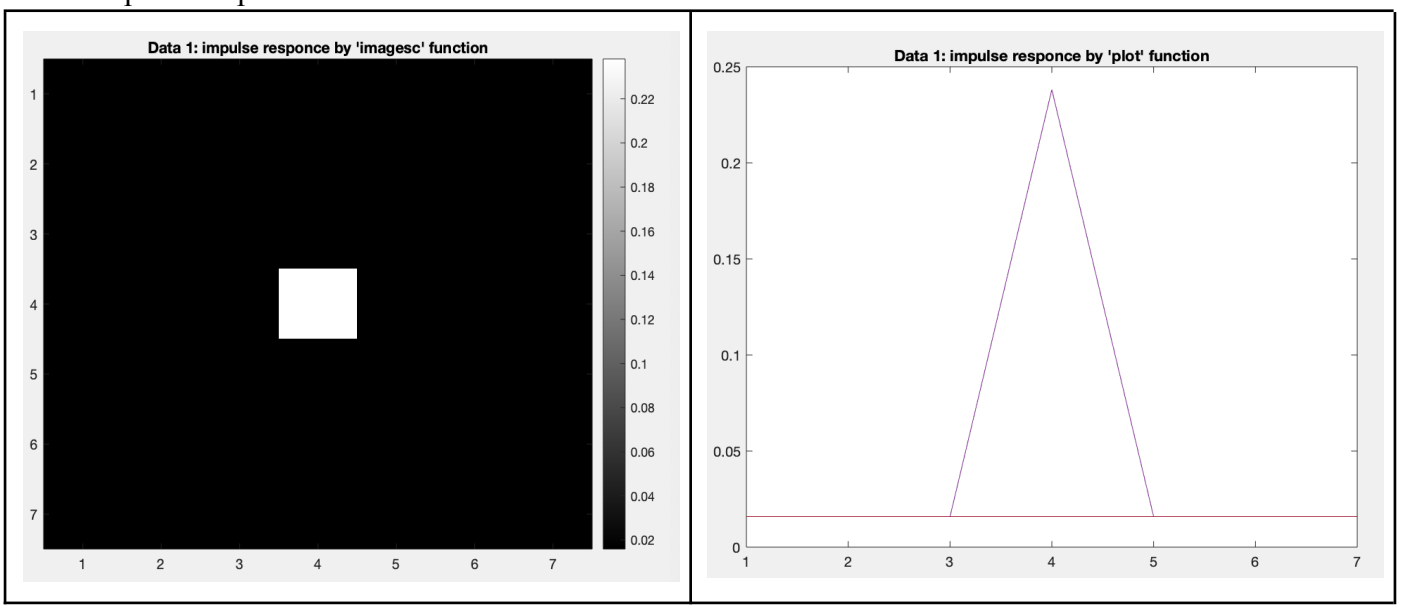


Fig 5. True Image in “logarithmic” frequency scale

By comparing the true image (Fig 5) and observed images (Fig 4) in the frequency domain, we can see that high frequencies are distributed on the edges of the graph, and low frequencies on the center. Based on this, we can assess the texture of the image determined by the radius of white circle in the center. In Fig 4, we can see that data 1 circle is wider which proves that data 2 is more blurred.

Exercise 4.

Impulse response



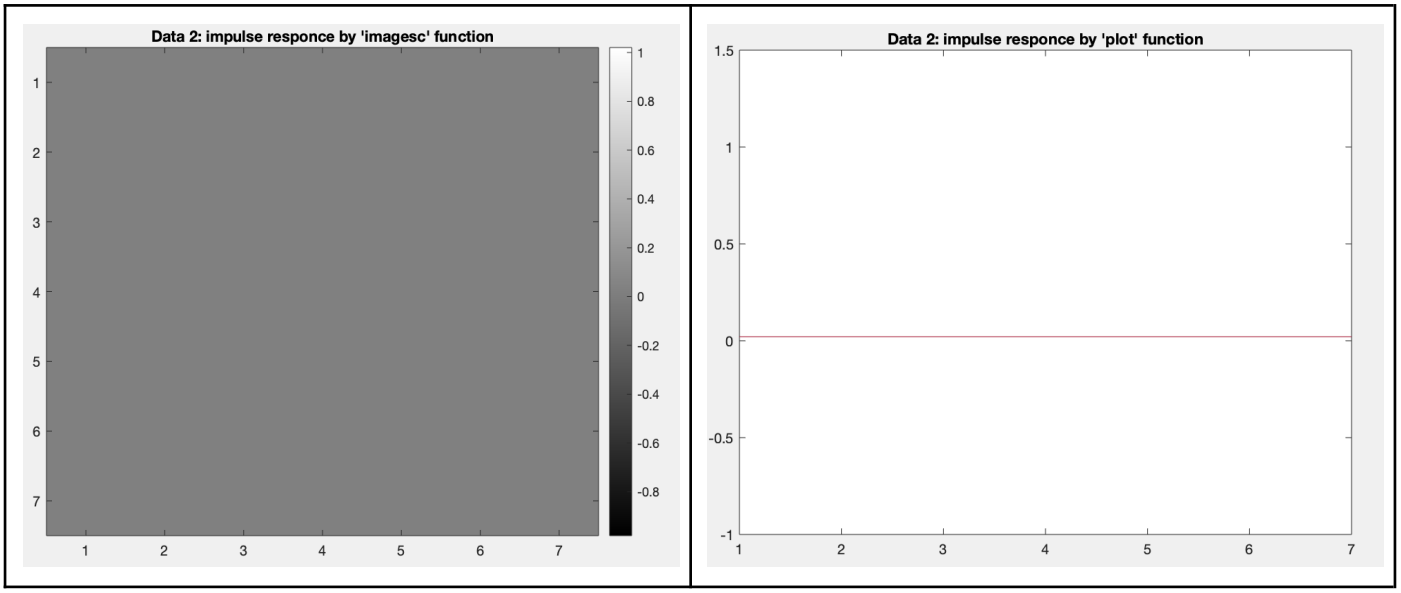


Fig 6. Impulse responses

We can see the impulse response of Data 1 leaves the values of the true image with the size of white box (Fig 6 first row first column), the rest parts are nullified. In other words, it is doing pixel averaging, and blurring the image.

Impulse response of Data 2 has the matrix of uniform numbers. According to its plot function (Fig 6, second row, second column), it is a very small number (≈ 0.05). During the convolution process, it simply divides the each pixel value, and lower them. Consequently, the intensity difference between neighbor pixels will decrease. For example, we have two numbers: 100 and 20. The difference is 100-20=80. After dividing by 4, numbers: 20 and 5. The difference is 20-5=15. It is much lower than the previous difference (80>15). As we know, high frequency of the image (texture) means high difference between neighbor pixel intensities. Since it is lowered, we lose the sharpness of the image.

According to the above assumptions and previous task examples, both impulse responses are low-pass filters.

The difference between them is that IR of Data 1 does averaging, and IR of Data 2 decreases the intensity of all pixels uniformly.

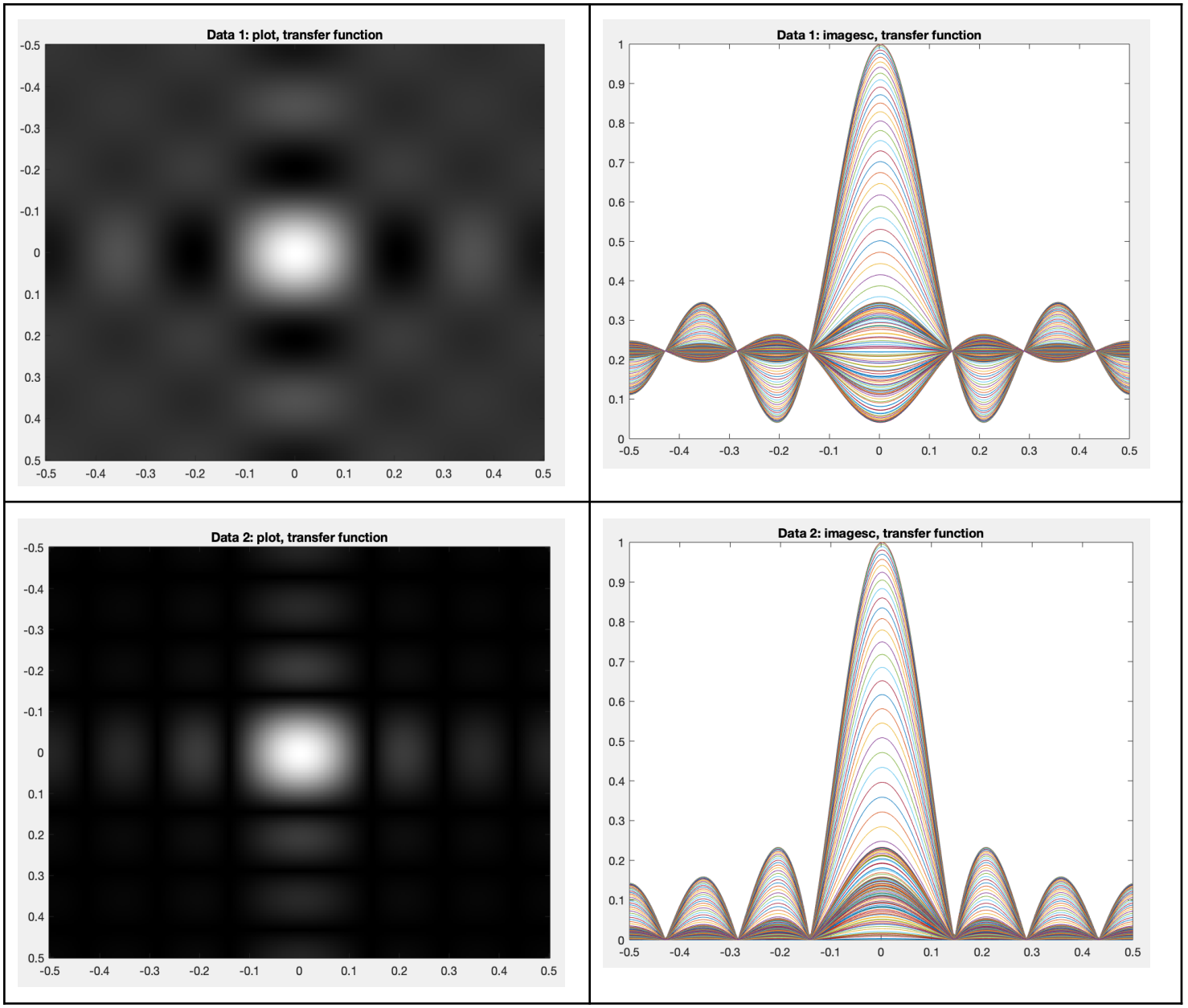


Fig 7. Transfer functions of impulse response by *plot* and *imagesc*

Exercise 5.
Deconvolution function:

```
function out = deconv(Data, IR, d0, mu)

% Output
h0 = MyFFT2RI(IR, 256);
y0 = MyFFT2(Data);
inv = MyIFFT2(y0./h0);

% Wiener filter
WF = conj(h0) ./ (abs(h0).^2 + mu*d0);
deconv_res = MyIFFT2(WF.*y0);
out = deconv_res;
end
```

Running the code:

```
clc, close all, clear all
load DataOne
%load DataTwo

% Filters
f1 = [0 0 0; 0 -1 1; 0 0 0];
f2 = [0 0 0; 0 -1 0; 0 1 0];

f3 = [0 -1 0; -1 4 -1; 0 -1 0];
f4 = [-1 -1 -1; -1 8 -1; -1 -1 -1];
f5 = [1 -2 1; -1 4 -1; 0 -1 0];

% regulizer
FD1 = MyFFT2RI(f1, 256);
FD2 = MyFFT2RI(f2, 256);
d0 = abs(FD1).^2 + abs(FD2).^2;

% deconvolution
mu = 0.001;
deconv_res = deconv(Data, IR, d0, mu);
imagesc(deconv_res);
colormap('gray');colorbar
axis('square','off')
```

We can also use the filters f3, f4, f5. In that case, regulizers d0 will be a squared sum of three of them. The obtained result is the same with filters f1, f2.

Exercise 6.

Mu = 0

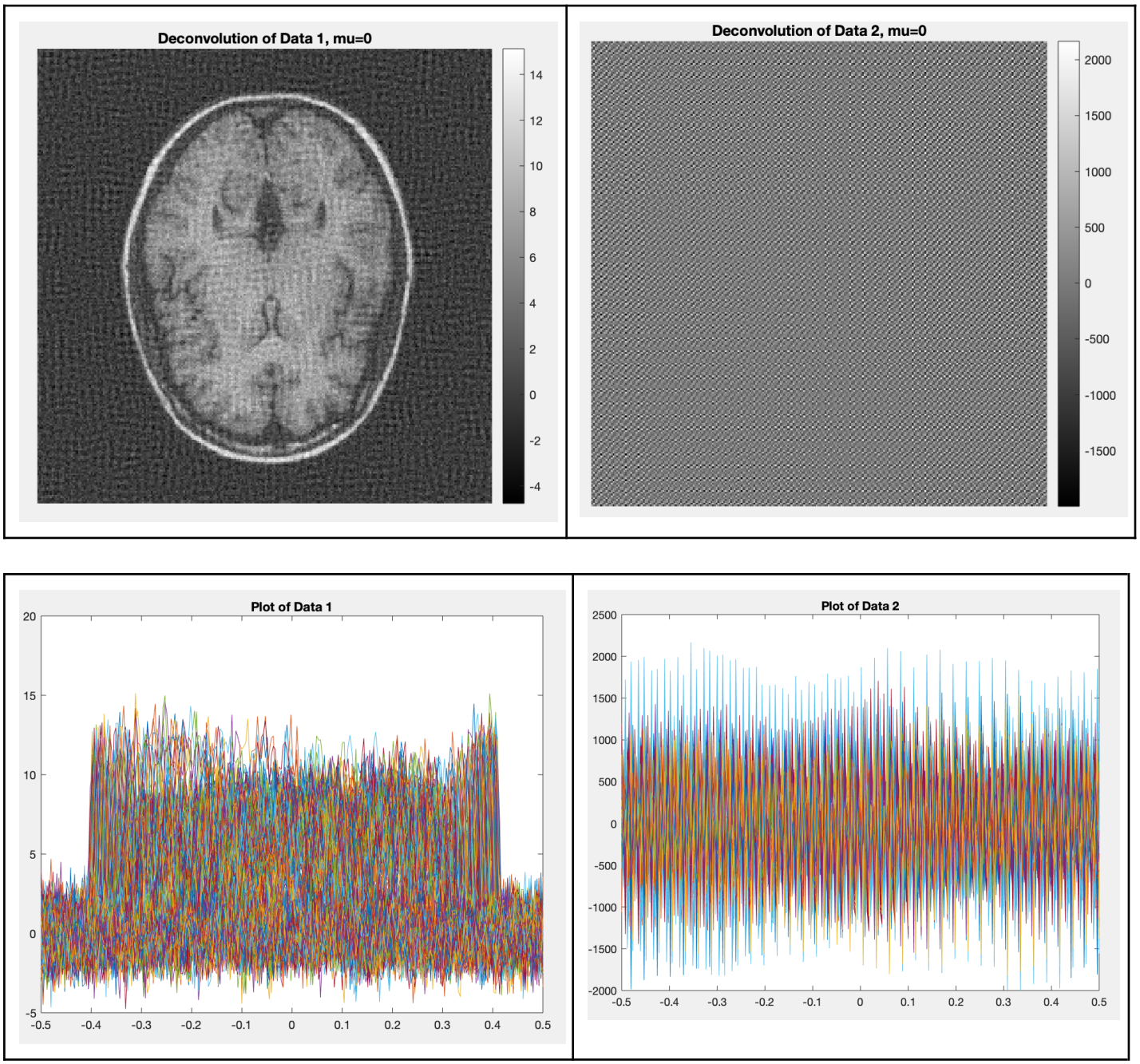
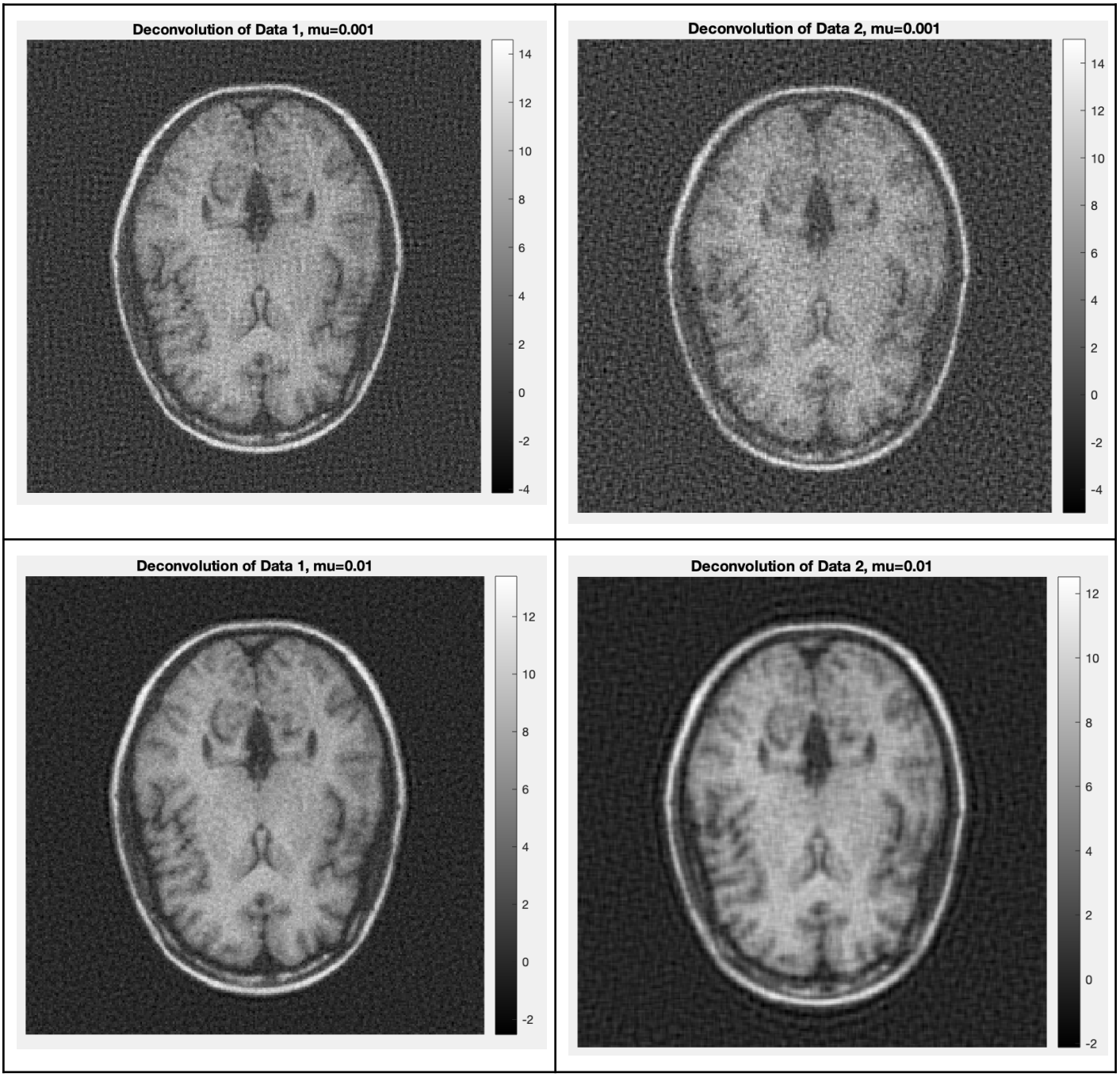
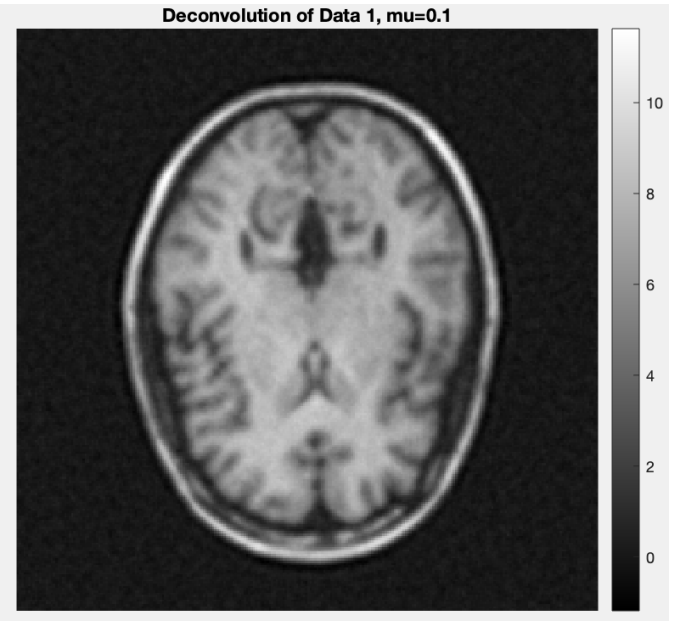


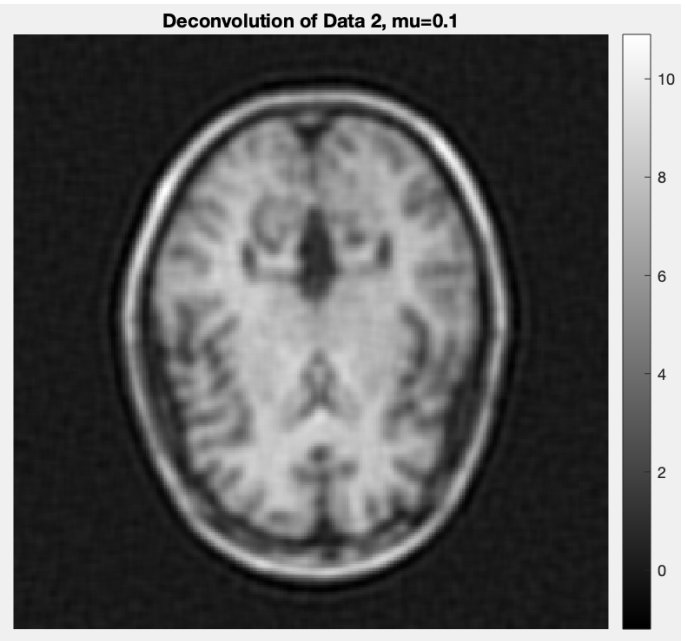
Fig 8. Deconvolution with mu=0

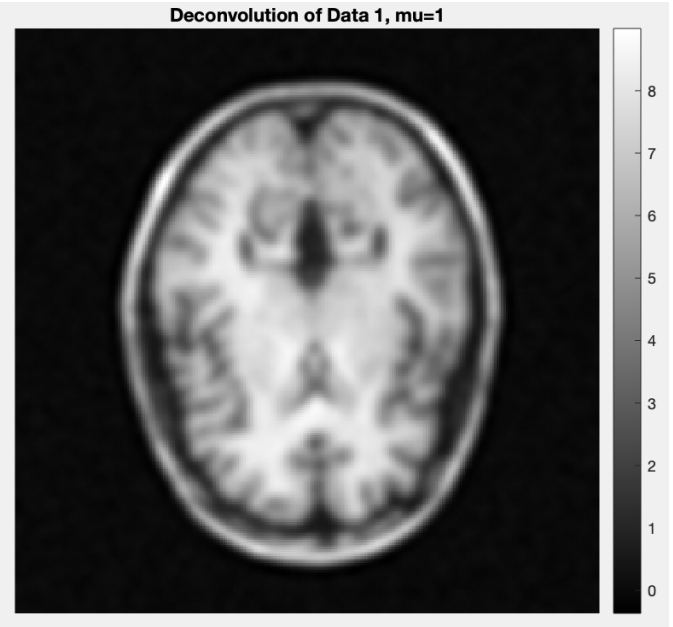
As we can see in Fig 8, deconvolution with mu=0, shows us the result of penalized least square method (Wiener filter) without regularizer which is a simple LS method (ref: exercise 1). So there is no bias. Result of data 2 is unacceptable, since observed data has relatively small intensity difference of neighbor pixel values because of applied impulse response. The result of data 1 is sharp, but it is still noisy. In order to improve the result, we need to tune the mu.

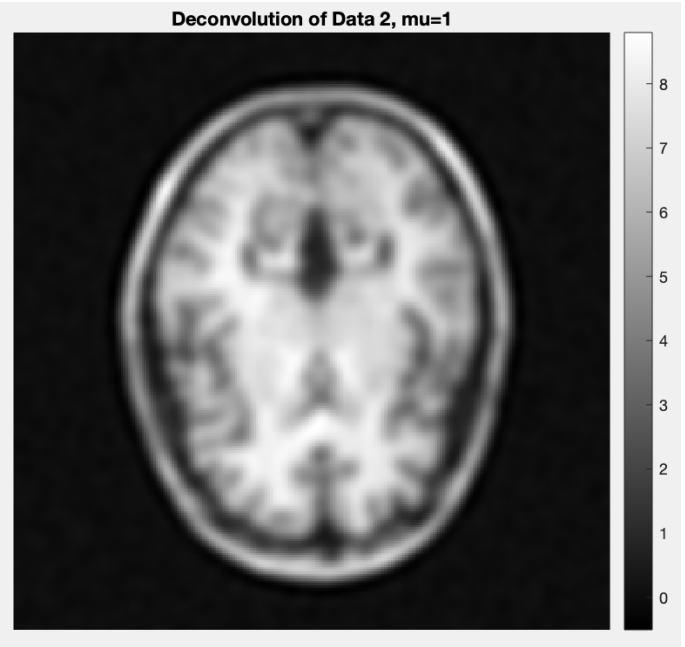
Exercise 7.


Deconvolution of Data 1, mu=0.001
14
12
10
8
6
4
2
0
-2
-4
Deconvolution of Data 2, mu=0.001
14
12
10
8
6
4
2
0
-2
-4
Deconvolution of Data 1, mu=0.01
12
10
8
6
4
2
0
-2
Deconvolution of Data 2, mu=0.01
12
10
8
6
4
2
0
-2


Deconvolution of Data 1, mu=0.1
10
8
6
4
2
0


Deconvolution of Data 2, mu=0.1
10
8
6
4
2
0


Deconvolution of Data 1, mu=1
8
7
6
5
4
3
2
1
0


Deconvolution of Data 2, mu=1
8
7
6
5
4
3
2
1
0

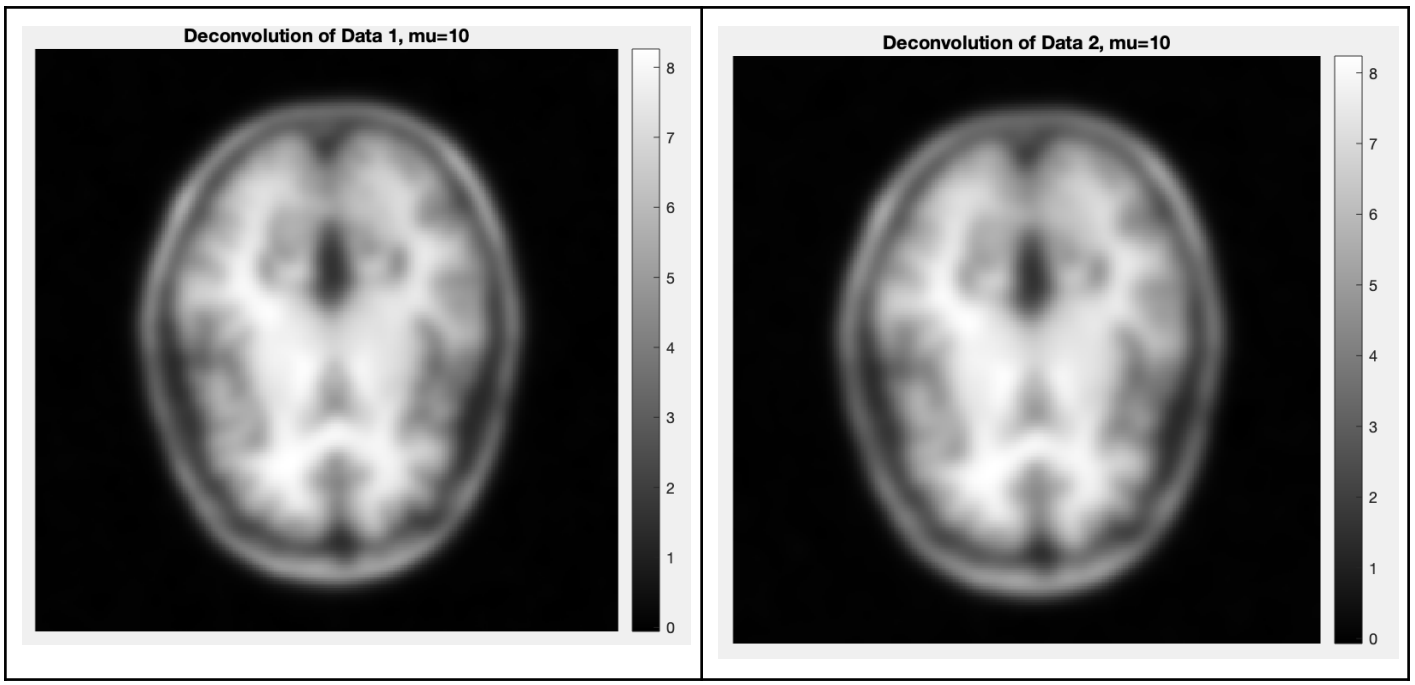
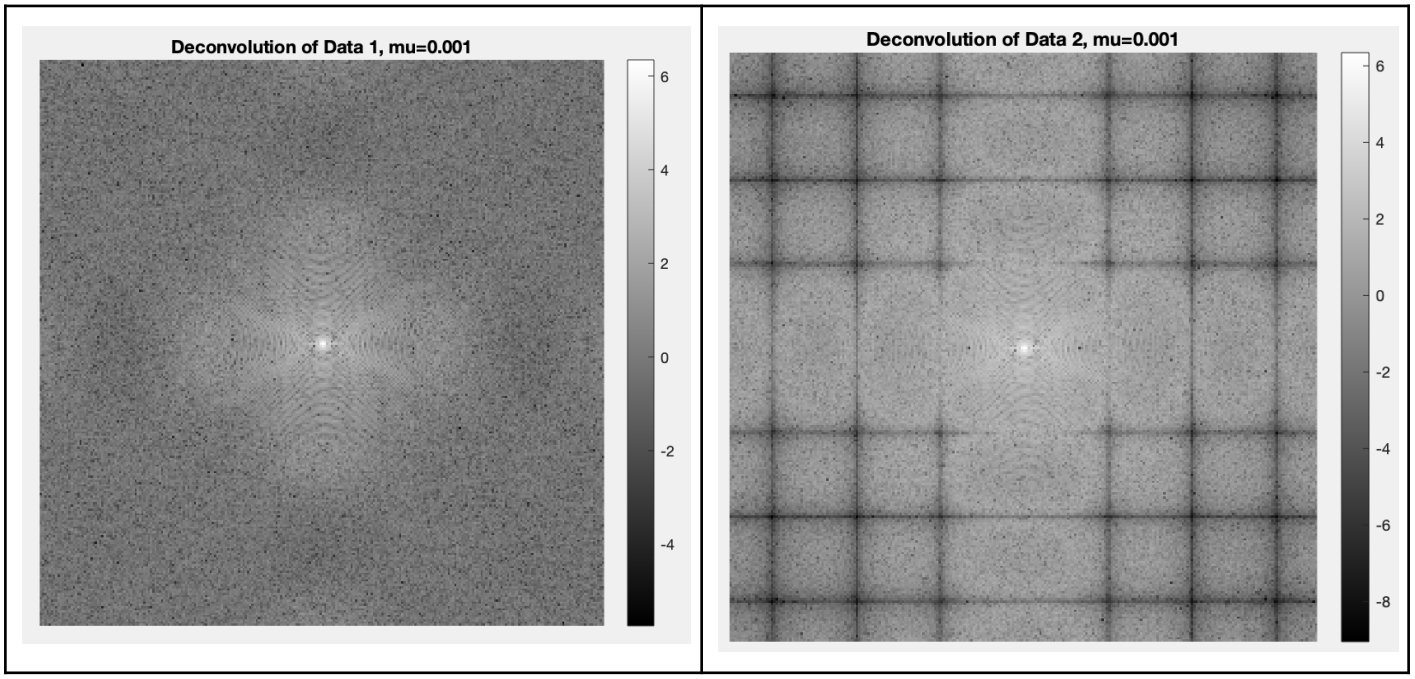
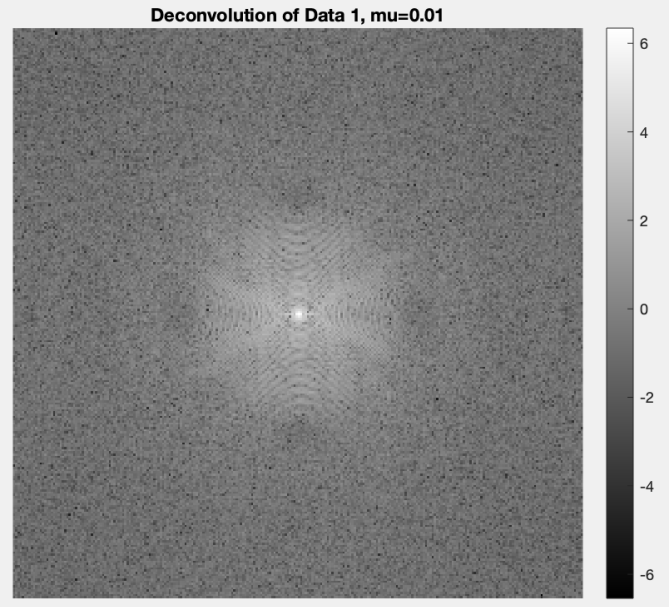
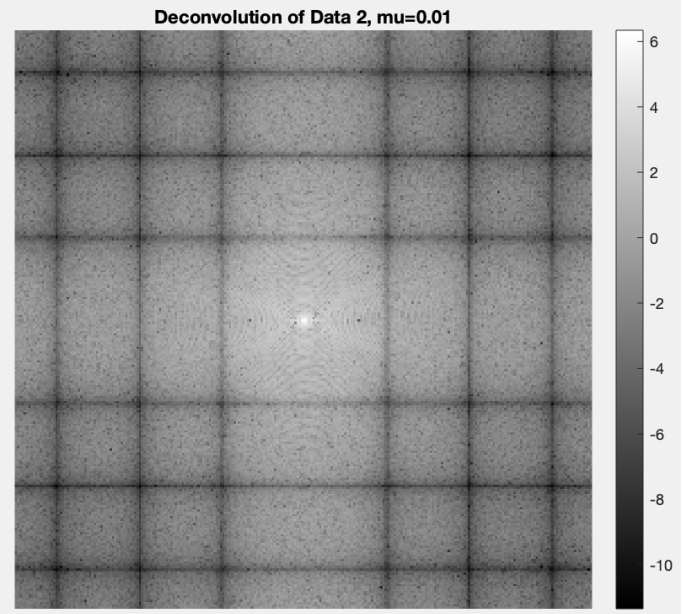
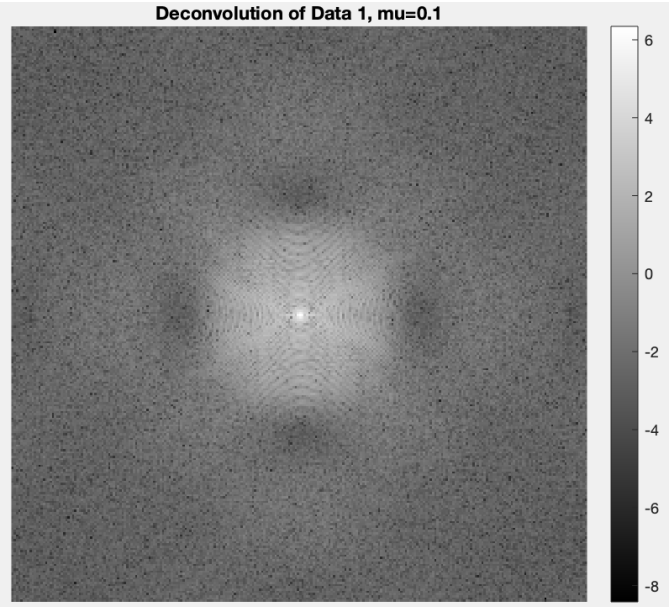


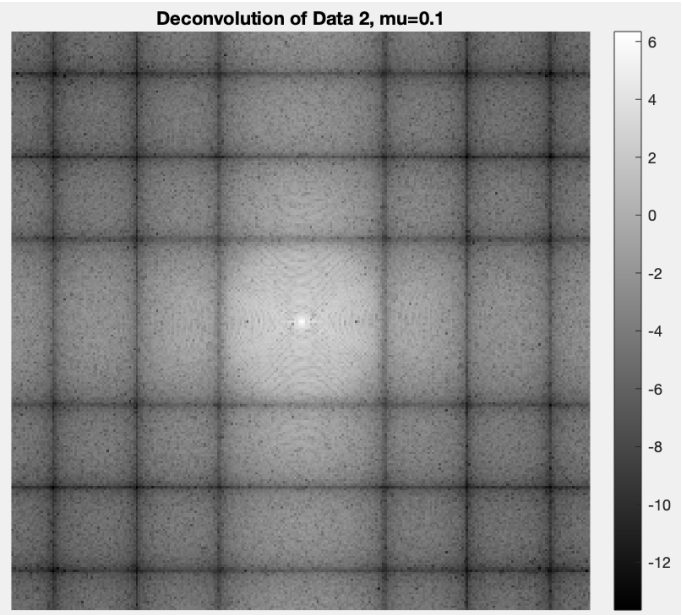
Fig 9. Deconvolution with mu=0.001:10 (Spatial domain)




Deconvolution of Data 1, mu=0.01
6
4
2
0
-2
-4
-6


Deconvolution of Data 2, mu=0.01
6
4
2
0
-2
-4
-6
-8
-10


Deconvolution of Data 1, mu=0.1
6
4
2
0
-2
-4
-6
-8


Deconvolution of Data 2, mu=0.1
6
4
2
0
-2
-4
-6
-8
-10
-12

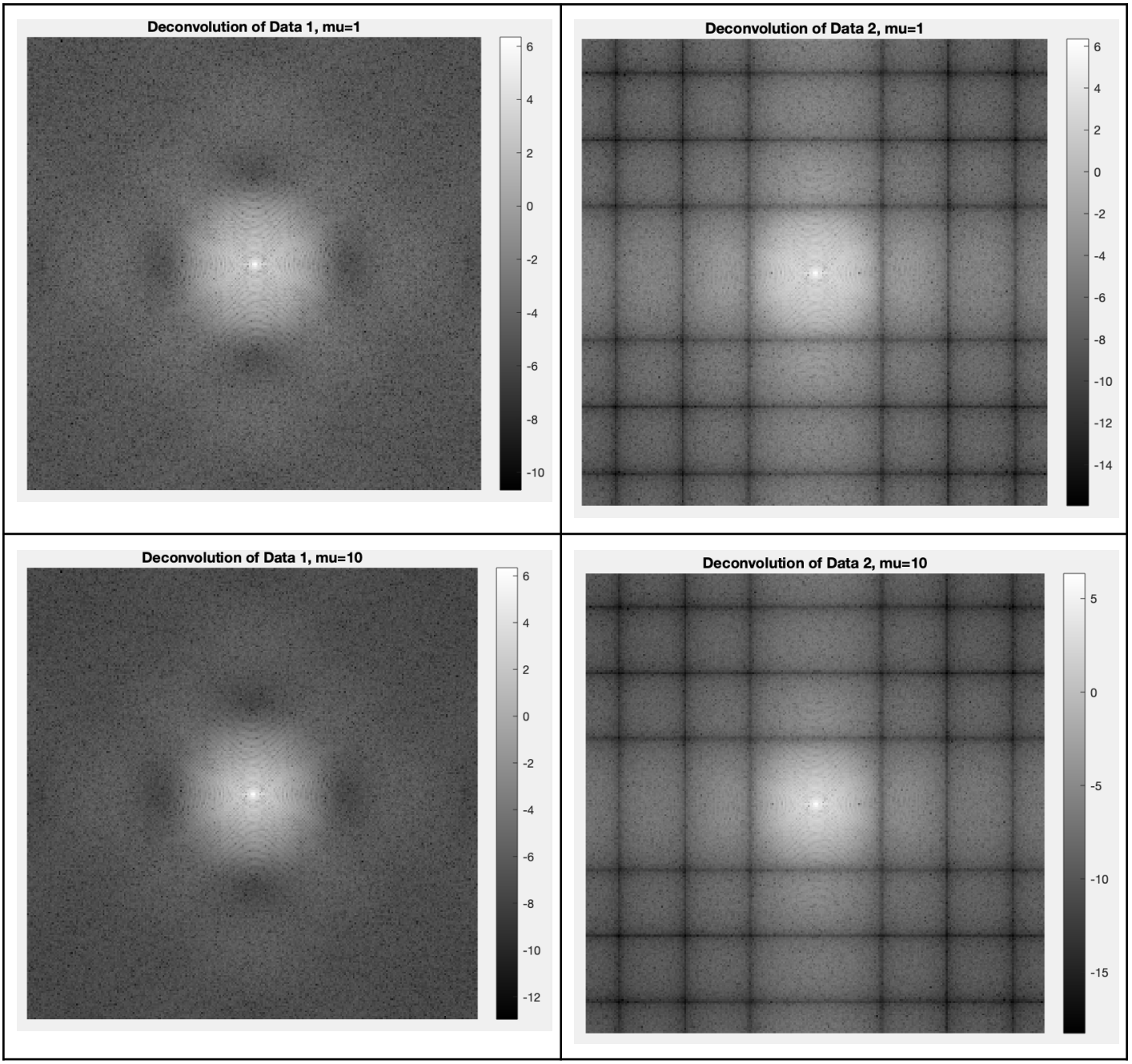


Fig 9. Deconvolution with mu=0.001:10 (Frequency domain)

According to Fig 8-9, by increasing the value of mu, reconstructed images get more smooth. It is due to the fact that mu coefficient tries to make neighbor pixels similar (less difference in intensity). Too small mu (mu=0.001) gives sharp images with noise, and too big (mu=10) is too blurry. From the visual examination, the optimal value of mu seems to lie between 0.01 and 0.1.

Exercise 8.

mu	d1	d2	d3
1,00E-10	35.209	78.2006	52.8683
1,00E-09	16.873,	40.3480	35.8102
1,00E-08	7.434	17.7084	21.0345
1,00E-07	2.635	8.1230	10.8424
1,00E-06	1.103	3.7790	5.4594
1,00E-05	0.411	1.6131	2.4603
1,00E-04	0.156	0.6450	0.9652
1,00E-03	0.058	0.2207	0.3423
1,00E-02	0.053	0.1335	0.2709
1,00E-01	0.074	0.1588	0.3746
1,00E+00	0.093	0.1943	0.5012
1,00E+01	0.122	0.2488	0.6550
1,00E+02	0.163	0.3458	0.7591
1,00E+03	0.351	0.4607	0.7949
1,00E+04	0.619	0.7005	0.8099
1,00E+05	0.688	0.7573	0.8141
1,00E+06	0.696	0.7638	0.8146
1,00E+07	0.697	0.7644	0.8146
1,00E+08	0.697	0.7645	0.8146
1,00E+09	0.697	0.7645	0.8146
1,00E+10	0.697	0.7645	0.8146

Table 1

Distances d1, d2, d3 estimate the error between true image and observed image. According to table 1, the optimum value of mu is when mu is around 0.01 which has the minimum distances.