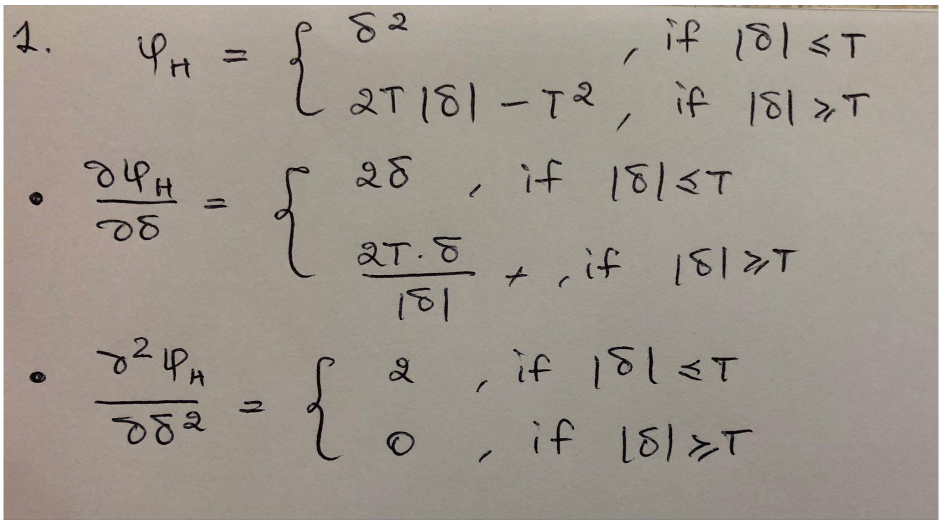
Convex regularization and contour aware image restoration

Baglan Aitu (IPCV)

Task 1.

a) Mathematically

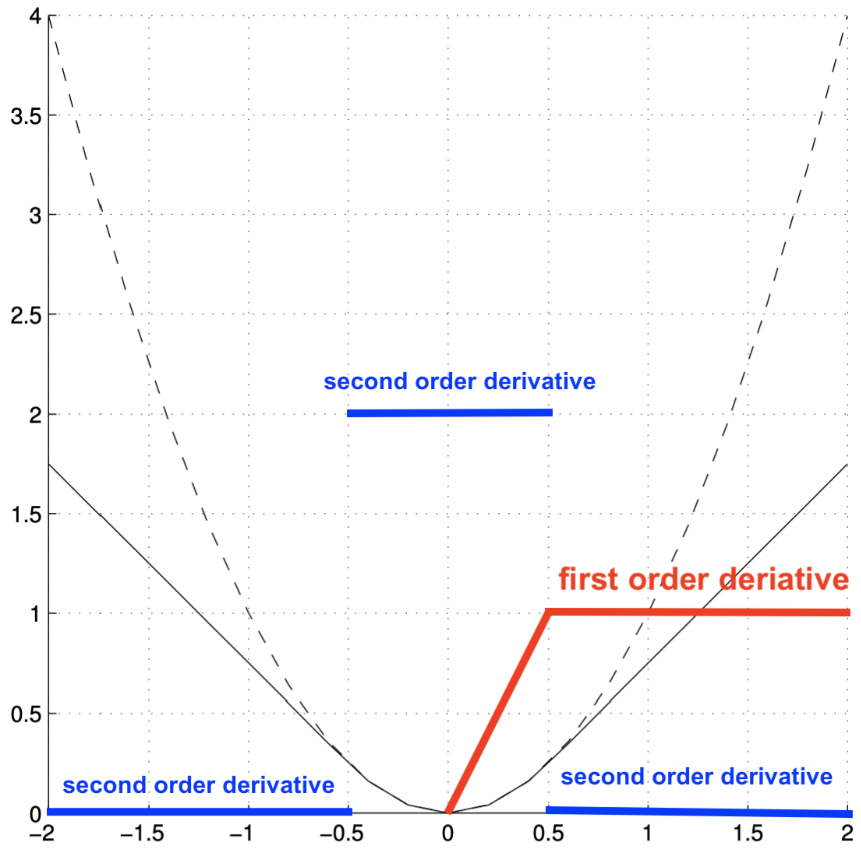


1. $$\varphi_H = \begin{cases} \delta^2 , & \text{if } |\delta| \leq T \\ 2T|\delta| - T^2 , & \text{if } |\delta| \geq T \end{cases}$$

- $$\frac{\partial \varphi_H}{\partial \delta} = \begin{cases} 2\delta , & \text{if } |\delta| \leq T \\ \frac{2T \cdot \delta}{|\delta|} , & \text{if } |\delta| \geq T \end{cases}$$

- $$\frac{\partial^2 \varphi_H}{\partial \delta^2} = \begin{cases} 2 , & \text{if } |\delta| \leq T \\ 0 , & \text{if } |\delta| \geq T \end{cases}$$

b) Theoretically



Task 2.

The problem of previously considered functions were they get more smooth in higher number of iterations and hyperparameter values, after which image loses fine details (high frequencies) and gets blurry. The reason for that was potentially penalizing the pixel difference in square manner. Since we need more smoothes in homogenous regions, and less in edges, a non-quadratic function was added. The idea is to act quadratically in low frequency regions (as Wiener-Hunt) and act non-quadratically in high frequency regions (Fig 1).

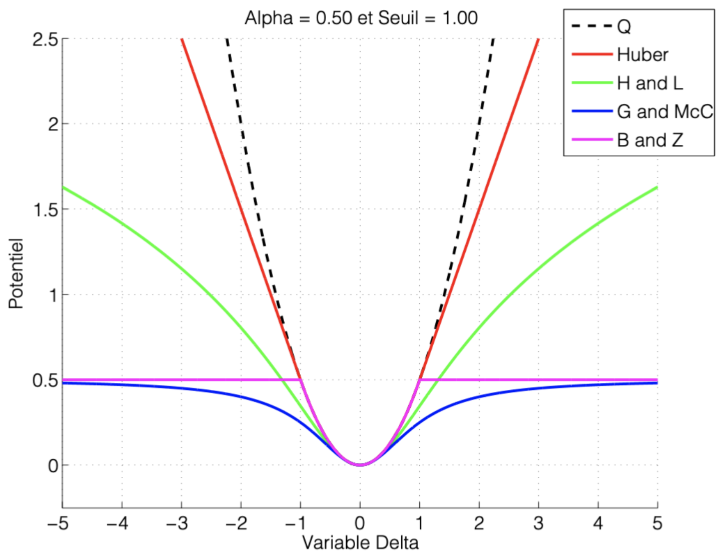
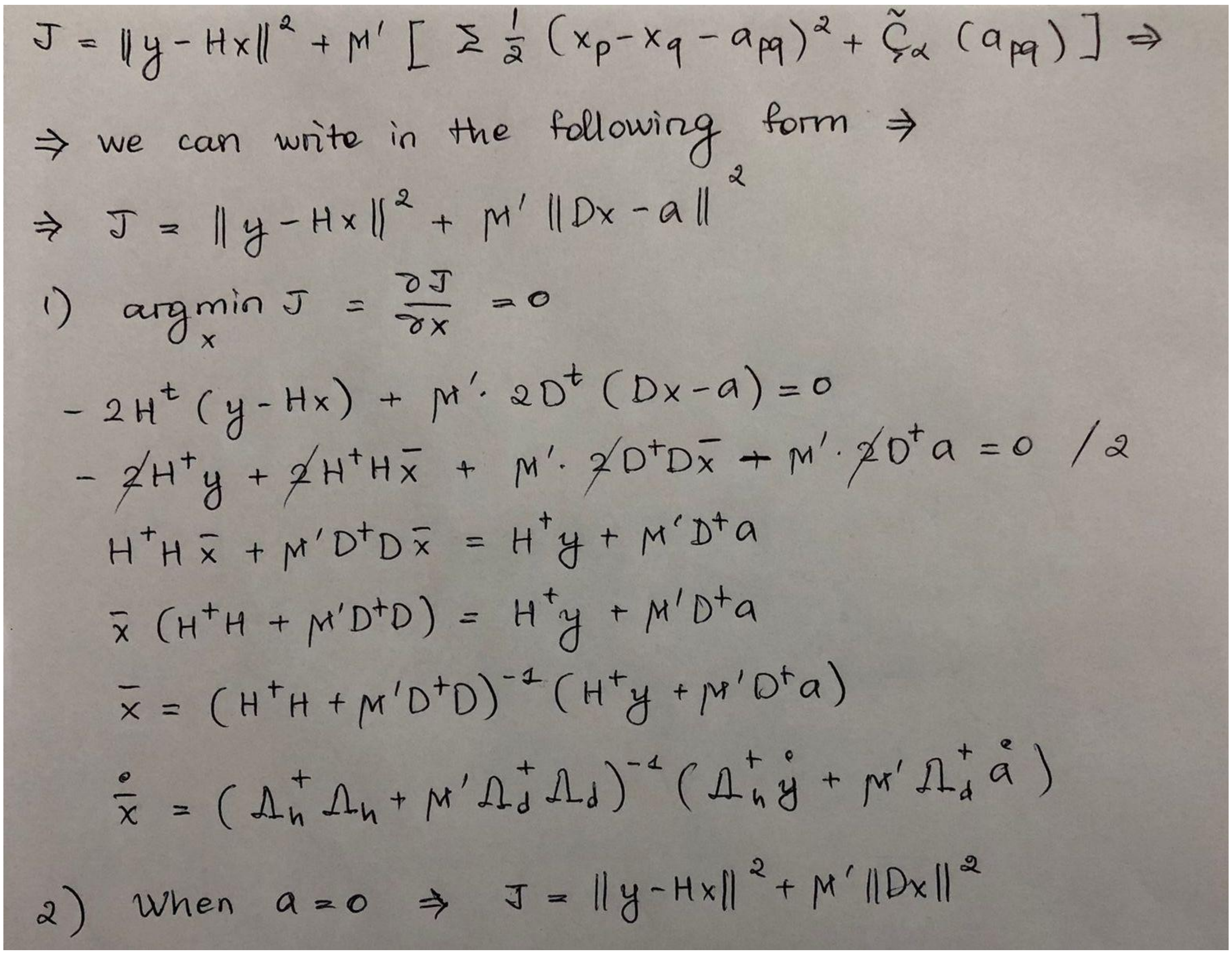


Fig 1. Non-quadratic penalization

In this lab, we are dealing with Huber penalization (Fig 1, red line). After threshold T, it starts to act linearly which enables more edge preservation. According to Fig 1, we can conclude that T values control the penalization term. To be exact, the higher values of T leads to more quadratic penalization effects. The more T we have, the more similar the deconvolution result with Wiener-Hunt we will get.

Task 4.



$$J = \|y - Hx\|^2 + \mu' \left[\sum \frac{1}{2} (x_p - x_q - a_{pq})^2 + \tilde{Ç}_{\alpha} (a_{pq}) \right] \Rightarrow$$

$\Rightarrow$ we can write in the following form $\Rightarrow$

$$\Rightarrow J = \|y - Hx\|^2 + \mu' \|Dx - a\|^2$$

1) $\underset{x}{\operatorname{argmin}} J = \frac{\partial J}{\partial x} = 0$

$$-2H^t (y - Hx) + \mu' \cdot 2D^t (Dx - a) = 0$$

$$-\cancel{2}H^+ y + \cancel{2}H^+ H \bar{x} + \mu' \cdot \cancel{2} D^+ D \bar{x} - \mu' \cdot \cancel{2} D^+ a = 0 \quad /2$$

$$H^+ H \bar{x} + \mu' D^+ D \bar{x} = H^+ y + \mu' D^+ a$$

$$\bar{x} (H^+ H + \mu' D^+ D) = H^+ y + \mu' D^+ a$$

$$\bar{x} = (H^+ H + \mu' D^+ D)^{-1} (H^+ y + \mu' D^+ a)$$

$$\mathring{\bar{x}} = (\Lambda_h^+ \Lambda_h + \mu' \Lambda_d^+ \Lambda_d)^{-1} (\Lambda_h^+ \mathring{y} + \mu' \Lambda_d^+ \mathring{a})$$

2) When $a = 0 \Rightarrow J = \|y - Hx\|^2 + \mu' \|Dx\|^2$

Task 5.

a. Since a_pq is independent of each other, their derivatives can be calculated in parallel to each other.

$$J = \|y - Hx\|^2 + \mu' \left[\sum \frac{1}{2} (x_p - x_q - a_{pq})^2 + \tilde{\xi}(a_{pq}) \right]$$

- $\frac{1}{2} \sum a_{pq}^2$

$$\Rightarrow \frac{1}{2} \nabla \sum a_{pq}^2 = \frac{1}{2} \begin{bmatrix} \frac{\partial}{\partial a_1} \sum a_{pq}^2 \\ \frac{\partial}{\partial a_2} \sum a_{pq}^2 \\ \vdots \\ \frac{\partial}{\partial a_n} \sum a_{pq}^2 \end{bmatrix} = \begin{bmatrix} a_{12} \\ a_{23} \\ \vdots \\ a_{n-1} a_n \end{bmatrix}$$

b.

$$a\varphi(\delta) = \begin{cases} \inf\limits_a \left[\frac{1}{2} (\delta - a)^2 + \frac{a^2 \cdot \alpha}{1 - 2\alpha} \right. , & \text{if } |a| \leq (1-2\alpha)T \\ \inf\limits_a \left[\frac{1}{2} (\delta - a)^2 + \alpha \left[2T|a| - (1-2\alpha)T^2 \right] \right], & \text{if } |a| \geq (1-2\alpha)T \end{cases}$$

$$\frac{\partial a\varphi(\delta)}{\partial a} = 0 = \begin{cases} a - \delta + \frac{2a\alpha}{1-2\alpha}, & \text{if } |a| \leq (1-2\alpha)T \\ a - \delta + 2\alpha|T|, & \text{if } |a| \geq (1-2\alpha)T \end{cases}$$

1) $a - \delta + \frac{2a\alpha}{1-2\alpha} = 0$

$\bar{a}\left(1 + \frac{2\alpha}{1-2\alpha}\right) = \delta$

$\bar{a} = (1-2\alpha)\delta$

if $|\delta| \not\leq T$

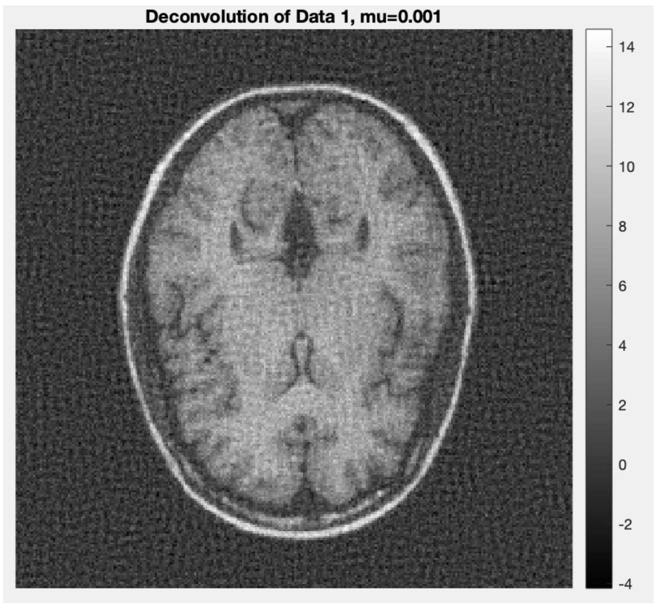
2) $a - \delta + 2\alpha|T| = 0$

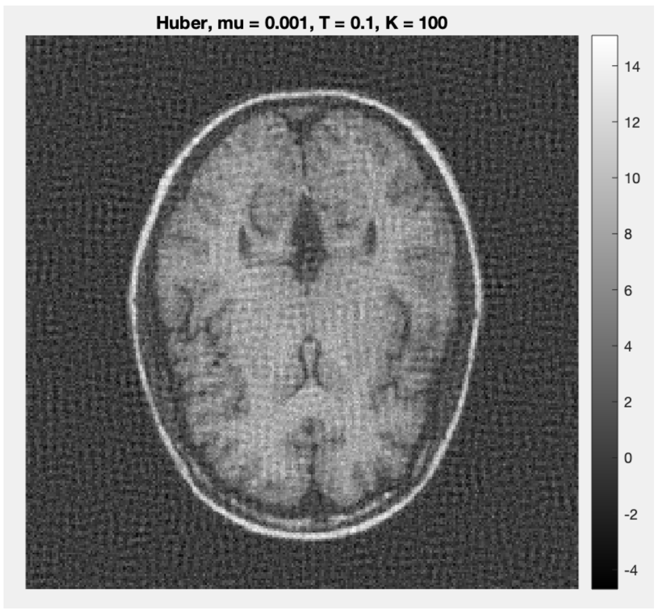
$\bar{a} = \delta - 2\alpha|T| \Rightarrow$

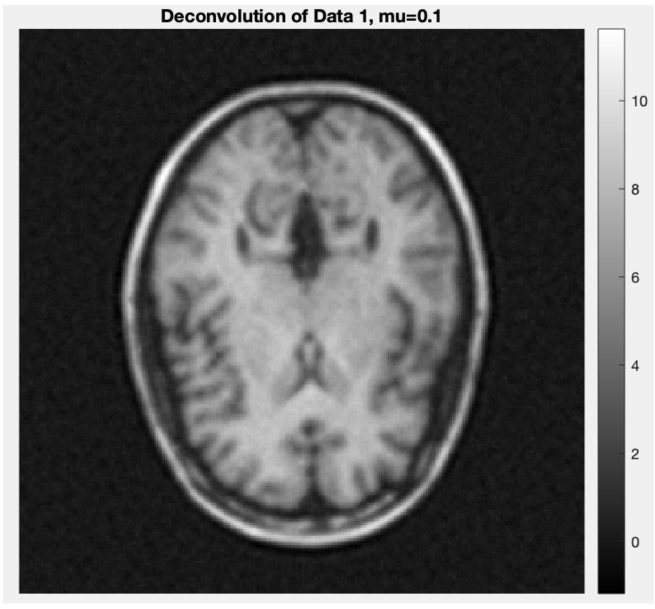
$$\Rightarrow \begin{cases} \delta - 2\alpha T, & \text{if } \delta \geq T \\ \delta + 2\alpha T, & \text{if } \delta \leq T \end{cases}$$

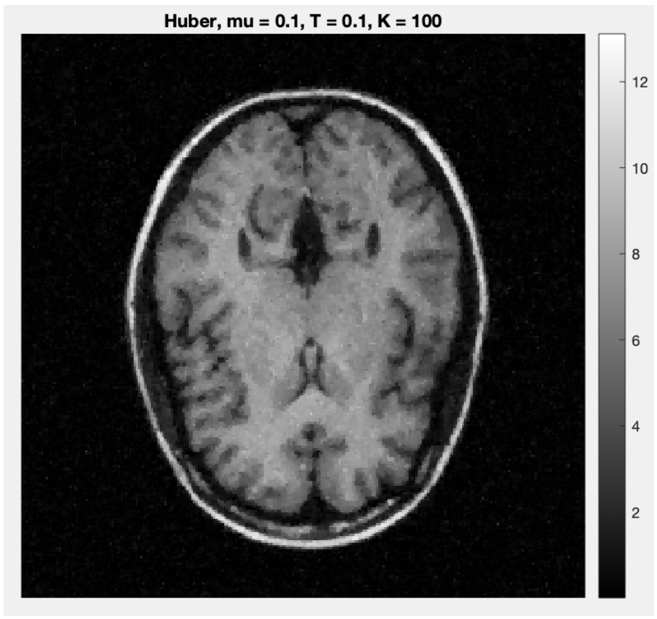
Task 6.

a. Influence of *mu*









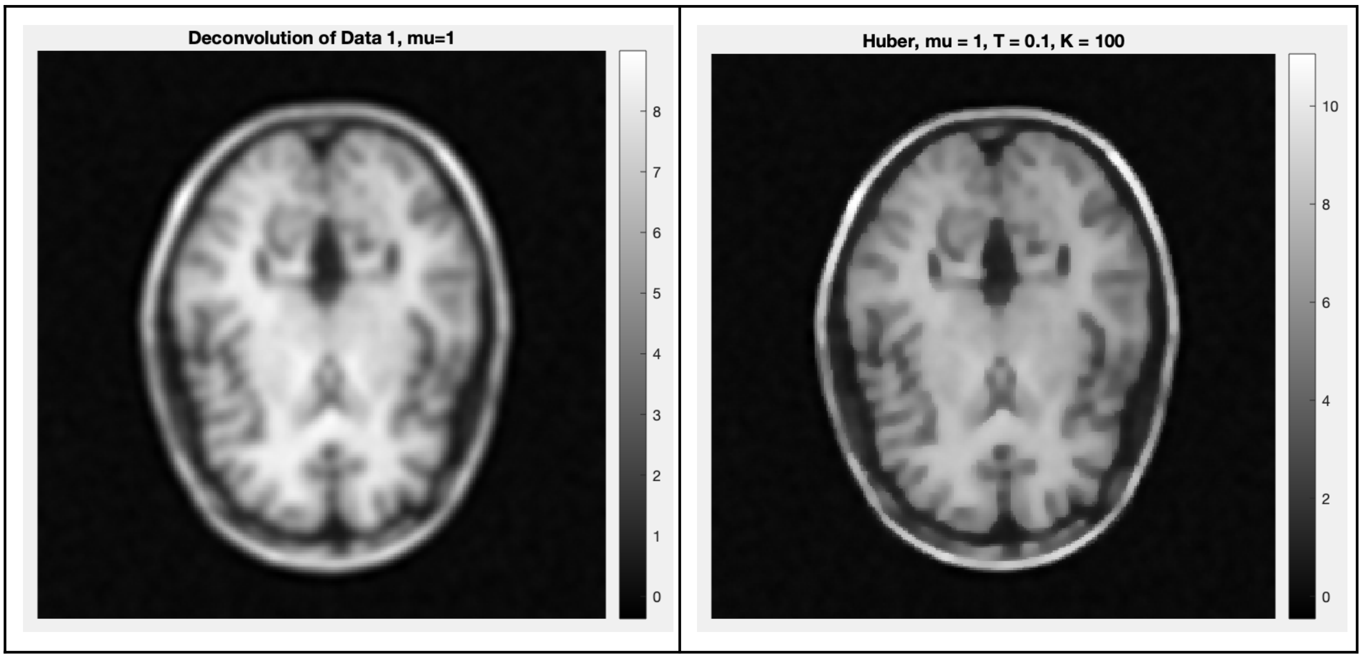
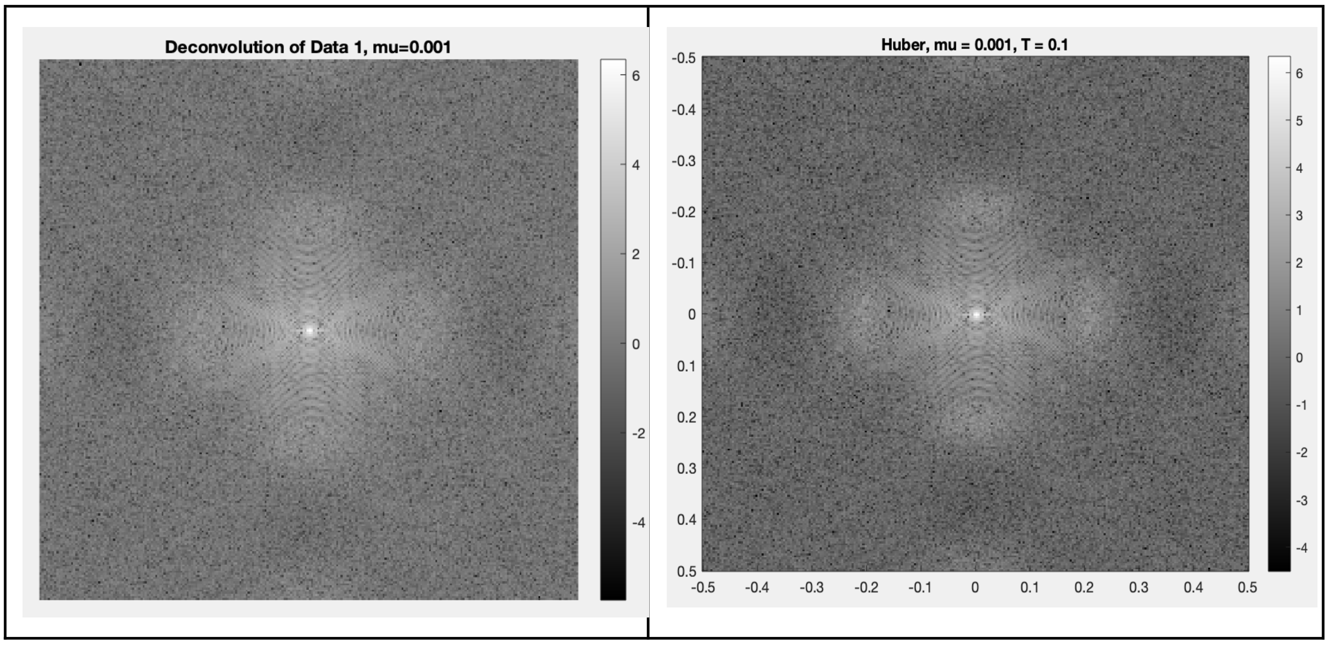

Deconvolution of Data 1, mu=1
Huber, mu = 1, T = 0.1, K = 100

Fig 2. Spatial domain. Left column is Wiener-Hunt, right column is Huber method



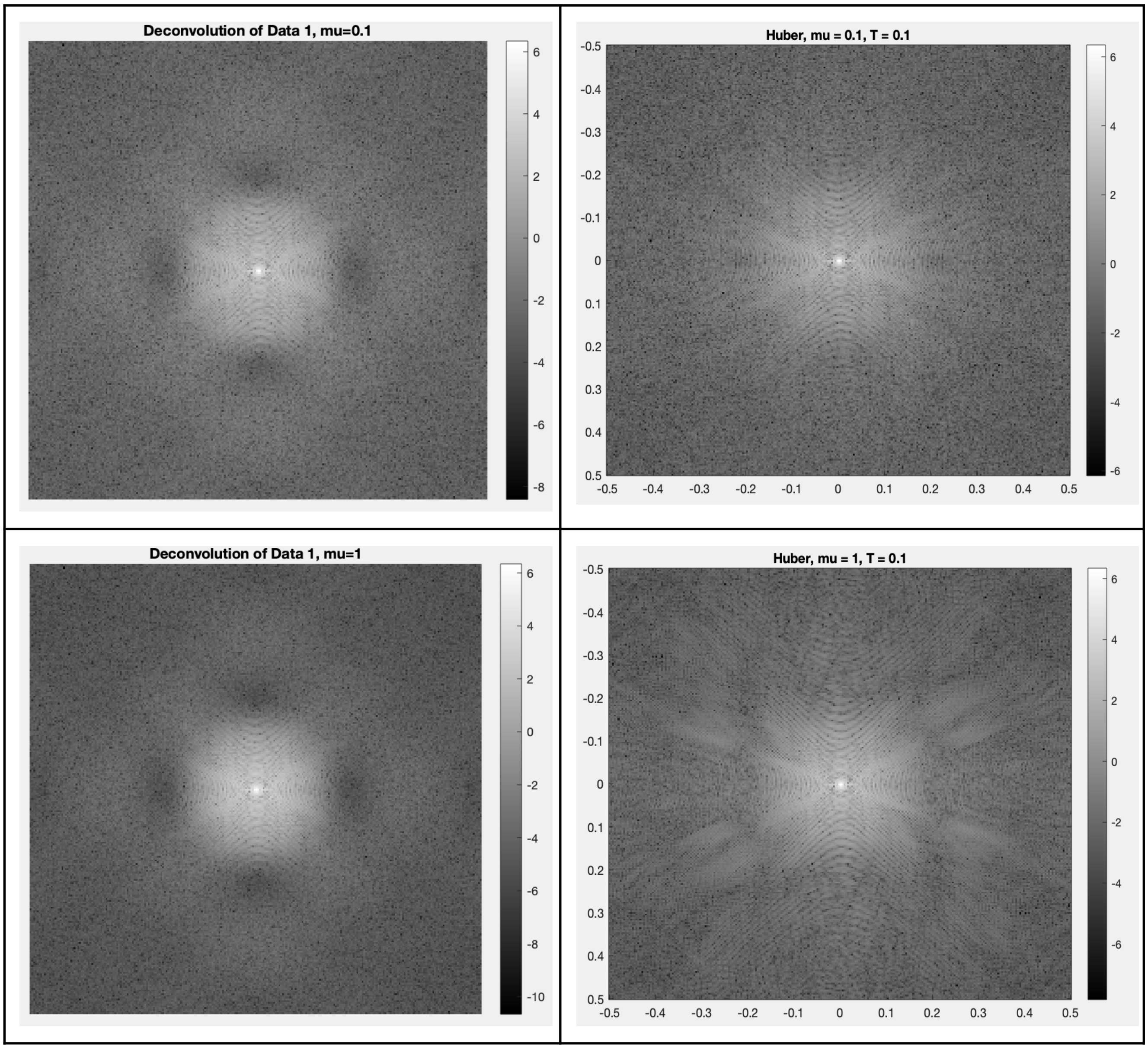


Fig 3. Frequency domain. Left column is Wiener-Hunt, right column is Huber method

As we can see in Fig 2, the Huber method preserves more edges than the Wiener-Hunt method. It smoothes homogenous regions as Wiener-Hunt and acts linearly in regions with high frequencies. We can notice that in the right column of Fig 3, the high frequencies are not completely eliminated as in the Wiener-Hunt. By tuning the mu parameter we can control the penalization term.

b. Influence of *T*

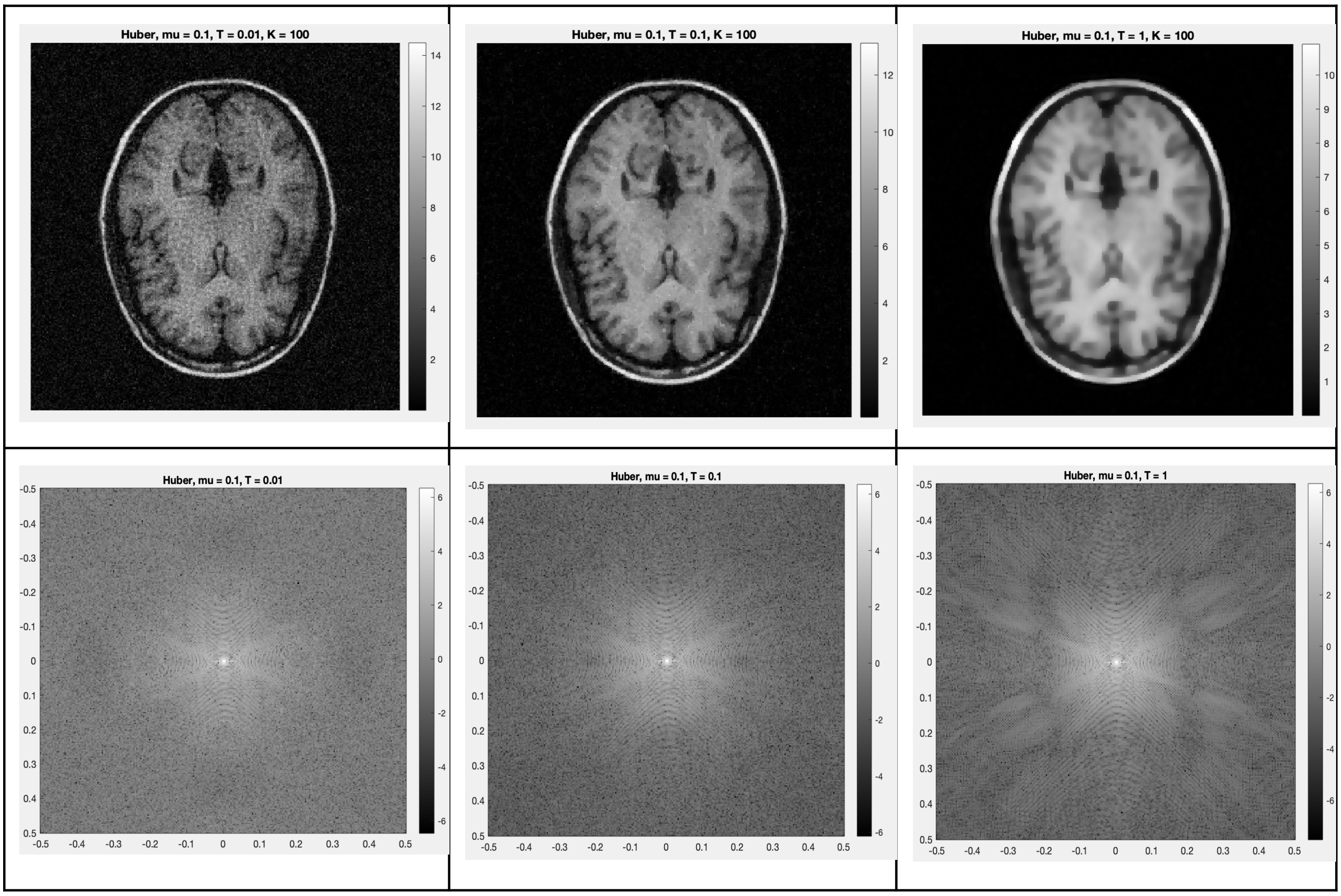

Huber, mu = 0.1, T = 0.01, K = 100
Huber, mu = 0.1, T = 0.1, K = 100
Huber, mu = 0.1, T = 1, K = 100
Huber, mu = 0.1, T = 0.01
Huber, mu = 0.1, T = 0.1
Huber, mu = 0.1, T = 1

Fig 4. T increases from left to right

Fig 4 shows us that by increasing the value of the T parameter, the image gets more smooth by eliminating the noise. However, we lose the sharpness of the image. We can also see that increasing the T value, penalization term becomes more quadratic than linear.

c. Convergence speed with respect to *alpha*

To evaluate the convergence of the algorithm, Euclidean norm (also known as L2-norm) was computed between neighbor values of deconvolved image y. As we can see in Table 1, convergence speed is increased exponentially by increasing the alpha values.

		alpha								
		0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
	1	0.1619	0.0366	0.0868	0.0879	0.0729	0.0603	0.0537	0.0440	0.0346
	25	0.0126	0.0232	0.0156	0.0092	0.0062	0.0038	0.0021	0.0010	0.0005
	50	0.0103	0.0070	0.0031	0.0014	0.0006	0.0003	0.0001	0.0000	0.0000
	100	0.0054	0.0024	0.0008	0.0002	0.0001	0.0000	0.0000	0.0000	0.0000
	125	0.0030	0.0009	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	150	0.0018	0.0004	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	175	0.0011	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	200	0.0006	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	225	0.0004	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Iterations	***250***	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	275	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	300	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	325	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	350	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	375	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	400	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	425	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	450	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	475	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 1. Convergence speed