1. FISTA and L-BFGS

Implement the accelerated proximal gradient method (FISTA) and L-BFGS method for the l-problem.

In order to recover the damaged images, the FISTA and L-BFGS will be used in this section. FISTA algorithm is an efficient tool to improve the performance and outcome.

To deal with the data of pictures, the information of masks are put into **Ind** function.

$$\mathbf{Ind}_{ij} = \begin{cases} 1 & \text{the pixel } (i,j) \text{ is not damaged,} \\ 0 & \text{the pixel } (i,j) \text{ is damaged.} \end{cases}$$

Then, set **ind** =vec(**Ind**). The information of damaged pictures are stored in U matrix, and set u= vec(U) to generate vector b, which has equation that $b = Pu$.

In l_1 -regularized image reconstruction problem, transformation is set as DCT-transformation.

Then, FISTA model is

$$\min_{x\in\mathbb{R}^{mn}} \frac{1}{2}\|Ax-b\|^2 + \mu\|x\|_1$$

Where $Ax = Pidct(x),\ P(x) = x(ind = 1)$

L-BFGS model is

$$\min_{x\in\mathbb{R}^{mn}} \frac{1}{2}\|Ax-b\|^2 + \mu\sum_{i=1}^{mn}\varphi_{\text{hub}}(x_i),$$

where the so-called *Huber-function* is defined as follows:

$$\varphi_{\text{hub}}(z) = \begin{cases} \frac{1}{2\delta}z^2 & \text{if } |z| \le \delta, \\ |z| - \frac{1}{2}\delta & \text{if } |z| > \delta, \end{cases} \quad z \in \mathbb{R}, \quad \delta > 0.$$

The parameter in FISTA model are supposed as followed to get better performance:

parameter	**value**	**parameter**	**value**
μ	0.01	opts.maxit	10000
opts.tol	1e-4	opts.L	1
opts.restart	0	opts.backMax	10

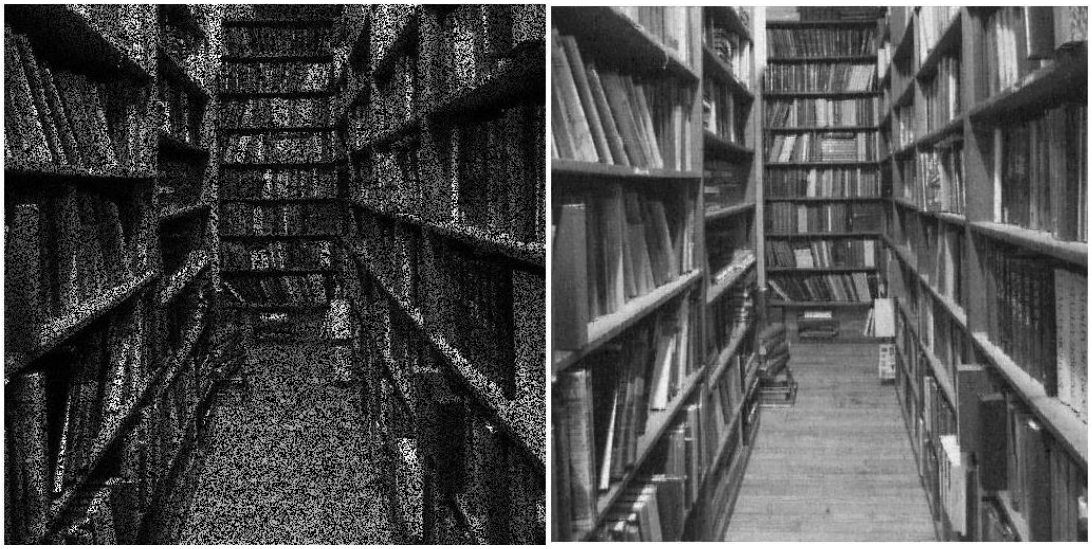
The parameter in L-BFGS model are supposed as followed to get better performance:

```
def L_BFGS(x0, H0, image_data, mask_data, memory_size, miu_data, delta_data):

    # step 1: Initial point x0 and parameter
    m = memory_size # memory size with [5, 7, 10, 12]
    miu = miu_data #0.001 # for Hunber-function [0.001, 0.1]
    delta = delta_data #0.01 # for Hunber-function [0.01, 0.5]

    sigma = 0.5
    gamma = 0.1
    tolerance = 1e-13 # epsilon
    max_k = 1000 # maximum iteration
```

The performance of model is shown as follow.



Damaged picture 1 VS recovered picture 1

Damaged picture 2 VS recovered picture 2

It can be seen that the picture is recovered well. Although the recovered pictures have some Unusually fuzzy places, they are acceptable.

Compare the results of the two methods and models with respect to the PSNR value, the number of iterations, and the required cpu-time. In this essay, three damaged images are taken as examples.

In FISTA method

Damaged picture 1	
measure	**result**
PSNR value	31.591600281428963
the number of iterations	403
cpu-time	6.62 sec
Damaged picture 2	
measure	**result**
PSNR value	29.869673619449753
the number of iterations	360
cpu-time	6.49 sec
Damaged picture 3	
measure	**result**
PSNR value	35.498263397099315
the number of iterations	266
cpu-time	4.48 sec

The PSNR values are approximately above 30, the time model cost is about 4 to 6 seconds. The number of iterations fluctuate from 250 to 400. The performance of the model is well.

3.1 In BFGS method

CPU time and number of iterations:

```
(5, 0.001, 0.01): [5.1392621994018555, 9],  (7, 0.001, 0.01): [5.2130656242370605, 9],
(5, 0.001, 0.1): [6.603349208831787, 13],   (7, 0.001, 0.1): [6.701087713241577, 13],
(5, 0.001, 0.5): [14.498246431350708, 41],  (7, 0.001, 0.5): [14.508219718933105, 41],
(5, 0.01, 0.01): [4.309480667114258, 6],    (7, 0.01, 0.01): [4.236675977706909, 6],
(5, 0.01, 0.1): [6.131610155105591, 10],    (7, 0.01, 0.1): [6.104681491851807, 10],
(5, 0.01, 0.5): [14.465334177017212, 41],   (7, 0.01, 0.5): [14.365600824356079, 41],
(5, 0.1, 0.01): [4.307485818862915, 6],     (7, 0.1, 0.01): [4.222712755203247, 6],
(5, 0.1, 0.1): [5.006617307662964, 6],      (7, 0.1, 0.1): [4.922842264175415, 6],
(5, 0.1, 0.5): [14.236944675445557, 40],    (7, 0.1, 0.5): [14.169125080108643, 40],

(10, 0.001, 0.01): [5.167187929153442, 9],  (12, 0.001, 0.01): [5.171176910400391, 9],
(10, 0.001, 0.1): [6.515583753585815, 13],  (12, 0.001, 0.1): [6.558471202850342, 13],
(10, 0.001, 0.5): [14.455361366271973, 41]  (12, 0.001, 0.5): [14.479295015335083, 41]
(10, 0.01, 0.01): [4.245650768280029, 6],   (12, 0.01, 0.01): [4.29452109336853, 6],
(10, 0.01, 0.1): [6.180479526519775, 10],   (12, 0.01, 0.1): [6.062793970108032, 10],
(10, 0.01, 0.5): [14.546121597290039, 41],  (12, 0.01, 0.5): [14.432422399520874, 41],
(10, 0.1, 0.01): [4.314464330673218, 6],    (12, 0.1, 0.01): [4.305491209030151, 6],
(10, 0.1, 0.1): [5.01060676574707, 6],      (12, 0.1, 0.1): [4.977694749832153, 6],
(10, 0.1, 0.5): [14.335680723190308, 40],   (12, 0.1, 0.5): [14.222981452941895, 40]}
```

PSNR value

(5, 0.001, 0.01): 7.127515571008097,
(5, 0.001, 0.1): 7.127515571008091,
(5, 0.001, 0.5): 7.127515571008098,
(5, 0.01, 0.01): 7.2173780593104855,
(5, 0.01, 0.1): 7.217378059310481,
(5, 0.01, 0.5): 7.217378059310487,
(5, 0.1, 0.01): 8.074518993157177,
(5, 0.1, 0.1): 8.074518993157177,
(5, 0.1, 0.5): 8.07451899315718,

(7, 0.001, 0.01): 7.127515571008097,
(7, 0.001, 0.1): 7.127515571008091,
(7, 0.001, 0.5): 7.127515571008098,
(7, 0.01, 0.01): 7.2173780593104855,
(7, 0.01, 0.1): 7.217378059310481,
(7, 0.01, 0.5): 7.217378059310487,
(7, 0.1, 0.01): 8.074518993157177,
(7, 0.1, 0.1): 8.074518993157177,
(7, 0.1, 0.5): 8.07451899315718,

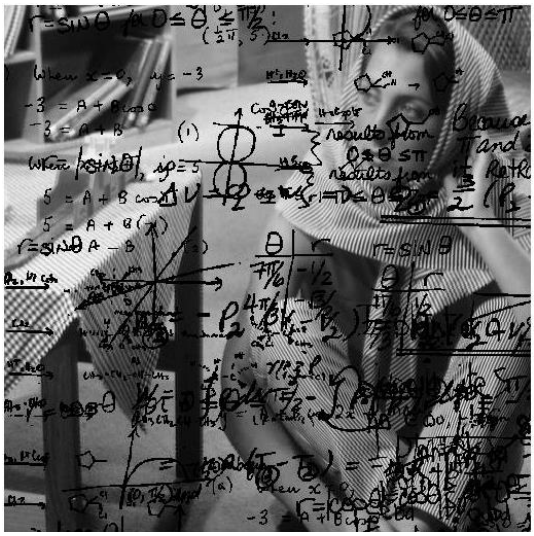
(10, 0.001, 0.01): 7.127515571008097,
(10, 0.001, 0.1): 7.127515571008091,
(10, 0.001, 0.5): 7.127515571008098,
(10, 0.01, 0.01): 7.2173780593104855,
(10, 0.01, 0.1): 7.217378059310481,
(10, 0.01, 0.5): 7.217378059310487,
(10, 0.1, 0.01): 8.074518993157177,
(10, 0.1, 0.1): 8.074518993157177,
(10, 0.1, 0.5): 8.07451899315718,

(12, 0.001, 0.01): 7.127515571008097,
(12, 0.001, 0.1): 7.127515571008091,
(12, 0.001, 0.5): 7.127515571008098,
(12, 0.01, 0.01): 7.2173780593104855,
(12, 0.01, 0.1): 7.217378059310481,
(12, 0.01, 0.5): 7.217378059310487,
(12, 0.1, 0.01): 8.074518993157177,
(12, 0.1, 0.1): 8.074518993157177,
(12, 0.1, 0.5): 8.07451899315718}

Plot the reconstructed images of FISTA

The reconstructed images are shown as follow.

Damaged picture 3 VS Recovered picture 3

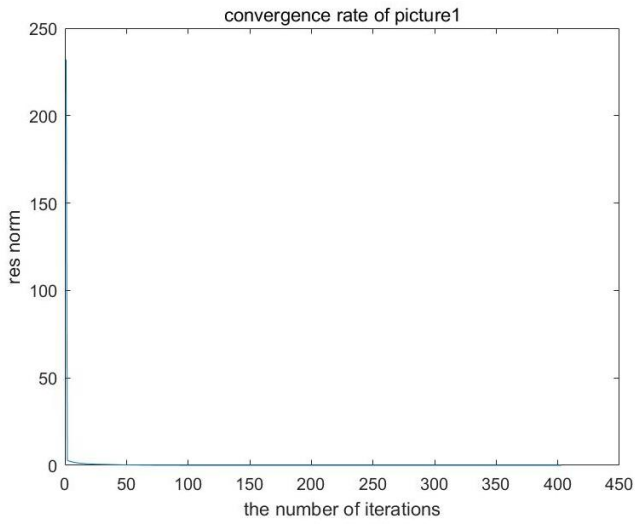


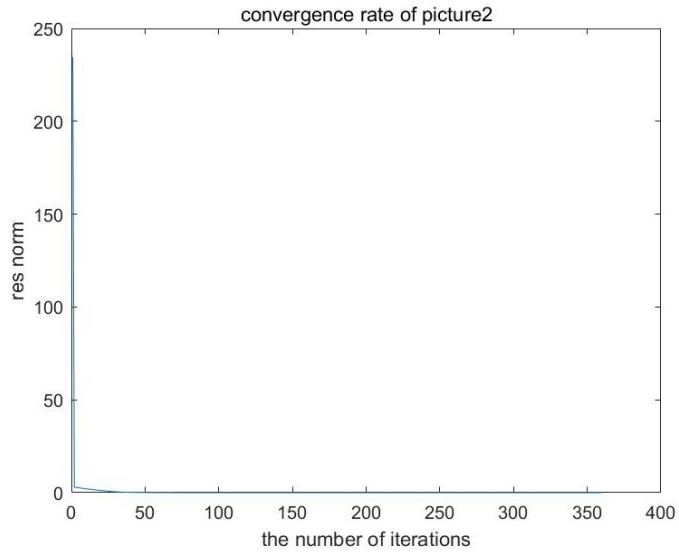

Damaged picture 4 VS Recovered picture 4

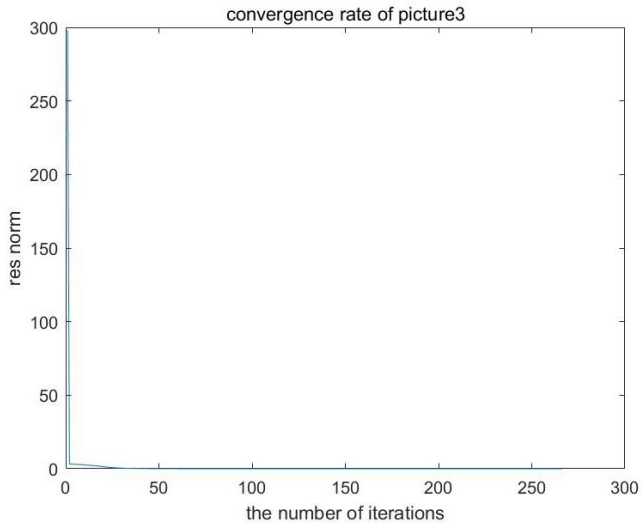

Damaged picture 5 VS Recovered picture 5

From the comparison of damaged images and recovered images, it can be seen that the picture is recovered well. There are still some places that we can see the shadow of mesh, but it doesn't influence the quality of picture.

The convergence behavior of FISTA







The convergence rate decreases rapidly at first, then converges to 0.

The convergence behavior of BFGS

The convergence rate decreases rapidly at first, then converges to 0.

Fixing m and μ

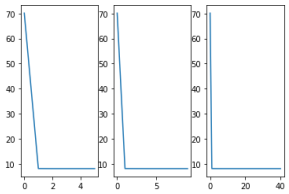
```
# compare at m = 5 and miu = 0.01
plt.figure(0.01)
plt.subplot(1,3,1)
plt.axis('off')
plt.imshow(result_image_data(m_data[(5, 0.01, 0.01)][0], test_image), 'gray')
plt.subplot(1,3,2)
plt.axis('off')
plt.imshow(result_image_data(m_data[(5, 0.01, 0.1)][0], test_image), 'gray')
plt.subplot(1,3,3)
plt.axis('off')
plt.imshow(result_image_data(m_data[(5, 0.01, 0.5)][0], test_image), 'gray')

```

```
<matplotlib.image.AxesImage at 0x1e3c4f81160>
```

```
# compare at m = 5 and miu = 0.01
plt.figure(0.01)
plt.subplot(1,3,1)
plt.plot(m_data[(5, 0.01, 0.01)][2], image_norm - m_data[(5, 0.01, 0.01)][1])
plt.subplot(1,3,2)
plt.plot(m_data[(5, 0.01, 0.1)][2], image_norm - m_data[(5, 0.01, 0.1)][1])
plt.subplot(1,3,3)
plt.plot(m_data[(5, 0.01, 0.5)][2], image_norm - m_data[(5, 0.01, 0.5)][1])
```

```
[<matplotlib.lines.Line2D at 0x1e3c96c2748>]
```



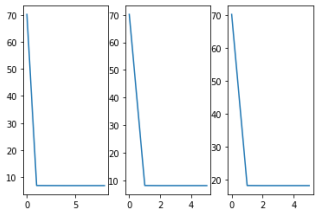
Fixing m and δ

```
# compare at m = 5 and delta = 0.01
plt.figure(0.01)
plt.subplot(1,3,1)
plt.axis('off')
plt.imshow(result_image_data(m_data[(5, 0.001, 0.01)][0], test_image), 'gray')
plt.subplot(1,3,2)
plt.axis('off')
plt.imshow(result_image_data(m_data[(5, 0.01, 0.01)][0], test_image), 'gray')
plt.subplot(1,3,3)
plt.axis('off')
plt.imshow(result_image_data(m_data[(5, 0.1, 0.01)][0], test_image), 'gray')
```

```
<matplotlib.image.AxesImage at 0x1e3c993f080>
```

```
# compare at m = 5 and delta = 0.01
plt.figure(0.01)
plt.subplot(1,3,1)
plt.plot(m_data[(5, 0.001, 0.01)][2], image_norm - m_data[(5, 0.001, 0.01)][1])
plt.subplot(1,3,2)
plt.plot(m_data[(5, 0.01, 0.01)][2], image_norm - m_data[(5, 0.01, 0.01)][1])
plt.subplot(1,3,3)
plt.plot(m_data[(5, 0.1, 0.01)][2], image_norm - m_data[(5, 0.1, 0.01)][1])
```

```
[<matplotlib.lines.Line2D at 0x1e3c99ee208>]
```



Fixing μ and δ

```
# compare at miu = 0.01 and delta = 0.5
plt.figure(0.01)
plt.subplot(2,2,1)
plt.axis('off')
plt.imshow(result_image_data(m_data[(5, 0.01, 0.5)][0], test_image), 'gray')
plt.subplot(2,2,2)
plt.axis('off')
plt.imshow(result_image_data(m_data[(7, 0.01, 0.5)][0], test_image), 'gray')
plt.subplot(2,2,3)
plt.axis('off')
plt.imshow(result_image_data(m_data[(10, 0.01, 0.5)][0], test_image), 'gray')
plt.subplot(2,2,4)
plt.axis('off')
plt.imshow(result_image_data(m_data[(12, 0.01, 0.5)][0], test_image), 'gray')
```

```
<matplotlib.image.AxesImage at 0x1e3c9618748>
```

```
# compare at miu = 0.01 and delta = 0.5
plt.figure(0.01)
plt.subplot(2,2,1)
plt.plot(m_data[(5, 0.01, 0.5)][2], image_norm - m_data[(5, 0.01, 0.5)][1])
plt.subplot(2,2,2)
plt.plot(m_data[(7, 0.01, 0.5)][2], image_norm - m_data[(7, 0.01, 0.5)][1])
plt.subplot(2,2,3)
plt.plot(m_data[(10, 0.01, 0.5)][2], image_norm - m_data[(10, 0.01, 0.5)][1])
plt.subplot(2,2,4)
plt.plot(m_data[(12, 0.01, 0.5)][2], image_norm - m_data[(12, 0.01, 0.5)][1])
```

```
[<matplotlib.lines.Line2D at 0x1e3c97f7be0>]
```