

# MATH473 HW2

Jiasen Zhang

## 1 Problem 1

$$f(x) = -|x| \Rightarrow f(x) = \begin{cases} x & x < 0 \\ -x & x \geq 0 \end{cases}$$

$$\begin{aligned} f'(0_+) &= \lim_{h \rightarrow 0} \frac{f(0+h) - f(0)}{h} = \frac{-h - 0}{h} = -1 \\ f'(0_-) &= \lim_{h \rightarrow 0} \frac{f(0) - f(0-h)}{h} = \frac{0 - (-h)}{h} = 1 \\ \Rightarrow f'(0_+) &\neq f'(0_-) \end{aligned}$$

So  $f(x)$  is not differentiable at 0.

Define  $v(x)$  as the weak derivative of  $f$ , then we have:

$$\int_{-\infty}^{\infty} f \phi' dx = - \int_{-\infty}^{\infty} \phi v dx \quad \text{for any } \phi(x) \in C_0^\infty(-\infty, +\infty)$$

$$\begin{aligned} \int_{-\infty}^{\infty} f \phi' dx &= \int_{-\infty}^0 x \phi' dx - \int_0^{\infty} x \phi' dx \\ &= \int_{-\infty}^0 x d\phi - \int_0^{\infty} x d\phi \\ &= x\phi(x) \Big|_{-\infty}^0 - \int_{-\infty}^0 \phi(x) dx - x\phi(x) \Big|_0^{\infty} + \int_0^{\infty} \phi(x) dx \end{aligned}$$

$$\begin{aligned} &\text{Becasue } \phi(\infty) = \phi(-\infty) = 0 \\ &= \int_0^{\infty} \phi(x) dx - \int_{-\infty}^0 \phi(x) dx \end{aligned} \tag{1}$$

$$= - \int_{-\infty}^{\infty} v(x) dx \tag{2}$$

By comparing equations (1) and (2), we can find a weak derivative of  $f$ :

$$v(x) = \begin{cases} 1 & x < 0 \\ 0 & x = 0 \\ -1 & x > 0 \end{cases}$$

## 2 Problem 2

The paper proposes to denoise images by minimizing the total variation norm of the estimated solution. A constrained minimization algorithm is derived as a time dependent nonlinear PDE, where the constraints are determined by the noise statistics.

First let  $u_0(x, y)$  denote the pixel values of a noisy image. Let  $u(x, y)$  denote the desired clean image and  $n(x, y)$  denote the additive noise.

$$u_0(x, y) = u(x, y) + n(x, y) \quad x, y \in \Omega \quad (1)$$

The paper uses total variation (TV) norm ( $L_1$  norms) to estimate the image. Although  $L_1$  estimation is nonlinear and computationally complex, it has been conjectured that  $L_1$  norm is more appropriate than  $L_2$  norm for image estimation. In fact  $L_1$  norm is good for shock calculation by removing spurious oscillations and preserve sharp signals. And it's the space of functions of bounded total variation BV. The derived constrained minimization problem is:

$$\text{minimize } \int_{\Omega} \sqrt{u_x^2 + u_y^2} dx dy \quad (2)$$

The two constraints make sure the mean and standard deviation of the noise  $n(x, y)$  are respectively 0 and  $\sigma$ :

$$\int_{\Omega} u dx dy = \int_{\Omega} u_0 dx dy \quad (3)$$

$$\int_{\Omega} \frac{1}{2} (u - u_0)^2 dx dy = \sigma^2 \quad , \text{ where } \sigma > 0 \text{ is given.} \quad (4)$$

The solution procedure of equations (2-4) is based on Euler-Lagrange equations, which converts the optimization problem to a PDE problem (5-6). The PDE is time dependent and as time increases, the image will approach its denoised version.  $\lambda$  can develop with time or be a constant.

$$u_t = \frac{\partial}{\partial x} \left( \frac{u_x}{\sqrt{u_x^2 + u_y^2}} \right) + \frac{\partial}{\partial y} \left( \frac{u_y}{\sqrt{u_x^2 + u_y^2}} \right) - \lambda(u - u_0) \quad t > 0, \quad x, y \in \Omega \quad (5)$$

$$u(x, y, 0) \text{ is given and } \frac{\partial u}{\partial n} = 0 \quad \text{on } \Omega \quad (6)$$

A numerical method is used for (5-6) and the algorithm is tested on graphs and real images. The concludes include that TV denoising is better than a Wiener filter denoising. The discontinuities are much clearer in TV denoising while Wiener filter denoising has oscillatory artifacts.

### 3 Problem 3

#### 3.1

To compare TV and Tikhonov inpainting, I set the same parameters: the ratio of missing pixels is 0.7, the time step is  $10^{-7}$  and the tolerance is  $10^{-5}$ . Their PSNR and relative errors are shown below the Figure 1 and 2. Although their PSNR and relative errors are close, the result of TV inpainting looks better with sharper edges.

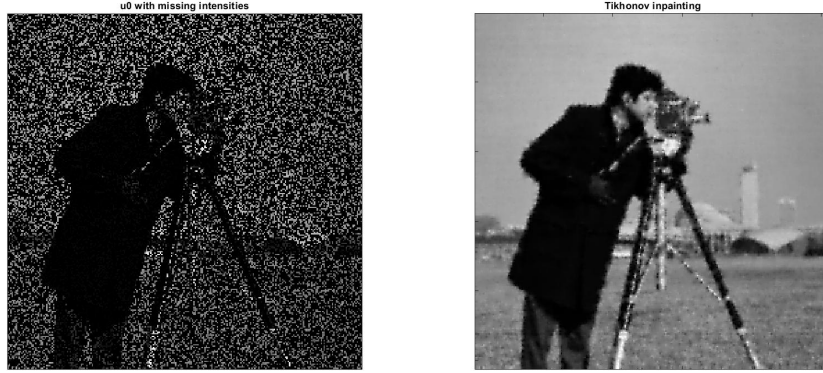


Figure 1: Tikhonov inpainting: PSNR=24.6106, relative error=0.1131.



Figure 2: TV inpainting.: PSNR=24.2277, relative error=0.1181

### 3.2

When the ratio of missing pixels increases, the image  $u_0$  with missing intensities will be less informative. Therefore, with a higher ratio of missing pixels, the quality of inpainting will be worse with smaller PSNR and larger relative errors.

Figure 3 and 4 shows Tikhonov inpainting with 0.8 and 0.9 missing intensities. Figure 5 and 6 shows TV inpainting with 0.8 and 0.9 missing intensities.



Figure 3: Tikhonov inpainting: 0.8 missing intensities, PSNR=23.1858, relative error=0.1332.



Figure 4: Tikhonov inpainting.: 0.9 missing intensities, PSNR=21.4394, relative error=0.1629



Figure 5: TV inpainting: 0.8 missing intensities, PSNR=22.5524, relative error=0.1433.



Figure 6: TV inpainting.: 0.9 missing intensities, PSNR=20.3209, relative error=0.1853

### 3.3

Implementing Tikhonov and TV denoising with  $\lambda = 10^d$ , Table 1 and 2 show PSNR and relative errors with different  $\lambda$  by changing values of  $d$ .

$\lambda = 10^d$	PSNR	Relative error
d=4	20.1052	0.1899
d=4.5	22.0334	0.1521
d=5	23.9936	0.1214
d=5.5	24.5808	0.1134
d=6	22.9051	0.1376
d=6.5	21.2215	0.1671

Table 1: Tikhonov denoising

$\lambda = 10^d$	PSNR	Relative error
d=2.5	20.2099	0.1877
d=3	22.9329	0.1372
d=3.5	26.6046	0.0899
d=4	24.3952	0.1159
d=4.5	21.4515	0.1627
d=5	20.4667	0.1822

Table 2: TV denoising

Obviously, when  $d = 5.5$ , the result of Tikhonov denoising has the largest PSNR. And when  $d = 3.5$ , the result of TV denoising has the largest PSNR. Show these two results in Figure 7 and 8.

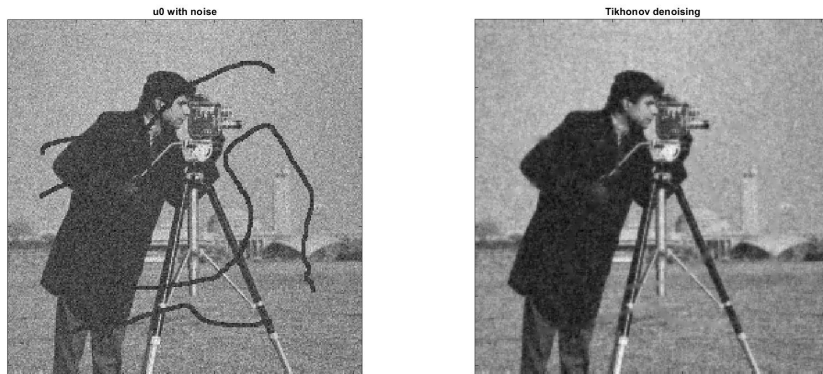


Figure 7: Tikhonov denoising: d=5.5

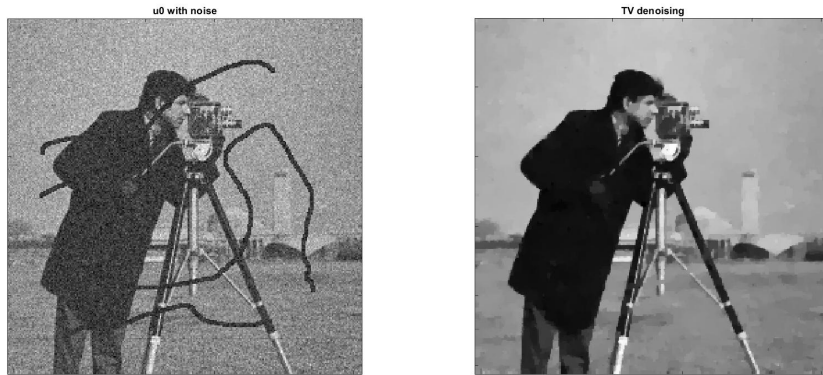


Figure 8: TV denoising:  $d=3.5$

### 3.4

Using the TV inpainting algorithm in 3.1 to remove the texts, the result is shown below in Figure 9.



Figure 9: Removing texts with TV denoising

## 4 Matlab Code

### 4.1 Tikhonov inpainting

```
1 function [u] = Tik_inpainting(u0, mask, e, dt)
2 %   TIK.INPAINTING by Jiasen Zhang
3
4 % input
5 % u0 = image with missing intensities
6 % mask = 1 as location of missing intensities
7 % e = tolerance
8 % dt = tiem step
9
10 % output
11 % u = result
12
13 [m,n]=size(u0);
14 h = 1/min(n,m); % here m=n
15 u=u0;
16 lu = zeros(m,n); % size Laplace u
17 for k=1:200000 % set a max iteration
18     % Laplace u
19     lu(2:m-1,2:n-1) = u(3:m,2:n-1) +u(1:m-2,2:n-1)+...
20         u(2:m-1,3:n) +u(2:m-1,1:n-2) -4*u(2:m-1,2:n-1);
21     unew=u+2*dt*lu/(h^2);
22     % boundary condition
23     unew(1,:)=unew(2,:);
24     unew(m,:)=unew(m-1,:);
25     unew(:,1)=unew(:,2);
26     unew(:,n)=unew(:,n-1);
27     % u=u0 on D
28     unew(mask~=1)=u0(mask~=1);
29     d=max(max(abs(u-unew))); % difference
30     if mod(k,200)==0
31         fprintf('%3e, %d\n',d,k);
32     end
33     u=unew;
34     if d<e
35         break
36     end
37 end
38
39 figure();
40 subplot(1,2,1);
41 imagesc(u0);
42 axis square;colormap(gray);set(gca,'XTickLabel','', 'YTickLabel','');
43 title('u0 with missing intensities');
44 subplot(1,2,2);
45 imagesc(u);
46 axis square;colormap(gray);set(gca,'XTickLabel','', 'YTickLabel','');
47 title('Tikhonov inpainting');
48
49 end
```

### 4.2 TV inpainting

```
1 function [u] = TV_inpainting(u0, mask, e, dt)
2 %   TV.INPAINTING by Jiasen Zhang
3
```



```

4 % input
5 % u0 = image with missing intensities
6 % mask = 1 as location of missing intensities
7 % e = tolerance
8 % dt = time step
9
10 % output
11 % u = result
12
13 [m,n]=size(u0);
14 h = 1/min(n,m); % here m=n
15 du = zeros(m,n);
16 divuy = du;
17 divux = du;
18 u=u0;
19 tau = 1e-6; % small term in the denominator
20 for k=1:200000 % set a max iteration
21     % given u
22     % get divux and divuy
23     duxf = u(3:m,2:n-1)-u(2:m-1,2:n-1); % forward difference of x direction
24     duxb = u(2:m-1,2:n-1)-u(1:m-2,2:n-1); % backward difference of x direction
25     duyf = u(2:m-1,3:n)-u(2:m-1,2:n-1); % forward difference of y direction
26     duyb = u(2:m-1,2:n-1)-u(2:m-1,1:n-2); % backward difference of y direction
27     divux(2:m-1,2:n-1) = duxf./(sqrt( tau*ones(m-2,n-2)+duxf.^2 ...
        +minmod(duyf,duyb).^2 ));
28     divuy(2:m-1,2:n-1) = duyf./(sqrt( tau*ones(m-2,n-2)+duyf.^2 ...
        +minmod(duxf,duxb).^2 ));
29     % get du
30     du(2:m-1,2:n-1) = divux(2:m-1,2:n-1)-divux(1:m-2,2:n-1)...
        + divuy(2:m-1,2:n-1)-divuy(2:m-1,1:n-2);
31     % unew
32     unew=u+dt*du/h;
33     % boundary condition
34     unew(1,:)=unew(2,:);
35     unew(m,:)=unew(m-1,:);
36     unew(:,1)=unew(:,2);
37     unew(:,n)=unew(:,n-1);
38     % u=u0 on D
39     unew(mask~=1)=u0(mask~=1);
40     d=max(max(abs(u-unew))); % difference
41     % if mod(k,200)==0
42     %     fprintf('%3e, %d\n',d,k);
43     % end
44     u=unew;
45     if d<e
46         break
47     end
48 end
49
50
51 figure();
52 subplot(1,2,1);
53 imagesc(u0);
54 axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
55 title('u0 with missing intensities');
56 subplot(1,2,2);
57 imagesc(u);
58 axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
59 title('TV inpainting');
60
61 % minmod
62 function result = minmod(a,b)
63 % minmod(a,b)
64 result=(sign(a)+sign(b)).*min(abs(a),abs(b))/2;
65 end
66 end

```

### 4.3 Tikhonov denoising

```
1 function [u] = Tik_denoising(J, scratch, e, dt, lambda)
2 %   TIK_DENOISING by Jiasen Zhang
3
4 % input
5 % J = image with missing intensities
6 % scratch = 0 as location of missing intensities
7 % e = tolerance
8 % dt = time step
9 % lambda = a parameter
10
11 % output
12 % u = result
13
14 [m,n]=size(J);
15 h = 1/min(n,m); % here m=n
16 u=J;
17 lu = zeros(m,n); % size Laplace u
18 for k=1:200000 % set a max iteration
19     % Laplace u
20     lu(2:m-1,2:n-1) = u(3:m,2:n-1) +u(1:m-2,2:n-1)+...
21         u(2:m-1,3:n) +u(2:m-1,1:n-2) -4*u(2:m-1,2:n-1);
22     uD = u;
23     uD(scratch==0)=0;
24     JD = J;
25     JD(scratch==0)=0;
26     unew=u + 2*dt*lu/(h^2) - dt*lambda*(uD-JD);
27
28     % boundary condition
29     unew(1,:)=unew(2,:);
30     unew(m,:)=unew(m-1,:);
31     unew(:,1)=unew(:,2);
32     unew(:,n)=unew(:,n-1);
33     d=max(max(abs(u-unew))); % difference
34     %d = norm(u-unew)/norm(u);
35     if mod(k,200)==0
36         fprintf('%e, %d\n',d,k);
37     end
38     u=unew;
39     if d<e
40         break
41     end
42 end
43 figure();
44 subplot(1,2,1);
45 imagesc(J);
46 axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
47 title('u0 with noise');
48 subplot(1,2,2);
49 imagesc(u);
50 axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
51 title('Tikhonov denoising');
52
53 end
```

### 4.4 TV denoising

```
1 function [u] = TV_denoising(J, scratch, e, dt, lambda)
2 %   TV_DENOISING by Jiasen Zhang
```

```

3
4 % input
5 % J = image with missing intensities
6 % scratch = 0 as location of missing intensities
7 % e = tolerance
8 % dt = tiem step
9 % lambda = a parameter
10
11 % output
12 % u = result
13
14 [m,n]=size(J);
15 h = 1/min(n,m); % here m=n
16 % size of matrices
17 u=zeros(m,n);
18 divux=u;divuy=u;
19 du = u;
20 u=J;
21 tau = 1e-8; % small term in the denominator
22 for k=1:200000 % set a max iteration
23     % given u
24     % get divux and divuy
25     duxf = u(3:m,2:n-1)-u(2:m-1,2:n-1); % forward difference of x direction
26     duxb = u(2:m-1,2:n-1)-u(1:m-2,2:n-1); % backward difference of x direction
27     duyf = u(2:m-1,3:n)-u(2:m-1,2:n-1); % forward difference of y direction
28     duyb = u(2:m-1,2:n-1)-u(2:m-1,1:n-2); % backward difference of y direction
29     divux(2:m-1,2:n-1) = duxf./ (sqrt( tau*ones(m-2,n-2)+duxf.^2 ...
        +minmod(duyf,duyb).^2 ));
30     divuy(2:m-1,2:n-1) = duyf./ (sqrt( tau*ones(m-2,n-2)+duyf.^2 ...
        +minmod(duxf,duxb).^2 ));
31     % get du
32     du(2:m-1,2:n-1) = divux(2:m-1,2:n-1)-divux(1:m-2,2:n-1)...
        + divuy(2:m-1,2:n-1)-divuy(2:m-1,1:n-2);
33     uD = u;
34     uD(scratch==0)=0;
35     JD = J;
36     JD(scratch==0)=0;
37     % unew
38     unew=u + dt*du/h - dt*lambda*(uD-JD);
39     %unew=u + dt*du/h - dt*lambda*(u-J);
40     % boundary condition
41     unew(1,:)=unew(2,:);
42     unew(m,:)=unew(m-1,:);
43     unew(:,1)=unew(:,2);
44     unew(:,n)=unew(:,n-1);
45     d=max(max(abs(u-unew))); % difference
46     %d = norm(u-unew)/norm(u);
47     %
48     %     if mod(k,200)==0
49     %         fprintf('%e, %d\n',d,k);
50     %     end
51     u=unew;
52     if d<e
53         break
54     end
55 end
56 figure();
57 subplot(1,2,1);
58 imagesc(J);
59 axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
60 title('u0 with noise');
61 subplot(1,2,2);
62 imagesc(u);
63 axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
64 title('TV denoising');
65

```

```

66 function result = minmod(a,b)
67 % minmod(a,b)
68 result=(sign(a)+sign(b)).*min(abs(a),abs(b))/2;
69 end
70 end

```

## 4.5 Problem 3.1 and 3.2

```

1 clear all;clc;
2 tic
3 I = imread('cameraman.tif');
4 I = double(I); % change I from unit8 to double precision format
5 % normalize I to intensity [0,1] for easier parameter selection
6 I = (I-min(I(:)))/(max(I(:))-min(I(:)));
7 [m,n] = size(I);
8 % amount of removed pixels.
9 perc = 0.7;
10 % random mask, Mask==1 for removed pixels
11 Mask = zeros(m,n);
12 Pick = randperm(m*n); Pick = Pick(1:round(perc*m*n));
13 Mask(Pick) = 1;
14 u0 = I;
15 u0(Mask == 1) = 0;
16 %imagesc(u0);
17
18 e = 1e-5; % tolerance
19 dt = 1e-7;
20
21 % choose one and comment the other one
22 u = Tik.inpainting(u0, Mask, e, dt); % Tikhonov inpainting
23 %u = TV.inpainting(u0, Mask, e, dt); % TV inpainting
24
25 % compute PSNR
26 mse = sum(sum((u-I).^2))/(m*n);
27 maxx = max(max(abs(I)));
28 psnr = 10*log10(maxx*maxx/mse);
29 fprintf('PSNR = %f \n', psnr);
30 % Compute relative error
31 re = norm(u-I,'fro')/norm(I,'fro');
32 fprintf('Relative error = %f \n', re);
33 toc

```

## 4.6 Problem 3.3

```

1 clear all;clc;
2 tic
3 I = imread('cameraman.tif');
4 I = double(I); % change I from unit8 to double precision format
5 % normalize I to intensity [0,1] for easier parameter selection
6 I = (I-min(I(:)))/(max(I(:))-min(I(:)));
7 [m,n] = size(I);
8
9 % Zero out intensity of I at scratch to get J
10 load scratch
11 J = I;
12 J(scratch == 0) = 0;
13 % Add noise
14 sigma = 0.1;

```

```

15 J = J + sigma*randn(size(J));
16
17 d0 = 3.5;
18 lambda = 10^d0;
19 e = 1e-5; % tolerance
20 dt = 1e-7;
21
22 % choose one and comment the other one
23 %u = Tik_denoising(J, scratch, e, dt, lambda); % Tik denoising
24 u = TV_denoising(J, scratch, e, dt, lambda); % TV denoising
25
26
27 % Compute PSNR
28 mse = sum(sum((u-I).^2))/(m*n);
29 maxx = max(max(abs(I)));
30 psnr = 10*log10(maxx*maxx/mse);
31 fprintf('PSNR = %f \n', psnr);
32 % Compute relative error
33 re = norm(u-I, 'fro')/norm(I, 'fro');
34 fprintf('Relative error = %f \n', re);
35 toc

```

## 4.7 Problem 3.4

```

1 clear all;clc;
2 tic
3 I = imread('cameraman.tif');
4 I = double(I); % change I from unit8 to double precision format
5 % normalize I to intensity [0,1] for easier parameter selection
6 I = (I-min(I(:)))/(max(I(:))-min(I(:)));
7 [m,n] = size(I);
8
9 % Zero out intensities of text
10 load text
11 u0 = I;
12 u0(text == 0) = 0.92;
13 % make sure missing intensities are 1, while other intensities are 2 here
14 text = text + 1;
15
16 % TV inpainting
17 e = 1e-5; % tolerance
18 dt = 1e-7;
19 u = TV.inpainting(u0, text, e, dt); % TV inpainting
20
21 % Compute PSNR
22 mse = sum(sum((u-I).^2))/(m*n);
23 maxx = max(max(I));
24 psnr = 10*log10(maxx*maxx/mse);
25 fprintf('PSNR = %f \n', psnr);
26 % Compute relative error
27 re = norm(u-I, 'fro')/norm(I, 'fro');
28 fprintf('Relative error = %f \n', re);
29 toc

```