MATH473 HW2

Jiasen Zhang

1 Problem 1

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

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

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

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

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

$$\begin{aligned}
\int_{-\infty}^{\infty} f\phi' dx &= \int_{-\infty}^{0} x\phi' dx - \int_{0}^{\infty} x\phi' dx \\
&= \int_{-\infty}^{0} x d\phi - \int_{0}^{\infty} x d\phi \\
&= x\phi(x)\Big|_{-\infty}^{0} - \int_{-\infty}^{0} \phi(x) dx - x\phi(x)\Big|_{0}^{\infty} + \int_{0}^{\infty} \phi(x) dx \\
& \text{Becasue} \quad \phi(\infty) = \phi(-\infty) = 0 \\
&= \int_{0}^{\infty} \phi(x) dx - \int_{-\infty}^{0} \phi(x) dx \qquad (1) \\
&= -\int_{-\infty}^{\infty} v(x) dx \qquad (2)
\end{aligned}$$

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.

$$u_0(x, y) = u(x, y) + n(x, y) \qquad x, y \in \Omega \tag{1}$$

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:

$$\text{minimize} \int_\Omega \sqrt{u_x^2 + u_u^2} \mathrm{d}x\mathrm{d}y \tag{2}$$

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

$$\int_\Omega u \mathrm{d}x\mathrm{d}y = \int_\Omega u_0 \mathrm{d}x\mathrm{d}y \tag{3}$$

$$\int_\Omega \frac{1}{2}(u - u_0)^2 \mathrm{d}x\mathrm{d}y = \sigma^2 \quad \text{, where} \quad \sigma > 0 \quad \text{is given.} \tag{4}$$

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. λ can develop with time or be a constant.

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

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

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 Weiner filter denoising. The discontinuities are much clearer in TV denoising while Weiner 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.

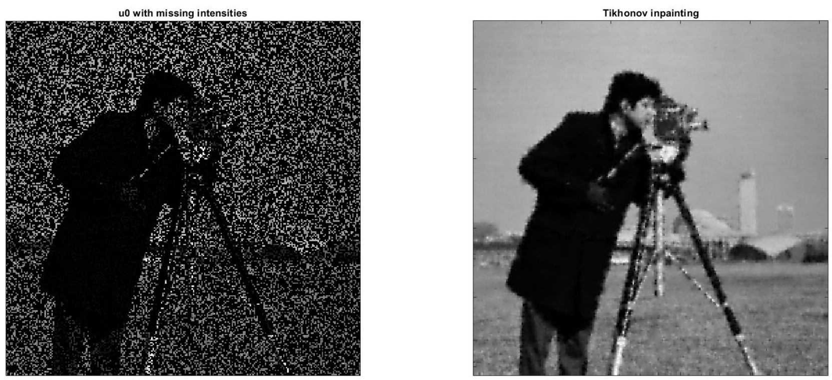

Figure 1: Tikhonov inpainting: PSNR=24.6106, relative error=0.1131.

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.

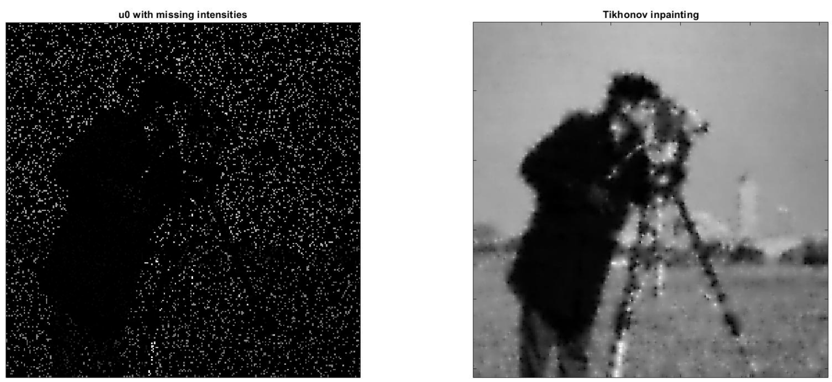

Figure 4: Tikhnov inpainting.: 0.9 missing intensities, PSNR=21.4394, relative error=0.1629

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

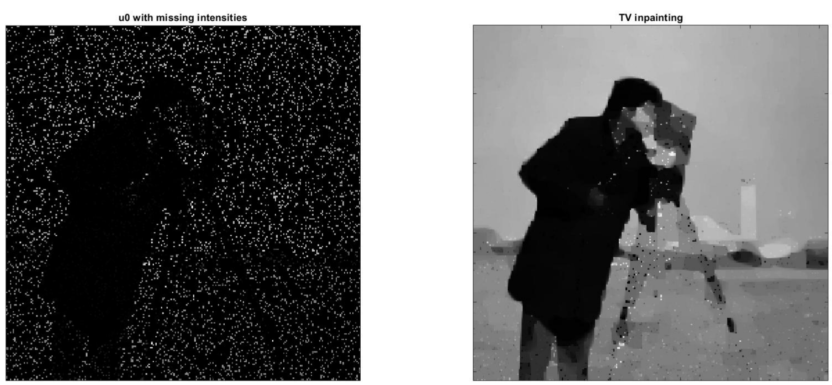

Figure 6: TV inpainting.: 0.9 missing intensities, PSNR=20.3209, relative error=0.1853

3.3

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

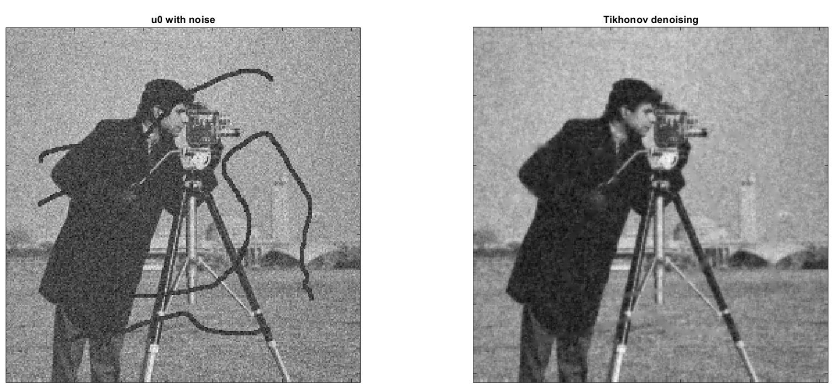

Figure 7: Tikhonov denoising: d=5.5

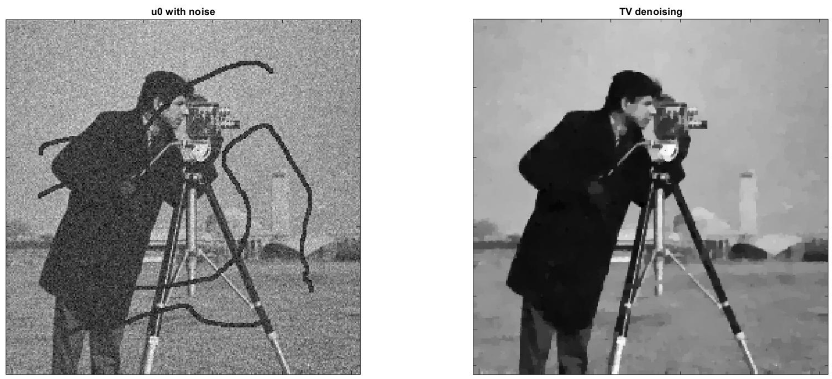

Figure 8: TV denoising: d=3.5

3.4

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

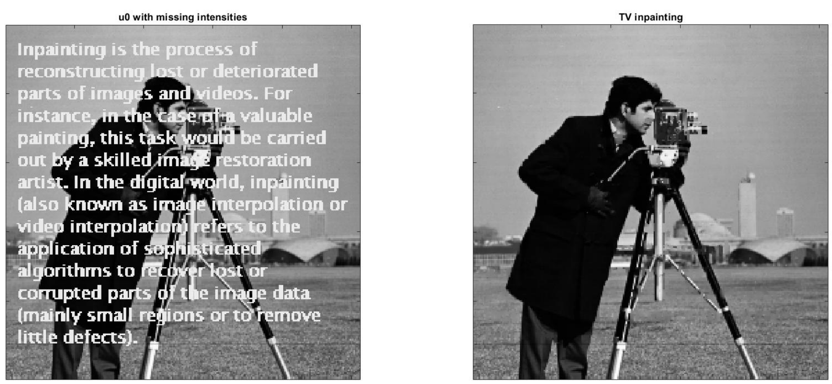

Figure 9: Removing texts with TV denoising

4 Matlab Code

4.1 Tikhonov inpainting

```
function [u] = Tik_inpainting(u0, mask, e, dt)
%   TIK_INPAINTING by Jiasen Zhang

% input
% u0 = image with missing intensities
% mask = 1 as location of missing intensities
% e = tolerance
% dt = tiem step

% output
% u = result

[m,n]=size(u0);
h = 1/min(n,m); % here m=n
u=u0;
lu = zeros(m,n); % size Laplace u
for k=1:200000 % set a max iteration
    % Laplace u
    lu(2:m-1,2:n-1) = u(3:m,2:n-1) +u(1:m-2,2:n-1)+...
        u(2:m-1,3:n) +u(2:m-1,1:n-2) -4*u(2:m-1,2:n-1);
    unew=u+2*dt*lu/(h^2);
    % boundary condition
    unew(1,:)=unew(2,:);
    unew(m,:)=unew(m-1,:);
    unew(:,1)=unew(:,2);
    unew(:,n)=unew(:,n-1);
    % u=u0 on D
    unew(mask≠1)=u0(mask≠1);
    d=max(max(abs(u-unew))); % difference
    if mod(k,200)==0
        fprintf('%3e, %d\n',d,k);
    end
    u=unew;
    if d<e
        break
    end
end

figure();
subplot(1,2,1);
imagesc(u0);
axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
title('u0 with missing intensities');
subplot(1,2,2);
imagesc(u);
axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
title('Tikhonov inpainting');

end
```

4.2 TV inpainting

```
function [u] = TV_inpainting(u0, mask, e, dt)
%   TV_INPAINTING by Jiasen Zhang

```

```
% input
% u0 = image with missing intensities
% mask = 1 as location of missing intensities
% e = tolerance
% dt = time step

% output
% u = result

[m,n]=size(u0);
h = 1/min(n,m); % here m=n
du = zeros(m,n);
divuy = du;
divux = du;
u=u0;
tau = 1e-6; % small term in the denominator
for k=1:200000 % set a max iteration
    % given u
    % get divux and divuy
    duxf = u(3:m,2:n-1)-u(2:m-1,2:n-1); % forward difference of x direction
    duxb = u(2:m-1,2:n-1)-u(1:m-2,2:n-1); % backward difference of x direction
    duyf = u(2:m-1,3:n)-u(2:m-1,2:n-1); % forward difference of y direction
    duyb = u(2:m-1,2:n-1)-u(2:m-1,1:n-2); % backward difference of y direction
    divux(2:m-1,2:n-1) = duxf./(sqrt( tau*ones(m-2,n-2)+duxf.^2 ...
        +minmod(duyf,duyb).^2 ));
    divuy(2:m-1,2:n-1) = duyf./(sqrt( tau*ones(m-2,n-2)+duyf.^2 ...
        +minmod(duxf,duxb).^2 ));
    % get du
    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);
    % unew
    unew=u+dt*du/h;
    % boundary condition
    unew(1,:)=unew(2,:);
    unew(m,:)=unew(m-1,:);
    unew(:,1)=unew(:,2);
    unew(:,n)=unew(:,n-1);
    % u=u0 on D
    unew(mask≠1)=u0(mask≠1);
    d=max(max(abs(u-unew))); % difference
%     if mod(k,200)==0
%         fprintf('%3e, %d\n',d,k);
%     end
    u=unew;
    if d<e
        break
    end
end

figure();
subplot(1,2,1);
imagesc(u0);
axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
title('u0 with missing intensities');
subplot(1,2,2);
imagesc(u);
axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
title('TV inpainting');

% minmod
function result = minmod(a,b)
% minmod(a,b)
result=(sign(a)+sign(b)).*min(abs(a),abs(b))/2;
end
end
```

4.3 Tikhonov denoising

```
function [u] = Tik_denoising(J, scratch, e, dt, lambda)
%   TIK_DENOISING by Jiasen Zhang

% input
% J = image with missing intensities
% scratch = 0 as location of missing intensities
% e = tolerance
% dt = tiem step
% lambda = a parameter

% output
% u = result

[m,n]=size(J);
h = 1/min(n,m); % here m=n
u=J;
lu = zeros(m,n); % size Laplace u
for k=1:200000 % set a max iteration
    % Laplace u
    lu(2:m-1,2:n-1) = u(3:m,2:n-1) +u(1:m-2,2:n-1)+...
        u(2:m-1,3:n) +u(2:m-1,1:n-2) -4*u(2:m-1,2:n-1);
    uD = u;
    uD(scratch==0)=0;
    JD = J;
    JD(scratch==0)=0;
    unew=u + 2*dt*lu/(h^2) - dt*lambda*(uD-JD);

    % boundary condition
    unew(1,:)=unew(2,:);
    unew(m,:)=unew(m-1,:);
    unew(:,1)=unew(:,2);
    unew(:,n)=unew(:,n-1);
    d=max(max(abs(u-unew))); % difference
    %d = norm(u-unew)/norm(u);
    if mod(k,200)==0
        fprintf('%e, %d\n',d,k);
    end
    u=unew;
    if d<e
        break
    end
end
figure();
subplot(1,2,1);
imagesc(J);
axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
title('u0 with noise');
subplot(1,2,2);
imagesc(u);
axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
title('Tikhonov denoising');

end
```

4.4 TV denoising

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

```
% input
% J = image with missing intensities
% scratch = 0 as location of missing intensities
% e = tolerance
% dt = tiem step
% lambda = a parameter

% output
% u = result

[m,n]=size(J);
h = 1/min(n,m); % here m=n
% size of matrices
u=zeros(m,n);
divux=u;divuy=u;
du = u;
u=J;
tau = 1e-8; % small term in the denominator
for k=1:200000 % set a max iteration
    % given u
    % get divux and divuy
    duxf = u(3:m,2:n-1)-u(2:m-1,2:n-1); % forward difference of x direction
    duxb = u(2:m-1,2:n-1)-u(1:m-2,2:n-1); % backward difference of x direction
    duyf = u(2:m-1,3:n)-u(2:m-1,2:n-1); % forward difference of y direction
    duyb = u(2:m-1,2:n-1)-u(2:m-1,1:n-2); % backward difference of y direction
    divux(2:m-1,2:n-1) = duxf./(sqrt( tau*ones(m-2,n-2)+duxf.^2 ...
        +minmod(duyf,duyb).^2 ));
    divuy(2:m-1,2:n-1) = duyf./(sqrt( tau*ones(m-2,n-2)+duyf.^2 ...
        +minmod(duxf,duxb).^2 ));
    % get du
    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);
    uD = u;
    uD(scratch==0)=0;
    JD = J;
    JD(scratch==0)=0;
    % unew
    unew=u + dt*du/h - dt*lambda*(uD-JD);
    %unew=u + dt*du/h - dt*lambda*(u-J);
    % boundary condition
    unew(1,:)=unew(2,:);
    unew(m,:)=unew(m-1,:);
    unew(:,1)=unew(:,2);
    unew(:,n)=unew(:,n-1);
    d=max(max(abs(u-unew))); % difference
    %d = norm(u-unew)/norm(u);
%     if mod(k,200)==0
%         fprintf('%e, %d\n',d,k);
%     end
    u=unew;
    if d<e
        break
    end
end
figure();
subplot(1,2,1);
imagesc(J);
axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
title('u0 with noise');
subplot(1,2,2);
imagesc(u);
axis square;colormap(gray);set(gca,'XTickLabel','','YTickLabel','');
title('TV denoising');

```

```
function result = minmod(a,b)
% minmod(a,b)
result=(sign(a)+sign(b)).*min(abs(a),abs(b))/2;
end
end
```

4.5 Problem 3.1 and 3.2

```
clear all;clc;
tic
I = imread('cameraman.tif');
I = double(I); % change I from unit8 to double precision format
% normalize I to intensity [0,1] for easier parameter selection
I = (I-min(I(:)))/(max(I(:))-min(I(:)));
[m,n] = size(I);
% amount of removed pixels.
perc = 0.7;
% random mask, Mask==1 for removed pixels
Mask = zeros(m,n);
Pick = randperm(m*n); Pick = Pick(1:round(perc*m*n));
Mask(Pick) = 1;
u0 = I;
u0(Mask == 1) = 0;
%imagesc(u0);

e = 1e-5; % tolerance
dt = 1e-7;

% choose one and comment the other one
u = Tik_inpainting(u0, Mask, e, dt); % Tikhonov inpainting
%u = TV_inpainting(u0, Mask, e, dt); % TV inpainting

% compute PSNR
mse = sum(sum((u-I).^2))/(m*n);
maxx = max(max(abs(I)));
psnr = 10*log10(maxx*maxx/mse);
fprintf('PSNR = %f \n', psnr);
% Compute relative error
re = norm(u-I,'fro')/norm(I,'fro');
fprintf('Relative error = %f \n', re);
toc
```

4.6 Problem 3.3

```
clear all;clc;
tic
I = imread('cameraman.tif');
I = double(I); % change I from unit8 to double precision format
% normalize I to intensity [0,1] for easier parameter selection
I = (I-min(I(:)))/(max(I(:))-min(I(:)));
[m,n] = size(I);

% Zero out intensity of I at scratch to get J
load scratch
J = I;
J(scratch == 0) = 0;
% Add noise
sigma = 0.1;
```

```
J = J + sigma*randn(size(J));

d0 = 3.5;
lambda = 10^d0;
e = 1e-5; % tolerance
dt = 1e-7;

% choose one and comment the other one
%u = Tik_denoising(J, scratch, e, dt, lambda); % Tik denoising
u = TV_denoising(J, scratch, e, dt, lambda); % TV denoising


% Compute PSNR
mse = sum(sum((u-I).^2))/(m*n);
maxx = max(max(abs(I)));
psnr = 10*log10(maxx*maxx/mse);
fprintf('PSNR = %f \n', psnr);
% Compute relative error
re = norm(u-I,'fro')/norm(I,'fro');
fprintf('Relative error = %f \n', re);
toc
```

4.7 Problem 3.4

```
clear all;clc;
tic
I = imread('cameraman.tif');
I = double(I); % change I from unit8 to double precision format
% normalize I to intensity [0,1] for easier parameter selection
I = (I-min(I(:)))/(max(I(:))-min(I(:)));
[m,n] = size(I);

% Zero out intensities of text
load text
u0 = I;
u0(text == 0) = 0.92;
% make sure missing intensities are 1, while other intensities are 2 here
text = text + 1;

% TV inpainting
e = 1e-5; % tolerance
dt = 1e-7;
u = TV_inpainting(u0, text, e, dt); % TV inpainting

% Compute PSNR
mse = sum(sum((u-I).^2))/(m*n);
maxx = max(max(I));
psnr = 10*log10(maxx*maxx/mse);
fprintf('PSNR = %f \n', psnr);
% Compute relative error
re = norm(u-I,'fro')/norm(I,'fro');
fprintf('Relative error = %f \n', re);
toc
```