Math 6590 Project II: Data denoising in 1D and 2D with total variation regularization: implementations with gradient flow, lagged diffusivity fixed point iteration and Chambolle projection methods

Name: Li Dong
RIN: 661243168

Mar 9, 2016

Abstract

In this report, the classical Rudin-Osher-Fatemi (ROF) model is used for 1D and 2D data denoising. Three methods, gradient flow, lagged diffusivity fixed point iteration and Chambolle projection, are implemented, where differentials were approximated with finite differences. A one-dimensional signal and two-dimensional image polluted with noise are used to validate the three methods. Total variation regularization (ROF model) appears superior for discontinuous reconstruction compared to the Tikhonov regularization in Project Report I last time. At last, the convergence of the three methods are compared. In the two numerical examples, Chambolle projection converges relatively faster than the other two methods in general. Compared to lagged diffusivity fixed point iteration, gradient flow performed better in the one-dimensional example and worse in the two-dimensional example.

1 Problems

In 1992, the revolutionary Rudin-Osher-Fatemi (ROF) model [1] injected new blood to the image processing business. It might be even too exaggerating to say that it is the model that starts the whole business. After the proposition of the suitable model, mathematicians immediately came up with numerous algorithms to tackle with the model. Due to the discontinuity in the denominator of the regularization term, ROF model proves to be quite hard to deal with numerically. A natural and simple yet a little bit awkward remedy is to add a small constant to make singularity vanish [2]. This is quite a popular method until a French mathematician Antonin Chambolle came up with an elegant approach [3] that, instead of dealing with the primal formulation of the total variation term, attacked its dual. It is noted that there are also mathematicians before Antonin Chambolle trying to attack the total variation term from its dual, such as in [4] and [5].

In the following sections, the gradient flow and lagged diffusivity fixed point iteration methods use the 'awkward' remedy that adds a small constant to the denominator of the regulation term and the Chambolle projection certainly deals with the dual of the total variation term without worrying about the singularity. Below are the requirements of this project.

Describe the following algorithms and use Matlab to implement the Rudin-Osher-Fatemi mdoel for 1D signal and 2D image denosing

$$\min_u \mathcal{E}(u) = \int_\Omega |Du| \dot{\mathrm{x}} + \frac{\lambda}{2} \int_\Omega (u - u_0)^2 \dot{\mathrm{x}} \tag{1}$$

1. Implement the gradient flow method using $\int_\Omega \sqrt{|\nabla u|^2+\epsilon}\ \mathrm{x}$ as an approximation of TV.
2. Solve the Euler-Lagrange equation $\frac{\delta \mathcal{E}}{\delta u}=0$ using the the lagged diffusivity fixed-point iteration.
3. Solve the above problem using Chambolle's dual projection method.
4. Use the attached data (a 1D signal and a 2D image) to test your codes.

2 Methodology

2.1 Gradient flow method

The method of gradient flow uses a 'fake' time evolution scheme and the ROF model becomes

$$\begin{aligned}\frac{\partial u}{\partial t} &= \nabla\left(\frac{\nabla u}{\sqrt{|\nabla u|^2+\alpha}}\right)+\lambda(u_0-u), &(2)\\ u(0,x) &= u_0(x), \quad x\in\Omega, &(3)\\ \frac{u}{\vec{n}}(t,x) &= 0, \quad x\in\partial\Omega. &(4)\end{aligned}$$

where u_0 is the noisy data and u is the reconstructed data. Then the iteration comes naturally as follows.

$$u^{n+1}=u^n+dt\left(\nabla^-\left(\frac{\nabla^+ u}{\sqrt{|\nabla^+ u|^2+\alpha}}\right)+\lambda(u_0-u)\right), \tag{5}$$

where $\nabla^+ u(i,j)=\left(\frac{u(i+1,j)-u(i,j)}{h},\frac{u(i,j+1)-u(i,j)}{h}\right), \nabla^- u(i,j)=\left(\frac{u(i,j)-u(i-1,j)}{h},\frac{u(i,j)-u(i,j-1)}{h}\right)$. The forward and backward difference are typically used together to cancel out the errors incurred by the difference operator. dt has to satisfy the stability condition $\frac{dt}{h^2}\le c$. α is a small constant to eliminate the singularity of the regularization term and takes the value of 1×10^{-10} here.

2.2 Lagged diffusivity fixed point iteration

Lagged diffusivity fixed point iteration is proposed by Vogel and Oman in [6]. In this method, a diffusion operator $L(u)$ is defined to be

$$L(u)v=-\nabla\cdot\left(\frac{\nabla v}{\sqrt{|\nabla u|^2+\alpha}}\right). \tag{6}$$

The iteration then is given by

$$(1+\lambda L(u^{(n)}))u^{(n+1)}=u_0, \quad n=1,2,\cdots. \tag{7}$$

In this report, finite difference is used to solve this diffusion equation. The discretization in 2D is illustrated in 1. For demonstration purpose, only 2D derivation is showed below and 1D only needs minor adaptation.

First, we let

$$v=\frac{\nabla u}{|\nabla u|}, \tag{8}$$

which gives

$$\nabla\cdot v\approx\frac{v_e-v_w}{h}+\frac{v_n-v_s}{h}, \tag{9}$$

where $v_e=\frac{u_x(e)}{\nabla u_e}\approx\frac{u_E-u}{h}$ and $|\nabla u_e|=\sqrt{u_x(e)^2+u_y(e)^2}\approx\frac{1}{h}\sqrt{(u_E-u)^2+(\frac{u_{NE}-u_{SE}+u_N-u_S}{4})^2}$. Now the iteration can be stated as

$$u^{n+1}=\sum_{Q\in\Lambda}h_q^n u_Q^n+h^n u_0, \tag{10}$$

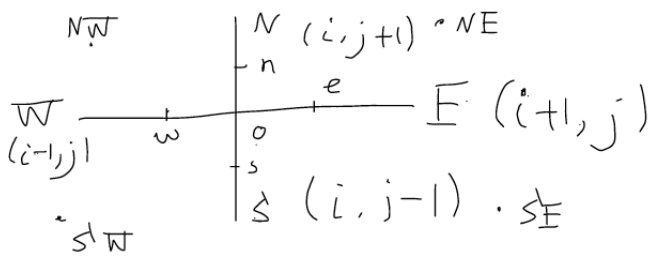


Figure 1: 2D discretization of the finite difference strategy for lagged diffusivity fixed point iteration.

where $h_q^n = \frac{d_q^n}{\sum_{Q\in\Lambda} d_q^n + \lambda}, h^n = \frac{\lambda}{\sum_{Q\in\Lambda} d_q^n + \lambda}, d_q^n = \frac{1}{|\nabla u_q|}, q \in \{n, s, e, w\}, Q \in \Lambda \equiv \{N, S, E, W\}$ and u_0 is the noisy data.

2.3 Chambolle projection

Chambolle projection is also named dual projection since it deals with the dual of the total variation term and the projection, in this case L_2, of the ROF model. The detailed derivation of this method is not discussed here and can certainly be found in [3]. The main iteration of the algorithm is

$$g^{n+1} = \frac{g^n + dtH(g^n)}{1 + dt|H(g^n)|}, \tag{11}$$

where $H(g) = -\nabla(u_0 - \frac{\nabla\cdot g}{\lambda})$ and g^1 can be initialized to 0. After the iteration converges, the reconstructed data $u = u_0 - \frac{\nabla\cdot g^n}{\lambda}$.

2.4 Remarks

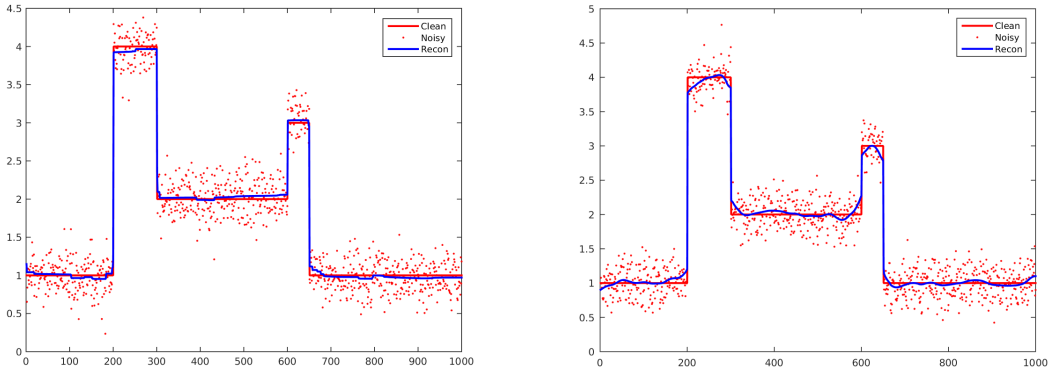
1. It is noted that, since the 1D and 2D data are distributed evenly with the spacing of one unit, the same discretization strategy ($h = 1$) is used for the above three methods.
2. For all three methods, the convergence criteria is set to 100,000 iterations or error tolerance of 1×10^{-6} in 1D and 10,000 iterations or error tolerance of 1×10^{-4} in 2D, whichever comes first terminates the algorithm. Error is defined to be $|\ ||u^n - u_0||_2 - ||u^{n-1} - u_0||_2\ |$.

3 Results

Apparently all three methods should give the same reconstruction given the same regularization parameter and terminated by the same error tolerance. Thus, only one result is displayed for 1D and 2D (see Figures 2 and 3), respectively.

For the 1D case, when $\lambda = 1$ the reconstructed signal is relatively accurate and there are some sharp edges along each plateau, which reflects the effect of the total variation and some people call it 'staircasing' effect (see Figure 2a); When λ decreases by half the reconstructed signal starts to deviate from the true signal and this is especially true for the corners, yet the sharp reconstruction still shows up in the transition parts (see Figure 2b).

For the 2D case, when $\lambda = 0.1$ the reconstructed image appears quite accurate and all the edges between different colors are preserved (see 3c); When λ, again, decreases by half the reconstructed image still looks good but start to look a little bit foggy or cartoon-like, which is the result of over-regularization of total variation (see Figure 3d).



(a) (b)

Figure 2: Results for ROF model of 1D signal, (a) $\lambda = 1$, (b) $\lambda = 0.5$

Figure 4 compare the convergence of the three methods for $\lambda = 1$ in one-dimensional data and $\lambda = 0.1$ in two-dimensional data.

For the noisy signal data, in Figure 4a, though lagged diffusivity fixed point iteration started from a relatively low error, Chambolle projection had a much faster convergence rate before around 70 iterations and achieved a lower error than the other two methods immediately after about 10 iterations. After roughly 70 iterations, gradient flow caught up quickly and reached convergence criteria first. Lagged diffusivity fixed point iteration started to converge relatively fast only after 10,000 iterations.

For the noisy image data, in Figure 4b, we can see that apparently the convergence rate of gradient flow is quite slow and lagged diffusivity fixed point iteration performs similar to Chambolle projection and only took about 1000 more iterations to reach convergence.

After profiling the temporal performance of the three algorithms, Chambolle projection method does not quite stand out. The possible reason can be that the problem size is not big enough, the implementation is not sufficiently mature or the coding style is not efficient.

4 Observation and Conclusions

1. Compared to the Tikhonov regularization in Project I, total variation regularization, or ROF model, is naturally superior in terms of reconstructing discontinuous data such as the image denoising where the pixel is most probably piece-wise constant.
2. The implementation of ROF model is not trivial. Compared to the straightforward way which adds a small constant to eliminate the singularity, such as the gradient flow and the lagged diffusivity fixed point iteration, the elegant formulation posed by Antonin Chambolle prevails in both theory and practice.
3. The results concluded from the convergence plots only reflect the performance of the algorithm for the specific implementation as well as for the specific problem and may not be general.

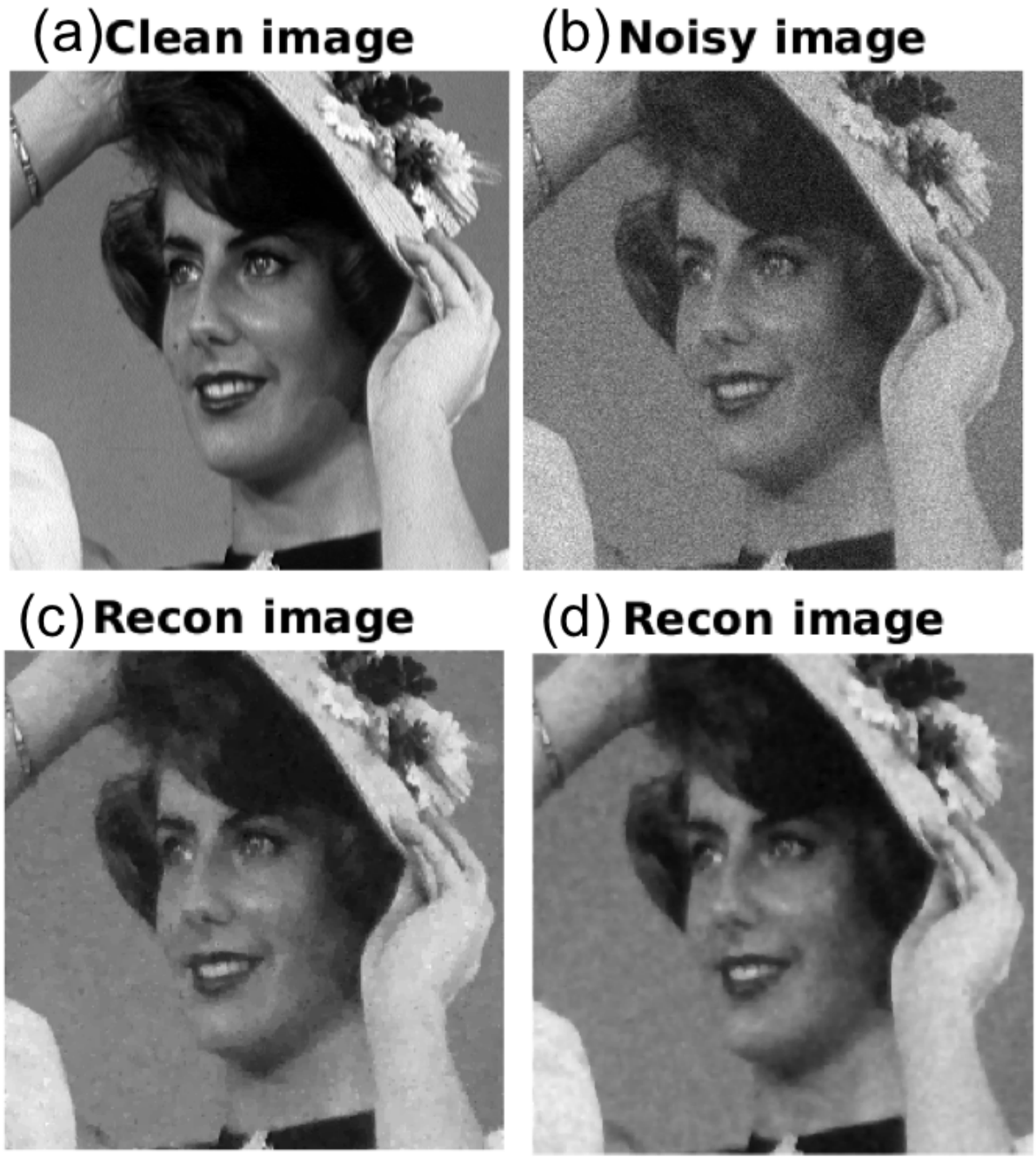
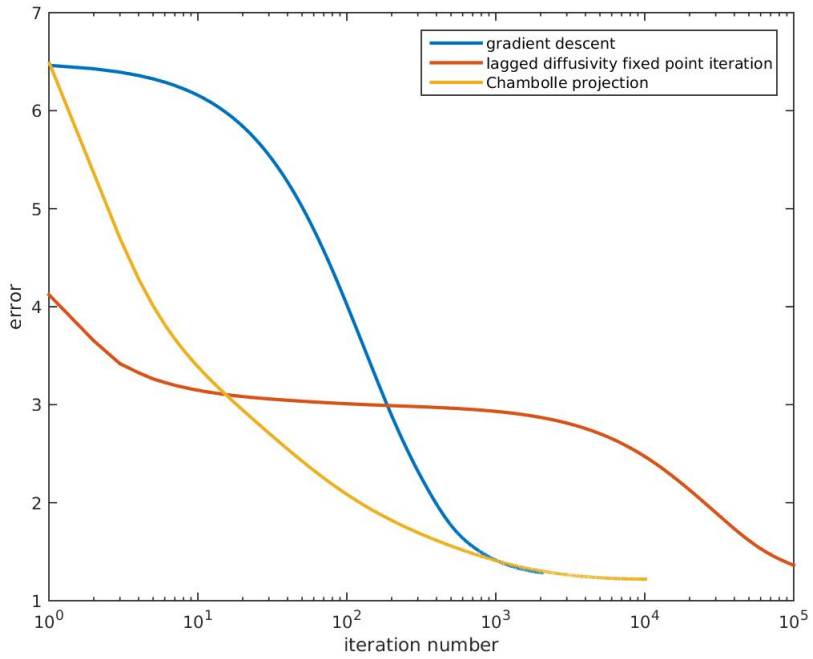
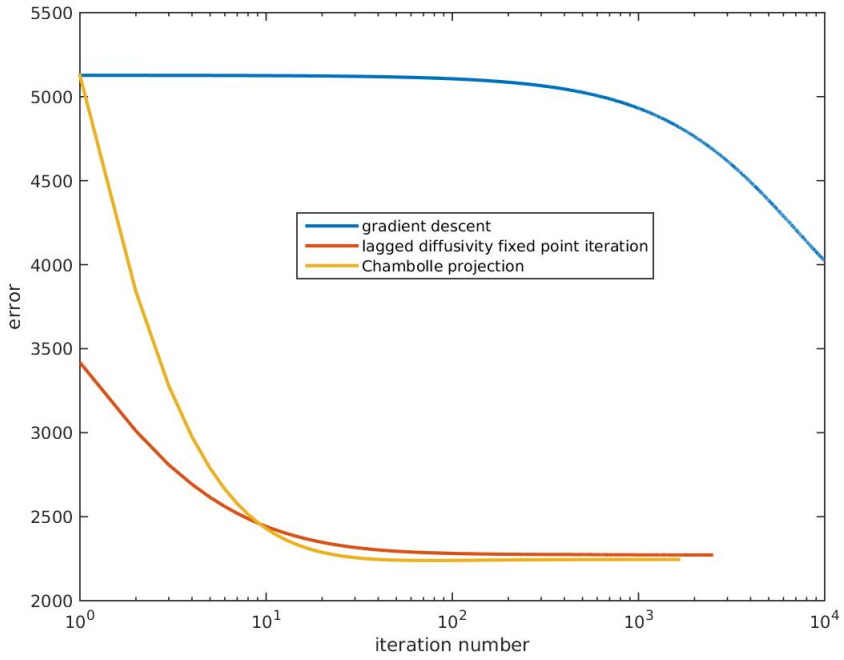


Figure 3: Results for ROF model of the 256x256 image (a) original image, (b) image with noise, (c) reconstructed image with $\lambda = 0.1$ and (d) $\lambda = 0.05$.



(a)



(b)

Figure 4: Convergence plots of the three methods for (a) 1D signal with $\lambda = 1$ and (b) 2D image with $\lambda = 0.1$.

References

[1] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[2] Curtis R Vogel. *Computational methods for inverse problems*, volume 23. Siam, 2002.

[3] Antonin Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical imaging and vision*, 20(1-2):89–97, 2004.

[4] Tony F Chan, Gene H Golub, and Pep Mulet. A nonlinear primal-dual method for total variation-based image restoration. *SIAM journal on scientific computing*, 20(6):1964–1977, 1999.

[5] Jamylle Laurice Carter. *Dual methods for total variation-based image restoration*. PhD thesis, University of California Los Angeles, 2001.

[6] Curtis R Vogel and Mary E Oman. Iterative methods for total variation denoising. *SIAM Journal on Scientific Computing*, 17(1):227–238, 1996.

Appendix

MATLAB codes.

main.m calls three methods and save results.

```
clear
close all
clc

[I1, I1_noise, I2, I2_noise] = LoadData;
tol=[1e-6 1e-4]; % relative error tolerance, [1D 2D]
% lambda=[5e-1 5e-2];
% iter=[1e6 1e4];
lambda=[1 1e-1];
iter=[1e5 1e4]; % iteration number, [1D 2D]
beta=1e-10; % small constant to make denominator of TV nonsingular for case 1 & 2
method=[1 2 3]; % 1: gradient descent;
                % 2: lagged diffusivity fixed point iteration;
                % 3: Chambolle projection
conv=[figure figure];
for i=1:length(method)
    switch method(i)
        case 1
            %% gradient descent
            [I1_smooth_GD, I2_smooth_GD, err1_GD, err2_GD] = ...
            gradDescent(I1_noise, I2_noise, lambda(1),lambda(2),beta,I1,I2,tol,iter);

            plot_creator(I1,I1_noise,I1_smooth_GD,I2,I2_noise,I2_smooth_GD,...
                        lambda,err1_GD,err2_GD,'GD',conv);

        case 2
            %% lagged diffusivity fixed point iteration
            [I1_smooth_LD, I2_smooth_LD,err1_LD,err2_LD] = ...
            lagDiff(I1_noise, I2_noise, lambda(1),lambda(2), beta, I1,I2,tol,iter);

            plot_creator(I1,I1_noise,I1_smooth_LD,I2,I2_noise,I2_smooth_LD,...
```

```
                        lambda,err1_LD,err2_LD,'LD',conv);

        case 3
            %% Chambolle projection
            [I1_smooth_CP, I2_smooth_CP,err1_CP,err2_CP] = ...
                Cproj(I1_noise, I2_noise,lambda(1),lambda(2),I1,I2,tol,iter);

            plot_creator(I1,I1_noise,I1_smooth_CP,I2,I2_noise,I2_smooth_CP,...
                        lambda,err1_CP,err2_CP,'CP',conv);

    end
end

figure(conv(1)); legend('gradient descent',...
                        'lagged diffusivity fixed point iteration',...
                        'Chambolle projection');
ylabel('error'); xlabel('iteration number'); saveas(gcf,'convergence_1D.jpg');
figure(conv(2)); legend('gradient descent',...
                        'lagged diffusivity fixed point iteration',...
                        'Chambolle projection','Location','best');
ylabel('error'); xlabel('iteration number'); saveas(gcf,'convergence_2D.jpg');
```

gradDescent.m implements the gradient flow method.

```
function [I1_s, I2_s, err1, err2]=gradDescent(I1_n,I2_n,lambda1,lambda2,beta,I1,I2,tol,iter)

% differential operator for 1D
nx = length(I1);
ex = ones(nx,1);
diff_f1=spdiags([-ex ex], [0 1], nx, nx); %forward difference
diff_b1=spdiags([-ex ex], [-1 0],nx, nx); %backward difference

% differential operator for 2D
[ny, nx] = size(I2);
e = ones(nx*ny,1);
diff_f2=spdiags([-e e], [0 1], nx*ny, nx*ny);
diff_b2=spdiags([-e e],[-1 0],nx*ny, nx*ny);

dt1 = 1e-3;
I1_k = I1_n;
err1 = 0;
tic;
for i=1:iter(1)
    I1_k_tmp = I1_k - dt1*(-diff_b1*( diff_f1*I1_k ./ sqrt((diff_f1*I1_k).^2+beta) )...
                           + lambda1*(I1_k-I1_n));
    err1 = [err1 norm(I1_k_tmp - I1)];
    if abs( err1(end)-err1(end-1) ) <tol
        fprintf('Gradient descent: 1D Reached tol @ iter %d!\n',i);
        break
    end
    I1_k = I1_k_tmp;
end
err1(1)=[];
t_1D_GD=toc/iter(1)
```

```
I1_s = I1_k;

dt2=1e-3;
I2_k = I2_n;
I2_k_tmp = zeros(size(I2_n));
err2 = 1e8;
tic
for i=1:iter(2)
    I2_kt=I2_k';
    I2_k_tmp(:) = I2_k(:) - dt2*(-diff_b2*( diff_f2*I2_k(:) ./ sqrt( (diff_f2*I2_k(:)).^2 ...
                                + (diff_f2*I2_kt(:)).^2 + beta) ) + lambda2*(I2_k(:)-I2_n(:)));
    err2 = [err2 norm(I2_k_tmp(:) - I2(:))];
    if abs( err2(end)-err2(end-1) )<tol
        fprintf('Gradient descent: 2D Reached tol @ iter %d!\n',i);
        break
    end
    I2_k = I2_k_tmp;
end
err2(1)=[];
t_2D_GD=toc/iter(2)
I2_s = I2_k;

end
```

lagDiff.m implements the lagged diffusivity fixed point iteration.

```
function [I1_s, I2_s, err1,err2]=lagDiff(I1_n,I2_n,lambda1,lambda2,beta,I1,I2,tol,iter)

I1_k=I1_n;
err1=0;
tic;
for i=1:iter(1)
    I1_k_n=circshift(I1_k,[ 1,0]);
    I1_k_s=circshift(I1_k,[-1,0]);
    I1_k_N = (I1_k-I1_k_n);
    I1_k_S = (I1_k-I1_k_s);
    dq_n = 1./sqrt(I1_k_N.^2+beta);
    dq_s = 1./sqrt(I1_k_S.^2+beta);
    dQ = (dq_n + dq_s);
    hq0_n = dq_n./(dQ+lambda1);
    hq0_s = dq_s./(dQ+lambda1);
    h00 = (lambda1)./(dQ+lambda1);

    I1_k_tmp=(hq0_n.*I1_k_n + hq0_s.*I1_k_s) + h00.*I1_n;
    err1=[err1 norm(I1-I1_k_tmp)];
    if abs( err1(end)-err1(end-1) )<tol(1)
        fprintf('Lagged diffusivity fixed point iteration: 1D Reached tol @ iter %d!\n',i);
        break
    end
    I1_k = I1_k_tmp;
end
t_1D_LD=toc/iter(1)
err1(1)=[];
I1_s = I1_k;
```

```
I2_k=I2_n;
err2=0;
tic;
for i=1:iter(2)
    I2_k_n =circshift(I2_k,[ 0, 1]);
    I2_k_s =circshift(I2_k,[ 0,-1]);
    I2_k_w =circshift(I2_k,[ 1, 0]);
    I2_k_e =circshift(I2_k,[-1, 0]);
    I2_k_ne=circshift(I2_k,[-1, 1]);
    I2_k_se=circshift(I2_k,[-1,-1]);
    I2_k_sw=circshift(I2_k,[ 1,-1]);
    I2_k_nw=circshift(I2_k,[ 1, 1]);

    dq_n = 1./ sqrt( (I2_k-I2_k_n).^2 + ((I2_k_ne-I2_k_nw+I2_k_e-I2_k_w)./4).^2 + beta);
    dq_s = 1./ sqrt( (I2_k-I2_k_s).^2 + ((I2_k_se-I2_k_sw+I2_k_e-I2_k_w)./4).^2 + beta);
    dq_w = 1./ sqrt( (I2_k-I2_k_w).^2 + ((I2_k_sw-I2_k_nw+I2_k_s-I2_k_n)./4).^2 + beta);
    dq_e = 1./ sqrt( (I2_k-I2_k_e).^2 + ((I2_k_se-I2_k_ne+I2_k_s-I2_k_n)./4).^2 + beta);

    dQ = dq_n + dq_s + dq_w + dq_e;
    hq0_n = dq_n./(dQ+lambda2);
    hq0_s = dq_s./(dQ+lambda2);
    hq0_w = dq_w./(dQ+lambda2);
    hq0_e = dq_e./(dQ+lambda2);
    h00 = lambda2./(dQ+lambda2);

    I2_k_tmp=hq0_n.*I2_k_n + hq0_s.*I2_k_s + hq0_w.*I2_k_w + hq0_e.*I2_k_e + h00.*I2_n;
    err2=[err2 norm(I2(:)-I2_k_tmp(:))];
    if abs( err2(end)-err2(end-1) )<tol(2)
        fprintf('Lagged diffusivity fixed point iteration: 2D Reached tol @ iter %d!\n',i);
        break
    end
    I2_k = I2_k_tmp;
end
t_2D_LD=toc/iter(2)
err2(1)=[];
I2_s = I2_k;

end
```

Cproj.m implements the Chambolle projection method.

```
function [I1_s, I2_s, err1, err2]=Cproj(I1_n,I2_n,lambda1,lambda2,I1,I2,tol,iter)

err1=0;
% dt1=1e-4; % for smaller reg param
dt1=1e-1;
g1=zeros(size(I1_n));
tic;
for i=1:iter(1)
   H=-grad(I1_n-div(g1)/lambda1);
   g1_tmp=(g1+dt1*H)./(1+dt1*abs(H));
   I1_k=I1_n-div(g1)/lambda1;
   err1=[err1 norm(I1_k-I1)];
```

```
   if abs( err1(end)-err1(end-1) )<tol(1)
        fprintf('Chambolle Projection: 1D Reached tol @ iter %d!\n',i);
        break
    end
   g1=g1_tmp;
end
t_1D_CP=toc/iter(1)
err1(1)=[];
I1_s=I1_k;

err2=0;
% dt2=1e-5; % for smaller reg param
dt2=1e-2;
g2=zeros(size(I2_n,1),size(I2_n,2),2);
tic;
for i=1:iter(2)
   H=-grad(I2_n-div(g2(:,:,1),g2(:,:,2))/lambda2);
   g2_tmp=(g2+dt2*H)./(1+dt2*abs(H));
   I2_k=I2_n-div(g2(:,:,1),g2(:,:,2))/lambda2;
   err2=[err2 norm(I2_k(:)-I2(:))];
   if abs( err2(end)-err2(end-1) )<tol(2)
        fprintf('Chambolle Projection: 2D Reached tol @ iter %d!\n',i);
        break
    end
   g2=g2_tmp;
end
t_2D_CP=toc/iter(2)
err2(1)=[];
I2_s=I2_k;

function [fx] = grad(M)

nbdims = 2;
if size(M,1)==1 || size(M,2)==1
    nbdims = 1;
end

fx = M([2:end end],:)-M;

if nbdims>=2
    fy = M(:,[2:end end])-M;
end

if nbdims==2
    fx = cat(3,fx,fy);
end

function fd = div(Px,Py)

nbdims = 2;
if size(Px,1)==1 || size(Px,2)==1
    nbdims = 1;
end
```

```
fx = Px-Px([1 1:end-1],:);
if nbdims>=2
    fy = Py-Py(:,[1 1:end-1]);
end

if nbdims==2
    fd = fx+fy;
else
    fd = fx;
end
```

plot_creator.m is a helper function that makes all the plots.

```
function plot_creator(I1,I1_noise,I1_smooth,I2,I2_noise,I2_smooth,lambda,err1,err2,method,fig_h)

figure;
plot(I1,'r-','LineWidth',2);
hold on;
plot(I1_noise,'r.');
plot(I1_smooth,'b-','LineWidth',2);
hlen = legend('Clean','Noisy','Recon');
img_name=sprintf('1D_%s_%g.png',method,lambda(1));
saveas(gcf,img_name);
hold off

figure;
subplot(1,3,1);
imshow(I2,[]);
title('Clean image');
subplot(1,3,2);
imshow(I2_noise,[]);
title('Noisy image');
subplot(1,3,3);
imshow(I2_smooth,[]);
title('Recon image');
img_name=sprintf('2D_%s_%g.png',method,lambda(2));
saveas(gcf,img_name);

figure(fig_h(1)); plot(1:length(err1),err1,'linewidth',2); set(gca,'xscale','log');
hold on

figure(fig_h(2)); plot(1:length(err2),err2,'linewidth',2); set(gca,'xscale','log');
hold on

end
```