INSA Toulouse
GMM 5ème année, Image

Lesson 7 : In the jungle of proximal algorithms

Introduction

During the last decade, each month, several *new* proximal algorithms are proposed. Most of these algorithms are designed to minimize a convex function F, defined from an Hilbert space E to $\mathbb{R} \cup +\{\infty\}$, as a sum of several convex functions :

$$F(x) = f(x) + \sum_{k=1}^{M} g_k(L_k x) \tag{1}$$

where L_k are linear operators, where g_k can be non smooth and f may be smooth or not. Most of them deals with specific cases, $f = 0$ or $L_k = Id$ or $M = 1$... It is literally impossible and useless to present all of them, see [**? ?**] for a more complete analysis. Some are variations of classical ones, Forward-Backward, Douglas-Rachford, or Primal Dual, and some are more complicated. All these algorithms share the property that they are iterative algorithms using the proximity operator of functions g and the applications of L_k and their adjoined L_k^*.

We recall here the definition of the proximity of the convex function g :

$$prox_g(x) = \underset{z \in \mathbb{R}^n}{\arg\min}\, g(z) + \frac{1}{2} \|z - x\|_2^2 \tag{2}$$

For most algorithms, only convergence proofs of iterates $(x_k)_{k \in \mathbb{N}}$ are available, for some of them such as Forward-Backward or FISTA, bounds on $F(x) - F(x^*)$, where x^* is a minimizer, are given. Hence for a given optimization problem, several algorithms may be available, but unfortunately, it is often hard to predict which one is the best. Moreover, some problems can be reformulated into equivalent ones, using for example duality, or dimension extension. It follows that a single algorithm can be used in different way to solve a given problem. In this document we try describe several classical algorithms and techniques to solve optimization problems using proximal algorithms.

1 Computation of proximity operators

Some proximity operators are widely used and have explicit closed form see [**?**] or the website http ://proximity-operator.net. Others may be computed via calculus rules. We give here some useful examples.

1.1 Explicit form of proximity operator.

Let's give 3 canonical examples of proximity operators :

— Let C be a closed convex set and g its indicator function defined by $g(x) = \iota_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases}$
The proximity operator of g is the projection on C.

— If g is a differentiable function, the proximity operator of γg is an implicit gradient descend on g with step γ.

— If $g(x) = \lambda \|x\|_1$ then $prox_g$ is a soft thresholding with threshold λ.

We will focus now on three computations of proximity operators useful in signal and image processing.

1.1.1 Proximity operator of a quadratic function

In many problems coming from statistics and image processing, we have to minimize functions with a quadratic term : $f(x) = \frac{\lambda}{2}\|Ax - b\|_2^2$. There is an explicit form for the proximity operator :

$$prox_f(x) = (Id + \lambda A^*A)^{-1}(x + \lambda A^*b) \tag{3}$$

Hence, the computation of this proximity operator is an inversion of a linear system.
In any cases the matrix $Id + \lambda A^*A$ is real and symmetric and a conjugate gradient may be used to solve the system. But be careful, most of the time the matrix A is not available, we may only be able to apply A and A^*. It exists some situations where this system can be solved exactly :

— If A is a circular convolution by a filter h. In the Fourier domain $(Id + \lambda A^*A)^{-1}$ is a simple multiplication by $(1 + \lambda|\hat{h}(k)|^2)^{-1}$.

— If A is a derivative operator like a discrete gradient, then if the derivative is circular, that may create strange boundary effect, it actually is a convolution and one may apply the previous formula. If the derivative is not shift invariant, the inversion in the Fourier domain is only an approximation and a conjugate gradient may provide more precise results.

— If A is a masking operator the operator $(Id + \lambda A^*A)$ is diagonal in the pixel domain and the inversion of the system is a simple division in the space domain.

Take away message : It is always possible to *prox* a quadratic function but depending on the function, it may be very fast or not.

1.1.2 Separable proximity operator

If a function f is separable, the proximity operator may be computed by blocks or even component by component. Hence if $x = (x_1, x_2)$ and if $f(x) = f_1(x_1) + f_2(x_2)$ then

$$prox_f(x) = \underset{z=(z_1,z_2)\in\mathbb{R}^n}{\arg\min}\ f_1(z_1) + f_2(z_2) + \frac{1}{2}\|z - x\|_2^2 = \underset{z=(z_1,z_2)\in\mathbb{R}^n}{\arg\min}\ f_1(z_1) + \frac{1}{2}\|z_1 - x_1\|_2^2 + f_2(z_2) + \frac{1}{2}\|z_2 - x_2\|_2^2 \tag{4}$$

it follows that

$$prox_f(x) = (prox_{f_1}(x_1), prox_{f_2}(x_2)) \tag{5}$$

This remark also applies to a sum of M functions $f(x) = \sum_{i=1}^M f_i(x_i)$, the proximity operator of f at the point x is a vector whose components p_i are $prox_{f_i}(x_i)$. A classical application of this rule is the proximity operator of $= g(x) = \|x\|_1$ which is computed component by component.

This simple remark is the corner stone of the dimension extension trick and particularly of the PPXA (Parallel ProXimal Algorithm) that will be described further.

1.1.3 Proximity operator of indicator functions of vectorial subspaces

Let A a linear operator defined from $\mathbb{R}^n$ to $\mathbb{R}^d$ and $K = \{(x_1, x_2)\ \text{such that}\ Ax_1 = x_2\}$ a vectorial subspace of $\mathbb{R}^{n+d}$. The proximity operator of the indicator function ι_K of K is defined by

$$(p_1, p_2) = prox_{\iota_K}(x_1, x_2) = \underset{(z_1,z_2)}{\arg\min}\frac{1}{2}\|z_1 - x_1\|_2^2 + \frac{1}{2}\|z_2 - x_2\|_2^2 + \iota_K(z_1, z_2) \tag{6}$$

It follows that $p_2 = Ap_1$ with

$$p_1 = \underset{z_1}{\arg\min}\frac{1}{2}\|z_1 - x_1\|_2^2 + \frac{1}{2}\|Az_1 - x_2\|_2^2 = (Id + A^*A)^{-1}(x_1 + A^*x_2) \tag{7}$$

Once again, the computation of this proximity operator may be fast or not depending on A. For example if $A = Id$ then $p_1 = \frac{1}{2}(x_1 + x_2)$ and $p_2 = p_1$. If we are able to diagonalize the matrix $I + A^*A$, for circular convolutions or circular derivative operators, the computation may be fast. If not, one needs to use an algorithm to solve this linear system, like conjugate gradient.

1.2 Computation rules from known proximity operator

1.2.1 Moreau Identity

As we will see further, many minimization problems can be associated to saddle point problems replacing the use of f by the use of its Fenchel conjugate f^*. Thus, to solve these saddle point problems, the proximity operator of f^* is needed. The Moreau Identity links the proximity operator of f and f^* .

The conjugate f^* of a function f is defined as follow :

$$f^*(u) = \sup_{x \in E} \langle x, u \rangle - f(x)$$

The function f^* describes the affine minorants of the function f. If f is convex, f is equal to the conjugate of f^* i.e $f = f^{**}$.

A quick computation (Exercice) shows that if $(x,u) \in E^2$

$$u \in \partial f(x) \Leftrightarrow f^*(u) + f(x) = \langle u,x \rangle \tag{8}$$

If f is convex, $f = f^{**}$ and then by symmetry we deduce that

$$u \in \partial f(x) \Leftrightarrow f^*(u) + f(x) = \langle u,x \rangle \Leftrightarrow f^*(u) + f^{**}(x) = \langle u,x \rangle \Leftrightarrow x \in \partial f^*(u) \tag{9}$$

Moreover, the proximity operator is defined by differential inclusion : If for $x \in E$ we denote $p = prox_f(x)$ and $q := x - p$ we have by the definition of the proximity operator

$$p = prox_f(x) \Leftrightarrow x - p \in \partial f(p)$$

Hence if $p = prox_f(x)$ we have $q \in \partial f(p)$ which is equivalent to $p \in \partial f^*(q)$ which implies that $q = prox_{f^*}(x)$. Which is the Moreau Identity :

Theorem 1. *If f is convex, proper and sci then*

$$prox_{f^*} + prox_f = Id. \tag{Moreau}$$

1.3 Composition with scaling and decomposition in orthogonal bases and Tight Frame operators

As an exercice...

1. If T is a linear operator such that $TT^* = Id$ but not necessarily such that $T^*T = Id$ (T is a tight frame but may be an orthogonal basis decomposition or not) and if $h = g \circ T^*$ then

$$prox_h(x) = T \circ prox_g \circ T^*(x)$$

   Examples of such operators T^* are overcomplete (or "à trous") wavelet decomposition associating to an image $x \in \mathbb{R}^n$ a vector of coefficients $c = T^*x \in \mathbb{R}^N$ with $N > n$, or some curvelet decompositions or decomposition in a union of orthogonal bases (be careful with the scaling factor).

2. If $\lambda > 0$ and f is a convex function , we define g by $g(z) = f(\lambda z)$, then

$$prox_{\lambda^2 f}(x) = \lambda prox_g(\frac{x}{\lambda})$$

2 Classical and useful algorithms

2.1 Forward-Backward algorithm and FISTA

If $F = f + g$, f is differentiable and ∇f is $L-Lipschitz$, for $\gamma > 0$ we define :

$$T = prox_{\gamma g}(Id - \gamma \nabla f) \tag{10}$$

2.1.1 Forward-Backward (FB)

The Forward-Backward algorithm is defined, if $\gamma < \frac{1}{L}$ by

$$x_{n+1} = Tx_n \tag{FB1}$$

or by

$$x_{n+1} = \lambda x_n + (1-\lambda)Tx_n \tag{FB2}$$

with any $\lambda \in (0,1)$ if $\gamma < \frac{2}{L}$.

A few properties about FB :

— The sequence $(F(x_n))_{n\in\mathbb{N}}$ is non increasing.

— The sequence $(x_n)_{n\in\mathbb{N}}$ converges to a minimizer x^* of F if F is sci, proper and convex and coercive.

— $F(x_n) - F(x^*) = O\left(\frac{1}{n\gamma}\right)$

— If F is strongly convex, the decay is linear : it exists $\kappa < 1$ such that $F(x_n) - F(x^*) = O(\kappa^n)$, but κ may be very close to 1.

This algorithm is explicit on f and implicit on g.

2.1.2 Fast iterative Thresholding Algorithm (FISTA)

FISTA [**? ?**] may be used in the same setting as FB and is based on the same operator T. The sequence is generated as follow for $\gamma < \frac{1}{L}$ and $\alpha \geqslant 3$, see [**?**]

$$x_{n+1} = Ty_n \quad \text{with } y_n = x_n + \frac{n}{n+\alpha}(x_n - x_{n-1}). \tag{FISTA}$$

A few remarks about FISTA

— The original algorithm by Beck and Teboulle [**?**] inspired by Nesterov [**?**] uses only $\alpha = 3$.

— The sequence $(F(x_n))_{n\in\mathbb{N}}$ may not be not increasing and in many practical cases, there are oscillations.

— The sequence $(x_n)_{n\in\mathbb{N}}$ converges to a minimizer x^* of F if F is sci, proper and convex and coercive when $\alpha > 3$, see [**?**].

— A choice of α not loo large, close to 3 is often relevant.

— Each time FB can be used, it it relevant to try FISTA which is often faster.

2.2 Douglas-Rachford

If $F = f + g$ and both proximity operator of f and g can be computed, the Douglas-Rachford algorithm can be used [**? ?**]. This algorithm is based on the reflected Prox :

$$rprox_f = 2prox_f - Id. \tag{rprox}$$

The algorithm can be stated as follow for any $\lambda \in (0,1)$ and any $\gamma > 0$ we compute the sequence $(x_n)_{n\in\mathbb{N}}$

$$x_{n+1} = \lambda x_n + (1-\lambda) rprox_{\gamma f} \circ rprox_{\gamma g}(x_n)$$

and the sequence $(u_n)_{n\in\mathbb{N}}$ defined by $u_n = prox_{\gamma g}(x_n)$ converge (weakly) to a minimizer of F.

A few remarks on the Douglas-Rachford algorithm :

— It is a full implicit algorithm needed the knowledge of the prox of both functions. It can be an advantage because it can be used when neither functions are differentiable and a drawback because sometimes it is easier to compute an explicit step.

— In the algorithm we compute a sequence $(x_n)_{n\in\mathbb{N}}$ but this sequence does not converge to a minimizer of F. We need to build $u_n = prox_{\gamma g}(x_n)$ at the end.

— There is no condition on $\gamma > 0$ to ensure the convergence but the choice of γ may be crucial on the numerical convergence rate.

— There are no convergence rate and no known bounds on $F(u_n) - F(x^*)$, but the DR algorithm may be very fast.

— It can be shown (Exercice) that this algorithm can be written

$$\begin{cases} y_{n+1} = & prox_{\gamma g}(x_n) \\ z_{n+1} = & prox_{\gamma f}(2y_{n+1} - x_n) \\ x_{n+1} = & x_n + 2(1-\lambda)(z_{n+1} - y_{n+1}) \end{cases} \tag{11}$$

2.3 Parallel ProXimal Algorithm (PPXA)

PPXA [**?**] uses the Douglas-Rachford algorithm in a smart way. It deals with more than two functions and all the proximity operators can be computed in a parallel way. This algorithm is based on the dimension extension. Hence, even when we consider only two functions and apply PPXA, PPXA is different from DR.

Let's consider the following minimization problem

$$\min_{x\in E} F(x) := \sum_{i=1}^{M} f_i(x) \tag{12}$$

where all functions f_i are convex, proper and sci and we suppose that this optimization problem has at least one solution.

We can reformulate this problem as an optimization problem in E^n. Each vector X of E^M can be written $X = (x_1, x_2, \cdots, x_M)$ where $x_i \in E$ for all $i \leqslant M$. More precisely the previous problem is equivalent to the following one :

$$\min_{X=(x_i)_{i\leqslant n}\in E^M} F(X) := \sum_{i=1}^{M} f_i(x_i) + \iota_D(X) \tag{13}$$

where $D = \{X = (x_k)_{k\leqslant M} \in E^M \text{ such that } x_k = x_1, \forall k \leqslant M\}$.

To solve this problem we can apply DR to $F = f + g$ with $f(X) = \sum_{i=1}^{M} f_i(x_i)$ and $g(X) = \iota_D(X)$.

The proximity operator of f can be computed in a parallel way using the separability of f, that is using the extension of equality (5). And the proximity operator of D is a simple mean as it was proved previously.

Some remarks on PPXA

— The algorithm is ... parallel and may be faster than DR even for two functions.

— This algorithm is also fully implicit with its advantages and drawbacks.

— The choice of the step γ may be crucial for the convergence rate.

— If functions f_i for $i \geqslant 2$ are composed with a linear operator :

$$\min_{x\in E} F(x) := f_1(x) + \sum_{i=2}^{M} f_i(L_i x) \tag{14}$$

the problem can also be reformulated using the dimension extension

$$\min_{X=(x_i)_{i\leqslant M}\in E^n} F(X) := \sum_{i=1}^{M} f_i(x_i) + \iota_{D_2}(X) \tag{15}$$

where $D_2 = \{X = (x_k)_{k\leqslant M} \in E^M \text{ such that } L_k x_1 = x_k, \forall k \leqslant M\}$. We can use DR but this time, one needs to perform the projection on D_2. It exists a closed form similar to (7) : if we denote $(p_1, p_2, \cdots, p_n) = Proj_{D_2}(X)$ then $\forall k \geqslant 2$ we have $p_k = L_k p_1$ and

$$p_1 = \left(Id + \sum_{k=2}^{M} L_k^* L_k\right)^{-1} (x_1 + \sum_{k=2}^{M} L_k^* x_k).$$

The problem is : How to perform such an inversion ? It may depend on operators L_k. One can always use a conjugate gradient if no diagonalization basis is known.

— Prove that for the optimization problem with two functions $F = f + g$, PPXA consists in computing two sequences $(x_n^1)_{n\in\mathbb{N}}$ and $(x_n^2)_{n\in\mathbb{N}}$ by :

$$\begin{cases} x_{n+1}^1 = & x_n^1 + 2(1-\lambda)\left(prox_{\gamma f}(x_n^2) - \frac{x_n^1+x_n^2}{2}\right) \\ x_{n+1}^2 = & x_n^2 + 2(1-\lambda)\left(prox_{\gamma g}(x_n^1) - \frac{x_n^1+x_n^2}{2}\right) \end{cases} \tag{16}$$

and considering $x_n = \frac{x_n^1+x_n^2}{2}$.

2.4 Chambolle Pock Primal Dual, a key algorithm.

Let's consider now the optimization problem :

$$\min_{x\in E} F(x) = f(Kx) + g(x) \tag{17}$$

where K is a linear operator and f and g two convex proper sci functions which proximity map are known.

The Primal Dual algorithm proposed by Chambolle and Pock [**?**] solves the previous problem dealing separately functions f and g and the operator K. If we denote $L = |||K|||$ the operator norm of K, the Primal Dual algorithm is defined for any (σ,τ) real non negative numbers satisfying $\sigma\tau L^2 < 1$:

$$\begin{cases} y_{n+1} = & prox_{\sigma f^*}(y_n + \sigma K\bar{x}_n) \\ x_{n+1} = & prox_{\tau g}(x_n - \tau K^* y_{n+1}) \\ \bar{x}_{n+1} = & 2x_{n+1} - x_n \end{cases} \tag{PD}$$

One can observe that the proximity operator of σf^* is needed but it can be expressed using the proximity operator of f and the Moreau Identity (Moreau)

$$prox_{\sigma f^*}(x) = x - \sigma prox_{\sigma^{-1} f}(\sigma^{-1}x) \tag{18}$$

This algorithm aims at solving a saddle point Primal Dual problem that is equivalent to (17). The idea is to replace the expression of f using its conjugate :

$$f(Ku) = \sup_{z\in E}\langle Ku, z\rangle - f^*(z) \tag{19}$$

Hence if x^* is a solution of optimization problem (17) then it exists a dual vector $y^* \in E$ such that (x^*, y^*) is a saddle point of the following optimization problem :

$$\min_{x\in E}\sup_{y\in E} g(x) + \langle Kx, y\rangle - f^*(y) \tag{20}$$

This last problem is called a Primal Dual problem and is solved by the Chambolle Pock algorithm. The sequence $(x_n)_{n\in\mathbb{N}}$ converges (weakly) to a minimizer of F.

Some remarks about the Primal Dual algorithm

— This algorithm (PD) is implicit in f and g but does not need any inversion of $(Id + K^*K)$.

— PD applies in many situations and does not need internal loop for any operator K.

— If we denote $h(x,y) = g(x) + \langle Kx, y\rangle - f^*(y)$, Chambolle and Pock proved that

$$h(x_n, y^*) - h(x^*, y_n) \leqslant \frac{1}{n}\left(\frac{\|y^* - y_0\|_2^2}{2\sigma} + \frac{\|x^* - x_0\|_2^2}{2\tau}\right) \tag{21}$$

— The choice of σ may be crucial for the actual convergence rate of $F(x_n) - F(x^*)$ but there is no generic rule to choose them. The optimal parameter may depend on the starting points.

— Some variations of PD have been proposed to allow an explicit step on one of the two functions or to deal with more than two functions. But to deal with this general situation we will see another algorithm proposed by Laurent Condat.

2.5 The Primal Dual algorithm proposed by Condat

The following algorithm du to Condat [**?**] aims to solve the general problem

$$F(x) = f(x) + g(x) + \sum_{m=1}^{M} h_k(L_m x). \tag{22}$$

where f is a differentiable function which gradient is L-Lipschitz and functions g and h_m have known proximity operators and L_k are linear operators. Let $\rho \in (0,1]$ and $\sigma > 0$, $\tau > 0$ such that

$$\tau \left(\frac{L}{2} + \sigma ||| \sum_{m=1}^{M} L_m^* L_m ||| \right) < 1 \tag{23}$$

Let's choose initial points $x_0 \in E$ and $(u_0^m)_{m \leqslant M}$ initial dual variables and

```
for $k = 0$ to $n-1$ do
    $\tilde{x}_{k+1} = prox_{\tau g}\left(x_k - \tau\nabla f(x_k) - \tau\sum_{m=1} L_m^* u_k^m\right)$
    $x_{k+1} = \rho\tilde{x}_{k+1} + (1-\rho)x_k$
    for $m = 1$ to $M$ do
        $\tilde{u}_{k+1}^m = prox_{\sigma h_m^*}\left(u_k^m + \sigma L_m(2\tilde{x}_{k+1} - x_k)\right)$
        $u_{k+1}^m = \rho\tilde{u}_{k+1}^m + (1-\rho)u_k^m$
    end
end
```

The sequence $(x_k)_{k\in\mathbb{N}}$ (weakly) converges to a minimizer of F.
Some remarks about this Primal Dual algorithm

- This algorithm allows to use an explicit gradient descend on the differentiable part, like Forward-Backward.
- It also deals separately all functions and operators, like the Chambolle Pock algorithm.
- The computations of $(u_k^m)_{m\leqslant M}$ are independent and can be done separately and allow parallel computing like PPXA.

3 Various optimization problems, various algorithms.

For any problem, several algorithms may be available. We won't give any rule to choose a priori one among the others, but it is important to know which algorithms can be used directly or using reformulation. By reformulation we think mainly about dualisation and dimension extension.

Let's give some examples, each time, we want to minimize the function F in $\mathbb{R}^n$. We will always suppose that a minimizer exists. Note that the Primal Dual algorithm by Condat applies in all situations without any inner loop and most of the time Chambolle Pock Primal Dual algorithm applies. Indeed many of these examples have been solved in [**?**].

1. Inpainting, with small gaussian noise with orthogonal wavelet regularization $T^*T = TT^* = Id$

   $$F(x) = \frac{1}{2}\|y - Mx\|_2^2 + \lambda\|Tx\|_1$$

   Here, we can use all the previous algorithms, using the proximity operator of $g(x) = \|Tx\|_1$ and an explicit or an implicit descent on $f(x) = \frac{1}{2}\|y - Mx\|_2^2$.

2. Deconvolution with small gaussian noise with orthogonal wavelet regularization $T^*T = TT^* = Id$

   $$F(x) = \frac{1}{2}\|y - h\star x\|_2^2 + \lambda\|Tx\|_1$$

   We can also add an undersampling operator to deal with the zooming problem. Same conclusions.

3. TV Denoising with gaussian noise.

$$F(x) = \frac{1}{2} \|y-x\|_2^2 + \lambda \|\nabla x\|_1 \tag{24}$$

The minimization of F is the computation of the proximity operator of $g(x) = \|\nabla x\|_1$. It can be computed directly by Primal Dual algorithms (Chambolle Pock and Condat). We can reformulate this problem into

$$\tilde{F}(x,z) = \underbrace{\frac{1}{2} \|y-x\|_2^2 + \lambda \|z\|_1}_{f} + \underbrace{\iota_D(x,z)}_{g} \text{ with } D = \{(x,z) \text{ such that } \nabla z = x\}$$

to apply DR or PPXA. The projection on D can be computed using conjugate gradient (needing inner loops) or directly if the gradient is periodic because the proximity operator can be computed in an explicit way in the Fourier domain.
Moreover we can build a dual problem associated to (24) :

$$G(p) = \frac{1}{2} \|y + div(p)\|_2^2 + \iota_{\mathscr{B}}(p) \quad \text{where } \mathscr{B} = \{p \text{ such that } \|p\|_\infty \leqslant \lambda\} \tag{25}$$

If p^* is a minimizer of (25) then $x^* = y + div(p^*)$ is a solution of (24) .
This dual problem can be solved by all presented algorithms including FB and FISTA.

4. TV Denoising with salt and paper noise.

$$F(x) = \frac{1}{2} \|y-x\|_1 + \lambda \|\nabla x\|_1$$

Both PD algorithms can be used here. We can also reformulate the problem as previously using dimension extension :

$$\tilde{F}(x,z) = \underbrace{\|y-x\|_1 + \lambda \|z\|_1}_{f} + \underbrace{\iota_D(x,z)}_{g} \text{ with } D = \{(x,z) \text{ such that } \nabla z = x\}$$

Once again DR or PPXA may be used.

5. Partial Radon measurements inversion. We consider here a Radon operator R where measurements are done only on a small number of angles : $y = Rx^0 + b$. If we assume that the noise b is gaussian and the target x^0 has a small TV norm, the inversion can be done minimizing

$$F(x) = \frac{1}{2} \|y - Rx\|_2^2 + \lambda \|\nabla x\|_1$$

Let's recall that R^* is the back-projection associated to R (without any filtering). It is not impossible to compute to proximity operator of $f(x) = \frac{1}{2} \|y - Rx\|_2^2$ but it's a bit tricky. From a theoretical point of view R^*R is a simple ramp filter but the interpolation step during the computations of R and R^*. That is why I recommend to use an explicit descent on f. Here the PD algorithm by Condat seems to be the more efficient : explicit on f and use the proximity operator of the ℓ_1 norm and the computation of discrete gradient and discrete divergence.

6. Inpainting with gaussian noise using TV regularisation :

$$F(x) = \frac{1}{2} \|y - Mx\|_2^2 + \lambda \|\nabla x\|_1$$

FB and FISTA are possible options but the proximity operator of $\lambda \|\nabla x\|_1$ will necessitate inner loops. PPXA is also possible with the following reformulation

$$\tilde{F}(x,z) = \underbrace{\|y - Mx\|_2^2 + \lambda \|z\|_1}_{f} + \underbrace{\iota_D(x,z)}_{g} \text{ with } D = \{(x,z) \text{ such that } \nabla z = x\}$$

which also necessitate inner loops except if the gradient ∇ is periodic.
Both PD algorithms can be used without inner loops.

7. Inpainting with gaussian noise by regularization of tight frame coefficients (for example frame of wavelets). Such regularizations may give better results that the one given classical wavelet orthogonal bases. If $x \in \mathbb{R}^n$ is an image, $c = T^*x \in \mathbb{R}^N$ are the wavelet coefficients. And $N > n$, if T is a tight frame operator then $Tc = x$ that is $TT^* = Id$ but $T^*T \neq Id$ because $N > n$. To recover x from $y = Mx + b$, we first estimate $c^* = T^*x$ minimizing

$$F(c) = \frac{1}{2} \|y - MTc\|_2^2 + \lambda \|c\|_1$$

Then we estimate the image x^* by $x^* = Tc^*$.
This approach is also available for any tight frame, for example Curvelet Tight frames.
To compute the proximity operator of $f(x) = \frac{1}{2} \|y - MTc\|_2^2$ one needs to inverse the operator $Id + \gamma T^*MT$ which is not so simple. That is why I recommend to compute an explicit descent on f which only necessitate to apply T and T^*. Here I recommend FB, FISTA or the PD of Condat.
Another way to solve this problem is to consider the following optimization problem

$$\tilde{F}(x) = \frac{1}{2} \|y - Mx\|_2^2 + \lambda \|T^*x\|_1$$

and to use any PD algorithm (Chambolle Pock or Condat) considering $K = T^*$, $f(x) = \|x\|_1$ and $g(x) = \frac{1}{2} \|y - Mx\|_2^2$.

8. Image fusion (Poisson Image Editing [**?**] : We consider a source image s and a target image y and a mask Ω. If we want to insert in a smooth way the source in the target image. the mask we can minimize the following function f.

$$F(x) = \frac{1}{2} \|\nabla s - \nabla x\|_\Omega^2 + \iota_D(x) \quad \text{where } D = \{z \text{ such that } z_{|\overline{\Omega}} = y_{|\overline{\Omega}}\} \tag{26}$$

This optimization problem may be solved using FB and FISTA. Depending on the definition of the discrete gradient (periodic or not) the proximity operator of $f(x) = \frac{1}{2} \|\nabla s - \nabla x\|_\Omega^2$ may be explicit or not. If it is, all algorithms can be used. If not, a direct DR may need inner loop to compute $prox_{\gamma f}$. But one can observe that if the mask doesn't touch the boundary of the image, the exact definition of the gradient on the boundary of the image doesn't have any impact... So under that condition, we can assume that the gradient is periodic.
Hence $f(x)$ can be written $\tilde{f}(\nabla x)$. Depending on the algorithm and of your choice you can make an explicit descent on f, an implicit (proximal) step on f or on $\tilde{f}$ if you use a Primal Dual algorithm.