The Proximity Operator Repository
User's Guide – Version 0.1

Giovanni Chierchia, Emilie Chouzenoux, Patrick Louis Combettes
and Jean-Christophe Pesquet

Contents

1 Introduction **2**

2 General features **2**
- 2.1 Installation 2
- 2.2 Terms of usage 2

3 Proximity operator **2**

4 List of available functions **3**
- 4.1 Functions of a scalar variable 3
- 4.2 Functions of multivariate variable 4
  - 4.2.1 Vector variable 4
  - 4.2.2 Matrix variable 4
  - 4.2.3 Perspective of convex functions 4
  - 4.2.4 φ-divergences 5
- 4.3 Indicator functions 5
  - 4.3.1 Polyhedra 5
  - 4.3.2 Lower-level sets 6
  - 4.3.3 Epigraphs 6
- 4.4 Nonconvex functions 6
  - 4.4.1 Scalar variable 6
  - 4.4.2 Vector variable 8

5 Tutorial: image recovery with proximal tools **9**
- 5.1 Degradation model 10
- 5.2 Problem formulation 11
- 5.3 Regularization 12
  - 5.3.1 Total variation 12
  - 5.3.2 Structure tensor 14

References **17**

1 Introduction

Proximal splitting methods have become increasingly popular for minimizing a sum of non-necessarily smooth functions. Their implementation is very easy when the proximity operators of the involved functions have a closed-form expression. In this respect, `The Proximity Operator Repository` website provides a comprehensive list of formulas for the proximity operator of convex and non-convex functions, along with the associated codes in Matlab and Python. These programs are provided as an aid for the implementation of proximal algorithms.

2 General features

2.1 Installation

Firstly, download the zipped-file `codes.zip` using the link provided in the home page, and unzip it in your Matlab/Python path. Secondly, search for your function in the `Programs` pages on the menu bar (see Section 4 for an overview of these pages). Then, go to the right column of the table, entitled "Matlab/Python codes", to download the programs that will allow you to evaluate the function (click on "function") and its associated proximity operator (click on "prox").

The provided codes are designed to work with Matlab® 7.0 (or above) or Python 3 along with the numpy scientific library.

2.2 Terms of usage

When your work benefits from the programs available on `The Proximity Operator Repository` website, please consider citing our user's guide.

The programs are provided as an aid for the implementation of proximal algorithms. As it is, they might help you, and it is our goal to provide you with the best possible codes. However, errors are always possible. Please, use our codes at your own risks!

The codes provided are distributed under the license CeCill-B.

3 Proximity operator

In 1962, Jean Jacques Moreau [21] proposed an extension of the projection operator to any lower semi-continuous (lsc) convex function from $\mathbb{R}^N$ to $]-\infty,+\infty]$, whose set is denoted by $\Gamma_0(\mathbb{R}^N)$. For every $f \in \Gamma_0(\mathbb{R}^N)$ and for every $x \in \mathbb{R}^N$, the function $f + \frac{1}{2}\|x - \cdot\|^2$ admits a unique minimizer, which is denoted by $\mathrm{prox}_f(x)$. The mapping $\mathrm{prox}_f : \mathbb{R}^N \mapsto \mathbb{R}^N$ just defined is the *proximity operator* of f.
There exist several extensions of the above definition.

- Let us consider a symmetric positive definite matrix $U \in \mathbb{R}^{N\times N}$. Then, the proximity operator of f within the metric induced by U is defined as the unique solution to:

$$\underset{y\in\mathbb{R}^N}{\text{minimize}}\ f(y) + \frac{1}{2}\|x-y\|_U^2$$

with $\|\cdot\|_2 = \langle .|U.\rangle^{1/2}$.

- The quadratic distance involved in the definition of the proximity operator can actually be modified into any Bregman distance [3]:

$$\underset{y\in\operatorname{int}\operatorname{dom}\psi}{\operatorname{minimize}}\ f(y)+D_\psi(x,y)$$

with $\psi\in\Gamma_0(\mathbb{R}^N)$ of Legendre type, and D_ψ its associated Bregman distance. The unique solution is denoted by $\operatorname{prox}_f^\psi(x)$.

- The notion of proximity operator is not restricted to convex functions. Actually, it can be generalized to any lsc proper function $f\colon\mathbb{R}^N\mapsto]-\infty,+\infty]$ that is not necessarily convex [17], leading to the multi-valued operator:

$$\operatorname{prox}_f\colon x\in\mathbb{R}^N\mapsto\underset{y\in\mathbb{R}^N}{\operatorname{Argmin}}\ f(y)+\frac{1}{2}\|x-y\|^2.$$

The proximity operator enjoys very nice properties [12]. This operator is at the core of an important class of optimization algorithms, known as *proximal algorithms*, which are acknowledged for their great efficiency for solving non-necessarily smooth optimization problems with a high number of variables [18, 4, 5, 6].

4 List of available functions

In order to simplify the search for a function and its associated proximity operator, the `Programs` section of our website is divided into four main categories, which are in turn organized in subsections. They are presented below.

4.1 Functions of a scalar variable

This page presents the proximity operators of some functions $f\in\Gamma_0(\mathbb{R})$. For every $x\in\mathbb{R}$ and for every $\gamma\in]0,+\infty[$, $\operatorname{prox}_{\gamma f}(x)$ is a singleton that belongs to $\mathbb{R}$. We also provide examples of proximity operators within a non euclidian Bregman distance, namely $\operatorname{prox}^\psi_{\gamma f}(x)$ with $\gamma\in]0,+\infty[$ and $\psi\in\Gamma_0(\mathbb{R})$ a Legendre-type function precised in the table.

Example [11]: The proximity operator of the absolute value is

$$(\forall x\in\mathbb{R})\qquad\operatorname{prox}_{\gamma|\cdot|}(x)=\operatorname{sign}(x)\max\{0,|x|-\gamma\}.$$

This can be checked by running the command `p = prox_abs(x, gamma)`.

Example [1]: The proximity operator of $x\mapsto|x-\delta|$, $\delta>0$, within the Bregman distance induced by $\psi:u\mapsto u\log u$ is

$$(\forall x\in]0,+\infty[)\qquad\operatorname{prox}^\psi_{\gamma|\cdot-\delta|}(x)=\begin{cases}\exp(\gamma)x & \text{if}\quad x<\exp(-\gamma)\delta\\ \delta & \text{if}\quad x\in[\exp(-\gamma)\delta,\exp(\gamma)\delta]\\ \exp(-\gamma)x & \text{if}\quad x>\exp(\gamma)\delta\end{cases}$$

This can be checked by running the command `p = prox_breg_abs(x, delta, gamma)`.

4.2 Functions of multivariate variable

This page is organized in the four subsections listed below.

4.2.1 Vector variable

This subpage presents the proximity operators of some functions $f \in \Gamma_0(\mathbb{R}^N)$. For every $x \in \mathbb{R}^N$ and for every $\gamma \in]0, +\infty[$, $\operatorname{prox}_{\gamma f}(x)$ is a singleton that belongs to $\mathbb{R}^N$.

Example [2]: The proximity operator of the Euclidean norm is

$$(\forall x \in \mathbb{R}^N) \qquad \operatorname{prox}_{\gamma\|\cdot\|_2}(x) = \begin{cases} 0 & \text{if} \quad \|x\|_2 \leq \gamma \\ x\left(1 - \dfrac{\gamma}{\|x\|_2}\right) & \text{otherwise.} \end{cases}$$

This can be checked by running the command `p = prox_L2(x, gamma)`.

4.2.2 Matrix variable

This subpage presents the proximity operators of some functions $f \in \Gamma_0(\mathbb{R}^{M\times N})$. For every $X \in \mathbb{R}^{M\times N}$ and for every $\gamma \in]0, +\infty[$, $\operatorname{prox}_{\gamma f}(X)$ is a singleton that belongs to $\mathbb{R}^{M\times N}$.

Example [19]: Consider the nuclear norm, which is defined as

$$(\forall X = U \operatorname{Diag}(s) V^\top \in \mathbb{R}^{M\times N}) \qquad \|X\|_* = \sum_{i=1}^{\min\{M,N\}} s_i.$$

The proximity operator of such a function is

$$(\forall X = U \operatorname{Diag}(s) V^\top \in \mathbb{R}^{M\times N}) \qquad \operatorname{prox}_{\gamma\|\cdot\|_*}(X) = U \operatorname{Diag}(\max\{0, s-\gamma\}) V^\top.$$

This can be checked by running the command `p = prox_nuclear(x, gamma)`.

4.2.3 Perspective of convex functions

This subpage presents the proximity operators of some functions $f_\varphi \in \Gamma_0(\mathbb{R}^N \times \mathbb{R})$ defined as

$$\big(\forall (x,\xi) \in \mathbb{R}^N \times \mathbb{R}\big) \qquad f_\varphi(x,\xi) = \begin{cases} \xi\varphi(x/\xi) & \text{if } \xi > 0 \\ \sigma_{\operatorname{dom}\varphi^*}(x) & \text{if } \xi = 0 \\ +\infty & \text{otherwise} \end{cases}$$

with $\varphi \in \Gamma_0(\mathbb{R}^N)$. For every $(x,\xi) \in \mathbb{R}^N \times \mathbb{R}$ and for every $\gamma \in]0, +\infty[$, $\operatorname{prox}_{\gamma f_\varphi}(x,\xi)$ is a singleton belonging to $\mathbb{R}^N \times \mathbb{R}$.

Example [10]: Consider the perspective of $\varphi = \|\cdot\|_2^2$, which is defined as

$$\big(\forall (x,\xi) \in \mathbb{R}^N \times \mathbb{R}\big) \qquad f_\varphi(x,\xi) = \begin{cases} \|x\|_2^2/\xi & \text{if } \xi > 0 \\ 0 & \text{if } x = 0 \text{ and } \xi = 0 \\ +\infty & \text{otherwise.} \end{cases}$$

The proximity operator of such a function is

$$\big(\forall (x,\xi)\in\mathbb{R}^N\times\mathbb{R}\big)\qquad \operatorname{prox}_{\gamma f_\varphi}(x,\xi)=\begin{cases}(0,0) & \text{if } \|x\|_2^2\le -4\gamma\xi\\ (0,\xi) & \text{if } x=0 \text{ and } \xi>0\\ \Big(x-\dfrac{\gamma t x}{\|x\|_2},\xi+\dfrac{\gamma t^2}{4}\Big) & \text{otherwise}\end{cases}$$

where $t\ge 0$ is the unique solution to $\gamma t^3+4(\xi+2\gamma)t-8\|x\|_2=0$. This can be checked by running the command `[p,t] = prox_perspective_square(x, xi, gamma)`.

4.2.4 φ-divergences

This subpage presents the proximity operators of some functions $f_\varphi\in\Gamma_0(\mathbb{R}\times\mathbb{R})$ defined as

$$\big(\forall (x,\xi)\in\mathbb{R}\times\mathbb{R}\big)\qquad f_\varphi(x,\xi)=\begin{cases}\xi\,\varphi\big(x/\xi\big) & \text{if } x\ge 0 \text{ and } \xi>0\\ x\displaystyle\lim_{\zeta\to+\infty}\varphi(\zeta)/\zeta & \text{if } x>0 \text{ and } \xi=0\\ 0 & \text{if } x=\xi=0\\ +\infty & \text{otherwise,}\end{cases}$$

with $\varphi\colon\mathbb{R}\to[0,+\infty]$ in $\Gamma_0(\mathbb{R})$, twice differentiable on $]0,+\infty[$, and such that $\varphi(1)=\varphi'(1)=0$. For every $(x,\xi)\in\mathbb{R}\times\mathbb{R}$ and $\gamma\in\,]0,+\infty[$, $\operatorname{prox}_{\gamma f_\varphi}(x,\xi)$ is a singleton belonging to $\mathbb{R}\times\mathbb{R}$.

Example [15]: Consider the I_α-divergence, which is defined as

$$\big(\forall (x,\xi)\in\mathbb{R}\times\mathbb{R}\big)\qquad f_\varphi(x,\xi)=\begin{cases}-\sqrt{x\xi} & \text{if } x\ge 0 \text{ and } \xi\ge 0\\ +\infty & \text{otherwise.}\end{cases}$$

The proximity operator of such a function is

$$\big(\forall (x,\xi)\in\mathbb{R}\times\mathbb{R}\big)\quad f_\varphi(x,\xi)=\begin{cases}\Big(x+\dfrac{\gamma}{2}\big(p-1\big),\xi+\dfrac{\gamma}{2}\big(p^{-1}-1\big)\Big) & \text{if } 2x\ge\gamma \text{ or } 1-\dfrac{2\xi}{\gamma}<\dfrac{\gamma}{\gamma-2x}\\ (0,0) & \text{otherwise}\end{cases}$$

where $p>\max\big\{1-2x\gamma^{-1},0\big\}$ is the unique solution to $p^4+\big(2x\gamma^{-1}-1\big)\,p^3+\big(1-2\xi\gamma^{-1}\big)\,p-1=0$. This can be checked by running the command `[p,t] = prox_Ialpha(x, xi, gamma)`.

4.3 Indicator functions

This page presents the proximity operators of some functions $f\in\Gamma_0(\mathbb{R}^N)$ defined as

$$f(x)=\begin{cases}0 & \text{if}\quad x\in C\\ +\infty & \text{otherwise}\end{cases}$$

where $C\subset\mathbb{R}^N$ is a closed convex set. For every $x\in\mathbb{R}^N$ and for every $\gamma\in\,]0,+\infty[$, $\operatorname{prox}_{\gamma f}(x)$ is the orthogonal projection of x onto C, denoted as $\mathrm{P}_C(x)$. The page is organized as follows.

4.3.1 Polyhedra

This subpage presents the projections onto sets delimited by polynomial inequalities.

Example [2]: Assume that $(\eta_1, \eta_2) \in \mathbb{R}^2$ with $\eta_1 \le \eta_2$. Then, the projection onto $C = [\eta_1, \eta_2]^N$ is

$$(\forall x \in \mathbb{R}^N) \qquad \mathrm{P}_C(x) = \big[\max\{\eta_1, \min\{\eta_2, x_n\}\}\big]_{1 \le n \le N}.$$

This can be checked by running the command `p = project_box(x, eta1, eta2)`.

4.3.2 Lower-level sets

This subpage presents the projections onto sets delimited by some functions $\varphi \in \Gamma_0(\mathbb{R}^N)$, namely

$$C = \big\{x \in \mathbb{R}^N \mid \varphi(x) \le \eta\big\}$$

where $\eta \in \operatorname{dom}\varphi$ is such that $C \neq \emptyset$.

Example [2]: The projection onto $C = \big\{x \in \mathbb{R}^N \mid \|x\|_2 \le \eta\big\}$ is

$$(\forall x \in \mathbb{R}^N) \qquad \mathrm{P}_C(x) = \begin{cases} x & \text{if} \quad \|x\|_2 \le \eta \\ x\dfrac{\eta}{\|x\|_2} & \text{otherwise.} \end{cases}$$

This can be checked by running the command `p = project_L2(x, eta)`.

4.3.3 Epigraphs

This subpage presents the projections onto epigraphs of some functions $\varphi \in \Gamma_0(\mathbb{R}^N)$, namely

$$\operatorname{epi}\varphi = \big\{(y, \zeta) \in \mathbb{R}^N \times \mathbb{R} \mid \varphi(y) \le \zeta\big\}.$$

Example [8]: The projection onto the epigraph of $\varphi = \tau\|\cdot\|_2$ is

$$\big(\forall (y, \zeta) \in \mathbb{R}^N \times \mathbb{R}\big) \qquad \mathrm{P}_{\operatorname{epi}\varphi}(y, \zeta) = \begin{cases} (y, \zeta) & \text{if} \quad \tau\|y\|_2 \le \zeta \\ (0, 0) & \text{if} \quad y = 0 \quad \text{and} \quad \zeta < 0 \\ \Big(y, \tau\|y\|_2\Big)\dfrac{\max\{0, \|y\|_2 + \tau\zeta\}}{(1+\tau^2)\|y\|_2} & \text{otherwise} \end{cases}$$

This can be checked by running the command `[p,t] = project_epi_L2(y, zeta)`.

4.4 Nonconvex functions

This page is organized into the two subsections listed below.

4.4.1 Scalar variable

This subpage presents the proximity operators of lsc nonconvex functions f from $\mathbb{R}$ to $]-\infty, +\infty]$. For every $x \in \mathbb{R}$ and for every $\gamma \in]0, +\infty[$, $\operatorname{prox}_{\gamma f}(x)$ is a set of values belonging to $\mathbb{R}$. In most of the considered examples, this set is reduced to a singleton. Otherwise, unless specified in the comment section of corresponding files, our codes provide an element of this set as the output.

Example [20]: Consider the ℓ_0 function, which is defined as

$$(\forall x \in \mathbb{R}) \qquad |x|^0 = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise.} \end{cases}$$

The proximity operator of such a function is

$$(\forall x \in \mathbb{R}) \qquad \operatorname{prox}_{\gamma f}(x) = \begin{cases} 0 & \text{if} \quad |x| < \sqrt{2\gamma} \\ \{x, 0\} & \text{if} \quad |x| = \sqrt{2\gamma} \\ x & \text{otherwise.} \end{cases}$$

This can be checked by running the command `p = prox_zero(x, gamma)`. In the case when $|x| = \sqrt{2\gamma}$, our code provides the input value x as the output of the proximity operator.

Example [7]: Consider the sum of the Shannon entropy and the ℓ_0 function, which is defined as

$$(\forall x \in \mathbb{R}) \qquad f(x) = \begin{cases} x \log x + \omega & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ +\infty & \text{otherwise,} \end{cases}$$

with $\omega \in]0, +\infty[$. The proximity operator of such a function is

$$(\forall x \in \mathbb{R}) \qquad \operatorname{prox}_{\gamma f}(x) = \begin{cases} 0 & \text{if} \quad p^2 + 2\gamma p < 2\omega\gamma \\ \{p, 0\} & \text{if} \quad p^2 + 2\gamma p = 2\omega\gamma \\ p & \text{otherwise,} \end{cases}$$

with

$$p = \gamma W\left(\frac{1}{\gamma}\exp\left(\frac{x}{\gamma} - 1\right)\right),$$

where $W(\cdot)$ denotes the W-Lambert function [14]. This can be checked by running the command `p = prox_entropy_zero(x, gamma, w)`. In the case $p^2 + 2\gamma p = 2\omega\gamma$, our code provides the value p as the output of the proximity operator.

4.4.2 Vector variable

This page presents the proximity operators of lsc nonconvex functions f from $\mathbb{R}^N$ to $]-\infty,+\infty]$. For every $x \in \mathbb{R}^N$ and for every $\gamma > 0$, $\mathrm{prox}_{\gamma f}(x)$ is a set of vectors belonging to $\mathbb{R}^N$. In most considered examples, this set is reduced to a singleton. Otherwise, unless specified in the comment section of corresponding files, our codes provide an element of this set as the output.

Example [22]: Consider the truncated quadratic form, which is defined as

$$(\forall x \in \mathbb{R}^N) \qquad f(x) = \min\{\|x\|_2^2, \omega\}$$

with $\omega > 0$. The proximity operator of such a function is

$$(\forall x \in \mathbb{R}^N) \qquad \mathrm{prox}_{\gamma f}(x) = \begin{cases} \dfrac{x}{1+2\gamma} & \text{if } \|x\|_2^2 < \omega(1+2\gamma) \\ \left\{x, \dfrac{x}{1+2\gamma}\right\} & \text{if } \|x\|_2^2 = \omega(1+2\gamma) \\ x & \text{otherwise} \end{cases}$$

This can be checked by running the command `p = prox_truncated_norm(x, gamma)`. In the case when $\|x\|_2^2 = \omega(1+2\gamma)$, our code provides x as the output of the proximity operator.

Example [16]: Consider the sum of the ℓ_0 function and the indicator of a conic domain:

$$(\forall (x,d) \in \mathbb{C}^2) \qquad f(x) = |x|^0 + \iota_S(x,d)$$

where

$$S = \{x \in \mathbb{C}, d \in \mathbb{C} | \quad \exists \delta \in [-\Delta, \Delta] \quad \text{such that} \quad d = \delta x\}$$

with $\Delta \in [0,+\infty)$. The proximity operator of such a function is

$$\mathrm{prox}_{\gamma f}(x,d) = \begin{cases} (0,0) & \text{if } |x|^2 + |d|^2 < \frac{|\widehat{\delta} x - d|^2}{1+\widehat{\delta}^2} + 2\gamma\lambda \\ \dfrac{x+\widehat{\delta} d}{1+\widehat{\delta}^2}(1,\widehat{\delta}) & \text{otherwise,} \end{cases}$$

where

$$\widehat{\delta} = \begin{cases} \min\left\{\frac{\eta + |d|^2 - |x|^2}{2|\mathrm{Re}(xd^*)|}, \Delta\right\} \mathrm{sign}\big(\mathrm{Re}(xd^*)\big) & \text{if } \mathrm{Re}(xd^*) \neq 0 \\ 0 & \text{if } \mathrm{Re}(xd^*) = 0 \\ & \text{ and } |x| \geq |d| \\ \Delta & \text{otherwise,} \end{cases}$$

and $\eta = \sqrt{\big(|d|^2 - |x|^2\big)^2 + 4\big(\mathrm{Re}(xd^*)\big)^2}$.

5 Tutorial: image recovery with proximal tools

The `tutorial` section describes how to tackle optimization problems with the *forward-backward primal-dual algorithm* (FBPD) [23, 13, 6], which is able to

$$\underset{x\in\mathbb{R}^N}{\text{minimize}} \;\; f(x) + g(x) + h(Fx)$$

where $f \in \Gamma_0(\mathbb{R}^N)$, $g \in \Gamma_0(\mathbb{R}^N)$ with β-Lipschitz continuous gradient for some $\beta > 0$, $F \in \mathbb{R}^{M\times N}$, and $h \in \Gamma_0(\mathbb{R}^M)$. In this regard, it can be shown that the sequence $(x^{[i]})_{i\in\mathbb{N}}$ generated by

$$\left\lfloor\begin{array}{l} x^{[i+1]} = \operatorname{prox}_{\tau f}\Big(x^{[i]} - \tau\big(\nabla g(x^{[i]}) + F^\top y^{[i]}\big)\Big) \\[2ex] y^{[i+1]} = \operatorname{prox}_{\sigma h^*}\Big(y^{[i]} + \sigma F\big(2x^{[i+1]} - x^{[i]}\big)\Big) \end{array}\right.$$

converges to a solution to the above problem, provided that

- $(x^{[0]}, y^{[0]}) \in \mathbb{R}^N \times \mathbb{R}^M$
- $\tau > 0$ and $\sigma > 0$ are such that $\tau\left(\frac{\beta}{2} + \sigma\|F^\top F\|\right) < 1$.

A generic implementation of FBPD is given below (the input parameters will be clarified later).

MATLAB CODE

```
function [x, it, time, crit] = FBPD(x, f, g, h, opt)

if nargin < 5, opt.tol = 1e-4; opt.iter = 500; end

% step-sizes
tau = 2 / (g.beta+2);
sigma = (1/tau - g.beta/2) / h.beta;

% initialization
y = h.dir_op(x);

% algorithm loop
time = zeros(1, opt.iter);
crit = zeros(1, opt.iter);
for it = 1:opt.iter

    tic;

    % primal forward-backward step
    x_old = x;
    x = x - tau * ( g.grad(x) + h.adj_op(y) );
    x = f.prox(x, tau);

    % dual forward-backward step
    y = y + sigma * h.dir_op(2*x - x_old);
    y = y - sigma * h.prox(y/sigma, 1/sigma);

    % time and criterion
    time(it) = toc;
    crit(it) = f.fun(x) + g.fun(x) + h.fun(h.dir_op(x));

    % stopping rule
    if norm( x(:) - x_old(:) ) < opt.tol * norm( x_old(:) ) && it > 10
        break;
    end
end
crit = crit(1:it);
time = cumsum(time(1:it));
```

Python code

```python
def FBPD(x_init, f, g, h, tol=None, iter=None):

    if tol  is None: tol  = 1e-4
    if iter is None: iter = 500

    # step-sizes
    tau   = 2 / (g.beta + 2);
    sigma = (1/tau - g.beta/2) / h.beta;

    # initialization
    x = x_init
    y = h.dir_op(x);

    # algorithm loop
    timing = np.zeros(max_iter)
    criter = np.zeros(max_iter)
    for it in range(0, max_iter):

        t = time.time()

        # primal forward-backward step
        x_old = x;
        x = x - tau * ( g.grad(x) + h.adj_op(y) );
        x = f.prox(x, tau);

        # dual forward-backward step
        y = y + sigma * h.dir_op(2*x - x_old);
        y = y - sigma * h.prox(y/sigma, 1/sigma);

        # time and criterion
        timing[it] = time.time() - t
        criter[it] = f.fun(x) + g.fun(x) + h.fun(h.dir_op(x));

        # stopping rule
        if np.linalg.norm(x - x_old) < tol * np.linalg.norm(x_old) and it > 10: break

    criter = criter[0:it+1];
    timing = np.cumsum(timing[0:it+1]);
    return x, it, timing, criter
```

5.1 Degradation model

The goal of *image recovery* is to restore the visual content of a corrupted image through the inversion of the corresponding degradation process. In this context, a popular task consists of recovering an image $\overline{x} \in \mathbb{R}^N$ as close as possible to some observations $z \in \mathbb{R}^K$ generated as

$$z = A\overline{x} + b,$$

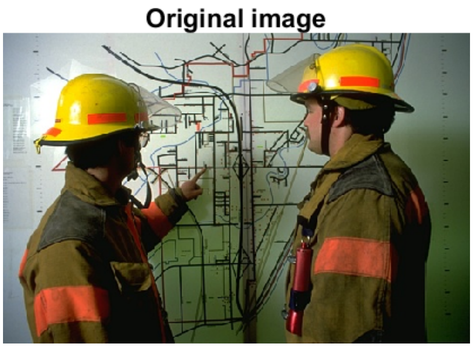
where $A \in \mathbb{R}^{K \times N}$ is a known matrix, and $b \in \mathbb{R}^K$ is a realization of zero-mean white Gaussian noise. For the purpose of this tutorial, the degraded image is generated as follows (with $K = N$).

Matlab code

```matlab
% original image
x_bar = double( imread('firemen.jpg') );

% blur operator
psf = fspecial('average', 3);

% noisy image
rng('default');
z = imfilter(x_bar, psf) + 20 * randn( size(x_bar) );

% visualization
figure; imshow(x_bar/255,[]); title('Original image');
figure; imshow(    z/255,[]); title('Noisy image');
```

Python code

```
# original image
x_bar = misc.imread('firemen.jpg');
x_bar = x_bar.astype(np.float64)

# blur operator
psf = (3, 3, 1)

# noisy image
z = fil.uniform_filter(x_bar, psf) + 20 * np.random.randn(*x_bar.shape);

# visualization
plt.imshow(x_bar/255)
plt.title('Original image')
plt.figure()
plt.imshow(np.clip(z/255,0,1))
plt.title('Noisy image')
```



5.2 Problem formulation

To recover $\overline{x}$ from z, one can follow a variational approach that aims at

$$\underset{x\in[0,255]^N}{\text{minimize}} \quad \underbrace{\frac{1}{2}\|Ax-z\|_2^2}_{\text{Data fidelity}} \quad + \quad \underbrace{h(Fx).}_{\text{Regularization}}$$

This formulation reverts to the general convex optimization problem by setting $f(x) = \iota_{[0,255]^N}(x)$ and $g(x) = \frac{1}{2}\|Ax-z\|_2^2$. As for the data fidelity term, FBPD needs $\text{prox}_{\tau f}(x) = P_{[0,255]^N}(x)$, $\nabla g(x) = A^\top(Ax-z)$, and $\beta = \|A\|^2$. This is illustrated in the code below.

Matlab code

```
% constraint
f.prox = @(x,tau) project_box(x, 0, 255);

% data fidelity
A_dir = @(x) imfilter(x, psf);
A_adj = @(x) imfilter(x, rot90(psf,2));  % WARNING: 'psf' must be a (2n+1)-by-(2n+1) matrix
g.grad = @(x) A_adj(A_dir(x) - z);
g.beta = sum(abs(psf(:)));

% criteria
f.fun = @(x) indicator_box(x, 0, 255);
g.fun = @(x) sum(sum(sum((A_dir(x)-z).^2)));
```

```
class LeastSquares:
   z      = None
   psf    = None
   beta   = None

   def __init__(self, z, psf):
      self.z    = z
      self.psf  = psf
      self.beta = np.prod(psf)

   def A_dir(self, x):
      return fil.uniform_filter(x, self.psf)

   def A_adj(self, x):
      return fil.uniform_filter(x, self.psf);     # WARNING: filter dimensions must be odd

   def grad(self, x):
      return self.A_adj( self.A_dir(x) - self.z )

   def fun(self, x):
      p = self.A_dir(x)
      return np.sum(np.square(p-z))

# constraint
f = BoxConstraint(0, 255)

# data fidelity
g = LeastSquares(z, psf)
```

5.3 Regularization

Various forms of regularization arise with specific choices of F and h, such as *total variation* and *structure tensor* (e.g., see [9]). They are discussed in the following.

5.3.1 Total variation

The *total variation* (TV) is defined as:

$$\mathrm{TV}(x) = \lambda \sum_{\ell=1}^{N} \Big(\big|x^{(\ell)} - x^{(n_{\ell,1})}\big|^2 + \big|x^{(\ell)} - x^{(n_{\ell,2})}\big|^2 \Big)^{1/2}$$

where $\lambda > 0$ and $(n_{\ell,1}, n_{\ell,2}) \in \{1, \dots, N\}^2$ denote the positions of the horizontal/vertical nearest neighbors of $x^{(\ell)}$. This penalty can be plugged into our problem formulation by setting:

$$y = Fx = \begin{bmatrix} x^{(1)} - x^{(n_{1,1})} \\ x^{(1)} - x^{(n_{1,2})} \\ \vdots \\ x^{(N)} - x^{(n_{N,1})} \\ x^{(N)} - x^{(n_{N,2})} \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} a \\ a \end{matrix}}\right\} y_1 \in \mathbb{R}^2 \\ \vdots \\ \left.\vphantom{\begin{matrix} a \\ a \end{matrix}}\right\} y_N \in \mathbb{R}^2 \end{matrix} \qquad \text{and} \qquad h(y) = \sum_{\ell=1}^{N} \lambda \|y_\ell\|_2.$$

Consequently, FBPD needs the following information: F, $F^\top$, $\|F^\top F\| = 8$, and

$$\mathrm{prox}_{\gamma h}(y) = \Big(\mathrm{prox}_{\gamma \lambda \|\cdot\|_2}(y_\ell) \Big)_{1 \le \ell \le N}.$$

Let us start with the implementation of F and its adjoint $F^\top$.

Matlab code

```matlab
% forward finite differences (with Neumann boundary conditions)
hor_forw = @(x) [x(:,2:end,:)-x(:,1:end-1,:), zeros(size(x,1),1,size(x,3))]; % horizontal
ver_forw = @(x) [x(2:end,:,:)-x(1:end-1,:,:); zeros(1,size(x,2),size(x,3))]; % vertical

% backward finite differences (with Neumann boundary conditions)
hor_back = @(x) [-x(:,1,:), x(:,1:end-2,:)-x(:,2:end-1,:), x(:,end-1,:)];    % horizontal
ver_back = @(x) [-x(1,:,:); x(1:end-2,:,:)-x(2:end-1,:,:); x(end-1,:,:)];    % vertical
```

Python code

```python
def hor_forward(x):
   """ Horizontal forward finite differences (with Neumann boundary conditions) """
   hor = np.zeros_like(x)
   hor[:,:-1,:] = x[:,1:,:] - x[:,:-1,:]
   return hor

def ver_forward(x):
   """ Vertical forward finite differences (with Neumann boundary conditions) """
   ver = np.zeros_like(x)
   ver[:-1,:,:] = x[1:,:,:] - x[:-1,:,:]
   return ver

def hor_backward(x):
   """ Horizontal backward finite differences (with Neumann boundary conditions) """
   Nr, Nc, Nb = x.shape
   zer = np.zeros((Nr,1,Nb))
   xxx = x[:,:-1,:]
   return np.concatenate((zer,xxx), 1) - np.concatenate((xxx,zer), 1)

def ver_backward(x):
   """ Vertical backward finite differences (with Neumann boundary conditions) """
   Nr, Nc, Nb = x.shape
   zer = np.zeros((1,Nc,Nb))
   xxx = x[:-1,:,:]
   return np.concatenate((zer,xxx), 0) - np.concatenate((xxx,zer), 0)
```

Let us continue with the implementation of prox_h. To do so, note that the method `h.dir_op()` generates a 4D array in which the *gradient vector* of each pixel is stored along the 4th dimension.

Matlab code

```matlab
% regularization parameter
lambda = 5;

% proximity operator and criterion
h.prox = @(y,gamma) prox_L2(y, lambda*gamma, 4);
h.fun  = @(y)       fun_L2(y, lambda,       4);

% direct and adjoint operators
h.dir_op = @(x) cat( 4, hor_forw(x), ver_forw(x) );
h.adj_op = @(y) hor_back( y(:,:,:,1) ) + ver_back( y(:,:,:,2) );

% operator norm
h.beta = 8;
```

Python code

```python
class TotalVariation(L2_Norm):
   beta = 8

   def __init__(self, gamma):
      L2_Norm.__init__(self, gamma, 3)

   def dir_op(self, x):
      return np.stack( (hor_forward(x),ver_forward(x)), 3)

   def adj_op(self, y):
      return hor_backward( y[:,:,:,0] ) + ver_backward( y[:,:,:,1] )
```

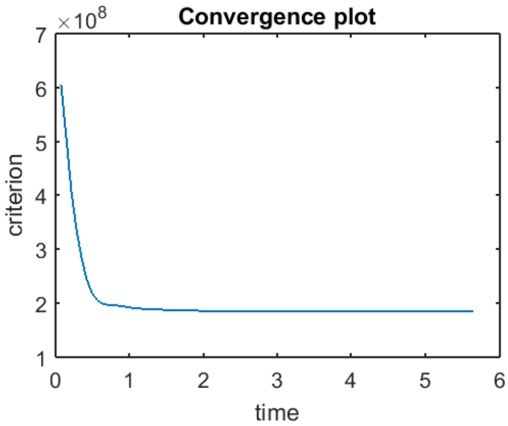
The inputs needed by FBPD are all set, and the optimization method can be executed.

Matlab code

```
% minimization
[x, it, time, crit] = FBPD(z, f, g, h);

psnr = 10 * log10( 255^2 / mean((x(:)-x_bar(:)).^2) );

% visualization
figure; imshow(x/255,[]); title(['Restored image - PSNR: ' num2str(round(psnr,2))])
figure; plot(time, crit); title('Convergence plot'); xlabel('seconds'); ylabel('criterion')
```

Python code

```
# minimization
x, it, time, crit = FBPD(z, f, g, h);

psnr = 10 * np.log10( 255*255 / np.mean(np.square(x-x_bar)) )

# visualization
plt.imshow(x/255); plt.title( 'Restored image - PSNR: ' + str(np.round(psnr,2)) )
plt.figure()
plt.plot(time, crit); plt.title('Convergence plot')
```



5.3.2 Structure tensor

In the previous example, the regularization is applied separately to each color channel of the image to be restored. The *structure tensor* (ST) is a natural extension of TV that allows one to process the channels jointly. Assume that the sought image x is composed by three channels (red, blue, and greed):

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

with $x_j \in \mathbb{R}^Q$ for every $j \in \{1,2,3\}$, and $N = 3Q$. The ST regularization is defined as [9]:

$$\mathrm{ST}(x) = \lambda \sum_{\ell=1}^{Q} \left\| \begin{bmatrix} x_1^{(\ell)} - x_1^{(n_{\ell,1})} & x_2^{(\ell)} - x_2^{(n_{\ell,1})} & x_3^{(\ell)} - x_3^{(n_{\ell,1})} \\ x_1^{(\ell)} - x_1^{(n_{\ell,2})} & x_2^{(\ell)} - x_2^{(n_{\ell,2})} & x_3^{(\ell)} - x_3^{(n_{\ell,2})} \end{bmatrix} \right\|_*$$

where $\lambda > 0$ and $\|\cdot\|_*$ denotes the nuclear norm.

This penalty can be plugged into our problem formulation by setting:

$$Y = Fx = \begin{array}{l} \left[\begin{matrix} x_1^{(1)} - x_1^{(n_{1,1})} & x_2^{(1)} - x_2^{(n_{1,1})} & x_3^{(1)} - x_3^{(n_{1,1})} \\ x_1^{(1)} - x_1^{(n_{1,2})} & x_2^{(1)} - x_2^{(n_{1,2})} & x_3^{(1)} - x_3^{(n_{1,2})} \\ \vdots & \vdots & \vdots \\ x_1^{(Q)} - x_1^{(n_{Q,1})} & x_2^{(Q)} - x_2^{(n_{Q,1})} & x_3^{(Q)} - x_3^{(n_{Q,1})} \\ x_1^{(Q)} - x_1^{(n_{Q,2})} & x_2^{(Q)} - x_2^{(n_{Q,2})} & x_3^{(Q)} - x_3^{(n_{Q,2})} \end{matrix}\right] \end{array} \begin{array}{l} \left.\vphantom{\begin{matrix} a \\ a \end{matrix}}\right\} Y_1 \in \mathbb{R}^{2\times 3} \\ \quad\vdots \\ \left.\vphantom{\begin{matrix} a \\ a \end{matrix}}\right\} Y_Q \in \mathbb{R}^{2\times 3} \end{array}$$

and

$$h(Y) = \sum_{\ell=1}^{Q} \lambda \|Y_\ell\|_*.$$

Consequently, FBPD needs the following information: F, $F^\top$, $\|F^\top F\| = 8$, and $\operatorname{prox}_{\gamma h}(Y) = \Big(\operatorname{prox}_{\gamma \lambda \|\cdot\|_*}(Y_\ell)\Big)_{1\leq\ell\leq Q}$. Let's start with the implementation of F and its adjoint $F^\top$.

Matlab code

```
% utility functions
  pack = @(x)   reshape( shiftdim(x,2), [1 size(x,3) size(x,1) size(x,2)] );
unpack = @(y,i) shiftdim( reshape( y(i,:,:,:), [size(y,2) size(y,3) size(y,4)] ), 1 );

% direct and adjoint operators
h.dir_op = @(x) [pack( hor_forw(x) ); pack( ver_forw(x) )];
h.adj_op = @(y) hor_back( unpack(y,1) ) + ver_back( unpack(y,2) );

% operator norm
h.beta = 8;
```

Python code

```
# TODO
```

Let us continue with the implementation of prox_h. Note that `h.dir_op` generates a 4D matrix in which the *gradient matrix* of each position is stored on the 1st and 2nd dimensions.

Matlab code

```
% regularization parameter
lambda = 7;

% proximity operator
h.prox = @(y,gamma) prox_nuclear(y, gamma*lambda);

% criterion
h.fun = @(y) fun_nuclear(y, lambda);
```

Python code

```
# TODO
```

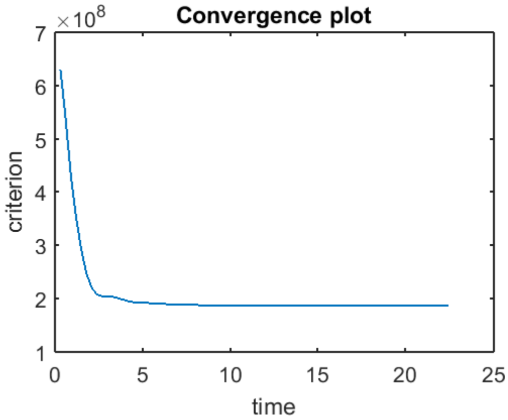
The inputs needed by FBPD are all set, and the optimization method can be executed.

Matlab code

```
% minimization
[x, it, time, crit] = FBPD(z, f, g, h);

% PSRN
psnr = 10 * log10( 255^2 / mean((x(:)-x_bar(:)).^2) );

% visualization
figure; imshow(x/255,[]); title(['Restored image - PSNR: ' num2str(round(psnr,2))])
figure; plot(time, crit); title('Convergence plot'); xlabel('seconds'); ylabel('criterion')
```

Python code

```
# TODO
```



References

[1] H. H. Bauschke, J. Bolte, and M. Teboulle. A descent lemma beyond lipschitz gradient continuity: First-order methods revisited and applications. *Mathematics of Operations Research*, (to appear), Nov. 2016.

[2] H. H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, January 2011.

[3] H.H. Bauschke, P. L. Combettes, and D. Noll. Joint minimization with alternating Bregman proximity operators. *Pacific Journal of Optimization*, 2(3):401–424, 2006.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–222, 2011.

[5] L. M. Briceños Arias and P. L. Combettes. A monotone + skew splitting model for composite monotone inclusions in duality. *SIAM J. Optim.*, 21(4):1230–1250, Oct. 2011.

[6] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2011.

[7] A. Cherni, E. Chouzenoux, and M.-A. Delsuc. Proximity operators for a class of hybrid sparsity+entropy priors. application to DOSY NMR signal reconstruction. In *Proceedings of the 8th International Symposium on Signal, Image, Video and Communications (ISIVC 2016)*, Tunis, Tunisia, 21–23 Nov. 2016.

[8] G. Chierchia, N. Pustelnik, J. C. Pesquet, and B. Pesquet-Popescu. Epigraphical projection and proximal tools for solving constrained convex optimization problems. *Signal, Image and Video Processing*, 9(8):1737–1749, November 2015.

[9] G. Chierchia, N. Pustelnik, B. Pesquet-Popescu, and J.-C. Pesquet. A nonlocal structure tensor based approach for multicomponent image recovery problems. *IEEE Transactions on Image Processing*, 23(12):5531–5544, December 2014.

[10] P. L. Combettes and C. L. Müller. Perspective functions: Proximal calculus and applications in high-dimensional statistics. *Journal of Mathematical Analysis and Applications*, 2017. http://http://arxiv.org/abs/1610.01478.

[11] P. L. Combettes and J.-C. Pesquet. Proximal thresholding algorithm for minimization over orthonormal bases. *SIAM Journal on Optimization*, 18(4):1351–1376, 2008.

[12] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In H. H. Bauschke, R. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, editors, *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer-Verlag, New York, 2010.

[13] L. Condat. A primal-dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms. *Journal of Optimization, Theory and Applications*, 158(2):460–479, Aug. 2013.

[14] R. M. Corless, G. H. Gonnet, and D. J. Hare, D. E. G. ad Jeffrey. On the lambert W function. *Advances in Computational mathematics*, 5(1):329–359, 1996.

[15] M. El Gheche, G. Chierchia, and J. C. Pesquet. Proximity operators of discrete information divergences. *Technical report*, 2016. http://arxiv.org/abs/1606.09552.

[16] A. Florescu, E. Chouzenoux, J.-C. Pesquet, and S. Ciochina. A constrained optimization approach for complex sparse perturbed models. In *Proceedings of the Signal Processing with Adaptive Sparse Structured Representations (SPARS 2013)*, Lausanne, Switzerland, 8-11 July 2013.

[17] J. B. Hiriart-Urruty and C. Lemaréchal. Convex analysis and minimization algorithms ii: Advanced theory and bundle methods. *Grundlehren der mathematischen Wissenschaften, Springer-Verlag, New York*, 306, 1993.

[18] N. Komodakis and J.-C. Pesquet. Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. *IEEE Signal Processing Magazine*, 32(6):31–54, Nov. 2014.

[19] A. S. Lewis. Derivatives of spectral functions. *Mathematics of Operations Research*, 21(3):576–588, 1996.

[20] T. Möllenhoff, E. Strekalovskiy, M. Möller, and D. Cremers. Low rank priors for color image regularization. In *Proceedings of EMMCVPR*, Hong Kong, January 2015.

[21] J.-J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

[22] E. Strekalovskiy and D. Cremers. Real-time minimization of the piecewise smooth mumford-shah functional. In *Proceedings of ECCV*, pages 127–141, Zürich, Switzerland, September 2014.

[23] B. C. Vũ. A splitting algorithm for dual monotone inclusions involving cocoercive operators. *Advances in Computational Mathematics*, 38(3):667–681, Apr. 2013.