A Predual Proximal Point Algorithm solving a Non Negative Basis Pursuit Denoising model

F. Malgouyres*and T. Zeng†

October 10, 2008‡

Abstract

This paper develops an implementation of a Predual Proximal Point Algorithm (PPPA) solving a Non Negative Basis Pursuit Denoising model. The model imposes a constraint on the l^2 norm of the residual, instead of penalizing it. The PPPA solves the predual of the problem with a Proximal Point Algorithm (PPA). Moreover, the minimization that needs to be performed at each iteration of PPA is solved with a dual method. We can prove that these dual variables converge to a solution of the initial problem.

Our analysis proves that we turn a constrained non differentiable convex problem into a short sequence of nice concave maximization problems. By nice, we mean that the functions which are maximized are differentiable and their gradient is Lipschitz.

The algorithm is easy to implement, easier to tune and more general than the algorithms found in the literature. In particular, it can be applied to the Basis Pursuit Denoising (BPDN) and the Non Negative Basis Pursuit Denoising (NNBPDN) and it does not make any assumption on the dictionary. We prove its convergence to the set of solutions of the model and provide some convergence rates.

Experiments on image approximation show that the performances of the PPPA are at the current state of the art for the BPDN.

AMS Classification : 41A45, 65D15, 49N30, 49N15, 65K05

Keywords : Basis Pursuit, algorithm, sparse representation, proximal point algorithm, ell1 minimization

*LAGA, CNRS UMR 7539, Université Paris 13, 99, avenue Jean-Baptiste Clement, 93430 Villetaneuse, FRANCE malgouy@math.univ-paris13.fr

†Dept. of Mathematics and Institute for Computational Mathematics Hong Kong Baptist University, Kowloon Tong, Hong Kong. zeng@hkbu.edu.hk

‡First release Feb. 23, 2007

1 Introduction

1.1 Recollection on Basis Pursuit Denoising

The use of the Basis Pursuit Denoising model (BPDN) [7] in image/signal processing is now a fairly developed field of research. For instance, it is commonly used for compression, source separation [37], feature selection for classification [5], and restoration [4]. Many theoretical results have also been established supporting this model. Most of them aim at understanding the equivalence between the usual BPDN (see below) and the search for the sparsest approximation (see, among others, [14, 15]). An extensive body of work has also been carried out under the name Compressed Sensing [33, 13, 6]. Other authors have shown that the BPDN is an efficient way to simplify a complex data distribution (see [30, 29]). The usual BPDN takes the form

$$\min_{x\in\mathbb{R}^P} \|Ax-b\|_2^2 + \lambda\|x\|_1, \tag{1}$$

where A is an N−by−P matrix (i.e. N rows and P columns), $b \in \mathbb{R}^N$ is the datum, $\lambda > 0$ is real, $\|.\|_2$ stands for the Euclidean norm on $\mathbb{R}^N$ and $\|.\|_1$ stands for the l^1 norm on $\mathbb{R}^P$.

We will always suppose that $P \geq N$ (and in practice we often have $P \gg N$). The columns of A are denoted $(A^i)_{i=1..P}$, they form a dictionary of atoms which are used to represent the datum b. The purpose of the BPDN is to express an approximation of b as a sparse linear expansion in this dictionary.

A nice geometric interpretation of the model (1) arises when we write it under the form

$$\min_{c\in\mathbb{R}^N} \|c-b\|_2^2 + \lambda E(c), \tag{2}$$

where E is defined, for every $c \in \mathbb{R}^N$, by

$$\begin{cases} E(c) \overset{\text{def}}{=} \min_{x\in\mathbb{R}^P} \|x\|_1 \\ \text{under the constraints } Ax = c. \end{cases}$$

Indeed, the strength of the functional E (the regularization term in (2)) is that its level sets are scaled versions of the convex hull of $(A^i)_{i=1..P} \cup (-A^i)_{i=1..P}$ (see [12, 30]). It is therefore possible to build a functional E that favor the use of specific structures; and these structures are explicitly chosen by the user. This functional can therefore be designed to favor the structures which are known to be important in a given application.

One drawback of the above functional E is that it favors both the apparition of A^i and $-A^i$. This might lead to the bad modeling of some structures which are only used with a given sign. For instance when dealing with images of text, the letters are always dark on a brighter background. If, in an expansion, an element of the dictionary representing a letter at a given location appears with a negative sign, it describes a content which is not a letter. This content should be represented by elements of the dictionary devoted to the background. (This holds also for astronomical image, images of faces...). This led some authors [15, 30, 26] to study the Non Negative Basis Pursuit Denoising (NNBPDN), where the above regularization term E is replaced by E_{nn} defined, for every $c \in \mathbb{R}^N$, by

$$\begin{cases} E_{nn}(c) = \min_x \quad x^t \mathbf{1}_P \\ \text{under the constraints } x_i \geq 0, \forall i = 1..P, \\ \text{and } Ax = c. \end{cases}$$

where $.^t$ stands for the transpose and $\mathbf{1}_P$ stands for a vector of size P with all its entries equal to 1.

The level sets of E_{nn} are scaled versions of the convex hull of $(A^i)_{i=1..P}$. Of course, if for all $i = 1..P$, $-A^i \in (A^i)_{i=1..P}$ one obtains a functional similar to E.

Another issue which we wanted to improve in (1) concerns the choice of the parameter λ. For practical applications, it is always preferable to solve the model in the form

$$\begin{cases} \min_{c \in \mathbb{R}^N} E_{nn}(c), \\ \text{under the constraints } \|c - b\|_2 \leq \tau, \end{cases} \tag{3}$$

for a parameter $\tau > 0$. Indeed, τ can be tuned automatically, according to some prescribed precision (in approximation) or a known noise level (in denoising, compressed sensing).

Notice that, as is well known, there is a correspondence between the parameter λ in (1) and the parameter τ in (3). However, this correspondence depends on the initial datum b.

These considerations led us to consider a NNBPDN model taking the form

$$(D) \begin{cases} \min_x x^t \mathbf{1}_P \\ \text{under the constraints } x_i \geq 0, \forall i = 1..P, \\ \text{and } \|Ax - b\|_2 \leq \tau, \end{cases}$$

for $\tau > 0$.

The purpose of the paper is to design an efficient algorithm (the PPPA) solving (D).

1.2 Existing algorithms solving the BPDN

As mentioned in the introduction, the literature on the resolution of the BPDN is rapidly growing. No algorithm is currently available for solving (D). Almost all the papers deal with the model under its form:

$$\min_{x \in \mathbb{R}^P} f(x), \tag{4}$$

with

$$f(x) \overset{\text{def}}{=} \|Ax - b\|_2^2 + \lambda \|x\|_1,$$

for $\lambda > 0$ and $x \in \mathbb{R}^P$.

1.2.1 Iterative Thresholding (IT)

The IT has been studied in [11, 1, 21, 20, 10, 9, 25].

This algorithm is proved to converge (with a linear convergence rate) as soon as the operator norm of A is strictly smaller than 1. This means

$$M \overset{\text{def}}{=} \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} < 1.$$

In Section 3.3.2, we present a trick (which has independently been noticed in [25]) to apply the IT to any matrix A.

1.2.2 Parallel Coordinate Descent (PCD) Algorithm

The PCD algorithm was proposed in [18] and extended in [19]. It is proved to converge under conditions which are not satisfied by the l^1 norm. We have observed however that it behaves well in this framework.

Notice that, in this algorithm, at the k^{th} iteration, a descent direction d^k is computed. Then an optimal step is computed along that direction. This requires the evaluation of $f\left(x^k + td^k\right)$ at different time steps t. The evaluation of $\|A(x^k+td^k)-b\|_2^2$ is rapid, since it is a second order polynomial in t. However, each evaluation of $\|x^k + td^k\|_1$ costs one multiplication and two additions per element of the dictionary. Depending on the matrix A this might represent a non negligible proportion of the total computational time.

1.2.3 Gradient Projection for Sparse Reconstruction (GPSR)

This algorithm is described in [23]. Like our algorithm, it benefits from the fact that it is possible to rewrite (4) in the form

$$\begin{cases} \min_{x^+, x^- \in \mathbb{R}^P} f(x^+ - x^-) \\ \text{under the constraints } x^+ \geq 0 \text{ and } x^- \geq 0. \end{cases}$$

This is a simpler form of the problem than (4) since the orthogonal projection onto the constraints is easy. Moreover, this permits to get rid of the non-differentiable l^1 norm. The GPSR also exploits the fact that $f(x^+ - x^-)$ is quadratic in x^+ and x^-.

1.2.4 Other methods

In [7], the authors propose an interior point method. (A better description is given in [36].) To our knowledge, the last development, in the direction of interior point methods is [26] (the algorithm is called *l1_ls*). It is reported, in [23], to be slower than GPSR.

The Block Coordinates Relaxation (BCR) algorithm introduced in [36] only applies when the dictionary of atoms corresponding to the columns of A is a union of orthonormal bases (its extension to a union of orthogonal bases is straightforward). Indeed, it uses the fact that the soft-thresholding operator provides an exact resolution of (4) when the dictionary is an orthonormal basis.

The homotopy algorithm (it computes an exact solution of (4) for a decreasing λ) proposed in [31, 16] is very elegant and has the advantage of being exact. Similar algorithms are used in the machine learning community (for computing SVMs). The history of this algorithm apparently goes back to the 50's in economics. The homotopy algorithm does require, at each iteration, the inversion of a matrix. At the end of the iterative process, the size of this matrix equals the number of non-zero coordinates of the result. This restricts its use to applications where this number remains very small.

At least one algorithm (see [17]) exists for solving the related problem (named LASSO)

$$\left\{ \begin{array}{l} \min_{x\in\mathbb{R}^P} \|Ax-b\|_2^2 \\ \text{under the constraint } \|x\|_1 \leq \tau', \end{array} \right. \tag{5}$$

for $\tau' > 0$. This "parameterization of the problem" is indeed relevant for the machine learning community.

Finally, there are many algorithms (in particular the Matching Pursuit family) which promote sparsity. They have the same goal and are in competition with the BPDN but do not solve the BPDN.

1.2.5 The continuation trick

This trick permits to deal better with the difficulty of the problem (4) when λ is small (or very small). It has been observed by several authors (see, among others, [23, 25]) that it is faster, in this case

- to compute a solution of the model for $\lambda' \geq \lambda$ and initialize the algorithm with this solution.
- than to compute directly a solution for λ, with a crude initialization.

Using this idea recursively, we can consider a decreasing set of values $(\lambda'_l)_{1\leq l\leq L}$ such that $\lambda'_L = \lambda$ and compute the solutions for all the values $(\lambda'_l)_{1\leq l\leq L}$.

This trick is called homotopy (because of its similarity with the homotopy algorithm) or continuation. It can be adapted to any algorithms solving (D), (4) and (5).

The drawback of this continuation trick is that it is difficult to construct a good sequence $(\lambda'_l)_{1\leq l\leq L}$ and to determine the correct level of convergence required when $l < L$. It is particularly difficult to determine these parameters automatically. However, when they are properly tuned (for a given problem), this strategy diminishes the computational time requirement significantly.

1.3 Sketch of the PPPA

The general description of the PPPA is as follows:

- Consider (P) the predual of (D).

- We solve the predual (P) with a PPA. (Thus the name of the algorithm.) Doing so, we obtain a sequence of minimization problems. As usual with the PPA, the solutions of this sequence of problems converge linearly to the solution of (P). Moreover, in each of these minimization problems, the objective function consists of the objective function of (P) plus the usual proximal term. We solve each of these problems by a dual method. This means that we first compute a Kuhn-Tucker vector of the problem in order to solve the problem. These Kuhn-Tucker vectors converge to the set of solutions of (D).

The above construction is possible because the objective function (in the maximization providing the above Kuhn-Tucker vectors) equals a known Moreau envelope plus a quadratic term. This has two consequences :

1. It has the smoothness of a Moreau envelope. In particular, its gradient is Lipschitz (which makes the maximization easy).
2. We can evaluate it and compute its gradient with closed form formulae.

1.4 Notation and hypotheses

The following notation and hypotheses hold throughout the paper.

For any positive integer k, we denote $v \in \mathbb{R}^k$ a column vector of length k. Moreover, we denote the entries of v by $(v_i)_{i=1..k}$. The usual norms are defined by

$$\|v\|_2 \overset{\text{def}}{=} \sqrt{\sum_{i=1}^{k} v_i^2},$$

$$\|v\|_1 \overset{\text{def}}{=} \sum_{i=1}^{k} |v_i|,$$

and

$$\|v\|_\infty \overset{\text{def}}{=} \max_{i=1..k} |v_i|.$$

We denote

$$\mathbb{R}^k_+ \overset{\text{def}}{=} \{v \in \mathbb{R}^k : \forall i = 1..k, v_i \geq 0\}.$$

For any positive integer k and any u and $v \in \mathbb{R}^k$, $u \leq v$ means that, for all $i = 1..k$, $u_i \leq v_i$. (We define $\geq$ similarly.) Also $\max(u, v) \overset{\text{def}}{=} (\max(u_i, v_i))_{i=1..k}$. The vector u^t is the transpose of u. The vector $\mathbf{1}_k$ is column vector of size k with all its entries equal to 1.

For any set $F \subset \mathbb{R}^k$, the indicator function of F is denoted by χ_F and it equals 0, on F, and $+\infty$ outside F.

The matrix A is an N-by-P (N and P are positive integers) matrix which positively generates $\mathbb{R}^N$. More precisely, the matrix A is such that

$$\forall d \in \mathbb{R}^N, \text{ there exists } x \in \mathbb{R}^P_+ \text{ such that } Ax = d.$$

Under this hypothesis, (D) has a solution. Notice also that, depending on A, this solution might not be unique. (A trivial example of non-uniqueness is when two columns of A are equal.)

For any set of columns $J \subset \{1, \dots, P\}$, the matrix A^J is a N-by-$\#J$ matrix (where $\#$ denotes the cardinal of a set) containing the columns of A whose index are in J. When J only contains one element, we write A^j instead of A^j.

We also assume that the datum $b \in \mathbb{R}^N$ and the parameter $\tau > 0$ are such that $\|b\|_2 > \tau$. Otherwise, the problem (D) is trivial ($x = 0$ solves (D)).

1.5 Organization of the paper

In section 2, we build the PPPA. The predual is computed in Section 2.1. The general form of the algorithm and main statement concerning its convergence are given in Section 2.2. The proof of the convergence is given in Section 2.3. In Section 2.4, the fact that the objective functions which are maximized have a Lipschitz gradient is established. Then, some calculations yield closed form formulae for the main computations of the algorithms (see Section 2.5). The pseudo-code of the PPPA is given in Section 2.6. It is easy to implement. Details and a variation around this algorithm are described in Section 2.7.

Then, some experiments are explained and commented in Section 3. The aims of these experiments are to understand the role of parameter of the algorithm and to compare it to the IT, the PCD and the GPSR-BB.

We finally give some perspectives in section 4.

2 Building the algorithms

2.1 The Predual formulation

In this section, we consider the optimization problem (P) below and show that its dual is indeed the above problem (D). In other words, we prove that (P) is the predual of (D).

$$(P) \left\{ \begin{array}{l} \min_{d \in \mathbb{R}^N} \tau \|d\|_2 - d^t b \\ \text{under the constraints } A^t d \leq \mathbf{1}_P. \end{array} \right.$$

The Lagrangian of the problem (P) is

$$L(d, x) \overset{\text{def}}{=} \tau \|d\|_2 - d^t b + x^t (A^t d - \mathbf{1}_P),$$

where the vector $x \in \mathbb{R}^P_+$ contains the dual variables.

The unique[1] solution c^* of (P) is also the first argument of any saddle point (c^*, x^*) of the Lagrangian. We also know that the second argument x^* of any saddle point of the Lagrangian solves the dual of (P) (x^* is called a Kuhn-Tucker vector of (P)).

[1]The existence and uniqueness of the solution of (P) are easy to establish and need not be detailed here. They are given in [38].

The dual of (P) is the following problem

$$\max_{x\in\mathbb{R}_+^P}\min_{d\in\mathbb{R}^N} L(d,x) = \max_{x\in\mathbb{R}_+^P}\left(\min_{d\in\mathbb{R}^N}\tau\|d\|_2 + d^t(Ax-b)\right) - x^t\mathbf{1}_P. \tag{6}$$

Finally, notice that we have

$$\min_{d\in\mathbb{R}^N}\tau\|d\|_2 + d^t(Ax-b) = \begin{cases} -\infty & \text{, if } b-Ax\notin\tau\partial\|.\|_2(0) \\ 0 & \text{, otherwise,}\end{cases}$$

with

$$\partial\|.\|_2(0) = \{d\in\mathbb{R}^N, \|d\|_2\le 1\}.$$

Integrating those results into (6), we finally obtain the dual of (P):

$$\begin{cases} \max_{x\in\mathbb{R}_+^P} & -x^t\mathbf{1}_P \\ \text{under the constraint } \|Ax-b\|_2\le\tau, \end{cases}$$

which is precisely the problem (D) considered in the preceding section.

It follows that the problem (D) can be solved by any algorithm solving (P) if this algorithm also provides one of its Kuhn-Tucker vectors x^*.

In the following, we will only consider a small family of such algorithms. (Our motivation for considering this family will be clear after Section 2.4 and 2.5) This family is described in the next section.

2.2 Applying the Proximal Point Algorithm to (P)

We propose to solve the predual (P) with a PPA (see, among other, [35, 24]). We denote

$$g_{c,\alpha}(d) \overset{\text{def}}{=} \alpha\|d-c\|_2^2 + \tau\|d\|_2 - d^tb + \chi_F(d),$$

where $F \overset{\text{def}}{=} \{d\in\mathbb{R}^N \ : \ A^td\le\mathbf{1}_P\}$ is the feasible set of (P).

Applying the PPA to (P) means that we compute the following sequence:

$$c^{m+1} = \text{argmin}_{d\in\mathbb{R}^N} g_{c^m,\alpha_m}(d), \tag{7}$$

for a given c^0 and a nonnegative bounded sequence $(\alpha_m)_{m\in\mathbb{N}}$.

General results on the PPA (see [35]) guarantee that $(c^m)_{m\in\mathbb{N}}$ converges linearly to the solution c^* of (P). (A more precise statement is given in Theorem 1).

For the PPPA, we need to go one step further and prove that this implies the convergence of the corresponding Kuhn-Tucker vectors. We also need to compute those vectors, since our actual goal is to solve (D). Therefore we solve (7) with a dual method. In order to do so, we consider the Lagrangian of (7) :

$$\begin{aligned} L'(d,x,c^m,\alpha_m) &\overset{\text{def}}{=} \alpha_m\|d-c^m\|_2^2 + L(d,x) \\ &= \alpha_m\|d-c^m\|_2^2 + \tau\|d\|_2 + d^t(Ax-b) - x^t\mathbf{1}_P. \end{aligned} \tag{8}$$

We know that a Kuhn-Tucker vector of (7) exists (see [34], Th 28.2, p.277) and maximizes on $\mathbb{R}_+^P$:

$$f_{c^m,\alpha_m}(x) \overset{\text{def}}{=} \min_{d\in\mathbb{R}^N} L'(d, x, c^m, \alpha_m).$$

Moreover, for any Kuhn-Tucker vector $x^* \in \text{argmax}_{x\in\mathbb{R}_+^P} f_{c^m,\alpha_m}(x)$, we have

$$c^{m+1} = \text{argmin}_{d\in\mathbb{R}^N} L'(d, x^*, c^m, \alpha_m).$$

The family of algorithms which we call Predual Proximal Point Algorithm (PPPA) is described in Table 1.

```
• Initialize $c^0$
• Repeat until convergence (loop in $m$)
    1. Use a gradient based algorithm for solving
           $x^m \in \text{argmax}_{x\in\mathbb{R}_+^P} f_{c^m,\alpha_m}(x)$
    2. Update $c^{m+1} = \text{argmin}_{d\in\mathbb{R}^N} L'(d, x^m, c^m, \alpha_m)$.
```

Table 1: General form of the algorithm. The gradient based algorithm still needs to be specified.

Notice that the function $f_{c^m,\alpha_m}(x)$ is concave because L' is concave in x. We also know that $-f_{c^m,\alpha_m}$ is coercive on $\mathbb{R}_+^P$ (for instance $f_{c^m,\alpha_m}(x) \leq L'(0, c^m, x, \alpha_m) = -\|x\|_1 + cst$).

We will show in Section 2.4 that $x \mapsto \nabla f_{c^m,\alpha_m}(x)$ is Lipschitz and we will provide an upper bound of its Lipschitz constant (this bound can be computed numerically). Together, this will guarantee the convergence of most gradient based algorithms considered in Step 1.

Notice that, besides the matrix-vector multiplication, the only difficulties in the implementation of the above algorithm are the computations of $\nabla f_{c^m,\alpha_m}(x)$, in Step 1, the resolution of Step 2 and, depending on the gradient based algorithm in Step 1, the evaluation of $f_{c^m,\alpha_m}(x)$. We will show in Section 2.5 that these computations can be performed with closed form formulae. Essentially, the cost of the evaluation of $\nabla f_{c^m,\alpha_m}(x)$ is two matrix-vector multiplications; the cost for computing $\text{argmin}_{c\in\mathbb{R}^N} L'(d, x^m, c^m, \alpha_m)$ and for evaluating $f_{c^m,\alpha_m}(x)$ is one matrix-vector multiplication.

Before going into those details, let us first state the following theorem, which guarantees that the PPPA approximates actual solutions of (P) and (D). It also guarantees that the loop in m of Table 1 converges rapidly and is short. The proof of the theorem is contained in the next section (the proof is independent of the rest of the paper and can be skipped); it is a straightforward application of a well known result which is true for many convex problem (see [35]). In particular, the linear convergence rate of the point (1) is similar to what is known for those convex problems. The main difficulty is to prove that (P) satisfies the hypotheses of Theorem 2, in [35]. The convergence of $(Ax^m)_{m\in\mathbb{N}}$ and $(x^m)_{m\in\mathbb{N}}$ are easily obtained once the point (1) is established.

Theorem 1 *Assume $(\alpha_m)_{m\in\mathbb{N}}$ is a bounded nonnegative sequence and $\|b\|_2 > \tau$, there exists $t > 0$ and $T > 0$ such that the sequences $(c^m)_{m\in\mathbb{N}}$ and $(x^m)_{m\in\mathbb{N}}$ defined in Table 1 satisfy*

1. *$(c^m)_{m\in\mathbb{N}}$ converges to c^*. If moreover, $(\alpha_m)_{m\in\mathbb{N}}$ is nonincreasing and for $t_m = \frac{t}{\sqrt{t^2+\alpha_m^{-2}}} < 1$*

$$\|c^{m+1} - c^*\|_2 \leq t_m\|c^m - c^*\|_2, \qquad \forall m \geq T.$$

In words, $(c^m)_{m\in\mathbb{N}}$ converges to c^* faster than linearly with convergence ratio $\frac{t}{\sqrt{t^2+\liminf_{m\to\infty}(\alpha_m^{-2})}}$.

2. *For any $x^* \in \mathcal{S}$, where $\mathcal{S}$ is the optimal set of (D), $(Ax^m)_{m\in\mathbb{N}}$ converges to Ax^*. If moreover, $(\alpha_m)_{m\in\mathbb{N}}$ is nonincreasing and for $T_m = 4\alpha_m + \frac{2\tau}{\|c^*\|_2}$,*

$$\|Ax^m - Ax^*\|_2 \leq T_m\|c^m - c^*\|_2, \qquad \forall m \geq T.$$

The left term converges to 0 with the same rate as $(\|c^m - c^*\|_2)_{m\in\mathbb{N}}$.

3. *Given $d(x^m, \mathcal{S}) \overset{\text{def}}{=} \min_{x^*\in\mathcal{S}} \|x^m - x^*\|_2$, we have*

$$\lim_{m\to+\infty} d(x^m, \mathcal{S}) = 0.$$

2.3 Proof of Theorem 1

2.3.1 Proof of Theorem 1, point (1)

The first statement is a direct application of Theorems 1 and 2 of [35]. Theorem 1 of [35] guarantees that $(c^m)_{m\in\mathbb{N}}$ converges to c^*. We can apply it because (P) has a solution. In order to apply Theorem 2 of [35], we only need to show that ∂g^{-1} is Lipschitz continuous at 0 (see [35]), for g defined for $d \in \mathbb{R}^N$ by

$$g(d) \overset{\text{def}}{=} \tau\|d\|_2 - d^t b + \chi_F(d),$$

where we recall that $F \overset{\text{def}}{=} \{d \in \mathbb{R}^N \ : \ A^t d \leq \mathbf{1}_P\}$ is the feasible set of (P).

The Lipschitz continuity of ∂g^{-1} at 0 follows from Proposition 7, in [35]: indeed, we show that, since $\|b\|_2 > \tau$, the function g satisfies the second statement (named b) of Proposition 7, in [35].

First, notice that, when $\|b\|_2 > \tau$, the minimizer c^* of g (i.e. the solution of (P)) is unique. The proof of this statement is straightforward and is given in [38].

Therefore, in order to apply Proposition 7 of [35], we only need to show that

$$\liminf_{d\to c^*} \frac{g(d)-g(c^*)}{\|d-c^*\|_2^2} > 0. \tag{9}$$

In order to prove this last statement, let us first remark that, since $\|b\|_2 > \tau$, $c^* \neq 0$. Therefore, $d \longrightarrow \tau\|d\|_2 - d^t b$ is infinitely differentiable at c^*. The second order Lagrange expansion holds for this function and, for all $d \in F$ (c^* trivially belongs to F),

$$\begin{aligned} g(d) = g(c^*) &+ \left(\tau\frac{c^*}{\|c^*\|_2} - b\right)^t (d-c^*) \\ &+ \frac{\tau}{2\|c^*\|_2^3}\left(\|d-c^*\|_2^2\|c^*\|_2^2 - \left((c^*)^t(d-c^*)\right)^2\right) + o(\|d-c^*\|_2^2). \end{aligned} \tag{10}$$

The difficulty, when trying to prove (9), is to prove that the first and second order terms of (10) cannot cancel simultaneously. We achieve this by splitting F into two complementary sets : one set on which the first order term is far from 0; its complement on which the second order term is far from 0. The details are given below. Before doing so, we establish some results which hold for any $d \in F$.

First, since c^* solves (P), there exists a Kuhn-Tucker vector $x \in \mathbb{R}_+^P$ such that (see [34], Th. 28.3, p. 281)

$$\forall i = 1..P,\ x_i((A^i)^t c^* - 1) = 0 \tag{11}$$

and

$$\tau\frac{c^*}{\|c^*\|_2} - b = -Ax. \tag{12}$$

Notice that, since $\|b\|_2 > \tau$, $x \neq 0$ and there exists $i_0 \in I$ such that $x_{i_0} > 0$. Then, by (11), for any $i = 1..P$ such that $x_i > 0$ (and in particular for i_0),

$$(A^i)^t c^* = 1 \tag{13}$$

and therefore $x_i(A^i)^t(c^*-d) \geq 0$, for all $d \in F$. Thus, using (12)

$$\begin{aligned} \left(\tau\frac{c^*}{\|c^*\|_2} - b\right)^t (d-c^*) &= (Ax)^t(c^*-d) \\ &= x^t\ [A^t(c^*-d)] \\ &= \sum_{i=1}^P x_i(A^i)^t(c^*-d) \\ &\geq x_{i_0}(A^{i_0})^t(c^*-d) \qquad (14) \\ &\geq 0. \qquad (15) \end{aligned}$$

Let us now constructs the sets which we evoked in the paragraph following (10).

To do so, notice that for any $d \in \mathbb{R}^N$ (and in particular for $d \in F$) there exists $\beta \in \mathbb{R}$ and $r \in \mathbb{R}^N$ such that $d = (1-\beta)c^* + r$, with $r^t c^* = 0$. Moreover, β and r are unique. (We use this notation throughout this proof without explicitly recalling it.) Notice that

$$d - c^* = r - \beta c^*. \tag{16}$$

Let us denote by

$$E = \{d = (1-\beta)c^* + r \in F, \text{with } r^t c^* = 0, \beta \geq 0 \text{ and } \|r\|_2 \leq \frac{\beta}{2\|A^{i_0}\|_2}\},$$

where we recall that i_0 is a given index such that $x_{i_0} > 0$.

We are going to show that the first order term in (10) cannot cancel on E.

We obtain, using successively (10) (and the convexity of g), (14), (16) and (13) that, for all $d \in E$,

$$\begin{aligned} g(d) - g(c^*) &\geq \left(\tau \frac{c^*}{\|c^*\|} - b\right)^t (d - c^*) + o(\|d - c^*\|_2^2) \\ &\geq x_{i_0} (A^{i_0})^t (\beta c^* - r) + o(\|d - c^*\|_2^2) \\ &\geq x_{i_0} (\beta - (A^{i_0})^t r) + o(\|d - c^*\|_2^2) \end{aligned}$$

Since, for all $d \in E$

$$\|(A^{i_0})^t r\|_2 \leq \|(A^{i_0})^t\|_2 \|r\|_2 \leq \frac{\beta}{2},$$

we obtain

$$g(d) - g(c^*) \geq x_{i_0} \frac{\beta}{2} + o(\|d - c^*\|_2^2).$$

Moreover, for $d \in E$,

$$\|d - c^*\|_2^2 = \beta^2 \|c^*\|_2^2 + \|r\|_2^2 \leq (\|c^*\|_2^2 + \frac{1}{4\|A^{i_0}\|_2^2})\beta^2.$$

Finally, for $d \in E$,

$$g(d) - g(c^*) \geq \frac{x_{i_0}}{2} \frac{\|d - c^*\|_2}{\sqrt{\|c^*\|_2^2 + \frac{1}{4\|A^{i_0}\|_2^2}}} + o(\|d - c^*\|_2^2).$$

So we get

$$\liminf_{\substack{d \to c^* \\ d \in E}} \frac{g(d) - g(c^*)}{\|d - c^*\|_2^2} = +\infty. \tag{17}$$

Let us now consider the situation where $d \in F \setminus E$, we deduce from (10) and (15) that

$$g(d) - g(c^*) \geq \frac{\tau}{2\|c^*\|_2^3} (\|d - c^*\|_2^2 \|c^*\|_2^2 - \left((c^*)^t (d - c^*)\right)^2) + o(\|d - c^*\|_2^2).$$

Decomposing again $d = (1-\beta)c^* + r$, with $r^t c^* = 0$, we obtain

$$\begin{aligned} g(d) - g(c^*) &\geq \frac{\tau}{2\|c^*\|_2^3}((\beta^2\|c^*\|_2^2 + \|r\|_2^2)\|c^*\|_2^2 - \beta^2\|c^*\|_2^4) + o(\|d-c^*\|_2^2) \\ &= \frac{\tau\|r\|_2^2}{2\|c^*\|_2} + o(\|d-c^*\|_2^2). \end{aligned} \tag{18}$$

Moreover, for $d \in F \setminus E$, we either have $\beta < 0$ or $\|r\|_2 > \frac{\beta}{2\|A^{i_0}\|_2} \geq 0$. In the latter case, we trivially have

$$\beta^2 \leq 4\|A^{i_0}\|_2^2\|r\|_2^2.$$

If $\beta < 0$, we have, using the definition of F and (13),

$$1 \geq (A^{i_0})^t d \geq (1-\beta) + (A^{i_0})^t r.$$

So

$$(A^{i_0})^t r \leq \beta$$

and

$$0 < -\beta \leq -(A^{i_0})^t r \leq \|A^{i_0}\|_2\|r\|_2.$$

This implies that $\beta^2 \leq \|A^{i_0}\|_2^2\|r\|_2^2$.

We finally obtain that, whenever $d \in F \setminus E$,

$$\beta^2 \leq 4\|A^{i_0}\|_2^2\|r\|_2^2$$

and therefore

$$\|d-c^*\|_2^2 = \beta^2\|c^*\|_2^2 + \|r\|_2^2 \leq (4\|A^{i_0}\|_2^2\|c^*\|_2^2 + 1)\|r\|_2^2.$$

Together with (18), this guarantees that

$$\liminf_{\substack{d \to c^* \\ d \in F \setminus E}} \frac{g(d) - g(c^*)}{\|d-c^*\|_2^2} \geq \frac{\tau}{2\|c^*\|_2(4\|A^{i_0}\|_2^2\|c^*\|_2^2 + 1)} > 0.$$

This result and (17) guarantee that (9) holds and concludes the proof of the first statement.

2.3.2 Proof of Theorem 1, point (2)

Again, since $\|b\|_2 > \tau$, $c^* \neq 0$. The objective function of (P) is differentiable at c^* and, for any $x^* \in \mathcal{S}$,

$$\tau\frac{c^*}{\|c^*\|_2} - b + Ax^* = 0. \tag{19}$$

Since c^{m+1} converges to c^*, for m large enough, c^{m+1} cannot be zero. Given the definition of c^{m+1} (see Table 1), we therefore know that

$$\frac{\partial L'}{\partial d}(c^{m+1}, x^m, c^m, \alpha_m) = 2\alpha_m(c^{m+1} - c^m) + \tau\frac{c^{m+1}}{\|c^{m+1}\|_2} + Ax^m - b = 0. \tag{20}$$

Using (19) and (20), we obtain

$$Ax^* - Ax^m = 2\alpha_m(c^{m+1} - c^m) + \tau\left(\frac{c^{m+1}}{\|c^{m+1}\|_2} - \frac{c^*}{\|c^*\|_2}\right).$$

Since $(c^m)_{m\in\mathbb{N}}$ converges to c^*, this leads to $(Ax^m)_{m\in\mathbb{N}}$ converges to Ax^*. Moreover, if $(\alpha_m)_{m\in\mathbb{N}}$ is nonincreasing,

$$\begin{aligned}
\|A(x^*-x^m)\|_2 &\leq 2\alpha_m(\|c^{m+1}-c^*\|_2 + \|c^m - c^*\|_2) + \tau\left\|\frac{\|c^*\|_2 c^{m+1} - \|c^{m+1}\|_2 c^*}{\|c^*\|_2\|c^{m+1}\|_2}\right\|_2 \\
&\leq 2\alpha_m\left(\frac{t}{\sqrt{t^2+\alpha_m^{-2}}}+1\right)\|c^m - c^*\|_2 \\
&\qquad + \tau\left\|\frac{(\|c^*\|_2 - \|c^{m+1}\|_2)c^{m+1} - \|c^{m+1}\|_2(c^* - c^{m+1})}{\|c^*\|_2\|c^{m+1}\|_2}\right\|_2 \\
&\leq 4\alpha_m\|c^m - c^*\|_2 + \tau\frac{\left|\|c^*\|_2 - \|c^{m+1}\|_2\right| + \|c^* - c^{m+1}\|_2}{\|c^*\|_2} \\
&\leq \left[4\alpha_m + \frac{2t\tau}{\sqrt{t^2+\alpha_m^{-2}}\|c^*\|_2}\right]\|c^m - c^*\|_2 \\
&\leq \left(4\alpha_m + \frac{2\tau}{\|c^*\|_2}\right)\|c^m - c^*\|_2
\end{aligned}$$

This concludes the proof of the second statement.

2.3.3 Proof of Theorem 1, point (3)

In order to establish the last statement of Theorem 1, we are going to show that $(x^m)_{m\in\mathbb{N}}$ is bounded in $\mathbb{R}_+^P$ and that any convergent sub-sequence of $(x^m)_{m\in\mathbb{N}}$ converges to an element in $\mathcal{S}$.

Let us first remark that, because of the definition of c^{m+1} and x^m, we have for any $x^* \in \mathcal{S}$ (as for any element of $\mathbb{R}_+^P$),

$$L'(c^{m+1}, x^m, c^m, \alpha_m) \geq L'(c^{m+1}, x^*, c^m, \alpha_m).$$

Using the definition of L' (see (8)), we obtain

$$(c^{m+1})^t Ax^m - (x^m)^t\mathbf{1}_P \geq (c^{m+1})^t Ax^* - (x^*)^t\mathbf{1}_P.$$

So,

$$(x^m)^t\mathbf{1}_P \leq (x^*)^t\mathbf{1}_P + \|c^{m+1}\|_2\|A(x^*-x^m)\|_2. \tag{21}$$

Since $\lim_{m\to+\infty} c^m = c^*$ and $\lim_{m\to+\infty} A(x^*-x^m) = 0$, we are guaranteed that there exists $B > 0$, such that, for all $m \in \mathbb{N}$,

$$(x^m)^t\mathbf{1}_P \leq B.$$

As a consequence, $(x^m)_{m\in\mathbb{N}}$ is bounded in $\mathbb{R}_+^P$.

Let $\overline{x}$ be an accumulation point of $(x^m)_{m\in\mathbb{N}}$. We know from the second statement of the theorem that $\lim_{m\to+\infty} Ax^m = Ax^*$ and therefore

$$A\overline{x} = Ax^*. \tag{22}$$

This guarantees that $\overline{x}$ belongs to the feasible set of (D). Moreover, using (21), this leads to

$$\overline{x}^t \mathbf{1}_P \leq (x^*)^t \mathbf{1}_P.$$

Now, (22) and the fact that $x^* \in \mathcal{S}$ guarantee that the converse inequality holds. We finally have

$$\overline{x}^t \mathbf{1}_P = (x^*)^t \mathbf{1}_P.$$

Altogether, this implies $\overline{x} \in \mathcal{S}$ and concludes the proof.

2.4 The gradient of $f_{c,\alpha}$ is Lipschitz

The Lipschitz continuity of $\nabla f_{c,\alpha}$ is very important since, in Step 1 of Table 1, we need to maximize $f_{c,\alpha}$. Remember that we already know that $f_{c,\alpha}$ is concave and $-f_{c,\alpha}$ is coercive on $\mathbb{R}_+^P$.

Before establishing this result, let us recall some facts about the Moreau envelope of $\|.\|_2$ and let us express $f_{c,\alpha}$ using this Moreau envelope.

We denote the Moreau envelope of $\|.\|_2$ by e_β. It is defined for $c \in \mathbb{R}^N$ and $\beta > 0$, by

$$e_\beta(c) \overset{\text{def}}{=} \min_{d\in\mathbb{R}^N} \beta\|d-c\|_2^2 + \|d\|_2. \tag{23}$$

As is usual for the Moreau envelope (see the introduction of [27]), for any d and $d' \in \mathbb{R}^N$,

$$\|\nabla e_\beta(d) - \nabla e_\beta(d')\|_2 \leq 2\beta\|d-d'\|_2. \tag{24}$$

Also, for this particular Moreau envelope, we know that the vector d^* solving the optimization problem on the right hand side of (23) is

$$d^* = \begin{cases} 0 & \text{, if } \|c\|_2 \leq \frac{1}{2\beta} \\ \frac{2\beta\|c\|-1}{2\beta\|c\|}c & \text{, otherwise.} \end{cases} \tag{25}$$

Recalling that, for $x \in \mathbb{R}_+^P$, $c \in \mathbb{R}^N$ and $\alpha > 0$,

$$L'(d,x,c,\alpha) = \alpha\|d-c\|_2^2 + \tau\|d\|_2 + d^t(Ax-b) - x^t\mathbf{1}_P$$

we can rewrite

$$\begin{array}{rcl} L'(d,x,c,\alpha) & = & \alpha\left\|d-c+\frac{1}{2\alpha}(Ax-b)\right\|_2^2 + \tau\|d\|_2 \\ & & +c^t(Ax-b) - \frac{1}{4\alpha}\|Ax-b\|_2^2 - x^t\mathbf{1}_P. \end{array} \tag{26}$$

Therefore, for any $x \in \mathbb{R}_+^P$,

$$\begin{array}{rcl} f_{c,\alpha}(x) & = & \tau e_{\frac{\alpha}{\tau}}\left(c - \frac{1}{2\alpha}(Ax-b)\right) \\ & & +c^t(Ax-b) - \frac{1}{4\alpha}\|Ax-b\|_2^2 - x^t\mathbf{1}_P. \end{array} \tag{27}$$

We can now state the main result of the section.

Theorem 2 *For any $c \in \mathbb{R}^N$, $\alpha > 0$, any x and x' in $\mathbb{R}_+^P$,*

$$\|\nabla f_{c,\alpha}(x) - \nabla f_{c,\alpha}(x')\|_2 \leq C\|x - x'\|_2,$$

where $C \overset{\text{def}}{=} \frac{M^2}{\alpha}$, for M the operator norm of A :

$$M \overset{\text{def}}{=} \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}.$$

Proof. Differentiating (27), we obtain that, for any $x \in \mathbb{R}_+^P$,

$$\nabla f_{c,\alpha}(x) = -\frac{\tau}{2\alpha} A^t \ \nabla e_{\frac{\alpha}{\tau}} \left(c - \frac{1}{2\alpha}(Ax - b) \right) + A^t c - \frac{1}{2\alpha} A^t (Ax - b) - \mathbf{1}_P.$$

Writing, for x and $x' \in \mathbb{R}_+^P$,

$$e \overset{\text{def}}{=} \nabla e_{\frac{\alpha}{\tau}}(c - \frac{1}{2\alpha}(Ax - b)) \tag{28}$$

and

$$e' \overset{\text{def}}{=} \nabla e_{\frac{\alpha}{\tau}}(c - \frac{1}{2\alpha}(Ax' - b)), \tag{29}$$

we obtain

$$\begin{aligned} \nabla f_{c,\alpha}(x) - \nabla f_{c,\alpha}(x') &= \frac{\tau}{2\alpha} A^t (e' - e) + \frac{1}{2\alpha} A^t A (x' - x), \\ &= \frac{1}{2\alpha} A^t \Big(\tau(e' - e) + A(x' - x) \Big). \end{aligned}$$

Notice that we have

$$M = \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sup_{d \neq 0} \frac{\|A^t d\|_2}{\|d\|_2}.$$

Indeed, they are both equal to the square root of the largest singular value of the matrix A.

We can therefore deduce that

$$\begin{aligned} \|\nabla f_{c,\alpha}(x) - \nabla f_{c,\alpha}(x)\|_2 &\leq \frac{M}{2\alpha} \|\tau(e' - e) + A(x - x')\|_2, \\ &\leq \frac{M}{2\alpha} \left(\tau\|e' - e\|_2 + \|A(x' - x)\|_2 \right). \end{aligned}$$

Moreover, using (28), (29) and (24)

$$\|e' - e\|_2 \leq 2\frac{\alpha}{\tau} \| \frac{1}{2\alpha} A(x - x')\|_2.$$

Finally, for any $x \in \mathbb{R}_+^P$ and $x' \in \mathbb{R}_+^P$,

$$\begin{aligned} \|\nabla f_{c,\alpha}(x) - \nabla f_{c,\alpha}(x')\|_2 &\leq \frac{M}{\alpha} \|A(x' - x)\|_2 \\ &\leq \frac{M^2}{\alpha} \|x' - x\|_2. \end{aligned}$$

$\square$

The above theorem is important since it guarantees that, when solving the first step of Table 1, most gradient based algorithms with constant step size converge for some known step size (see next sections). Together with Theorem 1, this ensures that the whole algorithm converges to the desired solution.

However, in order to chose the step size in these algorithms we need to have an estimate of the best possible Lipschitz constant. This can, of course, be done experimentally be running the algorithm for several step sizes, with all the other parameters fixed.

A more flexible way to chose the step size is to use the expression for C which is given in the Theorem 2. In order to compute M, we can use the algorithm described in Table 2. This algorithm computes the square root of the largest eigenvalue of the matrix A^tA, which equals M.

```
• Input : a matrix $A$
• Output : the constant $M$
• The algorithm :
    – Initialize $x^0 \neq 0$.
    – Repeat until convergence (loop in $k$)
        1. Normalize $x^k \leftarrow \frac{x^k}{\|x^k\|_2}$
        2. Compute $d^k = Ax^k$
        3. Update $x^{k+1} = A^t d^k$
    – Set $M = \|x^{k+1}\|_2^{\frac{1}{2}}$
```

Table 2: Iterative algorithm computing the constant M (see Theorem 2).

An alternative is to use the following upper bound of M. When the dictionary is the union of K orthonormal sets (for instance K orthonormal bases), we have

$$M \leq \sqrt{K}.$$

Finally, we will see from (31) and (33) that $f_{c,\alpha}$ does not satisfy any sort of ellipticity or strong convexity property. This rules out the application of theorems which require this property, when studying the convergence of a gradient based algorithm for solving Step 1 of Table 1.

2.5 Exact resolution of Step 2 and exact computation of $\nabla f_{c,\alpha}(x)$ and $f_{c,\alpha}(x)$

First, as is usual with the gradient of functions defined as a minimum, when computing $\nabla f_{c,\alpha}(x)$ many terms cancels out[2] and we have

$$\nabla f_{c,\alpha}(x) = A^t c^* - \mathbf{1}_P,$$

where

$$c^* = \mathrm{argmin}_{d\in\mathbb{R}^N} L'(d,x,c,\alpha). \tag{30}$$

As a consequence, the resolution of Step 2 in Table 1, the evaluation of $f_{c,\alpha}(x)$ and, modulo a multiplication by A^t, the computation of $\nabla f_{c^m,\alpha_m}(x)$ boil down to the same problem : the solution of (30).

In order to calculate the solution of (30), we use (26) which guarantees that

$$c^* = \mathrm{argmin}_{d\in\mathbb{R}^N} \frac{\alpha}{\tau}\left\| d - c + \frac{1}{2\alpha}(Ax-b)\right\|_2^2 + \|d\|_2.$$

Using (25), we obtain

$$c^* = \begin{cases} 0 & \text{, if } \|2\alpha c - (Ax-b)\|_2 \le \tau \\ \frac{\|2\alpha c-(Ax-b)\|-\tau}{2\alpha\|2\alpha c-(Ax-b)\|}(2\alpha c - (Ax-b)) & \text{, otherwise.} \end{cases}$$

We conclude that in Step 1 of the algorithm described in Table 1, the function $f_{c,\alpha}$ and its gradient can be computed with :

$$\nabla f_{c,\alpha}(x) = A^t c^* - \mathbf{1}_P, \tag{31}$$

and

$$f_{c,\alpha}(x) = L'(c^*,x,c,\alpha), \tag{32}$$

where

$$\begin{cases} c^* = \begin{cases} 0 & \text{, if } \|d\|_2 \le \tau \\ \frac{\|d\|_2-\tau}{2\alpha_m\|d\|_2}\, d & \text{, otherwise,} \end{cases} \\ \text{with } d = 2\alpha c + b - Ax. \end{cases} \tag{33}$$

Moreover, Step 2 of the algorithm in Table 1 is solved by applying (33) at x^m.

[2]Notice that the differentiation is not that trivial since, in L', the optimal c depends on x. Heuristically, as is common with such max min problems, the term $\frac{\partial L'}{\partial c}$ equals zero and it cancels the terms $\frac{\partial c^*}{\partial x}$ which appear in the calculation of $\nabla f_{c,\alpha}(x)$. For an example of such a calculation, see the construction of the Uzawa algorithm in [8]

2.6 A simple version of the PPPA

In this section, we present the algorithm obtained when the gradient based algorithm used to solve Step 1 of the algorithm described in Table 1 is a simple projected gradient ascent with a constant time step. Given Theorem 2, we know (see [32], Cor. 2.1.2, p. 70, and Th. 2.2.8, p. 88) that it converges as soon as the time step is in the range $(0, \frac{2}{C})$, where C is as given in Theorem 2. Moreover, the "best time step" is $\rho = \frac{1}{C}$.

We also know (see [32]) that, for $\rho = \frac{1}{C}$, $c^m \in \mathbb{R}^N$ and $\alpha^m > 0$, there exists a constant $C_1 > 0$ (which depends on the quality of the initialization) such that

$$|f^*_{c^m,\alpha_m} - f_{c^m,\alpha_m}\left(x^{m,k}\right)| \leq C_1 \frac{2C}{k+4},$$

where $x^{m,k}$ is the result of the algorithm at the k^{th} iteration, when solving Step 1 in Table 1 at the m^{th} iteration, and

$$f^*_{c^m,\alpha_m} \overset{\text{def}}{=} \max_{x \in \mathbb{R}^P_+} f_{c^m,\alpha_m}(x).$$

The final algorithm is described in Table 3.

```
• Inputs : $\tau > 0$, the initial image $b \in \mathbb{R}^N$, a matrix
  $A$ and $(\alpha_m)_{m \in \mathbb{N}}$
• Output : the coordinates $x$
• The algorithm :
    – Initialize $x^{0,0} \in \mathbb{R}^P_+$, $c^0 \in \mathbb{R}^N$ and $\rho = \frac{1}{C}$.
    – Repeat until convergence (loop in $m$)
        ∗ Repeat until convergence (loop in $k$)
          1. Compute $d^k = 2\alpha_m c^m + b - Ax^{m,k}$
          2. if $(\|d^k\|_2 \leq \tau)$, set $d^k = 0$
             otherwise, set $d^k \leftarrow a\ d^k$, with $a = \frac{\|d^k\|_2 - \tau}{2\alpha_m \|d^k\|_2}$
          3. Update $x^{m,k+1}$,
             $x^{m,k+1} = \max\left(0, x^{m,k} + \rho(A^t d^k - \mathbf{1}_P)\right)$
        ∗ Update $c^{m+1} = d^k$ and $x^{m+1,0} = x^{m,k+1}$
```

Table 3: PPPA solving (D) : Step 1 of the algorithm described in Table 1 is solved by a projected gradient descent with a constant step size.

The details on the initialization are given in Section 2.7.2. In the experiments, the constant C was estimated using Theorem 2 and the algorithm described in Table 2.

2.7 Details and variants of the algorithm

This section contains some details on the use of the above algorithm when solving the usual BPDN (instead of a NNBPDN), the initialization, the stopping criteria and the possibilities we investigated for $(\alpha_m)_{m\in\mathbb{N}}$.

Also, there exist many gradient based algorithms for solving the Step 1 in Table 1. In addition to the projected gradient algorithm with constant step described in the above section, we have implemented another version. This version is described in this section.

2.7.1 Symmetric and partly symmetric dictionaries

The algorithm presented so far solves a NNBPDN. We would like to emphasize that, this generalization is not made at any expense when the PPPA is used to solve the BPDN. The PPPA can be applied when the dictionary is symmetric (i.e. $\exists J \subset \{1,\ldots,P\}$, such that $(A^i)_{i=1..P} = (A^j)_{j\in J} \cup (-A^j)_{j\in J}$) or partly symmetric (i.e. $\exists J$ and $J' \subset \{1,\ldots,P\}$, such that $(A^i)_{i=1..P} = (A^j)_{j\in J'} \cup (A^j)_{j\in J} \cup (-A^j)_{j\in J}$).

For simplicity, let us consider a symmetric dictionary $(A^i)_{i=1..P} = (A^j)_{j=1..\frac{P}{2}} \cup (-A^j)_{j=\frac{P}{2}+1..P}$. When applied to the concatenation of two vectors x^+ and $x^- \in \mathbb{R}^{\frac{P}{2}}$, the multiplication by A can be computed by

$$A^{1..\frac{P}{2}}(x^+ - x^-),$$

where $A^{1..\frac{P}{2}}$ contains the $\frac{P}{2}$ first columns of A. This is exactly the matrix vector multiplication needed in the algorithms solving the BPDN.

Similarly, the multiplication of $d \in \mathbb{R}^N$ by A^t is performed by concatenating

$$(A^{1..\frac{P}{2}})^t d \quad \text{and} \quad -(A^{1..\frac{P}{2}})^t d$$

and only requires one matrix vector multiplication with a matrix of size $\frac{P}{2} \times N$.

As a conclusion, the cost for computing the matrix-vector multiplications with A is essentially the same as the cost for applying the corresponding operations with the matrix $A^{1..\frac{P}{2}}$.

A more serious issue is that the algorithm might converge more slowly, because it needs time to set a coordinate (for instance) x_i^- to 0 although $x_i^+ > 0$. In order to assess the extent of this problem, we evaluated

$$R \overset{\text{def}}{=} 100 \frac{\#\{i \in \{1,\ldots,\frac{P}{2}\}, x_i^+ > 0 \text{ and } x_i^- > 0\}}{\#J}$$

for a symmetric dictionary, throughout the iterative process (# denotes the cardinal of a set). The order of magnitude of the worst value we found is $R \approx 0.1$ and it always decayed rapidly to 0. This suggests that this is not a problem in practice.

However, when this occurs, we also observed that adding the "projection"

$$\forall i \in \{1,\ldots,\frac{P}{2}\}, (x_i^+, x_i^-) \leftarrow \begin{cases} (x_i^+ - x_i^-, 0) & \text{, if } x_i^+ \geq x_i^- \\ (0, x_i^- - x_i^+) & \text{, otherwise,} \end{cases}$$

as a fourth step, in the algorithm of Table 3, slightly improves the convergence. Notice that this “projection” obviously increases f_{c^m,α_m} (the objective function which is maximized). We have no theoretical proof of convergence with this “projection”, but we neither anticipate, nor have experimentally observed, any convergence problem when using this “projection”.

Although it does not seem to be a necessary step, all the experiments conducted in Section 3 use this “projection”.

Notice that the composition of the step 3 in Table 3 and the above “projection” is not a soft thresholding.

2.7.2 The initialization

In the algorithm of Table 3, we need to initialize $x^{0,0}$ and $c^0 \in \mathbb{R}^N$.

We have not studied the initialization of $x^{0,0}$. There are indeed many possibilities for this initialization and we postpone this study to future work. We therefore simply use

$$x^{0,0} = 0.$$

Concerning the initialization of c^0, let us first observe that $(c^m)_{m\in\mathbb{N}}$ converges to the solution c^* of (P). Therefore, c^0 should be close to c^*. Let us approximate c^*, given an estimate $x^{0,0}$ of a solution of (D).

If $x^{0,0}$ is properly initialized, the Kuhn-Tucker conditions for the problem (P) lead to

$$\tau \frac{c^*}{\|c^*\|_2} - b + Ax^{0,0} \approx 0.$$

Moreover, since c^* solves (P) and $\|b\|_2 > \tau$ we have

$$\|A^t c^*\|_\infty = 1.$$

So, we have

$$\begin{cases} c^* \approx \frac{1}{\|A^t c'\|_\infty} c', \\ \text{with } c' = b - Ax^{0,0} \end{cases} \tag{34}$$

Therefore, it seems reasonable to initialize c^0 at the approximate value of c^* given by (34). Of course, the advantage of this initialization is more striking when $x^{0,0}$ is close to an actual solution of (D).

2.7.3 Stopping criteria

Although well designed stopping criteria would improve the algorithm, this is an aspect which we have not studied in details. The stopping criteria used in the experiments are :

- for the loop in k : The loop continues while :

$$\frac{1}{\sqrt{N}} \|A(x^{m,k} - x^{m,k-1})\|_2 > \frac{10}{128} \text{ and } k < 50.$$

In practice, during the first iterations of the loop in m, the stopping criterion which is used is "$k < 50$". After that "$\frac{1}{\sqrt{N}}\|A(x^k - x^{k-1})\|_2 < \frac{10}{128}$" is used and the maximum number of iterations in k rapidly equals 1. The value $\frac{10}{128}$ was set empirically.

Notice in this respect that a better stopping criterion could be deduced from conditions A, A', B or B', in [35], p. 880. It would at least provide better theoretical guarantees of convergence.

- for the loop in m : in order to study the convergence of the algorithm, we simply use the stopping criterion : continue the loop in m while

$$\text{the number of matrix vector multiplications} \leq 6000.$$

  Of course, a better stopping criterion should be used if one wants to avoid useless iterations. This criterion is also motivated by the idea that the stopping criterion for the loop in m depends on the context (time constraints, needed accuracy ...). It should be customized for a specific application. A list of several possible stopping criteria is given in [23].

2.7.4 Setting the sequence $(\alpha_m)_{m\in\mathbb{N}}$

Although we know that the algorithm converges as soon as the sequence $(\alpha_m)_{m\in\mathbb{N}}$ is bounded, it is clear that the behavior of the sequence $(x^m)_{m\in\mathbb{N}}$ depends on $(\alpha_m)_{m\in\mathbb{N}}$. We have observed in many examples (one of them is detailed in Section 3.2) that, when the algorithm is stopped before it has converged and when $(\alpha_m)_{m\in\mathbb{N}}$ is constant : a larger $(\alpha_m)_{m\in\mathbb{N}}$ leads to a larger residual norm and a smaller regularity term. (The converse statement is also true.)

We have tried three strategies, in order to set this sequence.

- PPPA 1: The first one aims at understanding what can be expected from the algorithm, if one knows a reasonable choice for $(\alpha_m)_{m\in\mathbb{N}}$ (for instance by training the algorithm on similar problems). In such a scenario, the user knows an approximation of the best $(\alpha_m)_{m\in\mathbb{N}}$. We mimic this situation by running the algorithm for several sequences

$$\alpha_m \equiv \alpha_0, \qquad \text{for } \alpha_0 = 10^{-3},\ 10^{-2},\ 10^{-1},\ 1,\ 10,\ 10^2,\ 10^3.$$

  and selecting the best value for α_0 according to an external criterion (in practice in our experiments, it is the value of the objective function in (1)). Notice that the values α_0 are crudely sampled. This aims at mimicking the approximate knowledge of the ideal value for α_0. In section 3, the result for this sequence $(\alpha_m)_{m\in\mathbb{N}}$ is referred to as PPPA 1.

- PPPA 2: The second considers a more uncertain scenario where the sequence is automatically adjusted. In this case, we set $\alpha_{min} = 10^{-3}$ and $\alpha_{max} = 10^3$ and we adjust the new value α_m at the end of each iteration in m according to the criterion

$$\begin{aligned} &\text{if } \|Ax^m - b\|_2 < \frac{\tau}{1.1} && \text{, set } \alpha_{m+1} = \min(\alpha_m * 1.001, \alpha_{max}) \\ &\text{if } \|Ax^m - b\|_2 > 1.1\ \tau && \text{, set } \alpha_{m+1} = \max(\frac{\alpha_m}{1.001}, \alpha_{min}) \end{aligned}$$

We also set $\alpha_0 = \alpha_{min} + \frac{\alpha_{max}}{100}$. In section 3, the results for this sequence $(\alpha_m)_{m\in\mathbb{N}}$ is referred to as PPPA 2. Notice that this sequence might be increasing and it does not satisfy the hypotheses leading to the linear convergence rates in Theorem 1. We have not observed any problem with its convergence.

- PPPA 3: The third strategy uses the observations which are made in Section 3.2. At the end of each iteration in m it applies the following rule:

  $$\text{if } \|Ax^m - b\|_2 < \tau, \text{ then } \alpha_{m+1} = 2\alpha_m.$$

  We start from $\alpha_0 = 0.1$. After α_m has been changed, we wait for 25 iterations in m before applying this rule again. The purpose is to guarantee that the change of α_{m+1} already has had an impact on the residual error. In section 3, the results for this sequence $(\alpha_m)_{m\in\mathbb{N}}$ is referred to as PPPA 3. Notice that this sequence $(\alpha_n)_{n\in\mathbb{N}}$ is increasing.

2.7.5 Armijo Rule Along the Projection Arc

We also implemented a version of the algorithm where the gradient based algorithm used to solve Step 1 of Table 1 is the "Armijo Rule Along the Projection Arc" described in [3], Section 2.3.1, p. 230.

In short, the principle of this algorithm (for maximization) is to define at the iteration m and k

$$x(\rho) \overset{\text{def}}{=} \max\left(x^{m,k} + \rho\nabla f_{c^m,\alpha_m}(x^{m,k}), 0\right) \qquad \forall\rho > 0.$$

The algorithm uses the update

$$x^{m,k+1} = x(\rho),$$

where $\rho = \beta^l\rho_0$, for $\beta \in (0,1)$, a fixed $\rho_0 > 0$ and for the first nonnegative integer l such that

$$f_{c^m,\alpha_m}(x(\beta^l\rho_0)) - f_{c^m,\alpha_m}(x^k) \geq \sigma(\nabla f_{c^m,\alpha_m}(x^{m,k}))^t \left(x(\beta^l\rho_0) - x^{m,k}\right),$$

for $\sigma \in (0,1)$.

In the context of our problem, the drawback of this algorithm is that each test of a new value l requires one evaluation of $f_{c^m,\alpha_m}(x(\beta^l\rho_0))$. This evaluation is made using (32) and requires one matrix vector multiplication. The additional cost for these multiplications makes the use of the Armijo rule inefficient. Experimentally, we have not witnessed any situation where this step size rule improves the constant step size rule of the version of the PPPA described in Table 3.

In the following, we do not consider this option further.

3 Experimental results

In Section 3.1, we give all the details on the quantities which are used to assess the quality of the algorithms.

We describe in Section 3.2 some experiments on the convergence of the PPPA. In particular they emphasize the influence of $(\alpha_m)_{m\in\mathbb{N}}$ on the behavior of the algorithm.

Finally, in Section 3.3, we compare the PPPA to the other implementations of the BPDN (the IT, PCD and GPSR-BB).

All the programs and scripts which were used to produce those experiments are available on the web: [28]. Notice that the codes and scripts are adapted to the SPARCO toolbox (see [2]). It is therefore easy to test the PPPA for all the problems included in this toolbox.

Also, since the existing algorithms and problem solve the BPDN (not the NNBPDN) we restrict our experiments to this situation. We therefore always assume that the matrix A corresponds to a symmetric dictionary (i.e, in Matlab notations, $A = [A^{1..\frac{P}{2}}, (-A^{1..\frac{P}{2}})]$). Also, we write $x \in \mathbb{R}^{\frac{P}{2}}$ and, when necessary, implicitly assume that it is extended to $\tilde{x} \in \mathbb{R}_+^P$ according to

$$\begin{array}{lll} \tilde{x}_i = x_i & \text{and } \tilde{x}_{\frac{P}{2}+i} = 0 & \text{if } x_i \geq 0 \\ \tilde{x}_i = 0 & \text{and } \tilde{x}_{\frac{P}{2}+i} = x_i & \text{otherwise.} \end{array}$$

We therefore have

$$A\tilde{x} = A^{1..\frac{P}{2}}x.$$

3.1 Convergence criterion

To assess the quality of a decomposition $x \in \mathbb{R}^{\frac{P}{2}}$ such that $A^{1..\frac{P}{2}}x$ approximates an image $b \in \mathbb{R}^N$, we consider four quantities :

$$l^0 \stackrel{\text{def}}{=} \frac{100}{\frac{P}{2}} \#\{i = 1..\frac{P}{2}, x_i \neq 0\},$$

$$l^1 \stackrel{\text{def}}{=} \frac{1}{\frac{P}{2}} \|x\|_1,$$

and the $RMSE$

$$RMSE \stackrel{\text{def}}{=} \frac{1}{\sqrt{N}} \|A^{1..\frac{P}{2}}x - b\|_2.$$

To be consistent, we also consider the

$$\text{target } RMSE \stackrel{\text{def}}{=} \frac{\tau}{\sqrt{N}}.$$

We also display the value

$$f(x) \stackrel{\text{def}}{=} \|A^{1..\frac{P}{2}}x - b\|_2^2 + \lambda\|x\|_1,$$

when the Lagrange multiplier λ is available. It is indeed the functional which is minimized in the usual BPDN (see (1)).

Since most of the computational time is spent in computing matrix-vector multiplications and most algorithms use two matrix-vector multiplications per iteration, we evaluate time with

$$time \overset{\text{def}}{=} \frac{\#\text{matrix-vector multiplication}}{2}.$$

3.2 Practical convergence of the Proximal Point Algorithm and influence of $(\alpha_m)_{m\in\mathbb{N}}$

As can be seen in the preceding sections, beside the parameters of the problem A, τ and b, the only parameter of the algorithm is $(\alpha_m)_{m\in\mathbb{N}}$ (see (7)).

First, we know that, for any bounded positive sequence $(\alpha_m)_{m\in\mathbb{N}}$, the algorithm converges (see Theorem 1). The questions we would like to answer in this section are : What can we expect as $(x^m)_{m\in\mathbb{N}}$ converges to $\mathcal{S}$ (the set of solutions of (D))? Do we have a way to estimate a good α_{m+1} by observing x^m?

In order to understand these aspects, we run the algorithm for several sequences $\alpha_m \equiv \alpha_0$ and interrupt them before they have reached full convergence. Before describing the details of the experiments, let us summarize our findings.

- The criterion $RMSE$ increases with α_0. The $RMSE$ can even be smaller than the target $RMSE$ (i.e. $\frac{\tau}{\sqrt{N}}$), when α_0 is very small.
- The criteria l^0 and l^1 decay as α_0 increases.
- When α_0 increases, the functional f first decreases and then increases.
- The criterion $RMSE$ converges first. As the number of iterations increases more values α_0 permits to obtain an accurate $RMSE$. Once the $RMSE$ criterion has converged it remains in the vicinity of its target value, the criteria l^0 and l^1 still decay.

Each line of Table 4 and 5 contains the statistics described in Section 3.1 for a given value α_0 (and we recall that, in this experiment, $\alpha_m \equiv \alpha_0$) and for the SPARCO problem number 10 with the target $RMSE = 1$.

Table 4 corresponds to $time = 500$. The algorithms have not converged. The $RMSE$ criterion has reached its target for $\alpha_0 = 1$, 10 and 100. The general behavior described above holds.

Table 5 corresponds $time = 3000$. The $RMSE$ criterion has reached its target for $\alpha_0 = 0.1$, 1, 10, 100 and 1000. The l^1 and l^0 criterion still decay as α_0 increases. The best approximation of the solution is therefore for $\alpha_0 = 1000$. Notice that all the values have improved, compared to Table 4. In particular, for the value of α_0 which reached the $RMSE$ target after 500 iterations, the $RMSE$ is still around the target $RMSE$. The l^1 and l^0 criteria are improved.

α_0	l^0	l^1	$RMSE$
0.001	93.8477	0.50175	0.848469
0.01	88.1836	0.497928	0.8492
0.1	71.1914	0.463789	0.861673
1	47.4609	0.282127	1.00368
10	15.7227	0.231428	1.00074
100	6.73828	0.212524	1.019
1000	2.44141	0.119846	1.31399
10000	0.390625	0.0350802	1.91381
100000	0.195312	0.0293655	1.9707

Table 4: Convergence of the PPPA depending on $(\alpha_m)_{m\in\mathbb{N}}$. The experiment is on the SPARCO Problem number 10 with the target $RMSE = 1$. For every value α_0, the statistics are obtained after 500 iterations (i.e. $time = 500$).

α_0	l^0	l^1	$RMSE$
0.001	88.6719	0.498578	0.849038
0.01	73.8281	0.469742	0.858661
0.1	48.6328	0.301641	1.00005
1	18.5547	0.236177	0.99991
10	7.12891	0.220643	0.99986
100	2.53906	0.21497	0.999306
1000	0.78125	0.194661	1.04896
10000	0.488281	0.108985	1.36281
100000	0.0976562	0.0347324	1.91658

Table 5: Convergence of the PPPA depending on $(\alpha_m)_{m\in\mathbb{N}}$. The experiment is on the SPARCO Problem number 10 with the target $RMSE = 1$. For every value α_0, the statistics are obtained after 3000 iterations (i.e. $time = 3000$).

3.3 Comparison of the algorithms

3.3.1 The local DCT problem

We chose to compare the algorithms on the following problem because it is very difficult to solve and discriminate between the algorithms.

We consider the following dictionary of atoms (i.e. the matrix A): a translation invariant discrete local cosine dictionary. It consists in all the translations of the 64 small images displayed on Figure 1. The small image at the "frequency location" $(\xi, \eta) \in \{0, \dots, 7\}^2$ is

$$\phi^{\xi,\eta}_{m,n} = \frac{1}{\sqrt{C_{\xi,\eta}}} \begin{cases} \cos(\frac{\xi(2m+1)\pi}{16}) \cos(\frac{\eta(2n+1)\pi}{16}) & , \text{if } (m,n) \in \{0,\dots,7\}^2, \\ 0 & , \text{if } (m,n) \notin \{0,\dots,7\}^2, \end{cases}$$

with

$$C_{\xi,\eta} = \sum_{m.n=0}^{7} \left(\cos(\frac{\xi(2m+1)\pi}{16}) \cos(\frac{\eta(2n+1)\pi}{16}) \right)^2 .$$

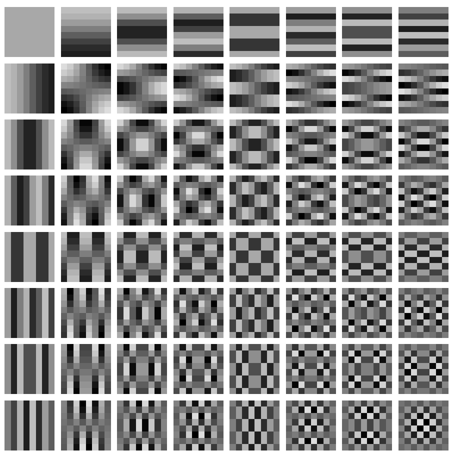

Figure 1: Small images defining the translation invariant discrete local cosine dictionary.

The dictionary is also symmetrized and we finally obtain

$$(\psi_i)_{i\in I} = (\phi_j)_{j\in J} \cup (-\phi_j)_{j\in J}, \tag{35}$$

where

$$(\phi_j)_{j\in J} = \left\{ \tau_{m,n}(\phi^{\xi,\eta}), \text{ for } (\xi,\eta) \in \{0,\dots 7\}^2 \text{ and } (m,n) \in \{0,\dots,\sqrt{N}-1\}^2 \right\},$$

for $\tau_{m,n}$, the translation of an image by the vector (m,n). Although we never actually build the matrix A, its columns would correspond to the elements ψ_i in (35).

Since this dictionary is symmetric, the problem can be solved by any algorithm solving the usual BPDN (as opposed to the NNBPDN). This allows comparisons.

We take b equal to an extracted part of the image Barbara (see Figure 2). (When necessary, the image is periodized outside its original support.) It is of size 128×128. We have $N = 128^2 = 16384$ and[3] $\frac{P}{2} = 64 * N \sim 10^6$

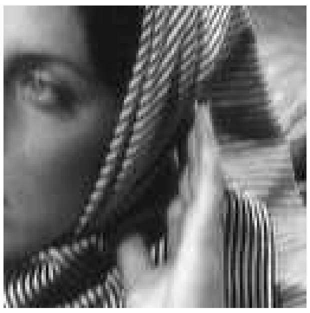

Figure 2: Image extracted from the image Barbara. It is used for the input b in all the experiments.

We make two experiments, one for $\lambda = 0.1$ and one for $\lambda = 100$. They respectively lead to target $RMSE = 0.0235988$ and 9.58555. The first case is, of course, much more difficult than the second.

3.3.2 The algorithm to which we compare the PPPA

We compare PPPA 1, PPPA 2 and PPPA 3 (see Section 2.7.4) to the IT (see [11]), PCD (see [18]) and the GPSR-BB (see [23]). The version of the IT and PCD to which we compare are available at [28], the GPSR-BB algorithm is at [22].

The implementation of the PCD is identical to that described in [18]. For the IT, we needed to improve it in order to apply it to the local DCT problem (for this dictionary, the operator norm of the matrix is larger than 1). However, we observe that the IT can be applied to any matrix A, since, for any $\beta > 0$:

$$x^* \in \text{argmin}_{x\in\mathbb{R}^P} \|Ax - b\|_2^2 + \lambda\|x\|_1 \tag{36}$$

$$\iff \quad \beta x^* \in \text{argmin}_{x\in\mathbb{R}^P} \|(\tfrac{1}{\beta}A)x - b\|_2^2 + \tfrac{\lambda}{\beta}\|x\|_1 \tag{37}$$

So one can solve (37) for β such that $\frac{M}{\beta} < 1$, and multiply the obtained solution by $\frac{1}{\beta}$. This provides a solution to (36)[4].

In practice, the user needs to compute M and chose β such that

$$\frac{M}{\beta} \in (0, 1).$$

[3] $\frac{P}{2}$ is the number of columns in the BPDN (as opposed to NNBPDN).

[4] This trick has independently been observed in [25]

We found experimentally that a larger $\frac{M}{\beta}$ leads to a faster convergence.

We downloaded the GPSR 5.0 toolbox at [22]. In the comparison, we use the Barzilai-Borwein Gradient projection (GPSR-BB), without debiasing. We use the options $'MINITERA'$ and $'MAXITERA'$ to fix the number of iterations.

We have, of course, tested on an easy problem that all these implementations converge to the same solution.

3.3.3 Influence of the parameters and how we fix them

In order to compare the PPPA 1, PPPA 2, PPPA 3, IT, PCD and GPSR-BB algorithms, we need to use them on the same problem. The purpose of our paper is obviously not to answer the question : How to fix λ in the model (4)? So our only choice is to follow the steps:

- Run the IT, PCD, and GPSR-BB algorithms for a given value λ.
- Compute τ (equivalently, the target $RMSE$) : the smallest l^2 norm of the residual amongst those obtained with the IT, PCD and GPSR-BB algorithms.[5]
- Run PPPA for this τ.

This results in an unfair comparison favoring the IT, GPSR-BB and PCD algorithm. Indeed, solving (4) for a fixed λ leads to a residual error (i.e. $\|Ax^* - b\|_2$) which depends on the input b (see Table 6). Therefore, although the required accuracy or noise level is fixed (and often known) the accuracy of the residual error depends on b. To be fair to the PPPA, we should find a strategy enabling the IT, GPSR-BB and PCD to find the correct λ. This would slow them. This cannot be seen on the experiments presented in Section 3.3.4.

clean image	RMSE for $\lambda = 0.1$	noisy image (standard deviation 20)	RMSE for $\lambda = 200$
pepper	0.02512	pepper + noise	17.74
lena	0.02504	lena + noise	18.30
barbara	0.02498	barbara + noise	20.11
baboon	0.02715	baboon + noise	21.41

Table 6: RMSE as a function of the input image b, when (1) is used with a fixed λ (the matrix A corresponds to the local DCT problem of section 3.3.1, we used PCD for computing the result of (4)). Even similar images (interior scenes with a woman + noise) lead to different $RMSE$. The $RMSE$ varies more when λ is large.

[5] In theory they should lead to the same residual norm but we observe a small difference because the convergence is not perfect.

algorithm	l^0	l^1	$RMSE$	f
IT	28.1609	0.284446	0.0294607	29840.5
PCD	29.9812	0.264319	0.026786	27727.6
GPSR-BB	20.6212	0.272771	0.0283202	28615.2
PPPA 1	3.27272	0.222995	0.200705	24042.7
PPPA 2	3.3596	0.223517	0.0999004	23601
PPPA 3	8.49972	0.246459	0.0243556	25852.8

Table 7: Experiment for the local DCT problem, $\lambda = 0.1$, target $RMSE = 0.0235988$, $time = 500$. In terms of f (which is the best criteria since the algorithms are far from convergence), all the PPPA produce a better solution.

algorithm	l^0	l^1	$RMSE$	f
IT	11.4758	0.254577	0.0254477	26704.9
PCD	6.95639	0.239962	0.0235988	25170.9
GPSR-BB	8.25691	0.245702	0.0239035	25773
PPPA 1	1.66874	0.216994	0.0521594	22798
PPPA 2	2.58303	0.219227	0.0270889	22999.7
PPPA 3	3.76682	0.226575	0.0237763	23767.4

Table 8: Experiment for the local DCT problem, $\lambda = 0.1$, target $RMSE = 0.0235988$, $time = 3000$. All the PPPA produce a better solution. In particular, the PPPA solution is much more sparse than the IT, PCD and GPSR-BB solutions. The values of α_{3000} are respectively 100, 2.0990, 1.6, for PPPA 1, PPPA 2 and PPPA 3.

3.3.4 Result of the comparison

We give in Tables 7 and 8, the convergence criteria for the results of the different algorithms, for the local DCT problem with $\lambda = 0.1$ at $time = 500$ and 3000. In is clear from these figures that all the versions of the PPPA perform better than the other algorithms. Concerning PPPA 1, notice that in Tables 7, 8 and 9, the value α_0 leading to the best value for f is not one of those for which the RMSE criterion has converged.

The same experiment for $\lambda = 100$ is less favorable to the PPPA. In particular PPPA 2 and PPPA 3 are less accurate than the other algorithms at $time = 500$. However, at $time = 3000$, PPPA 1 provides the best solution. In particular its result is more sparse. PPPA 3, also gives very good results, if one is not interested in a very accurate $RMSE$. Notice that, in this simpler experiment, all the algorithms give similar results at $time = 3000$.

algorithm	l^0	l^1	$RMSE$	f
IT	1.08442	0.176016	9.7777	$2.0023e+07$
PCD	0.973034	0.175485	9.62048	$1.99173e+07$
GPSR-BB	0.916862	0.17561	9.67858	$1.99488e+07$
PPPA 1	0.740337	0.173939	10.1404	$1.99236e+07$
PPPA 2	1.5254	0.180304	9.58529	$2.04116e+07$
PPPA 3	0.598907	0.175847	10.1887	$2.01397e+07$

Table 9: Experiment for the local DCT problem, $\lambda = 100$, target $RMSE = 9.58555$, $time = 500$.

algorithm	l^0	l^1	$RMSE$	f
IT	0.760555	0.175435	9.60997	$1.99087e+07$
PCD	0.61779	0.175344	9.58555	$1.98915e+07$
GPSR-BB	0.674343	0.175362	9.5923	$1.98956e+07$
PPPA 1	0.59557	0.17534	9.58555	$1.98911e+07$
PPPA 2	1.04094	0.176379	9.58554	$2.00001e+07$
PPPA 3	0.540924	0.17526	9.64634	$1.99019e+07$

Table 10: Experiment for the local DCT problem, $\lambda = 100$, target $RMSE = 9.58555$, $time = 3000$. The values of α_{3000} are respectively 1000, 9.98, 1.04, for PPPA 1, PPPA 2 and PPPA 3.

4 Perspectives

Several aspects of the algorithm and its convergence analysis could be improved. We give a non-exhaustive list below.

- One obvious improvement of the PPPA is to find a better gradient based algorithm for solving Step 1 in Table 1. The first tests we made did not lead to much improvements. A more systematic study/implementation of these algorithms would be interesting. Before studying this aspect, one should realize that this would only affect few iterations in m, since afterward the loop in k is very well initialized and the algorithm does not need many iterations in k.

- The main drawback of the current algorithm is that it requires the user to select $(\alpha_m)_{m\in\mathbb{N}}$. The algorithm seems to be relatively stable with regard to the sequence $(\alpha_m)_{m\in\mathbb{N}}$ but it is possible to select this sequence badly. It would be interesting to better develop a method selecting $(\alpha_m)_{m\in\mathbb{N}}$ in an automatic way.

  Another possibility might be to tune another hidden parameter of the algorithm, if this parameter has more meaning.

  In particular, it is possible to multiply the objective function in (P), by a constant factor β. The dual of this problem is similar to (D). This parameter is very much similar to the (hidden) parameter β used for the IT, in (36) and (37).

  Another parameter is hidden in $\mathbf{1}_P$ in (P). We can also multiply $\mathbf{1}_P$ by a factor β'. The dual of the new problem (P) is still (D). However, such a β' would appear and have an impact on the final algorithm.

- The convergence rate of the loop in k could perhaps be improved. In particular, the convergence rate stated in Section 2.6 concerns the value of the function $(f_{c,\alpha}(x^{m,k}))_{k\in\mathbb{N}}$ and not $(x^{m,k})_{k\in\mathbb{N}}$. It would be interesting to obtain a convergence rate which provides an upper bound for one of the quantities which are known not to perturb the convergence of the PPA. (See the criteria A,A',B,B' in the stability analysis of the PPA, in [35]).

- In Theorem 1, we have not been able to obtain convergence rates for the sequence $(x^m)_{m\in\mathbb{N}}$. We have not even been able to establish that it converges to a single point. Although we know that $(c^m)_{m\in\mathbb{N}}$ converges linearly, $\text{argmax}_{x\in\mathbb{R}_+^P} f_{c^m,\alpha_m}(x)$ might be a set and the element we pick in this set might vary a lot. This is due to the lack of uniqueness for the problem (D) and (possibly) for the maximization of $f_{c,\alpha}$. It might be possible to find a reasonable set of hypotheses for which the maximizer of $f_{c^m,\alpha_m}(x)$ is unique. If not, it seems possible to obtain convergence rates for $(x^m)_{m\in\mathbb{N}}$ by using both

  - the fact that we initialize the gradient based algorithm for solving Step 1 in Table 1 at the previous value x^m;

  - some techniques used to prove the stability of the maximizer of a function with regard to one of its parameters. (The parameter c in $f_{c,\alpha}$ does indeed converge.)

  Such a result could also be incorporated in the convergence rate of the loop in k. This loop benefits, in practice, a lot from the fact that it is initialized at the previous x^m and our analysis does not incorporate this information.

- Another interesting question is to determine the set of problems for which the "Predual Proximal Point Algorithm" (as described in this paper) can be successfully applied.

  In particular, it does not seem difficult to modify the data fidelity term in (D). The main modification seems to concern the problem (P) and the function e_β. Several other Moreau envelopes are known and can be used in place of e_β. This determines a class of problems (P) (and therefore of problems (D)) to which the PPPA can be applied. A good starting point for such a project seems to be [9], since it exploits similar ideas.

  This perspective is interesting since modifying the data fidelity term gives a chance to improve the performance of the $l^1 - l^2$ model, while keeping to l^1. It might also permit to apply the l^1 regularization to other problems (such as the restoration of compressed images).

- In order to answer a question of a referee, we mention that we do not know if the convergence rate of $(c^m)_{m\in\mathbb{N}}$ and $(Ax^m)_{m\in\mathbb{N}}$ can be improved or not. It could very well be that the particular structure of our function g permits to obtain a better convergence rate.

References

[1] J. Bect, L. Blanc-Féraud, G. Aubert, and A. Chambolle. A l1-unified variational framework for image restoration. *Lecture notes in Computer Science*, 2004. Proc. ECCV 2004.

[2] E. van den Berg, M. P. Friedlander, G. Hennenfent, F. Herrmann, R. Saab, and Ö. Yılmaz. Sparco: A testing framework for sparse reconstruction. Technical Report TR-2007-20, Dept. Computer Science, University of British Columbia, Vancouver, October 2007.

[3] D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, second edition edition, 2003.

[4] J.M. Bioucas-Dias. Bayesian wavelet-based image deconvolution: A gem algorithm exploiting a class of heavy-tailed priors. *IEEE, trans. on image processing*, 15(4):937–951, 2006.

[5] M. Brown and N.P. Costen. Exploratory basis pursuit classification. *Pattern Recognition Letters*, 26:1907–1915, 2005.

[6] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles : Exact signal reconstruction from highly incomplete frequency information. *IEEE, Trans. on Information Theory*, 52(2):489–509, Feb. 2006.

[7] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

[8] P. Ciarlet. *Introduction to numerical linear algebra and optimisation.* Cambridge University Press, 1989.

[9] P. L. Combettes and J.-C. Pesquet. Proximal thresholding algorithm for minimization over orthonormal bases. *SIAM Journal on Optimization*, 18(4):1351–1376, November 2007.

[10] P.L. Combettes and V.R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM, Journal on Multiscale Modeling and Simulation*, 4(4):1168–1200, November 2005.

[11] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problem with sparsity constraint. *Communication on Pure and Applied Mathematics*, 57(11):1413–1457, Aug. 2004.

[12] D. Donoho. Neighborly polytopes and sparse solution of underdetermined linear equations. Technical Report 2005-04, Dept of Statistics, Stanford University, January 2005.

[13] D. Donoho. Compressed sensing. *IEEE, Trans. on Information Theory*, 52(4):1289–1306, April 2006.

[14] D. Donoho, M. Elad, and V. Temlyakov. Stable recovery of sparse overcomplete representation in the presence of noise. *IEEE, Trans. on Information Theory*, 52:6–18, 2006.

[15] D. Donoho and J. Tanner. Sparse nonegative solution of underdetermined linear equations by linear programming. *Proceedings of the National Academy of Sciences*, 102(27):94446–9451, 2005.

[16] D. Donoho and Y. Tsaig. Fast solution of l1 - norm minimization problems when the solution may be sparse. Technical Report 2006-18, Stanford, dept of statistics, Oct. 2006.

[17] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32(2):407–499, 2004.

[18] M. Elad. Why simple shrinkage is still relevant for redundant transforms. *IEEE, Trans. on Information theory*, 52(12):5559–5569, Dec. 2006.

[19] M. Elad, B. Matalon, and M. Zibulevsky. Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization. *Journal on Applied and Computational Harmonic Analysis*, 23:346–367, Nov. 2007.

[20] M. Figueiredo and R. Nowak. An em algorithm for wavelet-based image restoration. *IEEE, transactionson image processing*, 12(8):906–916, August 2003.

[21] M. Figueiredo and R. Nowak. A bound optimization approach to wavelet-based image deconvolution. In *ICIP 2005*, volume 2, pages 782–785, 2005.

[22] M. Figueiredo, R. Nowak, and S. Wright. Gpsr 5.0. Matlab toolbox, available at http://www.lx.it.pt/ mtf/GPSR/, Dec. 2007.

[23] M. Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–598, December 2007.

[24] O. Güler. On the convergence of the proximal point algorithm for convex minimization. *SIAM, J. Control and optimization*, 29(2):403–419, 1991.

[25] E.T. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing. CAAM TR07-07, Rice University, August 2007.

[26] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. A method for large-scale l1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing*, 1(4):606–617, December 2007.

[27] C. Lemarechal and C. Sagastizabal. Practial aspects of the moreau-yoshida regularization 1 : theoretical properties. *SIAM, J. optimization*, 7:867–895, 1997.

[28] F. Malgouyres. Codes and scripts on basis pursuit denoising. http://www.math.univ-paris13.fr/∼malgouy/software/index.html.

[29] F. Malgouyres. Projecting onto a polytope simplifies data distributions. Technical Report 2006-1, University Paris 13, January 2006.

[30] F. Malgouyres. Rank related properties for basis pursuit and total variation regularization. *Signal Processing*, 87(11):2695–2707, Nov. 2007.

[31] S. Maria and J.J. Fuchs. Application of the global matched filter to stap data: an efficient algorithmic approach. In *Proceedings of ICASSP 2006*, volume 4, pages 1013–1016, Toulouse, France, May 2006.

[32] Y. Nesterov. *Introductory lectures on convex optimization : A basic course.* Kluwer Academic Publishers, 2004.

[33] Compressed Sensing ressources. http://www.dsp.ece.rice.edu/cs/.

[34] R.T. Rockafellar. *Convex analysis.* Princeton University Press, 1970.

[35] R.T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM, J. Control and optimization*, 14(5):877–898, 1976.

[36] S. Sardy, A.G. Bruce, and P. Tseng. Block coordinate relaxation methods for nonparametric wavelet denoising. *Journal of Computational and Graphical Statistics, Vol. No. 2 (Jun., 2000), pp.*, 9(2):361–379, June 2000.

[37] J-L. Starck, M. Elad, and D.L. Donoho. Image decomposition via the combination of sparse representations and a variational approach. *IEEE, Trans. on Image Processing*, 14(10):1570–1582, October 2005.

[38] T. Zeng. *Études de modèles variationnels et apprentissage de dictionnaires.* PhD thesis, Université Paris 13, 2007.