CompAMa
www.compama.co.usb.ve

Bull. Comput. Appl. Math., Vol.1, No.2, 2013
ISSN 2244-8659

Some Convergence Strategies for the Alternating Generalized Projection Method

M. Andrade [*], *R. Escalante* [†‡], *R. Espitia* [§]

CompAMa Vol.1, No.2, pp.47-77, 2013 - Accepted November 25, 2013

Abstract

In this paper we extend the application of the alternating projection algorithm to solve the problem of finding a point in the intersection of n sets ($n \geq 2$), which are not all of them convex sets. Here we term such method as alternating generalized projection (AGP) method. In particular, we are interested in addressing the problem of avoiding the so-called trap points, which may prevent an algorithm to obtain a feasible solution in two or more sets not all convex. Some strategies that allow us to reach the feasible solution are established and conjectured. Finally, we present simple numerical results that illustrate the efficiency of the iterative methods considered.

Keywords: Alternating generalized projection method; Method of generalized projections; Method of alternating projection; Error sums of distances; Product vector space; Feasible solution; Trap points; Intersection of sets.

[*]Escuela de Matemática, Facultad de Ciencias, Universidad Central de Venezuela, Caracas 1040, Venezuela (maricarmen.andrade@ciens.ucv.ve).

[†]Departamento de Cómputo Científico y Estadística, División de Ciencias Físicas y Matemáticas, Universidad Simón Bolívar, Ap. 89000, Caracas, 1080-A, Venezuela (rescalante@usb.ve).

[‡]This author was partially supported by the Decanato de Investigación y Desarrollo (DID) at USB.

[§]Escuela de Matemática, Facultad de Ciencias, Universidad Central de Venezuela, Caracas 1040, Venezuela (robert.espitia@ciens.ucv.ve).

1 Introduction and Preliminaries

Due to the large number of applications in science and engineering, algorithms for solving convex feasibility problems have received great attention. In particular, the *method of alternating projections* (MAP) has been widely studied from the numerical point of view [1]. However, this method can not guarantee convergence to a feasible solution if any of the sets is nonconvex [2]. Convergence to a feasible solution strongly depends on the starting point.

The important thing to note is that, in general, nonconvex sets may prevent the convergence of traditional methods of alternating projections. Indeed, this problem cannot generally be solved by the *method of projections onto convex sets* (POCS) [3]. A number of important practical problems involving the use of nonconvex constraints can be found, as occurs in certain types of neural nets, in analog and digital filter design, in digital communications, in designing structured tight frames, and pattern recognition [3], [4]. This led to consider a restricted form of the MAP called *generalized projection* [3], which here we prefer to call the *alternating generalized projection* (AGP) algorithm.

As a result of the direct application of the AGP algorithm, it is possible to generate iterates with the property that the summed distance error (SDE) from the involved sets is expected to decrease in each iteration (see also [5]). Thus SDE convergence means that the summed distance as defined in Theorem 1.1 and Algorithm 1.1 always decreases (or at least never increases). In practice, it is possible to detect starting points for which the algorithm converges to an infeasible fixed point called a *trap*, which is an undesirable convergence point, since the property of reducing SDE fails. This problem has not been sufficiently studied and the specialized literature on it is scarce.

The MAP dates back to von Neumann [6] who treated the problem of finding the projection of a given point in a Hilbert space onto the intersection of two closed subspaces. Later, Cheney and Goldstein [7] extended the analysis of von Neumann's alternating projection scheme to the case of two closed and convex sets. In particular, they established convergence under mild assumptions (the limit point need not be the closest in the intersection). When dealing with nonconvex sets, it is generally not possible to achieve strong or weak convergence. However, under certain conditions, an AGP algorithm can produce iterates, whose SDE decreases with the iteration number [3].

In [8], Combettes and Trussell extend the method of successive projections

to collections of approximately compact sets in metric spaces, establishing conditions for the convergence of a sequence of successive projections to a feasible point. However, this study requires defining a region called region of attraction, which depends on the projection operator onto the intersection of the involved sets and the geometry of the sets, which seems to be difficult to establish. In addition, the definition of a point of attraction is not constructive and such point may be difficult to find. So, one needs to know *a priori* the intersection set, in order to deduce which points can be points of attraction. Although the approach suggested in [8] is of unpractical interest, we think that it is one of the few attempts, as far as we know, to describe a method of alternating projections onto nonconvex sets.

Another interesting paper is that of Chrétien and Bondon [9], which analyzes projection methods for sets that admit a convex expansion, i.e. if it is the union of convex sets. It is also a theoretical approach for which the concerned geometry is somewhat difficult to establish.

Recently, Tam [10] examined, using an interactive geometry software named *Cinderella* [11], some of the difficulties encountered when dropping the assumption of convexity. Also recently Bauschke *et al.* [12] proposed an underrelaxed strategy. More specifically, they analyzed the Method of Alternating Relaxed Projections (MARP) for two sets not necessarily convex, establishing local linear convergence results for the MARP.

In our paper some strategies that try to avoid the trap points are proposed. They consist essentially of the proper application of relaxed projections, the use of the polar cone decomposition method of Moreau, and the use of separating hyperplanes. In this point it is necessary to warn the reader that theoretical work is needed to support the algorithms proposed in this paper.

The notation used here is, with slight modifications, that of [3], however some basic notations, assumptions, and definitions are given. If C is a set in a Hilbert space H and $x \in H$, then $d(x, C) = \inf_{y \in C} \|x - y\|$ is the *distance of a point x to C*. Let $C_1, C_2, \ldots, C_m$ be m sets in H, then the *summed distance error* (SDE) of a point x from the sets $C_1, C_2, \ldots, C_m$ is denoted by $J(x)$, i.e. $J(x) = \sum_{i=1}^{m} d(x, C_i)$. A point x is a trap point if it is a local minimum of $J(.)$ and $J(x) > 0$. If C is a closed set and there exists a point $x^* \in C$, which is not necessarily unique, such that the distance from x to C is achieved, then x^* is called the *generalized projection* of x onto C, i.e. $x^* = P_C x$, where P_C is the *generalized projector* onto C.

We are primarily concerned with the following algorithm:

$$x_{n+1} = T_{1,n}T_{2,n}\dots T_{m,n}x_n, \quad n = 0, 1, 2, \dots,$$

where

$$T_{i,n} = I + \lambda_{i,n}(P_i - I), \quad i = 1, 2, \dots, m,$$

with $P_i \equiv P_{C_i}$, and $\lambda_{i,n}$ is a relaxation parameter that can be adjusted from iteration to iteration. If $C_0 \equiv \cap_{i=1}^{m} C_i \neq \emptyset$ and sets C_i's, for $i = 1, \dots, m$, are closed convex sets, then this algorithm is the POCS method. A trap is a fixed point of the composition operator $T_{1,n}T_{2,n}\dots T_{m,n}$, which is not a fixed point of every individual $T_{i,n}$. For $m = 2$, it is preferred to write $x_{n+1} = T_1T_2x_n$, with x_0 arbitrary and $T_i = I + \lambda_i(P_i - I)$ $(i = 1, 2)$. Since here at least one nonconvex set is involved, we cannot use the concepts of weak or strong convergence to describe the algorithm's performance. Instead, we will consider the SDE concept (see for instance [3], [5]). In this case, $J(x_n) = \|P_1x_n - x_n\| + \|P_2x_n - x_n\|$. The following theorem, due to Levi [13], describes the SDE *reduction property* of the recursive algorithm.

Theorem 1.1 (Levi, 1983)**.** *The given algorithm, for $m = 2$, has the property*

$$J(x_{n+1}) \leq J(T_2x_n) \leq J(x_n) \tag{1}$$

for every λ_1 and λ_2 such that $0 \leq \lambda_i \leq \frac{A_i^2 + A_i}{A_i^2 + A_i - \frac{1}{2}(A_i + B_i)}$, $i = 1, 2$, where $A_1 = \frac{\|P_1T_2x_n - T_2x_n\|}{\|P_2T_2x_n - T_2x_n\|}$, $A_2 = \frac{\|P_2x_n - x_n\|}{\|P_1x_n - x_n\|}$, and $B_1 = \frac{\langle P_2T_2x_n - T_2x_n, P_1T_2x_n - T_2x_n\rangle}{\|P_2T_2x_n - T_2x_n\|^2}$, $B_2 = \frac{\langle P_1x_n - x_n, P_2x_n - x_n\rangle}{\|P_1x_n - x_n\|^2}$.

In particular, the algorithm defined by $x_{n+1} = P_1P_2x_n$ (x_0 arbitrary) will have the property of SDE reduction [3]. However, that does not make this algorithm optimal from convergence speed and performance point of view. Property defined in Theorem 1.1 does not imply a decreasing of $J(x_n)$ but a non-increasing sequence. This property does not imply convergence of the sequence $\{x_i\}$ to a feasible point, even to a trap point. In practice, the SDE reduction could be accomplished for a more extended range of λ_i than that given in the theorem. In practical problems we seek a solution in the intersection of many nonconvex sets. Although the SDE reduction property can not be extended to more than two sets, the *generalized projections in a product space* can be considered [3], which was possible by reformulating the problem in a product space, as it was proposed by Pierra in [14]. Theorem 1.1

Algorithm 1.1 Generalized projections algorithm (Levi, 1983)

Require: $x_0 \in H$, $\lambda_1, \lambda_2 \in (0,2)$ (which are chosen at random at each iteration), $\epsilon = tolerance$

Output x_{n+1}
for $n = 0, 1, 2, \ldots$ **do**
  for $i = 1, 2$ **do**
    $T_i = I + \lambda_i(P_i - I)$
  end for
  $A_1 \equiv \dfrac{\|P_1T_2x_n - T_2x_n\|}{\|P_2T_2x_n - T_2x_n\|}$
  $A_2 \equiv \dfrac{\|P_2x_n - x_n\|}{\|P_1x_n - x_n\|}$
  $B_1 \equiv \dfrac{\langle P_2T_2x_n - T_2x_n, P_1T_2x_n - T_2x_n\rangle}{\|P_2T_2x_n - T_2x_n\|^2}$
  $B_2 \equiv \dfrac{\langle P_1x_n - x_n, P_2x_n - x_n\rangle}{\|P_1x_n - x_n\|^2}$
  if $\lambda_i \leq \dfrac{A_i^2 + A_i}{A_i^2 + A_i - \frac{1}{2}(A_i + B_i)}$ **then**
    $x_{n+1} = T_1T_2x_n$
  else
    $\lambda_1, \lambda_2 \in (0,2)$ are chosen at random
    continue
  end if
  $J(x_n) \equiv \|P_1x_n - x_n\| + \|P_2x_n - x_n\|$
  Convergence test: If $J(x_{n+1}) \leq J(T_2x_n) \leq J(x_n) < \epsilon$, stop
end for

leads to the Algorithm 1.1. In Section 3 POCS is compared with algorithms endowed with the SDE reduction property.

We now recall some additional concepts. A nonempty set K in a vector space is called a *cone* if $x \in K$ implies that $\alpha x \in K$ for all $\alpha \geq 0$. If, in addition, K is convex, then K is called a *convex cone*. Now, let K be a cone in a Hilbert space H, then the *polar cone* of K, denoted by K°, is given by $K^\circ = \{x \in H : \langle x, y\rangle \leq 0 \ \forall y \in K\}$. Notice that K° is a closed convex cone and that if K is a closed convex cone then $(K^\circ)^\circ = K$. Besides, the decomposition result with respect to polar cones known as Moreau's Theorem

[15] will be needed.

Theorem 1.2 (Moreau, 1962)**.** *Let $x, y, z \in H$ and $K \subseteq H$. Moreover, if K is a closed convex cone, then* $[z = x + y$, *with* $x \in K$, $y \in K^{\circ}$, *and* $\langle x, y \rangle = 0]$ *if and only if* $[x = P_K z$ *and* $y = P_{K^{\circ}} z]$.

The remainder of the paper is organized as follows. In the next section, some iterative schemes and strategies to be applied to the nonconvex problem are proposed. In Section 3, preliminary numerical experimentation is presented to illustrate the advantages of the proposed strategies. Finally, in Section 4, some concluding remarks are presented.

2 Strategies of convergence

In order to avoid the pitfalls when applying the AGP method we propose to use strategies, not previously considered in the literature known by us, which involve an adaptation of the POCS algorithm and the decomposition method in polar cones, as established by Moreau for convex sets [15]. To avoid the convergence to trap points, another strategy will be needed, which will consider the use of separating hyperplanes. Thus, instead of projecting onto the sets directly, it is projected successively and cyclically onto one of the sets and the separating hyperplane to achieve convergence to a point in the intersection of the involved sets.

2.1 Decomposition method in polar cones

In this section, a strategy based on the orthogonal decomposition with respect to polar cones proved by Moreau (Theorem 1.2) will be proposed. Assuming that during the iterative process a trap point is reached, the intention is basically to try to leave the trapping area, and involve points and projections which were different to those used previously in the area of entrapment.

If one of the sets involved in the AGP method is a convex cone (which we will call *primitive cone*), the basic idea of the algorithm consists in using the Moreau's theorem to project onto the polar cone obtained from the convex set, to then projecting onto the nonconvex set. Next, we use the Moreau's theorem again to recover the projection onto the primitive cone. Thus, we expect that the algorithm iterates eventually leave the region where the trap point is located and may converge to a correct solution.

As far as we know the Moreau's theorem was never used as part of a strategy to exit a situation of entrapment. In numerical experimentation satisfactory and promising results were obtained. From the application of this strategy we propose Algorithm 2.1, where we assume that C_2 is the convex set and C_1 is the nonconvex set. In our numerical experiments, the relaxed T_i projections have been used (instead of P_{C_i} of Algorithm 2.1), which adds an element of randomness that we hope will help to prevent the succession of iterated falls into a situation of entrapment.

Algorithm 2.1 AGP algorithm using the decomposition method in polar cones

Require: $x \in H$, $\epsilon = tolerance$ $x = x_0$
1: **Output** x_{n+1}
2: **for** $n = 0, 1, 2, \ldots$ **do**
3: $x_{n+1} = P_{C_1} P_{C_2} x_n$
4: $J(x_n) = \|P_{C_2} x_n - x_n\| + \|P_{C_1} x_n - x_n\|$
5: **if** $J(x_n) < \epsilon$ **then**
6: the algorithm converges to $x_n \in (C_1 \cap C_2)$
7: **return** x_n
8: **end if**
9: **if** $|J(x_{n+1}) - J(x_n)| < \epsilon$, **then**
10: the algorithm converges to a trap point
11: $x_T = x_{n+1}$
12: $P_{C_2^o} x_T = x_T - P_{C_2} x_T$
13: $z = P_{C_1} P_{C_2^o} x_T$
14: $x_{n+1} = P_{C_1} P_{(C_2^o)^o} z = P_{C_1} P_{C_2} z$
15: **end if**
16: **end for**

2.2 AGP method using separating hyperplanes

It has been proved [16] that for P_1 linear and $P_1 T_2 x_n = x_n$, a solution x lies in a hyperplane orthogonal to the vector $P_2 x_n - x_n$. This fact tells us that when we are in a trap (or when $x_{n+1} \approx x_n$, and x_n is not a fixed point of both T_1 and T_2) we have to look for a solution along a direction orthogonal to the vector $P_2 x_n - x_n$. Although P_1 is not linear one can take this result as approximately true [3]. We will use this idea as inspiration to try to improve the algorithm, and thus avoid the traps.

In the case that one of the sets involved in the process of applying the AGP method is a convex set, but not a cone, and if the algorithm converges

to a trap point, a scheme involving the use of a succession of separating hyperplanes will be proposed. In each iteration a separating hyperplane is defined such that it separates the point to be projected onto the convex set and the same set. The reason for this is to apply the AGP method onto the nonconvex set and the separating hyperplanes, which are convex cones. Therefore, in this case, when the method seems to converge to a trap point, we can also apply the Moreau's theorem on the hyperplanes. For simplicity, we temporarily restrict ourselves to the particular case in which $H = \mathbb{R}^n$.

More specifically, we consider two nonempty closed sets $S_1, S_2 \in \mathbb{R}^n$ such that $S = S_1 \cap S_2 \neq \emptyset$. We also assume that S_1 is a nonconvex set and S_2 is convex, but not a cone (if it were a cone, we would apply Algorithm 2.1). Furthermore, let $x_0 \in \mathbb{R}^n$ be the starting point of the proposed algorithm. Our goal is to find a point x in the intersection of S_1 and S_2. It is a known result that if $y \notin S_2$ is a given point (initially $y = x_0$), then there exist a hyperplane that separates y and S_2. We proceed as follows; first, we project the point x_0 onto the convex set S_2 to get $P_{S_2}x_0$. Next, we select a point w in the line segment joining the points x_0 and $P_{S_2}x_0$, and define a hyperplane H_0 containing the point w, whose normal vector is the gradient vector of S_2 in the point $P_{S_2}x_0$. This hyperplane separates x_0 and S_2. We also expect that it is such that its intersection with S_1 is a nonempty set, i.e, $\mathrm{H}_0 \cap S_1 \neq \emptyset$ (see Figure 1(a)).

In case that a point w such that $\mathrm{H}_0 \cap S_1 \neq \emptyset$ can not be found, $P_{S_2}x_0$ is then projected onto the nonconvex set S_1, and continuing with this process the algorithm projects alternatively onto S_2 and S_1, until that it can be verified that there exists a point w such that the intersection of the corresponding separating hyperplane and S_1 is a nonempty set (see Figure 1(b)). Clearly, for this strategy to be successful, it is expected that there exists a point w. In our numerical tests the point was always found.

Let $\mathrm{H}_0 = \{z \in \mathbb{R}^n : \langle p, z\rangle = \alpha\}$ be the first separating hyperplane in the process just described, where p is the gradient vector of S_2 in the point $P_{S_2}x_0$ (it is assumed that S_2 is differentiable there) and $\alpha = \langle p, w\rangle$.

The projections are made alternatively onto the separating hyperplane H_0 and S_1 to obtain a sequence of elements that converges to a point $(x_0)_n \in S_1 \cap \mathrm{H}_0$. The existence of $(x_0)_n$ is guaranteed as long as the point w exists so that $S_1 \cap \mathrm{H}_0 \neq \emptyset$ and it is identified by the algorithm (see Figure 2(a)).

If the point $x_1 = (x_0)_n \notin S_2$ other hyperplane H_1 that separates x_1 and S_2 can be defined such that $S_1 \cap \mathrm{H}_1 \neq \emptyset$ (see Figure 2(b)), and again projections are made alternatively onto S_1 and H_1 to generate a sequence of elements

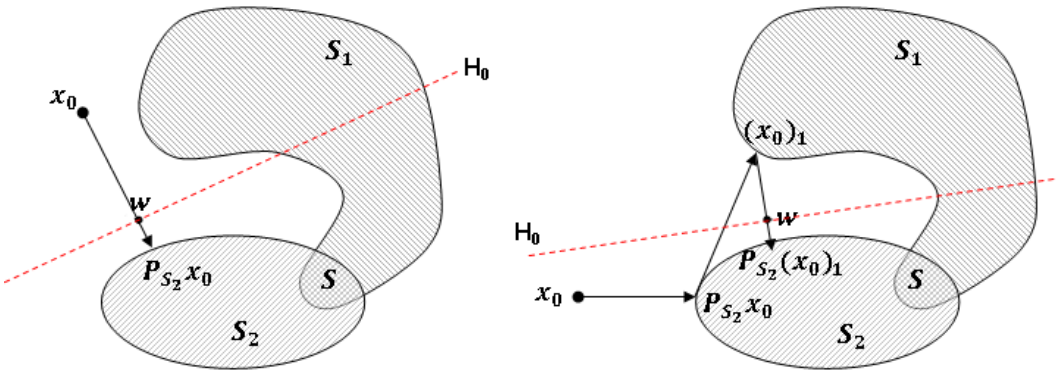


Fig. 1: (a) There is a point w in the line segment with endpoints x_0 and $P_{S_2}(x_0)$, which belongs to a hyperplane H_0 such that $\mathrm{H}_0 \cap S_1 \neq \emptyset$ is defined. (b) There is not a point w in the line segment with endpoints x_0 and $P_{S_2}(x_0)$ that belongs to a hyperplane H_0 such that $\mathrm{H}_0 \cap S_1 \neq \emptyset$. However, there may exist a point w in a line segment corresponding to a subsequent iterated.

that converges to a point $x_2 = (x_1)_n \in S_1 \cap \mathrm{H}_1$. Continuing with this process we expect that the algorithm eventually generates a sequence of elements that converges to a point in S. From which we get the following result.

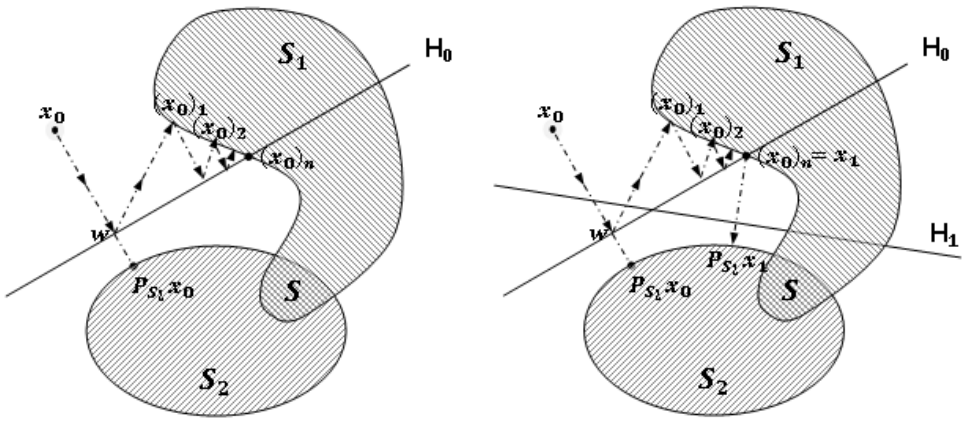
Theorem 2.1. *There exists a sequence of points* $\{x_m\}_{m\geq 0}$ *and a sequence of hyperplanes* $\{H_m\}_{m\geq 0}$ *such that, for each* m, $x_m \in S_1 \cap H_m$.

However, if the above sequence converges to a trap point the process is stopped. To exit the trap, a strategy previously considered could be applied temporarily, as would be the use of projections with appropriate relaxed parameters or the application of the Moreau's Theorem, and then continue with the process of defining new separating hyperplanes. So, a sequence of iterates $\{x_m\}_{m\geq 0} \subset S_1$, that we expect converges to a point in S, will be obtained. Furthermore, at the end of the process, a collection of separating hyperplanes H_m will have been defined.

These results are summarized in Algorithm 2.2.

The iterations in Algorithm 2.2 were stopped when, for $i \geq 0$,

$$k((x_i)_n) = \|P_{\mathrm{H}_i}(x_i)_n - (x_i)_n\| < tolerance,$$



(a) (b)

Fig. 2: (a) Alternating projections onto H_0 and S_1. (b) The hyperplane H_1 separates x_1 and S_2.

for different values of *tolerance*. As a second stop criterion, and using the same values of *tolerance*, the SDE reduction property associated was used, to ensure the convergence to an element in S.

2.2.1 Case $H_i \cap S_2 \neq \emptyset$

In this case a hyperplane which separates S_1 (a nonconvex set) of the current iterate, in each iteration, is considered. We have again two nonempty closed sets $S_1, S_2 \in \mathbb{R}^n$ such that $S = S_1 \cap S_2 \neq \emptyset$. We also assume that S_1 is a nonconvex set and S_2 is convex, but not a cone (if it were a cone, we would apply Algorithm 2.1). Furthermore, let $x_0 \in \mathbb{R}^n$ be the starting point of the proposed algorithm, and let P_{S_i} be a projection operator onto S_i $(i = 1, 2)$. Our goal is again to find a point x in the intersection of S_1 and S_2.

In this case, it is defined, in each iteration, a separating hyperplane between the projection of the current iterate onto S_2 and a projection of this projected point onto S_1. So, initially (and in successive iterations), we consider a separating hyperplane H_0 between $P_{S_2}x_0$ and $P_{S_1}P_{S_2}x_0$ such that $H_0 \cap S_2 \neq \emptyset$ (see Figure 3(a)). We proceed as follows; first, we project the point x_0 onto the convex set S_2 to get $P_{S_2}x_0$; next this point is projected onto the nonconvex set S_1 to obtain $P_{S_1}P_{S_2}x_0$. Then, we select a point w in the line segment joining the points $P_{S_2}x_0$ and $P_{S_1}P_{S_2}x_0$ (in practical compu-

Algorithm 2.2 AGP algorithm using separating hyperplanes

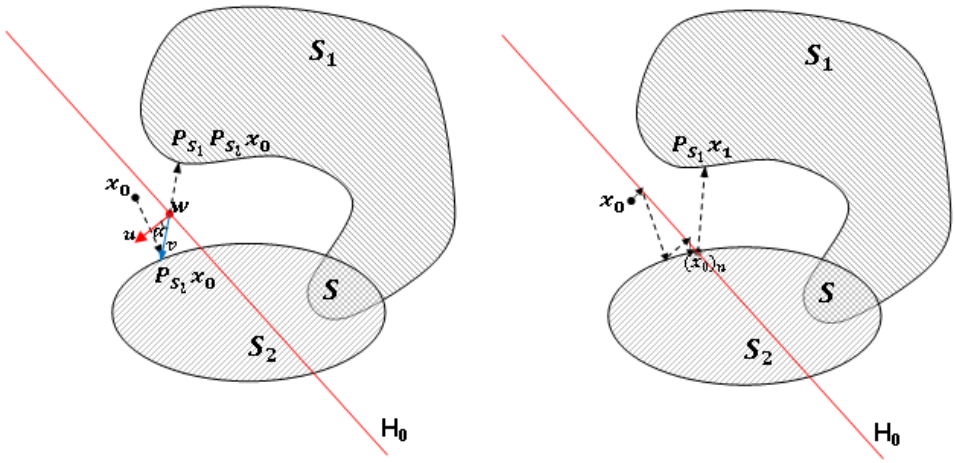
Require: $x \in H$, $\epsilon = tolerance$ $x = x_0$ **Output** $(x_i)_{n+1}$

for $i = 0, 1, 2, \ldots$ **do**
  $J(x_i) = \|P_{S_2}x_i - x_i\| + \|P_{S_1}x_i - x_i\|$
  for $n = 0, 1, 2, \ldots$ **do**
    $(x_i)_n = x_i$
    Choose a hyperplane H_i that separates $(x_i)_n$ and S_2,
    $(x_i)_{n+1} = P_{S_1}P_{\mathrm{H}_i}(x_i)_n$
    $k((x_i)_n) = \|P_{\mathrm{H}_i}(x_i)_n - (x_i)_n\|$
    if $k((x_i)_n) < \epsilon$ **then**
      the algorithm converges to $(x_i)_n \in (S_1 \cap \mathrm{H}_i)$
      return $(x_i)_n$
    end if
  end for
  $x_i = (x_i)_{n+1}$
  if $J(x_i) \geq \epsilon$ **then**
    continue
  end if
  if $J(x_i) < \epsilon$ **then**
    the algorithm converges to $x_i \in (S_1 \cap S_2)$
    return x_i
  end if
end for

tations it is sufficient to consider the midpoint as w, although it should be such that H_0 approximates somewhat S_1), and define a hyperplane H_0 containing the point w, whose normal vector, say u, form an angle α with the vector v defined as $v = P_{S_2}x_0 - w$ (see Figure 3(a)). So that, α is such that $\mathrm{H}_0 \cap S_2 \neq \emptyset$, and the H_0 separates the points $P_{S_2}x_0 \in S_2$ and $P_{S_1}P_{S_2}x_0 \in S_1$. Thus, $\mathrm{H}_0 = \{z \in \mathbb{R}^n : \langle u, z\rangle = c\}$ is the first separating hyperplane in the process, where u is a normal vector and $c = \langle u, w\rangle$.

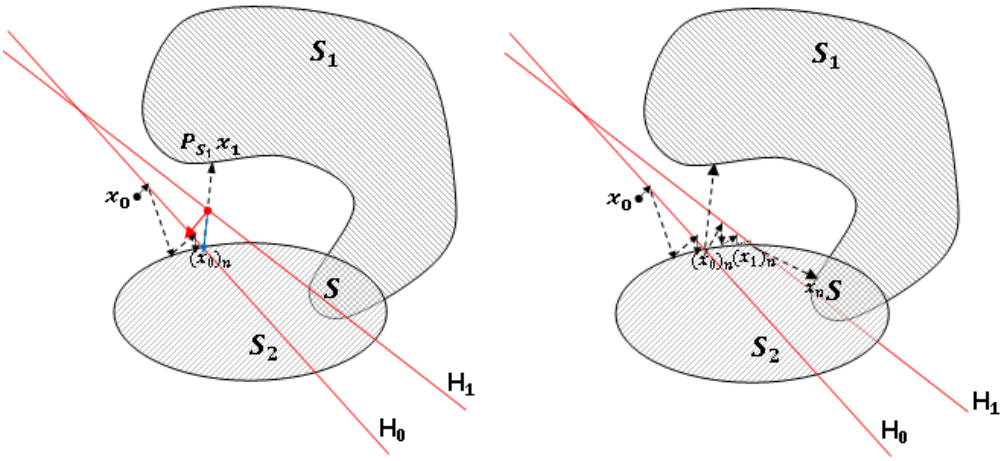
Although the figures show that the projections are made alternatively onto the separating hyperplane H_0 and S_2 until a sequence of elements that converges to a point $(x_0)_n \in S_2 \cap \mathrm{H}_0$ is obtained, this scheme is not the most efficient way to find $(x_0)_n \in S_2 \cap \mathrm{H}_0$. Indeed, there are quite a few of efficient algorithms that can be applied to find a point that belongs to the intersection of convex sets (see, for example, [17], [1]).

The existence of $(x_0)_n$ is guaranteed because we select an angle α such that $\mathrm{H}_0 \cap S_2 \neq \emptyset$. Moreover, successive projections converge to a point at the intersection $\mathrm{H}_0 \cap S_2$ since both are convex sets.



(a) (b)

Fig. 3: (a) A hyperplane H_0 with $w \in \mathrm{H}_0$, and $\mathrm{H}_0 \cap S_2 \neq \emptyset$ (with α appropriate). (b) Alternating projections onto H_0 and S_2. The point $(x_0)_n$ (close to $\mathrm{H}_0 \cap S_2$) is projected onto the nonconvex set.



(a) (b)

Fig. 4: (a) The hyperplane H_1 that separates $(x_0)_n = x_1$ and $P_{S_1}x_1$. (b) When the SDE of the current iterate is less than the SDE of the trap point, the MAP is applied to the sets involved.

Next, we project the point $(x_0)_n = x_1$ onto the set S_1 to obtain the point $P_{S_1}x_1$ (Figure 3(b)). If $x_1 \neq P_{S_1}x_1$ and $J(x_1) = \|P_{S_1}x_1 - x_1\| + \|P_{S_2}x_1 - x_1\| > tolerance$, then other hyperplane H_1 that separates x_1 and $P_{S_1}x_1$ such that $\mathrm{H}_1 \cap S_2 \neq \emptyset$ is defined (see Figure 4(a)). The alternating projections onto H_1 and S_2 are carried out again in order to generate a succession of elements which converge to a point $(x_1)_n = x_2 \in S_2 \cap \mathrm{H}_1$ (see Figure 4(b)). Continuing with this process, and if $J(x_i) < tolerance$, for some $i = 1, 2, ...$, we expect that the algorithm eventually left the region where a point trap was, and generates a sequence of elements that converges to a point in S. In our numerical experiments this strategy was more efficient when, after detecting potential traps points and to leave the trampping area, we continue the algorithm with the MAP implementation projecting alternately on both sets until we get a point in S (or a point close to S).

We summarize these results in Algorithm 2.3.

2.3 AGP method using separating hyperplanes and Moreau's Theorem

In the case that $k((x_i)_{n+1}) = k((x_i)_n) \geq tolerance$, the algorithm will reach a trap point, so that the Algorithm 2.2 may be adapted to use relaxed parameters or the Moreau's Theorem on the separating hyperplane, to leave the entrapment situation.

Theorem 1.2 can be applied when the AGP algorithm with separating hyperplanes, denoted as Algorithm 2.2, converges to a trap point $(x_i)_T \in S_1$. In this case the process is as follows: First, we calculate $P_{\mathrm{H}_i^\circ}(x_i)_T$, which is then projected onto the set S_1; this point is projected onto H_i°, and finally we use the Moreau's Theorem again to recover the projection onto the hyperplane H_i.

It is expected from the application of this strategy that the iterative process of successive projections may leave the region where the point trap $(x_i)_T$ is located.

Algorithm 2.4 is an extension of Algorithm 2.2 for the AGP method using separating hyperplanes and Moreau's Theorem.

As will be discussed in the next section, we can also apply Algorithm 2.4 when S_1 and S_2 are nonconvex sets.

Algorithm 2.3 AGP algorithm using separating hyperplanes ($\mathrm{H}_0 \cap S_2 \neq \emptyset$)

Require: $x \in H$, $\epsilon = tolerance$ $x = x_0$ **Output** x_n

for $i = 0, 1, 2, \ldots$ **do**
  $x_{i+1} = P_{S_1} P_{S_2} x_i$
  $J(x_i) = \|P_{S_1} x_i - x_i\| + \|P_{S_2} x_i - x_i\|$
  if $J(x_i) < \epsilon$, **then**
    return x_i
    Stop, the algorithm converges to a point en $S_1 \cap S_2$
  end if
  $a = J(x_i)$
  if $|J(x_{i+1}) - J(x_i)| < \epsilon$ **then**
    the algorithm converges to a trap point
    for $n = 0, 1, 2, \ldots$ **do**
      $J(x_n) = \|P_{S_1} x_n - x_n\| + \|P_{S_2} x_n - x_n\|$
      if $J(x_n) \geq a$ **then**
        for $p = 0, 1, 2, \ldots$ **do**
          Choose a hyperplane H_p that separates $P_{S_2} x_n$ and $P_{S_1} P_{S_2} x_n$. $\mathrm{H}_p \cap S_2 \neq \emptyset$
          $(x_p)_{n+1} = P_{S_2} P_{\mathrm{H}_p} (x_p)_n$
          Convergence test: if $\|P_{\mathrm{H}_p} (x_p)_n - (x_p)_n\| < \epsilon$, then stop
        end for
      end if
      if $J(x_n) < a$ **then**
        $x_{n+1} = P_{S_1} P_{S_2} x_n$
        Convergence test: if $\|P_{S_1} x_n - x_n\| + \|P_{S_2} x_n - x_n\| < \epsilon$,
        return x_{n+1}
      end if
    end for
  end if
end for

2.4 AGP method for nonconvex sets

We can also alternate projecting successively onto two nonempty closed sets S_1, S_2. When the projection algorithm seems to converge to a trap point x_T, we define a hyperplane H (in a similar manner as was described in Section 2.2) that separates x_T and say S_2. Next, the polar cone H° of the hyperplane H is defined. Then, we proceed to apply the Moreau's Theorem to obtain a point in H° such that $x_T = P_{\mathrm{H}} x_T + P_{\mathrm{H}^\circ} x_T$.

Subsequently, the point $P_{\mathrm{H}^\circ}(x_T)$ is projected onto S_1, which is then projected onto H°, and finally we use the Moreau's Theorem again to recover the projection onto the hyperplane H.

Algorithm 2.4 AGP algorithm using separating hyperplanes and decomposition in polar cones

Require: $x \in H$, $\epsilon = tolerance$, and $\delta < \epsilon$ $x = x_0$ **Output** $(x_i)_{n+1}$

for $i = 0, 1, 2, \ldots$ **do**
  $J(x_i) = \|P_{S_2}x_i - x_i\| + \|P_{S_1}x_i - x_i\|$
  for $n = 0, 1, 2, \ldots$ **do**
    $(x_i)_n = x_i$
    $(x_i)_{n+1} = P_{S_1}P_{\mathrm{H}_i}(x_i)_n$
    $k((x_i)_n) = \|P_{\mathrm{H}_i}(x_i)_n - (x_i)_n\|$
    $d = \|(x_i)_{n+1} - (x_i)_n\|$
    if $k((x_i)_n) < \epsilon$ **then**
      the algorithm converges to $(x_i)_n \in (S_1 \cap \mathrm{H}_i)$
      return $(x_i)_n$
    end if
    if $d < \delta$ and $k((x_i)_n) > \epsilon$, the algorithm converges to a trap point. **then**
      $P_{\mathrm{H}_i^o}(x_i)_T = (x_i)_T - P_{\mathrm{H}_i}(x_i)_T$
      $z = P_{S_1}P_{\mathrm{H}_i^o}(x_i)_T$
      $P_{(\mathrm{H}_i^o)^o}z = z - P_{\mathrm{H}_i^o}z$
      $(x_i)_{n+1} = P_{S_1}P_{(\mathrm{H}_i^o)^o}z = P_{S_1}P_{\mathrm{H}_i}z$
    end if
    if $\|P_{\mathrm{H}_i}z - z\| < \epsilon$ **then**
      stop, the algorithm converges to a point in $S_1 \cap \mathrm{H}_i$
    end if
  end for
  $x_i = (x_i)_{n+1}$
  if $J(x_i) \geq \epsilon$ **then**
    continue
  end if
  if $J(x_i) < \epsilon$ **then**
    the algorithm converges to $x_i \in (S_1 \cap S_2)$
    return x_i
  end if
end for

As before, it is expected from the application of this strategy that the iterative process of successive projections may leave the region where the point trap x_T is located. Then, we will continue applying successive projections alternately onto the sets S_1 and S_2 until a point in $S_1 \cap S_2$ is reached.

In Algorithm 2.5, as stopping criterion, the SDE involving x_n, S_1, and S_2

is calculated, i.e.

$$J(x_n) = \|P_{S_2}x_n - x_n\| + \|P_{S_1}x_n - x_n\|.$$

If $J(x_n) < \epsilon$ and ϵ is small enough, it can be safely assumed that the algorithm has converged to a point in $S_1 \cap S_2$.

The above process is summarized in the following version of previous algorithms, which is a combination of the MAP and the decomposition method in polar cones, using separating hyperplanes.

Algorithm 2.5 AGP method for nonconvex sets

Require: $x \in H$, $\epsilon = tolerance$ $x = x_0$ **Output** x_{n+1}

for $n = 0, 1, 2, \ldots$ **do**
  $x_{n+1} = P_{S_1}P_{S_2}x_n$
  $J(x_n) = \|P_{S_2}x_n - x_n\| + \|P_{S_1}x_n - x_n\|$
  if $J(x_n) < \epsilon$ **then**
    the algorithm converges to $x_{n+1} \in (S_1 \cap S_2)$
    return x_{n+1}
  end if
  if $J(x_{n+1}) = J(x_n) > \epsilon$, the algorithm converges to a trap point x_T, **then**
    $x_T = x_n$
    $P_{\mathrm{H}^o}x_T = x_T - P_{\mathrm{H}}x_T$
    $z = P_{S_1}P_{\mathrm{H}^o}x_T$
    $P_{(\mathrm{H}^o)^o}z = z - P_{\mathrm{H}^o}z$
    $x_n = P_{S_1}P_{(\mathrm{H}^o)^o}z = P_{S_1}P_{\mathrm{H}}z$
  end if
end for

The apparent advantage of this strategy with respect to the algorithms proposed so far is the exclusive use of separating hyperplanes to exit entrapment situations, and then continue applying the classical method of alternating projections onto S_1 and S_2. The application of this strategy may allow a substantial saving in computational work time.

3 Numerical experiments

In this section we compare numerically the different strategies proposed to avoid convergence to a trap point. A point x_i generated by the algorithm

will be declared as a trap point when $J(x_i) > 0$ and $|J(x_{i+1}) - J(x_i)|$ is less than a given tolerance.

All experiments in this section were obtained using MATLAB 7.8.0 package and were run on an Intel Core 2 Duo processor (2.20 GHz).

In all experiments we stop the process when the SDE $J(x_n)$ is less than 1×10^{-7} or when the number of iterations surpassess 1000. In general, when an algorithm performed over 1000 iterations, was because the algorithm had already reached a trap point in previous iterations. In practice, these stopping criteria appeared to be sufficient to demonstrate the performance of the algorithms considered in our experiments.

The results of experiments are reported in tables that show the number of iterations (IT) required by each algorithm, the CPU time in seconds (TIME), and the SDE $J(x)$. We observe that, for all algorithms, the total processing time is proportional to the required number of iterations. Also, the λ_i $(i = 1, 2)$ parameters were chosen uniformly at random in each iteration. In the first experiment we use the Frobenius norm, while for the other experiments we used the Euclidean norm. In all the experiments, when a projection onto a nonconvex set is performed more than one projection point is detected, then any one of them is chosen as the projection onto the set. We have included the MAP algorithm in all tables to uncover trap points (i.e., we do not use here the MAP for comparative purposes, but rather as a means for detecting trap points).

Experiment 1.
Our first experiment is chosen to find a matrix X in the intersection of the sets C_1 and C_2, which are defined by

$$C_1 = \{X \in \mathbb{R}^{n\times n} : X \in \mathfrak{B} \quad \text{and} \quad X \notin \mathfrak{C}\},$$

where

$$\mathfrak{B} = \{X \in \mathbb{R}^{n\times n} : A \leq X \leq B,\ A, B \in \mathbb{R}^{n\times n}\}$$

and

$$\mathfrak{C} = \{X \in \mathbb{R}^{n\times n} : C < X < D,\ C, D \in \mathbb{R}^{n\times n}\},$$

with $A \leq C < D \leq B$, and

$$C_2 = \{X \in \mathbb{R}^{n\times n} : X = X^T\}.$$

For given matrices A, B, C and D in $\mathbb{R}^{n\times n}$, the set C_1 is a nonconvex set and will be called a *hollow box* of matrices. The subspace C_2 is the convex

set of all $n \times n$ real symmetric matrices. The projections P_{C_1} and P_{C_2} are defined, for $i, j = 1, \ldots, n$, by

$$(P_{C_1}X)_{ij} = \begin{cases} X_{ij} & \text{if} \quad A_{ij} \leq X_{ij} \leq C_{ij} \\ X_{ij} & \text{if} \quad D_{ij} \leq X_{ij} \leq B_{ij} \\ A_{ij} & \text{if} \quad X_{ij} < A_{ij} \\ B_{ij} & \text{if} \quad X_{ij} > B_{ij} \\ C_{ij} & \text{if} \quad C_{ij} < X_{ij} < D_{ij} \text{ and } d(X_{ij}, C_{ij}) < d(X_{ij}, D_{ij}) \\ D_{ij} & \text{if} \quad C_{ij} < X_{ij} < D_{ij} \text{ and } d(X_{ij}, D_{ij}) < d(X_{ij}, C_{ij}) \\ C_{ij} \text{ or } D_{ij} & \text{if} \quad d(X_{ij}, C_{ij}) = d(X_{ij}, D_{ij}) \end{cases}$$

and

$$P_{C_2}X = \frac{1}{2}(X + X^T),$$

respectively.

Note that C_2 is a convex cone. In this experiment, several hollow boxes of 2×2 matrices were used, and in all the numerical tests similar results were obtained. For the purposes of presenting the experiment results, we consider the following matrices that form a hollow box:

$$A = \begin{pmatrix} 1 & 0 \\ 3 & 6 \end{pmatrix}, \quad C = \begin{pmatrix} 2 & 0 \\ 3 & 7 \end{pmatrix}, \quad D = \begin{pmatrix} 4 & 1 \\ 9 & 8 \end{pmatrix}, \quad B = \begin{pmatrix} 7 & 7 \\ 10 & 9 \end{pmatrix},$$

choosing as starting element the matrix

$$X_0 = \begin{pmatrix} 5 & 11 \\ 13 & -1 \end{pmatrix}.$$

Table 1 shows the numeric behavior of MAP and POCS algorithms and Algorithms 1.1-2.1. MAP revealed the existence of the trap point given by the matrix $\begin{pmatrix} 5 & 8 \\ 8 & 6 \end{pmatrix}$.

The POCS algorithm (second column from Table 1) does not satisfy the SDE reduction property, however it converges to a feasible solution. In general, the number of iterations required to reach a point in the intersection appears to vary widely and depends on the relaxation parameters used by the algorithm.

In this experiment, Algorithm 1.1 exhibited the SDE reduction property (1). After 282 iterations, the algorithm converged to a feasible solution given

MAP		**POCS**		**Algorithm 1.1**			**Algorithm 2.1**	
IT	$J(X_n)$	IT	$J(X_n)$	IT	$J(X_{n+1})$	$J(T_2X_n)$	IT	$J(X_n)$
1	10.01654	1	10.01654	1	3.65699	2.60416	1	10.01654
2	1.58114	2	2.38252	2	1.45284	1.57881	2	1.58114
3	1.41421	3	1.42935	3	1.41787	1.88496	3	1.41421
4	1.41421	4	4.13558	15	1.41421	1.41421	-	Moreau's T.
8	1.41421	46	1.41421	26	1.41421	2.47536	4	1.50000
16	1.41421	85	2.83000	58	1.41421	1.53212	5	0.75000
25	1.41421	144	7.35968	118	1.41421	1.41421	9	0.04688
50	1.41421	149	3.72134	263	0.66504	2.88424	14	0.00146
100	1.41421	154	0.00903	267	0.03145	0.02880	18	0.00009
300	1.41421	158	0.00047	271	0.00050	0.00065	22	0.00001
1000	1.41421	169	0.00000	282	0.00000	0.00000	29	0.00000
TIME:								
0.6240		0.2184		0.3900			0.0624	

Tab. 1: Results of Experiment 1.

by the matrix $\begin{pmatrix} 5 & 3 \\ 3 & 8.8 \end{pmatrix}$. It is worth noting that the number of iterations can vary greatly. This will depend on the relaxation parameters selected at the beginning of the algorithm and the length of the intervals at which the λ_i, $i = 1, 2$, belong (see Theorem 1.1).

In Algorithm 2.1 the SDE reduction property was initially used to locate a possible trap point, and then applying the Moreau's Theorem as was explained in Section 2.1. We see from Table 1 that after applying the Moreau's Theorem, the algorithm only needed 29 iterations to converge to a feasible solution.

In summary, for this first experiment the best algorithm was the AGP algorithm using the decomposition method in polar cones.

Experiment 2.
In our second experiment, a point in the intersection of the following sets is sought:

$$C_1 = \{(x, y, z) \in \mathbb{R}^3 : (x, y, z) \in \mathfrak{E}_1 \cup \mathfrak{E}_2\}$$

and

$$C_2 = \{(x, y, z) \in \mathbb{R}^3 : -2x - 6y - 5z = 0\},$$

where

$$\mathfrak{E}_1 = \{(x, y, z) \in \mathbb{R}^3 : (x - 6)^2 + (y - 2)^2 + z^2 \leq 25\}$$

and

$$\mathfrak{E}_2 = \{(x, y, z) \in \mathbb{R}^3 : x^2 + (y-6)^2 + z^2 \leq 9\}.$$

Here, C_1 is a nonconvex set in $\mathbb{R}^3$ and the subspace C_2 of $\mathbb{R}^3$ is a convex cone.

For $v \in \mathbb{R}^3$ arbitrary, the projections P_{C_1} and P_{C_2} are defined by

$$P_{C_1} v = \inf\{d(v, P_{\mathfrak{E}_1} v), d(v, P_{\mathfrak{E}_2} v)\},$$

where

$$P_{\mathfrak{E}_1} v = \left(\frac{5(x-6)}{\sqrt{(x-6)^2 + (y-2)^2 + z^2}} + 6, \frac{5(y-2)}{\sqrt{(x-6)^2 + (y-2)^2 + z^2}} + 2, \right.$$
$$\left. \frac{5z}{\sqrt{(x-6)^2 + (y-2)^2 + z^2}} \right),$$

$$P_{\mathfrak{E}_2} v = \left(\frac{3x}{\sqrt{x^2 + (y-6)^2 + z^2}}, \frac{3(y-6)}{\sqrt{x^2 + (y-6)^2 + z^2}} + 6, \frac{3z}{\sqrt{x^2 + (y-6)^2 + z^2}} \right),$$

and

$$P_{C_2} v = v - \frac{\langle v, r \rangle}{\langle r, r \rangle} r,$$

where r is the vector normal to the plane.

Table 2 shows the results for this experiment. The point $(-6, 2, 2)$ was the starting point in the implementation of the different algorithms. MAP converged to the trap point $(-0.74721, 3.76737, -1.86052)$.

Table 2 shows that POCS algorithm reached a point in the intersection of C_1 and C_2, though it fails to meet the SDE reduction property.

Algorithm 1.1 generates a sequence that converges to the point (3.2214, $0.7111, -2.1418) \in C_1 \cap C_2$.

At iteration 326 the algorithm was able to leave the region where the trap point was located and thereafter to converge to a feasible solution.

Finally, Algorithm 2.1 meets the SDE reduction property but converges to a trap point. After 49 iterations, the Moreau's Theorem is applied, and the algorithm converges to a feasible point in only 66 iterations.

We can conjecture that the application of Moreau's Theorem was crucial to abandon the entrampment region.

MAP		POCS		Algorithm 1.1			Algorithm 2.1	
IT	$J(v_n)$	IT	$J(v_n)$	IT	$J(v_{n+1})$	$J(T_2v_n)$	IT	$J(v_n)$
1	5.72366	1	5.72366	1	3.34745	5.48308	1	5.72366
2	2.81055	2	3.34745	2	1.68196	3.61667	2	2.81055
3	1.84820	3	1.72538	3	1.65250	1.57024	3	1.84820
4	1.61506	16	2.75426	15	1.46531	1.46532	8	1.47089
8	1.47089	40	1.73154	44	1.46525	1.46525	16	1.46526
25	1.46525	56	0.94326	230	2.44962	2.89477	-	Moreau's T.
50	1.46525	61	0.03789	326	1.20874	6.50142	50	0.48612
100	1.46525	64	0.00096	329	0.02550	0.02550	54	0.00607
300	1.46525	68	0.00003	331	0.00013	0.00013	59	0.00003
1000	1.46525	76	0.00000	340	0.00000	0.00000	66	0.00000
TIME:								
0.2808		0.1404		0.2340			0.0623	

Tab. 2: Results of Experiment 2.

Experiment 3.
In this experiment, one of the sets to be considered is a convex set but not a cone. In this case Algorithm 2.2 is applied, using separating hyperplanes (Section 2.2). If, when executing this algorithm, the successive projections converge to a trap point, we can apply Algorithm 2.4, which makes use of Moreau's Theorem on separating hyperplanes (see Section 2.3). Furthermore, in this experiment we also apply Algorithm 2.5, since this algorithm is also used when the set is not a convex cone (see Section 2.4).

Let now C_1 and C_2 be the sets defined by

$$C_1 = \left\{(x,y) \in \mathbb{R}^2 : (x,y) \in \mathfrak{E}_1 \cup \mathfrak{E}_2 \cup \mathfrak{E}_3\right\}$$

and

$$C_2 = \left\{(x,y) \in \mathbb{R}^2 : (x-6)^2 + (y+3)^2 \leq 9\right\},$$

where

$$\mathfrak{E}_1 = \left\{(x,y) \in \mathbb{R}^2 : x^2 + (y-3)^2 = 9\right\},$$

$$\mathfrak{E}_2 = \left\{(x,y) \in \mathbb{R}^2 : (x-5)^2 + \left(y - \frac{9}{2}\right)^2 = \frac{25}{4}\right\},$$

and

$$\mathfrak{E}_3 = \left\{(x,y) \in \mathbb{R}^2 : \left(x - \frac{17}{2}\right)^2 + \left(y - \frac{3}{2}\right)^2 = \frac{25}{4}\right\},$$

For $v = (x,y) \in \mathbb{R}^2$ arbitrary, the projections P_{C_1} and P_{C_2} are defined by

$$P_{C_1}v = \inf\{d(v, P_{\mathfrak{E}_1}v), d(v, P_{\mathfrak{E}_2}v), d(v, P_{\mathfrak{E}_3}v)\},$$

and

$$P_{C_2}v = \left(\frac{3(x-6)}{\sqrt{(x-6)^2+(y+3)^2}} + 6, \frac{3(y+3)}{\sqrt{(x-6)^2+(y+3)^2}} - 3\right),$$

respectively, and where

$$P_{\mathfrak{E}_1}v = \left(\frac{3x}{\sqrt{x^2+(y-3)^2}}, \frac{3(y-3)}{\sqrt{x^2+(y-3)^2}} + 3\right),$$

$$P_{\mathfrak{E}_2}v = \left(\frac{\frac{5}{2}(x-5)}{\sqrt{(x-5)^2+(y-\frac{9}{2})^2}} + 5, \frac{\frac{5}{2}(y-\frac{9}{2})}{\sqrt{(x-5)^2+(y-\frac{9}{2})^2}} + \frac{9}{2}\right),$$

and

$$P_{\mathfrak{E}_3}v = \left(\frac{\frac{5}{2}(x-\frac{17}{2})}{\sqrt{(x-\frac{17}{2})^2+(y-\frac{3}{2})^2}} + \frac{17}{2}, \frac{\frac{5}{2}(y-\frac{3}{2})}{\sqrt{(x-\frac{17}{2})^2+(y-\frac{3}{2})^2}} + \frac{3}{2}\right),$$

The projection of v onto the separating hyperplane

$$\mathrm{H}_i = \{z \in \mathbb{R}^2 : \langle p_i, z\rangle = \alpha_i\}, \quad \text{for } i = 0,1,2,...,$$

is given by:

$$P_{\mathrm{H}_i}v = v + \frac{\langle p_i, w_i\rangle - \langle p_i, v\rangle}{\langle p_i, p_i\rangle}p_i, \tag{2}$$

where $p_i = (-m_i, 1)$ with m_i the slope of the tangent line to the set C_2 at the point $P_{C_2}v_i$, and $w_i \in \mathrm{H}_i$ is a point between the points v and $P_{C_2}v$ such that $\mathrm{H}_i \cap C_1 \neq \emptyset$ (see Figure 1). It is also possible to choose $w_i = P_{C_2}v$ from which we will have a supporting hyperplane. In the case of Algorithm 2.5, we also defined separating hyperplanes, but rather than using the point v, it is used the trap point x_T.

Tables 3, 4, and 5 show the results for this experiment. The point $(-1,-1)$ was used as the starting point for the algorithms.

From Table 3 it is clear that the MAP fails to converge to a feasible solution and seems to converge to the trap point $(2.12132, 0.87868)$. It is also seen that POCS reached a point in the intersection of C_1 and C_2. The POCS

MAP		**POCS**		**Algorithm 1.1**		
IT	$J(v_n)$	IT	$J(v_n)$	IT	$J(v_{n+1})$	$J(T_2v_n)$
1	5.40322	1	5.40322	1	3.55743	3.84832
2	2.62059	2	3.45743	2	3.55293	3.04391
3	2.49662	3	2.90270	3	2.49033	2.65472
4	2.48629	9	3.69109	4	2.48662	2.67986
5	2.48537	13	3.27434	10	2.48528	2.48528
6	2.48529	18	3.50454	21	2.48528	2.48528
7	2.48528	21	2.72536	35	2.48528	2.48528
50	2.48528	31	0.00076	56	0.00643	0.03214
500	2.48528	36	0.00016	59	0.00005	0.00026
1000	2.48528	50	0.00000	65	0.00000	0.00000
TIME:						
0.3432		0.1560		0.1404		

Tab. 3: Results of Experiment 3.

algorithm fails to meet the SDE reduction property. But it was not until iteration 21 when the relaxation parameters $\lambda_1 = 0.57210$ and $\lambda_2 = 1.75803$ allowed to leave the region of entrapment.

Algorithm 1.1 reaches a point at the intersection of the involved sets and apparently verified the SDE reduction property. It is important to note that after 10 iterations the algorithm reached a trap point, but due to a suitable choice of the relaxation parameters, the algorithm converged to the correct solution after 64 iterations.

In Table 4 the numerical results by applying Algorithms 2.2 and 2.5 are shown. Although Algorithm 2.2 needed more iterations to achieve convergence to a correct solution than the POCS method and Algorithms 1.1, it can be guaranteed in this experiment that the algorithm did not fall in a region of entrapment and did not seem to have depended on a choice of relaxation parameters to achieve a feasible solution. Moreover, the SDE reduction property is satisfied. It is also seen that Algorithm 2.5 needs only 42 iterations to converge to a point in the intersection of C_1 and C_2. We think this was because when the iterates generated by the algorithm fell into a region of entrapment, it was only necessary to use a separating hyperplane and the strategy of decomposition in polar cones, so that the algorithm quickly left the state of entrapment and converged to a feasible solution.

Table 5 shows the numerical results obtained from the application of Al-

Algorithm 2.2					Algorithm 2.5	
Separ. hyperp.	$J(v_i)$	IT	$k((v_i)_n)$	Note	IT	$J(v_{n+1})$
H_1	5.40322	1	5.38650		1	5.40322
		2	2.21093		2	2.62059
		6	0.00000	it converges to $H_1 \cap C_1$	4	2.48629
H_2	2.36517	7	2.35593		10	2.48528
		8	0.16555		19	2.48528
		13	0.00000	it converges to $H_2 \cap C_1$	–	H. and M.T.
H_3	0.29052	14	0.28938		20	0.66636
		21	0.00112		21	0.18976
		35	0.00000	it converges to $H_3 \cap C_1$	22	0.08496
H_4	0.01042	36	0.01038		24	0.02286
		41	0.00056		26	0.00691
		57	0.00000	it converges to $H_4 \cap C_1$	28	0.00216
H_5	0.00006	58	0.00006		31	0.00038
		66	0.00000		34	0.00007
		71	0.00000	it converges to $H_5 \cap C_1$	37	0.00001
H_6	0.00000	72	0.00000	it converges to $C_1 \cap C_2$	42	0.00000
TIME:						
0.1092					0.0780	

Tab. 4: Results of Experiment 3. Algorithms 2.2 and 2.5.

gorithm 2.3 in the way as it was stated in Section 2.2.1. In the case of not obtaining an approximate intersection point in $H_i \cap C_2$ (see Figure 3), then $H_i \cap C_2 = \emptyset$ and we get another separating hyperplane, choosing another value of the angle such that $H_i \cap C_2 \neq \emptyset$.

Experiment 4.
In this experiment the sets C_1 and C_2 are defined by

$$\begin{aligned} C_1 &= \left\{(x,y,z) \in \mathbb{R}^3 : (x,y,z) \in \mathfrak{F}_1 \cup \mathfrak{F}_2 \cup \mathfrak{F}_3\right\} \quad \text{and} \\ C_2 &= \left\{(x,y,z) \in \mathbb{R}^3 : (x-6)^2 + (y+1)^2 + (z+1)^2 \leq 9\right\}, \end{aligned}$$

where

$$\mathfrak{F}_1 = \left\{(x,y,z) \in \mathbb{R}^3 : x^2 + (y-5)^2 + (z+1)^2 = 9\right\},$$

$$\mathfrak{F}_2 = \left\{(x,y,z) \in \mathbb{R}^3 : (x-5)^2 + \left(y - \frac{13}{2}\right)^2 + (z+1)^2 = \frac{25}{4}\right\},$$

Algorithm 2.3				
Separ. hyperp.	$J(v_i)$	IT	$J((v_p)_n)$	Note
H_1 $\alpha = \frac{\pi}{4}$	5.40322	1	5.20669	
		2	0.69934	
		6	0.00024	
		11	0.00000	it converges to $H_1 \cap C_2$
H_2 $\alpha = \frac{\pi}{3}$	2.52910	12	0.63227	
		14	0.02976	
		18	0.00015	
		25	0.00000	it converges to $H_2 \cap C_2$
H_3 $\alpha = \frac{\pi}{4}$	2.42506	26	0.85739	
		28	0.21901	
		35	0.00009	
		42	0.00000	it converges to $H_3 \cap C_2$
MAP	1.18485	43	1.18485	
		48	0.01919	
		57	0.00010	
		69	0.00000	it converges to $C_1 \cap C_2$
				TIME: 5.406351 seconds

Tab. 5: Results of Experiment 3. Algorithm 2.3.

and

$$\mathfrak{F}_3 = \left\{ (x,y,z) \in \mathbb{R}^3 : \left(x - \frac{17}{2}\right)^2 + \left(y - \frac{7}{2}\right)^2 + (z+1)^2 = \frac{25}{4} \right\}.$$

Here C_1 is a nonconvex set and C_2 is convex but not a cone. For $v = (x,y,z) \in \mathbb{R}^3$ arbitrary, the projections P_{C_1} and P_{C_2} are defined by

$$P_{C_1} v = \inf\{d(v, P_{\mathfrak{F}_1}(v)), d(v, P_{\mathfrak{F}_2}(v)), d(v, P_{\mathfrak{F}_3}(v))\},$$

where

$$P_{\mathfrak{F}_1} v = \left(\frac{3x}{\sqrt{r}}, \frac{3(y-5)}{\sqrt{r}} + 5, \frac{3(z+1)}{\sqrt{r}} - 1 \right)$$

with

$$r = x^2 + (y-5)^2 + (z+1)^2,$$

$$P_{\mathfrak{F}_2} v = \left(\frac{\frac{5}{2}(x-5)}{\sqrt{t}} + 5, \frac{\frac{5}{2}(y - \frac{13}{2})}{\sqrt{t}} + \frac{13}{2}, \frac{\frac{5}{2}(z+1)}{\sqrt{t}} - 1 \right)$$

with

$$t = (x-5)^2 + \left(y - \frac{13}{2}\right)^2 + (z+1)^2,$$

$$P_{\Im_3} v = \left(\frac{\frac{5}{2}(x - \frac{17}{2})}{\sqrt{u}} + \frac{17}{2}, \frac{\frac{5}{2}(y - \frac{7}{2})}{\sqrt{u}} + \frac{7}{2}, \frac{\frac{5}{2}(z+1)}{\sqrt{u}} - 1 \right)$$

with

$$u = \left(x - \frac{17}{2}\right)^2 + \left(y - \frac{7}{2}\right)^2 + (z+1)^2,$$

and

$$P_{C_2} v = \left(\frac{3(x-6)}{\sqrt{s}} + 6, \frac{3(y+1)}{\sqrt{s}} - 1, \frac{3(z+1)}{\sqrt{s}} - 1 \right)$$

with

$$s = (x-6)^2 + (y+1)^2 + (z+1)^2.$$

The purpose of this experiment is to compare the results of the first three algorithms with Algorithm 2.3, Algorithms 2.4, and 2.5. In the case of Algorithm 2.4 the separating hyperplane H_i is defined by

$$\mathrm{H}_i = \{z \in \mathbb{R}^3 : \langle p_i, z \rangle = \alpha_i\}, \quad i = 0, 1, 2, \ldots,$$

where p_i is the gradient vector of C_2 at the point $P_{C_2}(v_i)$, $\alpha_i = \langle p_i, w_i \rangle$ with $w_i \in \mathrm{H}_i$ is a point between v_i and $P_{C_2}(v_i)$ such that $\mathrm{H}_i \cap C_1 \neq \emptyset$, and the projection of v_i onto H_i is given by

$$P_{\mathrm{H}_i} v_i = v_i + \frac{\langle p_i, w_i \rangle - \langle p_i, v_i \rangle}{\langle p_i, p_i \rangle} p_i.$$

For Algorithm 2.5, the separating hyperplane is defined in the same way as for Algorithm 2.4, but instead of using v_i, we used the trap point x_T.

Tables 6, 7 and 8 show the results for this experiment. The point $(-4, -3, -4)$ was chosen as the initial point for all the algorithms.

From Table 6 it is clear that the MAP converges to a trap point (i.e., $(2.12132, 2.87868, -1)$). Furthermore, the iterates generated by the POCS algorithm converged to a point in the intersection of the involved sets after 110 iterations. In case of Algorithm 1.1, it needed 115 iterations to converge to a point in $C_1 \cap C_2$.

The numerical results obtained by application of Algorithm 2.3 are shown in Table 7 in the way as it was stated in Section 2.2.1. Here five separation

MAP		**POCS**		**Algorithm 1.1**		
IT	$J(v_n)$	IT	$J(v_n)$	IT	$J(v_{n+1})$	$J(T_2 v_n)$
1	14.06413	1	14.06413	1	2.76470	4.34215
2	2.76470	2	2.76470	2	2.57770	2.63536
3	2.50718	3	2.63572	3	3.07098	2.56447
4	2.48722	4	2.51592	4	2.50688	2.50846
5	2.48545	15	4.50135	14	2.62844	2.48528
6	2.48530	22	3.99686	32	3.05198	2.48528
7	2.48528	41	2.48528	63	1.72472	2.55259
12	2.48528	66	2.48528	65	0.22494	0.97333
50	2.48528	75	0.02920	70	0.05870	0.11546
500	2.48528	94	0.00006	96	0.00005	0.00011
1000	2.48528	110	0.00000	115	0.00000	0.00000
TIME:						
0.3431		0.1872		0.2340		

Tab. 6: Results of Experiment 4

hyperplanes were needed by Algorithm 2.3 for achieving feasible solution in 80 iterations (this number depends on the hyperplanes and angles α selected in the process), thereby avoiding entrapment points.

In Table 8 the results obtained by applying Algorithms 2.4 and 2.5 are shown. It is observed that for the separating hyperplane H_2, Algorithm 2.4 converges to a trap point. By using in iterations 37 and 38 the decomposition in polar cones and the Moreaus's Theorem (see Section 2.3), this algorithm achieved a correct solution after using 4 separating hyperplanes.

With regard to the results obtained by applying Algorithm 2.5, it is noted that this algorithm satisfies the given tolerance using 50% fewer iterations than those required for the previous algorithms. Demonstrating the efficiency of this algorithm, in our numerical experiments, with respect to the other algorithms considered in this paper. Also, the SDE reduction property was satisfied.

4 Final remarks

After a large number of numerical tests we reached the following concluding remarks:

- Despite the generalized projection algorithm proposed by Levi (Al-

Algorithm 2.3				
Separ. hyperp.	$J(v_i)$	IT	$J((v_p)_n)$	Note
H_1 $\alpha = \frac{3\pi}{4}$	14.06413	1	13.00065	
		2	0.82039	
		8	0.00002	
		12	0.00000	it converges to $H_1 \cap C_2$
H_2 $\alpha = \frac{3\pi}{4}$	4.02229	13	0.65002	
		15	0.00025	
		17	0.00000	it converges to $H_2 \cap C_2$
H_3 $\alpha = \frac{3\pi}{4}$	3.77055	18	0.76956	
		21	0.00004	
		24	0.00000	it converges to $H_3 \cap C_2$
H_4 $\alpha = \frac{5\pi}{6}$	3.48736	25	0.34982	
		28	0.00003	
		31	0.00000	it converges to $H_4 \cap C_2$
H_5 $\alpha = \frac{2\pi}{3}$	3.47305	32	1.16594	
		41	0.00055	
		54	0.00000	it converges to $H_5 \cap C_2$
MAP	2.20265	55	2.20265	
		68	0.00018	
		80	0.00000	it converges to $C_1 \cap C_2$
				TIME: 10.629808

Tab. 7: Results of Experiment 4. Algorithm 2.3.

gorithm 1.1) does not fulfill the SDE reduction property, in most of the experiments the algorithm converged to a feasible solution. The number of iterations depended on the values taken by the relaxation parameters and the starting point of the algorithm. A strategy that proved to be useful in our experiments was to generate the relaxation parameters randomly, until we detect that the algorithm abandons the entrapment region. Thereafter, we control the changes of the relaxation values. The SDE reduction proved to be an useful tool for monitoring whether the algorithm converged to a feasible solution or to a trap point.

- When the alternating projection method converges to a trap point and one of the sets is a cone, the method of decomposition in polar cones (Moreau's Theorem) has proved to be an effective approach to leave the entrapment region and converge to a feasible solution. The Moreau's

Algorithm 2.4					Algorithm 2.5	
Separ. hyperp.	$J(v_i)$	IT	$k((v_i)_n)$	Note	IT	$J(v_{n+1})$
H_1	14.06413	1	14.03432		1	14.06413
		4	0.02996		2	2.76470
		12	0.00000	it converges to $H_1 \cap C_1$	5	2.48545
H_2	2.48958	13	2.47986		11	2.48528
		20	2.47321		19	2.48528
		37	2.47321	it converges to a trap p.	–	H. and M.T.
		38	0.66317	using polar dec. on H_2	20	1.78476
		44	0.00531		21	0.22985
		57	0.00000	it converges to $H_2 \cap C_1$	22	0.09851
H_3	2.07393	58	2.06583		24	0.02588
		65	0.00043		26	0.00778
		72	0.00000	it converges to $H_3 \cap C_1$	28	0.00242
H_4	0.04294	73	0.04278		30	0.00076
		79	0.00107		32	0.00024
		92	0.00000	it converges to $H_4 \cap C_1$	34	0.00008
H_5	0.00052	93	0.00052		36	0.00002
		106	0.00000	it converges to $H_5 \cap C_1$	38	0.00001
H_6	0.00000	107	0.00000	it converges to $C_1 \cap C_2$	42	0.00000
TIME:						
0.1404					0.1092	

Tab. 8: Results of Experiment 4. Algorithms 2.4 and 2.5.

Theorem strategy is a novel attempt, which, to our knowledge, has not been previously reported in the context of the entrapment problem.

- The AGP method using separating hyperplanes and Moreau's Theorem (Algorithm 2.4) is a strategy that could be applied even if none of the sets involved in the algorithm is a convex cone (and could be applied even when none of the sets was convex). In all cases considered, the algorithm was able to avoid convergence to a trap point.

- In Algorithm 2.5 it was possible to simplify the strategy proposed by Algorithm 2.4 to obtain a numerically more efficient iterative algorithm, on all problems we tried, in terms of number of iterations and accuracy.

- The MAP proved to be useful for detecting trap points.

To give continuity to this study we think it is important to provide the necessary theoretical foundations that validate the conjectures used by the algorithms proposed in this paper.

A practical situation that arises in real modeling is related to the *false* modeling of a null intersection of a finite number of constraints. In the

proposed algorithms we always considered two closed sets with nonempty intersection. However, we expect that in the near future we extend these algorithms to this important practical case.

Currently most real world applications (e.g., in Digital Image Processing and Pattern Recognition) deal with nonconvex sets. We expect that in the near future our strategies of resolution can be applied to this kind of problems.

Acknowledgements: The authors thank Prof. Marcos Raydan from Universidad Simón Bolívar for his constructive comments. Special thanks also to reviewers for their valuable work.

References

[1] Escalante R. and Raydan M. *Alternating Projections Methods*. Society for Industrial and Applied Mathematics, Philadelphia, 2011.

[2] Bauschke H.H. and Borwein J.M. On Projection algorithms for solving convex feasibility problems. *SIAM Review*, 38:367–426, 1996.

[3] Stark H. and Yang Y. *Vector Space Projections: A Numerical Approach to Signal and Image Processing, Neural Nets, and Optics*. John Wiley & Sons, Inc., New York, 1998.

[4] Tropp J.A., Dhillon I.S., Heath Jr. R.W. and T. Strohmer. Designing structured tight frames via an alternating projection method. *IEEE Transactions on Information Theory*, 51(1):188–209, 2005.

[5] Stark H., Yang Y. and Gurkan D. Factors affecting convergence in the design of diffractive optics by iterative vector-space methods. *J. Opt. Soc. Am. A*, 16(1):149–159, 1999.

[6] Von Neumann J. *Functional Operators Vol. II. The Geometry of Orthogonal Spaces*. Annals of Math. Studies 22. Princeton University Press, Princeton, NJ, 1950. This is a reprint of mimeographed lecture notes first distributed in 1933.

[7] Cheney W. and Goldstein A. Proximity maps for convex sets. *Proc. Amer. Math. Soc.*, 10:448–450, 1959.

[8] Combettes P. and Trussell H.J. Method of successive projections for finding a common point of sets in metric spaces. *J. Optim. Theory Appl.*, 67:487–507, 1990.

[9] Chrétien S. and Bondon P. Cyclic Projection Methods on a Class of Nonconvex Sets. *Numer. Funct. Anal. and Optimiz.*, 17:37–56, 1996.

[10] Tam M.K. Alternating Projection Methods Failure in the Absence of Convexity, *Australian Mathematical Sciences Institute.* `http://www.amsi.org.au/images/stories/downloads/pdfs/education/VacationScholarships/2011-2012/MattTamfinal.pdf`, February 2012. On-line; accessed 12-August-2012.

[11] Richter-Gebert J. and Kortenkamp U.H. *Cinderella.2 Manual: Working with The Interactive Geometry Software.* Springer-Verlag, Berlin, 2012.

[12] Bauschke H.H., Phan H.M. and Wang X. The Method of Alternating Relaxed Projections for two nonconvex sets, arXiv:1305.4296 [math.OC]. `http://arxiv.org/abs/1305.4296`, May 2013.

[13] Levi A. Image Restoration by the Method of Projections with Applications to the Phase and Magnitud Retrieval Problems, 1983. Ph.D. Thesis, Rensselaer Polytechnic Institute, Dept. of ECSE, Troy, NY.

[14] Pierra G. Decomposition through formalization in a product space. *Math. Programming*, 28:96–115, 1984.

[15] Moreau J.-J. Décomposition orthogonale d'un espace hilbertien selon deux cônes mutuellement polaires. *C.R. Acad. Sci. Paris*, 255:238–240, 1962.

[16] Levi A. and Stark H. Restoration from phase and magnitude by generalized projections. In H. Stark, editor, *Image Recovery: Theory and Applications*, pages 277–320. Academic Press, Orlando, FL, 1987.

[17] Censor Y. and Zenios S.A. *Parallel Optimization. Theory, Algorithms, and Applications.* Oxford University Press, New York, 1997.