An acceleration scheme for solving convex feasibility problems using incomplete projection algorithms *

N. Echebest † M. T. Guardarucci † H. Scolnik ‡ M.C.Vacchino †

December 2001.

Abstract

The Projected Aggregation Methods (PAM) for solving linear systems of equalities and/or inequalities, generate a new iterate x^{k+1} by projecting the current point x^k onto a separating hyperplane generated by a given linear combination of the original hyperplanes or halfspaces. In H. Scolnik et al [12] we introduced acceleration schemes for solving systems of linear equations by applying optimization techniques to the problem of finding the optimal combination of the hyperplanes within a PAM like framework. In this paper we generalize those results, introducing a new accelerated iterative method for solving systems of linear inequalities, together with the corresponding theoretical convergence results. In order to test its efficiency, numerical results obtained applying the new acceleration scheme to two algorithms introduced by U. M. García-Palomares and F. J. González-Castaño [6] are given.

Key words. Aggregated projection methods, systems of inequalities, incomplete projections.

AMS subject classifications. 65F10

1 Introduction

The class of convex feasibility problems consisting in finding an element of a non-empty closed C convex set which is a subset of $\Re^n$, defined by

$$C = \{x \in \Re^n : g_i(x) \leq 0,\ i = 1, 2, ..., m\}$$

*Work supported by the universities of Buenos Aires and La Plata (Project 11/X243), Argentina.

†Departamento de Matemática, Facultad de Ciencias Exactas, Universidad Nacional de La Plata, Argentina(opti@mate.unlp.edu.ar).

‡Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina(hugo@dc.uba.ar).

such that the functions $g_i(.)$ are convex and differentiable in $\Re^n$, have a wide range of applications, like image reconstruction techniques [2, 8]. One of the well-known methods appearing in the literature for solving the problem $C = \bigcap^m{}_{i=1} C_i$, where each C_i is convex and closed, is the algorithm of sequential orthogonal projections of Gubin, Polyak and Raik [7]. That method is particularly useful when the projections onto C_i can be easily computed, as for instance in the particular case of hyperplanes or halfspaces. This last observation leads to generate for each iterate x^k, in more general cases, a suitable closed convex set $\mathcal{S}^k$, satisfying $C \subset \mathcal{S}^k$, that facilitates the computation of the exact projection onto it [6]. The choice of the sets $\mathcal{S}^k$ may have a significant influence on the convergence rate of the algorithms. García-Palomares and González-Castaño in [6] have proposed an incomplete projections algorithm (IPA) for obtaining an approximate projection of x^k onto special convex sets S_i^k, $C \subset S_i^k$, $i = 1, 2, \ldots, q$, arising from the separation of subsets of the violated constraints. The next iterate is defined by means of the projection of x^k onto a hyperplane H^k strictly separating C, generated by means of a combination of aggregate hyperplanes H_i^k relative to the incomplete projections onto each set S_i^k.

In this paper we present an accelerated iterative projection method for solving the convex feasibility problem defined by $Ax \leq b$, $A \in \Re^{m \times n}$, $b \in \Re^m$. This method is an extension of the projection methods for solving systems of linear equations given in [11] and [12]. The general scheme is similar to the IPA algorithm given in [6], and therefore is very convenient for parallel processing. The idea is, therefore, that at the current iterate the set of violated constraints is splitted into subsets or blocks S_i^k, in such a way that the required incomplete projection is obtained by combining exact projections onto simple convex sets. The new iterate x^{k+1} is defined by the projection of the current x^k onto a separating hyperplane given by a suitable combination of the hyperplanes obtained by incomplete projections onto the blocks.

The new acceleration schemes here presented appear in the algorithm DACCIM when exact projections are made, and later on they are applied to a more general method called ACIPA, both for computing incomplete projections onto each block as well as for obtaining the new iterate x^{k+1}. The basic idea is to consider at the current iterate the generated separating hyperplane, forcing the new iterate to lie on the convex region defined by it.

In Section 2 the acceleration scheme for solving systems of linear inequalities is presented and applied to a version of the EPA algorithm that employs exact projections as in [6], and prove the main results leading to an improved rate of convergence. In subsection 2 we apply the acceleration scheme to the ACIPA algorithm, that uses incomplete projections onto blocks of inequalities. In Section 3 the numerical results obtained with the new method are given, showing the efficiency of the acceleration scheme. The last section summarizes the conclusions.

2 Algorithms and convergence properties

Consider the non-empty convex set C characterized by a systems of m linear inequalities

$$C := \{x \in \Re^n : \ a_i^T x \leq b_i, \ i = 1, 2, \ldots, m\} \tag{2.1}$$

The matrix of the system will be denoted by $A \in \Re^{m \times n}$, and $\|x\|$ will be the 2-norm of $x \in \Re^n$. We will assume that each row a_i^T of A is such that $\|a_i\| = 1$. We will use the notation x^* for any feasible solution of $Ax \leq b$.

For $i = 1, 2, \ldots, m$, we define

$$\mathcal{C}_i = \{x \in \Re^n : \ a_i^T x \leq b_i, \ b_i \in \Re\}, \ \text{and} \ \ P_{\mathcal{C}_i}(x) = argmin_{y \in \mathcal{C}_i} \|x - y\|.$$

The general scheme of a parallel projection algorithm [6] for finding an element of C is as follows:

Given x^k, we define the set of violated constraints $J^k := \{j : \ a_j^T x^k - b_j > -\theta\}$, where θ is zero or a fixed positive constant as in [6]. This set is splitted into subsets $J_1^k, J_2^k, \ldots, J_q^k$, such that $J^k = \bigcup_{i=1}^q J_i^k$, leading to the corresponding subsets of inequalities

$$S_i^k := \{x \in \Re^n : \ a_j^T x \leq b_j, \ j \in J_i^k\} \tag{2.2}$$

In the exact projection methods for each S_i^k the projection of x^k,
$y_i^k = argmin_{y \in S_i^k} \|y - x^k\|$ is calculated.

As an alternative, García-Palomares and González-Castaño in [6] proposed an algorithm (IPA) for obtaining approximate projections y_i^k of x^k onto sets S_i^k, $i = 1, 2, \ldots, q$, satisfying

$$[x^* \in C] \Rightarrow [\|y_i^k - x^*\| < \|x^k - x^*\|] \tag{2.3}$$

This condition is in particular satisfied if y_i^k is the exact projection onto S_i^k.

In order to assure convergence to a solution of C, the next iterate x^{k+1} is defined using a combination of the directions $d_i^k = y_i^k - x^k$, $d^k = \sum_{i=1}^q w_i^k d_i^k$, $w_i^k \geq 0$, $\sum_{i=1}^q w_i^k = 1$. Then the next iterate is defined as $x^{k+1} = x^k + \omega_k \lambda_k d^k$, where $\eta \leq \omega_k \leq 2 - \eta$, $0 < \eta \leq 1$, and λ_k depends on the chosen algorithm.

The value defined for λ_k in [6], when $\omega_k = 1$, determines that the next iterate coincides with the projection of x^k onto a strictly separating hyperplane of C.

In the following, we will describe the adaptation of the algorithms IPA and EPA, given in [6], to systems of linear inequalities considering a choice of $\{w_i^k\}_{i=1}^q$, $\eta \leq \omega_k \leq 2 - \eta$, $0 < \eta \leq 1$.

Algorithm 1 *Parallel Incomplete projections Algorithm (IPA)[6](k-th iteration).*
Given $x^k \notin C$, $0 < \theta < 0.1$, $0 < \eta \leq 1$.

- *Define* $J^k := \{j : \ a_j^T x^k - b_j > -\theta\}$, *and* $q_k = card(J^k)$.
  Define sets $S_1^k, S_2^k, \ldots, S_q^k$ *according to (2.2).*

- *For* $i = 1, 2, \ldots, q$ *in parallel*
  Compute y_i^k *such that it satisfies (2.3),*
  Define $d_i^k = y_i^k - x^k$
  End For.

- *Define* $d^k = \sum_{i=1}^q w_i{}^k d_i^k$, $\quad \sum_{i=1}^q w_i{}^k = 1, \ \ w_i^k \geq 0$.
  Calculate

$$\lambda_k = \frac{\sum_{i=1}^q w_i^k \|d_i^k\|^2}{2\|d^k\|^2} \tag{2.4}$$

- *Define* $x^{k+1} = x^k + \omega_k \lambda_k d^k$, $\eta \leq \omega_k \leq 2 - \eta$. $\diamond$

Remark 1 *When* $\omega_k = 1$, *the iterate* x^{k+1} *is the projection of* x^k *onto the separating hyperplane [6]*

$$H^k := \{x : \ (d^k)^T (x - x^k) = \sum_{i=1}^q w_i^k \|d_i^k\|^2/2\} \tag{2.5}$$

that is a combination of the separating hyperplanes

$$H_i^k := \{x : \ (d_i^k)^T (x - x^k) = \|d_i^k\|^2/2\} \tag{2.6}$$

When the y_i^k are the exact projections onto each intermediate set S_i^k, the algorithm is modified [6] using a value of λ_k, that corresponds to the definition of x^{k+1} as the projection onto the separating hyperplane, combination of those arising from the exact projections onto each S_i^k.

Algorithm 2 *Parallel Algorithm using Exact Projections (EPA)[6] (k-th Iteration).*
Given $x^k \notin C$, $\theta \geq 0$, $0 < \eta \leq 1$.

- *Define* $J^k := \{j : \ a_j^T x^k - b_j > -\theta\}$, *and determines* $q_k = card(J^k)$.
  Define the sets $S_1^k, \ldots, S_q^k$ *according to (2.2).*

- *For* $i = 1, 2, \ldots, q$ *in parallel*
  Compute the projection of x^k *onto* S_i^k: $\ y_i{}^k$
  Define $d_i{}^k = y_i{}^k - x^k$
  End For.

- *Define* $d^k = \sum_{i=1}^{q} w_i{}^k d_i^k$, $w_i^k \geq 0$, $\sum_{i=1}^{q} w_i{}^k = 1$.
  Calculate

$$\lambda_k = \frac{\sum_{i=1}^{q} w_i^k \|d_i^k\|^2}{\|d^k\|^2}. \tag{2.7}$$

- *Define* $x^{k+1} = x^k + \omega_k \lambda_k d^k$, $\eta \leq \omega_k \leq 2 - \eta$. $\diamond$

Something characteristic of the PAM methods for linear systems, like those in [5] and [10], is the definition of $x^{k+1} = x^k + \lambda_k d^k$, with λ_k satisfying $\lambda_k = argmin_\lambda \|x^k + \lambda d^k - x^*\|^2$.

Remark 2 *If* x^* *is a solution to the system* $Ax \leq b$, *the ideal value of* λ_k *satisfying*

$$\min_\lambda \|x^k + \lambda d^k - x^*\|^2$$

requires to compute the solution of the problem

$$\min_\lambda \|x^k - x^*\|^2 - 2\lambda (d^k)^T (x^* - x^k) + \lambda^2 \|d^k\|^2 \tag{2.8}$$

by means of

$$\lambda_k = \frac{(d^k)^T (x^* - x^k)}{\|d^k\|^2}. \tag{2.9}$$

In general, the value of such an optimal λ *cannot be obtained by a practical formula. The formulas appearing in (2.4) and (2.7) used in [6], correspond to the related problem (2.8), avoid this difficulty.*

2.1 Algorithm DACCIM: Exact Projections

We consider a particular case of the EPA algorithm [6], fixing $\omega_k = 1$, $\theta = 0$, $card(J_i^k) = 1$, for all $i = 1, 2, \ldots, q_k$, where $q_k = card(J^k)$. Now, each set S_i^k (2.2) corresponds to a violated constraint in x^k.

The exact projection of x^k onto each halfspace S_i^k, $i = 1, 2, \ldots, q_k$, is easily calculated by $y_i^k = P_{C_{j(i)}}(x^k)$, if $j(i)$ is the original index of the corresponding inequality of the system (2.1).

From hereafter we denote by r_j^k the difference $b_j - a_j^T x^k$.

Remark 3 *In particular, when* $y_i^k = P_{C_{j(i)}}(x^k)$ *we get* $d_i^k = y_i^k - x^k = r_{j(i)}^k a_{j(i)}$, *being* $r_{j(i)}^k = b_{j(i)} - a_{j(i)}^T x^k < 0$, $i = 1, 2, \ldots, q_k$.

It is useful to point out that if x^* *is a solution to* $Ax \leq b$, $(d_i^k)^T (x^* - x^k) = (d_i^k)^T (x^* - y_i^k + y_i^k - x^k) \geq \|d_i^k\|^2$ *considering* $(d_i^k)^T (x^* - y_i^k) \geq 0$ *as a consequence of the convexity of* S_i^k *and the definition of* d_i^k.

From the assumptions made in this subsection, we can get the following results

Lemma 1 *Given* x^k, $J^k = \{j : \ b_j - a_j^T x^k < 0\}$, $q_k = card(J^k)$.
If $d^k = \sum_{i=1}^{q_k} w_i^k d_i^k$, $d_i^k = y_i^k - x^k$, $\sum_{i=1}^{q_k} w_i^k = 1$, $w_i^k = 1/q_k$, *then*

(i) For each $i = 1, \ldots, q_k$, $d_i^k = r_{j(i)}^k a_{j(i)}$ *where* $r_{j(i)}^k = b_{j(i)} - a_{j(i)}^T x^k$, *and* $j(i)$ *is the original index of the system of inequalities (2.1) corresponding to* i.

(ii) For each $i = 1, \ldots, q_k$, $r_{j(i)}^k = b_{j(i)} - a_{j(i)}^T x^k \geq a_{j(i)}^T x^* - a_{j(i)}^T x^k$, *if* x^* *is a solution to* $Ax \leq b$.

(iii) $(d^k)^T(x^* - x^k) = \sum_{i=1}^{q_k} w_i^k (d_i^k)^T (x^* - x^k) = \sum_{i=1}^{q_k} w_i^k r_{j(i)}^k a_{j(i)}^T (x^* - x^k)$,

(iv) $(d^k)^T(x^* - x^k) \geq \sum_{i=1}^{q_k} w_i^k (r_{j(i)}^k)^2 > 0$,

(v) If $x^{k+1} = x^k + \lambda d^k$, $\lambda > 0$ *then*

$$\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\lambda (d^k)^T (x^* - x^k) + \lambda^2 \|d^k\|^2 \leq \tag{2.10}$$

$$\|x^k - x^*\|^2 - 2\lambda \sum_{i=1}^{q_k} w_i^k (r_{j(i)}^k)^2 + (\lambda)^2 \|d^k\|^2 \tag{2.11}$$

(vi) If λ_k *is the argmin of (2.11), then*

$$\lambda_k = \frac{\sum_{i=1}^{q_k} w_i^k (r_{j(i)}^k)^2}{\|d^k\|^2}, \tag{2.12}$$

Furthermore

$$\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\lambda_k (d^k)^T (x^* - x^k) + (\lambda_k)^2 \|d^k\|^2, \tag{2.13}$$

satisfies

$$\|x^{k+1} - x^*\|^2 \leq \|x^k - x^*\|^2 - \alpha_k \tag{2.14}$$

where

$$\alpha_k = \lambda_k^2 \|d^k\|^2 = \frac{(\sum_{i=1}^{q_k} w_i^k (r_{j(i)}^k)^2)^2}{\|d^k\|^2} \tag{2.15}$$

(vii) $x^{k+1} = x^k + \lambda_k d^k$, *with* λ_k *given by (2.12) is the projection of* x^k *onto the hyperplane*

$$\{x : \ (d^k)^T (x - x^k) = \sum_{i=1}^{q_k} w_i^k (r_{j(i)}^k)^2\} \tag{2.16}$$

(viii) Moreover, (2.16) is the separating hyperplane of x^k with respect to C.

Proof. (i)-(iii) follow immediately from the definitions and the stated hypothesis. In order to prove (iv) we just have to take into account that for all $i = 1, 2, \ldots, q_k$, $r^k_{j(i)} < 0$, together with Remark 3 and (i)-(iii). By simple comparison (v) follows from (iv). The first part of (vi) follows directly by finding the argmin of problem (2.11).

The remaining results follow just replacing argmin in (2.11). For proving (vii) it is enough to replace x^{k+1} in (2.16). For proving (viii) consider the inequality given in (iv). That inequality shows that (2.16) is a separating hyperplane of C when $\sum_{i=1}^{q_k} w^k_i (r^k_{j(i)})^2 \neq 0$. □

Lemma 2 *Given x^k, if $d^k = \sum_{j \in J^k} w^k_j d^k_j$, where $d^k_j = P_{\mathcal{C}_j}(x^k) - x^k$ for all $j \in J^k$, and $x^{k+1} = x^k + \lambda_k d^k$, $\lambda_k = \frac{\sum_{j \in J^k} w^k_j \|d^k_j\|^2}{\|d^k\|^2}$, then the sequence $\{x^k\}$ satisfies*

$$\|x^{k+1} - x^*\|^2 \leq \|x^k - x^*\|^2 - \alpha_k, \text{ where} \tag{2.17}$$

$$\alpha_k = \lambda_k^2 \|d^k\|^2 = (\sum_{j \in J^k} w^k_j \|d^k_j\|^2)^2 / \|d^k\|^2. \tag{2.18}$$

Proof. It follows from (vi) of the previous Lemma. □

The following results are needed for justifying the acceleration scheme that will be applied to the EPA algorithm with unitary blocks.

Lemma 3 *Given x^k, $J_k = \{j : b_j - a_j^T x^k < 0\}$, d^k, λ_k and x^{k+1} as defined in Lemma 2, then*

(i) $(x^* - x^{k+1})^T d^k \geq 0$.

(ii) $(x^* - x^{k+1})^T d^k \geq \sum_{j \in J_k} w^k_j r^k_j r^{k+1}_j$.

(iii) $r^{k+1}_j = r^k_j - \lambda_k a_j^T d^k$, $j \in J^k$.

(iv) $\sum_{j \in J_k} w^k_j r^k_j r^{k+1}_j = 0$.

Proof. Using the definition of x^{k+1} and λ_k in $(x^* - x^{k+1})^T d^k$ (i) follows. Due to the fact that $(x^* - x^{k+1})^T d^k = \sum_{j \in J_k} w^k_j r^k_j a_j^T (x^* - x^{k+1})$, and considering $a_j^T x^* \leq b_j$, we get $(x^* - x^{k+1})^T d^k \geq \sum_{j \in J_k} w^k_j r^k_j (b_j - a_j^T x^{k+1}) = \sum_{j \in J_k} w^k_j r^k_j r^{k+1}_j$.
To prove (iv) we substitute (iii) in $\sum_{j \in J_k} w^k_j r^k_j r^{k+1}_j$. Then, replacing the expression of λ_k in $\sum_{j \in J_k} w^k_j (r^k_j)^2 - \lambda_k (\sum_{j \in J_k} w^k_j r^k_j a_j{}^T d^k)$ we obtain $\sum_{j \in J_k} w^k_j r^k_j r^{k+1}_j = 0$. □

Lemma 4 *Given x^k, consider J^k, d^k, λ_k and x^{k+1} as defined in Lemma 2. If $J^{k+1} := \{j : a_j^T x^{k+1} > b_j\}$, and $J_1^{k+1} := \{j : j \in J^{k+1}, a_j^T x^k \leq b_j\}$, $J_2^{k+1} := \{j : j \in J^k, a_j^T x^{k+1} > b_j\}$, then*

(i) For all $j \in J_1^{k+1}$, $a_j^T d^k > 0$.

(ii)If $t_1^{k+1} := \sum_{j \in J_1^{k+1}} w_j^{k+1} r_j^{k+1} a_j^T$, then $(t_1^{k+1})^T d^k < 0$.

(iii) If $t_2^{k+1} := \sum_{j \in J_2^{k+1}} w_j^{k+1} r_j^{k+1} a_j^T$, the sign of $(t_2^{k+1})^T d^k$ tends to be negative when the negative residuals of the constraints increase in absolute value. On the other hand, it tends to be positive when $|r_j^{k+1}| < |r_j^k|$.

Proof. To prove (i) we consider $j \in J_1^{k+1}$, $r_j^{k+1} = r_j^k - \lambda_k a_j^T d^k < 0$, y $r_j^k \geq 0$. Then, it follows that $a_j^T d^k > 0$. As a consequence of (i) $t_1^{k+1} := \sum_{j \in J_1^{k+1}} w_j^{k+1} r_j^{k+1} a_j^T$, satisfies $(t_1^{k+1})^T d^k < 0$. For all $j \in J_2^{k+1}$, we get $r_j^{k+1} < 0$ and $r_j^k < 0$. In that case, the sign of $r_j^{k+1} - r_j^k = -\lambda_k a_j^T d^k$, depends on the sign of $a_j^T d^k$. Thus, if $a_j^T d^k < 0$ then $|r_j^{k+1}| < |r_j^k|$, while if $a_j^T d^k > 0$ the opposite holds. Therefore, since $t_2^{k+1} = \sum_{j \in J_2^{k+1}} w_j^{k+1} r_j^{k+1} a_j^T$, the sign of $(t_2^{k+1})^T d^k$ can be either negative or positive depending on the absolute values of the negative residuals. □

From Lemma 4 we can infer the direction d^{k+1}, that combines the exact projections to the violated constraints at x^{k+1}, may satisfy $(d^{k+1})^T d^k < 0$. Such property has been observed in different numerical experiences in almost all iterations. Considering (ii) and (iii) from the previous Lemma, we see that such a situation is possible due to the zigzagging appearing when non-violated constraints in a given iteration are violated in the next, and the residuals of those constraints that remain violated ($|r_j^{k+1}| \simeq |r_j^k|$) do not decrease in a sensible way.

Moreover, if $(d^{k+1})^T d^k < 0$, then the next iterate along the direction d^{k+1} will lie outside of the convex region defined by the separating hyperplane (2.16) which contains the current point x^{k+1}. This observation led us to define a new algorithm such that the defined direction takes into account that property.

Lemma 5 *Given x^{k+1}, if $\sigma = (d^{k+1})^T v < 0$, and $v = d^k$, then the direction $\tilde{d}^{k+1} := P_{v^\perp} d^{k+1}$ satisfies*

(i) $\tilde{d}^{k+1} = d^{k+1} - \frac{(d^{k+1})^T d^k)}{\|d^k\|^2} d^k$.

(ii) $\tilde{d}^{k+1}(x^* - x^{k+1}) \geq (d^{k+1})^T (x^* - x^{k+1}) > 0$.

(iii) $\|P_{v^\perp} d^{k+1}\|^2 < \|d^{k+1}\|^2$.

(iv) $\|P_{v^\perp} d^{k+1}\| \neq 0$

Proof. From the definition of $\tilde{d}^{k+1}$, (i)follows. Multiplying (i) by $x^* - x^{k+1}$, and considering the result of Lemma 3 (i), we get $(\tilde{d}^{k+1})^T(x^* - x^{k+1}) \geq (d^{k+1})^T(x^* - x^{k+1})$. Then, since $(d^{k+1})^T(x^* - x^{k+1}) \geq \sum_{i=1}^{q_{k+1}} w_i^{k+1}(r_{j(i)}^{k+1})^2 > 0$, we obtain (ii). The results (iii) and (iv) follow directly from (i) and (ii) respectively. □

Lemma 6 *Given x^k, $v = x^k - x^{k-1}$, consider J^k, d^k as defined in Lemma 2. If $(d^k)^T v < 0$, and $\tilde{x}^{k+1} = x^k + \tilde{\lambda}_k \tilde{d}^k$, with $\tilde{d}^k$ defined as in Lemma 5, and*

$$\tilde{\lambda}_k = \frac{\sum_{i=1}^{q_k} w_i^k (r_i^k)^2}{\|\tilde{d}^k\|^2}, \tag{2.19}$$

then

$$\|\tilde{x}^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\tilde{\lambda}_k(\tilde{d}^k)^T(x^* - x^k) + \tilde{\lambda_k}^2\|\tilde{d}^k\|^2$$

satisfies

$$\|\tilde{x}^{k+1} - x^*\|^2 < \|x^{k+1} - x^*\|^2 \tag{2.20}$$

where $x^{k+1} = x^k + \lambda_k d^k$, and λ_k defined in (2.12).

Proof. To derive the inequality (2.20) it is enough to consider the definitions of $\tilde{x}^{k+1}$, x^{k+1} and their distances to x^*,

$\|\tilde{x}^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\tilde{\lambda}_k(\tilde{d}^k)^T(x^* - x^k) + \tilde{\lambda}_k^2\|\tilde{d}^k\|^2$, where $\tilde{\lambda}_k = \frac{\sum_{i=1}^{q_k} w_i^k (r_i^k)^2}{\|\tilde{d}^k\|^2}$,

$\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\lambda_k(d^k)^T(x^* - x^k) + \lambda_k^2\|d^k\|^2$, with $\lambda_k = \frac{\sum_{i=1}^{q_k} w_i^k (r_i^k)^2}{\|d^k\|^2}$.

The difference $\|x^{k+1} - x^*\|^2 - \|\tilde{x}^{k+1} - x^*\|^2$ coincides with

$$[-2\lambda_k(d^k)^T(x^* - x^k) + \lambda_k{}^2\|d^k\|^2] - [-2\tilde{\lambda}_k(\tilde{d}^k)^T(x^* - x^k) + \tilde{\lambda}_k^2\|\tilde{d}^k\|^2].$$

Reordering this expression we get

$$[2\frac{\sum_{i=1}^{q_k} w_i^k (r_i^k)^2}{\|\tilde{d}^k\|^2}(\tilde{d}^k)^T(x^* - x^k) - 2\frac{\sum_{i=1}^{q_k} w_i^k (r_i^k)^2}{\|d^k\|^2}(d^k)^T(x^* - x^k)] -$$

$$[\frac{(\sum_{i=1}^{q_k} w_i^k (r_i^k)^2)^2}{\|\tilde{d}^k\|^2} - \frac{(\sum_{i=1}^{q_k} w_i^k (r_i^k)^2)^2}{\|d^k\|^2}]$$

The first bracket

$$[2\frac{\sum_{i=1}^{q_k} w_i^k (r_i^k)^2}{\|\tilde{d}^k\|^2}(\tilde{d}^k)^T(x^* - x^k) - 2\frac{\sum_{i=1}^{q_k} w_i^k (r_i^k)^2}{\|d^k\|^2}(d^k)^T(x^* - x^k)] \geq$$

$c_1[\frac{1}{||\tilde{d}^k||^2} - \frac{1}{||d^k||^2}]$, where $c_1 = 2(d^k)^T(x^* - x^k)\sum_{i=1}^{q_k} w_i^k (r_i^k)^2$.
The second bracket coincides with
$c_2[\frac{1}{||\tilde{d}^k||^2} - \frac{1}{||d^k||^2}]$, where $c_2 = (\sum_{i=1}^{q_k} w_i^k (r_i^k)^2)^2$. Hence, since $c_1 \geq 2c_2$, $c_2 > 0$ and $\frac{1}{||\tilde{d}^k||^2} - \frac{1}{||d^k||^2} > 0$, we get (2.20). □

Now, we have the necessary results for presenting the new algorithm. Due to the hypotheses of Lemma 6, the iterate x^{k+1} is defined along the direction $\tilde{d}^k = P_{v^\perp}(\sum_{i=1}^{q_k} w_i^k d_i^k)$, where v is the direction $\tilde{d}^{k-1}$ from the previous iteration.

Given x^k, let us consider J^k, $card(J^k) = q_k$. We will denote by Q_k the projector onto the orthogonal subspace to $v = x^k - x^{k-1}$. In particular, $Q_0 = I_n$, where I_n is the identity matrix. The following scheme describes the iterative step ($k > 0$) of the new algorithm, in a version not adapted to parallel processing.

Algorithm 3 *DACCIM (k-th iteration):*

Given x^k, J_k, Q_k, $v = \tilde{d}^{k-1}$.

- *For* $i = 1, \ldots, q_k$ *do*
  Compute $y_i^k = P_{\mathcal{C}_{j(i)}}(x^k)$.
  Define $d_i^k = y_i^k - x^k$.
  End For.
- Define $d^k = \sum_{i=1}^{q_k} w_i^k d_i^k$,
- Compute $\sigma = v^T d^k$
  If $\sigma < 0$, define $\tilde{d}^k = Q_k(d^k)$, else $\tilde{d}^k = d^k$.
- Compute $x^{k+1} = x^k + \tilde{\lambda}_k \tilde{d}^k$, $\tilde{\lambda}_k$ defined by (2.19). ◇

This algorithm can be easily adapted to parallel multiprocessing.
The following Lemma proves that algorithm DACCIM is well defined.

Lemma 7 *Given* x^k, $x^k \neq x^*$, *the new direction* $\tilde{d}^k$ *and* $\tilde{\lambda}_k$ *as in (2.19) are well defined.*

Proof. It is of interest to consider the case $v^T d^k < 0$, $v = x^k - x^{k-1}$. Taking into account (ii) and (iii) of Lemma 5 it follows immediately that $\tilde{d}^k \neq 0$ and therefore, $\tilde{\lambda}_k$ is well defined. □

Lemma 8 *The sequence* $\{x^k\}$ *generated by DACCIM satisfies* $\|x^{k+1} - x^*\|^2 \leq \|x^k - x^*\|^2 - \tilde{\alpha}_k$, *where*

$$\tilde{\alpha}_k = (\tilde{\lambda}_k)^2 \|\tilde{d}^k\|^2 = \frac{(\sum_{i=1}^{q_k} w_i^k \|d_i^k\|^2)^2}{\|\tilde{d}^k\|^2}, \tag{2.21}$$

satisfying $\tilde{\alpha}_k \geq \alpha_k$, α_k *given in (2.18).*

Proof. The result follows using the expressions of $\tilde{\lambda}_k$, the Lemmas 2 and 6. □

2.1.1 Convergence

The convergence of the sequence generated by the DACCIM algorithm to a solution x^* is proved applying the theory developed by Gubin et al. in [7], taking into account that the sequence $\{\|x^k - x^*\|\}$ decreases, Lemma 6 and 8.

2.2 Accelerated incomplete projections algorithm (ACIPA)

In this subsection we apply the acceleration scheme to the ACIPA algorithm, by using incomplete projections onto blocks of inequalities as in the IPA method described in [6]. To compute the approximate projection onto each S_i^k, we use the DACCIM algorithm with a stopping condition such that it finds a y_i^k satisfying the condition (2.3) (one iteration guarantees this).

As a consequence of the procedure to find y_i^k, the next result justifies that $\tilde{H}_i^k$ is a deeper separating hyperplane the one given in [6].

Denoting by z^j the intermediate iterates of DACCIM until obtaining y_i^k, and using $z^0 = x^k$, we get

Lemma 9 *Given* y_i^k, $x^* \in C$. *If* $y_i^k - x^k = \sum_{j=1}^{j_i}(z^j - z^{j-1})$, *then*

(i) $\gamma_i = \sum_{j=1}^{j_i} \|z^j - z^{j-1}\|^2 > 0$,

(ii) $\|y_i^k - x^*\|^2 \leq \|x^* - x^k\|^2 - \sum_{j=1}^{j_i} \|z^j - z^{j-1}\|^2$

(iii) $(y_i^k - x^k)^T(x^* - x^k) \geq (\|y_i^k - x^k\|^2 + \gamma_i)/2$, *with* $\gamma_i > 0$

Proof. The first assertion follows immediately because if x^k is unfeasible, at least one iteration of DACCIM is performed and, therefore, we obtain the result given in (2.17). According to Lemma 2 each intermediate z^j satisfies $\|z^j - x^*\|^2 \leq \|x^* - z^{j-1}\|^2 - \|z^j - z^{j-1}\|^2$, hence obtaining (ii). Moreover, since $\|y_i^k - x^*\|^2 = \|x^k - x^*\|^2 - 2(y_i^k - x^*)^T(x^* - x^k) + \|y_i^k - x^k\|^2$ we derive (iii) using the inequality (ii). □

This implies that the hyperplane

$$\tilde{H}_i^k = \{x : \ (y_i^k - x^k)^T(x - x^k) = (\|y_i^k - x^k\|^2 + \gamma_i)/2\}$$

is deeper than H_i^k given in (2.6) and introduced in [6]. Likewise, the hyperplane generated from the convex combination of the previous ones

$$\tilde{H}^k = \{x : (d^k)^T(x - x^k) = \sum_{i=1}^{q} w_i^k(\|d_i^k\|^2 + \gamma_i)/2\} \tag{2.22}$$

where $d_i^k = y_i^k - x^k$, $d^k = \sum_{i=1}^q w_i^k d_i^k$, shares the same property when compared to (2.5).

In principle, as in the IPA algorithm, the new iterate will be obtained projecting x^k onto the deeper separating hyperplane $\tilde{H}^k$. Taking into account x^k is on the separating hyperplane $\tilde{H}^{k-1}$ (being $v = x^k - x^{k-1}$ the normal vector), when the new direction d^k satisfies $(d^k)^T v < 0$ and leads to a point exterior to the halfspace limited by $\tilde{H}^{k-1}$ containing C, the direction will be modified by projecting it onto the "correct" region. Such a modification is identical to the one proposed in the DACCIM algorithm when dealing with the same situation.

Therefore, if v is the direction at step $x^k - x^{k-1}$, and d^k satisfies $(d^k)^T v < 0$, we define $\tilde{d}^k := P_{v^\perp} d^k$, and $\tilde{x}^{k+1} = x^k + \tilde{\lambda}_k \tilde{d}^k$, being

$$\tilde{\lambda}_k = \frac{\sum_{i=1}^q w_i^k(\|d_i^k\|^2 + \gamma_i)}{2\|\tilde{d}^k\|^2} \tag{2.23}$$

the argmin of the problem

$$\|x^k - x^*\|^2 - 2\lambda \sum_{i=1}^{q} w_i^k(\|d_i^k\|^2 + \gamma_i)/2 + (\lambda)^2\|\tilde{d}^k\|^2 \tag{2.24}$$

that is an upper bound of the one given in (2.8), now using the new direction $\tilde{d}^k$.

We describe in the following the iterative step of the incomplete projections algorithm ACIPA.
Given x^k, denoting by Q_k the projector onto the orthogonal subspace to the one defined by the previous direction $v = x^k - x^{k-1}$, and defining $Q_0 = I_n$.

Algorithm 4 *ACIPA (k-th iteration).*
Given $x^k \notin C$, $v = \tilde{d}^{k-1}$, $0 < \theta < 0.1$, $0 < \eta \leq 1$.

- *Define* $J_k = \{j : \ a_j^T x^k - b_j \geq \ -\theta\}$, *and* S_i^k, $i = 1, 2, \ldots, q$.

- *For* $i = 1, 2, \ldots, q$ *in parallel*

  Compute y_i^k *using Algorithm DACCIM, and compute* γ_i.

  Define $d_i{}^k = y_i{}^k - x^k$.

  End For;

- *Define* $w_i^k \geq 0$, *such that* $\sum_{i=1}^q w_i^k = 1$.

  Define $d^k = \sum_{i=1}^q w_i^k d_i^k$, *and compute* $\sigma = (d^k)^T v$,

  Define $\tilde{d}^k = Q_k(d^k)$ *if* $\sigma < 0$, *else* $\tilde{d}^k = d^k$.

  Compute $\tilde{\lambda}_k$ *given in (2.23).*

- *Define* $x^{k+1} = x^k + \omega_k \tilde{\lambda}_k \tilde{d}^k$, *with* $\eta \leq \omega_k \leq 2 - \eta$. $\diamond$

We are now going to prove the algorithm is well defined and later on the convergence results.

Lemma 10 *Given* x^k, $x^k \neq x^*$, $x^* \in C$, *the direction* $\tilde{d}^k$ *of ACIPA is well defined and satisfies*

(i) $(\tilde{d}^k)^T(x^* - x^k) \geq (d^k)^T(x^* - x^k) > 0$, *being* $d^k = \sum_{i=1}^q w_i^k d_i^k$.

(ii) $\|\tilde{d}^k\| < \|d^k\|$, *if* $d^k = \sum_{i=1}^q w_i^k d_i^k$ *satisfies* $v^T d^k < 0$, *where* $v = \tilde{d}^{k-1}$, $k \geq 1$.

Proof. For $k = 0$, $\tilde{d}^0$ coincides with the direction given by the IPA algorithm.

Assuming the direction $\tilde{d}^{k-1}$, $k > 0$, is well defined, it is interesting to analyze the case when $\sigma = (d^k)^T \tilde{d}^{k-1} < 0$. According to the definition, $\tilde{d}^k$ satisfies $\tilde{d}^k = d^k - \sigma \frac{v}{\|v\|^2}$. Thus, multiplying both sides by $(x^* - x^k)$, and using $(\tilde{d}^{k-1})^T(x^* - x^k) \geq 0$ due to the definition of x^k, we get $(\tilde{d}^k)^T(x^* - x^k) = (d^k)^T(x^* - x^k) - \frac{v^T d^k}{(\|v\|)^2}(v^T(x^* - x^k)) \geq (d^k)^T(x^* - x^k) > 0$. From this inequality we derive (i) and also that $\|\tilde{d}^k\| > 0$. Therefore, the direction is well defined in the special case when its definition modifies the one of the IPA algorithm.

On the other hand, and for the same special case, the following holds
$\|\tilde{d}^k\|^2 = \|d^k\|^2 - (\sigma^2)/\|v\|^2$. Thus, $\tilde{d}^k$ satisfies (ii). $\square$

The differences between ACIPA and IPA [6] are the following:
1) we calculate y_i^k explicitly (algorithm DACCIM);
2) x^{k+1} is the projection of x^k onto a deeper separating hyperplane;
3) we preserve the fact that the new iterate does not fall outside of the region defined by the last separating hyperplane.

In order to theoretically analyze the comparative behaviour of the sequences generated by the algorithms ACIPA and IPA we proceed in two stages. First, we compare the step given by ACIPA with the one obtained by an algorithm using the same approximate projections y_i^k, but for defining x^{k+1} it uses directly d^k without projections. Second, we compare such a sequence with that obtained by the original IPA algorithm.

For comparison purposes we will denote by $\tilde{x}^{k+1}$ the ACIPA iterate and by $\underline{x}^{k+1}$ the one given by IPA.

Lemma 11 *Given x^k, y_i^k , $i = 1, 2, \ldots, q$, d^k, $\tilde{d}^k$ and $\tilde{\lambda}_k$ defined in ACIPA. If $\sigma = (d^k)^T \tilde{d}^{k-1} < 0$, and $\tilde{x}^{k+1} = x^k + \tilde{\lambda}_k \tilde{d}^k$, then*

(i) $\|\tilde{x}^{k+1} - x^*\| < \|x^{k+1} - x^*\|$ *if* $x^{k+1} = x^k + \lambda_k d^k$, *and* $\lambda_k = \frac{\sum_{i=1}^{p} w_i^k (\|d_i^k\|^2 + \gamma_i)}{2\|d^k\|^2}$

Furthermore,

(ii) If x^{k+1} is defined as in (i), then

$\|x^{k+1} - x^*\| < \|\underline{x}^{k+1} - x^*\|$ *when $\underline{x}^{k+1}$ is obtained using the original λ_k of [6] explicitly stated in (2.4):*

$\underline{\lambda}_k := \frac{\sum_{i=1}^{q} w_i^k (\|d_i^k\|^2)}{2\|d^k\|^2}$

Proof. To prove the inequality (i) let us consider,

$\|\tilde{x}^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\tilde{\lambda}_k (\tilde{d}^k)^T (x^* - x^k) + \tilde{\lambda}_k^2 \|\tilde{d}^k\|^2$, replacing $\tilde{\lambda}_k$ by the expression given in (2.23).

Analogously, replacing the expression of λ_k in $\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\lambda_k (d^k)^T (x^* - x^k) + \lambda_k^2 \|d^k\|^2$.

The difference $\|x^{k+1} - x^*\|^2 - \|\tilde{x}^{k+1} - x^*\|^2$ coincides with

$$[-2\lambda_k (d^k)^T (x^* - x^k) + \lambda_k^2 \|d^k\|^2] - [-2\tilde{\lambda}_k (\tilde{d}^k)^T (x^* - x^k) + \tilde{\lambda}_k^2 \|\tilde{d}^k\|^2].$$

Reordering as in Lemma 6 we observe that the involved expressions are similar to those appearing there, except by their numerators λ_k and $\tilde{\lambda}_k$, but they have no influence on the comparison we are interested in. Hence, repeating the steps of Lemma 6 we prove (i).

In order to get (ii) we consider as in (i)

$\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\lambda_k (d^k)^T (x^* - x^k) + \lambda_k^2 \|d^k\|^2$, replacing the expression corresponding to λ_k. Also $\|\underline{x}^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\underline{\lambda}_k (d^k)^T (x^* - x^k) + \|d^k\|^2$, and taking into account that the value of $\underline{\lambda}_k$ is the one given in (2.4).

It is easy to see the difference $\|\underline{x}^{k+1} - x^*\|^2 - \|x^{k+1} - x^*\|^2$, is equal to

$$[2(\lambda_k - \underline{\lambda}_k) d^k (x^* - x^k)] - [(\lambda_k^2 - \underline{\lambda}_k^2) \|d^k\|^2]$$

and using $(d^k)^T (x^* - x^k) \geq \lambda_k \|d^k\|^2$, $\lambda_k - \underline{\lambda}_k > 0$, (ii) follows. □

Lemma 12 *If* $\tilde{d}^k$ *is defined as in the ACIPA algorithm,* $x^{k+1} = x^k + w_k \tilde{\lambda}_k \tilde{d}^k$, *with* $\tilde{\lambda}_k$ *given in (2.23),* $\eta \leq w_k \leq 2 - \eta$ *then the generated sequence* $\{x^k\}$ *satisfies*

$$\|x^{k+1} - x^*\|^2 \leq \|x^k - x^*\|^2 - \tilde{\alpha}_k, \tag{2.25}$$

where

$$\tilde{\alpha}_k = w_k(2 - w_k)(\sum_{i=1}^{p} w_i^k(\|d_i^k\|^2 + \gamma_i))^2/(4\|\tilde{d}^k\|^2) \tag{2.26}$$

where d_i^k *and* γ_i *are those from ACIPA.*

Proof. The result follows from the definition of x^{k+1}, $\tilde{d}^k$ and that of $\tilde{\lambda}_k$ given in (2.23) . □

2.2.1 Convergence

The convergence of the sequence generated by the ACIPA algorithm is a consequence of the comparisons made in Lemma 11 with the IPA algorithm, together with Theorem 1 in [6] and Lemma 12.

3 Numerical experiences

The first purpose of the numerical experiences is to illustrate the behaviour of the acceleration of DACCIM (Algorithm 3), using exact projections onto the violated constraints, in comparison to the EPA method [6]. For that purpose a version of the EPA method was implemented, called EPACIM, using $card(J_i^k) = 1$, $w_i^k = 1/q_k$ if the number of violated inequalities in x^k, whose indexes are stored in $J^k = \{j : a_j^T x^k - b_j > 0\}$, is q_k. We briefly describe this version as follows:
EPACIM:Given x^k, J^k, $q_k = card(J^k)$ *(as in DACCIM), define*
$d^k = \sum_{i=1}^{q_k} w_i^k r_{j(i)}^k a_{j(i)}$ *where* $w_i^k = \frac{1}{q_k}$.
The new iterate is $x^{k+1} = x^k + \lambda_k d^k$, *with* λ_k *as in (2.12).*

The second purpose is to compare the results of ACIPA (Algorithm 4), that uses incomplete projections onto blocks of violated inequalities, with those obtained with a version of the IPA algorithm [6] called IPACIM. In both algorithms the computation of the approximate projections y_i^k is similarly done by means of the procedure of DACCIM, accepting an approximation if the condition described below is satisfied. Hence, in this version of IPA the convex combination $d^k = \sum_{i=1}^{q} w_i{}^k d_i{}^k$ where $d_i{}^k = y_i^k - x^k$ for $i = 1, 2, \ldots, q$, is similar to the one used in ACIPA.

Now, we will describe briefly the algorithms being compared, using an experimental code.

IPACIM (Incomplete projections algorithm: Given an iterate x^k, J^k, S_i^k, y_i^k *for each* $i = 1, 2, \ldots, q$, *we define the direction*

$d^k = \sum_{i=1}^{q} w_i{}^k d_i{}^k$ where $d_i^k = y_i^k - x^k$ for $i = 1, 2, \ldots, q$,

Define $x^{k+1} = x^k + \lambda_k d^k$, where λ_k is the original one of [6] as given in (2.4).

ACIPA (Accelerated incomplete projections algorithm: Given an iterate x^k, $\tilde{d}^{k-1}$, Q_k, J^k, S_i^k, y_i^k, $i = 1, 2, \ldots, q$, *we define the direction* d^k as in IPACIM,

Define $\tilde{d}^k = Q_k(d^k)$ if $(d^k)^T \tilde{d}^{k-1} < 0$, otherwise $\tilde{d}^k = d^k$,

Define $x^{k+1} = x^k + \tilde{\lambda}_k \tilde{d}^k$, where $\tilde{\lambda}_k$ is the one defined in (2.23).

In the implementations of ACIPA and IPACIM we consider:

(i) *A constraint is violated* at x^k if $a_j^T x^k - b_j \geq -(5 * 10^{-5})$. The set J^k of violated constraints is in principle splitted into $q = 4$ blocks of equal cardinality, adding if necessary the remaining inequalities to the fourth block.

(ii) *Incomplete projection onto each block:* We compute it by means of the DACCIM algorithm.

Given $x^k \in \Re^n$, and a block S_i^k, from $z^0 = x^k$ we compute $z^1, z^2, \ldots$ while none of the following conditions is satisfied:

(1) $r(z^l) < 10^{-2} * r(z^0)$, where $r(z^l) = \max_{j \in S_i^k}(0,\ a_j^T(z^l) - b_j)$ is not satisfied, or
(2) $\|z^l - z^{l-1}\| < 10^{-4}\|z^1 - z^0\|$, or
(3) the maximum number of allowed iterations (15) has not been reached.

Test problems.

We have run different problems of the type $Ax \leq b$, where the matrix $A \in \Re^{m \times n}$ has been chosen to reflect a variety of condition numbers and/or sparsity patterns. For that purpose we used the Zlatev routine from SPARSKIT2/Library [9] (www.cs.umn.edu/Research/darpa/SPARSKIT).

Another set of problems has been obtained randomly using different sparsity patterns according to predefined densities. More precisely, the indices of nonnull entries were generated randomly, as well as the corresponding matrix values. Approximately *density* $* m*n$ entries of the matrix will be nonnull. After generating the matrix, the code computes

the independent term b in such a way that $Ax \leq b$ is compatible. The initial approximation used was a vector x^0 whose components were zero.

Numerical results

The problems were run on a PC Pentium III, 800MHz, with 256 Mb Ram and 128 Mb Swap using FORTRAN 90.

The stopping criteria were:

If $Rm_k \leq 10^{-6} * \max\{1,\ Rm_0\}$, where $Rm_k = \max_{i=1,2,...,m}(0,\ a_i^T x^k - b_i)$, or if the number of iterations reaches the maximum allowed $ITMAX$, with $ITMAX = 5000$.

The obtained results are presented in the following Tables using the notation:

- **Iter:** Number of performed iterations.
- **$\mathbf{Rm_s}$:** $\max_{i=1,2,...,m}(0, a_i^T x^s - b_i)$, x^s being the iterate satisfying the stopping criteria.
- **CPU:** time measured in seconds.

The starting point was $x^0 = 0$.

In Table 1-2 we present the results obtained with the algorithms DACCIM and EPACIM for the problems of SPARSKIT2 Library derived from Zlatev's matrices. These matrices have been generated with different condition depend on the parameter α. The dimensions were $m = 12000$, $n = 10000$, and the problems were run with $\alpha = 2^i$, $i = 2, 4, 8, 12$, denoting each problem by Zla(α) according to the sort of conditioning with which the matrix was generated. In Table 1 results obtained using $index = 20$, a parameter that indicates the average number of non-zero elements per row are given, while in Table 2 the values correspond to $index = 100$.

Table 1: Zlatev's matrices, $m = 12000$, $n = 10000$

$index = 20$	Meth	Iter	Rm_s	CPU
	DACCIM	163	8.9d-7	2.8
$Zla(2^2)$	EPACIM	858	9.2d-7	19.5
	DACCIM	231	9.9d-7	3.8
$Zla(2^4)$	EPACIM	888	9.7d-7	20.0
	DACCIM	182	7.33d-7	3.1
$Zla(2^8)$	EPACIM	882	9.6d-7	20.6
	DACCIM	344	8.5d-7	5.4
$Zla(2^{12})$	EPACIM	896	9.7d-7	20.3

Table 2: Zlatev's matrices, $m = 12000$, $n = 10000$

$index = 100$	Meth	Iter	Rm_s	CPU
	DACCIM	1064	9.9d-7	82.9
$Zla(2^2)$	EPACIM	5000	1.3d-2	589.5
	DACCIM	1130	7.8d-7	119.0
$Zla(2^4)$	EPACIM	5000	9.4d-3	631.1
	DACCIM	2070	9.9d-7	227.2
$Zla(2^8)$	EPACIM	5000	1.2d-2	813.8
	DACCIM	1094	7.8d-7	126.6
$Zla(2^{12})$	EPACIM	5000	6.6d-3	518.7

These results show the effect of the acceleration scheme of the algorithm DACCIM in regard to the algorithm EPACIM, because the number of required iterations and the CPU time are drastically reduced in all problems.

Table 3: Random's matrices, density=1.. Average timing (sec.)

Meth	m	800	400	200	100
	n	200	100	50	25
DACCIM		0.44 (32)	0.09(28)	0.01(35)	0(37)
EPACIM		1.11(75)	0.22(63)	0.03(82)	0.01(105)

Tabla 3 shows a comparison of the DACCIM algorithm with the EPACIM version of EPA [6] using a set of dense randomly generated problems.

Results are given indicating the row and column dimensions m, n of the randomly generated problems with density 1. The reported CPU time is the average of solving each problem five times. Between parenthesis the required number of iterations for satisfying the stopping criteria as in Tables 1-2 are given.

In the next tables we report the results obtained with the ACIPA algorithm, using incomplete projections onto the violated constraints blocks, and IPACIM, the implemented version of IPA [6] as described at the beginning of this section. The key difference between them is the definition of x^{k+1}.

In the Tables 4-6 we present the results corresponding to the Zlatev matrices [9] considering different sparsity patterns defined by *index* in each case.

The results of Tables 4-6 show the efectiveness of the direction $\tilde{d}^k$ and the parameter

Table 4: Zlatev's matrices, $m = 12000$, $n = 10000$

$index = 20$	Meth	Iter	Rm_s	CPU
	ACIPA	59	4.2d-7	8.5
$Zla(2^2)$	IPACIM	115	9.3d-7	22.7
	ACIPA	82	7.6d-7	9.4
$Zla(2^4)$	IPACIM	71	4.3d-7	18.7
	ACIPA	71	9.9d-7	9.0
$Zla(2^8)$	IPACIM	898	9.9d-7	136.6
	ACIPA	75	7.4d-7	9.6
$Zla(2^{12})$	IPACIM	92	8.9d-7	20.9

Table 5: Zlatev's matrices, $m = 12000$, $n = 10000$

$index = 60$	Meth	Iter	Rm_s	CPU
	ACIPA	73	9.4d-7	22.2
$Zla(2^2)$	IPACIM	239	8.7d-7	154.6
	ACIPA	102	4.8d-7	25.6
$Zla(2^4)$	IPACIM	263	8.5d-7	154.7
	ACIPA	127	2.2d-7	27.7
$Zla(2^8)$	IPACIM	241	9.1d-7	149.8
	ACIPA	72	5.4d-7	22.7
$Zla(2^{12})$	IPACIM	247	8.2d-7	150.9

$\tilde{\lambda}_k$ used in the ACIPA algorithm. In particular, such a performance is more evident in Table 6, corresponding to more dense problems.

It is also worthwhile to point out that Table 4 shows that the number of iterations can be similar, while the CPU time is not. This means that the cardinality of the blocks corresponding to the violated constraints in IPACIM is greater than those in ACIPA. In other words, ACIPA generates better iterates as predicted by the theoretical results. Moreover, the version of IPA that we called IPACIM uses the same procedure DACCIM as in ACIPA for finding the approximations y_i^k to the blocks. Thus, the observed differences in the numerical results clearly arise from the different definitions of x^{k+1}.

We include in the next Table 7 the results obtained with ACIPA and IPACIM for the randomly generated problems with dimension $m = 4000$, $n = 2000$, and density 1. The reported CPU time corresponds to the average of five runs of each problem.

Table 6: Zlatev's matrices, $m = 12000$, $n = 10000$

$index = 100$	Meth	Iter	Rm_s	CPU
	ACIPA	118	8.9d-7	49.0
$Zla(2^2)$	IPACIM	537	7.2d-7	577.5
	ACIPA	147	9.6d-7	55.6
$Zla(2^4)$	IPACIM	637	8.4d-7	645.0
	ACIPA	166	4.9d-7	55.2
$Zla(2^8)$	IPACIM	568	5.5d-7	529.8
	ACIPA	216	9.8d-7	63.8
$Zla(2^{12})$	IPACIM	551	9.6d-7	560.2

Table 7: Random matrix, density =1.

m	n	Meth	Iter	Rm_s	CPU
4000	2000	ACIPA	245	1.5d-5	474.2
		IPACIM	5000	6.0d-3	8698.2

Finally, we run the same problems of Table 7 using a density of 0.1 and different dimensions as shown in Table 8 using the ACIPA and IPACIM algorithms.

Table 8: Random's matrices, $density = 0.1$

m	n	Meth	Iter	Rm_s	CPU
7500	2500	ACIPA	73	2.2d-6	71.6
		IPACIM	654	2.4 d-6	446.9
9500	2000	ACIPA	27	1.7d-6	30.9
		IPACIM	106	1.5d-6	85.5
10000	1900	ACIPA	23	1.5d-6	27.4
		IPACIM	74	1.2 d-6	62.3

In all Tables we observe the efficiency of the new algorithms that include the projection onto the half-space defined by the separating hyperplane defined in (2.22).

4 Conclusions

The DACCIM algorithm is an extension to linear inequalities problems of the ACIMM algorithm [12]. In that algorithm we introduced the basic idea of forcing a new iterate to belong to the convex region defined by the computed separating hyperplane. Both, the theoretical results and the numerical experiences, showed the advantages of this approach.

The acceleration scheme applied in the DACCIM algorithm is the basis for extending its applicability to other class of algorithms suitable for parallel processing. Among them is the IPA algorithm [6]. In particular, we used the same approach in the more general ACIPA algorithm based on the original IPA, both for computing incomplete projections onto each block, as well as for obtaining the new iterate x^{k+1}. As a consequence of the procedure to find the incomplete projection y_i^k onto each block using DACCIM, a hyperplane $\tilde{H}_i^k$ is obtained, deeper than the one given in [6]. Therefore, the new iterate x^{k+1} is the projection of x^k onto a deeper separating hyperplane (2.22). The direction $\tilde{d}^k$ defined by this algorithm preserves the fact that the new iterate does not fall outside of the region defined by the last separating hyperplane. The numerical results show a very competitive behaviour when dealing with blocks of inequalities by means of approximate projections such as in the ACIPA algorithm, as predicted by the theory.

References

[1] R. Bramley and A. Sameh, Row projection methods for large nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* **13** (1992) 168–193.

[2] Y. Censor, Parallel application of block-iterative methods in medical imaging and radiation therapy, *Math. Programming* **42** (1988) 307–325.

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

[4] G. Cimmino, Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari, *Ric. Sci.* **16** (1938) 326–333.

[5] U. M. García-Palomares, Parallel projected aggregation methods for solving the convex feasibility problem, *SIAM J. Optim.* **3** (1993) 882–900.

[6] U. M. García-Palomares and F.J. González-Castaño, Incomplete projection algorithms for solving the convex feasibility problem, *Numerical Algorithms* **18** (1998) 177–193.

[7] L. G. Gubin, B. T. Polyak, and E. V. Raik, The method of projections for finding the common point of convex sets, *USSR Comput. Math. and Math.Phys.* **7** (1967) 1–24.

[8] G. T. Herman and L.B. Meyer, Algebraic reconstruction techniques can be made computationally efficient, *IEEE Trans. Medical Imaging* **12** (1993) 600–609.

[9] Y. Saad, SPARSKIT: A basic tool kit for sparse matrix computations. *Technical Report 90-20, Research Institute for Avanced Computer Science. NASA Ames Research Center, Moffet Field, CA, l990* .

[10] H. D. Scolnik, N. Echebest, M. T. Guardarucci, M. C. Vacchino, A class of optimized row projection methods for solving large non-symmetric linear systems, *Report Notas de Matemática-74, Department of Mathematics, University of La Plata, AR, 2000* (to appear in Applied Numerical Mathematics).

[11] H. D. Scolnik, N. Echebest, M. T. Guardarucci, M. C. Vacchino, Acceleration scheme for Parallel Projected Aggregation Methods for solving large linear systems, 2000 (submmited to Annals of Operations Research).

[12] H. D. Scolnik, N. Echebest, M. T. Guardarucci, M. C. Vacchino, New Optimized and Accelerated PAM Methods for Solving Large Non-symmetric Linear Systems: Theory and Practice, in: *Inherently Parallel Algorithms in Feasibility and Optimization and their Applications*, D. Butnariu, Y. Censor and S. Reich (Editors), Studies in Computational Mathematics, Volume 8, Elsevier Science, Amsterdam, 2001.