Notas de Matemática

A class of optimized row projection methods for solving large non-symmetric linear systems

H. Scolnik, N. Echebest, M.T. Guardarucci, M.C. Vacchino

74

2000

UNIVERSIDAD NACIONAL DE LA PLATA

Departamento de Matemática - Facultad de Ciencias Exactas

A class of optimized row projection methods for solving large non-symmetric linear systems *

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

August 2000.

Abstract

We present in this paper optimal and accelerated row projection algorithms arising from new results that allow us to define the iterate x^{k+1} as the projection of x^k onto a hyperplane which minimizes its distance to the solution x^*. These algorithms also use a novel partition strategy into blocks based on sequential estimations of their condition numbers. Numerical results are given showing the new algorithms are more robust than Krylov subspace based methods, although the latter are generally faster when they converge.

Keywords: Projected aggregate methods, row partition strategies, parallel iterative methods.

AMS(MOS) subject classification. 65,15

*Work supported by the universities of Buenos Aires (Project EX.146) and La Plata (Project 11/X103), Argentina.

†Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina(scolnik@fd.com.ar).

‡Departamento de Matemática, Universidad of La Plata, Buenos Aires. Argentina (opti@mate.unlp.edu.ar).

1 Introduction.

This paper presents new iterative projection methods for computing a solution x^* of a non-symmetric compatible linear system $A^t x = b$, $A \in \Re^{n \times m}$.

Among the most frequently used iterative methods for solving large, sparse, non-symmetric linear systems $A^t x = b$, $A \in \Re^{n \times n}$ are those of the conjugate gradients type, like GMRES and Bi-CGSTAB ([14], [16]). An excellent implementation of these algorithms is available in SPARSKIT2.

Bi-CGSTAB is more convenient because its storage requirements do not grow with the number of iterations. These algorithms when convergent are extremely fast and efficient, but unfortunately numerical breakdowns occur.

The row projection methods avoid those difficulties [3] although in their classical formulations ([6], [13])convergence is usually very slow.

Block versions of these methods consider a partition of $A = [A_1, A_2, \ldots, A_q]$ and the corresponding partition of b. For each iterate x^k, the projection onto the orthogonal subspace to the one spanned by the rows of ${A_i}^t$ is computed by means of $A_i(A_i^t A_i)^{-1} A_i^t x^k$, for $i = 1, \ldots, q$. This approach is particularly useful for parallel implementation when the system is appropiately decomposed [3].

The PAM (Projected Aggregate Methods) sort of algorithms were introduced by Householder and Bauer in [12]. The idea is to generate the new point x^{k+1} as the projection of x^k onto an "aggregate" hyperplane usually arising from heuristic linear combinations of the hyperplanes defined by the blocks. García-Palomares gave conditions for using these methods in conjunction with parallel processing [9]. He also extended them for solving the important convex feasibility problem in [8] and [10].

Our first proposal to improve the speed of convergence is to use, instead of heuristic combinations, optimal aggregate hyperplanesin the sense that the distance between x^{k+1} and x^* is minimized. The origin of this idea arose from the merit function presented in [15]. This optimization demands the solution of a small quadratic subproblem whose dimension is at most the number of blocks.

The second idea is to accelerate even more the optimal algorithm by projecting the directions given by the blocks onto the aggregate hyperplane defined in the last iteration, and optimally combining them by means of the approach used in the first proposal.

On the other hand, the performance of the projection methods is highly

dependent on the way in which the rows of the matrix are splitted into blocks. Bramley and Sameh [3] gave conditions that a block splitting algorithm should satisfy. We also present in this paper a new block splitting algorithm which fulfills those conditions, based upon sequential estimations of condition numbers. A very useful byproduct of the new splitting algorithm is that it provides $(A_i{}^t A_i)^{-1}$ for each block, necessary for computing projections.

This paper is organized as follows: In Section 2 the new block projection algorithms are presented, together with the related convergence theory. In Section 3 the block splitting algorithm is given, as well as some of its theoretical properties. In Section 4 some numerical results comparing the performance of the new algorithms with the preconditioned methods contained in SPARSKIT2 and also with two from the SLATEC Library are presented.

2 The Algorithms: Basic Properties and Convergence Results.

Given a system $A^t x = b$, $A \in \Re^{n\times m}$, $m \leq n$, we will assume that it is compatible, and the matrix is not nil.

Let $x^* \in \Re^n$ be a solution of $A^t x = b$.

From hereafter $\|x\|$ will denote the Euclidean norm of $x \in \Re^n$.

Given $M \in \Re^{n\times l}$, $l \leq n$, P_M will denote the orthogonal projector onto $R(M)$, subspace spanned by the columns of M, and $P_{M^\perp}$ the orthogonal projector onto the orthogonal subspace to $R(M)$.

The block version of the projection methods consider a partition of A^t into q row blocks as follows: $A = [A_1, A_2, \ldots, A_q]$, $A_i \in \Re^{n\times m_i}$, with $\sum_{i=1}^{q} m_i = m$. and the corresponding partition of $b^t = (b_1^t, b_2^t, \ldots, b_q^t)$, where $1 \leq q \leq m$.

This yields a partition of the set $\{1, \ldots, m\} = I_1 \cup I_2 \cup \ldots \cup I_q$, where I_j contains indices rows of $A^t x = b$ belonging to $A_j^t x = b_j$, for each $j = 1, \ldots, q$.

We denote, for $i = 1, \ldots, q$

$$L_i = \{x \in \Re^n : A_i^t x = b_i, b_i \in \Re^{m_i}\}$$

We will assume that each block of the matrix has full rank, that is $rank(A_i) = m_i$. This hypothesis will be satisfied by the block partition strategy presented in Section 3.

For every $i = 1, \ldots, q$ the orthogonal projection onto L_i is the mapping $P_i : \Re^n \to L_i$ given by

$$P_i(x) = argmin\{\|x - y\| \ : \ y \in L_i\}$$

In the Parallel Block Iterative Algorithms described in [4], given a current iterate x^k, block iterations are first performed using x^k, on all blocks simultaneously, defining for $i = 1, \ldots, q$,

$$y_i^k = B_i(x^k, \{a_j\}_{j \in I_i})$$

The next iterate x^{k+1} is generated from the intermediate ones y_i^k by

$$x^{k+1} = S(\{y_i^k\}_{i=1}^q)$$

where S and B_i are algorithmic operators, and B_i generates y_i^k from the information contained in all rows of $A^t x = b$ whose indices belong to I_i.

The version of the block-iterative projections method described by Censor and Zenios (Algorithm 5.6.1 [5]), to find $x^* \in \bigcap_{i=1}^q L_i$, has the form

Initialization: $x^0 \in \Re^n$ arbitrary.

Iterative Step:

$$x^{k+1} = x^k + \lambda_k \sum_{i=1}^q w_i^k (P_i(x^k) - x^k),$$

$$k = k + 1$$

where λ_k are relaxation parameters and $w^k = (w_i^k)_{i=1}^q$ are varying weight vectors such that $w_i^k \geq 0$ and $\sum_{i=1}^q w_i^k = 1$.

In particular, choosing any sequence of weight vectors $\{w^k\}$ with $w_i^k > 0$, for all i, k, leads to a fully simultaneous Cimmino type algorithm [1]. Aharoni and Censor [1] have shown that if $\epsilon \leq \lambda_k \leq 2 - \epsilon$, for all $k \geq 0$, for an arbitrary $\epsilon > 0$, and if $\sum_{k=0}^{\infty} w_i^k = +\infty$, for $i = 1, \ldots, q$, then the sequence $\{x^k\}$ converges to a point $x^* \in \bigcap_{i=1}^q L_i$.

The basic idea in the parallel projection methods by blocks is given a point x^k, a closer point y_i^k to L_i is computed, and the next iterate is defined by means of a combination of the generated directions $d_i^k = y_i^k - x^k$.

In the special case where y_i^k is the projection onto L_i, for all $i = 1, \ldots, q$. we can get the following elementary results, which are here included because they are useful for understanding and proving the main theorems.

Lemma 2.1 *Given x^k, $x^k \neq x^*$. If $y_i^k = P_i(x^k)$, $d_i^k = y_i^k - x^k$ and $rank(A_i) = m_i$ for $i = 1, \ldots, q$, then*

(i) $d_i^k = A_i v_i^k$, with $v_i^k \in \Re^{m_i}$, being being

$$v_i^k = (A_i^t A_i)^{-1}(b_i - A_i^t x^k) = (A_i^t A_i)^{-1} A_i^t (x^* - x^k)$$

(ii) $\|d_i^k\|^2 = (d_i^k)^t (x^* - x^k)$

(iii) If $d^k = \sum_{i=1}^q w_i^k d_i^k$ then $(d^k)^t (x^* - x^k) = \sum_{i=1}^q w_i^k \|d_i^k\|^2$.

Proof. Considering $y_i^k = argmin\{\|y - x^k\|^2 \;\; y \in L_i\}$, according to the Kuhn-Tucker optimality condition [7] on the equivalent problem

$$min\{\|y - x^k\|^2 \;:\; A_i^t(y - x^k) = b_i - A_i^t x^k\},$$

there exists a vector $v_i^k \in \Re^{m_i}$ such that $d_i^k = y_i^k - x^k = A_i v_i^k$ satisfying $A_i^t A_i v_i^k = b_i - A_i^t x^k$. Therefore $v_i^k = (A_i^t A_i)^{-1}(b_i - A_i^t x^k)$, and $d_i^k = A_i(A_i^t A_i)^{-1}(b_i - A_i^t x^k)$.

Since $b_i = A_i^t x^*$, it follows that $d_i^k = A_i(A_i^t A_i)^{-1} A_i^t(x^* - x^k)$.

To see (ii) we consider the subsystem $A_i^t(y - x^k) = b_i - A_i^t x^k$, whose least norm solution is $d_i^k = y_i^k - x^k = A_i v_i^k$. Left multiplying by v_i^k we obtain $(v_i^k)^t A_i^t A_i v_i^k = (v_i^k)^t(b_i - A_i^t x^k)$. Therefore, $\|d_i^k\|^2 = (v_i^k)^t(b_i - A_i^t x^k)$. Moreover considering that x^* satisfies $A_i^t x^* = b_i$, we obtain

$$\|d_i^k\|^2 = (v_i^k)^t A_i^t(x^* - x^k) = (d_i^k)^t(x^* - x^k).$$

To see (iii), since $d^k = \sum_{i=1}^q w_i^k d_i^k$, it follows from (ii).□

In the PAM methods the iterative step is defined by means of the projection of the current iterate x^k onto a new hyperplane added to the system arising from a linear combination of the original hyperplanes [9].

U. M. García-Palomares in [10] described variations of PAMs for solving structured convex systems, and introduced novel methods (PPAMs) that allow a high degree of parallelism for nonstructured convex systems. The large system is splitted into smaller subsystems, not necessarily disjoint.

The parallel algorithms presented in [10], can be described for the particular case of systems of linear equations using our notation and hypotheses, in the following way.

Given an iterate x^k, compute for each block a weighted direction $w_i^k \lambda_i^k d_i^k$ aimed at finding a closer point y_i^k to L_i, considering $d_i^k = A_i v_i^k$, under the general scheme for PAMs for solving $A_i^t x = b_i$, and λ_i^k is defined as

$$\lambda_i^k = argmin_\lambda \|x^k + \lambda d_i^k - x^*\|^2$$

To ensure convergence to a solution of the system define the new iterate x^{k+1}, using a combination of directions d_i^k, $d^k = \sum_{i=1}^q w_i^k \lambda_i^k d_i^k$, and compute $x^{k+1} = x^k + \lambda_k d^k$, where

$$\lambda^k = argmin_\lambda \|x^k + \lambda d^k - x^*\|^2 \tag{2.1}$$

The parallel version for linear systems considering $y_i^k = P_i(x^k)$ for each block $i = 1, \ldots, q$, and a certain choice of $\{w_i^k\}_{i=1}^q$, $\eta \leq w_i^k \leq 2 - \eta$, $\eta > 0$. is

described by the iterative step.

Iterative Step : Given x^k

Do for $i = 1, \ldots, q$ **in parallel**
Compute $y_i^k = P_i(x^k)$
Define $d_i^k = y_i^k - x^k$.
Define w_i^k, $\eta \leq w_i^k \leq 2 - \eta$.
End do.
Define $d^k = \sum_{i=1}^q w_i^k d_i^k$,
$x^{k+1} = x^k + \lambda_k d^k$, $\lambda_k = argmin_\lambda \|x^k + \lambda d^k - x^*\|^2$.

Remark 1. As a consequence that each $d_i^k = P_i(x^k) - x^k$, it turns out that λ_i^k is 1. Furthermore, from (i) of Lemma 2.1 $d_i^k = A_i v_i^k$, $v_i^k \in \Re^{m_i}$, for all $i = 1, \ldots, q$, it follows that a vector $v^k \in \Re^m$ exists such that $d^k = Av^k$. This property is typical of the PAM methods.

Lemma 2.2 *If* $d^k = \sum_{i=1}^q w_i^k d_i^k$, *where* $d_i^k = P_i(x^k) - x^k$ *for all* $i = 1, \ldots, q$. *and* $\lambda_k = argmin_\lambda \|x^k + \lambda d^k - x^*\|^2$, *then*

$$\lambda_k = (d^k)^t (x^* - x^k)/\|d^k\|^2 = (\sum_{i=1}^q w_i^k \|d_i^k\|^2)/\|d^k\|^2 \tag{2.2}$$

If $x^{k+1} = x^k + \lambda_k d^k$, *then the sequence* $\{x^k\}$ *generated by this procedure satisfies* $\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - \alpha_k$, *where*

$$\alpha_k = \lambda_k^2 \|d^k\|^2 = (\sum_{i=1}^q w_i^k \|d_i^k\|^2)^2/\|d^k\|^2. \tag{2.3}$$

Proof. Considering the convex quadratic function
$\phi(\lambda) = \|x^k + \lambda d^k - x^*\|^2 = \|x^k - x^*\|^2 + 2\lambda (d^k)^t (x^k - x^*) + \lambda^2 \|d^k\|^2$
from its derivative equal to zero we get $\lambda_k = (d^k)^t (x^* - x^k)/\|d^k\|^2$.

Since $d^k = \sum_{i=1}^q w_i^k d_i^k$, and $(d_i^k)^t (x^* - x^k) = \|d_i^k\|^2$ by Lemma 2.1 (ii). (2.2) follows.

Considering $\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - 2\lambda_k (d^k)^t (x^* - x^k) + \lambda_k^2 \|d^k\|^2$. and the expression for λ_k given by (2.2), we get (2.3).□

The convergence of the sequence $\{x^k\}$ generated by this procedure is assured by Lemma 2.1 in [10].

The iterate x^{k+1} is the projection of x^k onto the aggregate hyperplane

$$H^k = \{x \in \Re^n : (d^k)^t(x - x^k) = (d^k)^t(x^* - x^k)\} \tag{2.4}$$

where $(d^k)^t(x^* - x^k) = \sum_{i=1}^q w_i^k \|d_i^k\|^2$, which is a combination of the hyperplanes $H_i^k = \{x \in \Re^n : (d_i^k)^t(x - x^k) = (d_i^k)^t(x^* - x^k)\}$ for $i = 1, \ldots, q$, considering their equivalent expression

$$H_i^k = \{x \in \Re^n : (v_i^k)^t A_i^t(x - x^k) = (v_i^k)^t A_i^t(x^* - x^k)\},$$

obtained by combining the equalities $A_i^t(x - x^k) = b_i - A_i^t x^k$, and using $b_i = A_i^t x^*$.

Our first proposal is to choose $\{w_i^k\}_{i=1}^q$ in such a way that an optimal combination of the directions $\{d_i^k\}_{i=1}^q$ is obtained in the sense that

$$min_{w \in \Re^q} \|x^k + \sum_{i=1}^q w_i d_i^k - x^*\|.$$

This idea leads to the iterative step, $x^{k+1} = x^k + \sum_{i=1}^q w_i^k d_i^k$, where $w^k \in \Re^q$ is the solution of the convex quadratic problem

$$min_{u \in \Re^q} \|x^k - x^*\|^2 + 2u^t(D^k)^t(x^k - x^*) + u^t(D^k)^t D^k u, \tag{2.5}$$

where $D^k = [d_1^k, \ldots, d_q^k]$.

Lemma 2.3 *The matrix D^k of directions has rank q_k, $0 < q_k \leq q$, if x^k is different from x^*.*

Proof. If x^k is different from x^* there exists at least $y_i^k \neq x^k$, such that $d_i^k = A_i v_i^k$, with $v_i^k \in \Re^{m_i}$, $v_i^k \neq 0$ by (i) of Lemma 2.1 , then we have that $rank(D^k) = q_k$ with $0 < q_k \leq q$.□

Remark 2 . Taking into account our general hypotheses, a direction d_i^k may exist such that is a linear combination of the other directions. Hence, the matrix D^k of problem (2.5) is defined using only linearly independent directions which means that $rank(D^k) = q_k$ with $q_k \leq q$. Therefore, the optimal solution of problem (2.5), by the optimality criteria, satisfies $w^k = ((D^k)^t D^k)^{(-1)}(D^k)^t(x^* - x^k)$. We can get the explicit expression of w^k. considering $(d_i^k)^t(x^* - x^k) = \|d_i^k\|^2$ by (ii) of Lemma 2.1. Thus, the optimal solution of (2.5) is $w^k = ((D^k)^t D^k)^{(-1)}(\|d_1^k\|^2, \|d_2^k\|^2, \ldots, \|d_{q_k}^k\|^2)^t$.

Algorithm 1 (ALG1).

Step 0. Split the matrix into blocks by rows using the method described in Section 3, obtaining $A = [A_1, A_2, \ldots, A_q]$, and the corresponding partition of $b^t = (b_1^t, b_2^t, \ldots, b_q^t)$. Compute for each block $i = 1, \ldots, q$, with m_i rows, the matrix $(A_i^t A_i)^{-1} = (L_{m_i}^t D_{mi} L_{m_i})^{-1}$.

Main step. Given the starting point $x^0 \in \Re^n$, $0 \leq \epsilon < 1$, k=0

While ($\|Ax^k - b\| > \epsilon$) **do**

For each $i = 1, \ldots, q$,

$d_i^k = A_i(A_i^t A_i)^{-1}(-r_i^k)$, where $r_i^k = A_i^t x^k - b_i$

Define $x^{k+1} = x^k + d^k$, where $d^k = D^k w^k$

$w^k = ((D^k)^t D^k)^{(-1)}(\|d_1^k\|^2, \|d_2^k\|^2, \ldots, \|d_{q_k}^k\|^2)^t$

$D^k = [d_1^k, d_2^k, \ldots, d_{q_k}^k]$.

$k = k + 1$

End while;

End procedure.

In order to derive some theoretical results we need the following:

Lemma 2.4 *In each iteration k the optimal direction $d^k = D^k w^k$ of the main step of Algorithm 1, satisfies*

i) $(D^k)^t D^k w^k = (D^k)^t (x^* - x^k)$, *where*
$(D^k)^t (x^* - x^k) = (\|d_1^k\|^2, \|d_2^k\|^2, \ldots, \|d_{q_k}^k\|^2)^t$.

ii) $\|d^k\|^2 = (d^k)^t (x^* - x^k) = \sum_{i=1}^{q_k} w_i^k \|d_i^k\|^2$

iii) $(d_i^k)^t d^k = \|d_i^k\|^2$, *for all* $i = 1, \ldots, q$, *and*
$\|d^k\| \geq \|d_i^k\|$, *for all* $i = 1, \ldots, q$

iv) The new iterate x^{k+1}, *is such that* $(d^k)^t (x^{k+1} - x^*) = 0$.

Proof. From the definition of $d^k = D^k w^k$, by the optimality criteria of the quadratic problem (2.5) which defines w^k, and considering $(d_i^k)^t (x^* - x^k) = \|d_i^k\|^2$ by (ii) of Lemma 2.1, we get (i).

(ii) arises from left multiplying (i) by $w^k \in \Re^{q_k}$, obtaining

$\|d^k\|^2 = (w^k)^t (D^k)^t (x^* - x^k) = (d^k)^t (x^* - x^k) = \sum_{i=1}^{q_k} w_i^k \|d_i^k\|^2$.

To see (iii), from (i) we get $(d_i^k)^t d^k = \|d_i^k\|^2$, for $i = 1, \ldots, q_k$. Thus, we obtain $\|d^k\| \geq \|d_i^k\|$, for all $i = 1, \ldots, q_k$.

If there exists a *jth* block such that its $d_j^k = 0$, it follows that $\|d^k\| \geq \|d_j^k\|$. If there exists a direction $d_j^k \neq 0$ such that it is a combination of columns of D^k, then $d_j^k = D^k u_j$, where $u_j \in \Re^{q_k}$. Since d^k satisfies $(D^k)^t d^k =$

$(D^k)^t(x^* - x^k)$, multiplying by $u_j \in \Re^{q_k}$ we get $(d_j^k)^t d^k = (d_j^k)^t(x^* - x^k)$. By (ii) of Lemma 2.1, $(d_j^k)^t d^k = \|d_j^k\|^2$, then follows $\|d^k\| \geq \|d_j^k\|$. It follows (iii).

(iv) is a consequence of the definition of $x^{k+1} = x^k + d^k$, and (ii) which allows us to write $\|d^k\|^2 = (d^k)^t(x^* - x^k)$, then

$$(d^k)^t(x^* - x^{k+1}) = (d^k)^t(x^* - x^k) - \|d^k\|^2 = 0.\square$$

Lemma 2.5 *The sequence $\{x^k\}$ generated by Algorithm 1 satisfies $\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - \alpha_k^*$, where*

$$\alpha_k^* = \sum_{i=1}^{q_k} w_i^k \|d_i^k\|^2 = \|d^k\|^2 \tag{2.6}$$

Proof. Taking into account that $x^{k+1} = x^k + d^k$, where the optimal combination d^k satisfies (ii) in Lemma 2.4, and considering the expression given in (2.3) for the PAM methods, the result follows.$\square$

Remark 3. From the optimality of the linear combination of $\{d_i^k\}_{i=1}^q$ used in Algorithm 1, it is obvious that $\alpha_k^* \geq \alpha_k$, where α_k is the one defined in (2.3).

2.1 An accelerated block projection algorithm.

Due to the fact that x^k is obtained as the projection of x^{k-1} onto the hyperplane $H^{k-1} = \{x \in \Re^n \ : \ (d^{k-1})^t(x - x^*) = 0\}$, and according to (2.4), it turns out that the optimal seeked increment $x^* - x^k$ belongs to it.

This leads to consider the possibility of performing from x^k the next iterative step on such hyperplane, which was optimally chosen in the sense described previously. One possibility is to project the optimal direction d^k, computed by the main step of Algorithm 1, onto the hyperplane $\{x \in \Re^n \ : \ (d^{k-1})^t(x - x^k) = 0\}$. Such a direction, denoted by $\hat{d}^k$, is $\hat{d}^k = \sum_{i=1}^{q_k} w_i^k P_{v^\perp}(d_i^k)$, where $v = x^k - x^{k-1}$.

The new iterate $\hat{x}^{k+1}$ could be defined as the point $x^k + \hat{\lambda}\hat{d}^k$ corresponding to the solution of the problem

$$min_\lambda \|x^k + \lambda \hat{d}^k - x^*\|^2 \tag{2.7}$$

In this case we obtain

Lemma 2.6 *If* $\hat{x}^{k+1} = x^k + \hat{\lambda}\hat{d}^k$, *where* $\hat{d}^k = P_{v^\perp}(d^k)$, $v = x^k - x^{k-1}$, *and* $\hat{\lambda}$ *is defined by (2.7), then* $\|\hat{x}^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - \hat{\alpha}_k$, *where*

$$\hat{\alpha}_k = (\|d^k\|^2/\|\hat{d}^k\|^2)\alpha_k^*, \tag{2.8}$$

with α_k^* *given in (2.6), is defined by the iterative step of Algorithm 1.*

Proof. We consider the optimal λ of problem (2.7), $\hat{\lambda} = (\hat{d}^k)^t(x^* - x^k)/\|\hat{d}^k\|^2$. On the other hand, from the definition of $\hat{d}^k = P_{v^\perp}(d^k)$, and using the results (ii) and (iv) of Lemma 2.4, we get $(\hat{d}^k)^t(x^* - x^k) = (d^k)^t P_{v^\perp}(x^* - x^k) = (d^k)^t(x^* - x^k) = \|d^k\|^2$. Hence, $\hat{\lambda} = \|d^k\|^2/\|\hat{d}^k\|^2$.

To compute $\hat{\alpha}_k$, we consider $(\hat{d}^k)^t(x^* - x^k) = \|d^k\|^2$, and the expression in (2.3) $\|\hat{x}^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - \hat{\lambda}^2\|\hat{d}^k\|^2$, and considering $\hat{\lambda}$, it follows that $\hat{\alpha}_k = (\|d^k\|^2/\|\hat{d}^k\|^2)\|d^k\|^2$. Then, from the definition of α_k^* in (2.6) we obtain (2.8).□

The following results will be used for explaining the accelerated convergence features of a procedure which at the *kth* iterate, $k \geq 1$, uses the direction $\hat{d}^k = P_{v^\perp}(d^k)$, where d^k is the optimal combination defined in the main step of Algorithm 1, and v is the previous step $x^k - x^{k-1}$, which satisfies the conditions:

(C1) $v^t(x^* - x^k) = 0$.

(C2) $(d_i^{k-1})^t v = \|d_i^{k-1}\|^2$, for $i = 1, \ldots, q$.

In particular, note that the iterative step $v = d^{k-1}$, defined in x^{k-1} by Algorithm 1, satisfies both conditions due to (iii) and (iv) of Lemma 2.4.

Lemma 2.7 *Given* x^k, $k \geq 1$, *and* x^{k-1} *the previous iterate, let us consider the directions* d_i^k *and* d_i^{k-1}, *respectively computed at the two iterates, for each* $i = 1, \ldots, q$, *then*

(i) $d_i^k = d_i^{k-1} - P_{A_i}(v)$, *being* $v = x^k - x^{k-1}$.

Furthermore, if $v = x^k - x^{k-1}$ *satisfies (C2), then*

(ii) If $d_i^k \neq 0$ *and* $d_i^{k-1} \neq 0$ *then* d_i^k *is orthogonal to* d_i^{k-1}.

(iii) If $d_i^k \neq 0$ *then* $\|d_i^k\|^2 = -(d_i^k)^t v$.

Moreover, if $v = x^k - x^{k-1}$ *satisfies (C1), then*

(iv) If $d_i^k \neq 0$ *then* $P_{v^\perp}(d_i^k) \neq 0$.

Proof. From the definition of d_i^k, and using (i) from Lemma 2.1. $d_i^k = A_i(A_i^t A_i)^{-1}A_i^t(x^* - x^k) = A_i(A_i^t A_i)^{-1}A_i^t(x^* - x^{k-1} + x^{k-1} - x^k)$. Hence, since $A_i(A_i^t A_i)^{-1}A_i^t = P_{A_i}$, is the projection matrix onto $R(A_i)$, (i) follows.

To see (ii), multiplying the current (i) by $d_i^{k-1} \neq 0$ and considering the properties of v, we get $(d_i^k)^t d_i^{k-1} = 0$.

(iii) follows by multiplying (i) by d_i^k and using (ii).

To see (iv), since *(C1)* holds, and considering (ii) of Lemma 2.1, $\|d_i^k\|^2 = (d_i^k)^t(x^* - x^k)$, we obtain $(d_i^k)^t P_{v^\perp}(x^* - x^k) = \|d_i^k\|^2$. Thus, $P_{v^\perp}(d_i^k) \neq 0$.□

Lemma 2.8 *If at x^k, $k \geq 1$, $x^k \neq x^*$, we consider the optimal direction $d^k = D^k w^k$ defined by the main step of Algorithm 1, and the previous step $v = x^k - x^{k-1}$ satisfying the conditions (C1) and (C2), then*

(i) $\|P_{v^\perp}(d^k)\| > 0$.

(ii) $\|P_{v^\perp}(d^k)\| < \|d_k\|$.

Proof. Taking into account the hypotheses on the iterative step we have that $v^t(x^* - x^k) = 0$. Hence $P_{v^\perp}(x^* - x^k) = (x^* - x^k)$. $P_{v^\perp}$ being the projection operator onto the orthogonal subspace to $R(v)$. From (ii) of Lemma 2.4. $\|d^k\|^2 = (d^k)^t(x^* - x^k)$, then using the properties of $P_{v^\perp}$, we obtain $\|d^k\|^2 = (d^k)^t\, P_{v^\perp}(x^* - x^k) = (P_{v^\perp}(d^k))^t(x^* - x^k)$. Hence, using the Cauchy-Schwarz inequality we get, $\|d^k\|^2 \leq \|P_{v^\perp}(d^k)\|\|x^* - x^k\|$. Thus, since $\|x^* - x^k\| > 0$ and $\|d^k\| \geq max\{\|d_i^k\|\}_{i=1}^q$ by (iii) of Lemma 2.4. $\|P_{v^\perp}(d^k)\| \geq (\|d^k\|^2/\|x^* - x^k\|) > 0$, and the result (i) follows.

To derive (ii) let us consider the norm of the projection of d^k onto the subspace spanned by the vector v, $\|P_v(d^k)\| = |(d^k)^t v|/\|v\|$.

From (iii) of Lemma 2.7 $\|d_i^k\|^2 = -(d_i^k)^t v$, for $i = 1, \ldots, q$, and considering $d^k = \sum_{i=1}^{q_k} w_i^k d_i^k$, it follows that $|(d^k)^t v| = \sum_{i=1}^{q_k} w_i^k \|d_i^k\|^2 = \|d^k\|^2$, by (ii) of Lemma 2.4. Therefore, $\|P_v(d^k)\| > 0$ and the result (ii) follows. □

Theorem 2.1 *If $\hat{x}^{k+1} = x^k + \hat{\lambda}\hat{d}^k$, is defined as in (2.7) at kth iterate, $k \geq 1$, where $\hat{d}^k = P_{v^\perp}(d^k)$, and $v = x^k - x^{k-1}$ satisfies (C1) and (C2), then*

$$\|\hat{x}^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - \hat{\alpha}_k, \quad with \quad \hat{\alpha}_k > \alpha_k^*$$

Proof. From Lemma 2.8 it follows that $\|d^k\|/\|\hat{d}^k\| > 1$. Hence, from the definition of $\hat{\alpha}_k$ in (2.8), we get $\hat{\alpha}_k > \alpha_k^*$. □

With the aim of arriving at the new accelerated procedure, we take into account Theorem 2.1, where the used direction belongs to the subspace defined by $[P_{v^\perp}(d_1^k), P_{v^\perp}(d_2^k), \ldots, P_{v^\perp}(d_q^k)]$. Then, it is natural to choose $\hat{d}^k$ at x^k, as the best combination of $\{P_{v^\perp}(d_i^k)\}_{i=1}^q$ such that the distance between $x^{k+1} = x^k + \hat{d}^k$ and x^* is minimized. This idea leads to define the iterative step, in the following way.

Given x^k, $k \geq 1$, $x^k \neq x^*$, the next iterate $x^{k+1} = x^k + D^k w^k$, where $w^k \in \Re^{q_k}$ is the solution of the quadratic problem

$$min_{u \in \Re^{q_k}} \|x^k + \hat{D}^k u - x^*\|^2, \tag{2.9}$$

where $v = x^k - x^{k-1}$, $\hat{D}^k = [P_{v^\perp}(d_1^k), P_{v^\perp}(d_2^k), \ldots, P_{v^\perp}(d_{q_k}^k)]$, and q_k is the number of linearly independent directions in $\{P_{v^\perp}(d_i^k)\}_{i=1}^q$.

At x^0, define $x^1 = x^0 + d^0$, where d^0 is defined as in the main step of Algorithm 1.

Now we will describe the iterative step which will be used for defining the Accelerated Block Algorithm (Algorithm 2).

We shall use the notation $Q_0 = I_n$, $Q_k = P_{v^\perp}$, $v = x^k - x^{k-1}$ for $k \geq 1$. and $q_k = rank(D^k)$ for all $k \geq 0$.

Iterative Step : Given x^k, Q_k

Do for $i = 1, \ldots, q$ in parallel

Compute $y_i^k = P_i(x^k)$

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

Define $\hat{d}_i^k = Q_k(d_i^k)$.

End do.

Define $x^{k+1} = x^k + \hat{d}^k$, **where** $\hat{d}^k = \hat{D}^k w^k$

$$w^k = argmin_{u \in \Re^{q_k}} \|x^k + \hat{D}^k u - x^*\|^2,$$
$$\hat{D}^k = [\hat{d}_1^k, \hat{d}_2^k, \ldots, \hat{d}_{q_k}^k].$$

Set $v = \hat{d}^k$

$k = k + 1$

Now, it is necessary to prove the iterative step of Algorithm 2 is well defined and that the sequence generated satisfies the conditions *(C1)* and *(C2)* needed for proving Theorem 2.1.

Lemma 2.9 *Let x^k, $k \geq 0$, $x^k \neq x^*$, then the next iterate x^{k+1} is well defined by Algorithm 2, and the step $x^{k+1} - x^k$ satisfies the conditions (C1) and (C2).*

Proof. For $k = 0$, the result holds as a consequence of $x^0 \neq x^*$, and that by the definition of x^1, the results of Lemmas 2.3 and 2.4 are valid.

corresponding to the iterates defined by Algorithm 1. Hence, x^1 is well defined by (2.5), and $x^1 - x^0$ satisfies the conditions *(C1)* and *(C2)*, as proved in (iii) and (iv) of Lemma 2.4.

In order to prove the result for any $k \geq 1$, we will do it by induction assuming that it holds for $k-1$.

Since $k \geq 1$, and $x^k \neq x^*$, and by the inductive hypothesis the previous step $v = x^k - x^{k-1}$ satisfies *(C1)* and *(C2)*, then (iv) of Lemma 2.7 holds for all $d_i^k \neq 0$, computed in x^k. Thus, there exists at least a direction $P_{v^\perp}(d_i^k) \neq 0$, as a consequence that $x^k \neq x^*$, then we have that $rank(\hat{D}^k) = q_k$, with $0 < q_k \leq q$. Hence, the problem (2.9) is well defined and so is $x^{k+1} = x^k + \hat{D}^k w^k$,where w^k is the solution of problem (2.9).

The new step $x^{k+1} - x^k$, by its definition is $\hat{D}^k w^k$, which, due to the optimality condition of problem (2.9), satisfies $(\hat{D}^k)^t \hat{D}^k w^k = (\hat{D}^k)^t (x^* - x^k)$.

Hence, $\hat{d}^k = \hat{D}^k w^k$, satisfies for all $i = 1, \ldots, q_k$, $(\hat{d}_i^k)^t \hat{d}^k = (\hat{d}_i^k)^t (x^* - x^k)$. Since $\hat{d}_i^k = P_{v^\perp}(d_i^k)$, $v = x^k - x^{k-1}$, and $v^t(x^* - x^k) = 0$ by condition *(C1)* , is easy to see using the properties of the projector $P_{v^\perp}$ that $(\hat{d}_i^k)^t \hat{d}^k = (d_i^k)^t \hat{d}^k$, and $(\hat{d}_i^k)^t (x^* - x^k) = (d_i^k)^t (x^* - x^k)$. Thus, for all $i = 1, \ldots, q_k$, $(d_i^k)^t \hat{d}^k = (d_i^k)^t (x^* - x^k) = \|d_i^k\|^2$, using result (ii) of Lemma 2.1.

In order to complete the proof we will see that the result is also valid for the remaining directions d_i^k which do not belong to $\hat{D}^k$.

If there exists a direction d_j^k such that it is a combination of columns of $\hat{D}^k$, then $d_j^k = \hat{D}^k u_j$, where $u_j \in \Re^{q_k}$. Since $\hat{d}^k$ satisfies $(\hat{D}^k)^t \hat{d}^k = (\hat{D}^k)^t (x^* - x^k)$, multiplying by $u_j \in \Re^{q_k}$ we get $(\hat{d}_j^k)^t \hat{d}^k = (\hat{d}_j^k)^t (x^* - x^k)$. By the properties of the projections, considering $(\hat{d}_j^k)^t (x^* - x^k) = (d_j^k)^t (x^* - x^k)$, then by (ii) of Lemma 2.1, $(d_j^k)^t \hat{d}^k = \|d_j^k\|^2$. Hence, the new step $x^{k+1} - x^k = D^k w^k = \hat{d}^k$, satisfies for all $i = 1, \ldots, q$, $(d_i^k)^t \hat{d}^k = \|d_i^k\|^2$, which is the condition *(C2)*.

The step $x^{k+1} - x^k = \hat{D}^k w^k$, satisfies by its definition

$$(\hat{D}^k)^t \hat{D}^k w^k = (\hat{D}^k)^t (x^* - x^k)$$

Hence, $(\hat{D}^k)^t (x^k + \hat{D}^k w^k - x^*) = 0$, then left multiplying by w^k, we obtain that $(\hat{d}^k)^t (x^{k+1} - x^*) = 0$. Therefore the new step also satisfies *(C1)*. □

Theorem 2.2 *The sequence $\{x^k\}$ generated by Algorithm 2, satisfies*

$$\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - \hat{\alpha}_k^*, \text{ with } \hat{\alpha}_k^* > \alpha_k^*. \tag{2.10}$$

Proof. Since for all x^k, $k \geq 1$, the previous step $x^k - x^{k-1}$, satisfies the conditions *(C1)* and *(C2)*, the point defined by the problem (2.7), $\hat{x}^{k+1} = x^k + \hat{\lambda} P_{v^\perp}(d^k)$, where $v = x^k - x^{k-1}$, satisfies the result of Theorem 2.1.

Since the new iterate $x^{k+1} = x^k + \hat{D}^k w^k$, defined by Algorithm 2, satisfies $\|x^{k+1} - x^*\|^2 \leq \|\hat{x}^{k+1} - x^*\|^2$, by the optimality condition of the direction $D^k w^k$ computed by the solution of problem (2.9), we get

$$\|x^{k+1} - x^*\|^2 = \|x^k - x^*\|^2 - \hat{\alpha}_k^*, \text{ with } \hat{\alpha}_k^* \geq \hat{\alpha}_k > \alpha_k^*. \ \square$$

Now we will describe the Accelerated Block Algorithm.

Algorithm 2 (ALG2).

Step 0. Split the matrix into blocks by rows using the method described in Section 3. $A = [A_1, A_2, \ldots, A_q]$, and the corresponding partition of $b^t = (b_1^t, \ldots, b_q^t)$, obtaining for each block $i = 1, \ldots, q$, with m_i rows, the matrix $(A_i^t A_i)^{-1} = (L_{m_i}^t D_{mi} L_{m_i})^{-1}$

Main Step. Given the starting point $x^0 \in \Re^n$, $\epsilon > 0$.

k=0

While ($\|Ax^k - b\| > \epsilon$) **do**

For each $i = 1, \ldots, q$,

$d_i^k = A_i(A_i^t A_i)^{-1}(-r_i^k)$, where $r_i^k = A_i^t x_k - b_i$

Define $\hat{d}_i^k = Q_k(d_i^k)$.

Set $x^{k+1} = x^k + \hat{d}^k$, where $\hat{d}^k = \hat{D}^k w^k$

$w^k = ((\hat{D}^k)^t \hat{D}^k)^{(-1)}(\|d_1^k\|^2, \|d_2^k\|^2, \ldots, \|d_{q_k}^k\|^2)^t$

$\hat{D}^k = [\hat{d}_1^k, \hat{d}_2^k, \ldots, \hat{d}_{q_k}^k]$.

Set $v = \hat{d}^k$

$k = k + 1$

End while;

End procedure.

2.2 Convergence.

For studying the convergence of our methods we use a theory developed by Gubin et al., [11].

We shall use the notation $\mathcal{P} = \{1, \ldots, q\}$, $L = \bigcap_{i \in \mathcal{P}} L_i$ where

$$L_i = \{x \in \Re^n \ : \ A_i^t x = b_i, b_i \in \Re^{m_i}\}.$$

Denote by $d(x, L_i)$ the Euclidean distance between a point $x \in \Re^n$ and a set L_i, and define $\Phi(x) = max_{i \in \mathcal{P}}\{d(x, L_i)\}$.

Definition. A sequence $\{x^k\}_0^\infty$ is called Fejér-monotone with respect to the set L, if for $x^* \in L$, and for all $k \geq 0$, $||x^{k+1} - x^*|| \leq ||x^k - x^*||$.

It is easy to check that every Fejér-monotone sequence is bounded.

The fundamental theorem of Gubin et al.[11], is:

Theorem 2.3 *Let $L_i \subset \Re^n$, be a closed convex set for each $i \in \mathcal{P}$, $L = \bigcap_{i \in \mathcal{P}} L_i$, $L \neq \emptyset$. If the sequence $\{x^k\}_0^\infty$ satisfies the properties :*

i) $\{x^k\}_0^\infty$ is Fejér-monotone with regard to L, and

ii) $lim_{k \to \infty} \Phi(x^k) = 0$,

then $\{x^k\}_0^\infty$ converges to x^*, $x^* \in L$.

Proof. It follows from Lemma 5 and Lemma 6 of Gubin et al.[11].□

Lemma 2.10 *Any sequence $\{x^k\}_0^\infty$ generated by Algorithm 1 or Algorithm 2 satisfies (i) and (ii) of Theorem 2.3, provided $x^k \notin L$, for all $k \geq 0$.*

Proof. The proof of (i) follows immediately from Lemma 2.5 and Theorem 2.2. Moreover, it satisfies (ii), taking into account that α_k^* of Algorithm 1, and $\bar{\alpha}_k^*$ of Algorithm 2 tend to 0 when $k \to \infty$, because the sequence $\{\|x^k - x^*\|\}$ converges since it is bounded and monotonically decreasing. Then, considering the results of Theorem 2.2, Lemma 2.5 and (iii) of Lemma 2.4, we have $\bar{\alpha}_k^* \geq \alpha_k^* \geq max_{i \in \mathcal{P}} \|d_i^k\|^2$. Thus, we get $lim_{k \to \infty} \Phi(x^k) = 0$.□

Theorem 2.4 *The sequence $\{x^k\}$ generated by Algorithm 1 or Algorithm 2, converges to a solution x^* of $A^t x = b$.*

Proof. It follows from the assumption that the system $A^t x = b$ has a solution. Lemma 2.10 and Theorem 2.3.□

3 Partition into blocks using estimations of the condition numbers.

In this section we will present a partitioning strategy for obtaining blocks with at most μ rows and such that the condition numbers of the matrices

$(A_i^t A_i)^{-1}$ necessary for computing the projectors remain bounded by a given tolerance κ.

We shall assume the rows of the matrix of the original system $A^t x = b$ had been normalized, that is $\|a_l\| = 1$, $l = 1, \ldots, m$. We shall denote by e_j the jth canonical vector and by d_{jj} the jth element of a diagonal matrix D.

Suppose we are generating a block $A_i^t x = b_i$ which already has j rows of $A^t x = b$, $j < \mu$, and we know the Cholesky decomposition of $B_j = (A_i^t A_i)^{-1} = (L_j D_j L_j^t)^{-1}$([2]). Let $\tau = max\{d_{hh}\}_{h=1}^{j}$ and $\delta = min\{d_{hh}\}_{h=1}^{j}$.

The estimation β_i of the $cond(A_i^t A_i)$ is defined as $\beta_i = \tau/\delta$.

We shall prove that $d_{11} = 1$ and $d_{hh} \leq 1$, for each $h = 2, \ldots, j$. and therefore $\beta_i = 1/\delta$.

We shall accept to add a new row of $A^t x = b$ to the ith block if and only if it has not been yet assigned to another block and the estimation of the condition number of the augmented block does not exceed κ.

In order to decide the acceptance of a new row a_l^t into the ith block, we define $\tilde{A}_i = [A_i, a_l]$ and from the knowledge of the Cholesky decomposition of B_j, we proceed to update the factorization of $B_{j+1} = (\tilde{A}_i{}^t \tilde{A}_i)^{-1}$ in a recursive way, recomputing the estimation of the condition number of the expanded block. If such estimation is still less than κ, the row a_l^t is added.

Initially, for $j = 1$, A_i^t has only one row, hence $A_i^t A_i = 1$, and $L_1^{-1} = 1$. $D_1^{-1} = 1$.

If a block $A_i^t x = b_i$ is composed by j rows of $A^t x = b$, with $j < \mu$, in order to analyze the possibility of adding a row a_l^t, we compute the decomposition of $B_{j+1} = (\tilde{A}_i^t \tilde{A}_i)^{-1} = L_{j+1}^{-1} D_{j+1}^{-1} L_{j+1}^{-t}$.

Taking into account that

$$D_{j+1} = \begin{bmatrix} D_j & 0 \\ 0 & \delta_{j+1} \end{bmatrix}, \text{ and if } L_{j+1} = \begin{bmatrix} L_j & 0 \\ l_{j+1}^t & 1 \end{bmatrix} \text{ then } L_{j+1}^{-1} = \begin{bmatrix} L_j^{-1} & 0 \\ -l_{j+1}^t L_j^{-1} & 1 \end{bmatrix}.$$

From $\tilde{A}_i{}^t \tilde{A}_i = L_{j+1} D_{j+1} L_{j+1}^t$ we get

$$\begin{bmatrix} A_i^t A_i & A_i^t a_l \\ a_l^t A_i & \|a_l\|^2 \end{bmatrix} = \begin{bmatrix} L_j D_j L_j^t & L_j D_j l_{j+1} \\ l_{j+1}^t D_j L_j^t & \delta_{j+1} + l_{j+1}^t D_j l_{j+1} \end{bmatrix}.$$

Therefore, we choose l_{j+1} and δ_{j+1} such that

$$\begin{cases} L_j D_j l_{j+1} = A_i^t a_l \\ \delta_{j+1} = 1 - l_{j+1}^t D_j l_{j+1} \end{cases}, \text{ that is } \begin{cases} l_{j+1} = D_j^{-1} L_j^{-1} A_i^t a_l \\ \delta_{j+1} = 1 - a_l^t A_i (A_i^t A_i)^{-1} A_i^t a_l \end{cases}$$

Then, if $1/\delta_{j+1} \leq \kappa$ the row a_l^t is accepted, and the ith block is now composed by the rows of $\bar{A}_i^t$. If the expanded block still has less than μ rows, the procedure is repeated for analyzing the inclusion of another row.

The following Lemmas show that the matrix D_{j+1} has all its elements $d_{hh} \leq 1$ and that β_i provides an underestimation of $cond(A_i^t A_i)$.

Lemma 3.1 *Let a_l^t be the row being analyzed for appending it to the ith block. If $\|a_l\| = 1$ and θ is the angle between a_l and $R(A_i)$, subspace spanned by the j rows of A_i^t, then $\delta_{j+1} = \sin^2\theta$.*

Proof. From the previous formulas and $\|a_l\| = 1$ we obtain $\delta_{j+1} = 1 - \|P_{A_i}(a_l)\|^2 = \|a_l\|^2 - \|P_{A_i}(a_l)\|^2 = \|P_{A_i^\perp}(a_l)\|^2$ with $0 \leq \delta_{j+1} \leq 1$.

Moreover, if θ is the angle between a_l and $R(A_i)$, $|\sin\theta| = \|P_{A_i^\perp}(a_l)\|/\|a_l\|$, from which it follows that $\delta_{j+1} = \sin^2\theta$.□

Lemma 3.2 *If $(A_i^t A_i)^{-1} = (L_j D_j L_j^t)^{-1}$, $\delta = min\{d_{hh}\}_{h=1}^j$ and $\beta_i = 1/\delta$. then $\epsilon_j > 0$ exists such that $\epsilon_j cond(A_i^t A_i) \leq \beta_i \leq cond(A_i^t A_i)$.*

Proof. Let λ_m and λ_M be the lowest and greatest eigenvalues of $A_i^t A_i$ from which it follows that $cond(A_i^t A_i) = \lambda_M/\lambda_m$. In the following, we shall use the inequality $\lambda_m \leq (z^t A_i^t A_i z)/\|z\|^2 \leq \lambda_M$ for all $z \in \Re^j$.

Considering that $\|a_h\|^2 = 1$ then $e_h^t A_i^t A_i e_h = 1$, $h = 1, \ldots, j$. Thus, $\lambda_M \geq 1$. If $z_h = L_j^{-t} e_h$ for $h = 1, \ldots, j$ then $e_h^t z_h = (L_j^{-1})_{hh} = 1$ and $\|z_h\|^2 \geq 1$ for $h = 1, \ldots, j$. Moreover, $z_h^t A_i^t A_i z_h = e_h^t D_j e_h = d_{hh}$ hence, $\lambda_m \leq (z_h^t A_i^t A_i z_h)/\|z_h\|^2 = d_{hh}/\|z_h\|^2 \leq d_{hh}$, for $h = 1, \ldots, j$. Thus , $\lambda_m \leq \delta$ and therefore, recalling that $\lambda_M \geq 1$, we get $\beta_i \leq cond(A_i^t A_i)$.

If v_m is an eigenvector of $A_i^t A_i$ corresponding to λ_m and v_M is an eigenvector of $A_i^t A_i$ corresponding to λ_M with $\|v_m\| = \|v_M\| = 1$ then $\lambda_M = v_M^t L_j D_j L_j^t v_M = \|D_j^{1/2} L_j^t v_M\|^2$. Since from Lemma 3.1 is $d_{hh} \leq 1$ for all $h = 1, \ldots, j$ then $\lambda_M \leq \|L_j^t v_M\|^2$. Analogously, since $\lambda_m = \|D_j^{1/2} L_j^t v_m\|^2$ and $d_{hh} \geq \delta$ for all $h = 1, \ldots, j$, it follows that $\lambda_m \geq \delta \|L_j^t v_m\|^2$. Therefore, $\beta_i = 1/\delta \geq \|L_j^t v_m\|^2/\lambda_m \geq (\|L_j^t v_m\|^2/\|L_j^t v_M\|^2)(\lambda_M/\lambda_m)$. In this way we see that for $\epsilon_j = \|L_j^t v_m\|^2/\|L_j^t v_M\|^2$ is $\epsilon_j cond(A_i^t A_i) \leq \beta_i \leq cond(A_i^t A_i)$.□

There is much empirical evidence to suggest that is very rare for β_i to considerably differ from $cond(A_i^t A_i)$ ([2], p.114).

In what follows we will describe the block partition algorithm. We shall denote $I = \{1, \ldots, m\}$, $I_a = \{h : a_h^t$ row of A^t assigned to some block $\}$.

Algorithm.
Initialization. Set $i = 1$. $I_a = \emptyset$
While $I_a \neq I$ **do**
Set $R = I - I_a$, $j = 1$, $I_i = \emptyset$.
Choose $l \in R$.
Set $A_i = [a_l]$, $D_1 = [1]$, $L_1^{-1} = [1]$,
$R = R - \{l\}$, $I_a = I_a \cup \{l\}$, $I_i = I_i \cup \{l\}$
While ($j < \mu$ and $R \neq \emptyset$) **do**
Choose $l \in R$
Compute $l_{j+1} = D_j^{-1} L_j^{-1} A_i^t a_l$,
$\delta_{j+1} = 1 - l_{j+1}^t D_j l_{j+1}$
If $1/\delta_{j+1} < \kappa$ **then**
Update $A_i = \left[\begin{array}{cc} A_i & a_l \end{array} \right]$
$L_{j+1}^{-1} = \left[\begin{array}{cc} L_j^{-1} & 0 \\ -l_{j+1}^t L_j^{-1} & 1 \end{array} \right]$ $D_{j+1} = \left[\begin{array}{cc} D_j & 0 \\ 0 & \delta_{j+1} \end{array} \right]$
$R = R - \{l\}$, $I_a = I_a \cup \{l\}$, $I_i = I_i \cup \{l\}$
$j = j + 1$
else
$R = R - \{l\}$
end;
end;
$i = i + 1$
end;
End procedure.

4 Numerical experiments.

In this section we present some preliminary numerical results obtained with the new algorithms. The first purpose of our experiments was to compare the behavior of the ALG1 and ALG2 introduced in this paper with two versions of the parallel block method described in [10]. In order to carry out the comparisons we wrote an experimental code for each algorithm.

The second purpose was to compare *ALG2* with the methods: *BCG*, *CGNR*, *TQMR*, *GMRES(k)* from the SPARSKIT2 Library

(http://www.cs.umn.edu/Research/darpa/SPARSKIT), and also with the programs DSDCGN and DGMRES(k) from the SLATEC Library implementing respectively Conjugate Gradients with diagonal scaling and GMRES.

All numerical experiences were made with $A \in \Re^{n\times n}$ and run on a PC Pentium III, 408MHz, with 256 Mb Ram and 128 Mb Swap using FORTRAN 77 for Linux and also with a CRAY J90 PVP using sequential, vectorized and parallelized versions of ALG2. The idea was to test the numerical behaviour using very different computers.

In what follows we will present a brief description of the implemented parallel block algorithms. In all of them the intermediate directions are defined by the projection of x^k onto each one of the blocks. They differ in the ways in which weights are chosen.

If d_i^k is the direction given by the projection of x_k onto

$L_i = \{x \in \Re^n : A_i^t x = b_i, b_i \in \Re^{m_i}\}$ in the *kth* iteration, the different algorithms compared in the following experiences can be described as follows:

PACI1 (Projected Aggregate Cimmino with equal weights): From an iterate x^k, we define the direction $d^k = \sum_{i=1}^q w_i^k d_i^k$ where $w_i^k = 1/q$. The new iterate is $x^{k+1} = x^k + \lambda_k d^k$, where λ_k is defined in (2.2).

PACI2 (Projected Aggregate Cimmino with weights defined by the residuals): From an iterate x^k, we define the direction $d^k = \sum_{i=1}^q w_i^k d_i^k$ where $w_i^k = \|A_i{}^t x^k - b_i\| / \sum_{j=1}^q \|A_j{}^t x^k - b_j\|$. The new iterate is $x^{k+1} = x^k + \lambda_k d^k$, where λ_k is defined in (2.2).

ALG1: From an iterate x^k, we define the direction $d^k = \sum_{i=1}^q w_i^k d_i^k$ where $w^k = (w_1^k, \ldots, w_q^k)$ is the solution of the quadratic problem (2.5):

$$w^k = ((D^k)^t D^k)^{-1}(\|d_1^k\|^2, \ldots, \|d_{q_k}^k\|^2)^t$$

where $D^k = [d_1^k, \ldots, d_{q_k}^k]$. The new iterate is $x^{k+1} = x^k + d^k$.

ALG2 : Define the direction $d^k = \sum_{i=1}^q w_i^k \hat{d}_i^k$ where $\hat{d}_i^k = P_{v^\perp}(d_i^k)$ with $v = d^{k-1}$, and

$$(w_i^k)_{i=1}^{q_k} = ((\hat{D}^k)^t \hat{D}^k)^{-1}(\|\hat{d}_1^k\|^2, \ldots, \|\hat{d}_{q_k}^k\|^2)^t$$

is the solution of the quadratic problem (2.9), and $\hat{D}^k = [\hat{d}_1^k, \hat{d}_2^k, \ldots, \hat{d}_{q_k}^k]$. and q_k is the number of linearly independent directions. The new iterate is $x^{k+1} = x^k + d^k$.

The new splitting method described in Section 3 is used for partitioning the matrix A into blocks and for obtaining the Cholesky decomposition of

the matrices $(A_i{}^t A_i)^{-1}$ used for computing projections. The intermediate directions used in all algorithms are calculated by means of the projectors obtained from the preprocessing.

For the block splitting procedure specified in Step 0 of *A*LG1 and *A*LG2 we used $\kappa = 10^5$. The maximum number of rows μ allowed for each block has been chosen as a function of the problem dimension. The stopping conditions are either the residual $\|A^t x^k - b\|^2$ is less than 10^{-9} or when more than ITMAX iterations have been performed.

In *A*LG1 and *A*LG2, the quadratic problems (2.5) and (2.9) are solved by means of the Cholesky decomposition of the involved matrices. In order to guarantee the numerical stability of *A*LG1 and *A*LG2, the Cholesky decomposition is computed recursively adding up only intermediate directions such that the estimates of the condition numbers of $(D^k)^t D^k$ and $(\bar{D}^k)^t D^k$ do not exceed the upper bound $\kappa = 10^{10}$.

Test problems. The first set of problems consisted of solving linear systems $A^t x = b$ with 500 equalities and 500 unknowns, where A in each case is a matrix whose entries were randomly generated between [-5,5] and $b = A^t e$ with $e = (1, \ldots, 1)$ to ensure the consistency of $A^t x = b$. The starting point was a random vector with all of its components belonging to [-1,1]. The maximum number of rows μ allowed for each block was 50.

The time required by the block splitting algorithm was 2.6 seconds. In all cases matrices were splitted into 10 blocks; in other words at most 10 directions had been combined in each iteration.

Figure 1 shows the total average time (preprocessing included), in minutes and seconds required to achieve convergence in 10 test problems.

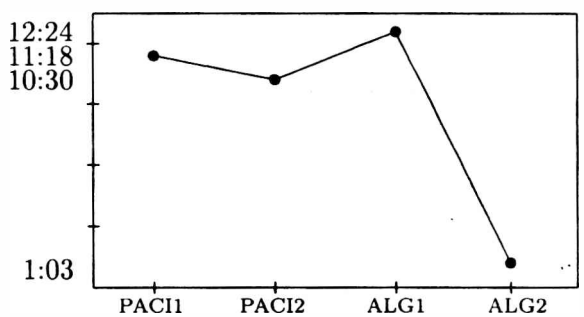

Figure 1. Average time in minutes and seconds for solving 10 random problems with each algorithm.

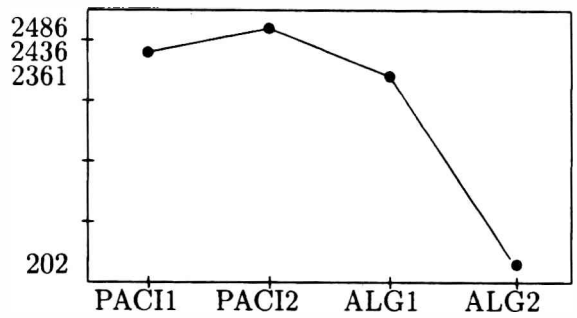

Figure 2. Average number of iterations used by each algorithm for the test problems.

Figure 2 shows the average number of iterations each method used for achieving convergence in all test problems.

The same problems were run using DGMRES(k)(GMRES), with k=10 and k=20, and DSDCGN(CG) from the SLATEC Library. GMRES failed in all problems due to "stagnated residual" and CG used an average of 194 iterations with an average time of 1 minute.

In all tests, the norms of the final residuals were of the same order for all methods. The CPU time required by ALG2 was the least for reaching convergence although it was similar to the one of DSDCGN(CG).

Therefore, all remaining experiences were carried out with ALG2.

In order to test the convenience of using the block splitting algorithm, we run ALG2 with the problem $A^t x = b$ where $A \in \Re^{n \times n}$ is the Hilbert matrix defined by $a_{ij} = 1/(i + j - 1)$, $i, j = 1, \ldots, n$, whose condition number is of the order $exp(3.5n)$. Consistency was guaranteed by defining $b = Ae$ with $e = (1, \ldots, 1)$. The starting point was $x^0 = (0, \ldots, 0)$. When using a dimension of 100, $\mu = 20$, and $\kappa = 10^5$, the block splitting algorithm partitioned the system into 31 blocks of different sizes, with an average of 3.2 rows each.

Figure 3 shows the number of rows of the different blocks and the frequency with which those numbers appear in the resulting splitting.

m=100	n. of rows	8	6	5	4	3	2	1
$\mu = 20$	n. of blocks	1	1	3	5	11	8	2

Figure 3 Number of rows per block and their frequencies

Convergence was achieved in 1 iteration with a final residual of $\|A^t x - b\| = 10^{-7}$ and an error $\|x - x^*\| = 10^{-4}$.

When for the same problem, the partition is made using blocks of 4 rows sequentially assigned, the condition number of the matrices involved in the projections turned out to be greater than 10^{10} for some blocks, and the resulting numerical errors affected convergence.

This shows that the block splitting algorithm is a very useful tool for dealing with numerically unstable problems.

In order to test ALG2 on dense and ill conditioned problems, we generated a family of systems where A^t was a non-symmetric matrix with $m = n = 1200$, defined by $a_{ij} = 1$ *if* $i \geq j$ and $a_{ij} = t$ *if* $i < j$ and $b = A^t c$ with $c = (c_i)$, c_i randomly chosen in [-1,1], to ensure the consistency of $A^t x = b$. The starting point was $x^0 = (0, \ldots, 0)$ and for the block sizes we used $\mu = 100$.

Comparisons with DGMRES(10) and DSDCGN(CG) were made for $t = 10$ (Figure 4), $t = 0.1$(Figure 5) and $t = .0001$(Figure 6). The average preprocessing time in ALG2 was 22 seconds.

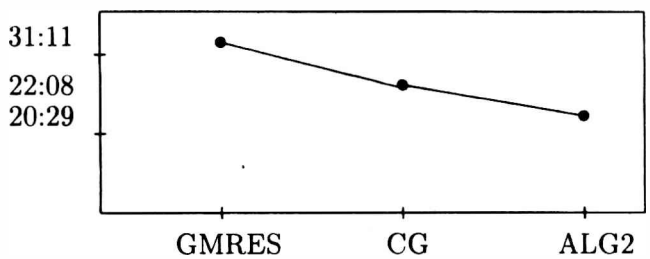

Figure 4. Total CPU time in minutes and seconds with $t = 10$

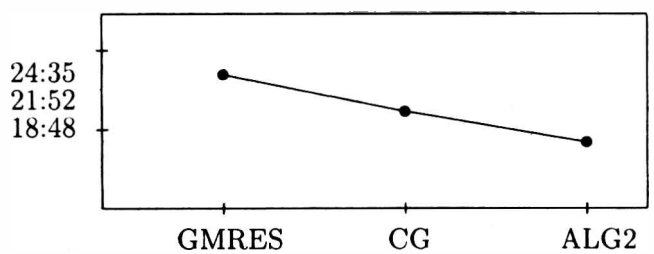

Figure 5. Total CPU time in minutes and seconds with $t = 0.1$

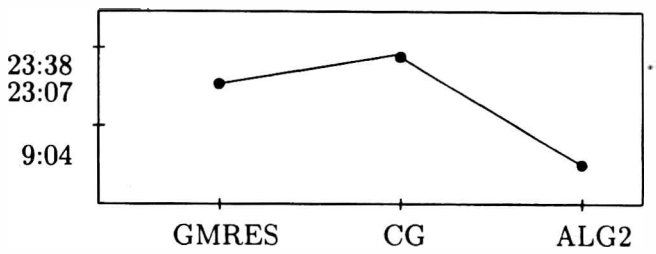

Figure 6. Total CPU time in minutes and seconds with $t = 0.0001$

Our code was very competitive in speed for these dense problems.

The previous figures show that when t decreases the total CPU time of ALG2 also goes down in spite of the condition number increases. This shows that ALG2 has a very good performance for dense ill-conditioned problems.

For testing the robustness of algorithm ALG2, the problems $P1 - P6$ proposed in [3], were run. Those problems arise from the discretization, using central differences, of elliptic partial differential equations of the form $au_{xx} + bu_{yy} + cu_{zz} + du_x + eu_y + fu_z + gu = F$, where $a - g$ are functions of (x, y, z) and the domain is the unit cube $[0, 1] \times [0, 1] \times [0, 1]$.

The Dirichlet boundary conditions were imposed in order to have a known solution against which errors can be calculated. When the discretization is performed using n_1 points in each direction, the resulting non-symmetric system is of order $n = n_1^3$, and therefore the dimension grows rapidly with n_1 when the grid is refined. If a grid of size $n_1 = 24$ is used, it leads to a problem of dimension $n = m = 13824$.

As mentioned in [3], CG without preconditioning and GMRES(k) with different values of $k, 10 \leq k \leq 40$, fail for problem P3. ALG2 was able to solve all test problems proposed in [3] for $n = 13824$:

P1 : $\Delta u + 1000u_x = F$ with solution $u(x, y, z) = xyz(1 - x)(1 - y)(1 - z)$.

P2 : $\Delta u + 10^3 e^{xyz}(u_x + u_y - u_z) = F$ with solution $u(x, y, z) = x + y + z$.

P3 : $\Delta u + 100xu_x - yu_y + zu_z + 100(x + y + z)(u/xyz) = F$ with solution $u(x, y, z) = e^{xyz} \sin(\pi x) \sin(\pi y) \sin(\pi z)$.

P4 : $\Delta u - 10^5 x^2(u_x + u_y + u_z) = F$ with solution idem P3.

P5 : $\Delta u - 10^3(1 + x^2)u_x + 100(u_y + u_z) = F$ with solution idem P3.

P6 : $\Delta u - 10^3((1 - 2x)u_x + (1 - 2y)u_y + (1 - 2z)u_z) = F$ with solution idem P3.

We ran the problems with all available algorithms in SPARSKIT2, and GMRES was used with all available preconditioners.

We present in the following Table 1 the obtained results for Problems P1-P6 with $n_1 = 24$, $n = 13824$, $\mu = m_i = n_1 * n_1$. The starting point was $x^0(i) = 0$ for $i = 1, .., n$, using the following notation:

Iter: number of iterations

Rsn: $||A^t x_s - b||_2$

Error: $||x_s - x^*||_2$

CPU: time measured in seconds

In the second row corresponding to each problem we indicate the method from SPARSKIT2 which obtained the best results and the sort of preconditioner (if one has been used).

In the third row corresponding to each problem we indicate the result of Kaczmarz method(KACZ), which was used due to its popularity in image reconstruction problems like computed tomography due to its robustness.

The CPU time required by ALG2 for block splitting and calculate $(L_i D_i L_i^t)^{-1}$. was 445 sec. for each problem. We include in the CPU column the execution time for ALG2 once the block partition has been performed.

	Meth	Iter	Error	Rsn	CPU
	ALG2	9	3.4d-6	1.4d-5	34.
P1	GMRES(P)	13	2.7d-8	3.5d-7	1.
	KACZ	617*n	2.4d-5	5.2d-5	1003.
	ALG2	128	6.4d-6	2.8d-5	479.
P2	GMRES	826	1.4d-7	9.9d-7	61.
	KACZ	1001*n	9.1d-2	2.3d-2	2500.
	ALG2	616	6.5d-5	3.1d-5	2287.
P3	CGNR	10000	5.9d-3	7.1d-5	57.
	KACZ	1001*n	9.2d-4	4.0d-5	3287.
	ALG2	442	9.0d-6	3.1d-5	1515.
P4	GMRES(P)	594	1.7d-6	3.1d-5	54.
	KACZ	1001*n	3.4d-1	9.9d-2	2500.
	ALG2	12	7.9d-6	2.6d-5	51.
P5	BCGSTAB(P)	19	7.4d-8	4.6d-7	4.
	KACZ	1001*n	2.1d-5	4.8d-5	1560.
	ALG2	32	2.8d-6	2.5d-5	110.
P6	BCGSTAB(P)	35	4.3d-8	6.1d-7	5.
	KACZ	170*n	1.3d-5	3.2d-5	325.

Table 1

Meth(P): with preconditioner

In Table 2 we describe, for each method in SPARSKIT2, the ratio between the number of problems P1-P6 successfully solved and the total (with or without preconditioning),using the notation
NPREC:= without preconditioning
WPREC:= with preconditioning
RSP := ratio of solved problems

Method	NPREC	WPREC	RSP
BCG	4/6	4/6	4/6
DBCG	4/6	4/6	4/6
CGNR	5/6	4/6	5/6
BCGSTAB	3/6	3/6	3/6
TFQMR	3/6	4/6	4/6
FOM	4/6	3/6	4/6
GMRES(*)	4/6	4/6	5/6
FGMRES	4/6	3/6	4/6
DQGMRES	4/6	3/6	4/6

Table 2.

(*): with all different preconditioners

Just for illustrative purposes we report in the following some numerical results obtained with a parallel implementation of ALG2 run on a CRAY J90 PVP(Parallel Vector Processing) with the support and technical assistance of the Federal University of Rio de Janeiro, Brazil. The following results compare a simulated sequential version using only one processor and inhibiting vectorization, another version using exclusively vectorization, and lastly a parallel implementation exploiting the fully simultaneous features of ALG2 and the corresponding synchronization of processes.

In the next table the CPU time in seconds obtained with the three versions for problems P1-P6, now run with $n_1 = 6$, $n = 216$, are given. Those figures were restricted by the computational resources available, but show the differences between the sequential version and the other two. Since the problem sizes are small, the overhead arising from the parallelization affects the total CPU time when the number of iterations is low and therefore in those cases, the vectorized version seems to be better.

When increasing the dimension the gap between the vectorized and parallel versions decreases. For instance, for $n_1 = 9$, $n = 729$, in the problems P1 and P3 , the parallelized version is the best. We show those results in the Table 3 and 4, considering the following notation:

AE1: sequential sparse

AE2: vectorized sparse

AE3: parallel sparse

Maximum number of rows per block(μ): n_1^2

Tolerance for row acceptance: 10^4

Stopping criteria: $Rsn^2 < 10^{-9}$ for odd-numbered problems, and $Rsn^2 < n * 10^{-9}$ for even-numbered problems.

n=216	Version	Preprocessing time	Processing time	Rsn	Iter
P1	AE1	0.226	0.008	3.653d-6	2
	AE2	0.007	0.003	3.651d-6	2
	AE3	0.111	0.003	3.651d-6	2
P2	AE1	0.225	20.336	6.623d-3	469
	AE2	0.007	6.359	6.623d-3	469
	AE3	0.009	4.472	6.623d-3	469
P3	AE1	0.227	0.254	2.964d-5	6
	AE2	0.007	0.008	2.964d-5	6
	AE3	0.009	0.009	2.964d-5	6
P4	AE1	0.227	11.946	6.294d-3	275
	AE2	0.007	3.750	6.300d-3	275
	AE3	0.009	2.442	6.808d-3	275
P5	AE1	0.225	0.169	1.103d-7	4
	AE2	0.007	0.005	1.103d-7	4
	AE3	0.009	0.004	1.103d-7	4
P6	AE1	0.228	4.517	6.610d-3	104
	AE2	0.007	1.398	6.610d-3	104
	AE3	0.009	0.987	6.610d-3	104

Table 3. Results using a Cray J90 PVP.

n=729	Version	Preprocessing time	Processing time	Rsn	Iter
P1	AE1	3.233	1.292	5.323d-4	5
	AE2	0.742	0.380	5.323d-4	5
	AE3	0.918	0.367	3.062d-5	5
P3	AE1	3.276	46.909	3.138d-5	177
	AE2	0.742	13.458	3.138d-5	177
	AE3	0.912	6.492	3.138d-5	177

Table 4. Results using a Cray J90 PVP.

Conclusion: None of the methods in SPARSKIT2 is able to solve all problems. and in order to get the solution it is necessary to try different algorithms with different preconditioners. On the other hand the new algorithm solved all problems, and its efficiency can be highly increased using a parallel implementation due to its fully simultaneous properties. Therefore. how to combine the speed of the Krylov subspace algorithms with the robustness of the projected aggregation methods seems to be an interesting field of research.

Acknowledgments. To U.M. García-Palomares for having introduced us to this field, and to Y. Censor who "forced" us to write up our ideas in a much clearer way, and for having reviewed this paper suggesting some changes which improved its readibility. We are very grateful to both of them for many enlightening discussions during their visits to Argentina.

References

[1] R. AHARONI AND Y. CENSOR, *Block-iterative projection methods for parallel computation of solutions to convex feasibility problems.* Linear Algebra Appl., 120 (1989), pp. 165-175.

[2] A. BJÖRCK, *Numerical Methods for Least Squares Problems.* SIAM. Filadelfia, 1996.

[3] R. BRAMLEY AND A. SAMEH, *Row projection methods for large nonsymmetric linear systems.* SIAM J. Sci. Statist. Comput.. 13 (1992), pp. 168-193.

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

[5] Y. CENSOR AND S. ZENIOS, *Parallel Optimization:Theory and Applications*, Oxford University Press, New York. 1997.

[6] G. CIMMINO, *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari*, Ric. Sci., 16 (1938), pp. 326-333.

[7] R.FLETCHER, *Practical Methods of Optimization* . J.Wiley and Sons. Second Edition. New York, 1987.

[8] U. M. GARCÍA-PALOMARES, *A class of methods for solving large convex systems*, Oper. Res. Lett., 9 (1990), pp. 181-187.

[9] U. M. GARCÍA-PALOMARES, *Projected aggregation methods for solving a linear system of equalities and inequalities*. in Mathematical Research. Parametric Programming and Related Topics II. Akademie-Verlag 62, Berlin, 1991. pp. 61-75.

[10] U. M. GARCÍA PALOMARES, *Parallel projected aggregation methods for solving the convex feasibility problem*, SIAM J. Optim., 3 (1993). pp. 882-900.

[11] 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), pp. 1-24.

[12] A. S. HOUSEHOLDER AND F. L. BAUER, *On certain iterative methods for solving linear systems*, Numer. Math.. 2 (1960). pp.55-59.

[13] S. KACZMARZ, *Angenäherte Auflösung von Systemen linearer Gleichungen*. Bull. Intern. Acad. Polonaise Sci. Lett.. 35 (1937). pp. 355-357.

[14] Y. SAAD AND M. SCHULTZ, *Conjugate gradient-like algorithms for solving nonsymmetric linear systems*, Math. Co., 44 (1985), pp.417-424.

[15] H.D.SCOLNIK, *New Algorithms for Solving Large Sparse Systems of Linear Equations and their Application to Nonlinear Optimization.* Investigación Operativa, 7 (1997), pp. 103-116.

[16] H. VAN DER VORST, *A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631-644.