Electronic Transactions on Numerical Analysis.
Volume 20, pp. 253-275, 2005.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

A LINEAR ACCELERATION ROW ACTION METHOD FOR PROJECTING ONTO SUBSPACES*

GLENN APPLEBY† AND DENNIS C. SMOLARSKI†

Abstract. This article describes an extension of projection or "row action" methods proposed first by Kaczmarz and by Cimmino. The method of this article constructs a line through two centroids computed by a modified Cimmino procedure and uses the actual or approximate intersection of this line with one of the hyperplanes associated with rows of the system matrix as an approximation to the solution. Comparisons are made with similar methods described by Pierra and Dax.

Key words. Linear systems, projection methods, row action methods, iterative methods, Kaczmarz, Cimmino, Pierra, Dax

AMS subject classifications. 15A06, 65F10, 65J05, 90C25, 90C55

1. Introduction. This article describes an iterative method, which will be called the "linear acceleration" (LA) method, to project a vector f in $\mathbb{R}^m$ onto the affine subspace of solutions to $Gu = c$ where G is $n \times m$ and, typically, $n < m$. This method builds upon algorithms first described by S. Kaczmarz [27] and by G. Cimmino [18], in which approximate solutions are successively projected onto a set of hyperplanes. Although terminology varies, such algorithms are often called *row projection* (cf. [9]) or *row-action* (cf. [10]) methods.

The LA method of this paper extends the algorithms of Kazmarcz and Cimmino (summarized in section 3 below) in this way: First, a vector f is projected independently onto each one of a finite set of hyperplanes. Next, these projected points are averaged to obtain a centroid $x_{\text{intermed.}_A}$ with which the projection step is repeated and a new centroid $x_{\text{intermed.}_B}$ computed. (We will make use of the term *centroid* rather than *barycentre* used by other authors, but we will consider the two terms equivalent in our context.) Then, unlike the original Kaczmarz and Cimmino algorithms, the LA method determines the line through the two centroids $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$, and, then, by moving along this line, obtains a new approximate solution point x_{new}.

A *saddle point* system $\mathcal{A}\widehat{x} = b$ is one in which, in its most general form, $\mathcal{A}$, $\widehat{x}$, and b have block structures as follows:

$$\begin{pmatrix} A_{m\times m} & G^T \\ G_{n\times m} & -D_{n\times n} \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} f \\ c \end{pmatrix} \tag{1.1}$$

(cf. [5, 8, 21]). A saddle point system can be seen as one way of formulating the problem solved by the LA method, namely, projecting a vector onto a constraint space. In our studies, we let A be the identity matrix and D be the zero matrix. With these specifications, equation (1.1) yields the following two systems of equations,

$$x + G^T\lambda = f \tag{1.2}$$

and

$$Gx = c \tag{1.3}$$

and the second system can be interpreted as a constraint space onto which f is projected. Thus, x (the upper part of the solution vector $\widehat{x}$) is the projection of f (the upper part of the

*Received August 6, 2003. Accepted for publication November 17, 2005. Recommended by M. Benzi.
†Department of Mathematics and Computer Science, Santa Clara University, Santa Clara, California 95053-0290 (gappleby@scu.edu, dsmolars@math.scu.edu).

ETNA
Kent State University
etna@mcs.kent.edu

constant vector b) onto the constraint space determined by $Gx = c$. In a situation such as this, what is desired is the value of x alone; the value of λ may be ignored, although it is necessary to compute it in some algorithms (cf. [5, p. 3].

Row projection methods may, in fact, be also used to solve a linear system and this is the context for the Kaczmarz and Cimmino algorithms. For such applications, row projection methods are usually not competitive in comparison to other methods. Nevertheless, projection methods are still of active interest in specialized areas, such as computed tomography where all the data may not be available at the start of a solution algorithm (cf. [4, 23, 25, 26, 28, 29, 32], and the various works of Censor [10, 11, 12, 13, 14, 15, 16, 17]). Since the Cimmino method is inherently parallelizable, variants of this method have also been of special interest (cf. [12, 17]).

Section 2 provides some background and a simple example of how projectors are used in this paper, and section 3 reviews the history of projection methods and describes the general approach to accelerating such methods. Section 4 describes the LA method and provides a proof of the convergence of one version of the LA method along with other details, and section 5 describes implementation issues. Section 6 shows the exact convergence of the LA method in one iteration for orthonormal spaces. Section 7 reports on numerical test results including those comparing the LA method with similar methods by G. Pierra [33] and A. Dax [19, 20], and section 8 briefly comments on the use of such projection methods with problems in computed tomography. The final sections include references to other approaches, a brief evaluation, and appendices describing alternative versions of the LA method.

2. Background and Example. Saddle point problems arise in a variety of applications (see [5, pp. 5–14] for some examples). In such problems, the solution of one linear system of m equations, $Ax + G^T\lambda = f$ (where f is in $\mathbb{R}^m$), is constrained by a set of conditions forming a second, underdetermined system of n equations, $Gx - D\lambda = c$ (where c is in $\mathbb{R}^n$). When $A = I_{m\times m}$ and $D = 0_{n\times n}$, these two linear systems reduce to the form given above in eqs. (1.2) and (1.3).

Let the i^{th} row of G, $(\gamma_{i1}, \gamma_{i2}, \ldots, \gamma_{im})$, be indicated by the vector g_i^T. If $c = (c_1, c_2, \ldots, c_n)^T$, then the vector g_i is orthogonal to the hyperplane $g_i^T x = c_i$. Thus, a point (or vector) $x^* \in \mathbb{R}^m$ is in the i^{th} hyperplane if and only if $g_i^T x^* = \langle g_i, x^*\rangle = c_i$. (When there is no possibility of confusion from context, we will simplify notation by referring to row g_i^T of G as a hyperplane.) Let $\mathcal{G}$ be the subspace obtained as the intersection of the n hyperplanes in $\mathbb{R}^m$ associated with the rows of the matrix G. Thus, $x \in \mathcal{G}$ if and only if $Gx = c$.

Computing the *projection* of some point (or vector) $f \in \mathbb{R}^m$ onto $\mathcal{G}$ is equivalent to writing f as $x + v$ where $x \in \mathcal{G}$ and $v \in \mathcal{G}^\perp$. The latter condition amounts to $v \in \text{Span}\{g_1, \ldots, g_n\}$, and so it suffices to write $v = G^T\lambda$ for some appropriate vector $\lambda \in \mathbb{R}^n$.

Thus, the problem of finding the projection of some $f \in \mathbb{R}^m$ onto $\mathcal{G}$ may be formulated as that of solving the saddle point system $\mathcal{A}\widehat{x} = b$, when $\mathcal{A}$, $\widehat{x}$, and b can be written in block form as

$$\begin{pmatrix} I_{m\times m} & G^T \\ G & 0_{n\times n} \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} f \\ c \end{pmatrix}, \tag{2.1}$$

and then extracting the upper block, x, of $\widehat{x}$ as the desired projected point in $\mathcal{G}$. We assume that G is $n \times m$ with $n < m$ and, thus, $\mathcal{A}$ is $(m+n) \times (m+n)$.

Assuming the solution to $\mathcal{A}\widehat{x} = b$ is $\widehat{x} = (x, \lambda)^T$, it follows (cf. equation (1.3)) that $Gx = c$, thus confirming that x does, in fact, lie in the constraint subspace $\mathcal{G}$ associated with G. Similarly, equation (1.2) yields $x + G^T\lambda = f$ indicating that f can be written as $x + v$ where x satisfies $Gx = c$ and $v = G^T\lambda$, which is in $\mathcal{G}^\perp$. Consequently, x is the projection of f onto $\mathcal{G}$.

If $b = (f, c)^T$ for c non-zero, it can be shown that, when G is of full rank, the desired solution $x = [I - G^T(GG^T)^{-1}G]f + G^T(GG^T)^{-1}c$. If $c = 0$, the solution reduces to $x = [I - G^T(GG^T)^{-1}G]f$. We can therefore write x as Pf where the projector $P = I - G^T(GG^T)^{-1}G = I - G^\dagger G$ and $G^\dagger$ is the Moore-Penrose pseudo-inverse of G. If G consists of a single row, g_i^T (corresponding to a single hyperplane), $GG^T = g_i^T g_i = \|g_i\|^2$ is a scalar, and the projector P_i (corresponding to row g_i) can be written as

$$P_i = I_m - \frac{g_i g_i^T}{\|g_i\|^2} \tag{2.2}$$

(cf. [22, p. 75], [37, pp. 41–46]).

The following simple example may help demonstrate the geometrical interpretation of this formulation of the problem. This system

$$\left(\begin{array}{ccc|cc} 1 & 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 0 \\ \hline 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 \end{array}\right) \left(\begin{array}{c} x \\ \hline \lambda \end{array}\right) = \left(\begin{array}{c} 1 \\ 2 \\ 3 \\ \hline 0 \\ 0 \end{array}\right)$$

yields a value for x of $(0, 0, 3)^T$. Note that the rows of G, contained in the last two rows of $\mathcal{A}$, are the coefficients of the planes $2t + y = 0$ and $t + 2y = 0$ whose intersection is the z-axis (in a tyz coordinate system). Thus, $x = (0, 0, 3)^T$ is, in fact, the projection of $f = (1, 2, 3)^T$ onto the z-axis. Also note that the value of λ is $(0, 1)^T$, a value not particularly useful but which, in many algorithms, must be computed with some accuracy to obtain x.

If, however, the constant vector f equals $(1, 2, 3, 1, 1)^T$, then x would equal $(1/3, 1/3, 3)$ and λ would equal $(-1/9, 8/9)^T$. The value of x here is consistent with the earlier value, since the earlier value was based on $Gx = 0$ resulting in an intersection of the two planes at $(0, 0)^T$, whereas the current value is based on $Gx = (1, 1)^T$ resulting in an intersection of the two planes at $(1/3, 1/3)^T$.

We shall consider the general problem of computing projections of vectors onto the intersection of hyperplanes. Many of our methods could be naturally extended to the more general problem of computing projections onto the intersections of convex sets. Note that, in our formulation of the problem, it will not be necessary to assume that the rows of G are independent.

3. History of Projection Methods. Row projection or row-action methods have been widely studied (cf. [10, 36]). Kaczmarz [27] was the first to show that iterated projections of a point onto intersecting hyperplanes will converge to the subspace of intersection (cf. [7, pp. 186–88], [10, p. 450], [14], [36]). This approach was rediscovered by Gordon et al. [23] and was called ART (algebraic reconstruction technique) (cf. [10, p. 451]). The basic approach of Kaczmarz can be modified by using a relaxation parameter α resulting in a point between the hyperplane and the point being projected (if $0 < \alpha < 1$) or between the hyperplane and the reflection of the point about the hyperplane (if $1 < \alpha < 2$).

Cimmino [18] proposed a method conceptually similar to Kaczmarz's approach, with this major difference: Instead of successive projections, Cimmino's algorithm projects the same point toward all the hyperplanes and then computes the centroid derived from these projected points. In actuality, Cimmino's original method determines the *reflection* of a point relative to each hyperplane, using a relaxation parameter of $\alpha = 2$. If one envisions the set of hyperplanes as all cutting a hypersphere through its center (which is the solution point, i.e., the point of intersection of all the hyperplanes), then reflecting a point on the surface of the hypersphere about a hyperplane (by means of an orthogonal projection through the hyperplane), results in another point also on the surface of the hypersphere. It is obvious that the centroid of a set of points on the surface of a hypersphere is interior to the hypersphere. Considering the centroid to be a point on the surface of a new hypersphere, concentric with the original hypersphere and smaller, the location of this point also determines the diameter of the new hypersphere. We can thus iterate these steps, slowly converging to the common center of all the hyperspheres, which is the desired solution point (cf. [18], [7, p. 187]).

We note in passing that the Cimmino and Kaczmarz methods are examples of linear stationary iterative methods that are related to other well known methods, such as Richardson iteration or the Jacobi method (cf. [6], [19, p. 612], [35, pp. 8–10]).

3.1. General Approach of Acceleration Methods. Pierra proposed an acceleration technique for algorithms based on projections and centroids (termed "barycentres" in [33]). Pierra notes that the "method of barycentres . . . is not very fast" and thus proposes the "extrapolated parallel projection method" (E.P.P.M.) [33, p. 99]. This method uses the line through the original point and a centroid (barycentre) and computes the average of the intersections of this line with all the hyperplanes to determine the next point [33, p. 101]. At regular intervals, the approximate solution point is centered using a relaxation factor.

Dax provided an alternative acceleration technique also based on projections and centroids [20]. This method also uses a line through the original point and a centroid and determines a distance to accelerate along this line. This distance is computed via the gradient of the residual $||Gx_{\text{new}} - c||$, where x_{new} is on the acceleration line, resulting in a point on the line closest to the actual solution point x.

The LA, Pierra, and Dax algorithms all follow a similar approach. Unlike the original Kaczmarz algorithm, in which each projected point is used as the base point for the next projection, all three of these methods, similar to the Cimmino algorithm, compute all the projections first and then determine the centroid.

All three methods determine a line along which the acceleration takes place and determine a distance to move along this acceleration line. In general, these methods proceed in this way: Given point x_{old}, one or more Cimmino-like steps are executed in which projections of x_{old} are computed and an intermediate point, the centroid $x_{\text{intermed.}_A}$, is obtained. A second intermediate centroid, $x_{\text{intermed.}_B}$, can also be computed. One then accelerates converging to the desired solution by moving a specific distance δ along the line through $x_{\text{intermed.}_A}$ with direction $x_{\text{intermed.}_A} - x_{\text{old}}$ or $x_{\text{intermed.}_B} - x_{\text{intermed.}_A}$. The three algorithms differ as to (1) whether one uses x_{old} or $x_{\text{intermed.}_A}$ as the base point from which one determines x_{new}, (2) how the distance δ is computed, and (3) how the direction of the line is computed.

Thus, for LA,

$$x_{\text{new}} = x_{\text{intermed.}_A} + \delta \cdot (x_{\text{intermed.}_B} - x_{\text{intermed.}_A})$$

for Pierra,

$$x_{\text{new}} = x_{\text{old}} + \delta \cdot (x_{\text{intermed.}_A} - x_{\text{old}})$$

and for Dax,

$$x_{\text{new}} = x_{\text{intermed.}_A} + \delta \cdot (x_{\text{intermed.}_A} - x_{\text{old}}).$$

The LA method described below incorporates the basic projection techniques developed by Kaczmarz and Cimmino and attempts to find an approximate solution point by determining a point on the line that passes close to the actual solution point, similar to the approaches proposed by Pierra or Dax.

ETNA
Kent State University
etna@mcs.kent.edu

3.2. Cimmino-projection Methods and Parallel Programming. The Cimmino-like projections used by the various algorithms are order independent and thus their computations can be coded to occur in parallel whenever possible (cf. [2, p. 49], [9, p. 174], [12]; also see the recent work of Censor and Zenios [17]). This inherent parallelization property is one of the reasons that makes these methods worthy of further study.

4. The Linear Acceleration (LA) Method. The LA method was motivated by an experimental study of the convergence of centroids computed by using the Cimmino algorithm on a constrained system. A sequence of centroids computed according to Cimmino's algorithm will converge to the solution x contained in the intersection subspace $\mathcal{G}$. Experimental data have shown that these centroids converge toward a line in n-space that intersects the desired solution. The LA method attempts to approximate the curve on which these centroids lie by a straight line, and then moves along this line in the direction of the solution point x to obtain an approximate solution, x_{new}. The use of an acceleration line through two centroids is the primary difference between the LA method and other algorithms

Figure 4.1 contains a 2-dimensional depiction that attempts to illustrate the rationale motivating the proposed LA method. The line through centroids $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$ intersects one of the hyperplanes near the common intersection $x \in \mathcal{G}$. Moving along this line a specific distance results in the new approximation, x_{new}.

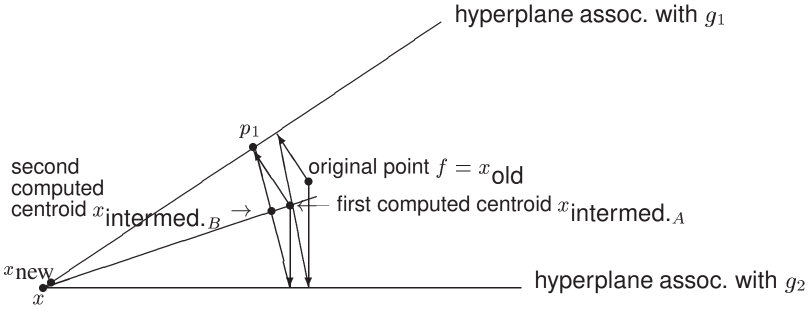


FIG. 4.1.

Thus, the LA method consists of the following steps (details below in section 5):

ALGORITHM 1 (LINEAR ACCELERATION): *Input*: g_i^T (rows of the matrix G), f (the initial point to be projected onto $\mathcal{G}$). *Output*: x_{new}, the approximate projection of f onto $\mathcal{G}$.

1. A starting point x_{old} (initially equal to f) is projected onto each hyperplane associated with the rows g_i^T of G.
2. The points of projection are averaged to produce a centroid $x_{\text{intermed.}_A}$.
3. The projection step is repeated, this time projecting the computed centroid $x_{\text{intermed.}_A}$ onto all the hyperplanes.
4. The new points of projection are averaged to produce a new centroid $x_{\text{intermed.}_B}$
5. *(The LA step.)* The line through centroids $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$ is computed and a point x_{new} on this line near the desired solution x is determined by some method.
6. Repeat steps 1–5 as desired, after first resetting x_{old} to x_{new}.

We note below in section 7.3 that the Cimmino-type projections used to compute the centroids of steps 2 and 4 is typically repeated several times before designating points as

ETNA
Kent State University
etna@mcs.kent.edu

$x_{\text{intermed.}A}$ and $x_{\text{intermed.}B}$.

4.1. Determining x_{new} on the Acceleration Line. To determine a point x_{new} on the line through $x_{\text{intermed.}A}$ and $x_{\text{intermed.}B}$ near the desired solution x, various options are possible, any one of which can be used in step 5 of Algorithm 1. Three distinct methods were studied; two of them are based on the *intersection* of the acceleration line with a hyperplane, and the third is based on travelling a specified *distance* along the line. We report on the procedure in which x_{new} is the point of intersection of the acceleration line with the *nearest* hyperplane and label the algorithm when using this distance procedure as LA_N. Other distance procedures will be commented on briefly in the appendix.

4.1.1. Algorithm LA_N: Intersecting with the Nearest Hyperplane. If the two centroids $x_{\text{intermed.}A}$ and $x_{\text{intermed.}B}$ are well-centered between enclosing hyperplanes, a line through $x_{\text{intermed.}A}$ and $x_{\text{intermed.}B}$ would be expected to intersect the hyperplanes near the common set of intersection as depicted in Figure 4.1. Determining the hyperplane that produces a point closest to the desired solution point x is non-trivial, however. One could choose a specific hyperplane to use in determining the intersection point (an approach described in the appendix). It is possible that the new point of intersection in this case will be further from the desired solution point than a point of intersection with some other hyperplane (cf. Figure 4.2), and might, in some cases, lead to a cyclic non-convergence.

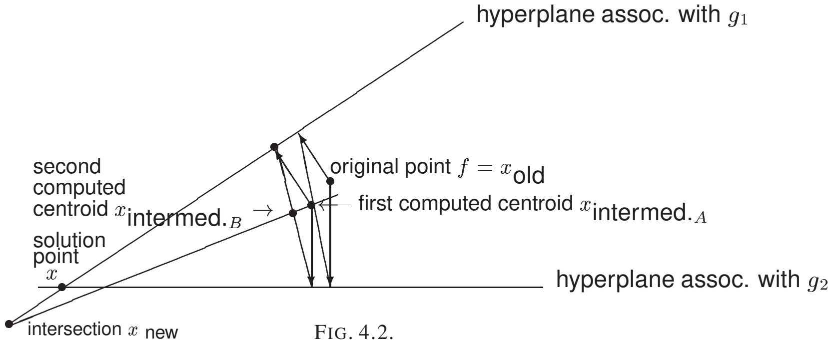


FIG. 4.2.

To avoid any possibility of obtaining a distant point of intersection, an alternative approach is to compute the intersection of the projection line with the *nearest* hyperplane and use this intersection point as the next approximate solution. In practice (as will be noted in section 5.3), one computes the intersection point x_{new} by first computing the distance δ to the nearest hyperplane and then moving from centroid $x_{\text{intermed.}A}$ in the direction $w = x_{\text{intermed.}B} - x_{\text{intermed.}A}$ by using the formula $x_{\text{new}} = x_{\text{intermed.}A} + \delta w$. Finding the nearest hyperplane consists only in computing the distance values δ_i along w from $x_{\text{intermed.}A}$ to the hyperplane associated with each row g_i^T of G, and then using the δ_i minimum in magnitude as δ to determine the intersection point x_{new}.

We claim that, when multiple iterations in the LA algorithm occur, distance procedure LA_N always converges. This is the case even when the centroid $x_{\text{intermed.}A}$ and $x_{\text{intermed.}B}$ are obtained via multiple Cimmino-type iterations. The proof is given as Theorem 4.1 and Corollary 4.4.

THEOREM 4.1. *Let $\mathcal{H}$ be a finite set of hyperplanes in $\mathbb{R}^m$ with non-empty intersection, $\mathcal{G} = \cap\mathcal{H}$. Let x_A be a point in $\mathbb{R}^m$, and let x_2 be the centroid obtained from x_A by averaging the projections of x_A onto all the hyperplanes $g \in \mathcal{H}$. Let x_3 be the centroid obtained by averaging projections of x_2 onto the hyperplanes $g \in \mathcal{H}$, etc., and finally set $x_B = x_k$, the*

ETNA
Kent State University
etna@mcs.kent.edu

k^{th} centroid obtained from from x_A. Define x^* to be the intersection of the ray $\overrightarrow{x_A x_B}$ with the hyperplane $g^* \in \mathcal{H}$ chosen so that x^* lies nearest to x_A among all other hyperplanes $g \in \mathcal{H}$. If $x \in \mathcal{G}$ is an arbitrary point in the intersection of the hyperplanes in $\mathcal{H}$, then $\|x^* - x\| < \|x_A - x\|$. If x_B is on the same side of the hyperplane g^* that x_A is, then $\|x^* - x\| < \|x_B - x\|$.

Before proceeding with the proof, we present two preliminary lemmas.

LEMMA 4.2. *Let H be a hyperplane in $\mathbb{R}^m$, and let Π be a plane (two-dimensional subspace) in $\mathbb{R}^m$, and let $\xi \in \mathbb{R}^m$. Let $l_H = H \cap \Pi$, a line in H. Let $P_H(y)$ denote the orthogonal projection of a vector y onto H, and similarly let $P_\Pi(y)$ denote the orthogonal projection of y onto Π. Finally, let y_H denote the orthogonal projection of $P_\Pi(\xi)$ on l_H. Then $P_\Pi(P_H(\xi))$ lies between y_H and $P_\Pi(\xi)$. (cf. Figure 4.3)*

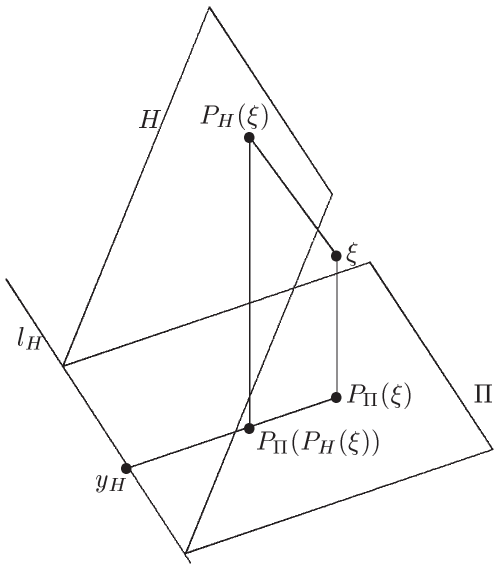


FIG. 4.3. *Depiction of projection of ξ onto plane Π and hyperplane H.*

Proof. Let $\overrightarrow{l_H}$ denote a vector that is parallel to the line l_H. Then $\overline{\xi P_H(\xi)} \perp \overrightarrow{l_H}$ by virtue of orthogonal projection, since l_H lies in H. Similarly, $\overline{\xi P_\Pi(\xi)} \perp \overrightarrow{l_H}$ since l_H lies in Π. So the plane (two dimensional subspace) containing ξ, $P_H(\xi)$, and $P_\Pi(\xi)$ is orthogonal to $\overrightarrow{l_H}$. Since $\overline{\xi P_\Pi(\xi)}$ is parallel to $\overline{P_H(\xi) P_\Pi(P_H(\xi))}$, then $P_\Pi(P_H(\xi))$ is in this plane, too, so that $\overline{P_H(\xi) P_\Pi(P_H(\xi))}$ is also orthogonal to $\overrightarrow{l_H}$, and so must lie on the line containing $P_\Pi(\xi)$ and y_H. The point $P_\Pi(P_H(\xi))$ lies between $P_\Pi(\xi)$ and y_h since $P_H(\xi)$ must lie closer to l_H (the intersection of Π and H) than does ξ, but then projecting both points onto Π must preserve this, and so $P_\Pi(P_H(\xi))$ must lie closer to y_H than does $P_H(\xi)$. ∎

LEMMA 4.3. *Let $\mathcal{H}$ be a set of n hyperplanes with a non-empty intersection, and let $x \in \cap\mathcal{H}$. Let l_{g^*} be a line through x, contained in some $g^* \in \mathcal{H}$. Let $y_{g^*} \in l_{g^*}$, $y_{g^*} \neq x$, and let ${g^*}^\perp$ be the hyperplane through y_{g^*} that is orthogonal to l_{g^*}. Suppose $\xi \in \mathbb{R}^m$ is either on ${g^*}^\perp$, or on the same side of ${g^*}^\perp$ that x is, with the further property that if Π is the plane*

ETNA
Kent State University
etna@mcs.kent.edu

(two-dimensional subspace) containing ξ and l_{g^*}, and $l^\perp = \Pi \cap g^{*\perp}$, then not only is ξ on the same side of $l^\perp$ as is x, but the orthogonal projection of ξ onto the line $l_g = g \cap \Pi$ is also on the same side of $l^\perp$ that x is, for all $g \in \mathcal{H}$. Then, the centroid $\overline{\xi} = \frac{1}{n}\sum_{g\in\mathcal{H}} P_g(\xi)$ is on the same side of $g^{*\perp}$ that x is, too. Further, each projection $P_\Pi(P_g(\overline{\xi}))$, for each $g \in \mathcal{H}$, is also on the same side of $l^\perp$ that x is.

Proof. By hypothesis, for each $g \in \mathcal{H}$, the line segment with endpoints ξ and z_g (where z_g is the orthogonal projection of ξ onto the line l_g in Π) is contained in the same half-plane (relative to $l^\perp$) that x is. By the first lemma, these segments must contain $P_\Pi(P_g(\xi))$. But this implies, for each $g \in \mathcal{H}$, that $P_g(\xi)$ is in the same half-space (relative to the hyperplane $g^{*\perp}$) as is x, so clearly the centroid $\overline{\xi} = \frac{1}{n}\sum_{g\in\mathcal{H}} P_g(\xi)$ is also on this half-space. It only remains to show that each projection of the centroid, $P_g(\overline{\xi})$, for $g \in \mathcal{H}$, projects onto Π to a point that lies in this same half-plane (relative to $l^\perp$) as x.

Fix one hyperplane $\widehat{g} \in \mathcal{H}$. Since $P_{\widehat{g}}(\overline{\xi}) = \frac{1}{n}\sum_{g\in\mathcal{H}} P_{\widehat{g}}(P_g(\xi))$, it will be enough to show that for each $g \in \mathcal{H}$ that $P_\Pi(P_{\widehat{g}}(P_g(\xi)))$ lies in the same side of $l^\perp$ that x does. Both ξ and z_g are on the same side of $l^\perp$ (i.e., on the same half-plane) as is x. So, by the first lemma, $P_\Pi(P_g(\xi))$ lies on the line between ξ and z_g, and hence is on the same half-plane as x. Similarly, $P_\Pi(P_{\widehat{g}}(\xi))$ also lies on the segment between ξ and $z_{\widehat{g}}$. Indeed, $P_\Pi(P_g(\xi))$ and $P_\Pi(P_{\widehat{g}}(\xi))$ lie on a quadrilateral in Π with vertices ξ, z_g, $z_{\widehat{g}}$ and x. Again using the first lemma, $P_\Pi(P_{\widehat{g}}(P_g(\xi)))$ must lie on the line segment in Π with endpoints $P_\Pi(P_g(\xi))$ and the orthogonal projection of $P_g(\xi)$ onto the line $l_{\widehat{g}}$. But this segment is contained within the quadrilateral, which is contained in the same half-plane as x, so we are done. ◻

Now we are ready to prove the theorem.

Proof. Let Π denote the plane (two-dimensional subspace) containing x_A, x_B and x. Denote by g^* the "nearest" hyperplane, that is, the hyperplane whose intersection with the ray $\overrightarrow{x_A x_B}$ lies closest to x_A. For each $g \in \mathcal{H}$, let $l_g = \Pi \cap g$, and let z_g denote the orthogonal projection of x_A onto l_g. Let $g^{*\perp}$ denote the hyperplane through z_{g^*} that is orthogonal to $\overline{x z_{g^*}}$, and let $l^\perp = \Pi \cap g^{*\perp}$. Identifying y_{g^*} with z_{g^*} and ξ with x_A in the second lemma, we see that all the hypotheses are satisfied (cf. Figure 4.4). Hence, the centroid x_2 obtained from x_A has the property that $P_\Pi(x_2)$ lies in the same half-plane (relative to $l^\perp$) that x does. In particular, x_2 is contained in the same half-space (relative to $g^{*\perp}$) as x. Since $P_\Pi(x_2)$ again satisfies the hypotheses of the second lemma, then the projections $z_g^{(2)}$ of $P_\Pi(x_2)$ onto each l_g, for all $g \in \mathcal{H}$, must lie in the same half-plane (relative to $l^\perp$) as ξ by the first lemma, since the segments with endpoints $P_\Pi(x_2)$ and $z_g^{(2)}$ contain $P_\Pi(P_g(x_2))$. But then x_3, the centroid obtained by averaging the projections of x_2 onto the hyperplanes $g \in \mathcal{H}$, has the property that $P_\Pi(x_3)$ satisfies the hypotheses of the second lemma, etc. We proceed until reaching $x_k = x_B$, and conclude inductively that x_B (which satisfies $x_B = P_\Pi(x_B)$ by hypothesis) lies in the same half-plane (relative to $l^\perp$) as x. This means the intersection of the ray $\overrightarrow{x_A x_B}$ with the hyperplane g^* (which occurs along the line l_{g^*} by construction) lies on the leg opposite x_A of the right triangle with vertices x_A, z_{g^*}, and x. From this we conclude that $\|x^* - x\| < \|x_A - x\|$. If x_B is on the same side of the hyperplane g^* that x_A is, then x_B lies in the interior of this right triangle, from which we conclude $\|x^* - x\| < \|x_B - x\|$. ◻

COROLLARY 4.4. *Given any initial point $x_0 \in \mathbb{R}^n$, the nearest hyperplane algorithm LA_N converges to a point $x \in \mathcal{G}$.*

Proof. We first observe not only that the method of computing successive centroids, as in the Cimmino algorithm, will converge to a point $x \in \mathcal{G}$, but also that the convergence is

ETNA
Kent State University
etna@mcs.kent.edu

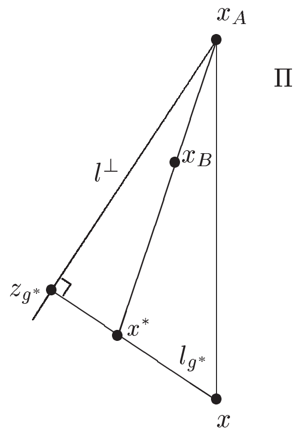


FIG. 4.4. *Depiction of plane* Π *containing* x_A, x_B, *and* x.

uniformly linear, since the rate of convergence is bounded below by the largest eigenvalue of a linear map (which must still be of absolute value less than one, since the method is convergent) (cf. [18]). For a given initial point x_0, there is a unique affine subset $\mathcal{F}$ of maximal dimension, orthogonal to $\mathcal{G}$, and containing x_0. The method of successive centroids yields a sequence of iterates all lying in $\mathcal{F}$, and converging to the single point $x \in \mathcal{F} \cap \mathcal{G}$. Inserting the LA step of computing the intersection to the nearest hyperplane keeps a point in the sequence within $\mathcal{F}$ because the line containing the two centroids lies in $\mathcal{F}$ since the centroids do. By Theorem 4.1, the projected point lies even closer to $x \in \mathcal{F} \cap \mathcal{G}$ than before, and since the convergence of the method of successive centroids is uniformly linear, the rate of convergence using the LA step cannot be any worse than the method of successive centroids alone. ◻

Procedure LA$_N$ demands that one compute the distance value δ_i for each hyperplane and find the minimum. If done in parallel, the computation of the δ_i values may not take significantly longer than computing a single distance value, but there may be a slight bottleneck in determining the minimum magnitude of all the δ_i. This may, in fact, be an acceptable overhead.

5. Implementation of the Linear Acceleration (LA) Algorithm and its Associated Formulas. The complete algorithm presented in section 4 consists of several steps. The projection steps 1 and 3 make use of the projector matrices P_i (cf. equation (2.2)) corresponding to each of the hyperplanes associated with row g_i^T of G. After computing the projections of a point onto each hyperplane, we next compute the centroids $x_{\text{intermed.}A}$ and $x_{\text{intermed.}B}$ (steps 2 and 4) and then determine the line through $x_{\text{intermed.}A}$ and $x_{\text{intermed.}B}$ (step 5). Finally, we need to determine a point x_{new} closer to the actual solution than x_{old} was (step 5) using the LA$_N$ distance procedure of section 4.1.1 (or some alternative, cf. appendix).

To simplify the discussion that follows, we will assume that the lower block c of the constant vector b is the zero vector, resulting in the constraint space $Gx = 0$. Geometrically, this amounts to shifting the space to the origin and simplifies some of the formulas (cf. sec. 2).

5.1. The Projections onto the Hyperplanes. To compute the projection of a point x^* onto the hyperplane associated with g_i for each of the n rows of G, we make use of the corresponding projector matrices, $P_i = I_m - g_i g_i^T/(g_i^T g_i) = I_m - g_i g_i^T/||g_i||^2$, where m is the number of columns of G (cf. equation (2.2)). To avoid computing and storing n different

ETNA
Kent State University
etna@mcs.kent.edu

$m \times m$ projector matrices, P_i, however, we note that

$$P_i x^* = \left(I_m - \frac{g_i g_i^T}{||g_i||^2} \right) x^* = I_m x^* - \frac{g_i g_i^T}{||g_i||^2} x^* = x^* - \frac{1}{||g_i||^2} g_i g_i^T x^*.$$

The product $g_i^T x^*$ is a scalar, and, since g_i^T would often be sparse, computing $g_i^T x^*$ is inexpensive if coded properly. Thus we have

$$P_i x^* = x^* - \frac{g_i^T x^*}{||g_i||^2} g_i \tag{5.1}$$

where $g_i^T x^*$ and $||g_i||^2$ are scalars. (Rather than computing the square of the norms of the row vectors $||g_i||^2$ at each iteration, one could compute them once prior to the iterative loop and store them for repeated use during the iteration process.)

If the constraint space includes a non-zero right hand side, $c = (c_1, c_2, \ldots, c_n)^T$, then the projector equation is

$$P_i x^* = x^* - \frac{g_i^T x^* - c_i}{||g_i||^2} g_i.$$

5.2. Optional Relaxation Parameters in the Projection Formula. One can interpret the projection formula given by equation (5.1) in this way: $P_i x_k$ (the projection of point x_k onto the hyperplane associated with projector P_i) is determined by starting at the original point x_k and moving in a direction normal to hyperplane associated with g_i by the distance $-\frac{g_i^T x^*}{||g_i||^2}$. As noted above in section 3, it is also possible to include a relaxation parameter α in this formula to vary the distance. The projection formula then becomes

$$P_i x^* = x^* - \alpha \frac{g_i^T x^*}{||g_i||^2} g_i. \tag{5.2}$$

If $\alpha = 1$, the new point is on the hyperplane (as is the Kaczmarz algorithm) and if $\alpha = 2$, the new point is the reflection of the original point with respect to the hyperplane (as in the Cimmino method). (Cf. [14] for studies involving various relaxation factors.)

5.3. Computing the Centroids $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$ and the Points of Intersection. After projecting an arbitrary point, x_k, onto the n hyperplanes g_i^T by computing $P_i x_k$ using equations (5.1) or (5.2), the n projection points p_i are averaged to determine the centroid, x_{k+1}. That is, one computes

$$\begin{aligned} x_{k+1} &= \frac{1}{n} \sum P_i x_k \\ &= \frac{1}{n} \sum (x_k - \alpha \frac{g_i^T x_k}{||g_i||^2} g_i) \\ &= \frac{1}{n} \sum (x_k) - \frac{1}{n} \sum (\alpha \frac{g_i^T x_k}{||g_i||^2} g_i) \\ &= x_k - \frac{\alpha}{n} \sum (\frac{g_i^T x_k}{||g_i||^2} g_i) \end{aligned} \tag{5.3}$$

As described above in section 4, this projection step is performed several times (as desired), with f as the initial value x_0, and results in two centroids, $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$

where $x_{\text{intermed.}A} = x_{k_1}$ and $x_{\text{intermed.}B} = x_{k_2}$ for some k_1 and $k_2 > k_1$. It is these two centroids that are used to determine the line used in the LA step of Algorithm 1. As noted above in section 3.1, $x_{\text{new}} = x_{\text{intermed.}A} + \delta w$ where the direction vector $w = x_{\text{intermed.}B} - x_{\text{intermed.}A}$.

We compute δ to correspond to the intersection point on the nearest hyperplane. First, the values of δ_i corresponding to the hyperplanes associated with each g_i are computed by substituting the equation of the line $L : x_{\text{new}} = x_{\text{intermed.}A} + \delta_i w$ in the equation for the hyperplane associated with g_i giving $g_i^T(x_{\text{intermed.}A} + \delta_i w) = 0$ and then solving for a specific scalar value of δ_i yielding $\delta_i = -(g_i^T x_{\text{intermed.}A})/(g_i^T w)$. (The case of a nonhomogeneous hyperplane $g_i^T u = c_i$ (for some $c_i \neq 0$) yields $\delta_i = (c_i - g_i^T x_{\text{intermed.}A})/(g_i^T w)$.) Let $\delta = \min\{|\delta_i|\}$ and $x_{\text{new}} = x_{\text{intermed.}A} + \delta w$ is the desired point of intersection with the nearest hyperplane.

6. Orthonormal Constraint Spaces. Before reporting the results of numerical tests, we first note an analytic result when the rows of G are *orthonormal*.

THEOREM 6.1. *Assume that the n hyperplanes associated with the rows of the constraint matrix G are orthonormal and that the dimension of f (i.e., the number of columns of G) is m (with $n \leq m$). Without loss of generality, we can assume that the rows g_i^T of G are the identity vectors $e_i = (0, \ldots, 1, \ldots, 0)^T$ where 1 is in the i^{th} location and $i \leq n \leq m$.*

Under these assumptions, the projection of $f = (\widetilde{\beta}_1, \widetilde{\beta}_2, \ldots, \widetilde{\beta}_n, \widetilde{\beta}_{n+1}, \ldots, \widetilde{\beta}_m)^T$ onto the subspace determined by G is the point $(0, 0, \ldots, 0, \widetilde{\beta}_{n+1}, \ldots, \widetilde{\beta}_m)^T$.

Moreover, this point is determined by a single iteration of the LA algorithm.

Proof. The projection of point $f = (\widetilde{\beta}_1, \widetilde{\beta}_2, \ldots, \widetilde{\beta}_m)^T$ onto the hyperplane corresponding to vector e_k produces $(\widetilde{\beta}_1, \widetilde{\beta}_2, \ldots, 0, \ldots, \widetilde{\beta}_m)^T$ where 0 appears in the k^{th} location. Projecting f onto each of the n hyperplanes will produce points that have a zero among the first n components and whose last $m - n$ components are all identical to those of f.

It thus follows that the projection of f onto the intersection of all the hyperplanes yields a point that has zeros in all of the first n components and whose last $m - n$ components are all identical to those of f.

To show that the LA algorithm yields this same point, if the n projection points are summed together and then divided by n, one obtains the first centroid,

$$x_1 = (\frac{n-1}{n}\widetilde{\beta}_1, \frac{n-1}{n}\widetilde{\beta}_2, \ldots, \frac{n-1}{n}\widetilde{\beta}_n, \widetilde{\beta}_{n+1}, \ldots, \widetilde{\beta}_m)^T.$$

Repeating the projection process on centroid x_1 yields the second centroid

$$x_2 = ((\frac{n-1}{n})^2\widetilde{\beta}_1, (\frac{n-1}{n})^2\widetilde{\beta}_2, \ldots, (\frac{n-1}{n})^2\widetilde{\beta}_n, \widetilde{\beta}_{n+1}, \ldots, \widetilde{\beta}_m)^T.$$

Subtracting x_1 from x_2 yields the direction vector

$$w = -(\frac{n-1}{n^2}\widetilde{\beta}_1, \frac{n-1}{n^2}\widetilde{\beta}_2, \ldots, \frac{n-1}{n^2}\widetilde{\beta}_n, 0, \ldots, 0)^T.$$

The general formula for δ in the equation for $x_{\text{new}} = x_1 + \delta w$ (the intersection point) is $\delta = -(g_i^T x_1)/(g_i^T w)$ (cf. section 5.3). For any hyperplane $g_i = e_i$, this simplifies to $\delta = (\frac{n-1}{n}\widetilde{\beta}_i)/(\frac{n-1}{n^2}\widetilde{\beta}_i) = n$.

Substituting the values for δ, w and x_1 into the formula for x_{new} yields a value of $(0, \ldots, 0, \widetilde{\beta}_{n+1}, \ldots, \widetilde{\beta}_m)^T$. ∎

ETNA
Kent State University
etna@mcs.kent.edu

One consequence is that, given an orthonormal constraint matrix G, the solution vector x can be found rather inexpensively and quickly using a single iteration of LA$_N$, a phenomenon also exhibited by standard row-action methods.

7. Numerical Tests.

7.1. Similar Algorithms. Tests were conducted comparing LA$_N$ with comparable algorithms by Cimmino, Pierra, and Dax. First, we briefly explain these other methods.

7.1.1. The Cimmino Algorithm. The Cimmino algorithm (cf. [18]) consists of repeated use of the following formula (cf. equation (5.3)) where x_0 is f, the upper block of the constant vector b. In the original Cimmino algorithm, the relaxation parameter α equals 2 (to obtain a point reflected about a given hyperplane), but other values may also be used.

$$x_{\text{new}} = \text{Cim}_\alpha(x_{\text{old}}) = x_{\text{old}} - \frac{\alpha}{n}\sum\left(\frac{g_i^T x_{\text{old}}}{\|g_i\|^2} g_i\right)$$

When used as part of another algorithm, it is often advantageous to repeat a Cimmino iteration multiple times. If the Cimmino iteration is repeated l times starting with x_{old} and using the intermediate centroid values to obtain new points, we will denote this as $x_{\text{new}} = \text{Cim}_\alpha^l(x_{\text{old}})$.

7.1.2. The Pierra E.P.P.M. Algorithm. The Pierra E.P.P.M. algorithm (cf. [33]) consists of these steps:

$$\begin{aligned} x_{\text{intermed.}} &= \text{Cim}_1^l(x_{\text{old}}) \\ w &= x_{\text{intermed.}} - x_{\text{old}} \\ x_{\text{new}} &= x_{\text{old}} + \frac{\lambda}{n}\frac{\sum \|P_i x_{\text{old}} - x_{\text{old}}\|^2}{\|w\|^2} w \end{aligned} \tag{7.1}$$

where λ is 1 except every k iterations. (In [33, p. 112], for one experiment, k was 3 and λ was 1/2.) Here the initial x_{old} is f, the upper block of the constant vector b.

7.1.3. The Dax Algorithm. The Dax algorithm (cf. [20]) consists of these steps:

$$x_{\text{intermed.}} = \text{Cim}_\alpha^l(x_{\text{old}}) \tag{7.2}$$

$$\begin{aligned} w &= x_{\text{intermed.}} - x_{\text{old}} \\ r &= Gx_{\text{old}} \\ z &= Gx_{\text{intermed.}} \\ \tau &= r - z \\ \theta &= \frac{\tau^T z}{\|\tau\|^2} \end{aligned}$$

$$x_{\text{new}} = x_{\text{intermed.}} + \theta w \tag{7.3}$$

where (7.2), the Cimmino iteration, may be repeated several times, $\alpha = 2$, and x_0 is f, the upper block of the constant vector b.

7.2. Comparison of Algorithms. Any distance procedure based on intersection with a hyperplane will obtain the exact answer in one step if the rows of G are orthonormal, as was shown above in section 6. If, however, the hyperplanes in G are nearly parallel, the computed intersection point of the line obtained through the centroids $x_{\text{intermed.}_A}$ and

$x_{\text{intermed.}_B}$ with one of the hyperplanes may, in fact, be quite distant from the desired solution point. It may also happen that the line through $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$ is parallel to one of the hyperplanes, thus making it impossible to compute an intersection. Given the approximate nature of the direction determined by the line through the centroids, the choice of a point that is the intersection with the *nearest* hyperplane is a "conservative" estimate, but one that does, in fact, guarantee eventual convergence.

All three methods compared make use of centroids and an acceleration line to obtain a new approximate solution point. The differences between the algorithms, however, are not insignificant (cf. sec. 3.1). The linear acceleration in LA is along a line determined by two centroids, whereas the line used by the Pierra and Dax algorithms uses a predetermined point and one newly-computed centroid. The LA_N algorithm uses the intersection with the nearest hyperplane as the starting point for the next iteration. In contrast, the Pierra algorithm averages the intersections with all the hyperplanes, some of which may be quite distant from the solution, to compute a distance and the Dax algorithm determines the point closest to the solution along the acceleration line as the basis for its distance. These differences give rise to the different convergence behaviors exhibited in our tests.

7.3. Modifications of the Basic Algorithm. In our tests, the basic LA_N algorithm was slightly modified to improve its accuracy. *First,* the rows of the constraint matrix G (cf. equation (2.1)) were *scaled by the norm of the row*. *Second,* as mentioned above, the computation of each centroid was, in fact, repeated several times before designating specific points as $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$, used to determine the line of projection. For comparison purposes, tests were conducted using different repetition values.

Repeating the computation of the centroids $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$ has two beneficial results: (1) the first centroid $x_{\text{intermed.}_A}$, derived from the initial vector f or a previous x_{new} (which is an intersection point on a hyperplane) becomes better centered between all the hyperplanes than it would be after merely one computation; and (2) the second centroid $x_{\text{intermed.}_B}$ is further separated from $x_{\text{intermed.}_A}$ than it would be if only a single computation were performed (resulting in a more accurate direction line toward the desired solution). The disadvantage is that each repeated computation of either centroid costs as much as the initial computation, but this computational cost may be offset improved by significantly faster convergence.

To indicate the number of repetitions of centroid computations, we will use a subscript with the LA method, e.g., LA_{N5} to indicate a 5-fold repetition of the centroid computations.

7.4. Experiments. Experiments were conducted with two different sets of matrices. The first set included constraint submatrices G with similar patterns but of different sizes. The second set used constraint submatrices derived from classical test matrices.

7.4.1. Description of Matrix Set I. Matrices 1–4 were matrices $\mathcal{A}$ in which the I submatrix was 75×75. In each case the f vector was the same, namely the vector $(1, 2, 3, \ldots, 75)^T$ and the initial approximation x_0 was f, the point being projected onto the constraint subspace $Gx = 0$. Matrix 5 was larger with a similar f vector. Tests were performed using Matlab 6.5.1 implementations of the various LA methods.

The structure of the G constraint submatrix of matrices 1–5 was based of this pattern: the first n rows and columns consisted of a square block with 2's on the diagonal and ones

ETNA
Kent State University
etna@mcs.kent.edu

elsewhere, such as

$$\begin{pmatrix} 2 & 1 & \dots & 1 \\ 1 & 2 & \dots & 1 \\ \vdots & \vdots & \ddots & 1 \\ 1 & 1 & \dots & 2 \end{pmatrix}$$

and the remaining elements of each row were all zeros or all ones, depending on the matrix. Table 7.1 summarizes the details and includes the condition numbers of each matrix (determined via the intrinsic Matlab `cond` command).

TABLE 7.1
Summary of sizes, structure, and condition numbers for Matrix Set I.

Matrix	Size of $\mathcal{A}$	Rows in G	Value in the remainder of G	Cond. Numb.
1	80×80	5	1	219.59
2	80×80	5	0	23.84
3	100×100	25	1	433.98
4	100×100	25	0	157.55
5	400×400	100	1	3190.80

7.4.2. Numerical Results for Matrix Set I. Table 7.2 summarizes the results from using the LA$_N$ algorithm with different repetition values on the matrices of Set I. The stopping criterion was when the norm of the error between the computed value and the exact value of x became less than 10^{-5}, i.e., when $\|x_{exact} - x_{\text{new}}\| < 10^{-5}$. (The exact value of x, x_{exact}, was determined by using Matlab to solve $\mathcal{A}\widehat{x} = b$ directly and then extracting the upper block of $\widehat{x}$ as x_{exact}.) It also includes results from the basic Cimmino, Pierra, and Dax algorithms applied to the five test matrices. For the LA$_N$ method, the number in parentheses indicates the total number of centroid computations performed. (This is the number of iterations times two [since two centroids are computed] times the number of repetitions to determine each centroid.) In the LA$_N$ and Dax algorithms, tests were run using different number of repetitions in the Cimmino step to compute the centroid and those values are indicated in the subscript. The relaxation value α for the Cimmino and Dax algorithms was 2. In the Pierra algorithm, every tenth iteration made use of a λ value of 0.90.

TABLE 7.2
Comparative results for different Cimmino-like algorithms. The error tolerance was 10^{-5}. The additional numbers in parentheses associated with the LA$_N$ and Dax algorithm indicate the total number of iterations used to compute the centroids.

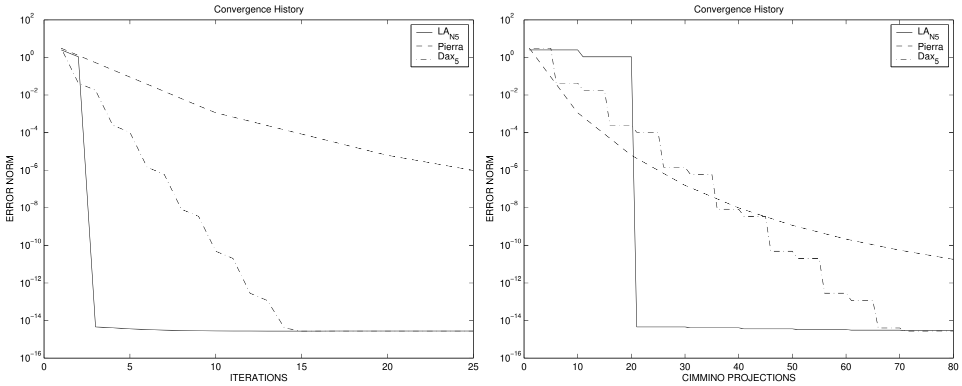
Matrix	LA$_{N2}$	LA$_{N5}$	LA$_{N10}$	Cimmino	Pierra	Dax$_5$	Dax$_{10}$
1	4 (16)	4 (40)	1 (20)	2464	4	3(15)	3(30)
2	15 (60)	3 (30)	2 (40)	247	20	6(30)	5(50)
3	2 (8)	2 (20)	1 (20)	14,713	8	4(20)	4(40)
4	18 (72)	4 (40)	2 (40)	5,277	34	5(25)	5(50)
5	6,391 (25,564)	2 (20)	1 (20)	260,241	9	4(20)	4(40)

The results indicate that the LA$_N$ algorithm performed significantly better than the Cimmino algorithm and that using a greater number of repetitions in determining the centroids

ETNA
Kent State University
etna@mcs.kent.edu

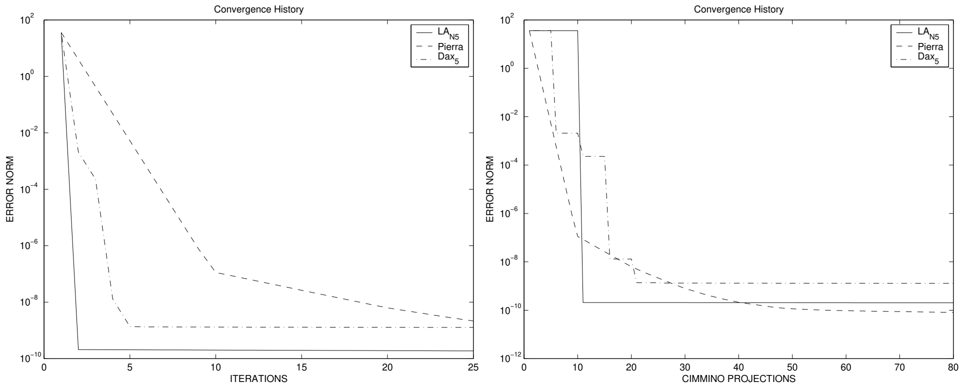
usually resulted in fewer iterations.

We include the following sample plots to illustrate the convergence of the LA$_N$ algorithm on matrix 2 (Figure 7.1) and matrix 3 (Figure 7.2) in comparison with the Pierra and Dax algorithms. The plots include a count of iterations and of the number of Cimmino-type projections used.



(Iteration Count) (Cimmino Projection Count)

FIG. 7.1. *Iteration counts and Cimmino projection counts for Matrix 2.*



(Iteration Count) (Cimmino Projection Count)

FIG. 7.2. *Iteration counts and Cimmino projection counts for Matrix 3.*

The plots suggest that, as a rule, the Pierra and Dax algorithms make small gains on each iteration, with Pierra making better progress than Dax on these test matrices. LA$_N$, however, demonstrates a slow convergence for a while and then exhibits a dramatic improvement. This behavior suggests that, even with the somewhat conservative distance algorithm of choosing the nearest hyperplane, once the two centroids determining the acceleration line align properly, convergence can be quite rapid. This motivates additional analysis of various modifications of the basic LA algorithm.

7.4.3. Description of Matrix Set II. This section reports on tests comparing the LA$_N$ algorithm with the Pierra and Dax algorithms for several test matrices with constraint spaces significantly different than those of Sections 7.4.1 and 7.4.2.

These additional matrices incorporated constraint spaces derived from test matrices chosen from the collections available from Matrix Market at the National Institute of Standards and Technology web site. In each case the test matrix $\mathcal{A}$ consists of an upper left hand block which was a square identity matrix matching the column dimension of the constraint matrix G derived from the chosen matrix.

Two test matrices derived from matrix FIDAP001, a real unsymmetric 216×216 matrix were created. For each matrix a specific number of the upper rows of the matrix was extracted and used as the constraint matrix G. For matrix 6 (FIDAP1_{108}), the G matrix consisted of the upper 108 rows (i.e., upper half) of FIDAP001. Matrix 7 (FIDAP1_{12}) was similar except that the G matrix was the upper 12 rows of FIDAP001. A KAHAN matrix of size 20×100 with $\theta = 1.2$ and perturbation = 25 was created and used as the G matrix for Matrix 8 ($\text{KAHAN}_{20\times 100}$). The upper 20 rows of matrix CAN_1054 (from the Cannes set of the Harwell-Boeing collection) were used as the G matrix for Matrix 9 (CAN1054_{20}).

Various other test matrices were also examined, but the behavior, as noted later, generally duplicated the results presented below.

Table 7.3 provides summary information about these four additional test matrices.

TABLE 7.3
Statistics for Matrices of Set II.

Matrix	Constraint Sp. G	Size of G / $\mathcal{A}$	Cond. Numb. $\mathcal{A}$
6	FIDAP001	$108 \times 216/324 \times 324$	3.839×10^{10}
7	FIDAP001	$12 \times 216/228 \times 228$	4.054×10^{6}
8	KAHAN	$20 \times 100/120 \times 120$	1.036×10^{2}
9	CAN_1054	$20 \times 1054/1074 \times 1074$	2.47×10^{1}

TABLE 7.4
Condition numbers for Original Matrices used for Constraint Spaces

Constraint Sp.	Cond. Numb. of Full Matrix
FIDAP001	3.298×10^{4}
CAN_1054	2.80×10^{34}

7.4.4. Numerical Results for Matrix Set II. In each of the following four plots in which the horizontal axis represents the number of iterations, the solid line represents the LA_{N5} algorithm, the dashed line the Pierra algorithm, and the dash-dotted line the Dax algorithm (with 5 repetitions of the Cimmino step to obtain the centroid). The vertical axis represents the norm of the error $\|x_{exact} - x_{\text{new}}\|$ where x_{exact} was determined by using Matlab to solve $\mathcal{A}\widehat{x} = b$ directly and then extracting the upper block of $\widehat{x}$ as x_{exact}. Similar to what was used in the previous experiments, f was the vector $(1, 2, 3, \ldots, m)^T$, where m is the number of columns of G.

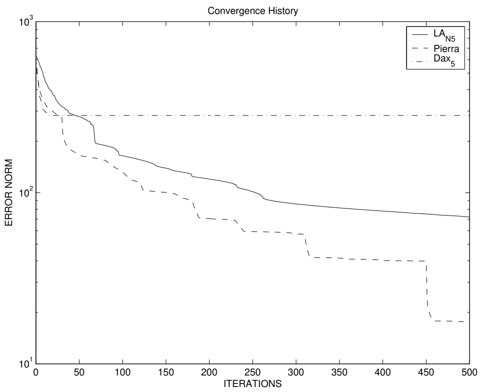
Each algorithm ran for the number of iterations indicated on the plot. (This number was chosen to show the convergence behavior without having to be identical for each matrix.)

In the Pierra algorithm, every tenth iteration made use of a λ value of 0.90.

The high condition numbers of matrix 7 and of the CAN_1054 matrix used for the constraint space of matrix 9 indicate that certain hyperplanes might be nearly parallel to others. Such a phenomenon does not affect the theory underlying the LA_N algorithm, however.

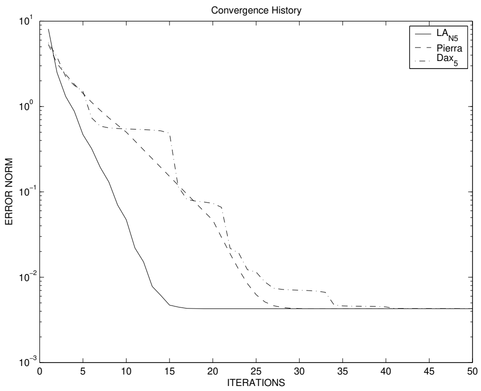
Figures 7.4, 7.5, and 7.6 show the show the superiority of the LA_N algorithm over the Pierra and Dax algorithm for these test matrices when counting iterations. We acknowledge,

ETNA
Kent State University
etna@mcs.kent.edu



(FIDAP1$_{108}$)

FIG. 7.3. *Matrix 6: The constraint space G is the upper half of the* FIDAP001 *matrix. Condition number of $\mathcal{A}$ was* 3.8×10^{10}.



(FIDAP1$_{12}$)

FIG. 7.4. *Matrix 7: The constraint space G is the upper 12 rows of the* FIDAP001 *matrix. Condition number of $\mathcal{A}$ was* 4.05×10^{6}.

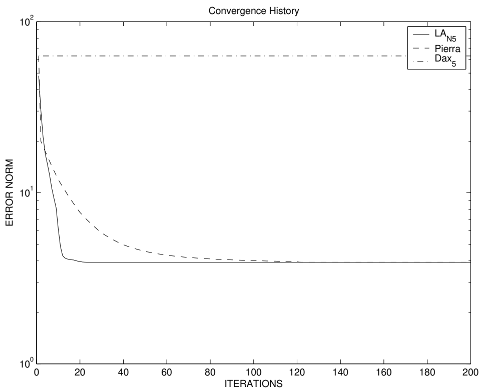
however, the different number of Cimmino-like projections involved per iteration for each algorithm: Pierra uses 1, Dax$_n$ uses n, and LA$_{Nn}$ uses $2n$. When counting Cimmino projections as in Figure 7.7, therefore, the plot for matrix 7 (with the FIDAP1$_{12}$ constraint space) indicates that Pierra converges more rapidly than the other two, but that there is essentially no difference between LA$_{N5}$ and Dax$_5$.

We note that Fig. 7.3 (Matrix 6 — FIDAP1$_{108}$) and 7.5 (Matrix 8 — KAHAN) demonstrate test matrices in which both the Pierra and LA$_N$ algorithms show a definite superiority over Dax, which seems to stagnate early in the process.

Moreover, one should realize that altering some of the parameters of the algorithms (e.g., the number of iterations in the Pierra algorithm before the λ correction value is applied or the number of Cimmino-like projections in the Dax and LA$_N$ algorithms) may also affect the convergence behavior.

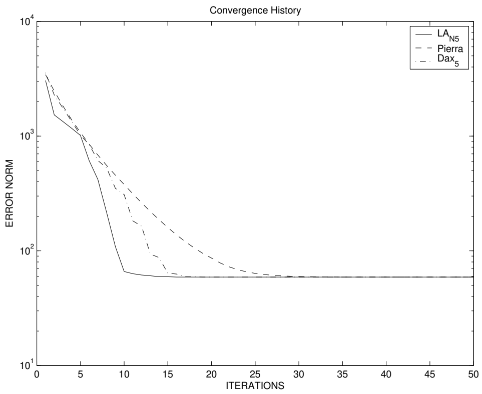
Other matrices were constructed and tested. For example, a test matrix derived from the PLAT362 (362×362) matrix (from the PLATZ set of the Harwell-Boeing collection) in which G consisted of the upper half of the PLAT362 exhibited behavior similar to that of

ETNA
Kent State University
etna@mcs.kent.edu



(KAHAN$_{20\times100}$)

FIG. 7.5. *Matrix 8: The constraint space G is the KAHAN*$_{20\times100}$ *matrix. Condition number of* $\mathcal{A}$ *was* 1.04×10^2.



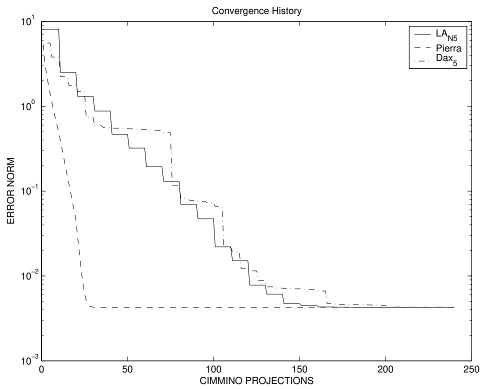
(CAN1054$_{20}$)

FIG. 7.6. *Matrix 9: The constraint space G is the upper 20 rows of the* CAN_1054 *matrix. Condition number of* $\mathcal{A}$ *was* 2.5×10^1.

Matrix 6. A matrix (with condition number of 1.7×10^{15}) derived from SAYLR1 and one (with condition number of 7.9×10^{12}) derived from BCSSTK04 (from the BCSSTRUC1 set of the Harwell-Boeing collection) were such that both test matrices caused all three methods to exhibit essentially identical non-converging stagnation behavior.

The plots of this section confirm that the basic insight used in the LA algorithm is good; namely, that determining an acceleration line via two centroids may be preferable to other approaches, and that, even when using a somewhat conservative measure in determining the distance accelerated along that line (such as intersecting with the *nearest* hyperplane), convergence is assured.

We note that the Pierra algorithm involves using a corrective step (i.e., the λ value) for the distance along the line every k iterations. These tests suggest further study as to whether using the *nearest* hyperplane to determine the acceleration distance along the projection line is too conservative a value especially in cases where the hyperplanes are nearly parallel and whether a hybrid scheme similar to what is used by Pierra would lead to better results. We note that results indicated for an alternative distance procedure given in the appendix suggest



(FIDAP1$_{12}$)

FIG. 7.7. *Matrix 7: Count of Cimmino projections. The constraint space G is the upper 12 rows of the FIDAP001 matrix. Condition number of $\mathcal{A}$ was* 4.05×10^6.

that certain hybrid schemes may be quite promising.

As a result of these tests, we believe that the use of an LA-like algorithm shows promise with a variety of matrices, when comparing with similar row action methods.

8. Applications to Computed Tomography. Although variants of the Kaczmarz and Cimmino methods have been studied in relation to saddle-point matrices and to the projection of a point in m-space onto the subspace determined by n hyperplanes (where $n \leq m$), the Kaczmarz approach has also been used to obtain an estimate when given an overdetermined system, as typically happens in problems related to computed tomography (cf. [28, pp. 277ff]). In fact, variants of the classical Kaczmarz algorithm were used in processing data in early CAT scanners, although other algorithms are now regularly used instead. Nevertheless, the Kaczmarz and Cimmino algorithms still enjoy favor among some researchers for specialized image reconstruction projects (cf. [25, 26, 31]).

In a typical computed tomography problem, the rows of $\mathcal{A}$ in the system $\mathcal{A}\widehat{x} = b$ correspond to the different scans (or rays) through an object and the columns correspond to the discretized blocks (or pixels) into which the object has been subdivided. The system matrix $\mathcal{A}$ is not a saddle point matrix and the number of rows it contains is usually significantly greater than the number of columns, thereby providing a redundancy of information and enabling the reconstruction of the image of the scanned object with greater accuracy. The right hand side, b, consists of the detected values associated with each scan. The origin of such a system means that the equations are, in general, not linearly independent (cf. [28, p. 281]), and that the equations are consistent, except for noise interfering with obtaining accurate values for the constant vector.

One could make use of a Cimmino-like projection algorithm, such as the LA algorithm of this paper, to solve a tomography problem. In such an application, every row of the system matrix $\mathcal{A}$ would be used to determine the centroid. Unlike the set of constraints associated with equation (2.1), $Gx = 0$, in which the right hand side is a zero vector, in this case, the right hand side, b, is non-zero. For that reason, the projector equation (corresponding to equation (5.1)) is

$$P_i x_k = x_k - \frac{a_i^T x_k - b_i}{\|a_i\|^2} a_i$$

where b_i is the i^{th} component of the right hand side vector b and a_i is the i^{th} row of $\mathcal{A}$.

ETNA
Kent State University
etna@mcs.kent.edu

9. Other Approaches. Other approaches to projection algorithms used in constrained systems often require the projector explicitly. For a detailed explanation see work by Bramley ([8], [9]). Such algorithms include the Uzawa algorithm studied by Elman and Golub ([21]).

In Bramley's approach (cf. [8, pp. 4, 6–7]), the projector $P = I - G^T(GG^T)^{-1}G = I - G^{\dagger}G$ and the pseudo-inverse $G^{\dagger}$ are computed explicitly and used repeated within the iteration loop. This means that the algorithm must explicitly compute the inverse of GG^T, a square matrix whose dimensions equal the number of rows of the constraint space G. In a large system with a significantly sized constraint space, it can be prohibitive to attempt such an explicit computation of $(GG^T)^{-1}$.

Bramley's implementation of the Uzawa algorithm (cf. [8, pp. 9–10]) does not explicitly compute $(GG^T)^{-1}$. Instead, it requires that the system $RR^T g_k = r_k$ be solved where RR^T is the full factorization of $GD^{-1}G^T$ where D is the diagonal of the upper left matrix of $\mathcal{A}$. For our examples, this upper left matrix was, in fact, I, and hence the system was equivalent to $GG^T g_k = r_k$. Such a step also occurs in the algorithm described in this paper.

One characterization of an Uzawa-type algorithm is as an iterative method that improves λ, the lower section of the solution vector, and then uses λ to determine x, the upper section of the solution vector, exactly. The updated x is easily computed from $x = f - G^T\lambda$ derived from the system equations. The updated λ is some variation of $\lambda_{old} + \alpha(Gx)$, where α may be chosen according to various schemes. As x improves, Gx approaches 0.

10. Evaluation and Future Work. Any row projection algorithm used to find the projected point in a constrained system as in (2.1) has the advantage of avoiding the computation of the orthogonal projector of the complete constraint matrix G, a very time-consuming step for large matrices. Yet, the computation and storage of the individual projector matrices P_i (cf. section 5.1) for each of the hyperplanes can also be prohibitive for large systems. The LA algorithm can avoid this difficulty if coded to compute the set of the norms of the row vectors, $g_i^T g_i$ once for reuse. This set is relatively small and should not cause any space difficulty.

It also attempts to take into account that something is known about the solution, namely, that it is the projection of f, the upper part of the constant vector b, onto the intersection subspace determined by the constraint space $Gx = c$. With certain constrained systems such as those used for tests in section 7, the LA step, that is, moving along a computed line that approximates a line intersecting the intersection subspace, can yield an approximate solution of desired accuracy in one iteration.

This paper has described the LA algorithm and demonstrated its success and even its dramatic convergence in specific cases when compared to similar line acceleration approaches. Future studies are planned with other types of matrices, both constrained systems and those arising from problems related to tomography as well as the effects various values for the number of times each centroid is computed may have on convergence rates. An additional topic for study is how using different number of repetitions of the Cimmino-projection in computing the two centroids affects the convergence rates.

Appendix: Alternative Distance Procedures.

Procedure A: Intersecting with a Specified Hyperplane. Procedure A involves computing the intersection of the projection line through $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$ with a *designated hyperplane*. This intersection point is considered to be the desired approximate solution x_{new}. For example, in Figure 4.1, one may assume that x_{new} is the intersection of the line through $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$ with the hyperplane associated with g_1. As depicted, x_{new} is closer to $x \in \mathcal{G}$ than are any of x_{old}, $x_{\text{intermed.}_A}$, or $x_{\text{intermed.}_B}$.

The simplest option for computing the point of intersection is to specify a hyperplane and compute the intersection point based on this hyperplane. In any subsequent iteration of

ETNA
Kent State University
etna@mcs.kent.edu

the algorithm, one could repeat the choice of the first hyperplane to obtain a new point of intersection. Such a choice does not, however, factor in any information latent in the other hyperplanes. An alternative approach would be to cycle through all the hyperplanes, changing the hyperplane of intersection with each iteration of this algorithm.

Table 10.1 summarizes the results from a *single* iteration of the LA algorithm with distance procedure A and using 5 iterations to determine the centroids, which we label LA$_{A5}$. In these experiments, we chose the first hyperplane, i.e., the hyperplane associated with the first row, g_1, of G. As can be seen, the accuracy achieved after one iteration was such that further repetitions of the algorithm were unneeded.

TABLE 10.1
Results after one iteration of the LA$_{A5}$ algorithm.

Matrix	Size	Cond. Numb.	$\|\|u_{exact} - x_N\|\|$
1	80×80	219.59	3.26×10^{-10}
2	80×80	23.84	1.02×10^{-5}
3	100×100	433.98	9.22×10^{-9}
4	100×100	157.55	1.31×10^{-6}
5	400×400	3190.80	3.78×10^{-8}

For Matrices 1–5, LA$_{A5}$ performed better than (or, with Matrix 5, the same as) LA$_{N5}$. With other sample matrices tested, however, the LA$_A$ algorithm sometimes cycled because of the selection of the same hyperplane for repeated iterations and a single computation of each centroid.

Procedure D: Distance. An alternative procedure for obtaining x_{new} is to determine the direction of the line through $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$ and then compute an appropriate *distance* to travel along this line in the direction of the desired solution x not dependent on intersecting with any hyperplane. This approach is similar to those used in the Pierra and Dax algorithms.

Procedure D makes use of approximations of similar right triangles to determine an appropriate distance and is based on two assumptions: (1) that many of the hyperplanes associated with row in G meet at an acute angle, and (2) that the line through $x_{\text{intermed.}_A}$ and $x_{\text{intermed.}_B}$ is nearly orthogonal to a line through $x_{\text{intermed.}_B}$ joining the projection of $x_{\text{intermed.}_A}$ on one hyperplane with the projection of $x_{\text{intermed.}_A}$ on another (as depicted in Figure 4.1).

To simplify the notation in the rest of this section, assume $x_1 = x_{\text{intermed.}_A}$ and $x_2 = x_{\text{intermed.}_B}$.

Let p_1 be the projection of centroid x_1 onto hyperplane g_1. It then follows that angle $\angle x_1 p_1 x_{\text{new}}$ is a right angle. Thus, the two angles $\angle x_1 p_1 x_2$ and $\angle x_2 p_1 x_{\text{new}}$ are complementary. Hence, since angle $\angle p_1 x_2 x_{\text{new}}$ is also assumed to be a right angle, angle $\angle p_1 x_{\text{new}} x_2$ must equal angle $\angle x_1 p_1 x_2$ and angle $\angle x_2 x_1 p_1$ must equal angle $\angle x_2 p_1 x_{\text{new}}$. One can therefore conclude that triangle $\triangle p_1 x_1 x_2$ is similar to triangle $\triangle p_1 x_2 x_{\text{new}}$. Thus, corresponding sides are proportional and $\dfrac{\|x_{\text{new}} - x_2\|}{\|p_1 - x_2\|} = \dfrac{\|p_1 - x_2\|}{\|x_2 - x_1\|}$. Therefore, the appropriate distance to travel from x_2 toward x_{new} (i.e., the distance $\|x_{\text{new}} - x_2\|$) in the direction along the line determined by x_1 and x_2 is $\dfrac{\|p_1 - x_2\|^2}{\|x_2 - x_1\|}$.

Procedure D thus yields $x_{\text{new}} = x_2 + \delta w$ where $w = x_2 - x_1$, x_2 is the second centroid and $\delta = \dfrac{\|p_1 - x_2\|^2}{\|x_2 - x_1\|}$.

In actual problems, centroid x_2 may be significantly closer to certain hyperplanes than to others, and some plausible approach is needed to obtain an appropriate value in place of $\|p_1 - x_2\|$ that also takes into account the contribution of all the hyperplanes to the system. One approach is to compute the average of the distances from x_2 to each of the hyperplanes, i.e., to compute the value $z_{ave} = (\sum_{i=1}^{n} \|p_i - x_2\|)/n$. This particular order of computations, however, has the disadvantage of requiring a significant amount of work, in computing $\|p_i - x_2\|$ for each hyperplane g_i and then computing the average, z_{ave} after first determining x_2.

The values of $\|p_i - x_2\|$, however, could be approximated by using $\|p_i - x_2\|^2 = \|p_i - x_1\|^2 - \|x_2 - x_1\|^2$ assuming that each triangle $\triangle p_i x_1 x_2$ is a right triangle (cf. Figure 4.1). But $\|p_i - x_1\|$ is merely $g_i^T x_1/(\|g_i\|^2)$ in projection formula (5.1) already being computed in the process of determining x_2.

As a result, the values for $\|p_i - x_1\|$ are merely by-products of computing the projections of x_1 and can be done at the same time, thus leading to an integrated LA algorithm in which the distance needed in step 5 of Algorithm 1 is actually computed in the process of steps 3 and 4. (We omit the details.)

Acknowledgements. Special thanks are given to Paul E. Saylor (Department of Computer Science, University of Illinois at Urbana-Champaign) and Richard A. Scott (Santa Clara University) for insightful comments and helpful suggestions.

REFERENCES

[1] R. ANSORGE, *Connections between the Cimmino-method and the Kaczmarz-method for the solution of singular and regular systems of equations*, Computing, 33 (1984), pp. 367–375.

[2] M. ARIOLI, I. DUFF, J. NOAILLES, AND D. RUIZ, *A block projection method for sparse matrices*, SIAM J. Sci. Statis. Comput., 13 (1992), pp. 47–70.

[3] S. F. ASHBY, T. A. MANTEUFFEL, AND P. E. SAYLOR, *A taxonomy for conjugate gradient methods*, SIAM J. Numer. Anal., 27 (1990), pp. 1542–1568.

[4] H. H. BAUSCHKE AND J. M. BORWEIN, *On projection algorithms for solving convex feasibility problems*, SIAM Rev., 38 (1996), pp. 367–426.

[5] M. BENZI, G. H. GOLUB, AND J. LIESEN, *Numerical solution of saddle point problems*, Acta Numer., (2005), pp. 1–137.

[6] A. BJÖRCK AND T. ELFVING, *Accelerated projection methods for comuting pseudoinverse solutions of systems of linear equations*, BIT, 19 (1979), pp. 145–163.

[7] E. BODEWIG, *Matrix Calculus*, North-Holland Publ. Co., Amsterdam, 1959.

[8] R. BRAMLEY, *An orthogonal projection algorithm for generalized Stokes problems*, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, January 1992, Report No. 1190.

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

[10] Y. CENSOR, *Row-action methods for huge and sparse systems and their applications*, SIAM Rev., 23 (1981), pp. 444–466.

[11] ———, *Finite series-expansion reconstruction methods*, Proceedings of the IEEE, 71 (1983), pp. 409–419.

[12] ———, *Parallel application of block-iterative methods in medical imaging and radiation therapy*, Math. Programming, 42 (1988), pp. 307–25.

[13] Y. CENSOR, M. D. ALTSCHULER, AND W. D. POWLIS, *On the use of Cimmino's simultaneous projections method for computing a solution of the inverse problem in radiation therapy treatment planning*, Inverse Problems, 4 (1988), pp. 607–23.

[14] Y. CENSOR, P. P. B. EGGERMONT, AND D. GORDON, *Strong underrelaxation in Kaczmarz's method for inconsistent systems*, Numer. Math., 41 (1983), pp. 83–92.

[15] Y. CENSOR, D. GORDON, AND R. GORDON, *Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems*, Paral. Comput., 27 (2001), pp. 777–808.

[16] Y. CENSOR AND G. T. HERMAN, *On some optimization techniques in image reconstruction from projections*, Appl. Numer. Math., 3 (1987), pp. 365–91.

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

[18] G. CIMMINO, *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari*, Ric. Sci. progr. tecn. econom. naz., XVI (1938), pp. 326–333.
[19] A. DAX, *The convergence of linear stationary iterative methods for solving singular unstructured systems of linear equations,* SIAM Rev., 32 (1990), pp. 611–635.
[20] A. DAX, *Line search accleration of iterative methods,* Lin. Alg. Appl., 130 (1990), pp. 43–63.
[21] H. C. ELMAN AND G. H. GOLUB, *Inexact and preconditioned Uzawa algorithms for saddle point problems*, SIAM J. Sci. Stat. Comput., 31 (1994), pp. 1645–1661.
[22] G. H. GOLUB, C. F. VANLOAN, *Matrix Computations* (Third ed.), Baltimore: The Johns Hopkins University Press, 1996.
[23] R. GORDON, R. BENDER, AND G. T. HERMAN, *Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography*, J. Theoret. Biol., 29 (1970), pp. 471–481.
[24] A. S. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis,* New York: Dover Publications, 1975.
[25] M. JIANG AND G. WANG, *Development of iterative algorithms for image reconstruction*, J. X-ray Sci. Tech., 10 (2001), pp. 77–86.
[26] M. JIANG AND G. WANG, *Convergence studies on iterative algorithms for image reconstruction*,IEEE Trans. Med. Imaging, 22 (2003), pp. 569–579.
[27] S. KACZMARZ, *Angenäherte Auflösung von Systemen linearer Gleichungen*, Bull. Intern. Acad. Polon. Sci. Lett., Ser. A, 35 (1937), pp. 355–357.
[28] A. C. KAK AND M. SLANEY, *Principles of Computerized Tomographic Imaging*, IEEE Press, New York, 1988.
[29] C. KAMATH AND S. WEERATUNGA, *Projection methods for the numerical solution of non-self-adjoint elliptic partial differential eauations*, Num. Methods for P.D.E., 8 (1992), pp. 59–76.
[30] S. L. LEE, *Krylov methods for the numerical solution of initial-value problems in differential-algebraic equations,* Department of Computer Science, University of Illinois at Urbana-Champaign, December 1993, Report No. UIUCDCS-R-93-1814.
[31] K. MULLER, R. YAGEL AND J. J. WHELLER, *Anti-aliased 3D cone-beam reconstruction of low-contrast objects with algebraic methods*, IEEE Trans. Med. Imaging, 18 (1999), pp. 519–537.
[32] F. NATTERER, *The Mathematics of Computerized Tomography,* New York: John Wiley and Sons, 1986. Reprinted: Philadelphia: SIAM, 2001.
[33] G. PIERRA, *Decomposition through formalization in a product space*, Math. Prog., 28 (1984), pp. 96–118.
[34] C. POPA, *Preconditioned Kaczmarz-extended algorithm with relaxation parameters*, Korean J. Comp. Appl. Math., 6 (1999), pp. 523–535.
[35] Y. SAAD AND H. A. VAN DER VORST, *Iterative solution of linear systems in the 20th century*, J. Comp. Appl. Math., 123 (2000), pp. 1–33.
[36] K. TANABE, *Projection method for solving a singular system of linear equations and its applications*, Numer. Math., 17 (1971), pp. 203–214.
[37] L. N. TREFETHEN AND D. BAU, III, *Numerical Linear Algebra,* Philadelphia: SIAM, 1997.
[38] M. R. TRUMMER, *Reconstructing pictures from projections*: *On the convergence of the ART algorithm with relaxation*, Computing, 26 (1981), pp. 189–195.
[39] M. R. TRUMMER, *A note on the ART of relaxation*, Computing, 33 (1984), pp. 349–352.