
ELSEVIER

Parallel Computing 27 (2001) 777–808

PARALLEL COMPUTING

www.elsevier.com/locate/parco

Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems

Yair Censor [a], Dan Gordon [b,*], Rachel Gordon [c]

[a] *Department of Mathematics, University of Haifa, Mt. Carmel, Haifa 31905, Israel*
[b] *Department of Computer Science, University of Haifa, Mt. Carmel, Haifa 31905, Israel*
[c] *Department of Aerospace Engineering, The Technion – Israel Institute of Technology, Haifa 32000, Israel*

Received 10 December 1998; received in revised form 8 June 1999; accepted 12 May 2000

Abstract

Component averaging (CAV) is introduced as a new iterative parallel technique suitable for large and sparse unstructured systems of linear equations. It simultaneously projects the current iterate onto all the system's hyperplanes, and is thus inherently parallel. However, instead of orthogonal projections and scalar weights (as used, for example, in Cimmino's method), it uses oblique projections and diagonal weighting matrices, with weights related to the sparsity of the system matrix. These features provide for a practical convergence rate which approaches that of algebraic reconstruction technique (ART) (Kaczmarz's row-action algorithm) – even on a single processor. Furthermore, the new algorithm also converges in the inconsistent case. A proof of convergence is provided for unit relaxation, and the fast convergence is demonstrated on image reconstruction problems of the Herman head phantom obtained within the SNARK93 image reconstruction software package. Both reconstructed images and convergence plots are presented. The practical consequences of the new technique are far reaching for real-world problems in which iterative algorithms are used for solving large, sparse, unstructured and often inconsistent systems of linear equations.

Keywords: Component averaging; Iterative techniques; Oblique projections; Sparse systems; Sparsity-related weights

* Corresponding author.
E-mail addresses: yair@math.haifa.ac.il (Y. Censor), gordon@cs.haifa.ac.il (D. Gordon), rgordon@tx.technion.ac.il (R. Gordon).

0167-8191/01/$ - see front matter
PII: S0167-8191(00)00100-9

1. Introduction and motivation

Large and sparse systems of linear equations arise in many important practical applications. Solving linear equations is a special case of the *convex feasibility problem* of finding a point $x^* \in C \triangleq \bigcap_{i=1}^{m} C_i \neq \emptyset$, where C is the nonempty intersection of finitely many closed convex sets $C_i \subseteq \mathbb{R}^n$, $i = 1, 2, \ldots, m$, in the Euclidean space. Systems of linear equations, linear inequalities, or convex inequalities are important special cases of the convex feasibility problem which repeatedly appear in various significant real-world applications.

We are interested here in iterative algorithms for this problem, which can be generally classified as being either *sequential* or *simultaneous* or *block-iterative* – see, e.g., [14, Section 1.3], and the review paper of Bauschke and Borwein [3] for a variety of specific algorithms of these kinds.

The prototype of simultaneous algorithms is the well-known Cimmino method. In this method the current iterate x^k is first projected on all sets to obtain "intermediate points"

$$x^{k+1,i} = P_i(x^k), \quad i = 1, 2, \ldots, m, \tag{1.1}$$

where P_i is the orthogonal (least Euclidean distance) projection onto C_i, and then the next iterate is

$$x^{k+1} = x^k + \lambda_k \left(\sum_{i=1}^{m} w_i x^{k+1,i} - x^k \right), \tag{1.2}$$

where w_i are fixed weights such that $w_i > 0$ for all i, and $\sum_{i=1}^{m} w_i = 1$, and $\{\lambda_k\}_{k \geqslant 0}$ are relaxation parameters, commonly confined to the interval $\epsilon \leqslant \lambda_k \leqslant 2 - \epsilon$, for some fixed but arbitrarily small $\epsilon > 0$. The origin of this method is in [15]; see also [20, Chapter 5, Section 22]. We can replace the system of fixed weights $\{w_i\}_{i=1}^{m}$ in (1.2) by $\{w_i^k\}_{i=1}^{m}$ with $w_i^k > 0$, and $\sum_{i=1}^{m} w_i^k = 1$, for all $k \geqslant 0$, and still have a convergent scheme in the consistent case, when the convex feasibility problem has a nonempty intersection set C– see [1, Theorem 1].

Besides its inherent parallelism, one advantage of the fully simultaneous algorithm over sequential methods such as the method of successive projections (also known as POCS, for "projections onto convex sets"), see, e.g., [14, Algorithm 5.2.1], or even over non-sequential block-iterative schemes which are not fully simultaneous, is its behavior in the inconsistent case. It is guaranteed to converge to a weighted least-squares solution which minimizes the weighted sum of the squares of the distances to the sets C_i, $i = 1, 2, \ldots, m$. This has been shown by Iusem and De Pierro [23], via a local convergence result, and globally by Combettes [16]. However, the slow rate of convergence of the method hampers its use, particularly for large and sparse unstructured systems. Throughout this paper, we use the term "rate of convergence" in an informal manner to refer to the practical behavior of the algorithm during a finite number of iterations starting from the first iterate x^0 and onwards. This usage is different from the standard mathematical definition of the rate of convergence, which describes the asymptotic behavior of the algorithm.

A widely used row-action sequential algorithm for solving systems of linear equations is algebraic reconstruction technique (ART) – see, e.g., [21], [14, Algorithm 5.4.3]. This algorithm, originally due to Kaczmarz [24], cyclically projects the current iterate onto the hyperplanes. In the inconsistent case, it is known to converge cyclically – see [34]. It has also been shown by Censor et al. [13] that for a fixed positive relaxation parameter, the limits of the cycles lie within a bounded distance from the geometric least-squares solution. This distance depends on the relaxation parameter, and it approaches zero as the relaxation parameter tends to zero. Thus, ART with small relaxation parameters can also be used to approach the least-squares solution arbitrarily close.

In this paper, we introduce a new algorithm, which we call "component averaging" (CAV). This technique is derived from the recently studied method of Byrne and Censor [10], which employs diagonal weighting matrices instead of the fixed weights $\{w_i\}$ in (1.2) or the iteration index dependent weights $\{w_i^k\}$. The weighting is also directly related to the sparsity of the system matrix. The new algorithm retains the much desired feature of convergence of Cimmino's algorithm in the inconsistent case, but accelerates the numerical convergence in practical computation very significantly. By diagonal sparsity-oriented weighting, the proposed acceleration is introduced into the mathematical structure of the algorithm and can be realized even on a single processor. It does not destroy the inherently parallel nature of the algorithm, thus allowing farther clock-time savings with parallel implementations.

It should be noted though that for sparse problems, the sequential row-action ART method can also be parallelized by simultaneously projecting the current iterate onto a set of mutually orthogonal hyperplanes (obtained by considering equations whose sets of nonzero components are pairwise disjoint). For the case of image reconstruction from projections, such sets of equations can be obtained by considering parallel rays that are sufficiently far apart so as to pass through disjoint sets of pixels, see Section 5.1. However, for image reconstruction, such a procedure produces very small granularity, and the amount of communications required between processors could make such a parallel implementation unattractive.

To motivate our work, let us consider the case of linear equations in which the sets C_i are hyperplanes

$$H_i \triangleq \{x \in \mathbb{R}^n \mid \langle a^i, x\rangle = b_i\} \tag{1.3}$$

for $i = 1, 2, \ldots, m$, where $\langle \cdot, \cdot \rangle$ is the inner product and $a^i = (a_j^i)_{j=1}^n \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ are given vectors and given real numbers, respectively. Then, for any $z \in \mathbb{R}^n$, the orthogonal projection of z onto H_i is

$$P_i(z) = z + \frac{b_i - \langle a^i, z\rangle}{\|a^i\|_2^2} a^i, \tag{1.4}$$

where $\|\cdot\|_2$ is the Euclidean norm.

In Cimmino's algorithm for the convex feasibility problem, with relaxation parameters and with equal weights $w_i = 1/m$, the next iterate x^{k+1} is the average of the projections of x^k on the closed and convex sets C_i, as follows:

Algorithm 1.1 (*Cimmino*).

Initialization: $x^0 \in \mathbb{R}^n$ is arbitrary.

Iterative step: Given x^k, compute

$$x^{k+1} = x^k + \frac{\lambda_k}{m} \sum_{i=1}^{m} \left(P_i(x^k) - x^k \right), \tag{1.5}$$

where $\{\lambda_k\}_{k \geqslant 0}$ are relaxation parameters.

Expanding the iterative step (1.5) according to (1.4) produces, for every component $j = 1, 2, \ldots, n$

$$x_j^{k+1} = x_j^k + \frac{\lambda_k}{m} \sum_{i=1}^{m} \frac{b_i - \langle a^i, x^k \rangle}{\|a^i\|_2^2} a_j^i, \tag{1.6}$$

which can be written in matrix notation as

$$x^{k+1} = x^k + \lambda_k A^{\mathrm{T}} D (b - A x^k), \tag{1.7}$$

where $b = (b_i) \in \mathbb{R}^m$, A^{T} (the transpose of A) has a^i in its ith column, and

$$D = \frac{1}{m} \operatorname{diag}\left(\frac{1}{\|a^1\|_2^2}, \frac{1}{\|a^2\|_2^2}, \ldots, \frac{1}{\|a^m\|_2^2} \right). \tag{1.8}$$

Our new CAV algorithm will be similar in form to (1.7), but with a totally different diagonal matrix D.

Consider now system (1.3). When it is sparse, only a relatively "small" number of the elements $a_j^1, a_j^2, \ldots, a_j^m$ are nonzero, but in (1.6) the sum of their contributions is divided by the relatively "large" m.

This observation led us to consider a modification of the algorithm in which the factor $1/m$ in (1.6) is replaced by a factor that depends only on the *nonzero* elements in the set $\{a_j^1, a_j^2, \ldots, a_j^m\}$. For each $j = 1, 2, \ldots, n$, we denote by s_j the number of nonzero elements of column j, and we wish to replace (1.6) by

$$x_j^{k+1} = x_j^k + \frac{\lambda_k}{s_j} \sum_{i=1}^{m} \frac{b_i - \langle a^i, x^k \rangle}{\|a^i\|_2^2} a_j^i. \tag{1.9}$$

Certainly, if the $m \times n$ system matrix $A = (a_j^i)$ is sparse then the s_j values will be much smaller than m. But this poses a theoretical difficulty. The iterative step (1.6) is a special case of

$$x^{k+1} = x^k + \lambda_k \sum_{i=1}^{m} w_i \frac{b_i - \langle a^i, x^k \rangle}{\|a^i\|_2^2} a^i, \tag{1.10}$$

where the fixed weights $\{w_i\}_{i=1}^m$ must be positive for all i and $\sum_{i=1}^m w_i = 1$. The attempt to use $1/s_j$ as weights in (1.9) does not fit into scheme (1.10), unless one can prove convergence of the iterates of a fully simultaneous iterative scheme with component-dependent (i.e., j-dependent) weights of the form

$$x_j^{k+1} = x_j^k + \lambda_k \sum_{i=1}^{m} w_{ij} \frac{b_i - \langle a^i, x^k \rangle}{\|a^i\|_2^2} a_j^i \tag{1.11}$$

for all $j = 1, 2, \ldots, n$.

As will be seen in the sequel, it turns out that to derive the proof of convergence for (1.11), we need to modify it further by replacing the orthogonal projections onto the hyperplanes H_i by certain *oblique projections* induced by appropriately defined weight matrices.

Simultaneous or block-iterative methods were extensively studied in recent years, see, e.g., [18,19], and the recent paper of Kotzer et al. [26], with interesting extensions to nonlinear equations such as, e.g., [17,29]. To the best of our knowledge, none of the earlier algorithms for the convex feasibility problem or for a system of linear equations, except for the recent method of Byrne and Censor [10], allow component-dependent weights.

The paper is organized as follows. In Section 2 we discuss oblique projections and define *generalized oblique projections*. The CAV algorithm is developed in Section 3 and in Section 4 we prove its convergence regardless of the consistency or inconsistency of the system of equations. Currently though, our proof is limited to the special case when all relaxation parameters are equal to unity, i.e., no relaxation is permissible. In Section 5 we compare the new algorithm against Cimmino and ART. We use the problem of image reconstruction from projections, in its fully discretized model, as a test bed for the comparisons of the three methods. We also study the parallel behavior of component averaging on varying number of processors on the IBM SP2 parallel machine.

2. Generalized oblique projections onto hyperplanes

Consider a hyperplane $H \triangleq \{x \in \mathbb{R}^n \,|\, \langle a, x \rangle = b\}$, with $a = (a_j) \in \mathbb{R}^n$, $b \in \mathbb{R}$ and $a \neq 0$. Let G be an $n \times n$ symmetric positive definite matrix and let $\|x\|_G^2 \triangleq \langle x, Gx \rangle$ be the associated ellipsoidal norm, see, e.g., [6, Proposition A.28]. Given a point $z \in \mathbb{R}^n$, the oblique projection of z onto H with respect to G is the unique point $P_H^G(z) \in H$ for which

$$P_H^G(z) = \arg\min\{\|x - z\|_G \,|\, x \in H\}. \tag{2.1}$$

Solving this minimization problem leads to

$$P_H^G(z) = z + \frac{b - \langle a, z \rangle}{\|a\|_{G^{-1}}^2} G^{-1} a, \tag{2.2}$$

where G^{-1} is the inverse of G. For $G = I$, the unit matrix, Eq. (2.2) yields the orthogonal projection of z onto H, as given by (1.4); see, e.g., [5, Section 2.6].

Oblique projections have been used in the past in several contexts. Kayalar and Weinert [25] promote oblique projections for local processing in sensor arrays and credit Murray [30] and Lorch [28] for pioneering work on oblique projections. Behrens and Scharf [4] use oblique projections for signal processing applications.

Oblique projections in Cimmino's algorithm have been proposed and used by Arioli et al. [2]. They showed that using a Cimmino algorithm in which all projections are oblique with respect to a given symmetric positive definite matrix G, for a system $Ax = b$, is equivalent to applying a Cimmino algorithm with all orthogonal projections to the post-conditioned system $AG^{-1/2}\tilde{x} = b$, where $\tilde{x} = G^{1/2}x$ and $G^{1/2}$, the symmetric square root of G, is the unique symmetric positive definite matrix obtained from $G = G^{1/2}G^{1/2}$.

We wish to consider oblique projections onto H with respect to a diagonal matrix $G = \mathrm{diag}(g_1, g_2, \ldots, g_n)$ for which some diagonal elements might be zero. Since this does not fit into formula (2.2), we introduce the following definition. We need, however, to allow $g_j = 0$ on the diagonal of G only for such j for which $a_j = 0$ in the normal vector a of the hyperplane H onto which we intend to project with respect to G.

Definition 2.1. Let $G = \mathrm{diag}(g_1, g_2, \ldots, g_n)$ with $g_j \geqslant 0$ for all $j = 1, 2, \ldots, n$, let $H = \{x \in \mathbb{R}^n | \langle a, x\rangle = b\}$ be a hyperplane with $a = (a_j) \in \mathbb{R}^n$ and $b \in \mathbb{R}$, and assume that $g_j = 0$ if and only if $a_j = 0$. The *generalized oblique projection* of a point $z \in \mathbb{R}^n$ onto H with respect to G is defined for all $j = 1, 2, \ldots, n$, by

$$\left(P_H^G(z)\right)_j \triangleq \begin{cases} z_j + \frac{b - \langle a, z\rangle}{\sum_{\substack{l=1 \\ g_l \neq 0}}^{n} a_l^2/g_l} \cdot \frac{a_j}{g_j} & \text{if } g_j \neq 0, \\ z_j & \text{if } g_j = 0. \end{cases} \tag{2.3}$$

In the following sections, $P_H^G(z)$ is always meant as in (2.3), which reduces to (2.2) if $g_j \neq 0$ for all $j = 1, 2, \ldots, n$. It is not difficult to verify that this $P_H^G(z)$ belongs to H, that it solves (2.1) if we just replace $\|x - z\|_G$ there by $\langle x - z, G(x - z)\rangle$, and that it is uniquely defined, although other solutions of (2.1) may exist due to the possibly zero-valued g_j's.

3. The component averaging (CAV) algorithm

Consider a set $\{G_i\}_{i=1}^m$ of real diagonal $n \times n$ matrices $G_i = \mathrm{diag}(g_{i1}, g_{i2}, \ldots, g_{in})$ with $g_{ij} \geqslant 0$ for all $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$, such that $\sum_{i=1}^m G_i = I$. The linear system $Ax = b$ is represented by the m hyperplanes

$$H_i = \{x \in \mathbb{R}^n \,|\, \langle a^i, x\rangle = b_i\}, \tag{3.1}$$

where $a^i = (a_j^i) \in \mathbb{R}^n$ is the ith row of A, $a^i \neq 0$ for all i, and $b = (b_i) \in \mathbb{R}^m$. Referring to the sparsity pattern of A we make the following definition.

Definition 3.1. A family $\{G_i\}_{i=1}^m$ of real diagonal $n \times n$ matrices with all diagonal elements $g_{ij} \geqslant 0$ such that $\sum_{i=1}^m G_i = I$ will be called *sparsity pattern oriented* (*SPO*, *for short*) *with respect to an* $m \times n$ *matrix* A if, for every $i = 1, 2, \ldots, m$, $g_{ij} = 0$ if and only if $a_j^i = 0$.

Our proposed CAV algorithm combines three features:

1. Each orthogonal projection onto H_i in (1.10) is replaced by a generalized oblique projection with respect to G_i.
2. The scalar weights $\{w_i\}$ in (1.10) are replaced by the diagonal weighting matrices $\{G_i\}$.
3. The actual weights are inversely proportional to the number of nonzero elements in each column, as motivated by the discussion preceding Eq. (1.9).

The iterative step resulting from the first two features has the form

$$x^{k+1} = x^k + \lambda_k \sum_{i=1}^{m} G_i\left(P_{H_i}^{G_i}(x^k) - x^k\right) \tag{3.2}$$

or equivalently, substituting from (2.3) for each $P_{H_i}^{G_i}$, we obtain:

Algorithm 3.1 (*Diagonal weighting for linear equations*).

Initialization: $x^0 \in \mathbb{R}^n$ is arbitrary.

Iterative step: Given x^k, compute x^{k+1} by using, for $j = 1, 2, \ldots, n$, the formula:

$$x_j^{k+1} = x_j^k + \lambda_k \sum_{\substack{i=1 \\ g_{ij}\neq 0}}^{n} \frac{b_i - \langle a^i, x^k\rangle}{\sum_{\substack{l=1 \\ g_{il}\neq 0}}^{n} \left(a_l^i\right)^2 / g_{il}} \cdot a_j^i, \tag{3.3}$$

where $\{G_i\}_{i=1}^m$ is a given family of diagonal SPO (with respect to A) weighting matrices as in Definition 3.1, and $\{\lambda_k\}_{k\geqslant 0}$ are relaxation parameters.

Finally, we specify how we construct the diagonal matrices $\{G_i\}_{i=1}^m$ in order to achieve the acceleration discussed in Section 1. Let s_j be the number of nonzero elements $a_j^i \neq 0$ in the jth column of A and define

$$g_{ij} \triangleq \begin{cases} \frac{1}{s_j} & \text{if } a_j^i \neq 0, \\ 0 & \text{if } a_j^i = 0. \end{cases} \tag{3.4}$$

With this particular SPO family of G_i's we obtain our CAV algorithm:

Algorithm 3.2 (*Component averaging*, *CAV*).

Initialization: $x^0 \in \mathbb{R}^n$ is arbitrary.

Iterative step: Given x^k, compute x^{k+1} by using, for $j = 1, 2, \ldots, n$, the formula:

$$x_j^{k+1} = x_j^k + \lambda_k \sum_{i=1}^{m} \frac{b_i - \langle a^i, x^k\rangle}{\sum_{l=1}^{n} s_l \left(a_l^i\right)^2} \cdot a_j^i, \tag{3.5}$$

where $\{\lambda_k\}_{k\geqslant 0}$ are relaxation parameters and $\{s_l\}_{l=1}^n$ are as defined above.

The CAV algorithm of (3.5) can be rewritten compactly in matrix notation. Let $S \triangleq \operatorname{diag}(s_1, s_2, \ldots, s_n)$, with the s_j's as defined above (we assume $s_j \neq 0$ for all $j = 1, 2, \ldots, n$), and define

$$D_S \triangleq \operatorname{diag}\left(\frac{1}{\|a^1\|_S^2}, \frac{1}{\|a^2\|_S^2}, \ldots, \frac{1}{\|a^m\|_S^2}\right). \tag{3.6}$$

Then (3.5) can be rewritten as

$$x^{k+1} = x^k + \lambda_k A^{\mathrm{T}} D_S (b - Ax^k). \tag{3.7}$$

This representation of CAV (suggested by an anonymous referee) does not reveal the logic that led us to (3.2). Note that (3.7) is similar in form to (1.7), but with a different diagonal matrix. It should not be confused with the *generalized Landweber iteration* which has the form

$$x^{k+1} = x^k + \alpha \Delta A^{\mathrm{T}} (b - Ax^k), \tag{3.8}$$

where α is related to the singular values of A, and Δ (called a *shaping matrix* in [32]) is a certain polynomial in $\alpha A^{\mathrm{T}} A$. For further information on the Landweber iteration see, e.g., [7,35]. Recent applications of Landweber-type algorithms in image reconstruction from projections appear in [31,32].

4. Convergence analysis of the CAV algorithm

We shall prove here that Algorithm 3.1, with $\lambda_k = 1$ for all $k \geqslant 0$, generates sequences $\{x^k\}$ which always converge, regardless of the initial point x^0 and independently from the consistency or inconsistency of the underlying system $Ax = b$. Moreover, it will always converge to a minimizer of a certain proximity function. Our analysis is an appropriately modified and revised adaptation of the unpublished study of Byrne and Censor [10], whose more general results also follow from Byrne and Censor [11, Algorithm 4.2].

Let $\{G_i\}_{i=1}^m$ be an SPO family with respect to A, as in Definition 3.1. For such diagonal matrices with nonnegative diagonal entries, we shall denote

$$|x|_{G_i} \triangleq \sqrt{\langle x, G_i x \rangle}. \tag{4.1}$$

Each $|x|_{G_i}$ is a *vector seminorm* (see, e.g., [27]) because it may be equal to zero for an $x \neq 0$ if G_i has at least one $g_{ij} = 0$. We define a *proximity function* F as follows:

$$F(x) \triangleq \sum_{i=1}^m |P_{H_i}^{G_i}(x) - x|_{G_i}^2. \tag{4.2}$$

Since we are dealing in our proof only with the case of unity relaxation ($\lambda_k = 1$ for all $k \geqslant 0$), we denote, for any $x \in \mathbb{R}^n$, the algorithmic operator of (3.2) by

$$T(x) \triangleq \sum_{i=1}^m G_i P_{H_i}^{G_i}(x). \tag{4.3}$$

We prepare the ground by proving several auxiliary lemmas.

Lemma 4.1. *If $\{G_i\}_{i=1}^m$ is SPO with respect to A, then for all $x, z \in \mathbb{R}^n$*

$$\sum_{i=1}^m |P_{H_i}^{G_i}(z) - x|_{G_i}^2 = \sum_{i=1}^m |P_{H_i}^{G_i}(z) - T(z)|_{G_i}^2 + \|T(z) - x\|_2^2. \tag{4.4}$$

Proof. We have

$$\sum_{i=1}^{m}\left(|P_{H_i}^{G_i}(z)-x|_{G_i}^2-|P_{H_i}^{G_i}(z)-T(z)|_{G_i}^2\right)$$
$$=\sum_{i=1}^{m}\left(|x|_{G_i}^2-|T(z)|_{G_i}^2+2\left\langle G_iP_{H_i}^{G_i}(z),T(z)-x\right\rangle\right). \tag{4.5}$$

Using (4.3) and the fact that, due to $\sum_{i=1}^{m}G_i=I$

$$\sum_{i=1}^{m}|y|_{G_i}^2=\|y\|_2^2 \tag{4.6}$$

for any $y\in\mathbb{R}^n$, we obtain the required result. □

Lemma 4.2. *For every sequence* $\{x^k\}_{k\geqslant 0}$, *generated by Algorithm* 3.1, *with* $\lambda_k=1$ *for all* $k\geqslant 0$, *the sequence* $\{F(x^k)\}_{k\geqslant 0}$, *with* F *defined as in* (4.2), *is decreasing and* $\lim_{k\to\infty}\|x^{k+1}-x^k\|_2=0$.

Proof. From (4.2) and Lemma 4.1 we obtain

$$F(x^k)=\sum_{i=1}^{m}|P_{H_i}^{G_i}(x^k)-T(x^k)|_{G_i}^2+\|T(x^k)-x^k\|_2^2. \tag{4.7}$$

From (4.3) and (3.2) with $\lambda_k=1$ for all $k\geqslant 0$, we have that $x^{k+1}=T(x^k)$, and since $P_{H_i}^{G_i}(x^{k+1})$ minimizes $|x-x^{k+1}|_{G_i}^2$ over all $x\in H_i$, we can continue (4.7) to obtain, for all $k\geqslant 0$

$$F(x^k)\geqslant\sum_{i=1}^{m}|P_{H_i}^{G_i}\left(x^{k+1}\right)-x^{k+1}|_{G_i}^2+\|x^{k+1}-x^k\|_2^2$$
$$=F(x^{k+1})+\|x^{k+1}-x^k\|_2^2\geqslant F(x^{k+1}). \tag{4.8}$$

The monotonicity and nonnegativity of $\{F(x^k)\}$ guarantee that the limit $\lim_{k\to\infty}F(x^k)$ exists, and thus, (4.8) implies $\lim_{k\to\infty}\|x^{k+1}-x^k\|_2=0$. □

The following lemma generalizes a classical result. We prove it here along the same lines as the proof of a similar result for Bregman generalized distances (see [8]) given in [14, Theorem 2.4.1].

Lemma 4.3. *Let* $H\subseteq\mathbb{R}^n$ *be a hyperplane*, G *a nonnegative diagonal matrix and* $z\in H$ *a given point. Then, for any* $y\in\mathbb{R}^n$, *the following inequality holds*:

$$|P_H^G(y)-y|_G^2\leqslant|z-y|_G^2-|z-P_H^G(y)|_G^2. \tag{4.9}$$

Proof. By expanding the function

$$E(u)\triangleq|u-y|_G^2-|u-P_H^G(y)|_G^2 \tag{4.10}$$

according to (4.1), we find that $E(u) = \langle u, \alpha \rangle + \beta$ for some $\alpha \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$ which are independent of u, thus $E(u)$ is convex. Denoting $u_\lambda \triangleq \lambda z + (1-\lambda)P_H^G(y)$, for any $0 \leqslant \lambda \leqslant 1$, we obtain, due to the convexity of $E(u)$

$$E(u_\lambda) \leqslant \lambda\Big(|z-y|_G^2 - |z-P_H^G(y)|_G^2\Big) + (1-\lambda)|P_H^G(y)-y|_G^2. \tag{4.11}$$

For $\lambda > 0$, this gives

$$\begin{aligned} &|z-y|_G^2 - |z-P_H^G(y)|_G^2 - |P_H^G(y)-y|_G^2 \\ &\quad \geqslant \frac{1}{\lambda}\Big(|u_\lambda - y|_G^2 - |P_H^G(y)-y|_G^2\Big) - \frac{1}{\lambda}|u_\lambda - P_H^G(y)|_G^2. \end{aligned} \tag{4.12}$$

The first term on the right-hand-side of (4.12) is nonnegative because of the minimization property of $P_H^G(y)$ and the second term tends to zero as $\lambda \to 0^+$. Therefore, for small enough positive values of λ, the right-hand-side of (4.12) is nonnegative. □

Definition 4.1. A sequence $\{x^k\}_{k \geqslant 0}$ is called *Fejér-monotone* with respect to a nonempty set $\Omega \subseteq \mathbb{R}^n$ if, for any $x \in \Omega$,

$$\|x - x^{k+1}\|_2 \leqslant \|x - x^k\|_2 \quad \text{for all } k \geqslant 0. \tag{4.13}$$

See, e.g., [14, Definition 5.3.1,]. It is easy to verify that any Fejér-monotone sequence is bounded. We denote the set of minimizers of the proximity function F by

$$\Phi \triangleq \Big\{\hat{x} \in \mathbb{R}^n | F(\hat{x}) \leqslant F(x) \text{ for all } x \in \mathbb{R}^n\Big\}. \tag{4.14}$$

The following lemma establishes the Fejér-monotonicity with respect to Φ of our CAV iterates.

Lemma 4.4. *If $\Phi \neq \emptyset$, then any sequence $\{x^k\}_{k \geqslant 0}$, generated by Algorithm* 3.1, *with $\lambda_k = 1$ for all $k \geqslant 0$, is Fejér-monotone with respect to Φ.*

Proof. Take any $\hat{x} \in \Phi$ and use (4.8) with $x^0 = \hat{x}$ and $x^1 = T(\hat{x})$. Then we get

$$F(\hat{x}) \geqslant F(T(\hat{x})) + \|T(\hat{x}) - \hat{x}\|_2^2. \tag{4.15}$$

Since $\hat{x}$ is a minimizer of F, $F(\hat{x}) \leqslant F(T(\hat{x}))$, yielding, by (4.15), that $T(\hat{x}) = \hat{x}$. Using Lemma 4.3 with $y = x^k$, $G = G_i$, $H = H_i$, $z = P_{H_i}^{G_i}(\hat{x})$, we obtain, for $i = 1, 2, \ldots, m$, that for any sequence $\{x^k\}_{k \geqslant 0}$, generated by Algorithm 3.1 with $\lambda_k = 1$ for all $k \geqslant 0$,

$$|P_{H_i}^{G_i}(\hat{x}) - x^k|_{G_i}^2 \geqslant |P_{H_i}^{G_i}(x^k) - x^k|_{G_i}^2 + |P_{H_i}^{G_i}(\hat{x}) - P_{H_i}^{G_i}(x^k)|_{G_i}^2. \tag{4.16}$$

Summing up all these inequalities, using Lemma 4.1 for the resulting left-hand side, and using (4.2), we obtain

$$\sum_{i=1}^{m} |P_{H_i}^{G_i}(\hat{x}) - T(\hat{x})|_{G_i}^2 + \|T(\hat{x}) - x^k\|_2^2 \geqslant F(x^k) + \Gamma(x^k), \tag{4.17}$$

where

$$\Gamma(v) \triangleq \sum_{i=1}^{m} |P_{H_i}^{G_i}(\hat{x}) - P_{H_i}^{G_i}(v)|_{G_i}^2. \tag{4.18}$$

Using (4.2) again and the fact that $T(\hat{x}) = \hat{x}$, (4.17) can be rewritten as

$$\|\hat{x} - x^k\|_2^2 \geqslant F(x^k) - F(\hat{x}) + \Gamma(x^k). \tag{4.19}$$

Since $\hat{x}$ minimizes F, we have $F(x^k) - F(\hat{x}) \geqslant 0$. Denoting

$$\gamma^i(v) \triangleq P_{H_i}^{G_i}(\hat{x}) - P_{H_i}^{G_i}(v), \tag{4.20}$$

we have, by (4.3)

$$\sum_{i=1}^{m} G_i\gamma^i(v) = \sum_{i=1}^{m} G_i P_{H_i}^{G_i}(\hat{x}) - \sum_{i=1}^{m} G_i P_{H_i}^{G_i}(v) = T(\hat{x}) - T(v) = \hat{x} - T(v). \tag{4.21}$$

Then, from (4.18) and (4.1), we get

$$\Gamma(v) = \sum_{i=1}^{m} \langle \gamma^i(v), G_i\gamma^i(v)\rangle \geqslant \left\langle \sum_{i=1}^{m} G_i\gamma^i(v), \sum_{i=1}^{m} G_i\gamma^i(v) \right\rangle = \|\hat{x} - T(v)\|_2^2, \tag{4.22}$$

which, along with (4.19) and $x^{k+1} = T(x^k)$, proves the required result. The inequality in (4.22) follows from convexity considerations: Due to $\sum_{i=1}^{m} G_i = I$ we have, for every $j = 1, 2, \ldots, n$, that $\sum_{i=1}^{m} g_{ij} = 1$ while $g_{ij} \geqslant 0$ for all i and all j. Convexity of the real-valued function $f(x) = x^2$, then implies that, for every $j = 1, 2, \ldots, n$,

$$\sum_{i=1}^{m} g_{ij}\left(\gamma_j^i(v)\right)^2 \geqslant \sum_{i=1}^{m} \left(g_{ij}\gamma_j^i(v)\right)^2. \tag{4.23}$$

Summing up all these inequalities over j yields the inequality in (4.22). □

Now we prove the convergence theorem.

Theorem 4.1. *If the proximity function F has minimizers, i.e., $\Phi \neq \emptyset$, then any sequence $\{x^k\}_{k\geqslant 0}$, generated by Algorithm* 3.1, *with $\lambda_k = 1$ for all $k \geqslant 0$, converges to a minimizer of F.*

Proof. It follows from the Fejér-monotonicity, established in Lemma 4.4, that $\{x^k\}_{k\geqslant 0}$ is bounded, thus it has at least one cluster point. We show now that any cluster point of $\{x^k\}_{k\geqslant 0}$ is a minimizer of F.

Let x^* be a cluster point of $\{x^k\}_{k\geqslant 0}$ and let $\hat{x} \in \Phi$. Assume, by way of negation, that $x^* \notin \Phi$. Using Lemma 4.3 with $y = x^*$, $G = G_i$, $H = H_i$, $z = P_{H_i}^{G_i}(\hat{x})$, for $i = 1, 2, \ldots, m$, summing up all inequalities, using Lemma 4.1 and Eqs. (4.2) and (4.18), we obtain

$$F(\hat{x}) + \|\hat{x} - x^*\|_2^2 \geqslant F(x^*) + \Gamma(x^*). \tag{4.24}$$

The second assertion of Lemma 4.2 guarantees that $T(x^*) = x^*$ because x^* is a cluster point, thus (4.24) and (4.22) lead to the contradiction $F(\hat{x}) \geqslant F(x^*)$.

Finally, if x^* is a cluster point of $\{x^k\}_{k\geqslant 0}$, then $x^* \in \Phi$ and Lemma 4.4 guarantees that, for all $k \geqslant 0$,

$$0 \leqslant \|x^* - x^{k+1}\|_2 \leqslant \|x^* - x^k\|_2. \tag{4.25}$$

Thus, the limit $\lim_{k\to\infty} \|x^* - x^k\|_2$ exists and since x^* is a cluster point, this limit must be zero, proving that x^* is the limit of $\{x^k\}_{k\geqslant 0}$. □

When all matrices $G_i = (1/m)I,\ i = 1, 2, \ldots, m$, then the proximity function F of (4.2) is the classical least-squares measure, and the projections $P_{H_i}^{G_i}$ are the orthogonal projections. For this case, Iusem and De Pierro [23] proved local convergence in the inconsistent case, while Combettes [16] showed global convergence in the inconsistent case by employing a product space formulation which can handle the inconsistent case and extends the one by Pierra [33].

5. Experimental results on a problem of image reconstruction from projections

Three algorithms were implemented on a problem of image reconstruction from projections, described in Subsection 5.1. The three algorithms are ART, Cimmino (referred to as CIM), and the new "component averaging", which we refer to as CAV. All three methods were implemented sequentially, and CIM and CAV were also implemented in parallel using MPI (message passing interface – a standard, well-known tool for parallel computing on distributed memory system). The implementations were done within SNARK93, a software package for testing and evaluating algorithms for image reconstruction, see [9].

The algorithms were implemented on an IBM SP2 parallel machine with 64 processors of type Thin2. 58 of the processors have a memory of 128 MB, and six of them have 512 MB. The inter-processor communication was based on US (user space) protocol, with point-to-point bandwidth of 35 MB/s, and 40 ms latency. The programs were compiled with the IBM AIX XL Fortran compiler version 4, with optimization flag O3.

We compare the algorithms on the basis of their qualitative and quantitative behavior. The parallel implementations were executed on various sized groups of processors in order to test their scalability and their efficiency.

5.1. Problem and algorithm description

In the medical application of transmission computerized tomography (TCT), a planar cross-section of the body is considered and the tissue's attenuation of X-rays everywhere in the cross-section has to be reconstructed. This unknown function of two variables has real nonnegative values and is called the *image* or *picture*. The fundamental model in the *finite series-expansion approach*, see, e.g., [12], is formulated as follows. A Cartesian grid of square picture-elements, called *pixels*, is introduced into the region of interest so that it covers the whole picture that has to be

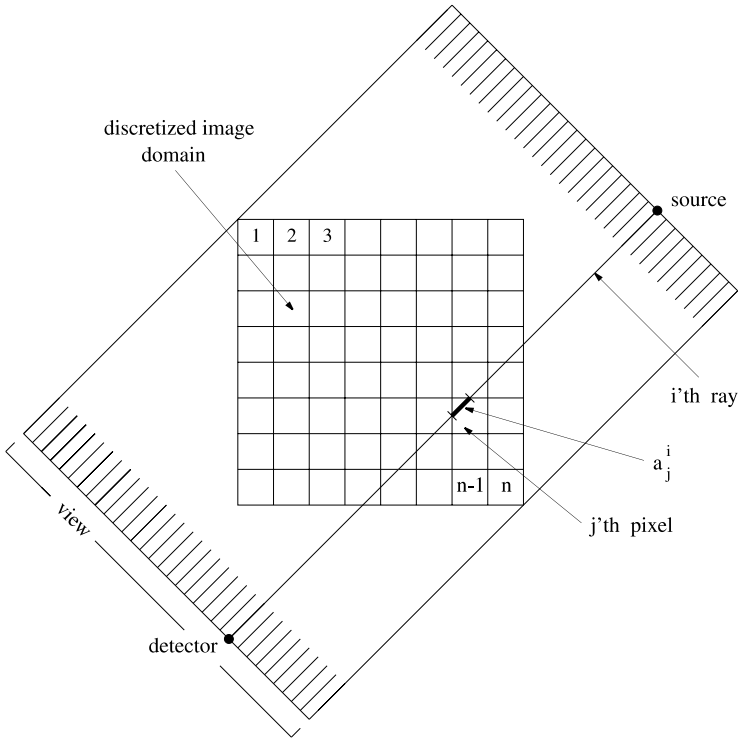

Fig. 1. The fully discretized model for transmission tomography image reconstruction.

reconstructed. The pixels are numbered in some agreed manner, say from 1 (top left corner pixel) to n (bottom right corner pixel); see Fig. 1.

The X-ray attenuation function is assumed to take a constant uniform value x_j throughout the jth pixel, for $j = 1, 2, \ldots, n$. Sources of X-rays and detectors are assumed to be points, and the rays between them – lines. It is further assumed that the length of intersection of the ith ray with the jth pixel, denoted by a_j^i for all $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$, represents the contribution of the jth pixel to the total attenuation along the ith ray.

The physical measurement of the total attenuation along the ith ray, denoted by b_i, represents the line integral of the unknown attenuation function along the path of the ray. Therefore, in this fully discretized model, each line integral is approximated by a finite sum and the model is described by a system of linear equations

$$\sum_{j=1}^{n} x_j a_j^i = b_i, \quad i = 1, 2, \ldots, m. \tag{5.1}$$

Here, $b = (b_i) \in \mathbb{R}^m$ is the *measurements vector*, $x = (x_j) \in \mathbb{R}^n$ is the *image vector* and the $m \times n$ matrix $A = (a_j^i)$ is the *projection matrix*.

We demonstrate the performance of the three algorithms on the reconstruction of the Herman head phantom [21, Section 4.3] and perform the experiment within the SNARK93 programming system of Browne et al. [9]. The Herman head phantom is specified by a set of ellipses, with a specific attenuation value attached to each elliptical region. The values of $b_i,\ i = 1, 2, \ldots, m$, are calculated by computing the line integrals through the elliptical regions (without reference to the discretization). Thus, system (5.1) is basically inconsistent, because the left-hand side is only an approximation to the actual integrals. This matches the real-life situation where the b_i's are actual X-ray readings through an object but the region of interest is discretized as above.

The performance of the new CAV algorithm is compared with CIM (Algorithm 1.1) with fixed scalar weights and with the row-action sequential ART algorithm – see, e.g., [21, Chapter 11] and [14, Algorithm 5.4.3]. In this experimental section of our paper we use the term *iteration* to mean a single whole sweep through all equations of the system. All our experiments were initiated with $x^0 = 0$.

In the sequential implementation of CAV, we compute all the values $s_j,\ j = 1, 2, \ldots, n$, during the first iteration, and then we compute the denominators of Eq. (3.5) and store them. These are then used in subsequent iterations.

The parallel implementation of CIM and CAV is straightforward. Each processor is assigned an equal (or almost equal) number of equations. For CIM, the sums in Eq. (1.6) are computed as follows: each processor computes the partial sums (for $j = 1, 2, \ldots, n$) associated with its equations. These partial sums are then gathered, summed up, and redistributed to all the processors (using the MPI routine MPI – ALLREDUCE). Each processor then multiplies the sums by λ_k/m and adds them to x^k to get the new iterate x^{k+1}.

For CAV, some initial processing is done as follows: each processor computes the number of nonzero elements in each column (i.e., for $j = 1, 2, \ldots, n$) in its set of equations. These numbers are then gathered, summed, and redistributed to all the processors. As a result, each processor has all the values of s_j for $j = 1, 2, \ldots, n$, and it uses them to compute the denominators of Eq. (3.5) for its set of equations. These denominators do not change during the iterations, so each processor stores its set of denominators in its local memory for use during the iterative steps. Subsequently, in each iteration, each processor computes the partial sums (related to its set of equations) of Eq. (3.5) for $j = 1, 2, \ldots, n$. These partial sums are then gathered, summed and redistributed. Each processor then multiplies the sums by λ_k and adds them to the current iterate x^k to obtain the new iterate x^{k+1}.

5.2. Qualitative differences

Qualitative differences indicate the different behavior and applicability of the various methods. Such differences provide a quick overview of the basic properties that a user might need from an algorithm. Table 1 summarizes the qualitative differences between ART, CIM, and CAV. Regarding the parallelization of ART, recall the comment made about this in Section 1.

Table 1
Qualitative differences between ART, CIM and CAV

Property	ART	CIM	CAV
Inherently parallel	No	Yes	Yes
Sequential convergence	Acceptable	Slow	Acceptable
Parallel convergence	n/a	Slow	Fast
Parallel efficiency (2–64 proc.)	n/a	1–0.5	1–0.5
Image quality (50 iterations)	Good	Unacceptable	Good

5.3. Sequential comparisons

In this subsection we show representative test results for one processor. Four different test cases, using the Herman head phantom, are examined. The four cases used differing image resolutions and differing numbers of projections and rays per projection, which resulted in differing numbers of variables and equations, as shown in Table 2.

The three algorithms were run with two different relaxation parameters, (which remained fixed throughout the iterations): CIM – 1.0, 2.0; ART – 0.1, 1.0; CAV – 1.0, 2.0. Three different measures were used for comparisons: *distance*, *relative error*, and *standard deviation*, calculated by SNARK93 [9], and defined there as follows.

Let x_j^k and $\tilde{x}_j$ denote the density assigned to the jth pixel of the reconstruction after k iterations, and the density of the jth pixel in the phantom, respectively. Let S denote the set of indices j of pixels which are in the region of interest and let α be the number of elements in S. The *average* value of the reconstructed image x^k is given by

$$\rho_k \triangleq \frac{1}{\alpha}\sum_{j\in S} x_j^k, \tag{5.2}$$

and the *variance* of x^k is given by

$$v_k \triangleq \frac{1}{\alpha}\sum_{j\in S}(x_j^k - \rho_k)^2. \tag{5.3}$$

The *standard deviation* of the reconstructed image x^k is then

$$\sigma_k \triangleq \sqrt{v_k}. \tag{5.4}$$

Similarly, we define the average value $\tilde{\rho}$, variance $\tilde{v}$ and standard deviation $\tilde{\sigma}$ for the phantom, in terms of the phantom values $\tilde{x}_j$.

Table 2
The four different test cases

Cases	Equations	Variables	Image size	Projections	Rays
1	13,137	13,225	115×115	151	87
2	26,425	13,225	115×115	151	175
3	126,655	119,025	345×345	365	347
4	232,275	119,025	345×345	475	489

The *distance* between x^k and the phantom $\tilde{x}$ is

$$\delta_k \triangleq \begin{cases} (1/\tilde{\sigma})\sqrt{(1/\alpha)\sum_{j\in S}(x_j^k - \tilde{x}_j)^2} & \text{if } \tilde{\sigma} > 0, \\ \sqrt{\sum_{j\in S}(x_j^k - \tilde{x}_j)^2} & \text{if } \tilde{\sigma} \leqslant 0. \end{cases} \tag{5.5}$$

With $\tau \triangleq \sum_{j\in S}|\tilde{x}_j|$ the *relative error* of x^k is defined by

$$\epsilon_k \triangleq \begin{cases} (1/\tau)\sum_{j\in S}|x_j^k - \tilde{x}_j| & \text{if } \tau > 0, \\ \sum_{j\in S}|x_j^k - \tilde{x}_j| & \text{if } \tau \leqslant 0. \end{cases} \tag{5.6}$$

Convergence results for the four test cases are shown in Figs. 2–13. For each case, we present the distance measure, the relative error, and the standard deviation. The actual time for a single iteration is the same for all three algorithms (since it means a complete sweep through all equations), hence the number of iterations is a true timing comparison measure between the algorithms.

The results indicate that CIM is by far much slower to converge than the other two methods. A comparison of ART and CAV shows that ART initially converges faster, but CAV is eventually better. In fact, ART even deteriorates after a certain number of iterations, while CAV continues to improve.

Some other facts emerge from these results: from the pairs of values that we used, the better relaxation parameters for each method are as follows: CIM – 2.0, ART – 0.1, and CAV – 2.0. These values were obtained after many runs with different relaxation parameters. Note that ART converges to different values with different relaxation parameters. This is due to the fact that in the inconsistent case, ART only converges cyclically, and that only the small relaxation parameter causes the limits of the cycles to approach the geometric least-squares point, see, e.g., [13]. On the other hand, CAV with different relaxation parameters approaches the same solution, and the differences are only in the initial rate of convergence.

Visual comparisons. Figs. 14–25 correspond to the four test cases, showing the phantom and the images reconstructed with ART, CIM and CAV, after 10, 20, 30, 40 and 50 iterations. The results for the Cimmino algorithm atest to its slow convergence. Only weak shadows of some of the internal structures can be noticed. Each test case is shown with the better relaxation parameter, as noted above. Generally, it can be seen that ART produces sharper images within a small number of iterations, and this is in accordance with the convergence plots, where the initial behavior of ART is superior. However, at some stage ART begins to deteriorate, while the images produced by CAV improve with increased iteration numbers. It can also be seen that both ART and CAV produce better images when more equations are used for the same phantom. Note also that in case 1, the circular artifacts produced by ART are more strongly pronounced than those produced by CAV, for all iteration numbers.

5.4. Parallel behavior of CAV

Of the three algorithms, we implemented only CAV in parallel. As noted in Section 1, ART is less suitable for parallelization for the problem of image

Table 3
Run times (in s) and efficiency of CAV for the four test cases

No. of processors		1	2	4	8	16	32	48	64
Case 1									
	Time	0.752	0.385	0.218	0.132	0.0845	0.065	0.063	0.08
	Efficiency	1.0	0.98	0.86	0.71	0.55	0.36	0.25	0.15
Case 2									
	Time	1.480	0.753	0.397	0.238	0.142	0.093	0.085	0.093
	Efficiency	1.0	0.98	0.93	0.78	0.65	0.50	0.36	0.25
Case 3									
	Time	22.10	11.15	5.875	3.56	2.126	1.323	1.01	0.92
	Efficiency	1.0	0.99	0.94	0.78	0.64	0.52	0.46	0.37
Case 4									
	Time	42.48	21.8	11.95	6.22	3.83	2.225	1.628	1.355
	Efficiency	1.0	0.975	0.89	0.85	0.693	0.60	0.543	0.49

reconstruction from projections, and the parallel behavior of CIM is essentially similar to that of CAV (as far as run-time per iteration and efficiency are concerned), but it converges much more slowly.

Table 3 shows the run-times (in s) and efficiency of CAV on the SP2, for the four test cases, with the number of processors varying from 1 to 64. The usual definition of efficiency is as follows: if T_p denotes the time for p processors, then the efficiency is $T_1/(T_p \times p)$. The run-times include the recalculation of the matrix elements during each iteration, which is the standard method employed by SNARK93, designed to save space for the nonzero matrix elements.

5.5. Other experiments

We have also implemented the iterative scheme with (1.9) instead of (3.5) in Algorithm 3.2. In matrix notation, (1.9) is equivalent to

$$x^{k+1} = x^k + \lambda_k SA^{\mathrm{T}}(mD)(b - Ax^k), \tag{5.7}$$

where $S = \mathrm{diag}(s_1, s_2, \ldots, s_n)$ and D is given by (1.8). This formulation might appear similar to the generalized Landweber iteration (Eq. (3.8)) but it is, in fact, quite distinct due to the appearance of mD in (5.7) and the specific values appearing to the left of A^{T} in (3.8), as noted at the end of Section 3.

In contrast with CAV, the resulting scheme uses orthogonal (instead of oblique) projections, but it retains features 2 and 3 mentioned in the text following Definition 3.1.

The convergence plots of this particular scheme were very similar to those of CAV. This first led us to believe that the schemes behave identically on problems of image reconstruction from projections. However, the resulting images were inferior

to those of CAV, with pronounced artifacts in some cases. Furthermore, one should note that we have no proof of convergence for that scheme.

Two conclusions can be drawn from this experiment regarding the implementation of these schemes on the problem of image reconstruction. One is that the *initial rate of convergence* of CAV is due to the sparsity-induced diagonal weighting rather than to the oblique projections. The second conclusion is that the oblique projections, in some sense, more accurately capture the essence of the problem and produce better images. The precise nature of this difference is a topic for further research.

Note that if all the s_j's are equal, then (3.5) reduces to (1.9). This might explain the similar rate of convergence of the two schemes, since a histogram of the values of the s_j's indicated that most of them do not differ much. Again, further research is required in order to precisely quantify how the difference between the schemes depends on the variability of the s_j's.

6. Conclusions

Cimmino's simultaneous algorithm, for the case of a very large system of sparse linear equations, exhibits a very slow convergence rate, due to the fact that the change between successive iterates is relatively small. This observation has led to a modified algorithm, in which the change depends only on the nonzero elements of each column. The resulting algorithm, which we called CAV, is a special case of the general diagonally weighted simultaneous algorithm of Byrne and Censor [10,11]. The behavior of CAV in the inconsistent case is similar to Cimmino's algorithm: with unity relaxation, it provably converges to the geometric least-squares solution, and it appears from experimental results that this property also holds for other values of the relaxation parameter.

CAV is as easily parallelizable as Cimmino's original algorithm, but its practical convergence rate is close to that of ART, which is essentially sequential in nature. With regard to sequential implementation, we noticed that after many iterations, ART deteriorated somewhat, whereas CAV continued to improve. However, it should be emphasized that we did not invest any additional efforts into the implementation of ART except for using the small relaxation parameter – see in this respect Herman and Meyer [22].

Future research on CAV will concentrate on its behavior on other cases of sparse systems, and on its generalization to systems of linear inequalities and nonlinear systems.

Acknowledgements

We gratefully acknowledge the fruitful discussions and constructive comments of our colleagues Martin Altschuler, Åke Björck, Lev Bregman, Charles Byrne, Tommy Elfving, Gabor Herman, Krzysztof Kiwiel, Arnold Lent and Robert Lewitt. In

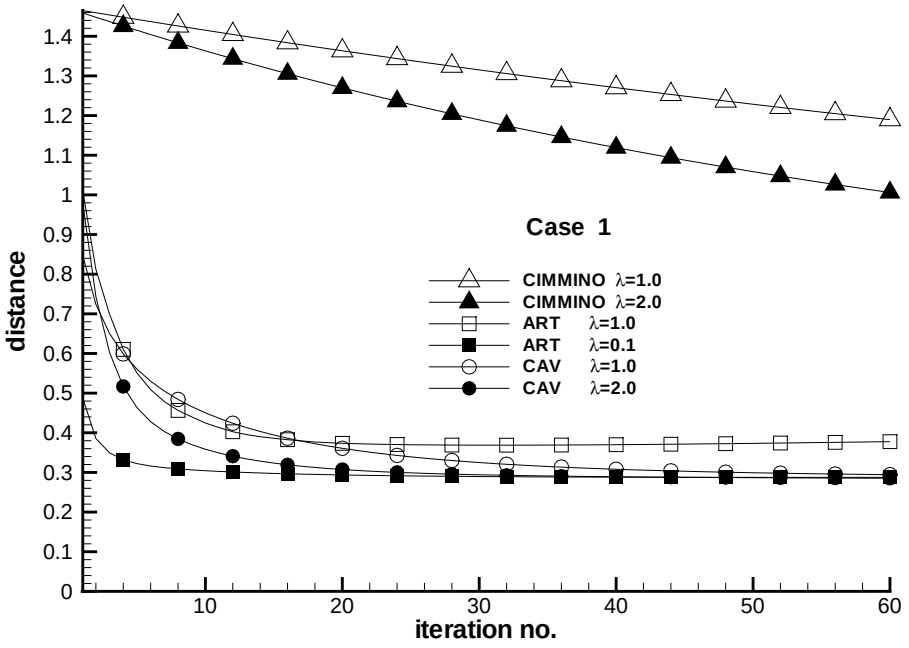

Fig. 2. Distance measure for case 1 (13,137 equations, 13,225 variables).

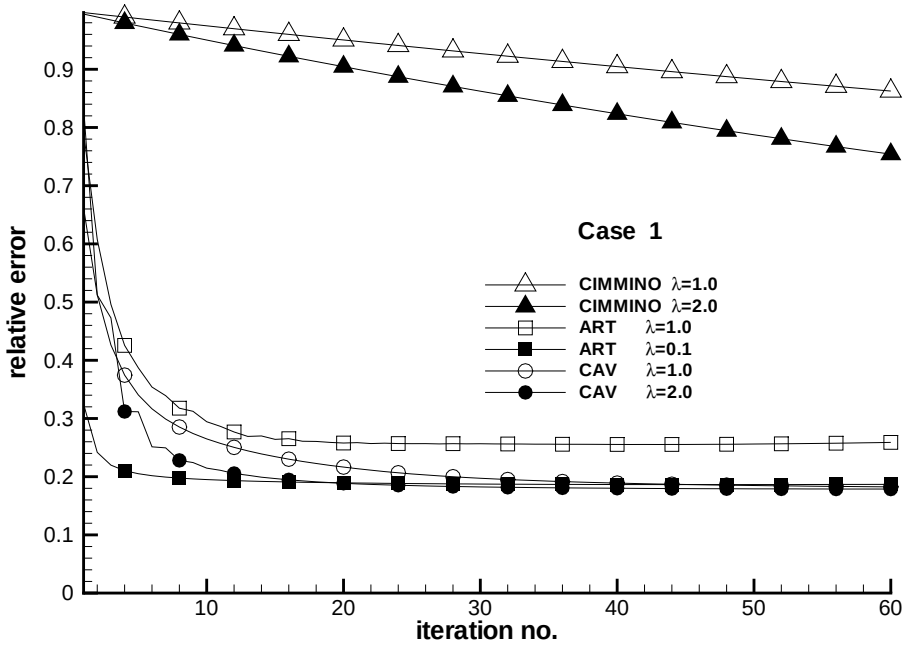

Fig. 3. Relative error for case 1 (13,137 equations, 13,225 variables).

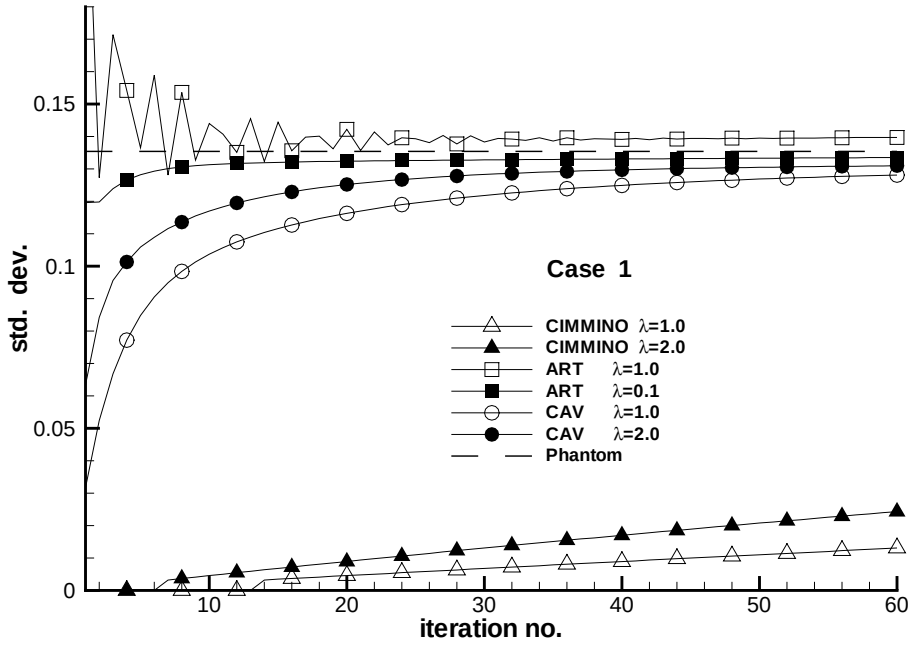

Fig. 4. Standard deviation for case 1 (13,137 equations, 13,225 variables).

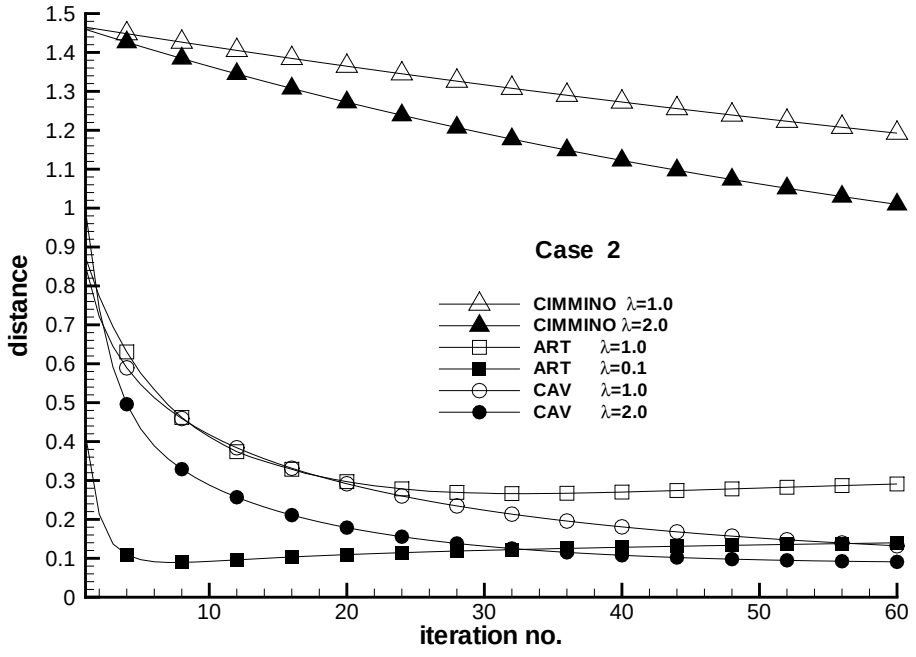

Fig. 5. Distance measure for case 2 (26,425 equations, 13,225 variables).

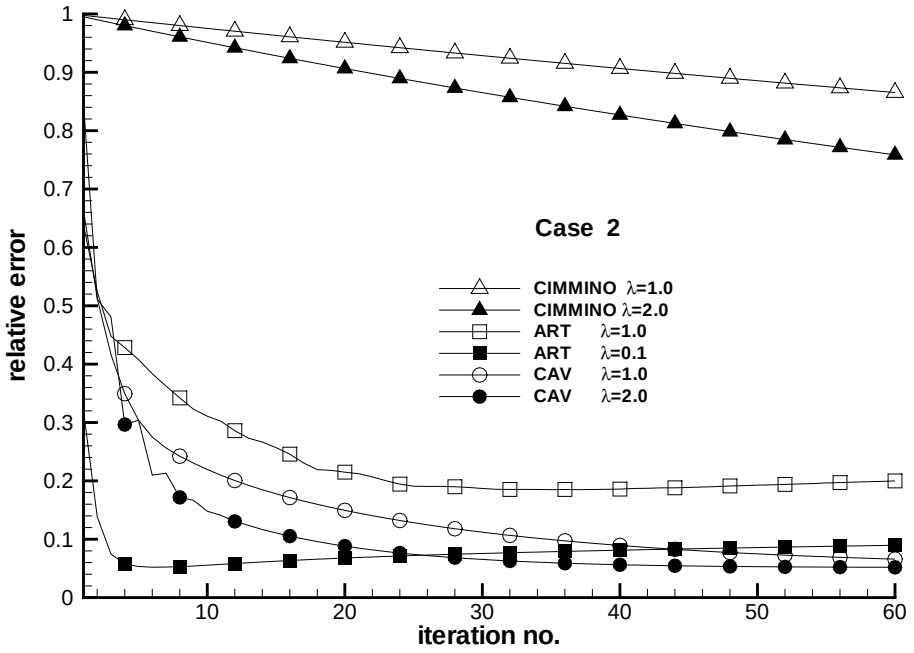

Fig. 6. Relative error for case 2 (26,425 equations, 13,225 variables).

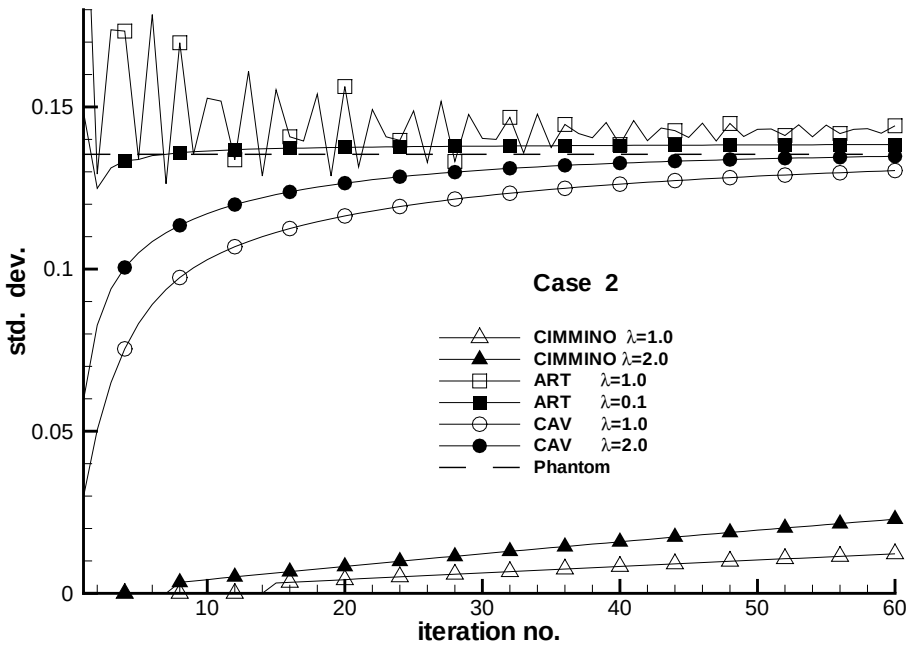

Fig. 7. Standard deviation for case 2 (26,425 equations, 13,225 variables).

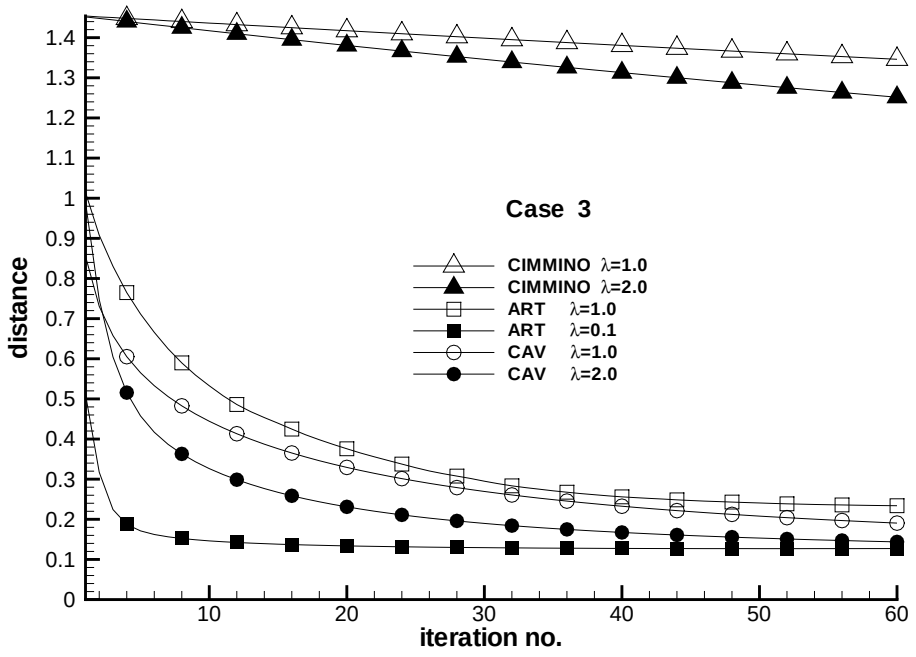

Fig. 8. Distance measure for case 3 (126,655 equations, 119,025 variables).

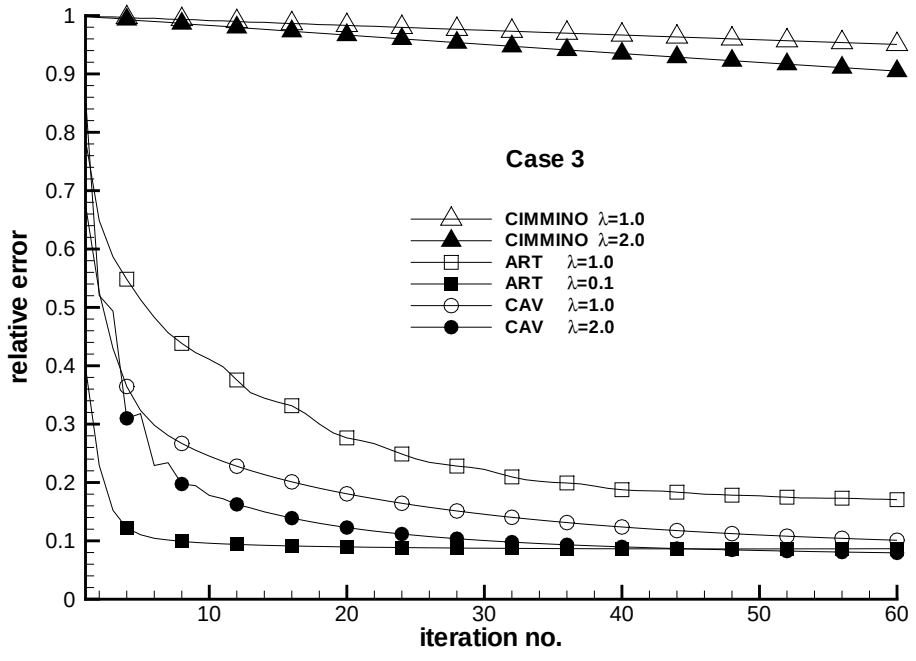

Fig. 9. Relative error for case 3 (126,655 equations, 119,025 variables).

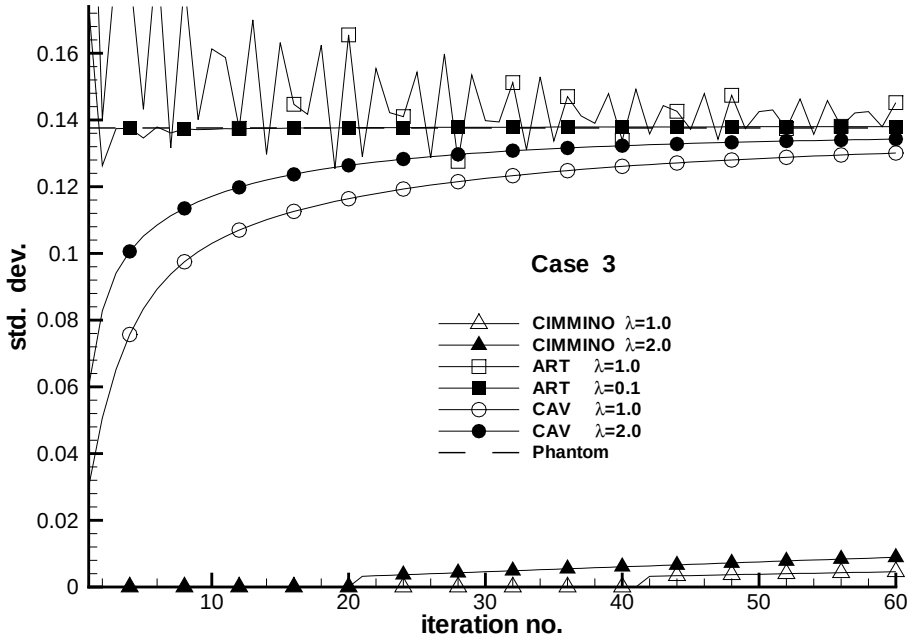

Fig. 10. Standard deviation for case 3 (126,655 equations, 119,025 variables).

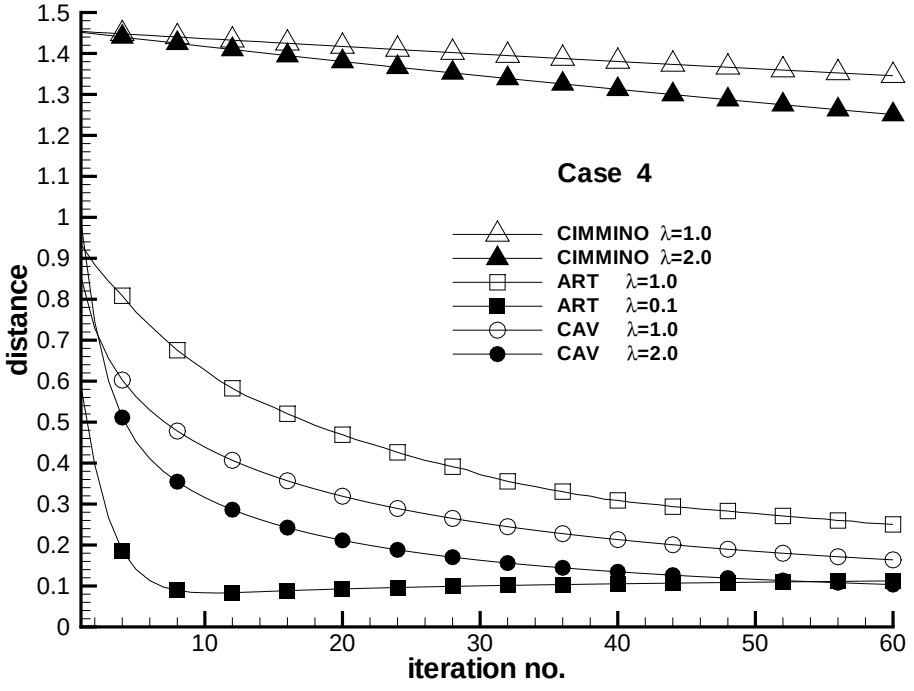

Fig. 11. Distance measure for case 4 (232,275 equations, 119,025 variables).

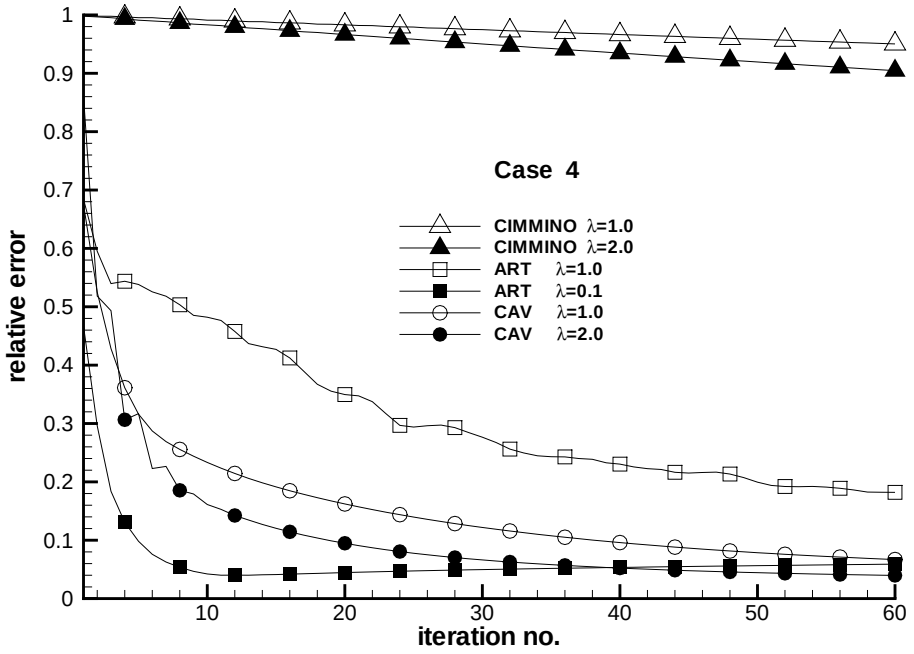

Fig. 12. Relative error for case 4 (232,275 equations, 119,025 variables).

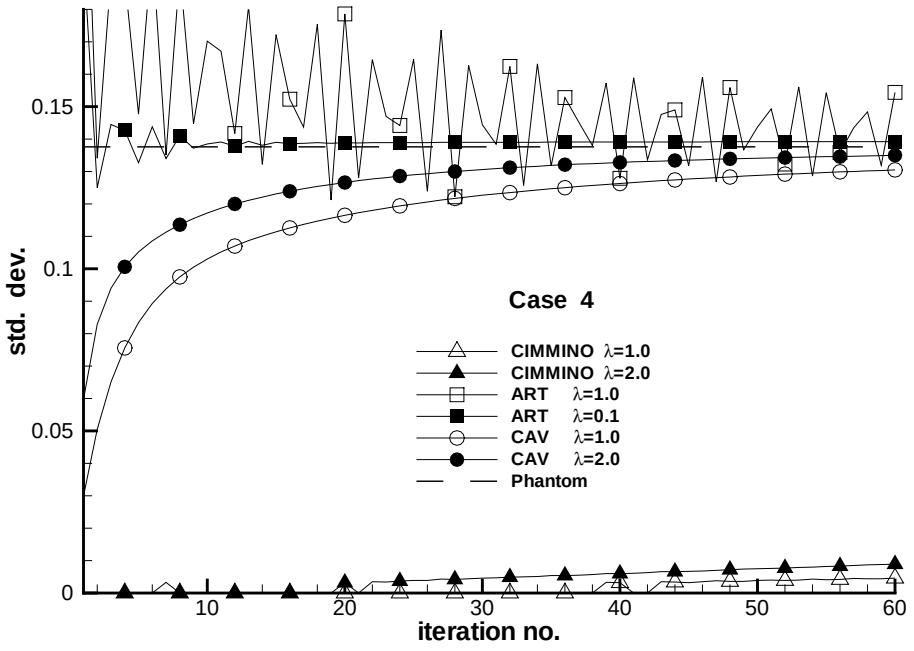

Fig. 13. Standard deviation for case 4 (232,275 equations, 119,025 variables).

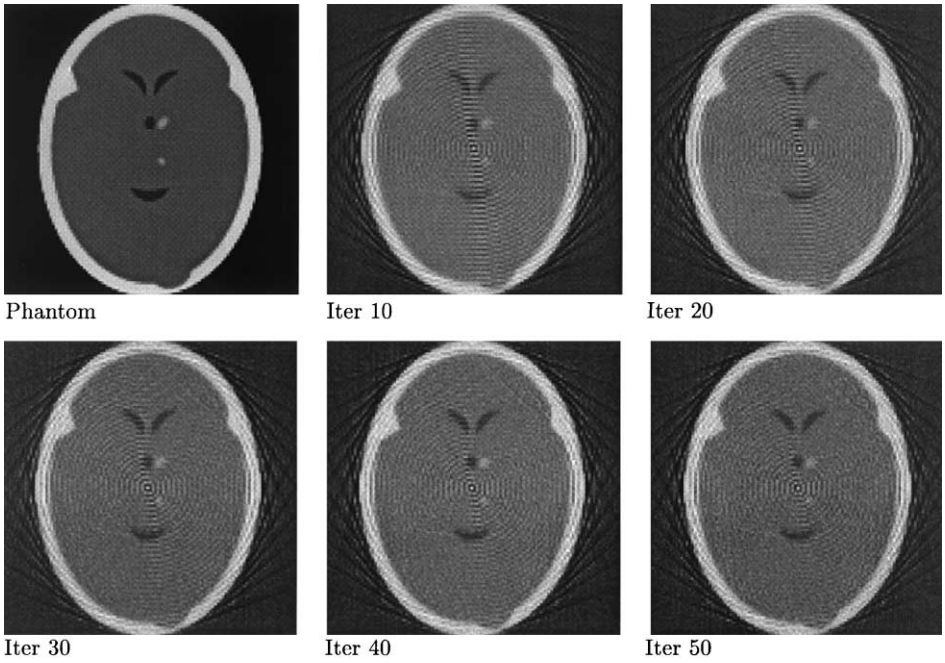

Fig. 14. Images produced by ART for case 1 (115×115 pixels, 13,137 equations, 13,225 variables).

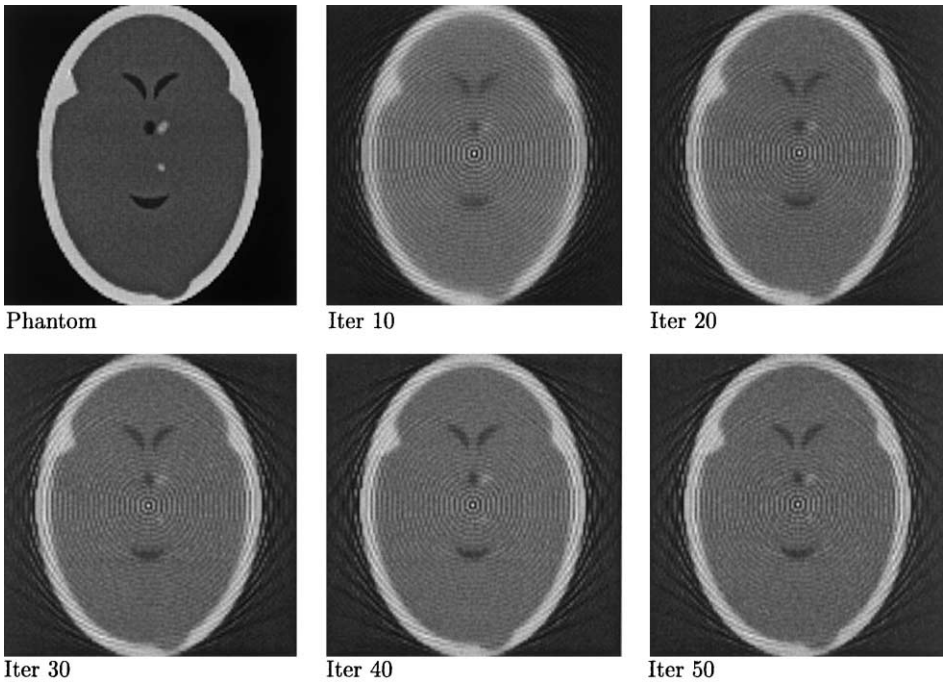

Fig. 15. Images produced by CAV for case 1 (115×115 pixels, 13,137 equations, 13,225 variables).

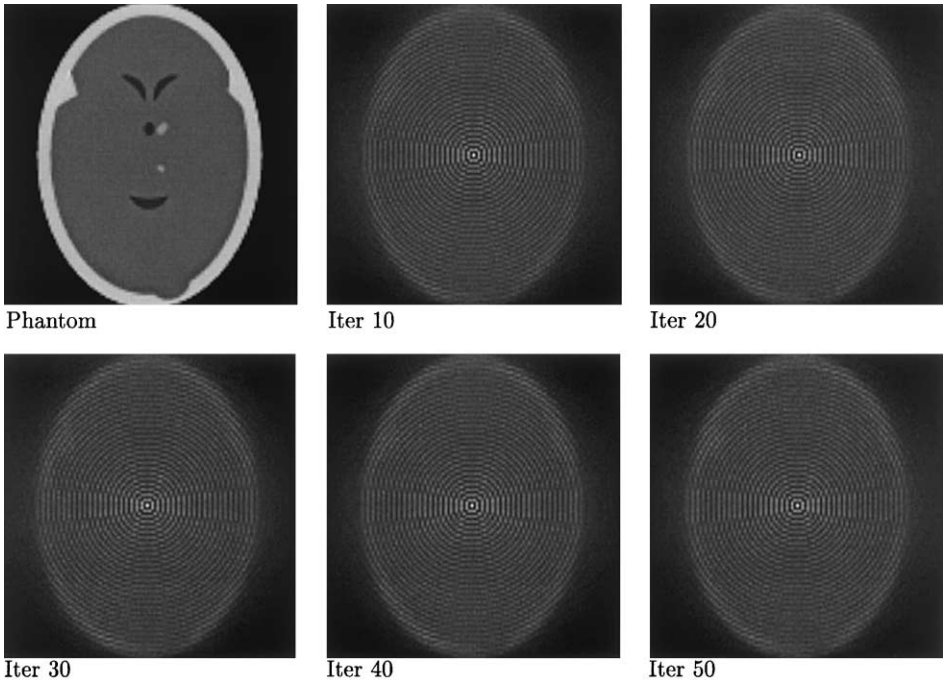

Fig. 16. Images produced by CIM for case 1 (115×115 pixels, 13,137 equations, 13,225 variables).

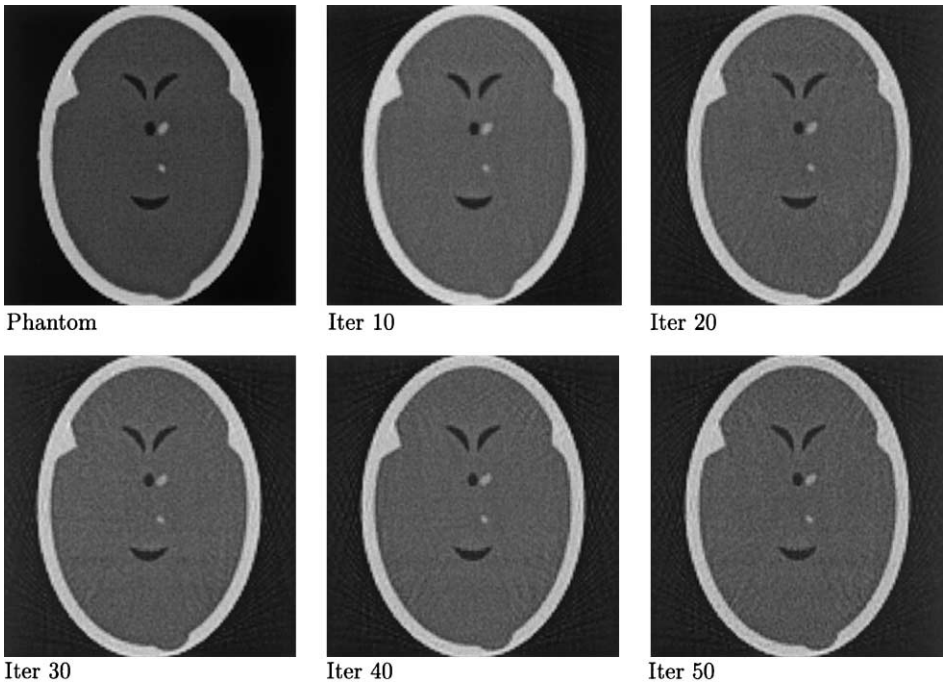

Fig. 17. Images produced by ART for case 2 (115×115 pixels, 26,425 equations, 13,225 variables).

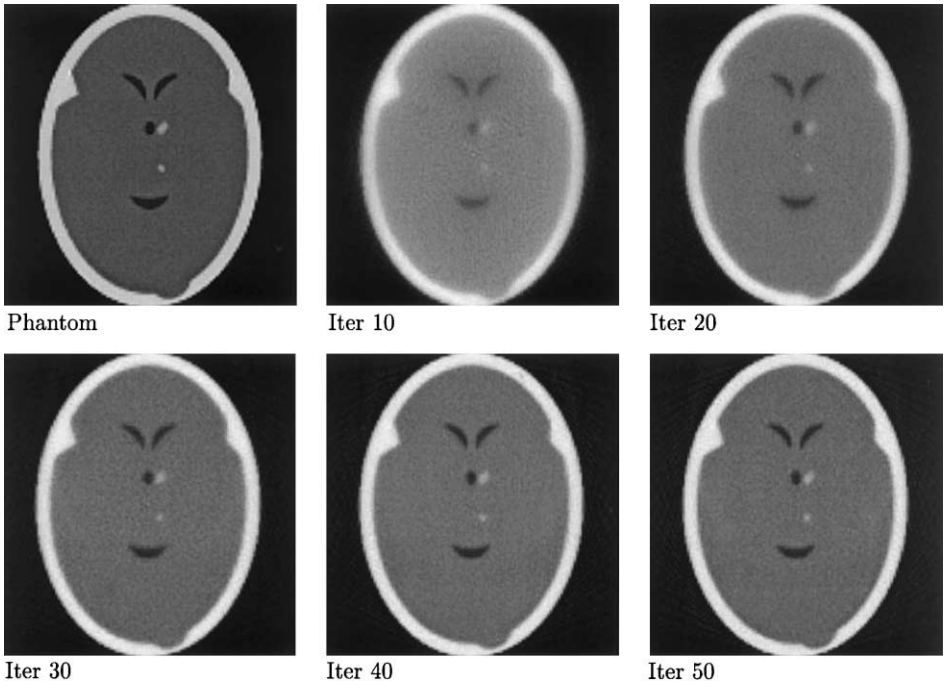

Fig. 18. Images produced by CAV for case 2 (115×115 pixels, 26,425 equations, 13,225 variables).

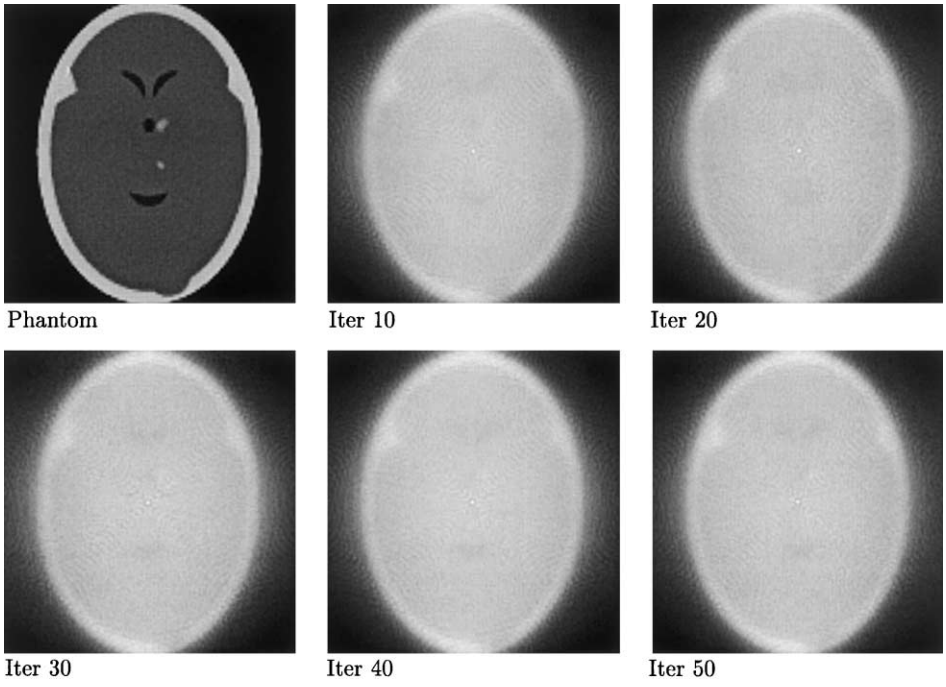

Fig. 19. Images produced by CIM for case 2 (115×115 pixels, 26,425 equations, 13,225 variables).

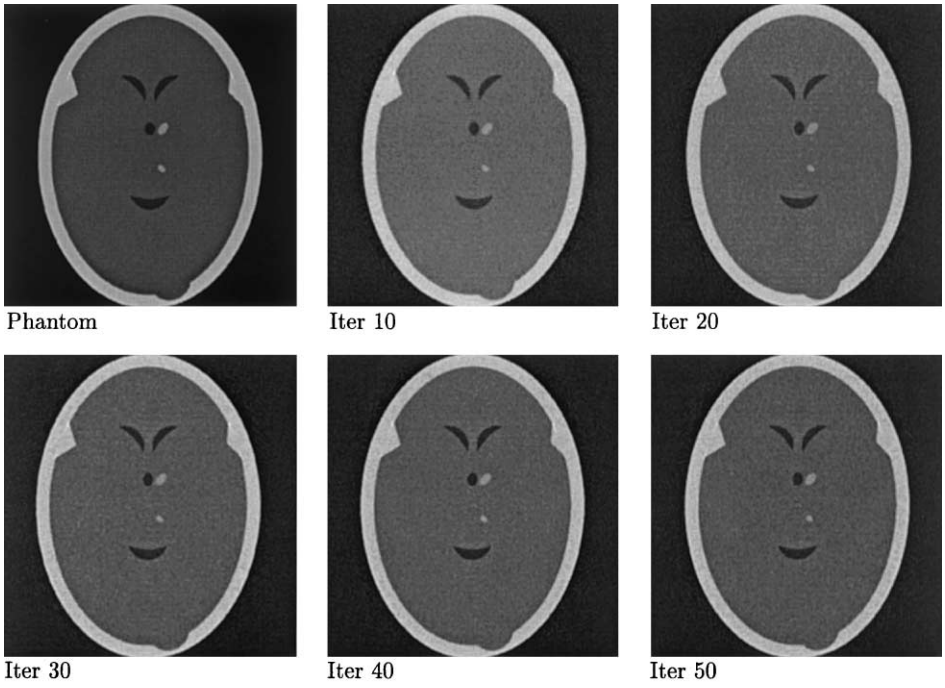

Fig. 20. Images produced by ART for case 3 (345 × 345 pixels, 126,655 equations, 119,025 variables).

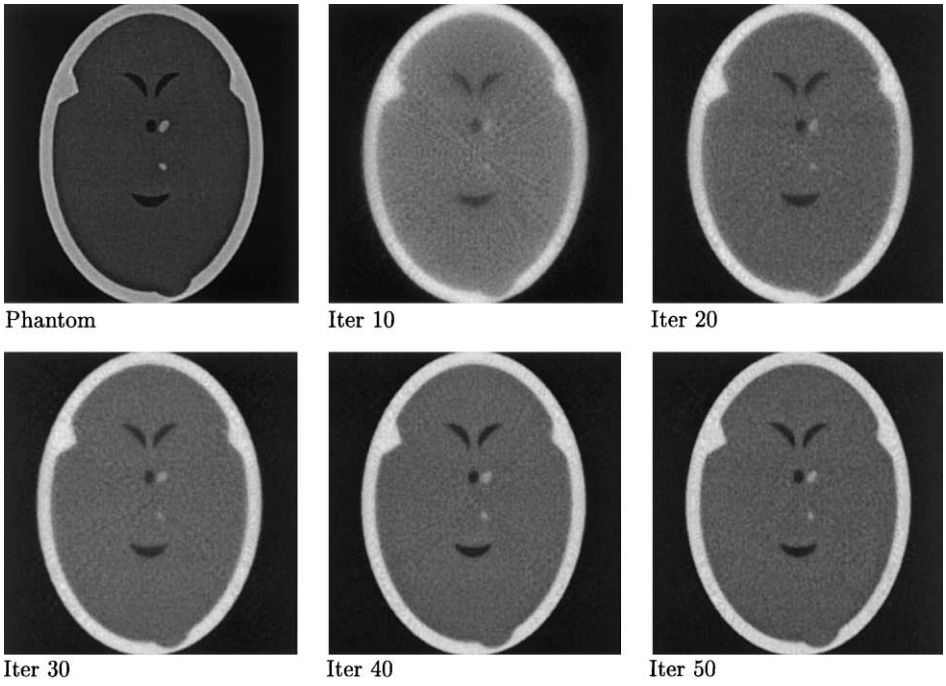

Fig. 21. Images produced by CAV for case 3 (345 × 345 pixels, 126,655 equations, 119,025 variables).

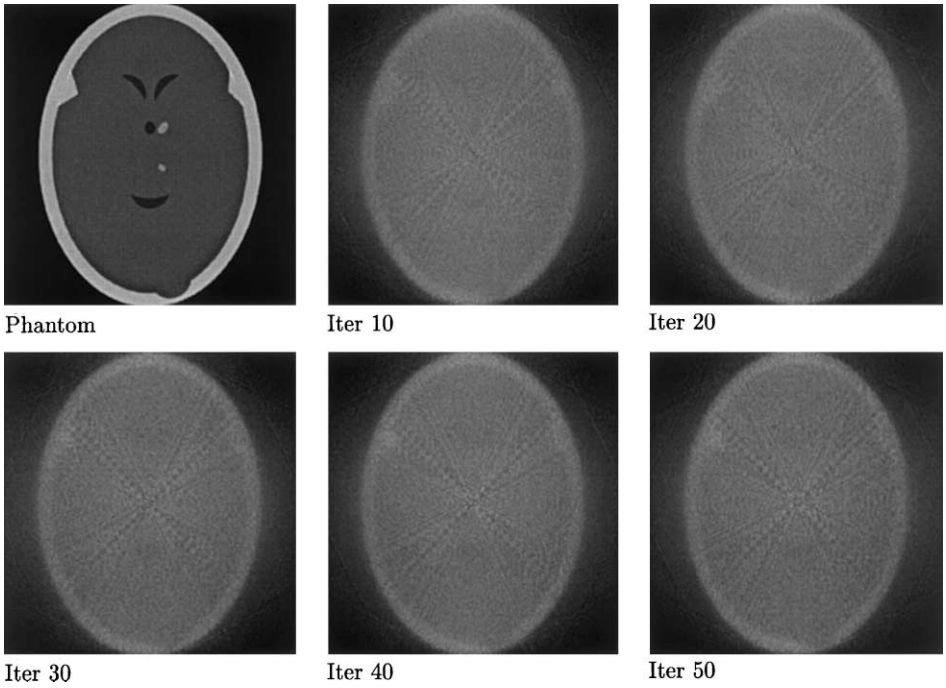

Fig. 22. Images produced by CIM for case 3 (345 × 345 pixels, 126,655 equations, 119,025 variables).

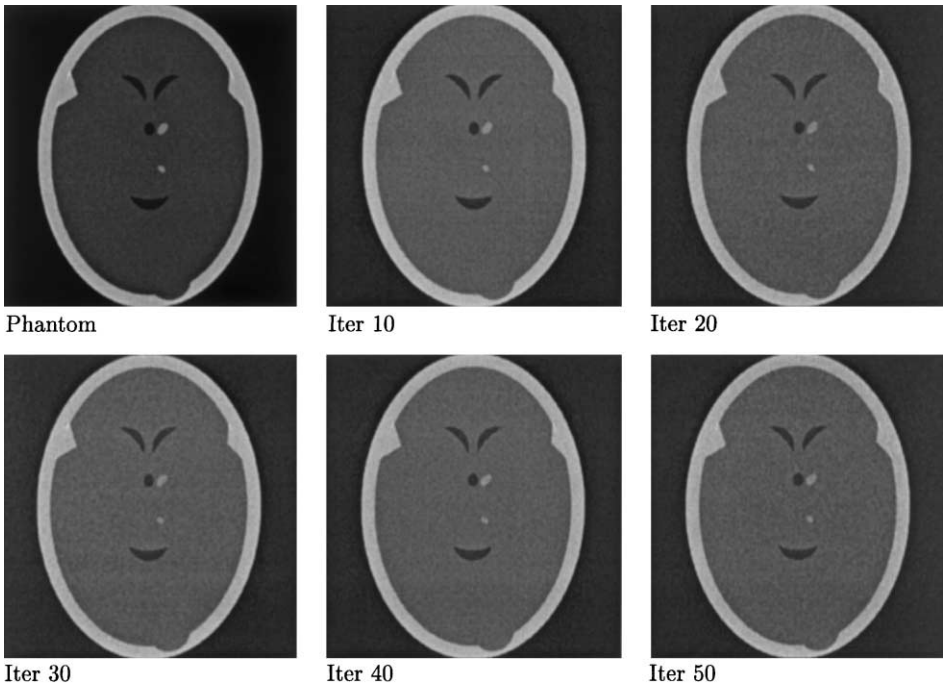

Fig. 23. Images produced by ART for case 4 (345 × 345 pixels, 232,275 equations, 119,025 variables).

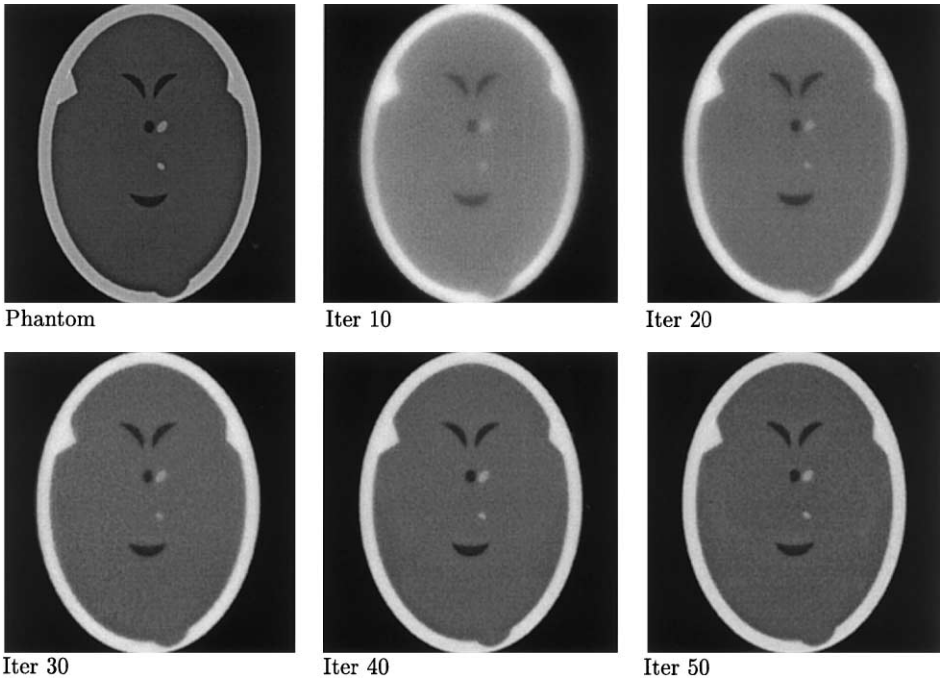

Fig. 24. Images produced by CAV for case 4 (345 × 345 pixels, 232,275 equations, 119,025 variables).

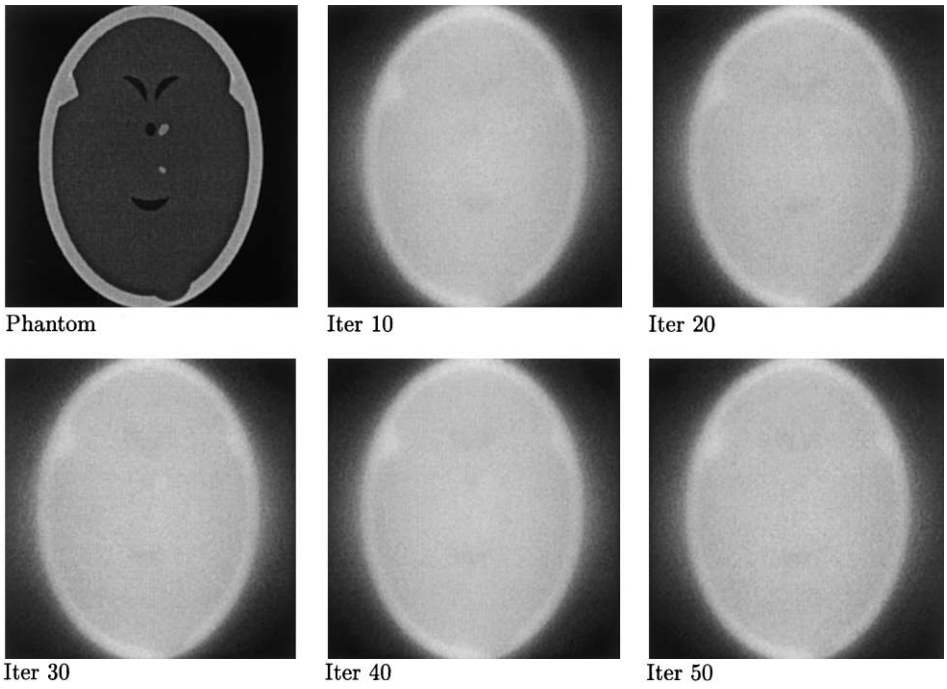

Fig. 25. Images produced by CIM for case 4 (345 × 345 pixels, 232,275 equations, 119,025 variables).

particular, our proof of convergence is a modified and revised adaptation of the first author's joint work with C. Byrne [10]. Thanks are also due to the anonymous referees whose detailed comments helped to improve this work. This research was initiated and supported by a research grant from the Israel Science Foundation, founded by the Israel Academy of Sciences and Humanities. The work of Y. Censor was also supported by NIH grant HL-28438 at the Medical Image Processing Group (MIPG), Department of Radiology, Hospital of the University of Pennsylvania, Philadelphia, PA, USA. The parallel computations were carried out on the supercomputers of the High Performance Computing Unit (HPCU) of the Israel Inter-University Computation Center (IUCC).

References

[1] R. Aharoni, Y. Censor, Block-iterative projection methods for parallel computation of solutions to convex feasibility problems, Linear Algebra Appl. 120 (1989) 165–175.
[2] M. Arioli, I. Duff, J. Noailles, D. Ruiz, A block projection method for sparse matrices, SIAM J. Sci. Stat. Comput. 13 (1992) 47–70.
[3] H.H. Bauschke, J.M. Borwein, On projection algorithms for solving convex feasibility problems, SIAM Rev. 38 (1996) 367–426.
[4] R.T. Behrens, L.L. Scharf, Signal processing applications of oblique projection operators, IEEE Trans. Signal Process. SP-42 (1994) 1413–1424.
[5] A. Ben-Israel, T.N.E. Greville, Generalized Inverses: Theory and Applications, Wiley, New York, 1974.
[6] D.P. Bertsekas, J.N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[7] Å. Björck, Numerical Methods for Least squares Problems, SIAM, Philadelphia, PA, 1996.
[8] L.M. Bregman, The relaxation method for finding the common point of convex sets and its application to the solution of problems in convex programming, USSR Comput. Math. Math. Phys. 7 (1967) 200–217.
[9] J.A. Browne, G.T. Herman, D. Odhner, SNARK93: a programming system for image reconstruction from projections, Technical Report No. MIPG198, The Medical Image Processing Group (MIPG). Department of Radiology, University of Pennsylvania, Philadelphia, PA, USA, August 1993.
[10] C. Byrne, Y. Censor, Proximity function minimization for separable, jointly convex Bregman distances, with applications, Technical Report, February 1998.
[11] C. Byrne, Y. Censor, Proximity function minimization using multiple Bregman projections, with applications to entropy optimization and Kullback–Leibler distance minimization, Technical Report, June 1999, revised: May 2000, Ann. Oper. Res. (to appear).
[12] Y. Censor, Finite series-expansion reconstruction methods, Proc. IEEE 71 (1983) 409–419.
[13] Y. Censor, P.P.B. Eggermont, D. Gordon, Strong underrelaxation in Kaczmarz's method for inconsistent systems, Numer. Math. 41 (1983) 83–92.
[14] Y. Censor, S.A. Zenios, Parallel Optimization: Theory, Algorithms and Applications, Oxford University Press, New York, 1997.
[15] G. Cimmino, Calcolo approssimato per soluzioni dei sistemi di equazioni lineari, La Ricerca Scientifica XVI, Series II, Anno IX 1 (1938) 326–333.
[16] P.L. Combettes, Inconsistent signal feasibility problems: least-squares solutions in a product space, IEEE Trans. Signal Process. SP-42 (1994) 2955–2966.
[17] M.A. Diniz-Ehrhardt, J.M. Martinez, A parallel projection method for overdetermined nonlinear systems of equations, Numer. Algo. 4 (1993) 241–262.
[18] P.P.B. Eggermont, G.T. Herman, A. Lent, Iterative algorithms for large partitioned linear systems, with applications to image reconstruction, Linear Algebra Appl. 40 (1981) 37–67.

[19] T. Elfving, Block-iterative methods for consistent and inconsistent linear equations, Numer. Math. 35 (1980) 1–12.
[20] N. Gastinel, Linear Numerical Analysis, Hermann, Paris, 1970.
[21] G.T. Herman, Image Reconstruction From Projections: The Fundamentals of Computerized Tomography, Academic Press, New York, 1980.
[22] G.T. Herman, L.B. Meyer, Algebraic reconstruction techniques can be made computationally efficient, IEEE Trans. Med. Imag. MI-12 (1993) 600–609.
[23] A.N. Iusem, A.R. De Pierro, Convergence results for an accelerated nonlinear Cimmino algorithm, Numer. Math. 49 (1986) 367–378.
[24] S. Kaczmarz, Angenäherte auflösung von systemen linearer gleichungen, Bulletin de l' Académie Polonaise des Sciences et Lettres A35 (1937) 355–357.
[25] S. Kayalar, H.L. Weinert, Oblique projections: formulas, algorithms, and error bounds, Math. Control Signals Syst. 2 (1989) 33–45.
[26] T. Kotzer, N. Cohen, J. Shamir, A projection-based algorithm for consistent and inconsistent constraints, SIAM J. Optim. 7 (1997) 527–546.
[27] E. Kreyszig, Introductory Functional Analysis with Applications, Wiley, New York, 1978.
[28] E.R. Lorch, On a calculus of operators in reflexive vector spaces, Trans. Am. Math. Soc. 45 (1939) 217–234.
[29] J.M. Martinez, R.J.B. De Sampaio, Parallel and sequential Kaczmarz methods for solving underdetermined nonlinear equations, J. Comput. Appl. Math. 15 (1986) 311–321.
[30] F.J. Murray, On complementary manifolds and projections in L_p and l_p, Trans. Am. Math. Soc. 41 (1937) 138–152.
[31] T.-S. Pan, A.E. Yagle, Acceleration of Landweber-type algorithms by suppression of projection on the maximum singular vector, IEEE Trans. Med. Imag. MI-11 (1992) 479–487.
[32] T.-S. Pan, A.E. Yagle, N.H. Clinthorne, W.L. Rogers, Acceleration and filtering in the generalized Landweber iteration using a variable shaping matrix, IEEE Trans. Med. Imag. MI-12 (1993) 278–286.
[33] G. Pierra, Decomposition through formalization in a product space, Math. Prog. 28 (1984) 96–115.
[34] K. Tanabe, Projection method for solving a singular system of linear equations and its applications, Numer. Math. 17 (1971) 203–214.
[35] H.J. Trussell, M.R. Civanlar, The Landweber iteration and projection onto convex sets, IEEE Trans. Acoust. Speech Signal Process. ASSP-33 (1985) 1632–1634.