ON DIAGONALLY-RELAXED ORTHOGONAL PROJECTION METHODS

YAIR CENSOR [†], TOMMY ELFVING [‡], GABOR T. HERMAN [§], AND TOURAJ NIKAZAD [¶]

September 2, 2005. Revised: May 29, 2007

Abstract. We propose and study a block-iterative projections method for solving linear equations and/or inequalities. The method allows diagonal component-wise relaxation in conjunction with orthogonal projections onto the individual hyperplanes of the system, and is thus called diagonally-relaxed orthogonal projections (DROP). Diagonal relaxation has proven useful in accelerating the initial convergence of simultaneous and block-iterative projection algorithms but until now it was available only in conjunction with generalized oblique projections in which there is a special relation between the weighting and the oblique projections. DROP has been used by practitioners and in this paper a contribution to its convergence theory is provided. The mathematical analysis is complemented by some experiments in image reconstruction from projections which illustrate the performance of DROP.

Key words. block-iterations, convex feasibility, diagonal relaxation, projection methods, simultaneous algorithms

AMS subject classifications. 15A06, 15A29, 15A39, 65F10, 65F50

1. Introduction.

1.1. Motivation and algorithmic structure. In this paper we investigate diagonal component-wise relaxation in simultaneous and in block-iterative projection methods for the linear feasibility problem. The method that we study, called diagonally-relaxed orthogonal projections (DROP), has been used by practitioners and our paper makes a contribution to its convergence theory. We also illustrate the performance of DROP as compared to similar methods on some examples, but these experiments, performed on a single processor, indicate that none of these methods is superior to all others, as long as the free parameters of the methods are carefully selected.

Our main contributions are convergence results (Theorem 2.3) for the fully-simultaneous DROP in the inconsistent case and (Theorems 2.9, 2.10) for the block-iterative DROP in the consistent case. The novel features of these results are the generality of the relaxation parameters intervals and the constructive explicit bounds for them. These are achieved by combining general results for block-iterative projection methods, such as those in Censor and Elfving [11] and Jiang and Wang [34], with a key lemma (Lemma 2.2) which generalizes a result of Lent that appeared in [11, Lemma 7.1].

Many problems in mathematics, in physical sciences and in real-world applications of various technological innovations can be modeled as a *convex feasibility problem*;

[†]Department of Mathematics, University of Haifa, Mt. Carmel, Haifa 31905, Israel (yair@math.haifa.ac.il).

[‡]Department of Mathematics, Linköping University, SE-581 83 Linköping, Sweden (toelf@mai.liu.se).

[§]Department of Computer Science, The Graduate Center, City University of New York, New York, NY 10016, USA (gabortherman@yahoo.com).

[¶]Department of Mathematics, Linköping University, SE-581 83 Linköping, Sweden (tonik@mai.liu.se) (on leave from the Department of Mathematics, Iran University of Science and Technology). His work is supported by a grant from the Iran Ministry of Science, Research & Technology.

i.e., a problem of finding a point $x^* \in Q := \cap_{i=1}^m Q_i$ in the intersection of finitely many closed convex sets $Q_i \subseteq R^n$ in a finite-dimensional Euclidean space. Projection algorithms play a central role in the area of constructive solution of such problems.

Although projections onto sets are used in many methods in optimization theory, not every method that uses projections really belongs to the class of projection algorithms. *Projection algorithms* are iterative algorithms that use projections onto sets while relying on the general principle that when a family of (usually closed and convex) sets is present, then projections onto the individual sets are easier to perform than projections onto other sets (intersections, image sets under some transformation, etc.) that are derived from the individual original sets. A projection algorithm reaches its goal that is related to the whole family of sets by performing projections onto the individual sets. Projection algorithms employ projections onto convex sets in various ways. They may use different kinds of projections and, sometimes, even use different projections within the same algorithm. They serve to solve a variety of problems that are either of the feasibility or the optimization types. They have different algorithmic structures (e.g., Butnariu, Censor and Reich [6], Censor, Elfving and Herman [10], Censor and Zenios [14, Section 1.3]) of which some are particularly suitable for parallel computing, and they demonstrate nice convergence properties and/or good initial behavior patterns.

This class of algorithms has witnessed great progress in recent years and its member algorithms have been applied with success to fully-discretized models of problems in image reconstruction from projections (e.g., Herman [30]), in image processing (e.g., Stark and Yang [37]), and in intensity-modulated radiation therapy (IMRT) (e.g., Censor [9]). Apart from theoretical interest, the main advantage of projection methods that makes them successful in real-world applications is computational. They commonly have the ability to handle huge-size problems of dimensions beyond which other, more sophisticated currently available, methods cease to be efficient. This is so because the building bricks of a projection algorithm are the projections onto the individual sets (that are assumed easy to perform) and the algorithmic structure is either sequential or simultaneous (or in-between).

If $Q \neq \emptyset$ the convex feasibility problem is called *consistent*, otherwise it is *inconsistent*. Our starting point is the well-known Cimmino algorithm for linear equations whose iterative step for equally-weighted linear equations has the form

$$x^{k+1} \;=\; x^k + \frac{\lambda_k}{m} \sum_{i=1}^{m} \frac{b_i - \langle a^i, x^k \rangle}{\|a^i\|_2^2} a^i, \tag{1.1}$$

where, here and throughout the paper, λ_k are relaxation parameters, and $a^i = (a^i_j)_{j=1}^n \in R^n$, $a^i \neq 0$, and $b_i \in R$ are the given data of the linear equations or inequalities. Fully-simultaneous (parallel) algorithmic schemes for general (not necessarily linear) convex sets employ a similar iterative step of the form

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

where $P_\Omega(x)$ stands for the *orthogonal (nearest Euclidean distance) projection* of a point x onto the closed convex set Ω, the parameters $\{w_i\}_{i=1}^m$ are a *system of weights* such that $w_i > 0$ for all $i = 1, 2, \ldots, m$ (sometimes $\sum_{i=1}^m w_i = 1$ is also required), and the *relaxation parameters* $\{\lambda_k\}_{k=0}^\infty$ are user-chosen and, in most convergence analyses, must remain in a certain fixed interval, in order to guarantee convergence.

This scheme (1.1) for linear equations was first proposed by Cimmino [15] (read about the profound impact of this paper on applied scientific computing in Benzi's paper [4]). Its generalization to general convex sets was done by Auslender [2].

The question of diagonal component-wise relaxation is: may the weights w_i be allowed to depend on the component index j as iterations proceed, without loosing the guaranteed convergence of the algorithm? Or, phrased in mathematical notation, may the iterations proceed according to

$$x_j^{k+1} = x_j^k + \lambda_k \left(\sum_{i=1}^{m} w_{ij} \left(\left(P_{Q_i}(x^k) \right)_j - x_j^k \right) \right), \text{ for } j = 1, 2, \ldots, n, \tag{1.3}$$

where the parameters $\{w_{ij}\}_{i=1}^m$ form n *systems of weights* such that, for $j = 1, 2, \ldots, n$, $w_{ij} \geq 0$ for all $i = 1, 2, \ldots, m$, and possibly also $\sum_{i=1}^m w_{ij} = 1$? If such component-wise relaxation is possible, then we can use it to exploit sparsity of the underlying problem and to control asynchronous (block) iterations, as will be explained in the sequel. To demonstrate the advantages of such diagonal relaxation let us consider linear equations and their associated hyperplanes

$$H_i = \left\{ x \in R^n \mid \langle a^i, x \rangle = b_i \right\}, \tag{1.4}$$

for $i = 1, 2, \ldots, m$. The orthogonal projection $P_i(z)$ of any point $z \in R^n$ onto H_i is

$$P_i(z) \;=\; z + \frac{b_i - \langle a^i, z \rangle}{\|a^i\|_2^2} a^i, \tag{1.5}$$

where $\| \cdot \|_2$ is the Euclidean norm. In a simultaneous projections method (1.2), see also Censor and Zenios [14, Algorithm 5.6.1], 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 hyperplanes H_i,

$$x^{k+1} \;=\; x^k + \frac{\lambda_k}{m} \sum_{i=1}^{m} \left(P_i(x^k) - x^k \right). \tag{1.6}$$

Equivalently, combining (1.6) and (1.5), we obtain the fully-simultaneous Cimmino algorithm with equal weights for the solution of linear equations.

ALGORITHM 1.1. ***Cimmino's Algorithm (CIM) with Equal Weights for Linear Equations***
Initialization: $x^0 \in R^n$ *is arbitrary.*
Iterative Step: *Given* x^k *compute, for* $j = 1, 2, \ldots, n$,

$$x_j^{k+1} \;=\; x_j^k + \frac{\lambda_k}{m} \sum_{i=1}^{m} \frac{b_i - \langle a^i, x^k \rangle}{\|a^i\|_2^2} a_j^i, \tag{1.7}$$

where $\{\lambda_k\}_{k=0}^{\infty}$ *are user-chosen relaxation parameters.*

When the $m \times n$ system matrix $A = (a_j^i)$ is sparse, as often happens in some important real-world applications, only a relatively small number of the elements $\{a_j^1, a_j^2, \ldots, a_j^m\}$ in the j-th column of A are nonzero, but in (1.7) the sum of their contributions is divided by the relatively large m – slowing down the progress of the algorithm. This observation led us in [12] to consider replacement of the factor $1/m$ in (1.7) by a factor that depends only on the number of *nonzero* elements in the set

$\{a_j^1, a_j^2, \ldots, a_j^m\}$. Specifically, for each $j = 1, 2, \ldots, n$, we denoted by s_j the number of nonzero elements in column j of the matrix A. Since we assume throughout this work that all columns of A are nonzero, we have that $s_j \neq 0$, for all j. Then we replace (1.7) by

$$x_j^{k+1} = x_j^k + \frac{\lambda_k}{s_j} \sum_{i=1}^{m} \frac{b_i - \langle a^i, x^k \rangle}{\|a^i\|_2^2} a_j^i, \text{ for } j = 1, 2, \ldots, n. \tag{1.8}$$

This iterative formula is the backbone of the proposed DROP algorithm. Certainly, if the matrix A is sparse, then the s_j values will be much smaller than m. Equation (1.8) can be rewritten and, at the same time generalized to a weighted case as in (1.2), in the form

$$x^{k+1} = x^k + \lambda_k \sum_{i=1}^{m} G_i \left(P_i(x^k) - x^k \right), \tag{1.9}$$

where

$$G_i = w_i S \quad \text{and} \quad S = \text{diag}(1/s_j). \tag{1.10}$$

All this leads us to consider the following algorithmic structure.

ALGORITHM 1.2. ***The Fully-Simultaneous Diagonally-Relaxed Orthogonal Projections (DROP) Method for Linear Equations***
Initialization: $x^0 \in R^n$ *is arbitrary.*
Iterative Step: *Given* x^k *compute, for* $j = 1, 2, \ldots, n$,

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

where $\{\lambda_k\}_{k=0}^{\infty}$ *are user-chosen relaxation parameters and* $w_i > 0$, *for all* $i = 1, 2, \ldots, m$, *are user-chosen weights.*

1.2. Relation to other works: History and terminology. Censor, Gordon and Gordon [12] proved convergence of a diagonally-relaxed simultaneous projections algorithmic scheme only in conjunction with certain *generalized oblique projections* that replaced the orthogonal projections, resulting in an iterative scheme for linear equations called the CAV (component averaging) method. In [13] BICAV was proposed as a block-iterative companion to CAV. (These algorithms are further discussed in Section 3 below.) In these methods the sparsity pattern of the matrix A is explicitly used when constructing the generalized oblique projections in the iteration formula. Using this scaling, considerable improvement was experimentally observed in [12] and [13] in problems of image reconstruction from projections. In [11] a complete convergence theory was given, including a generalization to linear inequalities of the CAV and BICAV schemes for equations; see also the work of Jiang and Wang [34].

The possibility of doing diagonal relaxation in conjunction with orthogonal projections (and not with oblique projections) was raised already in [12, Subsection 5.5] but was abandoned in view of the lack of any mathematical validation at that time. Specifically, Eq. (1.9) of [12] and Eq. (2.8) of [13] are identical with our Eq. (1.8) and with the iterative step of Algorithm 1.2 when all $w_i = 1$ and was experimented with in [12, Subsection 5.5]. It is also identical with the algorithm called CARP1 that is mentioned below. A block-iterative version of it was experimented with and reported in Figure 8 of [13].

We present DROP below as a block-iterative projections algorithm with diagonal relaxation that uses orthogonal projections (1.9), without resort to generalized oblique projections. This method was already used in applications by Herman, Matej and Carvalho [32] and by Sorzano et al. [36, see Eq. (3)], but without mathematical validation. Our present work makes a contribution here. Recently we learned that Gordon and Gordon [28] developed a new iterative method called component-averaged row projections (CARP). In one special case CARP and DROP coincide. We will discuss the details right after Algorithm 3.3 below. We wish to add here that in [7] we proposed and studied a simultaneous projections algorithm that employs *Bregman projections* and allows component-dependent weighting. There also has to be a specific relation between the component-dependent weights w_{ij} and the Bregman functions f_i according to which the projections are performed.

There is a terminological hurdle that should be observed here. When sparsity-pattern-induced diagonal relaxation (such as the use of s_j in our Algorithm 1.2) is applied in conjunction with sparsity-pattern-oriented generalized oblique projections, as is done in CAV and BICAV for linear equations [12, 13], the combined effect on the iteration formula amounts to orthogonal projections. This can be easily verified from the iterative formulas of CAV and BICAV, see also the explanation in [28, p. 1101]. The other side of the same phenomenon is that applying sparsity-pattern-induced diagonal relaxation in conjunction with orthogonal projections, as we do here in the fully-simultaneous or in the block-iterative versions of DROP, has the combined effect on the iterative formula of oblique (non-orthogonal) projections.

A variant of (1.2) called BIP, proposed by Aharoni and Censor [1], has the property that the weights are allowed to vary with the iteration index k (but not with the component index j).

ALGORITHM 1.3. ***The Block-Iterative Projections (BIP) Method for Linear Equations***
Initialization: $x^0 \in R^n$ *is arbitrary.*
Iterative Step: *Given* x^k *compute*

$$x^{k+1} = x^k + \lambda_k \left(\sum_{i=1}^{m} w_i^k \frac{b_i - \langle a^i, x^k \rangle}{\|a^i\|_2^2} a^i \right), \tag{1.12}$$

where $\{\lambda_k\}_{k=0}^{\infty}$ *are user-chosen relaxation parameters and* $\{w^k\}_{k=1}^{\infty}$ *is a user-chosen infinite sequence of weight vectors such that* $w^k = (w_i^k)_{i=1}^m$ *and, for every* $k \geq 0$, $w_i^k \geq 0$ *for all* $i = 1, 2, \ldots, m$.

An important feature of BIP is that some of the weights in every iteration are allowed to take the value zero. This enables creation of block-iterations (because when a $w_i^k = 0$ then the i-th constraint does not affect the k-th iteration) with variable block sizes, and varying assignments of constraints to blocks become permissible as long as a technical condition ($\sum_{k=0}^{\infty} w_i^k = +\infty$, for all $i = 1, 2, \ldots, m$, see [1]) is satisfied. BIP also encompasses as a special case fully-simultaneous CIM (Algorithm 1.1) by setting $w_i^k = 1/m$ for all $i = 1, 2, \ldots, m$ and all $k \geq 0$. The block-iterative methods of Elfving [22] and the block-iterative ART (Algebraic Reconstruction Technique) of Eggermont, Herman and Lent [21] were forerunners of the BIP methods for the convex feasibility problem. A related family of methods is studied [35, (5.49)] by Natterer and Wübbeling. None of these nor BIP allow for diagonal relaxation.

Another family of iterative methods for solving linear systems of equalities and/or inequalities are the *projected aggregation methods* (called PAM) of García-Palomares

[27] which were recently adapted to diagonally-scaled oblique projections by Echebest et al. [20].

2. DROP for linear equations and inequalities .

2.1. The fully-simultaneous DROP. We study first the fully-simultaneous algorithmic scheme of DROP for linear equations and prove its convergence regardless of the consistency of the system. With S as in (1.10) and

$$W = \text{diag}\left(w_i/\|a^i\|_2^2\right), \tag{2.1}$$

we rewrite Algorithm 1.2 in matrix-vector form.

ALGORITHM 2.1. ***The Fully-Simultaneous Diagonally-Relaxed Orthogonal Projections (DROP) Method for Linear Equations (in Matrix Form)***
Initialization: $x^0 \in R^n$ *is arbitrary.*
Iterative Step: *Given* x^k *compute*

$$x^{k+1} = x^k + \lambda_k S A^T W \left(b - Ax^k\right), \tag{2.2}$$

where $\{\lambda_k\}_{k=0}^{\infty}$ *are user-chosen relaxation parameters and* $w_i > 0$, *for all* $i = 1, 2, \ldots, m$, *are user-chosen weights.*

Here $b = (b_i) \in R^m$ and A^T (the transposed matrix of A) has a^i in its i-th column. Let $\rho(Q)$ denote the spectral radius (i.e., the largest, in absolute value, eigenvalue of a matrix Q). As will be seen later, the choice of relaxation parameters and weights is theoretically-limited by conditions in convergence theorems and is, in practice, chosen based on computational experience. A key result in our mathematical development is the following generalization of a lemma due to A. Lent that appeared in [11, Lemma 7.1].

LEMMA 2.2. *Assume that* $w_i > 0$, *for all* $i = 1, 2, \ldots, m$, *and that* $A \in R^{m\times n}$. *If* $W = diag(w_i/||a^i||_2^2) \in R^{m\times m}$ *and* $S = diag(1/s_j) \in R^{n\times n}$, *where* $s_j \neq 0$ *is the number of nonzero elements in the* j*-th column of* A, *then* $\rho(SA^TWA) \leq \max\{w_i \mid i = 1, 2, \ldots, m\}$.

Proof. Let (μ, v) be an eigenpair of SA^TWA, i.e., $SA^TWAv = \mu v$. Multiplying by A and putting $u = WAv$, we get

$$ASA^Tu = \mu W^{-1}u. \tag{2.3}$$

Hence,

$$\mu = \frac{u^TASA^Tu}{u^TW^{-1}u}. \tag{2.4}$$

Now,

$$\begin{aligned} u^TASA^Tu = ||S^{1/2}A^Tu||_2^2 &= \sum_{j=1}^{n}\left(\sum_{i=1}^{m}\left(1/\sqrt{s_j}\right)a_j^iu_i\right)^2 \\ &= \sum_{j=1}^{n}\left(1/s_j\right)\left(\sum_{i=1}^{m}a_j^iu_i\right)^2, \end{aligned} \tag{2.5}$$

which implies

$$\mu = \frac{\sum_{j=1}^{n}\left(1/s_j\right)\left(\sum_{i=1}^{m}a_j^iu_i\right)^2}{\sum_{i=1}^{m}u_i^2\left(||a^i||_2^2/w_i\right)}. \tag{2.6}$$

Cauchy's inequality guarantees that

$$\left(\sum_{i=1}^{m} a_j^i u_i\right)^2 \leq s_j \sum_{i=1}^{m} u_i^2 \left(a_j^i\right)^2. \tag{2.7}$$

Applying (2.7) to the numerator of (2.6), and changing the order of summation yields

$$\mu \leq \frac{\sum_{i=1}^{m} u_i^2 ||a^i||_2^2}{\sum_{i=1}^{m} (1/w_i)\, u_i^2 ||a^i||_2^2} \leq \max\{w_i \mid i = 1, 2, \ldots, m\}, \tag{2.8}$$

which completes the proof. ∎

Denoting by $||z||_W^2 = \langle z, Wz\rangle$ the W-norm, the following convergence results holds (which is our first main contribution).

THEOREM 2.3. *Assume that $w_i > 0$, for all $i = 1, 2, \ldots, m$. If, for all $k \geq 0$,*

$$0 < \epsilon \leq \lambda_k \leq (2 - \epsilon) \,/\, \max\{w_i \mid i = 1, 2, \ldots, m\}, \tag{2.9}$$

where ϵ is an arbitrarily small but fixed constant, then any sequence $\{x^k\}_{k=0}^{\infty}$, generated by Algorithm 2.1, converges to a weighted least squares solution $x^* = \arg\min\{||Ax - b||_W \mid x \in R^n\}$. If, in addition, $x^0 \in \mathcal{R}(SA^T)$, the range of SA^T, then x^* has minimal S^{-1}-norm.

Proof. Using the transformations

$$y^k = S^{-1/2} x^k, \quad \text{and } \bar{A} = AS^{1/2}, \tag{2.10}$$

the iterative step (2.2) becomes

$$y^{k+1} \;=\; y^k + \lambda_k \bar{A}^T W (b - \bar{A} y^k). \tag{2.11}$$

Assuming that

$$0 < \epsilon \leq \lambda_k \leq (2 - \epsilon)/\rho(\bar{A}^T W \bar{A}), \tag{2.12}$$

we apply [11, Theorem 6.1] to conclude that

$$\lim_{k\to\infty} y^k = y^* \;\text{ and }\; y^* = \operatorname{argmin}\{\, ||\bar{A}y - b||_W \;\; \mid y \in R^m\}, \tag{2.13}$$

which implies that

$$\lim_{k\to\infty} x^k = S^{1/2} y^* = x^* \;\text{ and }\; x^* = \operatorname{argmin}\{\, ||Ax - b||_W \;\; \mid x \in R^n\}. \tag{2.14}$$

Also, if $y^0 \in \mathcal{R}(\bar{A}^T)$ then y^* has minimal Euclidean norm. Hence, by using

$$||y^*||_2 = ||S^{-1/2} S^{1/2} y^*||_2 = ||x^*||_{S^{-1}}, \tag{2.15}$$

it follows that x^* has minimal S^{-1}-norm provided that $x^0 = S^{1/2} y^0 \in \mathcal{R}(SA^T)$. Finally,

$$\begin{aligned} \rho(\bar{A}^T W \bar{A}) &= \rho(S^{1/2} A^T W A S^{1/2}) = \rho(S^{1/2}(S^{1/2} A^T W A S^{1/2}) S^{-1/2}) \\ &= \rho(SA^T W A) \end{aligned} \tag{2.16}$$

and the result now follows from Lemma 2.2. ∎

This theorem allows us to replace the spectral radius $\rho(SA^T WA)$ by $\max\{w_i \mid i = 1, 2, \ldots, m\}$ in the sufficient condition (2.12) on the relaxation parameters. This maximal weight is user-provided and using it saves the need to calculate the spectral radius. However, for fast initial performance it may well be the case that the relaxation parameters ought to be selected in violation of the simpler, but more restrictive, condition (2.9). We will return to this point when discussing our experiments below.

2.2. The block-iterative DROP. We now move from the fully-simultaneous DROP to its "block-iterative" generalization. This algorithmic model of block iterations is a special case of *asynchronous iterations*, see, e.g., Frommer and Szyld [26] and Elsner, Koltracht and Neumann [23]. Those were called in early days *chaotic relaxation* by Chazan and Miranker [16]. In recent literature in image reconstruction from projections the term "ordered subsets" is used for "block-iterative", see, e.g., Hudson and Larkin [33]. The basic idea of a block-iterative algorithm is to partition the A and b of the system $Ax = b$ or $Ax \leq b$ into "blocks" of equations and/or inequalities and treat each block according to the rule used in the simultaneous algorithm for the whole system, passing, e.g., cyclically over all the blocks. Throughout the following, T will be the number of blocks (T is also used for matrix transposition but the meaning will always be clear from the way it is used). For $t = 1, 2, \ldots, T$, the block $B_t \subseteq \{1, 2, \ldots, m\}$ will be a subset of indices of the form $B_t = \{i_1^t, i_2^t, \ldots, i_{m(t)}^t\}$, where $m(t)$ is the number of elements in B_t. There is nothing to prevent different blocks from containing common indices; we will require, however, the following:

CONDITION 2.4. *Every element of* $\{1, 2, \ldots, m\}$ *appears in at least one of the sets* $B_t, \quad t = 1, 2, \ldots, T$.

A sequence $\{t(k)\}_{k=0}^{\infty}$ such that $1 \leq t(k) \leq T$, called a *control sequence,* governs which block is taken up at the k-th iteration of the algorithm. A common control is the cyclic control, $t(k) = k \bmod T + 1$. For other possibilities see, e.g., [14, Definition 5.1.1].

The block-iteration corresponding to the method (1.9) is

$$x^{k+1} \;=\; x^k + \lambda_k \sum_{i \in B_{t(k)}} G_i \left(P_i(x^k) - x^k \right). \tag{2.17}$$

This can be reformulated for the case of linear inequalities in matrix-vector notation as follows. For $t = 1, 2, \ldots, T$, let A_t denote the matrix formed by all rows of A whose indices belong to the block of indices B_t (and correspondingly for the right-hand side vector b), i.e.,

$$A_t := \begin{bmatrix} \left(a^{i_1^t}\right)^T \\ \left(a^{i_2^t}\right)^T \\ \vdots \\ \left(a^{i_{m(t)}^t}\right)^T \end{bmatrix}, \quad b^t := \begin{bmatrix} b_{i_1^t} \\ b_{i_2^t} \\ \vdots \\ b_{i_{m(t)}^t} \end{bmatrix}, \quad t = 1, 2, \ldots, T. \tag{2.18}$$

The classical partitioning with fixed non-overlapping blocks of equal sizes [21] is obtained by taking in (2.18) $m(t) = \ell, \; t = 1, 2, \ldots, T$ with $\ell \times T = m$. We first consider a system of linear inequalities

$$Ax \leq b, \tag{2.19}$$

which we will assume to be consistent, i.e., $\{x \in R^n \mid Ax \leq b\} \neq \emptyset$. For each $i = 1, 2, \ldots, m$, the closed half-space

$$L_i = \{x \in R^n \mid \langle a^i, x \rangle \leq b_i\} \tag{2.20}$$

has (1.4) as its bounding hyperplane. Define $L := \cap_{i=1}^m L_i$ and note that L is a closed convex set in R^n. The task of finding a member of L, i.e., a solution of (2.19), is

called the *linear feasibility problem*, which is a special case of the *convex feasibility problem*; see, e.g., the review papers of Bauschke and Borwein [3], Combettes [17] or [14, Chapter 5]. It is well-known, and is easy to verify, that the orthogonal projection $P_{L_i}(z)$ of a point $z \in R^n$ onto L_i is

$$P_{L_i}(z) = z + c_i(z)a^i, \quad \text{where} \quad c_i(z) = \min\left\{0, \frac{b_i - \langle a^i, z\rangle}{\|a^i\|_2^2}\right\}. \tag{2.21}$$

Note that, if $z \notin L_i$ then $c_i(z) < 0$, otherwise $c_i(z) = 0$. Further, define

$$I_t(z) := \{\ i \mid i \in \{i_1^t, i_2^t, \ldots, i_{m(t)}^t\} \text{ and } c_i(z) < 0\ \} \tag{2.22}$$

as the set of indices of the half-spaces in the t-th block that are violated by z. We also introduce $m(t) \times m(t)$ diagonal matrices $\{D_t\}_{t=1}^T$, corresponding to the blocks $\{A_t\}_{t=1}^T$, namely,

$$(D_t(z))_{qq} = \begin{cases} 1, & \text{if } q \in I_t(z), \\ 0, & \text{otherwise.} \end{cases} \tag{2.23}$$

Let $\{W_t\}_{t=1}^T$ be a finite set of some given positive definite diagonal matrices, and define

$$V_t(z) := D_t(z)W_tD_t(z), \quad \text{for all} \quad t = 1, 2, \ldots, T. \tag{2.24}$$

To define DROP in its general formulation let $\{U_t\}_{t=1}^T$ be some given collection of symmetric and positive definite matrices. Different choices of this collection may give rise to different realizations of the general DROP scheme. The specific choices that we used in our computations are given in Algorithms 3.4 and 3.5 below. Now the formulation of DROP for linear inequalities is as follows when written in matrix form.

ALGORITHM 2.5. ***DROP for Linear Inequalities (in Matrix Form)***
Initialization: $x^0 \in R^n$ *is arbitrary.*
Iterative Step: *Given* x^k *compute*

$$x^{k+1} \;=\; x^k + \lambda_k U_{t(k)} A_{t(k)}^T V_{t(k)}(x^k)(b^{t(k)} - A_{t(k)}x^k), \tag{2.25}$$

where $\{\lambda_k\}_{k=0}^\infty$ *are user-chosen relaxation parameters and* $\{t(k)\}_{k=0}^\infty$ *is a control sequence.*

Observe that, as an algorithm name, we designate the acronym DROP to this block-iterative formulation with any such collection of matrices $\{U_t\}_{t=1}^T$. This will make the fully-simultaneous case just a special case of Algorithm 2.5.

The corresponding block-iterative algorithmic scheme for linear equations

$$Ax = b \tag{2.26}$$

is formulated as follows. Let $\{W_t\}_{t=1}^T$ be a finite set of some given positive definite matrices (note that we now, for equalities, allow not only diagonal weight matrices).

ALGORITHM 2.6. ***DROP for Linear Equations (in Matrix Form)***
Initialization: $x^0 \in R^n$ *is arbitrary.*
Iterative Step: *Given* x^k *compute*

$$x^{k+1} \;=\; x^k + \lambda_k U_{t(k)} A_{t(k)}^T W_{t(k)}(b^{t(k)} - A_{t(k)}x^k), \tag{2.27}$$

where $\{\lambda_k\}_{k=0}^\infty$ *are user-chosen relaxation parameters and* $\{t(k)\}_{k=0}^\infty$ *is a control sequence.*

Algorithm 2.1 is a special case of Algorithm 2.6, obtained by choosing $T = 1$, $U_1 = S$ and $W_1 = W$.

Although the formulation of Algorithm 2.5 and 2.6 allows a family of matrices $\{U_t\}_{t=1}^T$ our current convergence results only include the case $U_t = U$, for $t = 1, 2, \cdots, T$, for some fixed matrix U.

Our first result is for systems of linear inequalities.

THEOREM 2.7. *Let U be a given symmetric and positive definite matrix and let $\{W_t\}_{t=1}^T$ be a given collection of positive definite diagonal matrices and define the matrices*

$$\Phi_k := UA_{t(k)}^T V_{t(k)}(x^k) A_{t(k)}, \tag{2.28}$$

where $V_t(z)$ is defined by (2.23) and (2.24). Assume that $L := \cap_{i=1}^m L_i \neq \emptyset$ and that the relaxation parameters are restricted, for all $k \geq 0$, to

$$0 < \epsilon \leq \lambda_k \leq (2 - \epsilon)/\rho(\Phi_k), \tag{2.29}$$

where ϵ is an arbitrarily small but fixed constant. If $\{t(k)\}_{k=0}^\infty$ is the cyclic control, then any sequence $\{x^k\}_{k=0}^\infty$, generated by Algorithm 2.5, converges to a solution of the system (2.19), provided that Condition 2.4 holds.

Proof. We simplify notation by writing $b^k = b^{t(k)}$, $A_k = A_{t(k)}$ and $V_k = V_{t(k)}(x^k)$, and multiply (2.25) by $U^{-1/2}$ (the inverse of the square root of U) to get

$$u^{k+1} = u^k + \lambda_k U^{1/2} A_k^T V_k (b^k - A_k U^{1/2} u^k), \tag{2.30}$$

where $u^k = U^{-1/2} x^k$. First note that the solution set of $AU^{1/2}u \leq b$ is feasible, by the assumption $L \neq \emptyset$. It follows by [11, Theorem 4.1] (here we need Condition 2.4) that $\lim_{k\to\infty} u^k = u^*$ such that $AU^{1/2}u^* \leq b$, provided that

$$0 < \epsilon \leq \lambda_k \leq (2 - \epsilon)/\rho(\Phi_k^U), \tag{2.31}$$

where $\Phi_k^U := U^{1/2} A_k^T V_k A_k U^{1/2}$. But

$$\Phi_k = UA_k^T V_k A_k = U^{1/2} \Phi_k^U U^{-1/2}, \tag{2.32}$$

so that $\rho(\Phi_k) = \rho(\Phi_k^U)$ follows. ∎

We also give the corresponding convergence result for linear equations.

THEOREM 2.8. *Let U be a given symmetric and positive definite matrix, let $\{W_t\}_{t=1}^T$ be given positive definite diagonal matrices and define the matrices*

$$\Psi_k := UA_{t(k)}^T W_{t(k)} A_{t(k)}. \tag{2.33}$$

Assume that $H := \cap_{i=1}^m H_i \neq \emptyset$ and that the relaxation parameters are restricted, for all $k \geq 0$, to

$$0 < \epsilon \leq \lambda_k \leq (2 - \epsilon)/\rho(\Psi_k), \tag{2.34}$$

where ϵ is an arbitrarily small but fixed constant. If $\{t(k)\}_{k=0}^\infty$ is a cyclic control then any sequence $\{x^k\}_{k=0}^\infty$, generated by Algorithm 2.6, converges to a solution of the system (2.26), provided that Condition 2.4 holds. If, in addition $x^0 \in \mathcal{R}(UA^T)$, then the solution has minimal U^{-1}-norm.

Proof. The proof is obtained by using a similar technique to that used in the proof of Theorem 2.3 and relying on [11, Theorems 4.2, 7.6]. ∎

Theorem 2.7 gives rise to the next theorem. We consider, and refer to, the block-iterative structure described just before Condition 2.4 and in (2.18). Let, for all $t = 1, 2, \ldots, T$, and all $q = 1, 2, \ldots, m(t)$, w_q^t denote a positive real number. In analogy with (2.1), we define, for all $t = 1, 2, \ldots, T$, the matrices

$$W_t = \mathrm{diag}(\mu_q^t) \in R^{m(t)\times m(t)}, \tag{2.35}$$

where

$$\mu_q^t = \frac{w_q^t}{\left\| a^{i_q^t} \right\|_2^2}, \quad q = 1, 2, \ldots, m(t). \tag{2.36}$$

THEOREM 2.9. *Let W_t be defined as in (2.35)–(2.36). Let $U = diag(1/\tau_j)$ where $\tau_j \geq \max\{s_j^t \mid t = 1, 2, \ldots, T\}$ with s_j^t being the number of nonzero elements in column j of block A_t. Assume that $L := \cap_{i=1}^m L_i \neq \emptyset$ and that the relaxation parameters are restricted, for all $k \geq 0$, to*

$$0 < \epsilon \leq \lambda_k \leq (2 - \epsilon)/\max\{w_q^{t(k)} \mid q = 1, 2, \ldots, m(t(k))\}, \tag{2.37}$$

where ϵ is an arbitrarily small but fixed constant. If $\{t(k)\}_{k=0}^\infty$ is a cyclic control, then any sequence $\{x^k\}_{k=0}^\infty$, generated by Algorithm 2.5, converges to a solution of the system (2.19), provided that Condition 2.4 holds.

Proof. To prove this theorem we need, by Theorem 2.7, to show that the upper bound in (2.29) can be replaced by that of (2.37). To see this we first observe that the matrices Φ_k^U, defined in the proof of Theorem 2.7 by $\Phi_k^U = U^{1/2} A_k^T V_k A_k U^{1/2}$, can be rewritten as

$$\Phi_k^U = \sum_{q \in I_{t(k)}(x^k)} \mu_q^{t(k)} (U^{1/2} a^{i_q^{t(k)}})(U^{1/2} a^{i_q^{t(k)}})^T, \tag{2.38}$$

where $I_{t(k)}(x^k)$ is defined by (2.22). Hence,

$$\begin{aligned} \rho(\Phi_k^U) &\leq \rho\left(\sum_{q=1}^{m(t(k))} \mu_q^{t(k)} (U^{1/2} a^{i_q^{t(k)}})(U^{1/2} a^{i_q^{t(k)}})^T \right) \\ &= \rho\left(U^{-1/2} \Psi_k U^{1/2} \right) = \rho(\Psi_k), \end{aligned} \tag{2.39}$$

where Ψ_k is defined in (2.33) and $W_{t(k)}$ is given in (2.35)–(2.36). Next we wish to use Lemma 2.2 to estimate $\rho(\Psi_k)$. To do so we identify $A_{t(k)}$ and $W_{t(k)}$ of (2.33) with A and with W in Lemma 2.2, respectively, but U of (2.33) is not identifiable with S of Lemma 2.2. However, we note that Lemma 2.2 remains true if instead of s_j we use τ_j in the definition of S. Nothing would change in the proof of Lemma 2.2 except that (2.7) would remain true also with τ_j because $\tau_j \geq s_j^{t(k)}$. The proof of the theorem is then complete by applying the modified version of Lemma 2.2. ∎

For linear equations we have now the next theorem which is our second main contribution.

THEOREM 2.10. *Let W_t be defined as in (2.35)–(2.36). Let $U = diag(1/\tau_j)$ where $\tau_j \geq \max\{s_j^t \mid t = 1, 2, \ldots, T\}$ with s_j^t being the number of nonzero elements in column j of block A_t. Assume that $H := \cap_{i=1}^m H_i \neq \emptyset$ and that the relaxation parameters are restricted, for all $k \geq 0$, to*

$$0 < \epsilon \leq \lambda_k \leq (2 - \epsilon)/\max\{w_q^{t(k)} \mid q = 1, 2, \ldots, m(t(k))\}, \tag{2.40}$$

where ϵ is an arbitrarily small but fixed constant. If $\{t(k)\}_{k=0}^{\infty}$ is a cyclic control, then any sequence $\{x^k\}_{k=0}^{\infty}$, generated by Algorithm 2.6, converges to a solution of the system (2.26), provided that Condition 2.4 holds. If, additionally, $x^0 \in \mathcal{R}(UA^T)$ then the limit vector has minimal U^{-1}-norm.

Proof. The last sentence in the statement of Theorem 2.10 is proved similar to the corresponding result in Theorem 2.3. The rest of the proof is similar almost verbatim to the proof of Theorem 2.9. □

We end this section with additional comments on the inconsistent case for linear equations. Jiang and Wang [34] recently extended our convergence results of [11] for Algorithm 2.6. In both these papers a constant U-matrix is used. By assuming for the parameters $\{\lambda_k\}_{k=0}^{\infty}$ that $\lim_{k\to\infty} \lambda_k = 0$ and $\sum_{k=0}^{\infty} \lambda_k = +\infty$, instead of what we have assumed, convergence towards a weighted least squares problem is proved in [34, Theorem II.1(2)]. Another possible way of coping with inconsistency in Algorithm 2.6 is to replace (2.26) by the normal equations. Consider, e.g., a regularized weighted least squares problem (here W_2 is a positive semidefinite and symmetric matrix and M is a positive definite and symmetric matrix)

$$\min\left\{||Ax - b||_M^2 + ||x - x^0||_{W_2}^2 \mid x \in R^n\right\}. \tag{2.41}$$

The normal equations corresponding to (2.41) are

$$(A^T M A + W_2)x = A^T M b + W_2 x^0. \tag{2.42}$$

This is a consistent system of equations, and its solution x solves (2.41) and is unique if W_2 is positive definite (or, of course, if A has full-rank). By introducing the residual vector

$$r = M^{1/2}(b - Ax) \tag{2.43}$$

and noting that

$$A^T M^{1/2} r = A^T M(b - Ax) = W_2(x - x^0), \tag{2.44}$$

the normal equations (2.42) can be written as two sets of equations

$$M^{1/2}Ax + r = M^{1/2}b, \quad W_2 x - A^T M^{1/2} r = W_2 x^0. \tag{2.45}$$

At least two options are now available. Either apply Algorithm 2.6 directly to the normal equations (2.42) or, to avoid the complexity associated with forming $A^T MA$, use the equations (2.45) at the expense of iterating in both x and r, as was done by Eggermont, Herman and Lent [21, Section 2.2].

3. The algorithms that are used in our experiments. In this section we give precise formulations of the algorithms that we are using in our experiments. As said at the beginning of Section 1, we are interested in projection methods because of their fundamental ability to handle huge-size problems of dimensions beyond which other, more sophisticated currently available, methods cease to be efficient. This is so because the building bricks of a projection algorithm are the projections onto the individual sets (that are assumed easy to perform) and the algorithmic structure which is either sequential or simultaneous (or in-between). Therefore, we compare our DROP algorithm with various sequential, simultaneous and block-iterative variants of commonly-used projection methods that are employed in the field of image reconstruction from projections.

We use the BIP method (Algorithm 1.3) in its block-iterative version. BIP also encompasses as a special case the algorithm CIM (Algorithm 1.1) that we use in our experiments. This occurs when in BIP one chooses $w_i^k = 1/m$ for all $i = 1, 2, \ldots, m$ and all $k \geq 0$. Another special case of BIP that we use in our experiments is obtained by choosing, for $i = 1, 2, \ldots, m$ and all $k \geq 0$,

$$w_i^k := \begin{cases} 1, & \text{if } i = i(k), \\ 0, & \text{otherwise,} \end{cases} \tag{3.1}$$

where $\{i(k)\}_{k=0}^{\infty}$ is a control sequence over the set $\{1, 2, \ldots, m\}$. In this case we get, for linear equations, the following fully-sequential algorithm.

ALGORITHM 3.1. ***Algebraic Reconstruction Technique (ART) for Linear Equations***
Initialization: $x^0 \in R^n$ *is arbitrary.*
Iterative Step: *Given* x^k *compute*

$$x^{k+1} = x^k + \lambda_k \frac{b_{i(k)} - \langle a^{i(k)}, x^k \rangle}{\left\| a^{i(k)} \right\|_2^2} a^{i(k)}, \tag{3.2}$$

where $\{\lambda_k\}_{k=0}^{\infty}$ *are user-chosen relaxation parameters and* $\{i(k)\}_{k=0}^{\infty}$ *is a control sequence that governs the order by which individual equations are employed as the iterations proceed.*

This algorithm has a long history and rich literature which we are not going to present here; see, e.g., [14, 30, 31]. In the block-iterative version of BIP that we used in our experiments we defined, for $i = 1, 2, \ldots, m$, and all $k \geq 0$,

$$w_i^k := \begin{cases} \dfrac{1}{m(t(k))}, & \text{if } i \in B_{t(k)}, \\ 0, & \text{otherwise,} \end{cases} \tag{3.3}$$

where $\{t(k)\}_{k=0}^{\infty}$ is a block control sequence over the set $\{1, 2, \ldots, T\}$.

ALGORITHM 3.2. ***The Block-Iterative Component Averaging (BICAV) Algorithm***
Initialization: $x^0 \in R^n$ *is arbitrary.*
Iterative Step: Given x^k*, compute* x^{k+1} *by using, for* $j = 1, 2, \ldots, n$*, the formula:*

$$x_j^{k+1} = x_j^k + \lambda_k \sum_{i \in B_{t(k)}} \frac{b_i - \langle a^i, x^k \rangle}{\sum_{\ell=1}^n s_\ell^{t(k)} (a_\ell^i)^2} a_j^i, \tag{3.4}$$

where λ_k *are user-chosen relaxation parameters, the number of nonzero elements* $a_j^i \neq 0$ *in the* j*-th column of the block* A_t *of the matrix* A *is* s_j^t*, and* $\{t(k)\}_{k=0}^{\infty}$ *is a control*

sequence that governs the order by which the blocks are employed as the iterations proceed.

For details about this algorithm consult [11, 13]. The special case when there is only one block ($T = 1$) has been referred to in the literature as CAV [12, Algorithm 3.2].

Next we describe the CARP algorithm of Gordon and Gordon [28, Algorithm 2]. To do so we need the following definitions. Construct a finite number of blocks, from the row indices of the matrix A, and denote them by $\{B_t\}_{t=1}^T$ where $B_t = \{i_1^t, i_2^t, \ldots, i_{m(t)}^t\}$ and $m(t)$ is the number of elements in B_t. Denote this family of blocks by $B = \{B_t\}_{t=1}^T$. For each $j = 1, 2, \ldots, n$, define the index set

$$I_j(B) = \{t \mid 1 \leq t \leq T,\ a_j^{i_q^t} \neq 0 \ \text{ for some } \ 1 \leq q \leq m(t)\}. \tag{3.5}$$

Given a set of T vectors $\{x^t\}_{t=1}^T$ in R^n, define the operator $CA_B(\{x^1, x^2, \ldots, x^T\})$ whose j-th component is

$$\left(CA_B(\{x^1, x^2, \ldots, x^T\})\right)_j := (1/s_j) \sum_{t \in I_j(B)} x_j^t, \tag{3.6}$$

where $s_j := |I_j(B)|$ is the number of elements in $I_j(B)$. The second definition describes the outcome of applying orthogonal projections sequentially along the equations of some block B_t, namely, given an index t and a point $x \in R^n$, define $KSWP(t, x)$ to be the vector $\overline{x}^t := x^{m(t)}$ obtained by running the loop:

$$x^0 = x, \text{ and } x^{q+1} = x^q + \lambda_q \frac{b_{i_q^t} - \langle a^{i_q^t}, x^q \rangle}{\left\| a^{i_q^t} \right\|_2^2} a^{i_q^t}, \text{ for } \ 0 \leq q < m(t), \tag{3.7}$$

where λ_q are relaxation parameters. With these definitions at hand, the CARP algorithm that we used is as follows.

ALGORITHM 3.3. ***The Component-Averaged Row Projection (CARP) Algorithm***
Initialization: $x^0 \in R^n$ is arbitrary.
Iterative Step: Given x^k,

$$x^{k+1} = CA_B(\{\overline{x}^1, \overline{x}^2, \ldots, \overline{x}^T\}), \tag{3.8}$$

where, for $t = 1, 2, \ldots, T$,

$$\overline{x}^t = (KSWP)^{p_k}(t, x^k), \tag{3.9}$$

and p_k is the number of times that $KSWP$ is applied consecutively to the block B_t.

In all our experiments with CARP we used $p_k = 1$, for all $k \geq 0$. We remark that Algorithm 2.1, with the choice $w_i = 1$, for all $i = 1, 2, \ldots, m$, is identical to the special case of CARP (called CARP1 in [28]) in which the whole system is partitioned into m single element blocks that contain one equation each. The method of proof in [28] is quite different from ours in Section 2 and is based on a clever way of expressing CARP in a product space. In this space the method BIP (1.12) is applied and, for the consistent case, CARP is a special case of BIP. In the inconsistent case one cannot have convergence of CARP, in [28] the authors prove cyclic convergence relying on

Eggermont, Herman and Lent [21, Theorem 3.1]. In contrast, we have here a complete convergence (not just cyclic) result (Theorem 2.3) for the fully-simultaneous DROP in the inconsistent case. Note also that when for CARP (Algorithm 3.3) one puts all constraints into a single block, then CARP becomes identical with ART (Algorithm 3.1).

We used in our computational experiments two special cases of DROP for linear equations (Algorithm 2.6). The first is with $U_t = U,\ t = 1, 2, \cdots, T$ with a specific choice of U. The second is with a specific choice of $\{U_t\}_{t=1}^{T}$.

ALGORITHM 3.4. ***DROP1***
Initialization: $x^0 \in R^n$ *is arbitrary.*
Iterative Step: *Given* x^k *compute*

$$x^{k+1} = x^k + \lambda_k U \sum_{q=1}^{m(t(k))} \mu_q^{t(k)} \left(b_{i_q^{t(k)}} - \langle a^{i_q^{t(k)}}, x^k \rangle \right) a^{i_q^{t(k)}}. \tag{3.10}$$

where $\{\lambda_k\}_{k=0}^{\infty}$ *are user-chosen relaxation parameters,* $\{t(k)\}_{k=0}^{\infty}$ *is a control sequence that governs the order according to which the blocks are employed as the iterations proceed,* $\mu_q^{t(k)}$ *are determined according to (2.36) with weights* $w_q^{t(k)} = 1$ *for all* $q = 1, 2, \ldots, m(t(k))$, *and* $U = diag(1/\tau_j)$ *where* $\tau_j = \max\{s_j^t \mid t = 1, 2, \ldots, T\}$ *with* s_j^t *being the number of nonzero elements in column* j *of block* A_t.

ALGORITHM 3.5. ***DROP2***
Initialization: $x^0 \in R^n$ *is arbitrary.*
Iterative Step: *Given* x^k *compute*

$$x^{k+1} = x^k + \lambda_k U_{t(k)} \sum_{q=1}^{m(t(k))} \mu_q^{t(k)} \left(b_{i_q^{t(k)}} - \langle a^{i_q^{t(k)}}, x^k \rangle \right) a^{i_q^{t(k)}}. \tag{3.11}$$

where $\{\lambda_k\}_{k=0}^{\infty}$ *are user-chosen relaxation parameters,* $\{t(k)\}_{k=0}^{\infty}$ *is a control sequence that governs the order according to which the blocks are employed as the iterations proceed,* $\mu_q^{t(k)}$ *are determined according to (2.36) with weights* $w_q^{t(k)} = 1$ *for all* $q = 1, 2, \ldots, m(t(k))$, *and* $U_{t(k)} = \mathrm{diag}(\min(1, 1/s_j^{t(k)}))$ *with* s_j^t *being the number of nonzero elements in column* j *of block* A_t.

4. Computational results. Our experimental work with the DROP algorithm is intended to assess the computational aspects of using it in the field of image reconstruction from projections [30]. We work with a medical test phantom and with a reconstruction problem in electron microscopy. In both of our examples a planar cross-section of the object is considered and the distribution of some physical parameter (the X-ray attenuation in medicine and the Coulomb potential in electron microscopy) in the cross-section has to be reconstructed from estimates of its line integrals for a finite number m of lines in the cross-section; we use b_i to represent the estimated line integral for the i-th line. The unknown function of two variables has real values and is called the *picture*.

A fundamental model for solving this task is provided by the *series-expansion* approach (e.g., Herman [30] or Censor [8]), which we formulate here 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 reconstructed. The pixels are numbered in some agreed manner, say from 1 (top left corner pixel) to n (bottom right corner pixel), see Figure 5.1.

The function to be reconstructed is approximated by one that takes a constant uniform value x_j throughout the j-th pixel, for $j = 1, 2, \ldots, n$. The value of x_j should closely approximate $\hat{x}_j$, which is defined as the average value of the function within the j-th pixel. Thus, the vector $\hat{x} = (\hat{x}_j)_{j=1}^n$ is the discretized version of the "true" function that is being reconstructed. We denote the length of intersection of the i-th line with the j-th pixel by a_j^i, for all $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$. Therefore, in this model, each line integral is approximated by a finite sum, and the task is represented by a system of linear equations

$$\sum_{j=1}^{n} a_j^i x_j = b_i\,, \quad i = 1, 2, \ldots, m\,. \tag{4.1}$$

For both our examples we simulated the parallel mode of data collection. In this mode, the set of all lines for which line integrals are estimated is divided into K sets of m/K lines in each. The lines within a set are parallel and equidistant. The information provided by the estimated line integrals in one set is called a *projection* and the angle that the lines of that set make with the horizontal axis is called its *projection angle.* (Observe the dual use of the word "projection" here and in the theory of projection algorithms.) Thus we can write

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_K \end{bmatrix}, \tag{4.2}$$

where each submatrix A_θ contains rows whose coefficients represent the line integrals that belong to the θ-th projection. Since only estimates of the line integrals b_i are known to us and the left-hand side of (4.1) involves an approximation due to the uniform pixels assumption, the system (4.1) is likely to be inconsistent. This matches the real-life situation.

Our experimental results demonstrate the performance of several algorithms in reconstructing two specific phantoms. The numerical experiments were performed using the SNARK93 software package [5]. Some of the algorithms that we implemented are dependent on the order in which the equations are processed. This data access ordering has a significant effect on the practical performance of the algorithms. We used the ordering proposed by Herman and Meyer [31] both for equations inside a projection and for the projections. The underlying intuitive principle of this ordering is that in a subsequence of sequential iterative steps the action should be as independent as possible of the previous actions. Phrased slightly more mathematically, the data vector (i.e., the row of the system matrix A) that is used should be as orthogonal as possible to the space spanned by the recently used corresponding data vectors. Heuristically, this aim is achieved by selecting data vectors so that vectors have very few nonzero components in common with any of the recently used vectors. As the initial vector in all our algorithmic runs we always picked the zero vector. In all our experiments we used a fixed relaxation parameter. (We are, however, aware that using non-constant relaxation strategies may improve convergence of individual methods; see, e.g., dos Santos [19] and Combettes [18] for CIM and [31] for ART.)

In our experiments we choose a function that will be reconstructed (i.e., a phantom picture) and calculate its line integrals. These line integrals along with the system matrix A are the input data to the reconstruction algorithm and the resulting vector

representation of the reconstructed image is then compared with the original (known to us since we created it) phantom picture. Denoting by $||\cdot||_1$ the ℓ_1-norm, we report below the *relative error* $||x^{cT} - \hat{x}||_1/||\hat{x}||_1$ versus the cycle number c, where a *cycle* is one pass through all data and the number of blocks T equals the number of iterations needed to complete a cycle. We do this because the iteration index k reflects different algorithmic progress in each algorithm depending on the amount of data that is used by the algorithm in passing from iteration k to the next $(k+1)$-th iteration. For example, ART uses a single equation in each iteration and takes m iterations to complete a cycle, while CIM uses m equations in each iteration and takes a single iteration to complete a cycle. The formulation in (4.2) naturally leads to a block-iterative approach in which each projection defines a block with the A_t as in (2.18). In this case T is the number of projections K. To make our comparisons equally relevant to all algorithms, we report on the relative error versus the number of cycles. Note that although in one cycle we go through all equations (rows of the matrix), this results in essentially the same number of arithmetic operations as doing two matrix-vector multiplications (by the matrix and its transpose, respectively).

4.1. The head phantom. Our first image is the standard head phantom from [30] (it appears on the top of Figure 5.2) which has been repeatedly used in the literature as a benchmark, discretized into 63×63 pixels, and so its vector representation $\hat{x}$ is $3,969$-dimensional. To model the scanning geometry, we are using 16 projections with 99 lines per projection (evenly distributed between 0 and 174 degrees). All detector readings (i.e., the line integrals) for all views (i.e., projections) can be represented as a *sinogram*. The intensities in the sinogram are proportional to the line integrals of the X-ray attenuation coefficient between the corresponding source and the detector positions. We present such 16×99 sinograms in Figure 5.2.

The resulting projection matrix A has, therefore, dimension $1,584 \times 3,969$, so that the system of equations is quite underdetermined. To make our experiments realistic we used, besides noiseless data, also data with Gaussian additive independent noise of mean 0 and standard deviation 0.2. The values of the $1,584$ line integrals of this phantom all lie between 0 and 4. This gives rise to a relative noise strength of (at least) $0.2/4 = 5\%$.

In our series of tests for the head phantom we compared the behavior of ART (Algorithm 3.1), BIP (Algorithm 1.3), DROP (Algorithm 3.4 and 3.5), BICAV (Algorithm 3.2) and CARP (Algorithm 3.3), with the last four algorithms tested for both $T = 1$ (all equations included in a single block resulting in a fully-simultaneous algorithm) and $T = 16$ (i.e., using 16 blocks each consisting of all equations associated with one projection of the line integrals). Note that for $T = 1$ Algorithm 3.4, DROP1 and Algorithm 3.5, DROP2 coincide.

To determine an optimal value of the relaxation parameter for each algorithm, we compared the behavior of that algorithm with different relaxation parameters λ within ranges that appear in the appropriate convergence results, such as (2.9), (2.12), (2.29), (2.31), (2.34), and (2.37). Since there is no convergence result for Algorithm 3.5 we picked up the relaxation parameter in $(0, 2)$. By "optimal value" we mean that constant value of the relaxation parameters that gives rise to the smallest relative error within 10 cycles. In Figures 5.3–5.6 we show examples of such plots for the sequential ART and for DROP with $T = 16$. Note the different character of the plots obtained when reconstructing noisy data as opposed to those obtained with noiseless data. The relaxation parameter and the number of iterations have a regularizing effect on the behavior of the algorithms when data are noisy. When searching for an

optimal value of the relaxation parameter for noisy data we ran each method, for a fixed value of λ, five times, using in each run a new sample of the noisy data. The displayed plots are based on the means of the individual five runs.

The optimal combinations of fixed relaxation parameter and stopping cycle number for the (sequential) ART and for BIP, DROP, BICAV and CARP algorithms with $T = 16$, when applied to the noiseless and to the noisy data of the head phantom, are shown in Table 4.1. The same information for the fully-simultaneous algorithms obtained with $T = 1$ is shown in Table 4.2 (ART is repeated in both tables).

Algorithm	ART	BIP	DROP1	DROP2	BICAV	CARP
Noiseless data	(0.15, 10)	(44, 3)	(0.4, 8)	(0.7, 2)	(0.81, 2)	(1, 10)
Noisy data	(0.05, 10)	(5, 9)	(0.1, 10)	(0.5, 1)	(0.075, 9)	(0.1, 10)

TABLE 4.1

Optimal pairs of relaxation parameter and stopping cycle number for the head phantom. All algorithms (except ART) use $T = 16$ blocks.

Algorithm	ART	BIP	DROP1	BICAV	CARP
Noiseless data	(0.15, 10)	(90, 10)	(1.57, 10)	(1.57, 10)	=ART
Noisy data	(0.05, 10)	(100, 7)	(1.7, 6)	(1.7, 6)	=ART

TABLE 4.2

Optimal pairs of relaxation parameter and stopping cycle number for the head phantom. All algorithms (except ART) use $T = 1$ and, therefore, BIP, DROP and BICAV are fully-simultaneous while CARP (with $T = 1$) is identical to ART.

In Figures 5.7–5.11 we compare the initial behavior of the various methods with the optimal fixed relaxation parameters obtained by the methodology of the previous paragraph. As an illustration of the point we have made following Theorem 2.3, in the case of the fully-simultaneous BIP algorithm (which is in fact CIM), we also indicate the performance with a relaxation parameter λ that is optimal for a smaller range (such as (2.9) rather than (2.12)). The performance with this λ is much worse than with the one that is optimal over the larger range. No such differences were found when similar experiments were done with DROP. Observe also that in all these figures the block-iterative versions of the algorithms are superior to their fully-simultaneous versions.

Looking at the behavior of fully-simultaneous DROP in Figure 5.9, we note that the relative error increases after the 6-th cycle. This does not contradict Theorem 2.3, which states that fully-simultaneous DROP converges to a weighted least squares solution. To illustrate this, we plot in Figure 5.10 the values of the weighted least squares functional $||Ax^c - b||_W$ versus cycle number c for fully-simultaneous DROP applied to the noisy data of the head phantom. In Figure 5.2 the head phantom and its reconstructions produced by ART, DROP1 and DROP2 using 16 blocks are displayed. We also present the sinograms of the projection data, calculated by SNARK93, for the noiseless and noisy data.

4.2. The mitochondrion phantom. Our second phantom is taken from Fernández et al. [24]. That paper is on electron microscopy tomographic reconstruction of biological specimens. One of their phantoms is designed to resemble a mitochondrion. It consists of hollow cylinders representing the membranes and a set of solid cylinders simulating the cristae. The cristae are embedded in a region of intermediate

density resembling the mitochondrial inner matter. In our experiments we took a two-dimensional (2D) slice of the phantom with size of 341×341 pixels. This phantom is shown on the top of Figure 5.18. The projections simulating a single-axis geometry (see Frank [25, pp. 186–187]) ranged over 72 projection angles evenly distributed between 1 degree and 143 degrees; each of the 72 projections consisted of 495 lines. This resulted in $116,281$ columns and $35,640$ rows in the projection matrix A. We generated both noiseless projection data and Gaussian zero-mean noisy projection data with standard deviation 1. The values of the $35,640$ line integrals lie between 0 and 22. So this gives a relative noise strength of (at least) $1/22 \approx 5\%$ (of the same order as for the head phantom). The difference in line integrals along horizontal lines that go through the cristae and those that do not is 5. For this phantom we compared the initial algorithmic behavior of ART (Algorithm 3.1) and DROP1 (Algorithm 3.4) with $T = 72$ blocks as in (4.2).

In Figures 5.12–5.15 we show the relative error versus number of cycles through the data for ART and DROP1 (using 72 blocks) with various relaxation parameters. Based on these we picked optimal pairs of relaxation parameter and stopping cycle number for the two methods both for the noiseless and noisy case, see Table 4.3.

Algorithm	ART	DROP1 ($T = 72$)
Noiseless data	(0.9, 10)	(1.8, 10)
Noisy data	(0.07, 3)	(0.15, 3)

TABLE 4.3
Optimal pairs of relaxation parameter and stopping cycle number for the mitochondrion phantom.

We display the behavior of the two algorithms with the optimal value of the relaxation parameter in Figure 5.16 (noiseless data) and in Figure 5.17 (noisy data), and in Figure 5.18 we show the mitochondrion phantom and its reconstructions that were produced by ART and by DROP1 with 72 blocks. Again, the 72×495 sinograms of the projection data, calculated by SNARK93, for the noiseless and noisy phantoms are presented.

4.3. General comments . In Table 4.4 we display the mean run times τ, in seconds, per cycle obtained for the head phantom with the various algorithms. In Table 4.5 we have listed the mean run times τ, in seconds, per cycle obtained for the mitochondrion phantom for ART, DROP1 ($T = 72$) and, for run-time comparison only, the fully-simultaneous DROP ($T = 1$). We see that the cost of performing block-iterations on the larger mitochondrion phantom is quite high on a single processor.

Algorithm	Run-time τ
ART (sequential)	0.042
BIP ($T = 1$)	0.044
DROP ($T = 1$)	0.043
BICAV ($T = 1$)	0.051
CARP ($T = 1$)	=ART
BIP ($T = 16$)	0.053
DROP1 ($T = 16$)	0.056
BICAV ($T = 16$)	0.063
CARP ($T = 16$)	0.055

TABLE 4.4
Run-times τ, in seconds, per cycle for the head phantom.

Algorithm	Run-time τ
ART (sequential)	4.776
DROP1 ($T = 72$)	8.463
DROP ($T = 1$)	5.261

TABLE 4.5
Run-times τ, in seconds, per cycle for the mitochondrion phantom.

The tests were performed on a single processor within a SUNRAY multi-user network with the Solaris operating system. As remarked above we used the SNARK93 program [5], which is a (Fortran) programming system for image reconstruction from projections. It facilitates simulating the model, running reconstruction algorithms and evaluating the results, including run times. Only the ART algorithm was used from the built-in algorithms of SNARK93; the other methods were all coded by us and given to SNARK93 as user's algorithms.

In all relative error versus cycle plots of noisy experiments we observe that the relative errors start to increase after a certain number of cycles. One reason for this is the error in the right-hand side of the linear system combined with the ill-conditioning of the matrix. The more ill-conditioned the matrix is the faster the divergence (for a fixed error in the right-hand side) and vice versa. The iterates converge to a least-squares solution which is based on the noisy right-hand side vector rather than the noise-free one. This can also be analyzed using the singular value expansion of the matrix, see, e.g., Hansen [29].

It is, therefore, important to correctly stop the iteration process. Iterating too long means that the noise-component in the right-hand side vector dominates the solution whereas performing too few iterations means loss of resolution in the iterates. Another aspect is that even the noise-free least-squares solution might not be a good approximation to the phantom. This could be so due to insufficient modelling for instance.

5. Summary. In the literature on reconstruction from projections, for example in [32] and [36, see Eq. (3)], researchers introduced diagonally-relaxed orthogonal projections (DROP) for heuristic reasons that are outlined in Section 1. However, there has been until now no mathematical study of the convergence behavior of such algorithms. Our paper makes a contribution here.

In Section 2 we have considered a fully-simultaneous DROP algorithm for linear equations and have proved its convergence without consistency assumptions. We have also introduced general (block-iterative) algorithms both for linear equations and for linear inequalities and have studied their convergence in special cases, but only for the consistent case. We have followed this by two sections, the first containing precise descriptions of a number of iterative algorithms that we have implemented for the purpose of an experimental study (both DROP-type and other classical algorithms) and the second reporting on the experimental study itself. A phantom based on a medical problem and another based on a problem of electron microscopy have been used to generate both noiseless and noisy projection data, and various algorithms have been applied to such data for the purpose of comparison. The results show that the use of DROP as an image reconstruction algorithm is not inferior to previously used methods. Those practitioners who used it without the mathematical justification offered here were indeed creating very good reconstructions. All our experiments are performed in a single processor environment. Further computational gains can be achieved by DROP in a parallel computing environment with appropriate block

choices but doing so and comparing it to other algorithms that were used in the comparisons made here calls for a separate study.

Acknowledgments. We thank Rachel Gordon and Dan Gordon for having sent us a preprint of their paper [28] and for providing advice in computational matters. Their detailed comments on an earlier version are also gratefully acknowledged. We thank two anonymous referees for their constructive reports. We thank Paulo J.S. Silva and Marlon Wisner for drawing our attention to an error in a previous version of the paper. This research is supported by the National Institutes of Health (NIH) through grant No. HL70472 and by grant No. 2003275 from the United States-Israel Binational Science Foundation (BSF). Part of this work was done at the Center for Computational Mathematics and Scientific Computation (CCMSC) at the University of Haifa and was supported by research grant No. 522/04 from the Israel Science Foundation (ISF).

REFERENCES

[1] R. Aharoni and Y. Censor, *Block-iterative projection methods for parallel computation of solutions to convex feasibility problems*, Linear Algebra and Its Applications, 120 (1989), pp. 165–175.

[2] A. Auslender, *Optimisation: Méthodes Numériques*, Masson, Paris, France, 1976.

[3] H. H. Bauschke and J. M. Borwein, *On projection algorithms for solving convex feasibility problems*, SIAM Review, 38 (1996), pp. 367–426.

[4] M. Benzi, *Gianfranco Cimmino's contributions to numerical mathematics*, Seminario di Analisi Matematica, Dipartimento di Matematica dell'Università di Bologna, Ciclo di Conferenze in Ricordo di Gianfranco Cimmino, Marzo-Maggio 2004. Tecnoprint, Bologna, Italy (2005), pp. 87–109.

[5] J. A. Browne, G. T. Herman and D. Odhner, *SNARK93: A programming system for image reconstruction from projections*, The Medical Imaging Processing Group (MIPG), Department of Radiology, The University of Pennsylvania, Technical Report MIPG198, 1993.

[6] D. Butnariu, Y. Censor and S. Reich, eds., *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, Elsevier Science Publishers, Amsterdam, The Netherlands, 2001.

[7] C. Byrne and Y. Censor, *Proximity function minimization using multiple Bregman projections, with applications to split feasibility and Kullback-Leibler distance minimization*, Annals of Operations Research, 105 (2001), pp. 77–98.

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

[9] ———, *Mathematical optimization for the inverse problem of intensity-modulated radiation therapy*, in Intensity-Modulated Radiation Therapy: The State of The Art, J. R. Palta and T. R. Mackie, eds., American Association of Physicists in Medicine (AAPM), Medical Physics Monograph No. 29, Medical Physics Publishing, Madison, Wisconsin, USA, 2003, pp. 25–49.

[10] ———, T. Elfving and G. T. Herman, *Averaging strings of sequential iterations for convex feasibility problems*, in Inherently Parallel Algorithms in Feasibility and Optimization and their Applications, D. Butnariu, Y. Censor, and S. Reich, eds., Elsevier Science Publishers, Amsterdam, The Netherlands, 2001, pp. 101–114.

[11] ———and T. Elfving, *Block-iterative algorithms with diagonally scaled oblique projections for the linear feasibility problem*, SIAM Journal on Matrix Analysis and Applications, 24 (2002), pp. 40–58.

[12] ———, D. Gordon and R. Gordon, *Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems*, Parallel Computing, 27 (2001), pp. 777–808.

[13] ———, ———,———, *BICAV: An inherently parallel algorithm for sparse systems with pixel-dependent weighting*, IEEE Transactions on Medical Imaging, 20 (2001), pp. 1050–1060.

[14] ———and S. A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*, Oxford University Press, New York, NY, USA, 1997.
[15] G. Cimmino, *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari*, La Ricerca Scientifica, XVI, Series II, Anno IX, 1 (1938), pp. 326–333.
[16] D. Chazan and W. L. Miranker, *Chaotic relaxation*, Linear Algebra and Its Application, 2 (1969), pp. 199–222.
[17] P. L. Combettes, *The convex feasibility problem in image recovery*, in Advances in Imaging and Electron Physics, P. W. Hawkes, ed., pp. 155–270, Academic Press, New York, NY, USA, 1996.
[18] ———, *Convex set theoretic image recovery by extrapolated iterations of parallel subgradient projections*, IEEE Transactions on Image Processing, 6 (1997), pp. 493–506.
[19] L.T. dos Santos, *A parallel subgradient projections method for the convex feasibility problem*, Journal of Computational and Applied Mathematics, 18 (1987), pp. 307–320.
[20] N. Echebest, M. T. Guardarucci, H.D. Scolnik and M.C. Vacchino, *An accelerated iterative method with diagonally scaled oblique projections for solving linear feasibility problems*, Annals of Operations Research, 138 (2005), pp. 235–257.
[21] P. P. B. Eggermont, G. T. Herman and A. Lent, *Iterative algorithms for large partitioned linear systems, with applications to image reconstruction*, Linear Algebra and Its Applications, 40 (1981), pp. 37–67.
[22] T. Elfving, *Block-iterative methods for consistent and inconsistent linear equations*, Numerische Mathematik, 35 (1980), pp. 1–12.
[23] L. Elsner, I. Koltracht and M. Neumann, *Convergence of sequential and asynchronous nonlinear paracontractions*, Numerische Mathematik, 62 (1992), pp. 305–319.
[24] J.-J. Fernández, A. F. Lawrence, J. Roca, I. García, M. H. Ellisman and J.-M. Carazo, *High-performance electron tomography of complex biological specimens*, Journal of Structural Biology, 138 (2002), pp. 6–20.
[25] J. Frank, *Three-Dimensional Electron Microscopy of Macromolecular Assemblies*, Academic Press, San Diego, CA, USA, 1996.
[26] A. Frommer and D. B. Szyld, *On asynchronous iterations*, Journal of Computational and Applied Mathematics, 123 (2000), pp. 201–216.
[27] U. M. García-Palomares, *Parallel projected aggregation methods for solving the convex feasibility problem*, SIAM Journal on Optimization, 3 (1993), pp. 882–900.
[28] D. Gordon and R. Gordon, *Component-averaged row projections: A robust block-parallel scheme for sparse linear systems*, SIAM Journal on Scientific Computing, 27 (2005), pp. 1092–1117.
[29] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia PA, USA, 1998.
[30] G. T. Herman, *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography*, Academic Press, New York, NY, USA, 1980.
[31] ———and L. B. Meyer, *Algebraic reconstruction technique can be made computationally efficient*, IEEE Transactions on Medical Imaging, 12 (1993), pp. 600–609.
[32] ———, S. Matej and B. M. Carvalho, *Algebraic reconstruction techniques using smooth basis functions for helical cone-beam tomography*, in Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications, D. Butnariu, Y. Censor and S. Reich, eds., Elsevier Science Publishers, Amsterdam, The Netherlands, 2001, pp. 307–324.
[33] H. M. Hudson and R. S. Larkin, *Accelerated image reconstruction using ordered subsets projection data*, IEEE Transactions on Medical Imaging, 13 (1994), pp. 601–609.
[34] M. Jiang and G. Wang, *Convergence studies on iterative algorithms for image reconstruction*, IEEE Transactions on Medical Imaging, 22 (2003), pp. 569–579.
[35] F. Natterer and F. Wübbeling, *Mathematical methods in image reconstruction*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, USA, 2001.
[36] C. O. S. Sorzano, R. Marabini, G. T. Herman and J.-M. Carazo, *Multiobjective algorithm parameter optimization using multivariate statistics in three-dimensional electron microscopy reconstruction*, Pattern Recognition, 38 (2005), pp. 2587–2601.
[37] H. Stark and Y. Yang, *Vector Space Projections: A Numerical Approach to Signal and Image Processing, Neural Nets, and Optics*, John Wiley & Sons, New York, NY, USA, 1998.

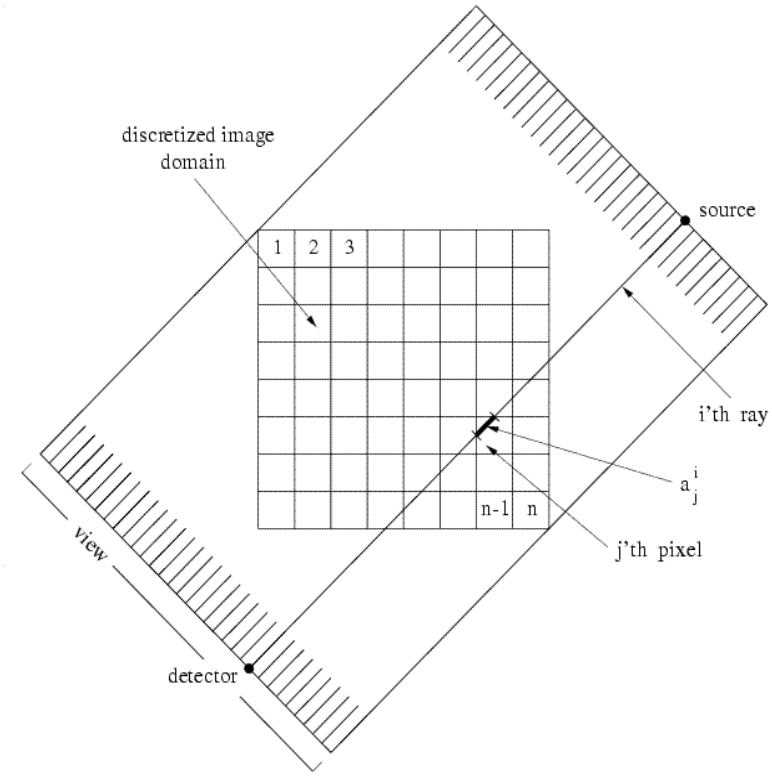

FIG. 5.1. *The fully-discretized model of the image reconstruction problem.*

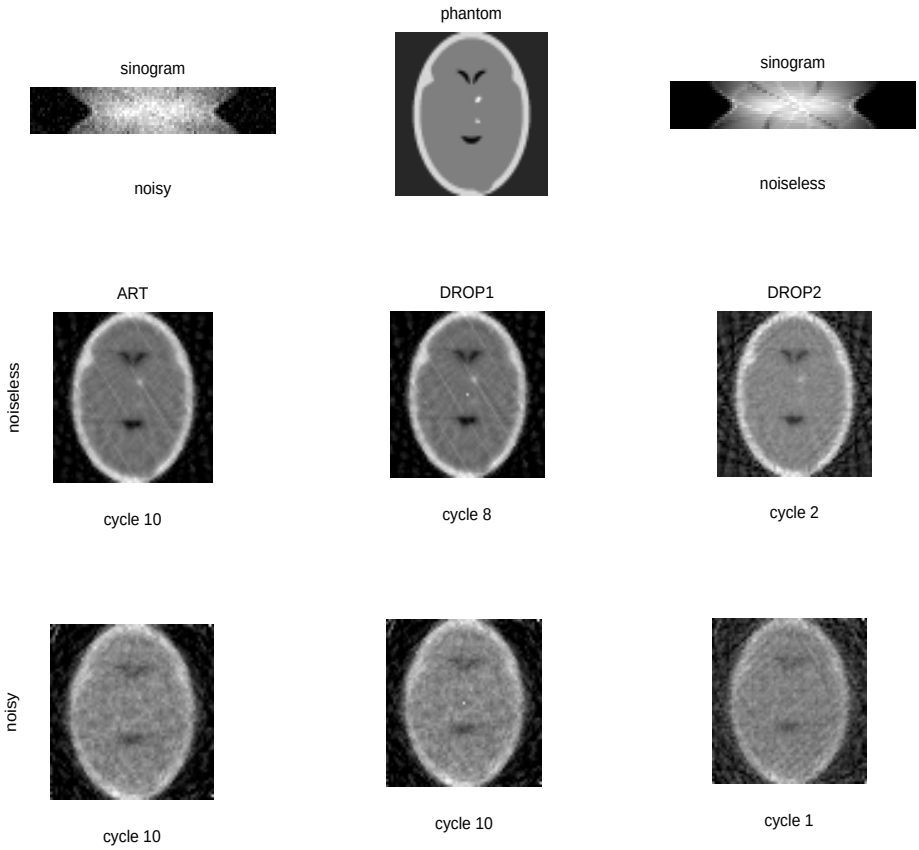

FIG. 5.2. *Images of the head phantom and of its reconstructions produced by ART, DROP1 and DROP2 (with 16 blocks), with and without noise. Beneath the phantom are displayed the sinograms of the noiseless and noisy data, respectively.*

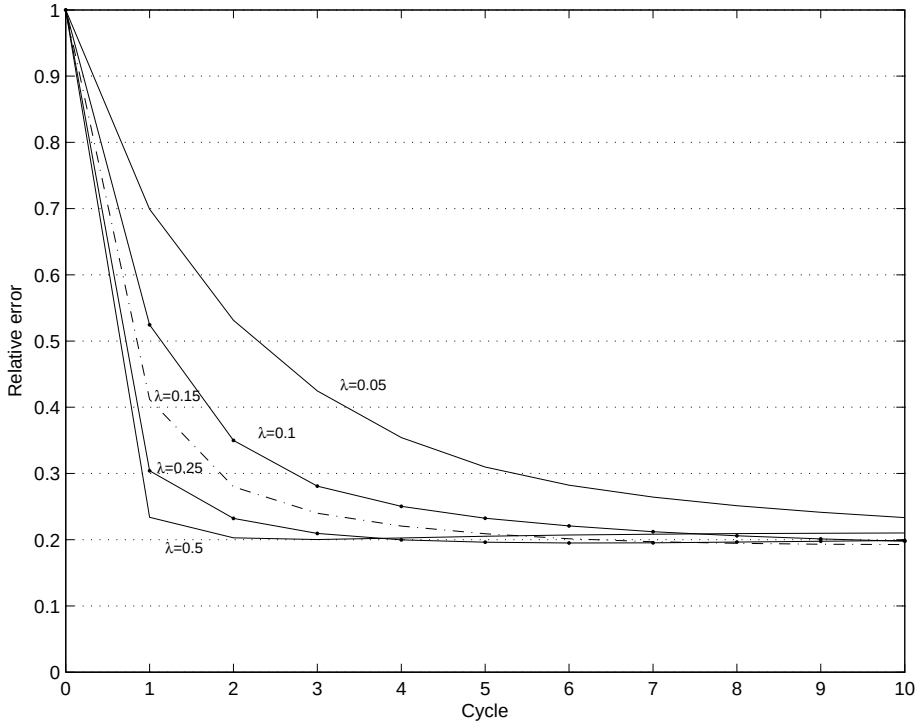

FIG. 5.3. *Plots of the relative error versus cycle for the ART algorithm on the head phantom noiseless data for different values of the relaxation parameter* λ.

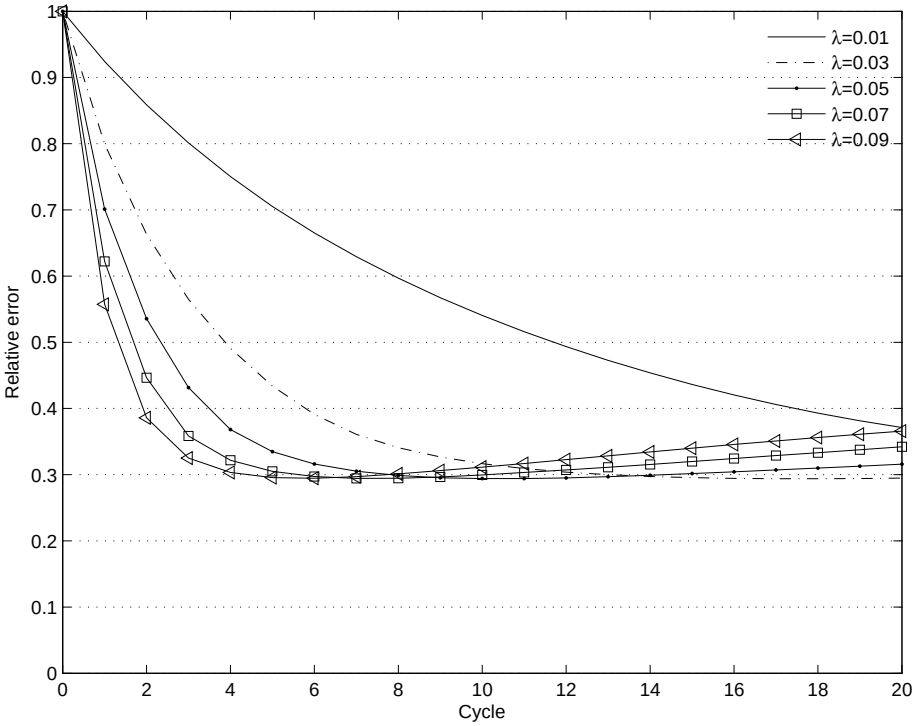

FIG. 5.4. *Plots of the relative error versus cycle for the ART algorithm on the head phantom noisy data for different values of the relaxation parameter* λ. *Each curve is based on the means of 5 independent runs.*

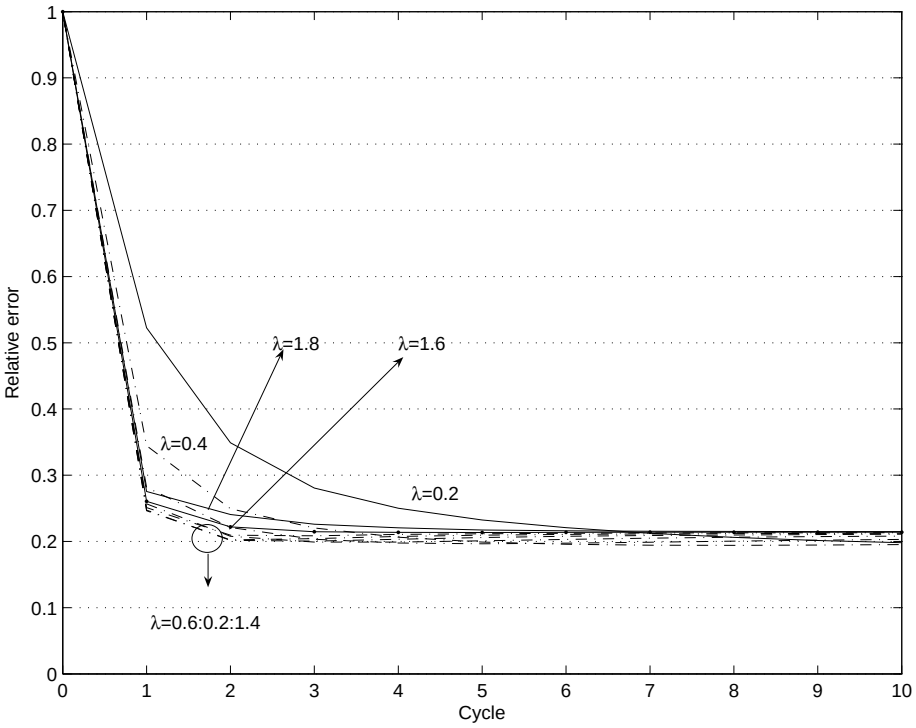

FIG. 5.5. *Plots of the relative error versus cycle for the DROP1 algorithm (with 16 blocks) on the head phantom noiseless data for different values of the relaxation parameter* λ. *The encircled area indicates almost overlapping plots for* λ *values between 0.6 and 1.4, gradually incremented by 0.2 steps.*

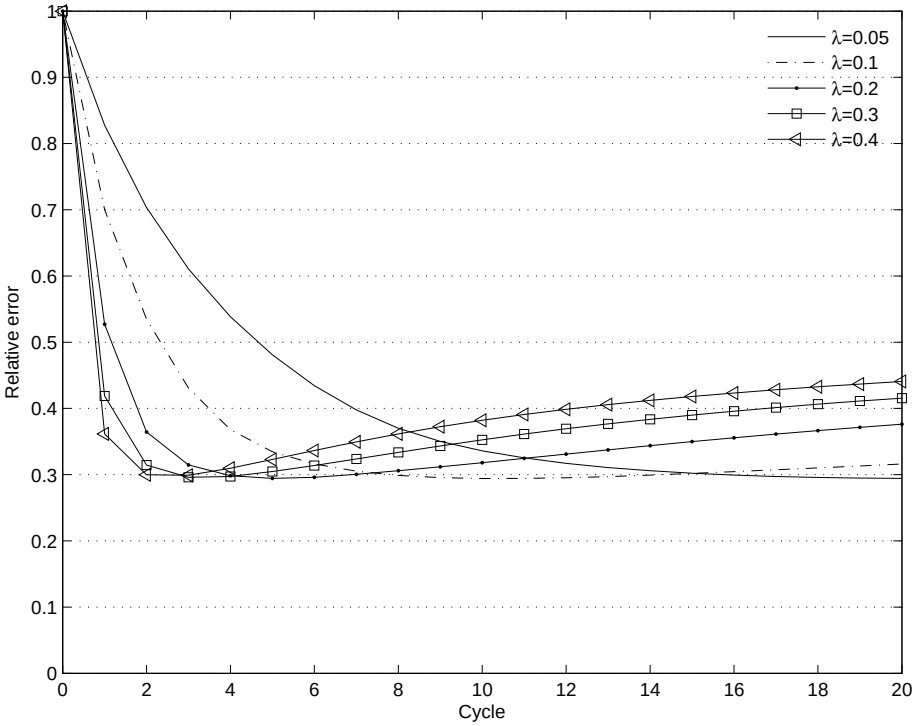

FIG. 5.6. *Plots of the relative error versus cycle for the DROP1 algorithm (with 16 blocks) on the head phantom noisy data for different values of the relaxation parameter* λ. *Each curve is based on the means of 5 independent runs.*

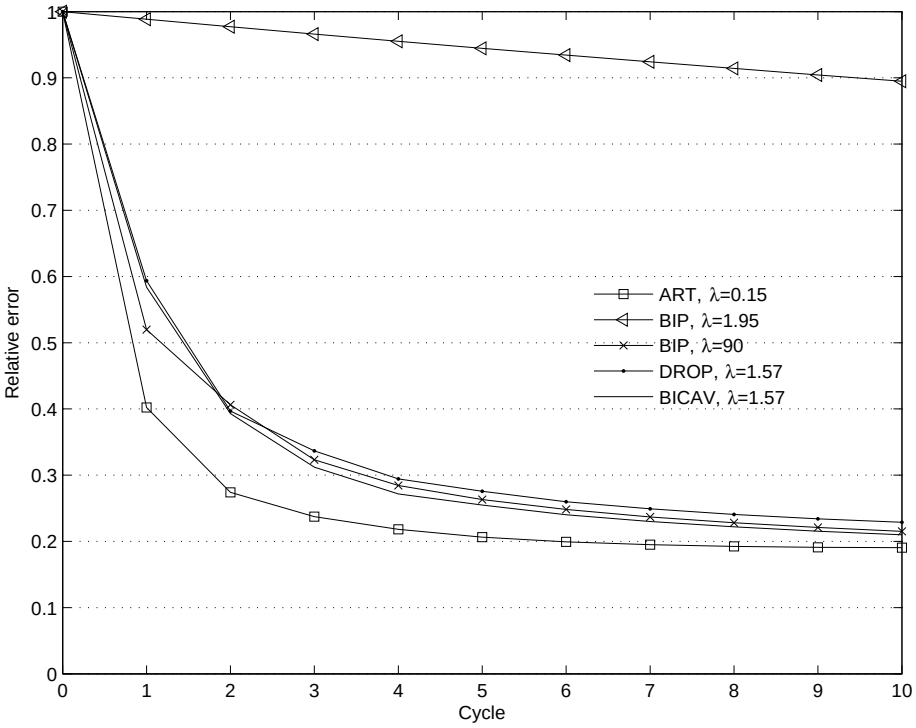

FIG. 5.7. *Relative error versus cycle performance plots. The noiseless head phantom data is being reconstructed here by the ART, BIP, DROP, BICAV and CARP algorithms, the last four in their fully-simultaneous versions, i.e., all equations form a single block. The fully-simultaneous versions of BIP and BICAV are also called CIM and CAV, respectively. CARP with a single block is identical to the (fully-sequential) ART algorithm. Each algorithm uses the optimal (fixed) relaxation parameter value that suites it best according to previous runs, and in the case of BIP we also indicate performance with a suboptimal relaxation parameter.*

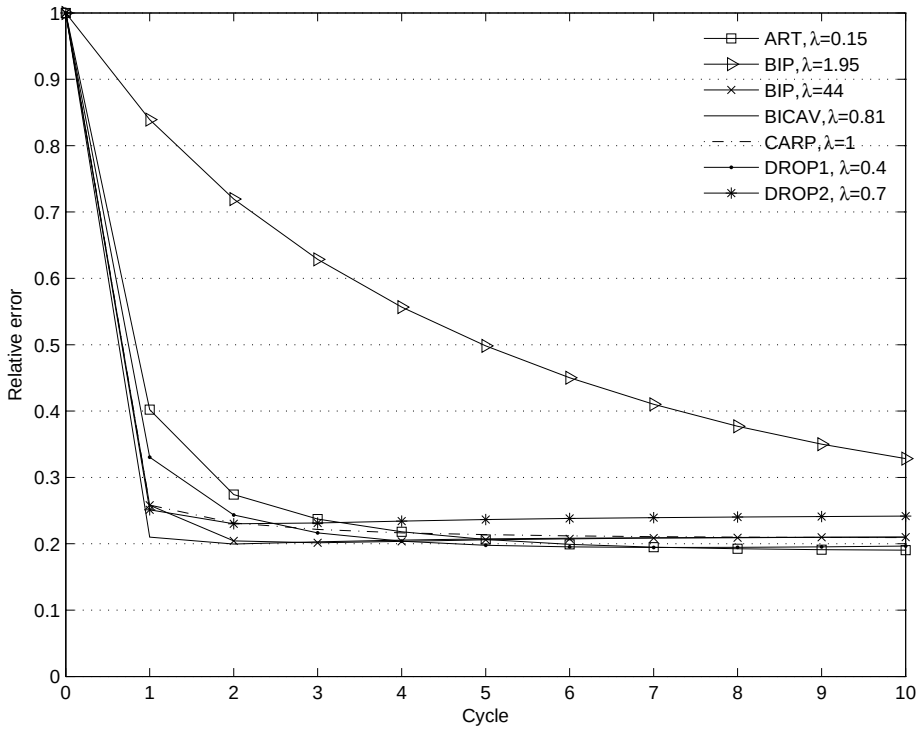

FIG. 5.8. *Relative error versus cycle performance plots. The noiseless head phantom data is being reconstructed here by the (fully-sequential) ART, BIP, DROP1, DROP2, BICAV and CARP algorithms, the last four using 16 blocks each. Each algorithm uses the optimal (fixed) relaxation parameter value that suites it best according to previous runs, and in the case of BIP we also indicate performance with a suboptimal relaxation parameter.*

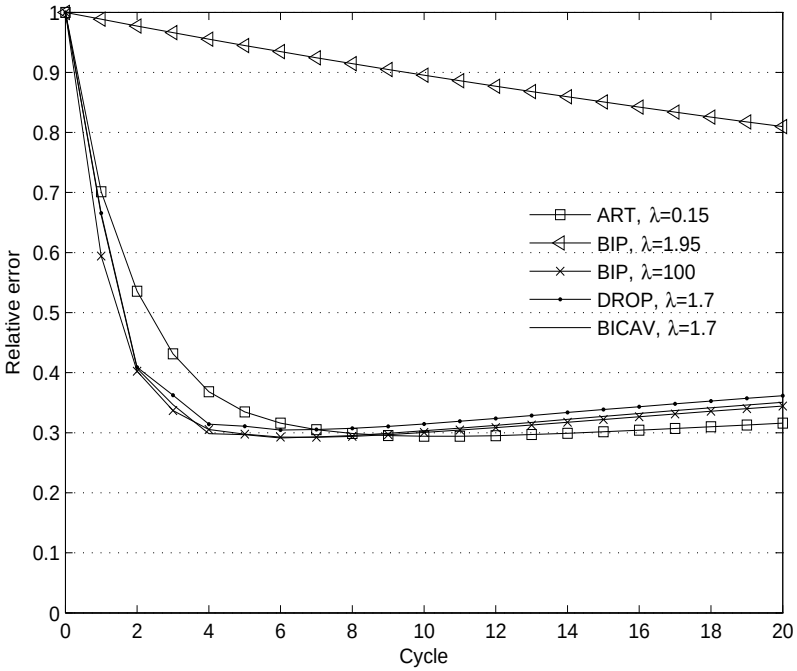

FIG. 5.9. *Relative error versus cycle performance plots. The noisy head phantom data is being reconstructed here by the ART, BIP, DROP, BICAV and CARP algorithms, the last four in their fully-simultaneous versions, i.e., all equations form a single block. CARP with a single block is identical to the fully-sequential ART algorithm. Each algorithm uses the optimal (fixed) relaxation parameter value that suites it best according to previous runs, and in the case of BIP we also indicate performance with a suboptimal relaxation parameter. Each curve is based on the means of 5 independent runs.*

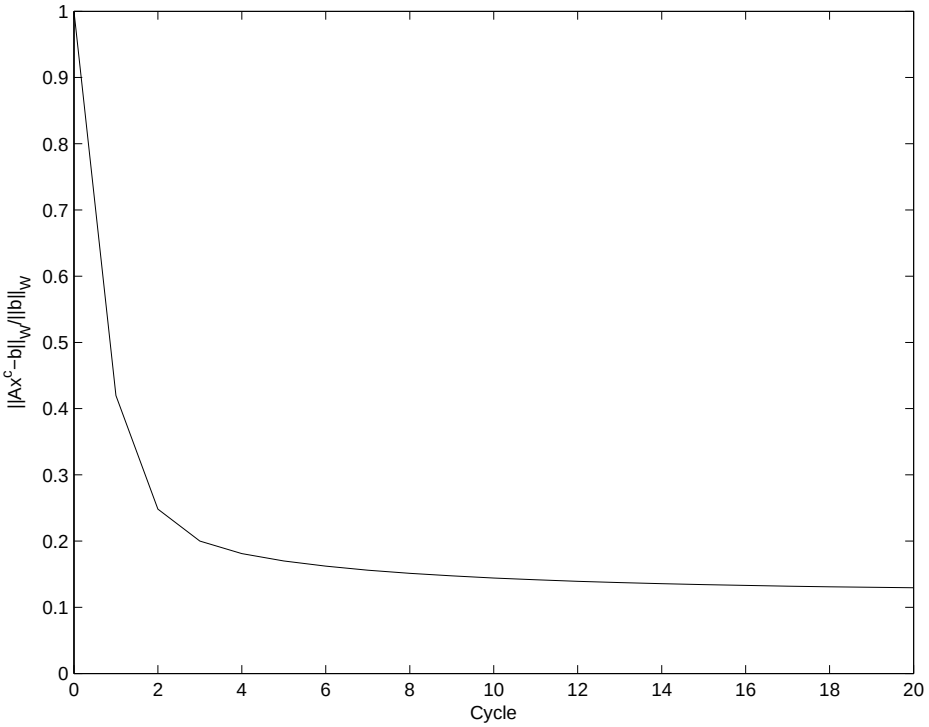

FIG. 5.10. *Plot of the values of the weighted least squares functional versus cycle number for the fully-simultaneous DROP algorithm applied to the noisy head phantom data. The relaxation parameter is the same as used for DROP in Figure 5.9, i.e.,* $\lambda = 1.7$.

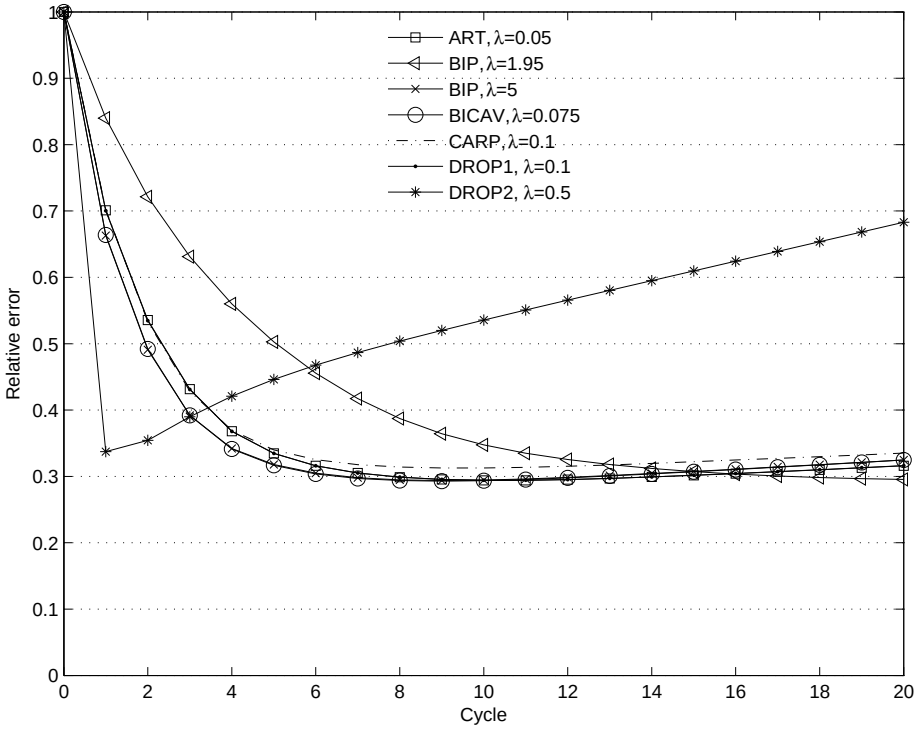

FIG. 5.11. *Relative error versus cycle performance plots. The noisy head phantom data is being reconstructed here by the ART, BIP, DROP1, DROP2, BICAV and CARP algorithms, the last four using 16 blocks each. Each algorithm uses the optimal (fixed) relaxation parameter value that suites it best according to previous runs, and in the case of BIP we also indicate performance with a suboptimal relaxation parameter. Each curve is based on the means of 5 independent runs.*

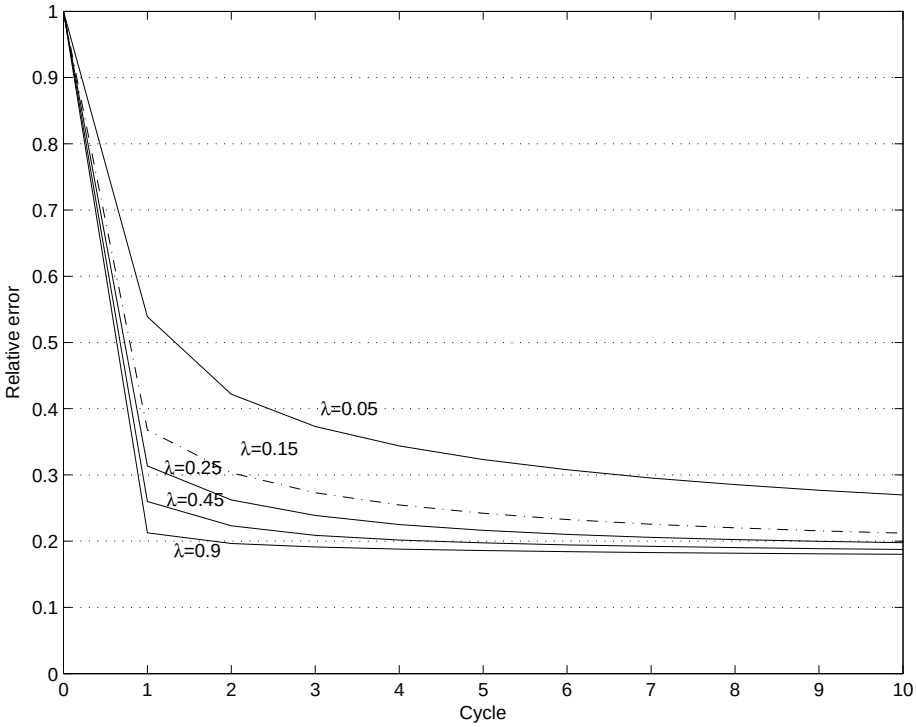

FIG. 5.12. *Plots of the relative error versus cycle for the ART algorithm on the noiseless mitochondrion phantom data for different values of the relaxation parameter λ, ranging from 0.05 to 0.9.*

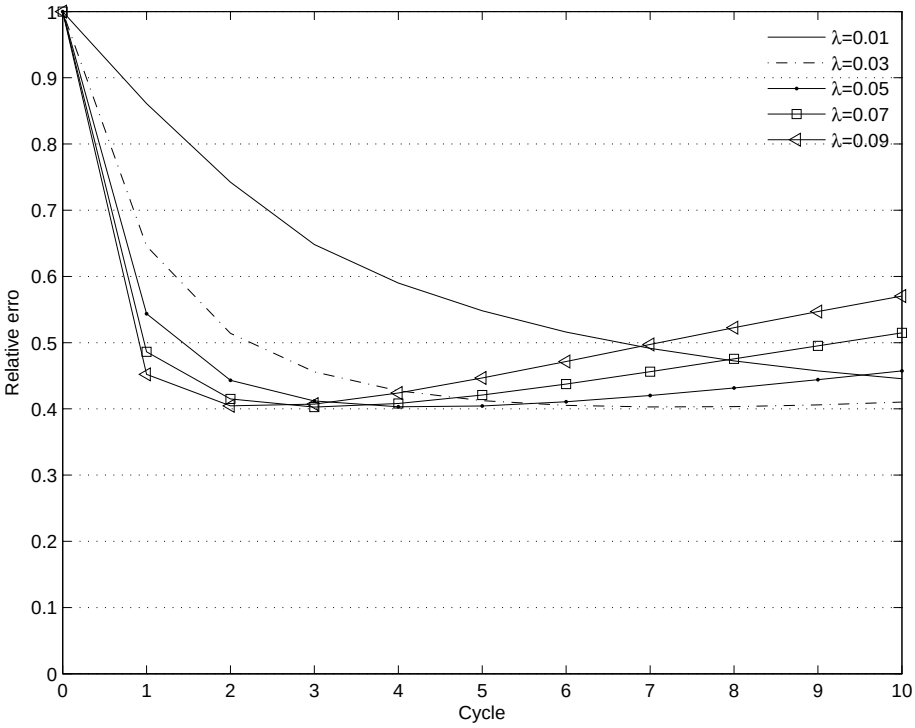

FIG. 5.13. *Plots of the relative error versus cycle for the ART algorithm on the noisy mitochondrion phantom data for different values of the relaxation parameter* λ*, ranging from 0.01 to 0.09 and gradually incremented by 0.02 steps. Each curve is based on the means of 5 independent runs.*

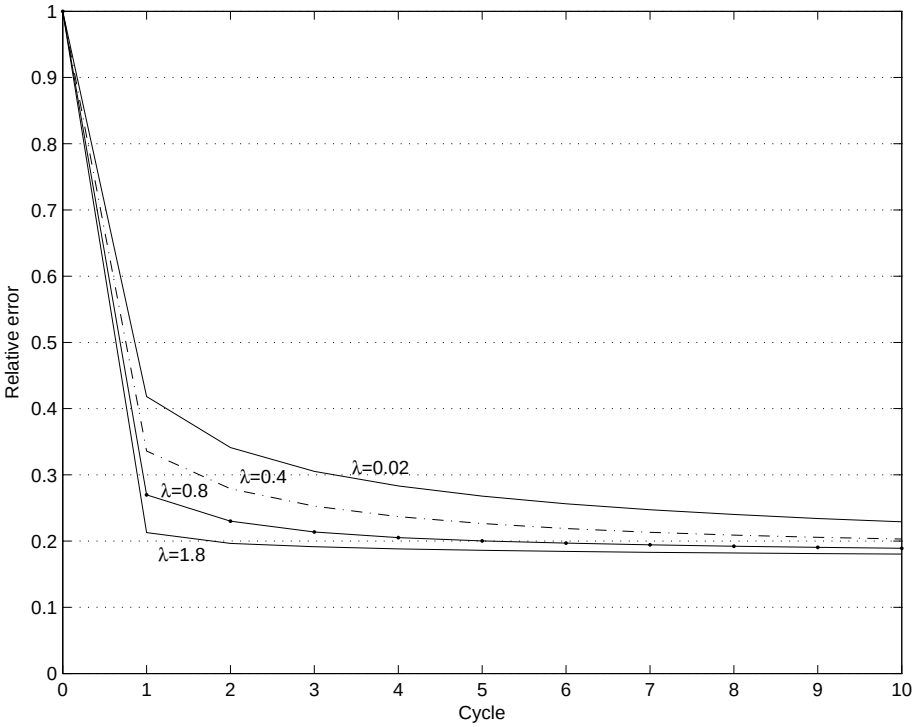

FIG. 5.14. *Plots of the relative error versus cycle, reconstructing the noiseless mitochondrion phantom. The DROP1 algorithm with 72 blocks is used with different (fixed per run) values of the relaxation parameter* λ.

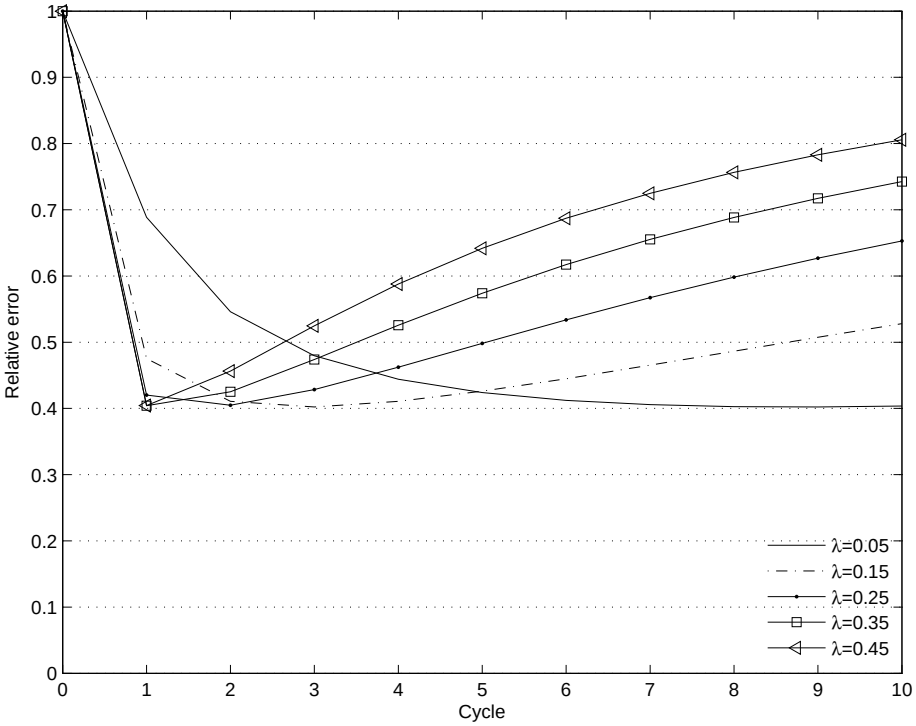

FIG. 5.15. *Plots of the relative error versus cycle, reconstructing the noisy mitochondrion phantom. The DROP1 algorithm with 72 blocks is used with different (fixed per run) values of the relaxation parameter* λ. *Each curve is based on the means of 5 independent runs.*

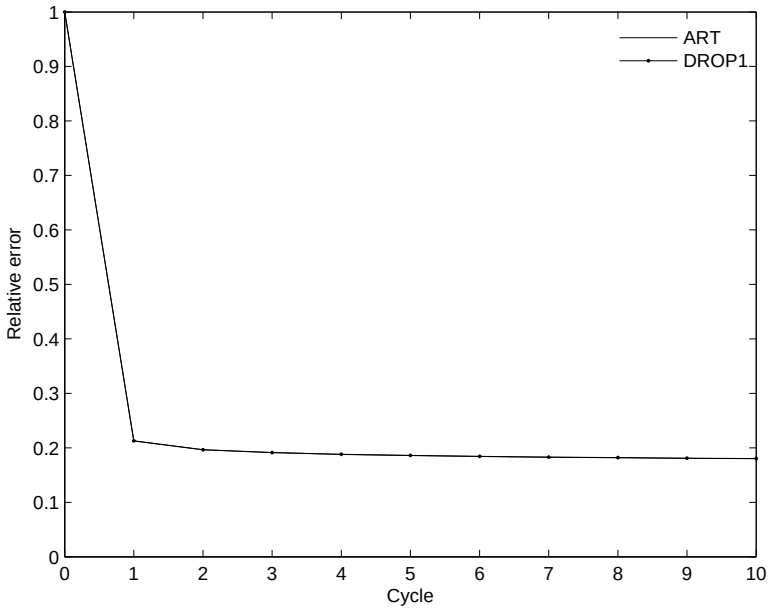

FIG. 5.16. *Relative error versus cycle performance plots, reconstructing the noiseless mitochondrion phantom with the DROP1 algorithm using 72 blocks, and the fully-sequential ART algorithm. Each algorithm uses the optimal (fixed) relaxation parameter value that suites it best according to previous runs.*

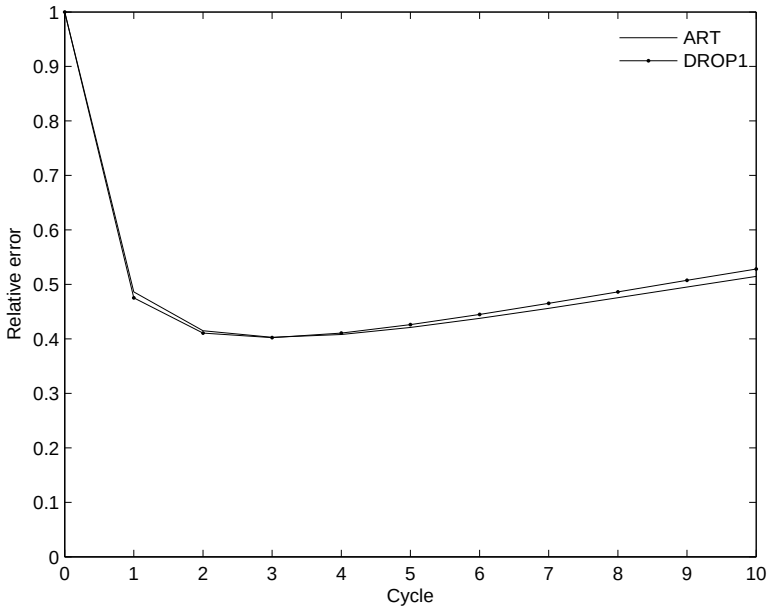

FIG. 5.17. *Relative error versus cycle performance plots, reconstructing the noisy mitochondrion phantom with the DROP1 algorithm sing 72 blocks, and the fully-sequential ART algorithm. Each algorithm uses the optimal (fixed) relaxation parameter value that suites it best according to previous runs. Each curve is based on the means of 5 independent runs.*

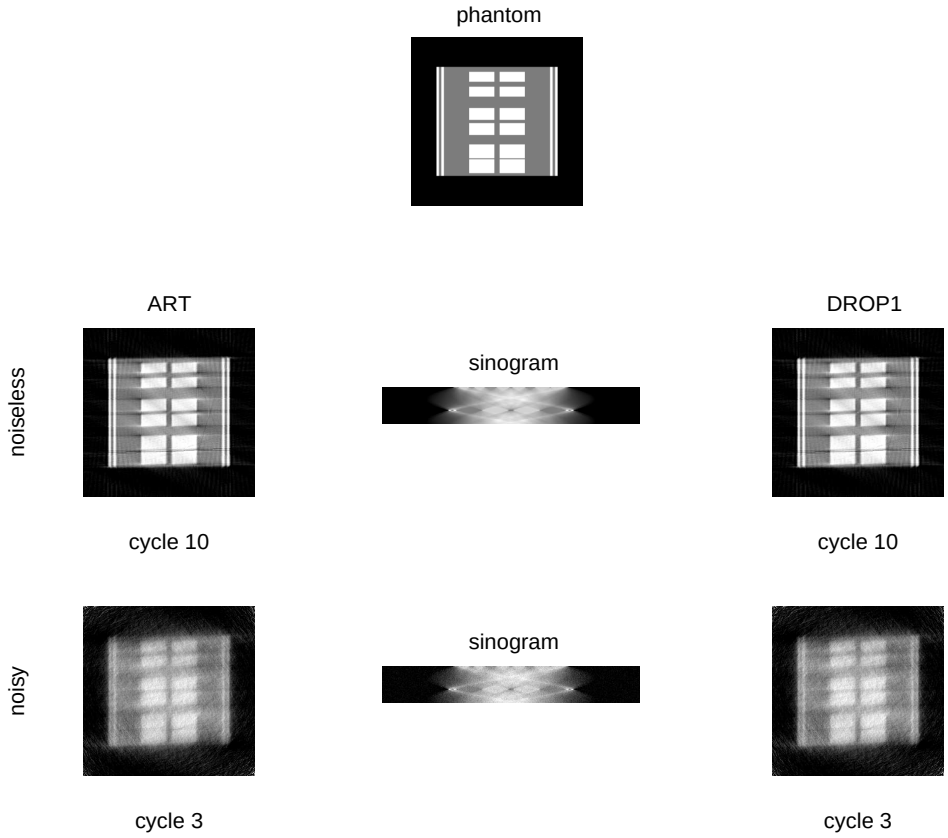

FIG. 5.18. *Images of the mitochondrion phantom and of reconstructions produced by ART and by DROP1 (with 72 blocks), with and without noise. The appearance of the reconstructions is influenced by the fact that the projection angles range over only 142 degrees (rather then the full 180 degrees). This accurately reflects the limitation of electron microscopy using a single-axis data collection geometry. Beneath the phantom are displayed the sinograms of the noiseless and noisy data, respectively.*