Applied Analysis and Optimization
Volume 2, Number 2, 2018, 253–270

Yokohama Publishers
ISSN 2189-1664 Online Journal

THE CIMMINO-KACZMARZ EQUIVALENCE AND RELATED RESULTS

DAN GORDON

Abstract. The Kaczmarz algorithm, which is a well-known solution method for linear systems, is based on orthogonal projections towards the hyperplanes determined by the equations. It is shown that a class of algorithms which use such projections in combination with averaging operations are equivalent to the Kaczmarz algorithm in some superspace of the problem space. There are two main consequences of these equivalences. Firstly, proofs of convergence for such algorithms can utilize the well-known results on the convergence of several variants of the Kaczmarz algorithm. Secondly, such algorithms can be accelerated by Conjugate Gradients (CG), similarly to the CGMN and CARP-CG algorithms, and convergence of such accelerations follow from the convergence of CG. The basis for the equivalence to Kaczmarz is the "Averaging Lemma" (AL) of Gordon & Gordon, which was used to show that CARP is equivalent to Kaczmarz in some superspace. AL shows that averaging of components of a single vector is equivalent to certain Kaczmarz projections. This work extends AL to the averaging of vectors, and presents detailed constructions for each equivalence case.

1. Introduction

The purpose of this work is to present a methodology for dealing with a certain class of iterative algorithms for the solution of linear systems. The algorithms considered are based on two operations: orthogonal projections of the current iterate onto the hyperplanes defined by the equations, and averaging operations between intermediate results.

The prototype of these algorithms was published by Kaczmarz in [32], where he proved convergence for consistent linear systems. The Kaczmarz algorithm (KACZ) does not involve any averaging operations, and convergence proofs for variants of KACZ are well known. Cimmino [13] presented an iterative algorithm that involves projections and averaging operations.

This paper shows that averaging operations are mathematically equivalent to projections in some superspace of the original problem space. There are two main consequences of such equivalences for algorithms that involve both projections and averaging:

(1) Convergence proofs follow from the convergence of the equivalent Kaczmarz methods, and these are well known for many variants of Kaczmarz.
(2) The algorithms can be accelerated similarly to CGMN, which is a Conjugate Gradients (CG) accelerations of Kaczmarz [5, 23], and CARP-CG [25]. The

2010 *Mathematics Subject Classification.* 15A06, 65F10, 68W10.
Key words and phrases. Averaging lemma, CARP, CARP-CG, Cimmino, Cimmino-Kaczmarz equivalence, convergence proofs, DROP, Kaczmarz, string averaging.

convergence of the accelerated methods then follows from the convergence of CG.

The algorithms which are shown to be equivalent to Kaczmarz in some superspace are the Cimmino algorithm with varying relaxation parameters, a subset version of Cimmino, CAV and BICAV [12, 11], and String Averaging [9]. The basic tool that will be used is the Averaging Lemma (AL) [22, 25], which shows that the operation of averaging several components of a vector is equivalent to Kaczmarz projections. AL is extended to show that averaging several vectors is also equivalent to Kaczmarz projections in some superspace of the problem space. The proofs of equivalence to Kaczmarz involve the construction of certain matrices that are specific for each case.

The rest of the paper is organized as follows. The necessary background on KACZ and AL is presented in the next section. The following sections show the equivalence of KACZ to Cimmino with varying relaxation parameters, a subset version of Cimmino, CAV, BICAV, and String Averaging.

2. Background

2.1 The Kaczmarz algorithm and its properties

The Kaczmarz algorithm (KACZ) is well-known as a solution method for linear systems in various applications, such as computerized tomography (CT), where it also known as ART (algebraic reconstruction technique) [27]. It is best described by its simple geometric explanation: starting from some selected point in the solution space $\mathbb{R}^n$, the current iterate is repeatedly projected orthogonally towards the hyperplane defined by one of the system's equations. Usually, the projections follow cyclically the given order of the linear system. It is also known that for some applications, a random selection of the equations can provide better results than the cyclic order [7, 29], and in some cases, randomness improves the rate of convergence [7, 29, 44, 21].

We consider an $m \times n$ linear system

$$Ax = b, \tag{2.1}$$

where $b = (b_1, \ldots, b_m)^T$, and the jth row of A is denoted by a^j.

Let x^0 be a chosen initial iterate, and for $k \geqslant 1$, let $i(k)$ be an integer between 1 and m which determines the order of the projections; $i(k)$ is also known as the "control" of the order of projections. For $k \geqslant 0$, x^{k+1} is obtained from x^k by projecting x^k towards the hyperplane determined by the $i(k)$th equation, i.e.,

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

where $0 < \lambda_k < 2$ is a relaxation parameter with the following significance: if $\lambda_k = 1$ then the projection is exactly on the hyperplane; if $0 < \lambda_k < 1$ then the projection lies between x^k and the hyperplane, and if $\lambda_k > 1$ then the projection is beyond the hyperplane. We can save some computation time by initially dividing each equation $\langle a^j, x \rangle = b_j$ by $\|a^j\|_2$, thus avoiding the division by $\|a^j\|_2^2$ at every step; this is sometimes called normalizing the equations. For ease of notation, we will assume henceforth that the system (2.1) has already been normalized.

For a square and consistent linear system, Kaczmarz proved convergence for the case where the projections were done in cyclic order. For an inconsistent system, Tanabe [45] proved *cyclic convergence*, meaning that if the control is cyclic, then the sequence of projections on each hyperplane converges to a point. Tanabe's result was proved for relaxation equal to 1. This actually means that if the control sequence is cyclic, then KACZ converges even on an inconsistent system because the sequence of points on the last hyperplane converges.

Herman et al. [28] proved convergence in the consistent case with relaxation parameters, provided that for some $\epsilon > 0$, $\epsilon \leqslant \lambda_k \leqslant 2-\epsilon$ for all k. In the inconsistent case, Censor et al. [8] showed that for a fixed value of the relaxation parameter λ, if the limits of the projections on the hyperplanes are denoted by $p_i(\lambda),\ 1 \leqslant i \leqslant m$, then for all $1 \leqslant i \leqslant m$, $p_i(\lambda) \xrightarrow[\lambda \to 0]{} p$, where p is a point that minimizes the sum of least squares of its distances to the hyperplanes. It follows that by successively reducing the size of λ, the limit can get arbitrarily closer to the least squares solution of the system.

We now introduce the following definition for notational convenience:

Definition 2.1. Given a normalized $m \times n$ linear system (2.1), a sequence of relaxation parameters $\Lambda = (\lambda_1, \ldots, \lambda_m)$, we define an operator called "Kaczmarz sweep", $\mathrm{KSWP}(A, b, x, \Lambda) : \mathbb{R}^n \to \mathbb{R}^n$, as follows. For $x \in \mathbb{R}^n$, we set $y^0 = x$, and for $i = 1, 2, \ldots, m$, define

$$y^i \ = \ y^{i-1} \ + \ \lambda_i \left(b_i - \langle a^i, y^{i-1} \rangle \right) a^i,$$

then $\mathrm{KSWP}(A, b, x, \Lambda) = y^m$.

Algorithm 2.2. (Kaczmarz with cyclic relaxation parameters)
begin algorithm
Given a sequence of relaxation parameters $\Lambda = (\lambda_1, \ldots, \lambda_m)$,
choose an initial iterate $x^0 \in \mathbb{R}^n$.
for $k = 0, 1, \ldots$ until some stopping criterion is satisfied:
$x^{k+1} := \mathrm{KSWP}(A, b, x^k, \Lambda)$
end algorithm

2.2 Parallelizing the Kaczmarz algorithm

This subsection presents the background that led to the Averaging Lemma, which is at the base of this paper. For huge systems of linear equations, it is essential distribute the computation among several processors that generally work in a two-step cyclic mode: in step 1, each processor works on its own allotted subtask, and in step 2, the processors exchange relevant information for the next step. The Kaczmarz algorithm, by its mathematical definition, is inherently sequential. However, in many situations, the system matrix is sparse, and so it may be possible divide the equations into blocks so that in each block, no two equations share the same variable; i.e., if a^i, a^j are two matrix rows whose equations are in the same block, then $\langle a^i, a^j \rangle = 0$; such equations are also called "independent". Given a block, all projections on hyperplanes of equations belonging to a block can be done in parallel

by several processors. After these projections the processors need to exchange information in order to process the next block of equations. This mode of parallelizing KACZ is called multi-coloring. Depending on the problem at hand, the exchange of information may or may not be a time consuming step. In the case of CT, equations derived from X-ray trajectories that are sufficiently far apart are independent, and this can be used for a parallel implementation of KACZ – see [20].

The numerical solution of differential equations over some domain involves the solution of a linear system whose unknown variables are the values of some unknown function(s) defined over some domain. A common approach to parallelizing the solution, called Domain Decomposition (DD), is to divide the domain into several subdomains, execute a solution step in each subdomain (in parallel), and then "merge" the subdomain solutions; this is repeated until some stopping criterion is satisfied. There are several approaches to the merge operation – see [15, 35, 39, 42, 47]. The merge operation is a problem in itself, and the latest approach is to use "perfectly matched layers" (PMLs), which were introduced in [3, 4] as a method for absorbing boundary conditions in wave problems, and adapted for merging subdomain solutions in DD [16, 43]. PMLs introduce certain equations in a relatively wide strip between subdomains, and thus place an extra burden on the computation. In some cases, PMLs present a particular problem at so-called cross-points, at which three or more subdomains meet, and require special treatment [6, 19, 38].

Gordon & Gordon [22] introduced a different method of parallelizing the Kaczmarz algorithm, called CARP (component-averaged row projections): the system is divided into blocks, which are not necessarily disjoint. For every variable that is common to two or more blocks, one of the blocks is determined as its "owner", and each of the other blocks operates on a "clone" of that variable. A CARP stage has two steps:

(1) Kaczmarz projections are done on every block, independently (and in parallel) of the other blocks. A shared variable is used as is in the processing of its owner block, but the processing of other blocks that share it uses its clone(s).
(2) After the separate processing of the blocks, every shared variable is averaged with all its clones, and the new value is distributed for the next stage.

The clones are new variables that help in distributing the computation among different processors. This mode of operation is, in fact, a new approach to DD, called CADD (component-averaged DD). Assume that the original problem space is $\mathbb{R}^n$ and there is a total of k clones. CARP then operates in the superspace $\mathbb{R}^{n+k}$ on a modified set of equations, because every clone replaces an original variable in some equations. It is easily seen that if CARP converges in $\mathbb{R}^{n+k}$ then the original n variables solve the original problem: every shared variable will be equal to all its clones, the modified equations are all satisfied in the superspace, and so all the original equations are satisfied by the original variables. A consequence of this is that there are no anomalies at subdomain boundaries and at cross-points.

For the formal proof of convergence of CARP, it was shown in [22] that the operation of averaging some components of a vector can be achieved by a sequence of Kaczmarz projections. This was proved in the following

Lemma 2.3. (The Averaging Lemma [22, 25]**)**
Let $x \in \mathbb{R}^n$ and $2 \leqslant m \leqslant n$. V_m is the following $(m-1)\times n$ matrix,

$$V_m = \begin{pmatrix} 1 & \text{-}1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & \text{-}2 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \text{-}3 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \\ & & & & & \vdots & & & & & \vdots & \\ 1 & 1 & 1 & 1 & 1 & \cdots & 1 & \text{-}(m\text{-}2) & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & \text{-}(m\text{-}1) & 0 & \cdots & 0 \end{pmatrix}, \tag{2.2}$$

0 *is a RHS vector of $m-1$ zeros, and $\Lambda = (1, 1, \ldots, 1)$ is a sequence of $m-1$ relaxation parameters of value one. Then*

$$\mathrm{KSWP}\,(V_m, \mathbf{0}, x, \Lambda) = \left(\underbrace{y, y, \ldots, y}_{m \text{ times}}, x_{m+1}, \ldots, x_n \right),$$

where $y = \frac{1}{m} \sum_{i=1}^{m} x_i$.

It was noted in [25] that the averaging equations are pairwise orthogonal, so they can be applied in any order. Consider now the number of equations that are added to the superspace. Suppose some variable, say x_0, has ℓ clones, so when averaging is done between x_0 and its clones, we need to average $\ell + 1$ components of a vector in the superspace. According to AL, this requires ℓ equations. Therefore, the number of additional equations in the superspace is exactly equal to the number of clones, so if the original system matrix was square, so is the corresponding superspace matrix.

There is a terminological point that needs to be explained: what is the difference between CARP (component-averaged row projections) and CAV (component-averaging) [12]? In CARP, components of a single vector (in the superspace) are averaged in order to unify partial solutions, while in CAV vectors are averaged by assigning weights that depend on the sparsity of the column components. Another point that needs mentioning is the special case when the blocks of CARP consist of single equations. This version of CARP, called CARP1 in [22], already appeared in [12, Eq. 1.9] and [11, Eq. 2.8]. A weighted version of this algorithm was studied further by Censor et al. in [10], where it is called DROP (diagonally relaxed orthogonal projections).

CARP was found to be useful on some problems of biomedical imaging such as electron tomography [17] and proton computed tomography (pCT) [34, 31]. A modified version of CARP was used for seismic tomography [33].

2.2 Accelerating Kaczmarz and CARP

One of the outcomes of this work is to enable CG acceleration of methods that involve both projections and averaging operations. This subsection presents the background of this topic and briefly summarizes the proven benefits of this approach. Kaczmarz is an extremely robust algorithm when compared against modern Krylov subspace algorithms such as CG (Conjugate Gradients) [30], GMRES [41], Bi-CGSTAB [48], even when these are combined with various preconditioners. In [22], CARP was compared against such methods, and although in one example it was the only method to converge, it was generally too slow.

In a landmark paper, Björck and Elfving [5] developed a CG acceleration of Kaczmarz, as follows: by running KACZ sequentially in one direction and then in the opposite direction, one obtains a symmetric and positive semi-definite iteration matrix, so acceleration by CG is made possible. It is not necessary to calculate this iteration matrix, instead, matrix-vector computations are done by KACZ sweeps (in the forward and backward direction). This acceleration was called CGMN. Surprisingly few works used CGMN for solving PDE problems, even though CGMN is very efficient – see [23] and the references therein.

Since CARP is KACZ in some superspace, it was natural to apply the principle of CGMN in order to accelerate it. This was done in [25]. The original CGMN assumed a fixed relaxation parameter for the KACZ projections, but in CARP, KACZ on the subdomains used some optimal relaxation parameter while the averaging equations use a relaxation parameter of 1. It was shown in [25] that the CGMN technique can be extended to cyclic relaxation parameters, and this enabled the CG acceleration of CARP, called CARP-CG. This method was found to be particularly useful on some of the most problematic cases involving the solution of partial differential equations (PDEs), namely linear systems with discontinuous coefficients and/or very large off-diagonal terms [24]. These problems include elliptic PDEs with very strong convection, the Helmholtz equation at high frequencies [26], and the elastic wave equation in the frequency domain. It was used by several researchers for solving the Helmholtz equation in exploration geophysics [49], solving the elastic wave equation in the frequency domain [37, 36], and for solving eigenvalue problems arising in quantum computations [18]. CARP-CG was also incorporated as one of the algorithms implemented in the exascale sparse solver repository [46].

3. Vector averaging

The Averaging Lemma shows how to average components of a given vector. For our purposes, we shall also need to average a set of vectors: Given $x^1, \ldots, x^m \in \mathbb{R}^n$, we wish to obtain their average vector $x = \frac{1}{m}\sum_{i=1}^m x^i$ using orthogonal projections in the superspace $\mathbb{R}^{m\times n}$. So, we define an "expansion" mapping $\mathcal{E} : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ as follows:

$$\mathcal{E}\left(x^1, \ldots, x^m\right) = \left(x_1^1, \ldots, x_n^1, \ldots, x_1^m, \ldots, x_n^m\right), \tag{3.1}$$

and a "contraction" mapping $\mathcal{C} : \mathbb{R}^{m\times n} \to \mathbb{R}^n$:

$$\mathcal{C}\left(x_1^1, \ldots, x_n^1, \ldots, x_1^m, \ldots, x_n^m\right) = \left(x_1^1, \ldots, x_n^1\right).$$

In order to obtain the average vector in $\mathbb{R}^{m\times n}$, we construct a matrix $\bar{V}$ of size $(m-1)n\times mn$, made up of copies of V_m (see Eq. (2.2)) interspersed with zeros. The first $m-1$ rows of $\bar{V}$ are comprised of V_m with $m-1$ zeros after every element of V_m. When one of these rows is multiplied by $\mathcal{E}(x)$, only the first component of each (original) vector is affected. So, after the first $m-1$ projections, all the components $x_1^i, 1 \leqslant i \leqslant m$, are replaced by their average, according to AL. The next $m-1$ rows are similar to the previous rows, but the nonzero elements are shifted by one position to the right, and so on. Eq. (3.2) is an example of $\bar{V}$ for $n = 3$ and $m = 4$.

$$\bar{V} = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -3 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -3 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -3 \end{pmatrix} \tag{3.2}$$

We can now state the following

Theorem 3.1. Vector Averaging. *Given a set of vectors $x^1, \ldots, x^m \in \mathbb{R}^n$, their average can be obtained by a Kaczmarz sweep in $\mathbb{R}^{m \times n}$:*

$$\frac{1}{m} \sum_{i=1}^{m} x^i \;=\; \mathcal{C}\left(\mathrm{KSWP}\left(\bar{V}, \mathbf{0}, \mathcal{E}\left(\mathbf{x^1}, \ldots, \mathbf{x^m}\right) \mathbf{\Lambda}\right)\right),$$

where $\mathbf{0}$ is a RHS vector of zeros, and $\mathbf{\Lambda}$ is a sequence of relaxation parameters of value 1.

4. The Cimmino-Kaczmarz equivalence

We assume that the given $m \times n$ linear system (2.1) is already normalized. The Cimmino algorithm presented here is a generalized version in which the relaxation parameters can vary with the equations and/or with the iteration number. The original algorithm presented by G. Cimmino in [13] used a constant relaxation parameter of 2 – see [2]. We introduce the following concept for ease of notation:

Definition 4.1. Given an $m \times n$ normalized linear system (2.1) and a sequence of relaxation parameters $\Lambda = (\lambda_1, \ldots, \lambda_m)$, we define an operator called a "Cimmino step", $\mathrm{CIMM}(A, b, x, \Lambda) : \mathbb{R}^n \to \mathbb{R}^n$, as follows. For $1 \leqslant i \leqslant m$, let $y^i = x + \lambda_i \left(b_i - \langle a^i, x \rangle\right) a^i$, then

$$\mathrm{CIMM}(A, b, x, \Lambda) = \frac{1}{m} \sum_{i=1}^{m} y^i = x + \frac{1}{m} \sum_{i=1}^{m} \lambda_i \left(b_i - \langle a^i, x \rangle\right) a^i. \tag{4.1}$$

Algorithm 4.2. (Cimmino with variable relaxation parameters)
Let $Ax = b$ be an $m \times n$ normalized linear system, $\Lambda^0, \Lambda^1, \ldots$ a sequence such that $\Lambda^k = (\lambda_1^k, \ldots, \lambda_m^k)$, and $x^0 \in \mathbb{R}^n$ an initial iterate.
begin algorithm
for $k = 0, 1, \ldots$ until some stopping criterion is satisfied:
$x^{k+1} := \mathrm{CIMM}(A, b, x^k, \Lambda^k)$
end algorithm

In the rightmost RHS of Eq. (4.1), we can actually consider the fraction $1/m$ to be a part of the relaxation parameter, which can also change with the step k.

Hence, we can rewrite the general step of Algorithm 4.2 as follows

$$x^{k+1} \;=\; x^k + \sum_{i=1}^{m} \mu_i^k \left(b_i - \langle a^i, x\rangle\right) a^i. \tag{4.2}$$

Consider the BIP algorithm of Aharoni and Censor [1]. BIP for linear equations has the following formulation for a normalized system (see [10, Alg. 1.3]):

$$x^{k+1} \;=\; x^k + \lambda_k \sum_{i=1}^{m} w_i^k \left(b_i - \langle a^i, x\rangle\right) a^i. \tag{4.3}$$

λ_k is superfluous in Eq. (4.3) because it can be considered as part of w_i^k, so equations (4.2) and (4.3) are actually identical. Hence, Cimmino with variable relaxation parameters is identical to BIP. See [1, Thm. 1] for details.

It should be noted that the convergence of Cimmino for the convex feasibility problem was proved by Combettes in [14], by using a method of alternating directions in a product space. Another point regarding Cimmino is that its CG acceleration is well known. Moreover, it was shown in [24] that this CG acceleration is actually the CGNR algorithm [40, §8.3.1] on the *normalized* system $Ax = b$.

We shall show that the basic Cimmino step, Eq. (4.1), is equivalent to a sequence of Kaczmarz projections in the superspace $\mathbb{R}^{m\times n}$. For this purpose we will use an expansion mapping $\mathcal{E}$ that is different from the one in Eq. (3.1); $\mathcal{E}$ replicates a single vector m times.

Definition 4.3. We define $\mathcal{E} : \mathbb{R}^n \to \mathbb{R}^{m\times n}$ as

$$\mathcal{E}\left(x_1, \ldots, x_n\right) = \left(\underbrace{\overbrace{x_1, \ldots, x_n}, \overbrace{x_1, \ldots, x_n}, \ldots, \overbrace{x_1, \ldots, x_n}}_{m \text{ times}}\right). \tag{4.4}$$

Theorem 4.4. (The Cimmino-Kaczmarz Equivalence)
Given an $m\times n$ normalized linear system $Ax = b$ and a sequence $\Lambda = (\lambda_1, \ldots \lambda_m)$ of relaxation parameters, we can construct a $2m\times mn$ linear system $\hat{A}x = \hat{b}$ and a sequence $\hat{\Lambda}$ of $2m$ relaxation parameters so that for any $x \in \mathbb{R}^n$,

$$\mathrm{CIMM}(A, b, x, \Lambda) \;=\; \mathcal{C}\left(\mathrm{KSWP}\left(\hat{A}, \hat{b}, \mathcal{E}(x), \hat{\Lambda}\right)\right),$$

i.e., a single CIMM *step in $\mathbb{R}^n$ can be obtained by applying a single* KSWP *operation in the superspace $\mathbb{R}^{m\times n}$, and contracting the result to $\mathbb{R}^n$.*

Proof. We will denote by a_j^i the element of A in row i and column j, and the ith row of A by a^i. We first construct a matrix $\bar{A}$ corresponding to A as follows: the first row of $\bar{A}$ is the first row of A, followed by $(m-1)n$ zeros. Starting from the second row, every row of $\bar{A}$ is obtained by shifting the corresponding row of A to the right by n positions w.r.t. the previous row of $\bar{A}$, and filling the vacant positions with zeros. This gives us the following $m\times mn$ matrix:

$$\bar{A} \;=\; \begin{pmatrix} a_1^1 \ldots a_n^1\ 0 \ldots\ldots\ldots\ldots 0 \\ 0 \ldots . 0\ a_1^2 \ldots a_n^2\ 0 \ldots\ldots . 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0 \ldots\ldots\ldots\ldots 0\ a_1^m \ldots a_n^m \end{pmatrix} \tag{4.5}$$

The ith row of $\bar{A}$ will be denoted by $\bar{a}^i$. Next, we construct $\hat{A}$ and $\hat{b}$ as follows:

$$\hat{A} \;=\; \begin{pmatrix} \bar{A} \\ \bar{V} \end{pmatrix}, \quad \hat{b} \;=\; \begin{pmatrix} b \\ \mathbf{0} \end{pmatrix},$$

where $\bar{V}$ is the $(m-1)n \times mn$ matrix for averaging vectors, as in §3, and $\mathbf{0}$ is a suitable vector of zeros. The required relaxation parameters are given by

$$\hat{\Lambda} = (\lambda_1, \ldots, \lambda_m, \underbrace{1, 1, \ldots, 1}_{(m-1)n \text{ times}}).$$

Let $y^0 = \mathcal{E}(x)$, and consider the first m projections of KSWP$(\hat{A}, \hat{b}, y^0, \hat{\Lambda})$:

$$\begin{aligned} \text{step } 1: \; y^1 &= y^0 + \lambda_1(b_1 - \langle \bar{a}^1, y^0 \rangle)\bar{a}^1 \\ \text{step } 2: \; y^2 &= y^1 + \lambda_2(b_2 - \langle \bar{a}^2, y^1 \rangle)\bar{a}^2 \\ &\vdots \\ \text{step } m: \; y^m &= y^{m-1} + \lambda_m(b_m - \langle \bar{a}^m, y^{m-1} \rangle)\bar{a}^m \end{aligned}$$

It is clear from the structure of $\bar{A}$ that after step 1, y^1 differs from y^0 only in the first n elements. Additionally, due to the structure of $y^0 = \mathcal{E}(x)$ and $\bar{A}$, we also have $\langle \bar{a}^1, y^0 \rangle = \langle a^1, x \rangle$, so step 1 can be replaced by

$$\text{step } 1': \; y^1 = y^0 + \lambda_1(b_1 - \langle a^1, x \rangle)\bar{a}^1.$$

Consider now the inner product $\langle \bar{a}^2, y^1 \rangle \bar{a}^2$ in step 2: the elements of y^1 that take part in the inner product with $\bar{a}^2$ are $y^1_{m+1}, \ldots, y^1_{2m}$, and, by the previous comment, these are identical to the corresponding elements in $y^0 = \mathcal{E}(x)$, which are $x_1, \ldots, x_n$. Hence, $\langle \bar{a}^2, y^1 \rangle \bar{a}^2 = \langle \bar{a}^2, y^0 \rangle \bar{a}^2$. And, by a similar argument to the above, we also have $\langle \bar{a}^2, y^0 \rangle = \langle a^2, x \rangle$, so step 2 can be replaced by

$$\text{step } 2': \; y^2 = y^1 + \lambda_2(b_2 - \langle a^2, x \rangle)\bar{a}^2.$$

Continuing similarly with all the steps, we can replace the first m steps by

$$\text{step } i': \; y^i = y^{i-1} + \lambda_i(b_i - \langle a^i, x \rangle)\bar{a}^i, \; 1 \leqslant i \leqslant m.$$

Hence, by sequential substitutions, we get

$$y^m \;=\; y^0 + c_1\bar{a}^1 + \cdots + c_m\bar{a}^m \;=\; \begin{pmatrix} x_1 + c_1 a^1_1 \\ \vdots \\ x_n + c_1 a^1_n \\ \vdots \\ x_1 + c_m a^m_1 \\ \vdots \\ x_n + c_m a^m_n \end{pmatrix},$$

where $c_i = \lambda_i(b_i - \langle a^i, x \rangle)$ for $1 \leqslant i \leqslant m$.

The next $(m-1)m$ projections are done with the rows of $\bar{V}$ (as the second part of $\hat{A}$), and these are the vector averaging projections of §3. So we get

$$y^{m+(m-1)n} \;=\; \begin{pmatrix} \frac{1}{m}\sum_{i=1}^{m}\left(x+c_i a^i\right) \\ \vdots \\ \frac{1}{m}\sum_{i=1}^{m}\left(x+c_i a^i\right) \end{pmatrix}.$$

By applying the contraction mapping we get

$$\mathcal{C}\left(y^{m+(m-1)n}\right) \;=\; x+\frac{1}{m}\sum_{i=1}^{m}\lambda_i\left(b_i-\langle a^i,x\rangle\right)a^i \;=\; \mathrm{CIMM}\left(A,b,x,\Lambda\right). \quad \square$$

Weighted averaging: suppose we wish to replace the basic Cimmino equation – Eq. (4.1) – by a weighted averaging version, i.e.,

$$\mathrm{CIMM}(A,b,x,\Lambda) \;=\; \sum_{i=1}^{m}\omega_i y^i \;=\; x+\sum_{i=1}^{m}\omega_i\lambda_i\left(b_i-\langle a^i,x\rangle\right)a^i,$$

where $0\leqslant\omega_i\leqslant 1$ and $\sum_{i=1}^{m}\omega_i=1$. This can be done by replacing λ_i in Thm. 4.4 by $m\omega_i\lambda_i$, for $1\leqslant i\leqslant m$.

3.1 Subset-Cimmino

This version of Cimmino works as follows: the $m\times n$ normalized system $Ax=b$ is divided into k blocks of equations (which are not necessarily disjoint): $A^\ell x=b^\ell$, for $1\leqslant\ell\leqslant k$, and block ℓ is associated with a sequence of relaxation parameters Λ^ℓ. Formally the algorithm is as follows.

Algorithm 4.5. (Subset-Cimmino)
begin algorithm
Choose an initial iterate $x^0\in\mathbb{R}^n$.
repeat
for $\ell=1,2,\ldots,k$
$x^\ell := \mathrm{CIMM}(A^\ell,b^\ell,x^{\ell-1},\Lambda^\ell)$
$x^0 := x^k$
until x^0 solves the linear system
end algorithm

Subset-Cimmino is clearly an intermediate algorithm between Cimmino and Kaczmarz: if $k=1$, then it is identical to Cimmino, and if $k=m$, then it is identical to Kaczmarz. Proving that Subset-Cimmino is also equivalent to Kaczmarz in a superspace is a modification of our previous constructions. The required modifications may be simple or complex, depending on the relative sizes of the subsets.

The simplest case is when all the subsets are the same size, which we denote by m. In this case, the required superspace is $\mathbb{R}^{m\times n}$. The matrix $\hat{A}$ is composed of matrices $\bar{A}^\ell$, where $\bar{A}^\ell$ is made up of the rows of A^ℓ shifted w.r.t. each other, as in Eq. (4.5). The order of the matrices is the same as their order in Algorithm 4.5. Each $\bar{A}^\ell$ is followed by a suitable $\bar{V}$, as in Eq. (3.2), for the averaging. The RHS $\hat{b}$

consists of the vectors b^ℓ, followed by zeros for the averaging. The rest is obvious so we omit further details.

The general case is more complicated. Denote by m_ℓ the number of rows of A^ℓ, for $1 \leqslant \ell \leqslant k$, and let $\bar{A}_\ell$ be as above. Since the matrices are not of the same size, we need to replicate them so that all elements of the vector y^0 will be affected. Let m' be least common multiple of $m_1, \ldots, m_k$. The superspace is $\mathbb{R}^{m' \times n}$, and the expansion mapping is $y^0 = (\underbrace{x, x, \ldots, x}_{m' \text{ times}})$.

The matrix $\hat{A}$ starts with a block, denoted B^1, consisting of $\bar{A}^1$ followed by a corresponding averaging matrix $\bar{V}'$. This block will perform a CIMM operation on the first m_1 copies of x. B^1 is followed by a copy of B^1, but shifted to the right w.r.t. B_1, so that it performs a CIMM operation on the next m_1 copies of x, and so on, for m'/m_1 times. The corresponding RHS's are obvious now from the proof of Theorem 4.4.

Since m' is divisible by m_1, after processing all the B^1 blocks, y^0 will be transformed to a vector holding m' copies of CIMM(A^1, b^1, x, Λ^1). Next, $\hat{A}$ will hold m'/m_2 copies of blocks B^2 constructed in a similar way, and so on. Clearly, after all projections of $\hat{A}$ are done, the resulting vector will consist of m' copies of KSWP$(\hat{A}, \bar{b}, y^0, \Lambda)$.

Recently, a randomized version of Subset-Cimmino was included in a study on recovering band-limited signals from random sampling – see [21]. The behavior of this algorithm as a function of the number of samples was typical of the behavior of all the Kaczmarz variants on this problem. The Cimmino-Kaczmarz equivalence now explains this behavior.

3.2 CAV and BICAV

The CAV algorithm was introduced in [12], where it was found to be a useful parallelizable algorithm for image reconstruction in CT. Its basic iterative step is given by

$$x^k \;=\; x^{k-1} + \lambda_k \sum_{i=1}^{m} \frac{b_i - \langle a^i, x^{k-1} \rangle}{\sum_{\ell=1}^{n} s_\ell (a^i_\ell)^2}\, a^i,$$

where λ_k is a relaxation parameter that may vary by the iteration number, and s_ℓ is the number of nonzero elements in column ℓ.

The proof of convergence given in [12] assumes that $\lambda_k = 1$ for all k. If we denote $\alpha_i = m/\left(\sum_{\ell=1}^{m} s_\ell (a^i_\ell)^2\right)$, we get the following representation for CAV (assuming $\lambda_k = 1$):

$$x^k \;=\; x^{k-1} + \frac{1}{m} \sum_{i=1}^{m} \alpha_i \left(b_i - \langle a^i, x^{k-1} \rangle\right) a^i.$$

This representation shows that CAV is actually a Cimmino-type algorithm with fixed relaxation parameters per equation, and the relaxation parameters α_i depend on the sparsity of the matrix. The experiments done in [12] were done on the

normalized system of equations for the purpose of comparisons with KACZ and Cimmino, so we can also assume that the equations are normalized.

The BICAV algorithm [11] divides the system (2.1) into blocks, and iteratively performs one CAV operation on every block of equations. The sparsity-oriented weights of every CAV operation on a particular block take into consideration only the equations that belong to that block. Hence, BICAV is a special case of subset-Cimmino, so it is also equivalent to Kaczmarz in some superspace.

5. String averaging

String Averaging (SA) was introduced by Censor, Elfving and Herman in [9]. It can be viewed as a certain combination of Kaczmarz and Cimmino in the following sense: whereas in Cimmino there is a set of projections that are averaged, in SA, the averaging is done after several *sequences* of projections. The most elementary SA will be described in a form that will make it easy to explain its relation to Kaczmarz in the superspace $\mathbb{R}^{m\times n}$.

Consider the (normalized) system (2.1), and some integer k, which will be the number of "strings". For $1\leqslant\ell\leqslant k$, let A^ℓ be a matrix made up of some of the rows of A, not necessarily in the same order as in A, and with possible repetitions of some of the rows in the same matrix or in other matrices. For $1\leqslant\ell\leqslant k$, let b^ℓ be a RHS vector made up of the elements of b corresponding to the rows of A^ℓ. Also, for $1\leqslant\ell\leqslant k$, let Λ^ℓ be a sequence of relaxation parameters which will be used with the equations of the system $A^\ell x=b^\ell$. We also assume that we are given a sequence of weights $\mathbf{\Omega}=(\omega_1,\ldots,\omega_k)$ s.t. $0\leqslant\omega_\ell\leqslant 1$ for $1\leqslant\ell\leqslant k$ and $\sum_{\ell=1}^k\omega_\ell=1$; these weights will be used for the weighted averaging of the strings. We will use the following notations:

$$\mathcal{A}=\left(A^1,\ldots,A^k\right),\boldsymbol{\mathcal{b}}=\left(b^1,\ldots,b^k\right),\mathbf{\Lambda}=\left(\Lambda^1,\ldots,\Lambda^k\right).$$

The quadruple $(\mathcal{A},\boldsymbol{\mathcal{b}},\mathbf{\Lambda},\mathbf{\Omega})$ will be called a "string-averaging setup" for the (normalized) linear system 2.1. For $x\in\mathbb{R}^n$, we define the operator SAVG : $\mathbb{R}^n\to\mathbb{R}^n$ as the weighted average of the string results:

$$\text{SAVG}(\mathcal{A},\boldsymbol{\mathcal{b}},x,\mathbf{\Lambda},\mathbf{\Omega})\;=\;\sum_{\ell=1}^k\omega_\ell\text{KSWP}\left(A^\ell,b^\ell,x,\Lambda^\ell\right). \tag{5.1}$$

In [9], SA is also described for Bregman projections, and also with a general operator on the results of the strings instead of averaging. Before proceeding with the main result, we need the following

Lemma 5.1. *Given an $m\times n$ normalized linear system (2.1), a sequence of m relaxation parameters Λ, and some constant α, then for any $x\in\mathbb{R}^n$*

$$\text{KSWP}(A,\alpha b,\alpha x,\Lambda)\;=\;\alpha\text{KSWP}(A,b,x,\Lambda).$$

Proof. Let $x\in\mathbb{R}^n$ be given. We define two finite sequences that lead to the two KSWP terms: $y^0=x$ and $z^0=\alpha x$, and for $1\leqslant i\leqslant m$,

$$y^i=y^{i-1}+\lambda_i\left(b_i-\langle a^i,y^{i-1}\rangle\right)a^i,\;\;z^i=z^{i-1}+\lambda_i\left(\alpha b_i-\langle a^i,z^{i-1}\rangle\right)a^i$$

Claim: For $0 \leqslant i \leqslant m$, $z_i = \alpha y_i$. The claim is proved by a simple induction on i. For $i = 0$, the claim follows from the definition of y^0 and z^0. Assume that the claim is true for i, then, for $i+1$, we have

$$\begin{aligned} z^{i+1} &= z^i + \lambda_{i+1}\left(\alpha b_{i+1} - \langle a^{i+1}, z^i\rangle\right) a^{i+1} \\ &= \alpha y^i + \lambda_{i+1}\left(\alpha b_{i+1} - \langle a^{i+1}, \alpha y^i\rangle\right) a^{i+1} \\ &= \alpha\left(y^i + \lambda_{i+1}\left(b_{i+1} - \langle a^{i+1}, y^i\rangle\right) a^{i+1}\right) \\ &= \alpha y^{i+1}. \end{aligned}$$

Therefore, $z^m = \alpha y^m \Rightarrow \mathrm{KSWP}(A, \alpha b, \alpha x, \Lambda) = \alpha \mathrm{KSWP}(A, b, x, \Lambda)$. □

Theorem 5.2. *Given a string-averaging setup* $(\mathcal{A}, \boldsymbol{\mathcal{b}}, \boldsymbol{\Lambda}, \boldsymbol{\Omega})$ *for the* $m \times n$ *normalized linear system* (2.1)*, let* $\bar{m} = \sum_{\ell=1}^{k} m_\ell$*, where* m_ℓ *is the number of rows of* A_ℓ *for* $1 \leqslant \ell \leqslant k$*. Then the following four items can be constructed*

- *a matrix* $\hat{A}$*,*
- *a corresponding RHS vector* $\hat{b}$*,*
- *a suitable sequence of relaxation parameters* $\hat{\Lambda}$*,*
- *an operator* $\mathcal{E} : \mathbb{R}^n \to \mathbb{R}^{k \times n}$*,*

so that for any point $x \in \mathbb{R}^n$*,*

$$\mathrm{SAVG}\left(\mathcal{A}, \boldsymbol{\mathcal{b}}, x, \boldsymbol{\Lambda}, \boldsymbol{\Omega}\right) = \mathcal{C}\left(\mathrm{KSWP}\left(\hat{A}, \hat{b}, \mathcal{E}(x), \hat{\Lambda}\right)\right). \tag{5.2}$$

Proof. Let $x = (x_1, \ldots, x_n)$ be given. We denote $\alpha_\ell = k\omega_\ell$ for $1 \leqslant \ell \leqslant k$; these constants are needed for weighted averaging. If only regular averaging is needed, they can all be set to 1. To construct $\hat{A}$ and $\hat{b}$, we first present a matrix $\bar{A}$ and a corresponding RHS $\bar{b}$: $\bar{A}$ is a generalization of the matrix used in the proof of Theorem 4.4: instead of shifting single rows of A by n positions, the matrices A^ℓ are taken as blocks and shifted by n positions w.r.t. each other, and the vacant positions are filled with zeros. Clearly, $\bar{A}$ is of size $\bar{m} \times kn$. The structures of $\bar{A}$ and $\bar{b}$ are illustrated in Fig. 5.1.

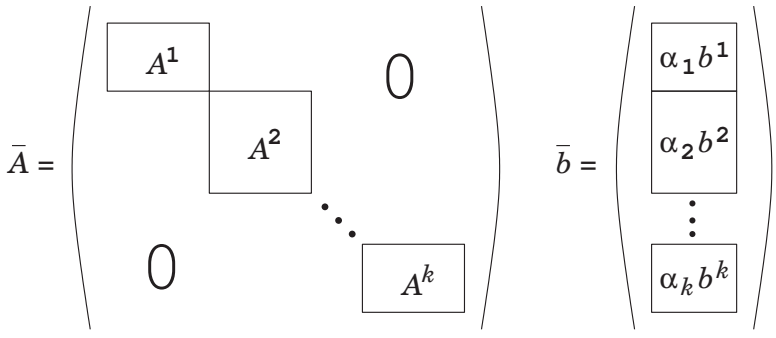

FIGURE 5.1. Matrix $\bar{A}$ and RHS $\bar{b}$ for the superspace equivalent of String Averaging, where k is the number of strings. For $1 \leqslant \ell \leqslant k$, A^ℓ consists of all the rows of A that are needed for the ℓth string and $\alpha_\ell b^\ell$ is the corresponding RHS.

We now set

$$\hat{A} = \begin{pmatrix} \bar{A} \\ \bar{V} \end{pmatrix}, \quad \hat{b} = \begin{pmatrix} \bar{b} \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

where $\bar{V}$ is the averaging matrix for vectors, as in Eq. (3.2) and the proof of Theorem 4.4, and the zeros in $\hat{b}$ are the RHS of $\bar{V}$.

The required sequence of relaxation parameters $\hat{\Lambda}$ is the concatenation of $\mathbf{\Lambda} = (\Lambda^1, \ldots, \Lambda^k)$ and a sequence of ones for the averaging operations. Similarly to the expansion mapping of Eq. (3.1), the expansion mapping $\mathcal{E}(x)$ consists of k copies of x, but now, each copy of x is multiplied by a different α factor:

$$\mathcal{E}(x_1, \ldots, x_n) = \left(\underbrace{\alpha_1 x_1, \ldots, \alpha_1 x_n}, \underbrace{\alpha_2 x_1, \ldots, \alpha_2 x_n}, \ldots, \underbrace{\alpha_k x_1, \ldots, \alpha_k x_n}\right)$$

Let $y^0 = \mathcal{E}(x)$, and consider now the operation of KSWP $\left(\hat{A}, \hat{b}, y^0, \hat{\Lambda}\right)$. The very first step is

$$y^1 = y^0 + \left(\alpha_1 b_1^1 - \langle \bar{a}^1, y^0 \rangle\right) \bar{a}^1$$

It is clear from the structure of $\bar{A}$ that y^1 differs from y^0 only in the first n elements. Also, after m_1 steps, the result, y^{m_1}, also differs from y^0 only in the first n elements. These n elements of y^{m_1} are exactly KSWP$(A^1, \alpha b^1, \alpha_1 x, \Lambda^1)$. So now we have

$$y^{m_1} = \begin{pmatrix} \text{KSWP}(A^1, \alpha_1 b^1, \alpha_1 x, \Lambda^1) \\ \alpha_2 x \\ \vdots \\ \alpha_k x \end{pmatrix}.$$

Consider now y^{m_1+1}: due to the structure of $\bar{A}$ and $\bar{b}$, y^{m_1+1} differs from y^{m_1} only in the next n elements. Eventually, after $\bar{m}$ steps, we get

$$\begin{aligned} y^{\bar{m}} &= \begin{pmatrix} \text{KSWP}(A^1, \alpha_1 b^1, \alpha_1 x, \Lambda^1) \\ \vdots \\ \text{KSWP}(A^k, \alpha_k b^k, \alpha_k x, \Lambda^k) \end{pmatrix} \\ &= \begin{pmatrix} \alpha_1 \text{KSWP}(A^1, b^1, x, \Lambda^1) \\ \vdots \\ \alpha_k \text{KSWP}(A^k, b^k, x, \Lambda^k) \end{pmatrix} \end{aligned}$$

The last equality follows from Lemma 5.1. Hence, after the averaging projections and the contraction, we get

$$\begin{aligned}\mathcal{C}\left(\mathrm{KSWP}\left(\hat{A},\hat{b},\mathcal{E}(x),\hat{\Lambda}\right)\right) &= \frac{1}{k}\sum_{\ell=1}^{k}\alpha_\ell \mathrm{KSWP}\left(A^\ell,b^\ell,x,\Lambda^\ell\right)\\ &= \frac{1}{k}\sum_{\ell=1}^{k}k\omega_\ell \mathrm{KSWP}\left(A^\ell,b^\ell,x,\Lambda^\ell\right)\\ &= \sum_{\ell=1}^{k}\omega_\ell \mathrm{KSWP}\left(A^\ell,b^\ell,x,\Lambda^\ell\right)\\ &= \mathrm{SAVG}\left(\mathcal{A},\boldsymbol{b},x,\boldsymbol{\Lambda},\boldsymbol{\Omega}\right).\end{aligned}$$

□

As noted previously, the general definition of String Averaging in [9] permits any type of operator on the string results. A case in point is CARP: the separate Kaczmarz operations on the different blocks are strings in the SA terminology, and the CARP operation of averaging components is an operator on the string results.

6. Conclusions

We have shown how projection methods which involve averaging are mathematically equivalent to the Kaczmarz algorithm in some superspace of the given problem space. Such equivalence is based on two elements:

- A structural transformation of the given linear system into a certain linear system in the superspace.
- An application of the Averaging Lemma [22, 25], according to which the averaging operations are equivalent to projections in the superspace.

Additionally, by using a modification of the transformation, weighted averaging can also done in the case of the Cimmino algorithm [13] and String Averaging [9].

Explicit constructions were given for a generalized Cimmino algorithm, with varying relaxation parameters, and String Averaging. In String Averaging, our results apply to orthogonal projections with weighted averaging. These results also hold for a so-called Subset-Cimmino, which is a block-iterative variant of Cimmino. It then follows that the results also hold for the CAV [12] and BICAV [11] algorithms.

A consequence of our results is that convergence proofs for methods that involve projections and averaging simply follow from well-known convergence proofs for the Kaczmarz algorithm. Another important consequence is that such methods can be accelerated by the Conjugate Gradients method, as in the CGMN algorithm [5, 23] and CARP-CG [25]. Note that the projections used by the Averaging Lemma use a relaxation parameter of 1, so if this is used in conjunction with different relaxation parameters, then one actually needs the cyclic version of CGMN, called CGMNC in [25]. Formal convergence proofs for such accelerated methods then follow from the known convergence of CG.

Acknowledgment

The author wishes to thank the reviewers for their helpful comments.

References

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

[2] M. Benzi, *Gianfranco Cimmino's Contributions to Numerical Mathematics*, Atti del Seminario di Analisi Matematica, Dipartimento di Matematica dell'Universita' di Bologna. Volume Speciale: Ciclo di Conferenze in Memoria di Gianfranco Cimmino, Marzo-Aprile 2004, Tecnoprint, Bologna, 2005, pp. 87–109.

[3] J.-P. Berenger, *A perfectly matched layer for the absorption of electromagnetic waves*, J. Computational Physics **114** (1994), 185–200.

[4] J.-P. Berenger, *Three-dimensional perfectly matched layer for the absorption of electromagnetic waves*, J. Computational Physics **127** (1996), 363–379.

[5] Å. Björck and T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT **19** (1979), 145–163.

[6] Y. Boubendir, A. Bendali and M. B. Fares, *Coupling of a non-overlapping domain decomposition method for a nodal finite element method with a boundary element method*, Int. l J. Numer. Methods Eng. **73** (2008), 1624–1650.

[7] C. Cenker, H. Feichtinger, M. Mayer, H. Steier and T. Strohmer, *New variants of the POCS method using affine subspaces of finite codimension, with applications to irregular sampling.* in: Visual Communications and Image Processing '92, P. Maragos (ed.), SPIE, Nov. 1992, pp. 299–310.

[8] Y. Censor, P. P. B. Eggermont and D. Gordon, *Strong underrelaxation in Kaczmarz's method for inconsistent systems*, Numerische Mathematik **41** (1983), 83–92.

[9] Y. Censor, 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.), vol. 8, Studies in Computational Mathematics, Elsevier, Amsterdam, 2001, pp. 101–113.

[10] Y. Censor, T. Elfving, G. T. Herman and T. Nikazad, *On diagonally relaxed orthogonal projection methods*, SIAM J. Scientific Computing **30** (2008), 473–504.

[11] Y. Censor, D. Gordon and R. Gordon, *BICAV: A block-iterative parallel algorithm for sparse systems with pixel-dependent weighting*, IEEE Trans. Medical Imaging **20** (2001), 1050–1060.

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

[13] G. Cimmino, *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari* La Ricerca Scientifica XVI, Series II, Anno IX **1** (1938), 326–333.

[14] P. L. Combettes, *Inconsistent signal feasibility problems: least-squares solutions in a product space*, IEEE Trans. on Signal Processing **42** (1994), 2955–2966.

[15] V. Dolean, P. Jolivet and F. Nataf, *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*, SIAM, Philadelphia, PA, 2015.

[16] B. Engquist and L. Ying, *Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers*, Multiscale Modeling & Simulation **9** (2011), 686–710.

[17] J.-J. Fernández, D. Gordon and R. Gordon, *Efficient parallel implementation of iterative reconstruction algorithms for electron tomography*, J. Parallel & Distributed Computing **68** (2008), 626–640.

[18] M. Galgon, L. Krämer, J. Thies, A. Basermann and B. Lang, *On the parallel iterative solution of linear systems arising in the FEAST algorithm for computing inner eigenvalues*, Parallel Computing **49** (2015), 153–163.

[19] M. J. Gander and F. Kwok, *Optimized Schwarz methods at cross points*, SIAM J. Scientific Computing **34** (2012), A1849–A1879.

[20] D. Gordon, *Parallel ART for image reconstruction in CT using processor arrays*, Int. J. Parallel, Emergent & Distributed Systems **21** (2006), 365–380.

[21] D. Gordon, *A derandomization approach to recovering bandlimited signals across a wide range of random sampling rates*, Numerical Algorithms, online, June 2017. DOI: 10.1007/s11075-017-0356-3.

[22] D. Gordon and R. Gordon, *Component-averaged row projections: A robust, block-parallel scheme for sparse linear systems*, SIAM J. Scientific Computing **27** (2005), 1092–1117.

[23] D. Gordon and R. Gordon, *CGMN revisited: robust and efficient solution of stiff linear systems derived from elliptic partial differential equations*, ACM Trans. Math. Soft. **35** (2008), 18:1–18:27.

[24] D. Gordon and R. Gordon, *Solution methods for linear systems with large off-diagonal elements and discontinuous coefficients*, Computer Modeling Engineering & Sciences **53** (2009), 23–45.

[25] D. Gordon and R. Gordon, *CARP-CG: a robust and efficient parallel solver for linear systems, applied to strongly convection-dominated PDEs*, Parallel Computing **36** (2010), 495–515.

[26] D. Gordon and R. Gordon, *Parallel solution of high frequency Helmholtz equations using high order finite difference schemes*, Applied Math. Comp. **218** (2012), 10737–10754.

[27] G. T. Herman, *Fundamentals of Computerized Tomography: Image Reconstruction From Projections*, Springer, 2nd edition, 2009.

[28] G. T. Herman, A. Lent and P. H. Lutz, *Relaxation methods for image reconstruction*, Communications ACM **21** (1978), 152–158.

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

[30] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. National Bureau Standards **49** (1952), 409–436.

[31] R. P. Johnson, *Review of medical radiography and tomography with proton beams*, Reports on Progress in Physics, accepted, 2017.

[32] S. Kaczmarz, *Angenäherte Auflösung von Systemen linearer Gleichungen*, Bulletin de l'Académie Polonaise des Sciences et Lettres **A35** (1937), 355–357.

[33] G. Kamath, L. Shi, W.-Z. Song and J. Lees, *Distributed travel-time seismic tomography in large-scale sensor networks*, J. Parallel & Distributed Computing **89** (2016), 50–64.5, 2016.

[34] N. T. Karonis, K. L. Duffin, C. E. Ordoñez, B. Erdelyi, T. D. Uram, E. C. Olson, G. Coutrakon and M. E. Papka, *Distributed and hardware accelerated computing for clinical medical imaging using proton computed tomography (pCT)*. J. Parallel & Distributed Computing **73** (2013), 1605–1612.

[35] J. Kruis, *Domain Decomposition Methods for Distributed Computing*, Saxe-Coburg Publications, 2007.

[36] Y. Li, B. Han, L. Métivier and R. Brossier, *Optimal fourth-order staggered-grid finite-difference scheme for 3D frequency-domain viscoelastic wave modeling*, J. Computational Physics **321** (2016), 1055–1078.

[37] Y. Li, L. Métivier, R. Brossier, B. Han and J. Virieux, *2D and 3D frequency-domain elastic wave modeling in complex media with a parallel iterative solver*, Geophysics **80** (2015), 101–118.

[38] S. Loisel, *Condition number estimates for the nonoverlapping optimized Schwarz method and the 2-Lagrange multiplier method for general domains and cross points*, SIAM J. Numer. Anal. **51** (2013), 3062–3083.

[39] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, Oxford, 1999.

[40] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, 2nd. edition, 2003.

[41] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Scientific Statistical Computing **7** (1986), 856–869.

[42] B. Smith, P. Bjørstad and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Diffferential Equations*, Cambridge University Press, Cambridge, UK, 1996.

[43] C. C. Stolk, *A rapidly converging domain decomposition method for the Helmholtz equation*, J. Computational Physics **241** (2013), 240–252.

[44] T. Strohmer and R. Vershynin, *A randomized Kaczmarz algorithm with exponential convergence*, J. Fourier Anal. Appl. **15** (2009), 262–278.

[45] K. Tanabe, *Projection method for solving a singular system of linear equations and its applications*, Numerische Mathematik **17** (1971), 203–214.
[46] J. Thies, M. Galgon, F. Shahzad, A. Alvermann, M. Kreutzer, A. Pieper, M. Röhrig-Zöllner, A. Basermann, H. Fehske, G. Hager, B. Lang and G. Wellein, *Towards an exascale enabled sparse solver repository*, in: Software for Exascale Computing – SPPEXA 2013-2015, H. J. Bungartz, P. Neumann, and W. Nagel (eds), vol. 113, Lecture Notes in Computational Science and Engineering, Springer, Cham, Switzerland, 2016, pp. 73–89.
[47] A. Toselli and O. B. Widlund, *Domain Decomposition Methods – Algorithms and Theory*, vol. 34, Springer Series in Computational Mathematics, Springer, Berlin, 2005.
[48] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Scientific Statistical Computing **13** (1992), 631–644.
[49] T. van Leeuwen and F. J. Herrmann, *3D frequency-domain seismic inversion with controlled sloppiness*, SIAM J. Scientific Computing **36** (2014), 192–217.

Manuscript received January 11 2018
revised February 14 2018

Dan Gordon
Dept. of Computer Science, University of Haifa, Haifa 3498838, Israel
E-mail address: gordon@cs.haifa.ac.il