A Kaczmarz Algorithm for Solving Tree Based Distributed Systems of Equations

Chinmay Hegde, Fritz Keinert, and Eric S. Weber

Abstract The Kaczmarz algorithm is an iterative method for solving systems of linear equations. We introduce a modified Kaczmarz algorithm for solving systems of linear equations in a distributed environment, i.e., the equations within the system are distributed over multiple nodes within a network. The modification we introduce is designed for a network with a tree structure that allows for passage of solution estimates between the nodes in the network. We prove that the modified algorithm converges under no additional assumptions on the equations. We demonstrate that the algorithm converges to the solution, or the solution of minimal norm, when the system is consistent. We also demonstrate that in the case of an inconsistent system of equations, the modified relaxed Kaczmarz algorithm converges to a weighted least squares solution as the relaxation parameter approaches 0.

1 Introduction

The Kaczmarz method [16] is an iterative algorithm for solving a system of linear equations $A\mathbf{x} = \mathbf{b}$, where A is an $m \times k$ matrix. Written out, the equations are $\mathbf{a}_i^* \mathbf{x} = b_i$ for $i = 1, \ldots, m$, where $\mathbf{a}_i^*$ is the ith row of the matrix A. Given a solution guess $\mathbf{x}^{(n-1)}$ and an equation number i, we calculate $r_i = b_i - \mathbf{a}_i^* \mathbf{x}^{(n-1)}$ (the residual for

Chinmay Hegde
Electrical and Computer Engineering, Iowa State University, Ames, IA 50011, e-mail: chinmayhegde@gmail.com, current: Electrical and Computer Engineering, New York University, New York, NY 10012

Fritz Keinert
Department of Mathematics, Iowa State University, 396 Carver Hall, Ames, IA 50011 e-mail: keinert@iastate.edu

Eric S. Weber
Department of Mathematics, Iowa State University, 396 Carver Hall, Ames, IA 50011 e-mail: esweber@iastate.edu

equation i), and define

$$\mathbf{x}^{(n)} = \mathbf{x}^{(n-1)} + \frac{r_i}{\|\mathbf{a}_i\|^2}\mathbf{a}_i. \tag{1}$$

This makes the residual of $\mathbf{x}^{(n)}$ in equation i equal to 0. Here and elsewhere, $\|\cdot\|$ is the usual Euclidean (ℓ^2) norm. We iterate repeatedly through all equations (i.e. we consider $\lim_{n\to\infty}\mathbf{x}^{(n)}$ where $n \equiv i \mod m$, so the equations are repeated cyclically). Kaczmarz proved that if the system of equations has a unique solution , then $\mathbf{x}^{(n)}$ converges to that solution. Later, it was proved in [33] that if the system is consistent (but the solution is not unique), then the sequence converges to the solution of minimal norm . Likewise, it was proved in [7, 21] that if inconsistent, a relaxed version of the algorithm can provide approximations to a weighted least-squares solution .

Obtaining the nth estimate requires knowledge only of the i-th equation ($n \equiv i \mod m$ as above) and the $n-1$-st estimate. We suppose that the equations are indexed by the nodes of a tree, representing a network in which the equations are distributed over many nodes. In our distributed Kaczmarz algorithm, solution estimates can only be communicated when there exists an edge between the nodes. The estimates for the solution will disperse through the tree, which results in several different estimates of the solution. When these estimates then reach the leaves of the tree, they are pooled together into a single estimate. Using this single estimate as a seed, the process is repeated, with the goal that the sequence of single estimates will converge to the true solution. We illustrate the dispersion and pooling processes in Figure 1.

1.1 Notation

For linear transformations T, we denote by $\mathcal{N}(T)$ and $\mathcal{R}(T)$ the kernel (nullspace) and range, respectively. We use $\rho(T)$ to denote the spectral radius .

When $\mathbf{a}^* = \mathbf{a}_v^*$ corresponds to a row of the matrix A indexed by a node v, we will denote the linear projection onto the subspace $\mathbf{a}_v^*\mathbf{z} = 0$ by:

$$P_v(\mathbf{z}) = \left(I - \frac{\mathbf{a}_v\mathbf{a}_v^*}{\mathbf{a}_v^*\mathbf{a}_v}\right)(\mathbf{z}) \tag{2}$$

and the affine projection onto the linear manifold $\mathbf{a}_v^*\mathbf{z} = b_v$ by:

$$Q_v(\mathbf{z}) = P_v(\mathbf{z}) + \mathbf{h}_v \tag{3}$$

where $\mathbf{h}_v$ is the vector that satisfies $\mathbf{a}_v^*\mathbf{h}_v = b_v$ and is in $\mathcal{N}(P_v)$.

A tree is a connected graph with no cycles. We denote arbitrary nodes (vertices) of a tree by v, u. Our tree will be rooted; the root of the tree is denoted by r. Following the notation from MATLAB, when v is on the path from r to u, we will say that v is a predecessor of u and write $u \prec v$. Conversely, u is a successor of v. By immediate successor of v we mean a successor u such that there is an edge between v and u (this

is referred to as a *child* in graph theory parlance [35]). Similarly, v is an immediate predecessor (i.e. *parent*). We denote the set of all immediate successors of node v by $C(v)$. A node without a successor is called a leaf; leaves of the tree are denoted by ℓ. We will denote the set of all leaves by $\mathcal{L}$. Often we will have need to enumerate the leaves as $\ell_1, \ldots, \ell_t$, hence t denotes the number of leaves.

A weight w is a nonnegative function on the edges of the tree; we denote this by $w(u, v)$, where u and v are nodes that have an edge between them. We assume $w(u, v) = w(v, u)$, though we will typically write $w(u, v)$ when $u \prec v$. When $u \prec v$, but u is not a immediate successor, we write

$$w(u, v) := \prod_{j=1}^{J-1} w(u_{j+1}, u_j) \tag{4}$$

where $v = u_1, \ldots, u_J = u$ is a path from v to u.

When the system of equations $A\mathbf{x} = \mathbf{b}$ has a unique solution, we will denote this by $\mathbf{x}^S$. When the system is consistent but the solution is not unique, we denote the solution of minimal norm by $\mathbf{x}^M$, which is given by

$$\mathbf{x}^M = \text{argmin } \{\|\mathbf{x}\| : A\mathbf{x} = \mathbf{b}\}. \tag{5}$$

1.2 The Distributed Kaczmarz Algorithm

The iteration begins with an estimate (say $\mathbf{x}^{(n)}$) at the root of the tree. When node u receives from its immediate predecessor v an input estimate $\mathbf{x}_v^{(n)}$, it generates a new estimate via the Kaczmarz update:

$$\mathbf{x}_u^{(n)} = \mathbf{x}_v^{(n)} + \frac{r_u(\mathbf{x}_v^{(n)})}{\|\mathbf{a}_u\|^2}\mathbf{a}_u, \tag{6}$$

where the residual is given by

$$r_u(\mathbf{x}_v^{(n)}) := b_u - \mathbf{a}_u^* \mathbf{x}_v^{(n)}. \tag{7}$$

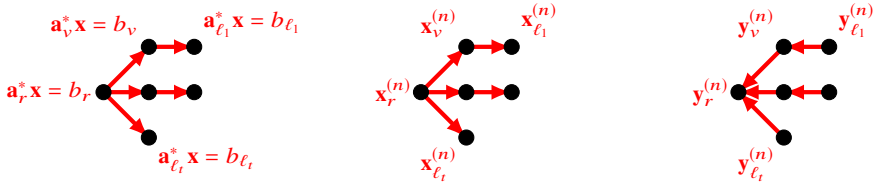
The root node updates the estimate that begins the iteration using its equation: $\mathbf{x}_r^{(n)} = \mathbf{x}^{(n)} + \frac{r_r(\mathbf{x}^{(n)})}{\|\mathbf{a}_r\|^2}\mathbf{a}_r$. Node u then passes this estimate to all of its immediate successors, and the process is repeated recursively. We refer to this as the *dispersion stage* . Once this process has finished, each leaf ℓ of the tree now possesses an estimate: $\mathbf{x}_\ell^{(n)}$.

The next stage, which we refer to as the *pooling stage* , proceeds as follows. For each leaf, set $\mathbf{y}_\ell^{(n)} = \mathbf{x}_\ell^{(n)}$. Each node v calculates an updated estimate as:

$$\mathbf{y}_v^{(n)} = \sum_{u \in C(v)} w(u,v)\mathbf{y}_u^{(n)}, \tag{8}$$

subject to the constraints that $w(u,v) > 0$ when $u \in C(v)$ and $\sum_{u \in C(v)} w(u,v) = 1$. This process continues until reaching the root of the tree, resulting in the estimate $\mathbf{y}_r^{(n)}$.

We set $\mathbf{x}^{(n+1)} = \mathbf{y}_r^{(n)}$, and repeat the iteration. The updates in the dispersion stage (Equation 6) and pooling stage (Equation 8) are illustrated in Figure 1.



(a) equations distributed across nodes

(b) updates disperse through nodes

(c) updates pool and pass to next iteration

Fig. 1 Illustration of updates in the distributed Kaczmarz algorithm with measurements indexed by nodes of the tree.

We note that the tree topology is fixed *a priori*, and remains fixed over all iterations.

1.3 Related Work

The Kaczmarz method was originally introduced in [16]. It became popular with the introduction of Computer Tomography, under the name of ART (Algebraic Reconstruction Technique). ART added non-negativity and other constraints to the standard algorithm [8]. Other variations on the Kaczmarz method allowed for relaxation parameters [33], re-ordering equations to speed up convergence [11], or considering block versions of the Kaczmarz method with relaxation matrices Ω_i [7]. Relatively recently, choosing the next equation randomly has been shown to dramatically improve the rate of convergence of the algorithm [32, 40, 28]. Moreover, this randomized version of the Kaczmarz algorithm has been shown to be comparable to the gradient descent method [26]. Recent advances in accelerating the Kaczmarz method include subsampling techniques, meaning subsampling the rows of the matrix [27], or sketching the full matrix with a preconditioner [9].

Our version of the Kaczmarz method differs from these versions in the following crucial sense. Each node has access to only its equation. Therefore, the next equation cannot be chosen randomly or otherwise, since the ordering of the equations is determined *a priori* by the network topology and thus is different from all randomized versions. Similarly, the block versions and sketched versions require access to several (but not necessarily all) of the rows simultaneously; this is also prohibited in our

distributed context. Our version here is most similar to the Cimmino method [6], which was extended in [3], as well as the greedy method given in [23]. Both of these methods involve averaging estimates, in addition to applying the Kaczmarz update, as we do here. The proofs in [23] require the system to be consistent, which we do not, and the method (and proofs) in [3] require access to *columns* of the matrix as well as rows. Because of these differences, we make no direct comparisons.

The situation we consider in the present paper can be considered a distributed estimation problem . Such problems have a long history in applied mathematics, control theory, and machine learning. At a high level, similar to our approach, they all involve averaging local copies of the unknown parameter vector interleaved with update steps [34, 36, 31, 2, 25, 15, 38, 29, 39, 30]. Recently, a number of protocols for gossip methods , including a variation of the Kaczmarz method, was analyzed in [20]. The protocols analyzed in that paper require the system to be consistent for convergence guarantees.

Following [38], a consensus problem takes the following form. Consider the problem of minimizing:

$$F(\mathbf{x}) = \sum_{v=1}^{m} f_v(\mathbf{x}),$$

where f_v is a function that is known (and private) to node v in the graph. Then, one can solve this minimization problem using decentralized gradient descent, where each node updates its estimate of $\mathbf{x}$ (say $\mathbf{x}_v$) by combining the average of its neighbors with the negative gradient of its local function f_v:

$$\mathbf{x}_v^{(n+1)} = \frac{1}{\deg v} \sum_u m(v,u)\mathbf{x}_u^{(n)} - \omega \nabla f_v(\mathbf{x}_v^{(n)}),$$

where $M = (m(v,u)) \in \{0,1\}^{m \times m}$ represents the adjacency matrix of the graph. Specializing $f_v(\mathbf{x}) = c_v(b_v - \mathbf{a}_v^* \mathbf{x})^2$ yields our least-squares estimation problem that we establish in Theorem 4 (where c_v is a fixed weight for each node).

However, our version of the Kaczmarz method differs from previous work in a few aspects: (i) we assume an *a priori* fixed tree topology (which is more restrictive than typical gossip algorithms); (ii) there is no master node as in parallel algorithms, and no shared memory architecture; (iii) as we will emphasize in Theorem 4, we make no strong convexity assumptions (which is typically needed for distributed optimation algorithms, but see [24, 22] for a relaxation of this requirement); and (iv) we make no assumptions on the matrix A, in particular we do not assume that it is nonnegative.

On the other end of the spectrum are algorithms that distribute a computational task over many processors arranged in a fixed network. These algorithms are usually considered in the context of parallel processing, where the nodes of the graph represent CPUs in a highly parallelized computer. See [1] for an overview.

The algorithm we are considering does not really fit either of those categories. It requires more structure than the gossip algorithms, but each node depends on results from other nodes, more than the usual distributed algorithms.

This was pointed out in [1]. For iteratively solving a system of linear equations, a Successive Over-Relaxation (SOR) variant of the Jacobi method is easy to parallelize; standard SOR, which is a variation on Gauss-Seidel, is not. The authors also consider what they call the *Reynolds method*, which is similar to a Kaczmarz method with all equations being updated simultaneously. Again, this method is easy to parallelize. A sequential version called RGS (Reynolds Gauss-Seidel) can only be parallelized in certain settings, such as the numerical solution of PDEs.

A distributed version of the Kaczmarz algorithm was introduced in [17]. The main ideas presented there are very similar to ours: updated estimates are obtained from prior estimates using the Kaczmarz update with the equations that are available at the node, and distributed estimates are averaged together at a single node (which the authors refer to as a fusion center, for us it is the root of the tree). In [17], the convergence analysis is limited to the case of consistent systems of equations, and inconsistent systems are handled by Tikhonov regularization [14, 12] rather than by varying the relaxation parameter. Another distributed version was proposed in [19], which has a shared memory architecture.

Finally, the Kaczmarz algorithm has been proposed for online processing of data in [13, 5]. In these papers, the processing is online, so neither distributed nor parallel.

2 Analysis of the Kaczmarz Algorithm for Tree Based Distributed Systems of Equations

In this section, we will demonstrate that the Kaczmarz algorithm for tree based equations as defined in Equations (6) and (8) converges. We consider three cases separately: (i) the system is consistent and the solution is unique; (ii) the system is consistent but there are many solutions; and (iii) the system is inconsistent. In Subsection 2.1, we prove that for case (i) the algorithm converges to the solution, and in Subsection 2.2, we prove that for case (ii) the algorithm converges to the solution of minimal norm. Also in Subsection 2.2, we introduce the relaxed version of the update in Equation (6). We prove that for every relaxation parameter $\omega \in (0, 2)$, the algorithm converges to the solution of minimal norm. Then in Subsection 2.3, we prove that for case (iii) the algorithm converges to a generalized solution $\mathbf{x}(\omega)$ which depends on ω, and $\mathbf{x}(\omega)$ converges to a weighted least-squares solution as $\omega \to 0$.

2.1 Systems with Unique Solutions

For our analysis, we need to trace the estimates through the tree. Suppose that the tree has t leaves; for each leaf ℓ, let $p_\ell - 1$ denote the length of the path between the root r and the leaf ℓ. We will denote the nodes on the path from r to ℓ by $r = (\ell, 1), (\ell, 2), \ldots, (\ell, p_\ell) = \ell$. During the dispersion stage, we have for $p = 2, \ldots, p_\ell$:

$$\mathbf{x}_{\ell,p}^{(n)} = \mathbf{x}_{\ell,p-1}^{(n)} + \left(\frac{r_{\ell,p}(\mathbf{x}_{\ell,p-1}^{(n)})}{\|\mathbf{a}_{\ell,p}\|^2} \right) \mathbf{a}_{\ell,p}.$$

Then at the beginning of the pooling stage, we have the estimates $\mathbf{y}_\ell^{(n)} := \mathbf{x}_\ell^{(n)}$ (we denote $\mathbf{x}_\ell^{(n)} := \mathbf{x}_{\ell,p_\ell}^{(n)}$ and $\mathbf{y}_\ell^{(n)} := \mathbf{y}_{\ell,p_\ell}^{(n)}$). These estimates then pool back at the root as follows (the proof is a straightforward induction argument):

Lemma 1 *The estimate at the root at the end of the pooling stage is given by:*

$$\mathbf{y}_r^{(n)} = \sum_{\ell \in \mathcal{L}} w(\ell, r) \mathbf{y}_\ell^{(n)}.$$

Note that also by induction, we have that

$$\sum_{\ell \in \mathcal{L}} w(\ell, r) = 1. \tag{9}$$

Theorem 1 *Suppose that the equation $A\mathbf{x} = \mathbf{b}$ has a unique solution, denoted by $\mathbf{x}^S$. There exists a constant $\alpha < 1$, such that*

$$\|\mathbf{x}^S - \mathbf{x}^{(n+1)}\| \leq \alpha \|\mathbf{x}^S - \mathbf{x}^{(n)}\|.$$

Consequently,

$$\lim_{n \to \infty} \mathbf{x}^{(n)} = \mathbf{x}^S,$$

and the convergence is linear in order.

Proof Along any path from the root r to the leaf ℓ, the dispersion stage is identical to the classical Kaczmarz algorithm, and so we can write (see [18]):

$$\mathbf{x}^S - \mathbf{x}_\ell^{(n)} = P_{\ell,p_\ell}(\mathbf{x}^S - \mathbf{x}_{\ell,p_\ell-1}^{(n)}) = P_{\ell,p_\ell} \dots P_{\ell,2} P_{\ell,1}(\mathbf{x}^S - \mathbf{x}^{(n)}), \tag{10}$$

from which it follows immediately that

$$\|\mathbf{x}^S - \mathbf{x}_\ell^{(n)}\| \leq \|\mathbf{x}^S - \mathbf{x}^{(n)}\|. \tag{11}$$

We claim that unless $\mathbf{x}^S = \mathbf{x}^{(n)}$, we must have a strict inequality for at least one leaf, say ℓ_0. Indeed, suppose to the contrary that for every leaf ℓ, we had equality in Equation (11), then by Equation (10), we must have for every node $v = (\ell, k)$ in the path from the root r to the leaf ℓ:

$$P_v(\mathbf{x}^S - \mathbf{x}^{(n)}) = \mathbf{x}^S - \mathbf{x}^{(n)}. \tag{12}$$

Therefore, we obtain

$$\mathbf{a}_v^*(\mathbf{x}^S - \mathbf{x}^{(n)}) = 0 \text{ for all nodes } v. \tag{13}$$

By our assumption that the equation has a unique solution, we obtain that $\mathbf{x}^S - \mathbf{x}^{(n)} = 0$.

By Equations (9) and (11) and our previous claim, we have

$$\|\mathbf{x}^S - \mathbf{x}^{(n+1)}\| < \sum_{\ell \in \mathcal{L}} w(\ell, r)\|\mathbf{x}^S - \mathbf{x}^{(n)}\| = \|\mathbf{x}^S - \mathbf{x}^{(n)}\|. \tag{14}$$

By continuity and compactness, there is a uniform constant α less than 1 that satisfies the claim. This completes the proof. □

As we shall see in the sequel, we can interpret the above proof in the following way: define the mapping

$$\mathcal{P} := \sum_{\ell \in \mathcal{L}} w(\ell, r) P_{\ell, p_\ell} \dots P_{\ell,2} P_{\ell,1},$$

then the mapping $\mathbf{z} \mapsto \mathbf{x}^S - \mathcal{P}(\mathbf{x}^S - \mathbf{z})$ is a contraction with unique fixed point $\mathbf{x}^S$. Moreover, the iteration of the algorithm can be expressed as:

$$\mathbf{x}^{(n+1)} = \mathbf{x}^S - \mathcal{P}(\mathbf{x}^S - \mathbf{x}^{(n)}). \tag{15}$$

2.2 Consistent Systems

We shall show in this section that the distributed Kaczmarz algorithm as defined in Equations (6) and (8) will converge to the solution with minimal norm in the case that there exists more than one solution. We first introduce the relaxed version of the algorithm (required to deal with inconsistent systems); we will show that for any appropriate relaxation parameter, the relaxed algorithm will converge to the solution of minimal norm.

The relaxed distributed Kaczmarz algorithm for tree based equations is as follows. Choose a relaxation parameter $\omega > 0$ (generally, we will require $\omega \in (0, 2)$, though see Section 3 for further discussion). At each node w during the dispersion stage of iteration n, the Kaczmarz update becomes:

$$\mathbf{x}_w^{(n)} = \mathbf{x}_v^{(n)} + \omega \frac{r_w(\mathbf{x}_v^{(n)})}{\|\mathbf{a}_w\|^2} \mathbf{a}_w. \tag{16}$$

We suppress the dependence of $\mathbf{x}_v^{(n)}$ on ω, but we will consider the limit

$$\mathbf{x}(\omega) := \lim_{n \to \infty} \mathbf{x}^{(n)} \tag{17}$$

which (in general) depends on ω. We will prove in Theorem 2 that when the system of equations is consistent, then this limit exists and is in fact independent of ω.

As in Equations (2) and (3), we use P_v and Q_v to denote the linear and affine projections, respectively. We will need the fact that Q_v is Lipschitz with constant 1:

$$\|Q_v \mathbf{z}_1 - Q_v \mathbf{z}_2\| \leq \|\mathbf{z}_1 - \mathbf{z}_2\|.$$

The relaxed Kaczmarz update in Equation (16) can be expressed as:

$$\mathbf{x}_u^{(n)} = [(1-\omega)I + \omega Q_u]\mathbf{x}_v^{(n)} =: Q_u^\omega \mathbf{x}_v^{(n)}.$$

Thus, the estimate $\mathbf{x}_\ell^{(n)}$ of the solution at leaf ℓ, given the solution estimate $\mathbf{x}^{(n)}$ as input at the root r, is:

$$\mathbf{x}_\ell^{(n)} = Q_{\ell,p_\ell}^\omega \cdots Q_{\ell,2}^\omega Q_{\ell,1}^\omega \mathbf{x}^{(n)} =: \mathcal{Q}_\ell^\omega \mathbf{x}^{(n)}. \tag{18}$$

We can now write the full update, with both dispersion and pooling stages, of the relaxed Kaczmarz algorithm as:

$$\mathbf{x}^{(n+1)} = \sum_{\ell \in \mathcal{L}} w(\ell, r) \mathcal{Q}_\ell^\omega \mathbf{x}^{(n)} =: \mathcal{Q}^\omega \mathbf{x}^{(n)}. \tag{19}$$

We note that, as above, each Q_v^ω is a Lipschitz map with constant 1 whenever $0 < \omega < 1$, but in fact, since $Q_v \mathbf{z}_1 - Q_v \mathbf{z}_2 = P_v \mathbf{z}_1 - P_v \mathbf{z}_2$, we have that Q_v^ω is Lipschitz with constant 1 whenever $0 < \omega < 2$. Moreover, as $\sum_{\ell \in \mathcal{L}} w(\ell, r) = 1$, we obtain:

Lemma 2 *For $0 < \omega < 2$, $\mathcal{Q}_\ell^\omega$ and $\mathcal{Q}^\omega$ are Lipschitz with constant* 1.

We note that the mappings $Q_{(\cdot)}^{(\cdot)}, \mathcal{Q}_{(\cdot)}^{(\cdot)}$ are affine transformations; we also have use for the analogous linear transformations. Similar to Equations (18) and (19), we write

$$\begin{aligned} P_v^\omega &:= (1-\omega)I + \omega P_v; \\ \mathcal{P}_\ell^\omega &:= P_{\ell,p_\ell}^\omega \cdots P_{\ell,2}^\omega P_{\ell,1}^\omega; \\ \mathcal{P}^\omega &:= \sum_{\ell \in \mathcal{L}} w(\ell, r) \mathcal{P}_\ell^\omega. \end{aligned}$$

Theorem 2 *If the system of equations given by $A\mathbf{x} = \mathbf{b}$ is consistent, then for any $0 < \omega < 2$, the sequence of estimates $\mathbf{x}^{(n)}$ as given in Equation (19) converges to the solution $\mathbf{x}^M$ of minimal norm as given by (5), provided the initial estimate $\mathbf{x}^{(0)} \in \mathcal{R}(A^*)$.*

We shall prove Theorem 2 using a sequence of lemmas. We follow the argument as presented in Natterer [21], adapting the lemmas as necessary. For completeness, we will state (without proof) the lemmas that we will use unaltered from [21]. (See also Yosida [37].)

Lemma 3 ([21], Lemma V.3.1)

Let T be a linear map on a Hilbert space H with $\|T\| \leq 1$. Then,

$$H = \mathcal{N}(I - T) \oplus \overline{\mathcal{R}(I - T)}.$$

Lemma 4 ([21], Lemma V.3.2)

Suppose $\{\mathbf{z}_k\}$ *is a sequence in* $\mathbb{C}^d$ *such that for any leaf* $\ell \in \mathcal{L}$,

$$\|\mathbf{z}_k\| \leq 1 \text{ and } \lim_{k\to\infty} \|\mathcal{P}_\ell^\omega \mathbf{z}_k\| = 1.$$

Then for $0 < \omega < 2$, *we have*

$$\lim_{k\to\infty} (I - \mathcal{P}_\ell^\omega)\mathbf{z}_k = 0.$$

Lemma 5 *Suppose* $\{\mathbf{z}_k\}$ *is a sequence in* $\mathbb{C}^d$ *such that*

$$\|\mathbf{z}_k\| \leq 1 \text{ and } \lim_{k\to\infty} \|\mathcal{P}^\omega \mathbf{z}_k\| = 1,$$

then for $0 < \omega < 2$, *we have*

$$\lim_{k\to\infty} (I - \mathcal{P}^\omega)\mathbf{z}_k = 0.$$

Proof Note that

$$(I - \mathcal{P}^\omega)\,\mathbf{z}_k = \sum_{\ell\in\mathcal{L}} w(\ell, r)\left(I - \mathcal{P}_\ell^\omega\right)\mathbf{z}_k,$$

so it is sufficient to show that the hypotheses of Lemma 4 are satisfied. Since $\|\mathcal{P}_\ell^\omega \mathbf{z}_k\| \leq 1$ and $\sum_\ell w(\ell, r) = 1$, we have

$$1 = \lim_{k\to\infty} \|\mathcal{P}^\omega \mathbf{z}_k\| \leq \lim_{k\to\infty} \sum_{\ell\in\mathcal{L}} w(\ell, r)\|\mathcal{P}_\ell^\omega \mathbf{z}_k\| \leq 1.$$

Thus, we must have $\lim \|\mathcal{P}_\ell^\omega \mathbf{z}_k\| = 1$ for every $\ell \in \mathcal{L}$. □

Lemma 6 *For* $0 < \omega < 2$, *we have*

$$\mathcal{N}(I - \mathcal{P}^\omega) = \bigcap_{v \text{ node}} \mathcal{N}(I - P_v). \tag{20}$$

Proof Suppose $P_v\mathbf{z} = \mathbf{z}$ for every node v. Then

$$\mathcal{P}^\omega\mathbf{z} = \sum_{\ell\in\mathcal{L}} w(\ell, r)P_{\ell,p_\ell}^\omega \ldots P_{\ell,1}^\omega\mathbf{z} = \sum_{\ell\in\mathcal{L}} w(\ell, r)\mathbf{z} = \mathbf{z}$$

thus the left containment follows.

Conversely, suppose that $\mathcal{P}^\omega\mathbf{z} = \mathbf{z}$. Again, we obtain

$$\|\mathbf{z}\| = \|\mathcal{P}^\omega\mathbf{z}\| \leq \sum_{\ell\in\mathcal{L}} w(\ell, r)\|P_{\ell,p_\ell}^\omega \cdots P_{\ell,1}^\omega\mathbf{z}\| \leq \|\mathbf{z}\|$$

which implies that

$$P_{\ell,p_\ell}^\omega \cdots P_{\ell,1}^\omega\mathbf{z} = \mathbf{z}$$

for every leaf ℓ. Hence, for every ℓ, and every $j = 1, \ldots, p_\ell$, $P^\omega_{\ell,j}\mathbf{z} = \mathbf{z}$. □

Lemma 7 ([21], Lemma V.3.5)

For $0 < \omega < 2$, $(\mathcal{P}^\omega)^k$ converges strongly, as $k \to \infty$, to the orthogonal projection onto

$$\bigcap_{v \text{ node}} \mathcal{N}(I - P_v) = \mathcal{N}(A).$$

The proof is identical to that in [21], using Lemmas 3, 5, and 6.

Proof (Proof of Theorem 2) Let $\mathbf{y}$ be any solution to the system of equations. We claim that for any $\mathbf{z}$,

$$\mathcal{Q}^\omega \mathbf{z} = \mathcal{P}^\omega(\mathbf{z} - \mathbf{y}) + \mathbf{y} \tag{21}$$

Indeed, for any nodes v and w, and consequently for any leaf ℓ, we have

$$\begin{aligned}
& Q^\omega_v \mathbf{z} = \mathbf{y} + P^\omega_v(\mathbf{z} - \mathbf{y}) \\
\Rightarrow \quad & Q^\omega_w Q^\omega_v \mathbf{z} = \mathbf{y} + P^\omega_w P^\omega_v(\mathbf{z} - \mathbf{y}) \\
\Rightarrow \quad & Q^\omega_\ell \mathbf{z} = \mathbf{y} + \mathcal{P}^\omega_\ell(\mathbf{z} - \mathbf{y}) \\
\Rightarrow \quad & \sum_{\ell \in \mathcal{L}} w(\ell, r) Q^\omega_\ell \mathbf{z} = \sum_{\ell \in \mathcal{L}} w(\ell, r) \left(\mathbf{y} + \mathcal{P}^\omega_\ell(\mathbf{z} - \mathbf{y})\right),
\end{aligned}$$

which demonstrates Equation (21).

Therefore, by Lemma 7, we have that for any $\mathbf{z}$,

$$(\mathcal{Q}^\omega)^k \mathbf{z} \to \mathbf{y} + Pr(\mathbf{z} - \mathbf{y}),$$

as $k \to \infty$, where Pr is the projection onto $\mathcal{N}(A)$. If $\mathbf{z} \in \mathcal{R}(A^*)$, we have that $\mathbf{y} + Pr(\mathbf{z} - \mathbf{y})$ is the unique solution to the system of equations that is in $\mathcal{R}(A^*)$, and hence is the solution of minimal norm. □

We can see that for $\mathbf{z} \in \mathcal{R}(A^*)$, the convergence rate of $(\mathcal{Q}^\omega)^k \mathbf{z} \to \mathbf{x}^M$ is linear, but we will formalize this in the next subsection (Corollary 1).

2.3 Inconsistent Equations

We now consider the case of inconsistent systems of equations. For this purpose, we must consider the relaxed version of the algorithm, as in the previous subsection. Again, we assume $0 < \omega < 2$ and consider the limit

$$\lim_{n \to \infty} \mathbf{x}^{(n)} = \mathbf{x}(\omega).$$

We will prove in Theorem 3 and Corollary 1 that the limit exists, but unlike in the case of consistent systems, the limit will depend on ω. Moreover, we will prove in Theorem 4 that the limit

$$\lim_{\omega \to 0} \mathbf{x}(\omega) = \mathbf{x}^{LS}$$

exists, and $\mathbf{x}^{LS}$ is a generalized solution which minimizes a weighted least-squares norm. We follow the presentation of the analogous results for the classical Kaczmarz algorithm as presented in [21]. Indeed, we will proceed by analyzing the distributed Kaczmarz algorithm using the ideas from Successive Over-Relaxation (SOR). We need to follow the updates as they disperse through the tree, and also how the updates are pooled back at the root, and so we define the following quantities.

We begin with reindexing the equations, which are currently indexed by the nodes as $a_v^* \mathbf{x} = b_v$. As before, for each leaf ℓ, we consider the path from the root r to the leaf ℓ, and index the corresponding equations as:

$$\mathbf{a}_{\ell,1}^* \mathbf{x} = b_{\ell,1}, \ \ldots, \mathbf{a}_{\ell,p_\ell}^* \mathbf{x} = b_{\ell,p_\ell}.$$

For each leaf ℓ, we can define:

$$\mathcal{A}_\ell = \begin{pmatrix} \mathbf{a}_{\ell,1}^* \\ \mathbf{a}_{\ell,2}^* \\ \vdots \\ \mathbf{a}_{\ell,p_\ell}^* \end{pmatrix}, \quad \mathbf{b}_\ell = \begin{pmatrix} b_{\ell,1} \\ b_{\ell,2} \\ \vdots \\ b_{\ell,p_\ell} \end{pmatrix}, \quad \mathcal{D}_\ell = \begin{pmatrix} \mathbf{a}_{\ell,1}^* \mathbf{a}_{\ell,1} & 0 & \ldots & 0 \\ 0 & \mathbf{a}_{\ell,2}^* \mathbf{a}_{\ell,2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \mathbf{a}_{\ell,p_\ell}^* \mathbf{a}_{\ell,p_\ell} \end{pmatrix}$$

and

$$\mathcal{L}_\ell = \begin{pmatrix} 0 & 0 & \ldots & 0 & 0 \\ \mathbf{a}_{\ell,2}^* \mathbf{a}_{\ell,1} & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{a}_{\ell,p_\ell}^* \mathbf{a}_{\ell,1} & \mathbf{a}_{\ell,p_\ell}^* \mathbf{a}_{\ell,2} & \ldots & \mathbf{a}_{\ell,p_\ell}^* \mathbf{a}_{\ell,p_\ell - 1} & 0 \end{pmatrix}$$

Then from input $\mathbf{x}^{(n)}$ at the root of the tree, the approximation at leaf ℓ after the dispersion stage in iteration n is given by:

$$\mathbf{x}_\ell^{(n)} = Q_\ell^\omega \mathbf{x}^{(n)} = \mathbf{x}^{(n)} + \sum_{j=1}^{p_\ell} u_j \mathbf{a}_{\ell,j} = \mathbf{x}^{(n)} + \mathcal{A}_\ell^* \mathbf{u},$$

where

$$\mathbf{u} := \left(u_1 \ldots u_{p_\ell}\right)^T = \omega \left(\mathcal{D}_\ell + \omega \mathcal{L}_\ell\right)^{-1} \left(\mathbf{b}_\ell - \mathcal{A}_\ell \mathbf{x}^{(n)}\right).$$

Therefore, we can write

$$\begin{aligned} \mathbf{x}_\ell^{(n)} &= \mathbf{x}^{(n)} + \omega \mathcal{A}_\ell^* \left(\mathcal{D}_\ell + \omega \mathcal{L}_\ell\right)^{-1} \left(\mathbf{b}_\ell - \mathcal{A}_\ell \mathbf{x}^{(n)}\right) \\ &= \left(I - \omega \mathcal{A}_\ell^* \left(\mathcal{D}_\ell + \omega \mathcal{L}_\ell\right)^{-1} \mathcal{A}_\ell\right) \mathbf{x}^{(n)} + \omega \mathcal{A}_\ell^* \left(\mathcal{D}_\ell + \omega \mathcal{L}_\ell\right)^{-1} \mathbf{b}_\ell. \end{aligned}$$

Combining these approximations back at the root yields:

$$\begin{aligned}\mathbf{x}^{(n+1)} &= \sum_{\ell\in\mathcal{L}} w(\ell,r)\mathbf{x}_\ell^{(n)} \\ &= \sum_{\ell\in\mathcal{L}} w(\ell,r)\left(I-\omega\mathcal{A}_\ell^*\left(\mathcal{D}_\ell+\omega\mathcal{L}_\ell\right)^{-1}\mathcal{A}_\ell\right)\mathbf{x}^{(n)} + \omega\sum_{\ell\in\mathcal{L}} w(\ell,r)\mathcal{A}_\ell^*\left(\mathcal{D}_\ell+\omega\mathcal{L}_\ell\right)^{-1}\mathbf{b}_\ell \\ &= \left(I-\omega\sum_{\ell\in\mathcal{L}} w(\ell,r)\mathcal{A}_\ell^*\left(\mathcal{D}_\ell+\omega\mathcal{L}_\ell\right)^{-1}\mathcal{A}_\ell\right)\mathbf{x}^{(n)} + \omega\sum_{\ell\in\mathcal{L}} w(\ell,r)\mathcal{A}_\ell^*\left(\mathcal{D}_\ell+\omega\mathcal{L}_\ell\right)^{-1}\mathbf{b}_\ell.\end{aligned} \tag{22}$$

We write

$$\mathbf{x}^{(n+1)} = \sum_{\ell\in\mathcal{L}} w(\ell,r)\mathcal{B}_\ell^\omega\mathbf{x}^{(n)} + \sum_{\ell\in\mathcal{L}} w(\ell,r)\mathbf{b}_\ell^\omega$$

where

$$\mathcal{B}_\ell^\omega := I-\omega\mathcal{A}_\ell^*\left(\mathcal{D}_\ell+\omega\mathcal{L}_\ell\right)^{-1}\mathcal{A}_\ell; \qquad \mathbf{b}_\ell^\omega := \omega\mathcal{A}_\ell^*\left(\mathcal{D}_\ell+\omega\mathcal{L}_\ell\right)^{-1}\mathbf{b}_\ell. \tag{23}$$

Written in this form, for each leaf ℓ, the input at the root undergoes the linearly ordered Kaczmarz algorithm. So, if the input at the root is $\mathbf{x}^{(n)}$, then the estimate at leaf ℓ is:

$$\mathbf{x}_\ell^{(n)} = Q_\ell^\omega\mathbf{x}^{(n)} = \mathcal{B}_\ell^\omega\mathbf{x}^{(n)} + \mathbf{b}_\ell^\omega.$$

As we shall see, for each leaf ℓ and $\omega\in(0,2)$, $\mathcal{B}_\ell^\omega$ has operator norm bounded by 1, and the eigenvalues are either 1 or strictly less than 1 in magnitude. We state these formally in Lemma 8.

We enumerate the leaves of the tree as $\ell_1,\ldots,\ell_t$, and write:

$$\mathcal{A} = \begin{pmatrix}\mathcal{A}_{\ell_1}\\ \vdots \\ \mathcal{A}_{\ell_t}\end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix}\mathbf{b}_{\ell_1}\\ \vdots \\ \mathbf{b}_{\ell_t}\end{pmatrix}$$

The system of equations $A\mathbf{x}=\mathbf{b}$ becomes:

$$\mathcal{A}\mathbf{x} = \mathbf{b} \tag{24}$$

where many of the equations are now repeated in Equation (24). However, we have $\mathcal{N}(\mathcal{A}) = \mathcal{N}(A)$ and $\mathcal{R}(\mathcal{A}^*) = \mathcal{R}(A^*)$.

We also write

$$\mathcal{D} := \begin{pmatrix}\mathcal{D}_{\ell_1} & 0 & \ldots & 0\\ 0 & \mathcal{D}_{\ell_2} & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & \mathcal{D}_{\ell_t}\end{pmatrix} \qquad \mathcal{L} = \begin{pmatrix}\mathcal{L}_{\ell_1} & 0 & \ldots & 0\\ 0 & \mathcal{L}_{\ell_2} & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & \mathcal{L}_{\ell_t}\end{pmatrix} \tag{25}$$

so

$$(\mathcal{D}+\omega\mathcal{L})^{-1} = \begin{pmatrix} (\mathcal{D}_{\ell_1}+\omega\mathcal{L}_{\ell_1})^{-1} & 0 & \dots & 0 \\ 0 & (\mathcal{D}_{\ell_2}+\omega\mathcal{L}_{\ell_2})^{-1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & (\mathcal{D}_{\ell_t}+\omega\mathcal{L}_{\ell_t})^{-1} \end{pmatrix} \tag{26}$$

We also define

$$\mathcal{W} = \begin{pmatrix} w(\ell_1,r)I_{p_{\ell_1}} & 0 & \dots & 0 \\ 0 & w(\ell_2,r)I_{p_{\ell_2}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w(\ell_t,r)I_{p_{\ell_t}} \end{pmatrix} \tag{27}$$

Note that since $\mathcal{D}+\omega\mathcal{L}$ and $\mathcal{W}$ are block matrices with blocks of the same size, and in $\mathcal{W}$ the blocks are scalar multiples of the identity, we have that the two matrices commute:

$$(\mathcal{D}+\omega\mathcal{L})^{-1}\mathcal{W} = \mathcal{W}(\mathcal{D}+\omega\mathcal{L})^{-1} = \mathcal{W}^{1/2}(\mathcal{D}+\omega\mathcal{L})^{-1}\mathcal{W}^{1/2}. \tag{28}$$

We can therefore write Equation (22) as

$$\begin{aligned} \mathbf{x}^{(n+1)} &= \left(I-\omega\mathcal{A}^*(\mathcal{D}+\omega\mathcal{L})^{-1}\mathcal{W}\mathcal{A}\right)\mathbf{x}^{(n)} + \omega\mathcal{A}^*(\mathcal{D}+\omega\mathcal{L})^{-1}\mathcal{W}\mathbf{b}. \\ &:= \mathcal{B}^\omega\mathbf{x}^{(n)} + \mathbf{b}^\omega. \end{aligned}$$

Note that $\mathcal{R}(\mathcal{A}^*)$ is an invariant subspace for $\mathcal{B}^\omega$, and that $\mathbf{b}^\omega \in \mathcal{R}(\mathcal{A}^*)$. We let $\widehat{\mathcal{B}}^\omega$ denote the restriction of $\mathcal{B}^\omega$ to the subspace $\mathcal{R}(\mathcal{A}^*)$. As we shall see, provided the input $\mathbf{x}^0 \in \mathcal{R}(\mathcal{A}^*)$, the sequence $\mathbf{x}^k$ converges. In fact, we will show that the transformation $\widehat{\mathcal{B}}^\omega$ is a contraction, and since $\mathbf{b}^\omega \in \mathcal{R}(\mathcal{A}^*)$, then the mapping

$$\mathbf{z} \mapsto \widehat{\mathcal{B}}^\omega\mathbf{z} + \mathbf{b}^\omega$$

has a unique fixed point within $\mathcal{R}(\mathcal{A}^*)$. We shall do so via a series of lemmas.

Lemma 8 *For each leaf ℓ and for $\omega \in (0,2)$, $\mathcal{B}_\ell^\omega$ is Lipschitz continuous with constant at most* 1 *(i.e. it has operator norm at most* 1*). Consequently, $\widehat{\mathcal{B}}^\omega$ is also Lipschitz continuous with constant at most* 1.

Moreover, for each leaf ℓ and $\omega \in (0,2)$, if λ is an eigenvalue of $\mathcal{B}_\ell^\omega$ with $|\lambda| = 1$, then $\lambda = 1$. Consequently, any eigenvalue $\lambda \neq 1$ has the property $|\lambda| < 1$.

Proof For input $\mathbf{z}_i$, we have that

$$Q_\ell^\omega\mathbf{z}_i = \mathcal{B}_\ell^\omega\mathbf{z}_i + \mathbf{b}_\ell^\omega,$$

hence

$$\|\mathcal{B}_\ell^\omega\mathbf{z}_1 - \mathcal{B}_\ell^\omega\mathbf{z}_2\| = \|Q_\ell^\omega\mathbf{z}_1 - Q_\ell^\omega\mathbf{z}_2\| \leq \|\mathbf{z}_1 - \mathbf{z}_2\|$$

by Lemma 2. Since $\mathcal{B}^\omega$ is a convex combination of the $\mathcal{B}_\ell^\omega$, it also has Lipschitz constant at most 1. The last conclusion follows from [21, Lemma V.3.9]. □

Theorem 3 *The spectral radius of $\widehat{\mathcal{B}}^\omega$ is strictly less than* 1.

Proof For each leaf ℓ, Lemma 8 implies that

$$\|\mathcal{B}_\ell^\omega\| \leq 1, \qquad |\langle \mathcal{B}_\ell^\omega \mathbf{v}, \mathbf{v}\rangle| \leq \|\mathbf{v}\|^2. \tag{29}$$

Let λ be an eigenvalue for $\widehat{\mathcal{B}}^\omega$. We must have $\lambda \neq 1$; if it were not so, then there exists a nonzero $\mathbf{z} \in \mathcal{R}(\mathcal{A}^*)$ with $\widehat{\mathcal{B}}^\omega \mathbf{z} = \mathbf{z}$. However, by Lemma 6 we must have $\mathbf{z} \in \mathcal{N}(A) = \mathcal{N}(S)$ which is a contradiction. Let $\mathbf{v}$ be a unit norm eigenvector for λ. We have

$$|\lambda| = |\langle \widehat{\mathcal{B}}^\omega \mathbf{v}, \mathbf{v}\rangle| \leq \sum_{\ell \in \mathcal{L}} w(\ell, r) |\langle \mathcal{B}_\ell^\omega \mathbf{v}, \mathbf{v}\rangle| \leq 1.$$

Now suppose that $|\lambda| = 1$, then we similarly obtain

$$\lambda = \sum_{\ell \in \mathcal{L}} \langle \mathcal{B}_\ell^\omega \mathbf{v}, \mathbf{v}\rangle \tag{30}$$

from which we deduce that the argument of the complex number $\langle \mathcal{B}_\ell^\omega \mathbf{v}, \mathbf{v}\rangle$ is independent of the leaf ℓ. Therefore, we must have for every leaf ℓ

$$\langle \mathcal{B}_\ell^\omega \mathbf{v}, \mathbf{v}\rangle = \lambda. \tag{31}$$

However, we know by the Cauchy-Schwarz inequality that equality in Equation (31) can only occur when $(\mathbf{v}, \lambda)$ is an eigenvector/eigenvalue pair for $\mathcal{B}_\ell^\omega$. However, Lemma 8 implies that none of the leaves ℓ have the property that λ is an eigenvalue, so we have arrived at a contradiction. □

Corollary 1 *For $\omega \in (0, 2)$ and for any initial input $\mathbf{x}^{(0)} \in \mathcal{R}(\mathcal{A}^*)$, we have that the sequence given by*

$$\mathbf{x}^{(n+1)} = \widehat{\mathcal{B}}^\omega \mathbf{x}^{(n)} + \mathbf{b}^\omega \tag{32}$$

converges to a unique point in $\mathcal{R}(\mathcal{A}^*)$, independent of $\mathbf{x}^{(0)}$, and the convergence rate is linear.

The following can be found in [21, Theorem IV.1.1]:

Lemma 9 *For each $\omega \in (0, 2)$, let*

$$\mathbf{x}(\omega) = \lim_{n \to \infty} \mathbf{x}^{(n)}$$

where $\mathbf{x}^{(n)}$ are as in Equation (32). Then, $\mathbf{x}(\omega)$ is the unique vector that satisfies the conditions

$$\mathcal{A}^* \left(\mathcal{D} + \omega \mathcal{L}\right)^{-1} \mathcal{W} \left(\mathbf{b} - \mathcal{A}\mathbf{z}\right) = \mathbf{0}; \qquad \mathbf{z} \in \mathcal{R}(\mathcal{A}^*). \tag{33}$$

Theorem 4 *For each $\omega \in (0, 2)$, let*

$$\mathbf{x}(\omega) = \lim_{n\to\infty} \mathbf{x}^n$$

as in Equation (32). Then,

$$\lim_{\omega\to 0} \mathbf{x}(\omega) = \mathbf{x}^{LS}$$

where $\mathbf{x}^{LS}$ *minimizes the functional*

$$\mathbf{z} \mapsto \langle \mathcal{D}^{-1}\mathcal{W}(\mathbf{b} - \mathcal{A}\mathbf{z}), (\mathbf{b} - \mathcal{A}\mathbf{z})\rangle. \tag{34}$$

Proof Let $\mathbf{x}^{LS}$ be the unique vector that satisfies the conditions

$$\mathcal{A}^*\mathcal{D}^{-1}\mathcal{W}\left(\mathbf{b} - \mathcal{A}\mathbf{x}^{LS}\right) = \mathbf{0}; \quad \mathbf{x}^{LS} \in \mathcal{R}(\mathcal{A}^*). \tag{35}$$

We have that $\mathbf{x}(\omega)$, as the unique solution of Equation (33) and $\mathbf{x}^{LS}$, as the unique solution of Equation (35), satisfy

$$\mathbf{x}(\omega) = \mathbf{x}^{LS} + O(\omega).$$

Indeed, this follows from the fact that $(\mathcal{D} + \omega\mathcal{L})^{-1} \to \mathcal{D}^{-1}$ as $\omega \to 0$, together with the fact that $\mathbf{x}(\omega), \mathbf{x}^{LS} \in \mathcal{R}(\mathcal{A}^*)$. □

We can re-write Equation (34) in the following way:

$$\mathbf{z} \mapsto \langle D^{-1}V(\mathbf{b} - A\mathbf{z}), (\mathbf{b} - A\mathbf{z})\rangle \tag{36}$$

where D is the diagonal matrix with entries given by $\|\mathbf{a}_v\|^2$, and V is the diagonal matrix whose entry for node v is given by:

$$V_{vv} = \sum_{\substack{\ell\in\mathcal{L}\\ \ell<v}} w(\ell, r).$$

2.4 Distributed Solutions

For each node v in the tree, the sequence of approximations $\mathbf{x}_v^{(n)}$ and $\mathbf{y}_v^{(n)}$ will have a limit, i.e. the following limits exist:

$$\lim_{n\to\infty} \mathbf{x}_v^{(n)} = \mathbf{x}_v; \qquad \lim_{n\to\infty} \mathbf{y}_v^{(n)} = \mathbf{y}_v. \tag{37}$$

In the relaxed case, these limits may depend on the relaxation parameter ω; if so we will denote this dependence by $\mathbf{x}_v(\omega)$ and $\mathbf{y}_v(\omega)$.

Corollary 2 *If the system of equations* $A\mathbf{x} = \mathbf{b}$ *is consistent, then for every node* v *and every* $\omega \in (0, 2)$*, the limits* $\mathbf{x}_v$ *and* $\mathbf{y}_v$ *as in Equation (37) equal* $\mathbf{x}^M$*, the solution of minimal norm.*

Proof We have by Theorem 2 that $\mathbf{x}(\omega) = \mathbf{x}^M$ for every $\omega \in (0, 2)$. For a node v, let the path from the root r to v be denoted by $r = (v, 1), \ldots, (v, p_v) = v$, where $p_v - 1$ is the length of the path. Then, we have that

$$\lim_{n\to\infty} \mathbf{x}_v^{(n)} = \lim_{n\to\infty} Q^{\omega}_{v,p_v} \cdots Q^{\omega}_{v,1} \mathbf{x}^{(n)} = Q^{\omega}_{v,p_v} \cdots Q^{\omega}_{v,1} \mathbf{x}(\omega) = \mathbf{x}^M.$$

This holds as a consequence of the fact that any solution to the system of equations is invariant under $Q^{\omega}_{(\cdot)}$.

Since we have that $\mathbf{y}_v^{(n)}$ is a convex combination of the vectors $\mathbf{x}_\ell^{(n)}$, which all converge to $\mathbf{x}(\omega)$, we have that $\mathbf{y}_v = \mathbf{x}^M$ also. □

Corollary 3 *If the system of equations $A\mathbf{x} = \mathbf{b}$ is inconsistent, then for every node v and every $\omega \in (0, 2)$, the limits $\mathbf{x}_v$ and $\mathbf{y}_v$ as in Equation (37) exist and depend on ω. Moreover, we have*

$$\lim_{\omega\to 0} \mathbf{x}_v(\omega) = \mathbf{x}^{LS} \qquad \lim_{\omega\to 0} \mathbf{y}_v(\omega) = \mathbf{x}^{LS},$$

where $\mathbf{x}^{LS}$ is the vector as in Theorem 4.

Proof We apply the SOR analysis of $\mathbf{x}_v^{(n)} = Q^{\omega}_{(v,p_v)} \cdots Q^{\omega}_{(v,1)} \mathbf{x}^{(n)}$ with input $\mathbf{x}^{(n)}$ to obtain

$$\mathbf{x}_v^{(n)} = \mathcal{B}_v^{\omega} \mathbf{x}^{(n)} + \mathbf{b}_v^{\omega}$$

where $\mathcal{B}_v^{\omega}$ and $\mathbf{b}_v^{\omega}$ are analogous to those in Equation (23). Taking limits on n, we obtain

$$\mathbf{x}_v(\omega) = \mathcal{B}_v^{\omega} \mathbf{x}(\omega) + \mathbf{b}_v^{\omega}.$$

Since, as $\omega \to 0$, we have that $\mathcal{B}_v^{\omega} \to I$, $\mathbf{b}_v^{\omega} \to 0$, and $\mathbf{x}(\omega) \to \mathbf{x}^{LS}$, we obtain

$$\lim_{\omega\to 0} \mathbf{x}_v(\omega) = \mathbf{x}^{LS}.$$

As previously, $\mathbf{y}_v(\omega)$ is a convex combination of $\mathbf{x}_\ell(\omega)$, so $\mathbf{y}_v(\omega) \to \mathbf{x}^{LS}$ as $\omega \to 0$ also. □

2.5 Error Analysis

We consider the question of how errors propagate through the iterations of the dispersion and pooling stages. We model errors as additive; the sources of errors could be machine errors, transmission errors, errors from compression to reduce communication complexity, etc. Additive errors then take on the form

$$\mathbf{x}_{v,e}^{(n)} = \mathbf{x}_v^{(n)} + \epsilon_v^{(n)}; \qquad \mathbf{y}_{v,e}^{(n)} = \mathbf{y}_v^{(n)} + \delta_v^{(n)} \tag{38}$$

Here, $\mathbf{x}_{v,e}^{(n)}$ and $\mathbf{y}_{v,e}^{(n)}$ are the error-riddled estimates which are passed to the successor (or predecessor) nodes in the dispersion (or pooling) stage, respectively, with additive

errors $\epsilon_v^{(n)}$ and $\delta_v^{(n)}$. Measurement errors, meaning errors in $\mathbf{b}$, are considered in [4, 10].

We trace the errors during the dispersion stage as follows: for node v on a path between the root r and leaf ℓ, and the path parameterized by $r = (\ell, 1), \ldots, (\ell, p_\ell) = \ell$, suppose that $v = (\ell, k)$. Then, the error introduced at node v (with errors introduced at no other node) results in the estimate

$$\begin{aligned} \mathbf{x}_{\ell,e}^{(n)} &= Q_{\ell,p_\ell}^{\omega} \cdots Q_{\ell,k+1}^{\omega}(\mathbf{x}_v^{(n)} + \epsilon_v^{(n)}) \\ &= Q_{\ell,p_\ell}^{\omega} \cdots Q_{\ell,k+1}^{\omega}(\mathbf{x}_v^{(n)}) + \widetilde{\epsilon}_{v,\ell}^{(n)} \\ &= Q_\ell^{\omega}(\mathbf{x}^{(n)}) + \widetilde{\epsilon}_{v,\ell}^{(n)}. \end{aligned} \tag{39}$$

Equation (39) follows for some $\widetilde{e}_v^{(n)}$ since the $Q_{(\cdot)}^{\omega}$ are affine transformations. We have that

$$\|\widetilde{\epsilon}_v^{(n)}\| = \|Q_{\ell,p_\ell}^{\omega} \cdots Q_{\ell,k+1}^{\omega}(\mathbf{x}_v^{(n)} + \epsilon_v^{(n)}) - Q_{\ell,p_\ell}^{\omega} \cdots Q_{\ell,k+1}^{\omega}(\mathbf{x}_v^{(n)})\| \leq \|\epsilon_v^{(n)}\|$$

since the $Q_{(\cdot)}^{\omega}$ have Lipschitz constant 1. The additive errors $\delta_v^{(n)}$ simply sum in the pooling stage, and thus we calculate the total errors from iteration n to iteration $n+1$.

Lemma 10 *Suppose we have additive errors as in Equation (38) introduced in iteration n. Suppose no errors were introduced in previous iterations. Then the estimate after iteration n is:*

$$\mathbf{x}_e^{(n+1)} = \mathbf{x}^{(n+1)} + \sum_v \sum_{\substack{\ell \in \mathcal{L} \\ \ell \prec v}} w(\ell, r)\tilde{e}_v^{(n)} + \sum_v w(v, r)\delta_{v,e}^{(n)}. \tag{40}$$

The magnitude of the error is bounded by:

$$\|\mathbf{x}_e^{(n+1)} - \mathbf{x}^{(n+1)}\| \leq K \; max \; \{\|\epsilon_v^{(n)}\|, \|\delta_v^{(n)}\|\} \tag{41}$$

where K is 2 *times the depth of the tree.*

We write

$$E^{(n)} = \sum_v \sum_{\substack{\ell \in \mathcal{L} \\ \ell \prec v}} w(\ell, r)\tilde{e}_v^{(n)} + \sum_v w(v, r)\delta_{v,e}^{(n)}. \tag{42}$$

Theorem 5 *If the additive errors in Equation (38) are uniformly bounded by M, and the system of equations $A\mathbf{x} = \mathbf{b}$ has a unique solution, then the sequence of approximations $\{\mathbf{x}_e^{(n)}\}$ has the property that*

$$\limsup_{n \to \infty} \|\mathbf{x}(\omega) - \mathbf{x}_e^{(n)}\| \leq \frac{2KM}{1 - \rho(\mathcal{B}^\omega)} \tag{43}$$

where K is the depth of the tree.

Proof We have

$$\mathbf{x}_e^{(n)} = \mathbf{x}^{(n)} + \sum_{k=1}^{n} (\mathcal{B}^{\omega})^{n-k} E^{(k)}.$$

As noted previously, $\|E^{(k)}\| \leq 2KM$, and if $A\mathbf{x} = \mathbf{b}$ has a unique solution, then $\rho(\mathcal{B}^{\omega}) < 1$ (see proof of Theorem 1).

Thus, for any matrix norm $\|\cdot\|$ with $\rho(\mathcal{B}^{\omega}) < \|\mathcal{B}^{\omega}\|$

$$\|\mathbf{x}^{(n)} - \mathbf{x}_e^{(n)}\| \leq \sum_{k=0}^{n-1} 2KM\|\mathcal{B}^{\omega}\|^k$$

from which Equation (43) follows. □

If the system of equations does not have a unique solution, then the mapping $\mathcal{B}^{\omega}$ has 1 as an eigenvalue, and so the parts of the errors that lie in that eigenspace accumulate. Hence, no stability result is possible in this case.

3 Implementation and Examples

For the standard Kaczmarz algorithm, it is well known that the method converges if and only if the relaxation parameter ω is in the interval $(0, 2)$. For our distributed Kaczmarz, the situation is not nearly as clear. The proofs of Theorems 2 and 4 require that $\omega \in (0, 2)$, but in numerical experiments, convergence occurred for $\omega \in (0, \Omega)$ for some $\Omega \geq 2$. The largest Ω observed was around 3.8. The precise upper limit depends on the equations themselves. In this section, we perform a preliminary analysis of the computation of Ω and the optimal ω_{opt} for a very simple setup, and give numerical results for several examples.

3.1 Examples

Example 1

We consider the matrix

$$A = \begin{pmatrix} -\sin\alpha & \cos\alpha \\ 0 & 1 \end{pmatrix}.$$

In geometric terms, the Kaczmarz method for this example corresponds to projection onto the x-axis and onto a line forming an angle α with the x-axis.

For standard Kaczmarz, the iteration matrix is

$$B_{\omega} = I - \omega A^*(D + \omega L)^{-1}A = \begin{pmatrix} 1 - \omega\sin^2\alpha & \omega\sin\alpha\cos\alpha \\ \omega(1-\omega)\sin\alpha\cos\alpha & (1-\omega)(1-\omega\cos^2\alpha) \end{pmatrix}.$$

The eigenvalues are

$$\lambda = \left[\frac{\omega^2 \cos^2 \alpha}{2} + (1-\omega)\right] \pm \omega \cos\alpha \sqrt{(\omega-2)^2 - \omega^2 \sin^2 \alpha}.$$

For small ω, the eigenvalues are real and decreasing as a function of ω. They become complex at

$$\omega_{opt} = \frac{2}{1+\sin\alpha},$$

which is between 1 and 2. After that point, both eigenvalues have magnitude $\omega - 1$, and the spectral radius increases in a straight line. The dependence of ρ on ω is illustrated below in the left half of Figure 3. Here $\alpha = \pi/3$, $\omega_{opt} \approx 1.0718$, $\rho_{opt} \approx 0.0718$.

As pointed out in [21], there is a strong connection between the classical Kaczmarz method and Successive Over-Relaxation (SOR). In SOR the relationship between ω and ρ shows the same type of behavior.

The example with two equations is too small to implement as distributed Kaczmarz, but we consider something similar. We project the same $\mathbf{x}^{(n)}$ onto each line, and average the result to get $\mathbf{x}^{(n+1)}$. We will refer to this as the *averaged Kaczmarz method*.

The iteration matrix is

$$B_\omega = \begin{pmatrix} 1 - \frac{\omega}{2}\sin^2\alpha & \frac{\omega}{2}\sin\alpha\cos\alpha \\ \frac{\omega}{2}\sin\alpha\cos\alpha & \frac{\omega}{2}\sin^2\alpha - \omega + 1. \end{pmatrix}$$

The eigenvalues here are always real and vary linearly with ω, namely

$$\lambda_{1,2} = 1 + \frac{\omega}{2}\left(\pm\cos\alpha - 1\right).$$

They both have the value 1 at $\omega = 0$, and are both decreasing with increasing ω. The first one reaches (-1) at

$$\Omega = \frac{4}{1+\cos\alpha}.$$

Thus, the upper limit Ω is somewhere between 2 and 4, depending on α. In numerical experiments with the distributed Kaczmarz method for larger matrices, we have observed Ω near 4, but never above 4. We conjecture that Ω can never be larger than 4.

The minimum spectral radius occurs at $\omega_{opt} = 2$, independent of α, with $\rho_{opt} = \cos\alpha$. The dependence of ρ on ω is illustrated below in the left half of Figure 3. In this example, the graph for the averaged Kaczmarz method consists of two line segments, with $\omega_{opt} = 2$, $\rho_{opt} = 0.5$.

Figure 2 illustrates the optimal ω for $\alpha = \pi/2$. The optimal ω for standard Kaczmarz is $\omega = 1$, with $\rho = 0$. Convergence occurs in a single step. For the averaged method, the optimal ω is 2, where again convergence occurs in a single step. The averaged method would still converge for a range of $\omega > 2$.

Numerical experiments with larger sets of equations indicate that the optimal ω for classical Kaczmarz is usually larger than 1, but of course cannot exceed 2.

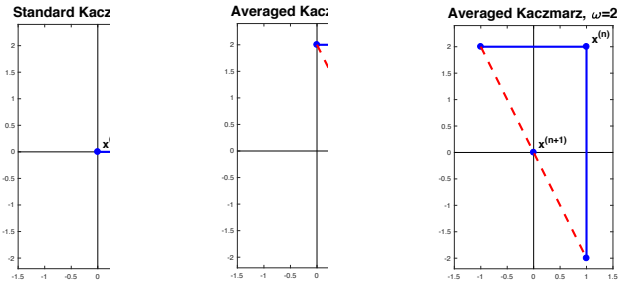


Fig. 2 Example 1 with $\alpha = \pi/2$. The pictures show one step of standard Kaczmarz with $\omega = 1$, and one step of averaged Kaczmarz for $\omega = 1$ and $\omega = 2$. This illustrates the need for a larger ω in the averaged Kaczmarz method.

The optimal ω for distributed Kaczmarz is usually larger than 2, sometimes even approaching 4.

Example 2

We used a random matrix of size 8×8, with entries generated using a standard normal distribution. For the distributed Kaczmarz method, we used the 8-node graph as shown on the right in Figure 4.

For the standard Kaczmarz method, the optimal relaxation parameter was $\omega_{opt} \approx 1.7354$, with spectral radius $\rho_{opt} \approx 0.93147$. For the distributed Kaczmarz method, the results were $\omega_{opt} \approx 3.7888$, with spectral radius $\rho_{opt} \approx 0.99087$. This is illustrated in on the right in Figure 3.

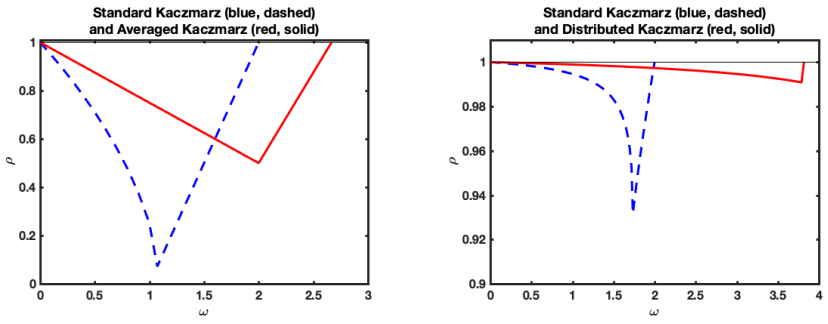


Fig. 3 Dependence of the spectral radius ρ of the iteration matrix on the relaxation parameter ω. The left graph shows Example 1 with $\alpha = \pi/3$. The right graph shows Example 2.

3.2 Implementation

The implementation of the distributed Kaczmarz algorithm is based on the Matlab Graph Theory toolbox. This toolbox provides support for standard graphs and directed graphs (digraphs), weighted or unweighted. We are using a weighted digraph. The graph is defined by specifying the edges, which automatically also defines the nodes. Specifying nodes is only necessary if there are additional isolated nodes. Both nodes and edges can have additional properties attached to them. We take advantage of that by storing the equations and right-hand sides, as well as the current approximate solution, in the nodes. The weights are stored in the edges. We are currently only considering tree-structured graphs. One node is the root. Each node other than the root has one incoming edge, coming from the predecessor, and zero or more outgoing edges leading to the successors. A node without a successor is called a *leaf*.

The basic Kaczmarz step has the form `x_new = update_node(node,omega,x)`. The graph itself is a global data structure, accessible to all subroutines; it would be very inefficient to pass it as an argument every time.

The `update_node` routine does the following:

- Use the equation(s) in the node to update `x`
- Execute the `update_node` routine for each successor node
- Combine the results into a new `x`, using the weights stored in the outgoing edges
- Return `x_new`

This routine needs to be called only once per iteration, for the root. It will traverse the entire tree recursively.

3.3 Numerical Experiments

We illustrate the methods with some simple numerical experiments. All experiments were run with three different nonsingular matrices each, of sizes 3×3 and 8×8. All matrices were randomly generated once, and then stored. The right-hand size vectors are also random, and scaled so that the true solution has L^2-norm 1. The test matrices are

- An almost orthogonal matrix, generated from a random orthogonal matrix by truncating to one decimal of accuracy
- A random matrix, based on a standard normal distribution
- A random matrix, based on a uniform distribution in $[-1, 1]$

In each case, we used the optimal ω, based on minimizing the spectral radius of the iteration matrix numerically. The distributed Kaczmarz method used the graphs shown in Figure 4. Results are shown in Tables 1 and 2. In all cases, we start with $\mathbf{x}_0 = \mathbf{0}$, so the initial L^2-error is $e_0 = 1$. e_{10} refers to the error after 10 iteration steps. For an orthogonal matrix, the standard Kaczmarz method converges in a single step. It is not surprising that it performs extremely well for the almost orthogonal matrices.

In all cases, the distributed Kaczmarz has larger spectral radius (and hence slower convergence). This is to be expected, since the distributed Kaczmarz averages several estimates into one, so bad estimates will increase the error of the average estimate.

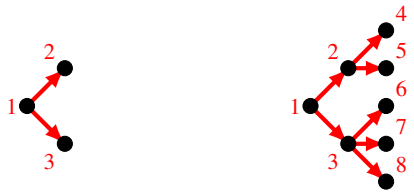


Fig. 4 The two graphs used in numerical experiments with the distributed Kaczmarz method. For these trees, all of the weights were uniform: $w(u, v) = (C(u))^{-1}$.

	Standard Kaczmarz			Distributed Kaczmarz		
	ω_{opt}	ρ_{opt}	e_{10}	ω_{opt}	ρ_{opt}	e_{10}
orthogonal	1.00030	0.00294	0	1.33833	0.33753	$1.5974 \cdot 10^{-5}$
normal	1.07213	0.20188	$1.2793 \cdot 10^{-6}$	1.82299	0.29611	$7.2461 \cdot 10^{-6}$
uniform	1.18634	0.37073	$9.0922 \cdot 10^{-4}$	1.92714	0.82562	$1.49608 \cdot 10^{-1}$

Table 1 Numerical results for a 3×3 system of equations.

	Standard Kaczmarz			Distributed Kaczmarz		
	ω_{opt}	ρ_{opt}	e_{10}	ω_{opt}	ρ_{opt}	e_{10}
orthogonal	1.01585	0.04931	$1.53 \cdot 10^{-13}$	1.76733	0.73919	$2.6757 \cdot 10^{-2}$
normal	1.73543	0.93147	$8.5663 \cdot 10^{-1}$	3.78883	0.99087	$9.0960 \cdot 10^{-1}$
uniform	1.88188	0.92070	$7.1463 \cdot 10^{-1}$	3.73491	0.99890	$7.7508 \cdot 10^{-1}$

Table 2 Numerical results for an 8×8 system of equations.

Acknowledgements This research was supported by the National Science Foundation and the National Geospatial-Intelligence Agency under awards DMS-1830254 and CCF-1750920.

References

1. Dimitri P. Bertsekas and John N. Tsitsiklis, *Parallel and distributed computation: Numerical methods*, Athena Scientific, Nashua, NH, 1997, originally published in 1989 by Prentice-Hall; available for free download at `http://hdl.handle.net/1721.1/3719`.

2. Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al., *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends® in Machine Learning **3** (2011), no. 1, 1–122.
3. Yair Censor, Dan Gordon, and Rachel Gordon, *Component averaging: an efficient iterative parallel algorithm for large and sparse unstructured problems*, Parallel Comput. **27** (2001), no. 6, 777–808. MR 1823354
4. Xuemei Chen, *The Kaczmarz algorithm, row action methods, and statistical learning algorithms*, Frames and harmonic analysis, Contemp. Math., vol. 706, Amer. Math. Soc., Providence, RI, 2018, pp. 115–127. MR 3796634
5. Yuejie Chi and Yue M Lu, *Kaczmarz method for solving quadratic equations*, IEEE Signal Processing Letters **23** (2016), no. 9, 1183–1187.
6. G. Cimmino, *Calcolo approssimato per soluzioni dei sistemi di equazioni lineari*, La Ricerca Scientifica XVI, Series II, Anno IX 1 (1938), 326–333.
7. P. P. B. Eggermont, G. T. Herman, and A. Lent, *Iterative algorithms for large partitioned linear systems, with applications to image reconstruction*, Linear Alg. Appl. **40** (1981), 37–67.
8. Richard Gordon, Robert Bender, and Gabor Herman, *Algebraic reconstruction techniques (ART) for threedimensional electron microscopy and x-ray photography*, Journal of Theoretical Biology **29** (1970), no. 3, 471–481.
9. Robert M. Gower and Peter Richtárik, *Randomized iterative methods for linear systems*, SIAM J. Matrix Anal. Appl. **36** (2015), no. 4, 1660–1690. MR 3432148
10. Jamie Haddock and Deanna Needell, *Randomized projections for corrupted linear systems*, AIP Conference Proceedings, vol. 1978, AIP Publishing, 2018, p. 470071.
11. C. Hamaker and D. C. Solmon, *The angles between the null spaces of X rays*, Journal of Mathematical Analysis and Applications **62** (1978), no. 1, 1–23.
12. Per Christian Hansen, *Discrete inverse problems*, Fundamentals of Algorithms, vol. 7, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2010, Insight and algorithms. MR 2584074
13. G. T. Herman, A. Lent, and H. Hurwitz, *A storage-efficient algorithm for finding the regularized solution of a large, inconsistent system of equations*, J. Inst. Math. Appl. **25** (1980), no. 4, 361–366. MR 578083
14. Gabor T. Herman, Henry Hurwitz, Arnold Lent, and Hsi Ping Lung, *On the Bayesian approach to image reconstruction*, Inform. and Control **42** (1979), no. 1, 60–71. MR 538379
15. Björn Johansson, Maben Rabi, and Mikael Johansson, *A randomized incremental subgradient method for distributed optimization in networked systems*, SIAM Journal on Optimization **20** (2009), no. 3, 1157–1170.
16. Stefan Kaczmarz, *Angenäherte Auflösung von Systemen linearer Gleichungen*, Bulletin International de l'Académie Polonaise des Sciences et des Lettres. Classe des Sciences Mathématiques et Naturelles. Série A, Sciences Mathématiques (1937), 355–357.
17. Goutham Kamath, Paritosh Ramanan, and Wen-Zhan Song, *Distributed randomized Kaczmarz and applications to seismic imaging in sensor network*, 2015 International Conference on Distributed Computing in Sensor Systems, 06 2015, pp. 169–178.
18. Stanisław Kwapień and Jan Mycielski, *On the Kaczmarz algorithm of approximation in infinite-dimensional spaces*, Studia Math. **148** (2001), no. 1, 75–86. MR 1881441 (2003a:60102)
19. Ji Liu, Stephen J Wright, and Srikrishna Sridhar, *An asynchronous parallel randomized Kaczmarz algorithm*, arXiv preprint arXiv:1401.4780 (2014).
20. Nicolas Loizou and Peter Richtárik, *Revisiting randomized gossip algorithms: General framework, convergence rates and novel block and accelerated protocols*, `arXiv:1905.08645`, 2019.
21. Frank Natterer, *The mathematics of computerized tomography*, Teubner, Stuttgart, 1986.
22. I. Necoara, Yu. Nesterov, and F. Glineur, *Linear convergence of first order methods for non-strongly convex optimization*, Math. Program. **175** (2019), no. 1-2, Ser. A, 69–107. MR 3942886
23. Ion Necoara, *Faster randomized block Kaczmarz algorithms*, `arXiv:1902.09946`, 2019.
24. Ion Necoara, Yurii Nesterov, and François Glineur, *Random block coordinate descent methods for linearly constrained optimization over networks*, J. Optim. Theory Appl. **173** (2017), no. 1, 227–254. MR 3626645

25. Angelia Nedic and Asuman Ozdaglar, *Distributed subgradient methods for multi-agent optimization*, IEEE Transactions on Automatic Control **54** (2009), no. 1, 48.
26. Deanna Needell, Nathan Srebro, and Rachel Ward, *Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm*, Math. Program. **155** (2016), no. 1-2, Ser. A, 549–573. MR 3439812
27. Deanna Needell and Joel A. Tropp, *Paved with good intentions: analysis of a randomized block Kaczmarz method*, Linear Algebra Appl. **441** (2014), 199–221. MR 3134343
28. Deanna Needell, Ran Zhao, and Anastasios Zouzias, *Randomized block Kaczmarz method with projection for solving least squares*, Linear Algebra Appl. **484** (2015), 322–343. MR 3385065
29. Ali H Sayed, *Adaptation, learning, and optimization over networks*, Foundations and Trends® in Machine Learning **7** (2014), no. 4-5, 311–801.
30. Kevin Scaman, Francis Bach, Sébastien Bubeck, Laurent Massoulié, and Yin Tat Lee, *Optimal algorithms for non-smooth distributed optimization in networks*, Advances in Neural Information Processing Systems, 2018, pp. 2740–2749.
31. Devavrat Shah, *Gossip algorithms*, Foundations and Trends® in Networking **3** (2008), no. 1, 1–125.
32. Thomas Strohmer and Roman Vershynin, *A randomized Kaczmarz algorithm with exponential convergence*, Journal of Fourier Analysis and Applications **15** (2009), no. 2, 262–278.
33. Kunio Tanabe, *Projection method for solving a singular system of linear equations and its application*, Numer. Math. **17** (1971), 203–214.
34. John Tsitsiklis, Dimitri Bertsekas, and Michael Athans, *Distributed asynchronous deterministic and stochastic gradient optimization algorithms*, IEEE Transactions on Automatic Control **31** (1986), no. 9, 803–812.
35. Douglas B. West, *Introduction to graph theory*, Prentice Hall, Inc., Upper Saddle River, NJ, 1996. MR 1367739
36. Lin Xiao, Stephen Boyd, and Seung-Jean Kim, *Distributed average consensus with least-mean-square deviation*, Journal of Parallel and Distributed Computing **67** (2007), no. 1, 33–46.
37. Kôsaku Yosida, *Functional analysis*, Second edition. Die Grundlehren der mathematischen Wissenschaften, Band 123, Springer-Verlag New York Inc., New York, 1968. MR 0239384
38. Kun Yuan, Qing Ling, and Wotao Yin, *On the convergence of decentralized gradient descent*, SIAM Journal on Optimization **26** (2016), no. 3, 1835–1854.
39. Xin Zhang, Jia Liu, Zhengyuan Zhu, and Elizabeth S. Bentley, *Compressed distributed gradient descent: Communication-efficient consensus over networks*, 2018, `arxiv.org/pdf/1812.04048`.
40. Anastasios Zouzias and Nikolaos M. Freris, *Randomized extended Kaczmarz for solving least squares*, SIAM J. Matrix Anal. Appl. **34** (2013), no. 2, 773–793. MR 3069089

Index