No Journal. manuscript No.
(will be inserted by the editor)

Asynchronous Sequential Inertial Iterations for Common Fixed Points Problems with an Application to Linear Systems

Howard Heaton · Yair Censor

Date: August 15, 2018

Abstract The common fixed points problem requires finding a point in the intersection of fixed points sets of a finite collection of operators. Quickly solving problems of this sort is of great practical importance for engineering and scientific tasks (e.g., for computed tomography). Iterative methods for solving these problems often employ a Krasnosel'skiĭ-Mann type iteration. We present an Asynchronous Sequential Inertial (ASI) algorithmic framework in a Hilbert space to solve common fixed points problems with a collection of nonexpansive operators. Our scheme allows use of out-of-date iterates when generating updates, thereby enabling processing nodes to work simultaneously and without synchronization. This method also includes inertial type extrapolation terms to increase the speed of convergence. In particular, we extend the application of the recent "ARock algorithm" [Peng, Z. et al, SIAM J. on Scientific Computing **38**, A2851-2879, (2016)] in the context of convex feasibility problems. Convergence of the ASI algorithm is proven with no assumption on the distribution of delays, except that they be uniformly bounded. Discussion is provided along with a computational example showing the performance of the ASI algorithm applied in conjunction with a diagonally relaxed orthogonal projections (DROP) algorithm for estimating solutions to large linear systems.

Keywords convex feasibility problem · asynchronous sequential iterations · nonexpansive operator · fixed point iteration · Kaczmarz method · DROP algorithm

Howard Heaton
Department of Mathematics, University of California Los Angeles, Los Angeles, CA, 90095, USA
E-mail: heaton@math.ucla.edu

Yair Censor
Department of Mathematics, University of Haifa, Mt. Carmel, Haifa, 3498838, Israel
E-mail: yair@math.haifa.ac.il

1 Introduction

In this paper, we investigate common fixed points problems with nonexpansive operators in Hilbert space. In this framework, we focus on the special case of convex feasibility problems. Solving convex feasibility problems is of practical interest because it has many applications, including areas in image recovery (e.g., computed tomography (CT)) [20], radiation therapy treatment planning [11], electron microscopy, seismology, and others, see, e.g., [20] for a collection of references. Since the speed of individual processing cores stopped increasing significantly and multi-core chips are becoming increasingly available [23], schemes for solving these problems in parallel are of great use. Notably, these include block-iterative methods (see, e.g., [1, 5, 10, 15, 30] and further references in [12]) and string-averaging methods [13].

Although parallel methods are being developed to utilize several processing nodes at once, these algorithms are often still expressible, at some level, as sequential algorithms. For example, individual block operators may be computed using several processing nodes in parallel, but these results must be merged together (e.g., via a convex combination) and then passed to the next block operator in the sequence. That is, the collection of block operators are applied successively. In this work, we show that such inherently sequential algorithms can be executed in parallel without the need for synchronization. This introduces robustness to dropped network transmissions, introduces a higher level of parallelization, and allows multiple processing nodes to run independently, thereby allowing faster computations of each iterate.

Related Works. Asynchronous algorithms date back at least to the early work of Chazan and Miranker [18], where they used the phrase "chaotic relaxations" (now often termed asynchronous relaxations) for the solution of linear systems. This was followed by numerous works (e.g., Strikwerda [39]). For a discussion on asynchronous algorithms, see the summary work of Frommer and Szyld [22]. In Bertsekas and Tsitsiklis [7], the distinction is made that *totally asynchronous* algorithms can tolerate arbitrarily large update delays while *partially asynchronous* algorithms are not guaranteed to work unless there is an upper bound on these delays. The analysis in [7] is both for totally and partially synchronous algorithms.

Elsner, Koltracht, and Neumann [21] proved convergence of a sequence generated by sequential application of partially asynchronous nonlinear paracontractions. Their work was done in finite-dimensional Euclidean space and used bounded delays of out-of-date information (i.e., partial asynchrony). This provides direct application to solving linear systems of equations, e.g., through application of Kaczmarz's method [29], which is also known in the image reconstruction from projections literature as the Algebraic Reconstruction Technique (ART), as it was discovered there by Gordon, Bender, and Herman [24]. More recently, Peng, Xu, Yan, and Yin [36] proposed the ARock algorithm, which is an asynchronous algorithmic framework. The task at hand there is to find a fixed point of a single separable nonexpansive operator. They accomplished this by generating updates on random blocks of coordinates, using potentially out-of-date information. A novelty in their approach was that it allowed for potentially unbounded delays on the out-of-date information. Similarly to this original work on ARock [36] and the subsequent work by Hannah and Yin [25], we achieve our results through establishing the monotonicity of a sequence that includes the classical error added to the error introduced from using out-of-date iterates to perform the updates.

Advantages of Asynchronous Algorithms. Asynchronous methods have several desirable features. First, each processing node performs its computations independently whereas traditionally when iterative methods are executed using multiple nodes the speed is limited by the slowest node. This means that asynchronous approaches may reduce the average time required to generate successive updates, especially when load-balancing differences do not occur among the nodes. Second, asynchronous methods are robust to failed transmissions through a network. In a synchronous model, if the output from a slave node is sent but does not make it to the master node used to generate a new iterate x^{k+1}, then the processing node must perform its computation and send its output again before x^{k+1} can be computed. However, in an asynchronous model, iterates are continually generated and need not be in a specific order. Lastly, this approach may simplify the software code to implement such algorithms since if we are able to estimate the bound on delays, we do not need to keep track of when information is received from each node. Instead, we need only fetch the most recent output, compute the update x^{k+1}, and then send this back to the node for the next computation (c.f. Algorithm 4 below).

Our Contribution. This work establishes the convergence of a general partially asynchronous iterative algorithmic framework in Hilbert space for common fixed points problems. This allows out-of-date iterates to be used when there is an upper bound on the delays. In practical terms, this enables processing nodes to run in parallel while still executing an inherently sequential algorithm. And, this allows for processors to be well-utilized even without load-balancing. The present work is distinct from that of [21] since there the operators were paracontractions on finite-dimensional Euclidean space while we use nonexpansive operators in Hilbert space of possibly infinite dimension. Moreover, our algorithmic structure includes an inertial term and our analysis takes a different approach. This work is chiefly an extension of the ARock algorithm in the context of convex feasibility problems. The ARock algorithm [25,36] and results in [25] generate a sequence stochastically while the current work need not generate iterations stochastically. Our work also differs from [7] since there fixed points were found for the special case of a nonexpansive operator with respect to the max norm on a Euclidean space.

Outline. In Section 2, we present the framework for the problem at hand and define the notation for asynchrony used in this work. Our proposed algorithm is given in Section 3. Section 4 presents our mathematical analysis, which leads to the proof of the main result, Theorem 1. We then direct our attention to apply Theorem 1 to solving large systems of linear equations in Section 5. There we show that the Diagonally Relaxed Orthogonal Projection (DROP) algorithm of Censor, Elfving, Herman, and Nikazad [14] and Kaczmarz's algorithm [29] for linear systems can be used within the ASI framework and provide a pseudo-code sample implementation of the ASI algorithm with this application in mind. A computational example is provided in Section 7 applying the ASI algorithm with DROP to a model CT image reconstruction problem. We make some concluding remarks in Section 8.

2 Convex Feasibility Problem in a Common Fixed Points Framework

Let $\{C_i\}_{i=1}^m$ be a finite family of closed convex sets in a Hilbert space $\mathcal{H}$ with the inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$, respectively. The associated *convex feasibility problem* (CFP) is

$$\text{Find } x^* \in C := \bigcap_{i=1}^{m} C_i, \tag{1}$$

where $C \neq \emptyset$. A common approach to solving (1) is to generate a sequence of iterates $\{x^k\}_{k\in\mathbb{N}}$ in $\mathcal{H}$ with $x^1 \in \mathcal{H}$ arbitrary via an iterative process

$$x^{k+1} := Q_k\left(x^k\right), \quad \text{for all } k \in \mathbb{N}, \tag{2}$$

where $\{Q_k\}_{k\in\mathbb{N}}$ is a sequence of operators. In this work, we instead use an iteration of the form

$$x^{k+1} := F_k\left(x^k, \hat{x}^k\right), \quad \text{for all } k \in \mathbb{N}, \tag{3}$$

where $\{F_k\}_{k\in\mathbb{N}}$ is a sequence of mappings from $\mathcal{H} \times \mathcal{H}$ to $\mathcal{H}$ and $\hat{x}^k$ either equals x^k, or equals a previous iterate x^{k-j} for some $j \in \mathbb{N}$. This enables out-of-date information to be used in creating successive iterates.

Suppose we have a collection of w processing nodes. For each iteration index k, we let $d^k \in \mathbb{Z}^w_{\geq 0}$ give the delay information for each of the nodes. This means that if the last output from the i-th node N_i was computed using the iterate x^{k-j}, then the i-th component of d^k gives this delay amount, i.e., $(d^k)_i = j$. Let $\{i_k\}_{k\in\mathbb{N}}$ be an index sequence identifying the index of the operator whose output will be used to compute x^{k+1}, where $i_k \in \{1, 2, \ldots, m\}$ for all $k \in \mathbb{N}$. Then $\hat{x}^k$ is the last iterate sent to the i_k-th node and we write

$$\hat{x}^k := x^{k-(d^k)_{i_k}}, \quad \text{for all } k \in \mathbb{N}. \tag{4}$$

For example, if $k = 12$ is the current iterative step, if $i_{12} = 4$, and if the last output from the 4-th node was generated using an iterate that is now 3 iterative steps out-of-date, then

$$(d^k)_{i_k} = (d_{12})_4 = 3, \tag{5}$$

and so

$$\hat{x}^k = x^{k-(d^k)_{i_k}} = x^{12-3} = x^9. \tag{6}$$

This shows that the iterate x^{13} will be computed using x^{12} and x^9.

The following definitions will be used in the sequel.

Definition 1 Let $\mathcal{H}$ be a Hilbert space and $D \subseteq \mathcal{H}$ be nonempty. Then an operator $T : D \to \mathcal{H}$ is

(i) *firmly nonexpansive* if

$$\forall\ x, y \in D, \quad \|T(x) - T(y)\|^2 \leq \langle x - y, T(x) - T(y) \rangle\,; \tag{7}$$

(ii) *nonexpansive* if T is Lipschitz with constant 1 so that

$$\forall\ x, y \in D, \quad \|T(x) - T(y)\| \leq \|x - y\|; \tag{8}$$

(iii) *quasi-nonexpansive* if, denoting its fixed points set $\mathrm{Fix}(T) := \{x \in \mathcal{H} \mid x = T(x)\}$,

$$\forall\ x \in D,\ y \in \mathrm{Fix}(T), \quad \|T(x) - y\| \leq \|x - y\|. \tag{9}$$

We say that T is *strictly nonexpansive* or *strictly quasi-nonexpansive* if the inequalities in (8) or (9), respectively, are strict.

Definition 2 Let $T : \mathcal{H} \to \mathcal{H}$ and $\alpha \in [0, 2]$. The operator $T_\alpha : \mathcal{H} \to \mathcal{H}$ defined by $T_\alpha := (1 - \alpha)\mathrm{Id} + \alpha T$ is called an α-relaxation of the operator T, where Id is the identity operator.

Definition 3 A *paracontraction* is a continuous strictly quasi-nonexpansive operator.

Remark 1 The collections of paracontractions and of nonexpansive operators form partially overlapping but distinct subsets of the collection of quasi-nonexpansive operators. Their intersection is nonempty since projections are contained in each of them. However, the identity mapping and reflections are nonexpansive, but are not paracontractions. Additionally, a paracontraction need not be Lipschitz continuous.

The following definition follows [17, Definition 5.1.1].

Definition 4 A sequence $\{i_k\}_{k\in\mathbb{N}}$ is called an *almost cyclic control* on $[m] := \{1, 2, \ldots, m\}$ if $i_k \in [m]$ for all $k \in \mathbb{N}$ and there exists an integer $M \geq m$ (called the *almost cyclicality constant*) such that, for each $k \in \mathbb{N}$, $[m] \subseteq \{i_{k+1}, i_{k+2}, \ldots, i_{k+M}\}$.

3 The Asynchronous Sequential Inertial (ASI) Algorithm

We now describe our proposed algorithm. Consider a collection of m nonexpansive operators $\{T_i\}_{i=1}^m$ on $\mathcal{H}$ with a common fixed point. We seek to solve (1) where $C_i = \mathrm{Fix}(T_i)$ for all $i \in [m]$. For notational convenience, we define

$$S_i := \mathrm{Id} - T_i, \quad \text{for all } i \in [m]. \tag{10}$$

Our Asynchronous Sequential Inertial (ASI) algorithm is as follows.

Algorithm 1: Asynchronous Sequential Inertial (ASI) Algorithm

Let $x^1 \in \mathcal{H}$ be arbitrary, $\{\lambda_k\}_{k\in\mathbb{N}}$ be such that $\lambda_k \in (0, 1)$ for all $k \in \mathbb{N}$, and $\{i_k\}_{k\in\mathbb{N}}$ be an almost cyclic control on $[m]$. For each $k \in \mathbb{N}$ set

$$x^{k+1} := \begin{cases} x^k, & \text{if } k \leq \sup_{k\in\mathbb{N}} \|d^k\|_\infty, \\ x^k - \lambda_k S_{i_k}\left(\hat{x}^k\right), & \text{otherwise.} \end{cases} \tag{11}$$

In the special case where $\hat{x}^k = x^k$ for each $k \in \mathbb{N}$ and we have a single operator T (i.e., $m = 1$), we obtain

$$x^{k+1} := T_\lambda \left(x^k\right) = (1-\lambda)x^k + \lambda T \left(x^k\right), \tag{12}$$

which is precisely the Krasnosel'skiĭ-Mann (KM) iteration, see, e.g., [31, 34] for the original works and [4, Section 5.2] for a summary. This iteration does not allow any delays and generates a sequence that weakly converges to a fixed point of T. The primary result of our current work is Theorem 1, which states that a sequence $\{x^k\}_{k\in\mathbb{N}}$ generated by the ASI algorithm converges weakly to a solution of (1) when the sequence of delays $\{d^k\}_{k\in\mathbb{N}}$ is uniformly bounded in the sup norm by some $\tau \geq 0$ and the step sizes $\{\lambda_k\}_{k\in\mathbb{N}}$ are bounded above by $1/(2\tau + 1)$.

Before presenting our analysis, we provide a remark and illustration of the ASI algorithm to give the reader some intuition. The computation of x^{k+1} can be expressed in two parts. The first part is a convex combination of x^k and $T_{i_k}(\hat{x}^k)$ to form the point

$$y^k := (1-\lambda_k)x^k + \lambda_k T_{i_k} \left(\hat{x}^k\right). \tag{13}$$

The second part is an inertial term that estimates the direction of the solution, given x^k and $\hat{x}^k$. The term y^k is added to the inertial term to yield x^{k+1}, i.e.,

$$\begin{aligned} x^{k+1} &= x^k - \lambda_k S_{i_k}\left(\hat{x}^k\right) \\ &= x^k - \lambda_k \left(\hat{x}^k - T_{i_k}\left(\hat{x}^k\right)\right) \\ &= \underbrace{(1-\lambda_k)x^k + \lambda_k T_{i_k}\left(\hat{x}^k\right)}_{\text{convex combination}} + \underbrace{\lambda_k\left(x^k - \hat{x}^k\right)}_{\text{inertial term}}. \end{aligned} \tag{14}$$

Since the iterates are, on average, moving closer to a solution, this inertial term may accelerate the convergence by using previous information along with the current iterate to estimate the direction toward a solution. This effect of inertial terms is known in the literature, see, e.g., [2, 32, 33, 35, 37]. The right-hand side of (14) also illustrates the distinction of ASI from the algorithm of [21], which uses only the convex combination part of (14) without the inertial term.

To illustrate the ASI algorithm graphically, let C_1 and C_2 be two closed convex sets with nonempty intersection and let T_1 and T_2 be relaxations of the projections P_{C_1} and P_{C_2} onto the sets C_1 and C_2, respectively. In this case, Figure 1 shows how x^{k+1} is generated from x^k and $\hat{x}^k$.

4 Mathematical Analysis of the ASI Algorithm

In this section, we provide several lemmas culminating in our primary convergence result in Theorem 1. We begin with the following lemma on the *demi-closedness principle*, which is a slight generalization of Cegielski [8, Lemma 3.2.5].

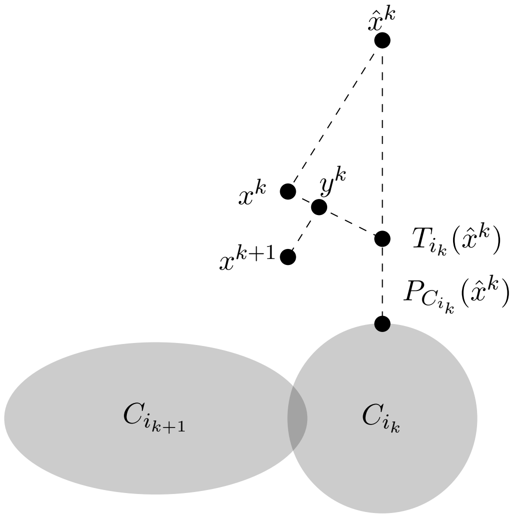


Fig. 1: Illustration of a full iteration of the ASI algorithm with two convex sets and the T_i's as relaxed projections onto the sets.

Lemma 1 *Let $\{T_i\}_{i=1}^m$ be a finite family of nonexpansive operators with a common fixed point and let $y \in \mathcal{H}$ be a weak cluster point of a sequence $\{x^k\}_{k\in\mathbb{N}}$. If $\|T_i x^k - x^k\| \longrightarrow 0$ for all $i \in [m]$, then*

$$y \in \bigcap_{i=1}^{m} \operatorname{Fix}(T_i). \tag{15}$$

Proof By hypothesis, there is a subsequence $\{x^{n_k}\}_{k\in\mathbb{N}} \subseteq \{x^k\}_{k\in\mathbb{N}}$ such that $x^{n_k} \rightharpoonup y$. Pick any $j \in [m]$. Then, using the triangle inequality, we deduce

$$\begin{aligned}
\liminf_{k\to\infty} \|x^{n_k} - y\| &\geq \liminf_{k\to\infty} \|T_j x^{n_k} - T_j y\| \\
&= \liminf_{k\to\infty} \|T_j x^{n_k} - x^{n_k} + x^{n_k} - T_j y\| \\
&\geq \liminf_{k\to\infty} (\|x^{n_k} - T_j y\| - \|T_j x^{n_k} - x^{n_k}\|) \\
&= \liminf_{k\to\infty} \|x^{n_k} - T_j y\|.
\end{aligned} \tag{16}$$

A 1967 lemma of Opial, see, e.g., [8, Lemma 3.2.4], states that if $z^k \rightharpoonup z \in \mathcal{H}$ and $z' \in \mathcal{H}$ with $z \neq z'$, then

$$\liminf_{k\to\infty} \|z^k - z'\| > \liminf_{k\to\infty} \|z^k - z\|. \tag{17}$$

Consequently, if $T_j y \neq y$, then (16) and (17) together imply

$$\liminf_{k\to\infty} \|x^{n_k} - y\| \geq \liminf_{k\to\infty} \|x^{n_k} - T_j y\| > \liminf_{k\to\infty} \|x^{n_k} - y\|, \tag{18}$$

a contradiction. Whence, $T_j y = y$ for each j and the result follows. ∎

This lemma helps obtain the following proposition, which is key in obtaining our main result. The second half of the proof of Proposition 1 is based on the result found in [8, Corollary 3.3.3], which is credited there to Bauschke and Borwein [3, Theorem 2.16(ii)]. This corollary in [8] established that Fejér monotone sequences with respect to a closed set have at most one weak cluster point. Their proof idea is modified below to apply to the current setting.

Proposition 1 *Let $\{T_i\}_{i=1}^m$ be a family of nonexpansive operators on $\mathcal{H}$ with a common fixed point and let $\{x^k\}_{k\in\mathbb{N}}$ be a sequence in $\mathcal{H}$. If for every*

$$z \in C := \bigcap_{i=1}^{m} \mathrm{Fix}(T_i) \tag{19}$$

the sequence $\{\|x^k - z\|\}_{k\in\mathbb{N}}$ converges and $\|T_i x^k - x^k\| \longrightarrow 0$ for all $i \in [m]$, then the sequence $\{x^k\}_{k\in\mathbb{N}}$ converges weakly to some $x^* \in C$.

Proof Let $z \in C$. The triangle inequality and the convergence of $\{\|x^k - z\|\}_{k\in\mathbb{N}}$ imply that the sequence $\{x^k\}_{k\in\mathbb{N}}$ is bounded. Consequently, it has a weakly convergent subsequence $\{x^{n_k}\}_{k\in\mathbb{N}} \subseteq \{x^k\}_{k\in\mathbb{N}}$. For each such subsequence, we may apply Lemma 1 to assert the weak cluster point is contained in C. We show below that the sequence $\{x^k\}_{k\in\mathbb{N}}$ has at most one weak cluster point in C. Thus, there is precisely one weak cluster point of $\{x^k\}_{k\in\mathbb{N}}$ and it is contained in C, from which the result will follow.

It remains to verify that there is at most one cluster point of $\{x^k\}_{k\in\mathbb{N}}$ in C. Define the function $f : C \to \mathbb{R}$ by

$$f(z) := \lim_{k\to\infty} \left(\|x^k - z\|^2 - \|z\|^2 \right), \tag{20}$$

which, by hypothesis, converges for each $z \in C$. Note that

$$\|x^k - z\|^2 - \|z\|^2 = \left(\|x^k\|^2 - 2\left\langle x^k, z\right\rangle + \|z\|^2 \right) - \|z\|^2 = \|x^k\|^2 - 2\left\langle x^k, z\right\rangle. \tag{21}$$

Now let $\{x^{m_k}\}_{k\in\mathbb{N}}$ and $\{x^{n_k}\}_{k\in\mathbb{N}}$ be subsequences of $\{x^k\}_{k\in\mathbb{N}}$ weakly converging to two distinct limit points p and q in C, respectively. Then

$$\begin{aligned} 2\lim_{k\to\infty} \left\langle x^k, p - q\right\rangle &= \lim_{k\to\infty} \left(\left(\|x^k\|^2 - 2\left\langle x^k, q\right\rangle \right) - \left(\|x^k\|^2 - 2\left\langle x^k, p\right\rangle \right) \right) \\ &= f(q) - f(p), \end{aligned} \tag{22}$$

i.e., the limit exists. However,

$$\lim_{k\to\infty} \left\langle x^{m_k}, p - q\right\rangle = \left\langle p, p - q\right\rangle, \tag{23}$$

and

$$\lim_{k\to\infty} \left\langle x^{n_k}, p - q\right\rangle = \left\langle q, p - q\right\rangle. \tag{24}$$

Since the limit in (22) exists, the limits in (23) and (24) must be equal, which implies

$$\left\langle p, p - q\right\rangle = \left\langle q, p - q\right\rangle \quad\Longrightarrow\quad \|p - q\|^2 = 0 \quad\Longrightarrow\quad p = q. \tag{25}$$

This shows the weak cluster point is unique and completes the proof. ■

The above proposition is essential for our convergence result. All our subsequent lemmas compiled together show that the assumptions of Proposition 1 hold for sequences generated by the ASI algorithm.

We outline the remainder of this section as follows. We first give in Lemma 2 a fundamental inequality about sequences $\{x^k\}_{k\in\mathbb{N}}$ generated by the ASI algorithm. This inequality is used in Lemma 3 to show that the sequence $\{\xi_k\}_{k\in\mathbb{N}}$ converges, where ξ_k is a sum of the classical distance $\|x^k - z\|$ for some $z \in C$ and finitely many terms of the form $c_i\|x^{k+1-i} - x^{k-i}\|$. We then use the inequality (39) in the proof of this lemma to verify that $\|x^{k+1} - x^k\| \longrightarrow 0$ and that the sequence $\{\|x^k - z\|\}_{k\in\mathbb{N}}$ converges for each $z \in C$. Following this, in Lemma 6 we prove that $\|T_i x^k - x^k\| \longrightarrow 0$ for each $i \in [m]$. With these results, we show that the hypotheses of Proposition 1 hold for any sequence $\{x^k\}_{k\in\mathbb{N}}$ generated by the ASI algorithm.

Lemma 2 *Let $z \in C$ and let $\mu > 0$. Let $\{x^k\}_{k\in\mathbb{N}}$ be any sequence generated by the ASI algorithm. If the delay vectors are uniformly bounded in the sup norm by some $\tau \geq 0$, then*

$$\begin{aligned} \left\|x^{k+1} - z\right\|^2 \leq \left\|x^k - z\right\|^2 + \mu \sum_{\ell=1}^{\tau} \left\|x^{k+1-\ell} - x^{k-\ell}\right\|^2 \\ - \lambda_k \left\|S_{i_k}\left(\hat{x}^k\right)\right\|^2 \Big(1 - \lambda_k\left(1 + \tau/\mu\right)\Big). \end{aligned} \tag{26}$$

Proof First observe that

$$\begin{aligned} \|x^{k+1} - z\|^2 &= \|x^k - \lambda_k S_{i_k}\left(\hat{x}^k\right) - z\|^2 \\ &= \|x^k - z\|^2 - 2\lambda_k \left\langle x^k - z, S_{i_k}\left(\hat{x}^k\right)\right\rangle + \lambda_k^2 \|S_{i_k}\left(\hat{x}^k\right)\|^2. \end{aligned} \tag{27}$$

We may split the cross-term into the two expressions α_k and β_k via

$$-2\lambda_k \left\langle S_{i_k}\left(\hat{x}^k\right), x^k - z\right\rangle = \underbrace{-2\lambda_k \left\langle S_{i_k}\left(\hat{x}^k\right), \hat{x}^k - z\right\rangle}_{\alpha_k} + \underbrace{2\lambda_k \left\langle S_{i_k}\left(\hat{x}^k\right), x^k - \hat{x}^k\right\rangle}_{\beta_k}. \tag{28}$$

Note that $\frac{1}{2}S_{i_k} = \frac{1}{2}\left(\mathrm{Id} - T_{i_k}\right)$ is firmly nonexpansive and that $\frac{1}{2}S_{i_k}(z) = 0$. Thus,

$$\begin{aligned} \|S_{i_k}\left(\hat{x}^k\right)\|^2 &= 4\|\frac{1}{2}S_{i_k}\left(\hat{x}^k\right)\|^2 \\ &= 4\|\frac{1}{2}S_{i_k}\left(\hat{x}^k\right) - \frac{1}{2}S_{i_k}(z)\|^2 \\ &\leq 4\left\langle \frac{1}{2}S_{i_k}\left(\hat{x}^k\right) - \frac{1}{2}S_{i_k}(z), \hat{x}^k - z\right\rangle \\ &= 2\left\langle S_{i_k}\left(\hat{x}^k\right), \hat{x}^k - z\right\rangle. \end{aligned} \tag{29}$$

This implies

$$\alpha_k = -2\lambda_k \left\langle S_{i_k}\left(\hat{x}^k\right), \hat{x}^k - z\right\rangle \leq -\lambda_k \|S_{i_k}\left(\hat{x}^k\right)\|^2. \tag{30}$$

Application of the triangle inequality and the fact that $\|d^k\|_\infty \leq \tau$ for all $k \in \mathbb{N}$ yield

$$\begin{aligned}
\beta_k &= -2\lambda_k \left\langle S_{i_k}\left(\hat{x}^k\right), x^k - \hat{x}^k \right\rangle \\
&= -2\lambda_k \sum_{\ell=1}^{(d^k)_{i_k}} \left\langle S_{i_k}\left(\hat{x}^k\right), x^{k+1-\ell} - x^{k-\ell} \right\rangle \\
&\leq 2\lambda_k \sum_{\ell=1}^{(d^k)_{i_k}} \|S_{i_k}\left(\hat{x}^k\right)\| \|x^{k+1-\ell} - x^{k-\ell}\| \\
&\leq 2\lambda_k \sum_{\ell=1}^{\tau} \|S_{i_k}\left(\hat{x}^k\right)\| \|x^{k+1-\ell} - x^{k-\ell}\|.
\end{aligned} \tag{31}$$

Note that the second line in (31) holds since the sum is telescoping and that the final line holds since $\|d^k\|_\infty \leq \tau$. Using the fact that $0 \leq (a-b)^2 = a^2 + b^2 - 2ab$ for $a, b \in \mathbb{R}$ implies $ab \leq \frac{1}{2}(a^2 + b^2)$, we deduce

$$\begin{aligned}
\beta_k &\leq \lambda_k \sum_{\ell=1}^{\tau} \frac{\lambda_k}{\mu} \|S_{i_k}\left(\hat{x}^k\right)\|^2 + \frac{\mu}{\lambda_k} \|x^{k+1-\ell} - x^{k-\ell}\|^2 \\
&= \frac{\tau\lambda_k^2}{\mu} \|S_{i_k}\left(\hat{x}^k\right)\|^2 + \mu \sum_{\ell=1}^{\tau} \|x^{k+1-\ell} - x^{k-\ell}\|^2.
\end{aligned} \tag{32}$$

Combining (27), (30) and (32), we obtain the desired result. ■

To prove the convergence of iterates to solutions of fixed point problems, we typically need some sort of monotonicity. However, when using out-of-date iterates to generate the sequence, the inequality $\|x^{k+1} - z\| \leq \|x^k - z\|$ does *not* necessarily hold. In [25], the authors were able to construct a sequence that is monotonic in expectation by including both the classical error $\|x^k - z\|$ and terms of the form $c_\ell \|x^{k+1-\ell} - x^{k-\ell}\|$. We also use the idea of adding terms of this form with the inequality in Lemma 2 to obtain convergence of a sequence that sums the classical error and a finite number of these discrepancy terms. This is stated formally in the following lemma.

Lemma 3 *Let $z \in C$ and let $\mu > 0$. Let $\{x^k\}_{k\in\mathbb{N}}$ be any sequence generated by the ASI algorithm. Assume that the delay vectors are uniformly bounded in the sup norm by some $\tau \geq 0$ and that there is $\varepsilon > 0$ such that*

$$0 < \varepsilon \leq \lambda_k \leq \frac{1}{1 + \tau(1/\mu + \mu) + \varepsilon}, \quad \textit{for all } k \in \mathbb{N}. \tag{33}$$

The sequence $\{\xi_k\}_{k\in\mathbb{N}}$ defined by

$$\xi_k := \|x^k - z\|^2 + \sum_{\ell=1}^{\tau} c_\ell \|x^{k+1-\ell} - x^{k-\ell}\|^2, \tag{34}$$

where

$$c_j := (\tau + 1 - j)\mu + \varepsilon, \quad \textit{for all } j \in [\tau + 1], \tag{35}$$

is a convergent sequence.

Proof The sequence $\{\xi_k\}_{k\in\mathbb{N}}$ is nonnegative; thus, it suffices to verify that it is monotonically decreasing. By Lemma 2, we have

$$\begin{aligned}\xi^{k+1} &= \|x^{k+1} - z\|^2 + \sum_{\ell=1}^{\tau} c_\ell \|x^{k+2-\ell} - x^{k+1-\ell}\|^2 \\ &\leq \|x^k - z\|^2 + \sum_{\ell=1}^{\tau} \mu \|x^{k+1-\ell} - x^{k-\ell}\|^2 \\ &\quad - \lambda_k \|S_{i_k}\left(\hat{x}^k\right)\|^2 \Big(1 - \lambda_k(1+\tau/\mu)\Big) + \sum_{\ell=1}^{\tau} c_\ell \|x^{k+2-\ell} - x^{k+1-\ell}\|^2.\end{aligned} \tag{36}$$

Reindexing the sums and noting $\mu + c_{j+1} = c_j$, this simplifies to

$$\begin{aligned}\xi^{k+1} &\leq \|x^k - z\|^2 + \sum_{j=1}^{\tau} (\mu + c_{j+1}) \|x^{k+1-j} - x^{k-j}\|^2 + c_1 \|x^{k+1} - x^k\|^2 \\ &\quad - \lambda_k \|S_{i_k}\left(\hat{x}^k\right)\|^2 \Big(1 - \lambda_k(1+\tau/\mu)\Big) - c_{\tau+1} \|x^{k+1-\tau} - x^{k-\tau}\|^2 \\ &= \xi_k + c_1 \|x^{k+1} - x^k\|^2 - c_{\tau+1} \|x^{k+1-\tau} - x^{k-\tau}\|^2 \\ &\quad - \lambda_k \|S_{i_k}\left(\hat{x}^k\right)\|^2 \Big(1 - \lambda_k(1+\tau/\mu)\Big).\end{aligned} \tag{37}$$

The ASI algorithm generates successive iterates such that

$$\|x^{k+1} - x^k\|^2 \leq \|x^k - \lambda_k S_{i_k}\left(\hat{x}^k\right) - x^k\|^2 = \lambda_k^2 \|S_{i_k}\left(\hat{x}^k\right)\|^2. \tag{38}$$

Consequently,

$$\xi_{k+1} \leq \xi_k - \lambda_k \|S_{i_k}\left(\hat{x}^k\right)\|^2 \Big(1 - \lambda_k(1+\tau/\mu + c_1)\Big) - c_{\tau+1} \|x^{k+1-\tau} - x^{k-\tau}\|^2. \tag{39}$$

With our assumption in (33), the fact $c_1 = \tau\mu + \varepsilon$, and the fact $c_{\tau+1} = \varepsilon > 0$, (39) shows that $\xi_{k+1} \leq \xi_k$, for all $k \in \mathbb{N}$, from which the result follows. ■

Remark 2 Naturally, we may seek to choose μ that maximizes the allowable step sizes. Such a choice of μ minimizes the function $f : (0,\infty) \to \mathbb{R}$ defined by $f(x) := x + 1/x$. Since the single critical point and minimizer of f occurs at $x = 1$, choosing $\mu = 1$ in Lemma 3 yields the optimal step size bound.

Although the above lemma establishes the convergence of $\{\xi_k\}_{k\in\mathbb{N}}$, it does not guarantee that a sequence generated by terms of the form $\|x^k - z\|$ will converge. However, we are able to verify this in the following lemma by noting the inequality in (39).

Lemma 4 *If the assumptions of Lemma 3 hold, then* $\|x^{k+1} - x^k\| \longrightarrow 0$ *and the sequence* $\{\|x^k - z\|\}_{k\in\mathbb{N}}$ *converges.*

Proof The inequality (39) implies

$$0 \leq c_{\tau+1}\|x^{k+1-\tau} - x^{k-\tau}\|^2 \leq \xi_k - \xi_{k+1}. \tag{40}$$

Letting $k \longrightarrow \infty$, the convergence of $\{\xi_k\}_{k\in\mathbb{N}}$ together with the squeeze (sandwich) theorem implies

$$\lim_{k\to\infty} c_{\tau+1}\|x^{k+1-\tau} - x^{k-\tau}\|^2 = 0. \tag{41}$$

Since $c_{\tau+1} = \varepsilon > 0$, we obtain

$$\lim_{k\to\infty} \|x^{k+1} - x^k\| = 0. \tag{42}$$

Let ξ_* be the limit of $\{\xi_k\}_{k\in\mathbb{N}}$. Then combining (34) and (42) and letting $k \longrightarrow \infty$ yields

$$\xi_* = \lim_{k\to\infty} \xi_k = \lim_{k\to\infty} \|x^k - z\|^2, \tag{43}$$

from which the result follows since the square root function is continuous. ■

Lemma 5 *If a sequence $\{x^k\}_{k\in\mathbb{N}}$ satisfies $\|x^{k+1}-x^k\| \longrightarrow 0$ and the delay vectors are uniformly bounded in the sup norm by some $\tau \geq 0$, then $\|x^k - \hat{x}^k\| \longrightarrow 0$.*

Proof The fact that $\|d^k\|_\infty \leq \tau$, for all $k \in \mathbb{N}$, yields

$$0 \leq \|x^k - \hat{x}^k\| = \|x^k - x^{k-(d^k)_{i_k}}\| \leq \sum_{j=1}^{\tau} \|x^{k+1-j} - x^{k-j}\|. \tag{44}$$

Letting $k \longrightarrow \infty$ in (44), the right-hand side goes to zero since the sum contains finitely many terms. The result is then obtained through the squeeze theorem. ■

Lemma 6 *Let $\{x^k\}_{k\in\mathbb{N}}$ be a sequence generated by the ASI algorithm. Assume there is $\varepsilon > 0$ such that $\lambda_k \geq \varepsilon$, for all $k \in \mathbb{N}$. If the delay vectors are uniformly bounded in the sup norm by some $\tau \geq 0$ and $\|x^{k+1}-x^k\| \longrightarrow 0$, then $\|T_i x^k - x^k\| \longrightarrow 0$, for each $i \in [m]$.*

Proof Let $i \in [m]$ and let $T_{i,\lambda}$ be the λ-relaxation of T_i. If $\{t_k\}$ is an increasing sequence such that $t_k \in \mathbb{N}$, for all $k \in \mathbb{N}$, then

$$\|T_{i,\lambda_{t_k}}(x^k)-x^k\| = \|(1-\lambda_{t_k})x^k+\lambda_{t_k}T_i(x^k)-x^k\| = \lambda_{t_k}\|T_i(x^k)-x^k\| \geq \varepsilon\|T_i(x^k)-x^k\|. \tag{45}$$

This inequality reveals that if $\|T_{i,\lambda_{t_k}}(x^k) - x^k\| \longrightarrow 0$, then $\|T_i(x^k) - x^k\| \longrightarrow 0$. Next, we verify that the first limit holds, by setting, for each $k \in \mathbb{N}$, t_k to be the smallest index greater than or equal to k such that $i_{t_k} = i$. This implies

$$\begin{aligned} x^{t_k+1} &= x^{t_k} - \lambda_{t_k} S_i\left(\hat{x}^{t_k}\right) \\ &= x^{t_k} - \lambda_{t_k}\hat{x}^{t_k} + \lambda_i T_i\left(\hat{x}^{t_k}\right) \\ &= T_{i,\lambda_{t_k}}\left(\hat{x}^{t_k}\right) + \left(\hat{x}^{t_k} - \hat{x}^{t_k}\right). \end{aligned} \tag{46}$$

Observe also that

$$
\begin{aligned}
\|T_{i,\lambda_{t_k}}\left(x^k\right) - x^{t_k+1}\| &= \|T_{i,\lambda_{t_k}}\left(x^k\right) - T_{i,\lambda_k}\left(\hat{x}^{t_k}\right) - \left(x^{t_k} - \hat{x}^{t_k}\right)\| \\
&\leq \|T_{i,\lambda_{t_k}}\left(x^k\right) - T_{i,\lambda_{t_k}}\left(\hat{x}^{t_k}\right)\| + \|x^{t_k} - \hat{x}^{t_k}\| \\
&\leq \|x^k - \hat{x}^{t_k}\| + \|x^{t_k} - \hat{x}^{t_k}\|,
\end{aligned}
\tag{47}
$$

where the first equality holds by (46) and the final inequality follows from the fact the λ_{t_k}-relaxation of a nonexpansive operator T_i is nonexpansive. Due to the bound on delays and the almost cyclicity of $\{i_k\}_{k\in\mathbb{N}}$, we know that $\hat{x}^{t_k} = x^j$ for some $k - \tau \leq j \leq t_k \leq k + M$, where M is the almost cyclicality constant of $\{i_k\}_{k\in\mathbb{N}}$. Thus, repeated application of the triangle inequality with (47) yields

$$
\begin{aligned}
\|T_{i,\lambda_{t_k}}\left(x^k\right) - x^{t_k+1}\| &\leq \|x^k - \hat{x}^{t_k}\| + \|x^{t_k} - \hat{x}^{t_k}\| \\
&\leq \sum_{\ell=-\tau}^{M-1} \|x^{k+\ell+1} - x^{k+\ell}\| + \sum_{\ell=-\tau}^{M-1} \|x^{k+\ell+1} - x^{k+\ell}\| \\
&= 2\sum_{\ell=-\tau}^{M-1} \|x^{k+\ell+1} - x^{k+\ell}\|.
\end{aligned}
\tag{48}
$$

In a similar fashion, we deduce

$$
\begin{aligned}
\|T_{i,\lambda_{t_k}}(x^k) - x^k\| &\leq \|T_{i,\lambda_{t_k}}(x^k) - x^{t_k+1}\| + \|x^{t_k+1} - x^k\| \\
&\leq 2\sum_{\ell=-\tau}^{M-1} \|x^{k+\ell+1} - x^{k+\ell}\| + \|x^{t_k+1} - x^k\| \\
&\leq 3\sum_{\ell=-\tau}^{M-1} \|x^{k+\ell+1} - x^{k+\ell}\|.
\end{aligned}
\tag{49}
$$

Letting $k \longrightarrow \infty$, the right-hand side goes to zero since it is the sum of a finite number of terms that, by hypothesis, converge to zero. This verifies $\|T_{i,\lambda_k}(x^k) - x^k\| \longrightarrow 0$, from which the result follows. ∎

Now we can state and prove the main result of this paper. As noted previously, this result is about a generalization of the procedure that successively applies the λ_k-relaxation of the operator T_{i_k}, for all $k \in \mathbb{N}$, which occurs when $\tau = 0$.

Theorem 1 *Let $\{x^k\}_{k\in\mathbb{N}}$ be a sequence generated by the ASI algorithm. If the delay vectors are uniformly bounded in the sup norm by some $\tau \geq 0$ and there is $\varepsilon > 0$ such that*

$$
0 < \varepsilon \leq \lambda_k \leq \frac{1}{2\tau + 1 + \varepsilon}, \quad \text{for all } k \in \mathbb{N}, \tag{50}
$$

then the sequence $\{x^k\}_{k\in\mathbb{N}}$ converges weakly to a common fixed point x^* of the family $\{T_i\}_{i=1}^m$, i.e.,

$$
x^k \rightharpoonup x^* \in C = \bigcap_{i=1}^m \mathrm{Fix}(T_i). \tag{51}
$$

Proof The given hypotheses make the assumptions of Lemma 3 hold, taking $\mu = 1$. With these, Lemma 4 then asserts that $\|x^{k+1} - x^k\| \longrightarrow 0$ and $\{\|x^k - z\|\}_{k\in\mathbb{N}}$ converges for any $z \in C$. The fact $\|x^{k+1} - x^k\| \longrightarrow 0$ enables Lemma 6 to be applied to deduce $\|T_i(x^k) - x^k\| \longrightarrow 0$, for each $i \in [m]$. This shows the assumptions of Proposition 1 hold and completes the proof. ■

5 Application to Linear Systems

In this section, we present an application for fast solution of linear systems of equations. Chiefly, we prove that the operators used in the method of Diagonally Relaxed Orthogonal Projections (DROP) [14] are nonexpansive and, thus, can be incorporated into the ASI algorithm. Let $A \in \mathbb{R}^{M\times N}$ and $b \in \mathbb{R}^M$, and consider the problem

$$\text{Find } x \in \mathbb{R}^N \text{ such that } Ax = b. \tag{52}$$

We assume that each row and column of the matrix A is nonzero, that A is large, and that the system is overdetermined. We provide background material pertaining to projection methods and then show that the ASI algorithm can be used to form an asynchronous version of DROP, which we call ASI–DROP.

Each equation in the linear system can be associated with a closed and convex subset of $\mathbb{R}^N$, namely, the hyperplane

$$H_i := \{x \in \mathbb{R}^N \mid \langle a^i, x\rangle = b_i\}, \quad \text{for all } i \in [M], \tag{53}$$

where a^i is the i-th row of A. The projection P_i onto H_i is given by

$$P_i(x) = x + \frac{b_i - \langle a^i, x\rangle}{\|a^i\|^2} a^i. \tag{54}$$

In order to discuss blocks of rows from the matrix A, let $\{B_t\}_{t=1}^r$ be a collection of sets of indices such that $B_t \subseteq [M]$, for all t, and

$$[M] = \bigcup_{t=1}^r B_t. \tag{55}$$

For each t, let A_t be the submatrix of A corresponding to the row indices in B_t and b_t be the subvector corresponding to the entries in B_t.[1] Note that there may be overlapping B_t's, i.e., it is possible that there exists $t \neq s$ such that $B_t \cap B_s \neq \emptyset$.

[1] In the next subsection, we define new matrices and include with these a subscript t. However, all of those matrices are constructed directly from A_t and are *not* themselves submatrices of any previously defined matrices.

5.1 Diagonally Relaxed Orthogonal Projections

The DROP algorithm is a modification of Cimmino's simultaneous projections method [19], which uses the iterative process

$$x^{k+1} := x^k - \frac{1}{M}\sum_{i=1}^{M} \frac{\langle a^i, x\rangle - b}{\|a^i\|^2} a^i. \tag{56}$$

When M is large, the term $1/M$ restricts the progress of the iteration. Letting s_j be the number of nonzero entries in the j-th column of A, the DROP algorithm replaces the $1/M$ term in (56) with $1/s_j$ to define component-wise updates via

$$x_j^{k+1} := x_j^k - \frac{1}{s_j}\sum_{i=1}^{M} \frac{\langle a^i, x\rangle - b}{\|a^i\|^2} a_j^i, \quad \text{for all } j \in [N]. \tag{57}$$

If A is sparse, then $s_j \ll M$ and this approach effectively makes the update x_j^{k+1} depend upon x_j^k and the average over the summands for which a_j^i is nonzero. For each $t \in [r]$, define the matrices

$$W_t := \operatorname{diag}\left(\|a^i\|^{-2}\right), \quad D_t := \operatorname{diag}\left(1/s_j\right), \quad \overline{A}_t := A_t D_t^{1/2}, \tag{58}$$

where the i-th diagonal entry of W_t is the inverse of the square of the norm of the i-th row in A_t and the j-th diagonal entry in D_t gives the number of nonzero entries in the j-th column of A_t. For each $t \in [r]$ define the operator $U_t : \mathbb{R}^N \to \mathbb{R}^N$ by

$$U_t(y) := y - \overline{A}_t^T W_t \left(\overline{A}_t y - b_t\right), \tag{59}$$

where b_t is the subvector of b corresponding to the row indices in B_t and here T stands for matrix transposition. Given an almost cyclic control $\{t_k\}_{k\in\mathbb{N}}$ on $[r]$, a special case of the DROP algorithm is given by the process

$$y^{k+1} := U_{t_k}\left(y^k\right). \tag{60}$$

The following proposition shows this formulation of DROP can be used with the ASI algorithm.

Proposition 2 *Each operator U_t defined by (59) is nonexpansive with respect to the Euclidean norm.*

Proof Let $t \in [r]$ and let (μ, v) be an eigenpair of $\overline{A}_t^T W_t \overline{A}_t$ so that $\mu v = \overline{A}_t^T W_t \overline{A}_t v$. Multiplying by the transpose of v and dividing by $\langle v, v\rangle$ yields

$$\mu = \frac{\langle v, \overline{A}_t^T W_t \overline{A}_t v\rangle}{\langle v, v\rangle} = \frac{\|\overline{A}_t v\|_{W_t}^2}{\|v\|_2^2} \geq 0, \tag{61}$$

where $\|v\|_2^2 = \langle v, v\rangle$ with $\langle \cdot, \cdot\rangle$ the inner product in $\mathbb{R}^N$ and $\|\cdot\|_{W_t}$ is the W_t-norm defined by $\|y\|_{W_t}^2 = \langle y, W_t y\rangle$. Note that W_t is symmetric and positive definite; thus, this is well-defined. Since the matrix $\overline{A}_t^T W_t \overline{A}_t$ is symmetric, its eigenvalues are real. Additionally, $\overline{A}_t^T W_t \overline{A}_t$ and $D_t A_t^T W_t A_t$ have the same eigenvalues. Indeed, using det for determinants and Id for the identity matrix, we use properties of determinants to see that

$$\begin{aligned}\det\left(\mu\mathrm{Id} - D_t A_t^T W_t A_t\right) &= \det(D_t^{-1/2})\det\left(\mu\mathrm{Id} - D_t A_t^T W_t A_t\right)\det(D_t^{1/2})\\ &= \det\left(\mu\mathrm{Id} - D_t^{1/2} A_t^T W_t A_t D_t^{1/2}\right)\\ &= \det\left(\mu\mathrm{Id} - \overline{A}_t^T W_t \overline{A}_t\right).\end{aligned} \tag{62}$$

By [14, Lemma 2.2], we know that $\rho(D_t A_t^T W_t A_t) \leq 1$. Consequently, $\mu \in [0,1]$. Note that

$$(\mathrm{Id} - \overline{A}_t^T W_t \overline{A}_t)v = (1-\mu)v, \tag{63}$$

and $(1-\mu) \in [0,1]$, which implies $\rho(\mathrm{Id} - \overline{A}_t^T W_t \overline{A}_t) \leq 1$. Moreover, because $(\mathrm{Id} - \overline{A}_t^T W_t \overline{A}_t)$ is symmetric,

$$\|\mathrm{Id} - \overline{A}_t^T W_t \overline{A}_t\|_2 = \rho(\mathrm{Id} - \overline{A}_t^T W_t \overline{A}_t) \leq 1. \tag{64}$$

Whence, for any $y^1, y^2 \in \mathbb{R}^N$

$$\begin{aligned}\|U_t(y^1) - U_t(y^2)\|_2 &= \left\|\left(\mathrm{Id} - \overline{A}_t^T W_t \overline{A}_t\right)\left(y^1 - y^2\right)\right\|_2\\ &\leq \|\mathrm{Id} - \overline{A}_t^T W_t \overline{A}_t\|_2 \|y^1 - y^2\|_2\\ &\leq \|y^1 - y^2\|_2,\end{aligned} \tag{65}$$

and we conclude that U_t is nonexpansive. ■

The ASI algorithm with these DROP operators, henceforth called the ASI–DROP algorithm, takes the following form. Its presentation is followed by a theorem guaranteeing its convergence.

Algorithm 2: ASI–DROP

Initialization: Let $A \in \mathbb{R}^{M\times N}$ and $b \in \mathbb{R}^M$ be given. Choose any $x^1 \in \mathbb{R}^N$, a sequence $\{\lambda_k\}_{k\in\mathbb{N}}$ such that $\lambda_k \in (0,1)$ for all $k \in \mathbb{N}$, an almost cyclic control $\{t_k\}_{k\in\mathbb{N}}$ on $[r]$, and a family of blocks of indices $\{B_t\}_{t=1}^r$ satisfying (55).

Iteration: For each $k \in \mathbb{N}$ set

$$x^{k+1} := \begin{cases} x^k, & \text{if } k \leq \sup_{k\in\mathbb{N}} \|d^k\|_\infty,\\ x^k - \lambda_k D_{t_k} A_{t_k}^T W_{t_k}\left(A_{t_k}\hat{x}^k - b_{t_k}\right), & \text{otherwise.}\end{cases} \tag{66}$$

Theorem 2 *Let $\{x^k\}_{k\in\mathbb{N}}$ be a sequence generated by the ASI–DROP algorithm. If the linear system (52) is consistent, and if the delay vectors are uniformly bounded in the sup norm by some $\tau \geq 0$, and if there is $\varepsilon > 0$ such that*

$$0 < \varepsilon \leq \lambda_k \leq \frac{1}{2\tau + 1 + \varepsilon}, \quad \textit{for all } k \in \mathbb{N}, \tag{67}$$

then the sequence $\{x^k\}_{k\in\mathbb{N}}$ converges to a solution of the linear system (52).

Proof For each t, set $S_t := \mathrm{Id} - U_t$. Proposition 2 and Theorem 1 imply that the sequence $\{y^k\}_{k\in\mathbb{N}}$ generated by the ASI algorithm converge to a fixed point of U_t. With the above notations and with U_t as in (59), define for each t

$$S_t := \mathrm{Id} - U_t. \tag{68}$$

Let $y^* = \lim_{k\to\infty} y^k$. Then for each $t \in [r]$

$$b_t = \overline{A}_t y^* = A_t D_t^{1/2} y^* = A_t x^*, \tag{69}$$

taking $x^* := D^{1/2}y^*$. This implies that $b = AD^{1/2}y^* = Ax^*$. For each $k \in \mathbb{N}$ set $x^k = D_{t_k}^{1/2} y^k$. Then $x^k \longrightarrow x^*$ where x^* is a solution to the linear system (52). Moreover, the active step describing the iterate updates is

$$\begin{aligned} x^{k+1} &= D_{t_k}^{1/2} y^{k+1} \\ &= D_{t_k}^{1/2}\left(y^k - \lambda_k S_{t_k}\left(\hat{x}^k\right)\right) \\ &= D_{t_k}^{1/2}\left(y^k - \lambda_k (I - U_{t_k})\left(\hat{y}^k\right)\right) \\ &= D_{t_k}^{1/2}\left(y^k - \lambda_k \overline{A}_{t_k}^T W_{t_k}\left(\overline{A}_{t_k}\hat{y}^k - b_{t_k}\right)\right) \\ &= x^k - \lambda_k D_{t_k} A_{t_k}^T W_{t_k}\left(A_{t_k}\hat{x}^k - b_{t_k}\right). \end{aligned} \tag{70}$$

This completes the proof. ■

5.2 Kaczmarz's and Other Methods

Kaczmarz Method. A popular method for approximating solutions to linear systems is that of Kaczmarz [29]. In the context of computed tomography, this method is known as the Algebraic Reconstruction Technique (ART) since it was rediscovered there by Gordon, Bender, and Herman in [24]. Kaczmarz's method generates updates by successively projecting each iterate x^k onto an individual hyperplane H_i. For an almost cyclic control $\{i_k\}_{k\in\mathbb{N}}$ on $[M]$ and a sequence of scalars $\{\lambda_k\}_{k\in\mathbb{N}}$, updates in the relaxed version of Kaczmarz's method are given by the iteration

$$x^{k+1} = (1-\lambda_k)x^k + \lambda_k P_{i_k}\left(x^k\right) = x^k + \lambda_k \left(\frac{b_{i_k} - \langle a^{i_k}, x^k\rangle}{\|a^{i_k}\|^2}\right) a^{i_k}, \tag{71}$$

where P_{i_k} is as in (54). Since the projection operators $\{P_i\}_{i=1}^M$ are nonexpansive, we use them in the ASI framework to construct an asynchronous generalization. For each $i \in [M]$ set, analogously to (10),

$$S_i(x) := (\mathrm{Id} - P_i)(x) = \frac{\langle a^i, x\rangle - b_i}{\|a^i\|^2} a^i. \tag{72}$$

Then the ASI–ART method is presented formally in Algorithm 3.

Algorithm 3: ASI-ART

Initialization: Let $A \in \mathbb{R}^{M\times N}$ and $b \in \mathbb{R}^M$ be given. Choose any $x^1 \in \mathbb{R}^N$, a sequence $\{\lambda_k\}_{k\in\mathbb{N}}$ such that $\lambda_k \in (0,1)$ for all $k \in \mathbb{N}$, and an almost cyclic control $\{i_k\}_{k\in\mathbb{N}}$ on $[M]$.

Iteration: For each $k \in \mathbb{N}$ set

$$x^{k+1} := \begin{cases} x^k, & \text{if } k \leq \sup_{k\in\mathbb{N}} \|d^k\|_\infty, \\ x^k - \lambda_k \dfrac{\langle a^{i_k}, \hat{x}^k\rangle - b_{i_k}}{\|a^{i_k}\|^2} a^{i_k}, & \text{otherwise.} \end{cases} \tag{73}$$

Other Methods. The ASI framework can be applied in conjunction with other projection methods. These include the valiant projection method (VPM) of Censor and Mansour [16], which is known as the automatic relaxation method (ARM) of Censor [9] when applied to interval linear inequalities. We conjecture the intrepid method of Bauschke, Iorio, and Koch [6], which is known as the ART3 method of Herman [27] when applied to linear systems, may also be used within the ASI framework.

6 ASI Algorithm Implementation

A sample pseudocode of the ASI algorithm is presented formally in Algorithm 4 and illustrated in Figure 2. Our model is for a master/slave type architecture. Referring to the notation of Section 3, let $x \in \mathcal{H}$ be arbitrary, fix $\lambda \in (0,1)$, and let $\{i_k\}_{k\in\mathbb{N}}$ be an almost cyclic control on $[m]$. First the initial iterate x is sent to each of the w processing nodes and we let the ℓ-th node compute $N_\ell := S_{i_\ell}(x)$. The iteration counter k is then set to $w+1$ and the time step counter θ to 1. Note the time step θ is distinct from the iteration step k since multiple nodes may complete their updates at the same time step, thereby enabling the iteration step to exceed the time step (i.e., $k \geq \theta$). The iterative process proceeds by fetching the collection of indices F_θ of nodes that produce outputs at time θ. Then, for each $\ell \in F_\theta$, we update x with $x - \lambda N_\ell$, where N_ℓ is the output of the ℓ-th node. Then x is sent to the ℓ-th node to compute $N_\ell = S_{i_k}(x)$. After the loop occurs over all elements of F_θ, we increment θ by 1 and repeat the iteration step if the stopping criteria are not met.

In the schematic Figure 2, each processing node is represented by a circle with N_ℓ inside. At iteration step k, the most recent output from the collection of nodes is fetched, which is precisely $N_\ell = S_{i_k}(\hat{x}^k)$ when the most recent output is from the ℓ-th node. This is then merged with x^k as in the ASI algorithm to form the new iterate x^{k+1}, overwriting x^k. The output x^{k+1} is then fed to the ℓ-th node to compute $N_\ell = S_{i_{k+1}}(x^{k+1})$. This effectively sets $k \leftarrow k+1$. Then the process repeats, fetching the most recent node outputs. In this master/slave framework,

Algorithm 4: A Pseudocode Implementation of the ASI Algorithm

Initialization:
  Let $x \in \mathcal{H}$, $\lambda \in (0,1)$, and $\{i_k\}_{k\in\mathbb{N}}$ an almost cyclic control on $[m]$.
  for $\ell \in [w]$
    Send x and i_ℓ to the ℓ-th node to compute $N_\ell = S_{i_\ell}(x)$
  endfor
  $k \leftarrow w+1$
  $\theta \leftarrow 1$

Master Node Iteration:
  while stopping criteria not met
    Fetch set of node indices F_θ for outputs received at time θ
    for $\ell \in F_\theta$
      $x \leftarrow x - \lambda N_\ell$
      $k \leftarrow k+1$
      Send x and i_k to ℓ-th slave node to compute $N_\ell = S_{i_k}(x)$
    endfor
    $\theta \leftarrow \theta + 1$
  end while

Slave Node ℓ Iteration:
  Read x and i_k as input
  Compute $N_\ell = S_{i_k}(x)$
  Output $N_\ell = S_{i_k}(x)$ to master node

each of the w slave nodes applies operators from the family $\{S_i\}_{i=1}^m$ and the master node continually computes the updates by merging x^k with $S_{i_k}(\hat{x}^k)$.

Remark 3 Implementation of Algorithm 2 can be done using the pseudocode in Algorithm 4 by taking $\mathcal{H} = \mathbb{R}^N$, $i_k = t_k$ for all $k \in \mathbb{N}$, $m = r$, and $\{U_t\}_{t=1}^r = \{T_i\}_{i=1}^m$, the family of DROP operators associated with the family of blocks of indices $\{B_t\}_{t=1}^r$ in Algorithm 2. Similarly, Algorithm 3 can be implemented using the pseudocode in Algorithm 4 by taking $\mathcal{H} = \mathbb{R}^N$ and $m = M$.

Remark 4 The indexing of the slave nodes can be set up as follows. First locally store subcollections of the family of operators $\{S_i\}_{i=1}^m$ to each slave node. Then let each node progress cyclically through the operators it has stored locally. Note that, although each slave node may proceed in a cyclic fashion, the order in which outputs arrive to the master node may not be cyclic. This can result from the variation in computation times on each node and, thus, arrival times to the master node. Despite this, the arrival of outputs to the master node will still occur in an almost cyclic fashion, as needed.

7 A Computational Example

The ASI algorithm is written in a sequential manner; however, we note that being able to use out-of-date iterations enables all the processing nodes to work *simultaneously*. The processing nodes are also able to work independently of each other.

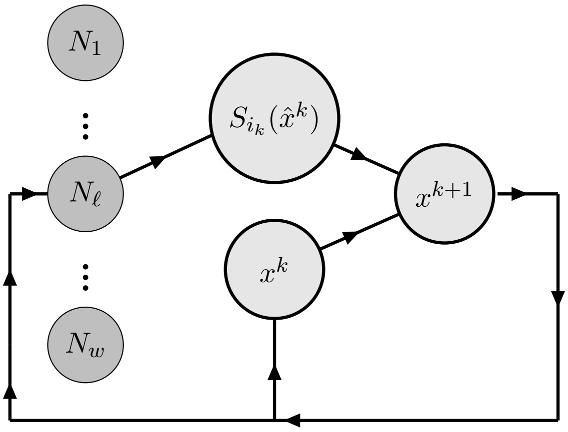


Fig. 2: Schematic architecture for the ASI algorithm. At the current iteration k the latest output N_ℓ from the ℓ-th node is merged with x^k via a linear combination to form the update x^{k+1}, overwriting x^k. Here $w \leq m$ is the number of processing nodes.

Below we apply our results in a computational example with a model computed tomography (CT) image reconstruction problem in Matlab. We use Algorithm 4 to implement Algorithm 2 and take $r = m = 40$ and $t_k = i_k$, for all $k \in \mathbb{N}$.

7.1 Experiment Setup

In our computational example, we provide results using the method of Diagonally Relaxed Orthogonal Projections (DROP) [14]. We implement DROP using both the ASI algorithm and the form of asynchrony of [21], i.e., without the inertial terms. We refer to these as ASI–DROP and EKN–DROP, respectively (EKN is for the authors' names of [21].) Note that convergence for EKN–DROP is not proven since we have not validated that the DROP operators are paracontractions. However, as expected, this method converged in all our experiments.

We consider an image reconstruction model of CT image reconstruction. Our specific aim is to show how an inherently sequential algorithm can be executed by multiple nodes working in parallel and asynchronously. This example illustrates the speedup of DROP for solving linear systems and the speedup that occurs when using the ASI–DROP algorithm. The task at hand was to solve (52) where A was a given 176,672×16,384 matrix and b was a vector with 176,672 entries. The computational work was done in Matlab and the quantities A and b were generated using the Shepp-Logan phantom [38] in Figure 3 and the AIR Tools Matlab package [26]. In our implementations, the initial iterate x^1 was set to be the zero vector and we generated $\{i_k\}_{k\in\mathbb{N}}$ following Remark 4, i.e., the family of operators $\{S_i\}_{i=1}^m$ was loaded into memory and each slave node accessed a subcollection of this family and applied the operators from that subcollection cyclically. Execution was stopped when sufficient proximity was reached. In particular, we stopped the

Fig. 3: A 128x128 digitization of the Shepp-Logan phantom [38].

Method	Measurement	number of slave nodes (w)				
		$w = 1$	$w = 2$	$w = 4$	$w = 8$	$w = 10$
ASI–DROP	time (sec)	751.5	418.0	226.9	140.3	163.8
	# epochs	353.9	357.4	352.5	352.4	445.0
	speedup	NA	1.80	3.31	5.35	4.59
EKN–DROP	time (sec)	767.1	540.8	387.9	361.1	395.6
	# epochs	353.9	427.5	561.4	840.7	989.0
	speedup	NA	1.42	1.98	2.12	1.94

Table 1: Reconstruction results with iterations stopped when $\|x^k - x\| < \varepsilon = 10^{-2}$. Reported values are averaged from 30 trials repeated on the same data set.

iterations when $\|x^k - x\| < \varepsilon = 10^{-2}$, where x is the true image vector, i.e., the "phantom" from which A and b were reconstructed for the purpose of the reconstruction experiment. We define an epoch to be the number of operators m, which in the case of our experiment was $m = 40$. The computation cluster used had 49.5 GB of RAM and 12 Intel Xeon X5650 processors with frequency 2.67 GHz. For more in-depth material on CT image reconstruction, we refer the reader, e.g., to Herman's book [28]. For the asynchronous calls to each processing node, we used the `parfeval` command in Matlab's Parallel Computing toolbox.

7.2 Numerical Results

Reconstruction results are provided in Table 1. Note that the number of iterations required for the EKN–DROP approach increases as the number of nodes w increases. However, in some cases, *fewer* iterations are required using the ASI–DROP algorithm. Furthermore, there is speedup as the number of nodes increases, given by 1.80, 3.31, 5.35, and 4.59 for 2, 4, 8, and 10 nodes, respectively. With $w = 12$ nodes, the ASI algorithm does not converge with the step size $\lambda = 0.20$, but does converge if the step-size λ is sufficiently reduced (e.g., by taking $\lambda = 0.15$). In comparison, for EKN–DROP the speedup was 1.42, 1.98, 2.12, 1.94 for 2, 4, 8, and 10 nodes, respectively. The fastest reconstruction time for ASI–DROP was

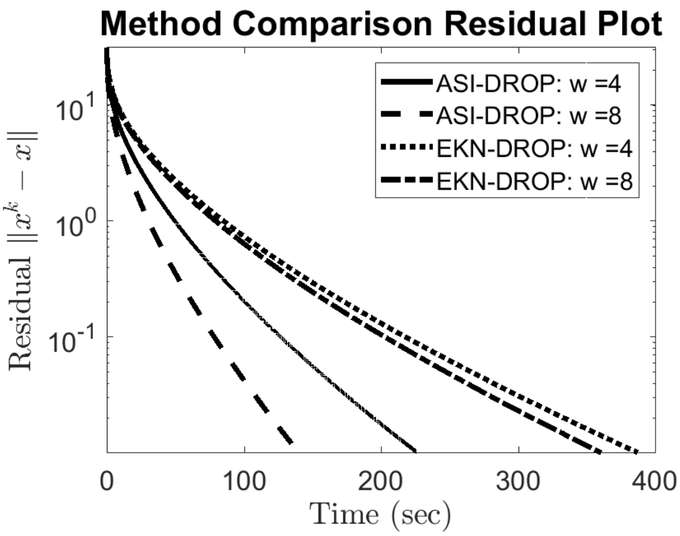


Fig. 4: Juxtaposition of averages of residual plots over 30 trials for the ASI–DROP and EKN–DROP algorithms.

140.3 seconds, which is more than twice as fast as that for EKN–DROP (361.1 seconds) and over five times faster than the sequential implementation of DROP.

For both forms of asynchrony, we see diminishing returns as w increases. It might even be more advantageous to choose w smaller rather than larger. This is seen by comparing the ASI–DROP algorithm performance when $w = 8$ and $w = 10$ in Table 1. However, for sufficiently small $w > 1$ we still see notable performance improvements over the existing synchronous sequential case $w = 1$. Plots of the averages of residuals for the ASI–DROP and EKN–DROP schemes are provided in Figures 4, 5, and 6. Figure 5(a) reveals that, when the step sizes are small enough, roughly the same number of iterations is needed to obtain convergence of the ASI–DROP algorithm as w increases, up to $w = 8$. Figure 5(b) shows that the amount of computation time decreases as w increases, up to $w = 8$, for the ASI–DROP algorithm. For the EKN–DROP algorithm, Figure 5(c) shows more iterations are needed to obtain convergence; however, even with the larger number of iterations, Figure 5(d) shows speedup is still obtained for $w > 1$. Figure 6 demonstrates the behavior of the ASI–DROP algorithm when the step sizes increase to nearly the maximal size that still yields convergence. There we see a wide plot, which demonstrates the variance among arrival times of the iterates generated by the ASI–DROP algorithm. Note also that the iterates do *not* necessarily approach the solution monotonically. In Figure 7, we see sample reconstructions that reveal each image high equality. In Figure 4, we see the primary comparison plot between the ASI–DROP and EKN–DROP methods of asynchrony, which shows increasing speedup as w increases for the ASI–DROP algorithm. In summary, compared to the sequential application of block operators, both versions of asynchrony display speedup while the ASI–DROP approach is faster when it converges.

7.3 Discussion

From the computational example, we learn that much larger step sizes may possibly be used in practice, with promise, than the bound given in Theorem 1. If the updates are uniformly random and independent of the distribution of delays, then the results of ARock [25, Table 2] can be used to deduce larger step sizes yield convergence, i.e., the upper bound for step sizes is $1/(1+2\tau/\sqrt{m})$, where m is the number of operators and τ is an upper bound on the delays. However, once the number of nodes w increases too much, then the step size must be reduced to maintain convergence. There appears to be an optimal pairing (λ, w) for obtaining the fastest reconstructions. The EKN asynchrony may be advantageous over a sequential algorithm, but requires an increasing number of iterations as w increases, thereby limiting the speedup. Due to the introduction of inertial terms (c.f. (14) and Figure 2), we see roughly the same number of iterations may be needed for the ASI algorithm as w increases, and in some cases *fewer* iterations are needed.

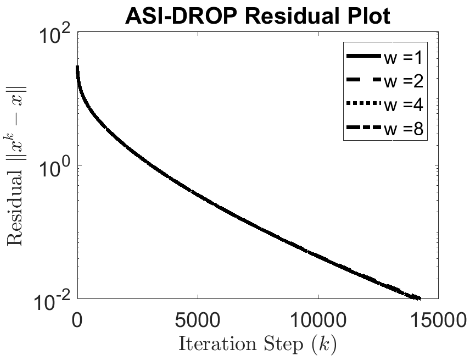
Although there is a speedup for the ASI–DROP algorithm, it is sublinear in our experiments. Initially, this may seem contradictory since fewer iterations are required and more processing nodes are used. However, when several nodes produce outputs during the same time step θ, a queue is formed for the updates to x^k (c.f. the `for` loop during the iteration of Algorithm 4). This leads to some delays and node idle time. If the operators are more computationally expensive, then the chances of simultaneous node outputs at the same time step goes down. Consequently, it may be advantageous to use few costly operators rather than many computationally cheap operators. Alternatively, one may choose a different pseudocode implementation of the ASI algorithm that does not utilize the master/slave architecture, but instead uses some form of peer to peer network.

8 Conclusion

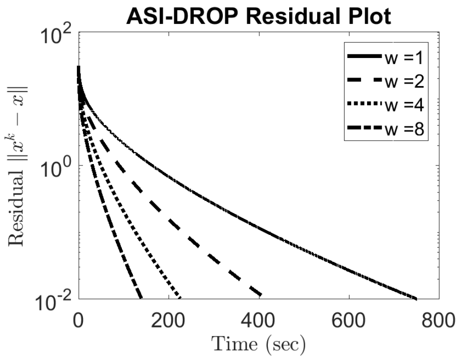
In this work, we present a KM-type iteration for common fixed point problems that allows for partial asynchronity, i.e., delays uniformly bounded in the sup norm by some $\tau \geq 0$. Convergence of this ASI algorithm is established. This provides robustness to dropped network transmissions, removes both the need to synchronize node outputs and to coordinate load-balancing, and reveals a method for attaining further speedup with block-iterative methods when solving large scale problems. Moreover, when there is a delay, inertial terms are introduced into the iteration to accelerate convergence. In some cases, this reduces the number of iterations needed to converge. Future work may test the ASI algorithm on massive scale problems, study application of the ASI algorithm to computing architectures other than the master/slave architecture, and investigate extensions to inconsistent convex feasibility problems.

Acknowledgments

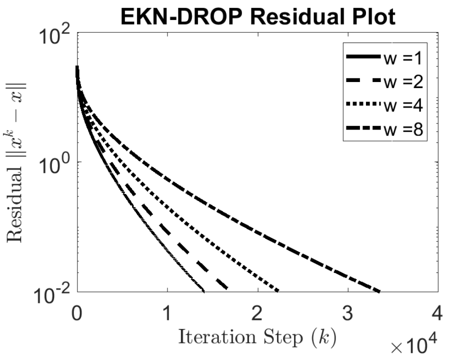
We thank Prof. Reinhard Schulte from Loma Linda Univeristy for valuable discussions and insights regarding the advantage of an asynchronous approach to



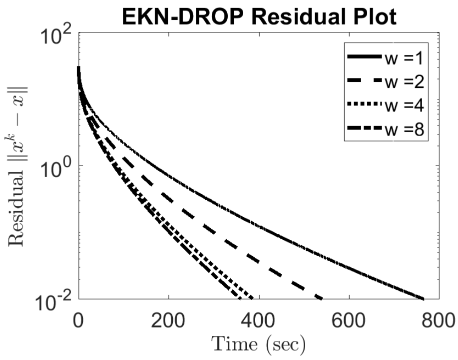
(a) ASI–DROP



(b) ASI–DROP



(c) EKN–DROP



(d) EKN–DROP

Fig. 5: Averages of residual plots over 30 trials for the ASI–DROP and EKN–DROP algorithms.

large scale problems in proton CT (pCT) image reconstruction and Intensity-Modulated Proton Therapy (IMPT). We appreciate constructive comments received from Prof. Ernesto Gomez from California State University San Bernardino and Prof. Keith Schubert from Baylor University. The first author's work is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1650604. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The second author's work is supported by research grant No. 2013003 of the United States-Israel Binational Science Foundation (BSF).

References

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

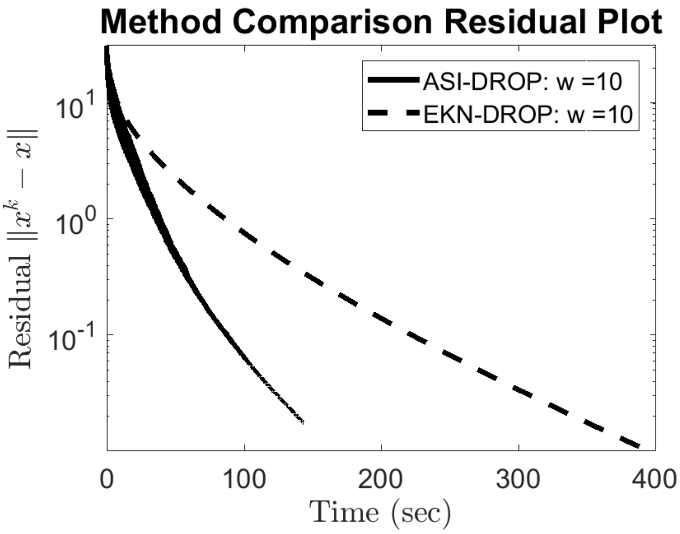


Fig. 6: Averages of residual plots over 30 trials for the ASI–DROP and EKN–DROP algorithms with $w = 10$ slave nodes.

2. Alvarez, F., Attouch, H.: An inertial proximal method for maximal monotone operators via discretization of a nonlinear oscillator with damping. Set-Valued Analysis **9**, 3–11 (2001)
3. Bauschke, H.H., Borwein, J.M.: On projection algorithms for solving convex feasibility problems. SIAM review **38**, 367–426 (1996)
4. Bauschke, H.H., Combettes, P.L.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces, 2nd edn. Springer (2017)
5. Bauschke, H.H., Combettes, P.L., Kruk, S.G.: Extrapolation algorithm for affine-convex feasibility problems. Numerical Algorithms **41**, 239–274 (2006)
6. Bauschke, H.H., Iorio, F., Koch, V.R.: The method of cyclic intrepid projections: Convergence analysis and numerical experiments. In: M. Wakayama, R.S. Anderssen, J. Cheng, Y. Fukumoto, R. McKibbin, K. Polthier, T. Takagi, K.C. Toh (eds.) The Impact of Applications on Mathematics, pp. 187–200. Springer Japan, Tokyo (2014)
7. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Prentice Hall, Englewood Cliffs, NJ, USA (1989)
8. Cegielski, A.: Iterative Methods for Fixed Point Problems in Hilbert Spaces, *Lecture Notes in Mathematics*, vol. 2057. Springer (2012)
9. Censor, Y.: An automatic relaxation method for solving interval linear inequalities. Journal of Mathematical Analysis and Applications **106**, 19–25 (1985)
10. Censor, Y.: Parallel application of block-iterative methods in medical imaging and radiation therapy. Mathematical Programming **42**, 307–325 (1988)
11. Censor, Y., Altschuler, M.D., Powlis, W.D.: On the use of Cimmino's simultaneous projections method for computing a solution of the inverse problem in radiation therapy treatment planning. Inverse Problems **4**, 607 (1988)
12. Censor, Y., Chen, W., Combettes, P.L., Davidi, R., Herman, G.T.: On the effectiveness of projection methods for convex feasibility problems with linear inequality constraints. Computational Optimization and Applications **51**, 1065–1088 (2012)
13. Censor, Y., Elfving, T., Herman, G.: Averaging strings of sequential iterations for convex feasibility problems. In: D. Butnariu, Y. Censor, S. Reich (eds.) Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications, pp. 101–113. Elsevier Science Publishers, Amsterdam, The Netherlands (2001)
14. Censor, Y., Elfving, T., Herman, G.T., Nikazad, T.: On diagonally relaxed orthogonal projection methods. SIAM Journal on Scientific Computing **30**, 473–504 (2008)

(a) ASI–DROP ($w = 1$)

(b) EKN–DROP ($w = 1$)

(c) ASI–DROP ($w = 8$)

(d) EKN–DROP ($w = 8$)

Fig. 7: Sample reconstructions.

15. Censor, Y., Gordon, D., Gordon, R.: BICAV: A block-iterative parallel algorithm for sparse systems with pixel-related weighting. IEEE Transactions on Medical Imaging **20**, 1050–1060 (2001)
16. Censor, Y., Mansour, R.: Convergence analysis of processes with valiant projection operators in hilbert space. Journal of Optimization Theory and Applications **176**, 35–56 (2018)
17. Censor, Y., Zenios, S.A.: Parallel Optimization: Theory, Algorithm, and Applications. Oxford University Press, New York, NY, USA (1997)
18. Chazan, D., Miranker, W.: Chaotic relaxation. Linear Algebra and its Applications **2**, 199–222 (1969)
19. Cimmino, G.: Cacolo approssimato per soluzioni dei sistemi di equazioni lineari. La Ricerca Scientifica XVI, *Series II, Anno IX* **1**, 326–333 (1938)
20. Combettes, P.L.: The convex feasibility problem in image recovery. In: Advances in imaging and electron physics, vol. 95, pp. 155–270. Academic Press, New York, NY, USA (1996)
21. Elsner, L., Koltracht, I., Neumann, M.: Convergence of sequential and asynchronous nonlinear paracontractions. Numerische Mathematik **62**, 305–319 (1992)
22. Frommer, A., Szyld, D.B.: On asynchronous iterations. Journal of Computational and Applied Mathematics **123**, 201–216 (2000)
23. Geer, D.: Chip makers turn to multicore processors. Computer **38**, 11–13 (2005)
24. Gordon, R., Bender, R., Herman, G.T.: Algebraic reconstruction techniques (art) for three-dimensional electron microscopy and x-ray photography. Journal of theoretical Biology **29**(3), 471–481 (1970)
25. Hannah, R., Yin, W.: On unbounded delays in asynchronous parallel fixed-point algorithms. Journal of Scientific Computing **76**, 299–326 (2018)
26. Hansen, P.C., Saxild-Hansen, M.: AIR Tools–A MATLAB package of algebraic iterative reconstruction methods. Journal of Computational and Applied Mathematics **236**, 2167–2178 (2012)

27. Herman, G.T.: A relaxation method for reconstructing objects from noisy x-rays. Mathematical Programming **8**, 1–19 (1975)
28. Herman, G.T.: Fundamentals of Computerized Tomography, 2nd edn. Springer-Verlag, London, UK (2009)
29. Kaczmarz, S.: Angenäherte Auflösung von Systemen linearer Gleichungen. Bulletin, International de l'Acadeémie Polonaise des Sciences et des Lettres **A35**, 355–357 (1937)
30. Kiwiel, K.C.: Block-iterative surrogate projection methods for convex feasibility problems. Linear Algebra and its Applications **215**, 225–259 (1995)
31. Krasnosel'skii, M.A.: Two remarks on the method of successive approximations. Uspekhi Matematicheskikh Nauk **10**, 123–127 (1955). (in Russian)
32. Lorenz, D.A., Pock, T.: An inertial forward-backward algorithm for monotone inclusions. Journal of Mathematical Imaging and Vision **51**, 311–325 (2015)
33. Maingé, P.E.: Convergence theorems for inertial KM-type algorithms. Journal of Computational and Applied Mathematics **219**, 223–236 (2008)
34. Mann, W.R.: Mean value methods in iteration. Proceedings of the American Mathematical Society **4**, 506–510 (1953)
35. Moudafi, A., Oliny, M.: Convergence of a splitting inertial proximal method for monotone operators. Journal of Computational and Applied Mathematics **155**, 447–454 (2003)
36. Peng, Z., Xu, Y., Yan, M., Yin, W.: ARock: An algorithmic framework for asynchronous parallel coordinate updates. SIAM Journal on Scientific Computing **38**, A2851–A2879 (2016)
37. Polyak, B.T.: Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics **4**, 1–17 (1964)
38. Shepp, L.A., Logan, B.F.: The Fourier reconstruction of a head section. IEEE Transactions on nuclear science **21**, 21–43 (1974)
39. Strikwerda, J.C.: A convergence theorem for chaotic asynchronous relaxation. Linear Algebra and its Applications **253**, 15–24 (1997)