Solving Linear Algebraic Equations with Limited Computational Power and Network Bandwidth[☆]

Shenyu Liu[a], Sonia Martinez[b]

[a]*School of Automation, Beijing Institute of Technology, Beijing 100089, China.*
[b]*Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA 92037, USA.*

Abstract

This work introduces a distributed algorithm for finding least square (LS) solutions of linear algebraic equations (LAEs). Unlike the methods studied in the literature, we assume that our distributed algorithm has limited computation power and network bandwidth, in the sense that each agent can only solve small-scale LAEs and the group of agents can only exchange messages of small size at a time. Our algorithm contains two layers of nested loops. A part of the solution is updated by a consensus algorithm in the inner loop, while an scheduling sequence in the outer loop decides which part of the solution to be updated. By appealing to the alternating projection theorem, we prove convergence of the proposed algorithm when the scheduling sequence is both spanning and periodic. The accuracy of our algorithm is verified through a numerical example.

Keywords: Distributed algorithm, linear algebraic equations, least square solutions.

1. Introduction

The solution to linear algebraic equations (LAEs) finds application in diverse scientific and engineering disciplines such as physics, chemistry, financial

[☆]This work was supported by the grants NSF of China 62203053, and ONR N00014-19-1-2471.

Email addresses: shenyuliu@bit.edu.cn (Shenyu Liu), soniamd@ucsd.edu (Sonia Martinez)

modeling, and network analysis. A particular area of interest concerns data-driven modeling and regression, which typically employ very large-dimensional, data-dependent LAEs for prediction. As the data can be spread over multiple nodes in a network, an efficient, privacy preserving, and dependable algorithm becomes crucial in the training of such models. In order to ensure this, the associated distributed algorithms must account for the limitations in computation and communication of the digital devices involved in the process. This work aims to contribute to the body of work that provides a solution to these problems under various communication and computation constraints.

Literature review: Solving LAEs in a distributed manner has many applications, such as distributed spectrum estimation [**?**] and linear and nonlinear (kernel-based) regression in predictive learning [1]. The distributed solution to LAEs has been considered in various forms in the literature starting with [2, 3, 4], see also the survey [5]. The main assumption of these early works is that each node in the network has access to a full set of rows of the coefficient matrix of LAEs and the corresponding entries in the independent vector. In this way, one can find discrete-time iterations [2, 3, 6] that produce an exact solution to the LAEs for underdetermined systems. However, these solutions struggle when there is an excessive number of constraints, which motivates the distributed optimization approach of [7], employing continuous flows to solve the problem approximately. Subsequently, [8] follows along these lines and further characterizes the step-size under which the discretization of one such type of continuous-time algorithms converges. More recently, leveraging continuous-time algorithms and dynamic consensus [9], the work [10] considers a special LAE problem in which each agent in the network has access to a full coefficient matrix and an independent vector of its own, which are then summed up to produce a global LAE. Convergence is established under general assumptions of time-varying, undirected graph connectivity.

However, the aforementioned distributed algorithms require the exchange of large-size messages, proportional to the solution's dimension. This presents a challenge due to limited bandwidth for communication among agents. Con-

sequently, these algorithms are not "scalable", in the sense that adding more agents does not facilitate solving LAEs with an excessive number of decision variables. Recent studies [11, 12] focus on scalable continuous-time and discrete-time algorithms. These works propose double-layered networks comprising higher-level aggregators coordinating with locally clustered agents. Each agent controls and broadcasts a state variable proportional to its known coefficient submatrix dimension. By achieving consensus and conservation through intra-cluster agent-agent and inter-cluster aggregator-aggregator communications, distributed algorithms are devised for agents to cooperatively achieve least square (LS) solutions to LAEs. We note that in these works, the number of required agents needs to be of the order of the total number of coefficients.

Statement of contributions: In this study, we present a novel scalable discrete-time distributed algorithm for determining the LS solutions of overdetermined LAEs. In addition to the usual assumption that each agent only knows some rows of coefficient matrix, our approach introduces communication "scheduling" to address the challenges of limited computation power and communication bandwidth. In contrast with previous work, even under the constraint of limited bandwidth, the number of agents in our algorithm does not need to be of the order of the total number of coefficients. Our algorithm consists of two integrated loops. An inner loop employs a consensus algorithm, which efficiently solves small-scale LAEs by the multi-agent network. The solution of the small-scale LAEs is then used to update a portion (a subset of the entries) of the guessed solution, determined by a set-valued scheduling sequence designed for the outer loop algorithm. By leveraging the alternating projection theorem, we establish convergence of our algorithm under the conditions that the outer loop's scheduling sequence is both spanning and periodic.

The rest of the paper is organized as follows. In Section 2, we provide the motivation behind considering limited computation power and network bandwidth, and we formally define the problem. Section 3 introduces the distributed algorithms proposed in this study. The convergence analysis of the algorithms is presented in Section 4. In Section 5, we provide a numerical example to demon-

strate the algorithms' effectiveness. Finally, Section 6 concludes the paper by summarizing the findings and offering insights into potential avenues for future research.

Notation. Let $\mathbb{R}, \mathbb{N}$ be the set of all real numbers and positive integers, respectively. For any $l, m, n \in \mathbb{N}$, define the sets $\mathsf{L} := \{1, \dots, l\}, \mathsf{M} := \{1, \dots, m\}, \mathsf{N} := \{1, \dots, n\}$. For any countable set Ω, let $|\Omega|$ denote its cardinality. For any vector $x \in \mathbb{R}^n$ and matrix $A \in \mathbb{R}^{m \times n}$, let $\|x\|, \|A\|$ denote the 2-norm and induced norm, respectively. The range and kernel spaces of a matrix $A \in \mathbb{R}^{m \times n}$ is denoted by $\mathrm{im}(A), \ker(A)$, respectively. The orthogonal space of a vector subspace $\mathcal{X}$ is denoted by $\mathcal{X}^\perp$. An undirected graph $G = (V, E)$ with l vertices consists of a vertex set $V = \mathsf{L}$ and edge set $E \subset \mathsf{L} \times \mathsf{L}$ such that $(i, j) \in E$ if and only if $(j, i) \in E$. A path is a sequence of vertices connected by edges, and the graph G is *connected* if there is a path between any pair of vertices. For any $i \in V$, the set of neighbors of i is $\mathcal{N}_i := \{j \in V : (i, j) \in E\}$ and the degree is $d_i := |\mathcal{N}_i|$.

2. Motivation and problem formulation

Consider the following problem of solving the *linear algebraic equation* (LAE)

$$Ax = b, \tag{1}$$

where $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$ are the known data, $x \in \mathbb{R}^n$ is the variable to be determined. Throughout this work, we assume A to be full column rank so that the problem (1) has a unique *least squares* (LS) solution, which can be equivalently characterized by the following lemma.

Lemma 1. *The following statements regarding $x^* \in \mathbb{R}^n$ are equivalent:*

1. *x^* is the LS solution of the LAE (1).*
2. *$x^* = \arg\min_x \|Ax - b\|^2$.*
3. *$x^* = A^\dagger b$, where $(\cdot)^\dagger$ denotes pseudo-inverse.*
4. *$(Ax^* - b) \perp \mathrm{im}(A)$.*

The motivation to account for limited computational power and network bandwidth stems from the challenge posed by large-scale problems with high-dimensional LS solutions. Note that, the direct computation of the LS solution using the formula in the third statement of Lemma 1 becomes impractical due to the burden associated with the computation of the pseudo-inverse $A^\dagger$. Hence, this consideration is crucial in developing efficient algorithms. Distributed algorithms, such as those proposed in [3, 13], offer a potential solution to this issue by distributing the workload among multiple agents in a network. However, these algorithms still require the exchange of full solution guesses among the agents. Given that the LS solution vector x is of dimension n, an efficient information exchange requires a network bandwidth of at least size n. Our objective is to propose a scalable distributed algorithm, where the number of agents only scales with respect to m, so that it solves LAEs of arbitrary values of m and n without the demand to increase the computational power of each agent or expand the network bandwidth for information exchange. More precisely,

Problem 1 (Scalable Solution to LAE)**.** *For some m^*, $n^* \in \mathbb{N}$, let $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ with $m > m^*, n > n^*$ be given. Assume the following holds for a group of l agents that interact over a connected, undirected communication graph G:*

- *(Limited data) Each agents knows at most a subset of m^* rows of A and the corresponding elements in b;*
- *(Limited computation power) Each agent can only compute the inverse of $r \times r$ square matrices (if invertible) for $r \leq m^*$;*
- *(Limited communication bandwidth) The i-th agent can broadcast and receive messages of size up to n^* to and from its neighbors in $\mathcal{N}_i$ at each time.*
- *(Number of agents) It holds that $l = O(m)$.*

Design a distributed algorithm which finds the LS solution of (1) *by this group of agents.*

3. The distributed algorithm

Before introducing our algorithm, we define a few more notions that are needed. For a given matrix $A \in \mathbb{R}^{m\times n}$, a vector $b \in \mathbb{R}^m$ and any sets $S_r \subseteq \mathsf{M}, S_c \subseteq \mathsf{N}$, denote $A_{S_r,S_c} \in \mathbb{R}^{|S_r|\times|S_c|}$ to be the submatrix of A consisting of elements on the rows indexed by S_r and columns indexed by S_c, $b_{S_r} \in \mathbb{R}^{|S_r|}$ to be the subvector of b consisting of elements on the rows indexed by S_r. In particular, if $S_r = \mathsf{M}$ or $S_c = \mathsf{N}$ then the notation A_{S_r,S_c} is abbreviated by $A_{:,S_c}$ or $A_{S_r,:}$, respectively.

Let $\{\Sigma(i)\}_{i\in\mathsf{L}}$ be a partition of M; i.e., $\cup_{i\in\mathsf{L}}\Sigma(i) = \mathsf{M}$ and $\Sigma(i) \cap \Sigma(j) = \emptyset$ for each pair of agents $i \neq j$. Also, let $\Omega : \mathbb{N} \rightrightarrows \mathsf{N}$ be a set-valued map and define its complement $\Omega^c : \mathbb{N} \rightrightarrows \mathsf{N}$ by $\Omega^c(i) := \mathsf{N}\backslash\Omega(i)$. Naturally, the elements in $\Omega(t), \Omega^c(t)$ are enumerated by the order of elements in N. We therefore define a permutation matrix $P(t) = [p_{jk}(t)] \in \mathbb{R}^{n\times n}$ by

$$p_{jk}(t) = \begin{cases} 1, & \text{if } k \text{ is the } j\text{-th element in } \Omega(t), \\ 1, & \text{if } k \text{ is the } (j - |\Omega(t)|)\text{-th element in } \Omega^c(t), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Informally speaking, our algorithm consists of two nested loops or layers. In the outer layer, the scheduling sequence $\Omega(t)$ tells which part of the solution the agents need to update. In the inner layer, the agents run a consensus algorithm based on their known data $A_{\Sigma(i),:}, b_{\Sigma(i)}$, in order to reach consensus on that part of the solution.

The following assumption is needed for our algorithm to be implementable.

Assumption 1. *It holds that all* $\Sigma(i)$ *are non-empty and*

$$|\Sigma(i)| \leq m^*, \quad \forall i \in \mathsf{L}, \tag{3}$$

$$|\Omega(t)| \leq \frac{n^*}{2}, \quad \forall t \in \mathbb{N}. \tag{4}$$

The inequality (3) can be satisfied by setting l sufficiently large. The sequence $\Omega(t)$ can also be easily made to satisfy the inequality (4).

Our distributed algorithm can be stated as follows.

$$\begin{bmatrix} x^{(i)}_{\Omega(t)} \\ y^{(i)}_{\Omega(t)} \end{bmatrix} \leftarrow F^{-1}_{i,t} \begin{bmatrix} d_i x^{(i)}_{\Omega(t)} + \sum_{j\in\mathcal{N}_i} y^{(j)}_{\Omega(t)} + A^\top_{\Sigma(i),\Omega(t)} \left(b_{\Sigma(i)} - A_{\Sigma(i),\Omega^c(t)} x^{(i)}_{\Omega^c(t)} \right) - \Sigma_{j\in\mathcal{N}_i} x^{(j)}_{\Omega(t)} + d_i y^{(i)}_{\Omega(t)} \end{bmatrix}. \tag{5}$$

$$F^{-1}_{i,t} := -\frac{1}{4d_i^2} \begin{bmatrix} I_{|\Omega(t)|} \\ I_{|\Omega(t)|} \end{bmatrix} A^\top_{\Sigma(i),\Omega(t)} \left(I_{|\Sigma(i)|} + \frac{1}{2d_i} A_{\Sigma(i),\Omega(t)} A^\top_{\Sigma(i),\Omega(t)} \right)^{-1} A_{\Sigma(i),\Omega(t)} \begin{bmatrix} I_{|\Omega(t)|} & I_{|\Omega(t)|} \end{bmatrix} + \frac{1}{2d_i} \begin{bmatrix} I_{|\Omega(t)|} & -I_{|\Omega(t)|} \\ I_{|\Omega(t)|} & I_{|\Omega(t)|} \end{bmatrix} \tag{6}$$

1. Initially, let all the agents start with a common initial guess $x^{(i)}(0) = x(0) \in \mathbb{R}^n$ and arbitrary auxiliary state vectors $y^{(i)}(0) \in \mathbb{R}^n$ for all $i \in \mathsf{L}$.
2. *Outer loop.* For the t-th iteration, the i-th agent, $i \in \mathsf{L}$, does the following:
   2.1) *Inner loop.* For the s-th iteration, update $x^{(i)}_{\Omega(t)}, y^{(i)}_{\Omega(t)}$ according to rule (5), where $F^{-1}_{i,t}$ is given by (6).
   2.2) Repeat Step 2.1) for $s \in \mathbb{N}$ until $x^{(i)}_{\Omega(t)}, y^{(i)}_{\Omega(t)}$ converge for all $i \in \mathsf{L}$.
3. Repeat Step 2) for $t \in \mathbb{N}$ until $x^{(i)}, y^{(i)}$ converge for all $i \in \mathsf{L}$.

We justify that under Assumption 1, our algorithm meets all the constraints in Problem 1. First of all, the i-th agent only knows $A_{\Sigma(i),:}, b_{\Sigma(i),:}$. This ensures local data privacy and, in the view of (3), the constraint of limited data is met. Meanwhile, in the state update rule (5), the computational complexity mainly arises from the inversion in the computation of the matrix $F^{-1}_{i,t}$, which has a dimension of $|\Sigma(i)| \times |\Sigma(i)|$. Again, our algorithm satisfies the constraint of limited computation power. Moreover, the state update rule (5) only requires a subset of the state variables, namely $x^{(j)}_{\Omega(t)}$ and $y^{(j)}_{\Omega(t)} \in \mathbb{R}^{|\Omega(t)|}$, to be broadcast. By referring to (4), we can ascertain that our algorithm also satisfies the constraint of limited network bandwidth. Lastly, l only needs to satisfy (3); we do not need $l = O(mn)$. As a result, the proposed distributed algorithm is scalable for solving LAEs of arbitrary dimension.

The sentences "until ... converge" in Step 2.1) and Step 3) mean to the accuracy of the algorithm as in Section 4. In practice, "convergence" is ob-

tained by taking sufficiently many iterations in both inner and outer loops, or the iterations are stopped when the updated values are observed to have negligible changes. Nevertheless, the latter method requires a varying number of iterations, which becomes challenging in the distributed setup as the agents may terminate the loop at different time.

We also remark that our approach can be reminiscent of the Gauss-Seidel method [2, Chapter 2.4], where the partial update of the solution can be interpreted as a type of scheduling. However, when the Gauss-Seidel method is directly implemented in a distributed manner [14], it requires the exchange of messages whose dimension equals to the dimension of the solution. This violates our constraint of limited communication bandwidth.

4. Convergence analysis

Denote the solution computed by the i-th agent after the t-th iteration of the outer loop by $x^{(i)}(t)$. These values are the output of the inner loop, which is essentially the consensus algorithm from [13], aiming to find a LS solution $z \in \mathbb{R}^{|\Omega(t)|}$ of the LAE

$$\begin{bmatrix} A_{\Sigma(1),\Omega(t)} \\ A_{\Sigma(2),\Omega(t)} \\ \vdots \\ A_{\Sigma(l),\Omega(t)} \end{bmatrix} z = \begin{bmatrix} b_{\Sigma(1)} - A_{\Sigma(1),\Omega^c(t)} x^{(1)}_{\Omega^c(t)}(t-1) \\ b_{\Sigma(2)} - A_{\Sigma(2),\Omega^c(t)} x^{(2)}_{\Omega^c(t)}(t-1) \\ \vdots \\ b_{\Sigma(l)} - A_{\Sigma(l),\Omega^c(t)} x^{(l)}_{\Omega^c(t)}(t-1) \end{bmatrix}. \tag{7}$$

In Section 4.1, we first show a convergence result under the assumption of *perfect* consensus from each inner loop. We then analyze the impact of inconsistencies among agents produced by an early termination of the inner loop in Section 4.2.

4.1. Convergence guarantees under perfect consensus

If perfect consensus is achieved after each inner loop, then

$$x^{(1)}(t) = x^{(2)}(t) = \cdots = x^{(l)}(t) =: x(t), \quad \forall t \in \mathbb{N}. \tag{8}$$

We first give an iterative property of the sequence of $x(t)$ in this case.

Lemma 2. *Define a sequence of matrices* $\Pi : \mathbb{N} \to \mathbb{R}^{n\times n}$ *by*

$$\Pi(t) := P^\top(t) \begin{bmatrix} 0_{|\Omega(t)|\times|\Omega(t)|} & -A^\dagger_{:,\Omega(t)}A_{:,\Omega^c(t)} \\ 0_{|\Omega^c(t)|\times|\Omega(t)|} & I_{|\Omega^c(t)|} \end{bmatrix} P(t), \tag{9}$$

where $P(t)$ *is defined by* (2)*. If* $x^* \in \mathbb{R}^n$ *is the unique LS solution of* (1) *and the agents reach perfect consensus for each inner loop in the sense of* (8)*, then*

$$x(t) - x^* = \Pi(t)(x(t-1) - x^*) \tag{10}$$

for all $t \in \mathbb{N}$*.*

Proof. Substitute (8) into (7) and re-arrange the rows, we obtain an equivalent expression

$$A_{:,\Omega(t)}z = b - A_{:,\Omega^c(t)}x_{\Omega^c(t)}(t-1). \tag{11}$$

By [13, Theorem 1], the consensus state is the LS solution of (7), which is used to update $x_{\Omega(t)}(t)$. According to the second statement in Lemma 1, $\|A_{:,\Omega(t)}x_{\Omega(t)}(t) + A_{:,\Omega^c(t)}x_{\Omega^c(t)}(t-1) - b\|$ is minimized. Now denote $\delta(t) := x(t) - x^*$. We have

$$\begin{aligned}
&\|A_{:,\Omega(t)}x_{\Omega(t)}(t) + A_{:,\Omega^c(t)}x_{\Omega^c(t)}(t-1) - b\|^2 \\
&= \|A_{:,\Omega(t)}(x^*_{\Omega(t)} + \delta_{\Omega(t)}(t)) \\
&\quad + A_{:,\Omega^c(t)}(x^*_{\Omega^c(t)} + \delta_{\Omega^c(t)}(t-1)) - b\|^2 \\
&= \|(Ax^* - b) + A_{:,\Omega(t)}\delta_{\Omega(t)}(t) + A_{:,\Omega^c(t)}\delta_{\Omega^c(t)}(t-1)\|^2 \\
&= \|(Ax^* - b)\|^2 + \|A_{:,\Omega(t)}\delta_{\Omega(t)}(t) + A_{:,\Omega^c(t)}\delta_{\Omega^c(t)}(t-1)\|^2,
\end{aligned}$$

where the last equality follows from the last statement of Lemma 1 and the Pythagorean Theorem. As a result, $\delta(t)$ is updated in each iteration of the outer loop such that $\|A_{:,\Omega(t)}\delta_{\Omega(t)}(t) + A_{:,\Omega^c(t)}\delta_{\Omega^c(t)}(t-1)\|$ is minimized. According to the equivalence between the second and third statements of Lemma 1, we have an explicit formula for $\delta_{\Omega(t)}(t)$:

$$\delta_{\Omega(t)}(t) = -A^\dagger_{:,\Omega(t)}A_{:,\Omega^c(t)}\delta_{\Omega^c(t)}(t-1). \tag{12}$$

Meanwhile, it follows from the property of the permutation matrix $P(t)$ that $\begin{bmatrix} \delta_{\Omega(t)}(t) \\ \delta_{\Omega^c(t)}(t) \end{bmatrix} = P(t)\delta(t)$. Thus, we obtain $\delta(t) = \Pi(t)\delta(t-1)$, which proves Lemma 2. □

Lemma 2 immediately leads to the following convergence result:

Corollary 1. *Let $x^* \in \mathbb{R}^n$ be the unique LS solution of* (1) *and suppose that agents reach perfect consensus after each inner loop in the sense of* (8)*. If the following property on the joint spectral radius of the matrices* (9) *holds*

$$\rho_\Pi := \limsup_{t\to\infty} |\lambda_{\max}(\Pi(t)\Pi(t-1)\cdots\Pi(1))|^{\frac{1}{t}} < 1, \tag{13}$$

then $x(t)$ converges to x^* exponentially with rate $\ln \rho_\Pi$.

We would like to emphasize that Corollary 1 is implicit, since the condition (13) depends on the specific matrix A, which cannot be determined a priori. However, it is worth noting that each $\Pi(t)$ matrix only possesses two eigenvalues, namely 0 and 1. As a result, the joint spectrum of the system must lie within the closed unit disk. In order to exclude eigenvalues residing on the unit circle, intuitively, one would require the sequence Ω to be "fully-mixed". We hence propose the following two assumptions on Ω.

Assumption 2 (Spanning set)**.** *It holds that*

$$\lim_{N\to\infty} \bigcup_{t=1}^{N} \Omega(t) = \mathsf{N}.$$

Assumption 3 (Periodicity)**.** *There exists $T \in \mathbb{N}$ such that*

$$\Omega(t+T) = \Omega(t) \quad \forall t \in \mathbb{N}.$$

We remark here that under Assumption 3, Assumption 2 is equivalent to

$$\bigcup_{i=1}^{T} \Omega(i) = \mathsf{N}. \tag{14}$$

The next theorem shows a surprising result, which is that under Assumptions 2 and 3, convergence is always guaranteed regardless of the value of A (although the convergence rate depends on A), so that our algorithm can always find the LS solution of (1).

Theorem 1. *Let $x^* \in \mathbb{R}^n$ be the unique LS solution of* (1) *and suppose the agents reach perfect consensus for each inner loop in the sense of* (8)*. Then under Assumptions 2 and 3, $x(t)$ converges to x^* exponentially.*

Proof. Denote $v(t) := A\delta(t)$, where recall $\delta(t) := x(t) - x^*$. We conclude from (12) that

$$\begin{aligned}
v(t) &= \begin{bmatrix} A_{:,\Omega(t)} & A_{:,\Omega^c(t)} \end{bmatrix} \begin{bmatrix} \delta_{\Omega(t)}(t) \\ \delta_{\Omega^c(t)}(t) \end{bmatrix} \\
&= \begin{bmatrix} A_{:,\Omega(t)} & A_{:,\Omega^c(t)} \end{bmatrix} \begin{bmatrix} 0_{|\Omega(t)|\times|\Omega(t)|} & -A^\dagger_{:,\Omega(t)} A_{:,\Omega^c(t)} \\ 0_{|\Omega^c(t)|\times|\Omega(t)|} & I_{|\Omega^c(t)|} \end{bmatrix} \begin{bmatrix} \delta_{\Omega(t)}(t-1) \\ \delta_{\Omega^c(t)}(t-1) \end{bmatrix} \\
&= \big(I_m - A_{:,\Omega(t)} A^\dagger_{:,\Omega(t)}\big) A_{:,\Omega^c(t)} \delta_{\Omega^c(t)}(t-1) \\
&= \big(I_m - A_{:,\Omega(t)} A^\dagger_{:,\Omega(t)}\big) \begin{bmatrix} A_{:,\Omega(t)} & A_{:,\Omega^c(t)} \end{bmatrix} \begin{bmatrix} \delta_{\Omega(t)}(t-1) \\ \delta_{\Omega^c(t)}(t-1) \end{bmatrix} \\
&= \big(I_m - A_{:,\Omega(t)} A^\dagger_{:,\Omega(t)}\big) v(t-1),
\end{aligned}$$

where we have also used the property of pseudo-inverse such that $A_{:,\Omega(t)} A^\dagger_{:,\Omega(t)} A_{:,\Omega(t)} = A_{:,\Omega(t)}$ for the second last equality. Note that $I_m - A_{:,\Omega(t)} A^\dagger_{:,\Omega(t)}$ is the orthogonal projection operator onto $\mathrm{im}(A_{:,\Omega(t)})^\perp$. Also, since $\mathrm{im}(A_{:,\Omega(t)})^\perp = \ker((A_{:,\Omega(t)})^\top)$, We conclude that

$$v(t) = \mathcal{P}_{\ker((A_{:,\Omega(t)})^\top)} v(t-1), \tag{15}$$

where $\mathcal{P}_{\mathcal{X}}$ denotes the projection onto the vector space $\mathcal{X}$. As a result, the sequence v is generated by *alternating projections*. Because of Assumption 3, we conclude from [15] that $\lim_{t\to\infty} v(t) = \mathcal{P}_{\mathcal{Y}} v(0)$, where

$$\mathcal{Y} = \bigcap_{t=1}^{T} \ker((A_{:,\Omega(t)})^\top) = \mathrm{im}\left(\begin{bmatrix} A_{:,\Omega(1)} & A_{:,\Omega(2)} & \cdots & A_{:,\Omega(T)} \end{bmatrix}\right)^\perp.$$

We can further conclude that $\mathcal{Y} = \mathrm{im}(A)^\perp$ by the equivalent condition (14) for Assumption 2. Now, recall $v(0) = A\delta(0) \in \mathrm{im}(A)$. Thus, $\mathcal{P}_{\mathcal{Y}} v(0) = \mathcal{P}_{\mathrm{im}(A)^\perp} v(0) = 0$. In other words, $\lim_{t\to\infty} v(t) = 0$. Because A is full column rank, this also implies that $\lim_{t\to\infty} \delta(t) = 0$. We have hence shown the convergence of $x(t)$ to x^*. To further show exponential convergence, note that it follows from (12) and Assumption 3 that

$$\delta(kT) = \big(\Pi(T-1)\cdots\Pi(1)\Pi(0)\big)^k \delta(0)$$

for any $k \in \mathbb{N}$. Hence, we must have $|\lambda_{\max}(\Pi(T-1)\cdots\Pi(1)\Pi(0))| < 1$, or

otherwise it contradicts with the convergence of the sequence δ. We thus have

$$\begin{aligned}\rho_\Pi &= \limsup_{t\to\infty} |\lambda_{\max}(\Pi(t)\Pi(t-1)\cdots\Pi(0))|^{\frac{1}{t}} \\ &= \limsup_{k\to\infty} |\lambda_{\max}(\Pi(T-1)\cdots\Pi(1)\Pi(0))|^{\frac{k}{kT}} < 1.\end{aligned}$$

By Corollary 1, $x(t)$ converges to x^* exponentially. □

4.2. Practical convergence result

Although it can be implied by [13, Theorem 1] that the convergence of our inner loop is exponential, in practice, consensus is never perfect due to the finite terminal condition. Errors between $x^{(i)}(t)$ and the LS solution of (7) exist, and we certainly do not wish such errors to accumulate per iteration of the outer loop. This can be ensured by the following practical convergence result.

Theorem 2. *Let $x^* \in \mathbb{R}^n$ be the unique LS solution of (1), and $z^*(t) \in \mathbb{R}^{|\Omega(t)|}$ be the LS solution of (7) at the t-th iteration of the outer loop. Suppose there exists $\bar{\epsilon} > 0$ such that the inner loop always terminates with $\|x^{(i)}_{\Omega(t)}(t) - z^*(t)\| \le \bar{\epsilon}$ for all $i \in \mathsf{L}, t \in \mathbb{N}$. Then, there exists $b, c > 0, \lambda \in (0,1)$ such that*

$$\|x^{(i)}(t) - x^*\| \le c\lambda^t \|x(0) - x^*\| + b\bar{\epsilon}$$

for all $i \in \mathsf{L}, t \in \mathbb{N}$.

Proof. Let $z : \mathbb{N} \to \mathbb{R}^n$ be constructed by

$$z(0) = x(0), \quad z(t) = P^\top(t) \begin{bmatrix} z^*(t) \\ z_{\Omega^c(t)}(t-1) \end{bmatrix},$$

where $z^*(t)$ is the LS square solution of (7). Denote $\epsilon^{(i)}(t) := x^{(i)}(t) - z(t)$. We have

$$\epsilon^{(i)}(0) = 0, \quad \epsilon^{(i)}(t) = P^\top(t) \begin{bmatrix} x^{(i)}_{\Omega(t)}(t) - z^*(t) \\ \epsilon^{(i)}_{\Omega^c(t)}(t-1) \end{bmatrix}.$$

Because of the assumption that $\|x^{(i)}_{\Omega(t)}(t) - z^*(t)\| \le \bar{\epsilon}$ and the fact that $P(t)$ is a permutation matrix, we conclude that each element of $\epsilon^{(i)}(t)$ must be bounded in the interval $[-\bar{\epsilon}, \bar{\epsilon}]$ and hence $\|\epsilon^{(i)}(t)\| \le \bar{\epsilon}\sqrt{n}$ for all $t \in \mathbb{N}$.

We now show that the sequence z admits an input-to-state-like property. By re-arrange the rows of (7) and substitute $x^{(i)}(t) = z(t) + \epsilon^{(i)}(t)$, we obtain

$$A_{:,\Omega(t)} z = b - A_{:,\Omega^c(t)} z(t-1) - u(t), \tag{16}$$

where $u(t)$ is a re-arrangement of the vector

$$\begin{bmatrix} A_{\Sigma(1),\Omega^c(t)} \epsilon^{(1)}_{\Omega^c(t)}(t-1) \\ A_{\Sigma(2),\Omega^c(t)} \epsilon^{(2)}_{\Omega^c(t)}(t-1) \\ \vdots \\ A_{\Sigma(l),\Omega^c(t)} \epsilon^{(l)}_{\Omega^c(t)}(t-1) \end{bmatrix}.$$

In particular, we have

$$\begin{aligned} \|u(t)\| &= \sqrt{\sum_{i=1}^{l} \|A_{\Sigma(i),\Omega^c(t)} \epsilon^{(i)}_{\Omega^c(t)}(t-1)\|^2} \\ &\leq \bar{\epsilon} \sqrt{n \sum_{i=1}^{l} \|A_{\Sigma(i),\Omega^c(t)}\|^2}. \end{aligned} \tag{17}$$

Because the LS solution of (7) is $z^*(t) = z_{\Omega(t)}(t)$, according to the second statement in Lemma 1, $\|A_{:,\Omega(t)} z_{\Omega(t)}(t) + A_{:,\Omega^c(t)} x_{\Omega^c(t)}(t-1) + u(t) - b\|$ is minimized. Now further denote $\delta(t) := z(t) - x^*$. Similar to the proof of Lemma 2, we conclude that this optimization problem is equivalent to minimizing

$$\|(Ax^* - b) + A_{:,\Omega(t)} \delta_{\Omega(t)}(t) + A_{:,\Omega^c(t)} \delta_{\Omega^c(t)}(t-1) + u(t)\|,$$

which, according to the equivalence between the second and third statements of Lemma 1, has an explicit formula

$$\begin{aligned} \delta_{\Omega(t)}(t) &= -A^{\dagger}_{:,\Omega(t)} \left(Ax^* - b + A_{:,\Omega^c(t)} \delta_{\Omega^c(t)}(t-1) + u(t) \right) \\ &= -A^{\dagger}_{:,\Omega(t)} A_{:,\Omega^c(t)} \delta_{\Omega^c(t)}(t-1) - A^{\dagger}_{:,\Omega(t)} u(t). \end{aligned}$$

Here, we have used the last statement of Lemma 1 such that $A^{\dagger}_{:,\Omega(t)}(Ax^* - b) = 0$.

As a result, we conclude that

$$\begin{aligned}\delta(t) &= P^\top(t)\begin{bmatrix}\delta_{\Omega(t)}(t)\\ \delta_{\Omega^c(t)}(t)\end{bmatrix}\\ &= P^\top(t)\left(\begin{bmatrix}0_{|\Omega(t)|\times|\Omega(t)|} & -A^\dagger_{:,\Omega(t)}A_{:,\Omega^c(t)}\\ 0_{|\Omega^c(t)|\times|\Omega(t)|} & I_{|\Omega^c(t)|}\end{bmatrix}\begin{bmatrix}\delta_{\Omega(t)}(t-1)\\ \delta_{\Omega^c(t)}(t-1)\end{bmatrix}\right.\\ &\quad\left.-\begin{bmatrix}A^\dagger_{:,\Omega(t)}\\ 0_{|\Omega^c(t)|}\end{bmatrix}u(t)\right)\\ &= \Pi(t)\delta(t-1) - P^\top(t)\begin{bmatrix}A^\dagger_{:,\Omega(t)}\\ 0_{|\Omega^c(t)|}\end{bmatrix}u(t),\end{aligned}$$

which gives a discrete-time linear time-varying (LTV) dynamics of δ with input u. Now note that the sequence of matrices Π is periodic by Assumption 3, and we have proven that 0 is exponentially stable for the unforced dynamics in Theorem 1. Hence, this LTV system of δ is exponentially input-to-state stable with respect to u; futhermore, because $\|u(t)\|$ is bounded as in (17), there exist $c, \tilde{b} > 0, \lambda \in (0,1)$ such that $\|\delta(t)\| \le c\lambda^t\|\delta(0)\| + \tilde{b}\bar{\epsilon}$ for all $t \in \mathbb{N}$. Finally, we conclude from triangle inequality that

$$\begin{aligned}\|x^{(i)}(t) - x^*\| &\le \|x^{(i)}(t) - z(t)\| + \|z(t) - x^*(t)\|\\ &\le c\lambda^t\|\delta(0)\| + (\tilde{b} + \sqrt{n})\bar{\epsilon},\end{aligned}$$

which proves Theorem 2 with $b = \tilde{b} + \sqrt{n}$. □

5. Simulation

We consider a numerical example where the dimension of the LAE is given by $m = n = 20$. Let there be 5 agents; that is, $l = 5$, such that they form a circular communication network. In addition, the partition Σ and the scheduling sequence Ω are designed such that

$$\begin{aligned}\Sigma(i) &= \{4i-3, 4i-2, 4i-2, 4i\} \quad \forall i \in \mathsf{L}\\ \Omega(t) &= \{2\bmod(t,10)+1, 2\bmod(t,10)+2\} \quad \forall t \in \mathbb{N}.\end{aligned}$$

With this setup, Assumption 1 is satisfied with $m^* = n^* = 4$, and both Assumptions 2 and 3 hold.

For a pair of randomly generated matrix and vector A, b, we consider four scenarios such that the inner loop of our algorithm terminates at $s_{\max}$ iterations, $s_{\max} = 10, 20, 50, 100$. The outer loop ends when $\|x^{(i)}(t) - x^{(i)}(t-1)\| \leq \epsilon_{\text{tol}}\|x^{(i)}(t-1)\|$ for all $i \in \mathsf{L}$, where $\epsilon_{\text{tol}} = 10^{-6}$. Although the incremental relative error, $\frac{\|x^{(i)}(t)-x^{(i)}(t-1)\|}{\|x^{(i)}(t-1)\|}$ depends on agents, in practice we observe that all of them are very close and hence we use $i = 1$ to represent the incremental relative error. The plots of incremental relative error versus t are shown in Figure 1, together with the function $\beta(t) := \rho_{\Pi}^{t}$, where we recall the joint spectral radius ρ_{Π} given by (13) in Lemma 2 is the convergence rate for perfect consensus. We observe that with larger $s_{\max}$ (50 and 100), the plots of incremental relative errors become parallel to the plot of $\beta(t)$, indicating that the convergence rate becomes almost identical to the case of perfect consensus. Meanwhile, with larger $s_{\max}$, the total number of outer loop iterations is also bounded if the terminal condition is based on a threshold on the incremental relative error.

In contrast to the incremental relative error, the true relative error is given by $\frac{\|x^{(i)}-x^*\|}{\|x^*\|}$. Similar to the incremental relative error, the true relative error is also insensitive to different agents. The total number of outer loop iterations, computation time and true relative error for these four scenarios are recorded in Table 1. While the computation time increases, the true relative error significantly reduces with respect to $s_{\max}$. We remark here that our current algorithm is merely a coarse prototype implemented in `MATLAB`, where there is still a huge room for improving efficiency.

$s_{\max}$	Outer loop iterations	Time (ms)	True relative error
10	3370	438	62.9%
20	11960	2884	19.2%
50	8690	4922	0.17%
100	8460	9210	0.00847%

Table 1: Results for different inner loop terminal conditions.

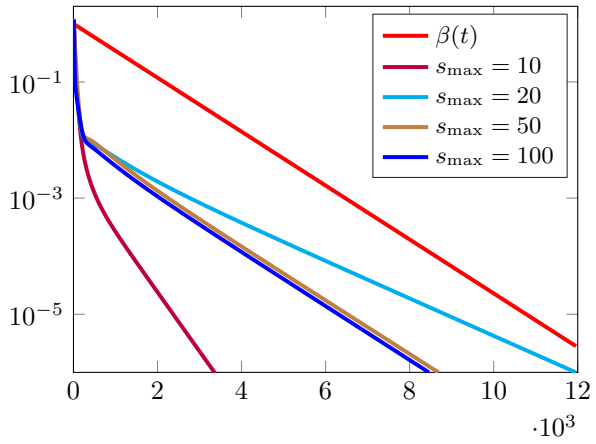

Figure 1: Plots of incremental relative error vs. outer loop iteration for different inner loop terminal conditions.

6. Discussion and conclusion

In this study, we introduced a novel scalable distributed algorithm tailored for solving LAEs while considering limitations in computational power and network bandwidth. Additionally, we conducted convergence analysis, addressing both the ideal scenario with perfect consensus and the practical scenario with errors.

There are several potential extensions to our study of the distributed algorithm that could be explored. First, we can investigate relaxing Assumptions 2 and 3 on the scheduling sequence Ω. It may be possible to consider a weaker condition based on *quasi-periodicity* [16], which could still be sufficient for the convergence of the algorithm. However, it is important to note that such an extension might add minimal practical value, as the scheduling sequence can be customized for the algorithm and made simple. On the other hand, determining which scheduling sequence Ω can expedite the convergence rate is both theoretically and practically intriguing. As highlighted in Corollary 1, the convergence rate of the algorithm is related to properties of the matrix A. Exploring and identifying the specific properties of A that impact the convergence rate is an important task. Developing techniques to compute or estimate these

properties within the distributed framework presents an additional challenging research problem. Lastly, instead of sequentially updating parts of the solution in the outer loop, an alternative direction of research could involve designing a distributed algorithm where each agent updates a distinct part of the solution. This approach would require non-trivial analysis to ensure convergence and efficiency.

References

[1] M. Anthony and P. L. Bartlett, *Neural network learning: Theoretical foundations*. Cambridge, UK: Cambridge University Press, 1999.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[3] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2863–2878, 2015.

[4] B. D. O. Anderson, S. Mou, A. S. Morse, and U. Helmke, "Decentralized gradient algorithm for solution of a linear equation," *Numerical Algebra, Control and Optimization*, vol. 6, no. 3, pp. 319–328, 2016.

[5] P. Wang, S. Mou, J. Lian, and W. Ren, "Solving a system of linear equations: From centralized to distributed algorithms," *Annual Reviews in Control*, vol. 47, pp. 306–322, 2019.

[6] J. Liu, A. S. Morse, A. Nedić, and T. Başar, "Exponential convergence of a distributed algorithm for solving linear algebraic equations," *Automatica*, vol. 83, pp. 37–46, 2017.

[7] G. Shi, B. D. O. Anderson, and U. Helmke, "Network flows that solve linear equations," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2659–2674, 2017.

[8] T. Yang, J. George, J. Qin, X. Yi, and J. Wu, "Distributed least squares solver for network linear equations," *Automatica*, vol. 113, p. 108798, 2020.

[9] S. S. Kia, B. V. Scoy, J. Cortés, R. A. Freeman, K. M. Lynch, and S. Martinez, "Tutorial on dynamic average consensus: The problem, its applications, and the algorithms," *IEEE Control Systems Magazine*, vol. 39, no. 3, pp. 40–72, 2019.

[10] P. Srivastava and J. Cortés, "Solving linear equations with separable problem data over directed networks," *IEEE Control Systems Letters*, vol. 6, pp. 596–601, 2022.

[11] X. Wang, S. Mou, and B. D. O. Anderson, "Scalable, distributed algorithms for solving linear equations via double-layered networks," *IEEE Transactions on Automatic Control*, vol. 65, no. 3, pp. 1132–1143, 2020.

[12] Y. Huang, Z. Meng, and J. Sun, "Scalable distributed least square algorithms for large-scale linear equations via an optimization approach," *Automatica*, vol. 146, p. 110572, 2022.

[13] X. Wang, J. Zhou, S. Mou, and M. J. Corless, "A distributed algorithm for least squares solutions," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4217–4222, 2019.

[14] Y. Shang, "A distributed memory parallel Gauss–Seidel algorithm for linear algebraic systems," *Computers & Mathematics with Applications*, vol. 57, no. 8, pp. 1369–1376, 2009.

[15] I. Halperin, "The product of projection operators," *Acta Sci. Math. (Szeged)*, vol. 23, no. 1, pp. 96–99, 1962.

[16] M. Sakai, "Strong convergence of infinite products of orthogonal projections in Hilbert space," *Applicable Analysis*, vol. 59, no. 1-4, pp. 109–120, 1995.