2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

A Resilient Distributed Algorithm for Solving Linear Equations*

Jingxuan Zhu Alvaro Velasquez Ji Liu

Abstract— **This paper presents a resilient distributed algorithm for solving a system of linear algebraic equations over a multi-agent network in the presence of Byzantine agents capable of arbitrarily introducing untrustworthy information in communication. It is shown that the algorithm causes all non-Byzantine agents' states to converge to the same least squares solution exponentially fast, provided appropriate levels of graph redundancy and objective redundancy are established. An explicit convergence rate is also provided.**

I. Problem

There has been considerable interest in designing distributed algorithms for solving a possibly large system of linear algebraic equations over a multi-agent network, stemming from the work of [1]. A review of this topic can be found in [2]. While various problem formulations have been proposed and studied, following [1] we focus on the following basic and important information distribution setting.

Consider a multi-agent network consisting of n agents labeled 1 through n for the purpose of presentation. Each agent is not aware of such a global identification number, but is capable of distinguishing between its neighbors. The neighbor relations among the n agents are characterized by a directed graph $\mathbb{G} = (\mathcal{V}, \mathcal{E})$ whose vertices correspond to agents and whose directed edges (or arcs) depict neighbor relations, where $\mathcal{V} = \{1, \ldots, n\}$ is the vertex set and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the directed edge set.[1] We say that agent i is a neighbor of agent j if $(i, j) \in \mathcal{E}$. The directions of arcs represent the directions of information flow in that each agent can receive information only from its neighbors.

Each agent $i \in \mathcal{V}$ knows a pair of "private" real-valued matrices $(A_i^{r_i \times d}, b_i^{r_i \times 1})$ which are only known to agent i. The problem of interest is to devise local algorithms, one for each agent, which will enable all n agents to simultaneously and iteratively compute the same least squares solution to the linear algebraic equation $Ax = b$ where

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{bmatrix}_{r \times d} \quad \text{and} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}_{r \times 1}$$

with $r = \sum_{i=1}^n r_i$. Such a distributed least squares problem has been studied in the literature, e.g., [1], [3]–[5]. Let

$$\mathcal{X}^* \triangleq \arg\min_x \|Ax - b\|_2^2 = \arg\min_x \sum_{i=1}^n \|A_i x - b_i\|_2^2$$

be the set of all least squares solutions, where $\|\cdot\|_2$ denotes the 2-norm. It is easy to see from the above equality that the distributed least squares problem can be reformulated as a distributed convex optimization problem and thus is solvable via a vast number of existing distributed optimization algorithms [6], [7]. It is well known that $\mathcal{X}^* = \{x : A'Ax = A'b\}$ and is always nonempty. We will use this fact of least squares solutions without special mention in the sequel.

In this paper we consider a more challenging variant of the distributed least squares problem in the presence of Byzantine agents capable of transmitting arbitrary values to other agents and transferring conflicting values to different agents at any time. We use $\mathcal{F}$ to denote the set of Byzantine agents and $\mathcal{H}$ to denote the set of normal (non-Byzantine) agents. Which agents are Byzantine is unknown to normal agents. It is assumed that the network may have at most β Byzantine agents. The goal of the normal agents is to cooperatively reach a consensus at the same least squares solution to $Ax = b$.

From the preceding discussion, the resilient distributed least squares problem just described can also be treated as a resilient distributed convex optimization problem. It turns out that very few papers accurately solve this resilient problem with theoretical guarantees.

The resilient distributed optimization algorithm in [8] is expected to be applicable to the problem under consideration with some modification. The algorithm in [8] is based on the subgradient method (for convex but not necessarily differentiable objective functions) and a nonempty $\mathcal{X}^*$ interior assumption (cf. Assumption 1 in [8]), and thus specialization of that algorithm to the problem of interest here needs a more careful treatment. Moreover, the algorithm in [8] makes use of time-varying diminishing stepsizes and thus cannot converge exponentially fast. It is worth mentioning that the state-of-the-art least squares and distributed least squares algorithms achieve (at least) exponential convergence. It will be clear shortly that the problem under consideration is

*Proofs of some assertions in this paper are omitted due to space limitations and will be included in an expanded version of the paper.

This material is based upon work supported in part by the National Science Foundation under Grant No. 2230101, by the Air Force Office of Scientific Research under award number FA9550-23-1-0175, and by Stony Brook University's Office of the Vice President for Research through a Seed Grant. J. Liu acknowledges Air Force Research Laboratory Information Directorate (AFRL/RI); part of this work took place while he was there via the 2022 Visiting Faculty Research Program (VFRP).

J. Zhu is with the Department of Applied Mathematics and Statistics at Stony Brook University (`jingxuan.zhu@stonybrook.edu`). A. Velasquez is with the Department of Computer Science at University of Colorado Boulder (`alvaro.velasquez@colorado.edu`). J. Liu is with the Department of Electrical and Computer Engineering at Stony Brook University (`ji.liu@stonybrook.edu`).

[1]We use $\mathcal{A} \subset \mathcal{B}$ to denote that $\mathcal{A}$ is a subset of $\mathcal{B}$.

closely related to a resilient version of so-called constrained consensus problem [9]. In general a discrete-time constrained consensus cannot be reached exponentially fast unless a certain constrained set regularity condition is satisfied and exploited in analysis [10]. We will further comment on this point in the next section.

This paper proposes a resilient distributed least squares algorithm with guaranteed exponentially fast convergence. The algorithm follows two quantifiable redundancy notions in [8],[2] namely objective redundancy and graph redundancy, with the former being tailored for distributed linear equations. An explicit rate of convergence is derived, reflecting the effects of quantified levels of both graph redundancy and objective redundancy.

II. Redundancy

It is easy to see that a multi-agent system without an attack detection/correction capability is unable to solve the resilient distributed least squares problem unless certain redundancy is established. We begin with the objective redundancy.

Definition 1: An n-agent network is called k-redundant, $k \in \{0, 1, \dots, n-1\}$, if for any subsets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathcal{V}$ with $|\mathcal{S}_1| = |\mathcal{S}_2| = n-k$, there holds[3]

$$\arg\min_x \sum_{i\in\mathcal{S}_1} \|A_i x - b_i\|_2^2 = \arg\min_x \sum_{i\in\mathcal{S}_2} \|A_i x - b_i\|_2^2.$$

Note that for any nonempty agent subset $\mathcal{U} \subset \mathcal{V}$, $\arg\min_x \sum_{i\in\mathcal{U}} \|A_i x - b_i\|_2^2$ is the least squares solution to the linear equation $A_{\mathcal{U}} x = b_{\mathcal{U}}$ where

$$A_{\mathcal{U}} = \begin{bmatrix} A_{\pi(1)} \\ A_{\pi(2)} \\ \vdots \\ A_{\pi(|\mathcal{U}|)} \end{bmatrix}, \qquad b = \begin{bmatrix} b_{\pi(1)} \\ b_{\pi(1)} \\ \vdots \\ b_{\pi(|\mathcal{U}|)} \end{bmatrix},$$

and $\pi : \mathcal{U} \to \mathcal{U}$ is any permutation map. Thus,

$$\arg\min_x \sum_{i\in\mathcal{U}} \|A_i x - b_i\|_2^2 = \{x : A'_{\mathcal{U}} A_{\mathcal{U}} x = A'_{\mathcal{U}} b_{\mathcal{U}}\}.$$

The above objective redundancy is a well-defined quantifiable notion because of the following properties.

Lemma 1: If an n-agent network is k-redundant, then for any subsets $\mathcal{S}, \mathcal{L} \subset \mathcal{V}$ with $|\mathcal{S}| = n-k$ and $|\mathcal{L}| \geq n-k$,

$$\arg\min_x \sum_{i\in\mathcal{S}} \|A_i x - b_i\|_2^2 = \arg\min_x \sum_{i\in\mathcal{L}} \|A_i x - b_i\|_2^2.$$

The lemma is essentially a special case of Lemma 1 in [8] and immediately implies the following results.

Corollary 1: If an n-agent network is k-redundant, then for any subset $\mathcal{S} \subset \mathcal{V}$ with $|\mathcal{S}| \geq n-k$, there holds

$$\arg\min_x \sum_{i\in\mathcal{S}} \|A_i x - b_i\|_2^2 = \mathcal{X}^*.$$

[2]The two notions were prompted by preexisting ones, e.g., [11] and [12]; see detailed discussion in [8, Sections 1 and 2.1].

[3]We use $|\mathcal{S}|$ to denote the cardinality of a set $\mathcal{S}$.

Corollary 2: If an n-agent network is $(k+1)$-redundant with $k \geq 0$, then it is k-redundant.

More can be said.

Proposition 1: If an n-agent network is k-redundant with $k \geq 1$, then $\mathcal{X}^* = \bigcap_{i=1}^n \{x : A'_i A_i x = A'_i b_i\}$.

Proof of Proposition 1: Note that $A'A = \sum_{i=1}^n A'_i A_i$ and $A'b = \sum_{i=1}^n A'_i b_i$. Then, $\mathcal{X}^* = \{x : \sum_{i=1}^n A'_i A_i x = \sum_{i=1}^n A'_i b_i\}$, which implies that $\mathcal{X}^* \supset \bigcap_{i=1}^n \{x : A'_i A_i x = A'_i b_i\}$. To prove the lemma, it suffices to prove $\mathcal{X}^* \subset \bigcap_{i=1}^n \{x : A'_i A_i x = A'_i b_i\}$, or equivalently, $\mathcal{X}^* \subset \{x : A'_i A_i x = A'_i b_i\}$ for all $i \in \mathcal{V}$. To this end, pick any $j \in \mathcal{V}$ and $x^* \in \mathcal{X}^*$. From Corollary 1 with $k \geq 1$,

$$x^* \in \arg\min_x \sum_{i\in\mathcal{V}\setminus\{j\}} \|A_i x - b_i\|_2^2,$$

which implies that $\sum_{i\in\mathcal{V}\setminus\{j\}} A'_i A_i x^* = \sum_{i\in\mathcal{V}\setminus\{j\}} A'_i b_i$. The difference between this equality and $\sum_{i=1}^n A'_i A_i x^* = \sum_{i=1}^n A'_i b_i$ yields $A'_j A_j x^* = A'_j b_j$. ■

Proposition 1 implies that with objective redundancy all agents share at least one common least squares solution and thus the resilient distributed least squares problem boils down to a resilient distributed linear equation problem. The algorithm to be presented exploits this important fact.

Proposition 1 also maps the resilient problem under study to a resilient constrained consensus problem. Resilient constrained consensus has been partially solved in [13] only for complete graphs and studied in [14] with an incomplete proof. It is worth emphasizing that discrete-time constrained consensus, first proposed in [9], in general does not enjoy exponentially fast convergence (see [9, Proposition 2], [10, Theorem 6], and [15, Theorem 1]), let alone a resilient variant. An exponentially fast constrained consensus can be achieved when all agents' local constrained sets meet a so-called set regularity condition, subsuming linear (in)equalities as a special case, but its analysis requires a more careful treatment even for a non-Byzantine multi-agent network [10, Section V.C].

We will specialize a constrained consensus approach, combining a multi-dimensional resilient consensus idea in [8], to the resilient problem of interest here and appeal to a distributed linear equation analysis tool developed in [1], [16]. This combination yields a fully resilient distributed least squares algorithm with any arbitrary initialization and an exponential convergence guarantee. Although exponential convergence is an intuitive result at the first glance, its establishment is not trivial. This is because major existing analyses for non-Byzantine distributed linear equations and constrained consensus processes over time-varying graphs are based on jointly strongly connected graphs [1] or jointly strongly connected components with the same vertex subset[4] [15], [17], whereas the analysis for the resilient algorithm

[4]A lifting approach is typically used to analyze these discrete-time distributed algorithms with bounded delays in which a nominal time-dependent expanded graph replaces the underlying neighbor graph. Jointly strongly connected neighbor graphs are jointly strongly connected components of the nominal expanded graph which thus share the same vertex subset.

here will cope with time-varying rooted graphs whose roots (maximal strongly connected components) may arbitrarily change over time (cf. Lemma 3). The definitions of rooted and strongly connected graphs are given as follows.

Let us call a vertex i in a directed graph $\mathbb{G}$ a root of $\mathbb{G}$ if for each other vertex j of $\mathbb{G}$, there is a directed path from i to j. In other words, i is a root of $\mathbb{G}$ if it is the root of a directed spanning tree of $\mathbb{G}$. We say that $\mathbb{G}$ is rooted at i if i is in fact a root. It is not hard to see that a rooted graph $\mathbb{G}$ has a unique maximal strongly connected component whose vertices are all roots of $\mathbb{G}$. A directed graph $\mathbb{G}$ is called strongly connected if there is a directed path from every vertex to every other vertex. Thus, every vertex in a strongly connected graph is a root. Any strongly connected graph must be rooted, but not vice versa.

We will also need a graph redundancy notion from [8].

Definition 2: An (r, s)-reduced graph of a directed graph $\mathbb{G}$ with n vertices, with $r, s \geq 0$ and $r + s \leq n - 1$, is a subgraph of $\mathbb{G}$ obtained by first picking any vertex subset $\mathcal{S} \subset \mathcal{V}$ with $|\mathcal{S}| = n - r$ and then removing from each vertex of the subgraph induced by $\mathcal{S}$, $\mathbb{G}_{\mathcal{S}}$, arbitrary s incoming edges in $\mathbb{G}_{\mathcal{S}}$. A directed graph $\mathbb{G}$ is called (r, s)-resilient if all its (r, s)-reduced graphs are rooted.

It is easy to see that if a directed graph is (r_1, s_1)-resilient, then for any nonnegative $r_2 \leq r_1$ and $s_2 \leq s_1$, the graph is also (r_2, s_2)-resilient. More can be said. If a directed graph is (r, s)-resilient, each of its vertices has at least $(r + s + 1)$ neighbors [8, Lemma 2].

Equipping a multi-agent network with certain levels of both objective and graph redundancy, allows us to present the following feasible resilient algorithm.

III. Algorithm

Each agent i has a time-dependent state vector $x_i(t)$ taking values in $\mathbb{R}^d$, which represents its estimate of a least squares solution at time t and can be arbitrarily initialized at $t = 0$. It is assumed that the information agent i receives from a normal neighbor j is only the current state vector of neighbor j. The algorithm will make use of the multi-dimensional resilient consensus idea[5] in [8] which needs the following notation.

Let $\mathcal{N}_i$ be the set of neighbors of agent i in the neighbor graph $\mathbb{G}$ and $\mathcal{A}_i$ denote the collection of all those subsets of $\mathcal{N}_i$ whose cardinality is $(d + 1)\beta + 1$. It is easy to see that the number of all such subsets is[6]

$$a_i \triangleq \binom{|\mathcal{N}_i|}{(d+1)\beta+1}, \tag{1}$$

assuming $|\mathcal{N}_i| \geq (d+1)\beta+1$, and label them $\mathcal{A}_{i1}, \ldots, \mathcal{A}_{ia_i}$. For each $j \in \{1, \ldots, a_i\}$, let $\mathcal{B}_{ij}$ denote the collection of all those subsets of $\mathcal{A}_{ij}$ whose cardinality is $d\beta + 1$. For any agent i and any subset of its neighbors $\mathcal{S} \subset \mathcal{N}_i$, we use $\mathcal{C}_{i\mathcal{S}}(t)$ to denote the convex hull of all $x_{ji}(t)$, $j \in \mathcal{S}$ where $x_{ji}(t)$ denotes the vector agent j sends to agent i. If agent j is a normal agent, $x_{ji}(t) = x_j(t)$ for all possible i. If agent j is a Byzantine agent, each $x_{ji}(t)$ can be an arbitrary vector.

[5]The idea was prompted by and simplified from the preexisting one in [14]; see detailed discussion in [8, Sections 1].

[6]$\binom{n}{k}$ denotes the number of k-combinations from a set of n elements.

Algorithm: At each discrete time $t \in \{0, 1, 2, \ldots\}$, each agent i first picks an arbitrary point

$$y_{ij}(t) \in \bigcap_{\mathcal{S} \in \mathcal{B}_{ij}} \mathcal{C}_{i\mathcal{S}}(t) \tag{2}$$

for each $j \in \{1, \ldots, a_i\}$, and then updates its state by setting

$$v_i(t) = \frac{1}{1 + a_i}\Big(x_i(t) + \sum_{j=1}^{a_i} y_{ij}(t)\Big), \tag{3}$$

$$x_i(t+1) = P_i v_i(t) + (A_i' A_i)^{\dagger} A_i' b_i, \tag{4}$$

where P_i is the orthogonal projection on the kernel of A_i and $M^{\dagger}$ denotes the Moore-Penrose inverse of matrix M. $\square$

Update (3) is the multi-dimensional resilient consensus step. The idea behind update (4) is as follows. Note that

$$P_i = I - A_i^{\dagger} A_i = I - (A_i' A_i)^{\dagger} A_i' A_i \tag{5}$$

since the kernel of $A_i' A_i$ equals the kernel of A_i. Using the standard quadratic programming with equality constraints, it is straightforward to show that $x_i(t + 1)$ in (4) is a solution to $\min_x \|x - v_i(t)\|_2$ subject to $A_i' A_i x = A_i b_i$. From this point of view, updates (3)–(4) can be regarded as a resilient variant of affine equality constrained consensus [10].

To state the main result, we need the following notation. For a directed graph $\mathbb{G}$, let $\mathcal{R}_{r,s}(\mathbb{G})$ denote the set of all (r, s)-reduced graphs of $\mathbb{G}$. For a rooted graph $\mathbb{G}$, we use $\kappa(\mathbb{G})$ to denote the size of the unique maximal strongly connected component whose vertices are all roots of $\mathbb{G}$; in other words, $\kappa(\mathbb{G})$ equals the number of roots of $\mathbb{G}$. For any (r, s)-resilient graph $\mathbb{G}$, define

$$\kappa_{r,s}(\mathbb{G}) \triangleq \min_{\mathbb{H} \in \mathcal{R}_{r,s}(\mathbb{G})} \kappa(\mathbb{H}),$$

which denotes the smallest possible number of roots in any (r, s)-reduced graphs of $\mathbb{G}$.

Theorem 1: If $\mathbb{G}$ is $(\beta, d\beta)$-resilient and the n-agent network is $(n - \kappa_{\beta,d\beta}(\mathbb{G}))$-redundant, then all $x_i(t)$, $i \in \mathcal{H}$ will converge to the same least squares solution to $Ax = b$ exponentially fast.

A $(\beta, d\beta)$-resilient $\mathbb{G}$ ensures that each agent has at least $(d + 1)\beta + 1$ neighbors at each time t [8, Lemma 2]. This further guarantees that $y_{ij}(t)$ in (2) must exist [8, Lemma 5].

Theorem 1 is a direct consequence of Theorems 2 and 3 in the next subsection. From the proof of Theorem 2 and using the same arguments as in the proof of Corollary 1 in [1], it is straightforward to obtain the following convergence rate result of the algorithm.

Corollary 3: Suppose that $Ax = b$ has a unique least squares solution. If $\mathbb{G}$ is $(\beta, d\beta)$-resilient and the n-agent network is $(n - \kappa_{\beta,d\beta}(\mathbb{G}))$-redundant, then all $x_i(t)$, $i \in \mathcal{H}$ converge to the least squares solution as $t \to \infty$ at the rate as λ^t converges to zero, where

$$\lambda = \Big(1 - (|\mathcal{H}| - 1)(1 - \rho)\eta^{\tau}\Big)^{\frac{1}{\tau}},$$

τ is a positive integer such that for any $p \geq \tau$, the matrix $P(W(t+p) \otimes I) \cdots P(W(t+1) \otimes I)P(W(t) \otimes I)$ is a contraction in the mixed matrix norm for all $t \geq 0$, $\rho = \max_{\mathcal{C}} \|P_{j_1}P_{j_2} \cdots P_{j_{\tau+1}}\|_2$ with $\mathcal{C}$ being the set of all those products of the orthogonal projection matrices in $\{P_1, P_2, \ldots, P_{|\mathcal{H}|}\}$ of length $\tau+1$ which are complete, and η is defined in (10).

The definition of "complete" products of orthogonal projection matrices will be given in the next subsection. It guarantees that, with set $\mathcal{C}$ being compact, ρ is strictly less than one. With this fact, it is not hard to show that $\lambda \in [0,1)$ provided more than one normal agent exists in the network. It is worth mentioning that, from the proof of Proposition 2, the value of τ is influenced by the levels of both graph redundancy and objective redundancy.

The above convergence rate result can be straightforwardly extended to the case when $Ax = b$ has more than one least squares solution using the proof of Theorem 3.

A. Analysis

To analyze the algorithm, we first derive the dynamics of normal agents which reject the influence of Byzantine agents because of step (2) of the algorithm. To this end, we need the following lemma.

Lemma 2: [8, Lemma 6] $v_i(t)$ in (3) can be expressed as a convex combination of $x_i(t)$ and $x_k(t)$, $k \in \mathcal{N}_i \cap \mathcal{H}$,

$$v_i(t) = w_{ii}(t)x_i(t) + \sum_{k \in \mathcal{N}_i \cap \mathcal{H}} w_{ik}(t)x_k(t), \qquad (6)$$

where $w_{ii}(t)$ and $w_{ik}(t)$ are nonnegative numbers satisfying $w_{ii}(t) + \sum_{k\in\mathcal{N}_i\cap\mathcal{H}} w_{ik}(t) = 1$, and there exists a positive constant η such that for all $i \in \mathcal{H}$ and t, $w_{ii}(t) \geq \eta$ and among all $w_{ik}(t)$, $k \in \mathcal{N}_i \cap \mathcal{H}$, at least $|\mathcal{N}_i \cap \mathcal{H}| - d\beta$ of them are bounded below by η.

From (4) and Lemma 2, the updates of all normal agents can be written as

$$\begin{aligned} x_i(t+1) = P_i\Big(w_{ii}(t)x_i(t) + \sum_{k\in\mathcal{N}_i\cap\mathcal{H}} w_{ik}(t)x_k(t)\Big) \\ + (A_i'A_i)^\dagger A_i'b_i, \qquad i \in \mathcal{H}, \qquad (7) \end{aligned}$$

which decouples from the dynamics of Byzantine agents. Let x^* be an arbitrary point in $\mathcal{X}^*$ and define $y_i(t) = x_i(t) - x^*$ for any $i \in \mathcal{H}$. Then, from (7), for all $i \in \mathcal{H}$,

$$\begin{aligned} y_i(t+1) &= P_i\Big(w_{ii}(t)y_i(t) + \sum_{k\in\mathcal{N}_i\cap\mathcal{H}} w_{ik}(t)y_k(t)\Big) \\ &\quad + (A_i'A_i)^\dagger A_i'b_i + P_ix^* - x^* \\ &= P_i\Big(w_{ii}(t)y_i(t) + \sum_{k\in\mathcal{N}_i\cap\mathcal{H}} w_{ik}(t)y_k(t)\Big), \qquad (8) \end{aligned}$$

where we used the fact that $(A_i'A_i)^\dagger A_i'b_i + P_ix^* - x^* = 0$ for all $i \in \mathcal{H}$ which can be straightforwardly proved by [18, Theorem 2] and (5).

Without loss of generality, we label all normal agents from 1 to $|\mathcal{H}|$, i.e., $\mathcal{H} = \{1, 2, \ldots, |\mathcal{H}|\}$, in the sequel.

To proceed, let $y(t)$ denote a stack of all $y_i(t)$, $i \in \mathcal{H}$ with the index in a top-down ascending order, i.e., $y(t) = [y_1'(t), y_2'(t), \ldots, y_{|\mathcal{H}|}'(t)]'$. Then, the updates in (8) can be written in the form of a state equation:

$$y(t+1) = P\big(W(t) \otimes I\big)y(t), \qquad (9)$$

where each $W(t) = [w_{ij}(t)]$ is a $|\mathcal{H}| \times |\mathcal{H}|$ stochastic matrix with positive diagonal entries, $\otimes$ denotes the Kronecker product, I denotes the $d \times d$ identity matrix, and P is a $d|\mathcal{H}| \times d|\mathcal{H}|$ block diagonal matrix whose ith diagonal block is P_i, $i \in \mathcal{H}$. It is easy to see that P is also an orthogonal projection matrix.

Define the graph of an $m \times m$ matrix M as a direct graph with m vertices and an arc from vertex i to vertex j whenever the ji-th entry of M is nonzero. We will write $\gamma(M)$ for the graph of a matrix M.

Lemma 3: [8, Lemma 7] If $\mathbb{G}$ is $(\beta, d\beta)$-resilient, the graph of each $W(t)$ in (9) has a rooted spanning subgraph and all the diagonal entries and those off-diagonal entries of $W(t)$ corresponding to the rooted spanning subgraph are uniformly bounded below by a positive number

$$\eta \triangleq \min_{i\in\mathcal{V}} \frac{1}{(d\beta+1)(1+a_i)\binom{(d+1)\beta+1}{d\beta+1}}. \qquad (10)$$

It is worth emphasizing that although (9) shares almost the same state equation as that in [1], which is $y(t+1) = P(S(t) \otimes I)Py(t)$, a critical difference between the two is that the graphs of stochastic matrices $S(t)$ in [1] are (jointly) strongly connected, whereas each graph of stochastic matrix $W(t)$ here is rooted (cf. Lemma 3) with possibly different roots. This is why analysis of the algorithm needs more careful treatment.

We first consider the case when $Ax = b$ has a unique least squares solution, i.e., $x^* \in \mathcal{X}^*$ is unique. From Proposition 1, all $A_ix = b_i$, $i \in \mathcal{V}$ share a unique least squares solution. Let $\mathcal{P}_i$ denote the column span of P_i for all i. Since $\mathcal{P}_i =$ kernel $A_i =$ kernel $A_i'A_i$, the least squares solution being unique is equivalent to

$$\bigcap_{i=1}^{n} \mathcal{P}_i = 0. \qquad (11)$$

More can be said. The following lemma is a direct consequence of Corollary 1.

Lemma 4: If the n-agent network is k-redundant and (11) holds, then for any subset $\mathcal{S} \subset \mathcal{V}$ with $|\mathcal{S}| \geq n-k$, there holds $\bigcap_{i\in\mathcal{S}} \mathcal{P}_i = 0$.

We appeal to some concepts and results from [16], [19]. Let us agree to call a vertex i in a directed graph $\mathbb{G}$ a sink of $\mathbb{G}$ if for any other vertex j of $\mathbb{G}$, there is a directed path from vertex j to vertex i. We say that $\mathbb{G}$ is sunk at i if i is in fact a sink, and that $\mathbb{G}$ is strongly sunk at i if i is reachable from each other vertex of $\mathbb{G}$ along a directed path of length one, i.e., any other vertex is a neighbor of vertex i. A directed graph is called a sunk graph if it possesses at least one sink, and a strongly sunk graph if it has at least one vertex at which it is strongly sunk.

The composition of two directed graphs $\mathbb{G}_p$, $\mathbb{G}_q$ with the same vertex set, denoted by $\mathbb{G}_q \circ \mathbb{G}_p$, is the directed graph with the same vertex set and arc set defined so that (i,j) is an arc in the composition whenever there is a vertex k such that (i,k) is an arc in $\mathbb{G}_p$ and (k,j) is an arc in $\mathbb{G}_q$. Since this composition is an associative binary operation, the definition extends unambiguously to any finite sequence of directed graphs with the same vertex set. Graph composition and matrix multiplication are closely related in that $\gamma(M_2M_1) = \gamma(M_2) \circ \gamma(M_1)$. For graphs with self-arcs at all vertices, it is easy to see that the arcs of both $\mathbb{G}_p$ and $\mathbb{G}_q$ are arcs of $\mathbb{G}_q \circ \mathbb{G}_p$.

Lemma 5: [16, Lemma 5] Let $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \dots, \mathbb{G}_{p_k}$ be a finite sequence of m-vertex directed graphs with self-arcs which are all sunk at v. If $k \leq m-1$, then v has at least $k+1$ neighbors in $\mathbb{G}_{p_k} \circ \mathbb{G}_{p_{k-1}} \circ \cdots \circ \mathbb{G}_{p_1}$. If $k \geq m-1$, then $\mathbb{G}_{p_k} \circ \mathbb{G}_{p_{k-1}} \circ \cdots \circ \mathbb{G}_{p_1}$ is strongly sunk at v.

Define the neighbor function of a directed graph $\mathbb{G}$ with vertex set $\mathcal{V}$, denoted by $\alpha(\mathbb{G}, \cdot)$, as the $2^{\mathcal{V}} \to 2^{\mathcal{V}}$ function which assigns to each subset $\mathcal{S} \subset \mathcal{V}$, the subset of vertices in $\mathcal{V}$ which are neighbors of $\mathcal{S}$ in $\mathbb{G}$. For any $v \in \mathcal{V}$, there is a unique largest subgraph sunk at v, namely the graph induced by the vertex set $\mathcal{V}(v) = \{v\} \cup \alpha(\mathbb{G}, v) \cup \cdots \cup \alpha^{|\mathcal{V}|-1}(\mathbb{G}, v)$, where $\alpha^i(\mathbb{G}, \cdot)$ denotes the composition of $\alpha(\mathbb{G}, \cdot)$ with itself i times. We call this induced graph the sunk graph generated by v. The sunk graph generated by any vertex of each $\gamma(W(t))$, $t \geq 0$ has the following property.

Lemma 6: If $\mathbb{G}$ is $(\beta, d\beta)$-resilient and the n-agent network is $(n - \kappa_{\beta,d\beta}(\mathbb{G}))$-redundant, then for any time $t \geq 0$ and each vertex v of $\gamma(W(t))$, there holds $\bigcap_{i \in \mathcal{V}(v)} \mathcal{P}_i = 0$.

Proof of Lemma 6: Since $\mathcal{V}(v)$ is the vertex set of the sunk graph generated by v, it follows that $\mathcal{V}(v)$ contains all roots of $\gamma(W(t))$ whose number is at least $\kappa_{\beta,d\beta}(\mathbb{G})$. From Lemma 4, $\bigcap_{i \in \mathcal{V}(v)} \mathcal{P}_i = 0$. ∎

We also make use of the following concepts and results from [1]. Define a route over a given sequence of directed graphs $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_q$ with the same vertex set as a sequence of vertices $i_0, i_1, \dots, i_q$ such that (i_{k-1}, i_k) is an arc in $\mathbb{G}_k$ for all $k \in \{1, 2, \dots, q\}$. A route over a sequence of graphs which are all the same directed graph $\mathbb{G}$, is thus a directed walk in $\mathbb{G}$. The definition implies that if $i_0, i_1, \dots, i_q$ is a route over $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_q$ and $i_q, i_{q+1}, \dots, i_p$ is a route over $\mathbb{G}_q, \mathbb{G}_{q+1}, \dots, \mathbb{G}_p$, then the concatenated sequence $i_0, i_1, \dots, i_{q-1}, i_q, i_{q+1}, \dots, i_p$ is a route over $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_{q-1}, \mathbb{G}_q, \mathbb{G}_{q+1}, \dots, \mathbb{G}_p$. This fact remains true if more than two sequences are concatenated.

More can be said if we focus exclusively on graphs with self-arcs. If $i = i_0, i_1, \dots, i_q = j$ is a route over a sequence $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_q$, then (i,j) must be an arc in the composed graph $\mathbb{G}_q \circ \mathbb{G}_{q-1} \circ \cdots \circ \mathbb{G}_1$. The converse is also true, namely that if (i,j) is an arc in $\mathbb{G}_q \circ \mathbb{G}_{q-1} \circ \cdots \circ \mathbb{G}_1$, then there must exist vertices $i_1, \dots, i_{q-1}$ for which $i = i_0, i_1, \dots, i_q = j$ is a route over $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_q$. Moreover, if $\mathbb{G}_{\tau_1}, \mathbb{G}_{\tau_2}, \dots, \mathbb{G}_{\tau_p}$ is a subsequence of $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_q$ with $p \leq q$ and $i_0, i_1, \dots, i_p$ being a route over $\mathbb{G}_{\tau_1}, \mathbb{G}_{\tau_2}, \dots, \mathbb{G}_{\tau_p}$, then there must exist a route over $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_q$ which contains $i_0, i_1, \dots, i_p$ as a subsequence. An important relation between routes and matrix multiplication is as follows.

Lemma 7: Let $S_1, S_2, \dots S_q$ be a sequence of $m \times m$ stochastic matrices with positive diagonal entries whose graphs are respectively $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_q$. If $j = i_0, i_1, \dots, i_q = i$ is a route over $\mathbb{G}_1, \mathbb{G}_2, \dots, \mathbb{G}_q$, then the matrix product $P_{i_q} \cdots P_{i_2} P_{i_1}$ is a component of the ijth block entry of $P(S_q \otimes I) \cdots P(S_2 \otimes I) P(S_1 \otimes I)$.

Lemma 7 can be proved using the same argument as in the proof of Lemma 4 in [1].

It is easy to see from (9) that analysis of the algorithm under study involves the matrix product $\cdots P(W(t) \otimes I) \cdots P(W(1) \otimes I) P(W(0) \otimes I)$ in which each $W(t)$, $t \geq 0$ is a $|\mathcal{H}| \times |\mathcal{H}|$ stochastic matrix with positive diagonal entries. Such a matrix product is a $d|\mathcal{H}| \times d|\mathcal{H}|$ block matrix whose each block is a projection matrix polynomial of the form

$$\mu(P_1, P_2, \dots, P_{|\mathcal{H}|}) = \sum_{i=1}^{b} \lambda_i P_{h_i(1)} P_{h_i(2)} \cdots P_{h_i(q_i)},$$

where q_i and b are positive integers, λ_i is a real positive number, and each $h_i(j)$, $j \in \{1, 2, \dots, q_i\}$ is an integer in $\{1, 2, \dots, |\mathcal{H}|\}$.

To study these block matrices, we need the following "mixed matrix norm" introduced in [1]. Let $\|\cdot\|_\infty$ denote the induced infinity norm and write $\mathbb{R}^{dm \times dm}$ for the vector space of all $m \times m$ block matrices $Q = [Q_{ij}]$ whose ijth block entry is a matrix $Q_{ij} \in \mathbb{R}^{d \times d}$. The mixed matrix norm of $Q \in \mathbb{R}^{dm \times dm}$, written $\|Q\|$, is defined as $\|Q\| = \|\langle Q \rangle\|_\infty$, where $\langle Q \rangle$ is the matrix in $\mathbb{R}^{m \times m}$ whose ijth entry is $\|Q_{ij}\|_2$. It has been shown in [1, Lemma 3] that the mixed matrix norm is sub-multiplicative. More can be said.

Let us agree to say that a projection matrix polynomial $\mu(P_1, P_2, \dots, P_{|\mathcal{H}|})$ is complete if it has a component $P_{h_i(1)} P_{h_i(2)} \cdots P_{h_i(q_i)}$ such that $\bigcap_{k=1}^{q_i} \mathcal{P}_{h_i(k)} = 0$. Such a complete component has the property that $\|P_{h_i(1)} P_{h_i(2)} \cdots P_{h_i(q_i)}\|_2 < 1$ [1, Lemma 2]. This property leads to a contraction condition for block matrices in $\mathbb{R}^{d|\mathcal{H}| \times d|\mathcal{H}|}$ whose blocks are projection matrix polynomials.

Lemma 8: Let $S_1, S_2, \dots S_q$ be a finite sequence of $|\mathcal{H}| \times |\mathcal{H}|$ stochastic matrices with positive diagonal entries and $M = P(S_q \otimes I) \cdots P(S_2 \otimes I) P(S_1 \otimes I)$. If at least one entry in each block row of M is complete, then M is a contraction in the mixed matrix norm, i.e., $\|M\| < 1$.

The lemma is a direct consequence of Proposition 1 in [1].

Proposition 2: Suppose that (11) holds. If $\mathbb{G}$ is $(\beta, d\beta)$-resilient and the n-agent network is $(n - \kappa_{\beta,d\beta}(\mathbb{G}))$-redundant, then there is a finite positive integer τ such that for any $p \geq \tau$ and $t \geq 0$, the matrix $P(W(t+p) \otimes I) \cdots P(W(t+1) \otimes I) P(W(t) \otimes I)$ is a contraction in the mixed matrix norm.

The following theorem establishes the correctness of the algorithm for the unique least squares solution case.

Theorem 2: Suppose that $Ax = b$ has a unique least squares solution. If $\mathbb{G}$ is $(\beta, d\beta)$-resilient and the n-agent network is $(n - \kappa_{\beta,d\beta}(\mathbb{G}))$-redundant, then there exists a nonnegative constant $\lambda < 1$ for which all $x_i(t)$, $i \in \mathcal{H}$ converge to the least squares solution as $t \to \infty$ as fast as λ^t converges to zero.

We next consider the case when $Ax = b$ has multiple least squares solutions, using the subspace "quotient out" technique from [1], and begin with the following lemma.

Lemma 9: Let Q' be any matrix whose columns form an orthonormal basis for the orthogonal complement of $\bigcap_{i\in\mathcal{H}} \mathcal{P}_i$ and define $\bar{P}_i = QP_iQ'$ for each $i \in \mathcal{H}$. Then,

1) Each $\bar{P}_i$, $i \in \mathcal{H}$ is an orthogonal projection matrix;
2) Each $\bar{P}_i$, $i \in \mathcal{H}$ satisfies $QP_i = \bar{P}_iQ$;
3) For any nonempty subset $\mathcal{E} \subset \mathcal{H}$, there holds $\bigcap_{i\in\mathcal{E}} \bar{\mathcal{P}}_i = 0$ if and only if $\bigcap_{i\in\mathcal{E}} \mathcal{P}_i = \bigcap_{i\in\mathcal{H}} \mathcal{P}_i$.

The lemma is a direct consequence of Lemma 7 in [16].

Lemma 10: If $\mathbb{G}$ is $(\beta, d\beta)$-resilient and the n-agent network is $(n - \kappa_{\beta,d\beta}(\mathbb{G}))$-redundant, then for any time $t \geq 0$ and each vertex v of $\gamma(W(t))$, there holds $\bigcap_{i\in\mathcal{V}(v)} \bar{\mathcal{P}}_i = 0$.

Proof of Lemma 10: Since $\mathcal{V}(v)$ is the vertex set of the sunk graph generated by v, it follows that $\mathcal{V}(v)$ contains all roots of $\gamma(W(t))$ whose number is at least $\kappa_{\beta,d\beta}(\mathbb{G})$. From Corollary 1, $\arg\min_x \sum_{i\in\mathcal{V}(v)} \|A_ix - b_i\|_2^2 = \mathcal{X}^*$. This fact and Proposition 1 imply that $\bigcap_{i\in\mathcal{V}(v)} \mathcal{P}_i = \bigcap_{i\in\mathcal{H}} \mathcal{P}_i$. From property 3) of Lemma 9, $\bigcap_{i\in\mathcal{V}(v)} \bar{\mathcal{P}}_i = 0$. ■

Theorem 3: Suppose that $Ax = b$ has more than one least squares solution. If $\mathbb{G}$ is $(\beta, d\beta)$-resilient and the n-agent network is $(n - \kappa_{\beta,d\beta}(\mathbb{G}))$-redundant, then there exists a nonnegative constant $\lambda < 1$ for which all $x_i(t)$, $i \in \mathcal{H}$ converge to the same least squares solution as $t \to \infty$ as fast as λ^t converges to zero.

IV. Conclusion

This paper has proposed a distributed least squares algorithm for solving a system of linear algebraic equations over a fixed multi-agent network, which converges exponentially fast and achieves full resilience in the presence of Byzantine agents provided appropriate redundancy in both graph connectivity and objective functions is established. The proposed algorithm and its convergence results can be easily extended to non-stationary networks provided that the time-varying neighbor graphs are always $(\beta, d\beta)$-resilient. Since the algorithm borrows the same design ideas from a recent resilient distributed optimization algorithm [8], it "inherits" the same limitations from the algorithm there (see discussions in Section 5 of [8]). A particular limitation is that it can hardly cope with high-dimensional cases.

An important future direction is thus to tackle the challenging high-dimensional issue. Although it is a natural idea to appeal to communication-efficient schemes in which each agent only needs to transmit low-dimensional signals (e.g., entry- or block-wise updating [20], [21]), our limited simulations indicate that simply partitioning high-dimensional state vectors into low-dimensional blocks in communication and computation is not promising, if without any additional design. Other possible approaches include exploiting consensus fusion with reduced information [22], leveraging dimension-independent filtering [13], [23], and combining these techniques together.

References

[1] S. Mou, J. Liu, and A.S. Morse. A distributed algorithm for solving a linear algebraic equation. *IEEE Transactions on Automatic Control*, 60(11):2863–2878, 2015.

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

[3] X. Wang, J. Zhou, S. Mou, and M.J. Corless. A distributed algorithm for least squares solutions. *IEEE Transactions on Automatic Control*, 64(10):4217–4222, 2019.

[4] Y. Liu, Y. Lou, B.D.O. Anderson, and G. Shi. Network flows that solve least squares for linear equations. *Automatica*, 120(09108), 2020.

[5] Y. Huang and Z. Meng. Distributed algorithms for the least square solution of linear equations. 2021. arXiv:2105.09298 [math.NA].

[6] A. Nedić and J. Liu. Distributed optimization for control. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:77–103, 2018.

[7] T. Yang, X. Yi, J. Wu, Y. Yuan, D. Wu, Z. Meng, Y. Hong, H. Wang, Z. Lin, and K.H. Johansson. A survey of distributed optimization. *Annual Reviews in Control*, 47:278–305, 2019.

[8] J. Zhu, Y. Lin, A. Velasquez, and J. Liu. Resilient distributed optimization. In *Proceedings of the 2023 American Control Conference*, pages 1307–1312, 2023.

[9] A. Nedić, A. Ozdaglar, and P.A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

[10] A. Nedić and J. Liu. On convergence rate of weighted-averaging dynamics for consensus problems. *IEEE Transactions on Automatic Control*, 62(2):766–781, 2017.

[11] N. Gupta and N.H. Vaidya. Resilience in collaborative optimization: Redundant and independent cost functions. 2020. arXiv:2003.09675v2 [cs.DC].

[12] N.H. Vaidya, L. Tseng, and G. Liang. Iterative approximate Byzantine consensus in arbitrary directed graphs. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 365–374, 2012.

[13] J. Zhu, Y. Lin, A. Velasquez, and J. Liu. Resilient constrained consensus over complete graphs via feasibility redundancy. In *Proceedings of the 2022 American Control Conference*, pages 3418–3422, 2022.

[14] X. Wang, S. Mou, and S. Sundaram. Resilience for distributed consensus with constraints. 2022. arXiv:2206.05662 [eess.SY].

[15] P. Lin and W. Ren. Constrained consensus in unbalanced networks with communication delays. *IEEE Transactions on Automatic Control*, 59(3):775–781, 2014.

[16] J. Liu, A.S. Morse, A. Nedić, and T. Başar. Exponential convergence of a distributed algorithm for solving linear algebraic equations. *Automatica*, 83:37–46, 2017. Full version is available at arXiv:1701.00554v2 [math.OC].

[17] J. Liu, S. Mou, and A.S. Morse. Asynchronous distributed algorithms for solving linear algebraic equations. *IEEE Transactions on Automatic Control*, 63(2):372–385, 2018.

[18] M. James. The generalised inverse. *The Mathematical Gazette*, 62(420):109–114, 1978.

[19] M. Cao, A.S. Morse, and B.D.O. Anderson. Reaching a consensus in a dynamically changing environment: A graphical approach. *SIAM Journal on Control and Optimization*, 47(2):575–600, 2008.

[20] J. Liu and B.D.O. Anderson. Communication-efficient distributed algorithms for solving linear algebraic equations over directed graphs. In *Proceedings of the 59th IEEE Conference on Decision and Control*, pages 5360–5365, 2020.

[21] I. Notarnicola, Y. Sun, G. Scutari, and G. Notarstefano. Distributed big-data optimization via blockwise gradient tracking. *IEEE Transactions on Automatic Control*, 66(5):2045–2060, 2021.

[22] J. Zhu, Y. Lin, J. Liu, and A.S. Morse. Reaching a consensus with limited information. *Systems & Control Letters*, 176:105524, 2023. A conference version also appears in Proceedings of the 62nd IEEE Conference on Decision and Control.

[23] N. Gupta, T.T. Doan, and N.H. Vaidya. Byzantine fault-tolerance in decentralized optimization under $2f$-redundancy. In *Proceedings of the 2021 American Control Conference*, pages 3632–3637, 2021.