Distributed and Communication-efficient Solutions to Linear Equations with Special Sparse Structure

Peng Wang[a], Yuanqi Gao[b], Nanpeng Yu[b], Wei Ren[b], Jianming Lian[c], Di Wu[a]

[a] *Pacific Northwest National Laboratory, 902 Battelle Blvd, Richland, 99354, WA, USA*
[b] *University of California, Riverside, 900 University Ave, Riverside, 92521, CA, USA*
[c] *Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge, 37831, TN, USA*

Abstract

In this paper, two distributed and communication-efficient algorithms based on the multi-agent system are proposed to solve a system of linear equations with the Laplacian sparse system matrix. One algorithm is based on the gradient descent method in optimization. In this algorithm, the agents only share partial information instead of all of their collective state vectors to save significant communication. The other algorithm is obtained by approximating Newton's method for a faster convergence rate. Although it requires twice as much communication as the first one, it is still communication-efficient given the low dimension of the information shared among agents. The convergence at a linear rate is proved for both algorithms, and a comprehensive comparison of their convergence rate, communication burden, and computation costs is also performed. The proposed algorithms can be applied to various systems to solve those problems that can be modeled as a system of linear equations with a Laplacian sparse system matrix. Simulation results with the electric power system illustrate their effectiveness.

Keywords: distributed method, multi-agent system, linear equations, power system

1. INTRODUCTION

Solving a system of linear equations, denoted by $Ax = b$ mathematically, is one of the most fundamental problems in many research fields. With the emergence of Internet of Things, an increasing amount of sensors and actuators are being integrated into the networked systems around us. Hence,

distributed methods to solve a system of linear equations are attracting more attention from researchers.

Many distributed algorithms to solve $Ax = b$ are proposed in the literature, e.g. [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]. These algorithms assume that each agent knows some rows of the augmented matrix $\begin{pmatrix} A & b \end{pmatrix}$. The algorithms in [14, 15, 16, 17, 18, 19] are continuous-time ones while those in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] are discrete-time ones. In this paper, we focus on discrete-time distributed algorithms to solve $Ax = b$. In [1], a distributed algorithm is proposed for both synchronous and asynchronous updates under repeatedly jointly strongly connected graphs, which requires locally feasible initialization. The convergence of the algorithm at a linear rate is also proved in [1]. Ref. [10] extends the results in [1] by considering the influence of communication and computation delays and arbitrary initialization. In [2], distributed algorithms are proposed to find the minimum norm solution to a system of linear equations associated with weighted inner products. Ref. [9] then broadens the results in [1] and [2], allowing arbitrary initialization for convergence to a general solution. It also shows that with special initialization, the distributed algorithms in [9] can converge to a solution to $Ax = b$ that is closest to a given point. When $Ax = b$ has a unique solution, a distributed algorithm is designed in [3] to allow arbitrary initialization with the feedback of the deviation from local systems of linear equations. The algorithm is proved to converge at a linear rate. Also, when $Ax = b$ has a unique solution, a distributed algorithm is proposed in [8] using the subgradient method and the linear convergence rate is proved. In [5, 6, 7], a distributed algorithm that converges in finite time is also proposed to solve $Ax = b$. The algorithm requires agents to share the information of kernels of local equations with their neighbors, which may lead to non-robustness. A distributed algorithm to solve $Ax = b$ is proposed in [4] using M-Fejer mappings and the convergence rates for two special cases are also specified.

In the literature mentioned above, the agents need to share their estimates of every entry of the vector x. However, in many applications in which matrix A is sparse and the system is large-scale, the distributed algorithms in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] will lead to significant communication overhead. In this case, communication-efficient distributed algorithms to solve $Ax = b$ are necessary. Communication-efficient distributed algorithms to solve $Ax = b$ are designed for time-varying undirected graphs in [12] in which each agent broadcasts the entries of its estimate in a cyclic manner and complete send-

ing the whole vector of its estimates in multiple rounds of communication. The algorithms in [12] reduce the communication burden in each round of communication, but the total communication burden does not change. A communication-efficient distributed algorithm is proposed in [11] for general sparse matrices, but the algorithm in [11] requires the information of common nonzero parts of agents' rows and their neighbors' columns. When matrix A is Laplacian sparse, the distributed algorithm in [11] requires sharing information of agents' common neighbors, which might not be available to the agents. The algorithms in [12, 11] are extended to directed graphs in [13].

In this paper, we develop communication-efficient distributed algorithms to solve $Ax = b$ with Laplacian sparse A based on multi-agent systems. Matrices with Laplacian sparse structure can be found in many problems such as network flow problems in various kinds of systems. In particular, the power flow problem in a power system involves network matrices with Laplacian sparse property. In our proposed communication-efficient distributed algorithms, an agent only transmits the parts of x related to itself and its neighbors, instead of every entry of x as in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] or the state of their common neighbors as in [11]. In the first proposed algorithm, only two parts of the vector x are transmitted through each communication link while in the second proposed algorithm, two parts of x and the corresponding parts of the gradient vector of the system are transmitted. As each part of x and that of the gradient vector are low dimensional, both algorithms significantly reduce the communication burden. We propose the first communication-efficient distributed algorithm to solve $Ax = b$ with Laplacian sparse A based on a gradient descent method and prove its convergence at a linear rate. Then, we propose an accelerated communication-efficient distributed algorithm based on an approximation to Newton's method. The algorithm based on the approximated Newton's method requires twice as much communication as the one based on the gradient method. As the information communicated between agents is low dimensional, the approximated Newton-based algorithm is still communication-efficient.

A preliminary version of this paper was presented in [20]. Compared with [20], Theorem 2 is extended to a more general case in which $Ax = b$ has multiple solutions. Such an extension requires a totally different proof from the one shown in [20]. In addition, the detailed proof of Lemma 3, which is the key to Theorem 1, is presented in this paper. Simulation examples are also added in this paper to illustrate the efficacy of the proposed algorithms.

The remaining of the paper is organized as follows. In Section 2, the pre-

liminaries on graph theory, Laplacian sparse matrix, and power flow problems are provided. In Section 4, the two communication-efficient distributed algorithms are proposed to solve $Ax = b$ with Laplacian sparse A with a detailed comparison. Simulation results are presented in Section 6 to illustrate their effectiveness. Finally, the conclusions can be found in Section 7.

2. Preliminaries

In this section, we provide some preliminaries on the graph theory, which is necessary for distributed algorithms, Laplacian sparse matrix, which is our research focus in this paper, and power flow problem, which serve as the practical application of our research topic.

2.1. Graph Theory

An M-th order undirected graph, denoted by $\mathcal{G}(V, E)$, is composed of a vertex set $V = \{1, \cdots, M\}$ and an edge set $E \subseteq V \times V$. We use the pair (j, i) to denote the edge between the j-th vertex and i-th vertex. We suppose that $(i, i) \notin E$, $\forall i \in V$. We say that j is a neighbor of i if there is an edge between i and j. The neighbor set N_i of vertex i is composed of the neighbors of the i-th vertex, i.e., $N_i = \{j : (j, i) \in E\}$. The number of the i-th vertex's neighbors is denoted by $|N_i|$. The Laplacian matrix $L = [l_{ij}]_{M\times M} \in \mathbb{R}^{M\times M}$ associated with the graph $\mathcal{G}$ is defined such that

$$l_{ij} = \begin{cases} 1, & j \in N_i \\ -|N_i|, & j = i; \\ 0, & \text{otherwise.} \end{cases}$$

A path between i and j is a sequence of edges (i, i_1), (i_1, i_2), $\cdots$, (i_p, j). An undirected graph is connected if, for every pair of vertices i and j $(i \neq j)$, there is a path between them.

2.2. Laplacian Sparse Matrix

In this work, we assume that matrix A has the following special sparse structure.

Definition 1 (Laplacian sparse matrix)**.** *A matrix A has the Laplacian sparse structure of an undirected graph $\mathcal{G}$ if $a_{ij} \neq 0$ only if i and j are neighbors in $\mathcal{G}$ or $i = j$, i.e., $l_{ij} \neq 0$, where a_{ij} is the (i, j)-th entry of the*

matrix A. A block matrix A has the Laplacian sparse structure of a graph $\mathcal{G}$ if A_{ij} is a nonzero matrix only if i and j are neighbors in $\mathcal{G}$ or $i = j$, i.e., $l_{ij} \neq 0$, where A_{ij} is the (i, j)-th block of matrix A.

Remark 1. *In many engineering systems, the graph $\mathcal{G}$ emerges from the communication network while matrix A is dependent on the network of physical connection, e.g., connection by electricity wires in a power system. In Definition 1, when l_{ij} is zero, a_{ij} (or A_{ij}) must be zero; but when l_{ij} is nonzero, a_{ij} (or A_{ij}) may be zero or nonzero. This allows us to apply the results in this paper to some problems in which the communication network differs from the physical network. For example, in a power system, the communication network may consist of regional centers which are in charge of many buses in a region while the physical network is composed of each bus. The Laplacian of the communication network does not have the same sparse structure as the power network. However, if there exists a communication link between regional centers that have physically connected buses, the results in this paper can be applied.*

A matrix that has the Laplacian sparse structure of a graph $\mathcal{G}$ is also called a Laplacian sparse matrix for simplicity if the graph $\mathcal{G}$ is clear from the context.

2.3. Power Flow Problem

Laplacian sparse matrices are common in many problems, e.g., power flow problems. In power flow problems, the nodal admittance matrix of a power system has a Laplacian sparse structure if the communication topology is the same as the physical one. Also, the Jacobian of the power flow equations, though more complex, can be regarded as a Laplacian sparse matrix.

The power flow problem is very fundamental in the steady-state analysis of electrical power systems. The power flow problem is typically formulated as solving a system of nonlinear equations, known as the power flow equations. Laplacian sparse matrices emerge in numerical solutions to power flow equations, e.g., the Newton-Raphson method. We will give a brief introduction to single-phase power flow problems and the Newton-Raphson method. A comprehensive description of power flow problems and the Newton-Raphson method can be found in [21].

Consider an electrical network modeled by a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is a set of nodes, $\mathcal{E}$ a set of links representing transmission/distribution

lines, and $\mathcal{W}$ a set of weights associated with $\mathcal{E}$. The value of $w \in \mathcal{W}$ depends on the electrical characteristics of the link, e.g., the impedance of the conductor.

Let v_i be the nodal voltage of node i, s_i be the net complex power injection at node i, and Y be the nodal admittance matrix. Notice that Y has the same sparse structure as the Laplacian matrix of $\mathcal{G}$ and is thus a Laplacian sparse matrix.

Let $v_i = |v_i|e^{\jmath\theta_i}$, $s_i = p_i + \jmath q_i$, and $Y = G + \jmath B$, where $\jmath$ is the imaginary unit, $|v_i|$ is the nodal voltage magnitude, θ_i is nodal voltage angle, p_i is net active power injection, q_i is the net reactive power injection, G is the conductance matrix, and B is the susceptance matrix. Let

$$\begin{aligned} p_i(x) &= \textstyle\sum_{k=1}^{|\mathcal{V}|} |v_i||v_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}), \\ q_i(x) &= \textstyle\sum_{k=1}^{|\mathcal{V}|} |v_i||v_k|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}), \end{aligned} \tag{1}$$

where G_{ik} and B_{ik} are the (i,k)-th entry of matrices G and B, respectively, $\theta_{ik} = \theta_i - \theta_k$ is the nodal voltage angle difference between nodes i and k, and $x = (|v_2|, |v_3|, \cdots, |v_{|\mathcal{V}|}|, |\theta_2|, |\theta_3|, \cdots, |\theta_{|\mathcal{V}|}|)^T$. As matrix Y has the Laplacian sparse structure of $\mathcal{G}$, so do G and B. Thus, if nodes i and k are not connected, G_{ik} and B_{ik} are both zero. The power flow problems can be solved through solving the following equations

$$\begin{aligned} p_i(x) - p_i = 0, i = 2, 3, \cdots, |\mathcal{V}|, \\ q_i(x) - q_i = 0, i = 2, 3, \cdots, |\mathcal{V}|. \end{aligned} \tag{2}$$

In the Newton-Raphson method, (2) is iteratively solved. In each iterative step, we need to solve a system of linear equations as follows:

$$-J\Delta x = y \tag{3}$$

where J is the Jacobian matrix of the functions in (2), y is a constant related to (1) in each step.

The Jacobian matrix can be partitioned as

$$J = \begin{pmatrix} J_{p\theta} & J_{p|v|} \\ J_{q\theta} & J_{q|v|} \end{pmatrix}, \tag{4}$$

where

$$J_{p\theta} = \frac{\partial p(x)}{\partial \theta}, J_{p|v|} = \frac{\partial p(x)}{\partial |v|}, J_{q\theta} = \frac{\partial q(x)}{\partial \theta}, J_{q|v|} = \frac{\partial q(x)}{\partial |v|}.$$

When G_{ik} and B_{ik} are zero, the (i,k)-th entries of $J_{p\theta}$, $J_{p|v|}$, $J_{q\theta}$, and $J_{q|v|}$ are zero, and thus they have the same sparse structure as matrices G or B. As G and B have the Laplacian sparse structure of $\mathcal{G}$, $J_{p\theta}, J_{p|v|}, J_{q\theta}, J_{q|v|}$ also have the Laplacian sparse structure of $\mathcal{G}$.

Remark 2. *Each entry of J can be computed locally. In the first block $J_{p\theta}$, its entries in the (i,j)-th sub-block depend only on the voltage magnitudes and phase angles of the i-th agent and the j-th agent, the conductance G_{ij} and the susceptance B_{ij} between them, which are all local information. The entries in other blocks $J_{p|v|}, J_{q\theta}, J_{q|v|}$ can similarly be computed with local information. Thus, all entries of J can be computed locally.*

3. Problem Formulation

Suppose that we have a group of M agents that form the vertex set V and communication links between the agents that form the edges between the vertices. The agent network can be represented by a graph $\mathcal{G}(V,E)$. For graph $\mathcal{G}$, we have the following assumption:

Assumption 1. *The communication topology of the agent network is fixed, undirected, and connected.*

Remark 3. *Fixed, undirected, and connected graphs exist in many engineering problems. For example, in a smart grid, the sensors to measure the electrical quantities, e.g., voltages, are installed in fixed locations. The communication channel between each pair of sensors is bidirectional, i.e., undirected. The communication topology of the sensor network will be fixed, undirected, and connected.*

Assumption 2. *The system matrix A has the Laplacian sparse structure of the communication graph of the agent network.*

Each agent knows some rows $A_i,\ i \in V$ of A and the corresponding entries b_i in b. All the agents work together to obtain a solution to $Ax = b$. Matrix A is assumed to have a Laplacian sparse structure of $\mathcal{G}$ as in Definition 1.

Let $A^{(i)}$ be the nonzero part in A_i, $x^{(i)}$ the corresponding parts in x, and $L^{(i)}$ the nonzero parts of L_i, where L_i is the i-th row of the Laplacian matrix L. Then the local equations $A_i x = b_i$ is equivalent to $A^{(i)}x^{(i)} = b_i$.

For clarity of notation, let $A_j^{(i)}$ denote the part (i.e., entry or block) of $A^{(i)}$ which corresponds to the j-th part of A_i, $x_j^{(i)}$ denote the part of $x^{(i)}$ which corresponds to the j-th part of x. In the rest part of the paper, $x_i^{(i)}$ is referred to as the i-th agent's state, $x^{(i)}$ as the i-th agent's augmented state, $x_j^{(i)}$ as the i-th agent's estimate on j-th agent's state, and x as the collective state.

Example 1. *Consider a network $\mathcal{G}$ composed of four agents with the following Laplacian matrix,*

$$L = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix},$$

and the following matrix A with Laplacian sparse structure of $\mathcal{G}$,

$$A = \begin{pmatrix} 1 & 3.4 & 0 & 0 \\ 0.8 & 5 & -9 & 0 \\ 0 & -6.23 & -3 & 6 \\ 0 & 0 & -5 & -0.96 \end{pmatrix}.$$

If each agent owns one row of A, then $A^{(1)} = (1\ 3.4)$, $A^{(2)} = (0.8\ 5\ -9)$, $A^{(3)} = (-6.23\ -36)$, $A^{(4)} = (-5\ -0.96)$, $A_1^{(1)} = 1$, and $A_2^{(1)} = 3.4$. Correspondingly, $x^{(1)} = (x_1\ x_2)^T$, $x^{(2)} = (x_1\ x_2\ x_3)^T$, $x^{(3)} = (x_2\ x_3\ x_4)^T$, and $x^{(4)} = (x_3\ x_4)^T$, $x_2^{(3)} = x_2$, $x_3^{(3)} = x_3$, and $x_4^{(3)} = x_4$. $L^{(1)} = (-1\ 1)$, $L^{(2)} = (1\ -2\ 1)$, $L^{(3)} = (1\ -2\ 1)$, and $L^{(4)} = (1\ -1)$.

If solving $Ax = b$ is formulated as the following distributed optimization problem,

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{i\in V} \|A_i x_i - b_i\|^2 \\ \text{subject to} \quad & x_i = x_j,\ \forall i, j \in V, \end{aligned}$$

then, under Assumption 1 and the Laplacian sparse structure of A, it is equivalent to solving

$$\begin{aligned} \text{minimize} \quad & f = \frac{1}{2}\sum_{i\in V} \|A^{(i)} x^{(i)} - b_i\|^2 \\ \text{subject to} \quad & x_i^{(i)} = x_i^{(j)},\ x_j^{(j)} = x_j^{(i)},\ (i,j) \in E, \end{aligned} \tag{5}$$

which can be easily proved as follows. If $x^\star$ is a solution to $Ax = b$, it is obvious that $x^{(i)} = \left(x_j^\star\right)_{j\in N_i\cup\{i\}}$, $\forall i \in V$, is also an optimal solution to (5). If $x^{(1)}$, $x^{(2)}, \cdots, x^{(M)}$ form a solution to (5), denoted $x^\star = \left(x_1^{(1)T}\ x_2^{(2)T}\ \cdots\ x_M^{(M)T}\right)^T$, for any the i-th agent, we have $x_j^\star = x_j^{(j)} = x_j^{(i)}$, $\forall j \in N_i$. So, $x^{(i)} = \left(x_j^\star\right)_{j\in N_i\cup\{i\}}$. Notice that A has the Laplacian sparse structure. Thus, we have that $A_i x^\star = A^{(i)}x^{(i)} = b_i$.

We can then transfer the constrained optimization problem in (5) to an unconstrained optimization problem with penalty functions as follows. Let

$$\begin{aligned} f_p =& \frac{1}{2}\left(\sum_{i\in V}\left\|A^{(i)}x^{(i)} - b_i\right\|^2\right. \\ &\left. + \sum_{(i,j)\in E}\left(\left\|x_i^{(i)} - x_i^{(j)}\right\|^2 + \left\|x_j^{(j)} - x_j^{(i)}\right\|^2\right)\right). \end{aligned} \tag{6}$$

If $Ax = b$ has solutions, (5) is equivalent to the following,

$$\text{minimize} \quad f_p. \tag{7}$$

If $Ax = b$ has solutions, the solutions to (5) and (7) are those satisfying that $A^{(i)}x^{(i)} = b_i$, $\forall i \in V$ and $x_i^{(i)} = x_i^{(j)}$, $x_j^{(j)} = x_j^{(i)}$, $\forall (i,j) \in E$. So (5) is equivalent to (7).

Remark 4. *In many applications in which $Ax = b$ does not have a solution, we may be interested in finding a least-squares solution. We can add a weight to the consensus item in (6) to obtain an approximated least-squares solution. Or we may explore other formulations for an exact least-squares solution.*

4. Communication-efficient Algorithms

In the following, we will propose two communication-efficient distributed algorithms to solve $Ax = b$ by solving (7). The first algorithm is based on the gradient descent method while the second one on an approximated Newton's method, which converges faster than the first one. The first algorithm only requires communication of two parts of the vector x through each communication link while the second algorithm also requires the communication of the

two corresponding parts of the gradient vector. For simplicity, it is supposed that $\dim\left(x_i^{(i)}\right) = 1,\ \forall i \in V$, i.e., the state of every agent is a scalar, where $\dim(\cdot)$ represents the dimension. It is not difficult to extend the results to the case in which different agents have states of different dimensions.

4.1. Algorithm Based on Gradient Descent Method

In this subsection, a communication-efficient distributed algorithm to solve $Ax = b$ is developed based on the gradient descent method with constant step size, and its convergence at a linear rate[1] is proved. First, the gradient descent method with constant step size α is applied to (7). For the gradient of f_p, we have that

$$\begin{aligned}\frac{\partial f_p}{\partial x^{(i)}} = & A^{(i)T}\left(A^{(i)}x^{(i)} - b_i\right) \\ & + \sum_{j\in N_i}\left(\left(x_i^{(i)} - x_i^{(j)}\right)e_i^{(i)} + \left(x_j^{(i)} - x_j^{(j)}\right)e_j^{(i)}\right),\end{aligned} \tag{8}$$

where $e_j^{(i)} = (0\ \cdots\ 0\ 1\ 0\ \cdots\ 0)^T$ with the position of 1 located at the i-th agent's local index of j. In addition,

$$\dim\left(e_j^{(i)}\right) = \sum_{j\in N_i\cup\{i\}}\dim\left(x_j^{(i)}\right) = \dim\left(x^{(i)}\right).$$

Let

$$\boldsymbol{x} = \left(x^{(1)T}\ x^{(2)T}\ \cdots\ x^{(M)T}\right)^T \tag{9}$$

be a stack of all agents' augmented states and

$$\nabla f(\boldsymbol{x}) = \left(\left(\frac{\partial f_p}{\partial x^{(1)}}\right)^T\ \left(\frac{\partial f_p}{\partial x^{(2)}}\right)^T\ \cdots\ \left(\frac{\partial f_p}{\partial x^{(M)}}\right)^T\right)^T. \tag{10}$$

be a stack of the gradients. From the gradient descent method

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \alpha\nabla f(\boldsymbol{x}(k)),$$

[1]A sequence $\{y_k\}$ is said to converge to $y^\star$ at a linear rate if there exists a constant $0 < C < 1$, $a > 0$, and $N > 0$ such that $\|y(k) - y^\star\| \le aC^k, \forall k > N$, where $\|\cdot\|$ can be any norm.

it can be obtained for the i-th agent,

$$\begin{aligned} x^{(i)}(k+1) =& x^{(i)}(k) - \alpha A^{(i)T}\left(A^{(i)}x^{(i)}(k) - b_i\right) \\ & - \alpha \sum_{j\in N_i}\left(x_i^{(i)}(k) - x_i^{(j)}(k)\right)e_i^{(i)} \\ & - \alpha \sum_{j\in N_i}\left(x_j^{(i)}(k) - x_j^{(j)}(k)\right)e_j^{(i)}, \end{aligned} \tag{11}$$

where α is the step size to be selected.

Remark 5. *Compared to the algorithms in the literature [1, 2, 3, 4, 8, 9, 10], the algorithm in* (11) *can significantly reduce communication. In the algorithms to solve linear equations in [1, 2, 3, 4, 8, 9, 10], the agents need to communicate their estimates on every entry of the collective state vector x, which requires significant communication resources for large-scale systems and leads to significant communication overhead for sparse systems. In contrast, in* (11)*, the agents need to communicate their estimates on two parts of the collective state vector x no matter how large the network size is, which saves communication resources. In the example of the single-phased power flow problems introduced in Section 2.3 in which each agent is assigned with a voltage magnitude and a phase angle, the i-th agent needs to transmit to j-th agent its estimate on the voltage magnitudes and phase angles of all agents in the algorithms in [1, 2, 3, 4, 8, 9, 10] while the i-th agent only transmits to j-th agent its estimates on the voltage magnitudes and phase angles assigned to i-th and j-th agents in* (11)*, if i-th and j-th agents are neighbors. The dimension of the transmitted vector in each communication channel in the algorithms in [1, 2, 3, 4, 8, 9, 10] is (2×network size) while it is* 4 *no matter how large the network size is in* (11)*. As a result, the gradient-based algorithm in* (11) *saves communication compared to those in [1, 2, 3, 4, 8, 9, 10]. Also, compared with [11], (11) does not require sharing estimates of the states of their common neighbors under Assumption 1 while the method in [11] requires the information of the agents' common neighbors, which may not be available. As a result, (11) needs less communication between agents than the method in [11].*

For the performance of (11), we have the following result:

Theorem 1. *If the system of linear equations $Ax = b$ has solutions and A is Laplacian sparse of the agent network, $x^{(i)}(k)$, $\forall i \in V$ in (11) converges at a*

linear rate to the optimal solution set of (7) if $0 < \alpha < 2/\lambda_{\max}(\nabla^2(f_p))$*, where* $\nabla^2(f_p)$ *is the Hessian of* f_p *which is defined in (6) and* $\lambda_{\max}(\cdot)$ *represents the maximal eigenvalue. Let*

$$x^\star(k) = \left(x_1^{(1)T}(k)\ x_2^{(2)T}(k)\ \cdots\ x_M^{(M)T}(k)\right)^T \tag{12}$$

be the collection of all agents' estimates of their states. Then $x^\star(k)$ *converges to a solution to* $Ax = b$*.*

The proof of Theorem 1 can be found in the appendix.

Remark 6 (A distributed method to select step size)**.** *The upper bound of* α *in Theorem 1 depends on* $\lambda_{\max}(\nabla^2 f_p)$*, which cannot be calculated in a distributed way. However, we can obtain another upper bound of* α *in a distributed way, which is more conservative than that in Theorem 1. Let* $H = \nabla^2 f_p$*. Then, the diagonal and off-diagonal blocks of* H *can be calculated as follows:*

$$\begin{aligned} H_{ii} &= \frac{\partial^2 f_p}{\partial x^{(i)2}} \\ &= A^{(i)T}A^{(i)} + \sum_{j\in N_i} e_i^{(i)}e_i^{(i)T} + e_j^{(i)}e_j^{(i)T} \\ &= A^{(i)T}A^{(i)} + \operatorname{diag}(|L^{(i)}|) \end{aligned} \tag{13}$$

and for $i \neq j$

$$H_{ij} = \frac{\partial^2 f_p}{\partial x^{(i)}\partial x^{(j)}} = \begin{cases} -e_i^{(i)}e_i^{(j)T} - e_j^{(i)}e_j^{(j)T}, & j \in N_i, \\ 0, & j \notin N_i, \end{cases} \tag{14}$$

where $\operatorname{diag}(|L^{(i)}|)$ *is a diagonal matrix whose diagonal entries are the absolute values of* $L^{(i)}$*, which is composed of the nonzero entries of the* i*-th row of the Laplacian matrix* L*. Let* $\mathcal{A} = \operatorname{diag}(A^{(i)T}A^{(i)})$ *be the block diagonal matrix of which the diagonal blocks are* $A^{(i)T}A^{(i)}$*. The eigenvalues of* $\mathcal{A}$ *are those of* $A^{(i)}A^{(i)T}$ *and* 0*. Let* $\mathcal{B} = H - \mathcal{A}$*. The diagonal blocks of* $\mathcal{B}$ *are* $\operatorname{diag}(|L^{(i)}|)$ *and the off-diagonal blocks are as computed in* (14)*. From Gersgorin discs theorem [22],* $\lambda_{\max}(\mathcal{B}) \leq 2\max_i |N_i|$*. Both* $\mathcal{A}$ *and* $\mathcal{B}$ *are real symmetric and thus Hermitian. From Weyl's inequalities [22],*

$$\begin{aligned} \lambda_{\max}(H) &\leq \lambda_{\max}(\mathcal{A}) + \lambda_{\max}(\mathcal{B}) \\ &= \max_{i\in V} \lambda_{\max}\left(A^{(i)}A^{(i)T}\right) + 2\max_{i\in V}|N_i|. \end{aligned}$$

Therefore,

$$\frac{2}{\lambda_{\max}(H)} \geq \frac{2}{\max_{i\in V}\lambda_{\max}\left(A^{(i)}A^{(i)T}\right)+2\max_{i\in V}|N_i|}.$$

As a result, the step size can be selected as

$$0 \leq \alpha \leq \frac{2}{\max_{i\in V}\lambda_{\max}\left(A^{(i)}A^{(i)T}\right)+2\max_{i\in V}|N_i|}$$

in order to guarantee the convergence of (11). *Note that* $\max_{i\in V}\lambda_{\max}\left(A^{(i)}A^{(i)T}\right)$ *and* $\max_{i\in V}|N_i|$ *can be obtained via maximum consensus algorithms in a distributed way.*

4.2. Algorithm Based on Approximated Newton's method

In the above, a communication-efficient distributed algorithm was proposed. However, we find in some simulation examples that it is slow. In this subsection, we will propose an accelerated distributed algorithm to solve $Ax = b$ by minimizing (6) based on an approximated Newton's method.

The centralized Newton's method to minimize f_p is

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \left(\nabla^2 f_p(\boldsymbol{x}(k))\right)^{-1}\nabla f_p(\boldsymbol{x}(k)).$$

For notational simplicity, denote $H = \nabla^2 f_p(\boldsymbol{x}(k))$ and $g(k) = \nabla f_p(\boldsymbol{x}(k))$. Note that H is a constant matrix for f_p in (6).

The gradient of f_p is computed in (8) and the diagonal and off-diagonal blocks of the Hessian of f_p are computed in (13) and (14), respectively. As we do not find any method to compute the inverse of the Hessian of f_p in a distributed way, we will next approximate it. Let $D_i = \gamma_i \frac{\partial^2 f_p}{\partial x^{(i)2}}$, where $\gamma_i > 1$ is a constant, and $D = \mathrm{diag}(D_1, \cdots, D_M)$. Also, let $F = H - D$. Then,

$$F_{ij} = \begin{cases} (1-\gamma_i)\left(A^{(i)T}A^{(i)} + \mathrm{diag}(|L^{(i)}|)\right), & i = j, \\ -e_i^{(i)}e_i^{(j)T} - e_j^{(i)}e_j^{(j)T}, & j \in N_i, \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

and F is Laplacian sparse. As $H = D + F$,

$$\begin{aligned} H^{-1} &= (D+F)^{-1} \\ &= D^{-\frac{1}{2}}(I + D^{-\frac{1}{2}}FD^{-\frac{1}{2}})^{-1}D^{-\frac{1}{2}} \\ &\approx D^{-\frac{1}{2}}(I - D^{-\frac{1}{2}}FD^{-\frac{1}{2}})D^{-\frac{1}{2}} \\ &= D^{-1} - D^{-1}FD^{-1} \end{aligned}$$

Replacing the inverse of the Hessian in Newton's method with its approximation, we obtain the following distributed approximated Newton-based algorithm to solve $Ax = b$:

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - (D^{-1} - D^{-1}FD^{-1})g(k), \tag{16}$$

where $\boldsymbol{x}$ is defined in (9) and $g(k) = \nabla f(\boldsymbol{x}(k))$. For the i-th agent, its estimate $x^{(i)}$ evolves as

$$\begin{aligned} &x^{(i)}(k+1) \\ &= x^{(i)}(k) - \left(D_i^{-1} g_i(k) - D_i^{-1} \sum_{j \in N_i \cup \{i\}} F_{ij} D_j^{-1} g_j(k) \right). \end{aligned} \tag{17}$$

Remark 7. *An approximated Newton's algorithm to solve distributed optimization problems can be found in [23], which requires the local objective functions to be strongly convex. But in our problem, the local objective functions $\frac{1}{2}\|A^{(i)}x^{(i)} - b_i\|$ are not strongly convex. Thus, the analysis in [23] does not apply to the problem in this paper.*

The convergence of $\boldsymbol{x}$ in (16) or $x^{(i)}$ in (17) is summarized in the following theorem.

Theorem 2. *Suppose that the system of linear equations $Ax = b$ has solutions and A is Laplacian sparse. Then under Assumption 1, $x^{(i)}(k)$, $\forall i \in V$ in (17) converges at a linear rate to the optimal solution set of (7). Hence, $x^\star(k)$ as defined in (12) converges to a solution to $Ax = b$.*

As the convergence of $x^\star(k)$ follows the convergence of $x^{(i)}(k)$, we will focus on the proof of the convergence of $x^{(i)}(k)$ at a linear rate to the optimal point of (7).

Proof. Let $\left(A^{(i)T}b_i\right)$ represent $\left(\left(A^{(1)T}b_1\right)^T \cdots \left(A^{(M)T}b_M\right)^T\right)^T$ for notational simplicity. It is easy to verify that

$$g(k) = H\boldsymbol{x}(k) - \left(A^{(i)T}b_i\right). \tag{18}$$

Let $\boldsymbol{x}^\star$ be an optimal solution to (7), then

$$g(\boldsymbol{x}^\star) = 0. \tag{19}$$

Combining (16) and (18), we obtain that

$$\begin{aligned}
&\boldsymbol{x}(k+1)\\
&= \boldsymbol{x}(k) - \left(D^{-1} - D^{-1}FD^{-1}\right) g(k)\\
&= \boldsymbol{x}(k) - \left(D^{-1} - D^{-1}FD^{-1}\right) \left(H\boldsymbol{x}(k) - \left(A^{(i)T}b_i\right)\right)\\
&= \boldsymbol{x}(k) - \left(D^{-1} - D^{-1}FD^{-1}\right) H\boldsymbol{x}(k)\\
&\quad + \left(D^{-1} - D^{-1}FD^{-1}\right) \left(A^{(i)T}b_i\right)\\
&= \left(I - \left(D^{-1} - D^{-1}FD^{-1}\right) H\right) \boldsymbol{x}(k)\\
&\quad + \left(D^{-1} - D^{-1}FD^{-1}\right) \left(A^{(i)T}b_i\right),
\end{aligned}$$

where $\boldsymbol{x}$ is defined in (9). Let $\boldsymbol{e}(k) = \boldsymbol{x}(k) - \boldsymbol{x}^\star$. Then it follows that

$$\begin{aligned}
\boldsymbol{e}(k+1) &= \boldsymbol{x}(k+1) - \boldsymbol{x}^\star\\
&= \left(I - \left(D^{-1} - D^{-1}FD^{-1}\right) H\right) (\boldsymbol{x}(k) - \boldsymbol{x}^\star)\\
&\quad + \left(I - \left(D^{-1} - D^{-1}FD^{-1}\right))H\right) \boldsymbol{x}^\star\\
&\quad + \left(D^{-1} - D^{-1}FD^{-1}\right) \left(A^{(i)T}b_i\right) - \boldsymbol{x}^\star\\
&= \left(I - \left(D^{-1} - D^{-1}FD^{-1}\right) H\right) (\boldsymbol{x}(k) - \boldsymbol{x}^\star)\\
&\quad - \left(D^{-1} - D^{-1}FD^{-1}\right) \left(H\boldsymbol{x}^\star - \left(A^{(i)T}b_i\right)\right).
\end{aligned}$$

On the other hand, it follows from (18) and (19) that $H\boldsymbol{x}^\star - \left(A^{(i)T}b_i\right) = 0$. Combining with the fact that

$$\begin{aligned}
&I - \left(D^{-1} - D^{-1}FD^{-1}\right) H\\
&\quad = I - \left(D^{-1} - D^{-1}FD^{-1}\right) (D + F)\\
&\quad = I - \left(I - D^{-1}F + D^{-1}F - D^{-1}FD^{-1}F\right)\\
&\quad = D^{-1}FD^{-1}F,
\end{aligned}$$

we obtain that

$$\begin{aligned}
\boldsymbol{e}(k+1) &= \left(I - \left(D^{-1} - D^{-1}FD^{-1}\right) H\right) \boldsymbol{e}(k)\\
&= D^{-1}FD^{-1}F\boldsymbol{e}(k).
\end{aligned} \tag{20}$$

To prove the convergence of $x^{(i)}(k)$ to the optimal point of (7) at a linear rate, it is sufficient to prove the following.

S1) The spectral radius of $D^{-1}FD^{-1}F$ is less than or equal to 1, that is, $\rho(D^{-1}FD^{-1}F) \leq 1$. Furthermore, 1 is the only eigenvalue lying on the unit circle.

S2) The eigenvalue 1 of $D^{-1}FD^{-1}F$ has the same algebraic and geometric multiplicity.

S3) The eigenvectors of $D^{-1}FD^{-1}F$ associated with eigenvalue 1 are in the kernel of H.

For notational simplicity, denote $\text{eig}_\lambda(S) = \{y|Sy = \lambda y\}$, where λ is an arbitrary real number. Then S3) is equivalent to $\text{eig}_1(D^{-1}FD^{-1}F) \subset \ker(H)$. It follows from S1) and S2) that $\boldsymbol{e}$ converges to a vector in $\text{eig}_1(D^{-1}FD^{-1}F)$, which is denoted as $\boldsymbol{e}(\infty)$. $\boldsymbol{x}$ then converges to $\boldsymbol{x}(\infty) = \boldsymbol{e}(\infty) + \boldsymbol{x}^\star$. From S3), $H\boldsymbol{e}(\infty) = 0$. Then

$$g(\infty) = H\left(\boldsymbol{e}(\infty) + \boldsymbol{x}^\star\right) - \left(A^{(i)T}b_i\right) = 0.$$

Therefore, $\boldsymbol{x}$ in (16) or $x^{(i)}$ in (17) converges to a solution to (7). Then, it follows from the linear dynamics of $\boldsymbol{e}$ that the convergence is at a linear rate. In the following, we will prove S1), S2) and S3), respectively.

Proof of S1) As $\rho(D^{-1}FD^{-1}F) = \rho(D^{-1}F)^2$, what we need to show is that $\rho(D^{-1}F) \leq 1$. As

$$D^{-1}F = D^{-\frac{1}{2}}(D^{-\frac{1}{2}}FD^{-\frac{1}{2}})D^{\frac{1}{2}},$$

we have that $\rho(D^{-1}F) = \rho\left(D^{-\frac{1}{2}}FD^{-\frac{1}{2}}\right)$. Next, we prove that

$$\rho\left(D^{-\frac{1}{2}}FD^{-\frac{1}{2}}\right) \leq 1,$$

which is equivalent to

$$\begin{aligned} D^{-\frac{1}{2}}FD^{-\frac{1}{2}} - I \leq 0 \\ D^{-\frac{1}{2}}FD^{-\frac{1}{2}} + I \geq 0. \end{aligned} \tag{21}$$

Note that (21) is valid if the following equations hold,

$$\begin{aligned} D^{-\frac{1}{2}}FD^{-\frac{1}{2}} - I < 0 \\ D^{-\frac{1}{2}}FD^{-\frac{1}{2}} + I \geq 0. \end{aligned} \tag{22}$$

Multiplying $D^{\frac{1}{2}}$ at both sides of (22), we obtain that

$$D - F > 0, \tag{23a}$$
$$F + D \geq 0. \tag{23b}$$

(23b) holds because $F + D = H$ and H is the Hessian of a non-negative function (6) and thus positive semidefinite. For (23a), Let FN be a matrix such that

$$FN_{ij} = \begin{cases} H_{ii}, & j = i, \\ -H_{ij}, & j \neq i. \end{cases}$$

and $D_{(\gamma)}$ be a block diagonal matrix such that its diagonal blocks are $\frac{2(\gamma_i - 1)}{\gamma_i} D_i$, $\forall i \in V$. Then, $D - F = D_{(\gamma)} + FN$. It is easy to verify that FN is the Hessian of the summation of the following,

$$\frac{1}{2} \sum_{i \in V} \left\| A^{(i)} x^{(i)} - b_i \right\|^2$$

and

$$\sum_{(i,j) \in E} \left(\left\| x_i^{(i)} + x_i^{(j)} \right\|^2 + \left\| x_j^{(j)} + x_j^{(i)} \right\|^2 \right).$$

Thus, FN is symmetric and positive semi-definite. Notice that $D_{ii} > 0$ and $\gamma_i > 1$, we have that $D_{(\gamma)} > 0$ and thus $D - F = D_{(\gamma)} + FN > 0$ and (23a) holds. Multiplying by $D^{-\frac{1}{2}}$ both sides of (23a), we obtain the first inequality in (22). So $\rho(D^{-1}FD^{-1}F) = \rho(D^{-1}F)^2 \leq 1$. Also, as $D^{-\frac{1}{2}}FD^{-\frac{1}{2}}$ is symmetric, all its eigenvalues are real and thus the eigenvalues of $D^{-1}F$ are also real. So, the eigenvalues of $D^{-1}FD^{-1}F$, as the squares of those of $D^{-1}F$, are real and non-negative. Therefore, 1 is the only eigenvalue lying on the unit circle.

Proof of S2) Notice that (23a) implies that $D^{-1}F$ does not have any eigenvalues equaling 1. So the algebraic multiplicity of the eigenvalue 1 of $D^{-1}FD^{-1}F$ is the same as that of the eigenvalue -1 of $D^{-1}F$ and $D^{-\frac{1}{2}}FD^{-\frac{1}{2}}$. From (23b), the algebraic multiplicity of the eigenvalue 1 of $D^{-\frac{1}{2}}FD^{-\frac{1}{2}}$ is the same as that of the eigenvalue 0 of $F+D$ which equals H. As H is symmetric, its eigenvalues have the same algebraic and geometric multiplicity. So, if we can show that the geometric multiplicity of eigenvalue 1 of $D^{-1}FD^{-1}F$ is the same as that of eigenvalue 0 of H, we can show that the eigenvalue 1 of $D^{-1}FD^{-1}F$ has the same algebraic and geometric multiplicity. This will be shown in the proof of S3) together with showing that the $\text{eig}_1(D^{-1}FD^{-1}F) = \text{eig}_0(H)$, where $\text{eig}_0(H)$ is $\ker(H)$.

Proof of S3) To prove S3), a stronger result is first proved that $\ker(H) = \text{eig}_1(D^{-1}FD^{-1}F)$, which is equivalent to proving $\text{eig}_1(D^{-1}FD^{-1}F) \subset \ker(H)$

and $\ker(H) \subset \text{eig}_1(D^{-1}FD^{-1}F)$. First, we have that

$$D^{-1}FD^{-1}F = D^{-1}FD^{-1}(H-D) = D^{-1}FD^{-1}H - D^{-1}F.$$

If $D^{-1}FD^{-1}Fx = x$, then

$$D^{-1}FD^{-1}Fx = D^{-1}FD^{-1}Hx - D^{-1}Fx = x.$$

Multiplying D on both sides of the second equal sign, we obtain that

$$FD^{-1}Hx - Fx = Dx,$$

and thus

$$FD^{-1}Hx = Fx + Dx = Hx.$$

Let $y = Hx$, then $FD^{-1}y = y$. Let $z = D^{-1}y$, $Fz = Dz$, i.e., $(D-F)z = 0$. As $D - F$ is positive definite from (23a) and thus invertible, $z = 0$ and $y = Hx = 0$. $\text{eig}_1(D^{-1}FD^{-1}F) \subset \ker(H)$ is proved. Second, if $Hx = 0$, then $(D+F)x = 0$, then $Dx = -Fx$ and $D^{-1}Fx = -x$, then $D^{-1}FD^{-1}Fx = -D^{-1}Fx = x$. $\ker(H) \subset \text{eig}_1(D^{-1}FD^{-1}F)$ is proved.

It follows from the above that S1)-S3) are proved and, as a result, $\boldsymbol{x}$ in (16) converges to the optimal solution to (7) and $x^\star$ in (12) converges to a solution to $Ax = b$. As (12) is linear, the convergence is at a linear rate. □

Remark 8. *The following approximation*

$$\left(I + D^{-\frac{1}{2}}FD^{-\frac{1}{2}}\right)^{-1} \approx I - D^{-\frac{1}{2}}FD^{-\frac{1}{2}}$$

is used when deriving H^{-1} in the approximated Newton's algorithm. In general, $D^{-\frac{1}{2}}FD^{-\frac{1}{2}}$ needs to be small (e.g., its norm is less than one) to make sense of the approximation. But the convergence analysis of the proposed approximated Newton-based algorithm does not depend on how close this approximation is. In the proof of Theorem 2, the convergence of the approximated Newton-based algorithm is proved through that of a linear dynamic system.

5. Algorithm Comparison

In this section, we make a brief comparison between existing algorithms and the proposed algorithms on the communication burdens. In addition, a

Table 1: Communication requirement per link

Algorithm	Communication burden
[1, 2, 3, 4]	2× network size
[11]	2+number of common neighbors
Gradient-based	2
Approximated Newton-based	4

comparison between the gradient-descent-based algorithm (11) and the approximated Newton-based algorithm (17) on the convergence rate and computation costs is also made in this section.

A comparison among different algorithms on the communication burden per link is made in Table 1 for a Laplacian sparse matrix on condition that the state of each agent is a scalar. The communication quantity for each link of the proposed gradient-based and approximated Newton-based algorithms is constant and does not change with system size or local network structure.

For the two proposed algorithms (11) and (17), although a quantitative convergence rate is not available, we find through simulations that the approximated Newton-based algorithm (17) converges much faster than the gradient-descent-based algorithm (11).

Remark 9. *No quantitative results on the faster convergence of the approximated Newton-based algorithm are obtained in this paper. The difficulty to obtain such quantitative results is caused by the difficulty to compare the eigenvalues of the two matrices with those of their product. The convergence of the gradient-based algorithm is equivalent to that of* (28) *and the convergence of the approximated Newton-based algorithm is equivalent to that of* (20)*, where H is the Hessian of the objective matrix with $H = D + F$. Thus, the convergence rate of the gradient-based algorithm is determined by* $\max\{|1 - \alpha\lambda_{\min}(H)|, |1 - \alpha\lambda_{\max}(H)|\}$ *and that of the approximate Newton-based algorithm is determined by* $\max\{|\lambda_{\min}(D^{-1}FD^{-1}F)|, |\lambda_{\max}(D^{-1}FD^{-1}F)|\}$*. The eigenvalues of $D^{-1}FD^{-1}F$ are the squares of those of $D^{-1}F$. $D^{-1}F$ is similar to the symmetric matrix $D^{-\frac{1}{2}}FD^{-\frac{1}{2}}$ of which the eigenvalues are all real. Thus, the eigenvalues of $D^{-1}F$ are all real, and thus the eigenvalues of*

$D^{-1}FD^{-1}F$ are non-negative. Then,

$$\begin{aligned}&\max\left\{\left|\lambda_{\min}\left(D^{-1}FD^{-1}F\right)\right|,\left|\lambda_{\max}\left(D^{-1}FD^{-1}F\right)\right|\right\}\\&\quad=\lambda_{\max}\left(D^{-1}FD^{-1}F\right)\\&\quad=\max\left\{\lambda^2_{\max}\left(D^{-1}F\right),\lambda^2_{\min}\left(D^{-1}F\right)\right\}.\end{aligned}$$

Also, $D^{-1}F = D^{-1}(H-D) = -(I-D^{-1}H)$. Therefore, the convergence rate of the approximated Newton-based algorithm is determined by the square of the eigenvalues of $I-D^{-1}H$. However, we do not know the relationship between the eigenvalues of H and $D^{-1}H$. Thus, we cannot compare the eigenvalues of $D^{-1}FD^{-1}F$ and $I-\alpha H$.

The communication burden of the approximated Newton-based algorithm doubles that of the gradient-descent-based algorithm in each iteration. We already know that the i-th agent in (11) only requires two parts, i.e., $x_i^{(j)}$ and $x_j^{(j)}$ through the communication link (j,i). In (17), let $v^{(j)} = D_j^{-1}g_j(k)$. If $j \in N_i$, $F_{ij} = -e_i^{(i)}e_i^{(j)T} - e_j^{(i)}e_j^{(j)T}$. Then, the i-th agent only requires $v_i^{(j)}$ and $v_j^{(j)}$ besides $x_i^{(j)}$ and $x_j^{(j)}$. The algorithm based on the approximated Newton's method thus requires twice as much communication as that based on the gradient descent method. As the dimensions of $x_i^{(j)}$ and $x_j^{(j)}$ are usually very low in spite of the scale of the whole system, the algorithm based on the approximated Newton's method is still communication-efficient. For example, in three-phase power flow problems, $\dim\left(x_i^{(j)}\right)$, $\dim\left(x_i^{(j)}\right)$, $\dim\left(v_i^{(j)}\right)$, and $\dim\left(v_j^{(j)}\right)$ are all at most three, so what is transmitted from agent j to i is at most twelve scalars no matter how large the system is.

The approximated Newton-based algorithm defined in (17) has a higher computation cost than the gradient-descent-based algorithm (11). First, both algorithms need to calculate $g_i(k)$ and perform the subtraction between two vectors of $\dim\left(x^{(i)}\right)$ dimension. Second, the i-th agent in (17) needs to perform more computations including

1) the computation of $D_i = A^{(i)T}A^{(i)} + \mathrm{diag}(|L^{(i)}|)$, a symmetric matrix of $\dim\left(x^{(i)}\right)$ dimension, which can be done at the initialization step;
2) the inverse of D_i^{-1}, a symmetric matrix of $\dim\left(x^{(i)}\right)$ dimension, which can be done at the initialization step;
3) the multiplication of D_i^{-1} and $g_i(k)$ per iteration;
4) the summation of $|N_i|$ vectors, $v_i^{(j)}$, of $\dim\left(x_i^{(i)}\right)$ dimension per iteration.

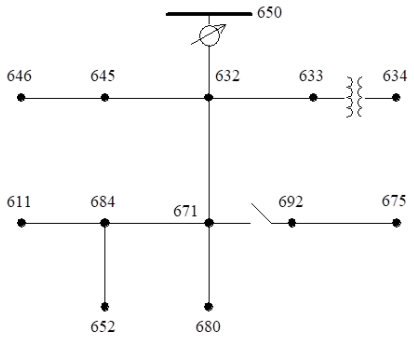

Figure 1: Illustration of IEEE 13-node test feeder.

It can be seen that the extra computational burdens associated with (17) greatly depend on the values of dim $\left(x^{(i)}\right)$ and dim $\left(x_i^{(i)}\right)$. In many practical problems, their values are usually very small in spite of the large system size. Take the three-phase power flow problem for example. The value of dim $\left(x_i^{(i)}\right)$ is at most three and dim $\left(x^{(i)}\right)$ at most $3|N_i|$, which is also small due to the sparse physical connections in the real-world systems. However, since (17) usually converges much faster than (11), the total communication and computation costs of (17) may be lower than those of (11).

6. Simulation Studies

In this section, simulation studies are carefully conducted to evaluate and illustrate the effectiveness of the proposed communication-efficient distributed algorithms that are defined in (11) and (17), respectively. We applied both algorithms to solve a system of linear equations (3) that is required by each iteration of the Newton-Raphson method. The Newton-Raphson method is widely used to compute the three-phase unbalance power flow problem in (2). The testing system is the IEEE 13-node test feeder [24] as shown in Fig. 1, where each line represents the electricity wire between the neighboring nodes. There is a switch between nodes 671 and 692 and a transformer between nodes 633 and 634. The line impedance and nodal power data can be found in [24].

In our simulation studies, we treat each node as an agent and assume that the communication network and the physical network share the same topology. The collective state of all agents consists of the magnitudes and

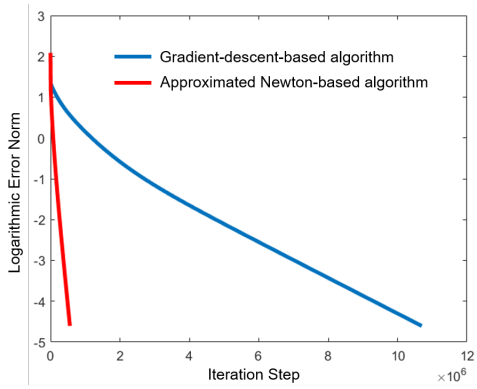

Figure 2: Norms of Estimation Errors.

phase angles of the complex voltages of all nodes. As the nodes may be of one-phase, two-phase, or three-phase, different agents have state vectors of different dimensional. The Jacobian matrix (4) consists of four blocks. Every block has the same sparse structure as the nodal admittance matrix of the power system and thus is Laplacian sparse. The Jacobian matrix is more complex than a Laplacian sparse matrix. However, we can still solve the system of linear equations in (3) with the algorithms in (11) and (17) because the Jacobian matrix, though consisting of four Laplacian sparse blocks, requires the same information flow as the nodal admittance matrix of the system. Each agent (or each node in the power system) knows the rows of the Jacobian matrix J which is generated from its active power equation and reactive power equation in (2). At each iterative step, each agent share with its neighbors its estimates on the voltage magnitudes and phase angles of itself and that neighbor.

The respective convergence results of the proposed algorithms are shown in Fig. 2. The step size α for the gradient-descent-based algorithm in (11) is chosen as 0.00002, and it is tested that if $\alpha = 0.00003$, the algorithm (11) diverges. So $\alpha = 0.0002$ might be very close to the largest step size that makes (11) converge in this example. The coefficient γ_i for the approximated Newton-based algorithm in (17) is selected to be 2. The error is defined as $x^\star(k) - x_{\text{sol}}$, where $x^\star(k)$ is defined in (12), x_{sol} is the solution to the problem obtained from a centralized method, and k is the iteration step. Note that in Fig. 2, the vertical axis is in the logarithmic scale. It can be observed that i) both algorithms can find the solution to the problem because the logarithmic

error norms decrease to a negative value and the error norms become very close to zero when the algorithms are terminated; ii) the error norms of both algorithms converge at a linear rate because the shapes of both curves are close to a straight line as iteration continues; iii) the approximated Newton's algorithm converges faster than the gradient descent one.

7. Conclusions

In this paper, two communication-efficient distributed algorithms were proposed based on multi-agent systems. The algorithms can be applied to various systems, e.g., power systems. The first algorithm was based on the gradient descent method while the second one was based on an approximation to Newton's method. In the gradient-descent-based algorithm, only the entries of the collective state vector which corresponds to each agent itself and one of its neighbors, instead of every entry of the collective state vector, were communicated through a communication link. In the approximated Newton-based algorithm, the corresponding entries in the gradient vector were also communicated. It was proven that both algorithms converged at a linear rate, while the approximated Newton-based algorithm converged faster than the gradient-descent-based algorithm at the price of heavier computation and communication burdens. The effectiveness of both algorithms was confirmed and illustrated by applying them to solve the three-phase unbalanced power flow problem in power systems through simulation studies.

In this paper, the proposed two algorithms have been developed under the assumption of synchronous communication. In other words, all the necessary communications at each iteration occur simultaneously. In some real-world applications, it may be difficult to coordinate all the agents at the same time and ensure them to communicate simultaneously. It becomes necessary to accommodate the inevitable asynchronous update and allow individual communication channels to be activated according to their own clocks. Therefore, our future work will focus on the design of communication-efficient distributed algorithms with asynchronous updates.

8. Proof of Theorem 1

When $Ax = b$ has a unique solution, f_p is strongly convex, and the convergence at a linear rate of (11) is a direct result of Theorem 2.1.14 in

[25]. But when $Ax = b$ has multiple solutions, f_p in (6) is not strongly convex, and thus the results in [25] cannot be used to prove Theorem 1.

In order to prove Theorem 1 when $Ax = b$ has multiple solutions, we need the following lemmas.

Lemma 1. *Let S be symmetric and positive semi-definite. Let $U = (U_1\ U_2)$ be the orthogonal matrix and $\Lambda = \begin{pmatrix} \Lambda_1 & \\ & 0 \end{pmatrix}$ be the diagonal matrix such that $S = U\Lambda U^T = U_1\Lambda_1 U_1^T$, where Λ_1 is a diagonal matrix with positive diagonal entries. Let $z = U^T y$, then the following statements are equivalent:*

1) $Sy = 0$,
2) $U_1\Lambda_1 U_1^T y = 0$,
3) $\Lambda z = 0$.

Proof. As $S = U_1\Lambda_1 U_1^T$, *1)* and *2)* are equivalent. Note that $Sy = U\Lambda U^T y = U\Lambda z$. As U is orthogonal and thus invertible, $Sy = 0$ is equivalent to $\Lambda z = 0$, and thus *1)* and *3)* are equivalent. As a result, *1)*, *2)*, and *3)* are equivalent. □

Lemma 2. *Let v be the projection of y_0 onto the hyper-plane $\{y|Cy = d\}$ and C be row full rank. Then*

$$v = y_0 - C^T \left(CC^T\right)^{-1} (Cy_0 - d),$$

and

$$\|v - y_0\|^2 = (Cy_0 - d)^T \left(CC^T\right)^{-1} (Cy_0 - d).$$

It is straightforward to see that $v = \operatorname{argmin}_{\{y:Cy=d\}} \frac{1}{2}\|y - y_0\|^2$. With the Karush–Kuhn–Tucker (KKT) condition, Lemma 2 can be readily proved and is thus omitted.

Lemma 3. *Let S be a nonzero symmetric positive semi-definite matrix and $f(y) = \frac{1}{2} y^T S y$. Let $Y^\star = \operatorname{argmin} f(y) = \{y^\star : Sy^\star = 0\}$ be the optimal set of $f(y)$. Then the gradient descent method $y(k+1) = y(k) - \alpha\nabla f(y(k)) = y(k) - \alpha S y(k)$ converges to $Y^\star$ at a linear rate provided that $0 < \alpha < \frac{2}{\lambda_{\max}(S)}$, where $\lambda_{\max}(S)$ is the maximum eigenvalue of S.*

Proof. Let $y^\star(k) = \operatorname{argmin}_{y^\star \in Y^\star} \|y(k) - y^\star\|$ as the closest optimal point to $y(k)$ with $Y^\star = \{y^\star : Sy^\star = 0\} = \{y^\star : U_1\Lambda_1 U_1^T y^\star = 0\}$, where U_1 is defined in Lemma 1. Let $C = U_1\Lambda_1 U_1^T$ in Lemma 2. It follows that

$$\|y(k) - y^\star(k)\|^2 = (y(k) - y^\star(k))^T U_1 U_1^T (y(k) - y^\star(k)). \tag{24}$$

Then,

$$\begin{aligned}
&\|y(k+1) - y^\star(k+1)\|^2 \\
&\quad \leq \|y(k+1) - y^\star(k)\|^2 \\
&\quad = \|y(k) - \alpha S y(k) - y^\star(k)\|^2 \\
&\quad = \|y(k) - y^\star(k)\|^2 + \alpha^2 y(k)^T S^T S y(k) \\
&\qquad - 2(y(k) - y^\star(k))^T S y(k).
\end{aligned}$$

As $Sy^\star(k) = 0$, it follows that

$$y(k)^T S^T S y(k) = (y(k) - y^\star(k))^T S^T S (y(k) - y^\star(k))$$

and

$$(y(k) - y^\star(k))^T S y(k) = (y(k) - y^\star(k))^T S (y(k) - y^\star(k)).$$

Thus,

$$\begin{aligned}
&\|y(k+1) - y^\star(k+1)\|^2 \\
&\quad \leq \|y(k) - y^\star(k)\|^2 \\
&\qquad + \alpha^2 (y(k) - y^\star(k))^T S^T S (y(k) - y^\star(k)) \\
&\qquad - 2(y(k) - y^\star(k))^T S (y(k) - y^\star(k)).
\end{aligned}$$

Let z and Λ be defined as in Lemma 1. Then,

$$\begin{aligned}
&\|y(k+1) - y^\star(k+1)\|^2 \\
&\quad \leq \|y(k) - y^\star(k)\|^2 \\
&\qquad + \alpha^2 (z(k) - z^\star(k))^T \Lambda^2 (z(k) - z^\star(k)) \\
&\qquad - 2\alpha (z(k) - z^\star(k))^T \Lambda (z(k) - z^\star(k)) \\
&\quad = \|y(k) - y^\star(k)\|^2 \\
&\qquad + (z(k) - z^\star(k))^T (\alpha^2 \Lambda^2 - 2\alpha\Lambda)(z(k) - z^\star(k)).
\end{aligned}$$

As

$$z(k) - z^\star(k) = U^T (y(k) - y^\star(k)) = \begin{pmatrix} U_1^T (y(k) - y^\star(k)) \\ U_2^T (y(k) - y^\star(k)) \end{pmatrix},$$

then,

$$\begin{aligned}
&(z(k) - z^\star(k))^T (\alpha^2 \Lambda^2 - 2\alpha\Lambda)(z(k) - z^\star(k)) \\
&\quad = [U_1^T (y(k) - y^\star(k))]^T (\alpha^2 \Lambda_1^2 - 2\alpha\Lambda_1)[U_1^T (y(k) - y^\star(k))].
\end{aligned}$$

As $0 < \alpha < 2/\lambda_{\max}(S) = 2/\lambda_{\max}(\Lambda_1)$ and $\lambda_{\max}(\Lambda_1) > 0$, $0 < \alpha\lambda_{\max}(\Lambda_1) < 2$. Thus, $(\alpha\lambda_{\max}(\Lambda_1))^2 - 2\alpha\lambda_{\max}(\Lambda_1) < 0$. Similarly, $(\alpha\lambda_{\min}(\Lambda_1))^2 - 2\alpha\lambda_{\min}(\Lambda_1) < 0$.

Let $\lambda(\Lambda_1)$ be any eigenvalue of Λ_1. We have that

$$\begin{aligned}
&\alpha^2\lambda(\Lambda_1)^2 - 2\alpha\lambda(\Lambda_1) \\
&= (\alpha\lambda(\Lambda_1))^2 - 2\alpha\lambda(\Lambda_1) \\
&\leq \max\left\{(\alpha\lambda_{\min}(\Lambda_1))^2 - 2\alpha\lambda_{\min}(\Lambda_1)\,,\right. \\
&\qquad\qquad \left.(\alpha\lambda_{\max}(\Lambda_1))^2 - 2\alpha\lambda_{\max}(\Lambda_1)\right\} \\
&< 0.
\end{aligned}$$

Denote

$$\begin{aligned}
\bar{\lambda} = \max\left\{(\alpha\lambda_{\min}(\Lambda_1))^2 - 2\alpha\lambda_{\min}(\Lambda_1)\,,\right. \\
\left.(\alpha\lambda_{\max}(\Lambda_1))^2 - 2\alpha\lambda_{\max}(\Lambda_1)\right\}.
\end{aligned}$$

Then it can be obtained that

$$\begin{aligned}
&(z(k) - z^\star(k))^T(\alpha^2\Lambda^2 - 2\alpha\Lambda)(z(k) - z^\star(k)) \\
&\leq \bar{\lambda}\left(U_1^T\left(y(k) - y^\star(k)\right)\right)^T\left(U_1^T\left(y(k) - y^\star(k)\right)\right) \\
&= \bar{\lambda}(y(k) - y^\star(k))^T U_1 U_1^T (y(k) - y^\star(k)) \\
&= \bar{\lambda}\left\|y(k) - y^\star(k)\right\|^2,
\end{aligned}$$

where the last equality results from (24).

Thus,

$$\begin{aligned}
&\|y(k+1) - y^\star(k+1)\|^2 \\
&\leq \|y(k) - y^\star(k)\|^2 + \bar{\lambda}\|y(k) - y^\star(k)\|^2 \\
&\leq (1 + \bar{\lambda})\|y(k) - y^\star(k)\|^2.
\end{aligned}$$

Note that $\bar{\lambda} < 0$, $y(k)$, $k = 1, 2, 3, \cdots$ converges to $Y^\star$ at a linear rate. Hence, (11) converges to $X^\star$ at a linear rate. □

The proof of Theorem 1 when $Ax = b$ has multiple solutions follows from Lemma 3.

Proof of Theorem 1. Notice that f_p in (7) is quadratic. So f_p can be reformulated as

$$f_p = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^\star)^T H(\boldsymbol{x} - \boldsymbol{x}^\star) \tag{25}$$

$$= \frac{1}{2}\boldsymbol{e}^T H \boldsymbol{e}, \tag{26}$$

where H is the Hessian of f_p, $\boldsymbol{x}^\star$ is an optimal point of f_p, and $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}^\star$. The optimal set of (25) is $X^\star = \{\boldsymbol{x} : H(\boldsymbol{x} - \boldsymbol{x}^\star) = 0\}$.

The gradient descend method to optimize (25) is

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \alpha H(\boldsymbol{x}(k) - \boldsymbol{x}^\star). \tag{27}$$

Note that $H(\boldsymbol{x}(k) - \boldsymbol{x}^\star)$ is the gradient of f_p. Thus, (27) can be calculated without any knowledge of the optimal point $\boldsymbol{x}^\star$. Subtracting $\boldsymbol{x}^\star$ from both sides of (25), it can be obtained that

$$\boldsymbol{e}(k+1) = \boldsymbol{e}(k) - \alpha H \boldsymbol{e}(k), \tag{28}$$

which is the gradient descent step to optimize (26). From Lemma 3, $\boldsymbol{e}(k)$ converges to the set $\{\boldsymbol{e} : H\boldsymbol{e} = 0\}$ at a linear rate. $\boldsymbol{x}(k) = \boldsymbol{x}^\star + \boldsymbol{e}(k)$ then converges at a linear rate to the set $\{\boldsymbol{x} : H\boldsymbol{x} = H\boldsymbol{x}^\star\}$, which is the optimal set minimizing (25), or equivalently, (6). □

References

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

[2] P. Wang, W. Ren, Z. Duan, Distributed minimum weighted norm solution to linear equations associated with weighted inner product, in: 2016 IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 5220–5225. doi:10.1109/CDC.2016.7799068.

[3] L. Wang, D. Fullmer, A. S. Morse, A distributed algorithm with an arbitrary initialization for solving a linear algebraic equation, in: 2016 American Control Conference (ACC), 2016, pp. 1078–1081.

[4] P. Wang, W. Ren, Z. Duan, Distributed algorithm to solve a system of linear equations with unique or multiple solutions from arbitrary initializations, IEEE Transactions on Control of Network Systems 6 (1) (2019) 82–93. doi:10.1109/TCNS.2018.2797805.

[5] F. Pasqualetti, R. Carli, F. Bullo, Distributed estimation via iterative projections with application to power network monitoring, Automatica 48 (5) (2012) 747 – 758. doi:http://dx.doi.org/10.1016/j.automatica.2012.02.025.

[6] F. Pasqualetti, R. Carli, A. Bicchi, F. Bullo, Distributed estimation and detection under local information, IFAC Proceedings Volumes 43 (19) (2010) 263 – 268, 2nd IFAC Workshop on Distributed Estimation and Control in Networked Systems. doi:http://dx.doi.org/10.3182/20100913-2-FR-4014.00032.

[7] F. Pasqualetti, R. Carli, F. Bullo, A distributed method for state estimation and false data detection in power networks, in: 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm), 2011, pp. 469–474. doi:10.1109/SmartGridComm.2011.6102368.

[8] K. You, S. Song, R. Tempo, A networked parallel algorithm for solving linear algebraic equations, in: 2016 IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 1727–1732. doi:10.1109/CDC.2016.7798514.

[9] X. Wang, S. Mou, D. Sun, Improvement of a distributed algorithm for solving linear equations, IEEE Transactions on Industrial Electronics 64 (4) (2017) 3113–3117. doi:10.1109/TIE.2016.2636119.

[10] J. Liu, S. Mou, A. S. Morse, Asynchronous distributed algorithms for solving linear algebraic equations, IEEE Transactions on Automatic Control 63 (2) (2018) 372–385. doi:10.1109/TAC.2017.2714645.

[11] S. Mou, Z. Lin, L. Wang, D. Fullmer, A. Morse, A distributed algorithm for efficiently solving linear equations and its applications (special issue jcw), Systems & Control Letters 91 (2016) 21 – 27.

[12] J. Liu, X. Gao, T. Başar, A communication-efficient distributed algorithm for solving linear algebraic equations, in: 2014 7th International

Conference on NETwork Games, COntrol and OPtimization (NetGCoop), 2014, pp. 62–69.

[13] J. Liu, B. D. O. Anderson, Communication-efficient distributed algorithms for solving linear algebraic equations over directed graphs, in: 2020 59th IEEE Conference on Decision and Control (CDC), 2020, pp. 5360–5365. doi:10.1109/CDC42340.2020.9304062.

[14] G. Shi, B. D. O. Anderson, U. Helmke, Network Flows that Solve Linear Equations, ArXiv e-prints (Oct. 2015). arXiv:1510.05176.

[15] B. D. O. Anderson, S. Mou, A. S. Morse, U. Helmke, Decentralized gradient algorithm for solution of a linear equation, ArXiv e-prints (Sep. 2015). arXiv:1509.04538.

[16] M. Yang, C. Y. Tang, A distributed algorithm for solving general linear equations over networks, in: 2015 54th IEEE Conference on Decision and Control (CDC), 2015, pp. 3580–3585.

[17] J. Liu, X. Chen, T. Basar, A. Nedic, A continuous-time distributed algorithm for solving linear equations, in: 2016 American Control Conference (ACC), 2016, pp. 5551–5556.

[18] J. Zhou, X. Wang, S. Mou, B. D. O. Anderson, Finite-time distributed linear equation solver for minimum l_1 norm solutions, arXiv preprint arXiv:1709.10154 (2017).
URL `http://arxiv.org/abs/1709.10154`

[19] X. Wang, S. Mou, B. D. O. Anderson, A double-layered framework for distributed coordination in solving linear equations, CoRR abs/1711.10947 (2017). arXiv:1711.10947.
URL `http://arxiv.org/abs/1711.10947`

[20] P. Wang, Y. Gao, N. Yu, W. Ren, J. Lian, D. Wu, Communication-efficient distributed solutions to a system of linear equations with laplacian sparse structure, in: 2018 IEEE Conference on Decision and Control (CDC), 2018, pp. 3367–3372.

[21] A. Bergen, V. Vittal, Power Systems Analysis, Prentice Hall, 2000.

[22] R. A. Horn, C. R. Johnson, Matrix Analysis, Cambridge University Press, 1985, cambridge Books Online.
URL http://dx.doi.org/10.1017/CBO9780511810817

[23] A. Mokhtari, Q. Ling, A. Ribeiro, Network newton distributed optimization methods, IEEE Transactions on Signal Processing 65 (1) (2017) 146–161. doi:10.1109/TSP.2016.2617829.

[24] IEEE PES AMPS DSAS Test Feeder Working Group, IEEE 13 Node Test Feeder.
URL https://site.ieee.org/pes-testfeeders/resources/

[25] I. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, Mathematics and its applications, Kluwer Academic Publishers, 2004.