Communication-efficient Distributed Solutions to a System of Linear Equations with Laplacian Sparse Structure

Peng Wang, Yuanqi Gao, Nanpeng Yu, Wei Ren, Jianming Lian, and Di Wu

***Abstract*— Two communication-efficient distributed algorithms are proposed to solve a system of linear equations $Ax = b$ with Laplacian sparse A. A system of linear equations with Laplacian sparse A can be found in many applications, e.g., the power flow problems and other network flow problems. The first algorithm is based on the gradient descent method in optimization and the agents only share two parts of the system state instead of that of the whole system state, which saves significant communication. The two parts shared by every agent through a communication link are the state information of its own and its neighbor connected by the communication link. The second method is obtained from an approximation to the Newton method, which converges faster. It requires twice as much communication as the first one but is still communication-efficient due to the low dimension of each part shared between agents. The convergence at a linear rate of both methods is proved. A comprehensive comparison of the convergence rate, communication burden, and computation costs between the methods is made. Finally, simulations are conducted to show the effectiveness of both methods.**

I. INTRODUCTION

Solving a system of linear equations $Ax = b$ 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 attentions from the researchers.

Many distributed algorithms to solve $Ax = b$ were proposed in the literature, e.g. [1]–[17]. These algorithms assume that each agent knows some rows of the augmented matrix $\begin{pmatrix} A & b \end{pmatrix}$. The algorithms in [12]–[17] are continuous-time ones while those in [1]–[11] are discrete-time ones. In this paper we focus on discrete-time distributed algorithms to solve $Ax = b$. In [1], a geometrically convergent distributed algorithm is proposed for both synchronous and asynchronous updates under repeatedly jointly strongly connected graphs, which requires locally feasible initializations. Ref. [10] extends the results in [1] by considering the influence of communication and computation delays and arbitrary initializations. In [2], distributed algorithms are proposed to find the minimum norm solution of a system of linear equations associated with weighted inner products. Ref. [9] then broadens the results in [1] and [2], allowing arbitrary initializations for convergence to a general solution and special initialization for convergence to a solution closest to a given point. When $Ax = b$ has a unique solution, a distributed algorithm is designed in [3] to allow arbitrary initializations with feedback of the deviation from local systems of linear equations and the geometric convergence rate is proved. 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]–[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 method 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 the state of the whole system. However, in many applications where the matrix A is sparse and the system is large-scale, the distributed methods in [1]–[10] will lead to significant communication overhead. In this case, communication-efficient distributed algorithms to solve $Ax = b$ are necessary. A communication-efficient distributed method is proposed in [11] for general sparse matrices, but the method in [11] requires the information of common nonzero parts of agents' rows and their neighbors' columns. When the matrix A is Laplacian sparse, the common nonzero parts of different rows and columns actually require sharing information of agents' common neighbors, which might not be available to the agents.

In this paper, we develop communication-efficient distributed algorithms to solve $Ax = b$ with Laplacian sparse A, which requires less communication than that in [11]. Matrices with Laplacian sparse structure can be found in many problems such as network flow problems. In particular, the power flow problem in the smart grid domain involves network matrices with Laplacian sparse property. In our proposed communication-efficient distributed algorithms, the agents only transmit the information of their own and one of their neighbors connected by the communication link, instead of the state of the whole system as in [1]–[10] or the state of their common neighbors as in [11]. In the first proposed method, only two parts of the state vector of the system are transmitted through each communication link while in the second proposed method, two parts of the state vector and the gradient vector, respectively, of the system are transmitted.

Peng Wang, Jianming Lian, and Di Wu are with the Pacific Northwest National Laboratory, 902 Battelle Boulevard, Richland, WA. 99352. Corresponding emails: peng.wang@pnnl.gov, jianming.lian@pnnl.gov, di.wu@pnnl.gov.

Yuanqi Gao, Nanpeng Yu, and Wei Ren are with the Department of Electrical and Computer Engineering at the University of California, Riverside, 900 University Ave., Riverside, CA. 92521. Corresponding emails: ygao024@ucr.edu, nyu@ece.ucr.edu, and ren@ece.ucr.edu

As each part of the state vector and the gradient vector is low dimensional, both methods 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 geometric convergence. Then we propose an accelerated communication-efficient distributed algorithm based on an approximation to the Newton method. The algorithm based on the approximated Newton method requires twice as much communication as the one based on the gradient method, but it converges faster.

The rest of the paper is organized as follows. In Section II, some preliminary knowledge on graph theory, Laplacian sparse matrix, and power flow problems is introduced. In Section III, the two communication-efficient distributed algorithms are proposed to solve $Ax = b$ with Laplacian sparse A and a comparison between them is made. Simulations are conducted to illustrate the effectiveness of the two methods in Section IV. Finally, the conclusions are stated in Section V.

II. Preliminaries

In this part, we will provide some preliminary knowledge on graph theory, which is necessary for distributed algorithms, Laplacian sparse matrix, which is our research focus in this paper, and power flow problems, which provide applications to our research focus.

A. Graph Theory

An $\bar{m}$th order undirected graph, denoted by $\mathcal{G}(V, E)$, is composed of a vertex set $V = \{1, \cdots, \bar{m}\}$ and an edge set $E \subseteq V \times V$. We use the pair (j, i) to denote the edge between vertex j and vertex i. 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 vertex i, i.e., $N_i = \{j : (j, i) \in E\}$. The number of vertex i's neighbors is denoted by $|N_i|$. The Laplacian matrix $L = [l_{ij}]_{\bar{m} \times \bar{m}} \in \mathbb{R}^{\bar{m} \times \bar{m}}$ 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.

B. Laplacian Sparse Matrix

We consider the matrix A with the following special sparse structure:

Definition 1 (Laplacian sparse matrix): A matrix A has the Laplacian sparse structure of a 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 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.

Laplacian sparse matrices appear in many problems, e.g., power flow problems. In power flow problems, the Ybus of a power system has 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 also be regarded as a Laplacian sparse matrix. See Section II-C for details.

C. The Power Flower Problem

In this subsection, we will detail the relationship between power flow problem and the 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-phased power flow problems and the Newton-Raphson method. A comprehensive description of power flow problems and the Newton-Raphson method can be found in [18].

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 bus admittance matrix. Notice that Y has the same sparse structure as the Laplacian matrix of $\mathcal{G}$ and has Laplacian sparse structure of $\mathcal{G}$.

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. Then the power flow equation is as follows:

$$\begin{aligned} \textstyle\sum_{k=1}^{|\mathcal{V}|} |v_i||v_k|(G_{ik} \cos\theta_{ik} + B_{ik} \sin\theta_{ik}) - p_i = 0, \\ \textstyle\sum_{k=1}^{|\mathcal{V}|} |v_i||v_k|(G_{ik} \sin\theta_{ik} - B_{ik} \cos\theta_{ik}) - q_i = 0, \end{aligned} \tag{1}$$

where G_{ik} and B_{ik} are the (i, k)th entry of matrices G and B, respectively, and θ_{ik} is the nodal voltage angle difference between nodes i and k. As the matrix Y has Laplacian sparse structure of $\mathcal{G}$, so do G and B. So when nodes i and k are not connected, G_{ik} and B_{ik} are both zero.

The Newton-Raphson method to solve (1) is introduced below. Define the vector of unknowns $x = [\theta_2, \cdots, \theta_{|\mathcal{V}|}, |v_2|, \cdots, |v_{|\mathcal{V}|}|]^T$. x does not contain $|v_1|, \theta_1$ since they are known. Let

$$\begin{aligned} p_i(x) = \textstyle\sum_{k=1}^{|\mathcal{V}|} |v_i||v_k|(G_{ik} \cos\theta_{ik} + B_{ik} \sin\theta_{ik}), \\ q_i(x) = \textstyle\sum_{k=1}^{|\mathcal{V}|} |v_i||v_k|(G_{ik} \sin\theta_{ik} - B_{ik} \cos\theta_{ik}), \end{aligned} \tag{2}$$

and then (1) is converted to

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

In the Newton-Raphson method, Eq. (3) is iteratively solved. In each iterative step, we need to solve a system of linear equations as follows:

$$-J\Delta x = y \tag{4}$$

where J is the Jacobian matrix of the functions in (3), y is a constant related to (2) in each step.

The Jacobian matrix can be partitioned as $J = \begin{pmatrix} J_{p\theta} & J_{p|v|} \\ J_{q\theta} & J_{q|v|} \end{pmatrix}$, where $J_{p\theta} = \frac{\partial p(x)}{\partial \theta}$, $J_{p|v|} = \frac{\partial p(x)}{\partial |v|}$, $J_{q\theta} = \frac{\partial q(x)}{\partial \theta}$, and $J_{q|v|} = \frac{\partial q(x)}{\partial |v|}$. When G_{ik} and B_{ik} are both zero, the (i,k)th entries of $J_{p\theta}, J_{p|v|}, J_{q\theta}, J_{q|v|}$ are all zeros. So each block of the Jacobian matrix has the same sparse structure as the matrix G or B. As G and B have Laplacian sparse structure of $\mathcal{G}$, $J_{p\theta}, J_{p|v|}, J_{q\theta}, J_{q|v|}$ also have Laplacian sparse structure of $\mathcal{G}$.

III. Communication-efficient Distributed Solutions to Linear Equations

In this section, we will develop two communication-efficient distributed algorithms to solve $Ax = b$. The first algorithm is based on the gradient descent method while the second one on the approximated Newton method, which converges faster than the first one. The first algorithm only requires communication of two parts of the state vector of the whole system through each communication link, i.e., the two parts owned by the two agents connected by the link, while the second algorithm also requires the communication of the two corresponding parts of the gradient vector of the whole system.

A. Problem Formulation

Suppose that we have a group of $\bar{m}$ agents which form the vertex set V and communication links between the agents which form the edges between the vertices. Then we have a graph $\mathcal{G}(V, E)$ to represent the agent network. For the graph $\mathcal{G}$, we have the following assumption:

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

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 the solution of $Ax = b$. We further assume that the matrix A has a Laplacian sparse structure of $\mathcal{G}$ as in Definition 1.

Remark 1: If the agent network $\mathcal{G}$, instead of being connected, consists of several connected components, a system of linear equations $Ax = b$ with Laplcian sparse A of $\mathcal{G}$ is indeed composed of several independent subsystems of linear equations. These subsystems of linear equations can be independently solved.

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

Example 1: We consider a network $\mathcal{G}$ composed of four agents with the Laplacian matrix $L = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}$. Then $A = \begin{pmatrix} 1 & 3.4 & 0 & 0 \\ 0.8 & 5 & -9 & 0 \\ 0 & -6.23 & -3 & 6 \\ 0 & 0 & -5 & -0.96 \end{pmatrix}$ is a matrix with Laplacian sparse structure of $\mathcal{G}$. If each agent owns one row of A, then $A_1^{(1)} = \begin{pmatrix} 1 & 3.4 \end{pmatrix}$, $A_2^{(2)} = \begin{pmatrix} 0.8 & 5 & -9 \end{pmatrix}$, $A_3^{(3)} = \begin{pmatrix} -6.23 & -3 & 6 \end{pmatrix}$, and $A_4^{(4)} = \begin{pmatrix} -5 & -0.96 \end{pmatrix}$. Correspondingly, $x^{(1)} = \begin{pmatrix} x_1 & x_2 \end{pmatrix}^T$, $x^{(2)} = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}^T$, $x^{(3)} = \begin{pmatrix} x_2 & x_3 & x_4 \end{pmatrix}^T$, and $x^{(4)} = \begin{pmatrix} x_3 & x_4 \end{pmatrix}^T$. $L_1^{(1)} = \begin{pmatrix} -1 & 1 \end{pmatrix}$, $L_2^{(2)} = \begin{pmatrix} 1 & -2 & 1 \end{pmatrix}$, $L_3^{(3)} = \begin{pmatrix} 1 & -2 & 1 \end{pmatrix}$, and $L_4^{(4)} = \begin{pmatrix} 1 & -1 \end{pmatrix}$.

If we formulate solving $Ax = b$ as a distributed optimization problem as follows

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

we then have the following lemma:

Lemma 1: Under Assumption 1 and the Laplacian sparse structure of A, solving $Ax = b$ is equivalent to solving

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

where $x_j^{(i)}$ is agent i's local estimate on $x_j,\ j \in N_i \cup \{i\}$.

Proof: If $x^\star$ is a solution of $Ax = b$, 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), where $x^{(i)} = \left(x_j^\star\right)_{j \in N_i \cup \{i\}}$ is a collection of the entries in $x^\star$ corresponds to the neighbors of agent i and itself.

If $x^{(1)}, x^{(2)}, \cdots, x^{(\bar{m})}$ form a solution of (5), denote $x^\star = \begin{pmatrix} x_1^{(1)T} & x_2^{(2)T} & \cdots & x_{\bar{m}}^{(\bar{m})T} \end{pmatrix}^T$. For any agent i, we have $x_j^\star = x_j^{(j)} = x_j^{(i)}, \forall j \in N_i$. Notice that A has Laplacian sparse structure, so $x^{(i)} = \left(x_j^\star\right)_{j \in N_i \cup \{i\}}$ and thus $A_i x^\star = A_i^{(i)} x^{(i)} = b_i$. ■

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

Lemma 2: Let

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

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

$$\min\ f_p. \tag{7}$$

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

Next, we will propose a communication-efficient distributed algorithm to solve $Ax = b$ by solving (7) based on the gradient descent method.

B. Communication-efficient Algorithm Based on Gradient Descent Method

In this subsection, we propose a communication-efficient distributed algorithm to solve $Ax = b$ based on the gradient descent method with a constant step size and prove its convergence at a geometric rate. For simplicity, we suppose that $\dim(x_i^{(i)}) = 1, \ \forall i \in V$, i.e., the state owned by every agent is a scalar. It is not difficult to extend the results to the cases when different agents own states of different dimensions.

We first apply the gradient descent method with a constant step size α to (7). For the gradient of f_p, we have that

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

where $e_j^{(i)} = \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}^T$ with the position of 1 located at agent i's local index of j. Also, $\dim(e_j^{(i)}) = \sum\limits_{j \in N_i \cup \{i\}} \dim(x_j^{(i)}) = \dim(x^{(i)})$.

Let

$$\boldsymbol{x} = \begin{pmatrix} x^{(1)T} & x^{(2)T} & \cdots & x^{(\bar{m})T} \end{pmatrix}^T \tag{9}$$

and

$$\nabla f(\boldsymbol{x}) = \begin{pmatrix} (\frac{\partial f_p}{\partial x^{(1)}})^T & (\frac{\partial f_p}{\partial x^{(2)}})^T & \cdots & (\frac{\partial f_p}{\partial x^{(\bar{m})}})^T \end{pmatrix}^T. \tag{10}$$

From the gradient descent method

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

we obtain that for agent i,

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

Remark 2: The communication from agent j to agent i is only the estimate of the states of agent i, e.g. $x_i^{(j)}$, and agent j, $x_j^{(j)}$, by agent j rather than the states of all agents. So (11) reduces significant communication compared with the algorithms in [1]–[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 in finite time or at a linear rate to the optimal point of (7) if $0 < \alpha < \frac{2}{\lambda_{\max}(\nabla^2(f_p))}$, where $\nabla^2(f_p)$ is the Hessian of f_p in (6) and $\lambda_{\max}(\cdot)$ represents the maximal eigenvalue. Let

$$x^\star(k) = \begin{pmatrix} x_1^{(1)T}(k) & x_2^{(2)T}(k) & \cdots & x_{\bar{m}}^{(\bar{m})T}(k) \end{pmatrix}^T. \tag{12}$$

Then $x^\star(k), \ k = 1, 2, 3, \cdots$ converges to a solution of $Ax = b$.

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 [19]. But when $Ax = b$ has multiple solutions, f_p in (6) is not strongly convex and thus the results in [19] cannot be used to prove Theorem 1. But the proof of the convergence of (11) when $Ax = b$ has more than one solution is omitted due to space limitation.

C. Communication-efficient Algorithm Based on Approximated Newton Method

A communication-efficient distributed algorithm was proposed in the previous subsection, but we find in some simulation examples that it is very slow. So in this subsection, we will propose an approximated Newton method to accelerate the process to solve $Ax = b$, i.e., minimize (6). For simplicity, we suppose that $\dim(x_i^{(i)}) = 1, \ \forall i \in V$, i.e., the state owned by every agent is a scalar. It is not difficult to extend the results to the cases where different agents own states of different dimensions.

Remark 3: We can find an approximated Newton method to solve distributed optimization problems in [20], which requires the local objective functions to be strongly convex. But in our problem, the local objective functions $\frac{1}{2}\|A_i^{(i)} x^{(i)} - b_i\|$ are not strongly convex. Thus, the analysis in [20] does not apply to the problem in this paper.

The centralized Newton method to minimize f_p is as

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

For simplicity, we denote $H = \nabla^2 f_p(x(k))$ and $g(k) = \nabla f_p(x(k))$. Notice that H is a constant matrix for f_p in (6).

As in (8), the gradient of f_p is

$$\begin{aligned} \frac{\partial f_p}{\partial x^{(i)}} =& A_i^{(i)T}(A_i^{(i)} x^{(i)} - b_i) \\ &+ \sum_{j \in N_i} [(x_i^{(i)} - x_i^{(j)}) \ e_i^{(i)} + (x_j^{(i)} - x_j^{(j)}) \ e_j^{(i)}]. \end{aligned}$$

For the Hessian of f_p, we have that

$$\begin{aligned} \frac{\partial^2 f_p}{\partial x^{(i)2}} &= A_i^{(i)T} A_i^{(i)} + \sum_{j \in N_i} e_i^{(i)} e_i^{(i)T} + e_j^{(i)} e_j^{(i)T} \\ &= A_i^{(i)T} A_i^{(i)} + \text{diag}(|L_i^{(i)}|) \end{aligned}$$

and

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

where diag$(|L_i^{(i)}|)$ is a diagonal matrix whole diagonal entries are the absolute values of $L_i^{(i)}$, the nonzero entries of the ith row of Laplacian matrix L.

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 $D_i = \gamma \frac{\partial^2 f_p}{\partial x^{(i)2}}$, where $\gamma > 1$ is a constant, and $D = \text{diag}(D_1, \cdots, D_{\bar{m}})$. Also, let $F = H - D$. Then we have that

$$F_{ij} = \begin{cases} (1-\gamma)(A_i^{(i)T} A_i^{(i)} + \text{diag}(|L_i^{(i)}|)), i = j, \\ -e_i^{(i)} e_i^{(j)T} - e_j^{(i)} e_j^{(j)T}, & j \in N_i, \\ 0, & \text{otherwise} \end{cases} \quad (13)$$

and we can see that F is Laplacian sparse.

Then, $H = D + F$ and

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

Then the distributed approximated Newton method to solve $Ax = b$ is

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - (D^{-1} - D^{-1} F D^{-1}) g(k), \quad (14)$$

where $\boldsymbol{x}$ is defined in (9). As matrix F is Laplacian sparse from (13), then for agent i, (14) becomes

$$\begin{aligned} x_i(k+1) = & x_i(k) - (D_i^{-1} g_i(k) \\ & - D_i^{-1} \sum_{j \in N_i \cup \{i\}} F_{ij} D_j^{-1} g_j(k)). \end{aligned} \quad (15)$$

Then we have the following result on the performance of (14) and (15):

Theorem 2: Suppose that the system of linear equations $Ax = b$ has a unique solution and A is Laplacian sparse. Then under Assumption 1, $x^{(i)}(k)$, $\forall i \in V$ in (15) converges at a linear rate to the optimal solution of (7). $x^\star(k)$, $k = 1, 2, 3, \cdots$ in (12) converges to the solution of $Ax = b$.
The proof of Theorem 2 is omitted due to space limitation.

D. Comparison Between the Two Algorithms

We make a brief comparison between the gradient descent based method (11) and the approximated Newton based method (15) on the convergence rate, communication burdens, and computation costs in this subsection.

In terms of the convergence rate, although a quantitative convergence rate is not available for either method, we find through simulations that the approximated Newton based method (15) converges much faster than the gradient descent based method (11).

The communication burden of the approximated Newton based algorithm doubles that of the gradient descent based algorithm in each iteration. We already know that agent i in (11) only requires two parts, i.e., $x_i^{(j)}$ and $x_j^{(j)}$ through the communication link (j, i). In (15), let $v^{(j)} = D_j^{-1} g_j(k)$. When $j \in N_i$, as $F_{ij} = -e_i^{(i)} e_i^{(j)T} - e_j^{(i)} e_j^{(j)T}$, agent i only requires $v_i^{(j)}$ and $v_j^{(j)}$ besides $x_i^{(j)}$ and $x_j^{(j)}$. So the algorithm based on approximated Newton method requires twice as much communication as that based on the gradient descent method. But 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 approximated Newton method is still communication-efficient. For example, in three-phase power flow problems, dim$(x_i^{(j)})$, dim$(x_i^{(j)})$, dim$(v_i^{(j)})$, and dim$(v_j^{(j)})$ are all at most three, so what is transmitted from agent j to i is at most twelve scalars no matter how large the whole system is in the approximated Newton based method.

The approximated network method (15) has a higher computation cost than the gradient descent based method (11).

First, both algorithms need to calculate $g_i(k)$ and then do a subtraction of two dim$(x^{(i)})$th order vectors. In addition, agent i in (15) needs to do more computations. These computations include

1) computation of a dim$(x^{(i)})$th order symmetric matrix, $D_i = A_i^{(i)T} A_i^{(i)} + \text{diag}(|L_i^{(i)}|)$, which can be done at the initialization step,

2) the inverse of a dim$(x^{(i)})$th order symmetric matrix, D_i^{-1}, which can be done at the initialization step,

3) the multiplication of D_i^{-1} and $g_i(k)$ per iteration,

4) summation of $|N_i|$ vectors, $v_i^{(j)}$, with dimension dim$(x_i^{(i)})$ after receiving the information from its neighbors per iteration.

5) a subtraction of two vectors with dimension dim$(x^{(i)})$ per iteration.

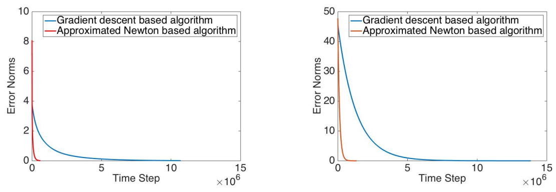
We can see that the extra computation burdens of (15) depends on dim$(x^{(i)})$ and dim$(x_i^{(i)})$. In many problems, dim$(x^{(i)})$ and dim$(x_i^{(i)})$ are usually very small in spite of the scale of the whole system. For example, in three-phase power flow problems, dim$(x_i^{(i)})$ is at most three and dim$(x^{(i)})$ is at most $3|N_i|$, which is also small due to the sparse physical connections in the real world systems. But as (15) usually converges much faster than (11), the total communication and computation costs of (15) may be lower than those of (11).

IV. Simulation Results

In this section, we conduct simulation studies to illustrate the effectiveness of the communication-efficient distributed algorithms (11) and (15).

The first simulation set up is based on solving one step of update equations in the Newton-Raphson approach to solve the power flow problem in (1). The testing system is the IEEE 13-bus test feeder.

We treat each bus as an agent and assume that the communication network and the physical network share the same topology. The state vector of the whole system consists of the magnitudes and phase angles of the complex voltages of all buses. As the buses may be of one-phase, two-phase, or three-phase, different agents have state vectors of differently dimensional.



(a) (b)

Fig. 1. Norms of Errors between $x_i^{(i)}(k)$ and Accurate Solutions

We can see that in Section II-C the Jacobian matrix consists of four blocks $J = \begin{pmatrix} J_{p\theta} & J_{p|v|} \\ J_{q\theta} & J_{q|v|} \end{pmatrix}$. Every block has the same sparse structure as the Ybus 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 (4) with the methods in (11) and (15).

The simulation results are shown in Fig. 1(a). The step size α in (11) is chosen as 0.00002, and we observe that if we choose $\alpha = 0.00003$, (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 in (15) is selected as 2.

The second simulation set up is based on solving a random generated system of linear equations with the Laplacian sparse structure of its communication topology. And the communication graph is a 30th-order undirected ring. The simulation results are shown in Fig. 1(b). The step size α in (11) is chosen as 0.1, and we observe that if we choose $\alpha = 0.15$, (11) diverges. So $\alpha = 0.1$ might be very close to the largest step size that makes (11) converge in this example. The coefficient γ in (15) is selected as 2.

From the two simulation examples, we can see that both algorithms converges to the solution of Eq.(4), and the approximated Newton based algorithm converges much faster than that of the gradient descent based algorithm (11).

V. Conclusions

Two communication-efficient distributed algorithms have been proposed in this paper. The first algorithm was based on gradient descent method while the second one was based on an approximation to the Newton method. The first algorithm only required two parts of the state vector to be communicated instead of the whole state vector, while the second algorithm required twice as much communication as the first one. In the first algorithm, the two parts communicated through each communication link were the states owned by the two agents connected by the communication link. In the second algorithm, two parts of the gradient vector were also transmitted besides the two parts of the state vector. The second algorithm converged faster than the first one at the price of heavier computation and communication burdens. It was proven that both algorithms converged at a linear rate. Future works may include convergence rate analysis and asynchronous updates.

References

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

[2] P. Wang, W. Ren, and 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)*, Dec 2016, pp. 5220–5225.

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

[4] P. Wang, W. Ren, and 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*, vol. PP, no. 99, pp. 1–1, 2018.

[5] F. Pasqualetti, R. Carli, and F. Bullo, “Distributed estimation via iterative projections with application to power network monitoring,” *Automatica*, vol. 48, no. 5, pp. 747 – 758, 2012.

[6] F. Pasqualetti, R. Carli, A. Bicchi, and F. Bullo, “Distributed estimation and detection under local information,” *IFAC Proceedings Volumes*, vol. 43, no. 19, pp. 263 – 268, 2010, 2nd IFAC Workshop on Distributed Estimation and Control in Networked Systems.

[7] F. Pasqualetti, R. Carli, and 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)*, Oct 2011, pp. 469–474.

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

[9] X. Wang, S. Mou, and D. Sun, “Improvement of a distributed algorithm for solving linear equations,” *IEEE Transactions on Industrial Electronics*, vol. 64, no. 4, pp. 3113–3117, April 2017.

[10] J. Liu, S. Mou, and A. S. Morse, “Asynchronous distributed algorithms for solving linear algebraic equations,” *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 372–385, Feb 2018.

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

[12] G. Shi, B. D. O. Anderson, and U. Helmke, “Network Flows that Solve Linear Equations,” *ArXiv e-prints*, Oct. 2015.

[13] B. D. O. Anderson, S. Mou, A. S. Morse, and U. Helmke, “Decentralized gradient algorithm for solution of a linear equation,” *ArXiv e-prints*, Sept. 2015.

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

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

[16] J. Zhou, X. Wang, S. Mou, and B. D. O. Anderson, “Finite-time distributed linear equation solver for minimum \$l_1\$ norm solutions,” *CoRR*, vol. abs/1709.10154, 2017. [Online]. Available: http://arxiv.org/abs/1709.10154

[17] X. Wang, S. Mou, and B. D. O. Anderson, “A double-layered framework for distributed coordination in solving linear equations,” *CoRR*, vol. abs/1711.10947, 2017. [Online]. Available: http://arxiv.org/abs/1711.10947

[18] A. Bergen and V. Vittal, *Power Systems Analysis*. Prentice Hall, 2000.

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

[20] A. Mokhtari, Q. Ling, and A. Ribeiro, “Network newton distributed optimization methods,” *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 146–161, Jan 2017.