Distributed Newton Optimization With Maximized Convergence Rate

Damián Marelli, Yong Xu, Minyue Fu, Fellow, IEEE, and Zenghong Huang

Abstract—The distributed optimization problem is set up in a collection of nodes interconnected via a communication network. The goal is to find the minimizer of a global objective function formed by the sum of local functions known at individual nodes. A number of methods, having different advantages, are available for addressing this problem. The goal of this article is to achieve the maximum possible convergence rate. As the first step toward this end, we propose a new method, which we show converges faster than other available options. As the second step toward our goal, we complement the proposed method with a fully distributed method for estimating the optimal step size that maximizes the convergence rate. We provide theoretical guarantees for the convergence of the resulting method in a neighborhood of the solution. We present numerical experiments showing that, when using the same step size, our method converges significantly faster than its rivals. Experiments also show that the distributed step-size estimation method achieves an asymptotic convergence rate very close to the theoretical maximum.

Index Terms—Distributed algorithms, local convergence, Newton method, step size.

I. INTRODUCTION

A networked system is a web of intelligent sensing and computing devices connected via a communication network. Its main goal is to carry out a computational task in a distributed manner, by executing a cooperative strategy over all the nodes of the network without centralized coordination. The design of distributed algorithms is constrained by the fact that each node is limited in computational power and communication bandwidth. Distributed algorithms are available for parameter estimation [1], Kalman filtering [2], control [3], and optimization [4], etc.

Manuscript received 25 September 2021; accepted 12 October 2021. Date of publication 27 October 2021; date of current version 27 September 2022. This work was supported in part by the Argentinian Agency for Scientific and Technological Promotion under Grant PICT-201-0985, in part by the National Natural Science Foundation of China under Grant 62121004, Grant 61633014, Grant 61876041, and Grant U1701264, in part by the Local Innovative and Research Teams Project of Guangdong Special Support Program under Grant 2019BT02X353, in part by Australian Research Council under grant DP200103507, and in part by the Key Area Research and Development Program of Guangdong Province under Grant 2021B0101410005. Recommended by Associate Editor Javad Lavaei. (Corresponding author: Yong Xu.)

Damián Marelli is with the Provincial Key Laboratory of Intelligent Decision and Cooperative Control, School of Automation, Guangdong University of Technology, Guangzhou 510006, China, and also with the French Argentine International Center for Information and Systems Sciences, National Scientific and Technical Research Council, Rosario 2000, Argentina (e-mail: damian.marelli@gmail.com).

Yong Xu and Zenghong Huang are with the Provincial Key Laboratory of Intelligent Decision and Cooperative Control, School of Automation, Guangdong University of Technology, Guangzhou 510006, China (e-mail: xyong8090@163.com; zenghong9527@foxmail.com).

Minyue Fu is with the School of Electrical Engineering and Computing, University of Newcastle, Callaghan, NSW 2308, Australia (e-mail: minyue.fu@newcastle.edu.au).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TAC.2021.3123244.

Digital Object Identifier 10.1109/TAC.2021.3123244

The goal of a distributed optimization method is to minimize an objective function formed by a sum of local functions that are only known by each node [4], [5]. It finds applications in power systems, sensor networks, smart buildings, and smart manufacturing, etc. The available distributed optimization methods can be classified according to different criteria. In the following, we describe those criteria used in this article.

One classification criterion is between sequential and simultaneous methods. In a sequential method, nodes take turns to tune its local variables using its local function as well as the information received from other nodes [6]. The main drawback of this approach is that it does not scale well for large networks, since many turns are needed to guarantee that all nodes are visited. Also, for a fully distributed implementation, a distributed mechanism is needed to guarantee that each node is regularly visited in the sequence. A popular approach within this method are algorithms based on alternating direction method of multipliers [7]. In contrast to sequential methods, a simultaneous method iterates over a computation step, in which all nodes carry out local computations, and a communication step, in which nodes communicate information with their neighbors. These two steps typically depend on the number of local neighbors and not on the network size. In this way, the simultaneous method avoids the scalability problems of the sequential method.

Another classification criterion is between methods using first-order derivatives and those using derivatives of the second order. There is a vast literature on first-order methods, a survey of which is given in [4]. As with centralized optimization methods, the advantage of the first-order distributed methods is that they are simpler to implement and analyze. However, the second-order distributed methods converge much faster, leading to less computational and communication requirements.

Upon convergence, a distributed optimization method needs to guarantee that all nodes obtain the same optimal value. Another classification criterion is based on how the distributed method guarantees this inter-node matching property. One approach consists in adding a constraint to the optimization program forcing this property. In [8] and [9], the resulting constrained optimization problem is solved by adding a penalization term. This gives an approximate solution that becomes exact as the step size used in the optimization recursions decreases to zero. This has the disadvantage of slowing down convergence. This is avoided in [10] by solving the constrained optimization program via its dual. However, at each iteration, each node needs to solve a local optimization problem needed to evaluate the Lagrange dual function. These two approaches were also considered in [11]. All these methods require the local functions at each node to be strongly convex, which is somehow a strong requirement, since these local functions typically represent partial information about the variables to be optimized. They also require, at each iteration, inverting a matrix using a recursive inversion formula. This requires running subiterations, each involving a computation/communication step, between every two main iterations. These drawbacks are avoided by forcing the inter-node matching using average consensus. By doing so, the resulting optimization program is unconstrained. As pointed out in [12], an additional advantage of this approach is that the large literature available for average consensus permits guaranteeing its robustness to asynchronous communications.
II. PROBLEM STATEMENT

Notation 1: The set of natural and real numbers are denoted by \( \mathbb{N} \) and \( \mathbb{R} \), respectively. For a vector \( x \in \mathbb{R}^N \), we use \( ||x|| \) to denote its 2-norm and \( ||x||_p = (\sum x^p)^{1/p} \) to denote its \( p \)-norm. For a matrix \( A \in \mathbb{R}^{N \times N} \), we use \( ||A|| \) to denote its Frobenius norm, \( ||A||_2 \) to denote its operator norm (induced by the vector 2-norm), and \( \rho(A) \) its spectral radius. Also, \( I_N \) denotes the \( N \times N \) identity matrix and \( 1_N \) denotes the \( N \)-dimensional column vector filled with ones.

To simplify the notation, we often omit the subscript in \( I_N \) and \( 1_N \) when the dimension can be clearly inferred from the context. We use \( \text{col}(x_1, \ldots, x_j) \) to denote the column vector formed by stacking the elements \( x_1, \ldots, x_j \) and \( \otimes \) to denote the Kronecker product. We use \( \leq \) to denote the nonstrict partial order on \( \mathbb{R} \), \( u \leq v \), for all \( u, v \in \mathbb{R} \), and \( < \) to denote the strict partial order corresponding to \( \leq \), i.e., \( u < v \) if \( u \leq v \) and \( u \neq v \). Finally, \( O(x) \) denotes Bachmann–Landau’s big O notation \( O(x) \) as \( x \to 0 \).

We have a network of \( I \) nodes. Node \( i \) can evaluate the function \( f_i : \mathbb{R}^N \to \mathbb{R} \) and send messages to its out-neighbors \( N_i \subseteq \{1, \ldots, I\} \) using a consensus network. The communication link from node \( i \) to node \( j \in N_i \) has time-invariant gain \( w_{ij}^{(i)} \). We assume that \( w_{ij}^{(i)} = 0 \) if \( j \notin N_i \). We also assume that the graph induced by the communication network is balanced (i.e., possibly directed) and strongly connected. This implies that matrix \( W = [w_{ij}]_{i,j=1}^I \) is doubly stochastic and primitive. As explained in [19], a consequence of this is that \( I^T W = 1_I^T \) and \( W 1_I = 1_I \). Also, for any \( x \in [x_1, \ldots, x_I] \in \mathbb{R}^I \), the sequence generated by \( x_{k+1} = W x_k \) satisfies

\[
\lim_{k \to \infty} x_k = 1_I \otimes \sum_{i=1}^I x_i^i
\]

In this article, we assume that the weights \( w_{ij}^{(i)} \) are given. However, they can be chosen to optimize different criteria, as done in [18] in the context of distributed optimization.

The goal of distributed optimization is to design a distributed method for solving the following minimization problem:

\[
x_* \in \arg\min_{x \in \mathbb{R}^N} f(x) \quad \text{with} \quad f(x) = \frac{1}{I} \sum_{i=1}^I f_i^i(x).
\]

Solving (1) using a centralized or distributed method requires certain assumptions on the objective function, e.g., convexity, quasi-convexity, everywhere positive definite Hessian matrix, etc. These assumptions may be too strong in certain applications. When none of these assumptions can be made, it is often enough to solve

\[
x_* \in \text{loc min } f(x) \quad \text{over } \mathbb{R}^N
\]

where \( \text{loc min} \) denotes the set of local minimizers of \( f \).

As mentioned in Section I, a number of distributed methods are available for solving the above-mentioned problems. In this article, we focus on the problem (2), which corresponds to the general case in which the objective function can have many minima. In Section III, we propose a new distributed method whose design aims at fast convergence. In Section IV, we complement this design by proposing a step-size selection criterion for maximizing convergence speed. In Section V, we present an example. Finally, Section VI, concludes this article.

III. PROPOSED METHOD

In this section, we introduce the basic form of the proposed distributed optimization method, i.e., with a fixed step size. In Section III-A, we introduce some background on static and dynamic average consensus. In Section III-B, we derive the proposed method, and in Section III-C, describe its differences with respect to a popular variant in literature.

A. Static and Dynamic Average Consensus

As described in Section I, we are interested in methods achieving inter-node matching of minimization parameters via average consensus. In this section, we briefly describe the options available for doing so.

Suppose that, in the network described in Section II, each node \( i \) knows a variable \( v^i \in \mathbb{V} \), \( i \in \mathbb{N} \), where \( \mathbb{V} \) is a vector space. In order to make the presentation valid in the general case, we assume that \( \mathbb{V} \) is an arbitrary vector space, i.e., each \( v^i \) can be either a scalar, a vector, a matrix, etc. The goal of static average consensus is to compute the average \( u = \frac{1}{I} \sum_{i=1}^I v^i \) in a distributed manner. This is done using the following iterations:

\[
v_k^i = \sum_{j=1}^f w_{ij}^{(i)} v_j^k
\]

initialized by \( v_0^i = v^i \) [19]. Letting \( v_k = \text{col}(v_1^k, \ldots, v_I^k) \in \mathbb{V}^I \), we can write the above-mentioned compactly as follows:

\[
v_k = W v_k
\]

Suppose now that each node \( i \) knows a time-varying sequence of variables \( u_{k}^i \in \mathbb{V} \), \( k \in \mathbb{N} \). The goal of the dynamic average consensus technique [20] is to obtain, at each \( k \), an estimate of the average \( u_k = \frac{1}{I} \sum_{i=1}^I u_{k}^i \). This is done as follows: Suppose that at time \( k \) node \( i \) knows an estimate \( u_{k-1}^i \) of \( u_{k-1} \). It then transmits the following message:

\[
s_k^i = u_{k-1}^i + v_k^i - u_{k-1}^i
\]
to its out-neighbors. On reception, node $i$ obtains
\[ u_k^i = \sum_{j=1}^{I} w^{i,j} s_k^j. \]  
(5)

The above-mentioned iterations are initialized by $s_k^i = v_i^k$. We can combine (4) and (5) in two ways, namely, in message form
\[ s_{k+1}^i = \sum_{j=1}^{I} w^{i,j} s_k^j + v_{k+1}^i - v_k^i, \]
or in estimate form
\[ u_{k+1}^i = \sum_{j=1}^{I} w^{i,j} (u_k^j + v_{k+1}^j - v_k^j). \]

B. Proposed Method

The essential idea consists in distributing the Newton iterations
\[ x_{k+1} = x_k - \alpha_k \left[ \nabla^2 f(x_k) \right]^{-1} \nabla f(x_k) \]  
(6)
where $\alpha_k$ is called the step size at time $k$. To this end, we make use of the dynamic average consensus technique [20]. Let
\[ \tilde{f}(x_1, \ldots, x_I) = \frac{1}{I} \sum_{i=1}^{I} f^i(x_k^i). \]

We can obtain an estimate $g_k^i$ of $\nabla \tilde{f}(x_1^i, \ldots, x_I^i) = \frac{1}{I} \sum_{i=1}^{I} \nabla f^i(x_k^i)$, at each node $i$, by applying dynamic average consensus on the inputs $\nabla f^i(x_k^i)$. This yields the following recursions written in estimate form:
\[ g_k^i = \sum_{j=1}^{I} w^{i,j} [g_{k-1}^j + \nabla f^j(x_k^j) - \nabla f^j(x_{k-1}^j)]. \]

We can do the same to obtain an estimate $H_k^i$ of the Hessian $\nabla^2 \tilde{f}(x_1^i, \ldots, x_I^i) = \frac{1}{I} \sum_{i=1}^{I} \nabla^2 f^i(x_k^i)$. This gives
\[ H_k^i = \sum_{j=1}^{I} w^{i,j} \left[ H_{k-1}^j + \nabla^2 f^j(x_k^j) - \nabla^2 f^j(x_{k-1}^j) \right]. \]

It is easy to see that $\tilde{f}(x_1, \ldots, x_I) = f(x)$. Hence, if
\[ x_k^i \approx x_k \quad \text{for all} \quad i \in \{1, \ldots, I\} \]  
(7)
and some $x_k$ then $g_k^i$ and $H_k^i$ are estimates of $\nabla f(x_k)$ and $\nabla^2 f(x_k)$, respectively. Thus, in principle, each node $i$ could use $g_k^i$ and $H_k^i$, in place of $\nabla f(x_k)$ and $\nabla^2 f(x_k)$, to locally carry out the iterations (6). This would yield, at node $i$, the following sequence of estimates of $x_k^i$:
\[ \tilde{x}_{k+1}^i = x_k^i - \alpha_k \left[ H_k^i \right]^{-1} g_k^i. \]
But the above requires (7) to hold, with $x_k^i$ replaced by $\tilde{x}_{k+1}^i$. In order to enforce that, once again we apply dynamic average consensus on the inputs $\tilde{x}_{k+1}^i$. Writing the result in form we obtain
\[ x_{k+1} = \sum_{j=1}^{I} w^{i,j} x_{k+1}^j + \tilde{x}_{k+1}^i - \tilde{x}_k^i = \sum_{j=1}^{I} w^{i,j} x_k^j - \alpha_k \left[ H_k^i \right]^{-1} g_k^i. \]

Finally, since the approximations $H_k^i$ to the Hessian are obtained via dynamic average consensus on the local Hessian matrices $\nabla^2 f^i(x_k^i)$, and the latter may fail to be positive definite, some mechanism is required to guarantee that $H_k^i$ is positive definite. To do so, we let $\beta > 0$ and use a map $B$ to guarantee that $B(H_k^i) \geq \beta^{-1}I$. The map $B$ is defined as follows. Let $H \in \mathbb{R}^{N}$ be symmetric and $H = U \Lambda U^T$ be its spectral decomposition, with $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$. Let $\tilde{\Lambda} = \text{diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_N)$ with $\hat{\lambda}_i = \lambda_i$ if $\lambda_i \geq \beta^{-1}$ and $\hat{\lambda}_i = \beta^{-1}$ otherwise. Then
\[ B(H) = U \tilde{\Lambda} U^T. \]

The question then arises as to how to choose the parameter $\beta$. This is given in Assumption 1 of our main results given in Section IV.

To summarize the above, the proposed algorithm is given by the following recursions:
\[ x_{k+1}^i = \sum_{j=1}^{I} w^{i,j} x_k^j - \alpha_k B \left( H_k^i \right) g_k^i \]
\[ g_{k+1}^i = \sum_{j=1}^{I} w^{i,j} \left[ g_k^j + \nabla f^j(x_{k+1}^j) - \nabla f^j(x_k^j) \right] \]
\[ H_{k+1}^i = \sum_{j=1}^{I} w^{i,j} \left[ H_k^j + \nabla^2 f^j(x_{k+1}^j) - \nabla^2 f^j(x_k^j) \right]. \]

which are initialized by
\[ x_1^i = x_1^\text{init} \quad \text{and} \quad g_1^i = \nabla f^i(x_1^\text{init}) \quad \text{and} \quad H_1^i = \nabla^2 f^i(x_1^\text{init}) \]  
(11)

Remark 1: Notice that the information exchanged by each node at each time step does not grow with the network size, as it depends on the number of out-neighbors of each node. This property is common to all simultaneous, second-order methods based on average consensus [12]–[16].

C. Comparison With Similar Algorithms

As mentioned previously, the proposed algorithm (8)–(10) is a variant of the algorithm used in [12]–[16]. The latter differ from (8)–(10) in essentially two aspects. The first one is that the consensus is not done on the parameters $x_k^i$, This means that (8) is replaced by
\[ x_{k+1}^i = x_k^i - \alpha_k B \left( H_k^i \right) \tilde{g}_k^i \]  
(12)
Together with (9) and (10), (12) forms an algorithm that, for latter reference, we refer to as Algorithm A.

The second difference consists in using the following transformation of (6):
\[ x_{k+1} = (1 - \alpha) x_k + \alpha_k \left( \nabla^2 f(x_k) \right)^{-1} \ell(x_k) \]  
(13)
where $\ell(x) = \nabla^2 f(x) x - \nabla f(x)$. Using dynamic average consensus, we can estimate $\ell(x_k)$ at each node using
\[ \ell_k^i = \sum_{j=1}^{I} w^{i,j} \left[ \ell_k^j + \ell^j(x_{k+1}^j) - \ell^j(x_k^j) \right] \]
\[ \ell_{k+1}^i = \sum_{j=1}^{I} w^{i,j} \left[ \ell_k^j + \ell^j(x_{k+1}^j) - \ell^j(x_k^j) \right] \]
\[ \ell_{k+1}^i = \sum_{j=1}^{I} w^{i,j} \left[ \ell_k^j + \ell^j(x_{k+1}^j) - \ell^j(x_k^j) \right] \]
\[ \ell_{k+1}^i = \sum_{j=1}^{I} w^{i,j} \left[ \ell_k^j + \ell^j(x_{k+1}^j) - \ell^j(x_k^j) \right] \]
\[ \ell_{k+1}^i = \sum_{j=1}^{I} w^{i,j} \left[ \ell_k^j + \ell^j(x_{k+1}^j) - \ell^j(x_k^j) \right] \]
\[ \ell_{k+1}^i = \sum_{j=1}^{I} w^{i,j} \left[ \ell_k^j + \ell^j(x_{k+1}^j) - \ell^j(x_k^j) \right] \]

where $\ell^j(x) = \nabla^2 f^j(x) x - \nabla f^j(x)$. We can then distribute iteration (13) as follows:
\[ x_{k+1}^i = (1 - \alpha \beta) \sum_{j=1}^{I} w^{i,j} x_k^j + \alpha_k B \left( H_k^i \right) \tilde{l}_k^i \]
(15)
We refer to the algorithm resulting from (10), (14), and (15) as Algorithm B.
Finally, the algorithm used in [12]–[16], apart from other minor differences, essentially consists in combining the modifications introduced by Algorithms A and B. This leads to the recursions formed by

\[
x_{k+1}^i = (1 - \alpha_k^i) x_k^i + \alpha_k^i B \left( H_k \right)^{-1} l_k^i
\]  

(16)
together with (10) and (14). We refer to it as Algorithm VZCPS, standing for the initials of the authors who proposed it.

As we show with experiments in Section V, the modifications (12) and (14)–(15), introduced by Algorithms A and B, respectively, drastically slow down convergence and can cause instability. More precisely, (12) does not guarantee the convergence of each \( x_k^i \) to \( x_* \), due to the lack of consensus on these parameters. Convergence only occurs if modification (14)–(15) is also considered, i.e., in Algorithm VZCPS, although at a much slower rate. However, a feature of the latter modification is that the first term in (15) pushes the local variables \( x_k^i \) toward zero at each iteration. This pushing is compensated by the second term, but only after consensus on the parameters \( x_k^i \) is reached. Before this happens, this zero pushing effect has a negative influence if the minimizing parameters \( x_* \) are far from zero. As we show in Section V, this can slow down convergence and even cause instability.

IV. Step-Size Selection for Fast Local Convergence

In this section, we derive a criterion for choosing the step size at each node to maximize the convergence rate in a neighborhood of the solution. In Section IV-A, we derive two state-space representations of the algorithm, which are instrumental for our analysis. In Section IV-B, we describe the design criterion. This criterion leads to an offline step-size choice, which is based on certain approximation. In order to obtain a more accurate selection, in Section IV-C, we propose a distributed adaptive method to estimate the optimal step size at each node. Finally, in Section IV-D, we analyze the local stability of the resulting distributed optimization algorithm when used in combination with the adaptive step-size selection.

A. State-Space Representation

We introduce the following required notation.

**Notation 2:** Let \( A = I_1 \otimes I_N \), \( B = A \otimes I_N \), \( a = \frac{1}{2} I_f \otimes I_N \) and \( I = - A \). Let also \( x_k = 1_f \otimes x_k \), \( x_k = \text{col}(x_k^1, \ldots, x_k^f) \). \( x_k = A x_k \) and \( \tilde{x}_k = x_k - \bar{x}_k \). We similarly define \( g_k = \text{col}(g_k^1, \ldots, g_k^f) \), \( \tilde{g}_k = A g_k \), and \( \bar{g}_k = g_k - g_k^* \) as well as \( h_k = \text{col}(H_k^1, \ldots, H_k^f) \), \( \tilde{h}_k = A h_k \), \( h_k = h_k - \bar{h}_k \), \( h_k = I_f \otimes \nabla^2 f(x_k) \), and \( h_k = I_f \otimes \nabla^2 f(x_k) \). Finally, \( \alpha = \text{diag}(\alpha_1^i, \ldots, \alpha_f^i) \otimes I_N \).

**Remark 2:** In the above-mentioned notation, \( \bar{x}_k \) is a block vector with all its subvectors equal to the average

\[
\bar{x}_k = \frac{1}{f} \sum_{i=1}^f x_k^i. \tag{17}
\]

Also, \( \bar{x}_k \) is a block vector whose ith subvector is given by \( \bar{x}_k^i = x_k^i - \bar{x}_k \). Finally, notice that \( h_k \) is a (column) vector of matrices, i.e., \( h_k \in \mathbb{R}^{I_N \times N} \).

**Notation 3:** For a block vector \( x = \text{col}(x^1, \ldots, x^f) \), let

\[
\begin{align*}
  g(x) &= \text{col} \left( \nabla f^1(x^1), \ldots, \nabla f^f(x^f) \right) \\
  h(x) &= \text{col} \left( \nabla^2 f^1(x^1), \ldots, \nabla^2 f^f(x^f) \right) \\
  \bar{h}(x) &= \text{diag}(h(x))
\end{align*}
\]

and for a block diagonal matrix \( H = \text{diag}(H^1, \ldots, H^f) \), let

\[
D(H) = \text{diag} \left( B \left( H^1 \right), \ldots, B \left( H^f \right) \right).
\]

Let also \( H_k = \text{diag}(h_k) \) and \( B_k = \mathcal{B}(H_k) \). Finally, let \( W = [w^{i,j}]_{i,j=1}^f \) and \( W = W \otimes I_N \).

Using the above-mentioned notation, we can write (8)–(10) in the following block state-space form:

\[
x_{k+1} = W x_k - \alpha_k B_k^{-1} g_k \tag{18}
\]

\[
g_{k+1} = W [g_k + g(x_{k+1}) - g(x_k)] \tag{19}
\]

\[
h_{k+1} = W [h_k + \bar{h}(x_{k+1}) - h(x_k)] \tag{20}
\]

A problem of the above-mentioned model for studying stability is that \( \rho(W) = 1 \). Our next step is to transform (18)–(20) into an equivalent model which avoids this drawback.

**Notation 4:** Let \( \bar{g}(x) = A g(x) \) and \( \bar{h}(x) = A h(x) \). Let also \( \bar{g}(x) = \text{diag}(\bar{g}(x)) \), \( \bar{H}_k = \text{diag}(h_k) \), and \( \bar{W} = W - A \).

**Lemma 1:** The following equivalences hold

\[
\bar{g}_{k+1} = \bar{g}(x_k), \quad \bar{h}_k = \bar{h}(x_k), \quad \bar{H}_k = \bar{h}(x_k) \tag{21}
\]

**Proof:** Since matrix \( W \) is doubly stochastic, it follows that \( AW = A \). Then, from (19)

\[
\bar{g}_{k+1} = A g_{k+1} + \bar{g}(x_{k+1}) - \bar{g}(x_k) \tag{22}
\]

The first equation in (21) then follows from (22) since \( g_1 = A g_1 = A g(x_1) = g(x_1) \). The other two equations follow using the same argument.

Using Lemma 1, we can write model (18)–(20) as follows:

\[
\begin{align*}
  \bar{x}_{k+1} &= \bar{x}_k - a_k \alpha \bar{H}_k^{-1} \bar{g}_k \tag{23} \\
  \bar{h}_{k+1} &= \bar{W} \bar{x}_k - \bar{I} \alpha \bar{H}_k^{-1} \bar{g}_k \tag{24} \\
  \bar{H}_{k+1} &= \bar{W} [\bar{g}_k + g(x_{k+1}) - g(x_k)] \tag{25} \\
  \bar{h}_{k+1} &= \bar{W} [\bar{h}_k + \bar{h}(x_{k+1}) - \bar{h}(x_k)] \tag{26}
\end{align*}
\]

where \( g_k = \bar{g}(x_k) + \bar{g}_k \), \( B_k = \mathcal{B}(\bar{h}(x_k) + \bar{H}_k) \), \( \bar{x}_k = \bar{x}_k + \bar{x}_k \), and \( \bar{x}_k = 1_f \otimes \bar{x}_k \).

B. Offline Step-Size Selection Criterion

We introduce the following notation.

**Notation 5:** Let \( \bar{x}_k = x_k - x_k \), \( \bar{h}_k = h_k - \bar{h}_k \), \( \xi_k = (\bar{x}_k, \bar{h}_k, \bar{H}_k) \) and \( \| \xi_k \|^2 = \| \bar{x}_k \|^2 + \| \bar{h}_k \|^2 + \| \bar{H}_k \|^2 \). Let also \( \Phi, \Psi : \mathbb{R} \to \mathbb{R} \) be the linear maps with matrix representation

\[
\Phi = \begin{bmatrix} W & 0 & 0 \\
W \bar{g}(x_k) (W - I) & 0 & 0 \\
W \bar{H}(x_k) H_k^{-1} (W - I) \end{bmatrix}
\]

\[
\Psi = \begin{bmatrix} 0 & H_k^{-1} & 0 \\
0 & W \bar{g}(x_k) H_k^{-1} & 0 \\
0 & W \bar{H}(x_k) H_k^{-1} & 0 \end{bmatrix}
\]

where \( \Phi(x_k)(\mathbb{M}) \) denotes the linear operator \( y \mapsto \Phi(x_k)(My) \), with \( \Phi(x_k) \) denoting the Fréchet derivative of \( \Phi \) at \( x_k \).

Our first result is given in Proposition 1. It states the local linear dynamics in a neighborhood of the local optimum \( x_* \).

**Assumption 1:** \( \| \nabla^2 f(x_*) \|^{-1} \leq \beta \).

**Proposition 1:** Under Assumption 1, if \( \alpha_k^i = \alpha \), for all \( k \in \mathbb{N} \) and \( i = 1, \ldots, I \), then

\[
\xi_{k+1} = (\Phi + \alpha \Psi) \xi_k + O \left( \| \xi_k \|^2 \right). \tag{27}
\]

**Proof:** With some abuse of notations, we use \( \xi_{k+1}(\xi_k) \) to denote the map \( \xi_k \mapsto \xi_{k+1} \) induced by (18)–(20). Doing a Taylor expansion
of this map, using Fréchet derivatives, around \( \xi_* \equiv (0, 0, 0) \), we obtain
\[
\xi_{k+1}(\xi_* ) = \xi_{k+1}(\xi_* ) + \mathcal{D}\xi_{k+1}(\xi_* ) (\xi_k) + O \left( \|\xi_k\|^2 \right). 
\]
Clearly, \( \xi_{k+1}(\xi_* ) = 0 \). Also
\[
\mathcal{D}\xi_{k+1}(\xi_* ) (\xi_k) = \mathcal{D}\xi_{k+1}(\xi_* ) (\xi_k) (\xi_{k+1}(\xi_* ) (\xi_k), \mathcal{D}\xi_{k+1}(\xi_* ) (\xi_k)) 
\]
The result then follows since
\[
\mathcal{D}\xi_{k+1}(\xi_* ) (\xi_k) = W^k - \alpha H^{-1}_1 g_k, \\
\mathcal{D}\xi_{k+1}(\xi_* ) (\xi_k) = W [g_k + \delta(x_\sigma) (\mathcal{D}\xi_{k+1}(\xi_* ) (\xi_k) - \delta(x_\sigma))] 
= W\delta(x_\sigma) (W - I) x_k 
+ W (I - \delta(x_\sigma) \alpha H^{-1}_1) g_k 
\]
\[
\mathcal{D}\xi_{k+1}(\xi_* ) (\xi_k) = W \mathcal{D}\delta(x_\sigma) ((W - I) x_k) 
+ W \mathcal{D}\delta(x_\sigma) (\alpha H^{-1}_1 g_k + W H). 
\]

The first-order local dynamics (1) reveals that \( \mathcal{D}\xi_{k+1} \) does not act as an input of neither \( \xi_k \) nor \( g_k \). Hence, to study the local convergence speed, we can focus on the local dynamics of the pair \( (\xi_k, g_k) \). This is determined by the following matrix:
\[
\Gamma(\alpha) = \begin{bmatrix}
W & 0 \\
W\delta(x_\sigma) (W - I) & W
\end{bmatrix} + \alpha \begin{bmatrix}
0 & -H^{-1}_1 \\
0 & -W\delta(x_\sigma) H^{-1}_1
\end{bmatrix}. 
\]
Clearly, when \( \alpha = 0 \), the spectrum of \( \Gamma(\alpha) \) consists of the eigenvalues of \( W \), each having multiplicity \( 2N \). Our main result describes the behavior of each of these eigenvalues when \( \alpha \) is small.

**Theorem 1:** Let \( \mu_0 \in \sigma(W) \) and \( u \) and \( v \) be, respectively, a right and left eigenvector of \( W \) associated with \( \mu_0 \). Under Assumption 1, for \( \alpha > 0 \), there exists \( \mu \in \sigma(\Gamma(\alpha)) \) and \( y \in \mathbb{R}^N \), with \( \|y\| = 1 \), satisfying
\[
\mu^2 - \mu_0 (2 - \alpha s) \mu + \mu_0 (\mu_0 - \alpha s) \simeq 0 
\]
where \( s = y^T R y \), with
\[
R = \frac{1}{\sigma} \sum_{i=1}^{N} v_i u_i \nabla^2 f_i (x_\sigma) \left[ \nabla^2 f (x_\sigma) \right]^{-1}. 
\]

**Proof:** Let \( \Gamma(\alpha) v = \mu v \). We then have
\[
\mu x = W x - \alpha H^{-1}_1 g, \\
\mu g = W\delta(x_\sigma) (W - I) x + (W - \alpha W\delta(x_\sigma) H^{-1}_1) g. 
\]

Then, since \( H_1 \) and \( W \) commute, we obtain \( -\alpha H^{-1}_1 (\mu I - W) x = g \). Let \( u = -\alpha H^{-1}_1 x \). Then, \( (\mu I - W) u = g \) and
\[
[\mu I - (W - \alpha \delta(x_\sigma) H^{-1}_1)] (\mu I - W) u = -\alpha \delta(x_\sigma) H^{-1}_1 (W - I) u. 
\]
Letting \( M = \delta(x_\sigma) H^{-1}_1 \), we obtain
\[
[\mu I - W + \alpha WM] (\mu I - W) u = -\alpha WM (W - I) u 
\Rightarrow (\mu I - W)^2 u = \alpha (1 - \mu) WMu 
\]
which means that \( \alpha (1 - \mu) \) is a generalized eigenvalue of the matrix pair \( (\mu I - W)^2, WM \).

Let \( u \) and \( v \) be, respectively, right and left eigenvectors of \( W \), associated with \( \mu_0 \). We have \( (\mu_0 I - W)^2 u = 0 \). It then follows from \( [21, \text{Lemma 6}] \) that
\[
\alpha (1 - \mu) \simeq \frac{2}{\mu_0 (2 - \alpha s) \mu + \mu_0 (\mu_0 - \alpha s)} 
\]
Clearly, \( u = u \otimes y \) and \( v = v \otimes z \), for any \( y, z \in \mathbb{R}^N \). Since we can choose \( z \) arbitrarily, we choose \( z = y \). We then get \( v^T u = v^T u \), \( v^T W = \mu_0 v^T u \), and \( v^T WMu = \mu_0 s v^T u \). The result follows by putting these identities into (30).

We now use Theorem 1 to analyze the trajectory of the relevant eigenvalues of \( \Gamma(\alpha) \), when \( \alpha \) is small.

The largest eigenvalue of \( \Gamma(0) \) is 1. For this case, we have \( u = v = 1 \). Then, \( R = I_N \) and \( s = 1 \). Hence, (28) becomes
\[
\mu^2 - (2 - \alpha) \mu + (1 - \alpha) \simeq 0 
\]
giving that either \( \mu \simeq 1 \) or \( \mu \simeq 1 - \alpha \). Since \( \mu_0 = 1 \) has multiplicity \( 2N \), this means that \( \Gamma(\alpha) \) will (approximately) have an eigenvalue at \( \mu = 1 \), with multiplicity \( N \) and another one at \( \mu = 1 - \alpha \) with the same multiplicity. The first set of \( N \) eigenvalues is a consequence of the fact that any point of the form \( (\xi_k, g_k) = (I_1 \otimes x_0) \), for any \( x_0 \in \mathbb{R}^N \), is a stationary point of the local linear dynamics determined by \( \Gamma(\alpha) \). However, we know from the global nonlinear model (18)–(20) that the only possibility of such stationary points is \( (I_1 \otimes x_0, 0) \). Hence, the convergence speed is determined by the remaining \( 2(1 - N) \) eigenvalues of \( \Gamma(\alpha) \). Hence, the second set of \( N \) eigenvalues describes a convergence mode of the distributed optimization algorithm.

The second largest eigenvalue of \( \Gamma(0) \) is \( \lambda_2 \). We have
\[
\mu^2 - \lambda_2 (2 - \alpha s) \mu + \lambda_2 (\lambda_2 - \alpha s) \simeq 0 
\]
which gives
\[
\mu \simeq \frac{\lambda_2}{2} \left( 2 - \alpha s \pm \sqrt{\alpha^2 s^2 + 4 \alpha s \frac{1}{\lambda_2 ^2 - 1}} \right). 
\]

As before, the above means that \( \Gamma(\alpha) \) will (approximately) have two eigenvalues with multiplicity \( N \), one moving up from \( \lambda_2 \) and another one moving down. Considering the one moving up, we can devise a criterion for choosing the design value \( \alpha_* \) of \( \alpha \). More precisely, we require this eigenvalue to be equal to \( 1 - \alpha_* \), i.e.
\[
1 - \alpha_* = \frac{\lambda_2}{2} \left( 2 - \alpha_* s + \sqrt{\alpha_*^2 s^2 + 4 \alpha_* s \left( \frac{1}{\lambda_2 ^2 - 1} \right)} \right). 
\]

Let now do in (29) the approximation
\[
R \simeq \frac{1}{\tau} \sum_{i=1}^{N} \nabla^2 f_i (x_\sigma) \left[ \nabla^2 f (x_\sigma) \right]^{-1} = I. 
\]

This gives \( s = 1 \), which when put in (31), gives the following step-size selection criterion:
\[
1 - \alpha_* = \frac{\lambda_2}{2} \left( 2 - \alpha_* + \sqrt{\alpha_*^2 + 4 \alpha_* \left( \frac{1}{\lambda_2 ^2 - 1} \right)} \right). 
\]
In particular, if \( \lambda_2 \in \mathbb{R} \), \( \alpha_* = 1 - \sqrt{\lambda_2} \).

**C. Distributed Step-Size Estimation**

The criterion (32) is based on the somehow coarse approximation \( R \simeq I \). A more accurate step-size selection can be achieved if the estimates of \( R, u, v \), and \( \lambda_2 \) are available at each node. The parameters \( u, v, \) and \( \lambda_2 \) depend on the communication network. These parameters can be either known in advance, or estimated during an initialization.
stage. In particular, if the network is undirected, the distributed method described in [21, Appendix C] can be used.

In contrast, matrix \( R \) depends on the optimization problem. Hence, it needs to be estimated. We can do so in a distributed manner using dynamic average consensus. Let \( R_i^0 \) denote the estimate of \( R \) obtained at node \( i \) and time \( k \), and let \( r_k = \text{col}(R_1^0, \ldots, R_N^0) \). The estimation is then initialized by \( R_1^0 = I_N \), for all \( i = 1, \ldots, I \), i.e., \( r_1 = \text{col}(I_N, \ldots, I_N) \), and proceeds as follows:

\[
 r_{k+1} = \mathbf{W} [r_k + \tau (x_{k+1}) - \tau (x_k)] 
\]  
(33)

where \( \tau(x_k) = \text{col}(\tau^1(x_k), \ldots, \tau^I(x_k)) \) with

\[
 \tau^i(x_k) = \frac{\tau^i u^i}{\nu^i} \nabla^2 f^i (\bar{x}^i_k) (H^i_k)^{-1} .
\]

In order to compute \( \alpha_k^i \) at each node and time \( k \) we need to solve (31). This requires computing an approximation \( s_k^i \) of \( s \) using \( R_i^0 \) in place of \( R \). Since we do not know the value of \( y \) for which (28) holds, we approximate it with the midpoint between the largest and smallest eigenvalues of \( R_i^0 \), i.e., we choose

\[
 s_k^i = \frac{1}{2} \left( \| R_i^0 \| + \| R_i^0 \|^{-1} \right) .
\]

We then compute \( \alpha_k^i \) by solving

\[
 1 - \alpha_k^i = \left[ 2 - \alpha_k^i s_k^i + \sqrt{(\alpha_k^i s_k^i)^2 + 4 \alpha_k^i s_k^i \left( \frac{1}{\lambda_k} - 1 \right)} \right] .
\]  
(34)

D. Local Stability Analysis

As mentioned in Section IV-B, \( \bar{x}_k \) does not act as input \( \bar{x}_k \) and \( g_k \) in the first-order dynamical model (27). Then, for analyzing local stability, we focus on that of \( \bar{x}_k \) and \( g_k \). A problem of doing so using the global model (18)–(19) is that its local linear dynamics (27) is marginally stable. We then base our study on the local dynamics of the global model (23)–(25).

Notation 6: Let \( \bar{x}_k = \bar{x}_k - x_k, \bar{\zeta}_k = (\bar{x}_k, \bar{x}_k, \bar{g}_k) \), and \( \| \bar{\zeta}_k \|^2 = \| \bar{x}_k \|^2 + \| \bar{x}_k \|^2 + \| \bar{g}_k \|^2 \). Let \( I = \mathbf{I} - \mathbf{A} \), and \( \alpha_1 = \frac{1}{2} \sum_{i=1}^{I} \alpha_i \). Let finally \( \Upsilon(\alpha) \) be given by (35) shown at the bottom of this page, and

\[
 \Theta = \begin{bmatrix}
 0 & 0 & 0 \\
 0 & \mathbf{W} & 0 \\
 0 & \mathbf{W} & 0
 \end{bmatrix} .
\]

Proposition 2: Under Assumption 1, for all \( k \in \mathbb{N} \)

\[
 \zeta_{k+1} = (\Theta - \Upsilon(\alpha)) \zeta_k + O \left(\| \zeta_k \|^2 \right) .
\]

Proof: The proof follows the step of that of Proposition 1. We use Fréchet derivatives to do a Taylor expansion of the map \( \zeta_{k+1}(\zeta_k) \) around \( \zeta_k = (0, 0, 0) \). We have \( \zeta_{k+1}(\zeta_k) = 0 \) and

\[
 \mathcal{D} \bar{x}_{k+1}(\zeta_k) = (1 - \alpha) \bar{x}_k - \alpha^i \mathbf{H}^i_k \mathbf{A} \delta(x_k) \bar{x}_k - \alpha^i \mathbf{H}^i_k \bar{g}_k .
\]

Also

\[
 \mathcal{D} \bar{x}_{k+1}(\zeta_k) = -\mathbf{I} \mathbf{A} (1_I \otimes I_N) \bar{x}_k - \mathbf{I} \mathbf{A} \mathbf{H}^i_k \bar{g}_k + \mathbf{W} \delta(x_k) (1_I \otimes I_N) \mathbf{A} \delta(x_k) \bar{x}_k - \mathbf{W} \delta(x_k) \mathbf{A} \delta(x_k) \bar{x}_k
\]

and

\[
 \mathcal{D} \bar{g}_{k+1}(\zeta_k) = -\mathbf{W} \delta(x_k) \alpha (1_I \otimes I_N) \bar{x}_k + \mathbf{W} [1 - \delta(x_k) \alpha \mathbf{H}^i_k] \bar{g}_k + \mathbf{W} \delta(x_k) \left( \mathbf{W} - I \right) - \alpha \mathbf{H}^i_k \mathbf{A} \delta(x_k) \bar{x}_k .
\]

We now state the main result of the section.

Theorem 2: Suppose Assumption 1 holds and, for all \( k \in \mathbb{N} \) and \( i = 1, \ldots, I, \alpha_k^i \) is chosen using (34). Then, there exists a neighborhood of \( (x_k^0, h_k^0, r_k^0) \), with \( r_k = 1_I \otimes R \), such that if \( (x_k, g_k, h_k, r_k) \) is inside that neighborhood

\[
 \lim_{k \to \infty} x_k = x_s, \quad \text{for all } i \in \{1, \ldots, I\} .
\]

Proof: Let \( \bar{r}_k = r_k - r_s \). In the adaptive step-size selection algorithm described in Section IV-C, \( \alpha_k^i \) is independent of \( \zeta_k \) and only depends on \( r_k \). With some abuse of notations we write \( \alpha_k^i = \alpha(\bar{r}_k) \) and

\[
 \zeta_{k+1} = \delta(\zeta_k, \alpha(\bar{r}_k))
\]

\[
 \bar{r}_{k+1} = \mathbf{W} [r_k + \tau (x_{k+1}) - \tau (x_k)]
\]

where \( \delta \) represents the mapping induced by (23)–(25). Clearly, \( (\zeta_k, r_k) = (0, r_s) \) is an equilibrium point of the above system. Let \( \alpha_s = \text{diag}(\alpha_s, \ldots, \alpha_s) \otimes I_N \). Clearly \( \alpha_s = \alpha(0, r_s) \).

Now, since \( \alpha_s \) always appears multiplying \( \bar{g}_k \), it follows that \( \mathcal{D} \delta(0, \alpha_s)(\zeta_k - \alpha_s) = 0 \). Hence, in the local dynamics of (37) and (38) around \( (0, \alpha_s) \), the term \( (\alpha_k - \alpha_s) \) is not an input of \( \zeta_{k+1} \). We also have, from the discussion in Section IV-C, that with the choice \( \alpha_s \), the algorithm is locally stable in a neighborhood of \( (\zeta_k, r_k) = (0, r_s) \). This means that \( \delta(\zeta_k, \alpha_s) \) is locally stable in that neighborhood. The local stability of (37) and (38) in the same neighborhood then immediately follows from the stability of (38). The result then follows since the local stability of (37) and (38) implies that of (18)–(20).

V. NUMERICAL EXAMPLE

A. Case Study

We consider a target localization problem. There are \( I = 30 \) nodes, measuring the distance to a target located at \( x_{\text{true}} \in \mathbb{R}^2 \). Node \( i \) is located at \( a_i^i \in \mathbb{R}^2 \), with \( a_i^i \sim \mathcal{N}(x_{\text{true}}, 100 \times I_2) \), and is initialized by \( x_1^1 \sim \mathcal{N}(x_{\text{true}}, I_2) \). It measures

\[
 z_i^i = \| x_{\text{true}} - a_i^i \|^2 + n_i^i, \quad n_i^i \sim \mathcal{N}(0, \sigma^2)
\]

with \( \sigma^2 = 0.01 \). Nodes are connected via a network with ring topology, whose gains are given by

\[
 w_{i,j} = \begin{cases}
 0.7, & i = j, \\
 0.15, & (i - j, I) = 1, \\
 0.15, & (i - j, I) = I - 2, \\
 0, & \text{otherwise}
\end{cases}
\]

where \( (a, b) \) denotes the \( a \) modulo \( b \) operation. This results in \( \lambda_2 = 0.9838 \).
Doing a maximum likelihood estimation of $x$ we obtain

$$x^\star = \arg \max_x (z^1, \ldots, z^n|x) = \arg \min_x \sum_{i=1}^n f^i(x)$$

with

$$f^i(x) = \left(\|x - a^i\|_2^2 - z^i\right)^2.$$

It is straightforward to obtain

$$\nabla f^i(x) = 4 \left(\|x - a^i\|_2^2 - z^i\right) (x - a^i)$$

$$\nabla^2 f^i(x) = 8 (x - a^i) (x - a^i)^\top + 4 \left(\|x - a^i\|_2^2 - z^i\right) I_2$$

Using the above, the optimal step size, in the sense of minimizing the largest modulus of the eigenvalues of $\Gamma(\alpha)$, which differs from 1, is given by $\alpha_{\text{opt}} = 6.197 \times 10^{-3}$.

### B. Numerical Experiments

In the first experiment, we evaluate the effect of considering the modification (12) introduced by Algorithm A, as described in Section III-C. We use $x_{\text{true}} = [0, 0]^\top$, $\beta = 0.1$ and the optimal step size $\alpha_{\text{opt}} = 6.197 \times 10^{-3}$. We also use $c_k^i \triangleq \|x_k^i - x_{\text{true}}\|$ as the performance metric for each node. In Fig. 1(a), we compare the performance of the proposed algorithm with Algorithm A. We see how the lack of a consensus stage prevents the local variables $x_k^i$ to converge to a common value. In Fig. 1(b), we see the effect of considering also the modification (14)–(15), i.e., Algorithm VZCPS. We see that it converges, although at a much slower rate than the proposed algorithm. We then conclude that it is this second modification that causes the convergence of Algorithm VZCPS.

In the second experiment, we remove modification (12), i.e., add consensus on variables, and study the effect of modification (14)–(15) introduced by Algorithm B. In Fig. 2(a), we see that Algorithm B converges at rate similar to that of the proposed algorithm. However, as explained in Section III, modification (14)–(15) has a negative effect when the minimizing parameters $x_k$ are far from zero. We show this in Fig. 2(b), where we repeat the previous experiment with $x_{\text{true}} = [300, 300]^\top$. We see how the local estimates of Algorithm B are pulled away from $x_{\text{true}}$ during the initial iterations, until consensus is reached. In Fig. 2(c), we repeat the experiment with $x_{\text{true}} = [1000, 1000]^\top$. We see that Algorithm B is not able to reach consensus in time, which causes its divergence.

In the third experiment, we evaluate the use of the distributed algorithm for estimating the step size. In Fig. 3(a), we compare the convergence of $\|x_k^i - x_{\text{true}}\|$ using both the optimal step size $\alpha_{\text{opt}}$ and the distributedly estimated one. We see that both methods converge at a very similar rate. We also show in the figure the theoretically optimal step size $\alpha_{\text{opt}} = 6.197 \times 10^{-3}$. Finally, Fig. 3(c) shows how the two eigenvalues used to compute $\alpha_{\text{opt}}$ depend on $\alpha_k$ and compares this with the approximated dependence given by Theorem 1. We see how, before the two eigenvalues meet, the true and approximated trajectories closely resemble each other. This results in $\alpha_{\text{opt}} = 6.117 \times 10^{-3}$ being a good approximation of $\alpha_{\text{opt}} = 6.197 \times 10^{-3}$.

### VI. Conclusion

We aimed at achieving the fastest convergence rate for distributed optimization. We did two steps toward this goal. In the first step, we proposed a new distributed optimization method which converges faster than other available options. In the second step, we proposed a distributed method to estimate the step size that maximizes this rate. We provided sufficient conditions for the convergence of the resulting method in a neighborhood of a local solution. We presented numerical experiments confirming our claims.

### References

[1] J. Xiao, A. Ribeiro, Z. Luo, and G. B. Giannakis, “Distributed compression-estimation using wireless sensor networks,” *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 27–41, Jul. 2006.

[2] A. Ribeiro, D. I. Stergios, I. S. Roumeliotis, and G. Giannakis, “Kalman filtering in wireless sensor networks,” *IEEE Control Syst. Mag.*, vol. 30, no. 2, pp. 66–86, Apr. 2010.

[3] P. Massoni and M. Verhaegen, “Distributed control for identical dynamically coupled systems: A decomposition approach,” *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 124–135, Jan. 2009.

[4] T. Yang et al., “A survey of distributed optimization,” *Annu. Rev. Control*, vol. 47, pp. 278–305, Jan. 2019.

[5] A. Nedić and J. Liu, “Distributed optimization for control,” *Annu. Rev. Control*, vol. 1, pp. 77–103, 2018.
[6] M. Gürbüzbalaban, A. Ozdaglar, and P. Parrilo, “A globally convergent incremental Newton method,” *Math. Prog.*, vol. 151, no. 1, pp. 283–313, 2015.

[7] S. N. Boyd, E. Parikh, B. C. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[8] A. Mokhtari, Q. Ling, and A. Ribeiro, “Network Newton distributed optimization methods,” *IEEE Trans. Signal Process.*, vol. 65, no. 1, pp. 146–161, Jan. 2017.

[9] F. Mansoori and E. Wei, “A fast distributed asynchronous Newton-based optimization algorithm,” *IEEE Trans. Autom. Control*, vol. 65, no. 7, pp. 2769–2784, Jul. 2020.

[10] R. Tutunov, H. Bou-Ammar, and A. Jadbabaie, “Distributed Newton method for large-scale consensus optimization,” *IEEE Trans. Autom. Control*, vol. 64, no. 10, pp. 3983–3994, Oct. 2019.

[11] A. Jadbabaie, A. Ozdaglar, and M. Zargham, “A distributed Newton method for network optimization,” in *Proc. IEEE Conf. Decis. Control*, 2009, pp. 2736–2741.

[12] D. Varagnolo, F. Zanella, A. Cenedese, G. Pillonetto, and L. Schenato, “Newton-Raphson consensus for distributed convex optimization,” *IEEE Trans. Autom. Control*, vol. 61, no. 4, pp. 994–1009, Apr. 2016.

[13] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, “Newton-Raphson consensus for distributed convex optimization,” in *Proc. IEEE Conf. Decis. Control*, 2011, pp. 5917–5922.

[14] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, “Asynchronous Newton-Raphson consensus for distributed convex optimization,” *IFAC Proc. Volumes*, vol. 45, no. 26, pp. 133–138, 2012.

[15] N. Bof, R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo, “Multiagent Newton-Raphson optimization over lossy networks,” *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2983–2990, Jul. 2019.

[16] Y. Li, H. Zhang, B. Huang, and J. Han, “A distributed Newton-Raphson-based coordination algorithm for multi-agent optimization with discrete-time communication,” *Neural Comput. Appl.*, vol. 32, no. 9, pp. 4649–4663, 2018.

[17] J. Zhang, K. You, and T. Başar, “Distributed adaptive Newton methods with globally superlinear convergence,” 2020, arXiv:2002.07378.

[18] T. Anderson, C. Chang, and S. Martínez, “Distributed approximate Newton algorithms and weight design for constrained optimization,” *Automatica*, vol. 109, 2019, Art. no. 108538.

[19] R. Olfati-Saber, J. Fax, and R. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[20] M. Zhu and S. Martínez, “Discrete-time dynamic average consensus,” *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.

[21] D. Marelli, Y. Xu, M. Fu, and Z. Huang, “Distributed newton optimization with maximized convergence rate,” 2021, arXiv:2102.08726.