A Hessian inversion-free exact second order method for distributed consensus optimization

Dušan Jakovetić, *Member, IEEE,* Nataša Krejić, Nataša Krklec Jerinkić

***Abstract*—We consider a standard distributed consensus optimization problem where a set of agents connected over an undirected network minimize the sum of their individual (local) strongly convex costs. Alternating Direction Method of Multipliers (ADMM) and Proximal Method of Multipliers (PMM) have been proved to be effective frameworks for design of exact distributed second order methods (involving calculation of local cost Hessians). However, existing methods involve explicit calculation of local Hessian inverses at each iteration that may be very costly when the dimension of the optimization variable is large. In this paper, we develop a novel method, termed INDO (Inexact Newton method for Distributed Optimization), that alleviates the need for Hessian inverse calculation. INDO follows the PMM framework but, unlike existing work, approximates the Newton direction through a generic fixed point method (e.g., Jacobi Overrelaxation) that does not involve Hessian inverses. We prove exact global linear convergence of INDO and provide analytical studies on how the degree of inexactness in the Newton direction calculation affects the overall method's convergence factor. Numerical experiments on several real data sets demonstrate that INDO's speed is on par (or better) as state of the art methods iteration-wise, hence having a comparable communication cost. At the same time, for sufficiently large optimization problem dimensions n (even at n on the order of couple of hundreds), INDO achieves savings in computational cost by at least an order of magnitude.**

***Index Terms*—Inexact Newton, proximal method of multipliers, distributed optimization, exact convergence, strongly convex problems.**

I. Introduction

We consider problems of the form

$$\min_{y\in\mathbb{R}^n} \sum_{i=1}^{N} f_i(y). \tag{1}$$

Here, $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., N$, is a strongly convex local cost function assigned to a node within a network of distributed agents able to perform local operations and communicate with their neighbours. Formulation (1) finds a number of applications in signal processing, e.g., [7], [18], control, e.g., [23], Big Data analytics, e.g., [26], social networks, e.g., [3], etc. The available methods for solving (1) include a large class of the so called exact methods that ensure convergence to the exact solution of (1) with different rates of convergence. The exact convergence is achieved in several ways – by utilizing diminishing step sizes in gradient methods for penalized reformulation [15], [36], by gradient tracking or second order methods that are defined within primal-dual framework, e.g, [8], [12], [13], [16], [21], [24], [27], [28], [30], [31], [33], [34], [38]–[41], or in the framework of alternating direction methods [6], [22]. Multiple consensus steps per each gradient update to ensure exact convergence are also considered [5].

For the current paper, of special relevance are two strategies available in the literature – the proximal method of multipliers as a framework to develop exact distributed methods in [21], and the well known theory of inexact Newton methods in centralized optimization, [9]. The main advantage of the inexact Newton methods is that they avoid oversolving of the Newtonian system of linear equations; as such, they have been employed in related problems, for example in minimizing finite sums in machine learning applications [4]. To be more specific, we consider the constrained reformulation of (1) in the augmented space (like in, e.g., [21]) and build up on the distributed second order approximation of the Augmented Lagrangian. The second order approximation of the Augmented Lagrangian conforms with the sparsity structure of the network and hence Newton-like step is well defined. The main obstacle for applying the Newton method is the fact that although the Hessian is sparse and distributed, the inverse Hessian is dense and challenging to compute efficiently in a distributed environment. One possibility is to approximate the inverse Hessian by inverting (local) Hessians of the f_i's and build a Taylor-like approximation to the inverse of the global Hessian as suggested in [20], [21], and exploited in [10], [19].

The approach we propose here is based on a recent result on distributed solution of systems of linear equations [14]. More precisely, the Newton-like equation that defines the update step is a solution of a system of linear equations defined by the global Hessian of the Augmented Lagrangian. So, instead of approximating the inverse Hessian as done in, e.g., [20], [21], we propose to solve the system of linear equations inexactly, in a distributed manner by a suitable iterative method. A distinctive feature of the approach is that it avoids inversion of the Hessians of the f_i's; instead, only diagonal elements of the Hessians are inverted. Another appealing feature of the approach is compatibility with the theory of inexact Newton methods in centralized optimization. The proposed approach is particularly suitable for the case of relatively large n, when the approximation of the inverse Hessian as

The authors are with the University of Novi Sad, Faculty of Sciences, Department of Mathematics and Informatics, Trg Dositeja Obradovica 4, Novi Sad, Serbia. Authors' emails: dusan.jakovetic@dmi.uns.ac.rs; natasak@uns.ac.rs; natasa.krklec@dmi.uns.ac.rs. This work is supported by the Ministry of Education, Science and Technological Development, Republic of Serbia. It is also supported in part by the Bilateral Cooperation Serbia – Croatia, project "Optimization Methods Application in Biomedicine," and by the European Union's Horizon 2020 Research and Innovation program under grant agreement No 871518. The paper reflects only the view of the authors and the Commission is not responsible for any use that may be made of the information it contains.

in [20], [21] might be expensive, and thus an iterative solver for the system of linear equations is a natural choice. The convergence theory developed here relies on the theory for the proximal methods of multipliers, as the proposed method fits this general framework, following, e.g., [21]. However, the error analysis carried out here – reflecting the inexact solution of the Newtonian system of linear equations and how it affects the overall proximal multipliers convergence – is very different for the proposed method.

To summarize, the main contributions of this paper are the definition and convergence theory of the novel exact second order method termed INDO (Inexact Newton method for Distributed Optimization).[1] Linear convergence to the exact solution is shown for strongly convex problems under a set of standard assumptions. The rate of convergence is also analysed, assuming that the linear solver is of fixed-point type, in particular we focused on the application of the Jacobi Overrelaxation method to derive an estimate of the convergence factor. The convergence of the JOR method is analysed with respect to the properties of the system arising in the corresponding primal-dual second order approximation. Furthermore, it is shown that one can choose a suitable relaxation parameter and estimate the number of JOR iterations needed to achieve desired error bound in solving the linear system.

To the best of our knowledge, INDO is the first exact second order method that completely eliminates *Hessian inverse* calculations from the variable updates. As a result, the proposed method can reduce computational cost by at least an order of magnitude for problems with sufficiently large variable dimensions, while at the same time achieving at least comparable communication cost. This advantage has been demonstrated through a number of numerical real and synthetic data test examples, by comparing INDO with a state-of-the-art method ESOM. Furthermore, the advantage observed in experiments has been corroborated with analytical analysis hints and insights, through the convergence factor comparisons for the inner loops of INDO and ESOM. In short, the analytical insights reveal that, for a range of tuning parameter values, the inner loop convergence factor can be maintained (incurring a minor degradation or even improvement), when the explicit Hessian inverse calculation is removed. Finally, we examine INDO's capabilities to handle highly heterogeneous data and highly heterogeneous local costs' cases. Numerical studies demonstrate that INDO exhibits improved or comparable performance on heterogeneous data examples, when compared with ESOM and EXTRA [27].

This paper is organized as follows. In Section 2 we introduce the problem model, the primal-dual framework based on the proximal method of multipliers and state the assumptions on the problem. The INDO method is introduced and theoretically analysed in Section 3. Details on distributed implementation as well as the analysis of the convergence factor are presented in Section 4, while Section 5 is devoted to practical implementation of INDO with the analysis of computational costs and comparison with ESOM [21] and EXTRA [27] methods through standard examples from machine learning. Some conclusions are drawn in Section 6. Finally, lengthy supporting proofs are delegated to the Appendix.

[1]To avoid confusion, we clarify that the wording "inexact Newton" here refers to approximately solving the Newtonian system of linear equations that defines the Newton direction; on the other hand, the wording "exact method" here signifies that INDO converges to the exact solution of (1).

II. Preliminaries

The notation used is the paper is the following. We use upper case blackboard bold letters to denote matrices in $\mathbb{R}^{nN\times nN}$, e.g., $\mathbb{A}, \mathbb{B}, \ldots$. For $\mathbb{A}\in\mathbb{R}^{nN\times nN}$, we use its block elements representation $\mathbb{A}=[A_{ij}]$, $A_{ij}\in\mathbb{R}^{n\times n}$. We denote scalar elements of $\mathbb{A}$ by $a_{ij}\in\mathbb{R}$. Similarly, upper normal letters $A, B, ...$, are used for stand-alone matrices in $\mathbb{R}^{n\times n}$. The vectors in $\mathbb{R}^{nN}$ are denoted by bold lowercase letters, e.g., $\mathbf{x}\in\mathbb{R}^{nN}$; their component blocks are $x_i\in\mathbb{R}^n$; similarly, we use normal lowercase letters, e.g., y, for stand-alone vectors in $\mathbb{R}^n$. The notation $\|\cdot\|_2$ stands for the 2-norm of its vector or matrix argument; if the subscript is omitted, $\|\cdot\|$ also designates the 2-norm, if not stated otherwise. We denote by $diag(A)$ the diagonal matrix with diagonal elements equal to those of matrix A, and by $\sigma(A)$ the spectral radius of A.

The network of connected computational nodes (agents) is represented by a graph $G=(V,\mathcal{E})$, where V is the set of nodes $\{1,...,N\}$ and $\mathcal{E}$ is the set of undirected edges (i,j). Denote by O_i the set of neighbors of node i and let $\bar{O}_i=O_i\bigcup\{i\}$. The communication network is accompanied by a communication (weight) matrix W with the following properties.

A 1: The matrix $W\in\mathbb{R}^{N\times N}$ is symmetric, doubly stochastic and

$$w_{ij}>0 \text{ if } j\in\bar{O}_i,\ w_{ij}=0 \text{ if } j\notin\bar{O}_i$$

Let us assume that each of the N nodes has its local cost function f_i and has access to computing its first and second derivatives. Under the assumption A1, the problem (1) has the equivalent form

$$\min_{\mathbf{x}\in\mathbb{R}^{nN}} f(\mathbf{x}):=\sum_{i=1}^{N} f_i(x_i) \quad \text{s. t.} \quad (\mathbb{I}-\mathbb{W})^{1/2}\mathbf{x}=0, \qquad (2)$$

where $\mathbf{x}=(x_1;...;x_N)\in\mathbb{R}^{nN}$, $\mathbb{W}=W\otimes I\in\mathbb{R}^{nN\times nN}$ and $\mathbb{I}\in\mathbb{R}^{nN\times nN}$ is the identity matrix.

A 2: The functions f_i's are twice continuously differentiable and the eigenvalues of the local Hessians are bounded by positive constants $0<m\leq M<\infty$, i.e.,

$$mI\preceq\nabla^2 f_i(y)\preceq MI,$$

for all $y\in\mathbb{R}^n$ and $i=1,\ldots,N$.

The above assumption implies that the function $f(\mathbf{x})$ is also strongly convex with constant m and its gradient ∇f is Lispchitz continuous with constant M. Given that we consider a method of Newton-type, the assumption on Lipschitz continuous Hessian is also needed, as usual in the theory of Newton methods and in [21].

A 3: The Hessian of the objective function $\nabla^2 f(\mathbf{x})$ is Lipschitz continuous, i.e., there exists $L>0$ such that

$$\|\nabla^2 f(\mathbf{x})-\nabla^2 f(\mathbf{z})\|\leq L\|\mathbf{x}-\mathbf{z}\|,\ \ \mathbf{x},\mathbf{z}\in\mathbb{R}^{nN}.$$

The following Lemma from [22] will be used in the convergence analysis.

Lemma 2.1: [22] Assume that A2-A3 hold. Then for all $\mathbf{x}, \mathbf{z} \in \mathbb{R}^{nN}$

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{z}) + \nabla^2 f(\mathbf{x})(\mathbf{z}-\mathbf{x})\| \leq \min\{2M, \frac{L}{2}\|\mathbf{x}-\mathbf{z}\|\}\|\mathbf{x}-\mathbf{z}\|.$$

Now, the Augmented Lagrangian function of (2) is defined as

$$\mathcal{L}(\mathbf{x}, \mathbf{v}) = f(\mathbf{x}) + \mathbf{v}^T(\mathbb{I}-\mathbb{W})^{1/2}\mathbf{x} + \frac{\alpha}{2}\mathbf{x}^T(\mathbb{I}-\mathbb{W})\mathbf{x},$$

where $\alpha > 0$ is a constant and $\mathbf{v}$ is dual variable. For a strictly positive proximal coefficient ε, the proximal method of multipliers is defined by the primal update

$$\mathbf{x}^{k+1} = \arg\min \mathcal{L}(\mathbf{x}, \mathbf{v}^k) + \frac{\varepsilon}{2}\|\mathbf{x}-\mathbf{x}^k\|^2, \tag{3}$$

while the dual update with the stepsize α is

$$\mathbf{v}^{k+1} = \mathbf{v}^k + \alpha(\mathbb{I}-\mathbb{W})^{1/2}\mathbf{x}^{k+1}. \tag{4}$$

The primal step is not implementable in the distributed environment due to the augmented term $\frac{\alpha}{2}\mathbf{x}^T(\mathbb{I}-\mathbb{W})\mathbf{x}$ and therefore an approximation of $\mathcal{L}(\mathbf{x}, \mathbf{v})$ is needed. We consider the second order Taylor approximation with respect to $\mathbf{x}$ at the point $(\mathbf{x}^k, \mathbf{v}^k)$,

$$\mathcal{L}(\mathbf{x}, \mathbf{v}^k) \approx \mathcal{L}(\mathbf{x}^k, \mathbf{v}^k) + \mathbf{g}^k(\mathbf{x}-\mathbf{x}^k) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^k)^T\mathbb{H}^k(\mathbf{x}-\mathbf{x}^k), \tag{5}$$

with

$$\mathbb{H}^k = \nabla^2 f(\mathbf{x}^k) + \alpha(\mathbb{I}-\mathbb{W}) + \varepsilon\mathbb{I}. \tag{6}$$

and

$$\mathbf{g}^k := \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^k, \mathbf{v}^k) = \nabla f(\mathbf{x}^k) + (\mathbb{I}-\mathbb{W})^{1/2}\mathbf{v}^k + \alpha(\mathbb{I}-\mathbb{W})\mathbf{x}^k.$$

Using the second order Taylor approximation (5) one can define the Newton step $\mathbf{d}_N^k$ as

$$\mathbb{H}^k\mathbf{d}_N^k = -\mathbf{g}^k, \tag{7}$$

and form the primal update as $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}_N^k$. Notice that the matrix $\mathbb{H}^k$ has the sparsity structure of the graph; hence, one can apply an iterative method of the fixed point-type to solve (7). The details of such procedure can be found in [14]. However, solving (7) exactly might be too costly and an approximate solution to (7) is in fact sufficient for the convergence as we will demonstrate further on. Thus the step we use is defined by $\mathbb{H}^k\mathbf{d}^k = -\mathbf{g}^k + \mathbf{r}^k$, for some residual $\mathbf{r}^k$ which obeys the classical Inexact Newton forcing condition [9]

$$\|\mathbf{r}^k\| \leq \eta_k\|\mathbf{g}^k\|,$$

for a forcing term $\eta_k \geq 0$.

The dual update is given by (4) but the matrix $\alpha(\mathbb{I}-\mathbb{W})^{1/2}$ is not neighbor sparse. Thus we apply the same variable transformation as in [21], defining a sequence of variables $\mathbf{q}^k$ as

$$\mathbf{q}^k = (\mathbb{I}-\mathbb{W})^{1/2}\mathbf{v}^k.$$

Now, multiplying the dual update (4) by $(\mathbb{I}-\mathbb{W})^{1/2}$ from the left and using the definition of $\mathbf{q}^k$ we obtain

$$\mathbf{q}^{k+1} = \mathbf{q}^k + \alpha(\mathbb{I}-\mathbb{W})\mathbf{x}^{k+1}, \tag{8}$$

which is computable in the distributed manner. Notice that with this change of variables we get

$$\mathbf{g}^k = \nabla f(\mathbf{x}^k) + \mathbf{q}^k + \alpha(\mathbb{I}-\mathbb{W})\mathbf{x}^k. \tag{9}$$

III. Distributed Inexact Newton Algorithm

With the notation introduced in the previous section we are now in a position to state the general INDO method. Further details regarding the inexact Newton step computation (step S1 in the INDO method below), are postponed to Section 4. Therein, we discuss different possibilities for fulfilling the Inexact Newton condition (10) in a distributed environment. Assume that the forcing sequence $\{\eta_k\}_{k=0}^{\infty}$ is nonnegative.

INDO method

Input : $\mathbf{x}^0 \in \mathbb{R}^{nN}$, $\mathbf{q}^0 = 0$, $k = 0, \{\eta_k\}_k, \alpha > 0$, $\varepsilon > 0$

S1 Find a step $\mathbf{d}^k$ such that

$$\mathbb{H}^k\mathbf{d}^k = -\mathbf{g}^k + \mathbf{r}^k, \ \|\mathbf{r}^k\| \leq \eta_k\|\mathbf{g}^k\|. \tag{10}$$

S2 Compute the primal update

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}^k.$$

S3 Compute the dual update

$$\mathbf{q}^{k+1} = \mathbf{q}^k + \alpha(\mathbb{I}-\mathbb{W})\mathbf{x}^{k+1},$$

set $k = k+1$ and return to S1.

As already stated, one can easily see that the algorithm fits the general proximal multipliers framework [21], with the primal step defined in S1-S2 and the dual update in S3. The key novelty is the direction computation in S1 where we generate a suitable inexact direction such that the residual is small enough with respect to the gradient norm, (10). This important property actually allows for flexible approach in the linear solver and avoids oversolving. That is, the proposed approach allows using a relatively large tolerance while the gradient is large, and implies stringent tolerance condition as the value of gradient decreases. The details of S1 implementation are presented in Section 4, where we also give INDO algorithm from the perspective of each node in the network.

Due to the convexity and the fact that we only have equality constraints, $\mathbf{x}^*$ is a solution of problem (2) if and only if there exists $\mathbf{v}^*$ such that

$$\nabla f(\mathbf{x}^*) + (\mathbb{I}-\mathbb{W})^{1/2}\mathbf{v}^* = 0 \tag{11}$$

$$(\mathbb{I}-\mathbb{W})^{1/2}\mathbf{x}^* = 0. \tag{12}$$

Strong convexity implies that $(\mathbf{x}^*, \mathbf{v}^*)$ is unique.

The convergence analysis we develop below relies on the reasoning from [21] in general but the error analysis, with respect to the proximal method of multipliers (see Section 2) is fundamentally different. We present the proof through a sequence of technical Lemmas, some of which correspond to parts of proofs in [21] that are rather similar. Proofs of the Lemmas similar to [21] are provided in Appendix for completeness.

Lemma 3.1: Let A1-A2 hold. A sequence generated by Algorithm INDO satisfies

$$\mathbf{q}^{k+1} - \mathbf{q}^k - \alpha(\mathbb{I} - \mathbb{W})(\mathbf{x}^{k+1} - \mathbf{x}^*) = 0.$$

Lemma 3.2: Let assumptions A1-A2 hold and $\{\mathbf{x}^k, \mathbf{q}^k\}$ be a sequence generated by Algorithm INDO. Then

$$\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*) + \varepsilon \mathbf{d}^k + \mathbf{q}^{k+1} - \mathbf{q}^* + \mathbf{e}^k = 0,$$

where

$$\mathbf{e}^k = \nabla^2 f(\mathbf{x}^k)\mathbf{d}^k + \nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k+1}) - \mathbf{r}^k, \ \mathbf{r}^k = \mathbb{H}^k \mathbf{d}^k + \mathbf{g}^k,$$

and

$$\mathbf{q}^* = (\mathbb{I} - \mathbb{W})^{1/2}\mathbf{v}^*.$$

Lemma 3.3: Let assumptions A1-A3 hold and $\{\mathbf{x}^k, \mathbf{q}^k\}$ be the sequence generated by INDO. Then

$$\begin{aligned} \|\mathbf{e}^k\| &\leq \min\{2M, \frac{L}{2}\|\mathbf{d}^k\|\}\|\mathbf{d}^k\| \\ &+ \eta_k(2\alpha + M)\|\mathbf{x}^k - \mathbf{x}^*\| + \sqrt{2}\eta_k\|\mathbf{v}^k - \mathbf{v}^*\|. \end{aligned}$$

In the convergence analysis below we will use the following inequality, [21]. Let a, b be two arbitrary vectors of the same dimension and $\xi > 0$ be an arbitrary real number. Then

$$-2a^T b \leq \frac{1}{\xi}\|a\|^2 + \xi\|b\|^2. \tag{13}$$

The technical lemma below is proved in [21], although it is not stated separately, see the proof of Theorem 2, inequality (93) in [21].

Lemma 3.4: Assume that A1-A3 hold and $\{\mathbf{x}^k, \mathbf{q}^k\}$ be a sequence generated by Algorithm INDO. Then

$$\begin{aligned} \|\mathbf{v}^{k+1} - \mathbf{v}^*\|^2 &\leq \frac{\beta\varepsilon^2}{(\beta-1)\lambda_2}\|\mathbf{d}^k\|^2 \\ &+ \frac{\phi\beta}{\lambda_2}\|\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)\|^2 + \frac{\beta\phi}{(\phi-1)\lambda_2}\|\mathbf{e}^k\|^2, \end{aligned}$$

where λ_2 is the smallest nonzero eigenvalue of $\mathbb{I} - \mathbb{W}$, and $\beta, \phi > 1$ are arbitrary constants.

Let us define the sequence of concatenated dual and primal errors as well as the Lyapunov function using the matrix $\mathcal{G}$ below,

$$\mathbf{u} = \begin{bmatrix} \mathbf{v} \\ \mathbf{x} \end{bmatrix} \quad \mathcal{G} = \begin{bmatrix} \mathbb{I} & 0 \\ 0 & \alpha\varepsilon\mathbb{I} \end{bmatrix}.$$

Analogously define the concatenated $\mathbf{u}^*$ and consider the sequence $\|\mathbf{u}^k - \mathbf{u}^*\|_{\mathcal{G}}^2 = \|\mathbf{v}^k - \mathbf{v}^*\|^2 + \alpha\varepsilon\|\mathbf{x}^k - \mathbf{x}^*\|^2$. We will prove that that the sequence $\|\mathbf{u}^k - \mathbf{u}^*\|_{\mathcal{G}}^2$ converges to zero linearly and hence the sequence of primal errors $\|\mathbf{x}^k - \mathbf{x}^*\|$ converges to zero linearly as well.

The following lemma is proved in the Appendix.

Lemma 3.5: Assume that A1-A3 hold and let $\{\mathbf{x}^k, \mathbf{q}^k\}$ be a sequence generated by Algorithm INDO. Then

$$\begin{aligned} &\|\mathbf{u}^{k+1} - \mathbf{u}^*\|_{\mathcal{G}}^2 - \|\mathbf{u}^k - \mathbf{u}^*\|_{\mathcal{G}}^2 \\ \leq\ & -\frac{2\alpha}{m+M}\|\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)\|^2 \\ -\ & \|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2_{(\frac{2\alpha m M}{m+M} - \frac{\alpha}{\zeta})\mathbb{I} + \alpha^2(\mathbb{I}-\mathbb{W})} \\ -\ & \alpha\varepsilon\|\mathbf{d}^k\|^2 + \alpha\zeta\|\mathbf{e}^k\|^2. \end{aligned} \tag{14}$$

The main convergence result is stated in the following Theorem.

Theorem 3.1: Assume that A1-A3 hold and let $\{\mathbf{x}^k, \mathbf{q}^k\}$ be a sequence generated by Algorithm INDO. Let $\beta, \phi > 1$ be arbitrary constants, λ_2 the smallest positive eigenvalue of $(\mathbb{I} - \mathbb{W})$ and $\zeta \in ((m+M)/(2mM), \varepsilon/(8M^2))$. Then there exists $\bar{\eta} > 0$ such that for $\eta_k \leq \bar{\eta}$, the sequence of Lyapunov functions $\|\mathbf{u}^k - \mathbf{u}^*\|_{\mathcal{G}}$ satisfies

$$\|\mathbf{u}^{k+1} - \mathbf{u}^*\|_{\mathcal{G}}^2 \leq \frac{1+\tilde{\delta}}{1+\delta}\|\mathbf{u}^k - \mathbf{u}^*\|_{\mathcal{G}}^2,$$

where $\tilde{\delta} < \delta \leq \min\{\delta_{\mathrm{a}}, \delta_{\mathrm{b}}\}$ with

$$\begin{aligned} \delta_{\mathrm{a}} &= \frac{2mM}{(m+M)\varepsilon} - \frac{1}{\varepsilon\zeta} \\ \delta_{\mathrm{b}} &= \min\{\frac{(\alpha\varepsilon - 8M^2\alpha\zeta)(\phi-1)(\beta-1)\lambda_2}{\beta\varepsilon^2(\phi-1) + 8M^2\beta(\beta-1)\phi}, \frac{2\alpha\lambda_2}{(m+M)\phi\beta}\}. \end{aligned}$$

Proof. To prove the statement we have to find δ and $\tilde{\delta}$ such that

$$\delta\|\mathbf{u}^{k+1} - \mathbf{u}^*\|_{\mathcal{G}}^2 - \tilde{\delta}\|\mathbf{u}^k - \mathbf{u}^*\|_{\mathcal{G}}^2 \leq \|\mathbf{u}^k - \mathbf{u}^*\|_{\mathcal{G}}^2 - \|\mathbf{u}^{k+1} - \mathbf{u}^*\|_{\mathcal{G}}^2. \tag{15}$$

Using the estimate for $\|\mathbf{v}^{k+1} - \mathbf{v}^*\|$ given in Lemma 3.4 and (14) in Lemma 3.5, the inequality (15) holds if

$$\begin{aligned} &\delta\alpha\varepsilon\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + \frac{\delta\beta\varepsilon^2}{(\beta-1)\lambda_2}\|\mathbf{d}^k\|^2 \\ +\ & \frac{\delta\phi\beta}{\lambda_2}\|\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)\|^2 \\ +\ & \frac{\beta\phi\delta}{(\phi-1)\lambda_2}\|\mathbf{e}^k\|^2 - \tilde{\delta}\alpha\varepsilon\|\mathbf{x}^k - \mathbf{x}^*\|^2 - \tilde{\delta}\|\mathbf{v}^k - \mathbf{v}^*\|^2 \\ \leq\ & \|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2_{(\frac{2\alpha m M}{m+M} - \frac{\alpha}{\zeta})\mathbb{I} + \alpha^2(\mathbb{I}-\mathbb{W})} + \alpha\varepsilon\|\mathbf{d}^k\|^2 \\ +\ & \frac{2\alpha}{m+M}\|\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)\|^2 - \alpha\zeta\|\mathbf{e}^k\|^2. \end{aligned} \tag{16}$$

By Lemma 3.3 we have

$$\|\mathbf{e}^k\|^2 \leq 8M^2\|\mathbf{d}^k\|^2 + 4\eta_k^2(M+2\alpha)^2\|\mathbf{x}^k - \mathbf{x}^*\|^2 + 8\eta_k^2\|\mathbf{v}^k - \mathbf{v}^*\|^2.$$

Substituting this bound for $\|\mathbf{e}^k\|^2$ and the corresponding lower bound for $-\|\mathbf{e}^k\|^2$ at both sides of (16) we get the inequality

$$\begin{aligned} 0 \quad \leq\ & \|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2_{(\frac{2\alpha m M}{m+M} - \frac{\alpha}{\zeta})\mathbb{I} + \alpha^2(\mathbb{I}-\mathbb{W})} \\ +\ & \|\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)\|^2 \left(\frac{2\alpha}{m+M} - \frac{\delta\phi\beta}{\lambda_2}\right) \\ +\ & \|\mathbf{d}^k\|^2 \left(\alpha\varepsilon - \frac{\delta\beta\varepsilon^2}{(\beta-1)\lambda_2} - 8M^2(\frac{\beta\phi\delta}{(\phi-1)\lambda_2} + \alpha\zeta)\right) \\ +\ & \|\mathbf{x}^k - \mathbf{x}^*\|^2 \left(-4\eta_k^2(M+2\alpha)^2 \left(\frac{\delta\beta\phi}{(\phi-1)\lambda_2} + \alpha\zeta\right)\right. \\ +\ & \left.\alpha\varepsilon\tilde{\delta}\right) \\ +\ & \|\mathbf{v}^k - \mathbf{v}^*\|^2 \left(\tilde{\delta} - 8\eta_k^2\left(\alpha\zeta + \frac{\delta\beta\phi}{(\phi-1)\lambda_2}\right)\right). \end{aligned} \tag{17}$$

The last inequality holds for δ specified in the statement and $\tilde{\delta} < \delta$ if

$$\begin{aligned} &\eta_k^2 \leq \bar{\eta}^2 \\ &\leq \min\{\frac{\tilde{\delta}\alpha\varepsilon}{4(M+2\alpha)^2(\alpha\zeta + \frac{\delta\beta\phi}{(\phi-1)\lambda_2})}, \frac{\tilde{\delta}}{8(\alpha\zeta + \frac{\delta\beta\phi}{(\phi-1)\lambda_2})}\} \end{aligned}$$

□

Notice that the error term of INDO method can be represented as $e^k = e^k_{ESOM-\infty} - r^k$, where $e^k_{ESOM-\infty}$ is the theoretical error for ESOM method with infinitely many inner iterations – that is, the error of Newton method applied to the Augmented Lagrangian. Theorem 3.1 reveals that $\tilde{\delta}$ can be chosen arbitrarily from the interval $(0, \delta)$, but the forcing term η_k depends on $\tilde{\delta}$ directly which can be seen at the end of the proof of Theorem 3.1. Smaller $\tilde{\delta}$ implies smaller η_k which further implies smaller $\|r^k\|$, i.e., the error in solving Newton's equation. So, smaller $\tilde{\delta}$ yields smaller theoretical convergence factor since we are approaching the Newton step, but on the other hand it increases the number of inner iterations which influences both communication and computational costs of the method.

IV. Distributed computation of Inexact Newton step

In this section we focus on the Step S1 in (10) of the algorithm INDO, for a fixed iteration k. We drop the iteration counter in the Hessian matrix $\mathbb{H}^k$ in (6) and remaining relevant quantities to simplify notation. That is, further on denote $\mathbb{H}^k = \mathbb{H}$, i.e.,

$$\mathbb{H} = \nabla^2 f(\mathbf{x}^k) + \alpha(\mathbb{I} - \mathbb{W}) + \varepsilon\mathbb{I}, \tag{18}$$

where $\varepsilon > 0$ is the proximal parameter and $\mathbf{g} = \mathbf{g}^k = \nabla f(\mathbf{x}^k) + \mathbf{q}^k + \alpha(\mathbb{I} - \mathbb{W})\mathbf{x}^k$. By assumption A2 the Hessian of the objective function f is positive definite with $m\mathbb{I} \preceq \nabla^2 f(\mathbf{x}^k)$, the matrix $\alpha(\mathbb{I} - \mathbb{W})$ is positive semidefinite and since $\varepsilon > 0$ we conclude that there holds

$$(m + \varepsilon)\mathbb{I} \preceq \mathbb{H}.$$

Notice that we can even make $\mathbb{H}$ strictly diagonally dominant by taking ε large enough. Furthermore, $\mathbb{H}$ has the same sparsity structure of the network and hence one can easily apply an iterative solver of the fixed point type in S1 of Algorithm INDO.

Let us look more closely at the step calculation in S1. For simplicity of exposition here we concentrate on the Jacobi Overrelaxation (JOR) method although other options for linear solver are possible. For each k we need to solve the system $\mathbb{H}\mathbf{d} = -\mathbf{g}$ approximately i.e. we are looking for $\mathbf{d}^k$ such that (10) holds. Thus we will consider the linear system

$$\mathbb{H}\mathbf{d} = -\mathbf{g}, \tag{19}$$

with $\mathbb{H}$ satisfying (18). With the splitting $\mathbb{H} = \mathbb{D} - \mathbb{G}$, where $\mathbb{D}$ is the diagonal part of $\mathbb{H}$, the Jacobi Overrelaxation matrix $\mathbb{T}_\gamma$ is defined as

$$\mathbb{T}_\gamma = \gamma\mathbb{D}^{-1}\mathbb{G} + (1 - \gamma)\mathbb{I} \tag{20}$$

and the JOR iterative method is defined as

$$\mathbf{d}^{\ell+1} = \mathbb{T}_\gamma\mathbf{d}^\ell - \gamma\mathbb{D}^{-1}\mathbf{g}, \tag{21}$$

with γ being the relaxation parameter, for arbitrary $\mathbf{d}^0$. For $\gamma = 1$ we get the Jacobi iterative method with the iterative matrix $\mathbb{T}_1 = \mathbb{D}^{-1}\mathbb{G}$.

The matrix $\mathbb{T}_\gamma$ has the same sparsity structure as $\mathbb{W}$ and all other matrices we considered so far, including the global Hessian. More precisely we can specify block-rows of $\mathbb{T}_\gamma$ that are used in the JOR method by each node as follows. Notice that, for matrix $\mathbb{D}$, we have its diagonal blocks $D_{ii} \in \mathbb{R}^{n\times n}, i = 1, \ldots, N$, given by

$$[D_{ii}]_j = \varepsilon + \alpha(1 - w_{ii}) + [\nabla^2 f_i(\mathbf{x}_i^k)]_{jj}, j = 1, \ldots, n, \tag{22}$$

where $[\nabla^2 f_i(x_i^k)]_{jj}$ denotes the j-th diagonal element of the local Hessian $\nabla^2 f_i(x_i^k)$. Similarly, with matrix G, consider its blocks $G_{ij} \in \mathbb{R}^{n\times n},\ i, j = 1, \ldots, N$. Then for $i \neq j$ we have that G_{ij} is diagonal with elements $\alpha w_{ij}\, I$ on the diagonal, and for $i = j$ we have

$$G_{ii} = diag(\nabla^2 f_i(x_i^k)) - \nabla^2 f_i(x_i^k), \tag{23}$$

with $diag(\nabla^2 f_i(x_i^k)) \in \mathbb{R}^{n\times n}$ being the diagonal of local Hessian $\nabla^2 f_i(x_i^k))$, for all $i = 1, \ldots, N$. Thus the matrix $\mathbb{T}_\gamma$ has block elements $[\mathbb{T}_\gamma]_{ij} = D_{ii}^{-1}G_{ij},\ i, j = 1, \ldots, N$. Let $[\mathbb{T}_\gamma]_i \in \mathbb{R}^{n\times nN}$ be the block row of $\mathbb{T}_\gamma,\ i = 1, \ldots, N$

$$[\mathbb{T}_\gamma]_i = [D_{ii}^{-1}G_{i1}, \ldots, D_{ii}^{-1}G_{ii}, \ldots, D_{ii}^{-1}G_{iN}]. \tag{24}$$

Similarly, we can define the partition of the gradient vector $\mathbf{g}$ with the ith component being

$$g_i^k = \nabla f_i(x_i^k) + q_i^k + \alpha[(1 - w_{ii})x_i^k - \sum_{j\in O_i} w_{ij}x_j^k],\ i = 1, \ldots, N. \tag{25}$$

Notice that each node i can compute $[\mathbb{T}_\gamma]_i$ and g_i^k.

Let us now state the $k+1$-th iteration of INDO algorithm in the node-wise manner.

Algorithm INDO-nodewise

Given : $\mathbf{x}^k, \mathbf{q}^k, \alpha > 0,\ \varepsilon > 0, \ell_k \geq 1, \ell \in \mathbb{N}$.

S1 Computing the direction $\mathbf{d}^k$

S1.1 Each node computes $[\mathbb{T}_\gamma]_i$ and g_i^k by (22) - (25) and chooses $d_i^0 \in \mathbb{R}^n$.

S1.2 Each node computes d_i^k as follows.

For $\ell = 0, \ldots, \ell_k - 1$

Each node sends d_i^ℓ to all its neighbors and receives $d_j^\ell, j \in O_i$.

Each node computes

$$d_i^{\ell+1} = [\mathbb{T}_\gamma]_i\mathbf{d}^\ell - \gamma D_{ii}^{-1}g_i^k \tag{26}$$

with $\mathbf{d}^\ell = (d_1^\ell, \ldots, d_N^\ell)$.

Endfor.

Set $d_i^k = d_i^{\ell_k}$.

S2 Primal update

S2.1 Each node updates the primal variable

$$x_i^{k+1} = x_i^k + d_i^k.$$

S2.2 Each node sends x_i^{k+1} to all its neighbours and receives $x_j^{k+1}, j \in O_i$.

S3 Each node computes the dual update

$$q_i^{k+1} = q_i^k + \alpha[(1 - w_{ii})x_i^{k+1} - \sum_{j\in O_i} w_{ij}x_j^{k+1}],$$

and sets $k = k + 1$.

Notice that Step 1.2 requires only the neighboring elements of $\mathbf{d}^\ell$ as $G_{ij} = 0$ if $w_{ij} = 0, i \neq j$, and thus the step is well defined and (26) is equivalent to (21). The algorithm above specifies a choice for Step S1 in the general INDO method in (10), but it is not straightforward to understand the connection. Namely, in Step 1.2 of the node-wise algorithm we state that each node should perform ℓ_k JOR iterations, while the main algorithm (see (10)) requires the step $\mathbf{d}^k$ such that the inexact forcing condition (10) with some $\eta_k > 0$ holds. In the sequel we first analyse the convergence conditions of the JOR method for solving (19) and then we show that one can in fact determine ℓ_k such that (10) holds after ℓ_k iterations in Step 1.2 of the above algorithm.

The JOR method is convergent for

$$\gamma \in (0, 2/\sigma(\mathbb{D}^{-1}\mathbb{H})) \tag{27}$$

for symmetric positive definite matrices, with $\sigma(\mathbb{D}^{-1}\mathbb{H})$ being the spectral radius of $\mathbb{D}^{-1}\mathbb{H}$, [11]. In the statement below we estimate the upper bound for the spectral radius of matrix $\mathbb{D}^{-1}\mathbb{H}$, using the block-wise Euclidean norm as in [2], [14]. For $\mathbb{T} \in \mathbb{R}^{nN\times nN}, \mathbb{T} = [T_{ij}], T_{ij} \in \mathbb{R}^{n\times n}$ we define

$$\|\mathbb{T}\|_{\mathrm{b}} = \max_{1\leq i\leq N} \sum_{j=1}^{N} \|T_{ij}\|_2. \tag{28}$$

Clearly, $\|\cdot\|_{\mathrm{b}}$ is a norm.

Proposition 4.1: Let Assumptions 1-3 hold and $\alpha > 0$. Then the Jacobi Overrrelaxation method (19) is convergent for

$$\gamma \in \left(0, 2\frac{m + \alpha(1 - w_d) + \varepsilon}{M + \varepsilon + \alpha(1 - w_m) + \alpha(1 - w_d)}\right),$$

where $w_d = \max_{1\leq i\leq N} w_{ii}$, $w_m = \min_{1\leq i\leq N} w_{ii}$.

Proof. Given the fact that for positive ε and α we have that $\mathbb{H}$ is positive definite, the convergence interval is determined by (27). On the other hand we have $\sigma(\mathbb{D}^{-1}\mathbb{H}) \leq \|\mathbb{D}^{-1}\mathbb{H}\|_{\mathrm{b}}$, so we have to estimate $\|\mathbb{D}^{-1}\mathbb{H}\|_{\mathrm{b}}$. By definition of $\mathbb{D}^{-1}\mathbb{H}$ and the norm we have

$$\begin{aligned}\|\mathbb{D}^{-1}\mathbb{H}\|_{\mathrm{b}} &= \max_{1\leq i\leq N} \sum_{j=1}^{N} \|[\mathbb{D}^{-1}\mathbb{H}]_{ij}\|_2 \\ &= \max_{1\leq i\leq N} \left(\|I - D_{ii}^{-1}G_{ii}\|_2 + \sum_{j\neq i} \| - D_{ii}^{-1}G_{ij}\|_2 \right).\end{aligned}$$

For each $i = 1, \ldots, N$ we have that D_{ii} is given by (22). Furthermore, the diagonal elements of local Hessian $\nabla^2 f_i(x_i^k)$ are in the interval $[m, M]$ by A2. Therefore

$$\|D_{ii}\|_2 \geq m + \alpha(1 - w_{ii}) + \varepsilon \geq m + \alpha(1 - w_d) + \varepsilon.$$

Next, for $i \neq j$, we have $\|G_{ij}\|_2 = \alpha\, w_{ij}$ and $\sum_{j\in O_i} w_{ij} = 1 - w_{ii} \leq 1 - w_m$ by the properties of W stated in A1. On the other hand, for $i = j$, we have by (23)

$$\|G_{ii}\|_2 \leq \|\nabla^2 f_i - diag(\nabla^2 f_i)\|_2.$$

It can be shown that $\|\nabla^2 f_i(x_i^k) - diag(\nabla^2 f_i(x_i^k))\|_2 \leq M - m$ and thus

$$\|G_{ii}\|_2 \leq M - m.$$

Combining the bounds for D_{ii} and G_{ii} we get

$$\begin{aligned}\|I - D_{ii}^{-1}G_{ii}\|_2 &\leq 1 + \|D_{ii}^{-1}G_{ii}\|_2 \\ &\leq 1 + \frac{M - m}{m + \alpha(1 - w_d) + \varepsilon} \\ &= \frac{M + \alpha(1 - w_d) + \varepsilon}{m + \alpha(1 - w_d) + \varepsilon}.\end{aligned}$$

Finally, using the bound above and the bounds for $\|D_{ii}^{-1}\|_2$ and $\|G_{ij}\|_2$, we get

$$\begin{aligned}\|\mathbb{D}^{-1}\mathbb{H}\|_{\mathrm{b}} &\leq \frac{\alpha(1 - w_m)}{m + \alpha(1 - w_d) + \varepsilon} + \frac{M + \alpha(1 - w_d) + \varepsilon}{m + \alpha(1 - w_d) + \varepsilon} \\ &= \frac{M + \varepsilon + \alpha(1 - w_m) + \alpha(1 - w_d)}{m + \alpha(1 - w_d) + \varepsilon},\end{aligned}$$

and the statement follows. □

Notice that the bound obtained above does not depend on k, and hence we can claim that the method in Step 1.2 of the node-wise algorithm will converge to the true solution $\mathbf{d}^*$ for an arbitrary $\mathbf{d}^0 \in \mathbb{R}^{nN}$ if we define the JOR matrix with the relaxation parameter γ as stated in Proposition 4.1.

The remaining question is the number of inner iterations in Step 1.2 ℓ_k that ensures that the forcing condition (10) is satisfied. In the statement bellow we will prove that such ℓ_k exists considering a special choice of d_i^0 for simplicity of exposition. Similar result can be proved for other choices of d_i^0.

Proposition 4.2: Let A1-A3 hold and assume that γ is chosen from the interval specified in Proposition 4.1. For given $\eta_k > 0$ denote by $\mathbf{d}^k$ a step obtained through Step 1.2 of the node-wise algorithm INDO with $d_i^0 = 0, i = 1, \ldots, N$ and ℓ_k inner iterations (26). Then $\|\mathbb{H}\mathbf{d}^k + \mathbf{g}\| \leq \eta_k\|\mathbf{g}\|$ after at most

$$\ell_k \geq \left\lceil \frac{\ln(\eta_k/c)}{|\ln(\sigma(\mathbb{T}_\gamma))|} \right\rceil,$$

where iterations, where $\lceil a \rceil$ denotes the smallest integer greater than or equal to a and

$$c := \frac{M + 2 + \varepsilon}{m + \varepsilon}\sqrt{\frac{\varepsilon + M + \alpha(1 - w_m)}{\varepsilon + m + \alpha(1 - w_d)}}.$$

Proof. Denote by $\mathbf{d}^*$ the exact solution of $\mathbb{H}\mathbf{d} = -\mathbf{g}$ and consider the sequence $\{d^\ell\}$ generated at Step 1.2 with $\mathbf{d}^0 = 0$. As $\mathbf{d}^{\ell+1} = \mathbb{T}_\gamma\mathbf{d}^\ell - \gamma\mathbb{D}^{-1}\mathbf{g}$, $\ell = 0, 1, \ldots$ and $\mathbf{d}^* = \mathbb{T}_\gamma\mathbf{d}^* - \gamma\mathbb{D}^{-1}\mathbf{g}$ we have

$$\mathbf{d}^\ell - \mathbf{d}^* = \mathbb{T}_\gamma^\ell(\mathbf{d}^0 - \mathbf{d}^*).$$

Notice that $\mathbb{T}_\gamma$ is not symmetric, but it has the same set of eigenvalues as the following symmetric matrix

$$\mathbb{T}'_\gamma = \mathbb{D}^{-1/2}(\gamma\mathbb{G} + (1 - \gamma)\mathbb{D})\mathbb{D}^{-1/2} := \mathbb{D}^{-1/2}\mathbb{C}_\gamma\mathbb{D}^{-1/2},$$

since $\mathbb{T}_\gamma = \mathbb{D}^{-1}\mathbb{C}_\gamma$ and both $\mathbb{D}^{-1}$ and $\mathbb{C}_\gamma$ are symmetric (see Remark 4.2 of [1] for instance). Therefore, we conclude

$$\sigma(\mathbb{T}_\gamma) = \sigma(\mathbb{T}'_\gamma) = \|\mathbb{T}'_\gamma\|$$

and according to the choice of γ and Proposition 4.1 we have $\sigma(\mathbb{T}_\gamma) < 1$. Moreover, it ca be shown that

$$\mathbb{T}_\gamma^\ell = \mathbb{D}^{-1/2}(\mathbb{T}'_\gamma)^\ell\mathbb{D}^{1/2}$$

and thus we obtain

$$\begin{aligned} \|\mathbf{d}^\ell - \mathbf{d}^*\| &\leq \|\mathbb{D}^{-1/2}\|\|\mathbb{T}'_\gamma\|^\ell\|\mathbb{D}^{1/2}\|\|\mathbf{d}^0 - \mathbf{d}^*\| \quad (29)\\ &= \|\mathbb{D}^{-1/2}\|(\sigma(\mathbb{T}_\gamma))^\ell\|\mathbb{D}^{1/2}\|\|\mathbf{d}^0 - \mathbf{d}^*\|. \end{aligned}$$

Considering that, by the structure of $\mathbb{D}$, we have

$$(\varepsilon + m + \alpha(1 - w_d))\mathbb{I} \preceq \mathbb{D} \preceq (\varepsilon + M + \alpha(1 - w_m))\mathbb{I},$$

we conclude that

$$\|\mathbb{D}^{-1/2}\|\|\mathbb{D}^{1/2}\| \leq \sqrt{\frac{\varepsilon + M + \alpha(1 - w_m)}{\varepsilon + m + \alpha(1 - w_d)}} := c_D.$$

Therefore, we obtain

$$\begin{aligned} \|\mathbf{d}^\ell - \mathbf{d}^*\| &\leq (\sigma(\mathbb{T}_\gamma))^\ell c_D\|\mathbf{d}^0 - \mathbf{d}^*\| && (30)\\ &= (\sigma(\mathbb{T}_\gamma))^\ell c_D\|\mathbb{H}^{-1}\mathbf{g}\| && (31)\\ &\leq (\sigma(\mathbb{T}_\gamma))^\ell c_D\|\mathbb{H}^{-1}\|\|\mathbf{g}\| && (32)\\ &\leq (\sigma(\mathbb{T}_\gamma))^\ell \frac{c_D}{m + \varepsilon}\|\mathbf{g}\|, && (33) \end{aligned}$$

since by A1 and A2 we can upper bound the Hessian and its inverse as

$$\|\mathbb{H}\| \leq M + 2 + \varepsilon,\ \|\mathbb{H}^{-1}\| < (m + \varepsilon)^{-1},$$

as $\|\mathbb{I} - \mathbb{W}\| \leq 2$. Finally, we have

$$\begin{aligned} &\|\mathbb{H}\mathbf{d}^\ell + \mathbf{g}\| = \|\mathbb{H}\mathbf{d}^\ell - \mathbb{H}\mathbf{d}^*\| \leq \|\mathbb{H}\|\|\mathbf{d}^\ell\\ -\ &\mathbf{d}^*\| \leq c_D\frac{M + 2 + \varepsilon}{m + \varepsilon}(\sigma(\mathbb{T}_\gamma))^\ell\|\mathbf{g}\| \end{aligned}$$

and we get

$$\|\mathbb{H}\mathbf{d}^{\ell_k} + \mathbf{g}\| \leq \eta_k\|\mathbf{g}\|$$

for ℓ_k given in the statement. □

The above statement implies that we can run INDO with the fixed number of inner iterations for all k, by taking $\ell_k = \ell(\bar{\eta})$ (see Theorem 3.1), to avoid the overhead needed to define suitable iteration-varying quantities η_k and ℓ_k. In fact, the numerical results we present in the next Section show that INDO behaves remarkably well if we take a very modest number of inner iterations at each outer iteration; namely, even $\ell_k = 1$ with the so-called warm start gives satisfactory results. In practice, this also alleviates the need to calculate the complicated expression for $\ell(\bar{\eta})$ as per Proposition 4.2 that depends on certain global quantities such as $\sigma(\mathbb{T}_\gamma)$.

Another possibility would be to define the forcing condition in the block infinity norm, i.e.,

$$\|\mathbb{H}\mathbf{d}^k + \mathbf{g}\|_\infty \leq \eta_k\|\mathbf{g}\|_\infty.$$

Then each node can compute the value of local residual $\|r_i^\ell\|_\infty$ at each ℓ; subsequently, at each ℓ, the nodes can run an iterative primitive for computing maximum (e.g., [42]) of their local residuals until the overall forcing condition in the block infinity norm is satisfied.

A couple of words on the forcing sequence $\{\eta_k\}$ are due here. Given that we are in the framework of the proximal method of multipliers, the most we can achieve is the overall linear convergence as the linear convergence is achieved even if the Newtonian linear systems are solved exactly i.e. for $\eta_k = 0$, see [21]. On the other hand we know from the classical optimization theory that η_k greatly influences the local convergence rate of Inexact Newton methods. In fact for $\eta_k \to 0$ we have superlinear local convergence while $\eta_k = \mathcal{O}(\|\mathbf{g}^k\|)$ recovers the quadratic convergence of the Newton method. In fact, one can show that the bound for $\bar{\eta}$ can be made arbitrarily close to 1 and still have a convergent method if a weighted norm is used, see [37]. But given the overall linear convergence at most in the assumed framework, it seems most efficient to work with a constant η_k during the whole process as such procedure maintains the linear rate of convergence while minimizing the effort needed for tuning η_k and ℓ_k. The key property – avoiding oversolving in the approximate Newton step calculation – is achieved with a constant η_k (and hence constant ℓ_k). The comparison with ESOM, both in terms of (inner iterations) convergence factor and numerically, presented in the next section strongly supports the approach we advocate here – approximate second order direction with the error proportional to the gradient value while incurring a small computational cost.

A. Inner iterations' convergence rate of INDO and ESOM

We provide here a more detailed comparison of INDO with the ESOM method proposed in [21] with respect to inner iterations' convergence. The two algorithms have identical dual variable updates, given by (8). Hence we focus on the primal variable updates. At each outer iteration k, both INDO and ESOM approximately solve the system of linear equations (19). While INDO solves (19) via (20)–(21), [21] adopts a different approach through a Taylor approximation; see equation (13) in [21]. However, the Taylor approximation approach admits a representation in the spirit of (20)–(21). Namely, it can be shown that the ESOM solver of (19) can be expressed as follows:

$$\mathbf{d}_E^{\ell+1} = \left(\mathbb{D}_E^{-1}\mathbb{B}\right)\mathbf{d}_E^\ell - \mathbb{D}_E^{-1}\mathbf{g}, \quad (34)$$

for $\ell = -1, 0, 1, ...$, with $\mathbf{d}_E^{-1} = 0$. Here, ESOM utilizes the splitting $\mathbb{H} = \mathbb{D}_E - \mathbb{B}$ of the Hessian $\mathbb{H}$ in (18), with

$$\begin{aligned} \mathbb{D}_E &= \nabla^2 f(\mathbf{x}^k) + 2\alpha\,(I - \operatorname{diag}(\mathbb{W})) + \varepsilon\, I \quad (35)\\ \mathbb{B} &= \alpha\,(I - 2\operatorname{diag}(\mathbb{W}) + \mathbb{W}). \quad (36) \end{aligned}$$

That is, an ESOM inner iteration ℓ in (34) is of a form similar to INDO; however, a major difference is that the utilized splitting involves a block-diagonal matrix $\mathbb{D}_E$ with non-sparse $n \times n$ diagonal blocks. As a result, (34) requires inverting a dense $n \times n$ positive definite matrix per node. In contrast, with the INDO approach in (20)–(21), the splitting matrix $\mathbb{D}$ is diagonal and hence no matrix inversion is required. A more detailed computational cost analysis per inner iteration for quadratic and logistic regression problems is presented in subsection 5.2.

We next compare convergence rates of inner iterations of ESOM and INDO. For both methods, it is easy to show that the error $\mathbf{c}^\ell = \mathbf{d}^\ell - \mathbf{d}^*$ with respect to the solution $\mathbf{d}^*$ of (18) evolves as:

$$\mathbf{c}^{\ell+1} = \mathbb{T}\mathbf{c}^\ell. \quad (37)$$

Here, for ESOM, we have that $\mathbf{c}^\ell = \mathbf{d}_E^\ell - \mathbb{H}^{-1}\mathbf{g}$, and $\mathbb{T} = \mathbb{D}_E^{-1}\mathbb{B}$; and for INDO, we have $\mathbf{c}^\ell = \mathbf{d}^\ell - \mathbb{H}^{-1}\mathbf{g}$, and

$\mathbb{T} = \mathbb{T}_\gamma = \gamma\,\mathbb{D}^{-1}\mathbb{G} + (1-\gamma)\mathbb{I}$. That is, provided that the spectral radius $\sigma(\mathbb{T})$ is less than one, $\mathbf{c}^\ell$ converges to zero linearly with the convergence factor determined by $\sigma(\mathbb{T})$. It is in general difficult to explicitly evaluate $\sigma(\mathbb{T})$ for INDO and ESOM. We hence compare the two methods in terms of the block-wise matrix norm upper bound $\|\mathbb{T}\|_{\mathrm{b}}$ on $\sigma(\mathbb{T})$ introduced in (28), that in turn admits intuitive upper bounds. Assuming for simplicity that all elements on the diagonal of W are mutually equal, $w_{ii} = w$, $i = 1, \ldots, N$, and that $\gamma = 1$ with INDO, it can be shown, similarly as in Proposition 4.1, that the following holds:

$$\text{ESOM}: \quad \|\mathbb{T}\|_{\mathrm{b}} \leq \frac{2\alpha\,(1-w)}{2\alpha\,(1-w)+\varepsilon+m} < 1 \quad (38)$$

$$\text{INDO}: \|\mathbb{T}\|_{\mathrm{b}} \leq \frac{M-m+\alpha\,(1-w)}{\alpha\,(1-w)+\varepsilon+m} < 1, \quad (39)$$

$$\text{for}\, \varepsilon > \max\{M-2m, 0\}.$$

We now comment on the convergence factor upper bounds in (38) and (39). First, we can see that, with both ESOM and INDO, the convergence factor can be made arbitrarily good (close to zero) by taking a sufficiently large ε. However, a too large ε comes at a price of slowing down the outer iterations, i.e., as too large ε makes small differences between $\mathbf{x}^{k+1}$ and $\mathbf{x}^k$, see equations (29) and (3). Second, we can see that, while we can take arbitrary (in fact, arbitrarily small) positive ε for ESOM, with INDO ε needs to be sufficiently large, i.e., we need to have $\varepsilon > \max\{M-2m, 0\}$. That is, by avoiding $n \times n$ matrix inversion at each node with INDO, we pay a price in that ε should be sufficiently large, i.e., of order M. However, extensive simulations in the next subsection show that this incurs no overall loss of INDO, i.e., INDO is comparable or faster iteration and communication-wise (and faster computational cost-wise) than ESOM with a best hand-tuned ε. Third, interestingly, when α and ε are large compared with M, INDO's inner iteration convergence factor upper bound (39) is comparable or even smaller than that of ESOM in (39). Extensive simulations show that taking α, ε to be of the same order and of the same order as M (in fact, we can take $\alpha = M = \varepsilon$) works well with INDO. This choice is in a good agreement with the theoretical upper bound in (39).

The recommended tuning parameter choice $\alpha = M = \varepsilon$ still requires a beforehand global knowledge of system parameters, specifically the constant M. Assuming that each node knows M_i–a Lipschitz constant of its own function f_i's gradient, $M = \max_{i=1,,,,,N} M_i$ can be obtained by running beforehand a distributed algorithm for computing maximum of scalar values held by the nodes, e.g., [42]. This can be done with a low overhead, wherein the number of required inter-neighbor (scalar-transmission) communication rounds is on the order of the network diameter. INDO as implemented in Section V also requires the beforehand knowledge of m and w_d; these quantities can be computed beforehand analogously to M.

V. Numerical results

In this section we analyze computational and communication costs of INDO and test its performance on a set of standard test examples. INDO is a second-order method based on the Proximal Method of Multipliers and differs significantly from the ESOM method, [21] in the primal variable construction update. Thus, our main aim is to investigate how INDO compares with ESOM. On the other hand, first order methods such as EXTRA [27] can also achieve exact linear convergence; henceforth, we compare INDO with EXTRA as a representative of exact first order methods.

First, we compare INDO with ESOM on quadratic cost functions (simulated data) and on logistic regression problems (real data). The network we consider has $N = 30$ nodes and is formed as follows, [16]. The points are sampled randomly and uniformly from $[0,1] \times [0,1]$. The edges between points exists if their Euclidean distance is smaller than $r = \sqrt{\log(N)/N}$. The resulting graph instance considered is connected. The weight coefficients in the communication matrix W are taken as $w_{i,j} = 1/(1 + \max\{deg(i), deg(j)\})$, where $deg(i)$ stands for the degree of node i, for directly connected nodes i and j, and the diagonal weights are $w_{i,i} = 1 - \sum_{j \neq i} w_{i,j}$. The matrix W generated in this way satisfies A1.

Let us now describe the test examples. Quadratic local cost functions are of the form

$$f_i(y) = \frac{1}{2}(y - b_i)^T B_{ii}(y - b_i) \quad (40)$$

and the data is simulated as in [16], i.e., vectors b_i are drawn from the Uniform distribution on $[1, 31]$, independently from each other. Matrices B_{ii} are of the form $B_{ii} = P_i S_i P_i$, where S_i are diagonal matrices with Uniform distribution on $[1, 101]$ and P_i are matrices of orthonormal eigenvectors of $\frac{1}{2}(C_i + C_i^T)$ where C_i have components drawn independently from the standard Normal distribution. Given that in this case we can compute the exact minimizer y^*, the error is measured as

$$E(\mathbf{x}^k) := \frac{1}{N}\sum_{i=1}^{N} \frac{\|x_i^k - y^*\|}{\|y^*\|}, \quad (41)$$

Both Algorithms require the Hessian lower and upper bound M, m and we calculate them as $M = \max_i M_i$, where M_i is the largest eigenvalue of B_{ii} and $m = \min_i m_i$, where $m_i > 0$ is the smallest eigenvalue of B_{ii}. The dimension of the problem is set to $n = 100$.

The two methods are also compared on binary classification problems and the following data sets: Mushrooms [32] ($n = 112$, total sample size $T = 8124$), LSVT Voice Rehabilitation [29] ($n = 309$, total sample size $T = 126$) and Parkinson's Disease Classification [25] ($n = 754$, total sample size $T = 756$). For each of the problems, the data is divided across 30 nodes of the graph described above. The logistic regression with the quadratic regularization is used and thus the local objective functions are of the form

$$f_i(y) = \frac{1}{|J_i|}\sum_{j \in J_i} \log(1 + e^{-\zeta_j p_j^T y}) + \frac{m}{2}\|y\|^2,$$

where J_i collects the indices of the data points assigned to node i, $p_j \in \mathbb{R}^n$ is the corresponding vector of attributes and $\zeta_j \in \{-1, 1\}$ represents the label. The data is scaled in a such way that $M = 1 + m$ with $m = 10^{-4}$. Since the solution is

unknown in general, the error is measured as the average value of the original objective function across the nodes' estimates

$$V(\mathbf{x}^k) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} f_j(x_i^k). \tag{42}$$

We fix the free parameters of INDO method to $\alpha = \varepsilon = M$. The reasoning behind this choice is explained in the previous section. We use this choice in all the tested examples, although it may not be the optimal choice. As discussed in the previous section, step S1 of INDO method can be implemented in different ways depending on the network characteristics, dimension of the problem, etc. We use a practical version of INDO in the tests by taking a fixed number ℓ of inner iterations of JOR method (e.g., $\ell = 1$) with $\gamma = 2(m + \varepsilon + \alpha(1 - w_d))/(M + 2\alpha + \varepsilon)$ denoted by INDO-ℓ in the sequel. Initial $\mathbf{d}^0$ is obtained by solving $\|\mathbb{H}^0\mathbf{d} + \mathbf{g}^0\|_\infty \leq \|\mathbf{g}^0\|_\infty$. In all the subsequent iterations we use the so called warm start, i.e., we set $\mathbf{d}^0 = \mathbf{d}^{k-1}$ at Step 1.2 in the node-wise INDO representation. Given that we implement the fixed number of inner iterations ℓ the forcing term η_k is not explicitly imposed.

Denote by ESOM-ℓ-α-ε the ESOM method with ℓ inner iterations and the corresponding free parameters α and ε. Notice that ESOM-ℓ requires the same amount of communications as INDO-ℓ.

Regarding the computational cost, the main advantage of INDO method with respect to ESOM is the following: ESOM requires inverting full $n\times n$ matrices at each node, while INDO only inverts the diagonal ones. The difference is more evident in non-quadratic case where the corresponding matrices need to be inverted in every outer iteration of ESOM method. This is even more significant in problems with relatively large n.

Let us now estimate the computational costs of the tested algorithms more precisely. We measure computational cost in terms of the total number of scalar (inner) products of vectors of size n incurred. Both ESOM and INDO are second order methods, so the cost of calculating the derivatives are the same. Moreover, the outer iterations updates are identical. The main difference lies in performing the inner iterations. In order to estimate the costs, let us observe the formula for INDO (26), i.e.,

$$d_i^{\ell+1} = \gamma D_{ii}^{-1}\left(G_{ii}d_i^\ell + \alpha\sum_{j\in O_i} w_{ij}d_j^\ell - g_i^k\right) + (1-\gamma)d_i^\ell. \tag{43}$$

and ESOM

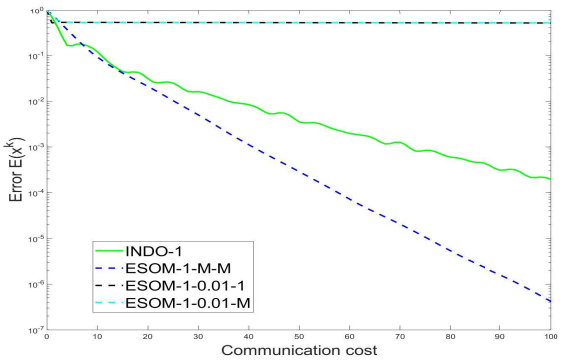
$$d_i^{\ell+1} = [D_E^{-1}]_{ii}\left(\alpha(1-w_{ii})d_i^\ell + \alpha\sum_{j\in O_i} w_{ij}d_j^\ell - g_i^k\right),$$
$$d_i^0 = -[D_E^{-1}]_{ii}g_i^k. \tag{44}$$

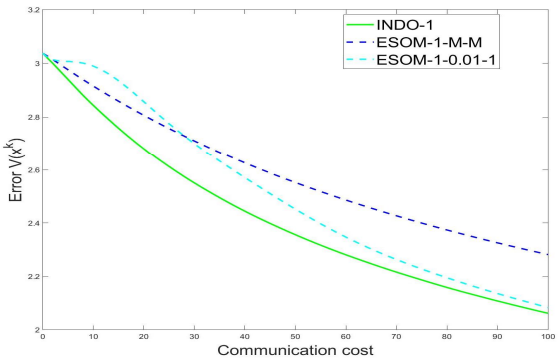
Both methods perform one consensus step ($\sum_{j\in O_i} w_{ij}d_j^\ell$) per inner iteration per node. Both methods perform one matrix-vector product per inner iteration per node as well: INDO calculates $G_{ii}d_i^\ell$ and ESOM calculates product of $[D_E^{-1}]_{ii}$ with the corresponding vector. The difference lies in the following. INDO inverts the diagonal matrix D_{ii} and multiplies it with the corresponding vector in each inner iteration, i.e., at each inner iteration we have component-wise division of two vectors. Thus, the cost of this operations can be estimated to $n\ell$ scalar products of vectors in $\mathbb{R}^n$ (SPs) per outer iteration per node for INDO-ℓ algorithm. On the other hand, ESOM calculates the inverse of possibly dense symmetric positive definite matrix $[D_E]_{ii}$ which can be estimated to $n^2/6$ SPs. For logistic regression problems, we estimate the common computational costs (see [17] for more details) of both algorithms to $|J_i|(2+n/2)+N+n\ell+N\ell/n$ SPs per node per (outer) iteration: $|J_i|(2+n/2)$ SPs is the cost of calculating the gradient and the Hessian of local cost function, $N = nN/n$ SPs comes from consensus step in calculating $\mathbf{g}^k$, matrix-vector products take $n\ell$ SPs and the consensus with respect to d_j vectors takes $\ell N/n$ SPs. Thus, the overall computational cost per node per iteration in logistic regression case is estimated to $|J_i|(2+n/2)+N+n\ell+N\ell/n+n\ell$ for INDO-ℓ and to $|J_i|(2+n/2)+N+n\ell+N\ell/n+n^2/6$ for ESOM-ℓ. Clearly, these costs differ by the order $\mathcal{O}(n)$. For the quadratic costs we do not have costs of calculating the local Hessian in every iteration while the cost of calculating the local gradient is n SPs. Other common costs are the same as in logistic regression case. INDO still needs $n\ell$ SPs in every iteration, while ESOM only inverts the Hessian in the initial phase with the cost of $n^2/6$ SPs.

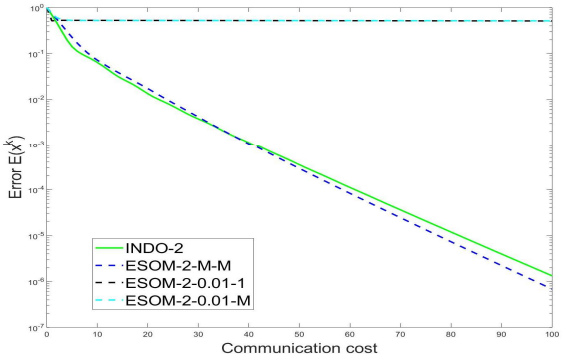
Figure 1 presents results on simulated quadratic costs for different number of inner iterations, i.e., with respect to communication cost–total number of n-dimensional vectors transmitted per node. We test different combinations of α and ε for ESOM method, while INDO is tested with fixed parameters as explained above. The best tested ESOM algorithm is ESOM-ℓ-M-M, and it outperforms the INDO method on this example. Other variants of ESOM start better, but they fail to converge to the same solution vicinity of the exact solution as the the remaining two methods.

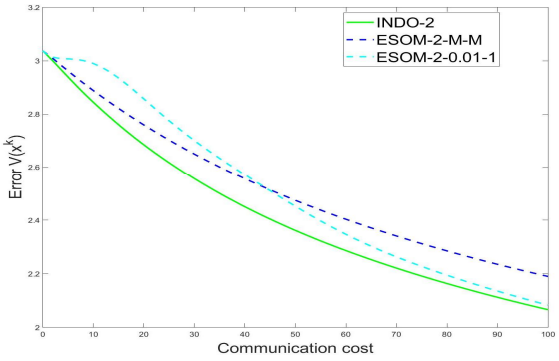
Figures 2-4 present analogous tests on logistic regression problems and the data sets Mushrooms, LSVT Voice Rehabilitation and Parkinson's Disease Classification. As already mentioned, for these problems $M = 1.0001$ so we omit the case $\alpha = 0.01, \varepsilon = 1$ for ESOM method in these tests. All the parameters for INDO algorithm are the same as in quadratic case. The results show that INDO algorithm is better or comparable with the ESOM algorithm with respect to iterations, i.e., communication cost. The difference seems to be bigger in the case of $\ell = 1$ than for the larger number of inner iterations $\ell = 2$. On the other hand, INDO outperforms ESOM method with respect to computational cost. Notice that for the Mushrooms dataset, n is relatively small with respect to $|J_i|$ and thus the common cost is the dominant cost, so the computational cost plots are very similar (in the sense of relative comparison INDO versus ESOM) to the ones that represent iterations and communication costs. For the remaining logistic regression problems, n is comparable to the total sample size T and thus the INDO's advantage in computational cost savings is more evident.

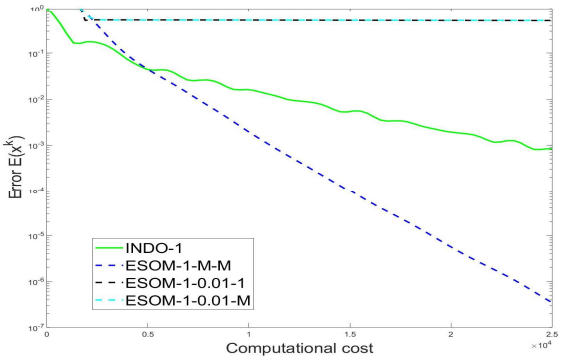
It is of significant interest to compare the methods on test examples with a high degree of data and local costs' heterogeneity across different nodes in the network. To this end,

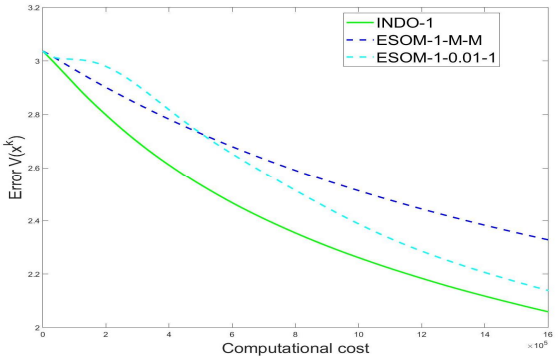


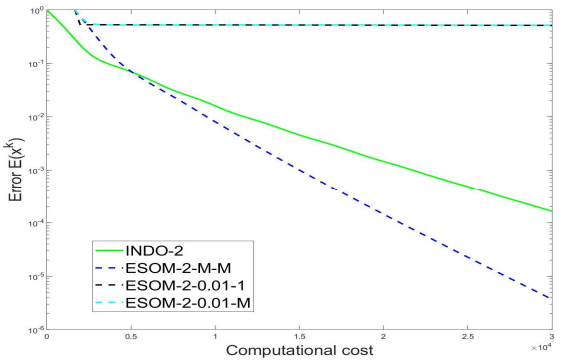












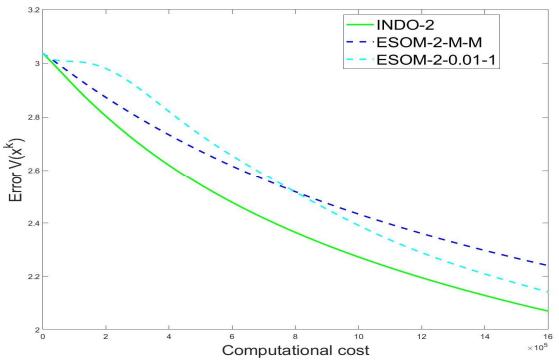
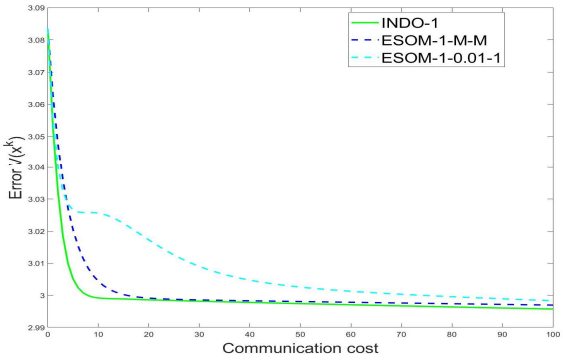
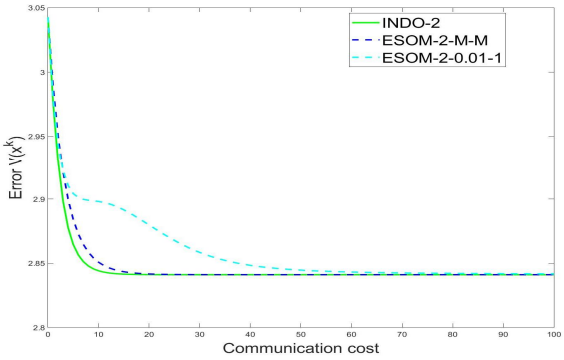
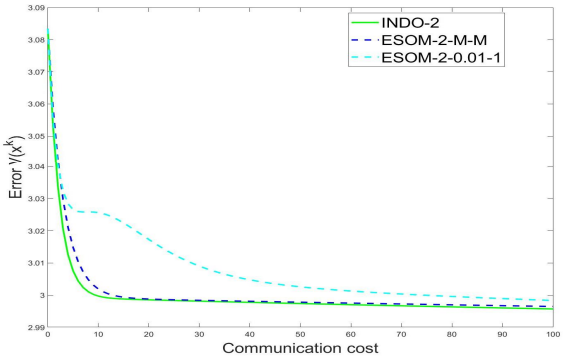


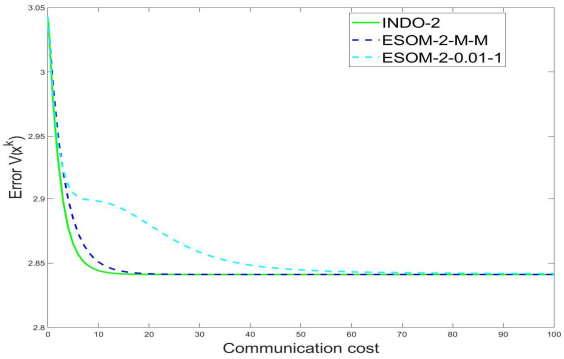
Fig. 1. INDO (solid line) versus ESOM methods (dotted lines): error (41) with respect to iterations/communication cost (the two top figures) and computational cost (the two bottom figures) for the number of inner iterations $\ell = 1$ (first and third figure from top) and $\ell = 2$ (second and fourth figure from top); Simulated quadratic costs (40) with $n = 100$ and $N = 30$.

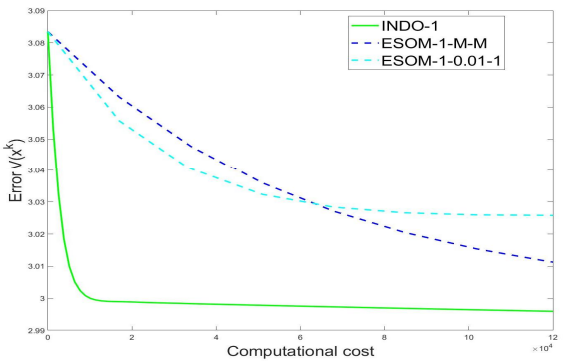
Fig. 2. INDO (solid line) versus ESOM methods (dotted lines): error (42) with respect to iterations/communication cost (the two top figures) and computational cost (the two bottom figures) for the number of inner iterations $\ell = 1$ (first and third figure from top) and $\ell = 2$ (second and fourth figure from top). Mushrooms dataset.

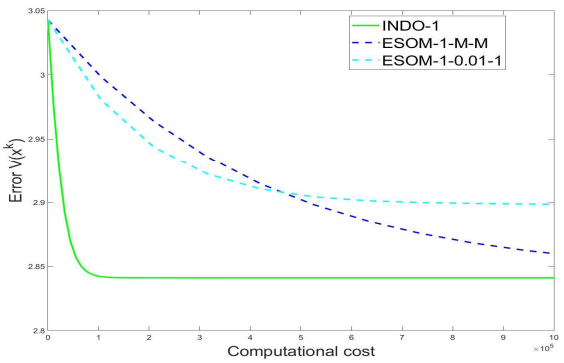


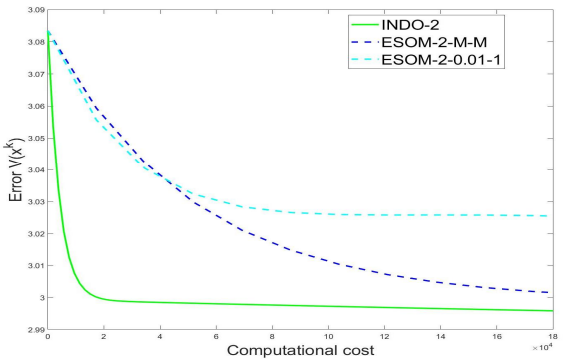












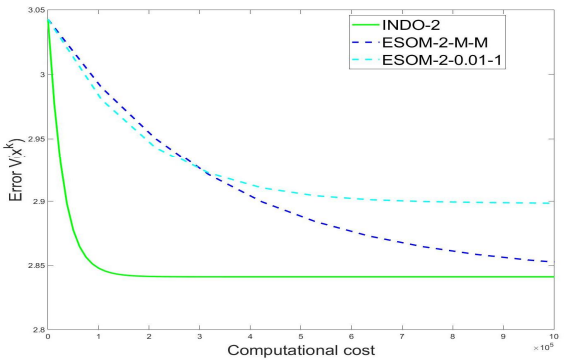


Fig. 3. INDO (solid line) versus ESOM methods (dotted lines): error (42) with respect to iterations/communication cost (the two top figures) and computational cost (the two bottom figures) for the number of inner iterations $\ell = 1$ (first and third figure from top) and $\ell = 2$ (second and fourth figure from top). LSVT Voice Rehabilitation dataset.

Fig. 4. INDO (solid line) versus ESOM methods (dotted lines): error (42) with respect to iterations/communication cost (the two top figures) and computational cost (the two bottom figures) for the number of inner iterations $\ell = 1$ (first and third figure from top) and $\ell = 2$ (second and fourth figure from top). Parkinson's Disease Classification dataset.

we compare INDO with ESOM on a test example with highly heterogeneous quadratic costs. In more detail, we use quadratic costs (40) where one half of nodes in the network have the B_{ii} matrices whose eigenvalues are generated from the standard Uniform distribution; the other half of nodes have the B_{ii} matrices whose eigenvalues are generated from the standard log-Normal distribution. All the vectors b_i are generated from the standard Normal distribution. The spectrum (the range that includes the minimal and the maximal eigenvalue of the matrix) varies from approximately $[0.002, 1]$ to $[0.04, 28]$; the corresponding condition numbers of the matrices vary from approximately 40 to 940. The test is performed on the same network with 30 nodes and the decision variable dimension $n = 100$. The results are shown in Figure 5. For this example, we can see that INDO exhibits better capabilities to cope with data and local functions heterogeneity than ESOM.

The same type of results is obtained on a a heterogeneous local costs example with the logistic regression losses. We continue to consider the same 30-node network and the Voice data set. However, now we vary the number of data points available at each node. More precisely, 5 nodes have 15 data points; 24 nodes have only two data points; and 1 node has 3 data points. Clearly, the local costs' minimizers can be very different here, where the nodes with 2 data points can have poor local classifiers (that correspond to local loss minimizers). Figure 6 shows the comparison of ESOM and INDO versus communication and computational costs. The results confirm significant advantages of INDO over ESOM computational cost-wise on the heterogeneous example also, while the two methods are still comparable communication cost-wise.

Now, we compare INDO with EXTRA as a representative first order method. The EXTRA update of the solution estimate sequence x_i^k at node i, using the notation in this paper is defined as:

$$x_i^0 = 0, \quad x_i^1 = \sum_{j=1}^{N} w_{ij} x_j^0 - \alpha g_i^0, \tag{45}$$

$$x_i^{k+2} = x_i^{k+1} + \sum_{j=1}^{N} w_{ij} x_j^{k+1} - \sum_{j=1}^{N} \tilde{w}_{ij} x_j^k - \alpha (g_i^{k+1} - g_i^k). \tag{46}$$

The standard choice of the matrix $\tilde{W} = (I + W)/2$ is used. Under the above setting, EXTRA requires only the x_j^k's exchange per iteration (broadcast of one n-dimensional vector per node), and thus its communication cost is $(\ell + 1)$ times smaller than the corresponding cost of INDO-ℓ, per iteration. The per-iteration computational cost of EXTRA is estimated analogously to the estimation of costs for INDO and ESOM. Namely, the cost per node per iteration of the EXTRA algorithm is $n + N/n$ and $|J_i|2 + N/n$ SPs for quadratic and logistic costs, respectively. Thus, it is obvious that EXTRA method is much cheaper than INDO per iteration, and hence in general it requires a smaller execution time per iteration, per node. We test INDO-1 against EXTRA with different choices of its step size α; we designate in plots the corresponding EXTRA variant as EXTRA-α. The considered network and the choice of the weight matrix W, same for both EXTRA and INDO, are same as before. The results are shown in Figures

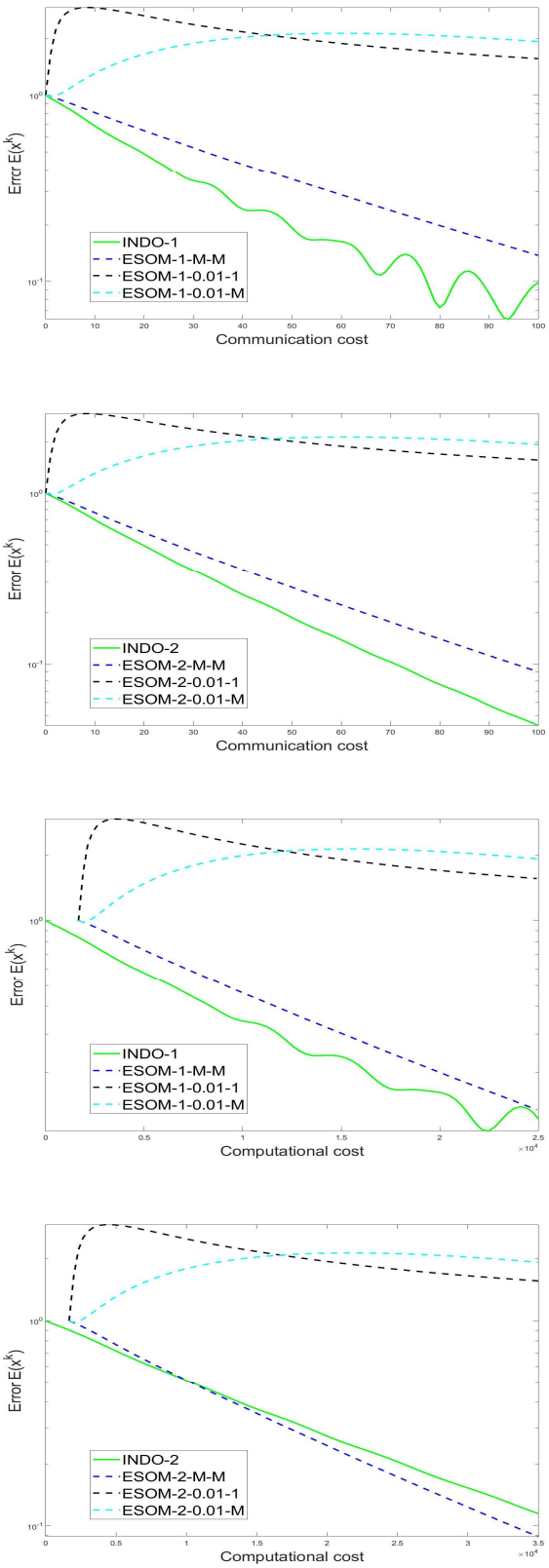


Fig. 5. INDO (solid line) versus ESOM methods (dotted lines): error (41) with respect to iterations/communication cost (the two top figures) and computational cost (the two bottom figures) for the number of inner iterations $\ell = 1$ (first and third figure from top) and $\ell = 2$ (second and fourth figure from top); Heterogeneous quadratic costs (40) with $n = 100$ and $N = 30$.

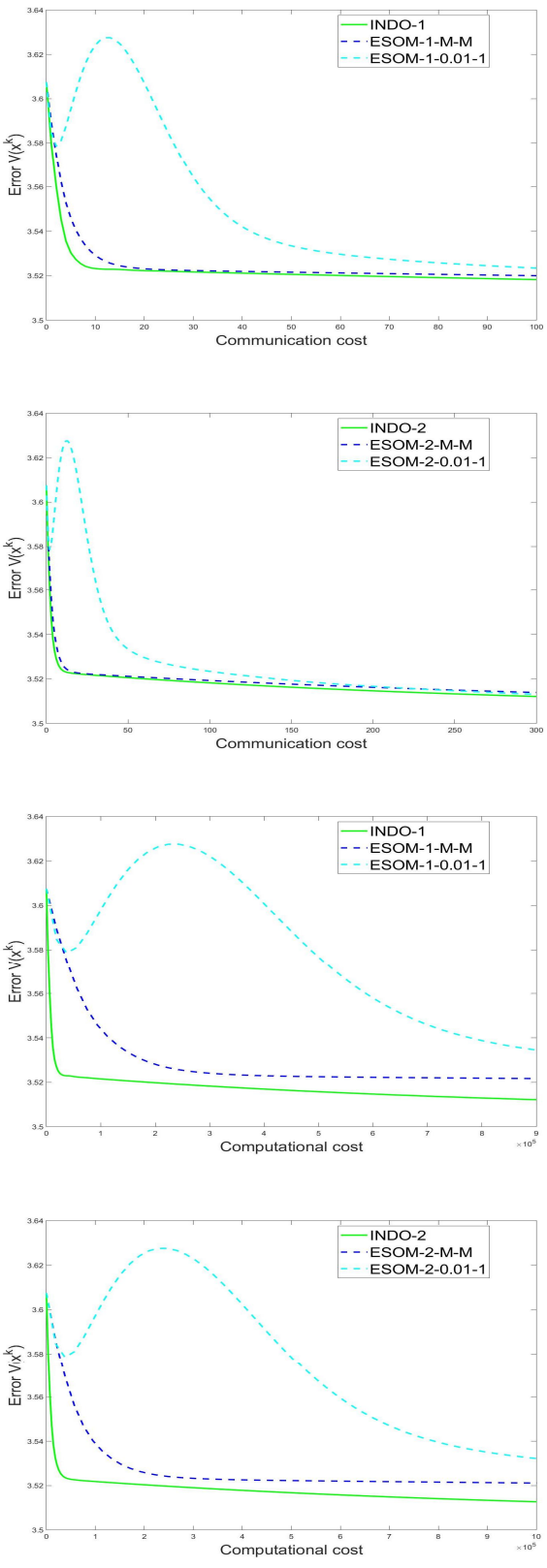


Fig. 6. INDO (solid line) versus ESOM methods (dotted lines): error (41) with respect to iterations/communication cost (the two top figures) and computational cost (the two bottom figures) for the number of inner iterations $\ell = 1$ (first and third figure from top) and $\ell = 2$ (second and fourth figure from top); Heterogeneous logistic regression costs, LSVT Voice Rehabilitation dataset with $N = 30$.

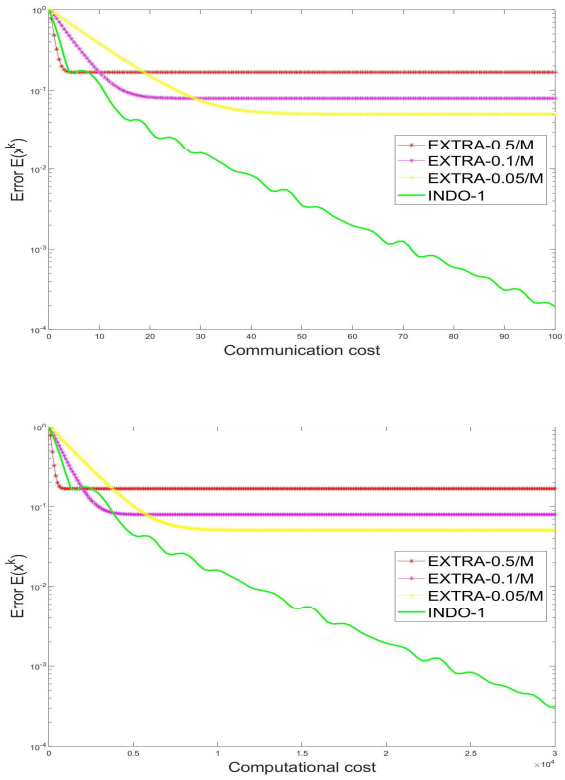


Fig. 7. INDO (solid line) versus EXTRA methods (marked lines): error (41) with respect to communication cost (top) and computational cost (bottom). Homogeneous quadratic costs (40) with $n = 100$ and $N = 30$.

7-9. Figure 7 presents results obtained on the homogeneous quadratic problem (40). The advantage of INDO is clear on this example. For the logistic regression problem (Figure 8), EXTRA outperforms INDO-1 in terms of computational cost, while the methods are at least comparable (or the comparison goes in favor of INDO) in terms of communication cost. Finally, INDO-1 outperforms EXTRA on the heterogeneous quadratic example described above (Figure 9), both in terms of computations and communications. This can be explained by the fact that, for a challenging heterogeneous problem with poor condition numbers here, second order cost functions' information, harnessed by INDO and ignored by (a first order method) EXTRA, seems to be crucial for good performance.

Figure 10 represents comparison of INDO performance on different graphs on simulated quadratic costs. We vary the number of nodes $N \in \{10, 20, 30, 40, 50\}$ and the connection radius $R \in \{0.7r, r, 2r\}$ with $r = \sqrt{\log(N)/N}$ which influences the spectral gap of the graph. We set the budget for computational cost to 10^4 SPs and report the resulting error of the method $E(x^k)$. Solid lines represent INDO-1 while dotted lines represent INDO-2 counterparts. INDO-2 seems to be a better option than INDO-1 in these circumstances, although INDO-2 requires more SPs per iteration. The advantage is even more evident when the graph has more links.

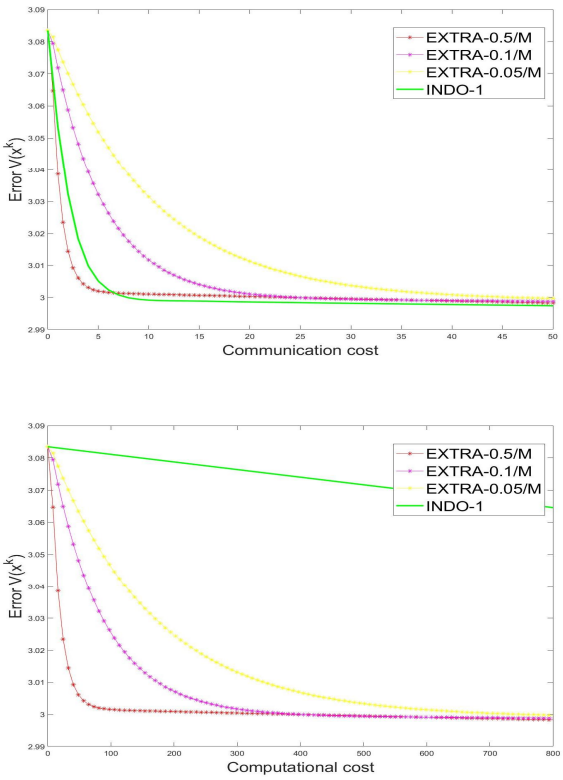


Fig. 8. INDO (solid line) versus EXTRA methods (marked lines): error (42) with respect to communication cost (top) and computational cost (bottom). LSVT Voice Rehabilitation dataset with $N = 30$.

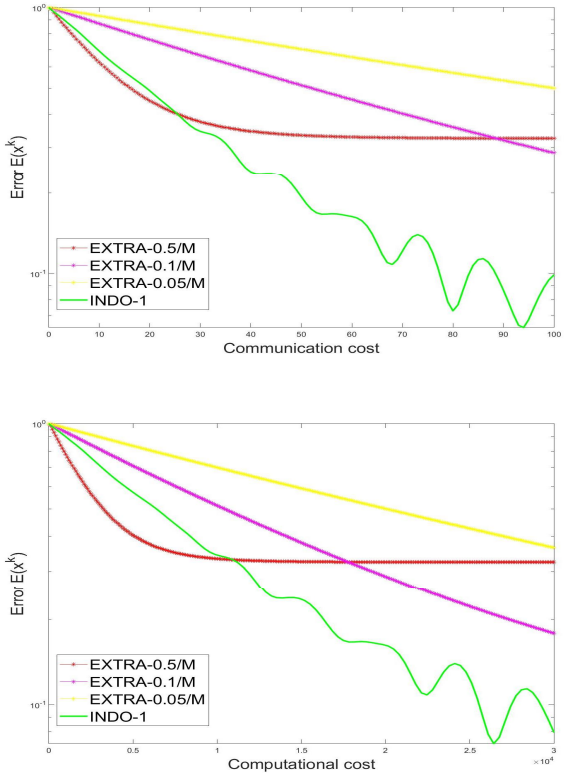


Fig. 9. INDO (solid line) versus EXTRA methods (marked lines): error (41) with respect to communication cost (top) and computational cost (bottom). Heterogeneous quadratic costs (40) with $n = 100$ and $N = 30$.

VI. Conclusions

In this paper, we proposed INDO, an exact distributed second order method for strongly convex distributed consensus optimization. INDO is design based on the framework of Proximal Method of Multipliers (PMM) and maintains a primal and a dual variable per iteration. Unlike existing exact distributed second order methods like ESOM [21] and DQM [22], INDO does not involve explicit Hessian inverse calculation when calculating the primal variable update. Instead, INDO calculates the primal variable update via a few inner iterations of a fixed point method (e.g., Jacobi overrelaxation) to approximately solve the system of linear equations that underlie the Newton direction evaluation; this update process alleviates the need for Newton inverse calculation. We prove that INDO achieves a global linear convergence to the exact solution of the problem of interest. Then, we provide analysis that reveals how the degree of inexactness in solving the Newton direction systems of linear equations affects the overall convergence rate. Furthermore, we provide intuitive bounds on the convergence factor of the involved linear system solver; these bounds shed light on the role of different method's parameters and provide guidelines on how these parameters should be set. Numerical experiments on several real data sets demonstrate that INDO achieves a comparable speed iteration-wise and communication cost-wise as ESOM, while at the same time reducing computational cost by at least an order of magnitude, when the dimension of the optimization variable is

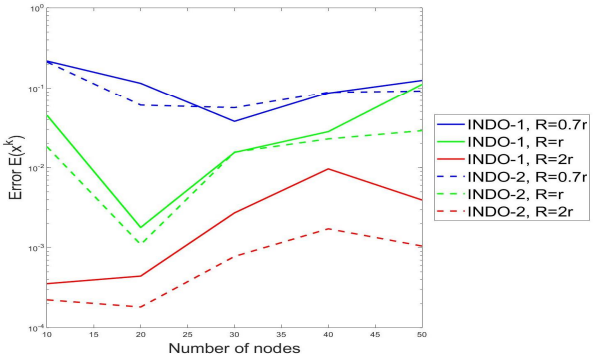


Fig. 10. Performance of INDO method on different graphs represented with different number of nodes and spectral gap controlled by the connection radius. INDO-1 (solid lines) vs INDO-2 (dotted lines). Simulated strongly convex quadratic costs (40).

on the order of couple of hundreds or larger. We also examined how INDO compares with EXTRA, a representative first order method, that does not involve Hessian calculations and hence has lower per-iteration computational cost. The results suggest that harnessing second order information is beneficial in terms of communication cost and can be beneficial for the overall computational cost, for heterogeneous and poorly conditioned quadratic problems.

REFERENCES

[1] Bai, Z., Silverstein, J. W., Spectral Analysis of Large Dimensional Random Matrices, Springer, 2010.

[2] Bajović, D., Jakovetić,D., Krejić, N., Krklec Jerinkić, N., Newton-like Method with Diagonal Correction for Distributed Optimization, SIAM Journal on Optimization, 27, 2 (2017), 1171-1203

[3] Baingana, B., Giannakis, G., B., Joint Community and Anomaly Tracking in Dynamic Networks, IEEE Transactions on Signal Processing, 64(8), (2016), pp. 2013-2025.

[4] Bellavia, S., Krejić, N., Krklec Jerinkić, N., Subsampled Inexact Newton Methods for minimizing large sums of convex functions, IMA J. Numer. Anal. 40,4 (2020), 2309-2341.

[5] Berahas, A. S., Bollapragada, R., Keskar, N. S., Wei, E., Balancing Communication and Computation in Distributed Optimization, IEEE Transactions on Automatic Control, 64(8), (2019), pp. 3141-3155.

[6] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning, 3(1), (2011) pp. 1-122.

[7] Cattivelli, F., Sayed, A. H., Diffusion LMS strategies for distributed estimation, IEEE Transactions on Signal Processing, 58(3), (2010) pp. 1035–1048.

[8] Eisen, M., Mokhtari, A., Ribeiro, A., Decentralized Quasi-Newton Methods, IEEE Trans. Signal Process, vol. 65, no. 10, pp. 2613–2628, 2017.

[9] Dembo, R.S., , Eisenstadt, S.C., Steihaugh, T., Inexact Newton Methods, SIAM Journal on Numerical Analysis, 19,2, (1982), 400-408.

[10] Eisen, M., Mokhtari, A., Ribeiro, A., A Primal-Dual Quasi-Newton Method for Exact Consensus Optimization, IEEE Transactions on Signal Processing, Vol. 67, No. 23, Dec. 2019., pp. 5983–5997.

[11] Greenbaum, A., Iterative Methods for Solving Linear Systems, SIAM, 1997.

[12] Jakovetić, D., A Unification and Generalization of Exact Distributed First Order Methods, IEEE Transactions on Signal and Information Processing over Networks, 5(1), (2019), pp. 31-46.

[13] Jakovetić, D., Bajović, D., Xavier, J., Moura, J.M.F., Primal-Dual Methods for Large-Scale and Distributed Convex Optimization and Data Analytics, Procedings of the IEEE, 108,11 (2020), 1923-1938.

[14] Jakovetić, D., Krejić, N., Krklec Jerinkić, N. , Malaspina, G. , Micheletti, A., Distributed Fixed Point Method for Solving Systems of Linear Algebraic Equations, Automatica (2021), Vol. 134.

[15] Jakovetić, D., Xavier, J., Moura, J. M. F., Fast distributed gradient methods, *IEEE Transactions on Automatic Control*, 59(5), (2014) pp. 1131–1146.

[16] Jakovetić, D., Krejić, N., Krklec Jerinkić, N., Exact spectral-like gradient method for distributed optimization, Computational Optimization and Applications, 74, (2019), pp. 703–728.

[17] Jakovetić, D., Krejić, N., Krklec Jerinkić, N., EFIX: Exact Fixed Point Methods for Distributed Optimization, arxiv preprint, arXiv:2012.05466v1 (2020).

[18] Lee, J. M., Song, I., Jung, S., Lee, J., A rate adaptive convolutional coding method for multicarrier DS/CDMA systems, MILCOM 2000 Proceedings 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No.00CH37155), Los Angeles, CA, (2000), pp. 932-936.

[19] Mansoori, F., Wei, E., A Fast Distributed Asynchronous Newton-Based Optimization Algorithm, IEEE Transactions on Automatic Control, Vol. 65, No. 7, July 2020, pp. 2769–2784.

[20] Mokhtari, A., Ling, Q., Ribeiro, A., Network Newton Distributed Optimization Methods, IEEE Transactions on Signal Processing, 65,1 (2017), 146-161.

[21] Mokhtari, A., Shi, W., Ling, Q., Ribeiro, A., A Decentralized Second Order Method with Exact Linear Convergence Rate for Consensus Optimization, IEEE Transactions on Signal and Information Processing over Networks, 2(4), (2016), pp. 507-522.

[22] Mokhtari, A., Shi, W., Ling, Q., Ribeiro, A., DQM: Decentralized quadratically approximated alternating direction method of multipliers, IEEE Transactions on Signal Processing, vol. 64, no. 19, pp. 5158-5173, Oct. 2016.

[23] Mota, J., Xavier, J., Aguiar, P., Püschel, M., Distributed optimization with local domains: Applications in MPC and network flows, IEEE Transactions on Automatic Control, 60(7), (2015), pp. 2004-2009.

[24] Qu, G., Li, N., Harnessing smoothness to accelerate distributed optimization, IEEE Transactions on Control of Network Systems, 5(3), (2018), pp. 1245-1260.

[25] Sakar, C., Serbes, Gorkem, Gunduz, Aysegul, Nizam, Hatice, Sakar, Betul. (2018). Parkinson's Disease Classification. UCI Machine Learning Repository.

[26] Scutari, G., Sun, Y., Parallel and Distributed Successive Convex Approximation Methods for Big-Data Optimization, Multi-agent Optimization, Lecture Notes in Mathematics, pp. 141-308, Springer, 2018.

[27] Shi, W., Ling, Q., Wu, G., Yin, W., EXTRA: an Exact First-Order Algorithm for Decentralized Consensus Optimization, SIAM Journal on Optimization, 2(25), (2015), pp. 944-966.

[28] Sun, Y., Daneshmand, A., Scutari, G., Convergence Rate of Distributed Optimization Algorithms based on Gradient Tracking, arXiv:1905.02637, (2019).

[29] Tsanas, Athanasios. (2014). LSVT Voice Rehabilitation. UCI Machine Learning Repository

[30] Tian, Y., Sun, Y., Scutari, G., Achieving Linear Convergence in Distributed Asynchronous Multi-agent Optimization, IEEE Trans. on Automatic Control, (2020).

[31] Tian, Y., Sun, Y., Scutari, G., Asynchronous Decentralized Successive Convex Approximation, arXiv:1909.10144, (2020).

[32] UCI Machine Learning Expository, https://archive.ics.uci.edu/ml/datasets/Mushroom.

[33] Xin, R., Khan, U. A., Distributed Heavy-Ball: A Generalization and Acceleration of First-Order Methods With Gradient Tracking, IEEE Transactions on Automatic Control, 65(6), (2020), pp. 2627-2633.

[34] Xu, J., Tian, Y., Sun, Y., Scutari G., Accelerated primal-dual algorithms for distributed smooth convex optimization over networks, International Conference on Artificial Intelligence and Statistics, PMLR, (2020), pp. 2381-2391.

[35] Xu, J., Tian, Y., Sun, Y., Scutari G., Distributed Algorithms for Composite Optimization: Unified Framework and Convergence Analysis, IEEE Transactions on Signal Processing, Vol. 69, pp. , June 2021.3555 - 3570, June 2021.

[36] Yousefian, F., Nedić, A., Shanbhag, U. V., On stochastic gradient and subgradient methods with adaptive steplength sequences, Automatica, 48(1), (2012), pp. 56-67.

[37] Ypma, T., Local convergence of Inexact Newton Methods, SIAM Journal on Numerical Analysis, 21(3), (1984), 583–590.

[38] Yuan, K., Ying, B., Zhao, X., Sayed, A. H., Exact diffusion for distributed optimization and learning – Part I: Algorithm development, IEEE Transactions on Signal Processing, Vol. 67, No. 3, Feb., 2019., pp. 708–723.

[39] Yuan, K., Ying, B., Zhao, X., Ali H. Sayed, Exact Diffusion for Distributed Optimization and Learning – Part II: Convergence Analysis, IEEE Trans. Signal Process., vol. 67, No. 3, pp. 724–739, 2019.

[40] Xin, R., Khan, U. A., and Kar, S., Variance-reduced decentralized stochastic optimization with accelerated convergence, IEEE Transactions on Signal Processing, vol. 68, pp. 6255–6271, Oct. 2020.

[41] Zhang, J., Ling, Q., So, A.-M., A Newton Tracking Algorithm with Exact Linear Convergence Rate for Decentralized Consensus Optimization, IEEE Transactions on Signal and Information Processing over Networks, 7, (2021), pp. 346–358.

[42] Zhang, S., Tepedelenlioglu, C., Banavar, M.K., Spanias, A., Max Consensus in Sensor Networks: Non-Linear Bounded Transmission and Additive Noise, IEEE Sensors Journal, Vol. 16, No. 24, pp. 9089–9098, Dec. 2016.

APPENDIX

Proof of Lemma 3.1 Starting from (8) and adding the zero term from (12) multiplied by $\alpha(\mathbb{I}-\mathbb{W})^{1/2)}$, i.e., $\alpha(\mathbb{I}-\mathbb{W})\mathbf{x}^*$, implies the statement. □

Proof of Lemma 3.2 By the definition of step $\mathbf{d}^k$ in S2, (8) and (11) we have

$$\begin{aligned}
0 &= \mathbb{H}^k\mathbf{d}^k + \mathbf{g}^k - \mathbf{r}^k \\
&= (\nabla^2 f(\mathbf{x}^k) + \alpha(\mathbb{I}-\mathbb{W}) + \varepsilon\mathbb{I})\mathbf{d}^k \\
&+ \mathbf{g}^k - \mathbf{r}^k \pm \mathbf{q}^{k+1} \\
&= \nabla^2 f(\mathbf{x}^k)\mathbf{d}^k + \alpha(\mathbb{I}-\mathbb{W})\mathbf{d}^k + \varepsilon\mathbf{d}^k + \nabla f(\mathbf{x}^k) + \mathbf{q}^k \\
&+ \alpha(\mathbb{I}-\mathbb{W})\mathbf{x}^k - \mathbf{r}^k \pm \mathbf{q}^{k+1} \\
&= \nabla^2 f(\mathbf{x}^k)\mathbf{d}^k + \alpha(\mathbb{I}-\mathbb{W})\mathbf{x}^{k+1} + \varepsilon\mathbf{d}^k + \nabla f(\mathbf{x}^k) \\
&+ -\mathbf{r}^k + \mathbf{q}^k - (\nabla f(\mathbf{x}^*) + \mathbf{q}^*) \pm \nabla f(\mathbf{x}^{k+1}) \\
&= \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*) + \varepsilon\mathbf{d}^k \\
&+ (\mathbf{q}^{k+1} - \mathbf{q}^*) + \mathbf{e}^k.
\end{aligned}$$

□

Proof of Lemma 3.3. By Lemma 2.1, Lemma 3.2 and the condition in S2 of the algorithm we have

$$\begin{aligned}
\|\mathbf{e}^k\| &\leq \|\nabla^2 f(\mathbf{x}^k)\mathbf{d}^k + \nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k+1})\| + \|\mathbf{r}^k\| \\
&\leq \min\{2M, \frac{L}{2}\|\mathbf{d}^k\|\}\|\mathbf{d}^k\| + \eta_k\|\mathbf{g}^k\|. \qquad (47)
\end{aligned}$$

Furthermore, the definition of $\mathbf{g}^k$ in (9) and optimality conditions (11), (12) imply

$$\begin{aligned}
\|\mathbf{g}^k\| &= \|\nabla f(\mathbf{x}^k) + (\mathbb{I}-\mathbb{W})^{1/2}\mathbf{v}^k + \alpha(\mathbb{I}-\mathbb{W})\mathbf{x}^k \\
&- (\nabla f(\mathbf{x}^*) + (\mathbb{I}-\mathbb{W})^{1/2}\mathbf{v}^*) - \alpha(\mathbb{I}-\mathbb{W})\mathbf{x}^*\| \\
&\leq \|\nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^*)\| + \|(\mathbb{I}-\mathbb{W})^{1/2}(\mathbf{v}^k - \mathbf{v}^*)\| \\
&+ \|\alpha(\mathbb{I}-\mathbb{W})(\mathbf{x}^k - \mathbf{x}^*)\| \\
&\leq M\|\mathbf{x}^k - \mathbf{x}^*\| + \sqrt{2}\|\mathbf{v}^k - \mathbf{v}^*\| \\
&+ 2\alpha\|\mathbf{x}^k - \mathbf{x}^*\| \\
&= (M+2\alpha)\|\mathbf{x}^k - \mathbf{x}^*\| + \sqrt{2}\|\mathbf{v}^k - \mathbf{v}^*\|.
\end{aligned}$$

Placing the last inequality into (47) we get the statement. □

Proof of Lemma 3.5. By assumption A2 we have that the aggregate function f is strongly convex with constant m and its gradient ∇f is Lipschitz continuous with constant M. Therefore the following inequality holds

$$\begin{aligned}
&\frac{mM}{m+M}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + \frac{1}{m+M}\|\nabla f(\mathbf{x}^{k+1}) \qquad (48) \\
- &\nabla f(\mathbf{x}^*)\|^2 \\
\leq &(\mathbf{x}^{k+1} - \mathbf{x}^*)^T(\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)).
\end{aligned}$$

On the other hand from Lemma 3.2 we have

$$\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*) = -\varepsilon\mathbf{d}^k - (\mathbb{I}-\mathbb{W})^{1/2}(\mathbf{v}^{k+1} - \mathbf{v}^*) - \mathbf{e}^k$$

Putting the right hand side of this inequality back in (48) and multiplying with 2α yields

$$\begin{aligned}
&\frac{2\alpha mM}{m+M}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \qquad (49) \\
+ &\frac{2\alpha}{m+M}\|\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)\|^2 \\
\leq &-2\alpha(\mathbf{x}^{k+1} - \mathbf{x}^*)^T(\mathbb{I}-\mathbb{W})^{1/2}(\mathbf{v}^{k+1} - \mathbf{v}^*) \\
- &2\alpha\varepsilon(\mathbf{x}^{k+1} - \mathbf{x}^*)^T\mathbf{d}^k - 2\alpha(\mathbf{x}^{k+1} - \mathbf{x}^*)^T\mathbf{e}^k.
\end{aligned}$$

Now, Lemma 3.1 implies $\alpha(\mathbf{x}^{k+1} - \mathbf{x}^*)^T(\mathbb{I}-\mathbb{W})^{1/2} = (\mathbf{v}^{k+1} - \mathbf{v}^k)^T$, and therefore

$$\begin{aligned}
&\frac{2\alpha mM}{m+M}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \qquad (50) \\
+ &\frac{2\alpha}{m+M}\|\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)\|^2 \\
\leq &-2(\mathbf{v}^{k+1} - \mathbf{v}^k)^T(\mathbf{v}^{k+1} - \mathbf{v}^*) - 2\alpha\varepsilon(\mathbf{x}^{k+1} - \mathbf{x}^*)^T\mathbf{d}^k \\
- &2\alpha(\mathbf{x}^{k+1} - \mathbf{x}^*)^T\mathbf{e}^k.
\end{aligned}$$

Notice further that

$$\begin{aligned}
&2(\mathbf{v}^{k+1} - \mathbf{v}^k)^T(\mathbf{v}^{k+1} - \mathbf{v}^*) = \|\mathbf{v}^{k+1} - \mathbf{v}^k\|^2 \\
+ &\|\mathbf{v}^{k+1} - \mathbf{v}^*\|^2 - \|\mathbf{v}^k - \mathbf{v}^*\|^2
\end{aligned}$$

and

$$2(\mathbf{x}^{k+1} - \mathbf{x}^*)^T\mathbf{d}^k = \|\mathbf{d}^k\|^2 + \|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 - \|\mathbf{x}^k - \mathbf{x}^*\|^2.$$

Going back to (50) we get

$$\begin{aligned}
&\frac{2\alpha mM}{m+M}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 + \frac{2\alpha}{m+M}\|\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)\|^2 \\
\leq &-\|\mathbf{v}^{k+1} - \mathbf{v}^k\|^2 - \|\mathbf{v}^{k+1} - \mathbf{v}^*\|^2 + \|\mathbf{v}^k - \mathbf{v}^*\|^2 \qquad (51) \\
- &\alpha\varepsilon\|\mathbf{d}^k\|^2 - \alpha\varepsilon\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \\
+ &\alpha\varepsilon\|\mathbf{x}^k - \mathbf{x}^*\|^2 - 2\alpha(\mathbf{x}^{k+1} - \mathbf{x}^*)^T\mathbf{e}^k.
\end{aligned}$$

Lemma 3.1 implies that $\|\mathbf{v}^{k+1} - \mathbf{v}^k\|^2 = \|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2_{\alpha^2(\mathbb{I}-\mathbb{W})}$ and by definition of Lyapunov function we have $\|\mathbf{u}^k - \mathbf{u}^*\|^2_{\mathcal{G}} - \|\mathbf{u}^{k+1} - \mathbf{u}^*\|^2_{\mathcal{G}} = \|\mathbf{v}^k - \mathbf{v}^*\|^2 - \|\mathbf{v}^{k+1} - \mathbf{v}^*\|^2 + \alpha\varepsilon\|\mathbf{x}^k - \mathbf{x}^*\|^2 - \alpha\varepsilon\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2$. Thus, inequality (53) reduces to

$$\begin{aligned}
&\frac{2\alpha mM}{m+M}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \qquad (52) \\
+ &\frac{2\alpha}{m+M}\|\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)\|^2 \\
\leq &\|\mathbf{u}^k - \mathbf{u}^*\|^2_{\mathcal{G}} - \|\mathbf{u}^{k+1} - \mathbf{u}^*\|^2_{\mathcal{G}} - \alpha\varepsilon\|\mathbf{d}^k\|^2 \\
- &\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2_{\alpha^2(\mathbb{I}-\mathbb{W})} \\
- &2\alpha(\mathbf{x}^{k+1} - \mathbf{x}^*)^T\mathbf{e}^k.
\end{aligned}$$

Regrouping the terms in the above inequality yields

$$\begin{aligned}
&\|\mathbf{u}^{k+1} - \mathbf{u}^*\|^2_{\mathcal{G}} - \|\mathbf{u}^k - \mathbf{u}^*\|^2_{\mathcal{G}} \qquad (53) \\
\leq &-\frac{2\alpha}{m+M}\|\nabla f(\mathbf{x}^{k+1}) \\
- &\nabla f(\mathbf{x}^*)\|^2 - \|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2_{\frac{2\alpha mM}{m+M}\mathbb{I}+\alpha^2(\mathbb{I}-\mathbb{W})} \\
- &\alpha\varepsilon\|\mathbf{d}^k\|^2 - 2\alpha(\mathbf{x}^{k+1} - \mathbf{x}^*)^T\mathbf{e}^k.
\end{aligned}$$

Using the bound $2(\mathbf{x}^{k+1} - \mathbf{x}^*)^T\mathbf{e}^k \geq -1/\zeta\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 - \zeta\|\mathbf{e}^k\|^2$, which holds for any $\zeta > 0$, we get

$$\begin{aligned}
&\|\mathbf{u}^{k+1} - \mathbf{u}^*\|^2_{\mathcal{G}} - \|\mathbf{u}^k - \mathbf{u}^*\|^2_{\mathcal{G}} \qquad (54) \\
\leq &-\frac{2\alpha}{m+M}\|\nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^*)\|^2 - \|\mathbf{x}^{k+1} \\
- &\mathbf{x}^*\|^2_{(\frac{2\alpha mM}{m+M} - \frac{\alpha}{\zeta})\mathbb{I}+\alpha^2(\mathbb{I}-\mathbb{W})} \\
- &\alpha\varepsilon\|\mathbf{d}^k\|^2 + \alpha\zeta\|\mathbf{e}^k\|^2.
\end{aligned}$$