Journal of Machine Learning Research 21 (2020) 1-51 Submitted 3/20; Revised 8/20; Published 8/20

Communication-Efficient Distributed Optimization in Networks with Gradient Tracking and Variance Reduction

Boyue Li, Shicong Cen {BOYUEL, SHICONGC}@ANDREW.CMU.EDU
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Yuxin Chen YUXIN.CHEN@PRINCETON.EDU
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544, USA

Yuejie Chi YUEJIECHI@CMU.EDU
Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Editor: Michael Mahoney

Abstract

There is growing interest in large-scale machine learning and optimization over decentralized networks, e.g. in the context of multi-agent learning and federated learning. Due to the imminent need to alleviate the communication burden, the investigation of communication-efficient distributed optimization algorithms — particularly for empirical risk minimization — has flourished in recent years. A large fraction of these algorithms have been developed for the master/slave setting, relying on the presence of a central parameter server that can communicate with all agents.

This paper focuses on distributed optimization over networks, or decentralized optimization, where each agent is only allowed to aggregate information from its neighbors over a network (namely, no centralized coordination is present). By properly adjusting the global gradient estimate via local averaging in conjunction with proper correction, we develop a communication-efficient approximate Newton-type method, called `Network-DANE`, which generalizes DANE to accommodate decentralized scenarios. Our key ideas can be applied, in a systematic manner, to obtain decentralized versions of other master/slave distributed algorithms. A notable development is `Network-SVRG/SARAH`, which employs variance reduction at each agent to further accelerate local computation. We establish linear convergence of `Network-DANE` and `Network-SVRG` for strongly convex losses, and `Network-SARAH` for quadratic losses, which shed light on the impacts of data homogeneity, network connectivity, and local averaging upon the rate of convergence. We further extend `Network-DANE` to composite optimization by allowing a nonsmooth penalty term. Numerical evidence is provided to demonstrate the appealing performance of our algorithms over competitive baselines, in terms of both communication and computation efficiency. Our work suggests that by performing a judiciously chosen amount of local communication and computation per iteration, the overall efficiency can be substantially improved.

Keywords: decentralized optimization, federated learning, communication efficiency, gradient tracking, variance reduction

1. Introduction

Distributed optimization has been a classic topic (Bertsekas and Tsitsiklis, 1989), yet is attracting significant attention recently in machine learning due to its numerous applications such as distributed training (Boyd et al., 2011), multi-agent learning (Nedic et al., 2010), and federated learning (Konečný et al., 2015, 2016; McMahan et al., 2017). At least two facts contribute towards this resurgence of interest: (1) the scale of modern datasets has oftentimes far exceeded the capacity of a single machine and requires coordination across multiple machines; (2) privacy and communication constraints disfavor information sharing in a centralized manner and necessitates distributed infrastructures.

Broadly speaking, there are two distributed settings that have received wide interest: 1) the *master/slave* setting, which assumes the existence of a central parameter server that can perform information aggregation and sharing with all agents; and 2) the *network* setting — also known as the *decentralized* setting — where each agent is only permitted to communicate with its neighbors over a locally connected network (in other words, no centralized coordination is present). Developing fast-convergent algorithms for the latter setting is in general more challenging.

Many algorithms have been developed for the master/slave setting to improve communication efficiency, including deterministic algorithms such as one-shot parameter averaging (Zhang et al., 2012), CoCoA (Smith et al., 2018), DANE (Shamir et al., 2014), CEASE (Fan et al., 2019), and stochastic algorithms like distributed SGD (Recht et al., 2011) and distributed SVRG (Lee et al., 2017; Konečný et al., 2015; Cen et al., 2020). In comparison, the network setting is substantially less explored. Recent work Lian et al. (2017) suggested that the network setting can effectively avoid traffic jams during communication on busy nodes, e.g. the parameter server, and be more efficient in wall-clock time than the master/slave setting. It is therefore natural to ask whether one can adapt more appealing algorithmic ideas to the network setting — particularly for the kind of network topology with a high degree of locality — without compromising the convergence guarantees attainable in the master/slave counterpart.

1.1. Our Contributions

In this paper, we investigate the problem of empirical risk minimization in the network (decentralized) setting, with the aim of achieving communication and computation efficiency simultaneously. The main algorithmic contribution of this paper is the development of communication-efficient network-decentralized (stochastic) optimization algorithms with primal-only formulations, with the assistance of proper gradient tracking. The proposed algorithmic ideas accommodate both approximate Newton-type methods and stochastic variance-reduced methods, and come with theoretical convergence guarantees.

Algorithmic developments. We start by studying an approximate Newton-type method called DANE (Shamir et al., 2014), which is among the most popular communication-efficient algorithms to solve empirical risk minimization. However, DANE was only designed for the master/slave setting in its original form. The current paper develops `Network-DANE`, which generalizes DANE to the network setting. The main challenge in developing such an algorithm is to track and adapt a faithful estimate of the global gradient at each agent, despite the lack of centralized information aggregation. Towards this end, we leverage the powerful idea of *dynamic average consensus* (originally proposed in the control literature Zhu and Martínez (2010) and later adopted in decentralized optimization Qu and Li (2018); Nedić et al. (2017); Di Lorenzo and Scutari (2016)) to track and correct the locally aggregated gradients at each agent — a scheme commonly referred to as *gradient tracking*. We then employ the corrected gradient in local computation, according to the subroutine adapted from DANE. This simple idea allows one to adapt approximate Newton-type methods to network-distributed optimization, without the need of communicating the Hessians.

Our ideas for designing `Network-DANE` can be extended, in a systematic manner, to obtain decentralized versions of other algorithms developed for the master/slave setting, by modifying the local computation step properly. As a notable example, we develop `Network-SVRG`, which performs variance-reduced stochastic optimization locally to enable further computational savings (Johnson and Zhang, 2013). The same approach can be applied to other distribute stochastic variance-reduced methods such as SARAH (Nguyen et al., 2017) to obtain `Network-SARAH`. We also demonstrate that `Network-DANE` can be extended to the proximal setting for nonsmooth composite optimization in a straightforward manner.

Performance analysis. The proposed algorithms achieve an intriguing trade-off between communication and computation efficiency. During every iteration, each agent only communicates the parameter and the gradient estimate to its neighbors, and is therefore communication-efficient globally; moreover, the local subproblems at each agent can be solved efficiently with accelerated or variance-reduced gradient methods, and is thus computation-efficient locally. When the network exhibits a high degree of locality, we show that by allowing multiple rounds of local mixing within each iteration, an improved overall communication complexity can be achieved as it accelerates the rate of convergence. Theoretically, we establish the linear convergence of `Network-DANE` for strongly convex losses, with an improved rate for quadratic losses, both with and without extra averaging. For `Network-SVRG`, we establish its linear convergence for the case of smooth strongly convex losses with extra rounds of averaging. Similar results are obtained for `Network-SARAH` for quadratic losses. Our analysis is highly nontrivial, as it needs to deal with the tight couplings of optimization and network consensus errors through a carefully-designed linear system of Lyapunov functions, especially in the context of approximate Newton-type methods which are known be harder to handle than simple gradient-type methods. Our results shed light on the impacts of data homogeneity and network connectivity upon the rate of convergence; in particular, the proposed algorithms provably obtain fast convergence if the local data are sufficiently similar. Table 1 summarizes the convergence rates of the proposed algorithms.

All in all, our work suggests that: by performing a judiciously chosen amount of local communication and computation per iteration, the overall efficiency can be remarkably improved. Extensive numerical experiments are provided to corroborate our theoretical findings, and to demonstrate the practical efficacy of the proposed algorithms over competitive baselines.

1.2. Related Work

First-order methods, which rely mainly on gradient information, are of core interest to big data analytics, due to their superior scalability. However, it is well-known that distributed

Algorithm	Communication Rounds	Extra Averaging	Loss Functions	β
`Network-DANE`	$O\left(\frac{\kappa(\beta/\sigma+1)\log(1/\varepsilon)}{(1-\alpha_0)^2}\right)$	✗	Quadratic	Arbitrary
	$O\left(\log\kappa\cdot\frac{(\beta^2/\sigma^2+1)\log(1/\varepsilon)}{(1-\alpha_0)^{1/2}}\right)$	✓		
	$O\left(\frac{\kappa^2\log(1/\varepsilon)}{(1-\alpha_0)^2}\right)$	✗	Strongly convex	
	$O\left(\log\kappa\cdot\frac{\kappa(\beta/\sigma+1)\log(1/\varepsilon)}{(1-\alpha_0)^{1/2}}\right)$	✓		
`Network-SVRG`	$O\left(\log\kappa\cdot\frac{\log(1/\varepsilon)}{(1-\alpha_0)^{1/2}}\right)$	✓	Strongly convex	$\beta\le\sigma/200$
`Network-SARAH`	$O\left(\log\kappa\cdot\frac{\log(1/\varepsilon)}{(1-\alpha_0)^{1/2}}\right)$	✓	Quadratic	
EXTRA	$O\left(\kappa^2\log(1/\varepsilon)\right)$	✗	Strongly convex	Arbitrary
DGD	$O\left(\frac{\kappa^2\log(1/\varepsilon)}{(1-\alpha_0)^2}\right)$	✗	Strongly convex	

Table 1: Communication complexity of the proposed algorithms for quadratic and strongly convex losses to reach ε-accuracy. Here, σ, L and $\kappa = L/\sigma$ are the strong convexity, smoothness, and condition number of the local loss functions f_j, $j = 1, \ldots, n$, $\beta \le L$ is the homogeneity parameter gauging the similarities of the local loss functions, and $\alpha_0 := \|\boldsymbol{W} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\|$ is the mixing rate over the network topology. Here, we assume the extra averaging step is implemented via the Chebyshev acceleration scheme (Arioli and Scott, 2014). EXTRA (Shi et al., 2015a) and DGD (Qu and Li, 2018) are listed as baselines.

gradient descent (DGD) suffers from a "speed" versus "accuracy" dilemma when naïvely implemented in a decentralized setting (Nedić et al., 2018). Various fixes (see e.g. the pioneering approaches such as EXTRA (Shi et al., 2015a) and NEXT (Di Lorenzo and Scutari, 2016)) have been proposed to address this issue. Similar gradient tracking ideas Zhu and Martínez (2010) have been incorporated to adjust DGD to ensure its linear convergence using a constant step size Nedić et al. (2017); Qu and Li (2018); Li et al. (2019); Xi et al. (2017); Yuan et al. (2018b); Scutari and Sun (2019); Xin et al. (2019b). The current paper is inspired by the use of gradient tracking in these early results. Our paper implements, and verifies the effectiveness of, gradient tracking for algorithms that involve approximate Newton and variance reduction steps, which are far from straightforward and require significant efforts.

Scaman et al. (2017) proposed a multi-step dual accelerated (MSDA) method for network-distributed optimization, which is optimal within a class of black-box procedures that satisfy the span assumption — the parameter updates fall in the span of the previous estimates and their gradients. Further optimal algorithms are proposed in Uribe et al. (2017) and Scaman et al. (2018) for loss functions that are not necessarily convex or smooth. Their algorithms require knowledge of the dual formulation. In contrast, our algorithms are directly applied to the primal problem, which are more friendly for problems whose dual formulations are hard to obtain. Our algorithms also do not require the span assumption and therefore do not fall into the class of procedures studied in Scaman et al. (2017). The recent work Hannah et al. (2018) suggested that algorithms that break the span assumption such as SVRG can be fundamentally faster than those that do not, and it is of future interest to study if similar conclusions hold in the distributed/decentralized setting.

The `Network-DANE` algorithm is closely related to DANE Shamir et al. (2014), which exhibits appealing performance in both theory and practice. Another recent work further extended DANE with an additional proximal term in the objective function and strengthened its analysis Fan et al. (2019). The proposed `Network-DANE` adapts DANE to the network setting with the aid of gradient tracking. During the preparation of this paper, it was brought to our attention that the SONATA algorithm Sun et al. (2019b), which also applies gradient tracking and subsumes many existing algorithms as special cases with convergence guarantees, can be specialized to obtain the same local sub-problem studied in `Network-DANE`, up to different mixing approaches. The connections between DANE and SVRG observed in Konečnỳ et al. (2015) motivate the development of `Network-SVRG` in this paper, which can be viewed as implementing the local optimization of `Network-DANE` with variance-reduced stochastic gradient methods. The same idea can be easily applied to obtain network-distributed versions of other algorithms such as Katyusha Allen-Zhu (2017), GIANT Wang et al. (2018), AIDE Reddi et al. (2016), among others. Compared with decentralized SGD Lan et al. (2017); Lian et al. (2017), the proposed `Network-SVRG/SARAH` employ variance reduction to achieve much faster convergence.

We note that variance-reduced methods have been adapted to the network setting recently in Mokhtari and Ribeiro (2016); Yuan et al. (2018a); Xin et al. (2019a); Sun et al. (2019a); however, they either have a large memory complexity or impose substantial communication burdens. To be more specific, to decentralize SVRG-type algorithms, these papers Yuan et al. (2018a); Xin et al. (2019a); Sun et al. (2019a) all require communication at every step of the inner loop; in contrast, the proposed `Network-SVRG` algorithm only requires communication at the end of the inner loop, allowing each agent to perform the inner loop efficiently without synchronization, and is therefore more communication-efficient.

Paper organization and notations. Section 2 introduces the formulation of distributed optimization in the decentralized setting, in addition to some preliminary facts. Section 3 presents the proposed `Network-DANE` together with its theoretical guarantees, and briefly discusses its extension to nonsmooth composite optimization. Section 4 introduces `Network-SVRG/SARAH`, which invokes the variance reduction idea to further reduce local computation, together with their theoretical guarantees. We provide numerical experiments in Section 5 and conclude in Section 6. The details of the proofs are deferred to the appendix. Throughout this paper, we use boldface letters to represent vectors and matrices. In addition, $\|\boldsymbol{A}\|$ denotes the spectral norm of a matrix $\boldsymbol{A}$, $\|\boldsymbol{a}\|_2$ represents the ℓ_2 norm of a vector $\boldsymbol{a}$, $\otimes$ stands for the Kronecker product, and $\boldsymbol{I}_n$ denotes the identity matrix of dimension n.

2. Problem Formulation and Preliminaries

2.1. Network-Distributed Optimization

Consider the following empirical risk minimization problem:

$$\operatorname*{minimize}_{\boldsymbol{x} \in \mathbb{R}^d} \quad f(\boldsymbol{x}) \triangleq \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{x}; \boldsymbol{z}_i), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ represents the parameter to optimize, $\ell(\boldsymbol{x}; \boldsymbol{z}_i)$ encodes certain empirical loss of $\boldsymbol{x}$ w.r.t. the ith sample $\boldsymbol{z}_i$ and N denotes the total number of samples we have available. This paper primarily focuses on the case where the function $\ell(\cdot; \boldsymbol{z})$ is both convex and smooth for any given $\boldsymbol{z}$, although we shall also study nonconvex problems in numerical experiments.

In a decentralized optimization framework, data samples are distributed over n agents. For simplicity, we assume throughout that data samples are split into disjoint subsets of equal size. The jth local data set, represented by $\mathcal{M}_j$, thus contains $m \triangleq N/n$ samples. As such, the global loss function can alternatively be represented by

$$f(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} f_j(\boldsymbol{x}), \qquad \text{with } \; f_j(\boldsymbol{x}) \triangleq \frac{1}{m}\sum_{\boldsymbol{z}\in\mathcal{M}_j} \ell(\boldsymbol{x}; \boldsymbol{z}). \tag{2}$$

Here, $f_j(\boldsymbol{x})$ denotes the local loss function at the jth agent ($1 \le j \le n$). In addition, there exists a network — represented by an undirected graph $\mathcal{G}$ of n nodes — that captures the local connectivity across all agents. More specifically, each node in $\mathcal{G}$ represents an agent, and two agents are allowed to exchange information only if there is an edge connecting them in $\mathcal{G}$. Throughout this paper, we denote by $\mathcal{N}_j$ the set of all neighbors of the jth agent over $\mathcal{G}$. The goal is to minimize $f(\cdot)$ in a decentralized manner, subject to the aforementioned network-based communication constraints.

2.2. Preliminaries

Before continuing, we find it helpful to introduce and explain two important concepts.

Mixing. Mathematically, the information mixing between neighboring nodes is often characterized by a mixing or gossiping matrix, denoted by $\boldsymbol{W} = [w_{ij}]_{1\le i,j\le n} \in \mathbb{R}^{n\times n}$. More specifically, $w_{ij} = 0$ if agent i and j are not connected, and $\boldsymbol{W}$ satisfies

$$\boldsymbol{W}^\top \mathbf{1}_n = \mathbf{1}_n \qquad \text{and} \qquad \boldsymbol{W}\mathbf{1}_n = \mathbf{1}_n, \tag{3}$$

where $\mathbf{1}_n \in \mathbb{R}^n$ is the all-one vector. The spectral quantity, which we call the *mixing rate*,

$$\alpha_0 \triangleq \|\boldsymbol{W} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\| \in [0, 1) \tag{4}$$

dictates how fast information mixes over the network. As an example, in a fully-connected network, one can attain $\alpha_0 = 0$ by setting $\boldsymbol{W} = \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$. Nedić et al. (2018) provides comprehensive bounds on $1/(1-\alpha_0)$ for various graphs. For instance, one has $\alpha_0 \asymp 1$ with high probability in an Erdös-Rényi random graph, as long as the graph is connected.

Dynamic average consensus. Assume that each agent generates some *time-varying* quantity $r_j^{(t)}$ (e.g. the current local parameter or gradient estimates). We are interested in tracking the dynamic average

$$\frac{1}{n}\sum_{j=1}^{n} r_j^{(t)} = \frac{1}{n}\mathbf{1}_n^\top \boldsymbol{r}^{(t)}$$

in each of the agents, where $\boldsymbol{r}^{(t)} = [r_1^{(t)}, \cdots, r_n^{(t)}]^\top$. To accomplish this, Zhu and Martínez (2010) proposed a simple tracking algorithm: suppose each agent maintains an estimate $q_j^{(t)}$ in the tth iteration, and the network collectively adopts the following update rule

$$\boldsymbol{q}^{(t)} = \boldsymbol{W}\boldsymbol{q}^{(t-1)} + \boldsymbol{r}^{(t)} - \boldsymbol{r}^{(t-1)}, \tag{5}$$

where $\boldsymbol{q}^{(t)} = [q_1^{(t)}, \cdots, q_n^{(t)}]^\top$. The first term $\boldsymbol{W}\boldsymbol{q}^{(t-1)}$ represents the standard local information mixing operation (meaning that each agent updates its own estimate by a weighted average of its neighbors' estimates), the second term $\boldsymbol{r}^{(t)} - \boldsymbol{r}^{(t-1)}$ tracks the temporal difference. A crucial property of (5) is

$$\mathbf{1}_n^\top \boldsymbol{q}^{(t)} = \mathbf{1}_n^\top \boldsymbol{r}^{(t)}, \tag{6}$$

which indicates that the average of $\{q_i^{(t)}\}_{1\leq i\leq n}$ dynamically tracks the average of $\{r_i^{(t)}\}_{1\leq i\leq n}$. We shall adapt this procedure in our algorithmic development, in the hope of reliably tracking the global gradients (i.e. the average of the local, and often time-varying, gradients at all agents).

3. `Network-DANE`: Algorithm and Convergence

In this section, we propose an algorithm called `Network-DANE` (cf. Alg. 1), which generalizes DANE (Shamir et al., 2014) to the network/decentralized setting. This is accomplished by carefully coordinating the information sharing mechanism and employing dynamic average consensus for gradient tracking.

3.1. The DANE Algorithm

The DANE algorithm is a popular communication-efficient approximate Newton method developed for the master/slave model, initially proposed by Shamir et al. (2014). Here, we review some key features of DANE. (i) Each agent performs an update using both the local loss function $f_j(\cdot)$ and the gradient $\nabla f(\cdot)$ of the global loss function (obtained via the parameter server). (ii) In the tth iteration, the jth agent solves the following problem to update its local estimate $\boldsymbol{x}_j^{(t)}$:

$$\boldsymbol{x}_j^{(t)} = \operatorname*{arg\,min}_{\boldsymbol{x}\in\mathbb{R}^d} \left\{ f_j(\boldsymbol{x}) - \left\langle \nabla f_j\big(\overline{\boldsymbol{x}}^{(t)}\big) - \nabla f\big(\overline{\boldsymbol{x}}^{(t)}\big), \boldsymbol{x} \right\rangle + \frac{\mu}{2}\big\|\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\big\|_2^2 \right\}, \tag{7}$$

where $\mu \geq 0$ is the regularization parameter.[1] Implementing this algorithm requires two rounds of communications per iteration.

[1] In Shamir et al. (2014), the second term in (7) takes the form $\nabla f_j(\overline{\boldsymbol{x}}^{(t)}) - \tilde{\eta}\nabla f(\overline{\boldsymbol{x}}^{(t)})$. We set $\tilde{\eta} = 1$ without loss of generality following the analysis in Fan et al. (2019).

(a) The parameter server first collects all local estimates $\{\boldsymbol{x}_j^{(t-1)}\}_{1\le j\le n}$ and computes the average global parameter estimate $\overline{\boldsymbol{x}}^{(t)} = \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{x}_j^{(t-1)}$; this is then sent back to all agents;

(b) The parameter server collects all local gradients evaluated at the point $\overline{\boldsymbol{x}}^{(t)}$, computes the global gradient $\nabla f(\overline{\boldsymbol{x}}^{(t)}) = \frac{1}{n}\sum_{j=1}^{n} \nabla f_j(\overline{\boldsymbol{x}}^{(t)})$, and shares it with all agents.

The DANE algorithm has been demonstrated as a competitive baseline whose communication efficiency improves, in some sense, with the increase of data size (Shamir et al., 2014); see Fan et al. (2019) for its proximal variation and improved theoratical analysis. To see the reason why DANE is an approximate Newton-type algorithm, consider the case when the local loss functions in all agents are quadratic and takes the form

$$f_j(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}_j \boldsymbol{x} + \boldsymbol{b}_j^\top \boldsymbol{x} + c_j, \tag{8}$$

where each $\boldsymbol{H}_j = \nabla^2 f_j(\boldsymbol{x}) \in \mathbb{R}^{d\times d}$ is a fixed symmetric and positive semidefinite matrix. The local optimization subproblem (7) in DANE can be solved in closed form, with $\boldsymbol{x}_j^{(t)}$ given by[2]

$$\boldsymbol{x}_j^{(t)} = \overline{\boldsymbol{x}}^{(t)} - \big(\underbrace{\boldsymbol{H}_j + \mu \boldsymbol{I}_d}_{\text{local Hessian}}\big)^{-1} \nabla f\big(\overline{\boldsymbol{x}}^{(t)}\big). \tag{9}$$

Clearly, this can be interpreted as

$$\boldsymbol{x}_j^{(t)} = \text{local parameter estimate} - \big(\text{local Hessian}\big)^{-1}\big(\text{global gradient}\big),$$

which is an approximate Newton-type update rule (since we invoke the local Hessian to approximate the true global Hessian). It is worth noting that the algorithm proceeds without actually communicating the local Hessians.

3.2. Algorithm Development

The DANE algorithm was originally developed for the master/slave setting. In the network setting, however, agents can no longer compute (7) locally, due to the absence of centralization enabled by the parameter server; more specifically, agents have access to neither $\overline{\boldsymbol{x}}^{(t)}$ nor $\nabla f(\overline{\boldsymbol{x}}^{(t)})$, both of which are required when solving (7). To address this lack of global information, one might naturally wonder whether we can simply replace global averaging by local averaging; that is, replacing $\overline{\boldsymbol{x}}^{(t)}$ and $\nabla f(\overline{\boldsymbol{x}}^{(t)})$ by $\frac{1}{|\mathcal{N}_j|}\sum_{i\in\mathcal{N}_j} \boldsymbol{x}_i^{(t-1)}$ and $\frac{1}{|\mathcal{N}_j|}\sum_{i\in\mathcal{N}_j} \nabla f_i(\boldsymbol{x}_i^{(t-1)})$, respectively, in the jth agent. However, this simple idea fails to guarantee convergence in local agents. For instance, the local estimation errors may stay flat (but nonvanishing) — as opposed to converging to zero — as the iterations progress, primarily due to imperfect information sharing.

With this convergence issue in mind, our key idea is composed of the following components.

[2] See Shamir et al. (2014) or Appendix A for a short derivation.

Algorithm 1 `Network-DANE`

```
input: initial parameter estimate $\boldsymbol{x}_j^{(0)} \in \mathbb{R}^d$ $(1 \le j \le n)$, regularization parameter $\mu$.
initialization: set $\boldsymbol{y}_j^{(0)} = \boldsymbol{x}_j^{(0)}$, $\boldsymbol{s}_j^{(0)} = \nabla f_j(\boldsymbol{y}_j^{(0)})$ for all agents $1 \le j \le n$.
for $t = 1, 2, \cdots$ do
    for Agents $1 \le j \le n$ in parallel do
        Set $\boldsymbol{y}_j^{(t),0} = \boldsymbol{x}_j^{(t-1)}$ and $\boldsymbol{s}_j^{(t),0} = \boldsymbol{s}_j^{(t-1)}$.
        for $k = 1, 2, \ldots, K$ do
            Receive information $\boldsymbol{y}_i^{(t),k-1}$ and $\boldsymbol{s}_i^{(t),k-1}$ from its neighbors $i \in \mathcal{N}_j$.
            Aggregate parameter estimates from neighbors:
                $$\boldsymbol{y}_j^{(t),k} = \sum\nolimits_{i \in \mathcal{N}_j} w_{ji} \boldsymbol{y}_i^{(t),k-1}, \quad \boldsymbol{s}_j^{(t),k} = \sum\nolimits_{i \in \mathcal{N}_j} w_{ji} \boldsymbol{s}_i^{(t),k-1} \tag{10}$$
        end for
        Set the local parameter estimate to $\boldsymbol{y}_j^{(t)} = \boldsymbol{y}_j^{(t),K}$.
        Update the global gradient estimate by aggregated local information and gradient
        tracking:
                $$\boldsymbol{s}_j^{(t)} = \boldsymbol{s}_j^{(t),K} + \underbrace{\nabla f_j\big(\boldsymbol{y}_j^{(t)}\big) - \nabla f_j\big(\boldsymbol{y}_j^{(t-1)}\big)}_{\text{gradient tracking}}. \tag{11}$$
        Update the parameter estimate by solving:
                $$\boldsymbol{x}_j^{(t)} = \operatorname*{argmin}_{\boldsymbol{z} \in \mathbb{R}^d} \Big\{ f_j(\boldsymbol{z}) - \big\langle \nabla f_j(\boldsymbol{y}_j^{(t)}) - \boldsymbol{s}_j^{(t)}, \boldsymbol{z} \big\rangle + \frac{\mu}{2} \big\| \boldsymbol{z} - \boldsymbol{y}_j^{(t)} \big\|_2^2 \Big\}. \tag{12}$$
    end for
end for
```

- The first ingredient is to maintain an additional estimate of the global gradient in each agent — denoted by $\boldsymbol{s}_j^{(t)}$ in the jth agent. This additional gradient estimate is updated via dynamic average consensus (11), in the hope of tracking the global gradient evaluated at $\boldsymbol{y}_j^{(t)}$ in the jth agent $(1 \le j \le n)$, i.e. $\boldsymbol{s}_j^{(t)}$ attempts to track $\nabla f(\boldsymbol{y}_j^{(t)})$. Here, $\boldsymbol{y}_j^{(t)}$ stands for the parameter estimate obtained by local neighborly averaging in the tth iteration (see Alg. 1 for details). As the algorithm converges, $\{\boldsymbol{y}_j^{(t)}\}_{1 \le j \le n}$ is expected to reach consensus, allowing $\boldsymbol{s}_j^{(t)}$ $(1 \le j \le n)$ to converge to the true global gradient as well.

- In addition, we also allow multiple rounds of mixing within each iteration, i.e. (10), which is helpful in accelerating convergence when the network exhibits a high degree of locality. In essence, by applying K rounds of mixing, we improve the mixing rate from α_0 to
$$\alpha = \alpha_0^K. \tag{13}$$
As we shall see later, choosing a proper (but not too large) K suffices to achieve the desired trade-off between the rate of information sharing and iteration complexity, which helps reduce the overall communication and computation cost. This step of extra

averaging can be implemented in an efficient manner via the Chebyshev acceleration scheme (Arioli and Scott, 2014; Scaman et al., 2017).

Armed with such improved global gradient estimates, we propose to solve a modified local optimization subproblem (12) in `Network-DANE`, which approximates the original Newton-type problem (7) by replacing $\nabla f(\overline{\boldsymbol{x}}^{(t)})$ with the local surrogate $\boldsymbol{s}_j^{(t)}$. The proposed local subproblem (12) is convex and can be solved efficiently via, say, Nesterov's accelerated gradient methods. The whole algorithm is presented in Alg. 1.

Remark 1 *It is certainly possible to employ more general mixing matrices in* (10). *For instance, in mobile computing scenarios with moving agents, one might prefer using time-varying mixing matrices in order to accommodate the topology changes over time. We omit such extensions for brevity.*

3.3. Assumptions and Key Parameters

Before stating theoretical convergence guarantees of `Network-DANE`, we formally introduce a few assumptions, key parameters, and error metrics.

Assumption 1 (strongly convex loss) *The loss function $f_j(\boldsymbol{x})$ at each agent is strongly convex and smooth, namely, $\sigma \boldsymbol{I} \preceq \nabla^2 f_j(\boldsymbol{x}) \preceq L\boldsymbol{I}$ $(1 \le j \le n)$ for some quantities $0 < \sigma \le L$, where $\kappa = L/\sigma$ is the condition number.*

Assumption 2 (quadratic loss) *The loss function $f_j(\boldsymbol{x})$ at each agent is quadratic w.r.t. $\boldsymbol{x}$, i.e. taking the form of* (8).

In the strongly convex setting, let the unique global optimizer of $f(\boldsymbol{x})$ be

$$\boldsymbol{y}^{\mathsf{opt}} := \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}). \tag{14}$$

In the following definition, we further define the homogeneity parameter (Cen et al., 2020; Fan et al., 2019).

Definition 2 (Homogeneity parameter) *Let $f(\cdot)$ and $f_j(\cdot)$ be as defined in* (2). *The homogeneity parameter β is defined as*

$$\beta := \max_{1\le j\le n} \beta_j \qquad \textit{with } \ \beta_j := \sup_{\boldsymbol{x}\in\mathbb{R}^d} \big\|\nabla^2 f_j(\boldsymbol{x}) - \nabla^2 f(\boldsymbol{x})\big\|. \tag{15}$$

As it turns out, β is bounded by the smoothness parameter of $f(\boldsymbol{x})$, i.e. $\beta \le L$.[3] On the other end, as the local loss functions f_j's become similar with each other, β will become smaller. Therefore, β is a key quantity measuring the similarity of data across agents.

[3] To see this, we note from the minimax theorem of eigenvalues and the triangle inequality that

$$\beta \le \max_j \left\{ \sup_{\boldsymbol{x}\in\mathbb{R}^d, \|\boldsymbol{v}\|_2=1} \boldsymbol{v}^\top \Big(\tfrac{n-1}{n}\nabla^2 f_j(\boldsymbol{x})\Big)\boldsymbol{v} - \inf_{\boldsymbol{x}\in\mathbb{R}^d, \|\boldsymbol{v}\|_2=1} \boldsymbol{v}^\top \Big(\tfrac{1}{n}\sum_{i:i\neq j} \nabla^2 f_i(\boldsymbol{x})\Big)\boldsymbol{v} \right\} = \big(1-\tfrac{1}{n}\big)(L-\sigma) \le L. \tag{16}$$

Remark 3 *If the local data follow certain statistical models, it is possible to show that β decreases as the local data size m grows. For example, Shamir et al. (2014) shows that if the data samples at all agents are i.i.d. (with $\ell(\boldsymbol{x};\boldsymbol{z})$ defined in* (2) *satisfying* $0 \preceq \nabla^2 \ell(\boldsymbol{x};\boldsymbol{z}) \preceq L\boldsymbol{I}$ *for all $\boldsymbol{z}$), then with probability at least $1-\delta$ over the samples, we have $\beta < \sqrt{\frac{32L^2}{m}\log\frac{nd}{\delta}}$ – implying β decreases at the rate of $1/\sqrt{m}$.*

Metrics and convergence. We define the following (nd)-dimensional vectors

$$\boldsymbol{x}^{(t)} := \left[\boldsymbol{x}_1^{(t)\top}, \cdots, \boldsymbol{x}_n^{(t)\top}\right]^\top, \quad \boldsymbol{y}^{(t)} := \left[\boldsymbol{y}_1^{(t)\top}, \cdots, \boldsymbol{y}_n^{(t)\top}\right]^\top, \quad \boldsymbol{s}^{(t)} := \left[\boldsymbol{s}_1^{(t)\top}, \cdots, \boldsymbol{s}_n^{(t)\top}\right]^\top. \tag{17}$$

The average of each (nd)-dimensional vector is defined by $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_{j=1}^n \boldsymbol{x}_j \in \mathbb{R}^d$. In addition, we introduce the distributed gradient $\nabla F(\boldsymbol{x}) \in \mathbb{R}^{nd}$ and the global gradient $\nabla f(\boldsymbol{x}) \in \mathbb{R}^{nd}$ of an (nd)-dimensional vector $\boldsymbol{x}$ as follows

$$\nabla F(\boldsymbol{x}) := [\nabla f_1(\boldsymbol{x}_1)^\top, \cdots, \nabla f_n(\boldsymbol{x}_n)^\top]^\top, \quad \nabla f(\boldsymbol{x}) := [\nabla f(\boldsymbol{x}_1)^\top, \cdots, \nabla f(\boldsymbol{x}_n)^\top]^\top. \tag{18}$$

To characterize the convergence behavior of our algorithm, we need to simultaneously track several interrelated error metrics as follows

(1) the convergence error: $\left\|\overline{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{\mathsf{opt}}\right\|_2$;

(2) the parameter consensus error: $\left\|\boldsymbol{y}^{(t)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\right\|_2$;

(3) the gradient estimation error: $\left\|\boldsymbol{s}^{(t)} - \boldsymbol{1}_n \otimes \nabla f(\boldsymbol{y}^{(t)})\right\|_2$.

In this paper, an algorithm is said to converge linearly at a rate $\rho \in (0,1)$ if there exists some constant $C > 0$ such that the following holds for all $t \geq 1$:

$$\max\left\{\sqrt{n}\left\|\overline{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{\mathsf{opt}}\right\|_2, \left\|\boldsymbol{y}^{(t)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\right\|_2, L^{-1}\left\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\right\|_2\right\} \leq C\rho^t.$$

In addition, an algorithm is said to reach ε-accuracy if the left-hand side of the above expression is bounded by ε.

3.4. Theoretical Guarantees of `Network-DANE` for Quadratic Losses

This subsection establishes linear convergence of `Network-DANE` when the objective functions are quadratic. The proofs are postponed to Appendix B.

Theorem 4 (`Network-DANE` under quadratic loss, arbitrary K) *Suppose that Assumptions 1 and 2 hold. Set $\alpha = \alpha_0^K$, and take μ large enough so that $\sigma + \mu \geq \frac{140L}{(1-\alpha)^2}\left(\frac{\beta}{\sigma}+1\right)$. Then* `Network-DANE` *converges linearly at a rate ρ_1 obeying*

$$\rho_1 := \max\left\{\frac{1+\theta_1}{2}, \alpha + \frac{140\kappa}{1-\alpha}\left(\frac{\sigma+\beta}{\sigma+\mu}\right), \frac{1+\alpha}{2} + \frac{2\beta}{\sigma+\mu}\right\}, \tag{19}$$

where θ_1 is defined by

$$\theta_1 := 1 - \frac{\sigma}{\sigma+\mu} + \frac{L}{L+\mu}\frac{\beta^2}{(\sigma+\mu)(\sigma+\mu-\beta)}. \tag{20}$$

Remark 5 *It turns out that $\theta_1 \in (0,1)$ is the convergence rate of DANE in the master/slave setting under quadratic losses (Shamir et al., 2014, Theorem 1).*

It is worth noting that we have spent no effort in optimizing the pre-constants in the above theorem. If the regularization parameter μ is sufficiently large, one can guarantee that $\theta_1 < 1$ and hence DANE converges at a linear rate when optimizing quadratic losses (Shamir et al., 2014). We can clearly see that (19) is always greater than θ_1, which is the price we pay for consensus under the network setting. Fortunately, by properly setting μ, we can still guarantee that $\rho_1 < 1$, which in turn enables linear convergence of `Network-DANE`.

In view of (19), if the network is sufficiently connected (i.e. α is small), or if the data are sufficiently homogeneous (i.e. β is small), we can use a smaller parameter μ, which makes θ_1 (defined in (20)) smaller and results in faster convergence. In summary, `Network-DANE` takes fewer iterations to converge when α and β are both small. After some basic calculations, the complexity of `Network-DANE` for quadratic losses is formalized in the following corollary.

Corollary 6 *Set $\mu + \sigma = \frac{180L}{(1-\alpha)^2}(\frac{\beta}{\sigma}+1)$. Under the assumptions of Theorem 4, one has*

$$\rho_1 \le 1 - \left(\frac{1-\alpha}{20}\right)^2 \frac{1}{\kappa}\frac{1}{(\beta/\sigma+1)}. \tag{21}$$

To reach ε-accuracy, `Network-DANE` takes at most $O\left(\frac{\kappa(\beta/\sigma+1)\log(1/\varepsilon)}{(1-\alpha)^2}\right)$ iterations, and $O\left(K \cdot \frac{\kappa(\beta/\sigma+1)\log(1/\varepsilon)}{(1-\alpha)^2}\right)$ communication rounds.

Recall that if we set the number of local averaging rounds to be $K = 1$, then one has $\alpha = \alpha_0$, and hence our iteration complexity can be readily compared with other existing results. If the homogeneous parameter β obeys $\beta = O(\sigma)$, then the convergence rate can be improved to $O\big(\kappa \log(1/\varepsilon)/(1-\alpha_0)^2\big)$; this is much faster than the corrected DGD (Qu and Li, 2018) with gradient tracking, which converges in $O(\kappa^2 \log(1/\varepsilon)/(1-\alpha_0)^2)$ iterations. The convergence rate of `Network-DANE` degenerates to that of DGD (Qu and Li, 2018) with gradient tracking under the worst condition $\beta = \Theta(L)$. This observation highlights the communication efficiency of `Network-DANE` by harnessing the homogeneity of data across different agents. We emphasize that this is an important feature of our analysis, where the convergence rate adapts with respect to the data homogeneity.

Benefits of extra local averaging (i.e. $K > 1$). The careful reader might have noticed that the rate established above scales poorly with respect to the network parameter, namely, $1 - \alpha_0$, when $K = 1$. One remedy is to consider the case with $K > 1$, where `Network-DANE` performs K rounds of communications per iteration. On the one hand, the effective network parameter $\alpha = \alpha_0^K$ can be made arbitrarily small by taking K sufficiently large, thus leading to faster convergence; on the other hand, the total number of communications is K times larger than the number of iterations, meaning that we might end up with a higher communication complexity. As an example, invoking Corollary 6, we see that: the total communication cost to reach ε-accuracy, in terms of the native network parameter α_0, is given by

$$O\big(K \cdot \kappa(1+\beta/\sigma)\log(1/\varepsilon)/(1-\alpha_0^K)^2\big).$$

Therefore, by judiciously choosing K, it is possible to significantly improve the overall communication complexity, especially when α_0 is close to 1. For example, by setting $K \asymp 1/\log(1/\alpha_0) = O(1/(1-\alpha_0))$, we can ensure $\alpha_0^K \asymp 1/2$ and reduce the communication complexity to $O\big(\kappa \cdot (\beta/\sigma+1)\log(1/\varepsilon)/(1-\alpha_0)\big)$, thus improving the dependence with the graph topology.

The following theorem shows an improved result following a refined analysis, which improves the dependence simultaneously with respect to both κ and $1-\alpha_0$.

Theorem 7 (Network-DANE under quadratic loss, optimized K) *Instate the assumptions of Theorem 4. Set K and μ large enough so that $\alpha = \alpha_0^K \leq 1/(2\kappa)$ and $\sigma + \mu \geq 360\sigma\left(\frac{\beta^2}{\sigma^2}+1\right)$. To reach ε-accuracy, Network-DANE takes at most $O\left((\beta^2/\sigma^2+1)\log(1/\varepsilon)\right)$ iterations, and $O\left(\log\kappa \cdot \frac{(\beta^2/\sigma^2+1)\log(1/\varepsilon)}{1-\alpha_0}\right)$ communications rounds.*

When we set K as suggested in Theorem 7, the iteration complexity becomes independent of the network topology. Moreover, it matches the rate of DANE in the master/slave setting (Shamir et al., 2014) when $\beta = O(\sigma)$, which is $O(\log(1/\varepsilon))$ and further independent of the condition number κ.

In terms of network dependence, the communication complexity improves from $O\big(1/(1-\alpha_0)^2\big)$ to $O\big(1/(1-\alpha_0)\big)$. By implementing the extra averaging step in an efficient manner via the well-known Chebyshev acceleration scheme (Arioli and Scott, 2014; Scaman et al., 2017), the dependence of the communication complexity with respect to $1-\alpha_0$ can be further improved to $O\left((1-\alpha_0)^{-1/2}\right)$. The final communication complexity of Network-DANE for quadratic losses thus becomes

$$O\left(\log\kappa \cdot \frac{(\beta^2/\sigma^2+1)\log(1/\varepsilon)}{(1-\alpha_0)^{1/2}}\right).$$

Therefore, the total amount of communication is significantly reduced using extra averaging, where it scales only logarithmically with respect to κ.

3.5. Theoretical Guarantees of Network-DANE for Strongly Convex Losses

This subsection establishes the linear convergence of Network-DANE for general smooth and strongly convex loss functions, where the rate is worse than that for quadratic losses. The proof can be found in Appendix C.

Theorem 8 *Suppose that Assumption 1 holds. Set $\alpha = \alpha_0^K$, and take μ large enough so that $\sigma+\mu \geq \frac{170\kappa L}{(1-\alpha)^2}$. Then Network-DANE converges linearly at a rate ρ_2 obeying*

$$\rho_2 := \max\left\{\frac{1+\theta_2}{2}, \alpha + \frac{170\kappa}{1-\alpha}\left(\frac{L}{\sigma+\mu}\right), \frac{1+\alpha}{2} + \frac{2\beta}{\sigma+\mu}\right\}, \tag{22}$$

where θ_2 is given by

$$\theta_2 := 1 - \frac{\sigma}{\sigma+\mu} + \frac{\beta}{\sigma+\mu}\sqrt{1-\Big(\frac{\mu}{\sigma+\mu}\Big)^2}. \tag{23}$$

Remark 9 *Note that $\theta_2 \in (0,1)$ is precisely the convergence rate of DANE in the master/slave setting (see (Fan et al., 2019, Theorem 3.1)).*

Similar to Theorem 4, one can guarantee $\theta_2 < 1$ and $\rho_2 < 1$ by setting the regularization parameter μ sufficiently large. Therefore, `Network-DANE` can converge at a linear rate for a general class of smooth and strongly convex problems. Comparing the convergence rates of `Network-DANE` derived for the above two different losses (i.e. comparing (20) with (23)), we see that: when the loss functions are non-quadratic, θ_2 is generally greater than θ_1[4]. This happens since the Hessian matrices associated with the non-quadratic loss functions may vary across different points, which is also the reason why the convergence rate of `Network-DANE` derived for the general case degenerates to the worst-case rate. After some basic calculations, the complexity of `Network-DANE` under strongly convex losses is formalized by the following corollary.

Corollary 10 *Set $\sigma + \mu = \frac{180\kappa L}{(1-\alpha)^2}$. Under the assumptions of Theorem 8, one has*

$$\rho_2 \leq 1 - \left(\frac{1-\alpha}{20}\right)^2 \frac{1}{\kappa^2}. \tag{24}$$

To reach ε-accuracy, `Network-DANE` takes at most $O\left(\frac{\kappa^2 \log(1/\varepsilon)}{(1-\alpha)^2}\right)$ iterations and $O\left(K \cdot \frac{\kappa^2 \log(1/\varepsilon)}{(1-\alpha)^2}\right)$ communication rounds.

When $K = 1$, the communication complexity of `Network-DANE` is $O\left(\frac{\kappa^2 \log(1/\varepsilon)}{(1-\alpha)^2}\right)$, which is rather pessimistic and does not improve with data homogeneity. Similar to Theorem 7, we can improve this by optimizing K properly. We have the following theorem, which is parallel to Theorem 7.

Theorem 11 (`Network-DANE` under strongly convex loss, optimized K) *Instate the assumptions of Theorem 8. Set K and μ large enough so that $\alpha = \alpha_0^K \leq 1/(2\kappa)$ and $\sigma + \mu \geq 360L\left(\frac{\beta}{\sigma} + 1\right)$. To reach ε-accuracy, `Network-DANE` takes at most $O\left(\kappa(\beta/\sigma + 1)\log(1/\varepsilon)\right)$ iterations and $O\left(\log\kappa \cdot \frac{\kappa(\beta/\sigma+1)\log(1/\varepsilon)}{1-\alpha_0}\right)$ communication rounds.*

The improved rate in Theorem 11 improves as the local data become more homogeneous, recovering a feature that has been highlighted previously. Similar to earlier discussions, by using the Chebyshev acceleration scheme (Arioli and Scott, 2014; Scaman et al., 2017), the final communication complexity of `Network-DANE` for strongly convex losses becomes

$$O\left(\log\kappa \cdot \frac{\kappa(\beta/\sigma + 1)\log(1/\varepsilon)}{(1-\alpha_0)^{1/2}}\right).$$

Remark 12 *The homogeneity parameter β defined in Definition 2 measures the largest deviation of local Hessians from the global Hessian. A refined analysis using local deviation β_j is possible by permitting different regularization parameters μ_j in (12) for different agents.*

[4] This is because $\sqrt{\frac{\sigma^2+2\sigma\mu}{(\sigma+\mu)^2}} \geq \frac{\sigma}{\sigma+\mu}$.

3.6. Extension to Nonsmooth Composite Optimization

The proposed algorithms can be extended for nonsmooth composite optimization, by properly adjusting the local optimization step, leveraging proximal variants of DANE (Fan et al., 2019) and SVRG (Xiao and Zhang, 2014). For simplicity, we present the proximal variant of `Network-DANE` and leave its theoretical analysis to future work.

Consider the following regularized empirical risk minimization problem:

$$\underset{\boldsymbol{x}\,\in\,\mathbb{R}^d}{\text{minimize}} \quad f(\boldsymbol{x}) + g(\boldsymbol{x}) \triangleq \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{x};\boldsymbol{z}_i) + g(\boldsymbol{x}), \tag{25}$$

where $f(\cdot)$ and $f_j(\cdot)$ are defined as in (2), and $g(\cdot)$ is a deterministic convex regularizer that can be nonsmooth. This type of problem has wide applications, where it is desirable to promote additional structures or incorporate prior knowledge about the solution through adding a deterministic regularization term $g(\boldsymbol{x})$. We can extend `Network-DANE` to solve (25) by adding the proximal term into the local optimization step, as detailed in Algorithm 2, which is a direct extension of Algorithm 1. Section 5 numerically verifies the effectiveness of Algorithm 2.

Algorithm 2 `Network-DANE` for nonsmooth composite optimization

```
1: Replace the local optimization sub-problem (12) of Network-DANE by the following:
2: Input: $\boldsymbol{y}_j^{(t)}$, $\boldsymbol{s}_j^{(t)}$, regularization parameter $\mu$.
3: Update the parameter estimate by solving:
```

$$\boldsymbol{x}_j^{(t)} = \underset{\boldsymbol{z}\in\mathbb{R}^d}{\text{argmin}} \left\{ f_j(\boldsymbol{z}) + g(\boldsymbol{z}) - \left\langle \nabla f_j(\boldsymbol{y}_j^{(t)}) - \boldsymbol{s}_j^{(t)}, \boldsymbol{z} \right\rangle + \frac{\mu}{2}\left\|\boldsymbol{z} - \boldsymbol{y}_j^{(t)}\right\|_2^2 \right\}. \tag{26}$$

4. Generalizing the Algorithm Design with Variance Reduction

The design of `Network-DANE` suggests a systematic approach to obtain decentralized versions of other algorithms. We illustrate this by reducing local computation of `Network-DANE` using variance reduction. Stochastic variance reduction methods are a popular class of stochastic optimization algorithms, developed to allow for constant step sizes and faster convergence in finite-sum optimization (Johnson and Zhang, 2013; Xiao and Zhang, 2014; Nguyen et al., 2017). It is therefore natural to ask whether such variance reduction techniques can be leveraged in a network setting to further save local computation without compromising communication.

Inspired by the connection between DANE and SVRG (Konečnỳ et al., 2015), we introduce `Network-SVRG/SARAH` in Alg. 3, a decentralized version of SVRG (Johnson and Zhang, 2013) and SARAH (Nguyen et al., 2017) tailored to the network setting, with the assistance of gradient tracking. In particular, the inner loops of SVRG (Johnson and Zhang, 2013) or SARAH (Nguyen et al., 2017) are adopted to replace the local computation subproblem (12) of `Network-DANE`, where the reference to the global gradient is replaced by $\boldsymbol{s}_j^{(t)}$ to calculate the variance-reduced stochastic gradient.

Algorithm 3 `Network-SVRG/SARAH`

```
Replace the local optimization subproblem (12) of Network-DANE by the following:
Input: $\boldsymbol{y}_j^{(t)}$, $\boldsymbol{s}_j^{(t)}$, step size $\delta$, number of local iterations $S$.
Initialization: set $\boldsymbol{u}_j^{(t),0} = \boldsymbol{y}_j^{(t)}$, $\boldsymbol{v}_j^{(t),0} = \boldsymbol{s}_j^{(t)}$.
for $s = 1, ..., S$ do
    $\boldsymbol{u}_j^{(t),s} = \boldsymbol{u}_j^{(t),s-1} - \delta \boldsymbol{v}_j^{(t),s-1}$.
    Sample $\boldsymbol{z}$ from $\mathcal{M}_j$ uniformly at random, then,
        $\boldsymbol{v}_j^{(t),s} = \begin{cases} \nabla \ell(\boldsymbol{u}_j^{(t),s}; \boldsymbol{z}) - \nabla \ell(\boldsymbol{u}_j^{(t),0}; \boldsymbol{z}) + \boldsymbol{v}_j^{(t),0}; & \texttt{(SVRG)} \quad (27a) \\ \nabla \ell(\boldsymbol{u}_j^{(t),s}; \boldsymbol{z}) - \nabla \ell(\boldsymbol{u}_j^{(t),s-1}; \boldsymbol{z}) + \boldsymbol{v}_j^{(t),s-1}. & \texttt{(SARAH)} \quad (27b) \end{cases}$
end for
Choose the new parameter estimate $\boldsymbol{x}_j^{(t)}$ from $\{\boldsymbol{u}_j^{(t),1}, \cdots, \boldsymbol{u}_j^{(t),S}\}$ uniformly at random.
```

The convergence analysis of Alg. 3 is more challenging due to the biased stochastic gradient involved in each local iteration. Encouragingly, the theorem below establishes the linear convergence of `Network-SVRG` for strongly convex losses, and of `Network-SARAH` for quadratic losses, as long as β is sufficiently small and the number of mixing rounds K is sufficiently large. Again, we have not strived to improve the pre-constants specified in the theorem.

Theorem 13 *Assume that the sample loss $\ell(\boldsymbol{x}; \boldsymbol{z})$ is convex and L-smooth w.r.t. $\boldsymbol{x}$ for all $\boldsymbol{z}$. If $\beta/\sigma \leq 1/200$, set K large enough such that $\alpha = \alpha_0^K \asymp 1/\kappa$ and S large enough, `Network-SVRG` converges linearly under Assumption 1; and `Network-SARAH` converges linearly under Assumptions 1 and 2. In particular, to reach ε-accuracy, `Network-SVRG` and `Network-SARAH` take at most $O\left(\log(1/\varepsilon)\right)$ iterations and $O\left(\log\kappa \cdot \frac{\log(1/\varepsilon)}{1-\alpha_0}\right)$ communication rounds under the aforementioned assumptions.*

The proof of Theorem 13 can be found in Appendix D. Theorem 13 implies that: as long as the local data are sufficiently similar (so that β does not exceed the order of σ), by performing $O\left(\log\kappa/(1-\alpha_0)\right)$ rounds of local communication per iteration, `Network-SVRG` and `Network-SARAH` converge in $O\left(\log(1/\varepsilon)\right)$ iterations independent of κ. This performance guarantee matches its counterpart in the master/slave setting (Cen et al., 2020). Altogether, `Network-SVRG/SARAH` achieves appealing computation and communication complexities simultaneously. By further adopting the Chebyshev acceleration scheme (Arioli and Scott, 2014; Scaman et al., 2017), the final communication complexity of `Network-SVRG/SARAH` is at most

$$O\left(\log\kappa \cdot \frac{\log(1/\varepsilon)}{(1-\alpha_0)^{1/2}}\right).$$

It is straightforward to extend this idea to obtain decentralized variants of other stochastic variance reduced algorithms such as Katyusha (Allen-Zhu, 2017), basically by replacing the local computation step (12) by the inner loop update rules of the stochastic methods of interest. For the sake of brevity, this paper does not pursue such “plug-and-play” extensions.

Remark 14 *Our convergence theory of* `Network-SVRG` *requires $\beta \lesssim \sigma$, which is consistent with its counterpart in the master/slave setting (Cen et al., 2020). In contrast,* `Network-DANE` *is guaranteed to converge linearly in the entire range of β by setting μ sufficiently large. One scheme to relax this requirement, as analyzed in Cen et al. (2020), is to add a regularization term, similar to the last term in* (12)*, that penalizes the distance to the previous estimate. However, this might come at a price of slower convergence. We leave this to future investigation.*

5. Numerical Experiments

We evaluate the performance of the proposed algorithms[5] for solving both strongly convex and nonconvex problems, in order to demonstrate the appealing performance in terms of communication-computation trade-offs. Code for our experiments can be found at

`https://github.com/liboyue/Network-Distributed-Algorithm/tree/JMLR`.

Throughout this section, we set the number of agents $n = 20$. We use symmetric fastest distributed linear averaging (FDLA) matrices (Xiao and Boyd, 2004) generated according to the communication graph as the mixing matrix $\boldsymbol{W}$ for aggregating $\boldsymbol{x}_j^{(t)}$ in (10). For aggregating $\boldsymbol{s}_j^{(t)}$ in (10), we use a convex combination of $\boldsymbol{I}$ and $\boldsymbol{W}$ such that its diagonal elements are greater than 0.1, which makes the algorithm more stable in practice. The same regularization parameter μ is used for DANE and `Network-DANE`. We generate connected random communication graphs using an Erdös-Rènyi graph with the probability of connectivity $p = 0.3$ (if not specified). For each experiment, we use the same random starting point $\boldsymbol{x}^{(0)}$ and mixing matrix $\boldsymbol{W}$ for all algorithms. To solve the local optimization subproblems, we use Nesterov's accelerated gradient descent for at most 100 iterations for DANE and `Network-DANE`.

5.1. Experiments On Synthetic Data

We conduct five synthetic numerical experiments based on linear regression to investigate the performance of our algorithms. The same data generation method is used for all synthetic experiments. We generate $m = 1000$ samples of dimension $d = 40$, denoted by $\boldsymbol{A}_i$, randomly from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ i.i.d. for each agent, where $\boldsymbol{\Sigma}$ is a diagonal matrix with $\boldsymbol{\Sigma}_{ii} = i^{-\varrho}$. By changing ϱ, we can change the condition number κ. Data samples are generated according the linear model $\boldsymbol{b}_i = \boldsymbol{A}_i \boldsymbol{x}_0 + \boldsymbol{\xi}_i$, with a random signal $\boldsymbol{x}_0$ and i.i.d. noise $\boldsymbol{\xi}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. For DANE and `Network-DANE`, we set $\mu = 5 \times 10^{-10}$ when $\kappa = 10$ and $\mu = 5 \times 10^{-4}$ when $\kappa = 10^4$. For `Network-SVRG/SARAH`, we set the step size $\delta = 0.1/(L + \sigma + 2\mu)$, the number of local iterations $S = 0.05m$.

Comparison with existing algorithms. To make a fair comparison with other algorithms, no extra local averaging is adopted in this experiment, i.e. the number of mixing rounds is set to $K = 1$. The loss function at each agent is given as $f_i(\boldsymbol{x}) = \frac{1}{2m}\|\boldsymbol{A}_i \boldsymbol{x} - \boldsymbol{b}_i\|_2^2$.

[5] In our experiments of `Network-SVRG/SARAH`, we use the last iterate $\boldsymbol{u}_j^{(t),S}$ as the new parameter estimate locally, which is more practical; our analysis only handles the case where the new parameter estimate is selected uniformly at random from previous iterates, though.

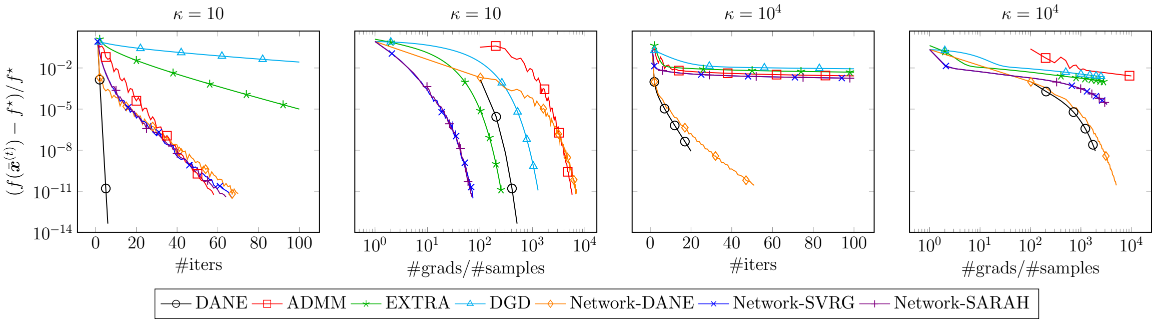

Figure 1: The relative optimality gap with respect to the number of iterations and gradient evaluations under different conditioning $\kappa = 10$ (left two panels) and $\kappa = 10^4$ (right two panels) for linear regression.

We plot the relative optimality gap, given as $(f(\overline{\boldsymbol{x}}^{(t)}) - f^\star)/f^\star$, where $\overline{\boldsymbol{x}}^{(t)}$ is the average parameter of all agents at the tth iteration, and $f^\star$ is the optimal value. We compare the proposed `Network-DANE` (Alg. 1) and `Network-SVRG/SARAH` (Alg. 3) with the master/slave algorithm DANE (Shamir et al., 2014) and ADMM (Boyd et al., 2011),[6] and two popular network-distributed gradient descent algorithms, referred to as DGD (Qu and Li, 2018) and EXTRA (Shi et al., 2015a).

Fig. 1 shows the relative optimality gap with respect to the number of iterations as well as the number of gradient evaluations under different condition numbers $\kappa = 10$ and $\kappa = 10^4$ for linear regression. In both experiments, `Network-DANE` and `Network-SVRG/SARAH` significantly outperform DGD and EXTRA in terms of the numbers of communication rounds. `Network-SVRG/SARAH` has similar communication rounds with ADMM but only communicates locally. `Network-DANE` is quite insensitive to the condition number, performing almost as well as the DANE algorithm in the ill-conditioned case, but operates in a fully decentralized setting. `Network-SVRG/SARAH` further outperforms other algorithms in terms of gradient evaluations in most settings, especially for well-conditioned cases. `Network-SVRG` and `Network-SARAH` are almost indistinguishable.

Benefits of extra local mixing (communication) per iteration. We conduct synthetic experiments to investigate the communication-computation trade-off observed in Corollary 11 when employing multiple rounds of mixing within every iteration. Following the suggestion of the theory, we use a poorly-connected network with mixing rate $\alpha_0 = 0.944$ for communication, which is generated by an Erdös-Rènyi graph with $p = 0.2$. For illustration, we consider the relative optimality gap for a linear regression problem with $\kappa = 10$, with respect to the number of iterations and communication rounds for `Network-DANE` and `Network-SVRG`, under different values of K (no Chebyshev acceleration is employed), shown

[6] We apply ADMM to the constrained optimization problem, which amounts to the centrally-distributed setting, $\min_{\boldsymbol{x}_i} \frac{1}{n} \sum f_i(\boldsymbol{x}_i)$ s.t. $\boldsymbol{x}_i = \boldsymbol{x}$. Note that ADMM can also be applied to the network-distributed setting, which is not shown here since our network algorithms already outperform ADMM in the centrally-distributed setting.

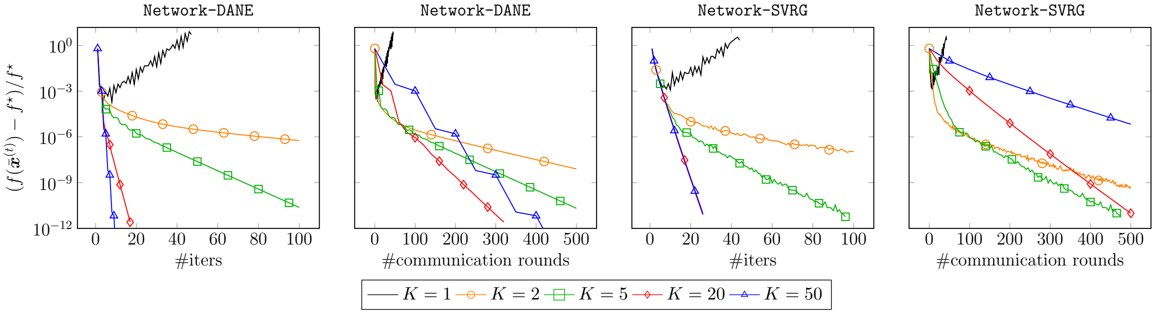

Figure 2: The relative optimality gap with respect to the number of iterations and communication rounds under different rounds of mixing K for `Network-DANE` (left two panels) and `Network-SVRG` (right two panels) over a poorly-connected graph.

in Fig. 2. Due to poor connectivity, `Network-DANE` and `Network-SVRG` fail to converge when using moderate parameters. However, by using a larger K, due to improvement in consensus, both algorithms converge faster in terms of the number of iterations. Notice that after certain threshold, further increasing K will not improve the convergence rate in terms of communication rounds.

Effects of local computation for `Network-SVRG`. We conduct an experiment to analyze the effect of different numbers of local stochastic iterations for `Network-SVRG`. Throughout this experiment, we run our algorithms on a linear regression problem with $\kappa = 10$ and Erdös-Rènyi graph ($p = 0.2$) as the communication graph. Fig. 3 shows the number of communication rounds and the number of gradient evaluations till converge for different numbers of local iterations. It is clear that with too few local iterations, `Network-SVRG` converges very slow and requires more communication. As soon as S is above a threshold, i.e. around $0.05m$ local iterations, the communication rounds no longer decreases. Therefore, in our experiments, we set the number of local iterations as $S = 0.05m$ to ensure satisfactory convergence rate while using an economical amount of local computation.

Effects of network topology. We conduct another experiment to compare the effect of network topology on linear regression problem with $\kappa = 10$. We generate communication graphs with different topology settings. Fig. 4 shows the relative optimality gap with respect to the number of iterations and gradient evaluations for `Network-DANE` and `Network-SVRG/SARAH` for Erdös-Rènyi graph ($p = 0.3$), a 4×5 grid graph, a star graph, and a ring graph. The performance degrades as the network becomes less connected (where $1 - \alpha_0$ becomes small) (Nedić et al., 2018).

Experiments for nonsmooth composite optimization We consider the ℓ_1-norm regularized linear regression, where the loss function of each agent is given as $\tilde{f}_i(\boldsymbol{x}) = f_i(\boldsymbol{x}) + g(\boldsymbol{x}) = \frac{1}{2m}\|\boldsymbol{A}_i\boldsymbol{x} - \boldsymbol{b}_i\|_2^2 + 0.01\|\boldsymbol{x}\|_1$, and the communication graph are generated in the same way as Fig. 1. The condition number κ is also defined in the same way as earlier. We compare the performance of `Network-DANE` with CEASE (Fan et al., 2019), which is the

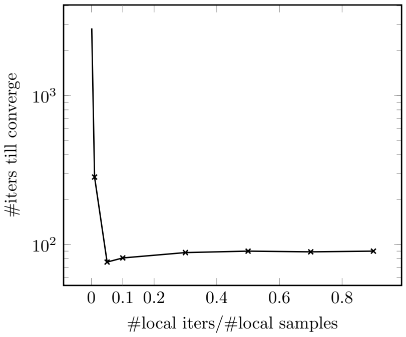

Figure 3: Number of communication rounds and number of gradient evaluations till converge with respect to different numbers of local iterations.

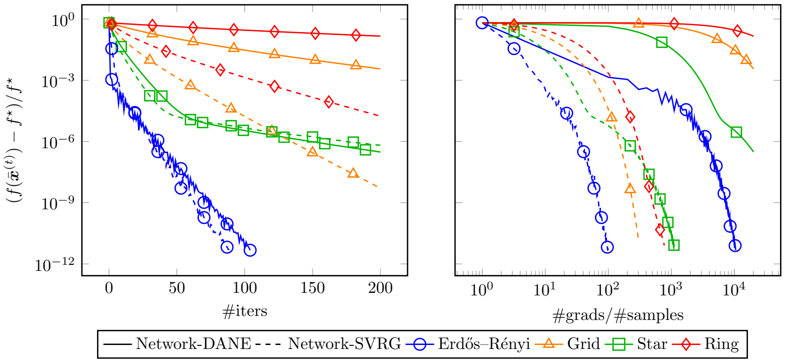

Figure 4: Performance of the proposed algorithms under different network topologies.

proximal version of DANE in the master/slave setting, ADMM, and PG-EXTRA, which is the proximal version of EXTRA (Shi et al., 2015b). For CEASE and `Network-DANE`, we set $\mu = 10^{-4}$ when $\kappa = 10$ and $\mu = 10^{-1}$ when $\kappa = 10^4$, and use FISTA (Beck and Teboulle, 2009) to solve the ℓ_1-norm regularized local problems for computation efficiency. Fig. 5 plots the relative optimality gap $\|\overline{\boldsymbol{x}}^{(t)} - \boldsymbol{x}^{\mathsf{opt}}\|_2/\|\boldsymbol{x}^{\mathsf{opt}}\|_2$ with respect to the number of iterations and the number of gradient evaluations for different algorithms under different condition numbers. In both experiments, `Network-DANE` outperformed ADMM and PG-EXTRA in both metrics, and achieves similar convergence behavior as CEASE, though at a slower rate due to optimizing over a decentralized topology.

5.2. Experiments On Real Data

We perform two experiments on real data to further evaluate the performance of the proposed algorithms for both convex and nonconvex problems.

Binary classification using logistic regression. We use regularized logistic regression to solve a binary classification problem using the Gisette dataset.[7] We split the Gisette

[7] The dataset can be found at https://archive.ics.uci.edu/ml/datasets/Gisette.

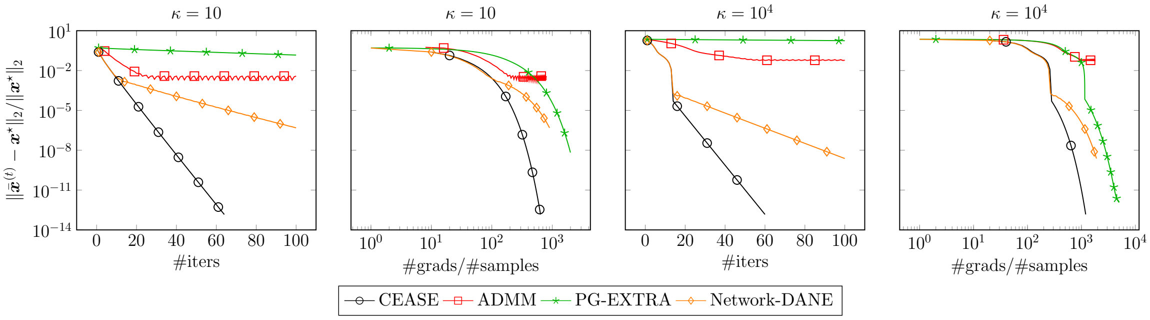

Figure 5: The relative optimality gap with respect to the number of iterations and gradient evaluations under different conditioning $\kappa = 10$ (left two panels) and $\kappa = 10^4$ (right two panels) for linear regression with ℓ_1-norm regularization.

dataset to $n = 20$ agents, where each agent receives $m = 300$ training samples of dimension $d = 5000$. The loss function at each agent is given as

$$f_i(\boldsymbol{x}) = -\frac{1}{m}\sum_{j=1}^{m}\left[b_i^{(j)}\log\left(\frac{1}{1+\exp(\boldsymbol{x}^\top \boldsymbol{a}_i^{(j)})}\right) + (1-b_i^{(j)})\log\left(\frac{\exp(\boldsymbol{x}^\top \boldsymbol{a}_i^{(j)})}{1+\exp(\boldsymbol{x}^\top \boldsymbol{a}_i^{(j)})}\right)\right] + \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2,$$

where $\boldsymbol{a}_i^{(j)} \in \mathbb{R}^d$ and $b_i^{(j)} \in \{0,1\}$ are samples stored at agent i. For DANE and `Network-DANE`, we set $\mu = 5 \times 10^{-9}$ when $\kappa = 2$ and $\mu = 5 \times 10^{-1}$ when $\kappa = 100$. The condition number is controlled by changing the regularization λ. In both cases, our algorithms exhibit compelling performance over other decentralized optimization algorithms especially in terms of communication efficiency.

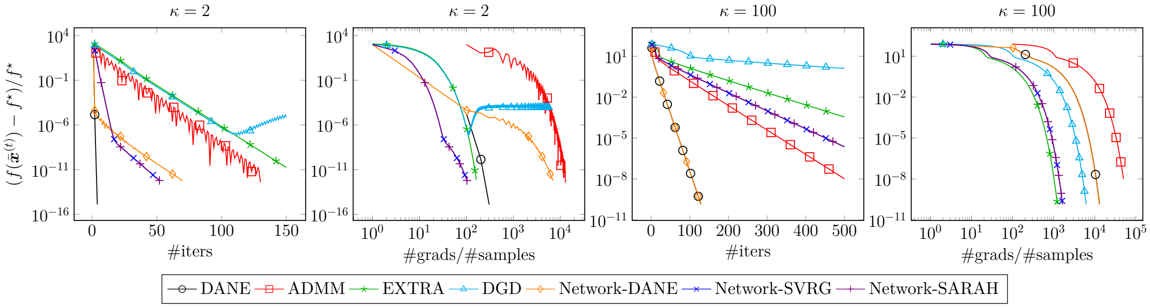

Figure 6: The relative optimality gap with respect to the number of iterations and gradient evaluations under different conditioning $\kappa = 2$ (left two panels) and $\kappa = 100$ (right two panels) for logistic regression using the Gisette dataset.

Neural network training. Though our theory only applies to the strongly convex case, we examine `Network-SVRG/SARAH` in the nonconvex case, by training a one-hidden-layer neural network with 64 hidden neurons and sigmoid activations for a classification task using the MNIST dataset. We split $60,000$ training samples to 20 agents and use an Erdös-Rènyi graph with $p = 0.3$ for communications. Fig. 7 plots the training loss and testing accuracy against the number of iterations and gradient evaluations for different algorithms, where centralized ADMM and decentralized stochastic algorithm (DSGD) are plotted as baselines. Being more communication-efficient than DSGD, and more computation-efficient than ADMM, `Network-SVRG/SARAH` reach a desirable balance between computation and communication efficacies.

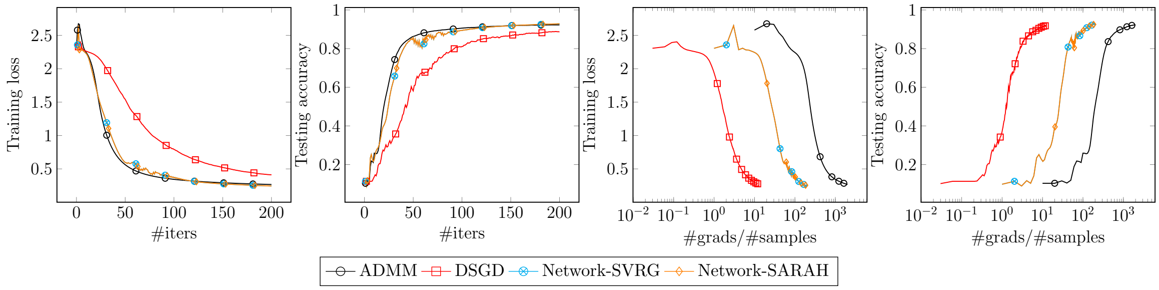

Figure 7: The training loss and testing accuracy with respect to the number of iterations (left two panels) and gradient evaluations (right two panels) for different algorithms on the MNIST dataset.

6. Conclusions

This paper proposes decentralized (stochastic) optimization algorithms that are communication-efficient over a network: (i) `Network-DANE` based on an approximate Newton-type local update, and (ii) `Network-SVRG/SARAH` based on stochastic variance-reduced local gradient updates. Theoretical convergence guarantees are developed for the proposed algorithms, highlighting the impact of network topology, data homogeneity across agents, and refined trade-offs between global communication and local computation. Moreover, extensive numerical experiments are conducted to verify the superior performance of the proposed algorithms. The idea can be easily extended to obtain decentralized versions of other master/slave distributed algorithms in a systematic manner. This work opens up many exciting directions for future investigation, including but not limited to establishing the convergence of `Network-DANE` and `Network-SVRG/SARAH` under general loss functions for both convex and nonconvex settings, with the possibility of asynchronous updates across agents.

Acknowledgments

The work of B. Li, S. Cen and Y. Chi is supported in part by ONR under the grants N00014-18-1-2142 and N00014-19-1-2404, by ARO under the grant W911NF-18-1-0303, and by NSF under the grants CAREER ECCS-1818571, CCF-1806154, CCF-1901199 and CCF-

2007911. Y. Chen is supported in part by the grants AFOSR YIP award FA9550-19-1-0030, ONR N00014-19-1-2120, ARO YIP award W911NF-20-1-0097, ARO W911NF-18-1-0303, NSF CCF-1907661, DMS-2014279 and IIS-1900140, and the Princeton SEAS Innovation Award.

Appendix A. Derivation of Equation (9)

We make the observation that

$$\begin{aligned} f_j(\boldsymbol{x}) - \big\langle \nabla f_j(\overline{\boldsymbol{x}}^{(t)}), \boldsymbol{x} \big\rangle =& \tfrac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}_j \boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{H}_j \overline{\boldsymbol{x}}^{(t)} + \text{constant} \\ =& \tfrac{1}{2}\big(\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\big)^\top \boldsymbol{H}_j \big(\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\big) + \text{constant}, \end{aligned}$$

which allows us to derive a closed-form expression for $\boldsymbol{x}_j^{(t)}$ as follows

$$\begin{aligned} \boldsymbol{x}_j^{(t)} &= \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \left\{ \frac{1}{2}\big(\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\big)^\top \boldsymbol{H}_j \big(\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\big) + \big\langle \nabla f\big(\overline{\boldsymbol{x}}^{(t)}\big), \boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)} \big\rangle + \frac{\mu}{2}\big\|\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\big\|_2^2 \right\} \\ &= \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \left\{ \frac{1}{2}\big(\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\big)^\top \left(\boldsymbol{H}_j + \mu \boldsymbol{I}\right) \big(\boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)}\big) + \big\langle \nabla f\big(\overline{\boldsymbol{x}}^{(t)}\big), \boldsymbol{x} - \overline{\boldsymbol{x}}^{(t)} \big\rangle \right\} \\ &= \overline{\boldsymbol{x}}^{(t)} - \left(\boldsymbol{H}_j + \mu \boldsymbol{I}_d\right)^{-1} \nabla f\big(\overline{\boldsymbol{x}}^{(t)}\big). \end{aligned}$$

Appendix B. Proof of Theorem 4 and Theorem 7

This sections proves the convergence rate of `Network-DANE` for quadratic losses. When local and global loss functions are quadratic, we can solve (12) explicitly. Specifically, Alg. 1 can be alternatively written as Alg. 4 below.

Algorithm 4 `Network-DANE` for quadratic losses (8)

```
for $t = 1, 2, \cdots$ do
    $\boldsymbol{y}^{(t)} = (\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\boldsymbol{x}^{(t-1)},$    (28a)
    $\boldsymbol{s}^{(t)} = (\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\boldsymbol{s}^{(t-1)} + \boldsymbol{H}\big(\boldsymbol{y}^{(t)} - \boldsymbol{y}^{(t-1)}\big),$    (28b)
    $\boldsymbol{x}^{(t)} = \boldsymbol{y}^{(t-1)} - (\boldsymbol{H} + \mu \boldsymbol{I}_{nd})^{-1}\boldsymbol{s}^{(t-1)},$    (28c)
    where $\boldsymbol{y}^{(t)}$ and $\boldsymbol{s}^{(t)}$ are defined in (17), $\boldsymbol{H} := \mathrm{diag}(\boldsymbol{H}_1, \cdots, \boldsymbol{H}_n) \in \mathbb{R}^{nd\times nd}$, and $\boldsymbol{H}_i$ is defined in (8).
end for
```

For notational convenience, we let $\overline{\boldsymbol{H}} = \nabla^2 f(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^n \boldsymbol{H}_j$ be the Hessian of the global loss function. From the definition of the homogeneity parameter β, we have $\|\overline{\boldsymbol{H}} - \boldsymbol{H}_j\|_2 \leq \beta$ for all $j = 1, \ldots, n$. In addition, we recall the notations in (14), (17) and (18), and define the error vector as follows

$$\boldsymbol{e}^{(t)} = \begin{bmatrix} \sqrt{n}\|\overline{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{\mathsf{opt}}\|_2 \\ \|\boldsymbol{y}^{(t)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\|_2 \\ L^{-1}\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2 \end{bmatrix}. \tag{29}$$

Establishing the convergence of `Network-DANE` relies on characterization of the per-iteration dynamics of $\boldsymbol{e}^{(t)}$ for quadratic losses. Towards this end, we state the following key lemma — which is established in Appendix E — that plays a crucial role in the analysis.

Lemma 15 *Let $\eta = \frac{1}{\sigma+\mu}$ and $\gamma = \frac{L}{L+\mu}$. Suppose that Assumptions 1 and 2 hold. Then one has*

$$\boldsymbol{e}^{(t)} \leq \underbrace{\begin{bmatrix} \theta_1 & \gamma\eta\beta + \eta\beta & \eta^2 L\beta \\ \alpha\gamma\eta\beta & \alpha + \alpha\eta L & \alpha\eta L \\ \frac{\beta}{L} + \theta_1\frac{\beta}{L} + \alpha\gamma\eta\beta\frac{\beta}{L} & \alpha\frac{\beta}{L} + \alpha + 1 + \gamma\eta\beta\frac{\beta}{L} + \eta\beta\frac{\beta}{L} + \alpha\frac{\beta}{L} + \alpha\eta\beta & \alpha + \gamma\eta\beta\frac{\beta}{L} + \alpha\eta\beta \end{bmatrix}}_{=:\boldsymbol{G}} \boldsymbol{e}^{(t-1)}. \tag{30}$$

Here, $\boldsymbol{a} \leq \boldsymbol{b}$ indicates that $a_i \leq b_i$ for all entries i.

In what follows, we invoke this result to establish Theorem 4 and Theorem 7 separately.

B.1. Proof of Theorem 4

By the choice of μ stated in Theorem 4, we can show that

$$\gamma < 1 \qquad \text{and} \qquad \eta\beta \leq \eta L < 1. \tag{31}$$

In view of Lemma 15, we can obtain

$$\boldsymbol{e}^{(t)} \leq \boldsymbol{G}_1 \boldsymbol{e}^{(t-1)}$$

with a simplified matrix

$$\boldsymbol{G}_1 := \begin{bmatrix} \theta_1 & 2\eta\beta & \eta^2 L\beta \\ \alpha\gamma\eta\beta & \alpha + \alpha\eta L & \alpha\eta L \\ 3\frac{\beta}{L} & 7 & \alpha + 2\eta\beta \end{bmatrix}, \tag{32}$$

where $\boldsymbol{e}^{(t)}$ is defined in (29). We first invoke an argument from Wai et al. (2018) to show that $\boldsymbol{e}^{(t)}$ converges linearly at a rate not exceeding $\rho(\boldsymbol{G}_1)$. Given that $\boldsymbol{G}_1$ is a positive matrix (i.e. all of its entries are strictly greater than zero), one can invoke the Perron-Frobenius Theorem to show that: there exists a real-valued positive number $\rho(\boldsymbol{G}_1) \in \mathbb{R}$ — the spectral radius of $\boldsymbol{G}_1$ — such that (i) $\rho(\boldsymbol{G}_1)$ is an algebraically simple eigenvalue of $\boldsymbol{G}_1$ associated with a strictly positive eigenvector $\boldsymbol{\chi}$, (ii) all other eigenvalues of $\boldsymbol{G}_1$ are strictly smaller in magnitude than $\rho(\boldsymbol{G}_1)$. Therefore, there exists some constant $C > 0$ such that $\boldsymbol{e}_0 \leq C\boldsymbol{\chi}$, and consequently,

$$\boldsymbol{e}^{(1)} \leq \boldsymbol{G}_1 \boldsymbol{e}^{(0)} \leq C\boldsymbol{G}_1\boldsymbol{\chi} = C\rho(\boldsymbol{G}_1)\boldsymbol{\chi}. \tag{33}$$

Invoking this argument recursively for all t, we arrive at

$$\boldsymbol{e}^{(t)} \leq C\big(\rho(\boldsymbol{G}_1)\big)^t \boldsymbol{\chi}. \tag{34}$$

Therefore, the rest of this proof boils down to upper bounding $\rho(\boldsymbol{G}_1)$. Rearrange the characteristic polynomial of $\boldsymbol{G}_1$, given by

$$\begin{aligned} f_1(\lambda) =& \det\big(\lambda \boldsymbol{I} - \boldsymbol{G}_1\big) \\ =&(\lambda-\theta_1)p_1(\lambda) + \alpha\gamma\eta^2\beta^2(2\alpha+4\eta\beta-2\theta_1-7\eta L) - 3\eta^2\beta^2(\alpha-\alpha\eta L+\theta_1), \end{aligned} \tag{35}$$

where $p_1(\lambda)$ is the following function obtained by direct computation

$$p_1(\lambda) = (\lambda-\alpha-\alpha\eta L)(\lambda-\alpha-2\eta\beta) - 7\alpha\eta L - 2\alpha\gamma\eta^2\beta^2 - 3\eta^2\beta^2. \tag{36}$$

From the Perron-Frobenius Theorem, we know that $\rho(\boldsymbol{G}_1)$ is a simple positive root of $f_1(\lambda)$ (so that $f_1(\rho(\boldsymbol{G}_1))=0$). However, it is difficult to compute it directly. In what follows, we seek to first upper bound $\rho(\boldsymbol{G}_1)$ by

$$\rho_1 := \lambda_0 = \max\left\{\frac{1+\theta_1}{2}, \alpha + \frac{140\eta L}{1-\alpha}\Big(\frac{\beta}{\sigma}+1\Big), \frac{1+\alpha}{2}+2\eta\beta\right\}, \tag{37}$$

and then demonstrate that $\lambda_0 < 1$, which in turn ensures linear convergence.

Step 1: bounding $\rho(\boldsymbol{G}_1)$ by λ_0. The following calculation aims to verify the fact that: for all $\lambda \geq \lambda_0$, one has $f_1(\lambda) > 0$, and hence $\rho(\boldsymbol{G}_1) \leq \lambda_0$. Recall the definition of θ_1 in (20). When $\lambda \geq \lambda_0 \geq \frac{1+\theta_1}{2}$, one has

$$\begin{aligned} \lambda - \theta_1 \geq& \frac{1-\theta_1}{2} \\ =& \frac{1}{2}\frac{\sigma}{\sigma+\mu}\Big(1 - \frac{L}{L+\mu}\frac{\beta}{\sigma+\mu-\beta}\frac{\beta}{\sigma}\Big) \\ \geq& \frac{1}{4}\frac{\sigma}{\sigma+\mu}. \end{aligned} \tag{38}$$

In order for the last inequality to hold, we must make sure that

$$\begin{cases} \sigma+\mu \geq \frac{3\beta^2}{\sigma}, & \text{if } \beta \geq \sigma; \\ \sigma+\mu \geq 3\sigma, & \text{otherwise.} \end{cases} \tag{39}$$

Note that the above relationship is guaranteed by the condition $\sigma+\mu \geq \frac{140L}{(1-\alpha)^2}\Big(\frac{\beta}{\sigma}+1\Big)$. When $\lambda \geq \lambda_0$, using (31), we can lower bound the first term of $p_1(\lambda)$ by

$$\begin{aligned} (\lambda-\alpha-\alpha\eta L)(\lambda-\alpha-2\eta\beta) \geq& \frac{1-\alpha}{2}\Big(\frac{140\eta L}{1-\alpha}\Big(\frac{\beta}{\sigma}+1\Big) - \alpha\eta L\Big) \\ >& 69\eta L\Big(\frac{\beta}{\sigma}+1\Big). \end{aligned}$$

We can lower bound $p_1(\lambda)$ by incorporating (31) as

$$\begin{aligned} p_1(\lambda) &= (\lambda-\alpha-\alpha\eta L)(\lambda-\alpha-2\eta\beta) - 7\alpha\eta L - 2\alpha\gamma\eta^2\beta^2 - 3\eta^2\beta^2 \\ &> 69\eta L\Big(\frac{\beta}{\sigma}+1\Big) - 12\eta L \end{aligned}$$

$$> 68\kappa\eta\beta. \tag{40}$$

As a result of (38) and (40), when $\lambda \geq \lambda_0$, the characteristic polynomial (35) satisfies

$$\begin{aligned} f_1(\lambda) &\geq (\lambda - \theta_1)p_1(\lambda) + \alpha\gamma\eta^2\beta^2(2\alpha + 4\eta\beta - 2\theta_1 - 7\eta L) - 3\eta^2\beta^2(\alpha - \alpha\eta L + \theta_1) \\ &> \frac{1}{4}\eta\sigma \cdot 68\kappa\eta\beta - 9\alpha\gamma\eta^2\beta^2 - 3\eta^2\beta^2(\alpha + \theta_1) \\ &> 17\eta\beta\eta L - 9\alpha\gamma\eta^2\beta^2 - 6\eta^2\beta^2 > 0. \end{aligned}$$

Therefore, any λ that exceeds λ_0 cannot be a root of $f_1(\cdot)$. This implies that the spectral radius $\rho(\boldsymbol{G}_1)$, of necessity, obeys $\rho(\boldsymbol{G}_1) < \lambda_0$.

Step 2: bounding λ_0. This step verifies that all three terms in (37) are smaller than 1, thus leading to the conclusion $\lambda_0 < 1$.

- First, observe that if (39) is satisfied, we have $\frac{1+\theta_1}{2} \leq 1 - \frac{1}{4}\eta\sigma < 1$.
- When $\sigma + \mu \geq \frac{140L}{(1-\alpha)^2}\Big(\frac{\beta}{\sigma} + 1\Big)$, the second term in (37) obeys $\alpha + \frac{140\eta L}{1-\alpha}\Big(\frac{\beta}{\sigma} + 1\Big) \leq 1$.
- Finally, the third term in (37) is also less than 1, since

$$\frac{1+\alpha}{2} + 2\eta\beta \leq \frac{1+\alpha}{2} + \frac{(1-\alpha)^2}{70}\frac{\beta}{\frac{\beta}{\sigma}+1}\frac{1}{L} \leq \frac{1+\alpha}{2} + \frac{(1-\alpha)^2}{70} \leq 1 - \frac{1-\alpha}{2} + \frac{1-\alpha}{70} < 1.$$

B.2. Proof of Theorem 7

By the assumption $\sigma + \mu \geq 360\sigma\Big(\frac{\beta^2}{\sigma^2} + 1\Big)$ and $\alpha \leq \frac{1}{2\kappa}$, we can prove that $\eta\beta < 1$ and $\alpha\eta L \leq \frac{1}{2}$. The characteristic polynomial (35) in Appendix B.1 can then be lower bounded by

$$\begin{aligned} f_1(\lambda) =&\ \det(\lambda \boldsymbol{I} - \boldsymbol{G}_1) \\ =&(\lambda - \theta_1)\Big((\lambda - \alpha - \alpha\eta L)(\lambda - \alpha - 2\eta\beta) - 7\alpha\eta L - 2\alpha\gamma\eta^2\beta^2 - 3\eta^2\beta^2\Big) \\ &+ \alpha\gamma\eta^2\beta^2(2\alpha + 4\eta\beta - 2\theta_1 - 7\eta L) - 3\eta^2\beta^2(\alpha - \alpha\eta L + \theta_1) \\ \geq&\ (\lambda - \theta_1)\Big((\lambda - \alpha - \frac{1}{2}\eta\sigma)(\lambda - \alpha - 2\eta\beta) - \frac{7}{2}\eta\sigma - \eta\sigma\eta^2\beta^2 - 3\eta^2\beta^2\Big) \\ &+ \alpha\gamma\eta^2\beta^2(2\alpha + 4\eta\beta - 2\theta_1 - 7\eta L) - 3\eta^2\beta^2(\alpha - \alpha\eta L + \theta_1), \end{aligned} \tag{41}$$

provided that λ obeys

$$\lambda \geq \max\left\{\frac{1+\theta_1}{2}, \alpha + 180\eta\sigma\Big(\frac{\beta^2}{\sigma^2} + 1\Big), \frac{1+\alpha}{2} + 2\eta\beta\right\}.$$

Given that all conditions in (39) are satisfied, we can show $\eta^2\beta^2 \leq \eta\sigma \cdot \frac{\beta^2}{360\sigma^2(\beta^2/\sigma^2+1)} < \eta\sigma < 1$. One can thus continue to lower bound (41) by

$$f_1(\lambda) > (\lambda - \theta_1)\Big((\lambda - \alpha - \frac{1}{2}\eta\sigma)(\lambda - \alpha - 2\eta\beta) - 8\eta\sigma\Big) - 11\eta^2\beta^2$$

$$
\begin{aligned}
&> \frac{1}{4}\eta\sigma\Big\{\frac{1}{4}\Big[180\eta\sigma\Big(\frac{\beta^2}{\sigma^2}+1\Big)-\frac{1}{2}\eta\sigma\Big]-8\eta\sigma\Big\}-11\eta^2\beta^2\\
&> \frac{1}{4}\eta\sigma\Big\{45\eta\beta\frac{\beta}{\sigma}+44\eta\sigma-8\eta\sigma\Big\}-11\eta^2\beta^2\\
&> \frac{45}{4}\eta\beta-11\eta^2\sigma^2\\
&> 0.
\end{aligned}
$$

Consequently, following similar arguments as in Appendix B.1, we can show that: under the conditions of Theorem 7, the spectral radius of $\boldsymbol{G}_1$ can be upper bounded by

$$
\rho(\boldsymbol{G}_1) \le 1 - \frac{C}{\frac{\beta^2}{\sigma^2}+1},
$$

where C is some sufficiently small positive constant. This immediately tells us that: to reach ε-accuracy, `Network-DANE` takes at most $O\Big(\big(\frac{\beta^2}{\sigma^2}+1\big)\log(1/\varepsilon)\Big)$ iterations. For each iteration, `Network-DANE` needs

$$
K \asymp \frac{\log(1/2\kappa)}{\log\alpha_0} \lesssim \frac{\log\kappa}{1-\alpha_0}
$$

rounds of communication, where we have used the elementary inequality $1-\alpha_0 < \log(1/\alpha_0)$. Putting all this together leads to a communication complexity at most $O\Big(\log\kappa\cdot\frac{(\beta^2/\sigma^2+1)\log(1/\varepsilon)}{1-\alpha_0}\Big)$.

Appendix C. Proofs of Theorem 8 and Theorem 11

This sections establishes the convergence rate of `Network-DANE` for smooth and strongly convex loss functions, following the analysis approach adopted in the proof of Theorem 4. In particular, the following key lemma plays a crucial role, which characterizes the per-iteration dynamics of the proposed `Network-DANE` for general smooth strongly convex losses. The proof of this lemma is deferred to Appendix F.

Lemma 16 *Recall the notations in Lemma 15. Suppose that Assumption 1 holds, and* $\big(\frac{\beta}{\sigma+\mu}\big)^2 \le \frac{\sigma}{\sigma+2\mu}$. *One has*

$$
\boldsymbol{e}^{(t)} \le \underbrace{\begin{bmatrix} \theta_2 & \eta L & \gamma\eta L \\ \alpha\gamma\eta L & \alpha+\alpha\eta L & \alpha\eta L \\ \frac{\beta}{L}+\theta_2\frac{\beta}{L}+\alpha\gamma\eta\beta & \alpha+1+\alpha\frac{\beta}{L}+\eta\beta+\alpha\frac{\beta}{L}+\alpha\eta\beta & \alpha+\gamma\eta\beta+\alpha\eta\beta \end{bmatrix}}_{=:\boldsymbol{G}'} \boldsymbol{e}^{(t-1)}. \tag{42}
$$

Here, $\boldsymbol{e}^{(t)}$ *is the error vector defined in* (29), *and the notation* $\boldsymbol{a} \le \boldsymbol{b}$ *indicates that* $a_i \le b_i$ *for all entries* i.

C.1. Proof of Theorem 8

Under the conditions of Theorem 8, the inequalities stated in (31) remain valid. In addition, when $\sigma+\mu = \frac{170\kappa L}{(1-\alpha)^2}$, we can verify that

$$
\Big(\frac{\beta}{\sigma+\mu}\Big)^2 = \frac{(1-\alpha)^4\beta^2}{170^2\kappa^2L^2} \le \frac{(1-\alpha)^2}{170^2\kappa^2} < \frac{1}{2}\cdot\frac{(1-\alpha)^2}{170\kappa^2} = \frac{1}{2}\cdot\frac{\sigma}{\sigma+\mu} < \frac{\sigma}{\sigma+2\mu}.
$$

When $\sigma+\mu \geq \frac{170\kappa L}{(1-\alpha)^2}$, the LHS decreases faster than the RHS, thus the requirement of Lemma 16 is met. In view of Lemma 16 as well as the fact $\theta_2 \leq 1$, we can replace $\boldsymbol{G}'$ by a simplified matrix that dominates $\boldsymbol{G}'$:

$$\boldsymbol{G}_2 := \begin{bmatrix} \theta_2 & 2\eta L & \gamma\eta L \\ \alpha\gamma\eta L & \alpha+\alpha\eta L & \alpha\eta L \\ 3\frac{\beta}{L} & 7 & \alpha+2\eta\beta \end{bmatrix}. \tag{43}$$

The above matrix $\boldsymbol{G}_2$ is similar to $\boldsymbol{G}_1$ in (32) in the quadratic case, except that the quantity β in the first two rows of $\boldsymbol{G}_1$ is replaced by L (thus leading to a worse convergence rate).

Similar to the proof of Theorem 4, we shall upper bound $\rho(\boldsymbol{G}_2)$ — the spectral radius of $\boldsymbol{G}_2$. To locate the eigenvalues of $\boldsymbol{G}_2$, we rearrange the characteristic polynomial of $\boldsymbol{G}_2$ as follows

$$\begin{aligned} f_2(\lambda) =& \det\left(\lambda\boldsymbol{I}-\boldsymbol{G}_2\right) \\ =&(\lambda-\theta_2)p_2(\lambda)+\alpha\gamma\eta^2L^2\left(2\alpha+4\eta\beta-2\theta_2-7\gamma\right)-3\eta\beta\left(2\alpha\eta L-\gamma(\alpha+\alpha\eta L-\theta_2)\right), \end{aligned} \tag{44}$$

where $p_2(\lambda)$ is the following function obtained by direct computation

$$p_2(\lambda)=(\lambda-\alpha-\alpha\eta L)(\lambda-\alpha-2\eta\beta)-7\alpha\eta L-2\alpha\gamma\eta^2L^2-3\gamma\eta\beta.$$

From the Perron-Frobenius Theorem, $\rho(\boldsymbol{G}_2)$ is a simple positive root of the equation $f_2(\lambda)=0$. However, it is hard to calculate it directly. In what follows, we seek to first upper bound $\rho(\boldsymbol{G}_2)$ by

$$\rho_2 := \lambda_0 = \max\left\{\frac{1+\theta_2}{2},\ \alpha+\frac{170\kappa\eta L}{1-\alpha}, \frac{1+\alpha}{2}+2\eta\beta\right\}, \tag{45}$$

and then demonstrate that $\lambda_0<1$, which in turn ensures linear convergence.

Step 1: bounding $\rho(\boldsymbol{G}_2)$ by λ_0. The following calculation aims to verify the fact that $f_2(\lambda)>0$ holds for all $\lambda\geq\lambda_0$, so that $\rho(\boldsymbol{G}_2)\leq\lambda_0$. Recalling the definition of θ_2 in Lemma 16, we see that when $\lambda\geq\lambda_0\geq\frac{1+\theta_2}{2}$,

$$\begin{aligned} \lambda-\theta_2 &\geq \frac{1-\theta_2}{2} \\ &= \frac{1}{2}\eta\Big(\sigma-\beta\sqrt{(1-\eta\mu)(1+\eta\mu)}\Big) \\ &\geq \frac{1}{2}\eta\Big(\sigma-\beta\sqrt{2(1-\eta\mu)}\Big) > \frac{1}{4}\eta\sigma, \end{aligned} \tag{46}$$

where we have used the fact $\eta\mu<1$ to reach the second inequality. For the last inequality to hold, we need to make sure

$$\begin{cases} \sigma+\mu\geq\frac{10\beta^2}{\sigma}, & \beta\geq\sigma \\ \sigma+\mu\geq 10\sigma, & \text{otherwise} \end{cases} \tag{47}$$

which is guaranteed by the assumption $\sigma+\mu\geq\frac{170\kappa L}{(1-\alpha)^2}$.

Similarly, when $\lambda \geq \lambda_0$, the first term of $p_2(\lambda)$ can be lower bounded by

$$(\lambda - \alpha - \alpha\eta L)(\lambda - \alpha - 2\eta\beta) \geq \frac{1-\alpha}{2}\Big(\frac{170\kappa\eta L}{1-\alpha} - \alpha\eta L\Big) > 80\kappa\eta L.$$

Then, using (31) we can bound $p_2(\lambda)$ by

$$\begin{aligned} p_2(\lambda) &= (\lambda - \alpha - \alpha\eta L)(\lambda - \alpha - 2\eta\beta) - 7\alpha\eta L - 2\alpha\gamma\eta^2 L^2 - 3\gamma\eta\beta \\ &> 80\kappa\eta L - 12\eta L \geq 68\kappa\eta L. \end{aligned} \tag{48}$$

By virtue of (46) and (48), it is seen that when $\lambda \geq \lambda_0$, the characteristic polynomial $f_2(\lambda)$ in (44) satisfies

$$f_2(\lambda) > \frac{1}{4}\eta\sigma \cdot 68\kappa\eta L - 8\alpha\gamma\eta^2 L^2 - 9\eta\beta\eta L > 0.$$

Therefore, any λ that exceeds λ_0 cannot possibly be a root of $f_2(\cdot)$. This implies that the spectral radius necessarily obeys $\rho(\boldsymbol{G}_2) < \lambda_0$.

Step 2: bounding λ_0. This step verifies that the three terms in the expression of λ_0 in (45) is smaller than 1, allowing us to conclude that $\lambda_0 < 1$.

- First, observe that if (47) is satisfied, then we have $\frac{1+\theta_2}{2} \leq 1 - \frac{1}{4}\eta\sigma < 1$.
- When $\sigma + \mu \geq \frac{170\kappa L}{(1-\alpha)^2}$, the second term is $\alpha + \frac{170\kappa\eta L}{1-\alpha} \leq 1$.
- We conclude the proof by checking that the third term is also less than 1, namely,

$$\frac{1+\alpha}{2} + 2\eta\beta \leq \frac{1+\alpha}{2} + \frac{(1-\alpha)^2}{85}\frac{1}{\kappa}\frac{\beta}{L} \leq \frac{1+\alpha}{2} + \frac{(1-\alpha)^2}{85} \leq 1 - \frac{1-\alpha}{2} + \frac{1-\alpha}{85}.$$

C.2. Proof of Theorem 11

We first verify the assumption of Lemma 16. When $\sigma + \mu = 360L\left(\frac{\beta}{\sigma} + 1\right)$,

$$\Big(\frac{\beta}{\sigma+\mu}\Big)^2 = \frac{\beta^2}{360^2 L^2 (\frac{\beta}{\sigma}+1)^2} \leq \frac{\beta}{360^2 \kappa L (\frac{\beta}{\sigma}+1)} < \frac{1}{2} \cdot \frac{1}{360\kappa(\frac{\beta}{\sigma}+1)} = \frac{1}{2} \cdot \frac{\sigma}{\sigma+\mu} < \frac{\sigma}{\sigma+2\mu}.$$

Therefore, Lemma 16 still holds.

By the assumption $\alpha \leq \frac{1}{2\kappa}$, we can further lower bound the characteristic polynomial (44) in Appendix C.1 as follows:

$$\begin{aligned} f_2(\lambda) =&\ \det\big(\lambda \boldsymbol{I} - \boldsymbol{G}_2\big) \\ =&\ (\lambda - \theta_2)\big((\lambda - \alpha - \alpha\eta L)(\lambda - \alpha - 2\eta\beta) - 7\alpha\eta L - 2\alpha\gamma\eta^2 L^2 - 3\gamma\eta\beta\big) \\ &+ \alpha\gamma\eta^2 L^2 \left(2\alpha + 4\eta\beta - 2\theta_2 - 7\gamma\right) - 3\eta\beta\left(2\alpha\eta L - \gamma(\alpha + \alpha\eta L - \theta_2)\right) \\ \geq&\ (\lambda - \theta_2)\Big((\lambda - \alpha - \frac{1}{2}\eta\sigma)(\lambda - \alpha - 2\eta\beta) - \frac{7}{2}\eta\sigma - \eta\sigma\eta^2 L^2 - 3\gamma\eta\beta\Big) \\ &- \eta\sigma\eta^2 L^2 (\theta_2 + \frac{7}{2}\gamma) - 3\eta\beta\Big(\eta\sigma + \gamma\theta_2\Big) \end{aligned}$$

$$> (\lambda - \theta_2)\Big((\lambda - \alpha - \frac{1}{2}\eta\sigma)(\lambda - \alpha - 2\eta\beta) - 8\eta\sigma\Big) - 5\eta\sigma\eta^2 L^2 - 6\eta\beta\eta L, \tag{49}$$

providing λ obeys

$$\lambda \geq \max\left\{\frac{1+\theta_2}{2}, \alpha + 180\eta L\Big(\frac{\beta}{\sigma}+1\Big), \frac{1+\alpha}{2} + 2\eta\beta\right\}.$$

We can further lower bound (49) by

$$f_2(\lambda) \geq \frac{1}{4}\eta\sigma\left\{\frac{1}{4}\Big[180\eta L\Big(\frac{\beta}{\sigma}+1\Big) - \frac{1}{2}\eta\sigma\Big] - 8\eta\sigma\right\} - 5\eta\sigma\eta^2 L^2 - 6\eta\beta\eta L > 0,$$

as long as μ satisfies $\sigma + \mu \geq 360L\big(\frac{\beta}{\sigma}+1\big)$. Therefore, following similar arguments as adopted in Appendix C.1, the spectral radius of $\boldsymbol{G}_2$ can be upper bounded by

$$\rho(\boldsymbol{G}_2) \leq 1 - \frac{C}{\kappa(\frac{\beta}{\sigma}+1)},$$

where C is a small positive constant. Consequently, to reach ε-accuracy, `Network-DANE` takes at most $O\left(\kappa\big(\frac{\beta}{\sigma}+1\big)\log(1/\varepsilon)\right)$ iterations and $O\left(\log\kappa \cdot \frac{\kappa(\beta/\sigma+1)\log(1/\varepsilon)}{1-\alpha_0}\right)$ communication rounds.

Appendix D. Proof of Theorem 13

The proof strategy of Theorem 13 is similar in spirit to the convergence proof of `Network-DANE`, where we will carefully build a linear system that tracks the coupling of the consensus error and the optimization error. Under the assumptions in Theorem 13, we can assume that $1 - 3\alpha\kappa - 3\beta/\sigma > 0$. Let

$$\zeta = 1/(1 - 3\alpha\kappa - 3\beta/\sigma).$$

In what follows, we first introduce two key lemmas that connect the convergence behavior of `Network-SVRG` and `Network-SARAH` in the network setting to their master/slave counterparts (namely, D-SVRG and D-SARAH) studied in Cen et al. (2020). Lemma 17, proved in Appendix G, creates the linear system characterizing the iteration dynamics of `Network-SVRG`. Similarly, Lemma 18 describes the dynamics of `Network-SARAH`, whose proof can be found in Appendix H.

Lemma 17 *Under the assumptions in Theorem 13, `Network-SVRG` satisfies*

$$\mathbb{E}[\boldsymbol{e}^{(t)}] \leq \underbrace{\begin{bmatrix} \Big(\nu(1+3\alpha\kappa+4\frac{\beta}{\sigma}) + \frac{\beta}{\sigma}\Big)\zeta & 8\frac{\beta}{\sigma}\zeta & \alpha\zeta/\kappa & \zeta/16 \\ 1/2 & 0 & 0 & 0 \\ 8\big(\frac{\beta}{\sigma}\big)^2 & 64\big(\frac{\beta}{\sigma}\big)^2 & 4\alpha^2 & \alpha\kappa/2 \\ 64\alpha\kappa & 0 & 0 & 0 \end{bmatrix}}_{:=\boldsymbol{G}_3} \mathbb{E}[\boldsymbol{e}^{(t-1)}], \tag{50}$$

where the error vector is defined as

$$\boldsymbol{e}^{(t)} = \begin{bmatrix} \sum_{j=1}^{n} \big(f(\boldsymbol{x}_j^{(t)}) - f(\boldsymbol{y}^{\mathsf{opt}})\big) \\ \sum_{j=1}^{n} \big(f(\boldsymbol{y}_j^{(t)}) - f(\boldsymbol{y}^{\mathsf{opt}})\big)/2 \\ \|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2^2/\sigma \\ 32L\|\boldsymbol{y}^{(t)} - \mathbf{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\|_2^2/\alpha \end{bmatrix}.$$

Here, $\nu \leq \frac{1}{2}\frac{\sigma-2\beta}{\sigma-3\beta}$ is the convergence rate of D-SVRG in the master/slave setting under the same assumptions (Cen et al., 2020, Theorem 1).

Lemma 18 *Under the assumptions of Theorem 13, and the loss functions are quadratic,* `Network-SARAH` *satisfies*

$$\mathbb{E}[\boldsymbol{e}^{(t)}] \leq \underbrace{\begin{bmatrix} \Big(\nu(1+3\alpha\kappa+4\frac{\beta}{\sigma}) + \frac{\beta}{\sigma}\Big)\zeta & 8\frac{\beta}{\sigma}\zeta & 2\alpha\zeta/\kappa & \zeta/8 \\ 1/2 & 0 & 0 & 0 \\ 4\big(\frac{\beta}{\sigma}\big)^2 & 32\big(\frac{\beta}{\sigma}\big)^2 & 4\alpha^2 & \alpha\kappa/2 \\ 32\alpha\kappa & 0 & 0 & 0 \end{bmatrix}}_{:=\boldsymbol{G}_4} \mathbb{E}[\boldsymbol{e}^{(t-1)}], \tag{51}$$

where the error vector is defined as

$$\boldsymbol{e}^{(t)} = \begin{bmatrix} \|\nabla f(\boldsymbol{x}^{(t)})\|_2^2 \\ \|\nabla f(\boldsymbol{y}^{(t)})\|_2^2/2 \\ \|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2^2 \\ 32L^2\|\boldsymbol{y}^{(t)} - \mathbf{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\|_2^2/(\alpha\kappa) \end{bmatrix}.$$

Here, $\nu \leq \frac{1}{2}\frac{1}{1-4\beta^2/\sigma^2}$ is the convergence rate of D-SARAH in the master/slave setting under the same assumptions (Cen et al., 2020, Theorem 2).

Since every term in the matrices of linear systems of Lemma 17 and Lemma 18 is non-negative, all eigenvalues of $\boldsymbol{G}_3$ and $\boldsymbol{G}_4$ are bounded by the maximum of the sum of rows according to the Gershgorin circle theorem. For `Network-SVRG`, by setting $\alpha = \frac{1}{70\kappa}$, which needs $K \asymp O(\log_{\alpha_0} 1/\kappa) = O\big(\log\kappa/(1-\alpha_0)\big)$, we can ensure that the sum of the first row is bounded by $5/6$, and the sums of other rows are also bounded by a constant smaller than 1, under the assumption $\beta \leq \sigma/200$. Therefore, invoking the Gershgorin circle theorem, the spectral radius is bounded by a constant smaller than 1. To achieve ε-accuracy, the total number of iterations needed is $O\left(\log(1/\varepsilon)\right)$ and thus the communication complexity is $O\left(\log\kappa \cdot \frac{\log(1/\varepsilon)}{1-\alpha_0}\right)$. Similar arguments hold true for `Network-SARAH`, which we omit for simplicity.

Appendix E. Proof of Lemma 15

The proof is divided into several steps. (i) In Appendix E.1, we bound the convergence error $\sqrt{n}\|\overline{\boldsymbol{x}}^{(t)} - \boldsymbol{y}^{\mathsf{opt}}\|_2$; (ii) in Appendix E.2, we bound the parameter consensus error $\|\boldsymbol{x}^{(t)} - \mathbf{1}_n \otimes \overline{\boldsymbol{x}}^{(t)}\|_2$; (iii) in Appendix E.3, we bound the gradient estimation error $\|\boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2$; (iv) finally, we create induction inequalities of $\|\boldsymbol{y}^{(t)} - \mathbf{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\|_2$, $\sqrt{n}\|\overline{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{\mathsf{opt}}\|_2$ and $\|\boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2$ in Appendix E.4 to conclude the proof.

E.1. Convergence error

We begin by defining an auxiliary variable $\boldsymbol{x}_j^+$, which can be seen as the result of one local iterate (12) of the original DANE algorithm initialized at $\overline{\boldsymbol{y}}^{(t-1)}$:

$$\boldsymbol{x}_j^+ = \operatorname*{argmin}_{\boldsymbol{x}} \left\{ f_j(\boldsymbol{x}) - \left\langle \nabla f_j(\overline{\boldsymbol{y}}^{(t-1)}) - \nabla f(\overline{\boldsymbol{y}}^{(t-1)}), \boldsymbol{x} \right\rangle + \frac{\mu}{2}\|\boldsymbol{x} - \overline{\boldsymbol{y}}^{(t-1)}\|_2^2 \right\}. \tag{52}$$

Following the same convention as in previous definitions, we also define

$$\overline{\boldsymbol{x}}^+ = \frac{1}{n}\sum_j \boldsymbol{x}_j^+. \tag{53}$$

Given that the function we optimize at each agent is strongly convex, the local optimality conditions of (52) and (12) are as follows:

$$\nabla f_j(\boldsymbol{x}_j^+) + \mu(\boldsymbol{x}_j^+ - \boldsymbol{y}^{\mathsf{opt}}) = \nabla(f_j - f)(\overline{\boldsymbol{y}}^{(t-1)}) + \mu(\overline{\boldsymbol{y}}^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}), \tag{54a}$$
$$\nabla f_j(\boldsymbol{x}_j^{(t-1)}) + \mu(\boldsymbol{x}_j^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}) = \nabla f_j(\boldsymbol{y}_j^{(t-1)}) - \boldsymbol{s}_j^{(t-1)} + \mu(\boldsymbol{y}_j^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}). \tag{54b}$$

Taking the average of (54) over $j = 1, \ldots, n$, we obtain another set of optimality conditions:

$$\frac{1}{n}\sum_j \nabla f_j(\boldsymbol{x}_j^+) + \mu(\overline{\boldsymbol{x}}^+ - \boldsymbol{y}^{\mathsf{opt}}) = \mu(\overline{\boldsymbol{y}}^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}), \tag{55a}$$
$$\frac{1}{n}\sum_j \nabla f_j(\boldsymbol{x}_j^{(t-1)}) + \mu(\overline{\boldsymbol{x}}^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}) = \mu(\overline{\boldsymbol{y}}^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}), \tag{55b}$$

where we use the fact $\sum_j \boldsymbol{s}_j^{(t-1)} = \sum_j \nabla f_j(\boldsymbol{y}_j^{(t-1)})$ due to the property of gradient tracking (6).

In view of the triangle inequality, the convergence error can be decomposed as

$$\|\overline{\boldsymbol{x}}^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}\|_2 \le \|\overline{\boldsymbol{x}}^{(t-1)} - \overline{\boldsymbol{x}}^+\|_2 + \|\overline{\boldsymbol{x}}^+ - \boldsymbol{y}^{\mathsf{opt}}\|_2, \tag{56}$$

where the first term is the error caused by inaccurate gradient estimate, and the second term is the progress of DANE initialized at $\overline{\boldsymbol{y}}^{(t-1)}$.

1. For the first term $\|\overline{\boldsymbol{x}}^{(t-1)} - \overline{\boldsymbol{x}}^+\|_2$, we first plug in the Hessian of the quadratic losses to solve for $\boldsymbol{x}_j^{(t-1)}$ and $\boldsymbol{x}_j^+$ explicitly as

$$\boldsymbol{x}_j^{(t-1)} = \boldsymbol{y}_j^{(t-1)} - (\boldsymbol{H}_j + \mu \boldsymbol{I}_d)^{-1}\boldsymbol{s}_j^{(t-1)}, \tag{57a}$$
$$\boldsymbol{x}_j^+ = \overline{\boldsymbol{y}}^{(t-1)} - (\boldsymbol{H}_j + \mu \boldsymbol{I}_d)^{-1}\nabla f(\overline{\boldsymbol{y}}^{(t-1)}). \tag{57b}$$

The first error term $\|\overline{\boldsymbol{x}}^{(t-1)} - \overline{\boldsymbol{x}}^+\|_2$ can be written as

$$\begin{aligned} &\|\overline{\boldsymbol{x}}^{(t-1)} - \overline{\boldsymbol{x}}^+\|_2 \\ =&\left\|\Big(\frac{1}{n}\boldsymbol{1}_n^\top \otimes \boldsymbol{I}_d\Big)(\boldsymbol{x}^{(t-1)} - \boldsymbol{x}^+)\right\|_2 \end{aligned}$$

$$
\begin{aligned}
=&\Big\|\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\Big(\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}-(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})+(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\boldsymbol{s}^{(t-1)}\Big)\Big\|_2\\
=&\Big\|\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\big(\boldsymbol{s}^{(t-1)}-\nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\big)\Big\|_2,
\end{aligned}
$$

where the last line follows from the definition of $\overline{\boldsymbol{y}}^{(t-1)}$. Then, we add and subtract $(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}$ and rearrange terms, obtaining

$$
\begin{aligned}
&\|\overline{\boldsymbol{x}}^{(t-1)}-\overline{\boldsymbol{x}}^{+}\|_2\\
=&\Big\|\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\Big((\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}-(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}\Big)\big(\boldsymbol{s}^{(t-1)}-\nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\big)\\
&+\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}\big(\boldsymbol{s}^{(t-1)}-\nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\big)\Big\|_2\\
=&\Big\|\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}-\boldsymbol{H})(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}\big(\boldsymbol{s}^{(t-1)}-\nabla f(\boldsymbol{y}^{(t-1)})\big)\\
&+\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}-\boldsymbol{H})(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}\big(\nabla f(\boldsymbol{y}^{(t-1)})-\nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\big)\\
&+\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}(\boldsymbol{H}-\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}})(\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\Big\|_2 \qquad (58)\\
\le&\Big\|\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\Big\|_2\Big\|(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}-\boldsymbol{H})(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}\Big\|_2\|\boldsymbol{s}^{(t-1)}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2\\
&+\Big\|\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\Big\|_2\Big\|(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}-\boldsymbol{H})\Big(\boldsymbol{I}_{nd}+\mu\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}^{-1}\Big)^{-1}\Big\|_2\|\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2\\
&+\Big\|\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\Big\|_2\Big\|(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}(\boldsymbol{H}-\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}})\Big\|_2\|\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2.
\end{aligned}
$$

The last term in (58) follows from the identity

$$
\begin{aligned}
&\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}\big(\boldsymbol{s}^{(t-1)}-\nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\big)\\
=&(\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_d)^{-1}\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\big(\boldsymbol{s}^{(t-1)}-\nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\big)\\
=&(\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_d)^{-1}\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\big(\boldsymbol{H}\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{H}}\overline{\boldsymbol{y}}^{(t-1)}\big)\\
=&(\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_d)^{-1}\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\big(\boldsymbol{H}\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\boldsymbol{H}\overline{\boldsymbol{y}}^{(t-1)}\big)\\
=&\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}\boldsymbol{H}(\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\\
=&\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}(\boldsymbol{H}-\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}})(\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}).
\end{aligned}
$$

Taken together with the identity $\|\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\|_2=\frac{1}{\sqrt{n}}$, the assumption $\|\boldsymbol{H}_j-\overline{\boldsymbol{H}}\|_2\le\beta$, and the bound $\|(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\|_2\le\frac{1}{\sigma+\mu}$ and $\Big\|\Big(\boldsymbol{I}_{nd}+\mu\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}^{-1}\Big)^{-1}\Big\|_2\le\frac{L}{L+\mu}$, we can further bound (58) by

$$
\begin{aligned}
\sqrt{n}\|\overline{\boldsymbol{x}}^{(t-1)}-\overline{\boldsymbol{x}}^{+}\|_2\le&\frac{1}{\sigma+\mu}\frac{\beta}{\sigma+\mu}\|\boldsymbol{s}^{(t-1)}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2\\
&+\Big(\frac{L}{L+\mu}\frac{\beta}{\sigma+\mu}+\frac{\beta}{\sigma+\mu}\Big)\|\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2. \qquad (59)
\end{aligned}
$$

2. Regarding the second term $\|\overline{\boldsymbol{x}}^{+}-\boldsymbol{y}^{\mathsf{opt}}\|_2$, we provide a slightly improved bound compared to Shamir et al. (2014). In view of (57b),

$$\begin{aligned}
\|\overline{\boldsymbol{x}}^{+}-\boldsymbol{y}^{\mathsf{opt}}\|_2 =& \Big\|\overline{\boldsymbol{y}}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}-\frac{1}{n}\sum_{j}(\boldsymbol{H}_j+\mu\boldsymbol{I}_d)^{-1}\nabla f(\overline{\boldsymbol{y}}^{(t-1)})\Big\|_2\\
=& \Big\|\Big(\boldsymbol{I}-\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{H}_i+\mu\boldsymbol{I})^{-1}\overline{\boldsymbol{H}}\Big)(\overline{\boldsymbol{y}}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}})\Big\|_2\\
\leq& \Big\|\boldsymbol{I}-\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{H}_i+\mu\boldsymbol{I})^{-1}\overline{\boldsymbol{H}}\Big\|_2\|\overline{\boldsymbol{y}}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}\|_2. \qquad (60)
\end{aligned}$$

Then, we use the triangle inequality to break the convergence rate in (60) into two parts:

$$\begin{aligned}
&\Big\|\boldsymbol{I}-\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{H}_i+\mu\boldsymbol{I})^{-1}\overline{\boldsymbol{H}}\Big\|_2\\
\leq& \Big\|\boldsymbol{I}-(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\overline{\boldsymbol{H}}\Big\|_2+\Big\|\frac{1}{n}\sum_{i=1}^{n}\Big((\boldsymbol{H}_i+\mu\boldsymbol{I})^{-1}-(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\Big)\overline{\boldsymbol{H}}\Big\|_2. \qquad (61)
\end{aligned}$$

When $\overline{\boldsymbol{H}}\succeq\sigma\boldsymbol{I}_d$, it is straightforward to check that the first term of (61) is upper bounded by

$$\Big\|\boldsymbol{I}-(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\overline{\boldsymbol{H}}\Big\|_2\leq 1-\frac{\sigma}{\sigma+\mu}.$$

Regarding the second term of (61), let $\boldsymbol{\Delta}_i:=\boldsymbol{H}_i-\overline{\boldsymbol{H}}$ and use the definition of β, one derives

$$\big\|(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\boldsymbol{\Delta}_i\big\|_2\leq\big\|(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\big\|_2\cdot\big\|\boldsymbol{\Delta}_i\big\|_2\leq\frac{\beta}{\sigma+\mu}<1 \qquad (62)$$

under our hypothesis $\beta<\mu+\sigma$. In addition,

$$\begin{aligned}
&\Big\|\frac{1}{n}\sum_{i=1}^{n}\Big((\boldsymbol{H}_i+\mu\boldsymbol{I})^{-1}-(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\Big)\overline{\boldsymbol{H}}\Big\|_2\\
=&\Big\|\frac{1}{n}\sum_{i=1}^{n}\Big(\sum_{m=0}^{\infty}(-1)^m[(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\boldsymbol{\Delta}_i]^m(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}-(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\Big)\overline{\boldsymbol{H}}\Big\|_2 \qquad (63)\\
=&\Big\|\frac{1}{n}\sum_{i=1}^{n}\Big(\sum_{m=2}^{\infty}(-1)^m[(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\boldsymbol{\Delta}_i]^m(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\Big)\overline{\boldsymbol{H}}\Big\|_2 \qquad (64)\\
\leq&\frac{1}{n}\sum_{i=1}^{n}\sum_{m=2}^{\infty}\|(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\|_2^m\cdot\|\boldsymbol{\Delta}_i\|_2^m\cdot\big\|(\boldsymbol{I}+\mu\overline{\boldsymbol{H}}^{-1})^{-1}\big\|_2\\
\leq&\sum_{m=2}^{\infty}(\sigma+\mu)^{-m}\beta^m\frac{L}{L+\mu}=\frac{L}{L+\mu}\frac{\beta^2}{(\sigma+\mu)(\sigma+\mu-\beta)}.
\end{aligned}$$

Here, the line (63) is an expansion based on the Neumann series (whose convergence is guaranteed by (62))

$$\begin{aligned}(\boldsymbol{H}_i+\mu\boldsymbol{I})^{-1}=(\overline{\boldsymbol{H}}+\mu\boldsymbol{I}+\boldsymbol{\Delta}_i)^{-1}&=\big(\boldsymbol{I}+(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\boldsymbol{\Delta}_i\big)^{-1}(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\\&=\left\{\sum_{m=0}^{\infty}(-1)^m\big[(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}\boldsymbol{\Delta}_i\big]^m\right\}(\overline{\boldsymbol{H}}+\mu\boldsymbol{I})^{-1}.\end{aligned}$$

The identity (64) holds since $\sum_{i=1}^n\boldsymbol{\Delta}_i=\boldsymbol{0}$, and hence the summation in (64) effectively starts at $m=2$.

Putting the above two bounds together back in (61), we arrive at

$$\Big\|\boldsymbol{I}-\frac{1}{n}\sum_{i=1}^n(\boldsymbol{H}_i+\mu\boldsymbol{I})^{-1}\overline{\boldsymbol{H}}\Big\|_2\le\theta_1=1-\frac{\sigma}{\sigma+\mu}+\frac{L}{L+\mu}\frac{\beta^2}{(\sigma+\mu)(\sigma+\mu-\beta)}. \tag{65}$$

Putting together (59) and (65), and plugging back into (56), we can bound the convergence error by:

$$\begin{aligned}\sqrt{n}\big\|\overline{\boldsymbol{y}}^{(t)}-\boldsymbol{y}^{\mathsf{opt}}\big\|_2=&\sqrt{n}\big\|\overline{\boldsymbol{x}}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}\big\|_2\\&\le\theta_1\sqrt{n}\big\|\overline{\boldsymbol{y}}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}\big\|_2+\frac{1}{\sigma+\mu}\frac{\beta}{\sigma+\mu}\big\|\boldsymbol{s}^{(t-1)}-\nabla f(\boldsymbol{y}^{(t-1)})\big\|_2\\&\quad+\Big(\frac{L}{L+\mu}\frac{\beta}{\sigma+\mu}+\frac{\beta}{\sigma+\mu}\Big)\big\|\boldsymbol{y}^{(t-1)}-\boldsymbol{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\big\|_2.\end{aligned}\tag{66}$$

E.2. Consensus error

Using the identity $\overline{\boldsymbol{y}}^{(t)}=\Big(\frac{1}{n}\boldsymbol{1}_n^\top\otimes\boldsymbol{I}_d\Big)\boldsymbol{y}^{(t)}$ and the update rule (28c), we can demonstrate that

$$\begin{aligned}&\Big\|\boldsymbol{y}^{(t)}-\boldsymbol{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\Big\|_2\\=&\Big\|\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top\otimes\boldsymbol{I}_d\Big)\boldsymbol{y}^{(t)}\Big\|_2\\=&\Big\|\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)\Big(\boldsymbol{y}^{(t-1)}-(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\boldsymbol{s}^{(t-1)}\Big)\Big\|_2\\\le&\Big\|\Big(\boldsymbol{W}^K-\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top\Big)\otimes\boldsymbol{I}_d\Big\|_2\Big\|\boldsymbol{y}^{(t-1)}-\boldsymbol{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}-\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top\otimes\boldsymbol{I}_d\Big)\Big((\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\boldsymbol{s}^{(t-1)}\Big)\Big\|_2 &(67)\\\le&\alpha\|\boldsymbol{y}^{(t-1)}-\boldsymbol{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2+\alpha\Big\|\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\boldsymbol{s}^{(t-1)}\Big\|_2, &(68)\end{aligned}$$

where (67) is due to the following equality:

$$\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)=\Big[\Big(\boldsymbol{W}^K-\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top\Big)\otimes\boldsymbol{I}_d\Big]\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top\otimes\boldsymbol{I}_d\Big),$$

which holds because the property of the averaging operator $\big(\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\big)$,

$$\Big(\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)=\Big[\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\Big(\boldsymbol{I}_d-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\Big]\otimes\boldsymbol{I}_n=\mathbf{0},$$

and the fact that $(\boldsymbol{A}\otimes\boldsymbol{B})(\boldsymbol{C}\otimes\boldsymbol{D})=(\boldsymbol{A}\boldsymbol{C})\otimes(\boldsymbol{B}\boldsymbol{D})$.

We rearrange the second term in (68) as

$$\begin{aligned}
&\Big\|\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\boldsymbol{s}^{(t-1)}\Big\|_2\\
=&\Big\|\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\Big(\boldsymbol{s}^{(t-1)}-\nabla f(\boldsymbol{y}^{(t-1)})\Big)\\
&+\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\Big(\nabla f(\boldsymbol{y}^{(t-1)})-\nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\Big)\\
&+\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\Big(\nabla f(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})-\nabla f(\mathbf{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}})\Big)\Big\|_2\\
=&\Big\|\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\Big(\boldsymbol{s}^{(t-1)}-\nabla f(\boldsymbol{y}^{(t-1)})\Big)\\
&+\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}})(\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\\
&+\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\Big((\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}-(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}+\mu\boldsymbol{I}_{nd})^{-1}\Big)(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}})(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}-\mathbf{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}})\Big\|_2.
\end{aligned}$$

Using similar trick as in (58), the above quantity can be further upper bounded as

$$\begin{aligned}
&\Big\|\Big(\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\boldsymbol{s}^{(t-1)}\Big\|_2\\
\le&\Big\|\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big\|_2\big\|(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\big\|_2\big\|\boldsymbol{s}^{(t-1)}-\nabla f(\boldsymbol{y}^{(t-1)})\big\|_2\\
&+\Big\|\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big\|_2\Big\|(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}\Big\|_2\big\|\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}\big\|_2\|\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2\\
&+\sqrt{n}\Big\|\boldsymbol{I}_{nd}-\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big\|_2\Big\|(\boldsymbol{H}+\mu\boldsymbol{I}_{nd})^{-1}(\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}-\boldsymbol{H})(\boldsymbol{I}_{nd}+\mu\boldsymbol{I}_n\otimes\overline{\boldsymbol{H}}^{-1})^{-1}\Big\|_2\|\overline{\boldsymbol{y}}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}\|_2.
\end{aligned}\tag{69}$$

Combine (68) and (69), we conclude that

$$\begin{aligned}
\|\boldsymbol{y}^{(t)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\|_2\le&\left(\alpha+\frac{\alpha L}{\sigma+\mu}\right)\|\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2+\frac{\alpha}{\sigma+\mu}\|\boldsymbol{s}^{(t-1)}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2\\
&+\frac{\alpha L}{L+\mu}\frac{\beta}{\sigma+\mu}\sqrt{n}\|\overline{\boldsymbol{y}}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}\|_2.
\end{aligned}\tag{70}$$

E.3. Gradient estimation error

In view of the fundamental theorem of calculus and the definition of β, it holds that

$$\|\nabla(f-f_j)(\boldsymbol{x})-\nabla(f-f_j)(\boldsymbol{y})\|_2=\left\|\left[\int_0^1\nabla^2(f-f_j)\big(c\boldsymbol{x}+(1-c)\boldsymbol{y}\big)dc\right](\boldsymbol{x}-\boldsymbol{y})\right\|_2\le\beta\|\boldsymbol{x}-\boldsymbol{y}\|_2.$$

To begin, the update formulas (10) and (11) are equivalent to

$$\boldsymbol{y}^{(t)} =(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\boldsymbol{x}^{(t-1)}, \tag{71}$$

$$\boldsymbol{s}^{(t)} =(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\boldsymbol{s}^{(t-1)} + \nabla F(\boldsymbol{y}^{(t)}) - \nabla F(\boldsymbol{y}^{(t-1)}). \tag{72}$$

Note that, since

$$\begin{aligned}\left(\boldsymbol{W} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\right)^K &= \left(\boldsymbol{W} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\right)\left(\boldsymbol{W} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\right)\cdots\left(\boldsymbol{W} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\right)\\ &= \left(\boldsymbol{W}^2 - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\right)\cdots\left(\boldsymbol{W} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\right) = \boldsymbol{W}^K - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top,\end{aligned}$$

we have the mixing rate of $\boldsymbol{W}^K$ is

$$\alpha := \|\boldsymbol{W}^K - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\| = \|\boldsymbol{W} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top\|^K = \alpha_0^K.$$

In view of the equivalent update rule (72),

$$\begin{aligned}\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2 =& \Big\|(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\boldsymbol{s}^{(t-1)} + \nabla F(\boldsymbol{y}^{(t)}) - \nabla F(\boldsymbol{y}^{(t-1)}) - \nabla f(\boldsymbol{y}^{(t)})\Big\|_2\\ =& \Big\|(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\Big(\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)})\Big) + (\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\nabla f(\boldsymbol{y}^{(t-1)})\\ &+ \nabla F(\boldsymbol{y}^{(t)}) - \nabla F(\boldsymbol{y}^{(t-1)}) - \nabla f(\boldsymbol{y}^{(t)})\Big\|_2\\ =& \Big\|(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\Big(\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)})\Big) + \nabla(F-f)(\boldsymbol{y}^{(t)})\\ &+ (\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\nabla f(\boldsymbol{y}^{(t-1)}) - \nabla F(\boldsymbol{y}^{(t-1)})\Big\|_2\end{aligned}$$

Subtract and add $\Big((\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top) \otimes \boldsymbol{I}_d\Big)\Big(\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)})\Big)$, $\nabla(f-F)(\mathbf{1}_n \otimes \overline{\boldsymbol{y}}^{(t)})$ and $\nabla(f-F)(\mathbf{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}})$ to the previous equation, and rearrange terms,

$$\begin{aligned}\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2 =& \Big\|\Big[(\boldsymbol{W}^K \otimes \boldsymbol{I}_d) - (\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top) \otimes \boldsymbol{I}_d\Big]\Big(\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)})\Big)\\ &+ \nabla(F-f)(\boldsymbol{y}^{(t)}) - \nabla(F-f)(\mathbf{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}})\\ &+ (\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\Big(\nabla f(\boldsymbol{y}^{(t-1)}) - \nabla f(\mathbf{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}})\Big) - \Big[\nabla F(\boldsymbol{y}^{(t-1)}) - \nabla F(\mathbf{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}})\Big]\\ &+ \Big[(\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top) \otimes \boldsymbol{I}_d\Big]\Big(\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)})\Big)\Big\|_2\\ \leq& \alpha\Big\|\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)})\Big\|_2 + \beta\|\boldsymbol{y}^{(t)} - \mathbf{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}}\|_2\\ &+ \Big\|(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\Big(\nabla f(\boldsymbol{y}^{(t-1)}) - \nabla f(\mathbf{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}})\Big) - \Big[\nabla F(\boldsymbol{y}^{(t-1)}) - \nabla F(\mathbf{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}})\Big]\\ &\quad + \Big[(\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top) \otimes \boldsymbol{I}_d\Big]\Big(\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)})\Big)\Big\|_2.\end{aligned} \tag{73}$$

Using the facts $\Big[(\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes\boldsymbol{I}_d\Big]\boldsymbol{s}^{(t-1)} = \Big[(\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes\boldsymbol{I}_d\Big]\nabla F(\boldsymbol{y}^{(t-1)})$ and $\Big[(\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes\boldsymbol{I}_d\Big]\nabla(F-f)(\mathbf{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}}) = \mathbf{0}$, the last term of (73) becomes

$$\Big\|\Big[(\boldsymbol{W}^K \otimes \boldsymbol{I}_d) - (\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top) \otimes \boldsymbol{I}_d\Big]\Big(\nabla(f-F)(\boldsymbol{y}^{(t-1)}) - \nabla(f-F)(\mathbf{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)})\Big)$$

$$
\begin{aligned}
&+\Big(\nabla(f-F)(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})-\nabla(f-F)(\mathbf{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}})\Big)\\
&+\Big[(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)-\boldsymbol{I}_{nd}\Big]\Big(\nabla F(\boldsymbol{y}^{(t-1)})-\nabla F(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\Big)\Big\|_2\\
\le&\Big\|(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)-(\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes\boldsymbol{I}_d\Big\|_2\|\nabla(f-F)(\boldsymbol{y}^{(t-1)})-\nabla(f-F)(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\|_2\\
&+\|\nabla(f-F)(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})-\nabla(f-F)(\mathbf{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}})\|_2\\
&+\Big\|(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)-\boldsymbol{I}_{nd}\Big\|_2\|\nabla F(\boldsymbol{y}^{(t-1)})-\nabla F(\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)})\|_2\\
\le&\alpha\beta\|\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2+\beta\sqrt{n}\|\overline{\boldsymbol{y}}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}\|_2+(\alpha+1)L\|\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2.
\end{aligned}
\tag{74}
$$

We used $\Big\|(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)-\boldsymbol{I}_{nd}\Big\|_2=\Big\|(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)-\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)+\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)-\boldsymbol{I}_{nd}\Big\|_2\le\Big\|(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)-\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)\Big\|_2+\Big\|\Big(\frac{1}{n}\mathbf{1}_n^\top\otimes\boldsymbol{I}_d\Big)-\boldsymbol{I}_{nd}\Big\|_2\le\alpha+1$ to obtain the last inequality.

Combining (73) and (74), we obtain the bound

$$
\begin{aligned}
\|\boldsymbol{s}^{(t)}-\nabla f(\boldsymbol{y}^{(t)})\|_2\le&\alpha\big\|\boldsymbol{s}^{(t-1)}-\nabla f(\boldsymbol{y}^{(t-1)})\big\|_2+\beta\|\boldsymbol{y}^{(t)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\|_2+\beta\sqrt{n}\|\overline{\boldsymbol{y}}^{(t)}-\boldsymbol{y}^{\mathsf{opt}}\|_2\\
&+\big(\alpha\beta+(\alpha+1)L\big)\|\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2+\beta\sqrt{n}\|\overline{\boldsymbol{y}}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}\|_2.
\end{aligned}
\tag{75}
$$

E.4. Linear system

Recall the definitions $\eta=\frac{1}{\sigma+\mu}$, $\gamma=\frac{L}{L+\sigma}$ and the error vector (20). Combining (66), (70) and (75) leads to the matrix $\boldsymbol{G}$ defined in (30).

Appendix F. Proof of Lemma 16

The proof follows the same procedures as the proof of Lemma 15. (i) In Appendix F.1, we bound the convergence error $\sqrt{n}\|\overline{\boldsymbol{y}}^{(t)}-\boldsymbol{y}^{\mathsf{opt}}\|_2$; (ii) in Appendix F.2, we bound the parameter consensus error $\|\boldsymbol{y}^{(t)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\|_2$; (iii) finally, using the bound we obtained in Appendix E.3 of the gradient estimation error, we create induction inequalities of $\|\boldsymbol{y}^{(t)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\|_2$, $\sqrt{n}\|\overline{\boldsymbol{y}}^{(t)}-\boldsymbol{y}^{\mathsf{opt}}\|_2$ and $L^{-1}\|\boldsymbol{s}_j^{(t)}-\nabla f(\boldsymbol{y}^{(t)})\|_2$ in Appendix F.3 to conclude the proof. For consistency and simplicity, we use the same definitions of $\boldsymbol{x}^+$ in (53), $\eta=\frac{1}{\sigma+\mu}$, and $\gamma=\frac{L}{L+\sigma}$ as in the proof of Lemma 15.

F.1. Convergence error

We continue to decompose the convergence error as (56), and bound the two terms respectively.

1. For the term $\|\overline{\boldsymbol{x}}^{(t-1)}-\overline{\boldsymbol{x}}^{+}\|_2$, we first subtract (54a) from (54b), which gives

$$
\begin{aligned}
\nabla f_j(\boldsymbol{x}_j^{(t-1)})-\nabla f_j(\boldsymbol{x}_j^{+})+\mu(\boldsymbol{x}_j^{(t-1)}-\boldsymbol{x}_j^{+})=&\nabla f(\boldsymbol{y}_j^{(t-1)})-\boldsymbol{s}_j^{(t-1)}\\
&+\nabla(f-f_j)(\overline{\boldsymbol{y}}^{(t-1)})-\nabla(f-f_j)(\boldsymbol{y}_j^{(t-1)})+\mu(\boldsymbol{y}_j^{(t-1)}-\overline{\boldsymbol{y}}^{(t-1)}),
\end{aligned}
$$

then use the strong convexity of $f_j(\cdot)$ and the definition of β to bound both sides,

$$\|\nabla f_j(\boldsymbol{x}_j^{(t-1)}) - \nabla f_j(\boldsymbol{x}_j^{+}) + \mu(\boldsymbol{x}_j^{(t-1)} - \boldsymbol{x}_j^{+})\|_2 \geq (\sigma+\mu)\|\boldsymbol{x}_j^{(t-1)} - \boldsymbol{x}_j^{+}\|_2,$$

$$\begin{aligned}&\big\|\nabla f(\boldsymbol{y}_j^{(t-1)}) - \boldsymbol{s}_j^{(t-1)} + \nabla(f-f_j)(\overline{\boldsymbol{y}}^{(t-1)}) - \nabla(f-f_j)(\boldsymbol{y}_j^{(t-1)}) + \mu(\boldsymbol{y}_j^{(t-1)} - \overline{\boldsymbol{y}}^{(t-1)})\big\|_2 \\ \leq&(\beta+\mu)\|\boldsymbol{y}_j^{(t-1)} - \overline{\boldsymbol{y}}^{(t-1)}\|_2 + \|\nabla f(\boldsymbol{y}_j^{(t-1)}) - \boldsymbol{s}_j^{(t-1)}\|_2.\end{aligned}$$

Therefore, combining the above two inequalities, we have

$$\|\boldsymbol{x}_j^{(t-1)} - \boldsymbol{x}_j^{+}\|_2 \leq \frac{1}{\sigma+\mu}\|\nabla f(\boldsymbol{y}_j^{(t-1)}) - \boldsymbol{s}_j^{(t-1)}\|_2 + \frac{\beta+\mu}{\sigma+\mu}\|\boldsymbol{y}_j^{(t-1)} - \overline{\boldsymbol{y}}^{(t-1)}\|_2. \tag{76}$$

Subtracting the optimality conditions in (55),

$$\begin{aligned}\boldsymbol{0} \in& \frac{1}{n}\sum_j \nabla f_j(\boldsymbol{x}_j^{(t-1)}) - \frac{1}{n}\sum_j \nabla f_j(\boldsymbol{x}_j^{+}) + \mu(\overline{\boldsymbol{x}}^{(t-1)} - \overline{\boldsymbol{x}}^{+}) \\ =& \frac{1}{n}\sum_j \left(\nabla f_j(\boldsymbol{x}_j^{(t-1)}) - L\boldsymbol{x}_j^{(t-1)}\right) - \frac{1}{n}\sum_j \Big(\nabla f_j(\boldsymbol{x}_j^{+}) - L\boldsymbol{x}_j^{+}\Big) + (L+\mu)(\overline{\boldsymbol{x}}^{(t-1)} - \overline{\boldsymbol{x}}^{+}).\end{aligned}$$

Note the gradient of the function $L\boldsymbol{x} - \nabla f_j(\boldsymbol{x})$ is a $(L-\sigma)$-Lipschitz function. Taking the ℓ_2 norm and plugging in (76), we have

$$\begin{aligned}\|\overline{\boldsymbol{x}}^{(t-1)} - \overline{\boldsymbol{x}}^{+}\|_2 \leq& \frac{1}{L+\mu}\Big\|\frac{1}{n}\sum_j \Big(\big[L\boldsymbol{x}_j^{(t-1)} - \nabla f_j(\boldsymbol{x}_j^{(t-1)})\big] - \big[L\boldsymbol{x}_j^{+} - \nabla f_j(\boldsymbol{x}_j^{+})\big]\Big)\Big\|_2 \\ \leq& \frac{1}{L+\mu}\frac{1}{n}\sum_j \Big\|\big[L\boldsymbol{x}_j^{(t-1)} - \nabla f_j(\boldsymbol{x}_j^{(t-1)})\big] - \big[L\boldsymbol{x}_j^{+} - \nabla f_j(\boldsymbol{x}_j^{+})\big]\Big\|_2 \\ \leq& \frac{L-\sigma}{L+\mu}\frac{1}{n}\sum_j \|\boldsymbol{x}_j^{(t-1)} - \boldsymbol{x}_j^{+}\|_2 \\ \leq& \frac{L-\sigma}{L+\mu}\frac{1}{\sigma+\mu}\frac{1}{n}\sum_j \big\|\nabla f(\boldsymbol{y}_j^{(t-1)}) - \boldsymbol{s}_j^{(t-1)}\big\|_2 + \frac{L-\sigma}{L+\mu}\frac{\beta+\mu}{\sigma+\mu}\frac{1}{n}\sum_j \big\|\boldsymbol{y}_j^{(t-1)} - \overline{\boldsymbol{y}}^{(t-1)}\big\|_2,\end{aligned} \tag{77}$$

where the last line follows (76).

2. For the second term $\|\overline{\boldsymbol{x}}^{+} - \boldsymbol{y}^{\mathsf{opt}}\|_2$, because of the assumption $\left(\frac{\beta}{\sigma+\mu}\right)^2 \leq \frac{\sigma}{\sigma+2\mu}$, we can invoke (Fan et al., 2019, Theorem 3.1), which is a careful analysis of the error of DANE, and bound the error as

$$\|\overline{\boldsymbol{x}}^{+} - \boldsymbol{y}^{\mathsf{opt}}\|_2 \leq \frac{\frac{\beta}{\sigma+\mu}\sqrt{\sigma^2+2\sigma\mu}+\mu}{\sigma+\mu}\|\overline{\boldsymbol{y}} - \boldsymbol{y}^{\mathsf{opt}}\|_2 := \theta_2\|\overline{\boldsymbol{y}}^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}\|_2. \tag{78}$$

Putting together (77) and (78), and plugging back into (56), we can bound the convergence error by:

$$\begin{aligned}\sqrt{n}\|\overline{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{\mathsf{opt}}\|_2 =& \sqrt{n}\|\overline{\boldsymbol{x}}^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}\|_2 \\ \leq& \theta_2\sqrt{n}\|\overline{\boldsymbol{y}}^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}\|_2 + \frac{1}{L+\mu}\frac{L}{\sigma+\mu}\|\nabla f(\boldsymbol{y}^{(t-1)}) - \boldsymbol{s}^{(t-1)}\|_2 \\ &+ \frac{\beta+\mu}{L+\mu}\frac{L}{\sigma+\mu}\|\boldsymbol{y}^{(t-1)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)}\|_2.\end{aligned} \tag{79}$$

F.2. Consensus error

Let $\boldsymbol{H}_j^{(t)} = \int_0^1 \nabla^2 f_j\big(c\boldsymbol{x}_j^{(t)} + (1-c)\boldsymbol{y}_j^{(t)}\big)\mathrm{d}c$ and $\boldsymbol{H}^{(t)} = \mathrm{diag}(\boldsymbol{H}_1^{(t)}, \boldsymbol{H}_2^{(t)}, \ldots, \boldsymbol{H}_n^{(t)})$. Via the fundamental theorem of calculus, we can solve for $\boldsymbol{x}_j^{(t-1)}$ from the optimality condition (54b) as

$$\boldsymbol{x}_j^{(t-1)} = \boldsymbol{y}_j^{(t-1)} - (\boldsymbol{H}_j^{(t-1)} + \mu\boldsymbol{I}_d)^{-1}\boldsymbol{s}_j^{(t-1)}. \tag{80}$$

Similar to (68), we decompose the consensus error as

$$\|\boldsymbol{y}^{(t)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\|_2 \le \alpha\|\boldsymbol{y}^{(t-1)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)}\|_2 + \alpha\Big\|\Big(\boldsymbol{I}_{nd} - (\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top) \otimes \boldsymbol{I}_d\Big)(\boldsymbol{H}^{(t-1)} + \mu\boldsymbol{I}_{nd})^{-1}\boldsymbol{s}^{(t-1)}\Big\|_2 \tag{81}$$

Then, we bound (81). Adding and subtracting terms and using the triangle inequality,

$$\begin{aligned}
&\Big\|\Big(\boldsymbol{I}_{nd} - (\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top) \otimes \boldsymbol{I}_d\Big)(\boldsymbol{H}^{(t-1)} + \mu\boldsymbol{I}_{nd})^{-1}\boldsymbol{s}^{(t-1)}\Big\|_2 \\
\le&\Big\|\Big(\boldsymbol{I}_{nd} - (\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top) \otimes \boldsymbol{I}_d\Big)(\boldsymbol{H}^{(t-1)} + \mu\boldsymbol{I}_{nd})^{-1}\Big(\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)}) + \nabla f(\boldsymbol{y}^{(t-1)}) - \nabla f(\boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)})\Big)\Big\|_2 \\
&+\Big\|\Big(\boldsymbol{I}_{nd} - (\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top) \otimes \boldsymbol{I}_d\Big)(\boldsymbol{H}^{(t-1)} + \mu\boldsymbol{I}_{nd})^{-1}\nabla f(\boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)})\Big\|_2
\end{aligned} \tag{82}$$

We can bound the first term in (82) as

$$\begin{aligned}
&\Big\|\Big(\boldsymbol{I}_{nd} - (\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top) \otimes \boldsymbol{I}_d\Big)(\boldsymbol{H}^{(t-1)} + \mu\boldsymbol{I}_{nd})^{-1}\Big(\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)}) + \nabla f(\boldsymbol{y}^{(t-1)}) - \nabla f(\boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)})\Big)\Big\|_2 \\
\le&\Big\|\Big(\boldsymbol{I}_{nd} - (\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top) \otimes \boldsymbol{I}_d\Big)(\boldsymbol{H}^{(t-1)} + \mu\boldsymbol{I}_{nd})^{-1}\Big\|_2\|\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)}) + \nabla f(\boldsymbol{y}^{(t-1)}) - \nabla f(\boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)})\|_2 \\
\le&\frac{1}{\sigma+\mu}\Big(\|\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)})\|_2 + \|\nabla f(\boldsymbol{y}^{(t-1)}) - \nabla f(\boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)})\|_2\Big) \\
\le&\frac{1}{\sigma+\mu}\Big(\|\boldsymbol{s}^{(t-1)} - \nabla f(\boldsymbol{y}^{(t-1)})\|_2 + L\|\boldsymbol{y}^{(t-1)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)}\|_2\Big)
\end{aligned} \tag{83}$$

Then, for the second term in (82),

$$\begin{aligned}
&\Big\|\Big(\boldsymbol{I}_{nd} - (\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top) \otimes \boldsymbol{I}_d\Big)(\boldsymbol{H}^{(t-1)} + \mu\boldsymbol{I}_{nd})^{-1}\nabla f(\boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)})\Big\|_2 \\
=&\Big\|\Big(\boldsymbol{I}_{nd} - (\frac{1}{n}\boldsymbol{1}_n\boldsymbol{1}_n^\top) \otimes \boldsymbol{I}_d\Big)\Big((\boldsymbol{H}^{(t-1)} + \mu\boldsymbol{I}_{nd})^{-1} - \big((L+\mu)\boldsymbol{I}_{nd}\big)^{-1}\Big)\nabla f(\boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)})\Big\|_2 \\
\le&\Big\|(\boldsymbol{H}^{(t-1)} + \mu\boldsymbol{I}_{nd})^{-1}(L\boldsymbol{I}_{nd} - \boldsymbol{H}^{(t-1)})\big((L+\mu)\boldsymbol{I}_{nd}\big)^{-1}\nabla f(\boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)})\Big\|_2 \\
\le&\frac{L-\sigma}{L+\mu}\frac{L}{\sigma+\mu}\sqrt{n}\|\overline{\boldsymbol{y}}^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}\|_2
\end{aligned} \tag{84}$$

Therefore, by combing (81), (82), (83) and (84), we can bound the consensus error by:

$$\begin{aligned}
\|\boldsymbol{y}^{(t)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\|_2 \le&\Big(\alpha + \frac{\alpha L}{\sigma+\mu}\Big)\|\boldsymbol{y}^{(t-1)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t-1)}\|_2 \\
&+\frac{\alpha}{\sigma+\mu}\|\nabla f(\boldsymbol{y}^{(t-1)}) - \boldsymbol{s}^{(t-1)}\|_2 + \frac{\alpha L}{L+\mu}\frac{L}{\sigma+\mu}\sqrt{n}\|\overline{\boldsymbol{y}}^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}\|_2.
\end{aligned} \tag{85}$$

F.3. Linear system

Combining (75), (85), (79), we reach the matrix claimed in (42).

Appendix G. Proof of Lemma 17

The proof follows similar procedures as the proof of Lemma 15. (i) In Appendix G.1, we bound the expected function value convergence errors $\mathbb{E}\big[\sum_{j=1}^{n}\big(f(\boldsymbol{x}_j^{(t)})-f(\boldsymbol{y}^{\mathsf{opt}})\big)\big]$ and $\mathbb{E}\big[\sum_{j=1}^{n}\big(f(\boldsymbol{y}_j^{(t)})-f(\boldsymbol{y}^{\mathsf{opt}})\big)\big]$; (ii) in Appendix G.2, we bound the expected parameter consensus error $\mathbb{E}\|\boldsymbol{y}^{(t)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\|_2^2$; (iii) in Appendix G.3, we bound the expected parameter consensus error $\mathbb{E}\|\boldsymbol{y}^{(t)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\|_2^2$; (iv) finally, we create induction inequalities of $\mathbb{E}\big[\sum_{j=1}^{n}\big(f(\boldsymbol{x}_j^{(t)})-f(\boldsymbol{y}^{\mathsf{opt}})\big)\big]$, $\mathbb{E}\big[\sum_{j=1}^{n}\big(f(\boldsymbol{y}_j^{(t)})-f(\boldsymbol{y}^{\mathsf{opt}})\big)\big]$, $\mathbb{E}\|\boldsymbol{y}^{(t)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\|_2^2$ and $\mathbb{E}\|\boldsymbol{y}^{(t)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\|_2^2$ to conclude the proof. Expectations in this section are conditioned on $\boldsymbol{x}^{(t-1)}$, $\boldsymbol{y}^{(t-1)}$ and $\boldsymbol{s}^{(t-1)}$, if not specified.

G.1. Function value convergence error

First, we bound the function value convergence error of $\boldsymbol{y}^{(t)}$ using the previous estimate $\boldsymbol{x}^{(t-1)}$. By the strong convexity of $f(\cdot)$ and the assumption of $\alpha\le 1/\kappa$,

$$\begin{aligned}\sum_{j=1}^{n} f(\boldsymbol{y}_j^{(t)}) \le& nf(\overline{\boldsymbol{y}}^{(t-1)})+\frac{L}{2}\|\boldsymbol{y}^{(t)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\|_2^2\\ \le& nf(\overline{\boldsymbol{x}}^{(t-1)})+\frac{\alpha^2 L}{2}\|\boldsymbol{x}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{x}}^{(t)}\|_2^2\\ \le& nf(\overline{\boldsymbol{x}}^{(t-1)})+\frac{\sigma}{2}\|\boldsymbol{x}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{x}}^{(t)}\|_2^2\\ =&\sum_{j=1}^{n}\Big(f(\overline{\boldsymbol{x}}^{(t-1)})+\Big\langle\nabla f(\overline{\boldsymbol{x}}^{(t-1)}),\boldsymbol{x}_j^{(t-1)}-\overline{\boldsymbol{x}}^{(t-1)}\Big\rangle+\frac{\sigma}{2}\|\boldsymbol{x}_j^{(t-1)}-\overline{\boldsymbol{x}}^{(t)}\|_2^2\Big)\\ \le&\sum_{j=1}^{n} f(\boldsymbol{x}_j^{(t-1)}).\end{aligned}\tag{86}$$

Next, we bound the function value convergence error after local update, $\sum_{j=1}^{n}\big(f(\boldsymbol{y}_j^{(t)})-f(\boldsymbol{y}^{\mathsf{opt}})\big)$. By constructing the following helper function, we can connect local updates of `Network-SVRG` to that of D-SVRG Cen et al. (2020), which is the counterpart of SVRG in the master/slave setting. For agent j at the tth time, we define the corrected sample loss function as

$$\tilde{\ell}^{(j)}(\boldsymbol{x};\boldsymbol{z})=\ell(\boldsymbol{x};\boldsymbol{z})+\Big\langle\boldsymbol{s}_j^{(t)}-\nabla f(\boldsymbol{y}_j^{(t)}),\boldsymbol{x}-\boldsymbol{y}_j^{(t)}\Big\rangle.$$

Then, define the corrected local and global loss functions as

$$\begin{aligned}h_i^{(t,j)}(\boldsymbol{x})&=\frac{1}{m}\sum_{\boldsymbol{z}\in\mathcal{M}_i}\tilde{\ell}^{(j)}(\boldsymbol{x};\boldsymbol{z})=f_i(\boldsymbol{x})+\Big\langle\boldsymbol{s}_j^{(t)}-\nabla f(\boldsymbol{y}_j^{(t)}),\boldsymbol{x}-\boldsymbol{y}_j^{(t)}\Big\rangle,\\ h^{(t,j)}(\boldsymbol{x})&=\frac{1}{n}\sum_{i}h_i^{(t,j)}(\boldsymbol{x})=f(\boldsymbol{x})+\Big\langle\boldsymbol{s}_j^{(t)}-\nabla f(\boldsymbol{y}_j^{(t)}),\boldsymbol{x}-\boldsymbol{y}_j^{(t)}\Big\rangle.\end{aligned}\tag{87}$$

Here, $h^{(t,j)}(\cdot)$ and $h_i^{(t,j)}(\cdot)$ are σ-strongly convex and L-smooth functions, and $\big\|h_i^{(t,j)}(\boldsymbol{x}) - h^{(t,j)}(\boldsymbol{x})\big\|_2 \le \beta$ by the definition of β. Let $h_*^{(t,j)}$ denote the optimum value of $h^{(t,j)}(\cdot)$.

The key observation is that the local update (27a) at agent j is the same as the update at agent j when applying D-SVRG to optimize $h^{(t,j)}$ initialized with $\boldsymbol{y}_j^{(t)}$. This is true because $\forall \boldsymbol{z} \in \mathcal{M}_j$, the sample gradient and global gradient used in D-SVRG updates at $\boldsymbol{y}_j^{(t)}$ satisfy

$$\nabla\tilde{\ell}^{(j)}(\boldsymbol{u};\boldsymbol{z}) - \nabla\tilde{\ell}^{(j)}(\boldsymbol{u}';\boldsymbol{z}) = \nabla\ell(\boldsymbol{u}';\boldsymbol{z}) - \nabla\ell(\boldsymbol{u};\boldsymbol{z}), \quad \text{and} \quad \nabla h^{(t,j)}(\boldsymbol{y}_j^{(t)}) = \boldsymbol{s}_j^{(t)},$$

which agree with (27a). Therefore, we can apply (Cen et al., 2020, Theorem 1) to bound the optimization error of optimizing $h^{(t,j)}$

$$\mathbb{E}\Big[h^{(t,j)}(\boldsymbol{x}_j^{(t)}) - h_*^{(t,j)}\Big] < \nu\Big(h^{(t,j)}(\boldsymbol{y}_j^{(t)}) - h_*^{(t)}\Big), \tag{88}$$

where $\boldsymbol{x}_j^{(t)}$ is the output at agent j produced by running one iteration of Alg. 3, which is also the output of running one iteration of D-SVRG at the same agent, ν is the convergence rate of D-SVRG, which can be bounded by $\nu \le 1 - \frac{1}{2}\frac{\sigma-2\beta}{\sigma-3\beta}$ when choosing step size $\delta = \frac{1}{40L}\big(1 - \frac{4\beta}{\sigma}\big)$ and the number of local updates $S = 160\frac{L}{\sigma}\big(1 - \frac{4\beta}{\sigma}\big)^{-2}$.

Next, we relate function value descent of $h^{(t,j)}$ to the function value descent of f. Plug in (87) and rearrange terms,

$$\begin{aligned}
f(\boldsymbol{x}_j^{(t)}) - f(\boldsymbol{y}^{\mathsf{opt}}) =& h^{(t,j)}(\boldsymbol{x}_j^{(t)}) - (1-\nu) f(\boldsymbol{y}^{\mathsf{opt}}) - \nu f(\boldsymbol{y}^{\mathsf{opt}}) - \Big\langle \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)}), \boldsymbol{x}_j^{(t)} - \boldsymbol{y}_j^{(t)} \Big\rangle \\
=& h^{(t,j)}(\boldsymbol{x}_j^{(t)}) - (1-\nu) h^{(t,j)}(\boldsymbol{y}^{\mathsf{opt}}) - \nu f(\boldsymbol{y}^{\mathsf{opt}}) \\
& - \Big\langle \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)}), \boldsymbol{x}_j^{(t)} - \boldsymbol{y}_j^{(t)} - (1-\nu)\Big(\boldsymbol{y}^{\mathsf{opt}} - \boldsymbol{y}_j^{(t)}\Big) \Big\rangle \\
\le& h^{(t,j)}(\boldsymbol{x}_j^{(t)}) - (1-\nu) h_*^{(t,j)} - \nu f(\boldsymbol{y}^{\mathsf{opt}}) \\
& - \Big\langle \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)}), \boldsymbol{x}_j^{(t)} - \boldsymbol{y}_j^{(t)} - (1-\nu)\Big(\boldsymbol{y}^{\mathsf{opt}} - \boldsymbol{y}_j^{(t)}\Big) \Big\rangle \\
=& h^{(t,j)}(\boldsymbol{x}_j^{(t)}) - h_*^{(t,j)} + \nu\Big(h_*^{(t,j)} - f(\boldsymbol{y}^{\mathsf{opt}})\Big) \\
& - \Big\langle \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)}), \boldsymbol{x}_j^{(t)} - \boldsymbol{y}_j^{(t)} - (1-\nu)\Big(\boldsymbol{y}^{\mathsf{opt}} - \boldsymbol{y}_j^{(t)}\Big) \Big\rangle,
\end{aligned}$$

where we used $h^{(t,j)}(\boldsymbol{y}^{\mathsf{opt}}) \ge h_*^{(t,j)}$ and $\nu \le 1$ to reach the last inequality.

Taking expectation on both sides and combining with (88), we reach the following function value descent of $f(\cdot)$:

$$\begin{aligned}
\mathbb{E}\left[f(\boldsymbol{x}_j^{(t)}) - f(\boldsymbol{y}^{\mathsf{opt}})\right] \le& \nu\left(h^{(t,j)}(\boldsymbol{y}_j^{(t)}) - h_*^{(t,j)}\right) + \nu\Big(h_*^{(t,j)} - f(\boldsymbol{y}^{\mathsf{opt}})\Big) \\
& - \mathbb{E}\Big[\Big\langle \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)}), \boldsymbol{x}_j^{(t)} - \boldsymbol{y}_j^{(t)} - (1-\nu)\Big(\boldsymbol{y}^{\mathsf{opt}} - \boldsymbol{y}_j^{(t)}\Big) \Big\rangle\Big] \\
=& \nu\left(f(\boldsymbol{y}_j^{(t)}) - f(\boldsymbol{y}^{\mathsf{opt}})\right) - \mathbb{E}\left[\Big\langle \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)}), \boldsymbol{x}_j^{(t)} - \boldsymbol{y}^{\mathsf{opt}} - \nu(\boldsymbol{y}_j^{(t)} - \boldsymbol{y}^{\mathsf{opt}}) \Big\rangle\right],
\end{aligned}$$

where the last line follows from (87). Summing the previous inequality over all agents and using matrix notations, we obtain the following inequality

$$\mathbb{E}\left[\sum_{j=1}^{n} f(\boldsymbol{x}_j^{(t)}) - f(\boldsymbol{y}^{\mathsf{opt}})\right] \le \nu\left[\sum_{j=1}^{n} f(\boldsymbol{y}_j^{(t)}) - f(\boldsymbol{y}^{\mathsf{opt}})\right] - \mathbb{E}\left[\Big\langle \boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)}), \boldsymbol{x}^{(t)} - \boldsymbol{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}} \Big\rangle\right]$$

$$+\nu\mathbb{E}\left[\left\langle \boldsymbol{s}^{(t)}-\nabla f(\boldsymbol{y}^{(t)}), \boldsymbol{y}^{(t)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\right\rangle\right]. \tag{89}$$

Our next step is to carefully bound the last two error terms in (89).

$$\begin{aligned}
&\left|\left\langle \boldsymbol{s}^{(t)}-\nabla f(\boldsymbol{y}^{(t)}), \boldsymbol{x}^{(t)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\right\rangle\right| \\
\le&\|\boldsymbol{s}^{(t)}-\nabla f(\boldsymbol{y}^{(t)})\|_2\|\boldsymbol{x}^{(t)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\|_2 \\
\le&\Big(\alpha\|\boldsymbol{s}^{t-1}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2+2L\|\boldsymbol{y}^{(t-1)}-\boldsymbol{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2 \\
&+2\beta\|\boldsymbol{y}^{(t-1)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\|_2+\beta\|\boldsymbol{y}^{(t)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\|_2\Big)\|\boldsymbol{x}^{(t)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\|_2 \\
\le&\frac{1}{2}\alpha L^{-1}\|\boldsymbol{s}^{t-1}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2^2+\alpha^{-1}L\|\boldsymbol{y}^{(t-1)}-\boldsymbol{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2^2+\frac{3}{2}\alpha L\|\boldsymbol{x}^{(t)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\|_2^2 \\
&+\beta\|\boldsymbol{y}^{(t-1)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\|_2^2+\frac{\beta}{2}\|\boldsymbol{y}^{(t)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\|_2^2+\frac{3\beta}{2}\|\boldsymbol{x}^{(t)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\|_2^2,
\end{aligned} \tag{90}$$

where the first inequality is due to (100), and the last inequality is obtained by Cauchy-Schwarz inequality. Similar to (89), because of the strong convexity of loss functions, we have

$$\|\boldsymbol{y}^{(t)}-\boldsymbol{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\|_2^2\le\frac{2}{\sigma}\sum_j\Big(f(\boldsymbol{y}_j^{(t)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big).$$

Then, we can further bound (90) as

$$\begin{aligned}
\left|\left\langle \boldsymbol{s}^{(t)}-\nabla f(\boldsymbol{y}^{(t)}), \boldsymbol{x}^{(t)}-\boldsymbol{y}^{\mathsf{opt}}\right\rangle\right| \le&\frac{1}{2}\alpha L^{-1}\|\boldsymbol{s}^{t-1}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2^2+\alpha^{-1}L\|\boldsymbol{y}^{(t-1)}-\overline{\boldsymbol{y}}^{(t-1)}\|_2^2 \\
&+\frac{2\beta}{\sigma}\sum_{j=1}^n\Big(f(\boldsymbol{y}_j^{(t-1)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big)+\frac{\beta}{\sigma}\sum_{j=1}^n\Big(f(\boldsymbol{x}_j^{(t-1)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big) \\
&+\Big(\frac{3\beta}{\sigma}+3\kappa\alpha\Big)\sum_{j=1}^n\Big(f(\boldsymbol{x}_j^{(t)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big).
\end{aligned} \tag{91}$$

Similarly, we have the same bound applicable for the last term of (89):

$$\begin{aligned}
\left|\left\langle \boldsymbol{s}^{(t)}-\nabla f(\boldsymbol{y}^{(t)}), \boldsymbol{y}^{(t)}-\boldsymbol{y}^{\mathsf{opt}}\right\rangle\right| \le&\frac{1}{2}\alpha L^{-1}\|\boldsymbol{s}^{t-1}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2^2+\alpha^{-1}L\|\boldsymbol{y}^{(t-1)}-\overline{\boldsymbol{y}}^{(t-1)}\|_2^2 \\
&+\frac{2\beta}{\sigma}\sum_{j=1}^n\Big(f(\boldsymbol{y}_j^{(t-1)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big)+\frac{\beta}{\sigma}\sum_{j=1}^n\Big(f(\boldsymbol{x}_j^{(t-1)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big) \\
&+\left(\frac{3\beta}{\sigma}+3\kappa\alpha\right)\sum_{j=1}^n\Big(f(\boldsymbol{x}_j^{(t-1)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big),
\end{aligned} \tag{92}$$

where the last term is due to (86).

Put together (90), (91) and (92) and taking expectation, we reach the following bound

$$\mathbb{E}\left[\sum_{j=1}^n\Big(f(\boldsymbol{x}_j^{(t)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big)\right]\le\Big(\nu\big(1+3\alpha\kappa+\frac{4\beta}{\sigma}\big)+\frac{\beta}{\sigma}\Big)\sum_{j=1}^n\Big(f(\boldsymbol{x}_j^{(t-1)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big)$$

$$
\begin{aligned}
&+\alpha L^{-1}\|\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)})\|_2^2 + 2\alpha^{-1}L\|\boldsymbol{y}^{(t-1)} - \overline{\boldsymbol{y}}^{(t-1)}\|_2^2 \\
&+\frac{4\beta}{\sigma}\sum_{j=1}^{n}\Big(f(\boldsymbol{y}_j^{(t-1)}) - f(\boldsymbol{y}^{\mathsf{opt}})\Big) + \Big(\frac{3\beta}{\sigma} + 3\kappa\alpha\Big)\,\mathbb{E}\left[\sum_{j=1}^{n}\Big(f(\boldsymbol{x}_j^{(t)}) - f(\boldsymbol{y}^{\mathsf{opt}})\Big)\right].
\end{aligned}
\tag{93}
$$

Rearranging terms, we proved the advertised bound.

G.2. Consensus error

We first bound the consensus error $\|\boldsymbol{y}^{(t)} - \mathbf{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\|_2^2/(\alpha L)$. Similar to (68),

$$
\begin{aligned}
\|\boldsymbol{y}^{(t)} - \mathbf{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\|_2^2 \leq& \alpha^2\|\boldsymbol{x}^{(t-1)} - \mathbf{1}_n \otimes \overline{\boldsymbol{x}}^{(t-1)}\|_2^2 \\
=& \alpha^2\|\boldsymbol{x}^{(t-1)} - \mathbf{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}}\|_2^2 - n\alpha^2\|\boldsymbol{y}^{\mathsf{opt}} - \overline{\boldsymbol{x}}^{(t-1)}\|_2 \\
\leq& \alpha^2\|\boldsymbol{x}^{(t-1)} - \mathbf{1}_n \otimes \boldsymbol{y}^{\mathsf{opt}}\|_2^2.
\end{aligned}
\tag{94}
$$

Then, using the strong convexity of $f(\cdot)$,

$$
\begin{aligned}
\|\boldsymbol{y}^{(t)} - \mathbf{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\|_2^2 \leq& \alpha^2\sum_{j=1}^{n}\|\boldsymbol{x}_j^{(t-1)} - \boldsymbol{y}^{\mathsf{opt}}\|_2^2 \\
\leq& \frac{2\alpha^2}{\sigma}\sum_{j=1}^{n}\Big(f(\boldsymbol{x}_j^{(t-1)}) - f(\boldsymbol{y}^{\mathsf{opt}})\Big).
\end{aligned}
\tag{95}
$$

G.3. Gradient estimation error

To bound the gradient estimation error, we note that

$$
\begin{aligned}
\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2 =& \|(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\boldsymbol{s}^{t-1} + \nabla F(\boldsymbol{y}^{(t)}) - \nabla F(\boldsymbol{y}^{(t-1)}) - \nabla f(\boldsymbol{y}^{(t)})\|_2 \\
=& \Big\|(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\Big(\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)})\Big) + (\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\nabla f(\boldsymbol{y}^{(t-1)}) - \nabla f(\boldsymbol{y}^{(t-1)}) \\
&+ \nabla F(\boldsymbol{y}^{(t)}) - \nabla F(\boldsymbol{y}^{(t-1)}) + \nabla f(\boldsymbol{y}^{(t-1)}) - \nabla f(\boldsymbol{y}^{(t)})\Big\|_2 \\
\leq& \Big\|(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\Big(\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)})\Big)\Big\|_2 + \Big\|(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\nabla f(\boldsymbol{y}^{(t-1)}) - \nabla f(\boldsymbol{y}^{(t-1)})\Big\|_2 \\
&+ \|\nabla(F - f)(\boldsymbol{y}^{(t)}) + \nabla(F - f)(\boldsymbol{y}^{(t-1)})\|_2.
\end{aligned}
\tag{96}
$$

We then bound the three terms in (96) respectively.

1. The first term can be bounded as

$$
\begin{aligned}
&\|(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)(\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)}))\|_2 \\
=& \Big\|(\boldsymbol{W}^K \otimes \boldsymbol{I}_d)\big(\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)})\big) - \Big((\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top) \otimes \boldsymbol{I}_d\Big)\big(\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)})\big)\Big\|_2 \\
&+ \Big\|\Big((\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top) \otimes \boldsymbol{I}_d\Big)\Big(\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)})\Big)\Big\|_2 \\
\leq& \alpha\|\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)})\|_2 + \Big\|\Big((\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top) \otimes \boldsymbol{I}_d\Big)\big(\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)})\big)\Big\|_2
\end{aligned}
$$

$$
\begin{aligned}
=&\alpha\|\boldsymbol{s}^{t-1}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2+\Big\|\Big((\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes\boldsymbol{I}_d\Big)\Big(\nabla(F-f)(\boldsymbol{y}^{t-1})-\nabla(F-f)(\boldsymbol{y}^{\mathsf{opt}})\Big)\Big\|_2\\
\leq&\alpha\|\boldsymbol{s}^{t-1}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2+\beta\|\boldsymbol{y}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}\|_2, \qquad (97)
\end{aligned}
$$

where we used the fact $\Big\|\Big((\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes\boldsymbol{I}_d\Big)\Big\|_2=1$ and the definition of β to reach the last inequality.

2. As for the second term in (96), we have

$$
\begin{aligned}
&\Big\|(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)\nabla f(\boldsymbol{y}^{(t-1)})-\nabla f(\boldsymbol{y}^{(t-1)})\Big\|_2\\
\leq&\Big\|(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)\nabla f(\boldsymbol{y}^{(t-1)})-\Big((\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes\boldsymbol{I}_d\Big)\nabla f(\boldsymbol{y}^{(t-1)})\Big\|_2\\
&+\Big\|\Big((\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes\boldsymbol{I}_d\Big)\nabla f(\boldsymbol{y}^{(t-1)})-\nabla f(\boldsymbol{y}^{(t-1)})\Big\|_2\\
\leq&2\Big\|\Big((\frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)\otimes\boldsymbol{I}_d\Big)\nabla f(\boldsymbol{y}^{(t-1)})-\nabla f(\boldsymbol{y}^{(t-1)})\Big\|_2\\
\leq&2\|\nabla f(\overline{\boldsymbol{y}}^{(t-1)})-\nabla f(\boldsymbol{y}^{(t-1)})\|_2\\
\leq&2L\|\boldsymbol{y}^{(t-1)}-\overline{\boldsymbol{y}}^{(t-1)}\|_2, \qquad (98)
\end{aligned}
$$

where the third inequality follows from the similar trick we used to obtain (94).

3. Using the triangle inequality and the definition of β, the last term in (96) can be bounded by

$$
\|\nabla(F-f)(\boldsymbol{y}^{(t)})+\nabla(F-f)(\boldsymbol{y}^{(t-1)})\|_2\leq\beta\|\boldsymbol{y}^{(t)}-\boldsymbol{y}^{\mathsf{opt}}\|_2+\beta\|\boldsymbol{y}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}\|_2. \qquad (99)
$$

Combining (96), (97), (98) and (99), the gradient estimation error can be bounded by

$$
\begin{aligned}
\|\boldsymbol{s}^{(t)}-\nabla f(\boldsymbol{y}^{(t)})\|_2\leq&\alpha\|\boldsymbol{s}^{t-1}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2+2\beta\|\boldsymbol{y}^{(t-1)}-\boldsymbol{y}^{\mathsf{opt}}\|_2\\
&+\beta\|\boldsymbol{y}^{(t)}-\boldsymbol{y}^{\mathsf{opt}}\|_2+2L\|\boldsymbol{y}^{(t-1)}-\overline{\boldsymbol{y}}^{(t-1)}\|_2. \qquad (100)
\end{aligned}
$$

Because of the strong convexity, $\|\boldsymbol{y}-\boldsymbol{y}^{\mathsf{opt}}\|_2^2\leq\frac{2}{\sigma}\sum_{j=1}^n\big(f(\boldsymbol{y}_j)-f(\boldsymbol{y}^{\mathsf{opt}})\big)$. Combining with (86), we reached the following bound

$$
\begin{aligned}
\|\boldsymbol{s}^{(t)}-\nabla f(\boldsymbol{y}^{(t)})\|_2^2\leq&4\alpha^2\|\boldsymbol{s}^{t-1}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2^2+\frac{32\beta^2}{\sigma}\sum_{j=1}^n\Big(f(\boldsymbol{y}_j^{(t-1)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big)\\
&+\frac{8\beta^2}{\sigma}\sum_{j=1}^n\Big(f(\boldsymbol{x}_j^{(t-1)})-f(\boldsymbol{y}^{\mathsf{opt}})\Big)+16L^2\|\boldsymbol{y}^{(t-1)}-\overline{\boldsymbol{y}}^{(t-1)}\|_2^2. \qquad (101)
\end{aligned}
$$

G.4. Linear System

Combining (86), (95), (93), and (101), we obtain the claimed linear system.

Appendix H. Proof of Lemma 18

Similar to the proof of Lemma 17, we bound the following four terms: (i) Expected gradient convergence errors $\mathbb{E}\|\nabla f(\boldsymbol{x}^{(t)})\|_2^2$ and $\mathbb{E}\|\nabla f(\boldsymbol{y}^{(t)})\|_2^2$ in Appendix H.1; (ii) Expected consensus error: $\mathbb{E}\|\boldsymbol{y}^{(t)} - \boldsymbol{1}_n \otimes \overline{\boldsymbol{y}}^{(t)}\|_2^2$ in Appendix H.2; (iii) Expected gradient estimation error: $\mathbb{E}\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2^2$ in Appendix H.3. Then conclude the proof by creating induction inequalities. Expectations in this section are also conditioned on $\boldsymbol{x}^{(t-1)}$, $\boldsymbol{y}^{(t-1)}$ and $\boldsymbol{s}^{(t-1)}$, if not specified.

H.1. Gradient convergence error

To bound the function gradient convergence error, we analyze the same helper function defined in (87), where we can apply (Cen et al., 2020, Theorem 2) to bound the convergence error of $h^{(t,j)}(\cdot)$ as

$$\mathbb{E}\left[\|\nabla h^{(t,j)}(\boldsymbol{x}_j^{(t)})\|_2^2\right] < \nu\|\nabla h^{(t,j)}(\boldsymbol{y}_j^{(t)})\|_2^2,$$

where ν is the convergence rate of D-SARAH in (Cen et al., 2020, Theorem 2) following similar reasonings as Section G.1. By setting $\delta = \frac{2}{L}\frac{1-8(\frac{\beta}{\sigma})^2}{9-8(\frac{\beta}{\sigma})^2}$ and $S = \frac{2L}{\sigma}\frac{9-8(\frac{\beta}{\sigma})^2}{\left(1-8(\frac{\beta}{\sigma})^2\right)^2}$, ν can be bounded by $\nu \le \frac{1}{2}\frac{1}{1-4(\frac{\beta}{\sigma})^2}$.

Then, plugging in (87) and taking expectation, we have

$$\begin{aligned}
&\mathbb{E}\left[\|\nabla f(\boldsymbol{x}_j^{(t)})\|_2^2\right] \\
&\quad= \mathbb{E}\left[\|\nabla h^{(t,j)}(\boldsymbol{x}_j^{(t)}) - \boldsymbol{s}_j^{(t)} + \nabla f(\boldsymbol{y}_j^{(t)})\|_2^2\right] \\
&\quad= \mathbb{E}\left[\|\nabla h^{(t,j)}(\boldsymbol{x}_j^{(t)})\|_2^2\right] + \|\boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)})\|_2^2 - 2\mathbb{E}\left[\left\langle \nabla h^{(t,j)}(\boldsymbol{x}_j^{(t)}), \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)})\right\rangle\right] \\
&\quad= \mathbb{E}\left[\|\nabla h^{(t,j)}(\boldsymbol{x}_j^{(t)})\|_2^2\right] - \|\boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)})\|_2^2 - 2\mathbb{E}\left[\left\langle \nabla f(\boldsymbol{x}_j^{(t)}), \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)})\right\rangle\right] \\
&\quad\le \nu\|\nabla f(\boldsymbol{y}_j^{(t)}) - \nabla f(\boldsymbol{y}_j^{(t)}) + \boldsymbol{s}_j^{(t)}\|_2^2 - \|\boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)})\|_2^2 - 2\mathbb{E}\left[\left\langle \nabla f(\boldsymbol{x}_j^{(t)}), \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)})\right\rangle\right] \\
&\quad= \nu\|\nabla f(\boldsymbol{y}_j^{(t)})\|_2^2 - 2\nu\left\langle \nabla f(\boldsymbol{y}_j^{(t)}), \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)})\right\rangle - 2\mathbb{E}\left[\left\langle \nabla f(\boldsymbol{x}_j^{(t)}), \boldsymbol{s}_j^{(t)} - \nabla f(\boldsymbol{y}_j^{(t)})\right\rangle\right],
\end{aligned}$$

where we apply D-SARAH's convergence result in the fourth step. Summing the previous inequality over all agents, we have

$$\begin{aligned}
&\mathbb{E}\left[\|\nabla f(\boldsymbol{x}^{(t)})\|_2^2\right] \\
&\quad\le \nu\|\nabla f(\boldsymbol{y}^{(t)})\|_2^2 - 2\nu\left\langle \nabla f(\boldsymbol{y}^{(t)}), \boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\right\rangle - 2\mathbb{E}\left[\left\langle \nabla f(\boldsymbol{x}^{(t)}), \boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\right\rangle\right] \\
&\quad\le \nu\|\nabla f(\boldsymbol{y}^{(t)})\|_2^2 + 2\nu\|\nabla f(\boldsymbol{y}^{(t)})\|_2\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2 + 2\mathbb{E}\big[\|\nabla f(\boldsymbol{x}^{(t)})\|_2\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2\big].
\end{aligned}$$

Using the same method as bounding (90), (91) and (92), we can prove

$$\begin{aligned}
2\mathbb{E}\big[\|\nabla f(\boldsymbol{x}^{(t)})\|_2\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2\big] \le \left(\frac{3\beta}{\sigma} + 3\alpha\kappa\right)\mathbb{E}\left[\|\nabla f(\boldsymbol{x}^{(t)})\|_2^2\right] + \frac{2\beta}{\sigma}\|\nabla f(\boldsymbol{y}^{(t-1)})\|_2^2 \\
+ \frac{\beta}{\sigma}\|\nabla f(\boldsymbol{x}^{(t-1)})\|_2^2 + \frac{\alpha}{\kappa}\|\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)})\|_2^2
\end{aligned}$$

$$
\begin{aligned}
&+\frac{2L^2}{\alpha\kappa}\|\boldsymbol{y}^{(t-1)}-\overline{\boldsymbol{y}}^{(t-1)}\|_2^2,\\
2\nu\|\nabla f(\boldsymbol{y}^{(t)})\|_2\|\boldsymbol{s}^{(t)}-\nabla f(\boldsymbol{y}^{(t)})\|_2 \leq&\nu\left(\frac{4\beta}{\sigma}+3\alpha\kappa\right)\|\nabla f(\boldsymbol{x}^{(t-1)})\|_2^2+\frac{2\beta}{\sigma}\|\nabla f(\boldsymbol{y}^{(t-1)})\|_2^2\\
&+\frac{\alpha}{\kappa}\|\boldsymbol{s}^{t-1}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2^2+\frac{2L^2}{\alpha\kappa}\|\boldsymbol{y}^{(t-1)}-\overline{\boldsymbol{y}}^{(t-1)}\|_2^2.
\end{aligned}
$$

To sum up,

$$
\begin{aligned}
\mathbb{E}\left[\|\nabla f(\boldsymbol{x}^{(t)})\|_2^2\right] \leq&\left(\nu\left(1+\frac{4\beta}{\sigma}+3\alpha\kappa\right)+\frac{\beta}{\sigma}\right)\|\nabla f(\boldsymbol{x}^{(t-1)})\|_2^2\\
&+3\left(\frac{\beta}{\sigma}+\alpha\kappa\right)\mathbb{E}\left[\|\nabla f(\boldsymbol{x}^{(t)})\|_2^2\right]+\frac{4\beta}{\sigma}\mathbb{E}\|\nabla f(\boldsymbol{y}^{(t-1)})\|_2^2\\
&+\frac{2\alpha}{\kappa}\|\boldsymbol{s}^{t-1}-\nabla f(\boldsymbol{y}^{(t-1)})\|_2^2+\frac{4L^2}{\alpha\kappa}\|\boldsymbol{y}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t-1)}\|_2^2. \qquad (102)
\end{aligned}
$$

We then show the proof for the term $\|\nabla f(\boldsymbol{y}^{(t)})\|_2^2$, which claims that the averaging process does not increase the sum of the squared norm of gradient when $\alpha \leq 1/\kappa$. We denote the Hessian of the quadratic function $f(\cdot)$ by $\overline{\boldsymbol{H}}=\nabla^2 f(\cdot)$, and have

$$
\begin{aligned}
\|\nabla f(\boldsymbol{y}^{(t)})\|_2^2 =&\sum_{j=1}^n\left\|\nabla f(\overline{\boldsymbol{y}}^{(t)})+\overline{\boldsymbol{H}}(\boldsymbol{y}_j^{(t)}-\overline{\boldsymbol{y}}^{(t)})\right\|_2^2\\
\leq& n\|\nabla f(\overline{\boldsymbol{y}}^{(t)})\|_2^2+L^2\sum_{j=1}^n\|\boldsymbol{y}_j^{(t)}-\overline{\boldsymbol{y}}^{(t)}\|_2^2\\
=& n\|\nabla f(\overline{\boldsymbol{x}}^{(t-1)})\|_2^2+L^2\|(\boldsymbol{W}^K\otimes\boldsymbol{I}_d)\boldsymbol{x}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{x}}^{(t-1)}\|_2^2\\
\leq& n\|\nabla f(\overline{\boldsymbol{x}}^{(t-1)})\|_2^2+\alpha^2L^2\|\boldsymbol{x}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{x}}^{(t-1)}\|_2^2\\
\leq& n\|\nabla f(\overline{\boldsymbol{x}}^{(t-1)})\|_2^2+\alpha^2\kappa^2\sum_{j=1}^n\|\overline{\boldsymbol{H}}(\boldsymbol{x}_j^{(t-1)}-\overline{\boldsymbol{x}}^{(t-1)})\|_2^2\\
\leq&\sum_{j=1}^n\left\|\nabla f(\overline{\boldsymbol{x}}^{(t-1)})+\overline{\boldsymbol{H}}(\boldsymbol{x}_j^{(t-1)}-\overline{\boldsymbol{x}}^{(t-1)})\right\|_2^2=\|\nabla f(\boldsymbol{x}^{(t-1)})\|_2^2. \qquad (103)
\end{aligned}
$$

H.2. Consensus error

By the property of $\boldsymbol{W}^K$ and the strong convexity of f, we have

$$
\begin{aligned}
\|\boldsymbol{y}^{(t)}-\mathbf{1}_n\otimes\overline{\boldsymbol{y}}^{(t)}\|_2^2 \leq&\alpha^2\|\boldsymbol{x}^{(t-1)}-\mathbf{1}_n\otimes\overline{\boldsymbol{x}}^{(t-1)}\|_2^2\\
\leq&\alpha^2\|\boldsymbol{x}^{(t-1)}-\mathbf{1}_n\otimes\boldsymbol{y}^{\mathsf{opt}}\|_2^2\\
\leq&\frac{\alpha^2}{\sigma^2}\|\nabla f(\boldsymbol{x}^{(t-1)})\|_2^2. \qquad (104)
\end{aligned}
$$

H.3. Gradient estimation error

Note that the bound (100) derived for `Network-SVRG` still holds, combining it with (104) and the strong convexity of f, we have

$$\begin{aligned}\|\boldsymbol{s}^{(t)} - \nabla f(\boldsymbol{y}^{(t)})\|_2^2 \leq & 4\alpha^2 \|\boldsymbol{s}^{t-1} - \nabla f(\boldsymbol{y}^{(t-1)})\|_2^2 + 16\Big(\frac{\beta}{\sigma}\Big)^2 \|\nabla f(\boldsymbol{y}^{(t-1)})\|_2^2 \\ & + 4\Big(\frac{\beta}{\sigma}\Big)^2 \|\nabla f(\boldsymbol{x}^{(t-1)})\|_2^2 + 16L^2 \|\boldsymbol{y}^{(t-1)} - \overline{\boldsymbol{y}}^{(t-1)}\|_2^2. \qquad (105)\end{aligned}$$

H.4. Linear System

Combining (102), (103), (104), (105), we obtain the claimed linear system.

References

Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1200–1205. ACM, 2017.

Mario Arioli and Jennifer Scott. Chebyshev acceleration of iterative refinement. *Numerical Algorithms*, 66(3):591–608, 2014.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Shicong Cen, Huishuai Zhang, Yuejie Chi, Wei Chen, and Tie-Yan Liu. Convergence of distributed stochastic variance reduced methods without sampling extra data. *IEEE Transactions on Signal Processing*, 68:3976–3989, 2020.

Paolo Di Lorenzo and Gesualdo Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

Jianqing Fan, Yongyi Guo, and Kaizheng Wang. Communication-efficient accurate statistical estimation. *arXiv preprint arXiv:1906.04870*, 2019.

Robert Hannah, Yanli Liu, Daniel O'Connor, and Wotao Yin. Breaking the span assumption yields fast finite-sum minimization. In *Advances in Neural Information Processing Systems*, pages 2318–2327, 2018.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

Jakub Konečný, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Guanghui Lan, Soomin Lee, and Yi Zhou. Communication-efficient algorithms for decentralized and stochastic optimization. *Mathematical Programming*, pages 1–48, 2017.

Jason D Lee, Qihang Lin, Tengyu Ma, and Tianbao Yang. Distributed stochastic variance reduced gradient methods by sampling extra data with replacement. *The Journal of Machine Learning Research*, 18(1):4404–4446, 2017.

Zhi Li, Wei Shi, and Ming Yan. A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates. *IEEE Transactions on Signal Processing*, 67(17):4494–4506, 2019.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

Aryan Mokhtari and Alejandro Ribeiro. DSA: Decentralized double stochastic averaging gradient algorithm. *The Journal of Machine Learning Research*, 17(1):2165–2199, 2016.

Angelia Nedic, Asuman Ozdaglar, and Pablo A Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4): 922–938, 2010.

Angelia Nedić, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pages 2613–2621, 2017.

Guannan Qu and Na Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2018.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

Sashank J Reddi, Jakub Konečnỳ, Peter Richtárik, Barnabás Póczós, and Alex Smola. Aide: fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *International Conference on Machine Learning*, pages 3027–3036, 2017.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pages 2740–2749, 2018.

Gesualdo Scutari and Ying Sun. Distributed nonconvex constrained optimization over time-varying digraphs. *Mathematical Programming*, 176(1-2):497–544, 2019.

Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate Newton-type method. In *International conference on machine learning*, pages 1000–1008, 2014.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. EXTRA: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015a.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. A proximal gradient algorithm for decentralized composite optimization. *IEEE Transactions on Signal Processing*, 63(22):6013–6023, 2015b.

Virginia Smith, Simone Forte, Chenxin Ma, Martin Takáč, Michael I Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *Journal of Machine Learning Research*, 18:230, 2018.

Haoran Sun, Songtao Lu, and Mingyi Hong. Improving the sample and communication complexity for decentralized non-convex optimization: A joint gradient estimation and tracking approach. *arXiv preprint arXiv:1910.05857*, 2019a.

Ying Sun, Amir Daneshmand, and Gesualdo Scutari. Convergence rate of distributed optimization algorithms based on gradient tracking. *arXiv preprint arXiv:1905.02637*, 2019b.

César A Uribe, Soomin Lee, Alexander Gasnikov, and Angelia Nedić. Optimal algorithms for distributed optimization. *arXiv preprint arXiv:1712.00232*, 2017.

Hoi-To Wai, Zhuoran Yang, Princeton Zhaoran Wang, and Mingyi Hong. Multi-agent reinforcement learning via double averaging primal-dual optimization. In *Advances in Neural Information Processing Systems*, pages 9649–9660, 2018.

Shusen Wang, Farbod Roosta-Khorasani, Peng Xu, and Michael W Mahoney. Giant: Globally improved approximate newton method for distributed optimization. In *Advances in Neural Information Processing Systems*, pages 2338–2348, 2018.

Chenguang Xi, Ran Xin, and Usman A Khan. ADD-OPT: Accelerated distributed directed optimization. *IEEE Transactions on Automatic Control*, 63(5):1329–1339, 2017.

Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53(1):65–78, 2004. ISSN 01676911. doi: 10.1016/j.sysconle.2004.02.022.

Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Ran Xin, Usman A Khan, and Soummya Kar. Variance-reduced decentralized stochastic optimization with gradient tracking. *arXiv preprint arXiv:1909.11774*, 2019a.

Ran Xin, Anit Kumar Sahu, Usman A Khan, and Soummya Kar. Distributed stochastic optimization with gradient tracking over strongly-connected networks. *arXiv preprint arXiv:1903.07266*, 2019b.

Kun Yuan, Bicheng Ying, Jiageng Liu, and Ali H Sayed. Variance-reduced stochastic learning by networked agents under random reshuffling. *IEEE Transactions on Signal Processing*, 67(2):351–366, 2018a.

Kun Yuan, Bicheng Ying, Xiaochuan Zhao, and Ali H Sayed. Exact diffusion for distributed optimization and learning – part I: Algorithm development. *IEEE Transactions on Signal Processing*, 67(3):708–723, 2018b.

Yuchen Zhang, Martin J Wainwright, and John C Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.

Minghui Zhu and Sonia Martínez. Discrete-time dynamic average consensus. *Automatica*, 46(2):322–329, 2010.