Linearized ADMM Converges to Second-Order Stationary Points for Non-Convex Problems

Songtao Lu, *Member, IEEE*, Jason D. Lee, Meisam Razaviyayn, and Mingyi Hong, *Senior Member, IEEE*

***Abstract*—In this work, a gradient-based primal-dual method of multipliers is proposed for solving a class of linearly constrained non-convex problems. We show that with random initialization of the primal and dual variables, the algorithm is able to compute second-order stationary points (SOSPs) with probability one. Further, we present applications of the proposed method in popular signal processing and machine learning problems such as decentralized matrix factorization and decentralized training of overparameterized neural networks. One of the key steps in the analysis is to construct a new loss function for these problems such that the required convergence conditions (especially the gradient Lipschitz conditions) can be satisfied without changing the global optimal points.**

***Index Terms*—First-order stationary points (FOSPs), second-order stationary points (SOSPs), alternating direction method of multipliers (ADMM), non-convex optimization, neural networks.**

I. Introduction

In this work, we consider the following linearly constrained non-convex problem,

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^{d_{\mathbf{x}}},\mathbf{y}\in\mathbb{R}^{d_{\mathbf{y}}}}{\text{minimize}} \quad & f(\mathbf{x})+g(\mathbf{y}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}=\mathbf{c} \end{aligned} \tag{1}$$

where $f(\mathbf{x}):\mathbb{R}^{d_{\mathbf{x}}}\to\mathbb{R}$ and $g(\mathbf{y}):\mathbb{R}^{d_{\mathbf{y}}}\to\mathbb{R}$ are smooth non-convex functions, $\mathbf{A}\in\mathbb{R}^{m\times d_{\mathbf{x}}}$, $\mathbf{B}\in\mathbb{R}^{m\times d_{\mathbf{y}}}$ are matrices and $\mathbf{c}\in\mathbb{R}^m$ is a vector.

An important application of problem (1) is in the non-convex distributed optimization setting – a problem that has gained considerable attention recently, and has found applications in training neural networks [2]–[4], distributed information processing and machine learning [5], [6], decentralized matrix factorization [7], and distributed signal processing [8]–[12].

In distributed optimization and learning, the common setup is that a network of N distributed agents collectively optimize the following problem

$$\underset{x\in\mathbb{R}}{\text{minimize}} \quad \sum_{i=1}^{N} f_i(x)+g(x), \tag{2}$$

where $f_i(x):\mathbb{R}\to\mathbb{R}$ is the local cost function of agent i. Here, for notational simplicity we assume that x is a scalar, i.e., $d_{\mathbf{x}}=N$. We also assume $g(x)$ represents a smooth regularization function known to all agents.

Let us briefly discuss two common settings related to problem (2). First, suppose that there are N agents in the system, and the agents are connected by a network defined by an *undirected* graph $\mathcal{G}=\{\mathcal{V},\mathcal{E}\}$, with $|\mathcal{V}|=N$ vertices and $|\mathcal{E}|=E$ edges. Each agent can only communicate with its immediate neighbors, and it can access one component function f_i. Define the node-edge incidence matrix $\mathbf{A}\in\mathbb{R}^{E\times N}$ as following: if $e\in\mathcal{E}$ and it connects vertex i and j with $i>j$, then $\mathbf{A}_{ev}=1$ if $v=i$, $\mathbf{A}_{ev}=-1$ if $v=j$ and $\mathbf{A}_{ev}=0$ otherwise. Introduce N local variables $\mathbf{x}=[x_1,\ldots,x_N]^T\in\mathbb{R}^N$, and suppose the graph $\{\mathcal{V},\mathcal{E}\}$ is connected. Then as long as as the graph is connected, the following formulation is equivalent to the global consensus problem, which is precisely problem (1), i.e.,

$$\begin{aligned} \underset{\mathbf{x}\in\mathbb{R}^N}{\text{minimize}} \quad & f(\mathbf{x})\triangleq\sum_{i=1}^{N}\left(f_i(x_i)+\frac{1}{N}g(x_i)\right), \\ \text{subject to} \quad & \mathbf{A}\mathbf{x}=0. \end{aligned} \tag{3}$$

Second, suppose that there is a central controller that the distributed agents can communicate with, but the agents are not able to directly communicate among themselves (without the help of the central controller). This setting, which is a special case of problem (3), appears in applications such as parallel computing; see [13], [14]. In this case problem (2) can be equivalently formulated into the following global consensus problem

$$\begin{aligned} \underset{\{x_i\}_{i=0}^{N}}{\text{minimize}} \quad & \sum_{i=1}^{N} f_i(x_i)+g(x_0), \\ \text{subject to} \quad & x_i=x_0,\ \forall\, i, \end{aligned} \tag{4}$$

where $\{x_i\}$ are copies of the global variable x_0; see [6], [15].

In this work, we develop an algorithm that can compute second-order stationary points (SOSPs) for problem (2). We also adapt the proposed algorithm to decentralized matrix factorization and decentralized training problems of certain machine

Manuscript received July 8, 2020; revised April 13, 2021; accepted May 25, 2021. Date of publication August 2, 2021; date of current version September 3, 2021. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. V. Gripon. The work of Minggyi Hong was supported in part by the National Science Foundation under Grants CIF-1910385 and CNS-2003033, and in part by AFOSR under Grant 19RT0424. This paper was presented in part at the International Conference on Machine Learning, Stockholm, Sweden [1]. *(Corresponding author: Mingyi Hong.)*

Songtao Lu is with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: songtao@ibm.com).

Jason D. Lee is with the Department of Electrical Engineering, Princeton University, NJ 08540 USA (e-mail: jasonlee@princeton.edu).

Meisam Razaviyayn is with the Department of Industrial and Systems Engineering, University of Southern California, CA 90089 USA (e-mail: razaviya@usc.edu).

Mingyi Hong is with the Department of Electrical and Computer Engineering, University of Minnesota Twin Cities, Minneapolis, MN 55455 USA (e-mail: mhong@umn.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSP.2021.3100976, provided by the authors.

Digital Object Identifier 10.1109/TSP.2021.3100976

learning models. We demonstrate that with proper modification, the algorithm can be used to compute global optimal solutions for many of these problems.

Algorithms that can find SOSPs or escape from strict saddle points – those stationary points that have negative eigenvalues – have wide applications in machine learing problems. Many recent works have analyzed the saddle points in machine learning problems [16]. Such as learning in shallow networks, the stationary points are either global minimum points or strict saddle points. In two-layer porcupine neural networks (PNNs), it has been shown that most local optima of PNN optimizations are also global optimizers [17]. Previous work in [18] has shown that the saddle points in tensor decomposition are indeed strict saddle points. Also, similar arguments have appeared in dictionary learning and phase retrieval problems, where it is verified that any saddle points are strict both theoretically and numerically [19]. More recently, authors in [20] propose a unified analysis of saddle points for a broad class of low-rank matrix factorization problems, including matrix factorization/sensing/completion and robust principal component analysis (PCA), and they proved that these saddle points are strict. These facts provide strong motivation to develop efficient algorithms that are able to escape from saddle points, leading to the iterates generated by these algorithms with convergence to the globally optimal solutions of the above mentioned machine learning problems.

A. Related Work

Many recent works have been focused on the performance analysis and/or design of algorithms with convergence guarantees to SOSPs for non-convex optimization problems. These include the trust region method [21], cubic regularized Newton's method [22], [23], and a mixed approach of the first-order and second-order methods [24], etc. However, these algorithms typically require second-order information, therefore they incur high computational complexity when the problem dimension becomes large. There is also a line of work analyzing the deterministic gradient descent (GD) type method. With random initializations, it has been shown that GD only converges to SOSPs for unconstrained smooth problems [25]. When manifold constraints are present, it is shown in [26] that manifold GD converges to SOSPs, provided that each time the iterates are always feasible (which is ensured by performing appropriate projection operations). Adding some noise occasionally to the iterates of the algorithm is an efficient way of finding the negative curvature. Perturbed GD (PGD) and alternating PGD have been proposed with convergence guarantees to SOSPs [27], [28], but they require the assumption on the Lipschitz Hessian continuity of the loss function and are only applicable to unconstrained problems.

As a special case of problem (1), the consensus problem has received much attention lately [13], [29]–[33]. When the objective functions are all convex, primal methods such as distributed gradient/subgradient descent (DGD) [31], [34], the EXTRA method [32], in-Network succEssive conveX approximaTion (NEXT) [10], [35], as well as primal-dual based methods such as alternating direction method of multipliers (ADMM) have been proposed [15]. There are also many recent works developing algorithms without the assumption of convexity of the objective function; see a recent survey [36]. It is worth noting that most of the algorithms surveyed in [36] can only guarantee to compute the first-order stationary points (FOSPs). More recently, a number of works such as [7], [11], [37]–[39] and the conference version of this work [1], have shown that algorithms such as (continuous-time) DGD and primal-dual can also converge to SOSPs. However, these works typically pose strong assumptions, such as a certain symmetric property of the objective function [7], or global Lipschitz gradient assumptions, therefore they cannot be directly applied to problems such as those mentioned in Sec. I. If the stronger assumptions, i.e., global Hessian Lipschitz continuity and bounded gradient disagreement, further hold, reference [40] gives the convergence rate of the stochastic DGD to a neighborhood of SOSPs.

B. Scope of This Work

The main contributions of this work are highlighted as follows.

- First, we show that the *linearized* alternating direction method of multipliers can converge to SOSPs of problem (1) by using constant step-sizes with probability one, under the global Lipschitz gradient assumption.
- Second, we apply the *linearized* ADMM algorithm to the decentralized training and matrix factorization problems, whose objective functions are both high-order polynomials that do not have global Lipschitz gradients. The key contribution here is to develop an alternative problem that is "friendly" to the *linearized* ADMM algorithm, in the sense that they possess global Lipschitz gradients while sharing the same set of the optimal solutions as the original ones.

We note that the conference version of this work [1] is, to our best knowledge, the first work that demonstrates that a decentralized algorithm is capable of converging to SOSPs. However, it has been mainly focused on theoretical analysis, without putting emphasis on concrete applications. This journal version improves upon [1] in the following aspects: 1) we unify the results in our conference paper by using a general two-block linearly constrained problem; 2) We show that it is not straightforward to apply the proposed algorithm to concrete signal processing and machine learning problems such as decentralized matrix factorization and neural network training, therefore a new technique based on spline interpolation is developed; 3) we provide thorough numerical experiments showcasing the performance of the proposed algorithm while comparing with the state-of-the-art.

Notation: bold symbols, e.g., $\mathbf{x}, \mathbf{y}$ represent vectors and plain ones represent scalars. Capital bold symbols, e.g., $\mathbf{A}, \mathbf{B}$, denote the matrices.

II. Linearized Alternating Direction Method of Multipliers

In this section, we propose an algorithm belonging to the class of method called an alternating direction method of

multipliers (ADMM). The algorithm leverages the block structure of problem (1), which updates the primal and dual variables in an alternating way. Although the main idea of the convergence analysis of the algorithm to FOSPs is similar to the existing works, the convergence to SOSPs significantly complicates the analysis.

For problem (1), the FOSPs satisfy the condition (5a) below, and the SOSPs satisfy both (5a) and (5b) below [41, Proposition 3.1.1]

$$\nabla_{\mathbf{x}} f(\mathbf{x}^*) + \mathbf{A}^T\boldsymbol{\lambda}^* = 0, \quad \nabla_{\mathbf{y}} g(\mathbf{y}^*) + \mathbf{B}^T\boldsymbol{\lambda}^* = 0,$$
$$\mathbf{A}\mathbf{x}^* + \mathbf{B}\mathbf{y}^* = \mathbf{c} \tag{5a}$$

$$\mathbf{z}^T \begin{bmatrix} \nabla^2_{\mathbf{xx}} f(\mathbf{x}^*) & 0 \\ 0 & \nabla^2_{\mathbf{yy}} g(\mathbf{y}^*) \end{bmatrix} \mathbf{z} \succeq 0, \quad \forall \mathbf{z} \in \mathcal{Z}, \tag{5b}$$

where we have defined:

$$\mathcal{Z} \triangleq \left\{ \mathbf{z} \in \mathbb{R}^d \mid \begin{bmatrix} \mathbf{A}^T\mathbf{A} & \mathbf{A}^T\mathbf{B} \\ \mathbf{B}^T\mathbf{A} & \mathbf{B}^T\mathbf{B} \end{bmatrix} \mathbf{z} = 0 \right\}, \quad d = d_{\mathbf{x}} + d_{\mathbf{y}}, \tag{6}$$

and $\boldsymbol{\lambda}^*$ denotes the dual variable.

We will refer to solutions satisfy (5a) as FOSPs, and those that satisfy both (5a) and (5b) as SOSPs. Therefore, a strict saddle point is defined as a point $(\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}^*)$ that satisfies the following conditions

$$\nabla_{\mathbf{x}} f(\mathbf{x}^*) + \mathbf{A}^T\boldsymbol{\lambda}^* = 0, \quad \nabla_{\mathbf{y}} g(\mathbf{y}^*) + \mathbf{B}^T\boldsymbol{\lambda}^* = 0,$$
$$\mathbf{A}\mathbf{x}^* + \mathbf{B}\mathbf{y}^* = \mathbf{c}, \tag{7a}$$

$$\mathbf{z}^T \begin{bmatrix} \nabla^2_{\mathbf{xx}} f(\mathbf{x}^*) & 0 \\ 0 & \nabla^2_{\mathbf{yy}} g(\mathbf{y}^*) \end{bmatrix} \mathbf{z} \leq -\sigma\|\mathbf{z}\|^2,$$
$$\text{for some } \sigma > 0, \mathbf{z} \in \mathcal{Z}. \tag{7b}$$

Here, it is obvious that this strict saddle point is an FOSP but not an SOSP.

A. Algorithm Description

Define the augmented Lagrangian (AL) function as

$$\mathcal{L}(\mathbf{x}, \mathbf{y}; \boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{y}) + \langle \boldsymbol{\lambda}, \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c} \rangle + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}\|^2 \tag{8}$$

where $\rho > 0$ is the penalty parameter.

Then, the *linearized* ADMM is given below, where $\beta > 0$ denotes the regularization parameter.

We note that in the *linearized* ADMM, the $\mathbf{x}$ and $\mathbf{y}$ steps perform gradient steps to optimize the AL, instead of performing the exact minimization as the original convex version of ADMM does [15], [42]. The reason is that the direct minimization may not be possible because the non-convexity of $f(\mathbf{x})$ and $g(\mathbf{y})$ makes the subproblem of minimizing the AL w.r.t. $\mathbf{x}$ and $\mathbf{y}$ also non-convex. Note that the gradient steps have been used in the primal updates of ADMM when dealing with convex problems, see [43], but their analyses do not extend to the non-convex setting.

Algorithm 1: Linearized ADMM.

At iteration 0, initialize $\boldsymbol{\lambda}^0$ and $\mathbf{x}^0$.

At each iteration $r+1$, update variables by:

$$\mathbf{x}^{r+1} = \arg\min_{\mathbf{x}} \langle \nabla_{\mathbf{x}} f(\mathbf{x}^r) + \mathbf{A}^T\boldsymbol{\lambda}^r, \mathbf{x} - \mathbf{x}^r \rangle + \rho \langle \mathbf{A}^T(\mathbf{A}\mathbf{x}^r + \mathbf{B}\mathbf{y}^r - \mathbf{c}), \mathbf{x} - \mathbf{x}^r \rangle + \frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^r\|^2 \tag{9a}$$

$$\mathbf{y}^{r+1} = \arg\min_{\mathbf{y}} \langle \nabla_{\mathbf{y}} g(\mathbf{y}^r) + \mathbf{B}^T\boldsymbol{\lambda}^r, \mathbf{y} - \mathbf{y}^r \rangle + \rho \langle \mathbf{B}^T(\mathbf{A}\mathbf{x}^{r+1} + \mathbf{B}\mathbf{y}^r - \mathbf{c}), \mathbf{y} - \mathbf{y}^r \rangle + \frac{\beta}{2}\|\mathbf{y} - \mathbf{y}^r\|^2 \tag{9b}$$

$$\boldsymbol{\lambda}^{r+1} = \boldsymbol{\lambda}^r + \rho\left(\mathbf{A}\mathbf{x}^{r+1} + \mathbf{B}\mathbf{y}^{r+1} - \mathbf{c}\right). \tag{9c}$$

We also note that there are quite a few recent works applying ADMM-type method to solve a number of non-convex problems; see, e.g., [44]–[46] and the references therein. However, to the best of our knowledge, these algorithms do not take exactly the same form as Algorithm 1 described above, despite the fact that their analyses all appear to be quite similar (i.e., some potential function based on the AL is shown to be descending at each iteration of the algorithm). In particular, in [46], both the $\mathbf{x}$ and $\mathbf{y}$ subproblems are solved using a proximal point method; In [47], the $\mathbf{x}$-step is solved using the gradient step, while the $\mathbf{y}$-step is solved using the conventional exact minimization. To the best of our knowledge, none of these works analyzed the convergence of these methods to SOSPs.

B. Main Assumptions

First we make the following assumptions.

- A1. Functions $f(\mathbf{x})$ and $g(\mathbf{y})$ are twice continuously differentiable and both have Lipschitz continuous gradients with respect to $\mathbf{x}, \mathbf{y}$, with constants $L_{\mathbf{x}}$, $L_{\mathbf{y}}$.
- A2. Functions $f(\mathbf{x})$ and $g(\mathbf{y})$ are lower bounded over $\mathbb{R}^{d_{\mathbf{x}}}$ and $\mathbb{R}^{d_{\mathbf{y}}}$. Without loss of generality, assume $f(\mathbf{x}) \geq 0$ and $g(\mathbf{y}) \geq 0$.
- A3. Constraint $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{c}$ is feasible over $\mathbf{x} \in \text{dom}\,(f)$ and $\mathbf{y} \in \text{dom}\,(g)$; the matrix $[\mathbf{A}\ \mathbf{B}] \in \mathbb{R}^{m \times (d_{\mathbf{x}} + d_{\mathbf{y}})}$ is *not* full column rank (so no trivial solutions exist).
- A4. Function $f(\mathbf{x}) + g(\mathbf{y}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}\|^2$ is a coercive function.
- A5. Function $f(\mathbf{x}) + g(\mathbf{y})$ is a Kurdyka-Łojasiewicz (KŁ) function. That is from [44], [48], at $\widehat{\mathbf{x}} \in \mathbb{R}^d$ if there exist $\eta \in (0\ \infty]$, a neighborhood $\mathcal{V}$ of $\widehat{\mathbf{x}}$ and a continuous concave function $\phi : [0,\ \eta) \to \mathbb{R}_+$ such that: 1) $\phi(0) = 0$ and ϕ is continuously differentiable on $[0,\ \eta]$ with positive derivatives; 2) for all $\mathbf{x} \in \mathbb{R}^d$, satisfying $h(\widehat{\mathbf{x}}) < h(\mathbf{x}) < h(\widehat{\mathbf{x}}) + \eta$, it holds that

$$\phi'(h(\mathbf{x}) - h(\widehat{\mathbf{x}}))\text{dist}(0, \partial h(\mathbf{x})) \geq 1, \tag{10}$$

where $\partial h(\mathbf{x})$ is the limiting subdifferential defined as

$$\partial h(\mathbf{x}) = \Big\{ \{\mathbf{v} \in \mathbb{R}^d : \exists \mathbf{x}^r \to \mathbf{x}, \mathbf{v}^r \to \mathbf{v}, \text{ with } \lim\inf_{\mathbf{z}\to\mathbf{x}^r} \frac{h(\mathbf{x}) - h(\mathbf{x}^r) - \langle \mathbf{v}^r, \mathbf{z} - \mathbf{x}^r\rangle}{\|\mathbf{x} - \mathbf{x}^r\|} \geq 0, \forall\, r \Big\}. \tag{11}$$

We comment that a wide class of functions enjoys the KŁ property, for example a semi-algebraic function is a KŁ function; for detailed discussions of the KŁ property we refer the readers to [44], [48] and references therein.

Below we will use $\sigma_i(\cdot)$, $\sigma_{\max}(\cdot)$, $\sigma_{\min}(\cdot)$ and $\widetilde{\sigma}_{\min}(\cdot)$ to denote the ith, the maximum, the minimum, and the smallest *non-zero* eigenvalues of a matrix, respectively.

Based on the above assumptions, the convergence of the *linearized* ADMM to FOSPs can be shown by the following similar line of arguments as in [44]–[47], [49]. However, since the exact form of this algorithm has not appeared before, for completeness we provide the proof outline in the appendix.

C. Convergence of Linearized ADMM to FOSPs

We have the following result about the convergence of the *linearized* ADMM to the FOSPs as defined in (5). The proof of the result is rather standard, and we include it in appendix Section A for completeness.

Theorem 1: Suppose Assumptions [A1] – [A5] are satisfied. For appropriate choices of β, ρ (shown in (68) for the precise expression), and starting from any point $(\mathbf{x}^0, \mathbf{y}^0, \boldsymbol{\lambda}^0)$, the linearized ADMM converges to the set of FOSPs of problem (1). Further, if $\mathcal{L}(\mathbf{x}^{r+1}, \mathbf{y}^{r+1}, \boldsymbol{\lambda}^{r+1})$ is a KŁ function, then the linearized ADMM converges globally to a unique point $(\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}^*)$.

Remark 1: The convergence analysis steps of the *linearized* ADMM to being able to find FOSPs are similar to the one provided in [50, Theorem 1]. Algorithmically, the main difference is that the algorithms analyzed in [50] do not linearize the penalty term $\frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{c}\|^2$ and make use of the same penalty and proximal parameters instead, i.e., $\rho = \beta$. In this work, in order to show the convergence of the algorithm to SOSPs, we need to have the freedom of tuning β while fixing ρ, therefore ρ and β have to be chosen differently. However, in terms of analysis, there is no fundamental difference between these versions.

D. Convergence of Linearized ADMM to SOSPs

In this section, we show one of the main contributions of this work, which demonstrates that the *linearized* ADMM can converge to solutions beyond FOSPs. To this end, first let us rewrite the optimality conditions for the $(\mathbf{x}, \mathbf{y})$ update as:

$$\nabla_{\mathbf{x}} f(\mathbf{x}^r) + \mathbf{A}^T\boldsymbol{\lambda}^r + \rho\mathbf{A}^T(\mathbf{A}\mathbf{x}^r + \mathbf{B}\mathbf{y}^r - \mathbf{c}) + \beta(\mathbf{x}^{r+1} - \mathbf{x}^r) = 0,$$

and

$$\nabla_{\mathbf{y}} g(\mathbf{y}^r) + \mathbf{B}^T\boldsymbol{\lambda}^r + \rho\mathbf{B}^T(\mathbf{A}\mathbf{x}^{r+1} + \mathbf{B}\mathbf{y}^r - \mathbf{c}) + \beta(\mathbf{y}^{r+1} - \mathbf{y}^r) = 0.$$

These conditions combined with the update rule of the dual variable give the following compact form of the algorithm

$$\begin{bmatrix} \mathbf{x}^{r+1} \\ \mathbf{y}^{r+1} \\ \boldsymbol{\lambda}^{r+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}^r - \frac{1}{\beta}\left(\nabla_{\mathbf{x}} f(\mathbf{x}^r) + \mathbf{A}^T\boldsymbol{\lambda}^r + \rho\mathbf{A}^T(\mathbf{A}\mathbf{x}^r + \mathbf{B}\mathbf{y}^r - \mathbf{c})\right) \\ \mathbf{y}^r - \frac{1}{\beta}\left(\nabla_{\mathbf{y}} g(\mathbf{y}^r) + \mathbf{B}^T\boldsymbol{\lambda}^r + \rho\mathbf{B}^T(\mathbf{A}\mathbf{x}^{r+1} + \mathbf{B}\mathbf{y}^r - \mathbf{c})\right) \\ \boldsymbol{\lambda}^r + \rho\left(\mathbf{A}\mathbf{x}^{r+1} + \mathbf{B}\mathbf{y}^{r+1} - \mathbf{c}\right) \end{bmatrix}.$$

To compactly write the iterations in the form of a linear dynamic system, define

$$\mathbf{z}^{r+1} \triangleq [\mathbf{x}^{r+1}; \mathbf{y}^{r+1}; \boldsymbol{\lambda}^{r+1}] \in \mathbb{R}^{d+m}.$$

Next we approximate the iteration around a stationary point $\mathbf{x}^*$. Suppose that $\nabla^2_{\mathbf{xx}} f(\mathbf{x}^*) = \mathbf{H}$ and $\nabla^2_{\mathbf{yy}} g(\mathbf{y}^*) = \mathbf{G}$. Then, we can write

$$\mathbf{P}\mathbf{z}^{r+1} = (\mathbf{T} + \mathbf{E}^r)\mathbf{z}^r + \mathbf{d}^r$$

where we have defined

$$\mathbf{P} \triangleq \begin{bmatrix} \mathbf{I}_{d_{\mathbf{x}}} & \mathbf{0} & \mathbf{0} \\ \frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & \mathbf{I}_{d_{\mathbf{y}}} & \mathbf{0} \\ -\rho\mathbf{A} & -\rho\mathbf{B} & \mathbf{I}_m \end{bmatrix}, \ \mathbf{E}^r \triangleq \begin{bmatrix} \Delta^r_{\mathbf{H}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Delta^r_{\mathbf{G}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \tag{12a}$$

$$\mathbf{d}^r \triangleq \begin{bmatrix} \frac{\rho}{\beta}\mathbf{A}^T\mathbf{c} + \nabla f(\mathbf{x}^*) - \Delta^r_{\mathbf{H}}\mathbf{x}^* - \mathbf{H}\mathbf{x}^* \\ \frac{\rho}{\beta}\mathbf{B}^T\mathbf{c} + \nabla g(\mathbf{x}^*) - \Delta^r_{\mathbf{G}}\mathbf{x}^* - \mathbf{G}\mathbf{y}^* \\ -\rho\mathbf{c} \end{bmatrix}, \tag{12b}$$

$$\mathbf{T} \triangleq \begin{bmatrix} \mathbf{I}_{d_{\mathbf{x}}} - \frac{1}{\beta}\mathbf{H} - \frac{\rho}{\beta}\mathbf{A}^T\mathbf{A} & -\frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} & -\frac{1}{\beta}\mathbf{A}^T \\ \mathbf{0} & \mathbf{I}_{d_{\mathbf{y}}} - \frac{1}{\beta}\mathbf{G} + \frac{\rho}{\beta}\mathbf{B}^T\mathbf{B} & -\frac{1}{\beta}\mathbf{B}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_m \end{bmatrix} \tag{12c}$$

with the following

$$\Delta^r_{\mathbf{H}} \triangleq \int_0^1 (\nabla^2 f(\mathbf{x}^* + \theta\mathbf{d}^r_{\mathbf{x}}) - \mathbf{H})\mathbf{d}^r_{\mathbf{x}} d\theta,$$

$$\Delta^r_{\mathbf{G}} \triangleq \int_0^1 (\nabla^2 g(\mathbf{y}^* + \theta\mathbf{d}^r_{\mathbf{y}}) - \mathbf{G})\mathbf{d}^r_{\mathbf{y}} d\theta,$$

$$\text{with} \quad \mathbf{d}^r_{\mathbf{x}} \triangleq -\mathbf{x}^* + \mathbf{x}^r, \quad \mathbf{d}^r_{\mathbf{y}} \triangleq -\mathbf{y}^* + \mathbf{y}^r.$$

By noting that $\mathbf{P}$ is an invertible matrix, we conclude that the new iteration $\mathbf{z}^{r+1}$ can be expressed as

$$\mathbf{z}^{r+1} = \mathbf{P}^{-1}(\mathbf{T} + \mathbf{E}^r)\mathbf{z}^r + \mathbf{P}^{-1}\mathbf{d}^r. \tag{13}$$

Now in order to analyze the stability at a point $(\mathbf{x}^*, \mathbf{y}^*)$, we need to analyze the eigenvalues of the matrix $\mathbf{P}^{-1}\mathbf{T}$ at a stationary solution, i.e., find the scalar μ that makes $\det[\mathbf{P}^{-1}\mathbf{T} - \mu\mathbf{I}] = 0$. We note that $\mathbf{P}$ is a lower triangular matrix and $\det[\mathbf{P}] = 1$. This implies that $\det[\mathbf{P}^{-1}\mathbf{T} - \mu\mathbf{I}] = \det[\mathbf{T} - \mu\mathbf{P}]$. We have the following characterization on the determinant of $\mathbf{T} - \mu\mathbf{P}$.

Lemma 1: We have the following for $\det[\mathbf{T} - \mu\mathbf{P}]$:

1) $\det[\mathbf{T} - \mathbf{P}] = 0$, i.e., 1 is an eigenvalue of $\mathbf{P}^{-1}\mathbf{T}$.

2) Suppose that the following condition is satisfied

$$\beta > \rho\sigma_{\max}(\mathbf{A}^T\mathbf{A}) + L_{\mathbf{x}}, \quad \beta > \rho\sigma_{\max}(\mathbf{B}^T\mathbf{B}) + L_{\mathbf{y}}. \tag{14}$$

Then $\det[\mathbf{T}] \neq 0$, i.e., the matrix $\mathbf{P}^{-1}\mathbf{T}$ is invertible.

3) Define a $d \times d$ matrix

$$\mathbf{U}(\mu) = [\mathbf{U}_{11}(\mu)\ \mathbf{U}_{12}(\mu); \mathbf{U}_{12}(\mu)\ \mathbf{U}_{22}(\mu)], \tag{15}$$

with

$$\mathbf{U}_{11}(\mu) \triangleq -\mu\left(2\mathbf{I} - \frac{2\rho}{\beta}\mathbf{A}^T\mathbf{A} - \frac{1}{\beta}\mathbf{H} - \mu\mathbf{I}\right) + \mathbf{I} - \frac{\rho}{\beta}\mathbf{A}^T\mathbf{A} - \frac{1}{\beta}\mathbf{H}, \tag{16a}$$

$$\mathbf{U}_{12}(\mu) \triangleq \mu\frac{2\rho}{\beta}\mathbf{A}^T\mathbf{B} - \frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} = (2\mu - 1)\frac{\rho}{\beta}\mathbf{A}^T\mathbf{B}, \tag{16b}$$

$$\mathbf{U}_{21}(\mu) \triangleq \mu^2\frac{\rho}{\beta}\mathbf{B}^T\mathbf{A}, \tag{16c}$$

$$\mathbf{U}_{22}(\mu) \triangleq -\mu\left(2\mathbf{I} - \frac{1}{\beta}\mathbf{G} - \frac{2\rho}{\beta}\mathbf{B}^T\mathbf{B} - \mu\mathbf{I}\right) + \mathbf{I} - \frac{1}{\beta}\mathbf{G} - \frac{\rho}{\beta}\mathbf{B}^T\mathbf{B}. \tag{16d}$$

Then we have $\det[\mathbf{U}(\mu)] = \det[\mathbf{T} - \mu\mathbf{P}]$, and that for any $\delta \in \mathbb{R}_+$ the eigenvalues of $\mathbf{U}(1+\delta)$ are real and the same as those of the following symmetric matrix

$$\begin{bmatrix} \mathbf{U}_{11}(1+\delta) & (\delta+1)\sqrt{2\delta+1}\frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} \\ (\delta+1)\sqrt{2\delta+1}\frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & \mathbf{U}_{22}(1+\delta) \end{bmatrix}. \tag{17}$$

Proof: **Part 1)** By using standard determinant for block matrices, we obtain (18) shown at the bottom of this page, where we have defined the matrix $\mathbf{U}(\mu) = [\mathbf{U}_{11}(\mu)\ \mathbf{U}_{12}(\mu); \mathbf{U}_{12}(\mu)\ \mathbf{U}_{22}(\mu)] \in \mathbb{R}^{d\times d}$ in (16).

We first verify the case with $\mu = 1$. In this case, it is easy to verify that

$$\mathbf{U}(1) = \frac{\rho}{\beta}\begin{bmatrix} \mathbf{A}^T\mathbf{A} & \mathbf{A}^T\mathbf{B} \\ \mathbf{B}^T\mathbf{A} & \mathbf{B}^T\mathbf{B} \end{bmatrix} = \frac{\rho}{\beta}\begin{bmatrix} \mathbf{A}^T \\ \mathbf{B}^T \end{bmatrix}[\mathbf{A}\ \mathbf{B}]. \tag{19}$$

Based on A3, we know that $\mathbf{U}(1)$ is rank deficient, therefore, $\det[\mathbf{U}(1)] = 0$, implying that $\mu = 1$ is an eigenvalue for the matrix $\mathbf{P}^{-1}\mathbf{T}$.

Part 2) Also let $\mu = 0$, we have

$$\mathbf{U}(0) = \begin{bmatrix} \mathbf{I} - \frac{\rho}{\beta}\mathbf{A}^T\mathbf{A} - \frac{1}{\beta}\mathbf{H} & -\frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} \\ \mathbf{0} & \mathbf{I} - \frac{1}{\beta}\mathbf{G} - \frac{\rho}{\beta}\mathbf{B}^T\mathbf{B} \end{bmatrix}. \tag{20}$$

Clearly, when β satisfies conditions (14), the matrix is invertible.

Part 3) We note that $\mathbf{U}(1+\delta)$ can be written in the following form

$$\mathbf{U}(1+\delta) = \begin{bmatrix} \mathbf{U}_{11}(1+\delta) & (2\delta+1)\frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} \\ (2\delta+1+\delta^2)\frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & \mathbf{U}_{22}(1+\delta) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{U}_{11}(1+\delta) & (2\delta+1)\frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} \\ \frac{(\delta+1)^2}{2\delta+1}(2\delta+1)\frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & \mathbf{U}_{22}(1+\delta) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{I} & 0 \\ 0 & \frac{\delta+1}{\sqrt{2\delta+1}} \end{bmatrix}\mathbf{W}\begin{bmatrix} \mathbf{I} & 0 \\ 0 & \frac{\sqrt{2\delta+1}}{\delta+1} \end{bmatrix}.$$

where

$$\mathbf{W} \triangleq \begin{bmatrix} \mathbf{U}_{11}(1+\delta) & (\delta+1)\sqrt{2\delta+1}\frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} \\ (\delta+1)\sqrt{2\delta+1}\frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & \mathbf{U}_{22}(1+\delta) \end{bmatrix}. \tag{21}$$

Therefore, matrix $\mathbf{U}(1+\delta)$ is similar to $\mathbf{W}$, which directly implies that these two matrices have the same eigenvalues. By noting the fact that $\delta + 1 > 0$, and $\mathbf{U}_{11}(1+\delta)$ and $\mathbf{U}_{22}(1+\delta)$ are both symmetric matrices, we conclude that $\mathbf{U}(1+\delta)$ has real eigenvalues. ■

Based on Lemma 1, we will show that the matrix $\mathbf{P}^{-1}\mathbf{T}$ has a *real* eigenvalue $\mu = 1 + \delta$, with $\delta > 0$ being a positive number. Next, plugging $\mu = 1 + \delta$ to the expression of the $\mathbf{U}$ matrix in (16) we have

$$\mathbf{U}_{11}(1+\delta) = \delta^2\mathbf{I} + \frac{\rho}{\beta}(1+2\delta)\mathbf{A}^T\mathbf{A} + \frac{\delta}{\beta}\mathbf{H} \tag{22a}$$

$$\mathbf{U}_{21}(1+\delta) = (1+\delta)^2\frac{\rho}{\beta}\mathbf{B}^T\mathbf{A}, \tag{22b}$$

$$\mathbf{U}_{12}(1+\delta) = (1+2\delta)\frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} \tag{22c}$$

$$\mathbf{U}_{22}(1+\delta) = \delta^2\mathbf{I} + \frac{\rho}{\beta}(1+2\delta)\mathbf{B}^T\mathbf{B} + \frac{\delta}{\beta}\mathbf{G}. \tag{22d}$$

Therefore, in this case we can express $\mathbf{U}(1+\delta)$ as

$$\mathbf{U}(1+\delta) = (2\delta+1)\mathbf{U}(1) + \frac{\delta}{\beta}\begin{bmatrix} \mathbf{H} & 0 \\ 0 & \mathbf{G} \end{bmatrix} + \delta^2\begin{bmatrix} \mathbf{I} & 0 \\ \frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & \mathbf{I} \end{bmatrix}.$$

It remains to show that there exists $\delta^* > 0$ such that the determinant of the above matrix is zero. To this end, we rewrite the above expression as follows

$$\mathbf{U}(1+\delta) = \delta\left(\frac{2\delta+1}{\delta}\mathbf{U}(1) + \frac{1}{\beta}\begin{bmatrix} \mathbf{H} & 0 \\ 0 & \mathbf{G} \end{bmatrix} + \delta\begin{bmatrix} \mathbf{I} & 0 \\ \frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & \mathbf{I} \end{bmatrix}\right) \triangleq \delta\left(\mathbf{F}(\delta) + \mathbf{E}(\delta)\right) \tag{23}$$

where for notational simplicity, we have defined

$$\mathbf{F}(\delta) = \frac{(2\delta+1)}{\delta}\mathbf{U}(1) + \frac{1}{\beta}\begin{bmatrix} \mathbf{H} & 0 \\ 0 & \mathbf{G} \end{bmatrix},\ \mathbf{E}(\delta) = \delta\begin{bmatrix} \mathbf{I} & 0 \\ \frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & \mathbf{I} \end{bmatrix}. \tag{24}$$

$$\det[\mathbf{T} - \mu\mathbf{P}] = \det\begin{bmatrix} (1-\mu)\mathbf{I}_N - \frac{1}{\beta}\mathbf{H} - \frac{\rho}{\beta}\mathbf{A}^T\mathbf{A} & -\frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} & -\frac{1}{\beta}\mathbf{A}^T \\ -\mu\frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & (1-\mu)\mathbf{I}_N - \frac{1}{\beta}\mathbf{G} + \frac{\rho}{\beta}\mathbf{B}^T\mathbf{B} & -\frac{1}{\beta}\mathbf{B}^T \\ \rho\mu\mathbf{A} & \rho\mu\mathbf{B} & (1-\mu)\mathbf{I}_M \end{bmatrix}$$

$$= (1-\mu)\det\left(\begin{bmatrix} (1-\mu)\mathbf{I} - \frac{1}{\beta}\mathbf{H} - \frac{\rho}{\beta}\mathbf{A}^T\mathbf{A} & -\frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} \\ -\mu\frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & (1-\mu)\mathbf{I}_N - \frac{1}{\beta}\mathbf{G} + \frac{\rho}{\beta}\mathbf{B}^T\mathbf{B} \end{bmatrix} - \frac{1}{1-\mu}\begin{bmatrix} -\frac{\rho\mu}{\beta}\mathbf{A}^T\mathbf{A} & -\frac{\rho\mu}{\beta}\mathbf{A}^T\mathbf{B} \\ -\frac{\rho\mu}{\beta}\mathbf{B}^T\mathbf{A} & -\frac{\rho\mu}{\beta}\mathbf{B}^T\mathbf{B} \end{bmatrix}\right) \triangleq \det[\mathbf{U}(\mu)] \tag{18}$$

Note that from (7b), we know that at a strict saddle point, there exists $\mathbf{y}$ such that

$$\mathbf{U}(1)\mathbf{y} = 0, \quad \mathbf{y}^T \begin{bmatrix} \mathbf{H} & 0 \\ 0 & \mathbf{G} \end{bmatrix} \mathbf{y} \leq -\sigma \|\mathbf{y}\|^2, \tag{25}$$

which implies

$$\mathbf{y}^T \left(\gamma \mathbf{U}(1) + \begin{bmatrix} \mathbf{H} & 0 \\ 0 & \mathbf{G} \end{bmatrix} \right) \mathbf{y} \leq -\sigma \|\mathbf{y}\|^2, \ \forall\, \gamma. \tag{26}$$

This further implies that the matrix $\mathbf{F}(\delta)$ has eigenvalue no greater than $-\sigma/\beta$ for any δ.

Next we invoke a matrix perturbation result [51] to argue that the matrix $\mathbf{F}(\delta) + \mathbf{E}(\delta)$ also has negative eigenvalues as long as the parameter $\delta > 0$ is small enough.

For a given matrix $\widetilde{\mathbf{F}} = \mathbf{F} + \mathbf{E} \in \mathbb{R}^{d\times d}$, let us define the following quantity, which is referred to as the optimal matching distance between $\mathbf{F}$ and $\widetilde{\mathbf{F}}$ [see Chapter 4, Section 1, Definition 1.2 in [51]]

$$\mathrm{md}(\mathbf{F}, \widetilde{\mathbf{F}}) \triangleq \min_{\Pi} \max_{j\in[d]} |\widetilde{\sigma}_{\Pi(j)} - \sigma_j| \tag{27}$$

where Π is taken over all permutations of $[d]$, and σ_j (resp $\widetilde{\sigma}_j$) is the jth eigenvalue of $\mathbf{F}$ (resp. $\widetilde{\mathbf{F}}$). We have the following results characterizing the matching distance of two matrices $\mathbf{F}$ and $\widetilde{\mathbf{F}}$ [51]:

Lemma 2: Suppose that $\mathbf{F}$ is diagonalizable with eigenvalue decomposition $\mathbf{X}^{-1}\mathbf{F}\mathbf{X} = \mathbf{\Upsilon}$. Then the following is true

$$\mathrm{md}(\mathbf{F}, \widetilde{\mathbf{F}}) \leq (2d-1)\|\mathbf{X}\|\|\mathbf{X}^{-1}\|\|\mathbf{E}\|. \tag{28}$$

Let us apply Lemma 2 to the matrices $\mathbf{F}(\delta)$ and $\mathbf{F}(\delta) + \mathbf{E}(\delta)$. Note that

$$\|\mathbf{E}\|_2 = \delta \sigma_{\max}^{1/2} \left(\begin{bmatrix} \mathbf{I} & \frac{\rho}{\beta}\mathbf{A}^T\mathbf{B} \\ \frac{\rho}{\beta}\mathbf{B}^T\mathbf{A} & \frac{\rho^2}{\beta^2}\mathbf{B}^T\mathbf{A}\mathbf{A}^T\mathbf{B} + \mathbf{I} \end{bmatrix} \right) \triangleq \delta q \tag{29}$$

where q is a fixed number independent of δ. By applying Lemma 2, and using the fact that $\|\mathbf{X}\| = 1$, we obtain the following

$$\mathrm{md}(\mathbf{F}(\delta), \mathbf{F}(\delta) + \mathbf{E}(\delta)) \leq (2d-1)\delta q. \tag{30}$$

Clearly, we can pick $\delta = \frac{\sigma}{2q\beta(2d-1)}$, which implies that

$$\mathrm{md}(\mathbf{F}(\delta), \mathbf{F}(\delta) + \mathbf{E}(\delta)) \leq \frac{\sigma}{2\beta}. \tag{31}$$

This combined with the fact that $\mathbf{F}(\delta)$ has an eigenvalue smaller or equal to $-\sigma/\beta$ regardless of the choice of δ, and that all the eigenvalues of $\mathbf{F}(\delta) + \mathbf{E}(\delta)$ are real (cf. Lemma 1), we conclude that there exists an index $i \in [d]$ such that

$$\sigma_i(\mathbf{F}(\delta) + \mathbf{E}(\delta)) \leq -\frac{\sigma}{2\beta}. \tag{32}$$

This implies that

$$\begin{aligned} \sigma_i(\mathbf{U}(1+\delta)) &\overset{(23)}{=} \delta \sigma_i(\mathbf{F}(\delta) + \mathbf{E}(\delta)) \\ &\leq -\frac{\sigma\delta}{2\beta} = -\frac{\sigma^2}{4\beta^2(2d-1)}. \end{aligned} \tag{33}$$

In conclusion, we have the following lemma.

Lemma 3: There exists $\widehat{\delta} > 0$ and $\widetilde{\delta} > 0$ such that

$$\sigma_{\min}(\mathbf{U}(1+\widehat{\delta})) < 0, \ \sigma_i(\mathbf{U}(1+\widetilde{\delta})) > 1, \quad \forall\, i. \tag{34}$$

Proof: The first inequality comes directly form our above discussion. The second inequality is also easy to see by analyzing the eigenvalues for the symmetric matrix in (17), for large positive δ.

Using the results in Lemma 1 and Lemma 3, and using the fact that the eigenvalues for $\mathbf{U}(1+\delta)$ are continuous functions of δ, we conclude that there exists $\delta^* > 0$ such that $\det[\mathbf{U}(1+\delta^*)] = 0$. The result below summarizes the proceeding discussion.

Proposition 1: Suppose Assumptions [A1] –[A5] hold true. Let $(\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}^*)$ be a FOSP satisfying (5a), and that it is a strict saddle point satisfying (7b). Let $\sigma_i(\mathbf{P}^{-1}\mathbf{T})$ be the ith eigenvalue for matrix $\mathbf{P}^{-1}\mathbf{T}$. Then the following holds:

$$\exists\, i \in [d], \quad \text{s.t.} \ \ |\sigma_i(\mathbf{P}^{-1}\mathbf{T})| > 1. \tag{35}$$

Further, when β satisfies (14), matrix $\mathbf{P}^{-1}\mathbf{T}$ is invertible.

We have the following result for the *linearized* ADMM.

Theorem 2: Suppose that Assumptions [A1] – [A5] hold, and β, ρ are chosen according to (68). Suppose that $(\mathbf{x}^0, \mathbf{y}^0, \boldsymbol{\lambda}^0)$ are initialized randomly. Then with probability one, the iterates generated by the linearized ADMM converge to SOSPs satisfying (5a).

Proof: We utilize the stable manifold theorem [25], [52]. We will verify that conditions given in [25, Theorem 7] hold, so the dynamic system (13) is not stable around strict saddle points.

Step 1: It is clear that $\mathbf{d}^r$ is a bounded sequence. As a direct result of Theorem 1, we can show that every fixed point of iteration shown in (13) is a FOSP for problem (1), i.e., every fixed point of the mapping $\varphi(\mathbf{z})$ defined below is a FOSP of problem (1):

$$\begin{aligned} &\varphi(\mathbf{z}) \triangleq \varphi([\mathbf{z}_1; \mathbf{z}_2; \mathbf{z}_3]) \\ &= \begin{bmatrix} \mathbf{z}_1 - \frac{1}{\beta}\left(\nabla_{\mathbf{z}_1} f(\mathbf{z}_1) + \mathbf{A}^T\mathbf{z}_3 + \rho\mathbf{A}^T(\mathbf{A}\mathbf{z}_1 + \mathbf{B}\mathbf{z}_2 - \mathbf{c})\right) \\ \mathbf{z}_2 - \frac{1}{\beta}\left(\nabla_{\mathbf{z}_2} g(\mathbf{z}_2) + \mathbf{B}^T\mathbf{z}_3 + \rho\mathbf{B}^T(\mathbf{A}\mathbf{z}_1 + \mathbf{B}\mathbf{z}_2 - \mathbf{c})\right) \\ \mathbf{z}_3 + \rho\left(\mathbf{A}\mathbf{z}_1 + \mathbf{B}\mathbf{z}_2 - \mathbf{c}\right) \end{bmatrix}. \end{aligned}$$

From (13), we have that the Jacobian matrix for the mapping φ at a FOSP is given by $D\varphi(\mathbf{z}^*) = \mathbf{P}^{-1}\mathbf{T}$. When β satisfies (14), we know from Lemma 1 that $\mathbf{P}^{-1}\mathbf{T}$ is invertible, implying that mapping $\varphi(\mathbf{z}^*), \forall \mathbf{z}^*$ is diffeomorphism.

Step 2: At a strict saddle point $(\mathbf{x}^*, \mathbf{y}^*)$, consider the Jacobian matrix $D\varphi(\mathbf{z}^*)$ evaluated at $\mathbf{z}^* = (\mathbf{x}^*, \mathbf{y}^*, \boldsymbol{\lambda}^*)$. Proposition 1 implies that $D\varphi(\mathbf{z}^*) = \mathbf{P}^{-1}\mathbf{T}$ has one eigenvalue that is strictly greater than 1, and this implies that the span of the eigenvectors corresponding to the eigenvalues of magnitude less than or equal to 1 is not the full space.

Step 3: Applying A5, we can know that the convergence of the entire sequence generated by the *linearized* ADMM to a stationary point exists, i.e., $\lim_{r\to\infty} \mathbf{z}^r = \mathbf{z}^*$. Recalling the center stable manifold theorem [25, Theorem 7], we know that if there is a map that diffeomorphically deforms a neighborhood of a stationary point, then it is implied that there exists a local stable center manifold $\mathcal{W}_{loc}^{cs}$ containing this stationary point with a dimension of being equal to the number of eigenvalues of the Jacobian of this stationary point that are less than 1. Also, $\mathcal{W}_{loc}^{cs}$ contains all points that are locally forward non-escaping. Combining the previous two steps, we can conclude that with

the random initialization, the *linearized* ADMM converges to SOSPs with probability one.

Remark 2: When $\mathbf{B} \equiv 0$ and $g(\mathbf{y}) \equiv 0$, problem (1) reduces to the case with a single optimization variables, that is:

$$\underset{\mathbf{x}\in\mathbb{R}^d}{\text{minimize}} \quad f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{c}. \tag{36}$$

Then the *linearized* ADMM becomes a gradient primal-dual algorithm (GPDA) proposed in [53], which is given Algorithm 2, and the convergence result given in Theorem 2 carries over.

Algorithm 2: Gradient Primal-Dual Algorithm (GPDA).

At iteration 0, initialize $\boldsymbol{\lambda}^0$ and $\mathbf{x}^0$.

At each iteration $r+1$, update variables by:

$$\mathbf{x}^{r+1} = \arg\min_{\mathbf{x}} \left\langle \nabla_{\mathbf{x}} f(\mathbf{x}^r) + \mathbf{A}^T\boldsymbol{\lambda}^r, \mathbf{x}-\mathbf{x}^r \right\rangle + \left\langle \rho\mathbf{A}^T(\mathbf{A}\mathbf{x}^r - \mathbf{c}), \mathbf{x}-\mathbf{x}^r \right\rangle + \frac{\beta}{2}\|\mathbf{x}-\mathbf{x}^r\|^2 \tag{37a}$$

$$\boldsymbol{\lambda}^{r+1} = \boldsymbol{\lambda}^r + \rho\left(\mathbf{A}\mathbf{x}^{r+1} - \mathbf{c}\right). \tag{37b}$$

The GPDA is closely related to the classical Uzawa primal-dual method [54], which has been utilized to solve *convex* saddle point problems and linearly constrained *convex* problems [55]. It is also related to the proximal method of multipliers (Prox-MM) first developed by Rockafellar in [56], in which a proximal term has been added to the augmented Lagrangian in order to make it strongly convex in each iteration. The latter method has also been applied for example, in solving certain large-scale linear programs; see [57].

Note that GPDA can be efficiently implemented in a decentralized manner to solve the consensus optimization problems. Specifically, the optimality condition of the $\mathbf{x}$-update step (37a) is given by

$$\nabla f(\mathbf{x}^r) + \mathbf{A}^T\boldsymbol{\lambda}^r + \rho\mathbf{A}^T\mathbf{A}\mathbf{x}^r + \beta(\mathbf{x}^{r+1}-\mathbf{x}^r) = 0. \tag{38}$$

Subtracting the same equation evaluated at the previous iteration and combining (37b), we can obtain

$$\mathbf{x}^{r+1} = 2\Phi\mathbf{x}^r - \Phi\mathbf{x}^{r-1} - \frac{1}{\beta}\left(\nabla f(\mathbf{x}^r) - \nabla f(\mathbf{x}^{r-1})\right) \tag{39}$$

where $\Phi \triangleq \mathbf{I} - \frac{\rho}{\beta}\mathbf{A}^T\mathbf{A}$. Note that $\mathbf{A}^T\mathbf{A}$ is the signed Laplacian matrix, so the updates of local $x_i, \forall i$ are fully decoupled.

Algorithm 3: Decentralized Implementation of GPDA.

At iteration 0, initialize $x_i^0, \forall i$.

At each iteration $r+1$, update variables $x_i, \forall i$ by:

$$x_i^{r+1} = \sum_{j\in\mathcal{N}_i} \Phi_{ij}(2x_j^r - x_j^{r-1}) - \frac{1}{\beta}(\nabla f_i(x_i^r) - \nabla f_i(x_i^{r-1})).$$

The detailed implementation of GPDA over a graph is shown in Algorithm 3. It can be seen that at each iteration GPDA only needs one round of communication among the nodes, which is the same as DGD [34] but doubly more efficient than gradient tracking (DGT) [11] and NEXT [10].

III. Applications

In this section, we will show how to apply the *linearized* ADMM and/or GPDA for two applications – the decentralized matrix factorization and the decentralized training of over-parameterized neural networks. Both problems have certain "benign" geometry in the sense that: 1) all saddle points of the problems are strict; 2) every local optimal point is also global optimal; 3) all stationary points are within a ball of the computable radius. Many problems have been shown to have these properties, such as matrix factorization for both symmetric and asymmetric cases [58], [59], phase retrieval [19], orthogonal tensor decomposition [18], dictionary recovery [60], and training shallow overparameterized neural networks [61], [62], etc. So this part of the work will explicitly present how these "benign" geometry and theoretical guarantees of the *linearized* ADMM can be kept simultaneously in real applications, which was not introduced before in [1]. Also, compared with the existing centralized algorithms, the *linearized* ADMM offers the ability to solve these problems to the global optimal points in a distributed/decentralized manner. In the following, we give several detailed examples of this class of problems and the corresponding benign structures.

A. Decentralized Matrix Factorization and Sensing

Matrix Factorization: Consider the following symmetric matrix factorization problem

$$\underset{\mathbf{X}\in\mathbb{R}^{d\times k}}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F^2 \tag{40}$$

where $\mathbf{M} \in \mathbb{R}^{d\times d}$. When $\mathbf{M}$ is Hermitian, the optimal solution of this problem can be obtained by performing eigenvalue decomposition. By applying the variable splitting technique of the *linearized* ADMM, problem (40) can be solved alternatively in the following formulation over networks, especially in some applications where the data matrices are collected at different locations,

$$\underset{\{\mathbf{X}_i\}\in\mathbb{R}^{d\times k}}{\text{minimize}} \quad \sum_{i=1}^N \frac{1}{2}\|\mathbf{X}_i\mathbf{X}_i^T - \mathbf{M}_i\|_F^2 \tag{41a}$$

$$\text{subject to} \quad \mathbf{X}_i = \mathbf{X}_j, \quad i,j \in \mathcal{N}_i \tag{41b}$$

where $\mathbf{M} = \frac{1}{N}\sum_{i=1}^N \mathbf{M}_i$ and $\mathcal{N}_i \triangleq \{j \mid j \neq i, (i,j) \in \mathcal{E}\}$ is the set of neighbors of node i.

Matrix Sensing: Matrix sensing is a generalized problem of compressive sensing in signal processing and could be solved by naturally performing distributed/decentralized processing since the sensing process could be implemented in a distributed way. A formal description of matrix sensing is as follows,

$$\underset{\mathbf{X}\in\mathbb{R}^{d\times k}}{\text{minimize}} \frac{1}{2n}\sum_{i=1}^n (\langle \mathbf{A}_i, \mathbf{X}\mathbf{X}^T\rangle - \widehat{y}_i)^2 \tag{42}$$

where n denotes number of data samples, $\widehat{y}_i = \langle \mathbf{A}_i, \mathbf{M}\rangle$ is the observed data, and $\{\mathbf{A}_i\}$ are known sensing matrices which could be *i.i.d.* zero mean sub-Gaussian entries with variance 1. Here, $\mathbf{M}$ is a rank-k matrix that is of interest to be recovered by the outer product of optimization variable $\mathbf{X}$, i.e., $\mathbf{X}\mathbf{X}^T$. Comparing formulation (42) and (41b), it is clear that matrix

sensing can be solved by the *linearized* ADMM in a distributed way through dividing the data into different computational resources, i.e.,

$$\underset{\{\mathbf{X}_i\}\in\mathbb{R}^{d\times k}}{\text{minimize}} \quad \frac{N}{2n}\sum_{i=1}^{N}\sum_{j=1}^{\frac{n}{N}}(\langle \mathbf{A}_{i,j}, \mathbf{X}_i\mathbf{X}_i^T\rangle - \widehat{y}_{i,j})^2 \tag{43a}$$

$$\text{subject to} \quad \mathbf{X}_i = \mathbf{X}_j, \quad i,j\in\mathcal{N}_i \tag{43b}$$

where each node has n/N number of data samples (we assume n/N is an integer), $\mathbf{A}_{i,j}$ denotes the jth sensing matrix at node i, and similarly $\widehat{y}_{i,j}$ represents the jth observation data sample at node i.

Low Rank Matrix Estimation: We can also solve the following nuclear-norm regularized (rank-constrained) optimization problem in a distributed way,

$$\underset{\widehat{\mathbf{M}}\in\mathbb{R}^{n\times m}}{\text{minimize}} \quad f(\widehat{\mathbf{M}}) + \lambda\|\widehat{\mathbf{M}}\|_*, \tag{44}$$

where λ is some regularizer. Note that this problem can be solved by the singular value thresholding algorithm to the global optimal solution when function $f(\cdot)$ is convex. But the algorithm requires the eigenvalue decomposition at each iteration, which is not computationally efficient for large scale problems. Instead, by leveraging Burer-Monteiro reformulation of the nuclear norm regularization [63], we can solve problem (44) by the *linearized* ADMM in a parallel fashion, i.e.,

$$\underset{\{\mathbf{X}_i\}\in\mathbb{R}^{n\times s},\{\mathbf{Y}_i\}\in\mathbb{R}^{m\times s}}{\text{minimize}} \quad f(\mathbf{X}_i\mathbf{Y}_i^T) + \lambda\left(\|\mathbf{X}_0\|_F^2 + \|\mathbf{Y}_0\|_F^2\right) \tag{45a}$$

$$\text{subject to} \quad \mathbf{X}_i = \mathbf{X}_0, \mathbf{Y}_i = \mathbf{Y}_0 \tag{45b}$$

where $i = 0,\ldots,N$. When s is greater than the rank of $\widehat{\mathbf{M}}^*$, problem (45b) is equivalent to (44) in the sense that they have a one-to-one correspondence of the global optimal solutions, where $\widehat{\mathbf{M}}^*$ denotes the global optimal solution of (44).

Remark 3: We remark that matrix factorization (40), matrix sensing (42), and low rank estimation problem (44) have the benign global geometry structure, where the corresponding theoretical results are shown in [20], [58], [63] respectively. Beyond these problem, matrix completion, robust PCA, phase retrieval also have this benign property [19], [20]. From (4) and/or (3), it can be easily checked that these centralized problems can be solved/formulated in a decentralized version naturally without loss of the benign structure.

B. Decentralized Training Overparameterized Neural Networks

Another popular machine learning problem is training neural networks. Consider the following loss function with two layers of networks [64]

$$\underset{\mathbf{v}\in\mathbb{R}^m,\mathbf{W}\in\mathbb{R}^{m\times d}}{\text{minimize}} \quad f(\mathbf{v},\mathbf{W}) = \frac{1}{2n}\sum_{i=1}^{n}(\widehat{y}_i - \mathbf{v}^T\phi(\mathbf{W}\mathbf{x}_i))^2 \tag{46}$$

where $\mathbf{W}$ denotes weights in the hidden layer, $\mathbf{v}$ is the weight at the output layer, $\mathbf{x}\in\mathbb{R}^d$ denotes the data, $\widehat{y}_i$ represents the label of the ith sample, and ϕ stands for the activation function.

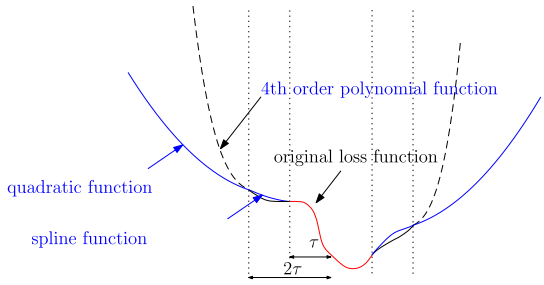


Fig. 1. An illustration of the loss function in a 2-D view.

In practice, the following regularized form of the two-layer network is desired from a perspective of improving generalization of a neural network [65]:

$$\underset{\mathbf{v}\in\mathbb{R}^m,\mathbf{W}\in\mathbb{R}^{m\times d}}{\text{minimize}} \frac{1}{2n}\sum_{i=1}^{n}(\widehat{y}_i - \mathbf{v}^T\phi(\mathbf{W}\mathbf{x}_i))^2 + \lambda(\|\mathbf{v}\|^2 + \|\mathbf{W}\|_F^2). \tag{47}$$

Recent works [62], [64] have shown that, when the neural networks are overparammeterized, both loss functions (46) and (47) have the benign property under some mild conditions.

C. Landscape of Benign Loss Functions With Global Lipschitz Continuity

Unfortunately, our analysis for the *linearized* ADMM and GPDA cannot be directly applied to the above problems. This is because the objective functions of these problems do not have a global Lipschitz gradient, i.e., Assumption A1 is not satisfied. It can be observed that the loss functions of these benign structured problems can all be viewed as a certain 4th-order polynomial with respect to the optimization variables, such as in matrix factorization/sensing (40)/(42), low rank matrix estimation (45a) when the loss function is quadratic, and distributed training (46) when the activation is quadratic. This fact motivates us to construct a *new* loss function, which shares a similar geometry as the original objective function in important regions (i.e., those regions that are known to contain global optimal solutions), while growing slowly when $\mathbf{x}$ is large. More specifically, the following new loss function $l(\sqrt{f(\mathbf{x})})$ will be considered, where loss $l(\cdot)$ is defined as

$$l(z) \triangleq \begin{cases} z^2+\alpha, & \text{if} \quad 0\le z\le\tau, \\ p(z), & \text{if} \quad \tau\le z\le 2\tau, \\ \beta z, & \text{if} \quad z\ge 2\tau, \end{cases} \tag{48}$$

with α,β,τ being some constants and $p(\cdot)$ being the spline function to be defined in (49). This construction uses spline function $p(\cdot)$ to connect the two functions $\sqrt{f(\mathbf{x})}$ and $f(\mathbf{x})$ smoothly [66, Chapter 5.1], and τ denotes the radius of the ball that contains all stationary points of the problem. In particular, the idea of constructing the new loss function for a 4th-order polynomial function is shown in Fig. 1 from a 2-dimensional view. In this section, we can show that the proposed objective function has global Lipschitz constant, while the important region (i.e., the red region in Fig. 1) is kept unchanged. As long as the algorithm can escape from the saddle points, the algorithm will converge to the global optimal solution. Below we detail the process of how to construct the new loss function $\ell(\cdot)$.

1) Construction of Spline Function: We construct the spline function $p(\cdot)$ to satisfy the properties given by Lemma 4.

Lemma 4: The following function

$$p(z) = -\frac{1}{3\tau}(z-\tau)^3 + (z-\tau)^2 + 2(z-\tau) + \frac{10}{3}\tau^2 \quad (49)$$

is a C^2 interpolation between the functions z^2 and z on the region $[\tau, 2\tau]$, which satisfies

P1 $p(\tau) = \tau^2 + \alpha$ and $p(2\tau) = 2\beta\tau$
P2 $p'(\tau) = 2\tau$ and $p'(2\tau) = \beta$
P3 $p''(\tau) = 2$ and $p''(2\tau) = 0$
P4 $p'([\tau, 2\tau]) > 0$.

where $\beta = 3\tau$ and $\alpha = \frac{7}{3}\tau^2$.

Proof: First, let us consider a third-order polynomial as the following,

$$p(z) = a(z-\tau)^3 + b(z-\tau)^2 + c(z-\tau) + d \quad (50)$$

where a, b, c, d are some constants. From P3, we can get $a = -\frac{1}{3\tau}$ and $b = 1$. Similarly, we can obtain $c = 2\tau$ and $\beta = 3\tau$ from P2. Finally, we have $d = \frac{10}{3}\tau^2$ and $\alpha = \frac{7}{3}\tau^2$ by using P1. Therefore, we have

$$p(z) = -\frac{1}{3\tau}(z-\tau)^3 + (z-\tau)^2 + 2\tau(z-\tau) + \frac{10}{3}\tau^2. \quad (51)$$

Also, we know

$$p'(z) = -\frac{1}{\tau}(z-\tau)^2 + 2z, \quad (52)$$

where the two roots of $p'(z) = 0$ are $(2 \pm \sqrt{3})\tau$, implying that $p'([\tau, 2\tau]) > 0$.

2) Choice of Parameter: Before constructing the new loss function, parameter τ needs to be chosen in advance such that there is no FOSPs when $\mathbf{x} \notin \mathcal{X}$, where $\mathcal{X} = \{\mathbf{x} | \sqrt{f(\mathbf{x})} \leq \tau\}$ denotes the set which includes the critical points. It is obvious that τ is dependent on the specific geometry of the loss function and its value is dependent on different optimization problems.

Below, we show that such a parameter can be chosen for the symmetric matrix factorization problem shown in (40).

Lemma 5: For the symmetric matrix factorization problem, i.e., $\min_{\mathbf{X}} \|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F^2$, all FOSPs are within the region $\mathcal{X} \triangleq \{\mathbf{X} | \|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F \leq 5\|\mathbf{M}\|_F\}$.

Proof: First, applying the triangle inequality, we have

$$\|\mathbf{X}\mathbf{X}^T\|_F + \|\mathbf{M}\|_F \geq \|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F. \quad (53)$$

When $\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F \geq 5\|\mathbf{M}\|_F$, it is obvious that $\|\mathbf{X}\mathbf{X}^T\| \geq 4\|\mathbf{M}\|_F$.

Second, from [58, Theorem 4], we know that when $\|\mathbf{X}\mathbf{X}^T\|_F \geq 4\|\mathbf{M}\|_F$ then $\|\nabla_{\mathbf{X}} f(\mathbf{X})\| > 3/4\sigma_{\max}^3(\mathbf{M}) > 0$, which implies that there is no FOSP of problem $\min_{\mathbf{X}} \|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F^2$ in region $\{\mathbf{X} | \|\mathbf{X}\mathbf{X}^T\|_F \geq 4\|\mathbf{M}\|_F\}$.

Combing the above two steps, we know that all the FOSPs of the symmetric matrix factorization problem must be within set $\mathcal{X}$. Therefore, according to the definition of τ, we choose $\tau \triangleq 5\|\mathbf{M}\|_F$.

Remark 4: For other problems, the size of τ can be also quantified by the above steps. The regions that include all critical points have been shown in the literature, such as phase retrieval [19, Theorem 2], asymmetric matrix factorization [67, Theorem 1].

For the problem of learning a shallow network (46), we can also quantify the radius of a ball that includes all the critical points. We follow the analysis and corresponding assumptions in [62, Theorem 2] and have the following new result of measuring the radius of the ball that includes all the critical points.

Lemma 6: Consider a shallow neural network. Input/label pairs $\{\mathbf{x}_i\} \in \mathbb{R}^d$ and $\{\widehat{y}_i\}$ for $i = 1, \ldots, n$ are generated by the form of

$$\widehat{y}_i = \langle \mathbf{v}^*, \phi(\mathbf{W}^*\mathbf{x}_i) \rangle, \quad (54)$$

where $\mathbf{x}_i \in \mathbb{R}^d$ are distributed in i.i.d. $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, $\phi(z)$ is a quadratic activation function, $\mathbf{W}^* \in \mathbb{R}^{m \times d}$ with $\sigma_{\min}(\mathbf{W}^*) > 0$ and $\mathbf{v}^* = [v_1^*, \ldots, v_m^*]^T \in \mathbb{R}^m$ denotes the weights in the hidden and output layer respectively. Further, we assume that all nonzero entries of $\mathbf{v}^*$ have the same sign (positive or negative). It is of interest to minimizing the loss function (46) by fixing $\mathbf{v}$, where we set the nonzero entries of $\mathbf{v}$ to have the same sign as $\mathbf{v}^*$ with at least d strictly nonzero entries. As long as $n \geq C(\delta, \delta')d$, when $\|\mathbf{W}\|_{\mathbf{D}_\mathbf{v}}^2 = |\mathrm{Tr}(\mathbf{W}^T\mathbf{D}_\mathbf{v}\mathbf{W})| \geq \frac{2(3+\delta)}{(1-\delta)^2(1-\delta')n} |\sum_{i=1}^n \widehat{y}_i|$ then $\|\nabla_{\mathbf{W}} f(\mathbf{W})\| > 0$ for any $\delta > 0$ and $0 < \delta' < 1$, where $\mathbf{D}_\mathbf{v} = \mathrm{diag}(v_1, \ldots, v_m)$ and $C(\delta, \delta')$ is some numerical constant depended on δ and δ'.

Proof:

Step 1: Substituting (54) into (46) and applying [62, Lemma 13], we know that for any $\delta > 0$ the following is true with high probability as long as $n \geq c(\delta)d\log(d)$

$$\langle \nabla_{\mathbf{W}} f(\mathbf{W}), \mathbf{W} \rangle \geq (1-\delta)^2 \mathrm{Tr}^2(\mathbf{W}^T\mathbf{D}_\mathbf{v}\mathbf{W}) - (3+\delta)\mathrm{Tr}((\mathbf{W}^*)^T\mathbf{D}_{\mathbf{v}^*}\mathbf{W}^*)\mathrm{Tr}(\mathbf{W}^T\mathbf{D}_\mathbf{v}\mathbf{W}) \quad (55)$$

where the detailed derivations are shown in [62, pages 753–754]. According to (55) and the sign assumptions on $\mathbf{v}^*$ and $\mathbf{v}$, it can be observed that when $|\mathrm{Tr}((\mathbf{W}^*)^T\mathbf{D}_{\mathbf{v}^*}\mathbf{W}^*)| \leq (1-\delta)^2/(2(3+\delta))|\mathrm{Tr}(\mathbf{W}^T\mathbf{D}_\mathbf{v}\mathbf{W})|$, then

$$\langle \nabla_{\mathbf{W}} f(\mathbf{W}, \mathbf{W} \rangle \geq \frac{(1-\delta)^2}{2}\mathrm{Tr}^2(\mathbf{W}^T\mathbf{D}_\mathbf{v}\mathbf{W}). \quad (56)$$

Combing the fact that $\langle \nabla_{\mathbf{W}} f(\mathbf{W}), \mathbf{W} \rangle \leq \|\nabla_{\mathbf{W}} f(\mathbf{W})\| \|\mathbf{W}\|$, we can have that when $\|\mathbf{W}\|_{\mathbf{D}_\mathbf{v}}^2 \geq \frac{2(3+\delta)}{(1-\delta)^2}\|\mathbf{W}^*\|_{\mathbf{D}_{\mathbf{v}^*}}^2 >^{(a)} 0$, then

$$\|\nabla_{\mathbf{W}} f(\mathbf{W})\| \geq \frac{(1-\delta)^2\|\mathbf{W}\|_{\mathbf{D}_\mathbf{v}}^4}{2\|\mathbf{W}\|} > 0 \quad (57)$$

where (a) is true because $\sigma_{\min}(\mathbf{W}^*) > 0$ and there is at least one entry of $\mathbf{v}$ which is strictly nonzero.

Step 2: From (54), we have

$$\frac{1}{n}\sum_{i=1}^n \widehat{y}_i = \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i^T((\mathbf{W}^*)^T\mathbf{D}_{\mathbf{v}^*}\mathbf{W}^*\mathbf{x}_i. \quad (58)$$

By standard concentration of sample covariance matrices or [62, Lemma 13], we have for any $0 < \delta' < 1$ $|\frac{1}{n}\sum_{i=1}^n \mathbf{x}_i^T(\mathbf{W}^*)^T\mathbf{D}_{\mathbf{v}^*}\mathbf{W}^*\mathbf{x}_i - \mathrm{Tr}((\mathbf{W}^*)^T\mathbf{D}_{\mathbf{v}^*}\mathbf{W}^*)| \leq \delta'|\mathrm{Tr}((\mathbf{W}^*)^T\mathbf{D}_{\mathbf{v}^*}\mathbf{W}^*)|$ when $n \geq \frac{c'}{\delta'^2}d$ with high probability where c' is a fixed numerical constant. Combining (58) and the sign assumptions on $\mathbf{v}^*$ and $\mathbf{v}$, we have

$$\frac{1}{n}\left|\sum_{i=1}^n \widehat{y}_i\right| \geq (1-\delta')\left|\mathrm{Tr}((\mathbf{W}^*)^T\mathbf{D}_{\mathbf{v}^*}\mathbf{W}^*)\right|, \quad (59)$$

meaning that $\|\mathbf{W}^*\|_{\mathbf{D}_{\mathbf{v}^*}}^2 = |\mathrm{Tr}((\mathbf{W}^*)^T\mathbf{D}_{\mathbf{v}^*}\mathbf{W}^*)| \leq \frac{1}{n(1-\delta')}|\sum_{i=1}^n \widehat{y}_i|$.

Step 3: Combining (57) in step 1 and (59) in step 2, we can conclude that when $\|\mathbf{W}\|^2_{\mathbf{D_v}} \geq \frac{2(3+\delta)}{(1-\delta)^2(1-\delta')n}|\sum_{i=1}^n \widehat{y}_i|$ we have $\|\nabla_{\mathbf{W}} f(\mathbf{W})\| > 0$ with $C(\delta, \delta')$ being chosen by $\max\{c(\delta)\log(d), c'/\delta'^2\}$.

Remark 5: When $m \gtrsim d$, the overparameterized shallow neural network also has the benign property [62, Theorem 2].

3) Global Lipschitz Continuity: From the new loss function (48), we can see that when $z \leq 2\tau$ the function is defined on a bounded region, meaning that the function is Lipschitz continuous. To show the global Lipschitz continuity, we only need to consider the case where $z > 2\tau$. Intuitively, after taking the square root of the objective function, the function in the 4-order polynomial of the optimization variable is reduced to a quadratic function. We can give the following example to show the Lipschitz constant when $\sqrt{f(\mathbf{x})}$ is large for the symmetric matrix factorization problem.

Lemma 7: When $\|\mathbf{X}\mathbf{X}^\tau\|_F \geq 2\|\mathbf{M}\|_F$, function $\sqrt{f(\mathbf{X})} \triangleq \|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F$ is Lipschitz continuous with constant $2.5 + 2\ k$, where $\mathbf{X} \in \mathbb{R}^{d\times k}$ and $\mathbf{M} \in \mathbb{R}^{d\times d}$.

Proof: The proof is elementary (i.e., checking the boundedness the second derivative of the objective function) but cumbersome. Please see Section B in the appendix.

Theorem 3: If all the FOSPs of $f(\mathbf{x})$ are in $\mathcal{X}$, where $\mathcal{X} = \{\mathbf{x}|\sqrt{f(\mathbf{x})} \leq \tau\}$ and τ is some constant, all critical points of the function $l(\sqrt{f(\mathbf{x})})$ in (48) have a one-to-one correspondence to the original loss function $f(\mathbf{x})$.

Proof: By the chain rule of the derivative, we know

$$\nabla l(f(\mathbf{x})) = l'(f(\mathbf{x}))\nabla f(\mathbf{x}), \tag{60}$$

so we have

$$\|\nabla l(f(\mathbf{x}))\| = l'(f(\mathbf{x}))\|\nabla f(\mathbf{x})\|. \tag{61}$$

Then, there are three cases:

1) When $\sqrt{f(\mathbf{x})} \leq \tau$: $l(\sqrt{f(\mathbf{x})}) = f(\mathbf{x}) + \alpha$, i.e., $\nabla_{\mathbf{x}} l(\sqrt{f(\mathbf{x})}) = \nabla_{\mathbf{x}} f(\mathbf{x})$ and $\nabla^2_{\mathbf{xx}} l(\sqrt{f(\mathbf{x})}) = \nabla^2_{\mathbf{xx}} f(\mathbf{x})$, meaning that all FOSPs and SOSPs between $l(\sqrt{f(\mathbf{x})})$ and $f(\mathbf{x})$ have a one-to-one correspondence.
2) When $\tau < \sqrt{f(\mathbf{x})} \leq 2\tau$: $l(\sqrt{f(\mathbf{x})}) = p(\sqrt{f(\mathbf{x})})$. From (61), we have $l'(\sqrt{f(\mathbf{x})}) = p'(\sqrt{f(\mathbf{x})})\nabla_{\mathbf{x}}\sqrt{f(\mathbf{x})} = 1/2p'(\sqrt{f(\mathbf{x})})\nabla_{\mathbf{x}} f(\mathbf{x})/\sqrt{f(\mathbf{x})}$. Because $p'(\sqrt{f(\mathbf{x})}) > 0$, $\sqrt{f(\mathbf{x})} > \tau > 0$, and $\nabla_{\mathbf{x}} f(\mathbf{x}) > 0$, we know that there is no FOSP in this region.
3) When $\sqrt{f(\mathbf{x})} > 2\tau$: $l(\sqrt{f(\mathbf{x})}) = \beta\sqrt{f(\mathbf{x})}$. We have $\nabla_{\mathbf{x}} l(\sqrt{f(\mathbf{x})}) = \nabla_{\mathbf{x}} 1/2\beta\nabla_{\mathbf{x}} f(\mathbf{x})/\sqrt{f(\mathbf{x})}$. Since $\sqrt{f(\mathbf{x})} > 2\tau > 0$ and $\nabla_{\mathbf{x}} f(\mathbf{x}) > 0$, we know that there is no FOSP in this region as well.

Combining the above three cases completes the proof.

Corollary 1: The loss function $l(\sqrt{f(\mathbf{x})})$ shown in (48) for the symmetric matrix factorization problem has the global Lipschitz continuity and all critical points of the function $l(\sqrt{f(\mathbf{x})})$ have a one-to-one correspondence to the original loss function $f(\mathbf{x})$.

Proof: The proof can be completed directly by combining Lemma 7, [67, Theorem 1] and Theorem 3.

Remark 6: For the finite sum problem, the loss function $f_i(\mathbf{x}_i)$ at each node can be constructed by $l(\sqrt{f_i(\mathbf{x}_i)})$ so that the whole problem $N^{-1}\sum_{i=1}^N l(\sqrt{f_i(\mathbf{x}_i)})$ still has the global gradient Lipschitz continuity. Also, τ can be approximated by several classic gossip steps through transmitting the size of averaged data samples at each node.

IV. Numerical Results

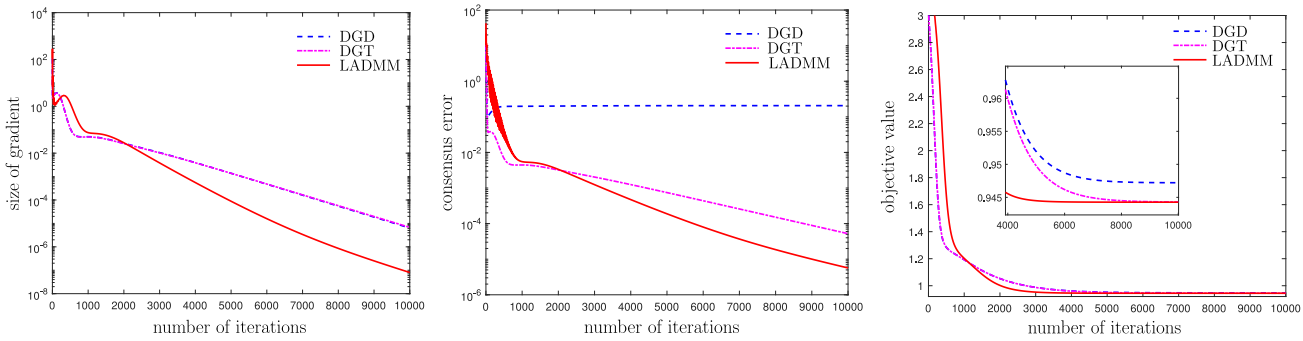
A toy example was used in [1] to show the convergence of the *linearized* ADMM (LADMM). In this section, we show the numerical results of applying LADMM or GPDA into problems of decentralized/distributed matrix factorization and overparameterized neural networking training, and compare LADMM with three other classic methods, i.e., DGD [34], DGT [11], and NEXT [10], [68]. There are a total number of 10 nodes over an Erdős–Rényi random graph generated with a connectivity of 0.5. In the results, the size of gradient refers to $\|\sum_{i=1}^n \nabla f_i(\mathbf{x}_i)\|$ and the consensus error is defined by $\sum_{i=1}^n \|\mathbf{x}_i - \bar{\mathbf{x}}\|$ where $\bar{x} \triangleq n^{-1}\sum_{i=1}^n \mathbf{x}_i$.

A. Decentralized Symmetric Matrix Factorization

First, we compare the algorithms for decentralized symmetric low-rank matrix factorization shown in Fig. 2, where $d = 10$ and $k = 3$. Each entry of the data matrices at node 1 to 5 is randomly generated, which follows the *i.i.d.* Gaussian distribution (i.e., $\mathcal{CN}(0, 1)$) while the rest of entries at node 6 to 10 follow the *i.i.d.* uniform distribution within the interval [0, 1]. The step-sizes of DGD and DGT are both 0.02 and the parameters of the *linearized* ADMM are $\rho = 1$ and $\beta = 50$, i.e., $1/\beta = 0.02$. The initial points of all the algorithms are the same and generated randomly, where each entry of $\mathbf{X}^0$ follows *i.i.d.* $\mathcal{CN}(0, 1)$. From Fig. 2, it can be observed that the *linearized* ADMM and DGT have similar convergence behaviour but DGD has a constant consensus error. We also test the algorithms for the case where $d = k = 10$. The step-sizes of DGD and DGT are both 0.04, the parameters of the *linearized* ADMM are $\rho = 3$ and $\beta = 25$, i.e., $1/\beta = 0.04$, and for NEXT the penalizer of the proximal term involved in local successive convex approximation (SCA) is 25 and step-size sequence $\{\alpha^r, r \geq 1\}$ is chosen according to Rule 2 as suggested in [10] where parameter $\mu = 0.01$ and $\alpha^0 = 0.5$. The rest of settings are the same as before and the results are shown in Fig. 3. It can be seen that the *linearized* ADMM and DGT are able to find the global optimal solution quickly (both of them show a linear convergence behaviour when the size of gradient is small) while DGD evaluated at the consensus space cannot converge to the global optimal solution. Here, NEXT shows a sublinear convergence rate behavior (when the size of gradient is small) and can also converge to the global optimal solution. It is worth noting that both DGT and NEXT need two rounds of communications at each iteration, illustrating the superiority of the *linearized* ADMM compared with DGT and NEXT in terms of the communication complexity.

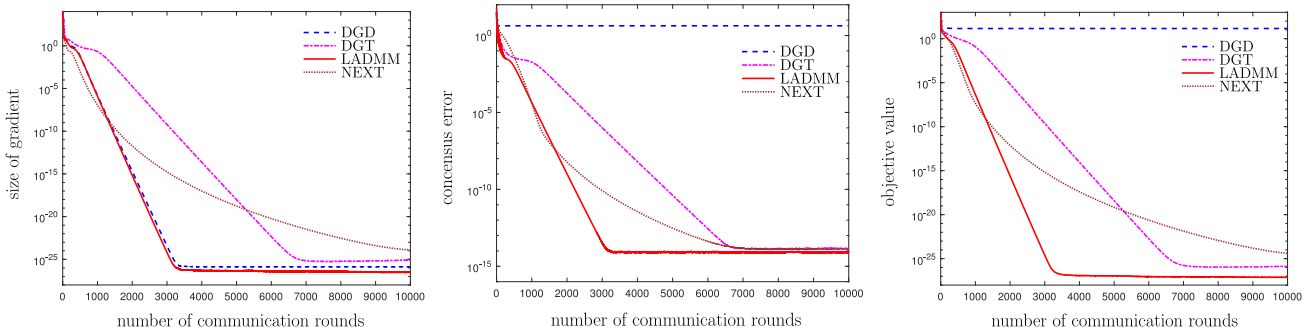
B. Testing Over Original and Constructed Loss Functions

In this section, we further test the algorithms on both original and proposed new loss functions to verify the effectiveness of the constructed objective in ensuring the convergence of the gradient-based distributed algorithms, where τ is chosen by the theoretical result shown in Lemma 5. When $\mathbf{X}^0$ is initialized



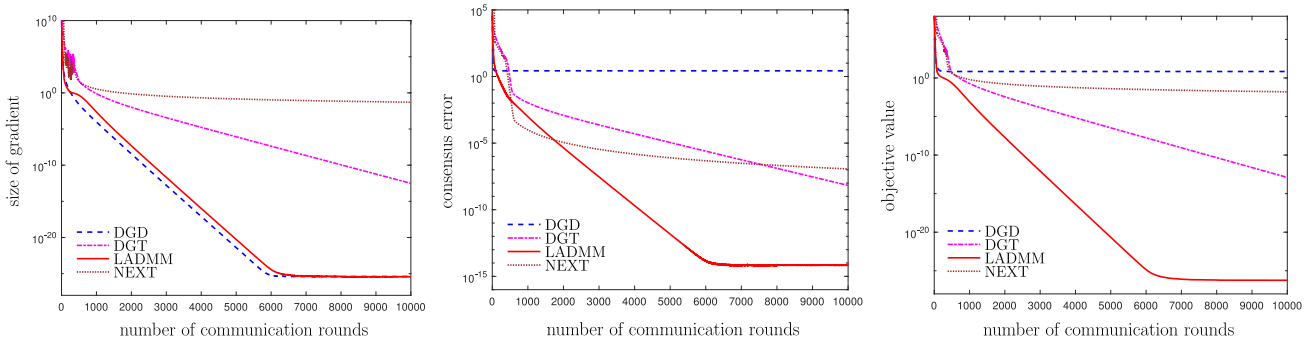
(a) Size of gradient versus # of iterations (b) Consensus error versus # of iterations (c) Objective value versus # of iterations

Fig. 2. The convergence behaviors of algorithms for low-rank matrix decomposition.



(a) Size of gradient versus # of communications (b) Consensus error versus # of communications (c) Objective value versus # of communications

Fig. 3. The convergence behaviors of the algorithms for symmetric matrix factorization.



(a) Size of gradient versus # of communications (b) Consensus error versus # of communications (c) Objective value versus # of communications

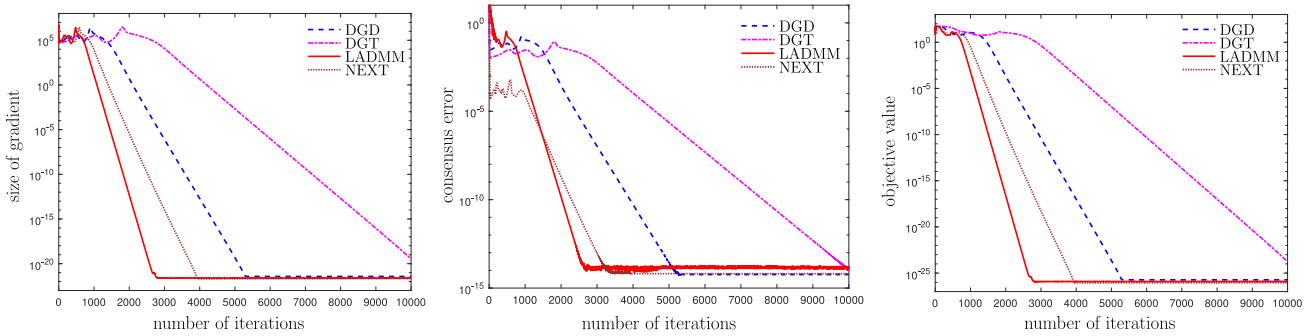
Fig. 4. The convergence behaviors of the algorithms for symmetric matrix factorization on the proposed loss function with global gradient Lipschitz continuity.

with a small size in the sense that each entry of $\mathbf{X}^0$ follows *i.i.d.* $\mathcal{CN}(0,1)$, the convergence curves of all the algorithms are the same on both loss functions. When $\mathbf{X}^0$ is initialized with a large size, i.e., each entry of $\mathbf{X}^0$ follows *i.i.d.* $\mathcal{CN}(0,1000)$, then all the algorithms either diverge or cannot even find FOSPs. However, they will converge based on the newly constructed loss function with global gradient Lipschitz continuity as shown in Fig. 4, where the step-sizes of DGD, DGT, β in the *linearized* ADMM, and the penalizer used in NEXT are 0.02, 0.012, 50, and 50, respectively. Note that these step-sizes are tuned such that the tested algorithms can converge, otherwise, if either the step-sizes are increased or equivalently the penalizer involved in the local function linearization process of NEXT (or β in the *linearized* ADMM) is decreased, the corresponding algorithm will diverge. This test shows the usefulness of the proposed loss function when the classic gradient-based methods are applied in solving the problems without global gradient Lipschitz continuity.

Next, we will show the convergence of these algorithms for training overparameterized neural networks to globally optimal solutions for *i.i.d.* Gaussian data.

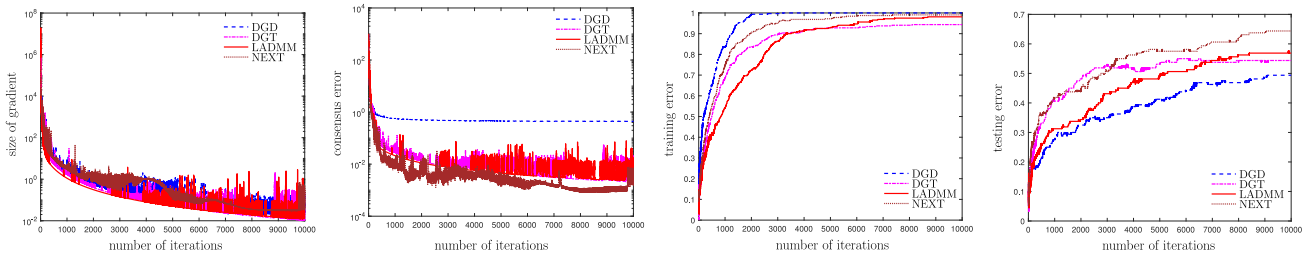
C. Decentralized Shallow Overparameterized Neural Nets Training

We compare the performance of these algorithms for training a two-layer neural network, where the number of neurons at the hidden layer is $m = 20$, and the activation is quadratic. There are 40 data samples at each node and the dimension of each data point is $d = 2$. The data and labels are both randomly generated, which follow Gaussian distribution. The step-sizes



(a) Size of gradient versus # of iterations (b) Consensus error versus # of iterations (c) Accuracy versus # of iterations

Fig. 5. The convergence behaviors of the algorithms for training an overparameterized neural net.



(a) Size of gradient v.s. iterations (b) Consensus error v.s. iterations (c) Training Accuracy v.s. iterations (d) Testing Accuracy v.s. iterations

Fig. 6. The convergence behaviors of algorithms for training a two-layer neural net on MNIST dataset.

of DGD and DGT are 1×10^{-4} and 5×10^{-5} (note that if the step-size of DGT is 1×10^{-4}, then DGT will diverge). The parameters of the *linearized* ADMM are $\rho = 80$ and $\beta = 5000$, i.e., $1/\beta = 2 \times 10^{-4}$. We also choose the step-size used in the local SCA optimization step of NEXT is 2×10^{-4}, hyperparameter $\mu = 0.001$, and the initial of step-size sequence α^r as $\alpha^0 = 0.1$. From Fig. 5 it is very interesting to see that all the algorithms converge to the global optimal solution in this case where the neural net is overparameterized and the data follow the *i.i.d* Gaussian distribution.

D. Decentralized Two-Layer Neural Nets Training With Real Dataset

We also implement the algorithms for training a two-layer neural net on the MNIST dataset [69]. The dimension of the input data is $d = 28 \times 28 = 784$ and there are 10 classes of handwritten digits from 0 to 1. We divide 640 training data samples equally to the 10 nodes and put each type of digits stored at each node, e.g., node 1 only has digit 1. In such a way, the data distribution is heterogeneous. We also divide 320 testing data samples equally to the 10 nodes randomly. The two-layer neural net has $m = 256$ neurons at the hidden layer and 10 neurons in the output layer, where the activation is sigmoid and one dimension of the bias is included. The step-sizes of DGD and DGT are both 1×10^{-2}. The parameters ρ and $1/\beta$ of the *linearized* ADMM are chosen as 8 and 1×10^{-2}, respectively. For NEXT, the step-size used in the local SCA is 1×10^{-2}, hyperparameter $\mu = 0.01$, and the initial of step-size sequence α^r is $\alpha^0 = 0.1$. It can be seen in Fig. 6 that DGD at each node can find a good solution but since there is almost a constant consensus error among the nodes, generalization performance achieved by DGD is very low. We can also observe that the *linearized* ADMM, DGT and NEXT achieve reasonable good training accuracy but there is some subtle difference in the testing accuracy.

V. Conclusion

The main contribution of this work is to show that primal-dual based first-order methods are capable of converging to second-order stationary points, for linearly constrained non-convex problems. The main techniques that we have leveraged are the stable manifold theorem and its recently developed connection to first-order optimization methods. One important implication of our result is that, properly designed distributed non-convex optimization methods (for both the global consensus problem and the distributed optimization problem over a multi-agent network) can also converge to SOSPs. To the best of our knowledge, this is the first algorithm for general linear constrained non-convex optimization problems with provable convergence guarantees to compute SOSPs. Numerical results also show that the proposed algorithm works well and can find the global optimal solutions for a class of non-convex learning problems.

Appendix

A. Proof of Theorem 1

Proof: we provide an outline of the proof. First, we construct an upper bound of $\|\boldsymbol{\lambda}^{r+1} - \boldsymbol{\lambda}^r\|$. Let us define

$$\mathbf{C}_1 \triangleq \beta \mathbf{I} - \rho \mathbf{A}^T \mathbf{A} \succ 0, \quad \mathbf{C}_2 \triangleq \beta \mathbf{I} - \rho \mathbf{B}^T \mathbf{B} \succ 0,$$

$$\mathbf{w}^{r+1} \triangleq (\mathbf{x}^{r+1} - \mathbf{x}^r) - (\mathbf{x}^r - \mathbf{x}^{r-1}),$$

$$\mathbf{v}^{r+1} \triangleq (\mathbf{y}^{r+1} - \mathbf{y}^r) - (\mathbf{y}^r - \mathbf{y}^{r-1}), \quad \mathbf{z}^{r+1} = [\mathbf{w}^{r+1}; \mathbf{v}^{r+1}],$$

$$\mathbf{W} \triangleq \begin{bmatrix} \mathbf{A}^T\mathbf{A} & \mathbf{A}^T\mathbf{B} \\ \mathbf{0} & \mathbf{B}^T\mathbf{B} \end{bmatrix}, \quad \mathbf{V} \triangleq [\mathbf{A}, \mathbf{B}], \quad \mathbf{C} \triangleq \beta\mathbf{I} - \rho\mathbf{W}.$$

From the optimality conditions of (9a) and (9b), we have

$$\begin{aligned} &\nabla_{\mathbf{x}} f(\mathbf{x}^r) + \mathbf{A}^T\boldsymbol{\lambda}^r + \rho\mathbf{A}^T(\mathbf{A}\mathbf{x}^{r+1} + \mathbf{B}\mathbf{y}^{r+1} - \mathbf{c}) \\ &+ \rho\mathbf{A}^T\mathbf{B}(\mathbf{y}^r - \mathbf{y}^{r+1}) + \rho\mathbf{A}^T\mathbf{A}(\mathbf{x}^r - \mathbf{x}^{r+1}) \\ &+ \beta(\mathbf{x}^{r+1} - \mathbf{x}^r) = 0, \end{aligned}$$

and

$$\begin{aligned} &\nabla_{\mathbf{y}} g(\mathbf{y}^r) + \mathbf{B}^T\boldsymbol{\lambda}^r + \rho\mathbf{B}^T(\mathbf{A}\mathbf{x}^{r+1} + \mathbf{B}\mathbf{y}^{r+1} - \mathbf{c}) \\ &+ \rho\mathbf{B}^T\mathbf{B}(\mathbf{y}^r - \mathbf{y}^{r+1}) + \beta(\mathbf{y}^{r+1} - \mathbf{y}^r) = 0. \end{aligned} \tag{62}$$

Using (9c) and (62) and the fact that $\boldsymbol{\lambda}^{r+1} - \boldsymbol{\lambda}^r$ lies in the column space of $[\mathbf{A}, \mathbf{B}]$, after some simple manipulation, we can show

$$\begin{aligned} \frac{1}{\rho}\|\boldsymbol{\lambda}^{r+1} - \boldsymbol{\lambda}^r\|^2 \le{}& \frac{2L_{\mathbf{x}}^2\|\mathbf{x}^r - \mathbf{x}^{r-1}\|^2 + 2L_{\mathbf{y}}^2\|\mathbf{y}^r - \mathbf{y}^{r-1}\|^2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})} \\ &+ \frac{2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})}\|\mathbf{z}^{r+1}\|^2_{\mathbf{C}^T\mathbf{C}}. \end{aligned} \tag{63}$$

Next, we construct an upper bound for the sum of squares of primal and dual variables. From the optimality conditions of (9a) and (9b), we have

$$\begin{aligned} &\langle \nabla_{\mathbf{x}} f(\mathbf{y}^r) + \mathbf{A}^T\boldsymbol{\lambda}^{r+1} + (\beta\mathbf{I} - \rho\mathbf{A}^T\mathbf{A})(\mathbf{x}^{r+1} - \mathbf{x}^r) \\ &+ \rho\mathbf{A}^T\mathbf{B}(\mathbf{y}^r - \mathbf{y}^{r+1}), \mathbf{x} - \mathbf{x}^r\rangle = 0, \ \forall\, \mathbf{x}, \end{aligned} \tag{64a}$$

$$\begin{aligned} &\langle \nabla_{\mathbf{y}} g(\mathbf{y}^r) + \mathbf{B}^T\boldsymbol{\lambda}^{r+1} \\ &+ (\beta\mathbf{I} - \rho\mathbf{B}^T\mathbf{B})(\mathbf{y}^{r+1} - \mathbf{y}^r), \mathbf{y} - \mathbf{y}^r\rangle = 0, \ \forall\, \mathbf{y}. \end{aligned} \tag{64b}$$

Then subtracting the previous iteration of the same condition in (64), and adding them together, we obtain

$$\begin{aligned} &\langle \boldsymbol{\lambda}^{r+1} - \boldsymbol{\lambda}^r, \mathbf{B}(\mathbf{y}^{r+1} - \mathbf{y}^r) + \mathbf{A}(\mathbf{x}^{r+1} - \mathbf{x}^r)\rangle \\ \le{}& \langle \nabla_{\mathbf{y}} g(\mathbf{y}^r) - \nabla_{\mathbf{y}} g(\mathbf{y}^{r-1}), \mathbf{y}^r - \mathbf{y}^{r+1}\rangle \\ &+ \langle \nabla_{\mathbf{x}} f(\mathbf{x}^r) - \nabla_{\mathbf{x}} f(\mathbf{x}^{r-1}), \mathbf{x}^r - \mathbf{x}^{r+1}\rangle \\ &- \langle (\beta\mathbf{I} - \rho\mathbf{B}^T\mathbf{B})\mathbf{v}^{r+1}, \mathbf{y}^{r+1} - \mathbf{y}^r\rangle \\ &- \langle (\beta\mathbf{I} - \rho\mathbf{A}^T\mathbf{A})\mathbf{w}^{r+1}, \mathbf{x}^{r+1} - \mathbf{x}^r\rangle \\ &+ \langle \rho\mathbf{A}^T\mathbf{B}\mathbf{v}^{r+1}, \mathbf{x}^{r+1} - \mathbf{x}^r\rangle. \end{aligned}$$

Collecting terms, and after some simple manipulations, we obtain

$$\begin{aligned} &\frac{1}{2\rho}\|\boldsymbol{\lambda}^{r+1} - \boldsymbol{\lambda}^r\|^2 + \frac{1}{2}\|\mathbf{x}^{r+1} - \mathbf{x}^r\|^2_{\mathbf{C}_3} + \frac{1}{2}\|\mathbf{y}^{r+1} - \mathbf{y}^r\|^2_{\mathbf{C}_4} \\ \le{}& \frac{1}{2\rho}\|\boldsymbol{\lambda}^r - \boldsymbol{\lambda}^{r-1}\|^2 + \frac{1}{2}\|\mathbf{x}^r - \mathbf{x}^{r-1}\|^2_{\mathbf{C}_3} + \frac{1}{2}\|\mathbf{y}^r - \mathbf{y}^{r-1}\|^2_{\mathbf{C}_4} \\ &+ \frac{1}{2}\|\mathbf{x}^{r+1} - \mathbf{x}^r\|^2_{\mathbf{C}_5} + \frac{1}{2}\|\mathbf{y}^{r+1} - \mathbf{y}^r\|^2_{\mathbf{C}_6} - \frac{1}{2c^2}\|\mathbf{w}^{r+1}\|^2_{\mathbf{C}_1} \\ &- \frac{1}{2}\|\mathbf{v}^{r+1}\|^2_{\mathbf{C}_2 - \rho^2\mathbf{B}^T\mathbf{B}} \end{aligned} \tag{65}$$

where matrices $\mathbf{C}_3$–$\mathbf{C}_6$ are defined by

$$\mathbf{C}_3 \triangleq (\beta\mathbf{I} - \rho\mathbf{A}^T\mathbf{A}) + L_{\mathbf{x}}\mathbf{I}, \quad \mathbf{C}_4 \triangleq (\beta\mathbf{I} - \rho\mathbf{B}^T\mathbf{B}) + L_{\mathbf{y}}\mathbf{I},$$
$$\mathbf{C}_5 \triangleq 2L_{\mathbf{x}}\mathbf{I} + \mathbf{A}^T\mathbf{A}, \mathbf{C}_6 \triangleq 2L_{\mathbf{y}}\mathbf{I}.$$

Next, by the standard descent estimate for the gradient-type algorithm, e.g., (9a) and (9b), we can show that the augmented Lagrangian decreases in the following manner

$$\begin{aligned} &\mathcal{L}(\mathbf{x}^{r+1}, \mathbf{y}^{r+1}, \boldsymbol{\lambda}^{r+1}) - \mathcal{L}(\mathbf{x}^r, \mathbf{y}^r, \boldsymbol{\lambda}^r) \\ \le{}& -\frac{\beta}{2}\|\mathbf{x}^{r+1} - \mathbf{x}^r\|^2 - \frac{\beta}{2}\|\mathbf{y}^{r+1} - \mathbf{y}^r\|^2 + \frac{1}{\rho}\|\boldsymbol{\lambda}^{r+1} - \boldsymbol{\lambda}^r\|^2 \\ \le{}& -\frac{\beta}{2}\|\mathbf{x}^{r+1} - \mathbf{x}^r\|^2 - \frac{\beta}{2}\|\mathbf{y}^{r+1} - \mathbf{y}^r\|^2 \\ &+ \frac{2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})}\|\mathbf{z}^{r+1}\|^2_{\mathbf{C}^T\mathbf{C}} \\ &+ \frac{2\,L_{\mathbf{y}}^2\|\mathbf{y}^r - \mathbf{y}^{r-1}\|^2 + 2\,L_{\mathbf{x}}^2\|\mathbf{x}^r - \mathbf{x}^{r-1}\|^2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})} \end{aligned} \tag{66}$$

whenever the following holds

$$\beta - L_{\mathbf{x}} - \sigma_{\max}(\rho\mathbf{A}^T\mathbf{A}) > 0, \quad \beta - L_{\mathbf{y}} - \sigma_{\max}(\rho\mathbf{B}^T\mathbf{B}) > 0.$$

Adding the above inequality with (65) multiplied by a constant $2c > 0$, we obtain

$$\begin{aligned} &\mathcal{L}(\mathbf{x}^{r+1}, \mathbf{y}^{r+1}, \boldsymbol{\lambda}^{r+1}) + \frac{c}{\rho}\|\boldsymbol{\lambda}^{r+1} - \boldsymbol{\lambda}^r\|^2 + c\|\mathbf{x}^{r+1} - \mathbf{x}^r\|^2_{\mathbf{C}_3} \\ &+ \left(\frac{2\,L_{\mathbf{x}}^2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})}\right)\|\mathbf{x}^{r+1} - \mathbf{x}^r\|^2 \\ &+ \left(\frac{2\,L_{\mathbf{y}}^2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})}\right)\|\mathbf{y}^{r+1} - \mathbf{y}^r\|^2 + c\|\mathbf{y}^{r+1} - \mathbf{y}^r\|^2_{\mathbf{C}_4} \\ \le{}& \mathcal{L}(\mathbf{x}^r, \mathbf{y}^r, \boldsymbol{\lambda}^r) + \frac{c}{\rho}\|\boldsymbol{\lambda}^r - \boldsymbol{\lambda}^{r-1}\|^2 + c\|\mathbf{x}^r - \mathbf{x}^{r-1}\|^2_{\mathbf{C}_3} \\ &+ \left(\frac{2\,L_{\mathbf{x}}^2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})}\right)\|\mathbf{x}^r - \mathbf{x}^{r-1}\|^2 \\ &+ \left(\frac{2\,L_{\mathbf{y}}^2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})}\right)\|\mathbf{y}^r - \mathbf{y}^{r-1}\|^2 + c\|\mathbf{y}^r - \mathbf{y}^{r-1}\|^2_{\mathbf{C}_4} \\ &- \left(\frac{\beta}{2} - c(2L_{\mathbf{x}} + \sigma_{\max}(\mathbf{A}^T\mathbf{A})) - \frac{2\,L_{\mathbf{x}}^2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})}\right)\|\mathbf{x}^{r+1} - \mathbf{x}^r\|^2 \\ &- \left(\frac{\beta}{2} - 2cL_{\mathbf{y}} - \frac{2L_{\mathbf{y}}^2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})}\right)\|\mathbf{y}^{r+1} - \mathbf{y}^r\|^2 \\ &- (\mathbf{z}^{r+1})^T\left(c\begin{bmatrix} \mathbf{C}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_2 - \rho^2\mathbf{B}^T\mathbf{B} \end{bmatrix} - \frac{2\mathbf{C}^T\mathbf{C}}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})}\right)\mathbf{z}^{r+1}. \end{aligned} \tag{67}$$

Therefore, to make the entire potential function decrease, we will need the following conditions

$$\beta - \sigma_{\max}(\rho\mathbf{B}^T\mathbf{B}) - L_{\mathbf{y}} > 0, \ \beta - \sigma_{\max}(\rho\mathbf{A}^T\mathbf{A}) - L_{\mathbf{x}} > 0, \tag{68a}$$

$$c\begin{bmatrix} \mathbf{C}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_2 - \rho^2\mathbf{B}^T\mathbf{B} \end{bmatrix} - \frac{2\mathbf{C}^T\mathbf{C}}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})} \succ 0, \tag{68b}$$

$$\frac{\beta}{2} - 2cL_{\mathbf{x}} - c\sigma_{\max}(\mathbf{A}^T\mathbf{A}) - \frac{2\,L_{\mathbf{x}}^2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})} > 0, \tag{68c}$$

$$\frac{\beta}{2} - 2cL_{\mathbf{y}} - \frac{2\,L_{\mathbf{y}}^2}{\rho\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})} > 0. \tag{68d}$$

These inequalities are consistent, meaning that there exists a choice of β, c, ρ such that they will all be satisfied. One particular choice could be

$$c = \frac{\beta}{8\varpi}, \quad \rho \geq \frac{16\varpi}{\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})} \tag{69}$$

with β chosen large enough, where $\varpi \triangleq \max\{L_{\mathbf{x}}, L_{\mathbf{y}}, \sigma_{\max}(\mathbf{A}^T\mathbf{A})\}$.

Further, it is easy to show that $\mathcal{L}(\mathbf{x}^{r+1}, \mathbf{y}^{r+1}, \boldsymbol{\lambda}^{r+1})$ is lower bounded. See Lemma 5 in [50]. It is also easy to show that there exists a constant $\widehat{c}$ in terms of β, ρ, $\sigma_{\max}(\mathbf{A}^T\mathbf{A})$, $\sigma_{\max}(\mathbf{B}^T\mathbf{B})$, $\widetilde{\sigma}_{\min}(\mathbf{V}^T\mathbf{V})$, which is greater than 0, such that the following holds

$$\|\mathcal{L}(\mathbf{x}^{r+1}, \mathbf{y}^{r+1}, \boldsymbol{\lambda}^{r+1})\| \leq \widehat{c}(\|\mathbf{x}^{r+1} - \mathbf{x}^r\| + \|\mathbf{y}^{r+1} - \mathbf{y}^r\|).$$

The following similar argument as in [50, Theorem 1], we can show that the first part of Theorem 1 is true. In particular, the boundedness of the primal and dual variable follows from part (2) of in [50, Theorem 1], which utilizes assumptions A1 and A4. Further, by using the standard argument in [48, Theorem 2.9], we can claim the global convergence of the sequence $\{\mathbf{x}^{r+1}, \mathbf{y}^{r+1}, \boldsymbol{\lambda}^{r+1}\}$ under the KŁ assumption of $\mathcal{L}(\mathbf{x}, \mathbf{y}, \boldsymbol{\lambda})$. We refer the readers to [44] for a similar argument.

B. Proof of Lemma 7

Proof: First, we have

$$\begin{aligned}\nabla f(\mathbf{X}) &= \frac{1}{2}\frac{\nabla f^2(\mathbf{X})}{f(\mathbf{X})} = \frac{1}{2}\frac{\nabla\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F^2}{\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F} \\ &= \frac{2(\mathbf{X}\mathbf{X}^T - \mathbf{M})\mathbf{X}}{\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F}\end{aligned} \tag{70}$$

and

$$\begin{aligned}\nabla^2 f(\mathbf{X}) &= \frac{1}{2}\frac{(\nabla^2 f^2(\mathbf{X}))f(\mathbf{X}) - \nabla f^2(\mathbf{X})\nabla f(\mathbf{X})^T}{f^2(\mathbf{X})} \\ &= \frac{1}{2}\frac{1}{\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F^2}\Big(\nabla^2\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F^2)\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F \\ &\quad - \nabla\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F^2(\nabla\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F)^T\Big).\end{aligned} \tag{71}$$

Second, let $\mathbf{y}$ denote the vectorization of $\mathbf{X}$, and $\mathbf{U}_i \in \mathbb{R}^{d\times dk}$ be a null matrix except the ith block being an $d \times d$ identity matrix. Hence, we have $\mathbf{X} = [\mathbf{U}_1\mathbf{y}\ \mathbf{U}_2\mathbf{y}\ \ldots\ \mathbf{U}_k\mathbf{y}]$.

It follows that

$$\begin{aligned}\mathbf{X}\mathbf{X}^T &= [\mathbf{U}_1\mathbf{y}\ \mathbf{U}_2\mathbf{y}\ldots\mathbf{U}_k\mathbf{y}][\mathbf{U}_1\mathbf{y}\ \mathbf{U}_2\mathbf{y}\ \ldots\mathbf{U}_k\mathbf{y}]^T \\ &= \sum_{i=1}^k \mathbf{U}_i\mathbf{y}\mathbf{y}^T\mathbf{U}_i^T.\end{aligned} \tag{72}$$

Then, objective function $f(\mathbf{X})$ can be written as

$$g(\mathbf{y}) = \sqrt{\psi_1(\mathbf{y}) - \psi_2(\mathbf{y}) + \|\mathbf{M}\|^2}$$

where

$$\psi_1(\mathbf{y}) \triangleq \mathrm{Tr}(\mathbf{X}\mathbf{X}^T\mathbf{X}\mathbf{X}^T)\sum_{i=1}^k\sum_{j=1}^k(\mathbf{y}^T\mathbf{U}_i^T\mathbf{U}_j\mathbf{y})(\mathbf{y}^T\mathbf{U}_j^T\mathbf{U}_i\mathbf{y}),$$

and $\psi_2(\mathbf{y}) \triangleq 2\mathrm{Tr}(\mathbf{X}\mathbf{X}^T\mathbf{M}) = 2\sum_{i=1}^k \mathbf{y}^T\mathbf{U}_i^T\mathbf{M}\mathbf{U}_i\mathbf{y}$.

Further, we have

$$\nabla\psi_1(\mathbf{y}) = \sum_{i=1}^k\sum_{j=1}^k \mathbf{y}^T(\mathbf{U}_i^T\mathbf{U}_j + \mathbf{U}_j^T\mathbf{U}_i)\mathbf{y}(\mathbf{U}_i^T\mathbf{U}_j + \mathbf{U}_j^T\mathbf{U}_i)\mathbf{y},$$

$$\nabla\psi_2(\mathbf{y}) = 4\sum_{i=1}^k \mathbf{U}_i^T\mathbf{M}\mathbf{U}_i\mathbf{y},$$

and

$$\begin{aligned}&\nabla^2\psi_1(\mathbf{y}) \\ &= \sum_{i=1}^k\sum_{j=1}^k \mathbf{y}^T(\mathbf{U}_i^T\mathbf{U}_j + \mathbf{U}_j^T\mathbf{U}_i)\mathbf{y}(\mathbf{U}_i^T\mathbf{U}_j + \mathbf{U}_j^T\mathbf{U}_i) \\ &\quad + 2\sum_{i=1}^k\sum_{j=1}^k(\mathbf{U}_i^T\mathbf{U}_j + \mathbf{U}_j^T\mathbf{U}_i)\mathbf{y}\mathbf{y}^T(\mathbf{U}_i^T\mathbf{U}_j + \mathbf{U}_j^T\mathbf{U}_i) \\ &= 4\sum_{i=1}^k \mathbf{X}_i^T\mathbf{X}_i\mathbf{U}_i^T\mathbf{U}_i \\ &\quad + 2\sum_{i=1}^k\sum_{j=1}^k(\mathbf{U}_i^T\mathbf{U}_j + \mathbf{U}_j^T\mathbf{U}_i)\mathbf{y}\mathbf{y}^T(\mathbf{U}_i^T\mathbf{U}_j + \mathbf{U}_j^T\mathbf{U}_i),\end{aligned} \tag{73}$$

$$\nabla^2\psi_2(\mathbf{y}) = 4\sum_{i=1}^k \mathbf{U}_i^T\mathbf{M}\mathbf{U}_i. \tag{74}$$

Therefore, the Hessian matrix of $f(\mathbf{X})$ is

$$\begin{aligned}\nabla^2 g(\mathbf{y}) &= \frac{1}{2}\frac{(\nabla^2\, g^2(\mathbf{y}))g(\mathbf{y}) - \nabla g^2(\mathbf{y})\nabla g(\mathbf{y})^T}{g^2(\mathbf{y})} \\ &= \frac{1}{2}\frac{(\nabla^2\, g^2(\mathbf{y}))g(\mathbf{y})}{g^2(\mathbf{y})} - \frac{1}{2}\frac{\nabla g^2(\mathbf{y})\nabla g(\mathbf{y})^T}{g^2(\mathbf{y})}\end{aligned} \tag{75}$$

which implies that

$$\|\nabla^2 g(\mathbf{y})\| \leq \frac{1}{2}\underbrace{\frac{\|\nabla^2\, g^2(\mathbf{y})\|}{g(\mathbf{y})}}_{\triangleq C_1} + \frac{1}{2}\underbrace{\frac{\|\nabla g^2(\mathbf{y})\nabla g(\mathbf{y})^T\|}{g^2(\mathbf{y})}}_{\triangleq C_2}. \tag{76}$$

Next, we will give the upper bounds of C_1 and C_2 as follows.

Upper bound of C_1: Substituting (73) and (74) into (76), we can obtain

$$\begin{aligned}C_1 &\leq \frac{\|\nabla^2\psi_1(\mathbf{y})\| + \|\nabla^2\psi_2(\mathbf{y})\|}{\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F} \\ &\overset{(73),(74)}{\leq} \frac{\|\sum_{i=1}^k 4\|\mathbf{X}_i\|^2 + 2k\mathbf{X}_i\mathbf{X}_i^T\|_F + 4k\|\mathbf{M}\|_F}{\|\mathbf{X}\mathbf{X}^T - \mathbf{M}\|_F} \\ &\leq \frac{\|\sum_{i=1}^k 4\|\mathbf{X}_i\|^2 + 2k\mathbf{X}_i\mathbf{X}_i^T\|_F + 4k\|\mathbf{M}\|_F}{\|\mathbf{X}\mathbf{X}^T\|_F - \|\mathbf{M}\|_F} \\ &\overset{(a)}{\leq} \frac{\|\sum_{i=1}^k 4\|\mathbf{X}_i\|^2 + 2k\mathbf{X}_i\mathbf{X}_i^T\|_F + 4k\|\mathbf{M}\|_F}{\|\mathbf{X}\mathbf{X}^T\|_F} \\ &\leq \frac{\sum_{i=1}^k 4\|\mathbf{X}_i\|^2 + 2k\|\sum_{i=1}^k \mathbf{X}_i\mathbf{X}_i^T\|_F + 4k\|\mathbf{M}\|_F}{\|\mathbf{X}\mathbf{X}^T\|_F} \\ &\overset{(b)}{\leq} 4 + 4\,k\end{aligned}$$

where in (a) we use the fact that $\|\mathbf{X}\mathbf{X}^T\|_F \geq 2\|\mathbf{M}\|_F$, (b) is true because $\sum_{i=1}^k \|\mathbf{X}_i\|^2 \leq \|\sum_{i=1}^k \mathbf{X}_i\mathbf{X}_i^T\|_F = \|\mathbf{X}\mathbf{X}^T\|_F$.

Upper bound of C_2: Substituting (73) and (74) into (76), we can obtain

$$\begin{aligned} C_2 &\leq \frac{1}{2}\frac{\|(\mathbf{X}\mathbf{X}^T-\mathbf{M})\mathbf{X}(\mathbf{X}\mathbf{X}^T-\mathbf{M})\mathbf{X})^T\|_F}{\|\mathbf{X}\mathbf{X}^T-\mathbf{M}\|_F^3} \\ &\leq \frac{1}{2}\frac{\|(\mathbf{X}\mathbf{X}^T-\mathbf{M})\mathbf{X}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T-\mathbf{M})\|_F}{\|\mathbf{X}\mathbf{X}^T-\mathbf{M}\|_F^3} \\ &\leq \frac{1}{2}\frac{\|\mathbf{X}\mathbf{X}^T-\mathbf{M}\|_F^2\|\mathbf{X}\mathbf{X}^T\|_F}{\|\mathbf{X}\mathbf{X}^T-\mathbf{M}\|_F^3} \\ &\leq \frac{1}{2}\frac{\|\mathbf{X}\mathbf{X}^T\|_F}{\|\mathbf{X}\mathbf{X}^T-\mathbf{M}\|_F} \\ &\leq \frac{1}{2}\frac{\|\mathbf{X}\mathbf{X}^T\|_F}{\|\mathbf{X}\mathbf{X}^T\|_F-\|\mathbf{M}\|_F} \\ &\leq \frac{1}{2}\frac{1}{1-\frac{\|\mathbf{M}\|_F}{\|\mathbf{X}\mathbf{X}^T\|_F}} \overset{(a)}{\leq} 1 \end{aligned}$$

where in (a) we use the fact that $\|\mathbf{X}\mathbf{X}^T\|_F \geq 2\|\mathbf{M}\|_F$.

REFERENCES

[1] M. Hong, M. Razaviyayn, and J. Lee, "Gradient primal-dual algorithm converges to second-order stationary solution for nonconvex distributed optimization over networks," in *Proc. Int. Conf. Mach. Learn.*, vol. 80, Jul. 2018, pp. 2009–2018.

[2] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 5330–5340.

[3] S. Lu, X. Zhang, H. Sun, and M. Hong, "GNSD: A gradient-tracking based nonconvex stochastic algorithm for decentralized optimization," in *Proc. IEEE Data Sci. Workshop*, Jun. 2019, pp. 315–321.

[4] L. Guan, Z. Yang, D. Li, and X. Lu, "pdlADMM: An ADMM-based framework for parallel deep learning training with efficiency," *Neurocomputing*, vol. 435, pp. 264–272, 2021.

[5] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed clustering using wireless sensor networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 707–724, Aug. 2011.

[6] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM J. Optim.*, vol. 26, no. 1, pp. 337–364, 2016.

[7] Z. Zhu, Q. Li, X. Yang, G. Tang, and M. B. Wakin, "Distributed low-rank matrix factorization with exact consensus," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8420–8430.

[8] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1035–1048, Mar. 2010.

[9] S. Lu, D. Liu, and J. Sun, "A distributed adaptive GSC beamformer over coordinated antenna arrays network for interference mitigation," in *Proc. Asilomar Conf. Signals, Syst. Comput.*, Nov. 2012, pp. 237–242.

[10] P. D. Lorenzo and G. Scutari, "NEXT: In-network nonconvex optimization," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 2, pp. 120–136, Jun. 2016.

[11] A. Daneshmand, G. Scutari, and V. Kungurtsev, "Second-order guarantees of distributed gradient algorithms," *SIAM J. Optim.*, vol. 30, no. 4, pp. 3029–3068, 2020.

[12] C. Battiloro and P. D. Lorenzo, "Distributed tensor completion over networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 8599–8603.

[13] Y. Zhang and X. Lin, "DiSCO: Distributed optimization for self-concordant empirical loss," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 362–370.

[14] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proc. Neural Inf. Process. Syst.*, 2014, pp. 19–27.

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

[16] K. Kawaguchi, "Deep learning without poor local minima," in *Proc. Neural Inf. Process. Syst.*, 2016, pp. 586–594.

[17] S. Feizi, H. Javadi, J. Zhang, and D. Tse, "Porcupine neural networks: Approximating neural network landscapes," in *Proc. Neural Inf. Process. Syst.*, 2018, pp. 4831–4841.

[18] R. Ge, F. Huang, C. Jin, and Y. Yuan, "Escaping from saddle points—Online stochastic gradient for tensor decomposition," in *Proc. Annu. Conf. Learn. Theory*, 2015, pp. 797–842.

[19] J. Sun, Q. Qu, and J. Wright, "A geometric analysis of phase retrieval," *Found. Comput. Math.*, vol. 18, no. 5, pp. 1131–1198, Oct. 2018.

[20] R. Ge, C. Jin, and Y. Zheng, "No spurious local minima in nonconvex low rank problems: A unified geometric analysis," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1233–1242.

[21] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*, SIAM, 2000.

[22] Y. Nesterov and B. T. Polyak, "Cubic regularization of Newton method and its global performance," *Math. Program.*, vol. 108, no. 1, pp. 177–205, 2006.

[23] Y. Carmon and J. Duchi, "Gradient descent finds the cubic-regularized nonconvex newton step," *SIAM J. Optim.*, vol. 29, no. 3, pp. 2146–2178, 2019.

[24] S. Reddi *et al.*, "A generic approach for escaping saddle points," in *Proc. Int. Conf. Artif. Intell. Statist.*, vol. 84, Apr. 2018, pp. 1233–1242.

[25] J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht, "Gradient descent only converges to minimizers," in *Proc. Annu. Conf. Learn. Theory*, 2016, pp. 1246–1257.

[26] J. D. Lee, I. Panageas, G. Piliouras, M. Simchowitz, M. I. Jordan, and B. Recht, "First-order methods almost always avoid strict saddle points," *Math. Program.*, vol. 176, no. 1, pp. 311–337, Jul. 2019.

[27] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, "How to escape saddle points efficiently," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1724–1732.

[28] S. Lu, M. Hong, and Z. Wang, "PA-GD: On the convergence of perturbed alternating gradient descent to second-order stationary points for structured nonconvex optimization," in *Proc. Int. Conf. Mach. Learn.*, vol. 97, 2019, pp. 4134–4143.

[29] I. D. Schizas, G. Mateos, and G. B. Giannakis, "Distributed LMS for consensus-based in-network adaptive processing," *IEEE Trans. Signal Process.*, vol. 57, no. 6, pp. 2365–2382, Jun. 2009.

[30] P. Bianchi and J. Jakubowicz, "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization," *IEEE Trans. Autom. Control*, vol. 58, no. 2, pp. 391–405, Feb. 2013.

[31] A. Nedic and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 601–615, Mar. 2015.

[32] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 944–966, 2015.

[33] S. Lu and C. W. Wu, "Decentralized stochastic non-convex optimization over weakly connected time-varying digraphs," in *Proc. Int. Conf. Acoust. Speech, Signal Process.*, 2020, pp. 5770–5774.

[34] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[35] R. Altilio, P. Di Lorenzo, and M. Panella, "Distributed data clustering over networks," *Pattern Recognit.*, vol. 93, pp. 603–620, 2019.

[36] T. Chang, M. Hong, H. Wai, X. Zhang, and S. Lu, "Distributed learning in the nonconvex world: From batch data to streaming and beyond," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 26–38, May 2020.

[37] B. Swenson, R. Murray, H. V. Poor, and S. Kar, "Distributed gradient descent: Nonconvergence to saddle points and the stable-manifold theorem," in *Proc. Annu. Allerton Conf. Commun., Control, Comput.*, 2019, pp. 595–601.

[38] B. Swenson, R. Murray, S. Kar, and H. V. Poor, "Distributed stochastic gradient descent and convergence to local minima," 2020, *arXiv:2003.02818*.

[39] B. Swenson, S. Kart, H. V. Poor, and J. M. F. Moura, "Distributed global optimization by annealing," in *Proc. IEEE Int. Workshop Comput. Adv. Multi-Sensor Adaptive Process.*, 2019, pp. 181–185.

[40] S. Vlaski and A. H. Sayed, "Distributed learning in non-convex environments—Part II: Polynomial escape from saddle-points," 2019, *arXiv:1907.01849*.

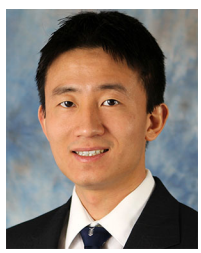
[41] D. P. Bertsekas, *Nonlinear Program.*, 2nd ed., Belmont, MA, USA: Athena Scientific, 1999.
[42] J. Eckstein and D. P. Bertsekas, "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators," *Math. Program.*, vol. 55, no. 1, pp. 293–318, 1992.
[43] X. Gao, B. Jiang, and S. Zhang, "On the information-adaptive variants of the ADMM: An iteration complexity perspective," *J. Sci. Comput.*, vol. 76, no. 1, pp. 327–363, Jul. 2018.
[44] G. Li and T. K. Pong, "Global convergence of splitting methods for nonconvex composite optimization," *SIAM J. Optim.*, vol. 25, pp. 2434–2460, 2015.
[45] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, no. 1, pp. 29–63, Jan. 2019.
[46] M. L. Gonçalves, J. G. Melo, and R. D. Monteiro, "Convergence rate bounds for a proximal ADMM with over-relaxation stepsize parameter for solving nonconvex linearly constrained problems," 2017, *arXiv:1702.01850*.
[47] B. Jiang, T. Lin, S. Ma, and S. Zhang, "Structured nonconvex and nonsmooth optimization: Algorithms and iteration complexity analysis," *Comput. Optim. Appl.*, vol. 72, no. 1, pp. 115–157, Jan. 2019.
[48] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Math. Program.*, vol. 146, no. 1, pp. 459–494, Aug. 2014.
[49] Q. Liu, X. Shen, and Y. Gu, "Linearized ADMM for nonconvex nonsmooth optimization with convergence analysis," *IEEE Access*, vol. 7, pp. 76131–76144, 2019.
[50] M. Hong, D. Hajinezhad, and M.-M. Zhao, "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1529–1538.
[51] G. W. Stewart, *Matrix Perturbation Theory*, Citeseer, 1990.
[52] M. Shub, *Global Stability of Dynamical Systems*, New York, NY, USA: Springer-Verlag, 1997.
[53] M. Hong, "Decomposing linearly constrained nonconvex problems by a proximal primal dual approach: Algorithms, convergence, and applications," 2016, *arXiv:1604.00543*.
[54] H. Uzawa, "Iterative methods for concave programming," in *Studies in Linear and Nonlinear Programming*. Stanford Univ. Press, 1958.
[55] A. Nedić and A. Ozdaglar, "Subgradient methods for saddle-point problems," *J. Optim. Theory Appl.*, vol. 142, no. 1, pp. 205–228, 2009.
[56] R. T. Rockafellar, "Augmented Lagrangians and applications of the proximal point algorithm in convex programming," *Math. Operations Res.*, vol. 1, no. 2, pp. 97–116, 1976.
[57] S. J. Wright, "Implementing proximal point methods for linear programming," *J. Optim. Theory Appl.*, vol. 65, no. 3, pp. 531–554, 1990.
[58] X. Li *et al.*, "Symmetry, saddle points, and global optimization landscape of nonconvex matrix factorization," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3489–3514, Jun. 2019.
[59] Z. Zhu, Q. Li, G. Tang, and M. B. Wakin, "Global optimality in low-rank matrix optimization," *IEEE Trans. Signal Process.*, vol. 66, no. 13, pp. 3614–3628, Jul. 2018.
[60] J. Sun, Q. Qu, and J. Wright, "Complete dictionary recovery over the sphere I: Overview and the geometric picture," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, pp. 853–884, Feb. 2017.
[61] R. Ge, J. D. Lee, and T. Ma, "Learning one-hidden-layer neural networks with landscape design," in *Proc. Int. Conf. Learn. Representations*, 2017.
[62] M. Soltanolkotabi, A. Javanmard, and J. D. Lee, "Theoretical insights into the optimization landscape of over-parameterized shallow neural networks," *IEEE Trans. Inf. Theory*, vol. 65, no. 2, pp. 742–769, Feb. 2019.
[63] Q. Li, Z. Zhu, and G. Tang, "The non-convex geometry of low-rank matrix optimization," *Inf. Inference: A, J. IMA*, vol. 8, no. 1, pp. 51–96, 2018.
[64] S. S. Du and J. D. Lee, "On the power of over-parametrization in neural networks with quadratic activation," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1328–1337.
[65] C. Wei, J. D. Lee, Q. Liu, and T. Ma, "Regularization matters: Generalization and optimization of neural nets vs their induced kernel," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 9709–9721.
[66] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*, New York, NY, USA: Springer, 2001.
[67] Z. Zhu, Q. Li, G. Tang, and M. B. Wakin, "The global optimization geometry of low-rank matrix optimization," *IEEE Trans. Inf. Theory*, vol. 67, no. 2, pp. 1308–1331, Feb. 2021.
[68] S. Scardapane and P. Di Lorenzo, "A framework for parallel and distributed training of neural networks," *Neural Netw.*, vol. 91, pp. 42–54, 2017.
[69] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

Songtao Lu (Member, IEEE) received the Ph.D. degree in electrical engineering from Iowa State University, Ames, IA, USA, in 2018. He is currently a Research Scientist with the Mathematics of Artificial Intelligence (AI) Group, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. From 2018 to 2019, he was a Postdoctoral Associate with the Department of Electrical and Computer Engineering, University of Minnesota Twin Cities, Minneapolis, MN, USA and from 2019 to 2020, an AI Resident with IBM Thomas J. Watson Research Center, Ossining, NY, USA. His primary research interests include signal processing, AI, machine learning, and optimization.

He was the recipient of the Graduate and Professional Student Senate Research Award at Iowa State University in 2015, the Research Excellence Award from the Graduate College at Iowa State University in 2017, the Student Travel Award from the 20th International Conference on Artificial Intelligence and Statistics, and the Postdoc Travel award from the 36th International Conference on Machine Learning.

Jason D. Lee received the Ph.D. degree from Stanford University, Stanford, CA, USA, advised by Trevor Hastie and Jonathan Taylor. He is currently an Assistant Professor of electrical engineering and computer science (courtesy) with Princeton University, Princeton, NJ, USA. Prior to that, he was with Data Science and Operations Department, University of Southern California, Los Angeles, CA, USA and a Postdoctoral Researcher with UC Berkeley, Berkeley, CA, USA, working with Michael I. Jordan. His research interests include the theory of machine learning, optimization, and statistics. He has worked on the foundations of deep learning, nonconvex optimization algorithm, and reinforcement learning. He was the recipient of ONR Young Investigator Award in Mathematical Data Science, Sloan Research Fellowship in 2019, NeurIPS Best Student Paper Award, and Finalist for the Best Paper Prize for Young Researchers in Continuous Optimization.

Meisam Razaviyayn received the Ph.D. degree in electrical engineering with minor in computer science from the University of Minnesota, Minneapolis, MN, USA. He is currently an Assistant Professor with Department of Industrial and Systems Engineering, University of Southern California (USC), Los Angeles, CA, USA. Prior to joining USC, he was a Postdoctoral Research Fellow with the Department of Electrical Engineering, Stanford University, Stanford, CA, USA. He was the recipient of the Signal Processing Society Young Author Best Paper Award in 2014, Best Paper Award in IEEE Data Science Workshop in 2019, ICCM Best Paper Award in Mathematics in 2020, and 3M's Non-Tenured Faculty Award in 2021. He was the finalist for Best Paper Prize for Young Researcher in Continuous Optimization in 2013 and 2016.



Mingyi Hong (Senior Member, IEEE) received the Ph.D. degree from the University of Virginia, Charlottesville, VA, USA, in 2011. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA. His research interests include optimization theory and applications in signal processing and machine learning. He is on the IEEE Signal Processing for Communications and Networking Technical Committee.