A Distributed and Secure Algorithm for Computing Dominant SVD Based on Projection Splitting

Lei Wang* Xin Liu† Yin Zhang‡

Abstract

In this paper, we propose and study a distributed and secure algorithm for computing dominant (or truncated) singular value decompositions (SVD) of large and distributed data matrices. We consider the scenario where each node privately holds a subset of columns and only exchanges "safe" information with other nodes in a collaborative effort to calculate a dominant SVD for the whole matrix. In the framework of alternating direction methods of multipliers (ADMM), we propose a novel formulation for building consensus by equalizing subspaces spanned by splitting variables instead of equalizing themselves. This technique greatly relaxes feasibility restrictions and accelerates convergence significantly, while at the same time yielding simple subproblems. We design several algorithmic features, including a low-rank multiplier formula and mechanisms for controlling subproblem solution accuracies, to increase the algorithm's computational efficiency and reduce its communication overhead. More importantly, unlike many existing distributed or parallelized algorithms, our algorithm preserves the privacy of locally-held data; that is, none of the nodes can recover the data stored in another node through information exchanged during communications. We present convergence analysis results, including a worst-case complexity estimate, and extensive experimental results indicating that the proposed algorithm, while safely guarding data privacy, has a strong potential to deliver a cutting-edge performance, especially when communication costs are high.

Key words: dominant singular value decomposition, distributed computing, data security, alternating direction method of multipliers

AMS subject classifications: 15A18, 65F15, 65K05, 90C06, 90C26

1 Introduction

Singular value decomposition (SVD) is a fundamental and ubiquitous technique in matrix computation with a wide and still rapidly growing variety of applications, such as principal component analysis [34], image compression [2], dictionary learning [1], facial recognition [49], latent semantic analysis [7], matrix completion [5], and so on.

Consider $A \in \mathbb{R}^{n\times m}$, an $n \times m$ real matrix, for which we will always assume $n \leq m$ without loss of generality, and let $p < n$ be a positive integer. The (economy-form) SVD of A is represented below as the decomposition on the left, while the p-term approximation of it on the right is called a dominant (or truncated) SVD of A,

$$A = U\Sigma V^\top = \sum_{i=1}^{n} \sigma_i u_i v_i^\top \approx \sum_{i=1}^{p} \sigma_i u_i v_i^\top.$$

*State Key Laboratory of Scientific and Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, and University of Chinese Academy of Sciences, China (wlkings@lsec.cc.ac.cn). Research is supported by the National Natural Science Foundation of China (No. 11971466).

†State Key Laboratory of Scientific and Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, and University of Chinese Academy of Sciences, China (liuxin@lsec.cc.ac.cn). Research is supported in part by the National Natural Science Foundation of China (No. 11991021, 11991020 and 11971466), Key Research Program of Frontier Sciences, Chinese Academy of Sciences (No. ZDBS-LY-7022), the National Center for Mathematics and Interdisciplinary Sciences, Chinese Academy of Sciences and the Youth Innovation Promotion Association, Chinese Academy of Sciences.

‡School of Data Science, The Chinese University of Hong Kong, Shenzhen, China, and Computational and Applied Mathematics, Rice University, Texas, United States (yinzhang@cuhk.edu.cn).

Here, $U = [u_1, \ldots, u_n] \in \mathbb{R}^{n\times n}$ and $V = [v_1, \ldots, v_n] \in \mathbb{R}^{m\times n}$ are orthogonal matrices, and $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix with diagonal entries, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$, being the singular values of A. The columns of U and V, i.e., u_i and v_i, are called left and right singular vectors of A, respectively.

As is well-known, the p-term dominant SVD provides the optimal rank-p approximation to A in Frobenius norm, and the first p left singular vectors of A form an orthonormal basis for the eigenspace associated with the p largest eigenvalues of $AA^\top$. In truly large-scale applications, computing a dominant SVD is practically affordable only for $p \ll n$. Often times, such a dominant SVD is sufficient since it contains the most relevant information about the underlying system or dataset represented by the matrix A.

1.1 Communication overhead and data security

To develop scalable capacities for SVD calculations in today's big-data environments, it is critical to study algorithms that can efficiently and securely process distributed and massively large-scale datasets. For a given large-scale data matrix $A \in \mathbb{R}^{n\times m}$, we call each column of A a sample with length n, and the number of samples is m which is assumed to be far bigger than n. In this paper, we consider the following setting: the data matrix A is divided into d blocks, each containing a number of samples; namely, $A = [A_1, A_2, \ldots, A_d]$ where $A_i \in \mathbb{R}^{n\times m_i}$ so that $m_1 + \cdots + m_d = m$. These submatrices A_i, $i = 1, \ldots, d$, are stored locally in d locations or nodes, possibly having been collected at different locations by different entities, and all the nodes are connected through a communication network.

In evaluating distributed algorithms, a key measure of performance is the total amount of communications required by algorithms. In general, during iterations heavy computations are mostly done at the local level within each node, and communications occur between iterations in order to exchange and share certain computed quantities. If the amount of communications remains more or less the same at each iteration, the total communication overhead is determined by the number of required iterations regardless of the underlying network topology. In this work, one of our primary considerations is to devise an algorithm that converges fast in terms of iteration counts. Issues related to network topologies and communication patterns will not be within the scope of the current paper.

In certain application areas, such as in those of healthcare and financial industry [33, 52], securing the privacy of local data is a primary requirement that many existing algorithms do not necessarily meet. For example, although common parallelization techniques in numerical linear algebra can in principle be adopted to distributed computing environments, such direct adaptations usually do not preserve data privacy, as will be discussed in Subsection 1.3. The ability to preserve data privacy is another core consideration of this paper.

1.2 Overview of related works

The subject of computing SVD (including dominant SVD) has been thoroughly studied over several decades and various iterative algorithms have been developed, mostly based on computing eigenpairs of the symmetric matrix $AA^\top$ (or $A^\top A$). We briefly review a small subset of algorithms closely related to the present work. In particular, we focus on those designed for distributed computing and capable of keeping data security.

Classical SVD algorithms are mostly based on Krylov subspace techniques, such as Arnoldi algorithms [3, 26], Lanczos algorithms [12, 14, 24], and Jacobi-Davidson algorithms [42, 43]. Due to their inherent sequential structure, Krylov subspace methods lack concurrency, making it difficult to achieve high scalability in parallel computing. There are works in parallelizing classical SVD algorithms at the linear algebra level, often focusing on reducing communication costs. However, these approaches are out of the scope of the present paper, we hence refer interested readers to [8, 17, 21, 48] and the references therein for more details.

Another class of methods is called block (updating) algorithms, as represented by the classic method of simultaneous subspace iteration (SSI) [39,44,45], that generally have higher scalability because their main computations are large matrix times (relatively) small and dense block matrices. Indeed there are various practical implementations of SSI under distributed settings, such as [10, 13, 25, 37, 47].

The main drawback of SSI lies in its slow convergence under unfavorable conditions. To improve convergence, other block algorithms are proposed, such as LOBPCG [22, 23] and LMSVD [30]. For these algorithms, a scalability bottleneck arises from the need for performing orthonormalization, an essentially sequential procedure, at each iteration. To increase the parallel scalability, another type of algorithms is constructed based on unconstrained optimization without the orthogonality constraint. Two members of this type are EigPen [50] and SLRPGN [31], both of which perform orthonormalization once for a while at low frequencies.

It is worth emphasizing that most block algorithms can be parallelized at the linear algebra level (we call this approach direct parallelization for short), but cannot preserve data privacy (see Subsection 1.3 below). Another class of methods use the approach of alternating direction methods of multipliers (ADMM) to achieve an algorithm-level parallelization. This ADMM approach is in the framework of augmented Lagrangian methods [38] while utilizing a variable-splitting technique to divide a computational subproblem at each iteration into smaller subproblems which are solved simultaneously and locally at the node level without communications. After each iteration, shared information is collected from all the nodes to update a consensus variable. In terms of data security, ADMM-based methods have an advantage in that the relationships between shared quantities and local data can be nonlinear and non-transparent, making it difficult, if possible at all, to uncover local data.

For example, the authors in [40] convert a low-rank matrix approximation model to a global variable consensus problem [4] by introducing local variables and then apply ADMM to solve the model. The resulting algorithm, called D-PCA, is able to avoid data leakage. However, convergence of ADMM algorithms based on variable-splitting can be highly sensitive to the choice of algorithm parameters and generally slow. Such variable-splitting algorithms usually require a large number of iterations to attain a high or even moderate accuracy, resulting in high communication overhead.

All the algorithms introduced above are designed for computing solutions to a relatively high precision. There are other algorithms, based on a very different philosophy, that are designed to quickly reach a relatively low accuracy (but considered sufficient for targeted applications). Such algorithms include Oja's methods [28,35,51], variance reduction method [41], and randomized sampling methods [18, 27, 29], all using some stochastic techniques one way or another. Other low-accuracy methods include [11, 19, 20]. In this paper, we only focus on algorithms designed for computing a dominant SVD to a relatively high accuracy.

1.3 Why direct parallelization is not secure

The distributed versions of block algorithms, including SSI, LOBPCG, LMSVD, EigPen, and SLRPGN, are all based on parallelization at the linear-algebra level, where the matrix-block multiplication $AA^\top X$ is divided and distributed into d nodes. At node i, a matrix-block multiplication $A_iA_i^\top X$ $(i = 1, \ldots, d)$ is calculated and then shared with other nodes by a certain communication strategy. In the following, we illustrate that such a parallelization does not preserve data security using the algorithm SLRPGN [31] as an example. That is, the data matrix $A_iA_i^\top$ can be recovered from a set of shared quantities at iterations $k = 1, 2, \ldots, N$ as long as N is sufficiently large.

Under the distributed setting, the main iteration of SLRPGN reads

$$X^{(k+1)} = X^{(k)} + \tau^{(k)} \left(\sum_{i=1}^{d} S_i(X^{(k)}) - \frac{1}{2} X^{(k)} \right), \tag{1.1}$$

for a step size $\tau^{(k)} > 0$, where, for $i = 1, \ldots, d$,

$$S_i(X) = (I_n - X \left(X^\top X\right)^{-1} X^\top/2) \left(A_iA_i^\top\right) X \left(X^\top X\right)^{-1}. \tag{1.2}$$

At iteration k, all nodes have access to both $X^{(k)}$ and $S_i(X^{(k)})$, which is computed at node i using local data A_i and then shared with others in order to compute the sum in (1.1). We note that, for given X (of full column-rank) and $S_i(X)$, (1.2) provides a set of linear equations for the "unknown" $A_iA_i^\top$. Should any other node aim to recover the private data $A_iA_i^\top$, it would only need to collect a sufficient number of publicly shared matrices $\{X^{(k)}\}$ and $\{S_i(X^{(k)})\}$, and then to solve the resulting

linear system of equations for $A_i A_i^\top$ as given in (1.2). Under mild conditions, $A_i A_i^\top$ would be uniquely determined by the (likely over-determined) linear system. Furthermore, once $A_i A_i^\top$ is known, A_i can in principle be determined up to a rotation.

Evidently, there exists a risk of data leakage when SLRPGN is employed to compute a dominant SVD under a distributed setting. Likewise, other block algorithms, such as SSI, LOBPCG, LMSVD, and EigPen, all suffer from the similar vulnerability in terms of data security.

1.4 Contributions

The use of variable splitting technique together with an ADMM algorithm is a popular approach to solving optimization problems distributively over networks. The first novelty of this paper is to propose a special splitting model for trace optimization problems with orthogonality constraints in the context of calculating a dominant SVD. Instead of splitting matrix variables, we split subspaces represented by orthogonal projections. Since the objective function varies with subspaces but not with particular orthonormal bases, the proposed strategy greatly relaxes feasibility restrictions, and at the same time yields nicely tractable subproblems. However, this projection-splitting strategy also gives rise to large-size matrix constraints corresponding to large-size multiplier matrices. To overcome this difficulty, we derived a close-form, low-rank multiplier formula that is computationally efficient. We also incorporated other algorithmic techniques such as inexact subproblem solving. The overall contribution of this work is the successful construction of a distributed algorithm that preserves data privacy, converges fast, and has a low communication overhead. In addition, we established a theoretical convergence result for our specialized ADMM algorithm for a nonconvex optimization model.

1.5 Notations

We use $\mathbb{R}$ and $\mathbb{N}$ to denote the sets of real and natural numbers, respectively. The $p \times p$ identity matrix is represented by I_p. The Euclidean inner product of two matrices Y_1 and Y_2 of the same size is defined as $\langle Y_1, Y_2 \rangle = \mathrm{tr}(Y_1^\top Y_2)$, where $\mathrm{tr}(B)$ is the trace of a square matrix B. The Frobenius norm and 2-norm of a matrix X are denoted by $\|X\|_{\mathrm{F}}$ and $\|X\|_2$, respectively. For a matrix X, the notation $\mathrm{rank}\,(X)$ stands for its rank; $\mathbf{orth}(X)$ refers to the set of orthonormal bases for its range space; and $\sigma_{\min}(X)$ denotes its smallest singular value. The set $\mathcal{S}_{n,p} := \left\{ X \in \mathbb{R}^{n\times p} \mid X^\top X = I_p \right\}$ is referred to as the Stiefel manifold [46]. For $X, Y \in \mathcal{S}_{n,p}$, we define $\mathbf{P}_X^\perp := I_n - XX^\top$, $\mathbf{D_p}\,(X,Y) := XX^\top - YY^\top$, and $\mathbf{d_p}\,(X,Y) := \|\mathbf{D_p}\,(X,Y)\|_{\mathrm{F}}$, standing for, respectively, the projection operator, the projection distance matrix and the projection distance. Other notations will be introduced at their first appearance.

1.6 Organization

The rest of this paper is organized as follows. In Section 2, we introduce a novel model with so-called projection splitting constraints, and investigate the structure of associated Lagrangian multipliers. Then we propose a distributed and secure algorithm for solving this model based on an ADMM framework in Section 3. Convergence properties of the proposed algorithm are studied in Section 4. Numerical experiments on a variety of test problems are presented in Section 5 to evaluate the performance of the proposed algorithm. We conclude the paper in the last section.

2 Projection Splitting Model

We first motivate the proposed projection-splitting model, then derive a low-rank formula for Lagrangian multipliers associated with the projection splitting constraints. This low-rank formula is essential to make the ADMM approach practical in its application to the proposed model with large-scale data matrices.

2.1 Pursuit of an optimal subspace, not basis

Computing a dominant SVD of a matrix A can be formulated as solving the following trace minimization problem with the orthogonality constraint:

$$\min_{X \in \mathcal{S}_{n,p}} \quad f(X) := -\frac{1}{2}\mathrm{tr}\left(X^\top A A^\top X\right). \tag{2.1}$$

It is worth emphasizing that both the objective function f and the feasible region $\mathcal{S}_{n,p}$ are invariant under the transformation $X \to XO$ for any orthogonal matrix $O \in \mathbb{R}^{p\times p}$. In essence, we are to pursue an optimal subspace rather than an optimal basis. Indeed, as is well-known, a global minimizer of (2.1) can be any orthonormal basis matrix for the optimal subspace spanned by the p left singular vectors associated with the largest p singular values of A. In addition, according to [30], the first-order stationarity condition of (2.1) can be expressed as:

$$\mathbf{P}_X^\perp A A^\top X = 0 \ \text{ and } \ X \in \mathcal{S}_{n,p}. \tag{2.2}$$

As is mentioned earlier, we have a division of $A = [A_1, \ldots, A_d]$ into d column blocks and the i-th block A_i is stored at the i-th node. Therefore, the objective function $f(X)$ can be recast as a finite sum function:

$$f(X) = \sum_{i=1}^{d} f_i(X) \ \text{ with } \ f_i(X) = -\frac{1}{2}\mathrm{tr}\left(X^\top A_i A_i^\top X\right), \tag{2.3}$$

so that the i-th component of the objective function $f_i(X)$ can be evaluated only at the i-th node since A_i is accessible only at the i-th node. We assume $p < m_i$, for $i = 1, \ldots, d$, hereinafter. To derive a distributed algorithm, we introduce a set of local variables, $\{X_i\}_{i=1}^d$, where at the i-th node $X_i \in \mathcal{S}_{n,p}$ is a local copy of the global variable $Z \in \mathcal{S}_{n,p}$ (here Z instead of X is used to avoid possible future confusion).

At this point, the conventional approach would impose constraints to equalize, one way or another, all the local variables $\{X_i\}_{i=1}^d$ with the global variable Z. For instance, one could formulate the following optimization problem with a separable objective function:

$$\min_{X_i,\, Z \in \mathcal{S}_{n,p}} \quad \sum_{i=1}^{d} f_i(X_i) \quad \text{s.t.} \quad X_i = Z, \ \ i = 1, \ldots, d. \tag{2.4}$$

In this model, the set of variables is $(\{X_i\}_{i=1}^d, Z)$ and f_i is defined in (2.3). When an ADMM scheme is applied to this model, the subproblems corresponding to the local variables can all be solved simultaneously and distributively. This type of variable-splitting schemes, referred to as consensus problems in [4], is essentially what was used by a recent distributed algorithm called D-PCA [40], which will be used in our numerical comparison.

However, we observe that the equalizing constraints in (2.4) require that all local variables $\{X_i\}$ must be equal to each other. In other words, model (2.4) dictates that every node must find exactly the same orthonormal basis for the optimal subspace, which is of course extremely demanding but totally unnecessary. Under such severely restrictive constraints, a consensus is much harder to reach than when each node is allowed to find its own orthonormal basis, independent of each other.

To relax the restrictive equalizing constraints in (2.4), we propose a new splitting scheme that equalizes subspaces spanned by local variables instead of the local variables (matrices) themselves. For this purpose, we replace the equalizing constraints in (2.4) by $X_i X_i^\top = Z Z^\top$ for $i = 1, \ldots, d$. Since both sides of the equations are orthogonal projections (recall $X_i, Z \in \mathcal{S}_{n,p}$), we call our new splitting scheme *projection splitting*. The resulting projection-splitting model is

$$\min_{X_i,\, Z \in \mathcal{S}_{n,p}} \quad \sum_{i=1}^{d} f_i(X_i) \quad \text{s.t.} \quad X_i X_i^\top = Z Z^\top, \ \ i = 1, \ldots, d. \tag{2.5}$$

For ease of reference, we will call the constraints in (2.5) *subspace constraints*. Obviously, these constraints are nonlinear and the optimization model (2.5) is nonconvex. Conceptually, subspace

constraints are easier to satisfy than the variable-splitting constraints $X_i = Z$. Computationally, however, subspace constraints do come with additional difficulties. Since $X_i X_i^\top = ZZ^\top$ are large-size, $n \times n$ matrix equations (compared to $n \times p$ in $X_i = Z$), their corresponding Lagrangian multipliers are also large-size, $n \times n$ matrices. How to treat such large-size multiplier matrices is a critical algorithmic issue that must be effectively addressed.

2.2 Existence of low-rank multipliers

By introducing dual variables, we derive a set of first-order stationarity conditions for the projection-splitting model (2.5) in the following proposition, whose proof will be given in Appendix A.

Proposition 2.1. *Let $(\{X_i \in \mathcal{S}_{n,p}\}_{i=1}^d, Z \in \mathcal{S}_{n,p})$ be a feasible point of the projection-splitting model. Then Z is a first-order stationary point of (2.1) if and only if there exist symmetric matrices $\Lambda_i \in \mathbb{R}^{n\times n}$, $\Gamma_i \in \mathbb{R}^{p\times p}$, and $\Theta \in \mathbb{R}^{p\times p}$ so that the following conditions hold:*

$$\sum_{i=1}^d \Lambda_i Z - Z\Theta = 0, \quad A_i A_i^\top X_i + X_i \Gamma_i + \Lambda_i X_i = 0, \;\; i = 1, \ldots, d. \tag{2.6}$$

The equations in (2.6) along with the feasibility represent the KKT conditions for the projection-splitting model (2.5). The dual variables $\Lambda_i \in \mathbb{R}^{n\times n}$, $\Gamma_i \in \mathbb{R}^{p\times p}$, and $\Theta \in \mathbb{R}^{p\times p}$ are the Lagrangian multipliers associated with the equality constraints $X_i X_i^\top = ZZ^\top$, $X_i^\top X_i = I_p$, and $Z^\top Z = I_p$, respectively.

It is straightforward (but rather lengthy, see Appendix A) to verify that at any first-order stationary point $(\{X_i\}, Z)$ of (2.5) (i.e., besides feasibility, (2.2) also holds at Z), the KKT conditions in (2.6) are satisfied by the following values of multipliers: $\Theta = 0$, $\Gamma_i = -X_i^\top A_i A_i^\top X_i$, and

$$\Lambda_i = -X_i X_i^\top A_i A_i^\top \mathbf{P}_{X_i}^\perp - \mathbf{P}_{X_i}^\perp A_i A_i^\top X_i X_i^\top, \quad i = 1, \ldots, d. \tag{2.7}$$

Clearly, all Λ_i satisfying (2.7) have a rank no greater than $2p$. In fact, they are symmetrization of rank-p matrices. As such, equation (2.7) provides a low-rank, closed-form formula for calculating an estimated multiplier Λ_i at a given X_i. This formulation will play a prominent role in our algorithm, for it effectively eliminates the costs of storing and updating $n \times n$ multiplier matrices.

Remark 1. *We note that multipliers associated with the subspace constraints are non-unique. For example, in addition to (2.7), the matrices*

$$\hat{\Lambda}_i = -A_i A_i^\top \mathbf{P}_{X_i}^\perp - \mathbf{P}_{X_i}^\perp A_i A_i^\top, \quad i = 1, \ldots, d,$$

also satisfy the KKT conditions in (2.6). However, for $p \ll n$, the matrix Λ_i in (2.7) has a much lower rank.

3 Algorithm Development

In this section, we develop a distributed algorithm for solving the projection-splitting model (2.5) based on the ADMM framework. Out of all the constraints, we only bring the subspace constraints in (2.5) into the augmented Lagrangian function:

$$\mathcal{L}(\{X_i\}, Z, \{\Lambda_i\}) = \sum_{i=1}^d \mathcal{L}_i(X_i, Z, \Lambda_i), \tag{3.1}$$

where for $i = 1, \ldots, d$,

$$\mathcal{L}_i(X_i, Z, \Lambda_i) = f_i(X_i) - \frac{1}{2}\left\langle \Lambda_i, \mathbf{D_p}\left(X_i, Z\right)\right\rangle + \frac{\beta_i}{4}\mathbf{d_p^2}\left(X_i, Z\right), \tag{3.2}$$

and $\beta_i > 0$ are penalty parameters. The quadratic penalty term $\mathbf{d_p^2}\left(X_i, Z\right)$ (see Subsection 1.5 for definition) measures the difference between the two subspaces spanned by X_i and Z, respectively.

At iteration k, our algorithm consists of the following three steps.

(1) Update the local variables,

$$X_i^{(k+1)} \approx \underset{X_i \in \mathcal{S}_{n,p}}{\arg\min}\ \mathcal{L}_i(X_i, Z^{(k)}, \Lambda_i^{(k)}),\quad i = 1, \ldots, d.$$

(2) Implicitly "update" the multipliers $\Lambda_i^{(k+1)}$, $i = 1, \ldots, d$.

(3) Update the global variable,

$$Z^{(k+1)} \approx \underset{Z \in \mathcal{S}_{n,p}}{\arg\min}\ \mathcal{L}(\{X_i^{(k+1)}\}, Z, \{\Lambda_i^{(k+1)}\}).$$

The first two steps can be concurrently carried out in d nodes, while the last step requires communications among the nodes. In the next three subsections, we specify in more concrete terms how the three steps are carried out. A detailed algorithm statement will be given in Subsection 3.4, and the issue of data security will be discussed in Subsection 3.5.

3.1 Subproblems for local variables

It is straightforward to derive that the subproblem for the local variables $\{X_i\}_{i=1}^d$ has the following equivalent form:

$$\min_{X_i \in \mathcal{S}_{n,p}} \quad h_i^{(k)}(X_i) := -\frac{1}{2}\mathrm{tr}\left(X_i^\top H_i^{(k)} X_i\right), \tag{3.3}$$

where, for $i = 1, \ldots, d$,

$$H_i^{(k)} = A_i A_i^\top + \Lambda_i^{(k)} + \beta_i Z^{(k)} (Z^{(k)})^\top. \tag{3.4}$$

Clearly, (3.3) is a standard eigenvalue problem where one computes a p-dimensional dominant eigenspace of an $n \times n$ real symmetric matrix. As a subproblem, (3.3) needs not to be solved to a high precision. In fact, we have discovered two inexact-solution conditions that ensure both theoretical convergence and good practical performance. It is important to note that using an iterative eigensolver, one does not need to compute nor store the $n \times n$ matrix $H_i^{(k)}$ since it is accessed through matrix-(multi)vector multiplications.

The first condition is a sufficient reduction in function value:

$$h_i^{(k)}(X_i^{(k)}) - h_i^{(k)}(X_i^{(k+1)}) \geq \frac{c_1}{c_1' \left\|A_i\right\|_2^2 + \beta_i} \left\|\mathbf{P}^\perp_{X_i^{(k)}} H_i^{(k)} X_i^{(k)}\right\|_{\mathrm{F}}^2, \tag{3.5}$$

where $c_1 > 0$ and $c_1' > 0$ are two constants independent of β_i. This kind of conditions has been used to analyze convergence of iterative algorithms for solving trace minimization problems with orthogonality constraints [16, 30].

The second condition is a sufficient decrease in KKT violation:

$$\left\|\mathbf{P}^\perp_{X_i^{(k+1)}} H_i^{(k)} X_i^{(k+1)}\right\|_{\mathrm{F}} \leq \delta_i \left\|\mathbf{P}^\perp_{X_i^{(k)}} H_i^{(k)} X_i^{(k)}\right\|_{\mathrm{F}}, \tag{3.6}$$

where $\delta_i \in [0, 1)$ is a constant independent of β_i. This condition frequently appears in inexact augmented Lagrangian based approaches [9, 32]. It will play a crucial role in our theoretical analysis.

The above two conditions, much weaker than optimality conditions of (3.3), are sufficient for us to derive global convergence of our ADMM algorithm framework. In practice, it usually takes very few iterations of certain iterative eigensolver, such as LMSVD [30] and SLRPGN [31], to meet these two conditions.

3.2 Formula for low-rank multipliers

Now we consider updating the multipliers $\{\Lambda_i\}_{i=1}^d$ associated with the subspace constraints in (2.5). In a regular ADMM algorithm, multiplier Λ_i would be updated by a dual ascent step:

$$\Lambda_i^{(k+1)} = \Lambda_i^{(k)} - \tau_i \beta_i \mathbf{D_p}\left(X_i^{(k+1)}, Z^{(k+1)}\right),$$

where $\tau_i > 0$ is a step size. However, the above dual ascent step requires to store an $n \times n$ matrix at each node, which can be prohibitive when n is large. In our search for an effective multiplier-updating scheme, we derived an explicit, low-rank formula (2.7) in Subsection 2.2 that is satisfied at any first-order stationary point, namely,

$$\Lambda_i^{(k+1)} = X_i^{(k+1)}(W_i^{(k+1)})^\top + W_i^{(k+1)}(X_i^{(k+1)})^\top, \tag{3.7}$$

where, for $i = 1, \ldots, d$,

$$W_i^{(k+1)} = -\mathbf{P}^{\perp}_{X_i^{(k+1)}} A_i A_i^\top X_i^{(k+1)}. \tag{3.8}$$

With this low-rank expression, one can produce matrix-(multi)vector products involving Λ_i without any storage besides X_i (optionally one more $n \times p$ matrix W_i for computational convenience). We note that Λ_i in formula (3.7) is independent of the global variable Z. Thus, we choose to "update" Λ_i after X_i and before Z.

3.3 Subproblem for global variable

The subproblem for the global variable Z can also be rearranged into a standard eigenvalue problem:

$$\min_{Z \in \mathcal{S}_{n,p}} \quad q^{(k)}(Z) := -\frac{1}{2}\mathrm{tr}\left(Z^\top Q^{(k)} Z\right), \tag{3.9}$$

where $Q^{(k)}$ is a sum of d locally held matrices:

$$Q^{(k)} = \sum_{i=1}^{d} Q_i^{(k)} \quad \text{with} \quad Q_i^{(k)} = \beta_i X_i^{(k+1)}(X_i^{(k+1)})^\top - \Lambda_i^{(k+1)}. \tag{3.10}$$

As is the case for local variables, we also approximately solve (3.9) by an iterative eigensolver. We will require the next iterate $Z^{(k+1)} \in \mathcal{S}_{n,p}$ to satisfy the following sufficient decrease condition in function value:

$$q^{(k)}(Z^{(k)}) - q^{(k)}(Z^{(k+1)}) \geq c_2 \left\| \mathbf{P}^{\perp}_{Z^{(k)}} Q^{(k)} Z^{(k)} \right\|_{\mathrm{F}}^2, \tag{3.11}$$

where $c_2 > 0$ is a constant dependent on the penalty parameters $\beta_i (i = 1, \ldots, d)$.

In practice, we have observed that (3.11) can be achieved by a single iteration of several algorithms for linear eigenvalue problems, such as SSI and SLRPGN. We refer the interested readers to [30, 31] for more details.

When we apply an iterative eigensolver, we need to produce matrix-(multi)vector products of the form $Q^{(k)}Y$ which is the sum of local products $Q_i^{(k)}Y$ for $i = 1, \ldots, d$. Invoking the definitions of $Q_i^{(k)}$ and $\Lambda_i^{(k+1)}$, we have

$$Q_i^{(k)}Y = \beta_i X_i^{(k+1)}(X_i^{(k+1)})^\top Y - X_i^{(k+1)}(W_i^{(k+1)})^\top Y - W_i^{(k+1)}(X_i^{(k+1)})^\top Y, \tag{3.12}$$

which can be carried out distributively at each node with $O(np^2)$ floating-point operations without forming any $n \times n$ matrix.

For simplicity, let us take one iteration of the classical SSI method as an example. Starting from the current iterate $Z^{(k)} \in \mathcal{S}_{n,p}$ stored at every node, one computes

$$Z^{(k+1)} \in \mathbf{orth}(Q^{(k)}Z^{(k)}) = \mathbf{orth}\left(\sum_{i=1}^{d} Q_i^{(k)} Z^{(k)}\right).$$

Here, local products $Q_i^{(k)}Z^{(k)}$, $i = 1, \ldots, d$, are calculated distributively at all nodes via formula (3.12). In a fully connected network, the summation of these local products can be achieved by the all-reduce type of communication. More specifically, one can adopt the butterfly algorithm [36]. In this case, the communication overhead per iteration is $O\left(np\log(d)\right)$.

3.4 Algorithm description

We now formally present the proposed algorithmic framework as Algorithm 1 below, named *distributed ADMM with projection splitting* and abbreviated to DAPS. In DAPS, the i-th node first solves its subproblem for X_i and updates its multiplier Λ_i. These two steps only use local data privately stored at the i-th node, and hence can be carried out concurrently. Then, the nodes collaboratively solve the common subproblem for the global variable Z via a certain communication strategy. This procedure is repeated until convergence. Upon termination, the final iterate $Z^{(k)}$ is an orthonormal basis for an approximately optimal eigenspace of $AA^\top$ from which the desired dominant SVD can be easily calculated by an extra procedure.

Algorithm 1: Distributed ADMM with projection splitting (DAPS)

Input: data matrix $A = [A_1, \ldots, A_d]$, penalty parameters $\{\beta_i\}$.
Set $k := 0$. Initialize $(\{X_i^{(0)}\}, Z^{(0)})$ and compute $\{\Lambda_i^{(0)}\}$ by (3.7).
while *"not converged"* **do**
  for *all* $i \in \{1, 2, \ldots, d\}$ **do**
    Find $X_i^{(k+1)} \in \mathcal{S}_{n,p}$ that satisfies (3.5) and (3.6).
    Update the multipliers $\Lambda_i^{(k+1)}$ by (3.7).
  Find $Z^{(k+1)} \in \mathcal{S}_{n,p}$ that satisfies (3.11).
  Set $k := k + 1$.
Output: $Z^{(k)}$. If requested, compute the dominant SVD of A from $Z^{(k)}$.

3.5 Preservation of data privacy

As is mentioned in Subsection 3.3, at iteration k the shared information from the i-th node is in the form of products $Q_i^{(k)}Y \in \mathbb{R}^{n\times p}$, see (3.12). For the sake of argument, suppose that one could collect enough such products for a set of Y-matrices (which may not be feasible by itself). Then would it be possible to recover the local data matrix $A_iA_i^\top$?

As we can see from the expressions (3.12) and (3.8), to recover $A_iA_i^\top$ from $\{Q_i^{(k)}Y\}$, it is necessary to know local quantities $X_i^{(k+1)}$ and β_i, which are kept private at node i at any given time. Consequently, there is practically no chance for an outsider to recover $A_iA_i^\top$ from the publicly shared quantities $\{Q_i^{(k)}Y\}$. Therefore, it can be claimed that as a distributed algorithm DAPS is secure and able to preserve privacy of local data.

4 Convergence Analysis

In this section, we rigorously establish the global convergence of our proposed Algorithm 1 under the following mild assumptions on the algorithm parameters.

Assumption 1. *(i) The algorithm parameter δ_i in (3.6) satisfies*

$$0 \le \delta_i < \frac{\underline{\sigma}}{2\sqrt{\rho d}}, \quad i = 1, \ldots, d,$$

where $\rho := \max_{i,j=1,\ldots,d}\{\beta_i/\beta_j\} \ge 1$ and $\underline{\sigma} := \sqrt{1 - 1/(2\rho d)} \in (0, 1)$.
(ii) For a sufficiently large constant $\omega_i > 0$, the penalty parameter β_i satisfies

$$\beta_i \ge \omega_i \|A\|_{\mathrm{F}}^2, \quad i = 1, \ldots, d.$$

Remark 2. *The above assumptions are imposed only for the purpose of theoretical analysis. An expression for ω_i will be given in Appendix B.*

We are now ready to present the global convergence and the worst-case complexity of DAPS. For brevity, we use the following simplified notations.

$$\mathbf{d}_i^{(k)} := \mathbf{d}_{\mathbf{p}}\left(X_i^{(k)}, Z^{(k)}\right), \quad \text{for} \quad i = 1, \ldots, d, \quad \text{and} \quad k \in \mathbb{N}. \tag{4.1}$$

Theorem 4.1. *Let $X_i^{(0)} \in \mathcal{S}_{n,p}$ and $Z^{(0)} \in \mathcal{S}_{n,p}$ satisfy*

$$\left(\mathbf{d}_i^{(0)}\right)^2 \leq \frac{1}{\rho d}, \quad i = 1, \ldots, d, \tag{4.2}$$

and the sequence $\{\{X_i^{(k)}\}_{i=1}^d, Z^{(k)}\}$ be generated by Algorithm 1. Under Assumption 1, $\{Z^{(k)}\}$ has at least one accumulation point, and any accumulation point is a first-order stationary point of problem (2.1). Moreover, there exists a constant $C > 0$ so that for any $N > 1$, it holds that

$$\min_{k=0,\ldots,N-1} \left\{ \left\| \mathbf{P}_{Z^{(k)}}^{\perp} A A^\top Z^{(k)} \right\|_{\mathrm{F}}^2 + \frac{1}{d} \sum_{i=1}^{d} \left(\mathbf{d}_i^{(k)}\right)^2 \right\} \leq \frac{C}{N}.$$

The proof of this theorem, being quite long and tedious, is left to Appendix B.

5 Numerical Experiments

In this section, we evaluate the performance of DAPS through comprehensive numerical experiments, which demonstrate its efficiency, robustness, and scalability. All the experiments are performed on a high-performance computing cluster LSSC-IV[1] maintained at the State Key Laboratory of Scientific and Engineering Computing (LSEC), Chinese Academy of Sciences. There are 408 nodes in the main part of LSSC-IV, and each node consists of two Inter(R) Xeon(R) Gold 6140 processors (at 2.30GHz $\times$18) with 192GB memory. The operating system of LSSC-IV is Red Hat Enterprise Linux Server 7.3.

5.1 Test problems

Two classes of test problems are used in our experiments. The first class consists of synthetic problems randomly generated as follows. We construct a test matrix $A \in \mathbb{R}^{n\times m}$ (assuming $n \leq m$ without loss of generality) by its (economy-form) singular value decomposition

$$A = U\Sigma V^\top, \tag{5.1}$$

where both $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{m\times n}$ are orthonormalization of matrices whose entries are random numbers drawn independently, identically and uniformly from $[-1, 1]$, and $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix with diagonal entries

$$\Sigma_{ii} = \xi^{1-i}, \quad i = 1, \ldots, n, \tag{5.2}$$

for a parameter $\xi > 1$ that determines the decay rate of the singular values of A. In general, smaller decay rates (with ξ closer to 1) correspond to more difficult cases.

The second class of test problems consists of matrices from four popular image datasets frequently used in machine learning, including MNIST[2], Fashion-MNIST[3], CIFAR-10[4], and CIFAR-100[4]. In both MNIST and Fashion-MNIST, the sample dimension is $n = 28 \times 28 = 784$ and the number of sample is $m = 60000$; while in both CIFAR-10 and CIFAR-100, the sample dimension is $n = 3 \times 32 \times 32 = 3072$ with $m = 50000$.

[1] More information at http://lsec.cc.ac.cn/chinese/lsec/LSSC-IVintroduction.pdf
[2] Available from http://yann.lecun.com/exdb/mnist/
[3] Available from https://github.com/zalandoresearch/fashion-mnist
[4] Available from https://www.cs.toronto.edu/~kriz/cifar.html

5.2 Implementation details

We use an adaptive strategy to tune the penalty parameters β_i, in which we periodically increase the penalty parameters value when the projection distance has not seen a sufficient reduction. Given initial values $\beta_i^{(0)}$, at iteration $k > 0$ we first compute the projection distance $\mathbf{d}_i^{(k)}$ and then update the penalty parameter by the recursion rule:

$$\beta_i^{(k+1)} = \begin{cases} (1+\theta)\,\beta_i^{(k)}, & \text{if mod } (k,5) = 0 \text{ and } \mathbf{d}_i^{(k-5)} \leq (1+\mu)\,\mathbf{d}_i^{(k)}, \\ \beta_i^{(k)}, & \text{otherwise.} \end{cases} \tag{5.3}$$

Bt default, we set $\beta_i^{(0)} = 0.15\,\|A_i\|_2^2$, $\theta = 0.1$, and $\mu = 0.01$ in our implementation.

We initialize the global variable $Z^{(0)}$ as orthonormalization of a random $n \times p$ matrix whose entries follow the i.i.d. uniform distribution in $[-1, 1]$. Then we set $X_i^{(0)} = Z^{(0)}$ $(i = 1, \ldots, d)$.

For solving subproblem (3.3) approximately, we choose to use SLRPGN [31] which, at outer iteration k, generates an inner-iteration sequence $X_i^{(k)}(j)$ for $j = 0, 1, \ldots,$ with the warm-start $X_i^{(k)}(0) = X_i^{(k)}$. As is suggested in [31], we use the following termination rule:

$$\left| \left\| X_i^{(k)}(j) \right\|_{\mathrm{F}} - \left\| X_i^{(k)}(j-1) \right\|_{\mathrm{F}} \right| \leq \epsilon_x \left\| X_i^{(k)}(j) \right\|_{\mathrm{F}}, \tag{5.4}$$

for a prescribed tolerance $\epsilon_x > 0$, which measures the relative change between two consecutive inner iterates. According to the analysis in [31], if $X_i^{(k)}(j)$ satisfies (5.4) with a sufficiently small ϵ_x, then $X_i^{(k+1)} \in \mathbf{orth}(X_i^{(k)}(j))$ will satisfy conditions (3.5) and (3.6). In our experiments, we set $\epsilon_x = 10^{-2}$ as the default value. For solving subproblem (3.9) approximately, starting from $Z^{(k)}$ we take a single iteration of SLRPGN followed by orthonormalization to obtain $Z^{(k+1)}$. For numerical convenience, the matrices $H_i^{(k)}$ and $Q^{(k)}$ in subproblems (3.3) and (3.9) are scaled by the reciprocals of $\beta_i^{(k)}$ and $\sum_{i=1}^d \beta_i^{(0)}$, respectively.

We terminate DAPS if either the following condition holds,

$$\left| \sum_{i=1}^d \left\| A_i^\top Z^{(k)} \right\|_{\mathrm{F}}^2 - \sum_{i=1}^d \left\| A_i^\top Z^{(k-1)} \right\|_{\mathrm{F}}^2 \right| \leq 10^{-10} \sum_{i=1}^d \left\| A_i^\top Z^{(k)} \right\|_{\mathrm{F}}^2, \tag{5.5}$$

or the maximum iteration number `MaxIter` = 20000 is reached. The condition (5.5) measures the relative change in objective function values.

Remark 3. *We choose not to use the KKT violation as the stopping criterion since it requires extra communication overheads under a distributed environment.*

In our experiments, we collect and compare four performance measurements: wall-clock time, total number of iterations, scaled KKT violation defined by

$$\frac{1}{\|A\|_{\mathrm{F}}^2} \left\| \mathbf{P}_{Z^{(k)}}^{\perp} A A^\top Z^{(k)} \right\|_{\mathrm{F}},$$

and relative error in singular values defined by $\left\| \Sigma^{(k)} - \Sigma^* \right\|_{\mathrm{F}} / \left\| \Sigma^* \right\|_{\mathrm{F}}$, where the diagonal matrices $\Sigma^* \in \mathbb{R}^{p \times p}$ and $\Sigma^{(k)} \in \mathbb{R}^{p \times p}$ hold, respectively, the exact and computed dominant singular values.

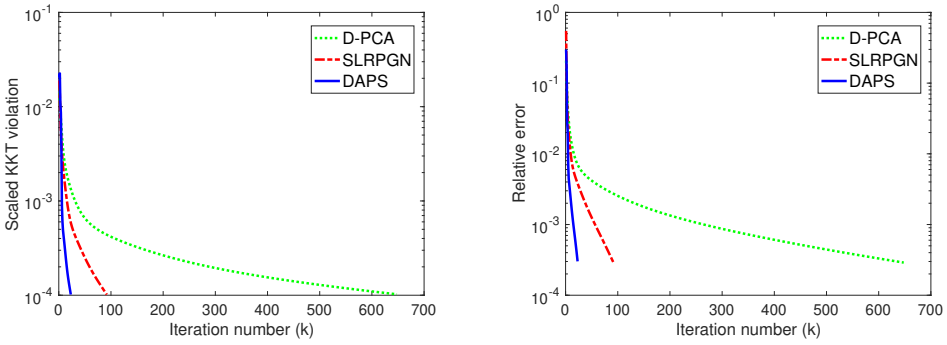
5.3 Competing algorithms

We compare the performances of DAPS mainly with two representative, state-of-the-art algorithms under the aforementioned distributed environment. The first competing algorithm is SLRPGN [31], which is a robust iterative eigensolvers developed in serial mode without any consideration of data security in distributed environments (see Subsection 1.3). The second competing algorithm is called D-PCA [40] (standing for distributed PCA). It is based on the framework of ADMM to solve a variable-splitting model derived from a low-rank matrix approximation problem. By design, it does preserve

data privacy in parallel mode (see Subsection 1.2). In our experiments, we always use default parameter settings for the three algorithms, and adopt the same initialization and stopping criterion as described in Subsection 5.2.

Since parallel versions of codes for these two algorithms are unavailable to us, as for DAPS we implemented them in C++ with MPI for inter-process communication to the best of our ability. In our implementation, we use the C++ linear algebra library *Eigen*[5] (version 3.3.8) for matrix computations. In particular, orthonormalization of an $n \times p$ matrix is done via the (economy-size) QR factorization at a cost of $O(np^2)$ operations. Moreover, SLRPGN and D-PCA require solving positive-definite linear systems which is realized by the class `LLT` in *Eigen*.

It has been proven in [31] that the algorithm SLRPGN has a linear rate of local convergence. Empirically, we have observed that all three algorithms appear to converge linearly, as is illustrated in Figure 1 where a synthetic matrix A, generated by (5.1) with $n = 1000$, $m = 160000$, and $\xi = 1.01$, is tested with $p = 10$ and $d = 128$. Not surprisingly, the convergence rate of DAPS is the fastest, followed by SLRPGN.



(a) Scaled KKT violation (b) Relative error

Figure 1: Comparison of empirical convergence rates

A more recent, decentralized algorithm called ADSA [15] was also tested. Although this algorithm may work well on matrices with fast-decaying singular values or when a low accuracy is sufficient, on our test problems it is clearly inferior to the other three algorithms, as is illustrated by the example given in Table 1 where a synthetic test matrix A, generated as in (5.1), is used. Consequently, we decide to exclude the ADSA algorithm from the rest of experimentations in this paper.

Algorithm	Wall-clock time (s)	Iteration	KKT violation	Relative error
ADSA	315.10	20000	1.61e-04	8.37e-04
D-PCA	94.13	5836	2.45e-06	1.69e-07
SLRPGN	5.95	381	3.77e-07	4.02e-09
DAPS	2.82	71	7.90e-08	9.08e-11

Table 1: Comparison of 4 algorithms with $n = 1000$, $m = 128000$, $p = 10$, $\xi = 1.01$, and $d = 128$

5.4 Parallel scalability

We first investigate parallel scalability of the three algorithms on synthetic test matrices generated as in (5.1), with $n = 1000$, $m = 512000$ and $\xi = 1.01$. The number of computed singular values is set to $p = 10$, and the minimum number of computing cores used in this experiment is $d = 16$.

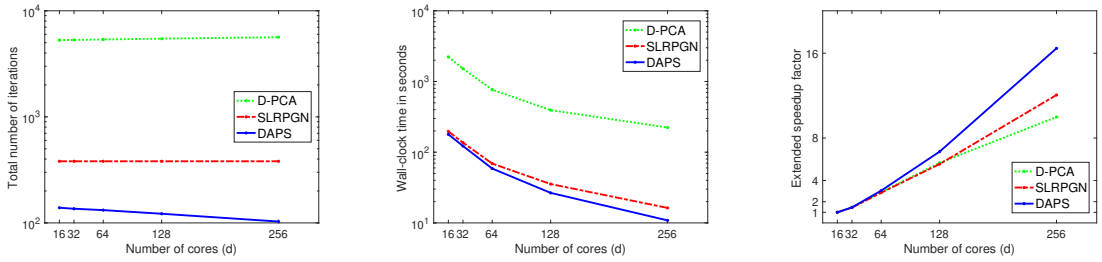
[5]Available from http://eigen.tuxfamily.org/index.php?title=Main_Page

The definition of *speedup factor* for running an algorithm on d cores is

$$\text{speedup-factor}(d) = \frac{\text{wall-clock time for a 16-core run}}{\text{wall-clock time for a } d\text{-core run}}, \quad d \geq 16,$$

which is upper-bounded by $d/16$ when traditional parallelization strategies are used. However, in our case we will call it the *extended speedup factor* since it is not necessarily upper-bounded by $d/16$ when we run DAPS or D-PCA. The reason is simple. Unlike SLRPGN which is parallelized at the linear algebra level, DAPS and D-PCA are parallelized at the algorithm level so that as d changes, the number of required iterations may also change, altering the pattern of change in wall-clock time.

We run the three algorithms with 16, 32, 64, 128, and 256 cores on LSSC-IV, and report numbers of iterations, wall-clock times, and extended speedup factors in Figure 2. In this experiment, all algorithms have reached a comparable level of accuracy.



(a) Total number of iterations (b) Wall-clock time in seconds (c) Extended speedup factor

Figure 2: Comparison of parallel scalability

We observe that, in this particular experiment, as d increases, the number of iterations required by DAPS decreases (so does the wall-clock time as expected), and the extended speedup factor for DAPS eventually breaks the traditional upper bound of $d/16 = 16$ at $d = 256$. On the other hand, the number of iterations for D-PCA slightly increases as d increases, and hence the extended speedup factor goes down below the corresponding values for SLRPGN (which is upper-bounded by $d/16$). In the present paper, due to space limitation we will not further study this issue of algorithm-level parallelism, but leave it to a future investigation instead.

5.5 Comprehensive comparison on synthetic data

We now compare the performances of the three algorithms on a variety of synthetic test problems, run under the afore-mentioned distributed environment with the number of computing cores fixed at $d = 128$. We construct four groups of test problems based on (5.1), in each of which there is only one parameter varying while all others are fixed. Specifically, the problem parameter settings for A are given as follows (recall that n is the number of rows, m is the number of columns, p is the number of singular values to be computed, and ξ determines the decay rate of singular values):

(1) $n = 1000 + 1000j$ for $j = 1, 2, 3, 4$, while $m = 128000$, $p = 20$, and $\xi = 1.01$;

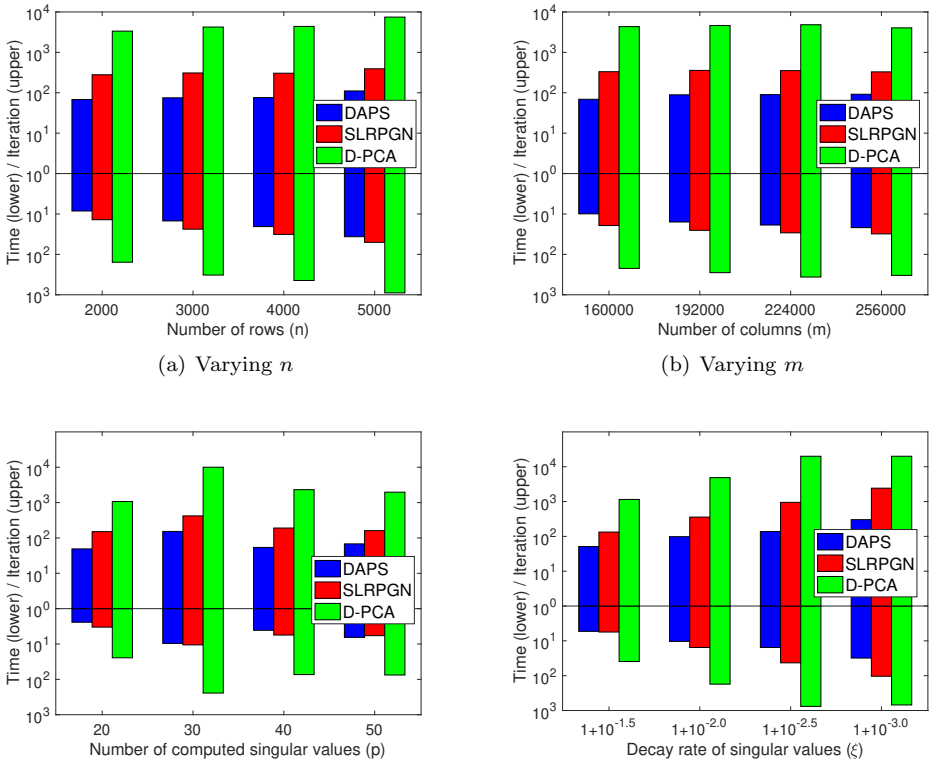
(2) $m = 128000 + 32000j$ for $j = 1, 2, 3, 4$, while $n = 2000$, $p = 10$, and $\xi = 1.01$;

(3) $p = 10 + 10j$ for $j = 1, 2, 3, 4$, while $n = 1000$, $m = 128000$, and $\xi = 1.01$;

(4) $\xi = 1 + 10^{-1-j/2}$ for $j = 1, 2, 3, 4$, while $n = 1000$, $m = 256000$, and $p = 10$.

The numerical results for the above four test scenarios are depicted in Figure 3, with two quantities, wall-clock time in seconds and number of iterations taken, recorded on a logarithmic scale for every experiment. The average scaled KKT violation and relative error of every experiment are tabulated in Table 2. It should be evident from these numerical results that DAPS clearly outperforms SLRPGN which in turn always outperforms D-PCA, in terms of both wall-clock time and iteration number. In

particular, we observe from Figure 3(d) that the advantage of DAPS is more pronounced on problems that are harder to solve (with slower decay in singular values). In some cases, the number of iterations required by DAPS is about one order of magnitude smaller than that by SLRPGN, and two order of magnitude smaller than that by D-PCA (noting that the maximum number of iterations is capped at 20,000).



(a) Varying n (b) Varying m

(c) Varying p (d) Varying ξ

Figure 3: Comparison of D-PCA, SLRPGN, and DAPS on synthetic data

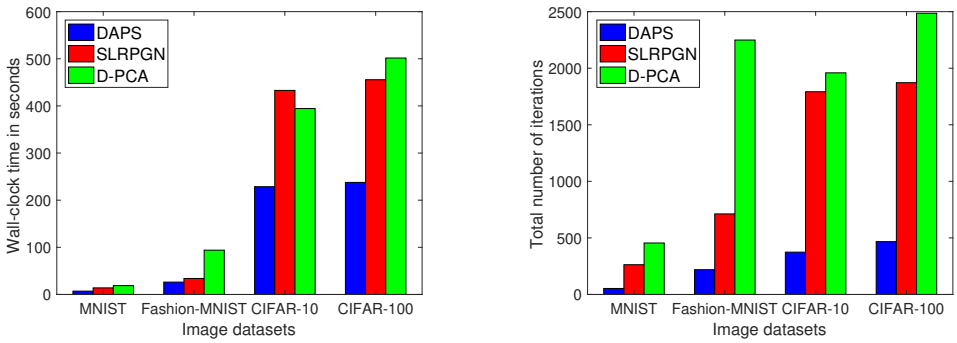
	Average scaled KKT violation			Average relative error		
	D-PCA	SLRPGN	DAPS	D-PCA	SLRPGN	DAPS
Varying n	2.98e-06	4.58e-07	1.89e-07	2.49e-07	5.91e-09	9.90e-10
Varying m	2.26e-06	3.76e-07	1.50e-07	2.21e-07	3.98e-09	6.40e-10
Varying p	2.96e-06	5.41e-07	2.36e-07	2.21e-07	8.13e-09	1.80e-09
Varying ξ	2.76e-06	3.43e-07	1.81e-07	2.88e-05	1.44e-08	1.04e-08

Table 2: Average errors of D-PCA, SLRPGN, and DAPS on synthetic data

It is worth emphasizing that since these three algorithms incur more or less the same amount of communication overhead per iteration, the total amount of information exchanged is roughly proportional to the numbers of iterations. Hence, the rapid convergence of DAPS (in terms of iteration number) translates into not only computational but also communicational efficiency. On the other hand, we caution that the advantage of DAPS may not always be as large as shown in our experiments when some parameter values go beyond the tested ranges .

5.6 Comparison on image datasets

We next evaluate the performances of the three algorithms on four image datasets popular in machine learning research. The numbers of computed singular values and computing cores in use are set to $p = 5$ and $d = 16$, respectively. Numerical results from this experiment are given in Figure 4 and Table 3. Again, in terms of both wall-clock time and number of iterations taken but especially the latter, DAPS always dominates SLRPGN which in turn outperforms D-PCA. These results indicate that the observed superior performance of DAPS is not just limited to synthetic matrices.



(a) Wall-clock time in seconds

(b) Total number of iterations

Figure 4: Comparison of D-PCA, SLRPGN, and DAPS on four image datasets

Average scaled KKT violation			Average relative error		
D-PCA	SLRPGN	DAPS	D-PCA	SLRPGN	DAPS
8.14e-06	3.23e-06	1.81e-06	2.16e-07	4.18e-08	1.13e-08

Table 3: Average errors of D-PCA, SLRPGN, and DAPS on four image datasets

5.7 Discussions on communication overhead

In this subsection, we take a closer look at the communication overhead of SLRPGN and DAPS (D-PCA is not included due to its non-competitiveness in previous experiments) on a synthetic test matrix A generated as in (5.1) with $n = 2000$, $m = 128000$, and $\xi = 1.01$. The numbers of computed singular values is set to $p = 10$, and the number of computing cores is $d = 128$. We list both the wall-clock time and the communication time consumed in Table 4. We observe that, to reach a comparable accuracy, the wall-clock time of SLRPGN is about 1.8 times that of DAPS, while for communication time the ratio is about 4.5 times, which is roughly equal to the ratio between the numbers of iterations taken by the two algorithms.

Algorithm	Wall-clock (s)	Communication (s)	Iteration	Relative error
SLRPGN	17.40	0.12042	326	4.01e-09
DAPS	9.82	0.02649	77	4.57e-11

Table 4: Comparison of SLRPGN and DAPS on timing

Since this experiment is done on a cluster computer where computing cores are tightly and fully connected, the resulting communication cost is extremely low. In fact, the cores share information through the all-reduce type of communication among each other with the cost of $O(np\log(d))$ operations at each iteration for computing a global summation of d matrices of size $n \times p$. In real-world applications, however, data may be distributed over a wide geographical range connected by a very

sparse network topology. For instance, on a linear array network the communication cost would become $O(npd)$ for computing the same global summation [6].

In the above experiment, the communication time ratio between SLRPGN and DAPS is about 4.5 times. We could infer the following scenario: as communication costs become more and more expensive to the extent that they dominate computational costs, then the wall-clock time ratio between SLRPGN and DAPS would eventually approach 4.5 times.

Finally, we note that by decreasing ϵ_x in (5.4), the tolerance for solving the local-variable subproblems, one can reduce the number of iterations in most cases at the cost of spending more time on solving the local subproblems. This strategy would become worthwhile when communication costs sufficiently dominate computing times.

6 Conclusions

Computing dominant SVD of large-scale matrices in distributed or parallel environments requires a balance of at least two goals: reducing communication overhead and achieving high efficiency in both time and space. In many modern applications, preserving data privacy is a third goal with an even higher priority. Towards achieving these three goals, we propose a novel projection splitting model and construct a distributed ADMM algorithm, called DAPS, that seeks consensus among subspaces rather than basis matrices. The projection splitting strategy also simplifies the subproblems in DAPS, giving rise to standard symmetric eigenvalue problems for which well-studied iterative algorithms and solvers exist. DAPS can maintain data privacy and is equipped with two key features: (i) multipliers are represented by a closed-form, low-rank formula; and (ii) solution accuracies for subproblems are appropriately controlled at fairly low levels. Together, these features make DAPS not only secure but also highly efficient.

Most existing works on analyzing the convergence of ADMM-based approaches for solving nonconvex optimization problems impose assumptions on iterates or multipliers. In our specific case for DAPS which solves a nonconvex optimization model, we are able to derive global convergence and worst-case complexity results under rather mild assumptions on algorithm parameters only.

We have implemented DAPS, along with two competing algorithms, in C++ with Message Passing Interface (MPI). Comprehensive numerical comparisons are carried out on a high-performance computing cluster with test results strongly in favor of DAPS. In particular, DAPS decisively dominates another recently proposed ADMM-type algorithm D-PCA. It also clearly outperforms a parallelized version of the state-of-the-art algorithm SLRPGN that lacks the ability to preserve data privacy. Most notably, the number of iterations required by DAPS is significantly fewer than that required by others up to one or two orders of magnitudes. This fast empirical convergence rate is made possible by the projection splitting strategy, representing the greatest advantage of DAPS.

Finally, we mention that there still remains a range of issues, theoretical or practical, to be further studied in order to fully understand the behavior and realize the potential of DAPS and its variants. We also note that the projection splitting idea can be generalized to a wider class of problems.

Appendices

A Proof of the existence of low-rank multipliers

In this part, we prove Proposition 2.1 to interpret the existence of low-rank multipliers associated with the subspace constraints in (2.5).

Proof of Proposition 2.1. We start with proving the "only if" part, and hence assume that Z is a first-order stationary point of (2.1). Let $\Theta = 0$, $\Gamma_i = -X_i^\top A_i A_i^\top X_i$, and $\Lambda_i = -\mathbf{P}_{X_i}^\perp A_i A_i^\top X_i X_i^\top - X_i X_i^\top A_i A_i^\top \mathbf{P}_{X_i}^\perp$ with $i = 1, \ldots, d$. Then matrices Θ, Γ_i and Λ_i are symmetric and $\operatorname{rank}(\Lambda_i) \le 2p$. And it can be readily verified that

$$A_i A_i^\top X_i + X_i \Gamma_i + \Lambda_i X_i = \mathbf{P}_{X_i}^\perp A_i A_i^\top X_i - \mathbf{P}_{X_i}^\perp A_i A_i^\top X_i = 0, \quad i = 1, \ldots, d.$$

Moreover, it follows from the fact $X_iX_i^\top = ZZ^\top$ and stationarity of Z that

$$\sum_{i=1}^d \Lambda_i Z - Z\Theta = \sum_{i=1}^d \left(-\mathbf{P}_{X_i}^\perp A_i A_i^\top X_i X_i^\top - X_i X_i^\top A_i A_i^\top \mathbf{P}_{X_i}^\perp\right) Z = -\mathbf{P}_Z^\perp A A^\top Z = 0.$$

Hence, $(\{X_i\}, Z)$ satisfies the condition (2.6) under the specific combination of Θ, Γ_i, and Λ_i.

Now we prove the "if" part and assume that there exist symmetric matrices Θ, Γ_i, and Λ_i such that the feasible point $(\{X_i\}, Z)$ satisfies the condition (2.6). By virtue of (2.6), we obtain $A_iA_i^\top X_i = -X_i\Gamma_i - \Lambda_i X_i$, and hence it holds that

$$\sum_{i=1}^d \mathbf{P}_{X_i}^\perp A_i A_i^\top X_i X_i^\top = -\sum_{i=1}^d \mathbf{P}_{X_i}^\perp \left(X_i\Gamma_i + \Lambda_i X_i\right) X_i^\top = -\mathbf{P}_Z^\perp \left(\sum_{i=1}^d \Lambda_i Z\right) Z^\top = 0,$$

where the second equality follows from the fact $X_iX_i^\top = ZZ^\top$, and the third equality follows from (2.6). On the other hand, we have

$$\sum_{i=1}^d \mathbf{P}_{X_i}^\perp A_i A_i^\top X_i X_i^\top = \sum_{i=1}^d \mathbf{P}_Z^\perp A_i A_i^\top ZZ^\top = \mathbf{P}_Z^\perp A A^\top ZZ^\top.$$

Combining the above two relationships, we arrive at $\mathbf{P}_Z^\perp AA^\top ZZ^\top = 0$, which further implies $\mathbf{P}_Z^\perp AA^\top Z = 0$. Therefore, Z is a first-order stationary point of (2.1). We complete the proof. □

B Proof of the global convergence

In this part, we prove Theorem 4.1 to establish the global convergence of Algorithm 1. To begin with, we give an explicit expression of the constant $\omega_i > 0$ in Assumption 1 as follows:

$$\omega_i = \max\left\{c_1', \ \frac{12\rho d\sqrt{p}}{c_1\underline{\sigma}^2}, \ \frac{4\sqrt{2}\left(1+\sqrt{2\rho d}\right)}{\underline{\sigma} - 2\sqrt{\rho d}\delta_i}, \ 16\rho d\sqrt{p}, \ \frac{4(1+\sqrt{2\rho d})}{c_1\underline{\sigma}^2\rho d}\right\}, \ \ i = 1, \ldots, d.$$

In addition, it is clear that $\|A\|_{\mathrm{F}} \geq \|A_i\|_{\mathrm{F}} \geq \|A_i\|_2$.

Next, in order to prove Theorem 4.1, we establish a few lemmas and corollaries to make preparations. In their proofs, we omit the superscript (k) to save space with a slight abuse of notations, and use the superscript $+$ to take the place of $(k+1)$.

Lemma B.1. *Suppose Assumption 1 holds and $(\{X_i^{(k)}\}, Z^{(k)})$ is the k-th iterate generated by Algorithm 1 and satisfies that $\mathbf{d}_i^{(k)} \leq \sqrt{1/(\rho d)}$, $i = 1, \ldots, d$. Then it holds that*

$$h_i^{(k)}(X_i^{(k)}) - h_i^{(k)}(X_i^{(k+1)}) \geq \frac{1}{4} c_1 \underline{\sigma}^2 \beta_i \left(\mathbf{d}_i^{(k)}\right)^2, \quad i = 1, \ldots, d.$$

Proof. It follows from Assumption 1 that $\beta_i > c_1' \|A\|_{\mathrm{F}}^2 \geq c_1' \|A_i\|_2^2$, which together with (3.5) yields that

$$h_i(X_i) - h_i(X_i^+) \geq \frac{c_1}{2\beta_i} \left\|\mathbf{P}_{X_i}^\perp H_i X_i\right\|_{\mathrm{F}}^2. \tag{B.1}$$

According to the definition of H_i and Λ_i, we have

$$\mathbf{P}_{X_i}^\perp H_i X_i = \mathbf{P}_{X_i}^\perp \left(A_i A_i^\top + \Lambda_i + \beta_i ZZ^\top\right) X_i = \beta_i \mathbf{P}_{X_i}^\perp ZZ^\top X_i. \tag{B.2}$$

Suppose $\hat{\sigma}_1, \ldots, \hat{\sigma}_p$ are the singular values of $X_i^\top Z$. It is clear that $0 \leq \hat{\sigma}_i \leq 1$ and $\mathbf{d}_i^2 = 2\sum_{j=1}^p \left(1 - \hat{\sigma}_j^2\right)$ for any $i = 1, \ldots, d$. By simple calculations, we have

$$\left\|\mathbf{P}_{X_i}^\perp ZZ^\top X_i\right\|_{\mathrm{F}}^2 = \mathrm{tr}\left(X_i^\top ZZ^\top X_i\right) - \mathrm{tr}\left(\left(X_i^\top ZZ^\top X_i\right)^2\right) = \sum_{j=1}^p \hat{\sigma}_j^2\left(1 - \hat{\sigma}_j^2\right).$$

Moreover, it follows from $\mathbf{d}_i^{(k)} \leq \sqrt{1/(\rho d)}$ that $\sigma_{\min}(X_i^\top Z) \geq \underline{\sigma}$, which implies that

$$\left\|\mathbf{P}_{X_i}^{\perp} Z Z^\top X_i\right\|_{\mathrm{F}}^2 = \sum_{j=1}^{p} \hat{\sigma}_j^2 \left(1 - \hat{\sigma}_j^2\right) \geq \underline{\sigma}^2 \sum_{j=1}^{p} \left(1 - \hat{\sigma}_j^2\right) = \frac{1}{2}\underline{\sigma}^2 \mathbf{d}_i^2.$$

This together with (B.1) and (B.2) completes the proof. □

Lemma B.2. *Suppose all the conditions in Lemma B.1 hold. Then for any $i = 1, \ldots, d$, we have*

$$\mathbf{d}_{\mathbf{p}}^2\left(X_i^{(k+1)}, Z^{(k)}\right) \leq \left(1 - c_1\underline{\sigma}^2\right)\left(\mathbf{d}_i^{(k)}\right)^2 + \frac{12}{\beta_i}\sqrt{p}\left\|A_i\right\|_{\mathrm{F}}^2.$$

Proof. According to Lemma B.1 and definitions of h_i and H_i, we can acquire

$$\frac{1}{2}\mathrm{tr}\left(ZZ^\top \mathbf{D}_{\mathbf{p}}\left(X_i^+, X_i\right)\right) + \frac{1}{2\beta_i}\mathrm{tr}\left(\left(A_i A_i^\top + \Lambda_i\right)\mathbf{D}_{\mathbf{p}}\left(X_i^+, X_i\right)\right) \geq \frac{1}{4}c_1\underline{\sigma}^2\mathbf{d}_i^2.$$

By straightforward calculations, we can further obtain the following two relationships

$$\begin{aligned}\mathrm{tr}\left(\left(A_i A_i^\top + \Lambda_i\right)\mathbf{D}_{\mathbf{p}}\left(X_i^+, X_i\right)\right) &\leq \left\|A_i A_i^\top + \Lambda_i\right\|_{\mathrm{F}} \mathbf{d}_{\mathbf{p}}\left(X_i^+, X_i\right) \leq 6\sqrt{p}\left\|A_i\right\|_{\mathrm{F}}^2, \\ \mathrm{tr}\left(ZZ^\top \mathbf{D}_{\mathbf{p}}\left(X_i^+, X_i\right)\right) &= \frac{1}{2}\mathbf{d}_i^2 - \frac{1}{2}\mathbf{d}_{\mathbf{p}}^2\left(X_i^+, Z\right).\end{aligned}$$

Combining the above three relationships, we complete the proof. □

Lemma B.3. *Suppose $\{\{X_i^{(k)}\}, Z^{(k)}\}$ is the iterate sequence generated by Algorithm 1. Then the inequality*

$$\left(\mathbf{d}_i^{(k+1)}\right)^2 \leq \rho\sum_{j=1}^{d}\mathbf{d}_{\mathbf{p}}^2\left(X_j^{(k+1)}, Z^{(k)}\right) + \frac{8\sqrt{p}}{\beta_i}\left\|A\right\|_{\mathrm{F}}^2$$

holds for $i = 1, \ldots, d$ and $k \in \mathbb{N}$.

Proof. The inequality (3.11) directly results in the relationship $q(Z) - q(Z^+) \geq 0$, which yields that

$$0 \leq \sum_{j=1}^{d}\beta_j \mathrm{tr}\left(X_j^+ (X_j^+)^\top \mathbf{D}_{\mathbf{p}}\left(Z^+, Z\right)\right) + \sum_{j=1}^{d}\mathrm{tr}\left(\Lambda_j^+ \mathbf{D}_{\mathbf{p}}\left(Z, Z^+\right)\right).$$

By straightforward calculations, we can deduce the following two relationships

$$\begin{aligned}\mathrm{tr}\left(\Lambda_j^+ \mathbf{D}_{\mathbf{p}}\left(Z, Z^+\right)\right) &\leq \left\|\Lambda_j^+\right\|_{\mathrm{F}} \mathbf{d}_{\mathbf{p}}\left(Z, Z^+\right) \leq 4\sqrt{p}\left\|A_j\right\|_{\mathrm{F}}^2, \\ \mathrm{tr}\left(X_j^+ (X_j^+)^\top \mathbf{D}_{\mathbf{p}}\left(Z^+, Z\right)\right) &= \frac{1}{2}\mathbf{d}_{\mathbf{p}}^2\left(X_j^+, Z\right) - \frac{1}{2}\left(\mathbf{d}_j^+\right)^2,\end{aligned}$$

which implies that

$$\sum_{j=1}^{d}\beta_j\left(\mathbf{d}_j^+\right)^2 \leq \sum_{j=1}^{d}\beta_j\mathbf{d}_{\mathbf{p}}^2\left(X_j^+, Z\right) + 8\sqrt{p}\left\|A\right\|_{\mathrm{F}}^2.$$

Now it can be readily verified that

$$\left(\mathbf{d}_i^+\right)^2 \leq \frac{1}{\beta_i}\sum_{j=1}^{d}\beta_j\left(\mathbf{d}_j^+\right)^2 \leq \rho\sum_{j=1}^{d}\mathbf{d}_{\mathbf{p}}^2\left(X_j^+, Z\right) + \frac{8\sqrt{p}}{\beta_i}\left\|A\right\|_{\mathrm{F}}^2.$$

This completes the proof. □

Lemma B.4. *Let $\Phi_i(Y) = -\mathbf{P}_Y^{\perp} A_i A_i^\top Y Y^\top - Y Y^\top A_i A_i^\top \mathbf{P}_Y^{\perp}$ for any $Y \in \mathcal{S}_{n,p}$ and $i = 1, \ldots, d$. Then for any $Y_1 \in \mathcal{S}_{n,p}$ and $Y_2 \in \mathcal{S}_{n,p}$, it holds that*

$$\left\|\Phi_i(Y_1) - \Phi_i(Y_2)\right\|_{\mathrm{F}} \leq 4\left\|A_i\right\|_2^2 \mathbf{d}_{\mathbf{p}}\left(Y_1, Y_2\right), \quad i = 1, \ldots, d.$$

Proof. This lemma directly follows from the triangular inequality. Hence, its proof is omitted. □

Lemma B.5. *Suppose Assumption 1 holds, and* $\{\{X_i^{(k)}\}, Z^{(k)}\}$ *is the iterate sequence generated by Algorithm 1 initiated from* $(\{X_i^{(0)}\}, Z^{(0)})$ *satisfying* (4.2). *Then for* $k \in \mathbb{N}$, *it holds that*

$$\left(\mathbf{d}_i^{(k)}\right)^2 \leq \frac{1}{\rho d}, \quad i = 1, \ldots, d. \tag{B.3}$$

Proof. We use mathematical induction to prove this lemma. The argument (B.3) directly holds at $\{\mathbf{d}_i^{(0)}\}_{i=1}^d$ resulting from (4.2). Now, we assume the argument holds at $\{\mathbf{d}_i\}_{i=1}^d$, and investigate the situation at $\{\mathbf{d}_i^+\}_{i=1}^d$.

According to Assumption 1, we have $\beta_i > 12\rho d\sqrt{p}\,\|A_i\|_{\mathrm{F}}^2/(c_1\underline{\sigma}^2)$. Without loss of generality, we assume that $c_1\underline{\sigma}^2 < 1$. Combining Lemma B.2 and (B.3), we can derive that

$$\mathbf{d}_{\mathbf{p}}^2\left(X_i^+, Z\right) \leq \frac{1 - c_1\underline{\sigma}^2}{\rho d} + \frac{c_1\underline{\sigma}^2}{\rho d} = \frac{1}{\rho d},$$

which infers that $\sigma_{\min}\left((X_i^+)^\top Z\right) \geq \underline{\sigma}$. Similar to the proof of Lemma B.1, we can deduce that

$$\left\|\mathbf{P}_{X_i^+}^{\perp} Z Z^\top X_i^+\right\|_{\mathrm{F}}^2 \geq \frac{\underline{\sigma}^2}{2}\mathbf{d}_{\mathbf{p}}^2\left(X_i^+, Z\right). \tag{B.4}$$

Together with condition (3.5) and equality (B.2), we have

$$\left\|\mathbf{P}_{X_i^+}^{\perp} H_i X_i^+\right\|_{\mathrm{F}} \leq \delta_i\beta_i \left\|\mathbf{P}_{X_i}^{\perp} Z Z^\top X_i\right\|_{\mathrm{F}} \leq \delta_i\beta_i\mathbf{d}_i.$$

On the other hand, it follows from the triangular inequality that

$$\left\|\mathbf{P}_{X_i^+}^{\perp} H_i X_i^+\right\|_{\mathrm{F}} \geq \left\|\mathbf{P}_{X_i^+}^{\perp}\left(A_i A_i^\top + \Lambda_i^+ + \beta_i Z Z^\top\right) X_i^+\right\|_{\mathrm{F}} - \left\|\mathbf{P}_{X_i^+}^{\perp}\left(\Lambda_i^+ - \Lambda_i\right) X_i^+\right\|_{\mathrm{F}}.$$

It follows from the inequality (B.4) that

$$\left\|\mathbf{P}_{X_i^+}^{\perp}\left(A_i A_i^\top + \Lambda_i^+ + \beta_i Z Z^\top\right) X_i^+\right\|_{\mathrm{F}} = \beta_i \left\|\mathbf{P}_{X_i^+}^{\perp} Z Z^\top X_i^+\right\|_{\mathrm{F}} \geq \frac{\sqrt{2}}{2}\underline{\sigma}\beta_i\mathbf{d}_{\mathbf{p}}\left(X_i^+, Z\right).$$

According to Lemma B.4, we have

$$\left\|\mathbf{P}_{X_i^+}^{\perp}\left(\Lambda_i^+ - \Lambda_i\right) X_i^+\right\|_{\mathrm{F}} \leq 4\left\|A_i\right\|_2^2 \mathbf{d}_{\mathbf{p}}\left(X_i^+, X_i\right) \leq 4\left\|A_i\right\|_2^2\left(\mathbf{d}_{\mathbf{p}}\left(X_i^+, Z\right) + \mathbf{d}_i\right).$$

Combing the above four inequalities, we further obtain that

$$(\sqrt{2}\underline{\sigma}\beta_i/2 - 4\left\|A_i\right\|_2^2)\mathbf{d}_{\mathbf{p}}\left(X_i^+, Z\right) \leq (\delta_i\beta_i + 4\left\|A_i\right\|_2^2)\mathbf{d}_i.$$

According to Assumption 1, we have $0 \leq \delta_i < \underline{\sigma}/\sqrt{4\rho d}$, and

$$\beta_i > \frac{4\sqrt{2}\left(1 + \sqrt{2\rho d}\right)}{\underline{\sigma} - 2\sqrt{\rho d}\delta_i}\left\|A_i\right\|_2^2 \geq \frac{4\sqrt{2}}{\underline{\sigma}}\left\|A_i\right\|_2^2.$$

Thus, we arrive at

$$\mathbf{d}_{\mathbf{p}}\left(X_i^+, Z\right) \leq \frac{2(\delta_i\beta_i + 4\left\|A_i\right\|_2^2)}{\sqrt{2}\underline{\sigma}\beta_i - 8\left\|A_i\right\|_2^2}\mathbf{d}_i \leq \sqrt{\frac{1}{2\rho d}}\mathbf{d}_i, \quad i = 1, \ldots, d. \tag{B.5}$$

Again, we have $\beta_i > 16\rho d\sqrt{p}\,\|A\|_{\mathrm{F}}^2$ according to Assumption 1. Combing Lemma B.3 and (B.3), we further acquire that

$$\left(\mathbf{d}_i^+\right)^2 \leq \rho\sum_{j=1}^d \mathbf{d}_{\mathbf{p}}^2\left(X_j^+, Z\right) + \frac{8\sqrt{p}}{\beta_i}\left\|A\right\|_{\mathrm{F}}^2 \leq \frac{1}{2d}\sum_{j=1}^d \mathbf{d}_j^2 + \frac{1}{2\rho d} \leq \frac{1}{\rho d},$$

which completes the proof. □

Corollary B.6. *Suppose all the conditions in Lemma B.5 hold. Then for any $k \in \mathbb{N}$, there holds*

$$\mathcal{L}(\{X_i^{(k)}\}, Z^{(k)}, \{\Lambda_i^{(k)}\}) - \mathcal{L}(\{X_i^{(k+1)}\}, Z^{(k)}, \{\Lambda_i^{(k)}\}) \geq \frac{1}{4} c_1 \underline{\sigma}^2 \sum_{i=1}^{d} \beta_i \left(\mathbf{d}_i^{(k)}\right)^2.$$

Proof. This corollary directly follows from Lemmas B.1 and B.5. □

Corollary B.7. *Suppose all the conditions in Lemma B.5 hold. Then for any $k \in \mathbb{N}$, it holds that*

$$\mathcal{L}(\{X_i^{(k+1)}\}, Z^{(k)}, \{\Lambda_i^{(k)}\}) - \mathcal{L}(\{X_i^{(k+1)}\}, Z^{(k)}, \{\Lambda_i^{(k+1)}\}) \geq -\frac{1+\sqrt{2\rho d}}{\rho d} \sum_{i=1}^{d} \|A_i\|_2^2 \left(\mathbf{d}_i^{(k)}\right)^2.$$

Proof. According to the Cauchy-Schwarz inequality, we can deduce that

$$\begin{aligned} &\left|\left\langle \Lambda_i^+ - \Lambda_i, \mathbf{D}_{\mathbf{p}}\left(X_i^+, Z\right)\right\rangle\right| = \left|\left\langle \Phi_i(X_i^+) - \Phi_i(X_i), \mathbf{D}_{\mathbf{p}}\left(X_i^+, Z\right)\right\rangle\right| \\ &\leq \left\|\Phi_i(X_i^+) - \Phi_i(X_i)\right\|_{\mathrm{F}} \mathbf{d}_{\mathbf{p}}\left(X_i^+, Z\right) \leq \sqrt{\frac{8}{\rho d}} \|A_i\|_2^2 \, \mathbf{d}_{\mathbf{p}}\left(X_i^+, X_i\right) \mathbf{d}_i, \end{aligned}$$

where the last inequality follows from Lemma B.4 and (B.5). Moreover, we have

$$\mathbf{d}_{\mathbf{p}}\left(X_i^+, X_i\right) \leq \mathbf{d}_{\mathbf{p}}\left(X_i^+, Z\right) + \mathbf{d}_i \leq \frac{1+\sqrt{2\rho d}}{\sqrt{2\rho d}} \mathbf{d}_i,$$

which further yields that

$$\left\langle \Lambda_i^+ - \Lambda_i, \mathbf{D}_{\mathbf{p}}\left(X_i^+, Z\right)\right\rangle \geq -\frac{2(1+\sqrt{2\rho d})}{\rho d} \|A_i\|_2^2 \, \mathbf{d}_i^2.$$

Combing the fact that

$$\mathcal{L}_i(X_i^+, Z, \Lambda_i) - \mathcal{L}_i(X_i^+, Z, \Lambda_i^+) = \frac{1}{2}\left\langle \Lambda_i^+ - \Lambda_i, \mathbf{D}_{\mathbf{p}}\left(X_i^+, Z\right)\right\rangle,$$

we complete the proof. □

Corollary B.8. *Suppose $\{\{X_i^{(k)}\}, Z^{(k)}\}$ is the iterate sequence generated by Algorithm 1. Let $\bar{Q}^{(k)} = \sum_{i=1}^{d}\left(\beta_i X_i^{(k+1)}(X_i^{(k+1)})^\top + \Phi_i(Z^{(k)}) - \Phi_i(X_i^{(k+1)})\right)$, and $G^{(k)} = \mathbf{P}_{Z^{(k)}}^{\perp} A A^\top Z^{(k)} + \mathbf{P}_{Z^{(k)}}^{\perp} \bar{Q}^{(k)} Z^{(k)}$. Then for any $k \in \mathbb{N}$, it holds that*

$$\mathcal{L}(\{X_i^{(k+1)}\}, Z^{(k)}, \{\Lambda_i^{(k+1)}\}) - \mathcal{L}(\{X_i^{(k+1)}\}, Z^{(k+1)}, \{\Lambda_i^{(k+1)}\}) \geq c_2 \left\|G^{(k)}\right\|_{\mathrm{F}}^2.$$

Proof. Recalling the definitions of Q and $\Phi_i(Z)$, we obtain that

$$\mathbf{P}_Z^{\perp} Q Z = \mathbf{P}_Z^{\perp} \bar{Q} Z - \sum_{i=1}^{d} \mathbf{P}_Z^{\perp} \Phi_i(Z) Z = \mathbf{P}_Z^{\perp} \bar{Q} Z + \mathbf{P}_Z^{\perp} A A^\top Z = G.$$

This together with (3.11) completes the proof. □

Next we show the monotonicity of the sequence of augmented Lagrangian function values $\{\mathcal{L}^{(k)}\}$ where $\mathcal{L}^{(k)} = \mathcal{L}(\{X_i^{(k)}\}, Z^{(k)}, \{\Lambda_i^{(k)}\})$.

Proposition B.9. *Suppose $\{\{X_i^{(k)}\}, Z^{(k)}\}$ is the iterate sequence generated by Algorithm 1 initiated from $(\{X_i^{(0)}\}, Z^{(0)})$ satisfying (4.2), and problem parameters satisfy Assumption 1. Then the sequence $\{\mathcal{L}^{(k)}\}$ is monotonically non-increasing and, for any $k \in \mathbb{N}$, satisfies the following two conditions:*

$$\mathcal{L}^{(k)} - \mathcal{L}^{(k+1)} \geq \sum_{i=1}^{d} \left(\frac{1}{4} c_1 \underline{\sigma}^2 \beta_i - \frac{1+\sqrt{2\rho d}}{\rho d} \|A_i\|_2^2\right) \left(\mathbf{d}_i^{(k)}\right)^2 + c_2 \left\|G^{(k)}\right\|_{\mathrm{F}}^2, \tag{B.6}$$

and

$$\mathcal{L}^{(k)} - \mathcal{L}^{(k+1)} \geq c_3 \left\|\mathbf{P}_{Z^{(k)}}^{\perp} A A^\top Z^{(k)}\right\|_{\mathrm{F}}^2, \tag{B.7}$$

where $c_3 > 0$ is a constant.

Proof. Combining Corollaries B.6 to B.8, we can easily verify the inequality (B.6). Recalling the condition $\beta_i > 4(1+\sqrt{2\rho d})\left\|A_i\right\|_2^2/(c_1\underline{\sigma}^2\rho d)$ in Assumption 1, we can conclude that $\mathcal{L}-\mathcal{L}^+ \geq 0$. Hence, the sequence $\{\mathcal{L}^{(k)}\}$ is monotonically non-increasing. It directly follows from the definition of $\bar{Q}$ that

$$\mathbf{P}_Z^{\perp}\bar{Q}Z = \mathbf{P}_Z^{\perp}\sum_{i=1}^{d}\left(\beta_i \mathbf{D}_{\mathbf{p}}\left(X_i^+, Z\right) + \Phi_i(Z) - \Phi_i(X_i^+)\right)Z.$$

Together with the triangular inequality and (B.5), we can obtain that

$$\left\|\mathbf{P}_Z^{\perp}\bar{Q}Z\right\|_{\mathrm{F}} \leq \sum_{i=1}^{d}(\beta_i \mathbf{d}_{\mathbf{p}}\left(X_i^+, Z\right) + \left\|\Phi_i(X_i^+) - \Phi_i(Z)\right\|_{\mathrm{F}}) \leq \sqrt{\frac{1}{2\rho d}}\sum_{i=1}^{d}(\beta_i + 4\left\|A_i\right\|_2^2)\mathbf{d}_i.$$

And we define a constant $c_4 := \min_{i=1,\ldots,d}\left\{c_1\underline{\sigma}^2\beta_i/4 - (1+\sqrt{2\rho d})\left\|A_i\right\|_2^2/(\rho d)\right\} > 0$. It can be readily verified that

$$\left\|\mathbf{P}_Z^{\perp}AA^{\top}Z\right\|_{\mathrm{F}} = \left\|G - \mathbf{P}_Z^{\perp}\bar{Q}Z\right\|_{\mathrm{F}} \leq \left\|G\right\|_{\mathrm{F}} + \left\|\mathbf{P}_Z^{\perp}\bar{Q}Z\right\|_{\mathrm{F}} \leq \sqrt{(\mathcal{L}-\mathcal{L}^+)/c_3},$$

where $c_3 := \left(\sum_{i=1}^{d}(\beta_i + 4\left\|A_i\right\|_2^2)/\sqrt{2\rho d c_4} + \sqrt{1/c_2}\right)^{-2} > 0$ is a constant, and the last inequality follows from the facts that, for $i = 1, \ldots, d$,

$$\left\|G\right\|_{\mathrm{F}} \leq \sqrt{(\mathcal{L}-\mathcal{L}^+)/c_2} \quad \text{and} \quad \mathbf{d}_i \leq \sqrt{(\mathcal{L}-\mathcal{L}^+)/c_4}. \tag{B.8}$$

We complete the proof. □

We are now ready to present the proof of Theorem 4.1.

Proof of Theorem 4.1. Since each of $X_i^{(k)}$ or $Z^{(k)}$ is orthonormal for any $i = 1, \ldots, d$ and $k \in \mathbb{N}$, the whole sequence $\{\{X_i^{(k)}\}, Z^{(k)}\}$ is naturally bounded. Then, it follows from the Bolzano-Weierstrass theorem that this sequence exists an accumulation point $(\{X_i^*\}, Z^*)$, where $X_i^* \in \mathcal{S}_{n,p}$ and $Z^* \in \mathcal{S}_{n,p}$. In addition, the boundedness of $\{\Lambda_i^{(k)}\}$ results from the multipliers updating formula (3.7). Hence, the lower boundedness of $\{\mathcal{L}^{(k)}\}$ is owing to the continuity of the augmented Lagrangian function. Namely, there exists a constant $\underline{L}$ such that $\mathcal{L}^{(k)} \geq \underline{L}$ holds for all $k \in \mathbb{N}$.

Let $R^{(k)} = \mathbf{P}_{Z^{(k)}}^{\perp}AA^{\top}Z^{(k)}$. It follows from (B.7) and (B.8) that there hold

$$\sum_{k=0}^{N-1}\left\|R^{(k)}\right\|_{\mathrm{F}}^2 \leq \frac{1}{c_3}\sum_{k=0}^{N-1}\left(\mathcal{L}^{(k)} - \mathcal{L}^{(k+1)}\right) \leq \frac{1}{c_3}\left(\mathcal{L}^{(0)} - \underline{L}\right) \tag{B.9}$$

and

$$\sum_{k=0}^{N-1}\sum_{i=1}^{d}\left(\mathbf{d}_i^{(k)}\right)^2 \leq \frac{d}{c_4}\sum_{k=0}^{N-1}\left(\mathcal{L}^{(k)} - \mathcal{L}^{(k+1)}\right) \leq \frac{d}{c_4}\left(\mathcal{L}^{(0)} - \underline{L}\right). \tag{B.10}$$

Taking the limit as $N \to \infty$ on the both sides of (B.9) and (B.10), we obtain that

$$\sum_{k=0}^{\infty}\left\|R^{(k)}\right\|_{\mathrm{F}}^2 < \infty \quad \text{and} \quad \sum_{k=0}^{\infty}\sum_{i=1}^{d}\left(\mathbf{d}_i^{(k)}\right)^2 < \infty,$$

which further imply

$$\lim_{k\to\infty}\left\|R^{(k)}\right\|_{\mathrm{F}} = 0 \quad \text{and} \quad \lim_{k\to\infty}\sum_{i=1}^{d}\mathbf{d}_i^{(k)} = 0.$$

Hence, it holds at any limit point that $\mathbf{P}_{Z^*}^{\perp}AA^{\top}Z^* = 0$ and $X_i^*(X_i^*)^{\top} = Z^*(Z^*)^{\top}$, for $i = 1, \ldots, d$. Therefore, Z^* is a first-order stationary point of the problem (2.1). Finally, it follows from the inequalities (B.9) and (B.10) that

$$\min_{k=0,\ldots,N-1}\left\{\left\|R^{(k)}\right\|_{\mathrm{F}}^2 + \frac{1}{d}\sum_{i=1}^{d}\left(\mathbf{d}_i^{(k)}\right)^2\right\} \leq \frac{1}{N}\sum_{k=0}^{N-1}\left\{\left\|R^{(k)}\right\|_{\mathrm{F}}^2 + \frac{1}{d}\sum_{i=1}^{d}\left(\mathbf{d}_i^{(k)}\right)^2\right\} \leq \frac{C}{N},$$

where $C = (\mathcal{L}^{(0)} - \underline{L})(1/c_3 + 1/c_4) > 0$ is a constant. The proof is completed. □

References

[1] M. Aharon, M. Elad, and A. Bruckstein, *K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation*, IEEE Trans. Signal Process., 54 (2006), pp. 4311–4322.

[2] H. Andrews and C. Patterson, *Singular value decomposition (SVD) image coding*, IEEE Trans. Commun., 24 (1976), pp. 425–432.

[3] W. E. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Q. Appl. Math., 9 (1951), pp. 17–29.

[4] 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., 3 (2011), pp. 1–122.

[5] E. J. Candès and B. Recht, *Exact matrix completion via convex optimization*, Found. Comput. Math., 9 (2009), p. 717.

[6] E. Chan, M. Heimlich, A. Purkayastha, and R. Van De Geijn, *Collective communication: theory, practice, and experience*, Concurr. Comput.-Pract. E., 19 (2007), pp. 1749–1783.

[7] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, *Indexing by latent semantic analysis*, J. Amer. Soc. Inf. Sci., 41 (1990), pp. 391–407.

[8] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, and I. Yamazaki, *The singular value decomposition: anatomy of optimizing an algorithm for extreme scale*, SIAM Rev., 60 (2018), pp. 808–865.

[9] J. Eckstein and P. J. Silva, *A practical relative error criterion for augmented Lagrangians*, Math. Program., 141 (2013), pp. 319–348.

[10] T. Elgamal, M. Yabandeh, A. Aboulnaga, W. Mustafa, and M. Hefeeda, *sPCA: Scalable principal component analysis for big data on distributed platforms*, in Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery, 2015, pp. 79–91.

[11] J. Fan, D. Wang, K. Wang, and Z. Zhu, *Distributed estimation of principal eigenspaces*, Ann. Statist., 47 (2019), pp. 3009–3031.

[12] H.-R. Fang and Y. Saad, *A filtered Lanczos procedure for extreme and interior eigenvalue problems*, SIAM J. Sci. Comput., 34 (2012), pp. A2220–A2246.

[13] J. Fellus, D. Picard, and P.-H. Gosselin, *Asynchronous gossip principal components analysis*, Neurocomputing, 169 (2015), pp. 262–271.

[14] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 137–158.

[15] A. Gang, H. Raja, and W. U. Bajwa, *Fast and communication-efficient distributed PCA*, in Proceedings of the 2019 International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 7450–7454.

[16] B. Gao, X. Liu, X. Chen, and Y.-X. Yuan, *A new first-order algorithmic framework for optimization problems with orthogonality constraints*, SIAM J. Optim., 28 (2018), pp. 302–332.

[17] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the bidiagonal SVD*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 79–92.

[18] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[19] C. Himpe, T. Leibner, and S. Rave, *Hierarchical approximate proper orthogonal decomposition*, SIAM J. Sci. Comput., 40 (2018), pp. A3267–A3292.

[20] M. A. Iwen and B. W. Ong, *A distributed and incremental SVD algorithm for agglomerative data analysis on large networks*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1699–1718.

[21] E. R. Jessup and D. C. Sorensen, *A parallel algorithm for computing the singular value decomposition of a matrix*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 530–548.

[22] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.

[23] A. V. Knyazev, M. E. Argentati, I. Lashuk, and E. E. Ovtchinnikov, *Block locally optimal preconditioned eigenvalue xolvers (BLOPEX) in HYPRE and PETSc*, SIAM J. Sci. Comput., 29 (2007), pp. 2224–2239.

[24] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Stand., 45 (1950).

[25] R. Lazcano, D. Madroñal, H. Fabelo, S. Ortega, R. Salvador, G. M. Callicó, E. Juarez, and C. Sanz, *Adaptation of an iterative PCA to a manycore architecture for hyperspectral image processing*, J. Signal Process. Syst., 91 (2019), pp. 759–771.

[26] R. B. Lehoucq, *Implicitly restarted Arnoldi methods and subspace iteration*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 551–562.

[27] H. Li, Y. Kluger, and M. Tygert, *Randomized algorithms for distributed computation of principal component analysis and singular value decomposition*, Adv. Comput. Math., 44 (2018), pp. 1651–1672.

[28] L. Li, A. Scaglione, and J. H. Manton, *Distributed principal subspace estimation in wireless sensor networks*, IEEE J. Sel. Top. Signal Process., 5 (2011), pp. 725–738.

[29] Y. Liang, M.-F. Balcan, V. Kanchanapally, and D. P. Woodruff, *Improved distributed principal component analysis*, in Proceedings of the 2014 Advances in Neural Information Processing Systems, 2014, pp. 3113–3121.

[30] X. Liu, Z. Wen, and Y. Zhang, *Limited memory block Krylov subspace optimization for computing dominant singular value decompositions*, SIAM J. Sci. Comput., 35 (2013), pp. A1641–A1668.

[31] ——, *An efficient Gauss–Newton algorithm for symmetric low-rank product matrix approximations*, SIAM J. Optim., 25 (2015), pp. 1571–1608.

[32] Y.-F. Liu, X. Liu, and S. Ma, *On the nonergodic convergence rate of an inexact augmented Lagrangian framework for composite convex programming*, Math. Oper. Res., 44 (2019), pp. 632–650.

[33] Y. Lou, L. Yu, S. Wang, and P. Yi, *Privacy preservation in distributed subgradient optimization algorithms*, IEEE Trans. Cybernet., 48 (2017), pp. 2154–2165.

[34] B. Moore, *Principal component analysis in linear systems: controllability, observability, and model reduction*, IEEE Trans. Automat. Control, 26 (1981), pp. 17–32.

[35] G. Morral, P. Bianchi, and J. Jakubowicz, *Asynchronous distributed principal component analysis using stochastic approximation*, in Proceedings of the 51st IEEE Conference on Decision and Control (CDC), IEEE, 2012, pp. 1398–1403.

[36] P. S. Pacheco, *An introduction to parallel programming*, Elsevier, 2011.

[37] H. Raja and W. U. Bajwa, *Cloud K-SVD: A collaborative dictionary learning algorithm for big, distributed data*, IEEE Trans. Signal Process., 64 (2015), pp. 173–188.

[38] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math. Oper. Res., 1 (1976), pp. 97–116.

[39] H. Rutishauser, *Simultaneous iteration method for symmetric matrices*, Numer. Math., 16 (1970), pp. 205–223.

[40] I. D. Schizas and A. Aduroja, *A distributed framework for dimensionality reduction and denoising*, IEEE Trans. Signal Process., 63 (2015), pp. 6379–6394.

[41] S. Shang-Guan and J. Yin, *A Fast Distributed Principal Component Analysis with Variance Reduction*, in Proceedings of the 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), IEEE, 2017, pp. 11–14.

[42] G. L. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM Rev., 42 (2000), pp. 267–293.

[43] A. Stathopoulos and C. F. Fischer, *A Davidson program for finding a few selected extreme eigenpairs of a large, sparse, real, symmetric matrix*, Comput. Phys. Commun., 79 (1994), pp. 268–290.

[44] G. W. Stewart, *Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices*, Numer. Math., 25 (1976), pp. 123–136.

[45] W. J. Stewart and A. Jennings, *A simultaneous iteration algorithm for real matrices*, ACM Trans. Math. Software, 7 (1981), pp. 184–198.

[46] E. Stiefel, *Richtungsfelder und fernparallelismus in n-dimensionalen mannigfaltigkeiten*, Comment. Math. Helv., 8 (1935), pp. 305–353.

[47] W. Suleiman, M. Pesavento, and A. M. Zoubir, *Performance analysis of the decentralized eigendecomposition and ESPRIT algorithm*, IEEE Trans. Signal Process., 64 (2016), pp. 2375–2386.

[48] F. Tisseur and J. Dongarra, *A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures*, SIAM J. Sci. Comput., 20 (1999), pp. 2223–2236.

[49] M. A. Turk and A. P. Pentland, *Face recognition using eigenfaces*, in Proceedings of the 1991 Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 1991, pp. 586–591.

[50] Z. Wen, C. Yang, X. Liu, and Y. Zhang, *Trace-penalty minimization for large-scale eigenspace computation*, J. Sci. Comput., 66 (2016), pp. 1175–1203.

[51] S. X. Wu, H.-T. Wai, A. Scaglione, and N. A. Jacklin, *The Power-Oja method for decentralized subspace estimation/tracking*, in Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017, pp. 3524–3528.

[52] C. Zhang, M. Ahmad, and Y. Wang, *ADMM based privacy-preserving decentralized optimization*, IEEE Trans. Inf. Foren. Secur., 14 (2018), pp. 565–580.