http://pubsonline.informs.org/journal/ijoo

INFORMS JOURNAL ON OPTIMIZATION
Vol. 2, No. 3, Summer 2020, pp. 192–208
ISSN 2575-1484 (print), ISSN 2575-1492 (online)

An Alternating Manifold Proximal Gradient Method for Sparse Principal Component Analysis and Sparse Canonical Correlation Analysis

Shixiang Chen,[a] Shiqian Ma,[b] Lingzhou Xue,[c] Hui Zou[d]

[a] Department of Industrial and Systems Engineering, Texas A&M University, College Station, Texas 77843-3131; [b] Department of Mathematics, University of California, Davis, Davis, California 95616; [c] Department of Statistics, Pennsylvania State University, University Park, Pennsylvania 16802; [d] School of Statistics, University of Minnesota, Minneapolis, Minnesota 55455

Contact: sxchen@tamu.edu, http://orcid.org/0000-0002-3261-0714 (SC); sqma@ucdavis.edu, http://orcid.org/0000-0003-1967-1069 (SM); lingzhou@psu.edu, http://orcid.org/0000-0002-8252-0637 (LX); zouxx019@umn.edu, http://orcid.org/0000-0003-4798-9904 (HZ)

Received: March 27, 2019
Revised: August 29, 2019; December 4, 2019
Accepted: March 25, 2020
Published Online in Articles in Advance: July 24, 2020

https://doi.org/10.1287/ijoo.2019.0032

Abstract. Sparse principal component analysis and sparse canonical correlation analysis are two essential techniques from high-dimensional statistics and machine learning for analyzing large-scale data. Both problems can be formulated as an optimization problem with nonsmooth objective and nonconvex constraints. Because nonsmoothness and nonconvexity bring numerical difficulties, most algorithms suggested in the literature either solve some relaxations of them or are heuristic and lack convergence guarantees. In this paper, we propose a new alternating manifold proximal gradient method to solve these two high-dimensional problems and provide a unified convergence analysis. Numerical experimental results are reported to demonstrate the advantages of our algorithm.

Funding: S. Ma was supported in part by the National Science Foundation [Grant DMS-1953210] and a startup package in the Department of Mathematics at the University of California, Davis. L. Xue was supported in part by the National Science Foundation [Grants DMS-1811552, DMS-1953189 and NIDA-P50DA039838]. H. Zou was supported in part by the National Science Foundation [Grant DMS-1915842].
Supplemental Material: The online appendices are available at https://doi.org/10.1287/ijoo.2019.0032.

Keywords: manifold optimization • sparse principal component analysis • sparse canonical correlation analysis

1. Introduction

In this paper, we consider two important problems arising from high-dimensional statistics and machine learning: sparse principal component analysis (PCA) and sparse canonical correlation analysis (CCA). Both problems are special cases of the following more general manifold optimization problem:

$$\min F(A,B) := H(A,B) + f(A) + g(B) \quad \text{subject to (s.t.)} \quad A \in \mathcal{M}_1, B \in \mathcal{M}_2, \tag{1}$$

where $H(A,B)$ is a smooth function of A,B with a Lipschitz continuous gradient, $f(\cdot)$ and $g(\cdot)$ are lower semicontinuous (possibly nonsmooth) convex functions, and $\mathcal{M}_1, \mathcal{M}_2$ are two embedded submanifolds in the Euclidean space. Problem (1) is numerically challenging because of the combination of nonsmoothness and nonconvexity.

1.1. Our Contributions

The contributions of this paper lie in several folds. First, we propose a new alternating manifold proximal gradient method for solving (1) and establish its convergence guarantee. Second, we show how to apply the proposed algorithm to solve sparse PCA and sparse CCA. Third, we conduct extensive numerical experiments to compare the performance of the proposed algorithm and existing methods for solving these two problems. Fourth, we analyze the convergence guarantee of an inexact (and more practical) version of the proposed method.

PCA was invented by Pearson (1901) and is widely used in dimension reduction. Let $X = [X_1, \ldots, X_p] \in \mathbb{R}^{n\times p}$ be a given data matrix whose column means are all zero. Assume that the singular-value decomposition (SVD) of X is $X = UDV^\top$. Then it is known that $Z = UD$ are the principal components (PCs), and the columns of V are the corresponding loadings of the PCs. In other words, the first PC can be defined as $Z_1 = \sum_{j=1}^{p} \alpha_{1j} X_j$ with $\alpha_1 = (\alpha_{11}, \ldots, \alpha_{1p})^\top$ maximizing the variance of Z_1; that is,

$$\alpha_1 = \arg\max_{\alpha} \alpha^\top \hat{\Sigma} \alpha \quad \text{s.t.} \quad \|\alpha_1\|_2 = 1,$$

where $\hat{\Sigma} = (X^\top X)/(n-1)$ is the sample covariance matrix. The remaining PCs are defined as

$$\alpha_{k+1} = \arg\max_{\alpha} \alpha^\top \hat{\Sigma}\alpha \quad \text{s.t.} \quad \|\alpha\|_2 = 1, \alpha^\top \alpha_1 = 0, \forall 1 \le l \le k.$$

CCA, introduced by Hotelling (1936), is another widely used tool that explores the relation between two sets of variables. For random variables $x \in \mathbb{R}^p$ and $y \in \mathbb{R}^q$, CCA seeks linear combinations of x and y such that the resulting values are mostly correlated. That is, it targets to solve the following optimization problem:

$$\max_{u\in\mathbb{R}^p, v\in\mathbb{R}^q} \frac{u^\top \Sigma_{xy} v}{\sqrt{u^\top \Sigma_x u}\sqrt{v^\top \Sigma_y v}}, \tag{2}$$

where Σ_x and Σ_y are variances of x and y, Σ_{xy} is their covariance matrix, and $u \in \mathbb{R}^p$ and $v \in \mathbb{R}^q$ are the first canonical vectors. It can be shown that solving (2) corresponds to computing the SVD of $\Sigma_x^{-1/2}\Sigma_{xy}\Sigma_y^{-1/2}$. In practice, given two centered data sets $X \in \mathbb{R}^{n\times p}, Y \in \mathbb{R}^{n\times q}$ with a joint covariance matrix $[\Sigma_x, \Sigma_{xy}; \Sigma_{yx}, \Sigma_y]$, CCA seeks the coefficients u, v such that the correlation of Xu and Yv is maximized. The classical CCA (Hotelling 1936) can be formulated as

$$\max_{u\in\mathbb{R}^p, v\in\mathbb{R}^q} \quad u^\top X^\top Y v \quad \text{s.t.} \quad u^\top X^\top X u = 1,\ v^\top Y^\top Y v = 1, \tag{3}$$

where $X^\top Y, X^\top X, Y^\top Y$ are used to estimate the true parameters $\Sigma_{xy}, \Sigma_x, \Sigma_y$ after scaling.

However, PCA and CCA perform poorly and often lead to wrong findings when modeling with high-dimensional data. For example, when the dimension is proportional to the sample size such that $\lim_{n\to\infty} p/n = \gamma \in (0,1)$ and the largest eigenvalue $\lambda_1 \le \sqrt{\gamma}$, the leading sample principal eigenvector could be asymptotically orthogonal to the leading population principal eigenvector under a multivariate Gaussian model (Paul 2007) or a spiked covariance model with a single component and finite fourth moment (Nadler 2008). The orthogonality result is built on the almost sure limits of sample eigenvalues (Baik and Silverstein 2006). Sparse PCA and sparse CCA are proposed as the more interpretable and reliable dimension-reduction and feature-extraction techniques for high-dimensional data. In what follows, we provide a brief overview of their methodological developments.

Sparse PCA seeks a sparse basis (loadings) of the subspace spanned by the data so that the leading PCs obtained are easier to interpret. Jolliffe et al. (2003) proposed the SCoTLASS procedure (Simplified Component Technique-LASSO) by imposing an ℓ_1 norm on the loading vectors, which can be formulated as the following optimization problem for given data $X \in \mathbb{R}^{n\times p}$:

$$\min_{A\in\mathbb{R}^{p\times r}} \quad -\mathrm{Tr}\big(A^\top X^\top X A\big) + \mu\|A\|_1 \quad \text{s.t.} \quad A \in \mathrm{St}(p,r), \tag{4}$$

where $\mathrm{Tr}(Z)$ denotes the trace of matrix Z, $\mu > 0$ is a weighting parameter, $\|A\|_1 = \sum_{ij}|A_{ij}|$, and $\mathrm{St}(p,r) := \{A \in \mathbb{R}^{p\times r} : A^\top A = I_r\}$ is the Stiefel manifold, where I_r is the $r \times r$ identity matrix. Note that the original SCoTLASS model in Jolliffe et al. (2003) uses an ℓ_1 constraint $\|A\|_1 \le t$ instead of penalizing $\|A\|_1$ in the objective. The SCoTLASS model (4) is numerically challenging. Algorithms for solving it have been limited. As a result, a new formulation of sparse PCA has been proposed by Zou et al. (2006), and it has been the main focus in the literature on this topic. Zou et al. (2006) formulate the sparse PCA problem as the following ridge regression problem plus a LASSO penalty:

$$\min_{A\in\mathbb{R}^{p\times r}, B\in\mathbb{R}^{p\times r}} \quad \sum_{i=1}^{n}\big\|\mathbf{x}_i - AB^\top \mathbf{x}_i\big\|_2^2 + \mu\sum_{j=1}^{r}\|B_j\|_2^2 + \sum_{j=1}^{r}\mu_{1,j}\|B_j\|_1 \quad \text{s.t.} \quad \mathrm{A} \in \mathrm{St}(p,r), \tag{5}$$

where $\mathbf{x}_i$ denotes the transpose of the ith row vector of X, B_j is the jth column vector of B, and $\mu > 0$ and $\mu_{1,j} > 0$ are weighting parameters. Given $A \in \mathrm{St}(p,r)$, we have the following:

$$\sum_{i=1}^{n}\|\mathbf{x}_i - AB^\top \mathbf{x}_i\|_2^2 = \|X - XBA^\top\|_F^2 = \mathrm{Tr}\big(X^\top X\big) + \mathrm{Tr}\big(B^\top X^\top X B\big) - 2\mathrm{Tr}\big(A^\top X^\top X B\big).$$

Therefore, (5) can be equivalently written as follows:

$$\min_{A\in\mathbb{R}^{p\times r},B\in\mathbb{R}^{p\times r}} \quad H(A,B)+\mu\sum_{j=1}^{r}\|B_j\|_2^2+\sum_{j=1}^{r}\mu_{1,j}\|B_j\|_1 \quad \text{s.t.} \quad A\in \mathrm{St}(p,r), \tag{6}$$

where

$$H(A,B)=\mathrm{Tr}\big(B^\top X^\top XB\big)-2\mathrm{Tr}\big(A^\top X^\top XB\big).$$

The sparse PCA (6) is in the form of (1) with $f(A)\equiv 0$, $g(B)=\mu\sum_{j=1}^{r}\|B_j\|_2^2+\sum_{j=1}^{r}\mu_{1,j}\|B_j\|_1$, $\mathcal{M}_1=\mathrm{St}(p,r)$, and $\mathcal{M}_2=\mathbb{R}^{p\times r}$. However, it should be noted that (6) is indeed still numerically challenging. Zou et al. (2006) proposed to solve it using an alternating minimization algorithm (AMA) that updates A and B alternatingly with the other variable fixed as the current iterate. A typical iteration of AMA is as follows:

$$\begin{aligned} A^{k+1} &:= \arg\min_{A\in\mathbb{R}^{p\times r}} H\big(A,B^k\big) \quad \text{s.t.} \quad A\in\mathrm{St}\big(p,r\big)\\ B^{k+1} &:= \arg\min_{B\in\mathbb{R}^{p\times r}} H\big(A^{k+1},B\big)+\mu\sum_{j=1}^{r}\|B_j\|_2^2+\sum_{j=1}^{r}\mu_{1,j}\|B_j\|_1. \end{aligned} \tag{7}$$

The A-subproblem in (7) is known as a *Procrustes rotation problem* and has a closed-form solution given by a SVD. The B-subproblem in (7) is a linear regression problem with an elastic net regularizer, and it can be solved by many existing solvers such as elastic net (Zou and Hastie 2005), coordinate descent (Friedman et al. 2010), and the fast iterative shrinkage-thresholding algorithm (FISTA; Beck and Teboulle 2009). However, there is no convergence guarantee of AMA (7). Recently, some new algorithms are proposed in the literature that can solve (6) with guarantees of convergence to a stationary point. We provide a summary of some representative algorithms in the next section.

We need to point out that there are other ways to formulate sparse PCA such as the ones in d'Aspremont et al. (2007, 2008), Ma (2013), Lu and Zhang (2012), Vu et al. (2013), d'Aspremont (2011), Shen and Huang (2008), Witten et al. (2009), Journee et al. (2010), Yuan and Zhang (2013), and Moghaddam et al. (2005). We refer interested readers to the recent survey paper by Zou and Xue (2018) for more details on these works on sparse PCA. In this paper, we focus on the formulation of (6) to estimate multiple PCs, which is a manifold optimization problem with nonsmooth objective function.

Sparse CCA (Wiesel et al. 2008, Parkhomenko et al. 2009, Witten et al. 2009, Hardoon and Shawe-Taylor 2011) is proposed to improve the interpretability of CCA, which can be formulated as follows:

$$\min_{u\in\mathbb{R}^p,v\in\mathbb{R}^q} \quad -u^\top X^\top Yv+f(u)+g(v) \quad \text{s.t.} \quad u^\top X^\top Xu=1,\ v^\top Y^\top Yv=1, \tag{8}$$

where $X\in\mathbb{R}^{n\times p}$, $Y\in\mathbb{R}^{n\times q}$, f, and g are regularization terms promoting the sparsity of u and v, and common choices for them include the ℓ_1 norm for sparsity and the $\ell_{2,1}$ norm for group sparsity. When multiple canonical vectors are needed, one can consider the matrix counterpart of (8) that can be formulated as follows:

$$\min_{A\in\mathbb{R}^{p\times r},B\in\mathbb{R}^{q\times r}} \quad -\mathrm{Tr}\big(A^\top X^\top YB\big)+f(A)+g(B) \quad \text{s.t.} \quad A^\top X^\top XA=I_r,\ B^\top Y^\top YB=I_r, \tag{9}$$

where r is the number of canonical vectors needed. From now on, we call (8) the *single sparse CCA model* and (9) the *multiple sparse CCA model*. Moreover, motivated by Gao et al. (2017), in this paper we choose f and g to be the $\ell_{2,1}$ norm to promote the group sparsity of A and B in (9). Specifically, we choose $f(A)=\tau_1\|A\|_{2,1}$ and $g(B)=\tau_2\|B\|_{2,1}$, where the $\ell_{2,1}$ norm is defined as $\|A\|_{2,1}=\sum_{j=1}^{p}\|A_{j\cdot}\|_2$, and $A_{j\cdot}$ denotes the jth row vector of matrix A, and $\tau_1>0$ and $\tau_2>0$ are weighting parameters. In this case, the multiple sparse CCA (9) reduces to

$$\min_{A\in\mathbb{R}^{p\times r},B\in\mathbb{R}^{q\times r}} -\mathrm{Tr}\big(A^\top X^\top YB\big)+\tau_1\|A\|_{2,1}+\tau_2\|B\|_{2,1} \quad \text{s.t. } A^\top X^\top XA=I_r,\ B^\top Y^\top YB=I_r, \tag{10}$$

which is in the form of (1) with $H(A,B)=-\mathrm{Tr}(A^\top X^\top YB)$, $\mathcal{M}_1=\{A\mid A^\top X^\top XA=I_r\}$, and $\mathcal{M}_2=\{B\mid B^\top Y^\top YB=I_r\}$. Note that in (8) and (9) we assumed that both $X^\top X$ and $Y^\top Y$ are positive definite to guarantee that $\mathcal{M}_1$ and $\mathcal{M}_2$ are submanifolds. If they are not positive definite, we can always add a small perturbation to make them so. These manifolds used in (8) and (9) are generalized Stiefel manifolds. Note that when $r=1$, the $\ell_{2,1}$ norm becomes the ℓ_1 norm of the vector because $\|u\|_{2,1}=\|u\|_1$ for any $u\in\mathbb{R}^p$. In this case, (8) reduces to

$$\min_{u\in\mathbb{R}^p,v\in\mathbb{R}^q} \quad -u^\top X^\top Yv+\tau_1\|u\|_1+\tau_2\|v\|_1 \quad \text{s.t.} \quad u^\top X^\top Xu=1,\ v^\top Y^\top Yv=1. \tag{11}$$

Manifold optimization has recently drawn a lot of research attention because of its success in a variety of important applications, including low-rank matrix completion (Boumal and Absil 2011, Vandereycken 2013), phase retrieval (Bendory et al. 2018, Sun et al. 2018), phase synchronization (Boumal 2016, Liu et al. 2017), blind deconvolution (Huang and Hand 2018), and dictionary learning (Cherian and Sra 2017, Sun et al. 2017). Most existing algorithms for solving manifold optimization problems rely on the smoothness of the objective (see the recent monograph by Absil et al. (2009)). Studies on manifold optimization problems with nonsmooth objectives such as (6), (10), and (11) have been limited. This urges us to study efficient algorithms that solve manifold optimization problems with nonsmooth objectives, and this is the main focus of this paper.

The rest of this paper is organized as follows. We review existing methods for sparse PCA and sparse CCA in Section 2. We propose a unified alternating manifold proximal gradient method with provable convergence guarantees for solving both sparse PCA and sparse CCA in Section 3. The numerical performance is demonstrated in Section 4. We provide preliminaries on manifold optimization and details of the global convergence analysis of our proposed method in the online appendix. We also discuss the convergence result of an inexact version of our algorithm in the online appendix.

2. Existing Methods

Before proceeding, we review existing methods for solving sparse PCA (6) and sparse CCA (8) and (9).

2.1. Existing Methods for Sparse PCA

For sparse PCA (6), other than the AMA algorithm suggested in Zou et al. (2006), there exist some other efficient algorithms for solving this problem. We now give a brief review of these works. We first introduce two powerful optimization algorithms for solving nonconvex problems: the proximal alternating minimization (PAM) algorithm (Attouch et al. 2010) and the proximal alternating linearization method (PALM) (Bolte et al. 2014). Surprisingly, it seems that these two methods have not been used to solve (6) yet. We now briefly describe how these two methods can be used to solve (6). PAM for (6) solves the following two subproblems in each iteration:

$$A_{k+1} := \arg\min_{A} H(A, B_k) + \frac{1}{2t_1}\|A - A_k\|_F^2 \quad \text{s.t.} \quad A \in \mathrm{St}(p, r), \tag{12}$$

$$B_{k+1} := \arg\min_{B} H(A_{k+1}, B) + \mu \sum_{j=1}^{r} \|B_j\|_2^2 + \sum_{j=1}^{r} \mu_{1,j}\|B_j\|_1 + \frac{1}{2t_2}\|B - B_k\|_F^2, \tag{13}$$

where $t_1 > 0$, $t_2 > 0$ are step sizes. Note that in each subproblem, PAM minimizes the objective function with respect to one variable by fixing the other, and a proximal term is added for the purpose of convergence guarantee. It is shown in Attouch et al. (2010) that the sequence of PAM converges to a critical point of (6) under the assumption that the objective function satisfies the Kurdyka–Łojasiewicz (KL) inequality.[1] We need to point out that the only difference between PAM ((12) and (13)) and the AMA (7) is the proximal terms, which together with the KL inequality helps establish the convergence result. Note that the A-subproblem in (12) corresponds to the reduced-rank Procrustes rotation and can be solved by a SVD. The B-subproblem in (13) is a LASSO-type problem and can be solved efficiently by first-order methods such as FISTA or block coordinate descent. A better algorithm that avoids an iterative solver for the subproblem is PALM, which linearizes the quadratic functions in the subproblems of (12) and (13). A typical iteration of PALM is the following:

$$A_{k+1} := \arg\min_{A} \langle \nabla_A H(A_k, B_k), A\rangle + \frac{1}{2t_1}\|A - A_k\|_F^2 \quad \text{s.t.} \quad A \in \mathrm{St}(p, r), \tag{14}$$

$$B_{k+1} := \arg\min_{B} \langle \nabla_B H(A_{k+1}, B_k), B\rangle + \frac{1}{2t_2}\|B - B_k\|_F^2 + \mu \sum_{j=1}^{r} \|B_j\|_2^2 + \sum_{j=1}^{r} \mu_{1,j}\|B_j\|_1, \tag{15}$$

where $\nabla_A H$ and $\nabla_B H$ denote the gradient of H with respect to A and B, respectively. The two subproblems in (14) and (15) are easier to solve than the ones in (12) and (13) because they both admit closed-form solutions. In particular, the solution of the A-subproblem in (14) corresponds to the projection onto the orthogonality constraint, which is given by a SVD; the solution of the B-subproblem in (15) is given by the ℓ_1 soft-thresholding operation. It is shown in Bolte et al. (2014) that the sequence of PALM converges to a critical point of (6) under the assumption that the objective function satisfies the KL inequality. Recently, Erichson

et al. (2020) proposed a projected gradient method based on variable projection (VP) for solving (6). Though the motivation of this algorithm is different, it can be viewed as a variant of PAM and PALM. Roughly speaking, the VP algorithm combines the A-subproblem (without the proximal term) in (12) and the B-subproblem in (15). That is, it updates the iterates as follows:

$$A_{k+1} := \arg\min_{A} \; H(A, B_k) \quad \text{s.t.} \quad A \in \mathrm{St}(p, r), \tag{16}$$

$$B_{k+1} := \arg\min_{B} \; \langle \nabla_B H(A_{k+1}, B_k), B\rangle + \frac{1}{2t_2}\|B - B_k\|_F^2 + \mu \sum_{j=1}^{r} \|B_j\|_2^2 + \sum_{j=1}^{r} \mu_{1,j}\|B_j\|_1. \tag{17}$$

Note that the difference of PALM ((14) and (15)) and VP ((16) and (17)) lies in the A-subproblem. The A-subproblem linearizes the quadratic function $H(A, B_k)$ in (14) but not in (16). This does not affect the performance of the algorithms much because in this specific problem, the A-subproblems correspond to a SVD in both algorithms. It is shown in Erichson et al. (2020) that the VP ((16) and (17)) converges to a stationary point of (6). Another recent work that can solve (6) is the manifold proximal gradient (ManPG) algorithm proposed by Chen et al. (2020). We will discuss it in more details later because it is closely related to the algorithm we propose in this paper. For other algorithms for solving sparse PCA, we refer interested readers to a recent survey paper (Zou and Xue 2018) for more details.

2.2. Existing Methods for Sparse CCA

Chen et al. (2013) studied a canonical correlation analysis via the precision-adjusted iterative thresholding (CAPIT) algorithm for solving the single sparse CCA (11). CAPIT alternates between an iterative thresholding step and a power method step to deal with the sparsity regularization and orthogonality constraints, respectively. The convex program with group LASSO refinement (CoLaR) method proposed by Gao et al. (2017) targets the multiple sparse CCA (9). CoLaR is a two-stage algorithm. In the first stage, a convex relaxation of (9) based on the matrix lifting technique is solved. In the second stage, the solution obtained from the first stage is refined by solving a group LASSO-type problem. Wiesel et al. (2008) suggested a greedy approach for solving (2) with cardinality constraints on u and v. Recently, Suo et al. (2017) developed an AMA for solving the single sparse CCA (11) that solves two subproblems in each iteration by solving (11) with respect to u (respectively, v) with v (respectively, u) fixed as v^k (respectively, u^k). The subproblems were then solved by a linearized alternating direction method of multipliers (ADMM) algorithm. We need to point out that none of these algorithms for sparse CCA has a convergence guarantee. There exist some other methods for sparse CCA (see, e.g., Witten et al. 2009, Hardoon and Shawe-Taylor 2011), but we omit their details here because they are not directly related to (8) and (9). We also point out that the PAM, PALM, and VP algorithms discussed in Section 2.1 are not good choices for sparse CCA ((8) and (9)) because the resulting subproblems are not easy to solve and require expensive subroutines. For instance, to apply PALM to (8), one needs to compute the proximal mapping of $f(u) + \iota(u^\top X^\top X u = 1)$, which does not admit a closed-form solution and is thus computationally expensive, where $\iota(\cdot)$ denotes the indicator function.

3. Unified Alternating Manifold Proximal Gradient Algorithm

In this section, we introduce our alternating manifold proximal gradient (A-ManPG) algorithm for solving (1). The following assumptions are made for problem (1).

Assumption 1. *We assume that the following are true:*

a. *The functions f and g are convex and Lipschitz continuous with Lipschitz constants L_f and L_g, respectively, and their proximal mappings are semismooth.*[2]

b. *The function $\nabla_A H(A, B)$ is Lipschitz continuous with respect to A when fixing B, and the Lipschitz constant is $L_A(B)$. Similarly, $\nabla_B H(A, B)$ is Lipschitz continuous with respect to B when fixing A, and the Lipschitz constant is $L_B(A)$. Moreover, there exist constants $L_A > 0$ and $L_B > 0$ such that $\sup_{B\in\mathcal{M}_2}\{L_A(B)\} \le L_A$ and $\sup_{A\in\mathcal{M}_1}\{L_B(A)\} \le L_B$.*

c. *The function F is lower bounded by a constant F^*.*

d. *The function F is coercive with respect to (A, B). An immediate consequence of this assumption is that the sublevel set $\{(A, B) \mid F(A, B) \le F(A_0, B_0)\}$ is bounded for any (A_0, B_0).*

Remark 1. In Assumption 1(a), the Lipschitz continuity and convexity are all defined in the Euclidean space. In Assumption 1(b), it is easy to obtain that $L_A(B) \equiv 0$ and $L_B(A) = 2\lambda_{\max}(X^\top X)$ for problem (6) and $L_A(B) = L_B(A) \equiv 0$ for problems (8) and (10). Assumptions 1(c) and 1(d) can be removed if $\mathcal{M}_1$ and $\mathcal{M}_2$ are compact submanifolds.

The ManPG algorithm proposed by Chen et al. (2020) can be applied to solve (1). In each iteration, ManPG linearizes $H(A,B)$ and solves the following convex subproblem:

$$\min_{D^A,D^B}\left\langle\begin{pmatrix}\nabla_A H(A_k,B_k)\\ \nabla_B H(A_k,B_k)\end{pmatrix},\begin{pmatrix}D^A\\ D^B\end{pmatrix}\right\rangle+\frac{1}{2t}\|D^A\|_F^2+\frac{1}{2t}\|D^B\|_F^2+f\left(A_k+D^A\right)+g\left(B_k+D^B\right)$$
$$\text{s.t. } D^A\in \mathrm{T}_{A_k}\mathcal{M}_1, D^B\in \mathrm{T}_{B_k}\mathcal{M}_2, \tag{18}$$

where $t<1/L$, and L is the Lipschitz constant of $\nabla H(A,B)$ on the tangent space $\mathrm{T}_{A_k}\mathcal{M}_1\times\mathrm{T}_{B_k}\mathcal{M}_2$. Here $\langle A,B\rangle=\mathrm{Tr}(A^\top B)$ denotes the inner product. Note that (18) is actually separable for D^A and D^B and thus reduces to two subproblems for D^A and D^B, respectively. As a result, ManPG (18) can be viewed as a Jacobi-type algorithm in this case because it computes D^A and D^B in parallel. We found from our numerical experiments that the algorithm converges faster if D^A and D^B are computed in a Gauss–Seidel manner. This leads to the following updating scheme, which is the basis of our A-ManPG algorithm:

$$D_k^A:=\arg\min_{D^A}\ \langle\nabla_A H(A_k,B_k),D^A\rangle+f\left(A_k+D^A\right)+\frac{1}{2t_1}\|D^A\|_F^2\ \text{ s.t. } D^A\in\mathrm{T}_{A_k}\mathcal{M}_1, \tag{19}$$

$$D_k^B:=\arg\min_{D^B}\ \langle\nabla_B H(A_{k+1},B_k),D^B\rangle+g\left(B_k+D^B\right)+\frac{1}{2t_2}\|D^B\|_F^2\ \text{ s.t. } D^B\in\mathrm{T}_{B_k}\mathcal{M}_2, \tag{20}$$

where A_{k+1} is obtained via a retraction operation (see Algorithm 1), $t_1\le 1/L_A$, and $t_2\le 1/L_B$. The Gauss–Seidel-type algorithm A-ManPG usually performs much better than the Jacobi-type algorithm ManPG, because the Lipschitz constants are smaller, and thus, larger step sizes are allowed. The details of the A-ManPG algorithm are described in Algorithm 1.

Algorithm 1 (A-ManPG Method)

1: Input: Initial point (A_0,B_0), parameter $\gamma\in(0,1)$, step sizes $t_1\le 1/L_A$ and $t_2\le 1/L_B$.
2: **for** $k=0,1,\ldots,$ **do**
3: Solve the A-subproblem in (19) to obtain D_k^A.
4: Set $\alpha_1^k=1$.
5: **while** $F(\mathrm{Retr}_{A_k}(\alpha_1^k D_k^A),B_k)>F(A_k,B_k)-\frac{\alpha_1^k}{2t_1}\|D_k^A\|_F^2$, **do**
6: $\alpha_1^k=\gamma\alpha_1^k$
7: **end while**
8: Set $A_{k+1}=\mathrm{Retr}_{A_k}(\alpha_1^k D_k^A)$.
9: Solve the B-subproblem in (20) to obtain D_k^B.
10: Set $\alpha_2^k=1$.
11: **while** $F(A_{k+1},\mathrm{Retr}_{B_k}(\alpha_2^k D_k^B))>F(A_{k+1},B_k)-\frac{\alpha_2^k}{2t_2}\|D_k^B\|_F^2$, **do**
12: $\alpha_2^k=\gamma\alpha_2^k$
13: **end while**
14: Set $B_{k+1}=\mathrm{Retr}_{B_k}(\alpha_2^k D_k^B)$.
15: **end for**

Remark 2. Note that the iterates A_k and B_k are kept on the manifolds through the retraction operations Retr_A and Retr_B. There exist many choices for the retraction operations, and in Algorithm 1, we did not specify which ones to use. We discuss common retractions for the Stiefel manifold and the generalized Stiefel manifold in the online appendix. In our numerical experiments in Section 4, we chose polar decomposition as the retraction. Lines 4–7 and 10–13 in Algorithm 1 are backtracking line search procedures. These are necessary to guarantee that the objective function has a sufficient decrease in each iteration, which is needed for the convergence analysis (see the online appendix).

From Lemma 3 (see the online appendix), we know that $D_k^A=0$ and $D_k^B=0$ imply that (A_k,B_k) is a stationary point (see the online appendix, Definition 5) for problem (1). As a result, we can define an ϵ-stationary point of (1) as follows.

Definition 1. We call (A,B) an *ϵ-stationary point* of (1) if D^A and D^B returned by (19) and (20) with step sizes $t_1=1/L_A, t_2=1/L_B$ satisfy $(\|D^A/t_1\|_F^2+\|D^B/t_2\|_F^2)\le\epsilon^2$.

We have the following convergence results for the A-ManPG algorithm (Algorithm 1), whose proof is given in the online appendix.

Theorem 1. *Under Assumption 1, any limit point of the sequence $\{(A_k, B_k)\}$ generated by Algorithm 1 is a stationary point of problem (1). Moreover, Algorithm 1 with step sizes $t_1 = 1/L_A$, $t_2 = 1/L_B$ returns an ϵ-stationary point (A_k, B_k) in at most $2(F(A_0, B_0) - F^*)/((\gamma\bar{\alpha}_1 t_1 + \gamma\bar{\alpha}_2 t_2)\epsilon^2)$ iterations, where $\bar{\alpha}_1 > 0$ and $\bar{\alpha}_2 > 0$ are constants.*

3.1. Semismooth Newton Method for the Subproblems

The main computational effort in each iteration of Algorithm 1 is to solve (19) and (20). For the Stiefel manifold and the generalized Stiefel manifold, (19) and (20) are both equality-constrained convex problems, given that both f and g are convex functions. Note that if f (respectively, g) vanishes, the A-subproblem (respectively, B-subproblem) becomes the projection onto the tangent space of $\mathcal{M}_1$ (respectively, $\mathcal{M}_2$), which reduces to a Riemannian gradient step and can be easily done. Here we discuss the general case where f and g do not vanish. The subproblem solver should be efficient and yields highly accurate solutions. We have tested several possible candidates of subproblem solvers including ADMM and the augmented Lagrangian method but found that they are either less efficient or do not yield a highly accurate solution. In the end, we found that an adaptive semismooth Newton (SSN) method (Xiao et al. 2018) is suitable for solving this kind of problem. The notion of semismoothness was originally introduced by Mifflin (1977) for real-valued functions and extended to vector-valued mappings by Qi and Sun (1993). In a pioneering work on the SSN method, Solodov and Svaiter (1998) proposed a globally convergent Newton's method by exploiting the structure of monotonicity and established a local superlinear rate under the conditions that the operator is semismooth and the generalized Jacobian is nonsingular at the global optimal point. The convergence rate is extended in Zhou and Toh (2005) to the setting where the generalized Jacobian is not necessarily nonsingular. Recently, the SSN method has received significant attention because of its success in solving structured convex problems to high accuracy (see Wang et al. 2010; Zhao et al. 2010; Qi and Sun 2011; Yang et al. 2013, 2015; Li et al. 2018; Xiao et al. 2018).

We now describe how to apply the adaptive SSN (ASSN) method in Xiao et al. (2018) to solve (19) and (20). The derivation here closely follows the one in Chen et al. (2020). For brevity, we only focus on the A-subproblem with $\mathcal{M}_1 = \{A \mid A^\top X^\top X A = I_r\}$ and $f(A) = \tau_1\|A\|_{2,1}$ as used in (10). Note that for the Stiefel manifold $\mathcal{M} = \mathrm{St}(p, r)$, its tangent space is given by $\mathrm{T}_A\mathcal{M} = \{D \mid D^\top A + A^\top D = 0\}$, and for the generalized Stiefel manifold $\mathcal{M} = \{A \mid A^\top M A = I_r\}$, its tangent space is given by $\mathrm{T}_A\mathcal{M} = \{D \mid D^\top M A + A^\top M D = 0\}$. For ease of notation, we denote $t = t_1$, $D = D^A$, $M := X^\top X$, and $h(A) := H(A, B_k)$. In this case, the A-subproblem in (19) reduces to

$$D_k := \arg\min_D \langle \nabla h(A_k), D\rangle + f(A_k + D) + \frac{1}{2t}\|D\|_F^2 \text{ s.t. } D^\top M A_k + A_k^\top M D = 0. \tag{21}$$

By associating a Lagrange multiplier Λ with the linear equality constraint, the Lagrangian function of (21) can be written as follows:

$$\mathcal{L}(D; \Lambda) = \langle \nabla h(A_k), D\rangle + \frac{1}{2t}\|D\|_F^2 + f(A_k + D) - \langle D^\top M A_k + A_k^\top M D, \Lambda\rangle, \tag{22}$$

and the Karush–Kuhn–Tucker system of (21) is given by

$$0 \in \partial_D \mathcal{L}(D; \Lambda) \text{ and } D^\top M A_k + A_k^\top M D = 0. \tag{23}$$

The first condition in (23) implies that D can be computed by

$$D(\Lambda) = \mathrm{prox}_{tf}(\mathrm{B}(\Lambda)) - A_k, \text{ with } B(\Lambda) = A_k - t(\nabla h(A_k) - 2MA_k\Lambda), \tag{24}$$

where $\mathrm{prox}_f(A)$ denotes the proximal mapping of function f at point A. By substituting (24) into the second condition in (22), we obtain that Λ satisfies

$$E(\Lambda) := D(\Lambda)^\top M A_k + A_k^\top M D(\Lambda) = 0, \tag{25}$$

and thus the problem reduces to finding a root of function E. Because E is a monotone operator (see Chen et al. 2020) and the proximal mapping of the ℓ_2 norm is semismooth,[3] we can apply SSN to find the zero of E.

The SSN method requires computation of the generalized Jacobian of E, and in the following, we show how to compute it. We first derive the vectorization of $E(\Lambda)$:

$$\begin{aligned}\mathrm{vec}(\mathrm{E}(\Lambda)) &= \big((MA_k)^\top \otimes I_r\big)\mathrm{vec}\big(D(\Lambda)^\top\big) + \big(I_r \otimes (MA_k)^\top\big)K_{rp}\mathrm{vec}\big(D(\Lambda)^\top\big)\\ &= (I_{r^2} + K_{rr})\big((MA_k)^\top \otimes I_r\big)\Big[\mathrm{prox}_{tf}\big(\mathrm{vec}\big((MA_k)^\top - t\nabla h(A_k)^\top\big)\\ &\quad + 2t((MA_k) \otimes I_r)\mathrm{vec}(\Lambda)) - \mathrm{vec}\big(A_k{}^\top\big)\Big],\end{aligned}$$

where K_{rp} and K_{rr} denote the commutation matrices. We define the following matrix:

$$\mathcal{G}(\mathrm{vec}(\Lambda)) = 2(I_{r^2} + K_{rr})\big((MA_k)^\top \otimes I_r\big)\mathcal{J}\big(y\big)\big|_{y=\mathrm{vec}(B(\Lambda)^\top)}((MA_k) \otimes I_r),$$

where $\otimes$ denotes the Kronecker product, and $\mathcal{J}(y)$ is the generalized Jacobian of $\mathrm{prox}_{tf}(y)$. The matrix $\mathcal{J}(y)$ is defined as $\mathcal{J}(y)|_{y=\mathrm{vec}(B(\Lambda)^\top)} = \mathrm{Diag}(\Delta_1, \ldots, \Delta_p)$, where the matrices $\Delta_j, j = 1, \ldots, p$ are defined as follows:

$$\Delta_j = \begin{cases} I_r - \frac{\tau_1 t}{\|b_j\|_2}\left(I_r - \frac{b_j b_j^\top}{\|b_j\|_2^2}\right), & \text{if } \|b_j\|_2 > t\tau_1,\\ \gamma\frac{b_j b_j^\top}{(t\tau_1)^2} : \gamma \in [0,1], & \text{if } \|b_j\|_2 = t\tau_1,\\ 0, & \text{otherwise.}\end{cases}$$

Here b_j is the jth column of matrix $B(\Lambda)^\top$. From the monotonicity of $\mathrm{vec}(E(\mathrm{vec}(\Lambda)))$, it follows that $\mathcal{G}(\mathrm{vec}(\Lambda))$ is positive semidefinite[4] (Xiao et al. 2018, lemma 2.4). From Hiriart-Urruty et al. (1984, example 2.5), we know that $\mathcal{G}(\mathrm{vec}(\Lambda))\xi = \partial\mathrm{vec}(E(\mathrm{vec}(\Lambda))\xi$, $\forall \xi \in \mathbb{R}^{r^2}$. So $\mathcal{G}(\mathrm{vec}(\Lambda))$ serves as an alternative of $\partial\mathrm{vec}(E(\mathrm{vec}(\Lambda)))$. It is known that the global convergence of ASSN is guaranteed if any element of $\mathcal{G}(\mathrm{vec}(\Lambda))$ is positive semidefinite (Xiao et al. 2018). For the local convergence rate, one needs more conditions on $\partial\mathrm{vec}(E(\mathrm{vec}(\Lambda)))$. We refer to Xiao et al. (2018) for more details. Note that because Λ is a symmetric matrix, we can work with the lower triangular part of Λ only and remove the duplicated entries in the upper triangular part. To do so, we use $\overline{\mathrm{vec}}(\Lambda)$ to denote the $\frac{1}{2}r(r+1)$-dimensional vector obtained from $\mathrm{vec}(\Lambda)$ by eliminating all superdiagonal elements of Λ. It is known that there exists a unique $r^2 \times \frac{1}{2}r(r+1)$ matrix U_r, which is called the *duplication matrix* (Magnus and Neudecker 2018, chapter 3.8), such that $U_r\overline{\mathrm{vec}}(\Lambda) = \mathrm{vec}(\Lambda)$. The Moore–Penrose inverse of U_r is $U_r^+ = (U_r^\top U_r)^{-1}U_r^\top$, and it satisfies $U_r^+\mathrm{vec}(\Lambda) = \overline{\mathrm{vec}}(\Lambda)$. The alternative of the generalized Jacobian of $\overline{\mathrm{vec}}(E(U_r\overline{\mathrm{vec}}(\Lambda)))$ is given by

$$\mathrm{G}(\overline{\mathrm{vec}}(\Lambda)) = tU_r^+\mathcal{G}(\mathrm{vec}(\Lambda))U_r = 4tU_r^+\big((MA_k)^\top \otimes I_r\big)\mathcal{J}\big(y\big)\big|_{y=\mathrm{vec}(B(\Lambda)^\top)}((MA_k) \otimes I_r)U_r, \tag{26}$$

where we use the identity $K_{rr} + I_{r^2} = 2U_rU_r^+$. Therefore, (26) can be simplified to

$$\begin{aligned}&G(\overline{\mathrm{vec}}(\Lambda))\\ &= 4tU_r^+\big((MA_k)^\top \otimes I_r\big)\begin{pmatrix}\Delta_1 & & \\ & \ddots & \\ & & \Delta_p\end{pmatrix}((MA_k) \otimes I_r)U_r\\ &= 4tU_r^+\begin{pmatrix}\sum_{j=1}^p (MA_k)_{j1}^2\Delta_j & \sum_{j=1}^p (MA_k)_{j1}(MA_k)_{j2}\Delta_j & \cdots & \sum_{j=1}^p (MA_k)_{j1}(MA_k)_{jr}\Delta_j\\ \sum_{j=1}^p (MA_k)_{j2}(MA_k)_{j1}\Delta_j & \sum_{j=1}^p (MA_k)_{j2}^2\Delta_j & \cdots & \sum_{j=1}^p (MA_k)_{j2}(MA_k)_{jr}\Delta_j\\ \vdots & \vdots & \vdots & \vdots\\ \sum_{j=1}^p (MA_k)_{jr}(MA_k)_{j1}\Delta_j & \sum_{j=1}^p (MA_k)_{jr}(MA_k)_{j2}\Delta_j & \cdots & \sum_{j=1}^p (MA_k)_{jr}^2\Delta_j\end{pmatrix}U_r.\end{aligned} \tag{27}$$

Because both U_r and U_r^+ have only r^2 nonzero elements, the computation flop of matrix $G(\overline{\mathrm{vec}}(\Lambda))$ is $2pr^4 + 2r(r+1)r^2$. The ASSN method in Xiao et al. (2018) first computes Newton's direction d_k by solving

$$\big(\mathrm{G}(\overline{\mathrm{vec}}(\Lambda_k)) + \eta I\big)d = -\overline{\mathrm{vec}}(E(\overline{\mathrm{vec}}(\Lambda_k))), \tag{28}$$

where $\eta > 0$ is a regularization parameter. Note that η is necessary here because $G(\overline{\mathrm{vec}}(\Lambda))$ could be singular if $\Delta_j = 0$ for some j. We then update Λ_k by

$$\overline{\mathrm{vec}}(\Lambda_{k+1}) = \overline{\mathrm{vec}}(\Lambda_k) + d_k.$$

Thus, the main computational cost of the SSN algorithm lies in solving (28), which is $\mathcal{O}(r^6)$ in each iteration. Therefore, performing a SSN step is faster than computing the gradient of the objective function in (1), which is $\mathcal{O}(p^2 r)$, if r is much smaller than p. We describe the detailed ASSN algorithm proposed in Xiao et al. (2018) in the online appendix.

4. Numerical Experiments

4.1. Sparse PCA

In this section, we apply our A-ManPG algorithm[5] to solve sparse PCA (6) and compare its performance with that of three existing methods: AMA (Zou et al. 2006), PALM (Bolte et al. 2014), and VP (Erichson et al. 2020). The details of the parameter settings of these algorithms are as follows:

- AMA (7): Maximum iteration number of AMA is set to 1,000. We use FISTA (Beck and Teboulle 2009) to solve the B-subproblem, and we set the maximum iterations of FISTA to 1,000 and stop it if $\|A_j - A_{j-1}\|/\|A_{j-1}\| \le 10^{-6}$, where $\{A_j\}$ is the sequence in FISTA.
- PALM ((14) and (15)): Here $t_1 := 1$, $t_2 := 1/(2\lambda_{\max}(X^\top X))$. Maximum iteration number is set to 10,000.
- VP ((16) and (17)): Here $t_2 := 1/(2\lambda_{\max}(X^\top X))$. Maximum iteration number is set to 10,000.
- A-ManPG: Here $t_1 = 100/p$, $t_2 := 1/(2\lambda_{\max}(X^\top X))$. Maximum iteration number is set to 10,000.

The algorithms are terminated using the following criteria. First, we use the PALM algorithm as a baseline, and we denote the objective function value in (6) as $F(A, B)$, that is., $F(A, B) = H(A, B) + \mu \sum_{j=1}^r \|B_j\|^2 + \sum_{j=1}^r \mu_{1,j}\|B_j\|_1$. We terminate PALM when we find that

$$|F_{PALM}(A_{k+1}, B_{k+1}) - F_{PALM}(A_k, B_k)| < 10^{-5}. \tag{29}$$

We then terminate the AMA, A-ManPG, and VP algorithms when their objective function values are smaller than F_{PALM} and the change of their objective values in two consecutive iterations is less than 10^{-5}.

We generate the data matrix X in the following manner. First, the entries of X are generated following the standard normal distribution $\mathcal{N}(0, 1)$. The columns of X are then centered so that the columns have zero mean, and they are then scaled by dividing the largest ℓ_2 norm of the columns. We report the comparison results of the four algorithms in Tables 1 and 2, where $r = 6$ for all cases. In particular, Table 1 reports the results for $n < p$, and we tested $\mu = 1$ and $\mu = 10$ because it is suggested in Zou et al. (2006) that μ should be relatively large in this case. Table 2 reports the results for $n > p$, and we set $\mu = 10^{-6}$ because it is suggested in Zou et al. (2006) that μ should be sufficiently small in this case. In these tables, central processing unit (CPU) times are in seconds, and "sp" denotes the percentage of zero entries of matrix B. From Tables 1 and 2, we see that the four algorithms generated solutions with similar objective function value $F(A, B)$ and similar sparsity sp. In terms of CPU time, AMA is the slowest, and the other three are comparable and are all much faster than AMA. This is because AMA needs an iterative solver to solve the B-subproblem, which is time-consuming in practice.

4.1.1. Comparison with ManPG-Ada.

We also compared the A-ManPG algorithm with the ManPG-Ada algorithm (a more practical version of ManPG) proposed in Chen et al. (2020) on the sparse PCA problem (6). The comparison results are reported in Figures 1 and 2. These results indicate that A-ManPG is at least two times faster than ManPG-Ada on the tested synthetic problems. More comparison results are presented in Figures 3 and 4 in the online appendix.

4.2. Sparse CCA: Vector Case

In this section, we report the numerical results of the A-ManPG algorithm for solving the single sparse CCA (11) and compare its performance with a recent approach proposed by Suo et al. (2017): AMA + linearized alternating direction method of multipliers (LADMM). More specifically, the AMA + LADMM method aims to solve the relaxation of (11) as follows:

$$\begin{aligned} \min_{u\in\mathbb{R}^p, v\in\mathbb{R}^q} \; & -u^\top X^\top Y v + \tau_1\|u\|_1 + \tau_2\|v\|_1 \\ \text{s.t. } & u^\top X^\top X u \le 1,\ v^\top Y^\top Y v \le 1. \end{aligned} \tag{30}$$

We generate the data in the same manner as Suo et al. (2017). Specifically, two data sets $X \in \mathbb{R}^{n\times p}$ and $Y \in \mathbb{R}^{n\times q}$ are generated from a Gaussian model with zero mean and covariance matrix $[\Sigma_x, \Sigma_{xy}; \Sigma_{yx}, \Sigma_y]$ with $\Sigma_{xy} = \hat{\rho}\Sigma_x \hat{u}\hat{v}^\top \Sigma_y$, $\hat{u}$

Table 1. Comparison of Algorithms for Solving the Sparse PCA Problem with $n < p$

	$\mu = 1$				$\mu = 10$			
Algorithm	$F(A,B)$	sp	CPU	Iteration	$F(A,B)$	sp	CPU	Iteration
				$(n,p) = (100,1{,}000)$, $\mu_{1,j} = 0.1, j = 1,\ldots,r$				
AMA	**−4.90778e+1**	59.7	11.48	648	−2.69714e+1	25.2	4.06	409
A-ManPG	−4.90771e+1	59.4	**0.36**	1,172	**−2.69716e+1**	25.4	**0.12**	375
PALM	−4.90769e+1	59.4	0.39	1,394	−2.69711e+1	25.3	0.15	518
VP	−4.90770e+1	59.4	0.37	1,335	−2.69712e+1	25.2	0.13	453
				$(n,p) = (100,1{,}000)$, $\mu_{1,j} = 0.2, j = 1,\ldots,r$				
AMA	**−4.16070e+1**	76.5	7.28	433	**−2.16374e+1**	42.6	2.78	259
A-ManPG	−4.16057e+1	76.4	**0.22**	712	−2.16371e+1	42.7	**0.09**	265
PALM	−4.16055e+1	76.4	0.23	855	−2.16371e+1	42.6	0.12	343
VP	−4.16056e+1	76.4	0.23	825	−2.16372e+1	42.6	0.10	301
				$(n,p) = (500,5{,}000)$, $\mu_{1,j} = 0.1, j = 1,\ldots,r$				
AMA	**−5.56171e+1**	75.8	728.29	1,000	**−3.18762e+1**	40.5	452.19	1,407
A-ManPG	−5.56134e+1	75.7	**8.90**	1,982	−3.18753e+1	40.5	**4.73**	1,021
PALM	−5.56131e+1	75.8	10.77	2,210	−3.18550e+1	40.3	4.94	1,023
VP	−5.56132e+1	75.7	10.87	2,147	−3.18759e+1	40.5	7.17	1,643
				$(n,p) = (500,5{,}000)$, $\mu_{1,j} = 0.2, j = 1,\ldots,r$				
AMA	**−4.25661e+1**	89.3	733.36	1,000	−2.18082e+1	63.7	171.90	545
A-ManPG	−4.25408e+1	89.0	9.05	2,017	**−2.18085e+1**	63.6	**3.28**	700
PALM	−4.25111e+1	89.1	7.59	1,713	−2.18079e+1	63.6	4.01	870
VP	−4.25115e+1	89.1	**7.28**	1,682	−2.18080e+1	63.6	3.57	773
				$(n,p) = (1{,}000,5{,}000)$, $\mu_{1,j} = 0.1, j = 1,\ldots,r$				
AMA	**−2.89684e+1**	79.6	306.42	437	**−1.34357e+1**	50.9	204.01	534
A-ManPG	−2.89676e+1	79.7	**9.61**	959	−1.34355e+1	50.9	**5.90**	535
PALM	−2.89675e+1	79.6	10.24	1,031	−1.34352e+1	50.8	8.15	794
VP	−2.89676e+1	79.6	9.64	975	−1.34355e+1	50.8	6.74	644
				$(n,p) = (1{,}000,5{,}000)$, $\mu_{1,j} = 0.2, j = 1,\ldots,r$				
AMA	**−1.94321e+1**	93.9	398.16	666	−7.41353e+0	77.1	317.83	841
A-ManPG	−1.94308e+1	93.9	23.33	2,346	**−7.41377e+0**	77.4	**10.38**	1,033
PALM	−1.94306e+1	93.9	21.54	2,104	−7.41316e+0	77.1	16.54	1,565
VP	−1.94306e+1	93.9	**19.27**	1,989	−7.41354e+0	77.1	11.33	1,138

Notes. CPU times are given in seconds; sp denotes the percentage of zero entries of matrix. The best ones among the four algorithms are highlighted using bold text.

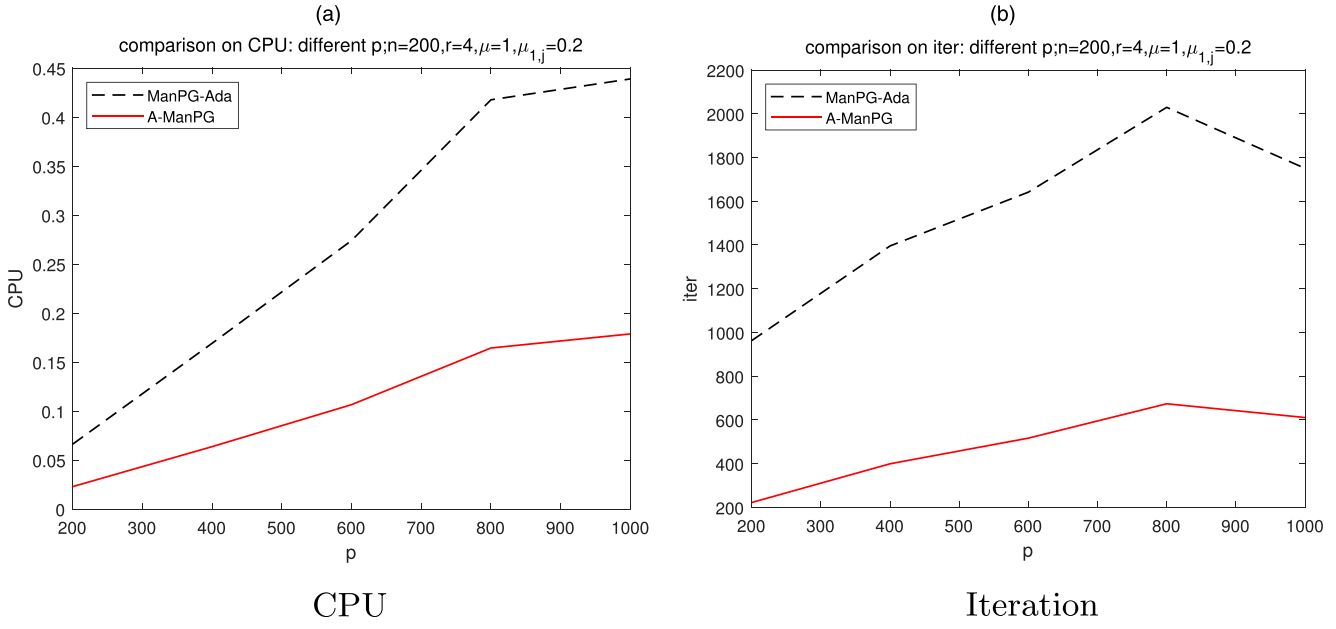
and $\hat{v}$ are the true canonical vectors, and $\hat{\rho}$ is the true canonical correlation. In our numerical tests, $\hat{u}$ and $\hat{v}$ are generated randomly such that they both have five nonzero entries and the nonzero coordinates are set at the $\{1,6,11,16,21\}$th coordinates. The nonzero entries are obtained from normalizing (with respect to Σ_x and Σ_y) random numbers drawn from the uniform distribution on the finite set $\{-2,-1,0,1,2\}$. We set $\hat{\rho} = 0.9$ in all tests. We tested three different ways to generate the covariance matrices Σ_x and Σ_y that are exactly the same as

Table 2. Comparison of Algorithms for Solving the Sparse PCA Problem with $n > p$ and $\mu = 10^{-6}$

Algorithm	$F(A,B)$	sp	CPU	Iteration	Algorithm	$F(A,B)$	sp	CPU	Iteration
	$(n,p) = (5{,}000,2{,}000)$, $\mu_{1,j} = 0.01, j = 1,\ldots,r$					$(n,p) = (5{,}000,2{,}000)$, $\mu_{1,j} = 0.05, j = 1,\ldots,r$			
AMA	**−1.29155e+1**	37.8	133.04	539	AMA	**−8.83497e+0**	89.4	88.04	425
A-ManPG	−1.29151e+1	37.6	**2.44**	493	A-ManPG	−8.83440e+0	89.4	3.87	842
PALM	−1.29142e+1	37.6	3.54	768	PALM	−8.83437e+0	89.3	4.36	977
VP	−1.29150e+1	37.5	2.93	640	VP	−8.83452e+0	89.3	**3.48**	773
	$(n,p) = (8{,}000,1{,}000)$, $\mu_{1,j} = 0.01, j = 1,\ldots,r$					$(n,p) = (8{,}000,1{,}000)$, $\mu_{1,j} = 0.05, j = 1,\ldots,r$			
AMA	**−9.04522e+0**	37.9	23.53	325	AMA	−6.59097e+0	95.6	44.30	636
A-ManPG	−9.04477e+0	37.6	**0.22**	276	A-ManPG	−6.58996e+0	95.7	**0.78**	897
PALM	−9.04471e+0	37.7	0.31	507	PALM	−6.58995e+0	95.7	0.82	1,486
VP	−9.04511e+0	37.8	0.24	360	VP	**−6.60764e+0**	95.9	1.22	1,907
	$(n,p) = (8{,}000,2{,}000)$, $\mu_{1,j} = 0.01, j = 1,\ldots,r$					$(n,p) = (8{,}000,2{,}000)$, $\mu_{1,j} = 0.05, j = 1,\ldots,r$			
AMA	**−1.07975e+1**	40.3	116.65	388	AMA	**−7.32162e+0**	95.3	167.36	597
A-ManPG	**−1.07975e+1**	40.2	**2.19**	363	A-ManPG	−7.31837e+0	95.0	3.34	578
PALM	−1.07966e+1	40.2	3.00	550	PALM	−7.30781e+0	95.0	3.97	780
VP	−1.07972e+1	40.4	2.70	437	VP	−7.30822e+0	95.0	**3.29**	606

Notes. CPU times are given in seconds; sp denotes the percentage of zero entries of matrix. The best ones among the four algorithms are highlighted using bold text.

Figure 1. (Color online) Comparison of A-ManPG and ManPG-Ada on the Sparse PCA Problem with $p = \{200, 400, 600, 800, 1{,}000\}$



presented in section 4.1 of Suo et al. (2017): identity matrices, Toeplitz matrices, and sparse inverse matrices. For succinctness, we omit them here but include them in the online appendix. The matrices X and Y are both divided by $\sqrt{n-1}$ such that $X^\top Y$ is the estimated covariance matrix. Note that if $n < p$ or $n < q$, the covariance matrix $X^\top X$ or $Y^\top Y$ is not positive definite. In this case, we replace $X^\top X$ by $(1-\alpha)X^\top X + \alpha I_p$ and $Y^\top Y$ by $(1-\alpha)Y^\top Y + \alpha I_q$ in the constraints of (11), so we can still keep them as manifold constraints. In our experiments, we chose $\alpha = 10^{-4}$. We adopted the same initialization procedure as suggested in Suo et al. (2017). We set $\tau_1 = \tau_2 = \frac{1}{2} b\sqrt{\log(p+q)/n}$ in (11), where b was set to $b = \{1, 1.2, 1.4, 1.6\}$. We report the best result among all the candidates. For each b, we solved (11) by A-ManPG with $\delta = 10^{-4}$, $\gamma = 0.5$, and $t_1 = t_2 = 1$. The A-ManPG algorithm was stopped if $\max\{\|D_k^A\|_F^2, \|D_k^B\|_F^2\} \le 10^{-8}$, and the ASSN algorithm was stopped if $\|E(\Lambda_k)\|_F \le 10^{-5}$ in (26). For AMA + LADMM, we set the stopping criteria of LADMM as $\|u_j - u_{j-1}\| \le 10^{-3}$ and $\|v_j - v_{j-1}\| \le 10^{-3}$, where u_j and v_j are iterates in LADMM. We set the stopping criteria of AMA as $\|u_k - u_{k-1}\| \le 10^{-3}$ and $\|v_k - v_{k-1}\| \le 10^{-3}$, where u_k and v_k are iterates in AMA.

Figure 2. (Color online) Comparison of A-ManPG and ManPG-Ada on the Sparse PCA Problem with $r = \{1, 2, 4, 6, 8\}$

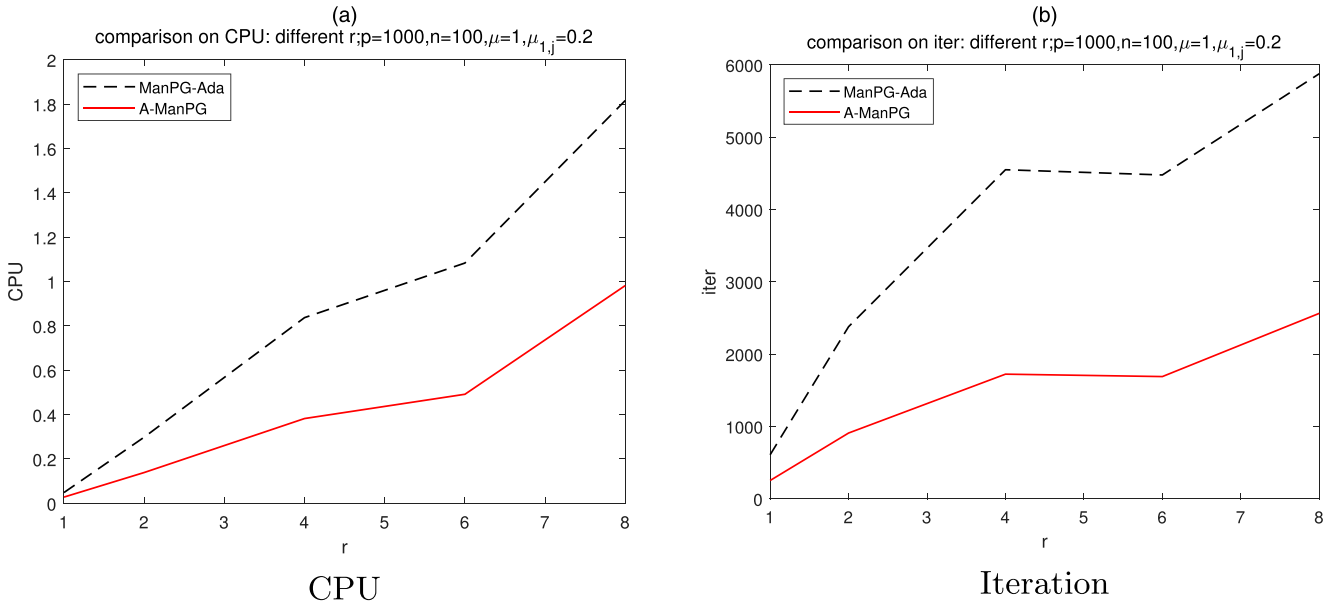


Table 3. Comparison of A-ManPG and AMA+LADMM for Solving Single Sparse CCA

	A-ManPG						AMA+LADMM					
(n, p, q)	CPU	lossu	lossv	ρ	nu	nv	CPU	lossu	lossv	ρ	nu	nv
Identity matrix												
500, 800, 800	**0.265**	3.955e-3	4.635e-3	0.900	4	4.5	0.737	3.955e-3	4.639e-3	0.900	4	4.5
1,000, 800, 800	**0.395**	2.477e-3	2.350e-3	0.899	4	4.5	1.240	2.470e-3	2.347e-3	0.899	4	4.5
500, 1,600, 1,600	**0.990**	6.071e-3	4.247e-3	0.898	5	4.5	2.475	6.050e-3	4.240e-3	0.898	5	4.5
1,000, 1,600, 1,600	**1.244**	1.351e-3	2.081e-3	0.900	5	5	3.880	1.350e-3	2.078e-3	0.900	5	5
Toeplitz matrix												
500, 800, 800	**0.279**	3.569e-3	5.570e-3	0.902	7	5.5	0.821	3.567e-3	5.570e-3	0.902	7	5.5
1,000, 800, 800	**0.395**	2.152e-3	2.165e-3	0.902	5	5	1.337	2.151e-3	2.159e-3	0.902	5	5
500, 1,600, 1,600	**0.955**	5.802e-3	4.758e-3	0.896	4	4.5	2.600	5.800e-3	4.751e-3	0.896	4	4.5
1,000, 1,600, 1,600	**1.172**	1.913e-3	1.602e-3	0.901	5	5.5	3.644	1.913e-3	1.604e-3	0.901	5	5.5
Sparse inverse matrix												
500, 800, 800	**0.527**	7.749e-3	1.248e-2	0.896	7	6.5	0.815	7.509e-3	1.209e-2	0.896	6.5	7
1,000, 800, 800	**0.618**	5.920e-3	4.631e-3	0.898	5	5	1.630	5.843e-3	4.624e-3	0.898	5	5
500, 1,600, 1,600	**1.589**	9.624e-3	1.052e-2	0.889	5	5	2.822	1.010e-2	1.031e-2	0.889	5	5
1,000, 1,600, 1,600	**1.951**	2.799e-3	3.812e-3	0.900	6.5	6	4.583	2.941e-3	3.807e-3	0.900	6.5	6

Note. CPU times are given in seconds; lossu and lossv are defined the same way as in Suo et al. (2017); nu and nv denote the number of nonzeros in u and v after setting their entries whose magnitudes are smaller than 10–4 to zero; and ρ denotes the canonical correlation computed from the solution returned by the algorithms. The best ones among the four algorithms are highlighted using bold text.

We report the numerical results in Table 3, where lossu and lossv are defined the same way as in Suo et al. (2017), nu and nv denote the number of nonzeros in u and v after setting their entries whose magnitudes are smaller than 10^{-4} to zero, and ρ denotes the canonical correlation computed from the solution returned by the algorithms. All reported values in Table 3 are the medians from 20 repetitions. From Table 3, we see that A-ManPG and AMA + LADMM achieve similar loss function values lossu and lossv, but A-ManPG is usually faster than AMA+LADMM, and for some cases, it is even two to three times faster. More important, AMA + LADMM lacks convergence analysis, but A-ManPG is guaranteed to converge to a stationary point (see the

Table 4. Losses Returned from the First-Stage Problem

	Init-1		Init-100	
(n, p, q)	lossu	lossv	lossu	lossv
Identity matrix				
200, 300, 300	0.304	0.374	0.107	0.124
500, 300, 300	0.114	0.103	0.050	0.037
200, 600, 600	0.394	0.393	0.146	0.116
500, 600, 600	0.137	0.139	0.048	0.035
Toeplitz matrix				
200, 300, 300	0.318	0.375	0.120	0.107
500, 300, 300	0.126	0.090	0.038	0.028
200, 600, 600	0.427	0.401	0.103	0.110
500, 600, 600	0.101	0.133	0.028	0.039
Sparse inverse matrix				
200, 300, 300	0.609	0.658	0.253	0.281
500, 300, 300	0.231	0.191	0.098	0.085
200, 600, 600	0.837	0.749	0.328	0.233
500, 600, 600	0.311	0.318	0.102	0.118

Note. The first-stage problem is given in (E.3) in the online appendix; lossu and lossv are defined the same way as in Suo et al. (2017).

Table 5. Comparison of A-ManPG and CoLaR for Sparse CCA

	A-ManPG-1					CoLaR-1					CoLaR-100				
b	0.8	1	1.2	1.4	1.6	0.8	1	1.2	1.4	1.6	0.8	1	1.2	1.4	1.6
								$(n,p,q) = (200,300,300)$							
CPU	0.387	0.320	0.288	**0.272**	0.274	0.763	0.760	0.725	0.667	0.619	5.071	5.020	4.844	4.724	4.663
lossu	0.064	0.043	0.038	**0.036**	0.045	0.094	0.062	0.049	0.046	0.051	0.081	0.051	0.038	0.041	0.047
lossv	0.075	0.053	0.044	**0.047**	0.058	0.110	0.079	0.072	0.080	0.094	0.096	0.063	0.061	0.059	0.072
nA	44	23.5	15.5	**10**	10	64	32.5	18	12	10	64	30	16.5	10	10
nB	45.5	24.5	16	**10**	10	64	32.5	18	12	11	63	32	19	12	10
ρ_1	0.919	0.907	0.900	**0.897**	0.894	0.925	0.910	0.900	0.895	0.893	0.925	0.911	0.900	0.897	0.895
ρ_2	0.863	0.833	0.822	**0.813**	0.811	0.877	0.840	0.822	0.813	0.804	0.879	0.841	0.821	0.815	0.810
								$(n,p,q) = (500,300,300)$							
CPU	0.300	0.275	0.264	**0.263**	0.265	0.693	0.680	0.612	0.524	0.431	2.995	2.924	2.780	2.653	2.647
lossu	0.031	0.021	0.017	**0.018**	0.020	0.032	0.018	0.017	0.018	0.019	0.032	0.018	0.015	0.017	0.020
lossv	0.031	0.019	0.018	**0.019**	0.021	0.038	0.023	0.022	0.023	0.022	0.037	0.022	0.019	0.019	0.021
nA	62	25	14	**10**	10	67	28.5	16	10.5	10	63	28.5	15	10	10
nB	58	30	16.5	11	**10**	64.5	31.5	17.5	11	10	69	31.5	18	12	10
ρ_1	0.907	0.901	0.898	**0.897**	0.897	0.906	0.901	0.898	0.897	0.896	0.907	0.901	0.898	0.897	0.896
ρ_2	0.833	0.816	0.808	**0.804**	0.803	0.838	0.817	0.808	0.804	0.803	0.838	0.818	0.808	0.804	0.802
								$(n,p,q) = (200,600,600)$							
CPU	1.329	1.133	1.062	1.021	0.992	1.441	1.373	1.321	1.229	1.217	63.371	63.236	63.092	62.835	62.706
lossu	0.101	0.068	0.056	0.062	**0.070**	0.182	0.117	0.094	0.097	0.103	0.141	0.095	0.071	0.071	0.085
lossv	0.091	0.069	0.059	0.057	**0.073**	0.162	0.115	0.093	0.093	0.091	0.127	0.085	0.065	0.066	0.081
nA	54.5	31	18	12	**10**	91.5	49.5	23	16	12	78	37	18	12	10
nB	55	29.5	18	13	10.5	100	49.5	25.5	15	12	78	35	21	13.5	**10**
ρ_1	0.926	0.915	0.910	0.906	**0.903**	0.934	0.918	0.907	0.904	0.903	0.932	0.912	0.905	0.903	0.902
ρ_2	0.879	0.843	0.821	0.804	**0.798**	0.903	0.858	0.823	0.808	0.795	0.904	0.852	0.823	0.805	0.799
								$(n,p,q) = (500,600,600)$							
CPU	1.094	1.019	0.989	**0.976**	0.978	1.385	1.327	1.270	1.149	1.042	17.822	17.734	17.649	17.488	17.289
lossu	0.032	0.020	0.016	**0.018**	0.020	0.041	0.023	0.018	0.017	0.019	0.041	0.024	0.017	0.018	0.020
lossv	0.032	0.018	0.014	**0.015**	0.016	0.041	0.023	0.016	0.017	0.017	0.039	0.019	0.016	0.015	0.017
nA	78	36	16	12	**10**	98	37	17.5	12	10	99	40	17	12	10
nB	74.5	32	16	**10**	10	93	37.5	15.5	10	10	98.5	39.5	16	10	10
ρ_1	0.914	0.906	0.904	**0.903**	0.903	0.916	0.906	0.903	0.903	0.902	0.917	0.907	0.904	0.903	0.902
ρ_2	0.846	0.822	0.807	**0.803**	0.802	0.852	0.824	0.809	0.804	0.803	0.855	0.823	0.808	0.803	0.802

Notes. Covariance matrix: identity matrix. The best ones among the four algorithms are highlighted using bold text.

Table 6. Comparison of A-ManPG and CoLaR for Sparse CCA

	A-ManPG-1					CoLaR-1					CoLaR-100				
b	0.8	1	1.2	1.4	1.6	0.8	1	1.2	1.4	1.6	0.8	1	1.2	1.4	1.6
						$(n,p,q) = (200,300,300)$									
CPU	0.380	0.327	0.292	0.283	**0.282**	0.791	0.761	0.729	0.666	0.622	8.229	8.079	7.931	7.801	7.725
lossu	0.069	0.045	0.043	0.049	**0.064**	0.103	0.079	0.069	0.070	0.085	0.088	0.054	0.043	0.049	0.061
lossv	0.075	0.057	0.046	0.050	**0.060**	0.116	0.079	0.065	0.067	0.075	0.096	0.066	0.056	0.055	0.062
nA	43	26	15.5	12	**10**	61.5	31	18.5	12	10	62.5	31.5	17	12	10
nB	44.5	27	16	12	**10**	64.5	36	20	14	12	57	30	19	12	10
ρ_1	0.921	0.911	0.906	0.902	**0.898**	0.925	0.912	0.905	0.902	0.900	0.926	0.912	0.906	0.902	0.899
ρ_2	0.864	0.835	0.818	0.803	0.794	0.869	0.839	0.818	0.803	0.797	0.875	0.838	0.814	**0.800**	0.792
						$(n,p,q) = (500,300,300)$									
CPU	0.310	0.287	0.266	**0.261**	0.260	0.707	0.667	0.646	0.492	0.431	3.220	3.160	3.067	2.858	2.839
lossu	0.025	0.015	0.010	**0.010**	0.010	0.029	0.017	0.012	0.013	0.014	0.030	0.017	0.012	0.010	0.012
lossv	0.027	0.016	0.012	**0.010**	0.012	0.031	0.015	0.012	0.011	0.013	0.035	0.019	0.013	0.012	0.014
nA	54	27	14.5	**10**	10	60.5	25.5	15	12	10	63.5	28	16	10	10
nB	56.5	28	16	**10**	10	63	31	18	10	10	65.5	33.5	17.5	11	10
ρ_1	0.905	0.899	0.896	**0.896**	0.895	0.906	0.900	0.897	0.896	0.896	0.906	0.900	0.897	0.896	0.895
ρ_2	0.835	0.819	0.810	**0.807**	0.806	0.838	0.820	0.810	0.807	0.805	0.840	0.822	0.810	0.807	0.806
						$(n,p,q) = (200,600,600)$									
CPU	1.427	1.214	1.120	1.048	**1.034**	1.504	1.445	1.343	1.273	1.272	64.845	64.700	64.465	64.460	64.255
lossu	0.077	0.055	0.050	0.051	**0.059**	0.158	0.108	0.077	0.079	0.090	0.106	0.068	0.056	0.059	0.069
lossv	0.079	0.063	0.044	0.044	**0.047**	0.158	0.112	0.112	0.105	0.116	0.105	0.075	0.055	0.052	0.064
nA	60	35	20	12	**10**	114	56	31.5	16	12	92.5	45	20	12	10
nB	59.5	33	20	12	**10**	104	53.5	25	16	12	86	39	20	12.5	10
ρ_1	0.925	0.911	0.903	**0.900**	0.897	0.936	0.915	0.901	0.897	0.894	0.933	0.913	0.902	0.899	0.896
ρ_2	0.879	0.842	0.815	**0.797**	**0.789**	0.896	0.853	0.823	0.796	0.788	0.900	0.851	0.816	0.796	0.786
						$(n,p,q) = (500,600,600)$									
CPU	1.142	1.070	1.029	1.019	**1.011**	1.419	1.349	1.290	1.178	1.071	16.082	15.965	15.844	15.795	15.676
lossu	0.033	0.022	0.018	**0.018**	0.020	0.041	0.022	0.015	0.014	0.015	0.042	0.022	0.019	0.019	0.022
lossv	0.031	0.018	0.014	**0.014**	0.017	0.034	0.018	0.012	0.011	0.012	0.040	0.019	0.013	0.013	0.016
nA	79.5	37.5	16	11	**10**	92.5	38.5	18.5	12	10	93.5	38	16	12	10
nB	77.5	34.5	16	12	**10**	91	36	17.5	11	10	93.5	38	17	12	10
ρ_1	0.913	0.904	0.902	**0.900**	0.898	0.913	0.904	0.902	0.900	0.899	0.915	0.904	0.902	0.900	0.899
ρ_2	0.840	0.816	0.806	**0.801**	0.799	0.846	0.818	0.806	0.802	0.800	0.847	0.819	0.807	0.800	0.798

Notes. Covariance matrix: Topelitz matrix. The best ones among the four algorithms are highlighted using bold text.

online appendix). Furthermore, AMA + LADMM is time-consuming for the multiple sparse CCA (10), but A-ManPG is suitable for (10), as we show in the next section.

4.3. Sparse CCA: Matrix Case

In this section, we apply the A-ManPG algorithm to solve the multiple sparse CCA (10) and compare its performance with the Convex program with group-LASSO Refinement (CoLaR) method proposed by Gao et al. (2017). CoLaR is a two-stage method based on convex relaxations. In all tests, we chose $r = 2$ and generated $\Sigma_{xy} = \Sigma_x U \Lambda V^\top \Sigma_y$, where $\Lambda \in \mathbb{R}^{r\times r}$ is a diagonal matrix with diagonal entries $\Lambda_{11} = 0.9$ and $\Lambda_{22} = 0.8$. The nonzero rows of both U and V are set at the $\{1,6,11,16,21\}$th rows. The values at the nonzero coordinates are obtained from normalizing (with respect to Σ_x and Σ_y) random numbers drawn from the uniform distribution on the finite set $\{-2,-1,0,1,2\}$. The two data sets $X \in \mathbb{R}^{n\times p}$ and $Y \in \mathbb{R}^{n\times q}$ are generated in the same way as in Section 4.2. The matrices X and Y are both divided by $\sqrt{n-1}$ such that $X^\top Y$ is the estimated covariance matrix. The loss between the estimation A and the ground truth U is measured by the subspace distance lossu $= \|P_U - P_A\|_F^2$, where P_U denotes the projection matrix onto the column space of U. Similarly, the loss for B and V is measured as lossv $= \|P_V - P_B\|_F^2$.

The codes of CoLaR were downloaded from the authors' web page.[6] We used all default settings of their codes. In particular, ADMM is used to solve the first-stage problem ((E.3) in the online appendix), and it is terminated when it does not make much progress or it reaches the maximum iteration number 100. For our A-ManPG algorithm, we run only one iteration of ADMM for (E.3) and use the returned solution as the initial point of A-ManPG, because we found that this already generates a very good solution for A-ManPG. To be fair, we also compare the same case for CoLaR, where only one iteration of ADMM is used for (E.3).

Table 7. Comparison of A-ManPG and CoLaR for Sparse CCA

	A-ManPG-1					CoLaR-1					CoLaR-100				
b	0.8	1	1.2	1.4	1.6	0.8	1	1.2	1.4	1.6	0.8	1	1.2	1.4	1.6
								$(n,p,q) = (200,300,300)$							
CPU	0.810	0.654	0.576	**0.538**	0.547	0.960	0.947	0.909	0.791	0.790	10.378	10.265	10.106	10.110	10.074
lossu	0.088	0.080	0.091	**0.113**	0.138	0.178	0.151	0.135	0.130	0.147	0.130	0.107	0.099	0.114	0.148
lossv	0.115	0.111	0.127	**0.157**	0.196	0.200	0.180	0.179	0.171	0.192	0.140	0.125	0.128	0.137	0.166
nA	43	24.5	16	12	**11**	79.5	50.5	34	23.5	18	59	36	23	17	13
nB	39	24.5	16	12	**10**	71.5	47	30	22	15	53	32	17	14	12
ρ_1	0.919	0.909	0.899	**0.893**	0.887	0.928	0.915	0.902	0.893	0.884	0.924	0.911	0.902	0.895	0.889
ρ_2	0.854	0.829	0.813	**0.803**	0.795	0.883	0.857	0.838	0.824	0.810	0.867	0.841	0.819	0.804	0.793
								$(n,p,q) = (500,300,300)$							
CPU	0.574	0.513	0.494	**0.472**	0.453	0.940	0.906	0.833	0.746	0.721	4.681	4.619	4.564	4.424	4.404
lossu	0.038	0.036	0.040	**0.051**	0.065	0.046	0.044	0.046	0.054	0.068	0.042	0.043	0.048	0.061	0.076
lossv	0.035	0.029	0.032	**0.042**	0.052	0.050	0.039	0.036	0.045	0.049	0.040	0.029	0.033	0.039	0.045
nA	46	25.5	14.5	**10**	10	64.5	38	23.5	14.5	11	57	30	15	11	10
nB	47.5	26	16	12	**10**	76.5	42	25.5	18	13	66	38	19	13	11
ρ_1	0.907	0.902	0.899	**0.897**	0.896	0.909	0.903	0.900	0.897	0.894	0.907	0.902	0.898	0.895	0.893
ρ_2	0.824	0.812	0.803	**0.800**	0.797	0.833	0.818	0.810	0.805	0.799	0.829	0.815	0.807	0.801	0.795
								$(n,p,q) = (200,600,600)$							
CPU	2.129	1.906	1.789	**1.683**	1.613	1.793	1.671	1.614	1.529	1.485	65.508	65.392	65.229	65.143	65.040
lossu	0.168	0.160	0.164	**0.164**	0.178	0.323	0.271	0.257	0.252	0.268	0.211	0.184	0.183	0.219	0.273
lossv	0.142	0.127	0.119	**0.128**	0.156	0.349	0.297	0.283	0.288	0.317	0.175	0.148	0.135	0.150	0.168
nA	50	29.5	20	13	**12**	131	81.5	55	31	23	90	44.5	22.5	16	13
nB	50	30	19	14	**10**	136.5	89	61	41	26	88.5	50	26.5	19.5	12.5
ρ_1	0.922	0.909	0.902	**0.899**	0.896	0.941	0.926	0.916	0.905	0.897	0.931	0.913	0.903	0.898	0.894
ρ_2	0.870	0.840	0.818	**0.801**	0.788	0.914	0.881	0.847	0.820	0.800	0.888	0.854	0.828	0.811	0.796
								$(n,p,q) = (500,600,600)$							
CPU	1.777	1.605	1.536	**1.467**	1.454	1.690	1.635	1.598	1.546	1.486	39.566	39.440	39.320	39.247	39.180
lossu	0.044	0.035	0.039	**0.049**	0.061	0.075	0.057	0.054	0.057	0.063	0.052	0.043	0.046	0.052	0.064
lossv	0.045	0.033	0.034	**0.042**	0.053	0.058	0.047	0.049	0.051	0.059	0.051	0.037	0.037	0.043	0.053
nA	69	33	17	12	**10**	120.5	63.5	34.5	20	13	83.5	40	18	13	10
nB	64	33.5	19	12	**10**	112.5	57.5	29	18	14	92	39.5	20.5	12.5	10
ρ_1	0.909	0.901	**0.897**	0.896	0.895	0.919	0.908	0.901	0.898	0.895	0.914	0.903	0.898	0.896	0.895
ρ_2	0.833	0.813	**0.802**	0.796	0.792	0.849	0.822	0.807	0.800	0.793	0.838	0.816	0.803	0.794	0.789

Notes. Covariance matrix: sparse inverse matrix. The best ones among the four algorithms are highlighted using bold text.

The parameter τ in (E.3) is set to $\tau = 0.55\sqrt{\log(p+q)/n}$. We set $\tau_1 = \tau_2 = \frac{1}{2}b\sqrt{\log(p+q)/n}$ in (10) and $\tau' = b$ in (E.4) (in the online appendix), where b was set to $b = \{0.8, 1, 1.2, 1.4, 1.6\}$. For each b, we solved (10) by A-ManPG with $\delta = 10^{-4}, \gamma = 0.5, t_1 = t_2 = 1$. The A-ManPG algorithm was stopped if $\max\{\|D_k^A\|_F^2, \|D_k^B\|_F^2\} \le 10^{-8}$ and the ASSN algorithm was stopped if $\|E(\Lambda_k)\|_F \le 10^{-5}$ in (25).

We report the numerical results in Tables 4–7, where CPU times are in seconds, and nA and nB denote the number of nonzeros of A and B, respectively, after truncating the entries whose magnitudes are smaller than 10^{-4} to zeros. The scalars ρ_1 and ρ_2 are the two canonical correlations, and they should be close to 0.9 and 0.8, respectively. All reported values in Tables 4–7 are the medians from 20 repetitions. More specifically, Table 4 reports the results obtained from the first stage where ADMM was used to solve (E.3). We use Init-1 to indicate that we only run one iteration of ADMM, and Init-100 to indicate the case where we run ADMM until it does not make much progress or the maximum iteration number 100 is reached. From Table 4, we see that solving the first-stage problem by running ADMM for 100 iterations indeed improves the two losses significantly. Tables 5–7 report the results for the three different types of covariance matrices. A-ManPG-1 and CoLaR-1 are the versions where we run only one iteration of ADMM for the first stage, and CoLaR-100 is the version where the first-stage problem ((E.3) in the online appendix) is solved more accurately by ADMM, as discussed earlier. We observed that running more iterations of ADMM in the first stage does not help much for A-ManPG; we thus only report the results of A-ManPG-1. From Tables 5–7, we see that CoLaR-100 gives much better results than CoLaR-1 in terms of the two losses lossu and lossv, especially when the sample size is relatively small compared with the matrix sizes. Moreover, we see that A-ManPG-1 outperforms both CoLaR-1 and CoLaR-100 significantly. In particular, A-ManPG-1 generates comparable and often better solutions than CoLaR-1 and

CoLaR-100 in terms of solution sparsity and losses lossu and lossv. Furthermore, A-ManPG-1 is usually faster than CoLaR-1 and much faster than CoLaR-100.

5. Conclusion

In this paper, we proposed an efficient algorithm for solving two important and numerically challenging optimization problems arising from statistics: sparse PCA and sparse CCA. These two problems are challenging to solve because they are manifold optimization problems with nonsmooth objectives, a topic that is still underdeveloped in optimization. We proposed an A-ManPG method to solve these two problems. Convergence and convergence rate to a stationary point of the proposed algorithm are established. Numerical results on statistical data demonstrate that A-ManPG is comparable to existing algorithms for solving sparse PCA and is significantly better than existing algorithms for solving sparse CCA.

Acknowledgments

The authors thank the anonymous referees for insightful and constructive comments that greatly improved the presentation of this paper.

Endnotes

[1] Without KL inequality, only subsequence convergence is obtained.

[2] The definition is given in the online appendix.

[3] The proximal mapping of $\ell_p (p \geq 1)$ norm is strongly semismooth (Facchinei and Pang 2007, Ulbrich 2011). From (Ulbrich 2011, proposition 2.26), if $F : V \to \mathbb{R}^m$ is a piecewise $\mathcal{C}^1$ (piecewise smooth) function, then F is semismooth. If F is a piecewise $\mathcal{C}^2$ function, then F is strongly semismooth. It is known that proximal mappings of many interesting functions are piecewise linear or piecewise smooth.

[4] We say a matrix A is positive semidefinite if $A + A^\top$ is positive semidefinite.

[5] Our MATLAB code is available at https://github.com/chenshixiang/AManPG.

[6] See http://www-stat.wharton.upenn.edu/~zongming/research.html.

References

Absil PA, Mahony R, Sepulchre R (2009) *Optimization Algorithms on Matrix Manifolds* (Princeton University Press, Princeton, NJ).

Attouch H, Bolte J, Redont P, Soubeyran A (2010) Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality. *Math. Oper. Res.* 35(2):438–457.

Baik J, Silverstein JW (2006) Eigenvalues of large sample covariance matrices of spiked population models. *J. Multivariate Anal.* 97(6):1382–1408.

Beck A, Teboulle M (2009) A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.* 2(1):183–202.

Bendory T, Eldar YC, Boumal N (2018) Non-convex phase retrieval from STFT measurements. *IEEE Trans. Inform. Theory* 64(1):467–484.

Bolte J, Sabach S, Teboulle M (2014) Proximal alternating linearized minimization or nonconvex and nonsmooth problems. *Math. Programming* 146(1–2):459–494.

Boumal N (2016) Nonconvex phase synchronization. *SIAM J. Optim.* 26(4):2355–2377.

Boumal N, Absil PA (2011) RTRMC: A Riemannian trust-region method for low-rank matrix completion. Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ, eds. *Proc. 24th Internat. Conf. Neural Inform. Processing Systems* (Curran Associates, Red Hook, NY), 406–414.

Chen M, Gao C, Ren Z, Zhou HH (2013) Sparse CCA via precision adjusted iterative thresholding. Preprint, submitted November 24, https://arxiv.org/abs/1311.6186.

Chen S, Ma S, So AMC, Zhang T (2020) Proximal gradient method for nonsmooth optimization over the Stiefel manifold. *SIAM J. Optim.* 30(1):210–239.

Cherian A, Sra S (2017) Riemannian dictionary learning and sparse coding for positive definite matrices. *IEEE Trans. Neural Networks Learn. Systems* 28(12):2859–2871.

d'Aspremont A (2011) Identifying small mean-reverting portfolios. *Quant. Finance* 11(3):351–364.

d'Aspremont A, Bach F, Ghaoui LE (2008) Optimal solutions for sparse principal component analysis. *J. Machine Learn. Res.* 9:1269–1294.

d'Aspremont A, Ghaoui LE, Jordan MI, Lanckriet GRG (2007) A direct formulation for sparse PCA using semidefinite programming. *SIAM Rev.* 49(3):434–448.

Erichson NB, Zheng P, Manoharz K, Brunton SL, Kutz JN, Aravkin AY (2020) Sparse principal component analysis via variable projection. *SIAM J. Appl. Math.* Forthcoming.

Facchinei F, Pang J (2007) *Finite-Dimensional Variational Inequalities and Complementarity Problems* (Springer Science & Business Media, Berlin).

Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. *J. Statist. Software* 33(1):1–22.

Gao C, Ma Z, Zhou HH (2017) Sparse CCA: Adaptive estimation and computational barriers. *Ann. Statist.* 45(5):2074–2101.

Hardoon DR, Shawe-Taylor J (2011) Sparse canonical correlation analysis. *Machine Learn.* 83(3):331–353.

Hiriart-Urruty JB, Strodiot JJ, Nguyen VH (1984) Generalized Hessian matrix and second-order optimality conditions for problems with $C^{1,1}$ data. *Appl. Math. Optim.* 11(1):43–56.

Hotelling H (1936) Relations between two sets of variates. *Biometrika* 28(3–4):321–377.

Huang W, Hand P (2018) Blind deconvolution by a steepest descent algorithm on a quotient manifold. *SIAM J. Imaging Sci.* 11(4):2757–2785.

Jolliffe I, Trendafilov N, Uddin M (2003) A modified principal component technique based on the LASSO. *J. Comput. Graph. Statist.* 12(3):531–547.

Journee M, Nesterov Y, Richtarik P, Sepulchre R (2010) Generalized power method for sparse principal component analysis. *J. Machine Learn. Res.* 11(February):517–553.

Li X, Sun D, Toh KC (2018) A highly efficient semismooth Newton augmented Lagrangian method for solving lasso problems. *SIAM J. Optim.* 28(1):433–458.
Liu H, Yue MC, So AMC (2017) On the estimation performance and convergence rate of the generalized power method for phase synchronization. *SIAM J. Optim.* 27(4):2426–2446.
Lu Z, Zhang Y (2012) An augmented Lagrangian approach for sparse principal component analysis. *Math. Programming* 135:149–193.
Ma S (2013) Alternating direction method of multipliers for sparse principal component analysis. *J. Oper. Res. Soc. China* 1(2):253–274.
Magnus JR, Neudecker H (2018) *Matrix Differential Calculus with Applications in Statistics and Econometrics* (Wiley, Hoboken, NJ).
Mifflin R (1977) Semismooth and semiconvex functions in constrained optimization. *SIAM J. Control Optim.* 15(6):959–972.
Moghaddam B, Weiss Y, Avidan S (2005) Spectral bounds for sparse PCA: Exact and greedy algorithms. Weiss Y, Scholkopf B, Platt JC, eds. *Proc. 18th Internat. Conf. Neural Inform. Processing Systems* (MIT Press, Cambridge, MA), 915–922.
Nadler B (2008) Finite sample approximation results for principal component analysis: A matrix perturbation approach. *Ann. Statist.* 36(6):2791–2817.
Parkhomenko E, Tritchler D, Beyene J (2009) Sparse canonical correlation analysis with application to genomic data integration. *Statist. Appl. Genetics Molecular Biol.* 8(1):1–34.
Paul D (2007) Asymptotics of sample eigenstructure for a large dimensional spiked covariance model. *Statistica Sinica* 17:1617–1642.
Pearson K (1901) LIII. On lines and planes of closest fit to systems of points in space. *London Edinburgh Dublin Philos. Magazine J. Sci.* 2(11):559–572.
Qi H, Sun D (2011) An augmented Lagrangian dual approach for the H-weighted nearest correlation matrix problem. *IMA J. Numer. Anal.* 31(2):491–511.
Qi L, Sun J (1993) A nonsmooth version of Newton's method. *Math. Programming* 58:353–367.
Shen H, Huang JZ (2008) Sparse principal component analysis via regularized low rank matrix approximation. *J. Multivariate Anal.* 99(6):1015–1034.
Solodov MV, Svaiter BF (1998) A globally convergent inexact Newton method for systems of monotone equations. *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods* (Springer, Dordrecht), 355–369.
Sun J, Qu Q, Wright J (2017) Complete dictionary recovery over the sphere I: Overview and the geometric picture. *IEEE Trans. Inform. Theory* 63(2):853–884.
Sun J, Qu Q, Wright J (2018) A geometrical analysis of phase retrieval. *Foundations Comput. Math.* 18(5):1131–1198.
Suo X, Minden V, Nelson B, Tibshirani R, Saunders M (2017) Sparse canonical correlation analysis. Preprint, submitted May 30, https://arxiv.org/abs/1705.10865.
Ulbrich M (2011) *Semismooth Newton Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces*, vol. 11 (SIAM, Philadelphia).
Vandereycken B (2013) Low-rank matrix completion by Riemannian optimization. *SIAM J. Optim.* 23(2):1214–1236.
Vu VQ, Cho J, Lei J, Rohe K (2013) Fantope projection and selection: A near-optimal convex relaxation of sparse PCA. Burges CJC, Bouttou L, Welling M, Ghahramani Z, Weinberger KQ, eds. *Proc. 26th Internat. Conf. Neural Inform. Processing Systems* (Curran Associates, Red Hook, NY), 2670–2678.
Wang C, Sun D, Toh KC (2010) Solving log-determinant optimization problems by a Newton-CG primal proximal point algorithm. *SIAM J. Optim.* 20(6):2994–3013.
Wiesel A, Kliger M, Hero AO III (2008) A greedy approach to sparse canonical correlation analysis. Preprint, submitted January 17, arXiv:0801.2748.
Witten DM, Tibshirani R, Hastie T (2009) A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatist.* 10(3):515–534.
Xiao X, Li Y, Wen Z, Zhang L (2018) A regularized semi-smooth Newton method with projection steps for composite convex programs. *J. Sci. Comput.* 76(1):364–389.
Yang J, Sun D, Toh KC (2013) A proximal point algorithm for log-determinant optimization with group lasso regularization. *SIAM J. Optim.* 23(2):857–893.
Yang L, Sun D, Toh KC (2015) SDPNAL+: A majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Math. Programming Comput.* 7(3):331–366.
Yuan XT, Zhang T (2013) Truncated power method for sparse eigenvalue problems. *J. Machine Learn. Res.* 14(1):899–925.
Zhao X, Sun D, Toh KC (2010) A Newton-CG augmented Lagrangian method for semidefinite programming. *SIAM J. Optim.* 20(4):1737–1765.
Zhou G, Toh KC (2005) Superlinear convergence of a Newton-type algorithm for monotone equations. *J. Optim. Theory Appl.* 125(1):205–221.
Zou H, Xue L (2018) A selective overview of sparse principal component analysis. *Proc. IEEE* 106(8):1311–1320.
Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. *J. Roy. Statist. Soc. B* 67(2):301–320.
Zou H, Hastie T, Tibshirani R (2006) Sparse principal component analysis. *J. Comput. Graphical Statist.* 15(2):265–286.