A Riemannian ADMM

Jiaxiang Li[1], Shiqian Ma[1,2], and Tejes Srivastava[1]

[1]Department of Mathematics, University of California, Davis
[2]Computational Applied Mathematics and Operations Research, Rice University

November 3, 2022

Abstract

We consider a class of Riemannian optimization problems where the objective is the sum of a smooth function and a nonsmooth function, considered in the ambient space. This class of problems finds important applications in machine learning and statistics such as the sparse principal component analysis, sparse spectral clustering, and orthogonal dictionary learning. We propose a Riemannian alternating direction method of multipliers (RADMM) to solve this class of problems. Our algorithm adopts easily computable steps in each iteration. The iteration complexity of the proposed algorithm for obtaining an ϵ-stationary point is analyzed under mild assumptions. To the best of our knowledge, this is the first Riemannian ADMM with provable convergence guarantee for solving Riemannian optimization problem with nonsmooth objective. Numerical experiments are conducted to demonstrate the advantage of the proposed method.

1 Introduction

Optimization over Riemannian manifolds has drawn a lot of attention due to its applications in machine learning and related disciplines, including low-rank matrix completion [6, 42], phase retrieval [3, 39], blind deconvolution [22] and dictionary learning [11, 37]. Riemannian optimization aims at minimizing an objective function over a Riemannian manifold. When the objective function is smooth, people have proposed to solve them using Riemannian gradient method, Riemannian quasi-Newton method, Riemannian trust-region method, etc. Work along this line has been summarized in the monographs [1, 5] as well as some other references. Recently, due to increasing demand from application areas such as machine learning, statistics, signal processing and so on, there is a line of work designing efficient and scalable algorithms for solving Riemannian optimization problems with nonsmooth objectives. For example, people have studied Riemannian subgradient method [29], Riemannian proximal gradient method [10, 21], Riemannian proximal point algorithm [9], Riemannian proximal-linear algorithm [44], zeroth-order Riemannian algorithms [28], and so on.

One thing that has not been widely considered is how to design alternating direction method of multipliers (ADMM) on manifolds. ADMM can be a perfect solver for the following nonsmooth optimization over Riemannian manifolds:

$$\begin{aligned} &\min_x F(x) := f(x) + g(Ax) \\ &\text{s.t. } x \in \mathcal{M}, \end{aligned} \tag{1}$$

where f is smooth and possibly nonconvex, g is nonsmooth but convex, $\mathcal{M}$ is an embedded submanifold in $\mathbb{R}^n$, and matrix $A \in \mathbb{R}^{m\times n}$. Throughout this paper, the smoothness, Lipschitz

continuity, and convexity of functions are interpreted as the functions are being considered in the ambient Euclidean space. If $\mathcal{M} = \mathbb{R}^n$, then problem (1) reduces to the Eucldiean case, and there exist efficient methods such as proximal gradient method, accelerated proximal gradient method, and ADMM for solving it. If the nonsmooth function vanishes, i.e., $g \equiv 0$, then problem (1) reduces to a smooth problem over manifold, and it can be solved by various methods for smooth Riemannian optimization. Therefore, the main challenge of solving (1) lies in the fact that there exist both manifold constraint and nonsmooth objective in the problem. As a result, a very natural idea to deal with this situation is to split the difficulty caused by the manifold constraint and nonsmooth objective. In particular, one can introduce an auxilliary variable y and rewrite (1) equivalently as

$$\begin{aligned} &\min_{x,y} \ f(x) + g(y) \\ &\text{s.t. } Ax = y, \ x \in \mathcal{M}. \end{aligned} \tag{2}$$

ADMM is a good candidate for solving (2), because it can deal with the nonsmooth objective and the manifold constraint separately and alternatingly.

The idea of splitting the nonsmooth objective and manifold constraint in (1) is not new. The first algorithm for this purpose is the SOC (splitting orthogonality constraints) algorithm proposed by Lai and Osher [26]. SOC for solving (1) splits the problem in the following way:

$$\begin{aligned} &\min_{x,y} \ f(x) + g(Ax) \\ &\text{s.t. } x = y, \ y \in \mathcal{M}, \end{aligned} \tag{3}$$

and iterates as follows:

$$\begin{aligned} x^{k+1} &:= \ \text{argmin}_x \, f(x) + g(Ax) + \langle \lambda^k, x - y^k \rangle + \frac{\rho}{2}\|x - y^k\|_2^2 \\ y^{k+1} &:= \ \text{argmin}_{y \in \mathcal{M}} \langle \lambda^k, x^{k+1} - y \rangle + \frac{\rho}{2}\|x^{k+1} - y\|_2^2 \\ \lambda^{k+1} &:= \lambda^k + \rho(x^{k+1} - y^{k+1}), \end{aligned} \tag{4}$$

where λ denotes the Lagrange multiplier and $\rho > 0$ is a penalty parameter. Note that the x-subproblem in (4) is an unconstrained problem, which can be solved by proximal gradient method and many others, and the y-subproblem corresponds to a projection onto the manifold $\mathcal{M}$. A closely related algorithm named MADMM (manifold ADMM), proposed in [25] for solving (2), iterates as follows:

$$\begin{aligned} x^{k+1} &:= \ \text{argmin}_{x \in \mathcal{M}} \, f(x) + \langle \lambda^k, Ax - y^k \rangle + \frac{\rho}{2}\|Ax - y^k\|_2^2 \\ y^{k+1} &:= \ \text{argmin}_y \, g(y) + \langle \lambda^k, Ax^{k+1} - y \rangle + \frac{\rho}{2}\|Ax^{k+1} - y\|_2^2 \\ \lambda^{k+1} &:= \lambda^k + \rho(Ax^{k+1} - y^{k+1}). \end{aligned} \tag{5}$$

In (5), the x-subproblem is a Riemannian optimization with smooth objective which can be solved by Riemannian gradient method, and the y-subproblem corresponds to the proximal mapping of function g. However, there lacks convergence guarantees for both SOC and MADMM.

When the nonsmooth term in (1) vanishes, i.e., $g \equiv 0$, an ADMM for nonconvex optimization can be used to solve (1) as illustrated in [43]. Since $g \equiv 0$, the problem (1) reduces to

$$\begin{aligned} &\min_{x,y} \ f(x) + I_{\mathcal{M}}(y) \\ &\text{s.t., } \ x = y, \end{aligned} \tag{6}$$

where $I_{\mathcal{M}}$ is the indicator function of manifold $\mathcal{M}$. The ADMM for solving (6) iterates as follows:

$$\begin{aligned} x^{k+1} &:= \operatorname{argmin}_x f(x) + \langle \lambda^k, x - y^k \rangle + \frac{\rho}{2}\|x - y^k\|_2^2 \\ y^{k+1} &:= \operatorname{argmin}_{y \in \mathcal{M}} \langle \lambda^k, x^{k+1} - y \rangle + \frac{\rho}{2}\|x^{k+1} - y\|_2^2 \\ \lambda^{k+1} &:= \lambda^k + \rho(x^{k+1} - y^{k+1}). \end{aligned} \tag{7}$$

The convergence of (7) is established in [43] under the assumption that f is Lipschitz differentiable. Note that the convergence only applies when $g \equiv 0$. The ADMM studied in [43] does not apply to (1) when the nonsmooth function g presents.

Another ADMM was proposed in [30] for solving a particular smooth Riemannian optimization problem: the sparse spectral clustering. This problem can be cast below.

$$\begin{aligned} &\min_{P,U} \ \langle L, UU^\top \rangle + g(P), \\ &\text{s.t. } P = UU^\top, U^\top U = I, \end{aligned} \tag{8}$$

where L is a given matrix, g is a smooth function that promotes the sparsity of $UU^\top$. The ADMM for solving (8) iterates as follows.

$$\begin{aligned} U^{k+1} &:= \operatorname{argmin}_{U^\top U = I} \langle L, UU^\top \rangle + \langle \Lambda^k, P^k - UU^\top \rangle + \frac{\rho}{2}\|P^k - UU^\top\|_F^2 \\ P^{k+1} &:= \operatorname{argmin}_P g(P) + \langle \Lambda^k, P - U^{k+1}(U^{k+1})^\top \rangle + \frac{\rho}{2}\|P - U^{k+1}(U^{k+1})^\top\|_F^2 \\ \Lambda^{k+1} &:= \Lambda^k + \rho(P^{k+1} - U^{k+1}(U^{k+1})^\top). \end{aligned} \tag{9}$$

Note that the ADMM in [30] requires the smoothness on the objective function as well, and it does not apply to the case where the objective function is nonsmooth. Zhang et al. [48] proposed a proximal ADMM which solves the following problem:

$$\begin{aligned} &\min \ f(x_1, \ldots, x_N) + \sum_{i=1}^{N-1} g_i(x_i) \\ &\text{s.t. } x_N = b - \sum_{i=1}^{N-1} A_i x_i \\ &\qquad x_i \in \mathcal{M}_i \cap \mathcal{X}_i, i = 1, \ldots, N-1, \end{aligned} \tag{10}$$

where f is a smooth function, g_i is a nonsmooth function, $\mathcal{M}_i$ is a Riemannian manifold, and $\mathcal{X}_i$ is a convex set. The authors of [48] established the iteration complexity of the proposed proximal ADMM for obtaining an ϵ-stationary point of (10). A notable requirement in (10) is that the last block variable (i.e., x_N) must not appear in the nonsmooth part of the objective, nor be subject to manifold constraints. This is in sharp contrast to problem (2), where one block variable is associated with the manifold constraint, and the other block variable is associate with the nonsmooth part of the objective.

Other than ADMM-type algorithms, there also exist some other algorithms for solving (1). Here we briefly discuss two of them: Riemannian subgradient method and Riemannian proximal gradient method. Because the objective function of (1) is nonsmooth, it is a natural idea to use Riemannian subgradient method [14, 4, 16, 17, 19, 18, 15, 29] to solve it. The Riemannian subgradient method for solving (1) updates the iterate by

$$x^{k+1} = \operatorname{Retr}_{x^k}(-\eta_k v^k),$$

where v^k is a Riemannian subgradient of F at $\mathcal{M}$, $\eta_k > 0$ is a stepsize, and Retr denotes the retraction operation. Convergence of this method is established in [14] when F is geodesically convex, and iteration complexity is analyzed in [29] when F is weakly convex over the Stiefel manifold. Another representative algorithm for solving (1) is the manifold proximal gradient method (ManPG), which was proposed recently by Chen et al. [10]. A typical iteration of ManPG is given below:

$$\begin{aligned} v^k &:= \operatorname{argmin}_{v \in \mathrm{T}_{x^k}\mathcal{M}} \langle \operatorname{grad} f(x^k), v\rangle + \frac{1}{2t}\|v\|^2 + g(A(x^k+v)) \\ x^{k+1} &:= \operatorname{Retr}_{x^k}(\alpha v^k), \end{aligned} \tag{11}$$

where $t > 0$ and $\alpha > 0$ are stepsizes, $\mathrm{T}_x\mathcal{M}$ denotes the tangent space of $\mathcal{M}$ at x, and $\operatorname{grad} f$ denotes the Riemannian gradient of f. Chen et al. [10] analyzed the iteration complexity of ManPG for obtaining an ϵ-stationary point of (1). Moreover, Chen et al. [10] suggested to solve the subproblem for determining v_k in (11) by a semi-smooth Newton method [10, 45].

Our contributions. In this paper, we propose a Riemannian ADMM (RADMM) for solving (2) based on a Moreau envelope smoothing technique. Our RADMM for solving (2) contains easily computable steps in each iteration. We analyze the iteration complexity of our RADMM for obtaining an ϵ-stationary point to (2) under mild assumptions. Numerical results of the proposed algorithm for solving sparse principal component analysis and dual principal component pursuit are reported, which demonstrate its superiority over exsiting methods.

Organizations. The rest of this paper is organized as follows. We propose our RADMM in Section 2, whose iteration complexity is analyzed in Section 3. Section 4 is devoted to applications and numerical experiments. We draw some concluding remarks in Section 5.

2 A Riemannian ADMM

In this section, we introduce our Riemannian ADMM algorithm. We first review some basics of Riemannian optimization.

2.1 Basics on Riemannian optimization

Let $\mathcal{M} \subset \mathbb{R}^n$ be a differentiable embedded submanifold. We have the following definition for the tangent space.

Definition 1 (Tangent space)**.** *Consider a manifold $\mathcal{M}$ embedded in a Euclidean space. For any $x \in \mathcal{M}$, the tangent space $\mathrm{T}_x\mathcal{M}$ at x is a linear subspace that consists of the derivatives of all differentiable curves on $\mathcal{M}$ passing through x:*

$$\mathrm{T}_x\mathcal{M} = \{\gamma'(0) : \gamma(0) = x, \gamma([-\delta, \delta]) \subset \mathcal{M} \text{ for some } \delta > 0, \gamma \text{ is differentiable}\}. \tag{12}$$

The manifold $\mathcal{M}$ is a Riemannian manifold if it is equipped with an inner product on the tangent space, $\langle \cdot, \cdot \rangle_x : \mathrm{T}_x\mathcal{M} \times \mathrm{T}_x\mathcal{M} \to \mathbb{R}$, that varies smoothly on $\mathcal{M}$. As an example, consider the Stiefel manifold $\mathcal{M} = \mathrm{St}(n,p) := \{X \in \mathbb{R}^{n\times p} : X^\top X = I_p\}$. The tangent space of $\mathrm{St}(n,p)$ is given by $\mathrm{T}_X\mathcal{M} = \{Y \in \mathbb{R}^{n\times p} : X^\top Y + Y^\top X = 0\}$. It is easy to verify that the projection onto the tangent space of $\mathrm{St}(n,p)$ is $\operatorname{Proj}_{\mathrm{T}_X\mathcal{M}}(Y) = (I - XX^\top)Y + X\operatorname{skew}(X^\top Y)$, where $\operatorname{skew}(A) := (A - A^\top)/2$. We refer to the monographs [1, 5] for more examples. We now introduce the concept of a Riemannian gradient.

Definition 2 (Riemannian Gradient). *Suppose f is a smooth function on $\mathcal{M}$. The Riemannian gradient* $\mathrm{grad} f(x)$ *is a vector in* $\mathrm{T}_x\mathcal{M}$ *satisfying* $\left.\frac{d(f(\gamma(t)))}{dt}\right|_{t=0} = \langle v, \mathrm{grad} f(x)\rangle_x$ *for any* $v \in \mathrm{T}_x\mathcal{M}$*, where* $\gamma(t)$ *is a curve as described in* (12).

Another useful concept is the retraction.

Definition 3 (Retraction). *A retraction mapping* Retr_x *is a smooth mapping from* $\mathrm{T}_x\mathcal{M}$ *to* $\mathcal{M}$ *(not necessary injective or surjective) such that:* $\mathrm{Retr}_x(0) = x$*, where* 0 *is the zero element of* $\mathrm{T}_x\mathcal{M}$*, and the differential of* Retr_x *at* 0 *is an identity mapping, i.e.,* $\left.\frac{d\mathrm{Retr}_x(t\eta)}{dt}\right|_{t=0} = \eta$*,* $\forall \eta \in \mathrm{T}_x\mathcal{M}$*. In particular, the exponential mapping* Exp_x *is a retraction that generates geodesics.*

In the theoretical analysis of our algorithm, we always assume that the retraction is injective from $\mathrm{T}_x\mathcal{M}$ to $\mathcal{M}$ for any point $x \in \mathcal{M}$, thus the existence of the inverse of the retraction function Retr_x^{-1} is guaranteed. For example, when $\mathcal{M}$ is complete, the exponential mapping Exp_x (which is a special example of retraction) is always defined for every $\xi \in \mathrm{T}_x\mathcal{M}$, and the inverse of the exponential mapping $\mathrm{Exp}_x^{-1}(y) \in \mathrm{T}_x\mathcal{M}$ (which is called the logarithm mapping), is always well defined for any $x, y \in \mathcal{M}$.

Throughout this paper, we consider the Riemannian metric on $\mathcal{M}$ that is induced from the Euclidean inner product; i.e., for any $\xi, \eta \in \mathrm{T}_x\mathcal{M}$, we have $\langle \xi, \eta\rangle_x = \mathrm{Tr}(\xi^\top \eta)$. The Euclidean gradient of a smooth function f is denoted as ∇f and the Riemannian gradient of f is denoted as $\mathrm{grad}\, f$. Note that by our choice of the Riemannian metric, we have $\mathrm{grad}\, f(x) = \mathrm{Proj}_{\mathrm{T}_x\mathcal{M}} \nabla f(x)$, the orthogonal projection of $\nabla f(x)$ onto the tangent space.

2.2 Our Riemannian ADMM

Now we are ready to introduce our RADMM algorithm. Our RADMM for solving (2) is based on the Moreau envelope smoothing technique. In particular, we consider to smooth the function g in (2) by adding a quadratic proximal term, which leads to:

$$\begin{aligned} &\min_{x,y,z}\ f(x) + g(y) + \frac{1}{2\gamma}\|y - z\|^2 \\ &\text{s.t. } Ax = z,\ x \in \mathcal{M}, \end{aligned} \tag{13}$$

where $\gamma > 0$ is a parameter. Equivalently, (13) can also be rewritten as

$$\begin{aligned} &\min_{x,z}\ f(x) + g_\gamma(z) \\ &\text{s.t. } Ax = z,\ x \in \mathcal{M}, \end{aligned} \tag{14}$$

where $g_\gamma(z) = \min_y \left\{ g(y) + \frac{1}{2\gamma}\|y - z\|^2 \right\}$ is the Moreau envelope of g [47], and it is known that g_γ is a smooth function when g is convex.

We need to point out that the idea of Moreau envelope smoothing has been proposed in [47] for solving the following problem in Euclidean space:

$$\min_x\ f(x) + g(x),\ \text{s.t.},\ Ax = b, \tag{15}$$

where f is smooth and g is weakly convex with easily computable proximal mapping. In particular, the authors of [47] proposed an augmented Lagrangian method for solving the Mereau envelope

smoothed problem of (15).We apply the same idea of Moreau envelope smoothing and design our RADMM algorithm. We define the augmented Lagrangian function of (14) as:

$$\mathcal{L}_{\rho,\gamma}(x,z;\lambda) = f(x) + g_\gamma(z) + \langle \lambda, Ax - z\rangle + \frac{\rho}{2}\|Ax - z\|^2. \tag{16}$$

A direct application of ADMM for solving (14) yields the following updating scheme:

$$\begin{aligned} x^{k+1} &:= \operatorname{argmin}_{x\in\mathcal{M}}\ \mathcal{L}_{\rho,\gamma}(x, z^k; \lambda^k) \\ z^{k+1} &:= \operatorname{argmin}_{z}\ \mathcal{L}_{\rho,\gamma}(x^{k+1}, z; \lambda^k) \\ \lambda^{k+1} &:= \lambda^k + \rho(Ax^{k+1} - z^{k+1}). \end{aligned} \tag{17}$$

Now since the x-subproblem in (17) is usually not easy to solve, we propose to replace it with a Riemannian gradient step, and this leads to our RADMM, which iterates as follows:

$$\begin{aligned} x^{k+1} &:= \operatorname{Retr}_{x^k}(-\eta_k \operatorname{grad}_x \mathcal{L}_{\rho,\gamma}(x^k, z^k; \lambda^k)) \\ z^{k+1} &:= \operatorname{argmin}_{z}\ \mathcal{L}_{\rho,\gamma}(x^{k+1}, z; \lambda^k) \\ \lambda^{k+1} &:= \lambda^k + \rho(Ax^{k+1} - z^{k+1}), \end{aligned} \tag{18}$$

where $\eta_k > 0$ is a stepsize, and $\operatorname{grad}_x \mathcal{L}_{\rho,\gamma}$ denotes the Riemannian gradient of $\mathcal{L}_{\rho,\gamma}$ with respect to x. The remaining thing is to discuss how to solve the z-subproblem in (18). It turns out that it is closely related to the proximal mapping of function g, and can be easily solved as long as the proximal mapping of g can be easily evaluated, as shown in the following lemma.

Lemma 1. *The solution of the z-subproblem in* (18) *is given by*

$$z^{k+1} := \frac{\gamma}{1+\gamma\rho}\left(\frac{1}{\gamma}y^{k+1} + \lambda^k + \rho A x^{k+1}\right), \tag{19}$$

where

$$y^{k+1} := \operatorname{prox}_{\frac{1+\rho\gamma}{\rho}g}\left(Ax^{k+1} + \frac{1}{\rho}\lambda^k\right), \tag{20}$$

where prox_h *denotes the proximal mapping of function h, which is defined as*

$$\operatorname{prox}_h(u) = \operatorname{argmin}_v\ h(v) + \frac{1}{2}\|u - v\|_2^2.$$

Proof. The z-subproblem in (18) can be equivalently rewritten as

$$(z^{k+1}, y^{k+1}) := \operatorname{argmin}_{z,y}\ g(y) + \frac{1}{2\gamma}\|y - z\|^2 + \langle \lambda^k, Ax^{k+1} - z\rangle + \frac{\rho}{2}\|Ax^{k+1} - z\|^2. \tag{21}$$

The optimality conditions of (21) are given by

$$0 = \frac{1}{\gamma}(z^{k+1} - y^{k+1}) - \lambda^k + \rho(z^{k+1} - Ax^{k+1}), \tag{22a}$$

$$0 \in \partial g(y^{k+1}) + \frac{1}{\gamma}(y^{k+1} - z^{k+1}). \tag{22b}$$

It is easy to see that (22a) immediately yields (19). Plugging (19) into (22b) gives

$$0 \in \frac{1+\gamma\rho}{\rho}\partial g(y^{k+1}) + y^{k+1} - \left(Ax^{k+1} + \frac{\lambda^k}{\rho}\right),$$

which implies

$$y^{k+1} = \text{argmin}_y \ \frac{1+\gamma\rho}{\rho} g(y) + \frac{1}{2}\left\| y - \left(Ax^{k+1} + \frac{\lambda^k}{\rho} \right) \right\|_2^2,$$

i.e., (20) holds. □

Our RADMM for solving (2) can therefore be summarized as in Algorithm 1. We can see that all the steps can be easily computed and implemented.

Algorithm 1: A Riemannian ADMM

Given $(x^0, z^0; \lambda^0)$, stepsize $\eta_k > 0$, parameters $\rho > 0$ and $\gamma > 0$;
for $k = 0, 1, ...$ **do**
- Update $x^{k+1} := \text{Retr}_{x^k}(-\eta_k \text{grad}_x \mathcal{L}_{\rho,\gamma}(x^k, z^k; \lambda^k))$;
- Update $y^{k+1} := \text{prox}_{\frac{1+\rho\gamma}{\rho} g}\left(Ax^{k+1} + \frac{1}{\rho}\lambda^k \right)$;
- Update $z^{k+1} := \frac{\gamma}{1+\gamma\rho}\left(\frac{1}{\gamma} y^{k+1} + \lambda^k + \rho A x^{k+1} \right)$;
- Update $\lambda^{k+1} := \lambda^k + \rho(Ax^{k+1} - z^{k+1})$.

3 Convergence Analysis

In this section, we analyze the iteration complexity of Algorithm 1 for obtaining an ϵ-stationary point of (2). The following assumption is needed in the analysis.

Assumption 1. *We assume f, g and $\mathcal{M}$ in (2) satisfy the following conditions.*

1. *$\mathcal{M} \subset \mathbb{R}^n$ is a compact and complete Riemannian manifold embedded in Euclidean space $\mathbb{R}^n$ with diameter D;*
2. *∇f is Lipschitz continuous with Lipschitz constant L in the ambient space $\mathbb{R}^n$;*
3. *g is convex and Lipschitz continuous with Lipschitz constant L_g in the ambient space $\mathbb{R}^m$.*

Also since $\mathcal{M}$ is compact and ∇f is continuous, we can denote the maximum of the norm of f as a constant M, i.e.,

$$\|\nabla f(x)\| \le M, \ \forall x \in \mathcal{M}. \tag{23}$$

Now we proceed to study the optimality of the problem (2). First, we note that the first-order optimality conditions of (2) are given by (see, e.g., [46]):

$$\begin{aligned} 0 &= \text{grad}_x \mathcal{L}(x^*, y^*, \lambda^*) = \text{Proj}_{\text{T}_{x^*}\mathcal{M}}\left(\nabla f(x^*) + A^\top \lambda^* \right), \\ 0 &\in \partial_y \mathcal{L}(x^*, y^*, \lambda^*) = \partial g(y^*) - \lambda^*, \\ 0 &= Ax^* - y^*, \\ x^* &\in \mathcal{M}, \end{aligned} \tag{24}$$

where the Lagrangian function of (2) is defined as

$$\mathcal{L}(x, y, \lambda) := f(x) + g(y) + \langle \lambda, Ax - y \rangle.$$

Based on this, we can define the ϵ-stationary point of (2) as follows.

Definition 4. *For (x, y, λ) with $x \in \mathcal{M}$, denote*

$$\partial \mathcal{L}(x, y, \lambda) := \begin{bmatrix} \mathrm{Proj}_{\mathrm{T}_x \mathcal{M}} \left(\nabla f(x) + A^\top \lambda \right) \\ \partial g(y) - \lambda \\ Ax - y \end{bmatrix}.$$

Then $(\bar{x}, \bar{y}, \bar{\lambda})$ with $\bar{x} \in \mathcal{M}$ is called an ϵ-stationary point of (2) if there exists $G \in \partial \mathcal{L}(\bar{x}, \bar{y}, \bar{\lambda})$ such that $\|G\|_2 \leq \epsilon$.

Before we present our main convergence results, we need the following lemmas. The first one is a brief recap of the properties of Moreau envelope (see e.g. [2] Chapter 6).

Lemma 2 (Properties of Moreau envelope)**.** *Suppose g is a L_g Lipschitz continuous and convex function. The Moreau envelope $g_\gamma(z) := \min_y g(y) + \frac{1}{2\gamma}\|z - y\|^2$ satisfies the following:*

1. $0 \leq g(z) - g_\gamma(z) \leq \gamma L_g^2$;
2. $\nabla g_\gamma(z) = \frac{1}{\gamma}(z - \mathrm{prox}_{\gamma g}(z))$;
3. $g_\gamma(z)$ *is L_g Lipschitz continuous;*
4. $g_\gamma(z)$ *is $1/\gamma$ Lipschitz smooth, i.e. $\nabla g_\gamma(z)$ is Lipschitz continuous with parameter $1/\gamma$.*

Now we proceed to bound the difference of dual sequence by the primal sequence.

Lemma 3 (Bound dual by primal)**.** *For the updates of Algorithm 1, we have:*

$$\|\lambda^{k+1} - \lambda^k\| \leq \frac{1}{\gamma}\|z^{k+1} - z^k\|. \tag{25}$$

Proof. Note that the optimality conditions of the z-subproblem in (18) is given by:

$$\nabla g_\gamma(z^{k+1}) - \lambda^k + \rho(z^{k+1} - Ax^{k+1}) = 0, \tag{26}$$

which, together with the λ update in (18), yields

$$\nabla g_\gamma(z^{k+1}) = \lambda^{k+1}. \tag{27}$$

The desired result (25) follows from Lemma 2. □

We now provide the smoothness notion over manifolds, which is also known as Lipschitz-type gradient for pullbacks.

Definition 5 ([7])**.** *Function f is called L_1-geodesic smooth on complete Riemannian manifold $\mathcal{M}$ if $\forall x \in \mathcal{M}$ and $\forall v \in \mathrm{T}_x\mathcal{M}$, it holds that*

$$f(\mathrm{Retr}_x(v)) \leq f(x) + \langle \mathrm{grad} f(x), v \rangle + \frac{L_1}{2}\|v\|^2. \tag{28}$$

The following lemma is from [7], which bridges the smoothness on the manifold with the smoothness in the ambient Euclidean space.

Lemma 4 ([7]). *Suppose $\mathcal{M} \in \mathbb{E}$ is a compact and complete Riemannian manifold embedded in Euclidean space $\mathbb{E}$ and f is L-Lipschitz smooth in $\mathbb{E}$, then f is also L_1-geodesic smooth, where L_1 is determined by the manifold $\mathcal{M}$ and f. Specifically, it can be shown (see [7]) that there exist positive constants α and β so that $\forall x \in \mathcal{M}$ and $\forall \eta \in \mathrm{T}_x\mathcal{M}$,*

$$\|\mathrm{Retr}_x(\eta) - x\| \leq \alpha\|\eta\|, \text{ and } \|\mathrm{Retr}_x(\eta) - x - \eta\| \leq \beta\|\eta\|^2.$$

As a result, it can be shown that

$$L_1 = \frac{L}{2}\alpha^2 + M\beta,$$

where M is the upper bound of the gradient, which is defined in (23).

Now we are ready to present the smoothness of the augmented Lagrangian function (16).

Lemma 5. *For any $\{(z^k, \lambda^k)\}$ generated in Algorithm 1, the augmented Lagrangian function $\mathcal{L}_{\rho,\gamma}(x, z^k, \lambda^k)$ defined in* (16) *is L_ρ-geodesic smooth with respect to $x \in \mathcal{M}$, where*

$$L_\rho = \frac{L + \rho\|A^\top A\|_2}{2}\alpha^2 + (M + \|A\|_2 L_g + \rho\|A^\top A\|_2 D + \|A\|_2(2L_g + \rho\|A\|_2 D))\beta, \tag{29}$$

and $\|B\|_2$ denotes the spectral norm of matrix B.

Proof. We first show that $\{z^k\}$, $\{\lambda^k\}$, $k = 0, 1, \ldots$, generated in Algorithm 1 are uniformly bounded. Note that from (27), we have

$$\|\lambda^k\| = \|\nabla g_\gamma(z^k)\| \leq L_g, \tag{30}$$

where the inequality follows from the facts that g is L_g-Lipschitz continuous (Assumption 1) and Lemma 2.

From the update of λ^{k+1}, i.e., $\lambda^{k+1} := \lambda^k + \rho(Ax^{k+1} - z^{k+1})$, we have

$$z^{k+1} = (\lambda^k - \lambda^{k+1})/\rho + Ax^{k+1},$$

which, together with (30) and Assumption 1, immediately implies

$$\|z^{k+1}\| \leq \frac{2L_g}{\rho} + \|A\|_2 D. \tag{31}$$

We now show that the gradient of $\mathcal{L}_{\rho,\gamma}(x, z^k, \lambda^k)$, i.e., $\nabla_x\mathcal{L}_{\rho,\gamma}(x, z^k, \lambda^k) = \nabla f(x) + A^\top\lambda^k + \rho A^\top(Ax - z^k)$, is uniformly upper bounded $\forall x \in \mathcal{M}$. To this end, we note that

$$\begin{aligned}\|\nabla_x\mathcal{L}_{\rho,\gamma}(x, z^k, \lambda^k)\| \leq& \|\nabla f(x)\| + \|A^\top\lambda^k\| + \rho\|A^\top(Ax - z^k)\| \\ \leq& \|\nabla f(x)\| + \|A\|_2\|\lambda^k\| + \rho\|A^\top A\|_2\|x\| + \rho\|A\|_2\|z^k\| \\ \leq& M + \|A\|_2 L_g + \rho\|A^\top A\|_2 D + \|A\|_2(2L_g + \rho\|A\|_2 D),\end{aligned} \tag{32}$$

where the last inequality is due to (30), (31) and Assumption 1. Moreover, we have

$$\begin{aligned}\|\nabla_x\mathcal{L}_{\rho,\gamma}(x_1, z^k, \lambda^k) - \nabla_x\mathcal{L}_{\rho,\gamma}(x_2, z^k, \lambda^k)\| \leq& \|\nabla f(x_1) - \nabla f(x_2)\| + \rho\|A^\top A(x_1 - x_2)\| \\ \leq& L\|x_1 - x_2\| + \rho\|A^\top A\|_2\|x_1 - x_2\|.\end{aligned} \tag{33}$$

By applying Lemma 4 together with (32) and (33), we immediately obtain the desired result. □

Now we give the following lemma regarding the decrease of the augmented Lagrangian function $\mathcal{L}_{\rho,\gamma}$.

Lemma 6. *For the iterates $\{(x^k, z^k, \lambda^k)\}$ generated in Algorithm 1, we have:*

$$\begin{aligned}&\mathcal{L}_{\rho,\gamma}(x^{k+1}, z^{k+1}, \lambda^{k+1}) - \mathcal{L}_{\rho,\gamma}(x^k, z^k, \lambda^k)\\ &\leq \left(\frac{1}{\rho\gamma^2} - \frac{\rho}{2}\right)\|z^{k+1} - z^k\|^2 - \left(\frac{1}{\eta_k} - \frac{L_\rho}{2}\right)\|\mathrm{Retr}_{x^k}^{-1}(x^{k+1})\|^2,\end{aligned} \tag{34}$$

where L_ρ is defined in (29).

Proof. First, we have

$$\begin{aligned}&\mathcal{L}_{\rho,\gamma}(x^{k+1}, z^{k+1}, \lambda^{k+1}) - \mathcal{L}_{\rho,\gamma}(x^{k+1}, z^{k+1}, \lambda^k)\\ &= \langle \lambda^{k+1} - \lambda^k, Ax^{k+1} - z^{k+1}\rangle\\ &= \frac{1}{\rho}\|\lambda^{k+1} - \lambda^k\|^2 \leq \frac{1}{\rho\gamma^2}\|z^{k+1} - z^k\|^2,\end{aligned} \tag{35}$$

where the inequality is from Lemma 3.

Second, we have,

$$\begin{aligned}&\mathcal{L}_{\rho,\gamma}(x^{k+1}, z^{k+1}, \lambda^k) - \mathcal{L}_{\rho,\gamma}(x^{k+1}, z^k, \lambda^k)\\ =&g_\gamma(z^{k+1}) - g_\gamma(z^k) + \langle \lambda^k, z^k - z^{k+1}\rangle + \frac{\rho}{2}(\|Ax^{k+1} - z^{k+1}\|^2 - \|Ax^{k+1} - z^k\|^2)\\ =&g_\gamma(z^{k+1}) - g_\gamma(z^k) + \langle \lambda^k + \rho(Ax^{k+1} - z^{k+1}), z^k - z^{k+1}\rangle - \frac{\rho}{2}\|z^{k+1} - z^k\|^2\\ \leq& -\frac{\rho}{2}\|z^{k+1} - z^k\|^2,\end{aligned} \tag{36}$$

where the inequality is by convexity of g_γ and $\nabla g_\gamma(z^{k+1}) = \lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} - z^{k+1})$.

Third, by Lemma 5 and (28), we obtain

$$\begin{aligned}&\mathcal{L}_{\rho,\gamma}(x^{k+1}, z^k, \lambda^k) - \mathcal{L}_{\rho,\gamma}(x^k, z^k, \lambda^k)\\ &\leq \langle \mathrm{grad}_x \mathcal{L}_{\rho,\gamma}(x^k, z^k, \lambda^k), \mathrm{Retr}_{x^k}^{-1}(x^{k+1})\rangle + \frac{L_\rho}{2}\|\mathrm{Retr}_{x^k}^{-1}(x^{k+1})\|^2\\ &= -\left(\frac{1}{\eta_k} - \frac{L_\rho}{2}\right)\|\mathrm{Retr}_{x^k}^{-1}(x^{k+1})\|^2,\end{aligned} \tag{37}$$

where the equality follows from the x-update in Algorithm 1.

Combining (35), (36), and (37) yields the desired result (34). □

The following lemma shows that the augmented Lagrangian function $\mathcal{L}_{\rho,\gamma}$ is lower bounded.

Lemma 7. *If $\rho\gamma \geq 1$, then the sequence $\{\mathcal{L}_{\rho,\gamma}(x^k, z^k, \lambda^k)\}$ is uniformly lower bounded by $F^* - \gamma L_g^2$, where F^* is the optimal value of* (1).

Proof. By the $1/\gamma$ Lipschitz smoothness of g_γ (see Lemma 2) and $\nabla g_\gamma(z^k) = \lambda^k$, we get

$$g_\gamma(Ax) \leq g_\gamma(z) + \langle \nabla g_\gamma(z), Ax - z\rangle + \frac{1}{2\gamma}\|Ax - z\|^2,$$

which implies

$$\begin{aligned}\mathcal{L}_{\rho,\gamma}(x^k, z^k, \lambda^k) &= f(x^k) + g_\gamma(z^k) + \langle \lambda^k, Ax^k - z^k\rangle + \frac{\rho}{2}\|Ax^k - z^k\|^2\\ &\geq f(x^k) + g_\gamma(Ax^k) + \left(\frac{\rho}{2} - \frac{1}{2\gamma}\right)\|Ax^k - z^k\|^2\\ &\geq f(x^k) + g_\gamma(Ax^k)\\ &\geq f(x^k) + g(Ax^k) - \gamma L_g^2\\ &\geq F^* - \gamma L_g^2,\end{aligned}$$

where the third inequality follows from Lemma 2. □

The following lemma gives an upper bound for $G^k \in \partial\mathcal{L}(x^k, y^k, \lambda^k)$.

Lemma 8. *There exists $G^k \in \partial\mathcal{L}(x^k, y^k, \lambda^k)$, $\forall k \geq 1$, as defined in Definition 4, such that:*

$$\|G^k\|^2 \leq \frac{2}{\eta_k^2}\|\mathrm{Retr}_{x^k}^{-1}(x^{k+1})\|^2 + \frac{2(\rho^2\|A\|_2^2+1)}{\rho^2\gamma^2}\|z^k - z^{k-1}\|^2 + 2\gamma^2 L_g^2.$$

Proof. From (22a), (22b) and the update of λ^{k+1} in Algorithm 1, we know that $\lambda^k \in \partial g(y^k)$ for $k = 1, 2, \ldots$. Therefore, there exist $G^k \in \partial\mathcal{L}(x^k, y^k, \lambda^k)$ such that

$$\begin{aligned}\|G^k\|^2 =& \|\,\mathrm{Proj}_{\mathrm{T}_{x^k}\mathcal{M}}\left(\nabla f(x^k) + A^\top\lambda^k\right)\|^2 + \|Ax^k - y^k\|^2\\ \leq& \|\,\mathrm{Proj}_{\mathrm{T}_{x^k}\mathcal{M}}\left(\nabla f(x^k) + A^\top\lambda^k\right)\|^2 + 2\|Ax^k - z^k\|^2 + 2\|z^k - y^k\|^2.\end{aligned}$$

Now from the x update of Algorithm 1, we know that

$$\mathrm{Proj}_{\mathrm{T}_{x^k}\mathcal{M}}\left(\nabla f(x^k) + A^\top\lambda^k\right) = -\frac{1}{\eta_k}\mathrm{Retr}_{x^k}^{-1}(x^{k+1}) - \mathrm{Proj}_{\mathrm{T}_{x^k}\mathcal{M}}(\rho A^\top(Ax^k - z^k)).$$

Therefore, we have

$$\begin{aligned}\|G^k\|^2 \leq& \left\|\frac{1}{\eta_k}\mathrm{Retr}_{x^k}^{-1}(x^{k+1}) + \mathrm{Proj}_{\mathrm{T}_{x^k}\mathcal{M}}(\rho A^\top(Ax^k - z^k))\right\|^2 + 2\|Ax^k - z^k\|^2 + 2\|z^k - y^k\|^2\\ \leq& \frac{2}{\eta_k^2}\|\mathrm{Retr}_{x^k}^{-1}(x^{k+1})\|^2 + 2\rho^2\|\,\mathrm{Proj}_{\mathrm{T}_{x^k}\mathcal{M}}(A^\top(Ax^k - z^k))\|^2 + 2\|Ax^k - z^k\|^2 + 2\|z^k - y^k\|^2\\ \leq& \frac{2}{\eta_k^2}\|\mathrm{Retr}_{x^k}^{-1}(x^{k+1})\|^2 + 2\rho^2\|A\|_2^2\|Ax^k - z^k\|^2 + 2\|Ax^k - z^k\|^2 + 2\|z^k - y^k\|^2\\ =& \frac{2}{\eta_k^2}\|\mathrm{Retr}_{x^k}^{-1}(x^{k+1})\|^2 + 2(\rho^2\|A\|_2^2+1)\|Ax^k - z^k\|^2 + 2\|z^k - y^k\|^2.\end{aligned}$$

Now by the update of λ^k in Algorithm 1 and (25) we have $\rho\|Ax^k - z^k\| = \|\lambda^k - \lambda^{k-1}\| \leq \frac{1}{\gamma}\|z^k - z^{k-1}\|$. By (22b) we have $z^k - y^k \in \gamma\partial g(y^k)$ so that $\|z^k - y^k\| \leq \gamma L_g$. Combining these results we get

$$\begin{aligned}\|G^k\|^2 \leq& \frac{2}{\eta_k^2}\|\mathrm{Retr}_{x^k}^{-1}(x^{k+1})\|^2 + 2(\rho^2\|A\|_2^2+1)\|Ax^k - z^k\|^2 + 2\|z^k - y^k\|^2\\ \leq& \frac{2}{\eta_k^2}\|\mathrm{Retr}_{x^k}^{-1}(x^{k+1})\|^2 + \frac{2(\rho^2\|A\|_2^2+1)}{\rho^2\gamma^2}\|z^k - z^{k-1}\|^2 + 2\gamma^2 L_g^2,\end{aligned}$$

which gives the desired result. □

Finally, we have the following convergence result for Algorithm 1.

Theorem 1. *Denote the iterates of Algorithm 1 by $\{(x^k, z^k, \lambda^k)\}$. For a given tolerance $\epsilon > 0$, we set $\rho = 1/\epsilon$, $\gamma = \sqrt{\frac{2}{\rho^2} + \frac{\rho^2\|A\|_2^2+1}{\rho^3 L_\rho}} = \mathcal{O}(\epsilon)$, also $\eta_k = \eta = \frac{1}{L_\rho}$. Note that our choices of ρ and γ guarantees that $\rho\gamma > 1$. Then there exist $G^k \in \partial\mathcal{L}(x^k, z^k, \lambda^k)$, $k = 1, 2, \ldots$, such that*

$$\min_{k=1,\ldots,K}\|G^k\|^2 \leq \epsilon^2,$$

provided that

$$K = \mathcal{O}\left(\frac{1}{\epsilon^4}\right).$$

That is, Algorithm 1 generates an ϵ-stationary point to Problem (2) in $\mathcal{O}(\epsilon^{-4})$ iterations.

Proof. From Lemma 8, we get

$$\|G^k\|^2 \leq \frac{2}{\eta_k^2}\|\mathrm{Retr}_{x^k}^{-1}(x^{k+1})\| + \frac{2(\rho^2\|A\|_2^2+1)}{\rho^2\gamma^2}\|z^k - z^{k-1}\|^2 + 2\gamma^2 L_g^2,$$

which, combining with (34) and $\eta_k = 1/L_\rho$, yields

$$\begin{aligned}\|G^k\|^2 \ &\leq \frac{4}{\eta_k}\left(\mathcal{L}_{\rho,\gamma}(x^k, z^k, \lambda^k) - \mathcal{L}_{\rho,\gamma}(x^{k+1}, z^{k+1}, \lambda^{k+1})\right) \\ &+ \left(\frac{2(\rho^2\|A\|_2^2+1)}{\rho^2\gamma^2}\|z^k - z^{k-1}\|^2 - \frac{4}{\eta_k}\left(\frac{\rho}{2} - \frac{1}{\rho\gamma^2}\right)\|z^k - z^{k+1}\|^2\right) + 2\gamma^2 L_g^2.\end{aligned}$$

Now by taking γ, ρ and $\eta_k = \eta$ as described in the theorem, it is easy to verify that

$$\frac{2(\rho^2\|A\|_2^2+1)}{\rho^2\gamma^2} \leq \frac{4}{\eta_k}\left(\frac{\rho}{2} - \frac{1}{\rho\gamma^2}\right).$$

Therefore, we have

$$\begin{aligned}\|G^k\|^2 &\leq \frac{4}{\eta_k}\left(\mathcal{L}_{\rho,\gamma}(x^k, z^k, \lambda^k) - \mathcal{L}_{\rho,\gamma}(x^{k+1}, z^{k+1}, \lambda^{k+1})\right) \\ &+ \left(\frac{4}{\eta_k}\left(\frac{\rho}{2} - \frac{1}{\rho\gamma^2}\right)\|z^k - z^{k-1}\|^2 - \frac{4}{\eta_k}\left(\frac{\rho}{2} - \frac{1}{\rho\gamma^2}\right)\|z^k - z^{k+1}\|^2\right) + 2\gamma^2 L_g^2.\end{aligned}$$

Now by summing this inequality over $k = 1, \ldots, K$ and using Lemma 7, we get

$$\frac{1}{K}\sum_{k=1}^{K}\|G^k\|^2 \leq \frac{4}{\eta K}(\mathcal{L}_{\rho,\gamma}(x^1, z^1, \lambda^1) - F^* + \gamma L_g^2) + \frac{2\rho}{\eta K}\|z^1 - z^0\|^2 + 2\gamma^2 L_g^2.$$

Since we take $\gamma = \mathcal{O}(\epsilon)$, $\rho = \frac{1}{\epsilon}$ and $\eta = \frac{1}{L_\rho} = \mathcal{O}(\epsilon)$, to ensure $\min_{k=1,\ldots,K}\|G^k\|^2 \leq \epsilon^2$, we need $K = \mathcal{O}(\frac{1}{\epsilon^4})$. □

4 Applications and Numerical Experiments

Problem (1) finds many applications in machine learning, statistics and signal processing. For example, K-means clustering [8], sparse spectral clustering [30, 32], and orthogonal dictionary learning [36, 13, 34, 37, 38] are all of the form of (1). In this section, we present two representative applications of (1) and then report the numerical results of our Algorithm 1 for solving them.

Example 1. Sparse Principal Component Analysis (PCA). Principal Component Analysis, proposed by Pearson [33] and later developed by Hotelling [20], is one of the most fundamental statistical tools in analyzing high-dimensional data. Sparse PCA seeks principal components with very few nonzero components. For given data matrix $A \in \mathbb{R}^{m\times n}$, the sparse PCA that seeks the leading p ($p < \min\{m, n\}$) sparse loading vectors can be formulated as

$$\begin{aligned}&\min_X F(X) := -\frac{1}{2}\mathrm{Tr}(X^\top A^\top A X) + \mu\|X\|_1 \\ &\text{s.t. } X \in \mathrm{St}(n, p),\end{aligned} \tag{38}$$

where $\mathrm{Tr}(Y)$ denotes the trace of matrix Y, the ℓ_1 norm is defined as $\|X\|_1 = \sum_{ij} |X_{ij}|$, $\mu > 0$ is a weighting parameter. This is the original formulation of sparse PCA as proposed by Jolliffe et al. in [23], where the model is called SCoTLASS and imposes sparsity and orthogonality to the loading vectors simultaneously. When $\mu = 0$, (38) reduces to computing the leading p eigenvalues and the corresponding eigenvectors of $A^\top A$. When $\mu > 0$, the ℓ_1 norm $\|X\|_1$ can promote sparsity of the loading vectors. There are many numerical algorithms for solving (38) when $p = 1$. In this case, (38) is relatively easy to solve because X reduces to a vector and the constraint set reduces to a sphere. However, there has been very limited literature for the case $p > 1$. Existing works, including [50, 12, 35, 24, 31], do not impose orthogonal loading directions. As discussed in [24], "Simultaneously enforcing sparsity and orthogonality seems to be a hard (and perhaps questionable) task." We refer the interested reader to [51] for more details on existing algorithms for solving sparse PCA.

Example 2. Orthogonal Dictionary Learning (ODL) and Dual principal component pursuit (DPCP). In ODL, one is given a set of p ($p \gg n$) data points $\mathbf{y}_1, \ldots, \mathbf{y}_p \in \mathbb{R}^n$ and aims to find an orthonormal basis of $\mathbb{R}^n$ to represent them compactly. In other words, by letting $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_p] \in \mathbb{R}^{n\times p}$, we want to find an orthogonal matrix $X \in \mathbb{R}^{n\times n}$ and a sparse matrix $A \in \mathbb{R}^{n\times p}$ such that $Y = XA$. Since X is orthogonal, we know that $A = X^\top Y$. This naturally leads to the following matrix version of ODL [36, 13, 34, 37, 38]:

$$\begin{aligned} &\min_X \ \|Y^\top X\|_1 \\ &\text{s.t. } X \in \mathrm{St}(n, n). \end{aligned} \tag{39}$$

Here, the ℓ_1 norm is used to promote the sparsity of $A = X^\top Y$, and the constraint set $\mathrm{St}(n, n)$ is known as the orthogonal group, which is a special case of the Stiefel manifold.

Another representative application of (39) is the so-called dual principal component pursuit (DPCP) for robust subspace recovery (RSR). RSR aims to fit a linear subspace to a dataset corrupted by outliers, which is a fundamental problem in machine learning and data mining. RSR can be described as follows. Given a dataset $Y = [\mathcal{X}, \mathcal{O}]\Gamma \in \mathbb{R}^{n\times(p_1+p_2)}$, where $\mathcal{X} \in \mathbb{R}^{n\times p_1}$ are inlier points spanning a d-dimensional subspace $\mathcal{S}$ of $\mathbb{R}^n$ ($d < p_1$), $\mathcal{O} \in \mathbb{R}^{n\times p_2}$ are outlier points without linear structure, and $\Gamma \in \mathbb{R}^{(p_1+p_2)\times(p_1+p_2)}$ is an unknown permutation, the goal is to recover the inlier space $\mathcal{S}$, or equivalently, to cluster the points into inliers and outliers. For a more comprehensive review of RSR, see the recent survey paper by Lerman and Maunu [27]. DPCP is a recently proposed approach to RSR that seeks to learn recursively a basis for the orthogonal complement $\mathcal{S}$ by solving (39) when X reduces to a vector, i.e.,

$$\min_{\mathbf{x}\in\mathbb{R}^n} \ f(\mathbf{x}) := \|Y^\top \mathbf{x}\|_1 \quad \text{s.t.} \quad \|\mathbf{x}\|_2 = 1, \tag{40}$$

The idea of DPCP is to first compute a normal vector $\mathbf{x}$ to a hyperplane $\mathcal{H}$ that contains all inliers $\mathcal{X}$. As outliers are not orthogonal to $\mathbf{x}$ and the number of outliers is known to be small, the normal vector $\mathbf{x}$ can be found by solving (40). It is shown in [40, 49] that under certain conditions, solving (40) indeed yields a vector that is orthogonal to $\mathcal{S}$, given that the number of outliers p_2 is at most on the order of $O(p_1^2)$. If d is known, then one can recover $\mathcal{S}$ as the intersection of the $p := n - d$ orthogonal hyperplanes that contain $\mathcal{X}$, which amounts to solving the following matrix optimization problem:

$$\min_{X\in\mathbb{R}^{n\times(n-d)}} \|Y^\top X\|_1 \quad \text{s.t.} \quad X^\top X = I_{n-d}. \tag{41}$$

Note that (38)-(41) are all in the form of (1).

4.1 Numerical Experiments on Sparse PCA

In this subsection, we conduct experiments to test the performance of our Riemannian ADMM for solving sparse PCA (38), and compare it with the performance of ManPG [10] and Riemannian subgradient method [14, 29]. To apply Riemannian ADMM, we first rewrite (38) as:

$$\begin{aligned} \min_{X,Y} \ & -\frac{1}{2}\mathrm{Tr}(X^\top A^\top A X) + \mu\|Y\|_1 \\ \text{s.t. } & X = Y,\ X \in \mathrm{St}(n,p). \end{aligned} \tag{42}$$

Now we see that the nonsmooth function $\|\cdot\|_1$ and the manifold constraint are associated with different variables. Thus, the two difficult terms are separated. Using the Moreau envelope smoothing, the smoothed problem of (42) is given by:

$$\begin{aligned} \min_{X,Z} \ & -\frac{1}{2}\mathrm{Tr}(X^\top A^\top A X) + g_\gamma(Z) \\ \text{s.t. } & X = Z,\ X \in \mathrm{St}(n,p), \end{aligned} \tag{43}$$

where $g_\gamma(Z) := \min_Y\{\mu\|Y\|_1 + \frac{1}{2\gamma}\|Y - Z\|_F^2\}$. The augmented Lagrangian function of (43) is given by

$$\mathcal{L}_{\rho,\gamma}(X, Z; \Lambda) = -\frac{1}{2}\mathrm{Tr}(X^\top A^\top A X) + g_\gamma(Z) + \langle \Lambda, X - Z\rangle + \frac{\rho}{2}\|X - Z\|_F^2.$$

Therefore, one iteration of our Riemannian ADMM 1 for solving (42) reduces to:

$$\begin{aligned} X^{k+1} &:= \mathrm{Retr}_{X^k}(-\eta_k \,\mathrm{Proj}_{\mathrm{T}_{X^k}\mathrm{St}(n,p)}(-A^\top A X^k + \Lambda^k + \rho(X^k - Z^k))) \\ Y^{k+1} &:= \mathrm{prox}_{\frac{\mu(1+\rho\gamma)}{\rho}\|\cdot\|_1}\left(X^{k+1} + \frac{1}{\rho}\Lambda^k\right) \\ Z^{k+1} &:= \frac{\gamma}{1+\gamma\rho}\left(\frac{1}{\gamma}Y^{k+1} + \Lambda^k + \rho X^{k+1}\right) \\ \Lambda^{k+1} &:= \Lambda^k + \rho(X^{k+1} - Z^{k+1}). \end{aligned} \tag{44}$$

The ManPG [10] for solving (38) updates the iterates as follows:

$$\begin{aligned} V^k &:= \mathrm{argmin}_{V \in \mathrm{T}_{X^k}\mathrm{St}(n,p)}\langle -A^\top A X^k, V\rangle + \frac{1}{2t}\|V\|^2 + \mu\|A(X^k + V)\|_1 \\ X^{k+1} &:= \mathrm{Retr}_{X^k}(\alpha V^k), \end{aligned} \tag{45}$$

where α and t are stepsizes. The authors of [10] suggest to solve the V subproblem by using a semi-smooth Newton method. The Riemannian subgradient method (RSG) [14] for solving (38) updates the iterates as follows:

$$X^{k+1} := \mathrm{Retr}_{X^k}(-\eta_k \,\mathrm{Proj}_{\mathrm{T}_{X^k}\mathrm{St}(n,p)}(-A^\top A X^k + \mu D^k)), \quad \text{with } D^k \in \partial\|X^k\|_1. \tag{46}$$

We now describe the setup of our numerical experiment. The data matrix $A \in \mathbb{R}^{m\times n}$ is generated randomly whose entries follow the standard Gaussian distribution. We choose μ from $\{0.5, 0.7, 1\}$, n from $\{100, 300, 500\}$, and p from $\{50, 100\}$. In our Riemannian ADMM, we set $\gamma = 10^{-8}$, $\rho = 10^2$ and $\eta_k = \eta = 10^{-2}$. The code of ManPG is downloaded from the authors' website of [10] and default settings of the parameters are used. In RSG (46), we set the stepsize $\eta_k = \eta = 10^{-2}$ as a result of a

simple grid search. For all three algorithms, we terminate them when the change of the objective function in two consecutive iterations is smaller than 10^{-8}, which means

$$|F(X^{k+1}) - F(X^k)| < 10^{-8}$$

for ManPG (45) and RSG (46), and

$$|F(Y^{k+1}) - F(Y^k)| < 10^{-8}$$

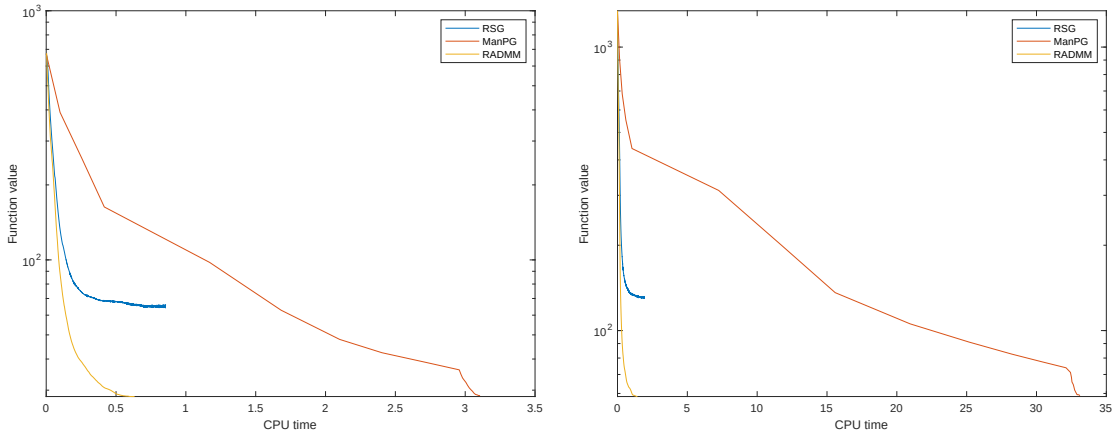
for our RADMM (44), where $F(X) := -\frac{1}{2}\mathrm{Tr}(X^\top A^\top AX) + \mu\|X\|_1$. Moreover, we also terminate the three algorithms when the maximal iteration number, which is set 1000, is reached. For different combinations of μ, n and p, we report the objective value "obj" ($F(X^k)$ for ManPG and RSG, and $F(Y^k)$ for RADMM), CPU time and the sparsity of the solution "Spa" in Table 1. Here the "sparsity" is the percentage of the zero entries of the iterate (X^k for ManPG and RSG, and Y^k for RADMM). Moreover, note that Y^k in RADMM (44) is not on the Stiefel manifold, we thus report the constraint violation "infeas", which is defined as $\|(Y^k)^\top Y_k - I_p\|_F$, in Table 1 for RADMM. From Table 1 we have the following observations: (i) both ManPG and RADMM generated very sparse solutions, while RSG cannot generate sparse solutions; (ii) RSG is very slow. It cannot decrease the objective value to the same level as ManPG and RADMM; (iii) RADMM is always faster than ManPG, sometimes is about 10 to 20 times faster. In most cases, RADMM yields iterates with much better objective value than ManPG; (iv) Though Y^k generated by RADMM is not on the Stiefel manifold, the constraint violation is small – usually in the order of 10^{-6}-10^{-8}.

Settings		RSG			ManPG			RADMM			
μ	(n,p)	obj	CPU	Spa	obj	CPU	Spa	obj	CPU	Spa	infeas
0.5	$(300,50)$	23.9783	0.5725	0	6.1015	1.6808	0.9964	6.0794	0.3550	0.9965	1.14e-6
	$(300,100)$	44.9207	1.4091	0	9.9683	16.9343	0.9966	9.4524	1.0113	0.9964	4.43e-6
	$(500,50)$	34.8607	1.1545	0	4.8868	1.7355	0.9977	4.7141	0.8379	0.9980	7.07e-8
	$(500,100)$	72.1180	2.2447	0	12.0830	15.4234	0.9980	11.7489	1.5738	0.9980	1.00e-7
0.7	$(300,50)$	50.0266	0.5584	0	14.9053	1.7990	0.9965	14.9497	0.2860	0.9967	9.90e-8
	$(300,100)$	99.1306	1.4196	0	29.0171	16.7438	0.9966	28.9101	0.8185	0.9967	1.40e-7
	$(500,50)$	73.4292	1.1515	0	14.3927	1.9293	0.9978	14.2181	0.7760	0.9980	9.90e-8
	$(500,100)$	147.0228	2.2224	0	29.8765	16.9296	0.9980	29.6908	1.2075	0.9980	1.40e-7
1.0	$(300,50)$	99.5018	0.5593	0	29.4374	2.2295	0.9967	29.6217	0.1879	0.9967	1.41e-7
	$(300,100)$	202.9473	1.4154	0	61.5334	16.0349	0.9965	61.0310	0.5699	0.9967	2.00e-7
	$(500,50)$	149.1125	1.1564	0	30.5119	1.8004	0.9980	30.4099	0.4336	0.9980	1.41e-7
	$(500,100)$	295.5895	2.2384	0	59.5210	18.3017	0.9980	59.5309	1.0377	0.9980	2.00e-7

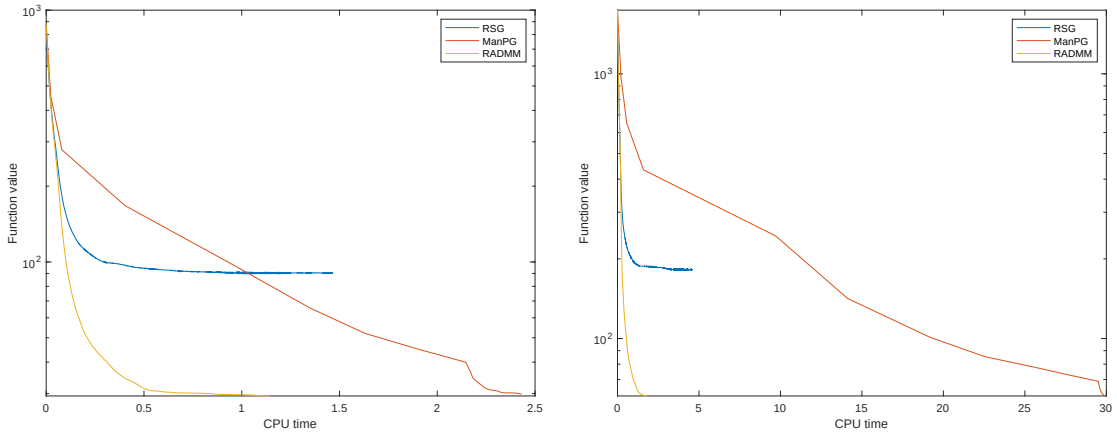
Table 1: Comparison of RSG (46), ManPG (45), and RADMM (44) for solving (38). The results are averaged for 10 repeated experiments with random initializations.

To better illustrate the behavior of the three algorithms, we further draw some figures in Figure 1, to show how the objective function value decreases along with the CPU time. From Figure 1 we can clearly see that RGS quickly stops decreasing the objective value, while both ManPG and RADMM can decrease the objective value to a much lower level. Moreover, RADMM is much faster than ManPG.

We also compare our RADMM (44) with SOC [26] and MADMM [25]. Before we present the numerical comparisons, we remind the reader that there is no convergence guarantee for SOC and MADMM. The SOC (4) algorithm for solving problem (38) actually solves the following equivalent



(a) $n = 300, p = 50$ (b) $n = 300, p = 100$



(c) $n = 500, p = 50$ (d) $n = 500, p = 100$

Figure 1: Comparison of the CPU time (in seconds) consumed among the ManPG, RADMM and Riemannian gradient methods for solving (38) with $\mu = 1$. Each figure is averaged for 10 repeated experiments with random initializations.

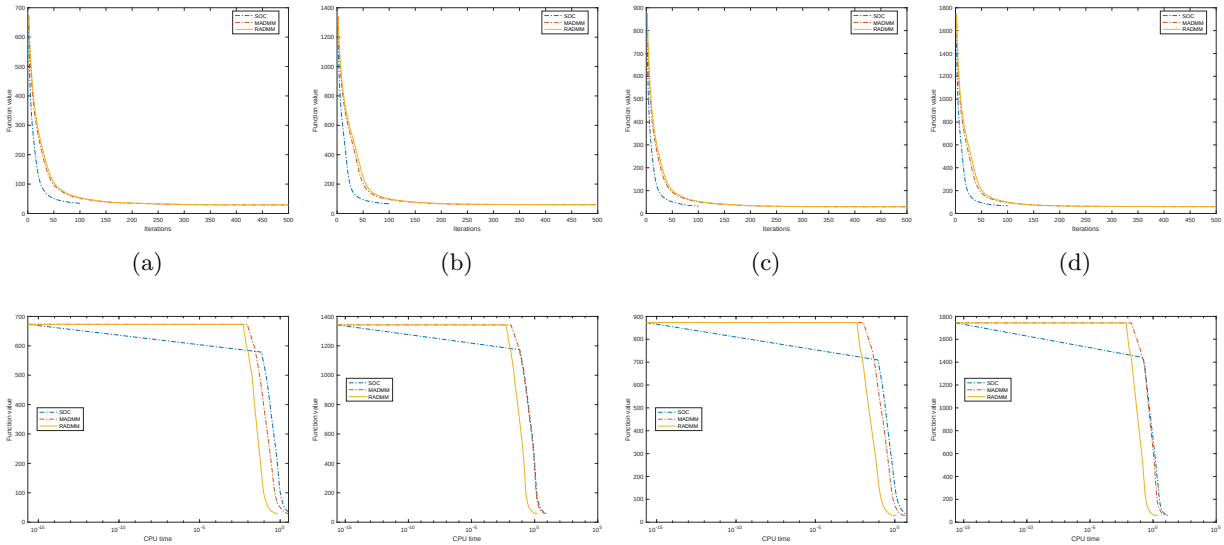
problem:

$$\begin{aligned} \min_{X,Y} \ & -\frac{1}{2}\mathrm{Tr}(X^\top A^\top A X) + \mu\|X\|_1 \\ \text{s.t. } & X = Y,\ Y \in \mathrm{St}(n,p). \end{aligned} \tag{47}$$

The SOC iterates as follows.

$$\begin{aligned} X^{k+1} :=\ & \mathrm{argmin}_X -\frac{1}{2}\mathrm{Tr}(X^\top A^\top A X) + \mu\|X\|_1 + \langle \Lambda^k, X - Y^k\rangle + \frac{\rho}{2}\|X - Y^k\|_F^2 \\ Y^{k+1} :=\ & \mathrm{argmin}_{Y\in\mathrm{St}(n,p)} \langle \Lambda^k, X^{k+1} - Y\rangle + \frac{\rho}{2}\|X^{k+1} - Y\|_F^2 \\ \Lambda^{k+1} :=\ & \Lambda^k + \rho(X^{k+1} - Y^{k+1}). \end{aligned} \tag{48}$$

In our numerical experiment, we chose to solve the X-subproblem using the proximal gradient



(a) (b) (c) (d)

(a) $n = 300, p = 50$ (b) $n = 300, p = 100$ (c) $n = 500, p = 50$ (d) $n = 500, p = 100$

Figure 2: Comparison of SOC, MADMM and RADMM for solving (38) with $\mu = 1$. The first row is the comparison of function value decrease w.r.t. number of iterations, and the second row is w.r.t. CPU time consumed. Each figure is averaged for 10 repeated experiments with random initializations.

method. The MADMM (5) solves (42), and iterates as follows:

$$\begin{aligned} X^{k+1} :=\ & \operatorname{argmin}_{X \in \mathrm{St}(n,p)} -\frac{1}{2}\mathrm{Tr}(X^\top A^\top A X) + \langle \Lambda^k, X - Y^k \rangle + \frac{\rho}{2}\|X - Y^k\|_F^2 \\ Y^{k+1} :=\ & \operatorname{argmin}_Y \mu\|Y\|_1 + \langle \Lambda^k, X^{k+1} - Y \rangle + \frac{\rho}{2}\|X^{k+1} - Y\|_F^2 \\ \Lambda^{k+1} :=\ & \Lambda^k + \rho(X^{k+1} - Y^{k+1}). \end{aligned} \tag{49}$$

In our numerical experiment, we chose to solve the X-subproblem using a Riemannian gradient method.

We test our RADMM with SOC and MADMM with the following parameters: for SOC we set $\rho = 50$ and $\eta = 10^{-2}$, where η is the stepsize for the proximal gradient method for solving the X-subproblem; for MADMM we set $\rho = 100$ and $\eta = 10^{-2}$, where η is the stepsize for the Riemannian gradient method for solving the X-subproblem; for RADMM we set $\rho = 100$, $\eta = 10^{-2}$ and $\gamma = 10^{-8}$. The parameters are obtained via simple grid searches, also we randomly initialize three algorithms at the same starting point. For all the three algorithms we record the function value and sparsity for the sequence on the manifold, i.e. X^k for MADMM and RADMM, and Y^k for SOC. For each algorithm, we terminate after 100 iterations. We present the function value change curve in Figure 2. We also report the objective function values of the outputs (denoted as "obj"), the sparsity (the percentage of zero entries, denoted as "Spa") and the constraint violation ($\|X^k - Y^k\|_F$ for all three algorithms, denoted as "infeas") in Table 2. From the top row of Figure 2 we can see that SOC is more efficient in terms of the iteration number, but from the bottom row of Figure 2 we see that RADMM is more efficient in terms of the CPU time. This is exactly because all steps in our RADMM are very easy to compute, and so the per-iteration complexity is very cheap.

Settings	SOC			MADMM			RADMM		
(n,p)	obj	Spa	infeas	obj	Spa	infeas	obj	Spa	infeas
$(300,50)$	34.8851	0.7609	0.0060	29.2059	0.9967	0.0000	29.1197	0.9967	0.0000
$(300,100)$	66.6870	0.6018	0.0072	59.6483	0.9967	0.0000	59.8210	0.9967	0.0000
$(500,50)$	32.7199	0.8819	0.0040	29.4007	0.9980	0.0000	29.5003	0.9742	0.0000
$(500,100)$	67.2337	0.7558	0.0082	59.7878	0.9977	0.0000	59.4491	0.9980	0.0000

Table 2: Comparison of SOC, MADMM and RADMM for solving (38) with $\mu = 1$. The results are averaged for 10 repeated experiments with random initializations.

4.2 Numerical Experiments on ODL and DPCP

In this section, we test Algorithm 1 on the DPCP problem (41), which can be equivalently written as:

$$\begin{aligned} &\min_{X,W} \|W\|_1 \\ &\text{s.t., } W = Y^\top X,\ X \in \mathrm{St}(n,p). \end{aligned} \tag{50}$$

Simple calculation shows that Algorithm 1 for the DPCP problem (41) iterates as follows.

$$\begin{aligned} X^{k+1} &:= \mathrm{Retr}_{X^k}(-\eta_k \,\mathrm{Proj}_{T_{X^k}\mathrm{St}(n,p)}(Y\Lambda^k + \rho Y(Y^\top X^k - Z^k))) \\ W^{k+1} &:= \mathrm{prox}_{\frac{1+\rho\gamma}{\rho}\|\cdot\|_1}(Y^\top X^{k+1} + \frac{1}{\rho}\Lambda^k) \\ Z^{k+1} &:= \frac{1}{1/\gamma+\rho}\left(\frac{1}{\gamma}W^{k+1} + \Lambda^k + \rho Y^\top X^{k+1}\right) \\ \Lambda^{k+1} &:= \Lambda^k + \rho(Y^\top X^{k+1} - Z^{k+1}). \end{aligned} \tag{51}$$

We compare the RADMM with iteratively reweighted least squares (IRLS) [40], projected subgradient method (PSGM) [49] and manifold proximal point algorithm (ManPPA) [9]. Note that the objective of the problem:

$$\begin{aligned} &\min_X\ F(X) := \|Y^\top X\|_1 \\ &\text{s.t.}\quad X \in \mathrm{St}(n,p) = \{X \in \mathbb{R}^{n\times p} | X^\top X = I_p\}. \end{aligned} \tag{52}$$

is separable column-wisely:

$$\begin{aligned} &\min_{x_1,\ldots,x_p} \sum_{i=1}^{p} \|Y^\top x_i\|_1 \\ &\text{s.t. } \{x_1,\ldots,x_p\} \text{ is orthonormal set.} \end{aligned} \tag{53}$$

PSGM and ManPPA conduct the minimization column-wisely. Therefore, in our experiment, we can only record the function value at the outputs of PSGM and ManPPA. Meanwhile the IRLS algorithm that we implemented here is a variant of the original column-wise algorithm for solving (41) which was also proposed in [40], IRLS iterates as follows: first we find the initialization by $X^0 := \mathrm{argmin}_{X\in\mathrm{St}(n,p)} \|Y^\top X\|_F^2$ and then the iterate is updated by

$$X^{k+1} \leftarrow \mathrm{argmin}_{X\in\mathrm{St}(n,p)} \sum_i \|X^\top Y_i\|_2^2 / \max\{\delta, \|(X^k)^\top Y_i\|_2\}. \tag{54}$$

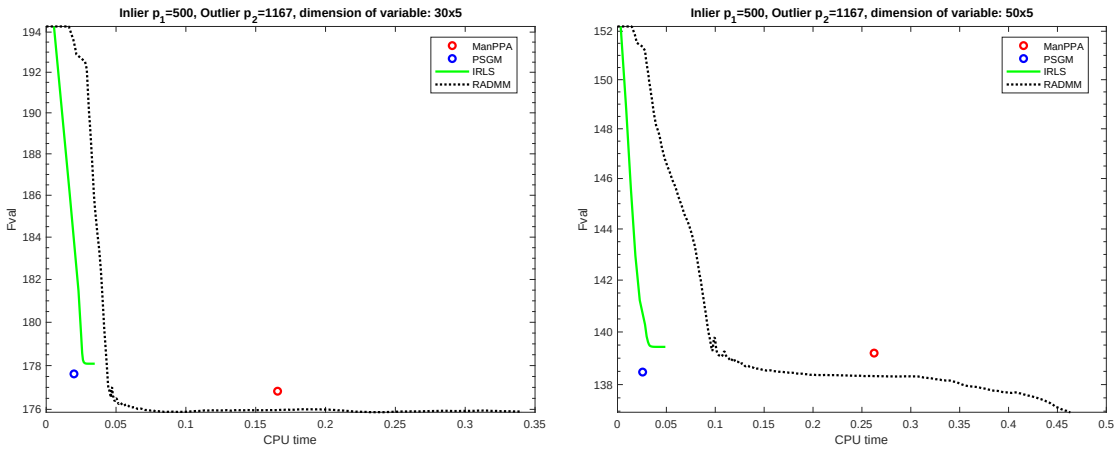


Figure 3: Function value $\|Y^\top X^k\|_1$ versus CPU time. In this experiment we set $n \in \{30, 50\}$, $p = 5$, $p_1 = 500$ and $p_2 = 1167$.

We follow the same experiment setting as [9]. More specifically, we construct the data to be $Y = [SR, O]$, $S \in \mathbb{R}^{n\times d}$ with orthogonal column vectors, $R \in \mathbb{R}^{d\times p_1}$, $O \in \mathbb{R}^{N\times p_2}$ both with random Gaussian entries. Here p_1 and p_2 are the numbers of inliers and outliers respectively as described in [10]. In our experiment we set $p = 5$, $p_1 = 500$ and $p_2 = 1167$, with different choice of n. For our RADMM algorithm we set $\rho = 40$, $\gamma = 4 \cdot 10^{-9}$, $\eta = 2 \cdot 10^{-4}$. For other algorithms, we use their default parameter settings from the papers [9, 49, 40]. For all the algorithm, we terminate them if the difference between two consecutive function values is smaller than 10^{-6}, i.e.

$$|F(X^{k+1}) - F(X^k)| < 10^{-6}.$$

We initialize IRLS and RADMM with the same initial point as in [49]. Note that PSGM and ManPPA sequentially solves the column-wise problems, and therefore they do not need the initial point to be on the Stiefel manifold. In Figure 3, we show how the objective function value changes along with the CPU time. We also record the CPU time and final objective function value in Table 3. For RADMM, we also include the constraint violation (i.e. $\|W^k - Y^\top X^k\|_F$, denoted as "infeas" in the table) in Table 3. It can be seen from Figure 3 and Table 3 that RADMM outputs the other three algorithms in terms of the objective function value.

Settings	PSGM		IRLS		ManPPA		RADMM		
(n, p)	obj	CPU	obj	CPU	obj	CPU	obj	CPU	vio
$(30, 5)$	180.59	0.0131	177.66	0.0230	177.90	0.1164	173.28	0.3177	0.0003
$(50, 5)$	141.66	0.0215	142.61	0.0404	138.78	0.1820	136.62	0.3971	0.0007
$(70, 5)$	125.94	0.0429	118.97	0.0881	119.50	0.3532	116.39	0.4526	0.0074

Table 3: Summary of function value, CPU time (seconds) of proposed RADMM Algorithm (51), comparing with PSGM [49], IRLS [41] and ManPPA [9] algorithm. The results are averaged for 10 repeated experiments with random generated data. In this experiment we set $p_1 = 500$ and $p_2 = 1167$.

We also compare our RADMM (51) with SOC [26] and MADMM [25]. The SOC (4) algorithm

for problem (41) actually solves the following equivalent problem:

$$\begin{aligned} &\min_{X,W} \ \|Y^\top X\|_1 \\ &\text{s.t., } X = W,\ W \in \mathrm{St}(n,p), \end{aligned}$$

and it iterates as:

$$\begin{aligned} X^{k+1} &:= \ \mathrm{argmin}_X \ \|Y^\top X\|_1 + \langle \Lambda^k, X - W^k\rangle + \frac{\rho}{2}\|X - W^k\|_F^2 \\ W^{k+1} &:= \ \mathrm{argmin}_{W\in \mathrm{St}(n,p)} \langle \Lambda^k, X^{k+1} - W\rangle + \frac{\rho}{2}\|X^{k+1} - W\|_F^2 \\ \Lambda^{k+1} &:= \Lambda^k + \rho(X^{k+1} - W^{k+1}). \end{aligned} \tag{55}$$

In our experiment, we chose to solve the X-subproblem by a subgradient method [2]. MADMM (5) solves (50), and updates the iterates as follows:

$$\begin{aligned} X^{k+1} &:= \ \mathrm{argmin}_{X\in \mathrm{St}(n,p)} \langle \Lambda^k, Y^\top X - W^k\rangle + \frac{\rho}{2}\|Y^\top X - W^k\|_F^2 \\ W^{k+1} &:= \ \mathrm{argmin}_W \ \|W\|_1 + \langle \Lambda^k, Y^\top X^{k+1} - W\rangle + \frac{\rho}{2}\|Y^\top X^{k+1} - W\|_F^2 \\ \Lambda^{k+1} &:= \Lambda^k + \rho(Y^\top X^{k+1} - W^{k+1}). \end{aligned} \tag{56}$$

In our experiment, we chose to solve the X-subproblem by a Riemannian gradient descent method.

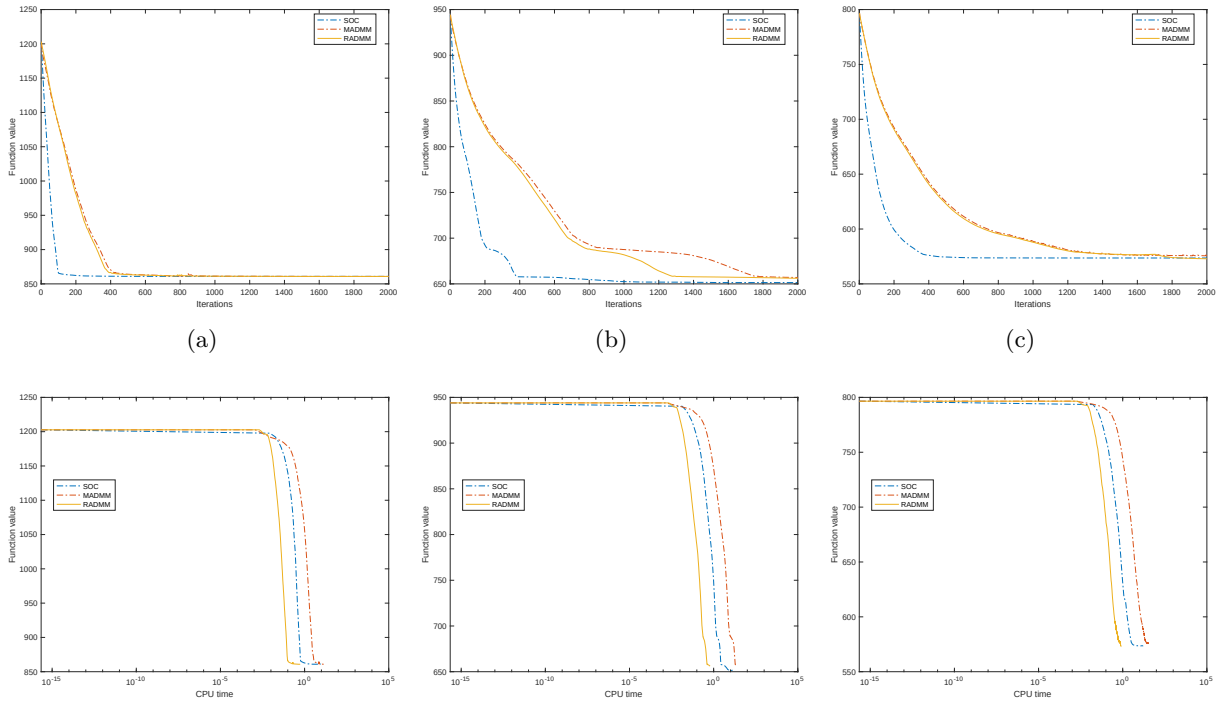
The parameters are set as follows. For SOC we set $\rho = 50$ and $\eta = 5 \cdot 10^{-6}$, where η is the stepsize for the subgradient step; for MADMM we set $\rho = 50$ and $\eta = 10^{-6}$, where η is the stepsize for the X update; for RADMM we set $\rho = 50$, $\eta = 10^{-4}$ and $\gamma = 10^{-9}$. Again, the parameters are obtained via simple grid searches, also we randomly initialize three algorithms at the same starting point. For all the three algorithms we record the function value for the sequence on the manifold, i.e. X^k for MADMM and RADMM, and W^k for SOC. We terminate the algorithms after 2000 iterations. We record the objective function values in Figure 4. We also report the objective function values of the final output (denoted as "obj") and the constraint violation ($\|X^k - W^k\|_F$ for SOC and $\|Y^\top X^k - W^k\|_F$ for MADMM and RADMM, denoted as "infeas") in Table 4. It can be seen from Figure 4 and Table 4 that RADMM is more efficient in terms of CPU time, despite small constraint violation.

Settings	SOC		MADMM		RADMM	
(n,p)	obj	infeas	obj	infeas	obj	infeas
$(30,5)$	860.9367	0.0000	860.8601	0.0019	860.8394	0.0021
$(50,5)$	651.4294	0.0000	656.9796	0.0062	656.1095	0.0066
$(70,5)$	551.2766	0.0000	564.4312	0.0137	563.8032	0.0097

Table 4: Comparison of SOC, MADMM and RADMM for solving (41). The results are averaged for 10 repeated experiments with random initializations.

5 Conclusion

In this paper, we proposed a Riemannian ADMM for solving a class of Riemannian optimization problem with nonsmooth objective function. To the best of our knowledge, this is the first Riemannian ADMM with provable convergence guarantee for solving Riemannian optimization problem with nonsmooth objective. All steps of our Riemannian ADMM are easy to compute and implement, which gives the potential to be applied to solving large-scale problems.



(a) (b) (c)

(a) $(n, p) = (30, 5)$ (b) $(n, p) = (50, 5)$ (c) $(n, p) = (70, 5)$

Figure 4: Comparison of SOC, MADMM and RADMM for solving (41). The first row is the comparison of function value decrease w.r.t. number of iterations, and the second row is w.r.t. CPU time consumed. Each figure is averaged for 10 repeated experiments with random initializations.

References

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[2] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[3] Tamir Bendory, Yonina C Eldar, and Nicolas Boumal. Non-convex phase retrieval from STFT measurements. *IEEE Transactions on Information Theory*, 64(1):467–484, 2017.

[4] Pierre B Borckmans, S Easter Selvan, Nicolas Boumal, and P-A Absil. A Riemannian subgradient algorithm for economic dispatch with valve-point effect. *Journal of Computational and Applied Mathematics*, 255:848–866, 2014.

[5] Nicolas Boumal. An introduction to optimization on smooth manifolds. *https://www.nicolasboumal.net/book/index.html*, 2020.

[6] Nicolas Boumal and Pierre-antoine Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. *NIPS*, 24:406–414, 2011.

[7] Nicolas Boumal, Pierre-Antoine Absil, and Coralia Cartis. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 39(1):1–33, 2019.

[8] Timothy Carson, Dustin G Mixon, and Soledad Villar. Manifold optimization for k-means clustering. In *2017 International Conference on Sampling Theory and Applications (SampTA)*, pages 73–77. IEEE, 2017.

[9] Shixiang Chen, Zengde Deng, Shiqian Ma, and Anthony Man-Cho So. Manifold proximal point algorithms for dual principal component pursuit and orthogonal dictionary learning. *IEEE Transactions on Signal Processing*, 69:4759–4773, 2021.

[10] Shixiang Chen, Shiqian Ma, Anthony Man-Cho So, and Tong Zhang. Proximal gradient method for nonsmooth optimization over the Stiefel manifold. *SIAM Journal on Optimization*, 30(1):210–239, 2020.

[11] Anoop Cherian and Suvrit Sra. Riemannian dictionary learning and sparse coding for positive definite matrices. *IEEE transactions on neural networks and learning systems*, 28(12):2859–2871, 2016.

[12] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007.

[13] L. Demanet and P. Hand. Scaling law for recovering the sparsest element in a subspace. *Information and Inference*, 3(4):295–309, 2014.

[14] OP Ferreira and PR Oliveira. Subgradient algorithm on Riemannian manifolds. *Journal of Optimization Theory and Applications*, 97(1):93–104, 1998.

[15] Philipp Grohs and Seyedehsomayeh Hosseini. Nonsmooth trust region algorithms for locally Lipschitz functions on Riemannian manifolds. *IMA Journal of Numerical Analysis*, 36(3):1167–1192, 2016.

[16] Philipp Grohs and Seyedehsomayeh Hosseini. ε-subgradient algorithms for locally Lipschitz functions on Riemannian manifolds. *Advances in Computational Mathematics*, 42(2):333–360, 2016.

[17] S. Hosseini. Convergence of nonsmooth descent methods via Kurdyka–Lojasiewicz inequality on Riemannian manifolds. *Hausdorff Center for Mathematics and Institute for Numerical Simulation, University of Bonn (2015,(INS Preprint No. 1523))*, 2015.

[18] Seyedehsomayeh Hosseini, Wen Huang, and Rohollah Yousefpour. Line search algorithms for locally Lipschitz functions on Riemannian manifolds. *SIAM Journal on Optimization*, 28(1):596–619, 2018.

[19] Seyedehsomayeh Hosseini and André Uschmajew. A Riemannian gradient sampling algorithm for nonsmooth optimization on manifolds. *SIAM Journal on Optimization*, 27(1):173–189, 2017.

[20] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[21] W. Huang and K. Wei. Riemannian proximal gradient methods. *Mathematical Programming*, 194:371–413, 2022.

[22] Wen Huang and Paul Hand. Blind deconvolution by a steepest descent algorithm on a quotient manifold. *SIAM Journal on Imaging Sciences*, 11(4):2757–2785, 2018.

[23] I. Jolliffe, N. Trendafilov, and M. Uddin. A modified principal component technique based on the Lasso. *Journal of computational and Graphical Statistics*, 12(3):531–547, 2003.

[24] M. Journee, Yu. Nesterov, P. Richtarik, and R. Sepulchre. Generalized power method for sparse principal component analysis. *J. Mach. Learn. Res.*, 11:517–553, 2010.

[25] A. Kovnatsky, K. Glashoff, and M. M. Bronstein. MADMM: a generic algorithm for non-smooth optimization on manifolds. In *ECCV*, pages 680–696, 2016.

[26] Rongjie Lai and Stanley Osher. A splitting method for orthogonality constrained problems. *Journal of Scientific Computing*, 58(2):431–449, 2014.

[27] G. Lerman and T. Maunu. An overview of robust subspace recovery. *Proceedings of the IEEE*, 106(8):1380–1410, 2018.

[28] J. Li, K. Balasubramanian, and S. Ma. Stochastic zeroth-order Riemannian derivative estimation and optimization. *accepted in Mathematics of Operations Research*, 2022.

[29] Xiao Li, Shixiang Chen, Zengde Deng, Qing Qu, Zhihui Zhu, and Anthony Man Cho So. Weakly convex optimization over Stiefel manifold using Riemannian subgradient-type methods. *SIAM J. Optimization*, 31(3):1605–1634, 2021.

[30] Canyi Lu, Jiashi Feng, Zhouchen Lin, and Shuicheng Yan. Nonconvex sparse spectral clustering by alternating direction method of multipliers and its convergence analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[31] S. Ma. Alternating direction method of multipliers for sparse principal component analysis. *Journal of the Operations Research Society of China*, 1(2):253–274, 2013.

[32] Seyoung Park and Hongyu Zhao. Spectral clustering based on learning similarity matrix. *Bioinformatics*, 34(12):2069–2076, 2018.

[33] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[34] Q. Qu, J. Sun, and J. Wright. Finding a sparse vector in subspace: linear sparsity using alternating directions. *IEEE Trans. Information Theory*, 62(10):5855–5880, 2016.

[35] H. Shen and J. Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99(6):1015–1034, 2008.

[36] D. Spielman, H. Wang, and J. Wright. Exact recovery of sparsely-used dictionaries. In *COLT*, 2012.

[37] J. Sun, Q. Qu, and J. Wright. Complete dictionary recovery over the sphere i: Overview and the geometric picture. *IEEE Trans. Information Theory*, 63(2):853–884, 2017.

[38] J. Sun, Q. Qu, and J. Wright. Complete dictionary recovery over the sphere ii: Recovery by Riemannian trust-region method. *IEEE Trans. Information Theory*, 63(2):885–914, 2017.

[39] Ju Sun, Qing Qu, and John Wright. A geometric analysis of phase retrieval. *Foundations of Computational Mathematics*, 18(5):1131–1198, 2018.

[40] M. C. Tsakiris and R. Vidal. Dual principal component pursuit. *Journal of Machine Learning Research*, 2018.

[41] Manolis C Tsakiris and René Vidal. Dual principal component pursuit. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 10–18, 2015.

[42] Bart Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

[43] Yu Wang, Wotao Yin, and Jinshan Zeng. Global convergence of ADMM in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, 78(1):29–63, 2019.

[44] Zhongruo Wang, Bingyuan Liu, Shixiang Chen, Shiqian Ma, Lingzhou Xue, and Hongyu Zhao. A manifold proximal linear method for sparse spectral clustering with application to single-cell rna sequencing data analysis. *INFORMS Journal on Optimization*, 4(2):200–214, 2022.

[45] Xiantao Xiao, Yongfeng Li, Zaiwen Wen, and Liwei Zhang. A regularized semi-smooth Newton method with projection steps for composite convex programs. *Journal of Scientific Computing*, 76(1):364–389, 2018.

[46] W. H. Yang, L.-H. Zhang, and R. Song. Optimality conditions for the nonlinear programming problems on Riemannian manifolds. *Pacific J. Optimization*, 10(2):415–434, 2014.

[47] Jinshan Zeng, Wotao Yin, and Ding-Xuan Zhou. Moreau envelope augmented lagrangian method for nonconvex optimization with linear constraints. *Journal of Scientific Computing*, 91(2):1–36, 2022.

[48] Junyu Zhang, Shiqian Ma, and Shuzhong Zhang. Primal-dual optimization algorithms over Riemannian manifolds: an iteration complexity analysis. *Mathematical Programming*, 184(1):445–490, 2020.

[49] Zhihui Zhu, Yifan Wang, Daniel Robinson, Daniel Naiman, Rene Vidal, and Manolis Tsakiris. Dual principal component pursuit: Improved analysis and efficient algorithms. *Advances in Neural Information Processing Systems*, 31, 2018.

[50] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *J. Comput. Graph. Stat.*, 15(2):265–286, 2006.

[51] Hui Zou and Lingzhou Xue. A selective overview of sparse principal component analysis. *Proceedings of the IEEE*, 106(8):1311–1320, 2018.