Proximal Gradient Method for Nonsmooth Optimization over the Stiefel Manifold

Shixiang Chen* Shiqian Ma† Anthony Man-Cho So‡ Tong Zhang§

September 2, 2019

Abstract

We consider optimization problems over the Stiefel manifold whose objective function is the summation of a smooth function and a nonsmooth function. Existing methods for solving this kind of problems can be classified into three classes. Algorithms in the first class rely on information of the subgradients of the objective function and thus tend to converge slowly in practice. Algorithms in the second class are proximal point algorithms, which involve subproblems that can be as difficult as the original problem. Algorithms in the third class are based on operator-splitting techniques, but they usually lack rigorous convergence guarantees. In this paper, we propose a retraction-based proximal gradient method for solving this class of problems. We prove that the proposed method globally converges to a stationary point. Iteration complexity for obtaining an ϵ-stationary solution is also analyzed. Numerical results on solving sparse PCA and compressed modes problems are reported to demonstrate the advantages of the proposed method.

Keywords— Manifold Optimization; Stiefel Manifold; Nonsmooth; Proximal Gradient Method; Iteration Complexity; Semi-smooth Newton Method; Sparse PCA; Compressed Modes

1 Introduction

Optimization over Riemannian manifolds has recently drawn a lot of attention due to its applications in many different fields, including low-rank matrix completion [18, 76], phase retrieval [10, 73], phase synchronization [17, 57], blind deconvolution [47], and dictionary learning [23, 72]. Manifold optimization seeks to minimize an objective function over a smooth manifold. Some commonly encountered manifolds include the sphere, Stiefel manifold, Grassmann manifold, and Hadamard manifold. The recent monograph by Absil et al. [4] studies this topic in depth. In particular, it studies several important classes of algorithms for manifold optimization with smooth objective, including line-search method, Newton's method, and trust-region method. There are also many gradient-based algorithms for solving manifold optimization problems, including [79, 68, 69, 56, 49, 87]. However, all these methods require computing the derivatives of the objective function and do not apply to the case where the objective function is nonsmooth.

In this paper, we focus on a class of nonsmooth nonconvex optimization problems over the Stiefel manifold that takes the form

$$\min \ F(X) := f(X) + h(X), \ \text{s.t.}, X \in \mathcal{M} := \mathrm{St}(n,r) = \{X : X \in \mathbb{R}^{n\times r}, X^\top X = I_r\}, \tag{1.1}$$

where I_r denotes the $r\times r$ identity matrix ($r \le n$). Throughout this paper, we make the following assumptions about (1.1):

[*] Department of Industrial & Systems Engineering, Texas A& M University

[†] Department of Mathematics, University of California, Davis

[‡] Department of Systems Engineering and Engineering Management, and, by courtesy, CUHK-BGI Innovation Institute of Trans-omics, The Chinese University of Hong Kong

[§] Departments of Computer Science and Mathematics, The Hong Kong University of Science and Technology

Assumption 1.1. *(i) f is smooth, possibly nonconvex, and its gradient ∇f is Lipschitz continuous with Lipschitz constant L.*

(ii) h is convex, possibly nonsmooth, and is Lipschitz continuous with constant L_h.

Note that here the smoothness, Lipschitz continuity and convexity are interpreted when the function in question is considered as a function in the ambient Euclidean space.

We restrict our discussions in this paper to (1.1) because it already finds many important applications in practice. In the following we briefly mention some representative applications of (1.1). For more examples of manifold optimization with nonsmooth objectives, we refer the reader to [3].

Example 1. Sparse Principal Component Analysis. Principal Component Analysis (PCA), proposed by Pearson [63] and later developed by Hotelling [46], 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 r ($r < \min\{m, n\}$) sparse loading vectors can be formulated as

$$\begin{array}{ll} \min_{X\in\mathbb{R}^{n\times r}} & -\mathrm{Tr}(X^\top A^\top AX) + \mu\|X\|_1 \\ \text{s.t.} & X^\top X = I_r, \end{array} \tag{1.2}$$

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 [50], where the model is called SCoTLASS and imposes sparsity and orthogonality to the loading vectors simultaneously. When $\mu = 0$, (1.2) reduces to computing the leading r 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 (1.2) when $r = 1$. In this case, (1.2) 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 $r > 1$. Existing works, including [94, 25, 70, 51, 58], do not impose orthogonal loading directions. As discussed in [51], "Simultaneously enforcing sparsity and orthogonality seems to be a hard (and perhaps questionable) task." We refer the interested reader to [95] for more details on existing algorithms for solving sparse PCA. As we will discuss later, our algorithm can solve (1.2) with $r > 1$ (i.e., imposing sparsity and orthogonality simultaneously) efficiently.

Example 2. Compressed Modes in Physics. This problem seeks spatially localized ("sparse") solutions of the independent-particle Schrödinger's equation. Sparsity is achieved by adding an L_1 regularization of the wave functions, which leads to solutions with compact support ("compressed modes"). For 1D free-electron case, after proper discretization, this problem can be formulated as

$$\begin{array}{ll} \min_{X\in\mathbb{R}^{n\times r}} & \mathrm{Tr}(X^\top HX) + \mu\|X\|_1 \\ \text{s.t.} & X^\top X = I_r, \end{array} \tag{1.3}$$

where H denotes the discretized Schrödinger operator. Note that the L_1 regularization reduces to the ℓ_1 norm of X after discretization. We refer the reader to [62] for more details of this problem. Note that (1.2) and (1.3) are different in the way that H and $A^\top A$ have totally different structures. In particular, H is the discretized Schrödinger Hamiltonian, which is a block circulant matrix, while A in (1.2) usually comes from statistical data and thus $A^\top A$ is usually dense and unstructured. These differences may affect the performance of algorithms for solving them.

Example 3. Unsupervised Feature Selection. It is much more difficult to select the discriminative features in unsupervised learning than supervised learning. There are some recent works that model this task as a manifold optimization problem in the form of (1.1). For instance, [85] and [74] assume that there is a linear classifier W which classifies each data point x_i (where $i = 1, \ldots, n$) in the training data set to a class, and by denoting $G_i = W^\top x_i$, $[G_1, \ldots, G_n]$ gives a scaled label matrix which can be used to define some local discriminative scores. The target is to train a W such that the local discriminative scores are the highest for all the training data $x_1, \ldots, x_n$. It is suggested in [85] and [74] to solve the following model to find W:

$$\begin{array}{ll} \min_{W\in\mathbb{R}^{n\times r}} & \mathrm{Tr}(W^\top MW) + \mu\|W\|_{2,1} \\ \text{s.t.} & W^\top W = I_r, \end{array}$$

where M is a given matrix computed from the input data, the $\ell_{2,1}$ norm is defined as $\|W\|_{2,1} = \sum_{i=1}^n \|W(i,:)\|_2$ with $W(i,:)$ being the i-th row of W, which promotes the row sparsity of W, and the orthogonal constraint is imposed to avoid arbitrary scaling and the trivial solution of all zeros. We refer the reader to [85] and [74] for more details.

Example 4. Sparse Blind Deconvolution. Given the observations

$$y = a_0 \circledast x_0 \in \mathbb{R}^m,$$

how can one recover both the convolution kernel $a_0 \in \mathbb{R}^k$ and signal $x_0 \in \mathbb{R}^m$? Here x_0 is assumed to have a sparse and random support and $\circledast$ denotes the convolution operator. This problem is known as sparse blind deconvolution. Some recent works on this topic suggest the following optimization formulation to recover a_0 and sparse x_0 (see, e.g., [90]):

$$\begin{array}{ll} \min_{a,x} & \|y - a \circledast x\|_2^2 + \mu\|x\|_1 \\ \text{s.t.} & \|a\|_2 = 1. \end{array}$$

Note that the sphere constraint here is a special case of the Stiefel manifold; i.e., $\mathrm{St}(k, 1)$.

Example 5. Nonconvex Regularizer. Problem (1.1) also allows nonconvex regularizer functions. For example, instead of using the ℓ_1 norm to promote sparsity, we can use the MCP (minimax concave penalty) function [86], which has been widely used in statistics. The MCP function is nonconvex and is given by

$$P(x) = \begin{cases} \lambda|x| - \frac{x^2}{2\lambda}, & \text{if } |x| \le \gamma\lambda, \\ \frac{1}{2}\gamma\lambda^2, & \text{otherwise,} \end{cases}$$

where λ and γ are given parameters, and $x \in \mathbb{R}$. If we replace the ℓ_1 norm in sparse PCA (1.2) by MCP, it reduces to

$$\begin{array}{ll} \min_{X \in \mathbb{R}^{n \times r}} & -\mathrm{Tr}(X^\top A^\top A X) + \mu \sum_{ij} P(X_{ij}) \\ \text{s.t.} & X^\top X = I_r. \end{array} \tag{1.4}$$

It is easy to see that the objective function in (1.4) can be rewritten as $f_1(X) + f_2(X)$, with $f_1(X) = -\mathrm{Tr}(X^\top A^\top A X) + \mu(\sum_{ij} P(X_{ij}) - \lambda\|X\|_1)$ and $f_2(X) = \mu\lambda\|X\|_1$. Note that f_1 is smooth and its gradient is Lipschitz continuous. Therefore, (1.4) is an instance of (1.1).

Our Contributions. Due to the needs of the above-mentioned applications, it is highly desirable to design an efficient algorithm for solving (1.1). In this paper, we propose a proximal gradient method for solving it. The proposed method, named ManPG (Manifold Proximal Gradient Method), is based on the proximal gradient method with a retraction operation to keep the iterates feasible with respect to the manifold constraint. Each step of ManPG involves solving a well-structured convex optimization problem, which can be done efficiently by the semi-smooth Newton method. We prove that ManPG converges to a stationary point of (1.1) globally. We also analyze the iteration complexity of ManPG for obtaining an ϵ-stationary point. Numerical results on sparse PCA (1.2) and compressed modes (1.3) problems show that our ManPG algorithm compares favorably with existing methods.

Notation. The following notation is adopted throughout this paper. The tangent space to $\mathcal{M}$ at point X is denoted by $\mathrm{T}_X\mathcal{M}$. We use $\langle A, B\rangle = \mathrm{Tr}(A^\top B)$ to denote the Euclidean inner product of two matrices A, B. 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 $\langle \xi, \eta\rangle_X = \mathrm{Tr}(\xi^\top \eta)$. We use $\|X\|_\mathrm{F}$ to denote the Frobenius norm of X and $\|\mathcal{A}\|_{op}$ to denote the operator norm of a linear operator $\mathcal{A}$. 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. According to [4], the projection of Y onto the tangent space at $X \in \mathrm{St}(n, r)$ is given by $\mathrm{Proj}_{T_X \mathrm{St}(n,r)} = (I_n - XX^\top)Y + \frac{1}{2}X(X^\top Y - Y^\top X)$. We use Retr to denote the retraction operation. For a convex function h, its Euclidean subgradient and Riemannian subgradient are denoted by ∂h and $\hat{\partial} h$, respectively. We use $\mathrm{vec}(X)$ to denote the vector formed by stacking the column vectors of X. The set of $r \times r$ symmetric matrices is denoted by S^r. Given an $X \in S^r$, we use $\overline{\mathrm{vec}}(X)$ to denote the $\frac{1}{2}r(r+1)$-dimensional vector obtained from $\mathrm{vec}(X)$ by eliminating all super-diagonal elements of X. We denote $Z \succeq 0$ if $(Z + Z^\top)/2$ is positive semidefinite. The proximal mapping of h at point X is defined by $\mathrm{prox}_h(X) = \mathrm{argmin}_Y \frac{1}{2}\|Y - X\|_\mathrm{F}^2 + h(Y)$.

Organization. The rest of this paper is organized as follows. In Section 2 we briefly review existing works on solving manifold optimization problems with nonsmooth objective functions. We introduce some preliminaries of manifolds in Section 3. The main algorithm ManPG and the semi-smooth Newton method for solving the subproblem are presented in Section 4. In Section 5, we establish the global convergence of ManPG and analyze its iteration complexity for obtaining an ϵ-stationary solution. Numerical results of ManPG on solving compressed modes problems in physics and sparse PCA are reported in Section 6. Finally, we draw some concluding remarks in Section 7.

2 Nonsmooth Optimization over Riemannian Manifold

Unlike manifold optimization with a smooth objective, which has been studied extensively in the monograph [4], the literature on manifold optimization with a nonsmooth objective has been relatively limited. Numerical algorithms for solving manifold optimization with nonsmooth objectives can be roughly classified into three categories: subgradient-oriented methods, proximal point algorithms, and operator-splitting methods. We now briefly discuss the existing works in these three categories.

2.1 Subgradient-oriented Methods

Algorithms in the first category include the ones proposed in [30, 16, 40, 42, 45, 43, 9, 26, 39], which are all subgradient-oriented methods. Ferreira and Oliveria [30] studied the convergence of subgradient method for minimizing a convex function over a Riemannian manifold. The subgradient method generates the iterates via

$$X_{k+1} = \exp_{X_k}(t_k V_k),$$

where $\exp_{X_k}$ is the exponential mapping at X_k and V_k denotes a Riemannian subgradient of the objective. Like the subgradient method in Euclidean space, the stepsize t_k is chosen to be diminishing to guarantee convergence. However, the result in [30] does not apply to (1.1) because it is known that every smooth function that is convex on a compact Riemannian manifold is a constant [14]. This motivated some more advanced works on Riemannian subgradient method. Specifically, Dirr et al. [26] and Borckmans et al. [16] proposed subgradient methods on manifold for the case where the objective function is the pointwise maximum of smooth functions. In this case, some generalized gradient can be computed and a descent direction can be found by solving a quadratic program. Grohs and Hosseini [40] proposed a Riemannian ε-subgradient method. Hosseini and Uschmajew [45] proposed a Riemannian gradient sampling algorithm. Hosseini et al. [43] generalized the Wolfe conditions and extended the BFGS algorithm to nonsmooth functions on Riemannian manifolds. Grohs and Hosseini [39] generalized a nonsmooth trust region method to manifold optimization. Hosseini [42] studied the convergence of some subgradient-oriented descent methods based on the Kurdyka-Łojasiewicz (KŁ) inequality. Roughly speaking, all the methods studied in [26, 16, 40, 45, 43, 39, 42] require subgradient information to build a quadratic program to find a descent direction:

$$\hat{g} \longleftarrow \min_{g \in \mathrm{conv}(W)} \|g\|, \tag{2.1}$$

where $\mathrm{conv}(W)$ denotes the convex hull of set $W = \{G_j, j = 1, \ldots, J\}$, G_j is the Riemannian gradient of a differentiable point around the current iterate X, and J usually needs to be larger than the dimension of $\mathcal{M}$. Subsequently, the iterate X is updated by $X^+ = \mathrm{Retr}_X(\alpha\hat{g})$, where the stepsize α is found by line search. For high-dimensional problems on the Stiefel manifold $\mathrm{St}(n, r)$, (2.1) can be difficult to solve because n is large. Since subgradient algorithm is known to be slower than the gradient algorithm and proximal gradient algorithm in Euclidean space, it is expected that these subgradient-based algorithms are not as efficient as gradient algorithms and proximal gradient algorithms on manifold in practice.

2.2 Proximal Point Algorithms

Proximal point algorithms (PPAs) for solving manifold optimization are also studied in the literature. Ferreira and Oliveira [31] extended PPA to manifold optimization, which in each iteration needs to minimize the original function plus a proximal term over the manifold. However, there are two issues that limit its applicability. The first is that the subproblem can be as difficult as the original problem. For example, Bacak et al. [9] suggested to use the subgradient method to solve the subproblem, but they require the subproblem to be in the form of the pointwise maximum of smooth functions tackled in [16]. The second is that the discussions in the literature mainly focus on the Hadamard manifold and exploit heavily the convexity assumption of the objective function. Thus, they do not apply to compact manifolds such as $\mathrm{St}(n,r)$. Bento et al. [12] aimed to resolve the second issue and proved the convergence of the PPA for more general Riemannian manifolds under the assumption that the KŁ inequality holds for the objective function. In [11], Bento et al. analyzed the convergence of some inexact descent methods based on the KŁ inequality, including the PPA and steepest descent method. In a more recent work [13], Bento et al. studied the iteration complexity of PPA under the assumption that the constraint set is the Hadamard manifold and the objective function is convex. Nevertheless, the results in [31, 12, 11, 13] seem to be of theoretical interest only because no numerical results were shown. As mentioned earlier, this could be due to the difficulty in solving the PPA subproblems.

2.3 Operator Splitting Methods

Operator-splitting methods do not require subgradient information, and existing works in the literature mainly focus on the Stiefel manifold. Note that (1.1) is challenging because of the combination of two difficult terms: Riemannian manifold and nonsmooth objective. If only one of them is present, then the problem is relatively easy to solve. Therefore, the alternating direction method of multipliers (ADMM) becomes a natural choice for solving (1.1). ADMM for solving convex optimization problems with two block variables is closely related to the famous Douglas-Rachford operator splitting method, which has a long history [36, 33, 55, 32, 35, 27]. The renaissance of ADMM was initiated by several papers around 2007-2008, where it was successfully applied to solve various signal processing [24] and image processing problems [82, 37, 6]. The recent survey paper [21] popularized this method in many areas. Recently, there have been some emerging interests in ADMM for solving manifold optimization of the form (1.1); see, e.g., [53, 52, 89, 78]. However, the algorithms presented in these papers either lack convergence guarantee ([53, 52]) or their convergence needs further conditions that do not apply to (1.1) ([78, 89]).

Here we briefly describe the SOC method (Splitting method for Orthogonality Constrained problems) presented in [53]. The SOC method aims to solve

$$\min \ J(X), \ \text{s.t.}, \ X \in \mathcal{M}$$

by introducing an auxiliary variable P and considering the following reformulation:

$$\min \ J(P), \ \text{s.t.}, P = X, X \in \mathcal{M}. \tag{2.2}$$

By associating a Lagrange multiplier Λ to the linear equality constraint, the augmented Lagrangian function of (2.2) can be written as

$$\mathcal{L}_\beta(X, P; \Lambda) := J(P) - \langle \Lambda, P - X\rangle + \frac{\beta}{2}\|P - X\|_F^2,$$

where $\beta > 0$ is a penalty parameter. The SOC algorithm then generates its iterates as follows:

$$\begin{array}{lcl} P^{k+1} & := & \operatorname{argmin}_P \ \mathcal{L}_\beta(P, X^k; \Lambda^k), \\ X^{k+1} & := & \operatorname{argmin}_X \ \mathcal{L}_\beta(P^{k+1}, X; \Lambda^k), \text{s.t.}, X \in \mathcal{M}, \\ \Lambda^{k+1} & := & \Lambda^k - \beta(P - X). \end{array}$$

Note that the X-subproblem corresponds to the projection onto $\mathcal{M}$, and the P-subproblem is an unconstrained problem whose complexity depends on the structure of J. In particular, if J is smooth, then the P-subproblem can be solved iteratively by the gradient method; if J is nonsmooth and has an easily computable proximal mapping, then the P-subproblem can be solved directly by computing the proximal mapping of J.

The MADMM (manifold ADMM) algorithm presented in [52] aims to solve the following problem:

$$\min_{X,Z} \ f(X) + g(Z), \ \text{s.t.}, \ Z = AX, \ X \in \mathrm{St}(n,r), \tag{2.3}$$

where f is smooth and g is nonsmooth with an easily computable proximal mapping. The augmented Lagrangian function of (2.3) is

$$\mathcal{L}_\beta(X, Z; \Lambda) := f(X) + g(Z) - \langle \Lambda, Z - AX\rangle + \frac{\beta}{2}\|Z - AX\|_F^2$$

and the MADMM algorithm generates its iterates as follows:

$$\begin{array}{lll} X^{k+1} & := & \mathrm{argmin}_X \ \mathcal{L}_\beta(X, Z^k; \Lambda^k), \ \text{s.t.}, X \in \mathrm{St}(n,r), \\ Z^{k+1} & := & \mathrm{argmin}_Z \ \mathcal{L}_\beta(X^{k+1}, Z; \Lambda^k), \\ \Lambda^{k+1} & := & \Lambda^k - \beta(Z^{k+1} - AX^{k+1}). \end{array}$$

Note that the X-subproblem is a smooth optimization problem on the Stiefel manifold, and the authors suggested to use the Manopt toolbox [20] to solve it. The Z-subproblem corresponds to the proximal mapping of function g.

As far as we know, however, the convergence guarantees of SOC and MADMM are still missing in the literature. Though there are some recent works that analyze the convergence of ADMM for nonconvex problems [78, 89], their results need further conditions that do not apply to (1.1) and its reformulations (2.2) and (2.3).

More recently, some other variants of the augmented Lagrangian method are proposed to deal with (1.1). In [22], Chen et al. proposed a PAMAL method which hybridizes an augmented Lagrangian method with the proximal alternating minimization method [7]. More specifically, PAMAL solves the following reformulation of (1.1):

$$\min_{X,Q,P} \ f(P) + h(Q), \ \text{s.t.}, \ Q = X, P = X, X \in \mathrm{St}(n,r). \tag{2.4}$$

By associating Lagrange multipliers Λ_1 and Λ_2 to the two linear equality constraints, the augmented Lagrangian function of (2.4) can be written as

$$\mathcal{L}_\beta(X,Q,P;\Lambda_1,\Lambda_2) := f(P) + h(Q) - \langle \Lambda_1, Q - X\rangle - \langle \Lambda_2, P - X\rangle + \frac{\beta}{2}\|Q - X\|_F^2 + \frac{\beta}{2}\|P - X\|_F^2,$$

where $\beta > 0$ is a penalty parameter. The augmented Lagrangian method for solving (2.4) is then given by

$$\begin{array}{lll} (X^{k+1}, Q^{k+1}, P^{k+1}) & := & \mathrm{argmin}_{X,Q,P} \ \mathcal{L}_\beta(X,Q,P;\Lambda_1^k,\Lambda_2^k), \ \text{s.t.}, \ X \in \mathrm{St}(n,r), \\ \Lambda_1^{k+1} & := & \Lambda_1^k - \beta(Q^{k+1} - X^{k+1}), \\ \Lambda_2^{k+1} & := & \Lambda_2^k - \beta(P^{k+1} - X^{k+1}). \end{array} \tag{2.5}$$

Note that the subproblem in (2.5) is still difficult to solve. Therefore, the authors of [22] suggested to use the proximal alternating minimization method [7] to solve the subproblem in (2.5) inexactly. They named the augmented Lagrangian method (2.5) with subproblems being solved by the proximal alternating minimization method as PAMAL. They proved that under certain conditions, any limit point of the sequence generated by PAMAL is a KKT point of (2.4). It needs to be pointed out that the proximal alternating minimization procedure involves many parameters that need to be tuned in order to solve the subproblem inexactly. Our numerical results in Section 6 indicate that the performance of PAMAL significantly depends on the setting of these parameters.

In [93], Zhu et al. studied another algorithm called EPALMAL for solving (1.1) that is based on the augmented Lagrangian method and the PALM algorithm [15]. The difference between EPALMAL and PAMAL is that they use different algorithms to minimize the augmented Lagrangian function inexactly. In particular, EPALMAL uses the PALM algorithm [15], while PAMAL uses PAM [7]. It is also shown in [93] that any limit point of the sequence generated by EPALMAL is a KKT point. However, their result assumes that the iterate sequence is bounded, which holds automatically if the manifold in question is bounded but is hard to verify otherwise.

3 Preliminaries on Manifold Optimization

We first introduce the elements of manifold optimization that will be needed in the study of (1.1). In fact, our discussion in this section applies to the case where $\mathcal{M}$ is any embedded submanifold of an Euclidean space. To begin, we say that a function F is locally Lipschitz continuous if for any $X \in \mathcal{M}$, it is Lipschitz continuous in a neighborhood of X. Note that if F is locally Lipschitz continuous in the Euclidean space $\mathcal{E}$, then it is also locally Lipschitz continuous when restricted to the embedded submanifold $\mathcal{M}$ of $\mathcal{E}$.

Definition 3.1. *(Generalized Clarke subdifferential [44]) For a locally Lipschitz function F on $\mathcal{M}$, the Riemannian generalized directional derivative of F at $X \in \mathcal{M}$ in the direction V is defined by*

$$F^{\circ}(X, V) = \limsup_{Y \to X, t \downarrow 0} \frac{F \circ \phi^{-1}(\phi(Y) + tD\phi(X)[V]) - F \circ \phi^{-1}(\phi(Y))}{t},$$

where (ϕ, U) is a coordinate chart at X. The generalized gradient or the Clarke subdifferential of F at $X \in \mathcal{M}$, denoted by $\hat{\partial} F(X)$, is given by

$$\hat{\partial} F(X) = \{\xi \in \mathrm{T}_X\mathcal{M} : \langle \xi, V \rangle \leq F^{\circ}(X, V),\ \forall V \in \mathrm{T}_X\mathcal{M}\}.$$

Definition 3.2. *([84]) A function f is said to be regular at $X \in \mathcal{M}$ along $\mathrm{T}_X\mathcal{M}$ if*

- *for all $V \in \mathrm{T}_X\mathcal{M}$, $f'(X; V) = \lim_{t \downarrow 0} \dfrac{f(X + tV) - f(X)}{t}$ exists, and*
- *for all $V \in \mathrm{T}_X\mathcal{M}$, $f'(X; V) = f^{\circ}(X; V)$.*

For a smooth function f, we know that $\operatorname{grad} f(X) = \operatorname{Proj}_{\mathrm{T}_X\mathcal{M}} \nabla f(X)$ by our choice of the Riemannian metric. According to Theorem 5.1 in [84], for a regular function F, we have $\hat{\partial} F(X) = \operatorname{Proj}_{\mathrm{T}_X\mathcal{M}}(\partial F(X))$. Moreover, the function $F(X) = f(X) + h(X)$ in problem (1.1) is regular according to Lemma 5.1 in [84]. Therefore, we have $\hat{\partial} F(X) = \operatorname{Proj}_{\mathrm{T}_X\mathcal{M}}(\nabla f(X) + \partial h(X)) = \operatorname{grad} f(X) + \operatorname{Proj}_{\mathrm{T}_X\mathcal{M}}(\partial h(X))$. By Theorem 4.1 in [84], the first-order necessary condition of problem (1.1) is given by $0 \in \operatorname{grad} f(X) + \operatorname{Proj}_{\mathrm{T}_X\mathcal{M}}(\partial h(X))$.

Definition 3.3. *A point $X \in \mathcal{M}$ is called a stationary point of problem (1.1) if it satisfies the first-order necessary condition; i.e., $0 \in \operatorname{grad} f(X) + \operatorname{Proj}_{\mathrm{T}_X\mathcal{M}}(\partial h(X))$.*

A classical geometric concept in the study of manifolds is that of an exponential mapping, which defines a geodesic curve on the manifold. However, the exponential mapping is difficult to compute in general. The concept of a retraction [4], which is a first-order approximation of the exponential mapping and can be more amenable to computation, is given as follows.

Definition 3.4. *[4, Definition 4.1.1] A retraction on a differentiable manifold $\mathcal{M}$ is a smooth mapping* Retr *from the tangent bundle $\mathrm{T}\mathcal{M}$ onto $\mathcal{M}$ satisfying the following two conditions (here Retr_X denotes the restriction of* Retr *onto $\mathrm{T}_X\mathcal{M}$):*

1. *$\mathrm{Retr}_X(0) = X, \forall X \in \mathcal{M}$, where 0 denotes the zero element of $\mathrm{T}_X\mathcal{M}$.*
2. *For any $X \in \mathcal{M}$, it holds that*

$$\lim_{\mathrm{T}_X\mathcal{M} \ni \xi \to 0} \frac{\|\mathrm{Retr}_X(\xi) - (X + \xi)\|_F}{\|\xi\|_F} = 0.$$

Remark 3.5. *Since $\mathcal{M}$ is an embedded submanifold of $\mathbb{R}^{n \times r}$, we can treat X and ξ as elements in $\mathbb{R}^{n \times r}$ and hence their sum is well defined. The second condition in Definition 3.4 ensures that $\mathrm{Retr}_X(\xi) = X + \xi + \mathcal{O}(\|\xi\|_F^2)$ and $D\mathrm{Retr}_X(0) = \mathrm{Id}$, where $D\mathrm{Retr}_X$ is the differential of Retr_X and Id denotes the identity mapping. For more details about retraction, we refer the reader to [4, 19] and the references therein.*

The retraction onto the Euclidean space is simply the identity mapping; i.e., $\mathrm{Retr}_X(\xi) = X + \xi$. For the Stiefel manifold $\mathrm{St}(n, r)$, common retractions include the exponential mapping [28]

$$\mathrm{Retr}_X^{\exp}(t\xi) = [X, Q] \exp\left(t \begin{bmatrix} -X^\top \xi & -R^\top \\ R & 0 \end{bmatrix}\right) \begin{bmatrix} I_r \\ 0 \end{bmatrix},$$

where $QR = -(I_n - XX^\top)\xi$ is the unique QR factorization; the polar decomposition

$$\mathrm{Retr}_X^{\mathrm{polar}}(\xi) = (X + \xi)(I_r + \xi^\top \xi)^{-1/2};$$

the QR decomposition

$$\mathrm{Retr}_X^{\mathrm{QR}}(\xi) = \mathrm{qf}(X + \xi),$$

where $\mathrm{qf}(A)$ is the Q factor of the QR factorization of A; the Cayley transformation [79]

$$\mathrm{Retr}_X^{\mathrm{cayley}}(\xi) = \left(I_n - \frac{1}{2}W(\xi)\right)^{-1}\left(I_n + \frac{1}{2}W(\xi)\right)X,$$

where $W(\xi) = (I_n - \frac{1}{2}XX^\top)\xi X^\top - X\xi^\top(I_n - \frac{1}{2}XX^\top)$.

For any matrix $Y \in \mathbb{R}^{n\times r}$ with $r \le n$, its orthogonal projection onto the Stiefel manifold $\mathrm{St}(n,r)$ is given by $UI_rV^\top$, where U, V are the left and right singular vectors of Y, respectively. If Y has full rank, then the projection can be computed by $Y(Y^\top Y)^{-1/2}$, which is the same as the polar decomposition. The total cost of computing the projection $UI_rV^\top$ is $8nr^2 + \mathcal{O}(r^3)$ flops, where the SVD needs $6nr^2 + \mathcal{O}(r^3)$ flops [38] and the formation of $UI_rV^\top$ needs $2nr^2$ flops. By comparison, if $Y = X + \xi$ and $\xi \in T_X\mathcal{M}$, then the exponential mapping takes $8nr^2 + \mathcal{O}(r^3)$ flops and the polar decomposition takes $3nr^2 + \mathcal{O}(r^3)$ flops, where $\xi^\top\xi$ needs nr^2 flops and the remaining $2nr^2 + \mathcal{O}(r^3)$ flops come from the final assembly. Thus, polar decomposition is cheaper than the projection. Moreover, the QR decomposition of $X + \xi$ takes $2nr^2 + \mathcal{O}(r^3)$ flops. For the Cayley transformation of $X + \xi$, the total cost is $7nr^2 + \mathcal{O}(r^3)$ [79, 48]. In our algorithm that will be introduced later, we need to perform one retraction operation in each iteration. We need to point out that retractions may also affect the overall convergence speed of the algorithm. As a result, determining the most efficient retraction used in the algorithm is still an interesting question to investigate in practice; see also the discussion after Theorem 3 of [56].

The retraction Retr has the following properties that are useful for our convergence analysis:

Fact 3.6. *([19, 56]) Let $\mathcal{M}$ be a compact embedded submanifold of an Euclidean space. For all $X \in \mathcal{M}$ and $\xi \in \mathrm{T}_X\mathcal{M}$, there exist constants $M_1 > 0$ and $M_2 > 0$ such that the following two inequalities hold:*

$$\|\mathrm{Retr}_X(\xi) - X\|_F \le M_1\|\xi\|_F, \quad \forall X \in \mathcal{M}, \xi \in \mathrm{T}_X\mathcal{M}, \tag{3.1}$$

$$\|\mathrm{Retr}_X(\xi) - (X + \xi)\|_F \le M_2\|\xi\|_F^2, \quad \forall X \in \mathcal{M}, \xi \in \mathrm{T}_X\mathcal{M}. \tag{3.2}$$

4 Proximal Gradient Method on the Stiefel Manifold

4.1 The ManPG Algorithm

For manifold optimization problems with a smooth objective, the Riemannian gradient method [1, 4, 61] has been one of the main methods of choice. A generic update formula of the Riemannian gradient method for solving

$$\min_X \ F(X), \ \text{s.t.}, \ X \in \mathcal{M}$$

is

$$X_{k+1} := \mathrm{Retr}_{X_k}(\alpha_k V_k),$$

where F is smooth, V_k is a descent direction of F in the tangent space $\mathrm{T}_{X_k}\mathcal{M}$, and α_k is a step size. Recently, Boumal et al. [19] established the sublinear rate of the Riemannian gradient method for returning a point X_k satisfying $\|\mathrm{grad}\, F(X_k)\|_F < \epsilon$. Liu et al. [56] proved that the Riemannian gradient method converges linearly for quadratic minimization over the Stiefel manifold. Other methods for solving manifold optimization problems with a smooth objective have also been studied in the literature, including the conjugate gradient methods [4, 2], trust region methods [4, 19], and Newton-type methods [4, 67].

We now develop our ManPG algorithm for solving (1.1). Since the objective function in (1.1) has a composite structure, a natural idea is to extend the proximal gradient method from the Euclidean setting to the manifold setting. The proximal gradient method for solving $\min_X F(X) := f(X) + h(X)$ in the Euclidean setting generates the iterates as follows:

$$X_{k+1} := \operatorname*{argmin}_Y f(X_k) + \langle \nabla f(X_k), Y - X_k\rangle + \frac{1}{2t}\|Y - X_k\|_{\mathrm{F}}^2 + h(Y). \tag{4.1}$$

In other words, one minimizes the quadratic model $Y \mapsto f(X_k) + \langle \nabla f(X_k), Y - X_k\rangle + \frac{1}{2t}\|Y - X_k\|_{\mathrm{F}}^2 + h(Y)$ of F at X_k in the k-th iteration, where $t > 0$ is a parameter that can be regarded as the stepsize. It is known that the quadratic model is an upper bound of F when $t \leq 1/L$, where L is the Lipschitz constant of ∇f. The subproblem (4.1) corresponds to the proximal mapping of h and the efficiency of the proximal gradient method relies on the assumption that (4.1) is easy to solve. For (1.1), in order to deal with the manifold constraint, we need to ensure that the descent direction lies in the tangent space. This motivates the following subproblem for finding the descent direction V_k in the k-th iteration:

$$\begin{aligned} V_k := \operatorname{argmin}_V \quad & \langle \operatorname{grad} f(X_k), V\rangle + \tfrac{1}{2t}\|V\|_F^2 + h(X_k+V) \\ \text{s.t.} \quad & V \in \mathrm{T}_{X_k}\mathcal{M}, \end{aligned} \tag{4.2}$$

where $t > 0$ is the stepsize. Here and also in the later discussions, we can interpret $X_k + V$ as the sum of X_k and V in the ambient Euclidean space $\mathbb{R}^{n\times r}$, as $\mathcal{M}$ is an embedded submanifold of $\mathbb{R}^{n\times r}$. Note that (4.2) is different from (4.1) in two places: (i) the Euclidean gradient ∇f is changed to the Riemannian gradient $\operatorname{grad} f$; (ii) the descent direction V_k is restricted to the tangent space. Following the definition of $\operatorname{grad} f$, we have

$$\langle \operatorname{grad} f(X_k), V\rangle = \langle \nabla f(X_k), V\rangle, \quad \forall V \in \mathrm{T}_{X_k}\mathcal{M},$$

which implies that (4.2) can be rewritten as

$$\begin{aligned} V_k := \operatorname{argmin}_V \quad & \langle \nabla f(X_k), V\rangle + \tfrac{1}{2t}\|V\|_F^2 + h(X_k+V) \\ \text{s.t.} \quad & V \in \mathrm{T}_{X_k}\mathcal{M}. \end{aligned} \tag{4.3}$$

As a result, we do not need to compute the Riemannian gradient $\operatorname{grad} f$. Rather, only the Euclidean gradient ∇f is needed. Note that without considering the constraint $V \in \mathrm{T}_{X_k}\mathcal{M}$, (4.3) computes a proximal gradient step. Therefore, (4.3) can be viewed as a proximal gradient step restricted to the tangent space $\mathrm{T}_{X_k}\mathcal{M}$. Since for an arbitrary stepsize $\alpha_k > 0$, $X_k + \alpha_k V_k$ does not necessarily lie on the manifold $\mathcal{M}$, we perform a retraction to bring it back to $\mathcal{M}$.

Our ManPG algorithm for solving (1.1) is described in Algorithm 1. Note that ManPG involves an Armijo line search procedure to determine the stepsize α. As we will show in Section 5, this backtracking line search procedure is well defined; i.e., it will terminate after finite number of steps.

Algorithm 1 Manifold Proximal Gradient Method (ManPG) for Solving (1.1)

```
Input: initial point $X_0 \in \mathcal{M}$, $\gamma \in (0,1)$, stepsize $t > 0$
for $k = 0, 1, \ldots$ do
    obtain $V_k$ by solving the subproblem (4.3)
    set $\alpha = 1$
    while $F(\mathrm{Retr}_{X_k}(\alpha V_k)) > F(X_k) - \frac{\alpha\|V_k\|_F^2}{2t}$ do
        $\alpha = \gamma\alpha$
    end while
    set $X_{k+1} = \mathrm{Retr}_{X_k}(\alpha V_k)$
end for
```

4.2 Regularized Semi-Smooth Newton Method for Subproblem (4.3)

The main computational effort of Algorithm 1 lies in solving the convex subproblem (4.3). We have conducted extensive numerical experiments and found that the semi-smooth Newton method (SSN) is very suitable for this purpose. The notion of semi-smoothness was originally introduced by Mifflin [60] for real-valued functions and later extended to vector-valued mappings by Qi and Sun [65]. A pioneering work on the SSN method was due to Solodov and Svaiter [71], in which the authors proposed a globally convergent Newton method by exploiting the structure of monotonicity and established a local superlinear convergence rate under the conditions that the generalized Jacobian is semi-smooth and non-singular at the global optimal solution. The convergence rate guarantee was later extended in [92] to the setting where the generalized Jacobian is not necessarily non-singular. Recently, the SSN method has received significant amount of attention due to its success in solving structured convex problems to a high accuracy. In particular, it has been successfully applied to solving SDP [91, 83], LASSO [54], nearest correlation matrix estimation [64], clustering [77], sparse inverse covariance selection [81], and composite convex minimization [80].

In the following we show how to apply the SSN method to solve the subproblem (4.3) with $\mathcal{M} = \mathrm{St}(n, r)$. The tangent space to $\mathcal{M} = \mathrm{St}(n, r)$ is given by

$$\mathrm{T}_X\mathcal{M} = \{V \mid V^\top X + X^\top V = 0\}.$$

For ease of notation, we define the linear operator $\mathcal{A}_k$ by $\mathcal{A}_k(V) := V^\top X_k + X_k^\top V$ and rewrite the subproblem (4.3) as

$$\begin{array}{rl} V_k := \mathrm{argmin}_V & \langle \nabla f(X_k), V\rangle + \frac{1}{2t}\|V\|_F^2 + h(X_k + V) \\ \text{s.t.} & \mathcal{A}_k(V) = 0. \end{array} \tag{4.4}$$

By associating a Lagrange multiplier Λ to the linear equality constraint, the Lagrangian function of (4.4) can be written as

$$\mathcal{L}(V; \Lambda) = \langle \nabla f(X_k), V\rangle + \frac{1}{2t}\|V\|_F^2 + h(X_k + V) - \langle \mathcal{A}_k(V), \Lambda\rangle,$$

and the KKT system of (4.4) is given by

$$0 \in \partial_V \mathcal{L}(V; \Lambda), \quad \mathcal{A}_k(V) = 0. \tag{4.5}$$

The first condition in (4.5) implies that V can be computed by

$$V(\Lambda) = \mathrm{prox}_{th}(B(\Lambda)) - X_k \quad \text{with} \quad B(\Lambda) = X_k - t(\nabla f(X_k) - \mathcal{A}_k^*(\Lambda)), \tag{4.6}$$

where $\mathcal{A}_k^*$ denotes the adjoint operator of $\mathcal{A}_k$. By substituting (4.6) into the second condition in (4.5), we see that Λ satisfies

$$E(\Lambda) \equiv \mathcal{A}_k(V(\Lambda)) = V(\Lambda)^\top X_k + X_k^\top V(\Lambda) = 0. \tag{4.7}$$

We will use the SSN method to solve (4.7). To do so, we need to first show that the operator E is monotone and Lipschitz continuous. For any $\Lambda_1, \Lambda_2 \in S^r$, we have

$$\begin{aligned} & \|E(\Lambda_1) - E(\Lambda_2)\|_\mathrm{F} \\ \leq & \|\mathcal{A}_k\|_{op}\|\mathrm{prox}_{th}(B(\Lambda_1)) - \mathrm{prox}_{th}(B(\Lambda_2))\|_\mathrm{F} \\ \leq & \|\mathcal{A}_k\|_{op}\|B(\Lambda_1) - B(\Lambda_2)\|_\mathrm{F} \\ \leq & t\|\mathcal{A}_k\|_{op}^2\|\Lambda_1 - \Lambda_2\|_\mathrm{F}, \end{aligned} \tag{4.8}$$

where the second inequality holds since the proximal mapping is non-expansive. Moreover,

$$\begin{aligned} & \langle E(\Lambda_1) - E(\Lambda_2), \Lambda_1 - \Lambda_2\rangle \\ = & \langle V(\Lambda_1) - V(\Lambda_2), \mathcal{A}_k^*(\Lambda_1 - \Lambda_2)\rangle \\ = & \frac{1}{t}\langle \mathrm{prox}_{th}(B(\Lambda_1)) - \mathrm{prox}_{th}(B(\Lambda_2)), B(\Lambda_1) - B(\Lambda_2)\rangle \\ \geq & \frac{1}{t}\|\mathrm{prox}_{th}(B(\Lambda_1)) - \mathrm{prox}_{th}(B(\Lambda_2))\|_\mathrm{F}^2 \\ \geq & \frac{1}{t\|\mathcal{A}_k\|_{op}^2}\|E(\Lambda_1) - E(\Lambda_2)\|_\mathrm{F}^2 \geq 0, \end{aligned}$$

where the first inequality holds since the proximal mapping is firmly non-expansive and the second inequality is due to (4.8). In particular, we see that E is actually $1/(t\|\mathcal{A}_k\|_{op}^2)$-coercive. Therefore, E is indeed monotone and Lipschitz continues, and we can apply the SSN method to find a zero of E. In order to apply the SSN method, we need to compute the generalized Jacobian of E.[1] Towards that end, observe that the vectorization of $E(\Lambda)$ can be represented by

$$\begin{aligned} \mathrm{vec}(E(\Lambda)) &= (X_k^\top \otimes I_p)K_{nr}\mathrm{vec}(V(\Lambda)) + (I_r \otimes X_k^\top)\mathrm{vec}(V(\Lambda)) \\ &= (K_{rr} + I_{r^2})(I_p \otimes X_k^\top)\left[\mathrm{prox}_{th(\cdot)}(\mathrm{vec}(X_k - t\nabla f(X_k)) + 2t(I_r \otimes X_k)\mathrm{vec}(\Lambda)) - \mathrm{vec}(X_k)\right], \end{aligned}$$

[1] See Appendix A for a brief discussion of the semi-smoothness of operators related to the proximal mapping.

where K_{nr} and K_{rr} denote the commutation matrices. We define the matrix

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

where $\mathcal{J}(y)$ is the generalized Jacobian of $\mathrm{prox}_{th}(y)$. From [41, Example 2.5], we know that $\mathcal{G}(\mathrm{vec}(\Lambda))\xi = \partial \mathrm{vec}(E(\mathrm{vec}(\Lambda))\xi,\ \forall \xi \in \mathbb{R}^{r^2}$. Thus, $\mathcal{G}(\mathrm{vec}(\Lambda))$ can serve as a representation of $\partial\mathrm{vec}(E(\mathrm{vec}(\Lambda)))$. Note that since Λ is a symmetric matrix, we only need to focus on the lower triangular part of Λ. It is known that there exists a unique $r^2 \times \frac{1}{2}r(r+1)$ matrix U_r, called the duplication matrix [59, Ch 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 satisfies $U_r^+\mathrm{vec}(\Lambda) = \overline{\mathrm{vec}}(\Lambda)$. Note that both U_r and U_r^+ have only r^2 nonzero elements. As a result, we can represent the generalized Jacobian of $\overline{\mathrm{vec}}(E(U_r\overline{\mathrm{vec}}(\Lambda)))$ by

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

where we use the identity $K_{rr} + I_{r^2} = 2U_rU_r^+$. It should be pointed out that $\mathcal{G}(\overline{\mathrm{vec}}(\Lambda))$ can be singular. Therefore, the vanilla SSN method cannot be applied directly and we need to resort to a regularized SSN method proposed in [71] and further studied in [92, 80]. It is known that the global convergence of the regularized SSN method is guaranteed if any element in $\mathcal{G}(\overline{\mathrm{vec}}(\Lambda))$ is positive semidefinite [80], which is the case here because it can be shown that $\mathcal{G}(\overline{\mathrm{vec}}(\Lambda)) + \mathcal{G}(\overline{\mathrm{vec}}(\Lambda))^\top$ is positive semidefinite. We find that the adaptive regularized SSN (ASSN) method proposed in [80] is very suitable for solving (4.7). The ASSN method first computes the Newton direction d_k by solving

$$(\mathcal{G}(\overline{\mathrm{vec}}(\Lambda_k)) + \eta I)d = -\overline{\mathrm{vec}}(E(\Lambda_k)), \tag{4.9}$$

where $\eta > 0$ is a regularization parameter. If the matrix size is large, then (4.9) can be solved inexactly by the conjugate gradient method. The authors then designed a strategy to decide whether to accept this d_k or not. Roughly speaking, if there is a sufficient decrease from $\|E(\Lambda_k)\|_2$ to $\|E(\Lambda_{k+1})\|_2$, then we accept d^k and set

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

Otherwise, a safeguard step is taken. For more details on the ASSN method, we refer the reader to [80].

5 Global Convergence and Iteration Complexity

In this section, we analyze the convergence and iteration complexity of our ManPG algorithm (Algorithm 1) for solving (1.1). Our convergence analysis consists of three steps. First, in Lemma 5.1 we show that V_k in (4.3) is a descent direction for the objective function in (4.3). Second, in Lemma 5.2 we show that V_k is also a descent direction for the objective function in (1.1) after applying a retraction to it; i.e., there is a sufficient decrease from $F(X_k)$ to $F(\mathrm{Retr}_{X_k}(\alpha V_k))$. This is motivated by a similar result in Boumal et al. [19], which states that the pullback function $\hat{F}(V) := F(\mathrm{Retr}_X(V))$ satisfies certain Lipschitz-type property. Therefore, the results here can be seen as an extension of the ones for smooth problems in [19] to the nonsmooth problem (1.1). Third, we establish the global convergence of ManPG in Theorem 5.5.

Now, let us begin our analysis. The first observation is that the objective function in (4.3) is strongly convex, which implies that the subproblem (4.3) is also strongly convex. Recall that a function g is said to be α-strongly convex[2] on $\mathbb{R}^{n\times r}$ if

$$g(Y) \geq g(X) + \langle \partial g(X), Y - X\rangle + \frac{\alpha}{2}\|Y - X\|_\mathrm{F}^2, \quad \forall X, Y \in \mathbb{R}^{n\times r}. \tag{5.1}$$

The following lemma shows that V_k obtained by solving (4.3) is indeed a descent direction in the tangent space to $\mathcal{M}$ at X_k:

[2] A function $g : \mathbb{R}^n \to \mathbb{R}$ is called $\alpha-$strongly convex [66, Definition 12.58] if there exists $\alpha > 0$ such that $g((1-t)x + ty) \leq (1-t)g(x) + tg(y) - \frac{1}{2}\alpha t(1-t)\|x - y\|^2$, for all x, y when $t \in (0,1)$. It is equivalent to that $g - \frac{1}{2}\alpha\|\cdot\|^2$ is convex [66, Exercise 12.59]. Thus, we have the definition in (5.1).

Lemma 5.1. *Given the iterate X_k, let*

$$g(V) := \langle \nabla f(X_k), V\rangle + \frac{1}{2t}\|V\|_F^2 + h(X_k + V) \tag{5.2}$$

denote the objective function in (4.3). Then, the following holds for any $\alpha \in [0,1]$:

$$g(\alpha V_k) - g(0) \le \frac{(\alpha-2)\alpha}{2t}\|V_k\|_F^2. \tag{5.3}$$

Proof. Since g is $(1/t)$-strongly convex, we have

$$g(\hat{V}) \ge g(V) + \langle \partial g(V), \hat{V} - V\rangle + \frac{1}{2t}\|\hat{V} - V\|_{\mathrm{F}}^2, \quad \forall V, \hat{V} \in \mathbb{R}^{n\times r}. \tag{5.4}$$

In particular, if $V, \hat{V}$ are feasible for (4.3) (i.e., $V, \hat{V} \in \mathrm{T}_{X_k}\mathcal{M}$), then

$$\langle \partial g(V), \hat{V} - V\rangle = \langle \mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}} \partial g(V), \hat{V} - V\rangle.$$

From the optimality condition of (4.3), we have $0 \in \mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}} \partial g(V_k)$. Letting $V = V_k$ and $\hat{V} = 0$ in (5.4) yields

$$g(0) \ge g(V_k) + \frac{1}{2t}\|V_k\|_{\mathrm{F}}^2,$$

which implies that

$$h(X_k) \ge \langle \nabla f(X_k), V_k\rangle + \frac{1}{2t}\|V_k\|_{\mathrm{F}}^2 + h(X_k + V_k) + \frac{1}{2t}\|V_k\|_{\mathrm{F}}^2.$$

Moreover, the convexity of h yields

$$h(X_k + \alpha V_k) - h(X_k) = h(\alpha(X_k + V_k) + (1-\alpha)X_k) - h(X_k) \le \alpha\left(h(X_k + V_k) - h(X_k)\right).$$

Upon combining the above two inequalities, we obtain

$$\begin{aligned} g(\alpha V_k) - g(0) &= \langle \nabla f(X_k), \alpha V_k\rangle + \frac{\|\alpha V_k\|_{\mathrm{F}}^2}{2t} + h(X_k + \alpha V_k) - h(X_k) \\ &\le \alpha\left(\langle \nabla f(X_k), V_k\rangle + \alpha\frac{\|V_k\|_{\mathrm{F}}^2}{2t} + h(X_k + V_k) - h(X_k)\right) \\ &\le \frac{\alpha^2 - 2\alpha}{2t}\|V_k\|_{\mathrm{F}}^2, \end{aligned}$$

as desired. □

The following lemma shows that $\{F(X_k)\}$ is monotonically decreasing, where $\{X_k\}$ is generated by Algorithm 1.

Lemma 5.2. *For any $t > 0$, there exists a constant $\bar{\alpha} > 0$ such that for any $0 < \alpha \le \min\{1, \bar{\alpha}\}$, the condition in Step 5 of Algorithm 1 is satisfied, and the sequence $\{X_k\}$ generated by Algorithm 1 satisfies*

$$F(X_{k+1}) - F(X_k) \le -\frac{\alpha}{2t}\|V_k\|_F^2.$$

Proof. Let $X_k^+ = X_k + \alpha V_k$. Following Boumal et al. [19], we first show that $f(\mathrm{Retr}_{X_k}(V))$ satisfies certain Lipschitz smooth condition. By the L-Lipschitz continuity of ∇f, for any $\alpha > 0$, we have

$$\begin{aligned} f(\mathrm{Retr}_{X_k}(\alpha V_k)) - f(X_k) &\le \langle \nabla f(X_k), \mathrm{Retr}_{X_k}(\alpha V_k) - X_k\rangle + \frac{L}{2}\|\mathrm{Retr}_{X_k}(\alpha V_k) - X_k\|_{\mathrm{F}}^2 \\ &= \langle \nabla f(X_k), \mathrm{Retr}_{X_k}(\alpha V_k) - X_k^+ + X_k^+ - X_k\rangle + \frac{L}{2}\|\mathrm{Retr}_{X_k}(\alpha V_k) - X_k\|_{\mathrm{F}}^2 \qquad (5.5) \\ &\le M_2\|\nabla f(X_k)\|_{\mathrm{F}}\|\alpha V_k\|_{\mathrm{F}}^2 + \alpha\langle \nabla f(X_k), V_k\rangle + \frac{LM_1^2}{2}\|\alpha V_k\|_{\mathrm{F}}^2, \end{aligned}$$

where the last inequality follows from (3.1) and (3.2). Since ∇f is continuous on the compact manifold $\mathcal{M}$, there exists a constant $G > 0$ such that $\|\nabla f(X)\|_{\mathrm{F}} \le G$ for all $X \in \mathcal{M}$. It then follows from (5.5) that

$$f(\mathrm{Retr}_{X_k}(\alpha V_k)) - f(X_k) \le \alpha\langle \nabla f(X_k), V_k\rangle + c_0\alpha^2\|V_k\|_{\mathrm{F}}^2, \tag{5.6}$$

where $c_0 = M_2 G + L M_1^2/2$. This implies that

$$\begin{aligned} F(\mathrm{Retr}_{X_k}(\alpha V_k)) - F(X_k) &\overset{(5.6)}{\le} \alpha\langle \nabla f(X_k), V_k\rangle + c_0\alpha^2\|V_k\|_{\mathrm{F}}^2 + h(\mathrm{Retr}_{X_k}(\alpha V_k)) - h(X_k^+) + h(X_k^+) - h(X_k) \\ &\le \alpha\langle \nabla f(X_k), V_k\rangle + c_0\alpha^2\|V_k\|_F^2 + L_h\|\mathrm{Retr}_{X_k}(\alpha V_k) - X_k^+\|_{\mathrm{F}} + h(X_k^+) - h(X_k) \\ &\overset{(3.2)}{\le} (c_0 + L_h M_2)\|\alpha V_k\|_F^2 + g(\alpha V_k) - \frac{1}{2t}\|\alpha V_k\|_{\mathrm{F}}^2 - g(0) \\ &\overset{(5.3)}{\le} \left(c_0 + L_h M_2 - \frac{1}{\alpha t}\right)\|\alpha V_k\|_{\mathrm{F}}^2, \end{aligned}$$

where g is defined in (5.2) and the second inequality follows from the Lipschitz continuity of h. Upon setting $\bar{\alpha} = 1/(2(c_0 + L_h M_2)t)$, we conclude that for any $0 < \alpha \le \min\{\bar{\alpha}, 1\}$,

$$F(\mathrm{Retr}_{X_k}(\alpha V_k)) - F(X_k) \le -\frac{1}{2\alpha t}\|\alpha V_k\|_{\mathrm{F}}^2 = -\frac{\alpha}{2t}\|V_k\|_{\mathrm{F}}^2.$$

This completes the proof. □

The following lemma shows that if one cannot make any progress by solving (4.3) (i.e., $V_k = 0$), then a stationary point is found.

Lemma 5.3. *If $V_k = 0$, then X_k is a stationary point of problem (1.1).*

Proof. By Theorem 4.1 in [84], the optimality conditions of the subproblem (4.2) are given by

$$0 \in \frac{1}{t}V_k + \mathrm{grad}\, f(X_k) + \mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}}\partial h(X_k + V_k), \qquad V_k \in \mathrm{T}_{X_k}\mathcal{M}.$$

If $V_k = 0$, then $0 \in \mathrm{grad}\, f(X_k) + \mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}}\partial h(X_k)$, which is exactly the first-order necessary condition of problem (1.1) since $X_k \in \mathcal{M}$. □

From Lemma 5.3, we know that $V_k = 0$ implies the stationarity of X_k with respect to (1.1). This motivates the following definition of an ϵ-stationary point of (1.1):

Definition 5.4. *We say that $X_k \in \mathcal{M}$ is an ϵ-stationary point of (1.1) if the solution V_k to (4.4) with $t = 1/L$ satisfies $\|V_k\|_F \le \epsilon/L$.*

We use $\|V_k\|_{\mathrm{F}} \le \epsilon/L$ as the stopping criterion of Algorithm 1 with $t = 1/L$. From Lemma 5.2, we obtain the following result which is similar to the one in [19, Theorem 2] for manifold optimization with smooth objectives.

Theorem 5.5. *Under Assumption 1.1, every limit point of the sequence $\{X_k\}$ generated by Algorithm 1 is a stationary point of problem (1.1). Moreover, Algorithm 1 with $t = 1/L$ will return an ϵ-stationary point of (1.1) in at most $\lceil 2L(F(X_0) - F^*)/(\gamma\bar{\alpha}\epsilon^2)\rceil$ iterations, where $\bar{\alpha}$ is defined in Lemma 5.2 and F^* is the optimal value of (1.1).*

Proof. Since F is bounded below on $\mathcal{M}$, by Lemma 5.2, we have

$$\lim_{k\to\infty} \|V_k\|_F^2 = 0.$$

Combining with Lemma 5.3, it follows that every limit point of $\{X_k\}$ is a stationary point of (1.1). Moreover, since $\mathcal{M}$ is compact, the sequence $\{X_k\}$ has at least one limit point. Furthermore, suppose that Algorithm 1 with $t = 1/L$ does not terminate after K iterations; i.e., $\|V_k\|_{\mathrm{F}} > \epsilon/L$ for all $k = 0, 1, \ldots, K-1$. Let α_k

be the stepsize in the k-th iteration; i.e., $X_{k+1} = \mathrm{Retr}_{X_k}(\alpha_k V_k)$. From Lemma 5.2, we know that $\alpha_k \geq \gamma\bar{\alpha}$. Thus, we have

$$F(X_0) - F^* \geq F(X_0) - F(X_K) \geq \frac{t}{2}\sum_{k=0}^{K-1} \alpha_k \|V_k/t\|_{\mathrm{F}}^2 > \frac{t\epsilon^2}{2}\sum_{k=0}^{K-1}\alpha_k \geq \frac{tK\epsilon^2}{2}\gamma\bar{\alpha}.$$

Therefore, Algorithm 1 finds an ϵ-stationary point in at most $\lceil 2L(F(X_0) - F^*)/(\gamma\bar{\alpha}\epsilon^2)\rceil$ iterations. □

Remark 5.6. *When the objective function F in (1.1) is smooth (i.e., the nonsmooth function h vanishes), the iteration complexity in Theorem 5.5 matches the result given by Boumal et al. in [19]. Zhang and Sra [88] analyzed the iteration complexity of some first-order methods, but they assumed that the objectives are geodesically convex. Such an assumption is rather restrictive, as it is known that every smooth function that is geodesically convex on a compact Riemannian manifold is constant [14]. Bento et al. [13] also established some iteration complexity results for gradient, subgradient, and proximal point methods. However, their results for gradient and subgradient methods require the objective function to be convex and the manifold to be of nonnegative curvature, while those for proximal point methods only apply to convex objective functions over the Hadamard manifold.*

6 Numerical Experiments

In this section, we apply our ManPG algorithm[3] (Algorithm 1) to solve the sparse PCA (1.2) and compressed modes (CM) (1.3) problems. We compare ManPG with two existing methods SOC [53] and PAMAL [22]. For both problems, we set the parameter $\gamma = 0.5$ and use the polar decomposition as the retraction mapping in ManPG. The latter is because it is found that the MATLAB implementation of QR factorization is slower than the polar decomposition; see [5]. Moreover, we implement a more practical version of ManPG, named ManPG-Ada and described in Algorithm 2, that incorporates a few tricks including adaptively updating the stepsize t. We set the parameters $\gamma = 0.5$ and $\tau = 1.01$ in ManPG-Ada. All the codes used in this section were written in MATLAB and run on a standard PC with 3.70 GHz I7 Intel microprocessor and 16GB of memory.

6.1 A More Practical ManPG: ManPG-Ada

In this subsection, we introduce some tricks used to further improve the performance of ManPG in practice. First, a warm-start strategy is adopted for SSN; i.e., the initial point Λ_0 in SSN is set as the solution of the previous subproblem. For the ASSN algorithm, we always take the semi-smooth Newton step as suggested by [80]. Second, we adaptively update t in ManPG. When t is large, we may need smaller total number of iterations to reach an ϵ-stationary point. However, it increases the number of line search steps and the SSN steps. For sparse PCA and CM problems, we found that setting $t = 1/L$ leads to fewer number of line search steps. We can then increase t slightly if no line search step was needed in the previous iteration. This new version of ManPG, named ManPG-Ada, is described in Algorithm 2. We also applied ManPG-Ada to solve sparse PCA and CM problems and compared its performance with ManPG, SOC, and PAMAL.

6.2 Numerical Results on CM

For the CM problem (1.3), both SOC [53] and PAMAL [22] rewrite the problem as

$$\begin{array}{ll} \min_{X,Q,P\in\mathbb{R}^{n\times r}} & \mathrm{Tr}(P^\top HP) + \mu\|Q\|_1 \\ \mathrm{s.t.} & Q = P, X = P, X^\top X = I_r. \end{array} \tag{6.1}$$

[3] Our MATLAB code is available at https://github.com/chenshixiang/ManPG.

Algorithm 2 ManPG-Ada for Solving (1.1)

```
Input: initial point $X_0 \in \mathcal{M}$, $\gamma \in (0,1)$, $\tau > 1$ and Lipschitz constant $L$
set $t = 1/L$
for $k = 0, 1, \ldots$ do
    obtain $V_k$ by solving the subproblem (4.3)
    set $\alpha = 1$ and linesearchflag $= 0$
    while $F(\mathrm{Retr}_{X_k}(\alpha V_k)) > F(X_k) - \frac{\alpha \|V_k\|_{\mathrm{F}}^2}{2t}$ do
        $\alpha = \gamma\alpha$
        linesearchflag $= 1$
    end while
    set $X_{k+1} = \mathrm{Retr}_{X_k}(\alpha V_k)$
    if linesearchflag $= 1$ then
        $t = \tau t$
    else
        $t = \max\{1/L, t/\tau\}$
    end if
end for
```

SOC employs a three-block ADMM to solve (6.1), which updates the iterates as follows:

$$\begin{array}{ll} P_{k+1} & := \mathrm{argmin}_P \ \mathrm{Tr}(P^\top H P) + \frac{\beta}{2}\|P - Q_k + \Lambda_k\|_F^2 + \frac{\beta}{2}\|P - X_k + \Gamma_k\|_F^2, \\ Q_{k+1} & := \mathrm{argmin}_Q \ \mu\|Q\|_1 + \frac{\beta}{2}\|P_{k+1} - Q + \Lambda_k\|_F^2, \\ X_{k+1} & := \mathrm{argmin}_X \ \frac{\beta}{2}\|P_{k+1} - X + \Gamma_k\|_F^2, \ \text{s.t.}, X^\top X = I_r, \\ \Lambda_{k+1} & := \Lambda_k + P_{k+1} - Q_{k+1}, \\ \Gamma_{k+1} & := \Gamma_k + P_{k+1} - X_{k+1}. \end{array} \tag{6.2}$$

PAMAL uses an inexact augmented Lagrangian method to solve (6.1) with the augmented Lagrangian function being minimized by the proximal alternating minimization algorithm proposed in [8]. Both SOC and PAMAL need to solve a linear system $(H + \beta I)X = B$, where B is a given matrix.

In our numerical experiments, we tested the same problems as in [62] and [22]. In particular, we consider the time-independent Schrödinger equation

$$\hat{H}\phi(x) = \lambda\phi(x), x \in \Omega,$$

where $\hat{H} = -\frac{1}{2}\Delta$ denotes the Hamiltonian, Δ denotes the Laplacian operator, and H is a symmetric matrix formed by discretizing the Hamiltonian $\hat{H}$. We focus on the 1D free-electron (FE) model. The FE model describes the behavior of valence electron in a crystal structure of a metallic solid and has $\hat{H} = -\frac{1}{2}\partial_x^2$. We consider the system on a domain $\Omega = [0, 50]$ with periodic boundary condition and discretize the domain with n equally spaced nodes. The stepsize t in Algorithm 1 was set to $1/(2\lambda_{\max}(\hat{H}))$, where $\lambda_{\max}(\hat{H})$ denotes the largest eigenvalue of $\hat{H}$ and is given by $2n^2/50^2$ in this case.

Since the matrix H is circulant, we used FFT to solve the linear systems in SOC and PAMAL, which is more efficient than directly inverting the matrices. We terminated ManPG when $\|V_k/t\|_{\mathrm{F}}^2 \leq \epsilon := 10^{-8}nr$ or the maximum iteration number 30000 was reached. For the inner iteration of ManPG (i.e., using SSN to solve (4.3)), we terminated it when $\|E(\Lambda)\|_{\mathrm{F}}^2 \leq \max\{10^{-13}, \min\{10^{-11}, 10^{-3}t^2\epsilon\}\}$ or the maximum iteration number 100 was reached. In all the tests of the CM problem, we ran ManPG first and let F_M denote the returned objective value. We then ran SOC and PAMAL and terminated them when $F(X_k) \leq F_M + 10^{-7}$ and

$$\frac{\|Q_k - P_k\|_F}{\max\{1, \|Q_k\|_F, \|P_k\|_F\}} + \frac{\|X_k - P_k\|_F}{\max\{1, \|X_k\|_F, \|P_k\|_F\}} \leq 10^{-4}. \tag{6.3}$$

Note that (6.3) measures the constraint violation of the reformulation (6.1). If (6.3) was not satisfied in 30000 iterations, then we terminated SOC and PAMAL. We also ran ManPG-Ada (Algorithm 2) and terminated it if $F(X_k) \leq F_M + 10^{-7}$.

In our experiments, we found that SOC and PAMAL are very sensitive to the choice of parameters. The default setting of the parameters of SOC and PAMAL suggested in [62] and [22] usually cannot achieve our desired accuracy. Unfortunately, there is no systematic study on how to tune these parameters. We spent a significant amount of effort on tuning these parameters, and the ones we used are given as follows. For SOC (6.2), we set the penalty parameter $\beta = nr\mu/25 + 1$. For PAMAL, we found that the setting of the parameters given on page B587 of [22] did not work well for the problems we tested. Instead, we found that the following settings of these parameters worked best and they were thus adopted in our tests: $\tau = 0.99$, $\gamma = 1.001$, $\rho^1 = 2\,|\lambda_{\min}(H)| + r/10 + 2, \overline{\Lambda}_{p,\min} = -100, \overline{\Lambda}_{p,\max} = 100, \Lambda_p^1 = 0_{nr}, p = 1, 2$, and $\epsilon^k = (0.995)^k$, $k \in \mathbb{N}$. For the meaning of these parameters, we refer the reader to page B587 of [22]. We used the same parameters of PAM in PAMAL as recommended by [22]. For different settings of (n, r, μ), we ran the four algorithms with 50 instances whose initial points were obtained by projecting randomly generated points onto $\mathrm{St}(n, r)$. Since problem (1.3) is nonconvex, it is possible that ManPG, ManPG-Ada, SOC and PAMAL return different solutions from random initializations. To increase the chance that all four solvers found the same solution, we ran the Riemannian subgradient method for 500 iterations and used the resulting iterate as the initial point. The Riemannian subgradient method is described as follows:

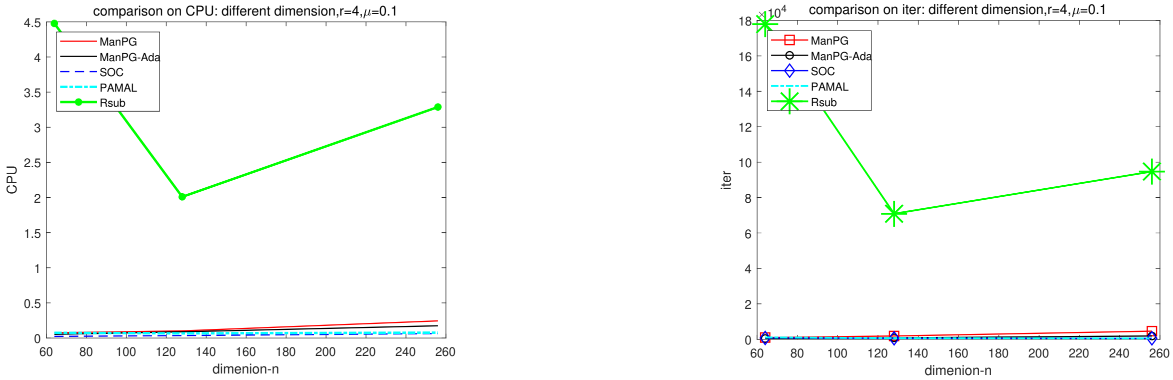
$$\begin{aligned} \hat{\partial} F(X_k) &:= \mathrm{Proj}_{\mathrm{T}_{X_k}\mathrm{St}(n,r)}(2HX_k + \mu \mathrm{sign}(X_k)), \\ X_{k+1} &:= \mathrm{Retr}_{X_k}\left(-\frac{1}{k^{3/4}}\hat{\partial} F(X_k)\right), \end{aligned} \tag{6.4}$$

where $\mathrm{sign}(\cdot)$ denotes the element-wise sign function. Moreover, we tried to run the Riemannian subgradient method (6.4) until it solved the CM problem. However, this method is extremely slow and we only report one case in Figure 1. We report the averaged CPU time, iteration number, and sparsity in Figures 1 to 4, where sparsity is the percentage of zeros and when computing sparsity, X is truncated by zeroing out its entries whose magnitude is smaller than 10^{-5}. For SOC and PAMAL, we only took into account the solutions that were close to the one generated by ManPG. Here the closeness of the solutions is measured by the distance between their column spaces. More specifically, let X_M, X_S, and X_P denote the solutions generated by ManPG, SOC, and PAMAL, respectively. Then, their distances are computed by $\mathrm{dist}(X_M, X_S) = \|X_M X_M^\top - X_S X_S^\top\|_{\mathrm{F}}$ and $\mathrm{dist}(X_M, X_P) = \|X_M X_M^\top - X_P X_P^\top\|_{\mathrm{F}}$. We only counted the results if $\mathrm{dist}^2(X_M, X_S) \leq 0.1$ and $\mathrm{dist}^2(X_M, X_P) \leq 0.1$.

In Figure 1, we report the results of Riemannian subgradient method with respect to different n's. We terminated the Riemannian subgradient method (6.4) if $F(X_k) < F_M + 10^{-3}$. We see that this accuracy tolerance 10^{-3} is too large to yield a good solution with reasonable sparsity level, yet it is already very time consuming. As a result, we do not report more results on the Riemannian subgradient method. In Figures 2, 3, and 4, we see that the solutions returned by ManPG and ManPG-Ada have better sparsity than SOC and PAMAL. We also see that ManPG-Ada outperforms ManPG in terms of CPU time and iteration number. In Figure 2, the iteration number of ManPG increases with the dimension n, because the Lipschitz constant $L = 2\lambda_{\max}(H) = 4n^2/50^2$ increases quadratically, which is consistent with our complexity result. In Figure 3, we see that the CPU times of ManPG and ManPG-Ada are comparable to those of SOC and PAMAL when r is small, but are slightly more when r gets large. In Figure 4, we see that the performance of the algorithms is also affected by μ. In terms of CPU time, ManPG and ManPG-Ada are comparable to SOC and PAMAL when μ gets large.

The first five CMs of the 1D FE model computed by the ManPG-Ada, SOC and PAMAL methods are shown in Figure 5. We found that the CMs generated by ManPG and ManPG-Ada were the same, so we only report the results of ManPG-Ada. We flip the CMs if necessary so that most values on the support of the CMs are positive, as sign ambiguities do not affect the minimal values of the objective function in (1.3). It can be seen that the CMs obtained from the three methods are compactly supported functions, and their localization degree is almost the same. We next examine the approximation behavior of the unitary transformations derived from the CMs to the eigenmodes of the Schrödinger operator. The approximation accuracy is measured by comparing the first r eigenvalues $(\sigma_1, \ldots, \sigma_r)$ of the matrix $X^\top \hat{H} X$ with the first r eigenvalues $(\lambda_1, \ldots, \lambda_r)$ of the corresponding Schrödinger operator $\hat{H}$. Figure 6 reports the results for different values of r. We can see the approximation errors of the ManPG-Ada, SOC and PAMAL are similar, and that $(\sigma_1, \ldots, \sigma_r)$ converges to $(\lambda_1, \ldots, \lambda_r)$ as r increases.

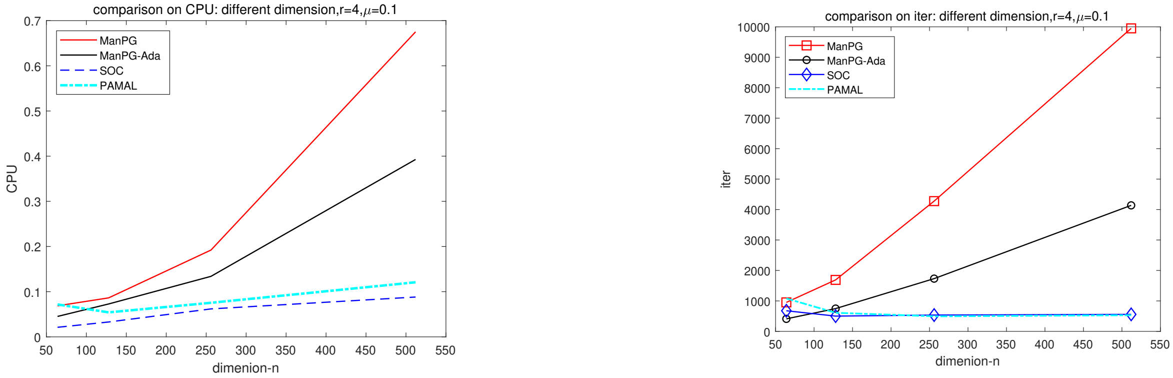
We also report the total number of line search steps and the averaged iteration number of SSN in ManPG and ManPG-Ada in Table 1. We see that ManPG-Ada needs more line search steps and SSN iterations, but as we show in Figures 2, 3, and 4, ManPG-Ada is faster than ManPG in terms of CPU time. This is mainly because the computational costs of retraction and SSN steps in this problem are both nearly the same as computing the gradient. In the last two columns of Table 1, '#s|d' denotes the number of instances for which SOC and PAMAL generate same, different solutions as ManPG with the closeness measurement discussed above; '# f' denotes the number of instances that SOC and PAMAL fail to converge. We see that for the tested instances of the CM problem, all algorithms converged thanks to the parameters that we chose, although sometimes the solutions generated by PAMAL are different from those generated by ManPG and SOC.



(a) CPU

(b) Iteration

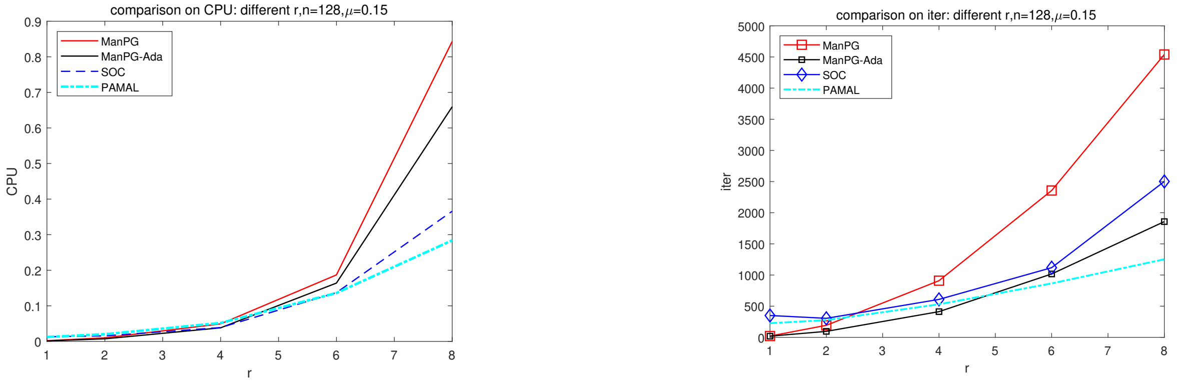
Figure 1: Comparison on CM problem (1.3), different $n = \{64, 128, 256\}$ with $r = 4$ and $\mu = 0.1$.



(a) CPU

(b) Iteration

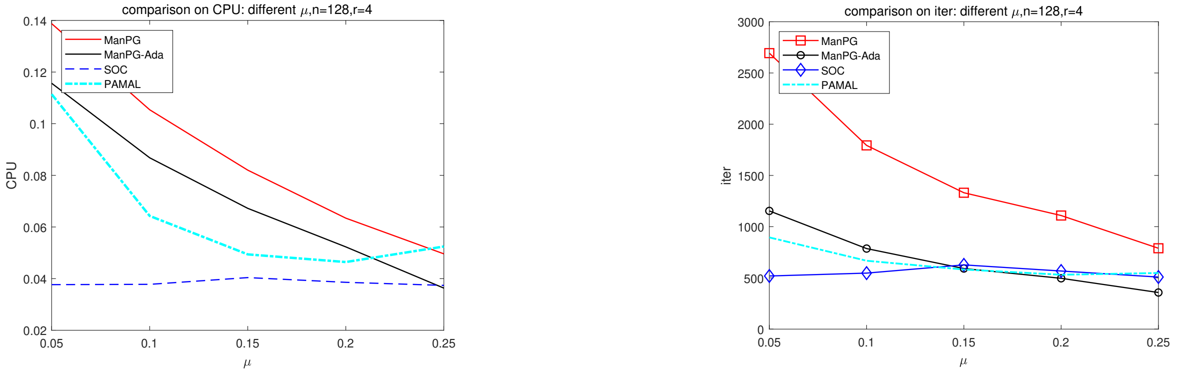
Figure 2: Comparison on CM problem (1.3), different $n = \{64, 128, 256, 512\}$ with $r = 4$ and $\mu = 0.1$.



(a) CPU

(b) Iteration

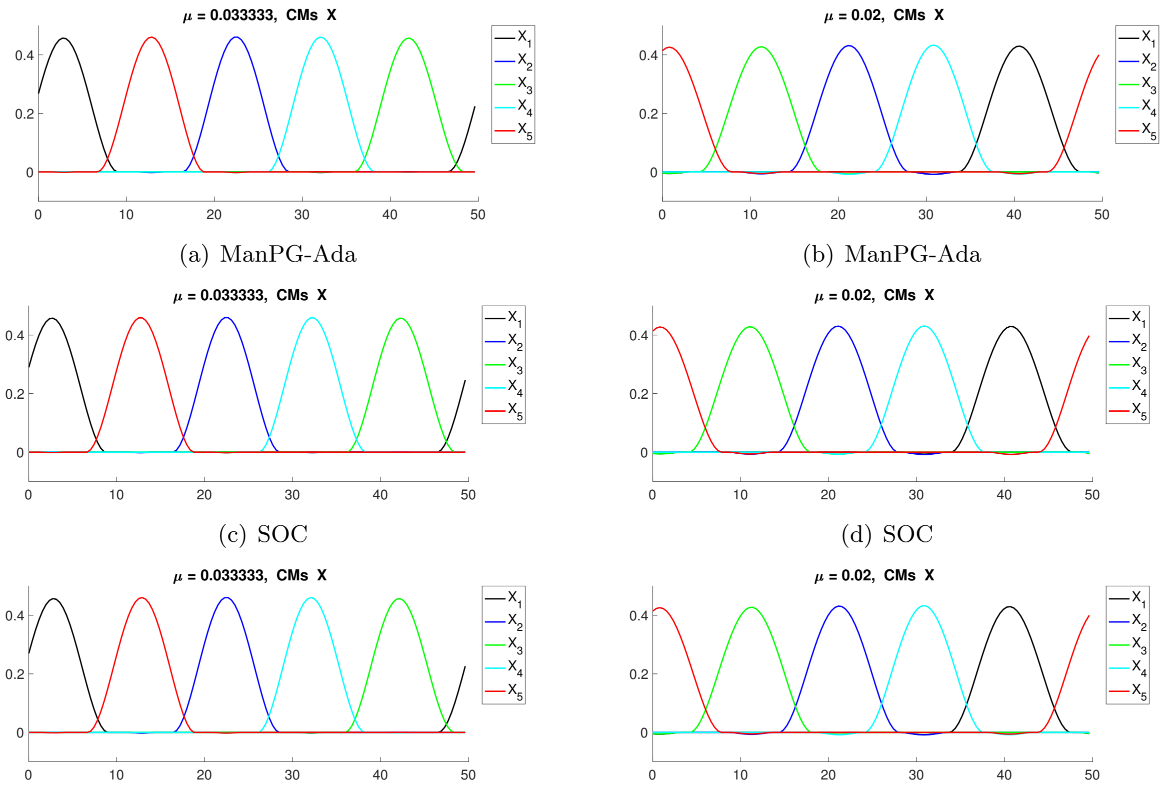
Figure 3: Comparison on CM problem (1.3), different $r = \{1, 2, 4, 6, 8\}$ with $n = 128$ and $\mu = 0.15$.



(a) CPU

(b) Iteration

Figure 4: Comparison on CM problem (1.3), different $\mu = \{0.05, 0.1, 0.15, 0.2, 0.25\}$ with $n = 128$ and $r = 4$.



(a) ManPG-Ada (b) ManPG-Ada

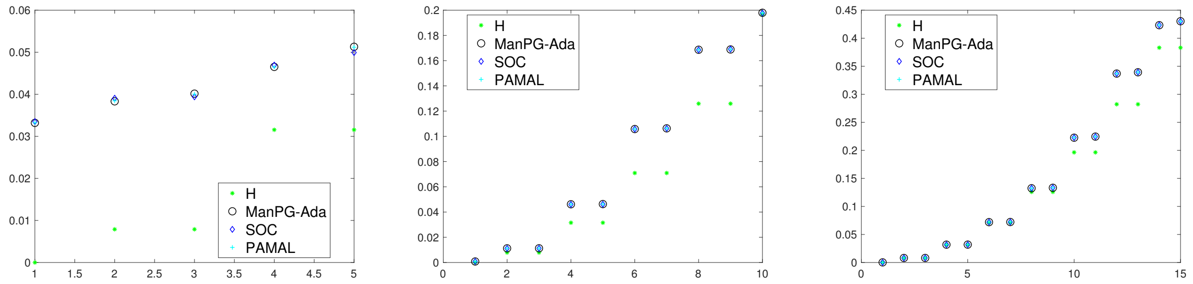
(c) SOC (d) SOC

(e) PAMAL (f) PAMAL

Figure 5: Comparison of the first five modes obtained for the 1D FE model with different values of μ. Left column: $\mu = 1/30$; Right column: $\mu = 1/50$.

Table 1: Number of line search steps and averaged SSN iterations for different (n, r, μ).

	ManPG		ManPG-Ada		SOC	PAMAL
	# line search	SSN iter	# line search	SSN iter	# s| d| f	# s| d| f
n			$r = 4, \mu = 0.1$			
64	85.94	1.0005	165.98	1.3307	50|0|0	48|2|0
128	70.5	0.64414	540.76	1.2237	50|0|0	50|0|0
256	84.06	0.39686	1191.5	0.60652	50|0|0	50|0|0
512	55.1	0.16622	2720.6	0.2417	50|0|0	49|1|0
μ			$n = 128, r = 4$			
0.05	49.2	0.30933	695.6	0.83637	50|0|0	50|0|0
0.1	74.38	0.54915	572.42	1.1514	50|0|0	50|0|0
0.15	102.62	0.82093	439.6	1.2899	50|0|0	50|0|0
0.2	82.52	0.81565	350.86	1.2114	50|0|0	50|0|0
0.25	93.3	0.57232	209.12	1.0122	50|0|0	48|2|0
r			$n = 128, \mu = 0.15$			
1	0	0.8971	0	0.98694	50|0|0	50|0|0
2	3.48	1.0001	61.02	1.1135	50|0|0	50|0|0
4	86.92	0.91814	311	1.2812	50|0|0	50|0|0
6	169.8	0.60206	719.42	1.5195	50|0|0	49|1|0
8	216.54	1.2011	1198.8	2.8667	50|0|0	42|8|0



(a) $r = 5$

(b) $r = 10$

(c) $r = 15$

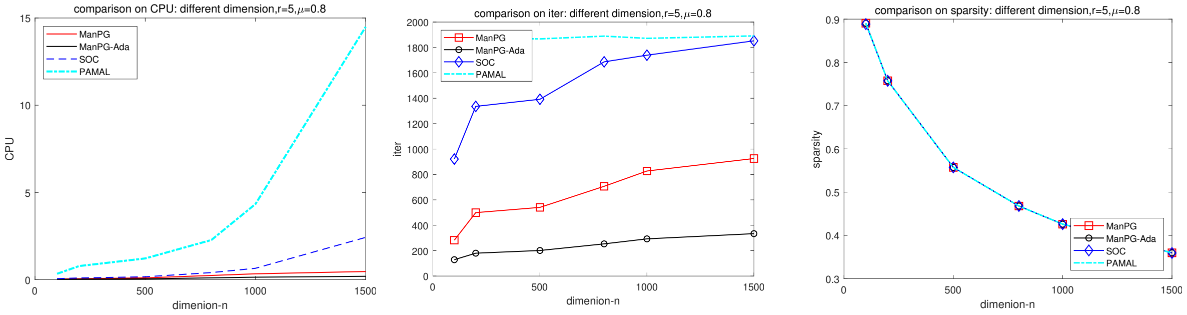
Figure 6: Comparisons of the first r eigenvalues of the 1D free electron model. $*$: The first r eigenvalues of the matrix $\hat{H}$. $\circ$: The first r eigenvalues of the matrix $X^\top \hat{H} X$, where X is the solution obtained by ManPG-Ada. $\diamond$: The first r eigenvalues of the matrix $X^\top \hat{H} X$, where X is the solution obtained by SOC. $+$: The first r eigenvalues of the matrix $X^\top \hat{H} X$, where X is the solution obtained by PAMAL.

6.3 Numerical Results on Sparse PCA

In this section, we compare the performance of ManPG, ManPG-Ada, SOC, and PAMAL for solving the sparse PCA problem (1.2). Note that there are other algorithms for sparse PCA such as the ones proposed in [50, 25], but these methods work only for the special case when $r = 1$; i.e., the constraint set is a sphere. The algorithm proposed in [34] needs to smooth the ℓ_1 norm in order to apply existing gradient-type methods and thus the sparsity of the solution is no longer guaranteed. Algorithms proposed in [94, 70, 51] do not impose orthogonal loading directions. In other words, they cannot impose both sparsity and orthogonality on the same variable. Therefore, we chose not to compare our ManPG with these algorithms.

The random data matrices $A \in \mathbb{R}^{m\times n}$ considered in this section were generated in the following way. We first generate a random matrix using the MATLAB function $A = randn(m, n)$, then shift the columns of A so that their mean is equal to 0, and lastly normalize the columns so that their Euclidean norms are equal to one. In all tests, m is equal to 50. The Lipschitz constant L is $2\sigma^2_{\max}(A)$, so we use $t = 1/(2\sigma^2_{\max}(A))$ in Algorithms 1 and 2, where $\sigma_{\max}(A)$ is the largest singular value of A. Again, we spent a lot of effort on tuning the parameters for SOC and PAMAL and found that the following settings of the parameters worked best for our tested problems. For SOC, we set the penalty parameters $\beta = 2\sigma^2_{\max}(A)$. For PAMAL, we set $\tau = 0.99$, $\gamma = 1.001$, $\rho^1 = 5\sigma^2_{\max}(A)$, $\overline{\Lambda}_{p,\min} = -100$, $\overline{\Lambda}_{p,\max} = 100$, $\Lambda^1_p = 0_{nr}, p = 1, 2$, and $\epsilon^k = (0.996)^k$, $k \in \mathbb{N}$. We again refer the reader to page B587 of [22] for the meanings of these parameters. We used the same parameters of PAM in PAMAL as suggested in [22]. We used the same stopping criterion for ManPG, ManPG-Ada, SOC, and PAMAL as for the CM problems. For different settings of (n, r, μ), we ran the four algorithms with 50 instances whose initial points were obtained by projecting randomly generated points onto $\mathrm{St}(n, r)$. We then ran the Riemannian subgradient method (6.4) for 500 iterations and used the returned solution to be the initial point of the compared solvers.

The CPU time, iteration number, and sparsity are reported in Figures 7, 8, and 9, respectively. Same as the CM problem, all the values were averaged over those instances that yielded solutions that were close to the ones given by ManPG. In Figures 7, 8 and 9, we see that ManPG and ManPG-Ada significantly outperformed SOC and PAMAL in terms of CPU time required to obtain the same solutions. We also see that ManPG-Ada greatly improved the performance of ManPG. We also report the total number of line search steps and the averaged iteration number of SSN in ManPG and ManPG-Ada in Table 2. We observe from Table 2 that SOC failed to converge on one instance, and for several instances, SOC and PAMAL generated different solutions when compared to those generated by ManPG.

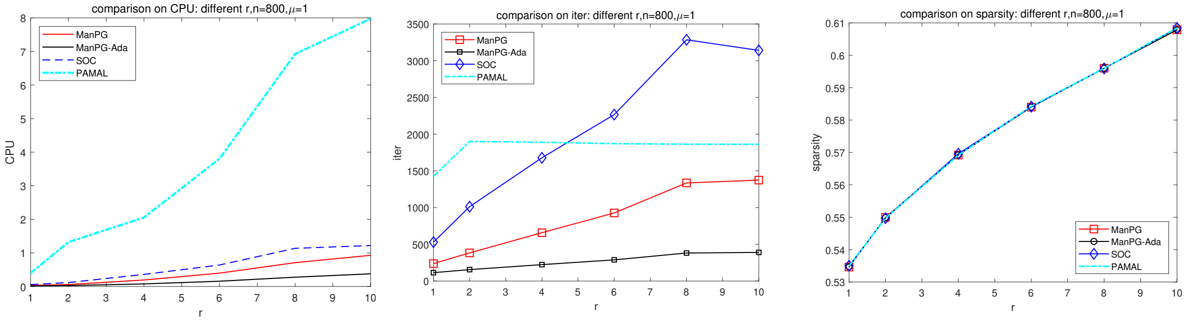


(a) CPU

(b) Iteration

(c) Sparsity

Figure 7: Comparison on sparse PCA problem (1.2), different $n = \{100, 200, 500, 800, 1000, 1500\}$ with $r = 5$ and $\mu = 0.8$.

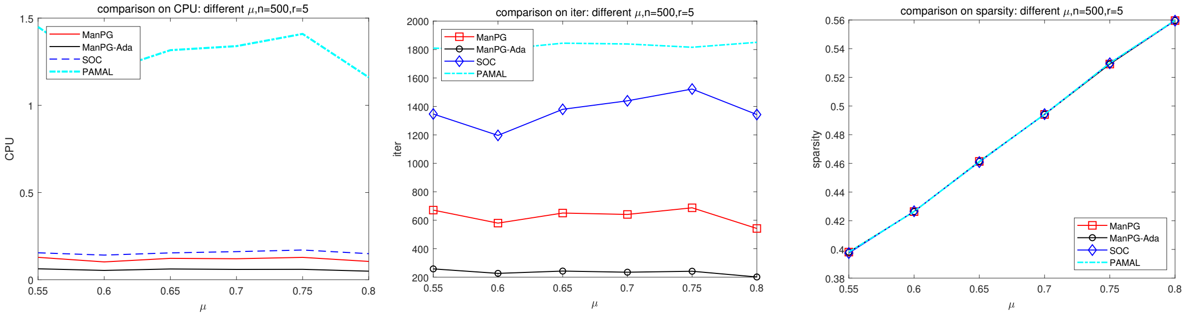


(a) CPU

(b) Iteration

(c) Sparsity

Figure 8: Comparison on sparse PCA problem (1.2), different $r = \{1, 2, 4, 6, 8, 10\}$ with $n = 800$ and $\mu = 1$.



(a) CPU

(b) Iteration

(c) Sparsity

Figure 9: Comparison on sparse PCA problem (1.2), different $\mu = \{0.55, 0.6, 0.65, 0.7, 0.75, 0.8\}$ with $n = 500$ and $r = 5$.

7 Discussions and Concluding Remarks

Manifold optimization has attracted a lot of attention recently. In this paper, we proposed a proximal gradient method (ManPG) for solving the nonsmooth nonconvex optimization problem over the Stiefel manifold

Table 2: Sparse PCA: Number of line search steps and averaged SSN iterations for different (n, r, μ).

	ManPG		ManPG-Ada		SOC	PAMAL
	# line search	SSN iter	# line search	SSN iter	# s| d| f	# s| d| f
n			$r = 5, \mu = 0.8$			
100	0.8	1.1881	0.08	1.5221	46|3|1	50|0|0
200	2.98	1.0722	15.1	1.3705	48|2|0	48|2|0
500	0.4	1.025	29.4	1.2066	50|0|0	50|0|0
800	0	1.0167	59.36	1.1847	49|1|0	50|0|0
1000	3.08	1.016	82.04	1.1712	49|1|0	49|1|0
15	11	1.0121	108.94	1.1035	48|2|0	49|1|0
μ			$n = 500, r = 5$			
0.55	0	1.0155	68.7	1.1463	48|2|0	50|0|0
0.60	0	1.0197	48.82	1.1431	50|0|0	49|1|0
0.65	0	1.019	57.96	1.1841	48|2|0	48|2|0
0.70	0	1.0246	52.5	1.2098	49|1|0	50|0|0
0.75	0.36	1.0238	55.88	1.2252	48|2|0	49|1|0
0.80	0	1.0286	28.98	1.1966	49|1|0	49|1|0
r			$n = 800, \mu = 0.6$			
1	0	0.90182	4.12	1.0335	50|0|0	50|0|0
2	82.06	1.0041	10.74	1.0767	49|1|0	50|0|0
4	8.52	1.0229	39.04	1.1453	48|2|0	50|0|0
6	0	1.0243	72.22	1.3198	46|4|0	49|1|0
8	0.34	1.0309	125.64	1.5325	46|4|0	50|0|0
10	0.76	1.0579	132.58	1.6894	42|8|0	47|3|0

(1.1). Different from existing methods, our ManPG algorithm relies on proximal gradient information on the tangent space rather than subgradient information. Under the assumption that the the smooth part of the objective function has a Lipschitz continuous gradient, we proved that ManPG converges globally to a stationary point of (1.1). Moreover, we analyzed the iteration complexity of ManPG for obtaining an ϵ-stationary solution. Our numerical experiments suggested that when combined with a regularized semi-smooth Newton method for finding the descent direction, ManPG performs efficiently and robustly. In particular, ManPG is more robust than SOC and PAMAL for solving the compressed modes and sparse PCA problems, as it is less sensitive to the choice of parameters. Moreover, ManPG significantly outperforms SOC and PAMAL for solving the sparse PCA problem in terms of the CPU time needed for obtaining the same solution.

It is worth noting that the convergence and iteration complexity analyses in Section 5 also hold for other, not necessarily bounded, embedded submanifolds of an Euclidean space, provided that the objective function F satisfies some additional assumptions (e.g., F is coercive and lower bounded on $\mathcal{M}$). We focused on the Stiefel manifold because it is easier to discuss the semi-smooth Newton method in Section 4.2 for finding the descent direction. As demonstrated in our tests on the compressed modes and sparse PCA problems, the efficiency of ManPG highly relies on that of solving the convex subproblem to find the descent direction. For general Riemannian submanifolds, it remains an interesting question whether the operator $\mathcal{A}_k$ in (4.4) can be easily computed and the resulting subproblem can be solved efficiently.

Acknowledgements

We are grateful to the associate editor and two anonymous referees for constructive and insightful comments that significantly improved the presentation of this paper. We would like to thank Rongjie Lai for discussions on SOC and the compressed modes problem, Lingzhou Xue for discussions on the sparse PCA problem, Zaiwen Wen for discussions on semi-smooth Newton methods, and Wen Huang and Ke Wei for discussions on manifolds. Part of this work was conducted while Shixiang Chen was an intern in Tencent AI Lab. S. Ma was supported in part by a startup package in the Department of Mathematics at University of California, Davis. A. M.-C. So was supported in part by the Hong Kong Research Grants Council (RGC) General Research Fund (GRF) project CUHK 14203218 and in part by the CUHK Research Sustainability of Major RGC Funding Schemes project 3133236.

A Semi-smoothness of Proximal Mapping

Definition A.1. *Let $E : \Omega \to \mathbb{R}^q$ be locally Lipschitz continuous at $X \in \Omega \subset \mathbb{R}^p$. The B-subdifferential of E at X is defined by*

$$\partial_B E(X) := \left\{ \lim_{k\to\infty} E'(X_k) \,\middle|\, X^k \in D_E, X_k \to X \right\},$$

where D_E is the set of differentiable points of E in Ω. The set $\partial E(X) = conv(\partial_B E(X))$ is called Clarke's generalized Jacobian, where conv denotes the convex hull.

Note that if $q = 1$ and E is convex, then the definition is the same as that of standard convex subdifferential. Thus, we use the notation ∂ in Definition A.1.

Definition A.2. *[60, 65] Let $E : \Omega \to \mathbb{R}^q$ be locally Lipschitz continuous at $X \in \Omega \subset \mathbb{R}^p$. We say that E is semi-smooth at $X \in \Omega$ if E is directionally differentiable at X and for any $J \in \partial E(X + \Delta X)$ with $\Delta X \to 0$,*

$$E(X + \Delta X) - E(X) - J\Delta X = o(\|\Delta X\|).$$

We say that E is strongly semi-smooth at X if E is semi-smooth at X and

$$E(X + \Delta X) - E(X) - J\Delta X = O(\|\Delta X\|^2).$$

We say that E is semi-smooth on Ω if it is semi-smooth at every $X \in \Omega$.

The proximal mapping of ℓ_p ($p \geq 1$) norm is strongly semi-smooth [29, 75]. From [75, Prop. 2.26], if $E : \Omega \to \mathbb{R}^m$ is a piecewise $\mathcal{C}^1$ (piecewise smooth) function, then E is semi-smooth. If E is a piecewise $\mathcal{C}^2$ function, then E is strongly semi-smooth. It is known that proximal mappings of many interesting functions are piecewise linear or piecewise smooth.

References

[1] T. E. Abrudan, J. Eriksson, and V. Koivunen. Steepest descent algorithms for optimization under unitary matrix constraint. *IEEE Transactions on Signal Processing*, 56(3):1134–1147, 2008.

[2] T. E. Abrudan, J. Eriksson, and V. Koivunen. Conjugate gradient algorithm for optimization under unitary matrix constraint. *Signal Processing*, 89(9):1704–1714, 2009.

[3] P.-A. Absil and S. Hosseini. A collection of nonsmooth Riemannian optimization problems. *Springer International Series of Numerical Mathematics (ISNM)*, 2017.

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

[5] P.-A. Absil and I. V Oseledets. Low-rank retractions: a survey and new results. *Computational Optimization and Applications*, 62(1):5–29, 2015.

[6] M. Afonso, J. Bioucas-Dias, and M. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *IEEE Transactions on Image Processing*, 19(9):2345–2356, 2010.

[7] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, 2010.

[8] H. Attouch, Z. Chbani, J. Peypouquet, and P. Redont. Fast convergence of inertial dynamics and algorithms with asymptotic vanishing viscosity. *Mathematical Programming*, 168(1-2):123–175, 2018.

[9] M. Bacak, R. Bergmann, G. Steidl, and A. Weinmann. A second order nonsmooth variational model for restoring manifold-valued images. *SIAM Journal on Scientific Computing*, 38:A567–A597, 2016.

[10] T. Bendory, Y. C. Eldar, and N. Boumal. Non-convex phase retrieval from STFT measurements. *IEEE Transactions on Information Theory*, 64(1):467–484, 2018.

[11] G. C. Bento, J. X. Cruz Neto, and P. R. Oliveira. Convergence of inexact descent methods for nonconvex optimization on Riemannian manifolds. *https://arxiv.org/abs/1103.4828v1*, 2011.

[12] G. C. Bento, J. X. Cruz Neto, and P. R. Oliveira. A new approach to the proximal point method: convergence on general Riemannian manifolds. *J. Optim Theory Appl*, 168:743–755, 2016.

[13] G. C. Bento, O. P. Ferreira, and J. G. Melo. Iteration-complexity of gradient, subgradient and proximal point methods on Riemannian manifolds. *J. Optim Theory Appl*, 173:548–562, 2017.

[14] R. L. Bishop and B. O'Neill. Manifolds of negative curvature. *Transactions of the American Mathematical Society*, 145:1–49, 1969.

[15] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146:459–494, 2014.

[16] P. B. Borckmans, S. Easter Selvan, N. Boumal, and P.-A. Absil. A Riemannian subgradient algorithm for economic dispatch with valve-point effect. *J. Comput. Appl. Math.*, 255:848–866, 2014.

[17] N. Boumal. Nonconvex phase synchronization. *SIAM Journal on Optimization*, 26(4):2355–2377, 2016.

[18] N. Boumal and P.-A. Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In *NIPS*, 2011.

[19] N. Boumal, P.-A. Absil, and C. Cartis. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 39(1):1–33, 2019.

[20] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15(1):1455–1459, 2014.

[21] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[22] W. Chen, H. Ji, and Y. You. An augmented Lagrangian method for ℓ_1-regularized optimization problems with orthogonality constraints. *SIAM Journal on Scientific Computing*, 38(4):B570–B592, 2016.

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

[24] P. L. Combettes and J.-C. Pesquet. A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):564–574, 2007.

[25] 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.

[26] G. Dirr, U. Helmke, and C. Lageman. Nonsmooth Riemannian optimization with applications to sphere packing and grasping. *Lagrangian and Hamiltonian Methods for Nonlinear Control*, pages 28–45, 2006.

[27] J. Eckstein. *Splitting methods for monotone operators with applications to parallel optimization.* PhD thesis, Massachusetts Institute of Technology, 1989.

[28] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353 (electronic), 1999.

[29] F. Facchinei and J. Pang. *Finite-dimensional variational inequalities and complementarity problems.* Springer Science & Business Media, 2007.

[30] O. P. Ferreira and P. R. Oliveira. Subgradient algorithm on Riemannian manifolds. *Journal of Optimization Theory and Applications*, 97(1):93–104, 1998.

[31] O. P. Ferreira and P. R. Oliveira. Proximal point algorithm on Riemannian manifold. *Optimization*, 51:257–270, 2002.

[32] M. Fortin and R. Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems.* North-Holland Pub. Co., 1983.

[33] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite-element approximations. *Comp. Math. Appl.*, 2:17–40, 1976.

[34] M. Genicot, W. Huang, and N. T. Trendafilov. Weakly correlated sparse components with nearly orthonormal loadings. *Geometric Science of Information*, pages 484–490, 2015.

[35] R. Glowinski and P. Le Tallec. *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics.* SIAM, Philadelphia, Pennsylvania, 1989.

[36] R. Glowinski and A. Marrocco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, pages 41–76, 1975.

[37] T. Goldstein and S. Osher. The split Bregman method for L1-regularized problems. *SIAM J. Imaging Sci.*, 2:323–343, 2009.

[38] G. H. Golub and C. F. Van Loan. *Matrix Computations.* JHU Press, 2012.

[39] P. Grohs and S. Hosseini. Nonsmooth trust region algorithms for locally Lipschitz functions on Riemannian manifolds. *IMA J. Numer. Anal.*, 36:1167–1192, 2016.

[40] P. Grohs and S. Hosseini. ε-subgradient algorithms for locally lipschitz functions on riemannian manifolds. *Advances in Computational Mathematics*, 42(2):333–360, 2016.

[41] J.-B. Hiriart-Urruty, J.-J. Strodiot, and V. H. Nguyen. Generalized hessian matrix and second-order optimality conditions for problems with $C^{1,1}$ data. *Applied mathematics and optimization*, 11(1):43–56, 1984.

[42] S. Hosseini. Convergence of nonsmooth descent methods via Kurdyka-Łojasiewicz inequality on Riemannian manifolds. *Technical Report. Hausdorff Center for Mathematics and Institute for Numerical Simulation, University of Bonn*, 2015.

[43] S. Hosseini, W. Huang, and R. Yousefpour. Line search algorithms for locally Lipschitz functions on Riemannian manifolds. *SIAM Journal on Optimization*, 28(1):596–619, 2018.

[44] S. Hosseini and M. R. Pouryayevali. Generalized gradients and characterization of epi-Lipschitz sets in Riemannian manifolds. *Nonlinear Analysis: Theory, Methods & Applications*, 72(12):3884–3895, 2011.

[45] S. Hosseini and A. Uschmajew. A Riemannian gradient sampling algorithm for nonsmooth optimization on manifolds. *SIAM J. Optim.*, 27(1):173–189, 2017.

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

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

[48] B. Jiang and Y.-H. Dai. A framework of constraint preserving update schemes for optimization on Stiefel manifold. *Mathematical Programming*, 153(2):535–575, 2015.

[49] B. Jiang, S. Ma, A. M.-C. So, and S. Zhang. Vector transport-free SVRG with general retraction for Riemannian optimization: Complexity analysis and practical implementation. *https://arxiv.org/abs/1705.09059v1*, 2017.

[50] 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.

[51] 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.

[52] A. Kovnatsky, K. Glashoff, and M. M. Bronstein. MADMM: a generic algorithm for non-smooth optimization on manifolds. In *European Conference on Computer Vision*, pages 680–696. Springer, 2016.

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

[54] X. Li, D. Sun, and K.-C. Toh. A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems. *SIAM J. Optim.*, 28:433–458, 2018.

[55] P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16:964–979, 1979.

[56] H. Liu, A. M.-C. So, and W. Wu. Quadratic optimization with orthogonality constraint: Explicit Łojasiewicz exponent and linear convergence of retraction-based line-search and stochastic variance-reduced gradient methods. *Mathematical Programming Series A*, 2018. Doi: https://link.springer.com/article/10.1007%2Fs10107-018-1285-1.

[57] H. Liu, M.-C. Yue, and A. M.-C. So. On the estimation performance and convergence rate of the generalized power method for phase synchronization. *SIAM J. Optim.*, 27(4):2426–2446, 2017.

[58] 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.

[59] J. R. Magnus and H. Neudecker. Matrix differential calculus with applications in statistics and econometrics. *Wiley Series in Probability and Mathematical Statistics*, 1988.

[60] R. Mifflin. Semismooth and semiconvex functions in constrained optimization. *SIAM J. Control Optim.*, 15:959–972, 1977.

[61] Y. Nishimori and S. Akaho. Learning algorithms utilizing quasi-geodesic flows on the Stiefel manifold. *Neurocomputing*, 67:106–135, 2005.

[62] V. Ozoliņš, R. Lai, R. Caflisch, and S. Osher. Compressed modes for variational problems in mathematics and physics. *Proceedings of the National Academy of Sciences*, 110(46):18368–18373, 2013.

[63] 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.

[64] H. Qi and D. Sun. An augmented Lagrangian dual approach for the H-weighted nearest correlation matrix problem. *IMA Journal of Numerical Analysis*, 31:491–511, 2011.

[65] L. Qi and J. Sun. A nonsmooth version of Newton's method. *Math. Program.*, 58:353–367, 1993.

[66] R.T. Rockafellar and R. J-B Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

[67] B. Savas and L.-H. Lim. Quasi-Newton methods on Grassmannians and multilinear approximations of tensors. *SIAM Journal on Scientific Computing*, 32(6):3352–3393, 2010.

[68] O. Shamir. A stochastic PCA and SVD algorithm with an exponential convergence rate. In *ICML*, 2015.

[69] O. Shamir. Fast stochastic algorithms for SVD and PCA: Convergence properties and convexity. In *ICML*, 2016.

[70] 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.

[71] M. V. Solodov and B. F. Svaiter. A globally convergent inexact Newton method for systems of monotone equations. In *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, pages 355–369. Springer, 1998.

[72] 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.

[73] J. Sun, Q. Qu, and J. Wright. A geometrical analysis of phase retrieval. *Foundations of Computational Mathematics*, 18(5):1131–1198, 2018.

[74] J. Tang and H. Liu. Unsupervised feature selection for linked social media data. In *SIGKDD*, pages 904–912. ACM, 2012.

[75] M. Ulbrich. *Semismooth Newton methods for variational inequalities and constrained optimization problems in function spaces*, volume 11. SIAM, 2011.

[76] B. Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM J. Optim.*, 23(2):1214–1236, 2013.

[77] C. Wang, D. Sun, and K.-C. Toh. Solving log-determinant optimization problems by a Newton-CG primal proximal point algorithm. *SIAM J. Optimization*, 20:2994–3013, 2010.

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

[79] Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[80] X. Xiao, Y. Li, Z. Wen, and L. Zhang. A regularized semi-smooth Newton method with projection steps for composite convex programs. *Journal of Scientific Computing*, 76(1):364–389, 2018.

[81] J. Yang, D. Sun, and K.-C. Toh. A proximal point algorithm for log-determinant optimization with group Lasso regularization. *SIAM J. Optim.*, 23:857–893, 2013.

[82] J. Yang, W. Yin, Y. Zhang, and Y. Wang. A fast algorithm for edge-preserving variational multichannel image restoration. *SIAM Journal on Imaging Sciences*, 2(2):569–592, 2008.

[83] L. Yang, D. Sun, and K.-C. Toh. SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.

[84] 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.

[85] Y. Yang, H. Shen, Z. Ma, Z. Huang, and X. Zhou. $\ell_{2,1}$-norm regularized discriminative feature selection for unsupervised learning. In *IJCAI*, volume 22, page 1589, 2011.

[86] C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Annuals of Statistics*, pages 894–942, 2010.

[87] H. Zhang, S. Reddi, and S. Sra. Fast stochastic optimization on Riemannian manifolds. In *NIPS*, 2016.

[88] H. Zhang and S. Sra. First-order methods for geodesically convex optimization. *Proceedings of the 29th Conference on Learning Theory*, 49:1617–1638, 2016.

[89] J. Zhang, S. Ma, and S. Zhang. Primal-dual optimization algorithms over Riemannian manifolds: an iteration complexity analysis. *Mathematical Programming Series A*, 2019. Doi: https://link.springer.com/article/10.1007%2Fs10107-019-01418-8.

[90] Y. Zhang, Y. Lau, H.-W. Kuo, S. Cheung, A. Pasupathy, and J. Wright. On the global geometry of sphere-constrained sparse blind deconvolution. In *CVPR*, 2017.

[91] X. Zhao, D. Sun, and K.-C. Toh. A Newton-CG augmented Lagrangian method for semidefinite programming. *SIAM Journal on Optimization*, 20:1737–1765, 2010.

[92] G. Zhou and K.-C. Toh. Superlinear convergence of a Newton-type algorithm for monotone equations. *Journal of optimization theory and applications*, 125(1):205–221, 2005.

[93] H. Zhu, X. Zhang, D. Chu, and L. Liao. Nonconvex and nonsmooth optimization with generalized orthogonality constraints: An approximate augmented Lagrangian method. *Journal of Scientific Computing*, 72(1):331–372, 2017.

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

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