Proximal Quasi-Newton Method for Composite Optimization over the Stiefel Manifold

Qinsi Wang[1] and Wei Hong Yang[1*]

[1*]School of Mathematical Sciences, Fudan University, 220 Handan Street, Shanghai, 200433,China.

*Corresponding author(s). E-mail(s): whyang@fudan.edu.cn; Contributing authors: qinsiwang20@fudan.edu.cn;

Abstract

In this paper, we consider the composite optimization problems over the Stiefel manifold. A successful method to solve this class of problems is the proximal gradient method proposed by [Chen et al., SIAM J. Optim., 30 (2020), pp. 210–239]. Motivated by the proximal Newton-type techniques in the Euclidean space, we present a Riemannian proximal quasi-Newton method, named ManPQN, to solve the composite optimization problems. The global convergence of the ManPQN method is proved and iteration complexity for obtaining an ϵ-stationary point is analyzed. Under some mild conditions, we also establish the local linear convergence result of the ManPQN method. Numerical results are encouraging, which shows that the proximal quasi-Newton technique can be used to accelerate the proximal gradient method.

Keywords: Proximal Newton-type method. Stiefel manifold. Quasi-Newton method. Nonmonotone line search. Linear convergence

Mathematics Subject Classification: 90C30

1 Introduction

In this paper, we consider the following composite optimization problem

$$\begin{aligned} &\min_{X}\ F(X) := f(X) + h(X), \\ &s.t.\ \ X^\top X = I_r, \end{aligned} \tag{1.1}$$

where $f : \mathbb{R}^{n\times r} \to \mathbb{R}$ is a smooth function and $h : \mathbb{R}^{n\times r} \to \mathbb{R}$ is a convex nonsmooth function. The feasible set $\mathrm{St}(n, r) := \{X \in \mathbb{R}^{n\times r} : X^\top X = I_r\}$ is referred to as the Stiefel manifold.

Problem (1.1) has wide applications in many fields such as machine learning, signal processing and numerical linear algebra. For example, when $f(X) = -\mathrm{tr}(X^T A^T AX)$ and $h(X) = \|X\|_1$, (1.1) is just the sparse PCA problem; When $f(X) = \mathrm{tr}(X^T MX)$ and $h(X) = \|X\|_{2,1}$, where M is a given matrix and $\|X\|_{2,1} := \sum_{i=1}^n \|X(i, :)\|_2$, (1.1) becomes the UFS (unsupervised feature selection) problem, which finds features that represent the distribution of the input data best, both to reduce the dimension of the data and to eliminate the noisy features. For more applications of composite optimization over the Stiefel manifold, we refer the readers to [1, 2, 6].

In the Euclidean setting, we can use the subgradient method to solve the composite optimization problems. In [36], Shor proved the convergence of subgradient method for nonsmooth convex optimization. There are several works which extend subgradient methods from Euclidean space to Riemannian manifolds. Ferreira and Oliveira [9] studied the convergence of subgradient method. Grohs and Hosseini [14] proposed an ε-subgradient method and establish the global converge result. In [15], they proposed an algorithm which combines the subgradient method with the trust-region model.

Another interesting approach is to use the operator splitting method to separate the manifold constraint part and the nonsmooth function part. In the Euclidean space, it is known that the operator splitting method needs more iterations to achieve the same accuracy as the proximal gradient method. Chen et al. [6] showed similar results in numerical experiments for the Riemannian setting.

A popular and efficient method for composite optimization in the Euclidean space is the proximal gradient (PG) method [26]. For a nonsmooth convex function h, the proximal mapping of h is defined by

$$\mathrm{prox}_h(x) = \arg\min_y \left\{ h(y) + \frac{1}{2}\|y - x\|_2^2 \right\}. \tag{1.2}$$

To minimize $f(x) + h(x)$, where f is smooth, the PG method takes the step

$$x_{k+1} = \mathrm{prox}_{th}(x_k - t\nabla f(x_k)),$$

where $t > 0$. It is well known that

$$x_{k+1} = \arg\min_x \left\{ \langle \nabla f(x_k), x\rangle + \frac{1}{2t}\|x - x_k\|_2^2 + h(x) \right\}. \tag{1.3}$$

There are several techniques can be used to accelerate the PG method. Beck and Teboulle [3] use Nesterov's acceleration technique to propose an algorithm, named FISTA, which shows the improved rate $O(1/k^2)$.

Another acceleration technique is the proximal Newton-type method, which is proposed in [11, 20]. Based on (1.2), Lee et al. [20] define the scaled proximal mapping as

$$\mathrm{prox}_h^B(x) = \arg\min_y \left\{ h(y) + \frac{1}{2}(y - x)^T B(y - x) \right\}. \tag{1.4}$$

where B is a positive definite matrix. The motivation of the proximal Newton-type method is to replace the term $\|x - x_k\|_2^2/(2t)$ in (1.3) by $(x - x_k)^T B_k(x - x_k)/2$, where B_k is an approximate matrix of $\nabla^2 f(x_k)$. The proximal Newton-type method takes the step

$$\begin{aligned} x_{k+1} &= \mathrm{prox}_h^{B_k}(x_k - {B_k}^{-1}\nabla f(x_k)) \\ &= \arg\min_x \left\{ \langle \nabla f(x_k), x\rangle + \frac{1}{2}(x-x_k)^T B_k (x-x_k) + h(x) \right\}. \end{aligned} \tag{1.5}$$

Superlinear convergence rate of the proximal Newton-type method has been established in [20, 23]. In [25], Nakayama proved the local linear rate of the inexact proximal quasi-Newton method.

Recently, many researchers have extended proximal-type methods to Riemannian manifolds. See [4, 5, 10, 12, 17, 18, 29, 42]. In [12], Gao et al. propose a parallelized proximal linearized augmented Lagrangian algorithm for solving optimization problems over the Stiefel manifold and show its convergence property. Oviedo [29] designs an iterative proximal point algorithm to minimize a continuously differentiable function over the Stiefel manifold, which avoids to construct geodesics and can be proved to converge globally. Zhang et al. [42] extend the smoothing steepest descent method for nonconvex and non-Lipschitz optimization from Euclidean space to Riemannian manifolds. The method can be applied to solve composite optimization problems. In [6], Chen et al. propose a retraction-based proximal gradient method, named ManPG, for composite optimization over the Stiefel manifold. Numerical experiments in [6] show that the performance of ManPG outperforms subgradient-based methods and ADMM-type methods for sparse PCA and CM problems. At the k-th iteration, the ManPG method solves the following subproblem to get a descent direction at X_k:

$$V_k = \arg\min_{V \in \mathrm{T}_{X_k}\mathcal{M}} \left\{ \langle \mathrm{grad} f(X_k), V\rangle + \frac{1}{2}\|V\|^2 + h(X_k + V) \right\}. \tag{1.6}$$

To accelerate the proximal gradient method, we propose a Riemannian proximal quasi-Newton method (ManPQN) to solve (1.1), which replaces $\frac{1}{2}\|V\|^2$ in (1.6) by $\frac{1}{2}\|V\|^2_{\mathcal{B}_k}$, where $\mathcal{B}_k$ is a linear operator on $\mathrm{T}_{X_k}\mathcal{M}$ updated by a quasi-Newton strategy. The ManPQN method can be regarded as an extension of the proximal quasi-Newton method from Euclidean space to Riemannian manifolds.

Our main contributions can be summarized as follows. We propose a proximal quasi-Newton method named ManPQN to solve (1.1). Under some mild conditions, we establish the global convergence and local linear convergence results of ManPQN. Since the computational cost of updating $\mathcal{B}_k$ by the quasi-Newton strategy is very large, we use a strategy for updating $\mathcal{B}_k$, which can reduce the amount of computation significantly. Numerical experiments demonstrate that the proximal quasi-Newton technique can be used to accelerate the ManPG method.

The organization of the paper is shown as follows. In section 2, we introduce some notations and definitions that will be frequently used throughout the paper. In section 3, we propose the ManPQN algorithm in detail. The global convergence of ManPQN is proved and iteration complexity for obtaining an ϵ-stationary point is analyzed in section 4. Under some mild conditions, we also establish the local linear convergence result of the ManPQN method in this section. In section 5, we compare the ManPQN method with ManPG related methods in the numerical experiments. The paper ends with some conclusions and a short discussion on possible future works.

2 Notations and Preliminaries

In this section, we introduce some notations and definitions which will be used in the rest of the paper. Throughout this section, $\mathcal{M}$ denotes a general manifold.

Definition 2.1 (Tangent space [2, p.33]) Let $X \in \mathcal{M}$. A tangent vector ξ_X to a manifold $\mathcal{M}$ at X is a mapping from $\mathfrak{F}_X(\mathcal{M})$ to $\mathbb{R}$ such that there exists a smooth curve γ on $\mathcal{M}$ with $\gamma(0) = X$, satisfying

$$\xi_X f = \dot{\gamma}(0) f := \left.\frac{\mathrm{d}(f(\gamma(t)))}{\mathrm{d}t}\right|_{t=0},$$

for all $f \in \mathfrak{F}_X(\mathcal{M})$, where $\mathfrak{F}_X(\mathcal{M})$ denotes the set of smooth real-valued functions defined on a neighborhood of X over the manifold $\mathcal{M}$. The tangent space to $\mathcal{M}$ at X, denoted by $\mathrm{T}_X\mathcal{M}$, is the set of all tangent vectors to $\mathcal{M}$ at X. The tangent bundle $\mathrm{T}\mathcal{M} := \cup_X \mathrm{T}_X\mathcal{M}$ consists of all tangent vectors to $\mathcal{M}$.

The Riemannian manifold $\mathcal{M}$ is a manifold whose tangent spaces are endowed with a smoothly varying inner product $\langle \xi, \zeta \rangle_X$, where $\xi, \zeta \in \mathrm{T}_X\mathcal{M}$. The norm induced by the Riemannian inner product is denoted by $\|\cdot\|_X$, where $\|\xi\|_X := (\langle \xi, \xi \rangle_X)^{1/2}$. In this work, for the Stiefel manifold $\mathrm{St}(n, r)$, we consider the Riemannian metric defined by $\langle \xi, \zeta \rangle_X := tr(\xi^\top \zeta)$. For simplicity of notation, we write $\|\cdot\|$ instead of $\|\cdot\|_X$ in the rest of the paper.

The tangent space of the Stiefel manifold $\mathrm{St}(n, r)$ at X is [2, p.42]

$$\mathrm{T}_X\mathrm{St}(n, r) = \{V \mid V^\top X + X^\top V = 0\}.$$

The projection operator onto $\mathrm{T}_X\mathrm{St}(n, r)$ is given by [2, p.48]

$$\mathrm{Proj}_{\mathrm{T}_X\mathrm{St}(n,r)} Z = Z - \frac{1}{2} X (X^\top Z + Z^\top X). \tag{2.1}$$

For a differentiable function $f : \mathcal{M} \to \mathbb{R}$, the derivative of f at $X \in \mathcal{M}$, denoted by $\mathrm{D}f(X)$, is an element of the dual space to $T_X\mathcal{M}$ which satisfies $\mathrm{D}f(X)\xi = \xi f$ for all $\xi \in T_X\mathcal{M}$. The gradient of a smooth function f at X, denoted by $\mathrm{grad} f(X)$ [2, p.46], is defined as the unique element of $\mathrm{T}_X\mathcal{M}$ that satisfies

$$\langle \mathrm{grad} f(X), \xi \rangle = \mathrm{D}f(X)[\xi] \quad \forall \xi \in \mathrm{T}_X\mathcal{M}.$$

If $\mathcal{M}$ is an embedded submanifold of an Euclidean space E, then by [2, p.48],

$$\mathrm{grad} f(X) = \mathrm{Proj}_{\mathrm{T}_X\mathcal{M}} \nabla f(X), \tag{2.2}$$

where $\mathrm{Proj}_{\mathrm{T}_X\mathcal{M}}$ is the projection operator from E onto $\mathrm{T}_X\mathcal{M}$ and $\nabla f(X)$ is the Euclidean gradient of f at X. For any $\xi \in \mathrm{T}_X\mathcal{M}$, from (2.2), it follows that

$$\langle \mathrm{grad} f(X), \xi \rangle = \langle \nabla f(X), \xi \rangle.$$

Next we introduce the definitions of the retraction and the vector transport.

Definition 2.2 (Retraction [2, Definition 4.1.1]) A retraction on a manifold $\mathcal{M}$ is a smooth mapping $\mathbf{R}$ from the tangent bundle $\mathrm{T}\mathcal{M}$ onto $\mathcal{M}$ with the following properties. Let $\mathbf{R}_X$ denote the restriction of $\mathbf{R}$ to $\mathrm{T}_X\mathcal{M}$.

(1) $\mathbf{R}_X(0_X) = X$, where 0_X denotes the zero element of $\mathrm{T}_X\mathcal{M}$.

(2) With the canonical identification $\mathrm{T}_{0_X}(\mathrm{T}_X\mathcal{M}) \simeq \mathrm{T}_X\mathcal{M}$, $\mathbf{R}_X$ satisfies

$$\mathrm{D}\mathbf{R}_X(0_X) = \mathbf{id}_{\mathrm{T}_X\mathcal{M}}, \tag{2.3}$$

where $\mathrm{D}\mathbf{R}_X(0_X)$ denotes the differential of the retraction $\mathbf{R}_X$ at the zero element $0_X \in \mathrm{T}_X\mathcal{M}$ and $\mathbf{id}_{\mathrm{T}_X\mathcal{M}}$ denotes the identity mapping on $\mathrm{T}_X\mathcal{M}$.

The following properties for a retraction $\mathbf{R}$ will be used in section 4. For a proof, see [22].

Proposition 2.1 *([22]) Suppose $\mathcal{M}$ is a compact embedded submanifold of an Euclidean space E, and $\mathbf{R}$ is a retraction. Then there exists $M_1, M_2 > 0$ such that for all $X \in \mathcal{M}$ and for all $\xi \in \mathrm{T}_X\mathcal{M}$,*

$$\|\mathbf{R}_X(\xi) - X\| \le M_1\|\xi\|, \tag{2.4}$$

$$\|\mathbf{R}_X(\xi) - X - \xi\| \le M_2\|\xi\|^2. \tag{2.5}$$

Next we will consider the transport of a vector from one tangent space $\mathrm{T}_X\mathcal{M}$ into another one $\mathrm{T}_Y\mathcal{M}$.

Definition 2.3 (Vector Transport [2, Definition 8.1.1]) A vector transport associated with a retraction $\mathbf{R}$ is defined as a continuous function $\mathcal{T} : \mathrm{T}\mathcal{M} \times \mathrm{T}\mathcal{M} \to \mathrm{T}\mathcal{M}, (\eta_X, \xi_X) \mapsto \mathrm{T}_{\eta_X}(\xi_X)$, which satisfies the following conditions:
(i) $\mathcal{T}_{\eta_X} : \mathrm{T}_X\mathcal{M} \to \mathrm{T}_{\mathbf{R}_X(\eta_X)}\mathcal{M}$ is a linear invertible map,
(ii) $\mathcal{T}_{0_X}(\xi_X) = \xi_X$.

Let $Y := \mathbf{R}_X(\eta_X)$, where $\eta_X \in \mathrm{T}_X\mathcal{M}$. For ease of notation, we denote

$$\mathcal{T}_{X,Y}(\xi_X) := \mathcal{T}_{\eta_X}(\xi_X),$$

where $\xi_X \in \mathrm{T}_X\mathcal{M}$.

For any $h \in \mathrm{T}_X\mathcal{M}$, we use $h^\flat$ to denote the linear function on $T_X\mathcal{M}$ induced by

$$h^\flat\eta := \langle h, \eta\rangle = \mathrm{tr}(h^T\eta) \quad \forall \eta \in \mathrm{T}_X\mathcal{M}.$$

Let $H = [h_1, \ldots, h_m]$, where $h_i \in T_X\mathcal{M}$ for any $i = 1, \ldots, m$. Define $H^\flat : \mathrm{T}_X\mathcal{M} \to \mathbb{R}^m$ by

$$H^\flat\eta = [h_1^\flat\eta, \ldots, h_m^\flat\eta]^T \quad \forall \eta \in \mathrm{T}_X\mathcal{M}.$$

Finally, we introduce the definitions of generalized Calrke subdifferential and regular function.

Definition 2.4 (Generalized Calrke subdifferential [6, 41]) For a locally Lipschitz function ψ on a Riemannian manifold $\mathcal{M}$, the Riemannian generalized directional derivative of F at $X \in \mathcal{M}$ in the direction $V \in \mathrm{T}_X\mathcal{M}$ is defined by

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

where (ϕ, U) is a coordinate chart at X. The generalized Clarke subdifferential of ψ at $X \in \mathcal{M}$, denoted by $\hat{\partial}\psi(X)$, is a subset of $\mathrm{T}_X\mathcal{M}$ whose support function is $\psi^\circ(X;\ \cdot)$, defined by

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

In the rest of this section, we assume that $\mathcal{M}$ be an embedded submanifold of an Euclidean space E.

Definition 2.5 (Regular function [41]) Let f be a function defined on E. We say that f is regular at $X \in \mathcal{M}$ along $\mathrm{T}_X\mathcal{M}$ if

(1) for all $V \in \mathrm{T}_X\mathcal{M}$,

$$f'(X;\ V) = \lim_{t \downarrow 0} \frac{f(X + tV) - f(X)}{t},$$

exists, and

(2) for all $V \in \mathrm{T}_X\mathcal{M}$, $f'(X;\ V) = f^\circ(X;\ V)$.

For a regular nonsmooth function $\psi(X)$, it is proved in [41, Theorem 5.1] that $\hat{\partial}\psi(X) = \mathrm{Proj}_{\mathrm{T}_X\mathcal{M}}(\partial\psi(X))$, where $\partial\psi(X)$ is the Euclidean generalized Clarke subdifferential of ψ at X. Since f is continuously differentiable and h is convex, by [41, Lemma 5.1], the objective function $F = f + h$ is regular. Then the first-order necessary condition of problem (1.1) can be written as

$$0 \in \hat{\partial}F(X^*) = \mathrm{grad} f(X^*) + \mathrm{Proj}_{\mathrm{T}_{X^*}\mathcal{M}}(\partial h(X^*)), \tag{2.6}$$

where X^* is a local optimal solution of (1.1).

3 Proximal Quasi-Newton Method

In this section, we propose a proximal quasi-Newton algorithm over the Stiefel manifold, named ManPQN, which takes a proximal quasi-Newton step at each iteration, and uses the nonmonotone line search strategy to determine the stepsize. In the rest of the paper, we use $\mathcal{M}$ to denote the Stiefel manifold.

3.1 The ManPQN Algorithm

At the current iterate X_k, the ManPQN method solves the following problem to get the descent direction V_k:

$$V_k = \arg\min\ \langle \nabla f(X_k), V \rangle + \frac{1}{2}\|V\|^2_{\mathcal{B}_k} + h(X_k + V) \tag{3.1}$$

$$\text{s.t.} \quad \mathcal{A}_k(V) := V^T X_k + X_k^T V = 0, \tag{3.2}$$

where $\mathcal{B}_k$ is a symmetric positive definite operator on $\mathrm{T}_{X_k}\mathcal{M}$, and $\|V\|^2_{\mathcal{B}_k} = \langle V, \mathcal{B}_k[V]\rangle$. From (3.2), we know that $V_k \in \mathrm{T}_{X_k}\mathcal{M}$.

To guarantee the positive definiteness of $\mathcal{B}_k$, we use a damped LBFGS strategy to generate $\mathcal{B}_k$. We leave the details of the updating in the next subsection.

After V_k is obtained, we apply the nonmonotone line search technique to determine the stepsize α_k. The nonmonotone technique was first introduced in [13]. Specifically, α_k is set to be γ^{N_k}, where N_k is the smallest integer such that

$$F(\mathbf{R}_{X_k}(\alpha_k V_k)) \leq \max_{\max\{0,k-m\}\leq j\leq k} F(X_j) - \frac{1}{2}\sigma\alpha_k\|V_k\|^2_{\mathcal{B}_k}, \tag{3.3}$$

in which $m > 0$ is an integer, $\sigma, \gamma \in (0,1)$. For simplicity, we use the notation

$$l(k) := \operatorname*{arg\,max}_{\max\{k-m,0\}\leq j\leq k} F(X_j). \tag{3.4}$$

Then $F(X_{l(k)}) = \max_{\max\{k-m,0\}\leq j\leq k} F(X_j)$.

Now we summarize the ManPQN method as follows:

Algorithm 1 Proximal quasi-Newton algorithm with nonmonotone line search to solve problem (1.1)

Require: Initial point $X_0 \in \mathcal{M}$, $\gamma, \sigma \in (0,1)$, $m, p > 0$ are integers.

for $k = 0, 1, \ldots$ **do**
    if $k \geq 1$ **then**
        Update $\mathcal{B}_k$ by quasi-Newton strategy;
    else
        Set $\mathcal{B}_k = I$;
    end if
    Solve the subproblem (3.1) to get the search direction V_k;
    Set initial stepsize $\alpha_k = 1$;
    while (3.3) is satisfied **do**
        $\alpha_k = \gamma\alpha_k$;
    end while
    Set $X_{k+1} = \mathbf{R}_{X_k}(\alpha_k V_k)$;
end for

3.2 Damped LBFGS Method

If the operator $\mathcal{B}_k$ in (3.1) is updated by the Riemannian BFGS method, the total amount of computation is very large. It is well known that the Limited memory BFGS method (LBFGS) [28] is more suitable for large scale problems. Comparing with the BFGS method, the LBFGS method only needs the information of the last p steps, where p is the memory size.

To give the Riemannian LBFGS update formula, we introduce some notations first. For $k \geq 0$, let $g_k = \mathrm{grad} f(X_k) \in \mathrm{T}_{X_k}\mathcal{M}$ and $\mathcal{T}_{k,k+1} := \mathcal{T}_{X_k, X_{k+1}}$. Define

$$S_k := \mathcal{T}_{k,k+1}(\mathbf{R}^{-1}_{X_k}(X_{k+1})), \; Y_k := g_{k+1} - \mathcal{T}_{k,k+1}(g_k).$$

Given an initial estimate $\mathcal{B}_{k,0}$ at iteration k and two sequences $\{S_j\}$ and $\{Y_j\}$, $j = k-p, \ldots, k-1$, the Riemannian LBFGS method updates $\mathcal{B}_{k,i}$ recursively as

$$\begin{aligned}\mathcal{B}_{k,i} &= \widetilde{\mathcal{B}}_{k,i-1} - \frac{\widetilde{\mathcal{B}}_{k,i-1} S_j (\widetilde{\mathcal{B}}_{k,i-1} S_j)^\flat}{S_j^\flat \widetilde{\mathcal{B}}_{k,i-1} S_j} + \rho_j Y_j Y_j^\flat, \\ j &= k-(p-i+1),\ i = 1, \ldots, p,\end{aligned} \tag{3.5}$$

where

$$\widetilde{\mathcal{B}}_{k,i-1} = \mathcal{T}_{j,j+1} \circ \mathcal{B}_{k,i-1} \circ \mathcal{T}_{j,j+1}^{-1} \text{ and } \rho_j = \frac{1}{Y_j^\flat S_j}.$$

The output $\mathcal{B}_{k,p}$ is set to be $\mathcal{B}_k$, the Riemannian LBFGS operator. It is easy to see $\mathcal{B}_k$ is a linear operator on $\mathrm{T}_{X_k}\mathcal{M}$.

The inverse of $\mathcal{B}_{k,i}$ is denoted by $\mathcal{H}_{k,i}$, $i = 0, 1, \cdots, p$. By the Sherman-Morrison-Woodbury formula [35], we have

$$\begin{aligned}\mathcal{H}_{k,i} &= (\mathbf{id} - \rho_j S_j Y_j^\flat) \mathcal{T}_{j,j+1} \mathcal{H}_{k,i-1} \mathcal{T}_{j,j+1}^{-1} (\mathbf{id} - \rho_j Y_j S_j^\flat) + \rho_j S_j S_j^\flat, \\ j &= k-(p-i+1),\ i = 1, \ldots, p,\end{aligned}$$

where $\mathbf{id}$ denotes the identity map. It is obvious that $\mathcal{H}_k = \mathcal{H}_{k,p}$, where $\mathcal{H}_k$ is the inverse of $\mathcal{B}_k$.

From (3.5), we know that the update of $\mathcal{B}_k$ is computationally expensive because it involves the calculation of vector transports. Since the Stiefel manifold $\mathcal{M}$ is a submanifold of a $\mathbb{R}^{n\times r}$, we can calculate the Euclidean differences s_k and y_k to replace S_k and Y_k, that is

$$s_k = X_{k+1} - X_k \in \mathbb{R}^{n\times r}, \qquad y_k = g_{k+1} - g_k \in \mathbb{R}^{n\times r}.$$

Calculating s_k and y_k is much cheaper than calculating S_k and Y_k. Such a strategy was used in [16, 38].

To reduce the computational cost further, we will use a simple and easily computed $\mathbf{B}_k$ to approximate $\mathcal{B}_k$. Such a strategy was used in [21, 31] for the Euclidean setting. In the ManPQN method, we solve the following subproblem to get V_k:

$$V_k = \operatorname*{arg\,min}_{V \in \mathrm{T}_{X_k}\mathcal{M}} \left\{ \langle \nabla f(X_k), V \rangle + \frac{1}{2} \mathrm{tr}(V^T \mathbf{B}_k[V]) + h(X_k + V) \right\}, \tag{3.6}$$

where

$$\mathbf{B}_k[V] = \mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}}\big((\mathrm{diag} B_k)(\mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}} V)\big), \tag{3.7}$$

in which $B_k \in \mathbb{R}^{n\times n}$ is a symmetric matrix. By (3.7), we can deduce that

$$\mathrm{tr}(V^T \mathbf{B}_k[V]) = \mathrm{tr}(V^T (\mathrm{diag} B_k) V) \quad \forall V \in \mathrm{T}_{X_k}\mathcal{M}. \tag{3.8}$$

To ensure V_k is a descent direction of F at X_k, a sufficient condition is that B_k is a positive definite matrix. To guarantee the positive definitness of B_k, we use the damped technique employed in [32, 37]. Specifically, for $k \geq 0$, define

$$\overline{y}_{k-1} = \beta_{k-1} y_{k-1} + (1 - \beta_{k-1}) H_{k,0}^{-1} s_{k-1}, \tag{3.9}$$

where

$$\beta_{k-1} = \begin{cases} \frac{0.75\mathrm{tr}(s_{k-1}^T H_{k,0}^{-1} s_{k-1})}{\mathrm{tr}(s_{k-1}^T H_{k,0}^{-1} s_{k-1}) - \mathrm{tr}(s_{k-1}^T y_{k-1})}, & \text{if } \mathrm{tr}(s_{k-1}^T y_{k-1}) < 0.25\mathrm{tr}(s_{k-1}^T H_{k,0}^{-1} s_{k-1}); \\ 1, & \text{otherwise.} \end{cases}$$

We set $H_{k,0} = (1/\delta)I$ for some $\delta > 0$. Then we can define B_k as follows:

$$\begin{cases} B_k = B_{k,p}, \\ B_{k,i} = B_{k,i-1} - \frac{B_{k,i-1} s_j s_j^T B_{k,i-1}}{\mathrm{tr}(s_j^T B_{k,i-1} s_j)} + \frac{\overline{y}_j \overline{y}_j^T}{\mathrm{tr}(s_j^T \overline{y}_j)}, \\ j = k - (p - i + 1), i = 1, \ldots, p, \\ B_{k,0} = \delta I_n. \end{cases} \tag{3.10}$$

Similar to the proof of [37, Lemma 3.1], we can show that B_k are positive definite matrices for all k. We omit the detail. The inverse of $B_{k,i}$ and B_k are denoted by $H_{k,i}$ and H_k. Thus, it holds that

$$\begin{cases} H_k = H_{k,p}, \\ H_{k,i} = (I - \overline{\rho}_j s_j \overline{y}_j^T) H_{k,i-1} (I - \overline{\rho}_j \overline{y}_j s_j^T) + \overline{\rho}_j s_j s_j^T, \\ j = k - (p - i + 1), \ i = 1, \ldots, p, \\ H_{k,0} = (1/\delta) I_n, \end{cases} \tag{3.11}$$

where $\overline{\rho}_j = 1/\mathrm{tr}(s_j^T \overline{y}_j)$.

The following lemma shows that $\mathbf{B}_k$ and its inverse are all bounded operators. In the proof of the lemma, we will use the notation

$$\varrho := \sup_{X \in \mathcal{M}} \|\nabla f(X)\|. \tag{3.12}$$

Since $\mathcal{M}$ is a compact set in $\mathbb{R}^{n \times r}$, it follows that $\varrho < \infty$. We will also use the notations

$$\kappa_1 = \left(\frac{\upsilon^{2p} - 1}{\upsilon^2 - 1} \cdot \frac{4}{\delta} + \frac{\upsilon^{2p}}{\delta} \right)^{-1}, \qquad \kappa_2 = \delta + \frac{4(L + \varrho + \delta)^2}{\delta} p, \tag{3.13}$$

where $\upsilon := (4L + 4\varrho + 5\delta)/\delta$, and δ is as in (3.10).

Lemma 3.1 *Suppose that ∇f is Lipschitz continuous with the Lipschitz constant L. Suppose $\mathbf{B}_k$ and B_k are defined by (3.7) and (3.10) respectively. Denote $\|V\|_{\mathbf{B}_k}^2 := \mathrm{tr}(V^T \mathbf{B}_k[V])$. Then for all k, we have*

$$\kappa_1 \|V\|^2 \le \|V\|_{\mathbf{B}_k}^2 \le \kappa_2 \|V\|^2 \quad \forall V \in \mathrm{T}_{X_k}\mathcal{M}. \tag{3.14}$$

Proof. To prove (3.14), by (3.8), we only need to show $\lambda_{\max}(B_k) \le \kappa_2$ and $\lambda_{\max}(H_k) \le \kappa_1^{-1}$. By (3.9), We have

$$\mathrm{tr}(s_{k-1}^T \bar{y}_{k-1}) = \max\{0.25\delta\|s_{k-1}\|^2, \mathrm{tr}(s_{k-1}^T y_{k-1})\} \ge 0.25\delta\|s_{k-1}\|^2. \tag{3.15}$$

For simplicity, we use the notation $P_k = \mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}}$. From (2.1), it follows that $\|P_k - P_{k-1}\| \le \|X_k - X_{k-1}\|$, Thus, by (2.2), we have

$$\begin{aligned}\|y_{k-1}\| &= \|P_k\nabla f(X_k) - P_{k-1}\nabla f(X_{k-1})\| \\ &\le \|P_k\nabla f(X_k) - P_k\nabla f(X_{k-1})\| + \|P_k\nabla f(X_{k-1}) - P_{k-1}\nabla f(X_{k-1})\| \\ &\le (L+\varrho)\|s_{k-1}\|,\end{aligned}$$

which together with (3.9) implies

$$\|\bar{y}_{k-1}\| \le (L+\varrho+\delta)\|s_{k-1}\| \quad \forall k. \tag{3.16}$$

By (3.10), (3.15) and (3.16), it holds that

$$\begin{aligned}\|B_{k,i}\|_2 &\le \left\|B_{k,i-1} - \frac{B_{k,i-1}s_j s_j^T B_{k,i-1}}{\mathrm{tr}(s_j^T B_{k,i-1}s_j)}\right\|_2 + \left\|\frac{\overline{y}_j\overline{y}_j^T}{\mathrm{tr}(s_j^T\overline{y}_j)}\right\|_2 \\ &\le \|B_{k,i-1}\|_2 + \frac{(L+\varrho+\delta)^2\|s_j\|^2}{0.25\delta\|s_j\|^2}, \quad j = k-(p-i+1), i = 1,\ldots,p,\end{aligned}$$

which together with $B_{k,0} = \delta I$ implies $\lambda_{\max}(B_k) \le \kappa_2$.

Recall that $\overline{\rho}_j = 1/\mathrm{tr}(s_j^T\overline{y}_j)$. By (3.16) again, we have $\overline{\rho}_j \left\|s_j\overline{y}_j^T\right\| \le 4(L+\varrho+\delta)/\delta$. Thus, by (3.11), (3.15) and (3.16), we have

$$\begin{aligned}\|H_{k,i}\|_2 &\le \|I - \overline{\rho}_j s_j\overline{y}_j^T\|_2 \cdot \|H_{k,i-1}\|_2 \cdot \|I - \overline{\rho}_j\overline{y}_j s_j^T\|_2 + \left\|\overline{\rho}_j s_j s_j^T\right\|_2 \\ &\le (1+\overline{\rho}_j\|s_j\overline{y}_j^T\|_2)^2\|H_{k,i-1}\|_2 + 4/\delta \\ &\le (1+4(L+\varrho+\delta)/\delta)^2\|H_{k,i-1}\|_2 + 4/\delta, \quad j = k-(p-i+1), i = 1,\ldots,p.\end{aligned}$$

By the above inequality, it is easy to prove $\lambda_{\max}(H_k) \le \kappa_1^{-1}$ by induction. We omit the detail.

□

Lemma 3.1 shows that $\{\mathbf{B}_k\}$ are uniformly positive definite if $\mathbf{B}_k$ is generated by (3.7). In the following, we show how to solve the subproblem (3.1). Let Λ be the Lagrange multiplier for the constraint (3.2). Then $\Lambda \in \mathbb{S}^r$, where $\mathbb{S}^r$ denotes the set of symmetric $r\times r$ matrices. The Lagrangian function for (3.1) is

$$\begin{aligned}\mathcal{L}_k(V,\Lambda) &= \langle\nabla f(X_k), V\rangle + \frac{1}{2}\|V\|^2_{\mathbf{B}_k} + h(X_k+V) - \langle\mathcal{A}_k^*(\Lambda), V\rangle \\ &= \langle\nabla f(X_k) - \mathcal{A}_k^*(\Lambda), V\rangle + \frac{1}{2}\mathrm{tr}(V^T(\mathrm{diag}B_k)V) + h(X_k+V), \end{aligned}\tag{3.17}$$

where $\mathcal{A}_k^*$ denotes the adjoint operator of $\mathcal{A}_k$. For a fixed Λ, $\mathcal{L}_k(V,\Lambda)$ is a strongly convex function of V since B_k is a positive definite matrix. We use $V(\Lambda)$ to denote the unique minimum of $\min_V \mathcal{L}_k(V,\Lambda)$. By (1.4) and (3.17), we have

$$\begin{aligned}V(\Lambda) &= \mathrm{prox}_h^{\mathbf{B}_k}\left(X_k - (\mathrm{diag}H_k)(\nabla f(X_k) - \mathcal{A}_k^*(\Lambda))\right) - X_k \\ &= \mathrm{prox}_h^{\mathbf{B}_k}(B(\Lambda)) - X_k,\end{aligned}\tag{3.18}$$

where $B(\Lambda) := X_k - (\mathrm{diag}H_k)(\nabla f(X_k) - \mathcal{A}_k^*(\Lambda))$ and H_k is the inverse of B_k. Substituting (3.18) into (3.2) yields

$$E(\Lambda) \equiv \mathcal{A}_k(V(\Lambda)) = V(\Lambda)^T X_k + X_k^T V(\Lambda) = 0. \tag{3.19}$$

Similar to the discussion at [6, p.221], we can prove that $E(\Lambda)$ is a monotone and Lipschitz continuous operator on $\mathbb{S}^r$. Then the adaptive regularized semismooth Newton (ASSN) method can be used to solve (3.19). We give a brief description in the following.

The vectorization of (3.19) can be written as

$$\begin{aligned}\mathrm{vec}(E(\Lambda)) \;=\; &(K_{rr}+I_{r^2})(I_r\otimes X_k^T)\big[\mathrm{prox}_h^{\mathbf{B}_k}(\mathrm{vec}(X_k-(\mathrm{diag}H_k)\nabla f(X_k))\\ &\qquad +2(I_r\otimes((\mathrm{diag}H_k)X_k))\mathrm{vec}(\Lambda))-\mathrm{vec}(X_k)\big],\end{aligned}$$

where K_{nr} and K_{rr} are comutation matrices. Define

$$\mathcal{G}(\mathrm{vec}(\Lambda)) \;:=\; 2(K_{rr}+I_{r^2})(I_r\otimes X_k^T)\mathcal{J}(y)|_{y=\mathrm{vec}(B(\Lambda))}(I_r\otimes((\mathrm{diag}H_k)X_k)), \quad (3.20)$$

where $\mathcal{J}(y)$ is the generalized Jacobian of $\mathrm{prox}_h^{\mathbf{B}_k}(y)$. Then, we know that $\mathcal{G}(\mathrm{vec}(\Lambda))\in\partial\mathrm{vec}(E(\mathrm{vec}(\Lambda)))$.

Denote $\overline{\mathrm{vec}}(\Lambda)$ as the vectorization of the lower triangular part of Λ. Then there exists a duplication matrix $U_r\in\mathbb{R}^{r^2\times\frac{1}{2}r(r+1)}$ such that $\overline{\mathrm{vec}}(\Lambda)=U_r^+\mathrm{vec}(\Lambda)$, where $U_r^+=(U_r^TU_r)^{-1}U_r$ is the Moore-Penrose inverse of U_r. By (3.20), the generalized Jacobian of $\overline{\mathrm{vec}}(E(U_r\overline{\mathrm{vec}}(\Lambda)))$ can be written as

$$\begin{aligned}\mathcal{G}(\overline{\mathrm{vec}}(\Lambda)) \;&=\; U_r^+\mathcal{G}(\mathrm{vec}(\Lambda))U_r\\ &=\; 4U_r^+(I_r\otimes X_k^T)\mathcal{J}(y)|_{y=\mathrm{vec}(B(\Lambda))}(I_r\otimes((\mathrm{diag}H_k)X_k))U_r.\end{aligned}$$

At the current iterate Λ_l, to get the Newton direction d_l, we can apply the conjugate gradient method to solve the following equation

$$(\mathcal{G}(\overline{\mathrm{vec}}(\Lambda_l))+\eta I)d=-\overline{\mathrm{vec}}(E(\Lambda_l)), \quad (3.21)$$

where $\eta>0$ is a regularization parameter. Then, we use the same strategy as that in [40] to obtain the next iterate Λ_{l+1}. For more details, we refer the reader to [6, 40].

4 Convergence Analysis of ManPQN Algorithm

In this section, we study the convergence properties of the ManPQN algorithm. Under mild assumptions, we prove the global convergence of Algorithm 1. We also analyze the local convergence rate of the ManPQN method. It is proved that the iterates of the algorithms converge locally linearly to the nondegenerate local minimum point.

4.1 Global Convergence

First, we give the following assumption which is required in the rest of the paper.

Assumption 4.1 *$f:\mathbb{R}^{n\times r}\to\mathbb{R}$ is smooth, and ∇f is Lipschitz continuous with Lipschitz constant L; $h:\mathbb{R}^{n\times r}\to\mathbb{R}$ is a convex but nonsmooth function, and h is Lipschitz continuous with Lipschitz constant L_h.*

Given X_k, we denote the objective function of (3.1) by ϕ_k, that is

$$\phi_k(V):=\langle\nabla f(X_k),V\rangle+\frac{1}{2}\|V\|_{\mathcal{B}_k}^2+h(X_k+V), \quad (4.1)$$

where $\mathcal{B}_k$ is a linear operator on $\mathbb{R}^{n\times r}$ satisfying $\mathcal{B}_k \mathrm{T}_{X_k}\mathcal{M} \subseteq \mathrm{T}_{X_k}\mathcal{M}$. In our convergence analysis, the only requirement of $\mathcal{B}_k$ is that it satisfies (3.14), that is, $\kappa_1\|V\|^2 \leq \|V\|^2_{\mathcal{B}_k} \leq \kappa_2\|V\|^2$ for all $V \in \mathrm{T}_{X_k}\mathcal{M}$. This and Assumption 4.1 imply that ϕ_k is a convex function on $\mathrm{T}_{X_k}\mathcal{M}$.

Since $V_k = \arg\min_{V\in \mathrm{T}_{X_k}\mathcal{M}} \phi_k(V)$, by (2.2), we have

$$0 \in \mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}}\partial\phi_k(V_k) = \mathrm{grad} f(X_k) + \mathcal{B}_k[V_k] + \mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}}\partial h(X_k + V_k). \tag{4.2}$$

If $V_k = 0$, then X_k satisfies (2.6), and therefore is a stationary point of (1.1); If $V_k \neq 0$, the following result shows that V_k is a descent direction of ϕ_k. The proof is similar to that of [6, Lemma 5.1]. We give a proof for completeness.

Lemma 4.1 *Suppose Assumption 4.1 holds. For any $\alpha \in [0, 1]$, it holds that*

$$\phi_k(\alpha V_k) - \phi_k(0) \leq \frac{\alpha(\alpha-2)}{2}\|V_k\|^2_{\mathcal{B}_k}. \tag{4.3}$$

Proof. By (4.2), there exists $\xi \in \partial h(X_k + V_k)$ such that $\mathrm{grad} f(X_k) + \mathcal{B}_k[V_k] + \mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}}\xi = 0$. From $\xi + \mathrm{grad} f(X_k) \in \partial\left(\phi_k - \frac{1}{2}\|\cdot\|^2_{\mathcal{B}_k}\right)(V_k)$, it follows that

$$\begin{aligned}\phi_k(0) - \phi_k(V_k) &\geq \langle \mathrm{grad} f(X_k) + \xi, -V_k\rangle - \frac{1}{2}\|V\|^2_{\mathcal{B}_k} \\ &= \langle \mathrm{grad} f(X_k) + \mathrm{Proj}_{\mathrm{T}_{X_k}\mathcal{M}}\xi + \mathcal{B}_k[V], -V_k\rangle + \frac{1}{2}\|V_k\|^2_{\mathcal{B}_k} \\ &= \frac{1}{2}\|V_k\|^2_{\mathcal{B}_k}. \end{aligned}\tag{4.4}$$

Since h is a convex function, for all $0 \leq \alpha \leq 1$, we have

$$h(X_k + \alpha V_k) - h(X_k) \leq \alpha(h(X_k + V_k) - h(X_k)). \tag{4.5}$$

Combining (4.4) and (4.5) yields

$$\begin{aligned}\phi_k(\alpha V_k) - \phi_k(0) &= \langle \nabla f(X_k), \alpha V_k\rangle + \frac{1}{2}\|\alpha V_k\|^2_{\mathcal{B}_k} + h(X_k + \alpha V_k) - h(X_k) \\ &\leq \alpha\left(\langle \nabla f(X_k), V_k\rangle + \frac{\alpha}{2}\|V_k\|^2_{\mathcal{B}_k} + h(X_k + V_k)) - h(X_k)\right) \\ &= \alpha(\phi_k(V_k) - \phi_k(0) + \frac{\alpha-1}{2}\|V_k\|^2_{\mathcal{B}_k}) \\ &\leq \frac{\alpha(\alpha-2)}{2}\|V_k\|^2_{\mathcal{B}_k}. \end{aligned}$$

The assertion holds. $\square$

In the rest of the paper, we use F_k to denote $F(X_k)$. By the definition of $l(k)$ (see (3.4)), we have $F_{l(k)} = \max_{\max\{k-m,0\}\leq j\leq k} F_j$. We will use the notation

$$\overline{\alpha} := \min\{1, \frac{(2-\sigma)\kappa_1}{2(\varrho M_2 + \frac{1}{2}LM_1^2 + L_h M_2)}\}, \tag{4.6}$$

where L and L_h are Lipschitz constants, M_1, M_2, σ, ϱ and κ_1 are parameters in (2.4), (2.5), (3.3), (3.12) and (3.13) respectively.

Lemma 4.2 *Suppose Assumption 4.1 holds, and $\mathcal{B}_k$ satisfies (3.14). Let α_k be the stepsize of the k-th iteration of Algorithm 1. Then $\alpha_k \geq \gamma\bar{\alpha}$, where $\overline{\alpha}$ is defined by (4.6) and γ is the parameter of Algorithm 1. Moreover,*

$$F_{k+1} - F_{l(k)} \leq -\frac{1}{2}\sigma\alpha_k \|V_k\|^2_{\mathcal{B}_k}.$$

Proof. Since ∇f is Lipschitz continuous with constant L, for any $\alpha > 0$, we have

$$\begin{aligned}
& f(\mathbf{R}_{X_k}(\alpha V_k)) \\
\leq\ & f(X_k) + \langle \nabla f(X_k), \mathbf{R}_{X_k}(\alpha V_k) - X_k \rangle + \frac{L}{2}\|\mathbf{R}_{X_k}(\alpha V_k) - X_k\|^2 \\
\leq\ & f(X_k) + \langle \nabla f(X_k), \mathbf{R}_{X_k}(\alpha V_k) - (X_k + \alpha V_k)\rangle + \langle \nabla f(X_k), \alpha V_k\rangle + \frac{1}{2}LM_1^2\|\alpha V_k\|^2, \\
\leq\ & f(X_k) + \langle \nabla f(X_k), \alpha V_k\rangle + (\varrho M_2 + \frac{1}{2}LM_1^2)\|\alpha V_k\|^2 \\
=\ & f(X_k) + \langle \nabla f(X_k), \alpha V_k\rangle + c_1\|\alpha V_k\|^2,
\end{aligned} \tag{4.7}$$

where $c_1 := \varrho M_2 + \frac{1}{2}LM_1^2$, the second inequality follows from (2.4), and the third inequality follows from (2.5) and (3.12).

By Assumption 4.1 and (2.5), it holds that

$$\begin{aligned}
h(\mathbf{R}_{X_k}(\alpha V_k)) - h(X_k + \alpha V_k) \ &\leq\ L_h\|\mathbf{R}_{X_k}(\alpha V_k) - X_k - \alpha V_k\| \\
&\leq\ L_h M_2\|\alpha V_k\|^2.
\end{aligned} \tag{4.8}$$

Combining (4.7) and (4.8) yields

$$\begin{aligned}
& F(\mathbf{R}_{X_k}(\alpha V_k)) \\
\leq\ & f(X_k) + \langle \nabla f(X_k), \alpha V_k\rangle + (c_1 + L_h M_2)\|\alpha V_k\|^2 + h(X_k + \alpha V_k) \qquad (4.9) \\
=\ & f(X_k) + \phi_k(\alpha V_k) + c_2\|\alpha V_k\|^2 - \frac{1}{2}\|\alpha V_k\|^2_{\mathcal{B}_k} \\
\leq\ & F_{l(k)} + \phi_k(\alpha V_k) - \phi_k(0) + (c_2\kappa_1^{-1} - \frac{1}{2})\|\alpha V_k\|^2_{\mathcal{B}_k}, \qquad (4.10)
\end{aligned}$$

where $c_2 := c_1 + L_h M_2 = \varrho M_2 + \frac{1}{2}LM_1^2 + L_h M_2$.

For any $0 < \alpha \leq 1$, by (4.3) and (4.10), we have

$$F(\mathbf{R}_{X_k}(\alpha V_k)) \leq F_{l(k)} + (c_2\kappa_1^{-1} - \alpha^{-1})\alpha^2\|V_k\|^2_{\mathcal{B}_k}.$$

Thus, if $0 < \alpha \leq \overline{\alpha}$, it holds that

$$F(\mathbf{R}_{X_k}(\alpha V_k)) - F_{l(k)} \leq (c_2\kappa_1^{-1}\overline{\alpha} - 1)\alpha\|V_k\|^2_{\mathcal{B}_k} \leq -\frac{1}{2}\sigma\alpha\|V_k\|^2_{\mathcal{B}_k}. \tag{4.11}$$

By steps 8-11 of Algorithm 1, we conclude that $\alpha_k \geq \gamma\overline{\alpha}$. Substituting $\alpha = \alpha_k$ into (4.11) yields

$$F_{k+1} - F_{l(k)} \ =\ F(\mathbf{R}_{X_k}(\alpha_k V_k)) - F_{l(k)} \leq -\frac{1}{2}\sigma\alpha_k\|V_k\|^2_{\mathcal{B}_k}. \tag{4.12}$$

The proof is complete. □

Now we are in a position to present the main result of this section, the global convergence of Algorithm 1.

Theorem 4.1 *Suppose Assumption 4.1 holds, and $\mathcal{B}_k$ satisfies (3.14). Then all accumulation points of $\{X_k\}$ are stationary points of problem (1.1).*

Proof. Let X^* be an accumulation point of sequence $\{X_k\}$. We need to prove that X^* satisfies (2.6). By Lemma 4.2, we have that

$$\begin{aligned} F_{l(k+1)} &= \max_{0\le j\le \min\{m,k+1\}} F_{k+1-j} \\ &= \max\{F_{k+1}, \max_{0\le j\le \min\{m-1,k\}} F_{k-j}\} \\ &\le \max\{F_{l(k)} - \frac{1}{2}\sigma\alpha_k\|V_k\|^2_{\mathcal{B}_k}, F_{l(k)}\}, \text{ by}(4.12) \\ &\le F_{l(k)}, \end{aligned}$$

which implies that $\{F_{l(k)}\}_k$ is a monotone nonincreasing sequence. Since $\mathcal{M}$ is compact, F_k is bounded below. Thus, there exists a scalar F^* such that

$$\lim_{k\to\infty} F_{l(k)} = F^*. \tag{4.13}$$

By (3.14) and (4.12), we have

$$\begin{aligned} F_{l(k)} &\le F_{l(l(k)-1)} - \frac{1}{2}\sigma\alpha_{l(k)-1}\|V_{l(k)-1}\|^2_{\mathcal{B}_{l(k)-1}} \\ &\le F_{l(l(k)-1)} - \frac{1}{2}\sigma\kappa_1\alpha_{l(k)-1}\|V_{l(k)-1}\|^2. \end{aligned}$$

From Lemma 4.2, it follows that $\alpha_k \ge \gamma\overline{\alpha}$ for all k, Thus, we must have

$$\lim_{k\to\infty} V_{l(k)-1} = 0.$$

Combining it with (4.13) yields that

$$\begin{aligned} \lim_{k\to\infty} F_{l(k)-1} &= \lim_{k\to\infty} F(\mathbf{R}_{X_{l(k)-1}}(\alpha_{l(k)-1}V_{l(k)-1})) \\ &= \lim_{k\to\infty} F(X_{l(k)}) = F^*. \end{aligned}$$

For all $1 \le j \le m$, we can prove by induction that

$$\lim_{k\to\infty} V_{l(k)-j} = 0, \text{ and } \lim_{k\to\infty} F_{l(k)-j} = F^*. \tag{4.14}$$

The proof is similar to that of [39, Theorem 1], and so we omit it.

For any k, there exists an integer $1 \le j(k) \le m$ such that $k = l(k+m) - j(k)$, which together with (4.14) implies

$$\lim_{k\to\infty} V_k = \lim_{k\to\infty} V_{l(k+m)-j(k)} = 0, \tag{4.15}$$

and

$$\lim_{k\to\infty} F_k = \lim_{k\to\infty} F_{l(k+m)-j(k)} = F^*. \tag{4.16}$$

By (4.15) and (4.2), we can deduce that X^* satisfies (2.6), which completes the proof. □

In the following, we introduce the definition of an ϵ-stationary point of the problem (1.1).

Definition 4.1 (ϵ-stationary point [6]) Given $\epsilon > 0$ and X_k generated by Algorithm 1, we say that $X_k \in \mathcal{M}$ is an ϵ-stationary point of (1.1) if the solution V_k to (3.1) satisfies $\|V_k\|_F \le \epsilon$.

In Algorithm 1, we use $\|V_k\|_F \le \epsilon$ as our stopping criterion. Similar to [6, Theorem 5.5], we give the iteration complexity analysis of Algorithm 1.

Corollary 4.1 *Algorithm 1 will find an ϵ-stationary point in at most $m\lceil 2(F(X_0) - F^*)/(\sigma\kappa_1\gamma\overline{\alpha}\epsilon^2)\rceil$ iterations, where m, σ, κ_1, $\overline{\alpha}$ are defined in (3.3), (3.13), (4.6) respectively and F^* is the optimal value of (1.1).*

Proof. Similar to the proof of Theorem 3.2 in [7], we can obtain that for $j \geq 0$,

$$\begin{aligned} F(X_{l((j+1)m)}) - F(X_{l(jm)}) &\leq \max_{0\leq i\leq m-1}\{-\frac{1}{2}\sigma\alpha_{jm+i}\|V_{jm+i}\|^2_{\mathcal{B}_{jm+i}}\} \\ &\leq \max_{0\leq i\leq m-1}\{-\frac{1}{2}\sigma\kappa_1\alpha_{jm+i}\|V_{jm+i}\|^2_F\}. \end{aligned} \tag{4.17}$$

Given $K > 0$, suppose that after mK iterations, Algorithm 1 does not terminate, which means that $\|V_k\|^2_{\mathcal{B}_k} > \kappa_1\epsilon^2$ for any $0 \leq k \leq Km - 1$. It follows that

$$\begin{aligned} F(X_0) - F^* &\geq F(X_{l(0)}) - F(X_{l(Km)}) \\ &\geq \sum_{j=0}^{K-1} \min_{0\leq i\leq m-1}\{\frac{1}{2}\sigma\kappa_1\alpha_{jm+i}\|V_{jm+i}\|^2_F\} > \frac{1}{2}\sigma\kappa_1\epsilon^2 K\gamma\overline{\alpha}, \end{aligned}$$

where the second inequality follows from (4.17). Thus, Algorithm 1 will return an ϵ-stationary point in at most $m\lceil 2(F(X_0) - F^*)/(\sigma\kappa_1\gamma\overline{\alpha}\epsilon^2)\rceil$ iterations. □

4.2 Locally Linear Convergence

The objective of this subsection is to show that Algorithm 1 has a local linear convergence rate around the nondegenerate local minimum point. Let $\{X_k\}$ be the sequence of iterates generated by Algorithm 1 and $\overline{X}^*$ be any accumulation point of $\{X_k\}$. By (4.16), we know that

$$F(\overline{X}^*) = F^*, \tag{4.18}$$

where F^* is the scalar in (4.16).

We need the following assumption before presenting our main results.

Assumption 4.2 *The function f is twice continuously differentiable. The sequence $\{X_k\}$ has an accumulation point X^* such that*

$$\lambda_{\min}(\mathrm{Hess}(f \circ \mathbf{R}_{X^*})(0_{X^*})) \geq \widetilde{\eta}, \tag{4.19}$$

where $\widetilde{\eta} > 5L_hM_2$.

The constant M_2 is defined in (2.5). We should point out Assumption 4.2 is not a strong condition. In some typical applications, $h(X) = \mu\|X\|_1$. We can see that if

$$\mu < \frac{1}{5M_2}\lambda_{\min}(\mathrm{Hess}(f \circ \mathbf{R}_{X^*})(0_{X^*})),$$

then (4.19) will be satisfied .

Definition 4.2 (ς-strongly convex function [27, Definition 2.1.3]) The function $g(x)$ is said to be a ς-strongly convex function if $g(x) - \frac{1}{2}\varsigma\|x\|^2$ is convex, where ς is called the convexity parameter of g.

By the definition, if $g(x)$ is strongly convex with parameter ς, it is easy to prove

$$g(x) - g(x^*) \geq \frac{\varsigma}{2}\|x - x^*\|^2 \quad \forall x, \tag{4.20}$$

where x^* is the unique minimizer of g. We also use the following property of strongly convex functions. If g is twice continuously differentiable, then

$$\lambda_{\min}(\nabla^2 g(x)) \geq \varsigma, \quad \forall x \Longleftrightarrow g \text{ is strongly convex with parameter } \varsigma. \tag{4.21}$$

For a proof, see [27].

To establish the main results, we need some preparing results.

Lemma 4.3 *Suppose Assumptions 4.1 and 4.2 hold. Let X^* be the accumulation point satisfying (4.19). Then there exists a neighbourhood $\mathcal{U}_{X^*}$ of X^* and $\epsilon > 0$ such that:*

(1) *For all $X \in \mathcal{U}_{X^*}$, $f \circ \mathbf{R}_X$ is a convex function on the set $\{\xi \in \mathrm{T}_X\mathcal{M} : \|\xi\| < \epsilon\}$.*

(2) *For all $X \in \mathcal{U}_{X^*}$,*

$$F(X) - F(X^*) \geq \eta\|\mathbf{R}_X^{-1}(X^*)\|^2, \tag{4.22}$$

for some $\eta > L_h M_2$.

Proof. (1). Since f is twice continuously differentiable and $\mathbf{R}$ is smooth, $\lambda_{\min}(\mathrm{Hess}(f \circ \mathbf{R}))$ is a continuous function of $\mathrm{T}\mathcal{M}$, which together with (4.19) implies that there exists a neighbourhood $\mathcal{U}_{X^*}$ of X^* and $\epsilon > 0$ such that

$$\lambda_{\min}(\mathrm{Hess}(f \circ \mathbf{R}_X)(\xi_X)) > \frac{9}{10}\widetilde{\eta}, \tag{4.23}$$

for all $X \in \mathcal{U}_{X^*}$ and all $\xi_X \in \{\xi \in \mathrm{T}_X\mathcal{M} : \|\xi\| < \epsilon\}$. Thus $f \circ \mathbf{R}_X$ is a convex function on $\{\xi \in \mathrm{T}_X\mathcal{M} : \|\xi\| < \epsilon\}$.

(2). By (4.21) and (4.23), we know that $f(\mathbf{R}_{X^*}(\xi))$ is $9\widetilde{\eta}/10$-strongly convex on $\mathcal{D}_{X^*}$ where $\mathcal{D}_{X^*} := \{\xi \in \mathrm{T}_{X^*}\mathcal{M} : \|\xi\| < \epsilon\}$.

Define $\Gamma : \mathrm{T}_{X^*}\mathcal{M} \to \mathbb{R}$ by $\Gamma(\xi) = f(\mathbf{R}_{X^*}(\xi)) + h(X^* + \xi)$. Then Γ is also $9\widetilde{\eta}/10$-strongly convex on $\mathcal{D}_{X^*}$. By (2.3), we have

$$\begin{aligned}\partial\Gamma(0_{X^*}) &= \mathrm{Proj}_{\mathrm{T}_{X^*}\mathcal{M}}\big(\mathrm{D}\mathbf{R}_{X^*}(0_{X^*})\nabla f(X^*) + \partial h(X^*)\big)\\ &= \mathrm{grad} f(X^*) + \mathrm{Proj}_{\mathrm{T}_{X^*}\mathcal{M}}(\partial h(X^*)),\end{aligned}$$

which together with (2.6) implies $0 \in \partial\Gamma(0_{X^*})$. Thus 0_{X^*} is the unique minimizer of Γ in $\mathcal{D}_{X^*}$. By (4.20), we have

$$\Gamma(\xi) - F(X^*) = \Gamma(\xi) - \Gamma(0_{X^*}) \geq \frac{9}{20}\widetilde{\eta}\|\xi\|^2 \quad \forall \xi \in \mathcal{D}_{X^*}. \tag{4.24}$$

Let $\varphi(\xi) = h(\mathbf{R}_{X^*}(\xi)) - h(X^* + \xi)$. From (2.5) and Assumption 4.1, it holds that

$$|\varphi(\xi)| \leq L_h M_2\|\xi\|^2 \quad \forall \xi \in \mathcal{D}_{X^*}. \tag{4.25}$$

By (4.24) and (4.25), we can deduce that

$$\begin{aligned}F(\mathbf{R}_{X^*}(\xi)) - F(X^*) &= f(\mathbf{R}_{X^*}(\xi)) + h(\mathbf{R}_{X^*}(\xi)) - F(X^*)\\ &= \Gamma(\xi) + \varphi(\xi) - \Gamma(0_{X^*})\end{aligned}$$

$$\geq (\frac{9}{20}\widetilde{\eta} - L_h M_2)\|\xi\|^2. \tag{4.26}$$

Substituting $\xi = \mathbf{R}_{X^*}^{-1}(X)$ into (4.26) yields

$$F(X) - F(X^*) \geq \frac{1}{4}\widetilde{\eta}\|\mathbf{R}_{X^*}^{-1}(X)\|^2. \tag{4.27}$$

By [33, Lemma 6], for any $\varepsilon > 0$, there exists a neighbourhood $\mathcal{U}_{X^*}$ of X^* and $\varepsilon' > 0$ such that for all $X \in \mathcal{U}_{X^*}$ and $V, W \in \mathrm{T}_X\mathcal{M}$ with $\|V\|, \|W\| < \varepsilon'$,

$$(1-\varepsilon)\|V - W\| \leq \mathrm{dist}(\mathbf{R}_X(V), \mathbf{R}_X(W)) \leq (1+\varepsilon)\|V - W\|. \tag{4.28}$$

Assume that $\mathcal{U}_{X^*}$ is small enough such that $\|\mathbf{R}_{X^*}^{-1}(X)\| < \varepsilon'$ and $\|\mathbf{R}_X^{-1}(X^*)\| < \varepsilon'$ for any $X \in \mathcal{U}_{X^*}$. By (4.28), we have

$$(1-\varepsilon)\|\mathbf{R}_X^{-1}(X^*)\| \leq \|X - X^*\| \leq (1+\varepsilon)\|\mathbf{R}_{X^*}^{-1}(X)\|, \tag{4.29}$$

where the first inequality follows from substituting $V = 0_X$ and $W = \mathbf{R}_X^{-1}(X^*)$ into (4.28), and the second inequality from $V = \mathbf{R}_{X^*}^{-1}(X)$ and $W = 0_{X^*}$ in (4.28). We can choose ε satisfying $\varepsilon < (\sqrt{5}-2)^2$. By (4.27) and (4.29), we have

$$F(X) - F(X^*) \geq \frac{(1-\varepsilon)^2}{4(1+\varepsilon)^2}\widetilde{\eta}\|\mathbf{R}_X^{-1}(X^*)\|^2 \geq \frac{1}{5}\widetilde{\eta}\|\mathbf{R}_X^{-1}(X^*)\|^2 \quad \forall X \in \mathcal{U}_{X^*},$$

which together with $\widetilde{\eta} > 5L_h M_2$ implies (4.22). $\square$

Under the condition of Assumption 4.2, from the following result, we know that the sequence $\{X_k\}$ has only one accumulation point X^*, which is of course the limit of $\{X_k\}$.

Theorem 4.2 *Suppose Assumptions 4.1 and 4.2 hold, and X^* is the accumulation point satisfying (4.19). Then, X_k converges to X^*.*

Proof. Let $\mathcal{U}_{X^*}$ be the neighbourhood of X^* as in Lemma 4.3. Define $\overline{\mathcal{U}}_{X^*} := \{\mathbf{R}_{X^*}(\xi) : \xi \in \mathcal{U}_{X^*}\}$. By (4.22), X^* is the unique minimizer of F in $\overline{\mathcal{U}}_{X^*}$, which together with (4.18) implies that X^* is an isolated accumulation point of $\{X_k\}$. Since $V_k \to 0$ by (4.15), we have

$$\|X_{k+1} - X_k\| = \|\mathbf{R}_{X_k}(\alpha_k V_k) - X_k\| \leq M_1 \alpha_k \|V_k\| \to 0.$$

By [24, Lemma (4.10)], we can obtain that $X_k \to X^*$. $\square$

From Lemma 4.2, we know that the stepsize α_k satisfies $\gamma\overline{\alpha} < \alpha_k \leq 1$ for all $k \geq 0$, where $\overline{\alpha}$ is defined in (4.6). We will use this fact to prove the local linear convergence of Algorithm 1. Our proof is based on a technique used in [19, Theorem 3.4].

Theorem 4.3 *Suppose Assumptions 4.1 and 4.2 hold, X^* is the accumulation point satisfying (4.19), and $\mathcal{B}_k$ satisfies (3.14). Then there exists an integer K, $\mu > 0$ and $\tau \in (0,1)$ such that*

$$F(X_k) - F(X^*) \leq \mu\tau^{k-K}\big(F(X_{l(K)}) - F(X^*)\big), \text{ for all } k > K. \tag{4.30}$$

Proof. Let $\mathcal{U}_{X^*}$ be the neighbourhood of X^* and $\epsilon > 0$ be the constant such that all statements of Lemma 4.3 hold. By Theorem 4.2, there exists an integer $K > 0$ such that $X_k \in \mathcal{U}_{X^*}$ for all $k \geq K$. From $X_k \to X^*$, it follows that $\|\mathbf{R}_{X_k}^{-1}(X^*)\| \to 0$. Without loss of generality, assume that $\|\mathbf{R}_{X_k}^{-1}(X^*)\| < \epsilon$ for all $k \geq K$.

We separate our proof into three parts.

Part (1). By (4.9), we have

$$F_{k+1} \leq f(X_k) + \langle \nabla f(X_k), \alpha_k V_k \rangle + c_2 \|\alpha_k V_k\|^2 + h(X_k + \alpha_k V_k), \tag{4.31}$$

where $c_2 := \varrho M_2 + \frac{1}{2} L M_1^2 + L_h M_2$. Since h is convex and $\alpha_k \in (0, 1]$, it holds that

$$h(X_k + \alpha_k V_k) \;\leq\; \alpha_k h(X_k + V_k) + (1 - \alpha_k) h(X_k).$$

Combining it with (4.31) yields

$$F_{k+1} \leq (1 - \alpha_k) F_k + \alpha_k (f(X_k) + \phi_k(V_k)) + (c_2 - \frac{\kappa_1}{2\alpha_k}) \|\alpha_k V_k\|^2. \tag{4.32}$$

From Lemma 4.3, we know that $f \circ \mathbf{R}_X$ is convex on the set $\{V \in \mathrm{T}_X \mathcal{M} : \|V\| < \epsilon\}$. For $V \in \mathrm{T}_X \mathcal{M}$, it holds that $\mathrm{grad}(f \circ \mathbf{R}_X)(0_X) = \mathrm{Proj}_{\mathrm{T}_X \mathcal{M}} \nabla f(X)$. If $\|V\| < \epsilon$, then

$$f(\mathbf{R}_X(V)) - f(X) \geq \langle \nabla f(X), V \rangle. \tag{4.33}$$

By the definition of ϕ_k (see (4.1)) and $V_k = \arg\min_{V \in \mathrm{T}_{X_k} \mathcal{M}} \phi_k(V)$, we have

$$f(X_k) + \phi_k(V_k) \;=\; \min_{V \in T_{X_k} \mathcal{M}} \{f(X_k) + \langle \nabla f(X_k), V \rangle + \frac{1}{2} \|V\|_{\mathcal{B}_k}^2 + h(X_k + V)\}.$$

By combining it with (4.33), for all $k \geq K$ and $\theta \in [0, 1]$, we have

$$\begin{aligned}
& f(X_k) + \phi_k(V_k) \\
\leq\; & \min_{V \in T_{X_k} \mathcal{M}, \|V\| < \epsilon} \{f(\mathbf{R}_{X_k}(V)) + \frac{1}{2} \|V\|_{\mathcal{B}_k}^2 + h(X_k + V)\} \\
\leq\; & \theta f(X^*) + (1 - \theta) F_k + \frac{1}{2} \theta^2 \kappa_2 \|\mathbf{R}_{X_k}^{-1}(X^*)\|^2 + \theta h(X_k + \mathbf{R}_{X_k}^{-1}(X^*)) \\
\leq\; & \theta f(X^*) + (1 - \theta) F_k + \frac{1}{2} \theta^2 \kappa_2 \|\mathbf{R}_{X_k}^{-1}(X^*)\|^2 + \theta h(X^*) + \theta L_h M_2 \|\mathbf{R}_{X_k}^{-1}(X^*)\|^2 \\
\leq\; & \theta F(X^*) + (1 - \theta) F_k + (\frac{1}{2} \theta^2 \kappa_2 + \theta L_h M_2) \|\mathbf{R}_{X_k}^{-1}(X^*)\|^2.
\end{aligned} \tag{4.34}$$

Denote $\Upsilon := (\frac{1}{2} \theta^2 \kappa_2 + \theta L_h M_2) \|\mathbf{R}_{X_k}^{-1}(X^*)\|^2$. From (4.34) and (4.32), it follows that

$$\begin{aligned}
& F_{k+1} \\
\leq\; & (1 - \alpha_k) F_k + \alpha_k \big(\theta F(X^*) + (1 - \theta) F_k + \Upsilon\big) + (c_2 - \frac{\kappa_1}{2\alpha_k}) \|\alpha_k V_k\|^2 \\
\leq\; & F_{l(k)} + \alpha_k \big(\theta (F(X^*) - F_{l(k)}) + \Upsilon\big) + c_2 \|\alpha_k V_k\|^2.
\end{aligned} \tag{4.35}$$

By (4.22), it holds that

$$F_{l(k)} - F(X^*) \geq F_k - F(X^*) \geq \eta \|\mathbf{R}_{X_k}^{-1}(X^*)\|^2. \tag{4.36}$$

Since $\alpha_k \leq 1$, by (4.36) and (4.35), we can deduce that

$$F_{k+1} \leq F_{l(k)} + \big(F_{l(k)} - F(X^*)\big) (\frac{1}{2\eta} \kappa_2 \theta^2 - (1 - \frac{1}{\eta} L_h M_2) \theta) + c_2 \|V_k\|^2. \tag{4.37}$$

Part (2). Now we prove that there exists a $\nu \in (0, 1)$ such that for all $k > K$,

$$F_{k+1} - F(X^*) \leq \nu (F_{l(k)} - F(X^*)). \tag{4.38}$$

Let ω be a positive real number such that

$$\omega < \min\{\frac{2}{\sigma \gamma \overline{\alpha}}, \frac{\kappa_2}{2\eta c_2}, \frac{(\eta - L_h M_2)^2}{2\eta \kappa_2 c_2}\}.$$

Next, we consider two cases of the value of $\|V_k\|^2$.

(i) $\|V_k\|^2 \geq \omega(F_{l(k)} - F(X^*))$. By (4.12), we have

$$\frac{2}{\sigma\gamma\bar{\alpha}}(F_{l(k)} - F_{k+1}) \geq \frac{2}{\sigma\alpha_k}(F_{l(k)} - F_{k+1}) \geq \|V_k\|^2 \geq \omega(F_{l(k)} - F(X^*)).$$

Thus,

$$F_{k+1} - F(X^*) \leq (1 - \frac{\sigma\gamma\bar{\alpha}\omega}{2})(F_{l(k)} - F(X^*)),$$

which implies (4.38).

(ii) $\|V_k\|^2 < \omega(F_{l(k)} - F(X^*))$. Combining it with (4.37) yields

$$F_{k+1} \leq F_{l(k)} + (F_{l(k)} - F(X^*))(\frac{1}{2\eta}\kappa_2\theta^2 - (1 - \frac{1}{\eta}L_h M_2)\theta + c_2\omega).$$

Denote $r_{k+1} := F_{k+1} - F(X^*)$. Then, we have

$$r_{k+1} \leq [1 + \frac{1}{2\eta}\kappa_2\theta^2 - (1 - \frac{1}{\eta}L_h M_2)\theta + c_2\omega] \cdot r_{l(k)}. \tag{4.39}$$

Define $q(\theta) := 1 + \frac{1}{2\eta}\kappa_2\theta^2 - (1 - \frac{1}{\eta}L_h M_2)\theta + c_2\omega$. Let $\theta_{\min} := \arg\min_{0\leq\theta\leq 1} q(\theta)$. Then

$$\theta_{\min} = \min\{1, (\eta - L_h M_2)/\kappa_2\}.$$

Consider the following two cases:

(a) If $\theta_{\min} = 1$, then

$$\frac{1}{2\eta}\kappa_2 \leq \frac{1}{2}(1 - \frac{1}{\eta}L_h M_2),$$

which together with $\omega < \kappa_2/(2\eta c_2)$ implies

$$\begin{aligned} q(\theta_{\min}) &= 1 + [\frac{1}{2\eta}\kappa_2 - (1 - \frac{1}{\eta}L_h M_2)] + c_2\omega \\ &\leq 1 - \frac{1}{2\eta}\kappa_2 + c_2\omega < 1. \end{aligned}$$

(b) Otherwise, $\theta_{\min} = (\eta - L_h M_2)/\kappa_2 < 1$. By $\omega < (\eta - L_h M_2)^2/(2\eta\kappa_2 c_2)$, we have

$$q(\theta_{\min}) = 1 + (-\frac{1}{2\eta\kappa_2}(\eta - L_h M_2)^2 + c_2\omega) < 1.$$

In either case (a) or (b), substituting $\theta_{\min}$ into (4.39), we can see that (4.38) holds.

Part (3). For any $k > K$, there exists an integer $i \geq 1$ such that $(i-1)m < k - K \leq im$ where m is the memory size parameter of the nonmonotone line search in (3.3). By using (4.38) recursively, we have

$$\begin{aligned} r_k &\leq r_{l(k)} \leq \nu r_{l(l(k)-1)} \leq \nu r_{l(k-m)} \leq \dots \\ &\leq \nu^{i-1} r_{l(k-(i-1)m)} \leq \nu^{i-1} r_{l(K)} \leq \nu^{(k-K)/m-1} r_{l(K)}. \end{aligned}$$

Then (4.30) follows from the above inequality by taking

$$\mu := \frac{1}{\nu},\ \tau := \nu^{1/m}.$$

The proof is complete. □

Corollary 4.2 *Suppose the same assumptions hold as in Theorem 4.3. Then there exists an integer K and a constant $C_K > 0$ such that*

$$\|X_k - X^*\| \leq C_K \sqrt{\tau}^k, \text{ for all } k > K, \tag{4.40}$$

where $\tau \in (0, 1)$ is as in Theorem 4.3.

Proof. By (2.4), (4.27) and (4.30), for any $k \geq K$, we have

$$\begin{aligned} \|X_k - X^*\| &\leq M_1 \|\mathbf{R}_{X^*}^{-1}(X_k)\| \\ &\leq 2M_1 \big(\frac{1}{\widetilde{\eta}} (F_k - F^*) \big)^{1/2} \\ &\leq 2M_1 \big(\frac{1}{\widetilde{\eta}} \mu \tau^{-K} (F_{l(K)} - F^*) \big)^{1/2} \sqrt{\tau}^k. \end{aligned}$$

Taking $C_K := 2M_1 \big(\mu \tau^{-K} (F_{l(K)} - F^*) / \widetilde{\eta} \big)^{1/2}$ in the above inequality yields (4.40). □

Remark 1 The constant τ in (4.40) only depends on f and h, while K and C_K depend on more factors. Different initial point X_0 and stepsize α_k may lead to different K and C_K.

5 Numerical Experiments

In this section, we report our numerical experiments comparing our method with Riemannian proximal gradient methods, including ManPG and ManPG-Ada in [6], and NLS-ManPG which equips ManPG with the nonmonotone line search strategy. Our objective is to show the efficiency of the proximal quasi-Newton method for composite optimization problems over the Stiefel Manifold.

Our test problems include the compressed modes (CM) problem, sparse principle component analysis (Sparse PCA), and the joint diagonalization problem with a regularization term. All of these experiments[1] were conducted in MATLAB R2018b on a PC using Windows 10 (64bit) system with Intel Core i5 CPU (2.3GHz) and 8GB memory.

For the stopping criterion, we terminate our algorithm when $\|V_k\|^2 \leq 10^{-8} nr$, where $V_k \in \mathbb{R}^{n \times r}$ is defined by (3.6), or the algorithm reaches the maximum iteration number 30000. For other parameters, the maximum iteration number of the inner loop is set to be 100. In the implementation of ManPQN, we set $m = 10$ and $p = 5$ (for m and p, see (3.3) and (3.11)). The parameters used in ManPG, ManPG-Ada and NLS-ManPG are set to be the default values in [6]. For all the problems, we use the singular value decomposition (SVD) as the retraction mapping in ManPQN, ManPG, ManPG-Ada and NLS-ManPG.

We report the numerical results obtained by solving randomly generated instances. Specifically, we randomly generate 50 instances and record the averaged numerical performance of these instances. Numerical results are shown in several figures and tables. In each figure, **CPU** denotes the CPU time in seconds, **Iter** represents the number of iterations, **Sparsity** denotes the percentage of zeros in the local minimum X^*. In each table, the total number of line search steps and the averaged iteration number of the adaptive regularized semismooth Newton (ASSN) method are reported.

5.1 CM Problem

The compressed modes (CM) problem aims to find sparse solutions of systems of equations in physics, including the Schrödinger equation in quantum mechanics. The

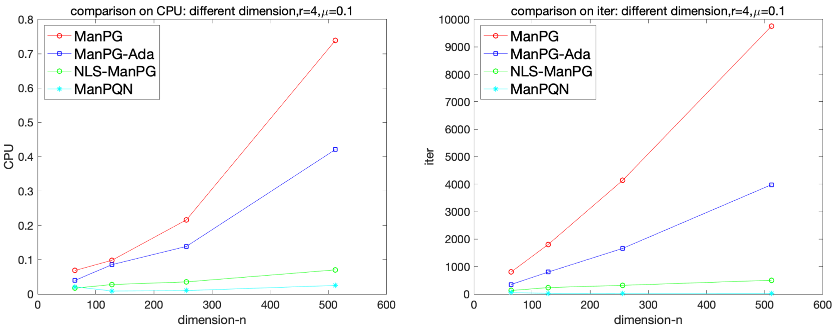
[1] Our MATLAB code is available at https://github.com/QinsiWang2022/ManPQN.

CM problem can be written as

$$\min_{X \in \mathcal{M}} tr(X^T HX) + \mu \|X\|_1, \tag{5.1}$$

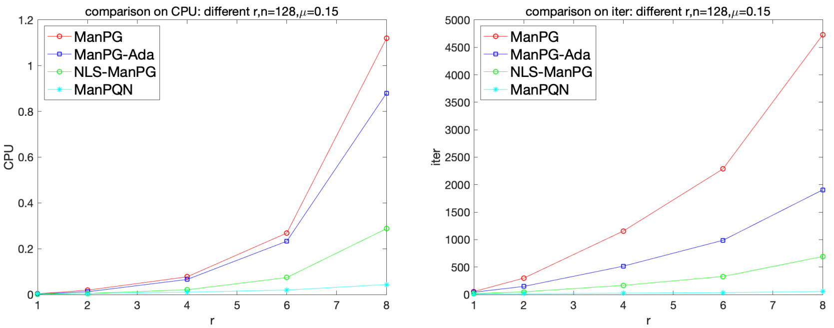
where H is the discretized Schrödinger operator. For details of the CM problem, the reader is referred to [30].

We can observe from Figures 1-3 and Tables 1-3 that the ManPQN method outperforms ManPG, ManPG-Ada and NLS-ManPG, which demonstrates the efficiency of the quasi-Newton strategy used in ManPQN. ManPQN requires less computational time and less iterations than ManPG related methods, especially when n and r are large. From these results, we can see that the quasi-Newton technique can accelerate the proximal gradient method for composite optimization problems over the Stiefel manifold. ManPG related methods can achieve a solution with slightly better sparsity than ManPQN. The reason for this is that we use an approximate quasi-Newton strategy in our method (see (3.6) and (3.7)).



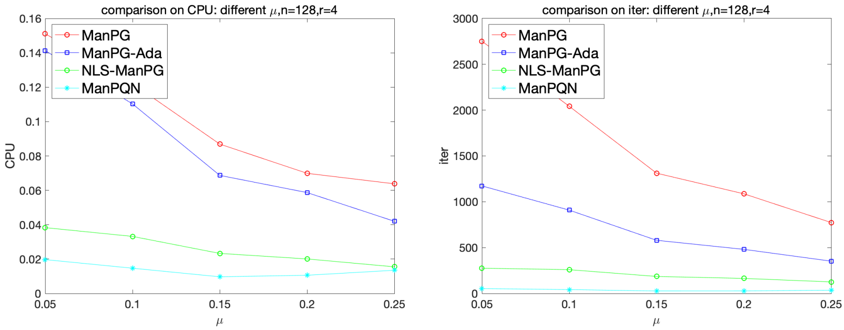
(a) CPU (b) Iter

Fig. 1 Comparison on CM problem, different $n = \{64, 128, 256, 512\}$ with $r = 4$ and $\mu = 0.1$.



(a) CPU (b) Iter

Fig. 2 Comparison on CM problem, different $r = \{1, 2, 4, 6, 8\}$ with $n = 128$ and $\mu = 0.15$.



(a) CPU (b) Iter

Fig. 3 Comparison on CM problem, different $\mu = \{0.05, 0.10, 0.15, 0.20, 0.25\}$ with $n = 128$ and $r = 4$.

Table 1 Comparison on CM problem, different $n = \{64, 128, 256, 512\}$ with $r = 4$ and $\mu = 0.1$.

$n = 64$	Iter	$F(X^*)$	sparsity	CPU time	# line-search	SSN iters
ManPG	800.74	1.424	0.82	0.0685	130.18	0.97
ManPG-Ada	347.80	1.424	0.82	0.0396	130.72	1.26
NLS-ManPG	128.74	1.424	0.82	0.0170	0.16	1.57
ManPQN	56.32	1.432	0.80	0.0213	116.20	5.91
$n = 128$						
ManPG	1808.54	1.885	0.83	0.0985	86.98	0.53
ManPG-Ada	801.16	1.885	0.83	0.0857	566.30	1.07
NLS-ManPG	235.20	1.885	0.83	0.0277	10.12	1.47
ManPQN	22.52	1.890	0.81	0.0088	18.76	4.44
$n = 256$						
ManPG	4141.94	2.489	0.85	0.2162	72.14	0.45
ManPG-Ada	1662.62	2.489	0.85	0.1388	1161.84	0.62
NLS-ManPG	317.06	2.489	0.85	0.0354	19.06	1.33
ManPQN	17.60	2.497	0.84	0.0101	19.00	4.33
$n = 512$						
ManPG	9755.60	3.286	0.87	0.7385	50.66	0.16
ManPG-Ada	3983.06	3.286	0.87	0.4208	2623.98	0.25
NLS-ManPG	501.92	3.286	0.87	0.0704	25.06	0.92
ManPQN	16.54	3.293	0.86	0.0250	15.58	3.28

The total number of line-search steps and the averaged iteration number of the ASSN method are shown in Tables 1-3. ManPQN and NLS-ManPG need less line-search steps than the other methods since they use the nonmonotone line search technique. From Tables 1-3, we can see that the ASSN method in ManPQN needs more iterations than that of the other three methods. The reason for this is that the matrix diagH_k is involved in $\mathcal{G}(\overline{\text{vec}}(\Lambda_l))$ and therefore the condition number of $\mathcal{G}(\overline{\text{vec}}(\Lambda_l))$ becomes larger (see (3.20) and (3.21)).

Table 2 Comparison on CM problem, different $r = \{1, 2, 4, 6, 8\}$ with $n = 128$ and $\mu = 0.15$.

$r = 1$	Iter	$F(X^*)$	sparsity	CPU time	# line-search	SSN iters
ManPG	54.12	0.6513	0.87	0.0034	0.00	0.93
ManPG-Ada	42.20	0.6513	0.87	0.0027	0.00	1.04
NLS-ManPG	16.94	0.6513	0.87	0.0015	0.08	1.12
ManPQN	12.78	0.6603	0.86	0.0039	4.94	2.29
$r = 2$						
ManPG	298.62	1.302	0.86	0.0190	15.04	0.91
ManPG-Ada	147.52	1.302	0.86	0.0120	94.92	1.06
NLS-ManPG	49.04	1.302	0.86	0.0048	1.46	1.24
ManPQN	16.58	1.303	0.86	0.0047	5.24	2.71
$r = 4$						
ManPG	1156.72	2.605	0.86	0.0775	95.56	0.82
ManPG-Ada	516.68	2.605	0.86	0.0659	371.62	1.25
NLS-ManPG	167.60	2.605	0.86	0.0210	5.56	1.65
ManPQN	26.26	2.610	0.85	0.0104	16.86	4.33
$r = 6$						
ManPG	2287.98	3.909	0.85	0.2688	163.14	0.60
ManPG-Ada	987.78	3.909	0.85	0.2330	709.94	1.57
NLS-ManPG	329.22	3.909	0.85	0.0742	8.98	1.79
ManPQN	34.08	3.920	0.84	0.0197	29.38	4.32
$r = 8$						
ManPG	4727.60	5.214	0.84	1.1193	220.42	1.24
ManPG-Ada	1905.44	5.214	0.84	0.8793	1265.02	2.82
NLS-ManPG	692.14	5.214	0.84	0.2878	15.86	2.87
ManPQN	56.56	5.248	0.82	0.0437	51.76	5.96

5.2 Sparse PCA

The sparse PCA model can be formulated as

$$\min_{X \in \mathcal{M}} -tr(X^T A^T A X) + \mu \|X\|_1, \tag{5.2}$$

where $A \in \mathbb{R}^{m \times n}$. We apply ManPQN and ManPG related algorithms to (5.2) and compare their performance. The matrix $A^T A$ is generated by the following two ways. In Section 5.2.1, A is generated by normal distribution. In Section 5.2.2, the matrix $A^T A$ is chosen from real symmetric positive definite matrices in "UF Sparse Matrix Collection" [8].

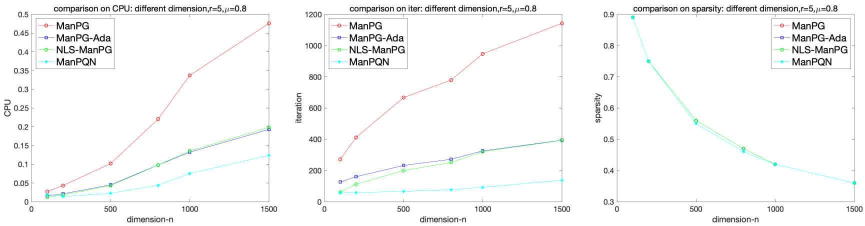
Table 3 Comparison on CM problem, different $\mu = \{0.05, 0.10, 0.15, 0.20, 0.25\}$ with $n = 128$ and $r = 4$.

$\mu = 0.05$	Iter	$F(X^*)$	sparsity	CPU time	# line-search	SSN iters
ManPG	2751.68	1.083	0.76	0.1511	57.08	0.28
ManPG-Ada	1173.30	1.083	0.76	0.1413	710.14	0.85
NLS-ManPG	276.12	1.083	0.76	0.0383	14.76	1.37
ManPQN	53.78	1.111	0.73	0.0197	92.40	5.35
$\mu = 0.10$						
ManPG	2041.74	1.885	0.83	0.1221	79.06	0.52
ManPG-Ada	909.02	1.885	0.83	0.1103	703.42	1.10
NLS-ManPG	259.48	1.885	0.83	0.0332	11.30	1.47
ManPQN	42.04	1.902	0.81	0.0147	37.10	4.27
$\mu = 0.15$						
ManPG	1312.30	2.605	0.86	0.0869	130.72	0.77
ManPG-Ada	580.08	2.605	0.86	0.0687	424.32	1.28
NLS-ManPG	185.76	2.605	0.86	0.0233	5.74	1.63
ManPQN	27.54	2.610	0.85	0.0097	18.36	4.15
$\mu = 0.20$						
ManPG	1087.28	3.278	0.88	0.0699	90.14	0.88
ManPG-Ada	481.32	3.278	0.88	0.0587	365.34	1.35
NLS-ManPG	164.26	3.278	0.88	0.0201	3.88	1.80
ManPQN	28.12	3.280	0.87	0.0106	23.40	4.91
$\mu = 0.25$						
ManPG	773.86	3.916	0.89	0.0638	86.08	0.55
ManPG-Ada	352.60	3.916	0.89	0.0420	211.18	1.00
NLS-ManPG	126.10	3.916	0.89	0.0155	1.70	1.22
ManPQN	36.24	3.920	0.88	0.0135	21.82	3.34

5.2.1 Random generated sparse PCA problem

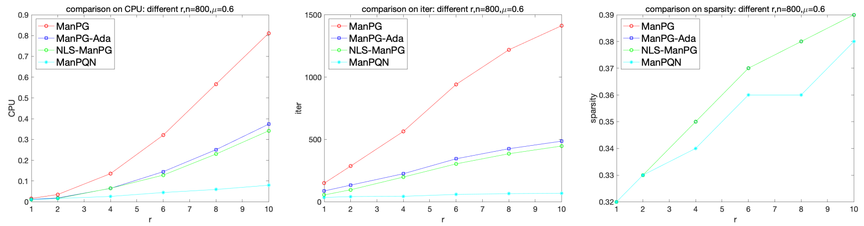
From Figures 4-6 and Tables 4-6, we can see that ManPQN shows better performance than ManPG, ManPG-Ada and NLS-ManPG in terms of CPU time and iteration number, especially when n and r are large. In some cases, ManPG related methods can achieve a solution, the sparsity of which is a little better than that of ManPQN. Taking CPU time, the total number of line search steps and the averaged SSN iteration number in Tables 4-6 into account, we can deduce that the quasi-Newton strategy can have a significant effect on accelerating our method.

Next, we investigate the convergence of $F(X_k)$ generated by ManPQN, ManPG and ManPG-Ada. We select six cases with different n and r, and plot numerical results of these three algorithms in Figures 7-9. We can see that when X_k is close to X^*, $F(X_k)$ converges to $F(X^*)$ approximately at a linear rate, which matches our theoretical results in Theorem 4.3.



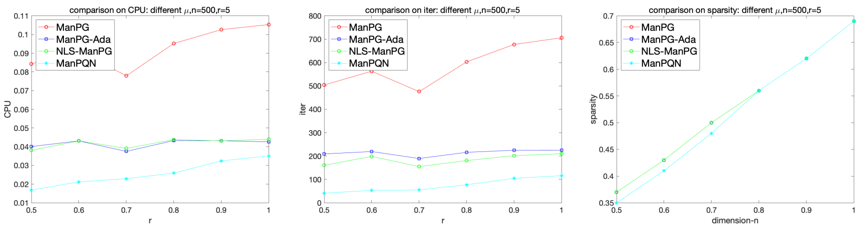
(a) CPU (b) Iter (c) Sparsity

Fig. 4 Comparison on Sparse PCA problem, different $n = \{100, 200, 500, 800, 1000, 1500\}$ with $r = 5$ and $\mu = 0.8$.



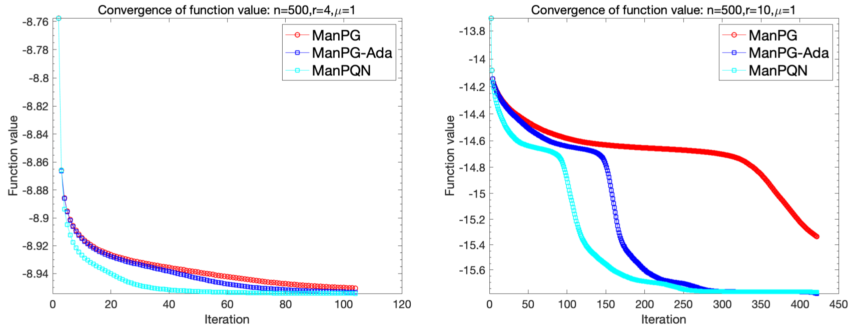
(a) CPU (b) Iter (c) Sparsity

Fig. 5 Comparison on Sparse PCA problem, different $r = \{1, 2, 4, 8, 10\}$ with $n = 800$ and $\mu = 0.6$.



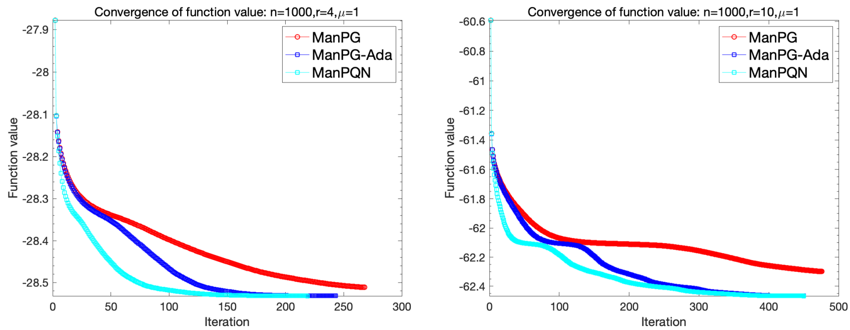
(a) CPU (b) Iter (c) Sparsity

Fig. 6 Comparison on Sparse PCA problem, different $\mu = \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ with $n = 500$ and $r = 5$.



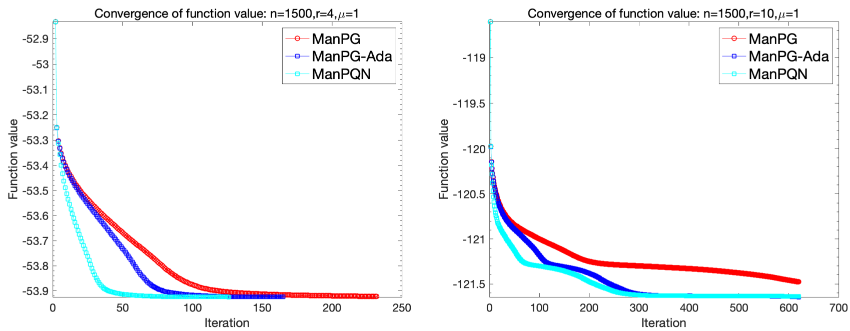
(a) $r = 4$ (b) $r = 10$

Fig. 7 Convergence of function value on Sparse PCA problem with $n = 500, \mu = 1.0$.



(a) $r = 4$ (b) $r = 10$

Fig. 8 Convergence of function value on Sparse PCA problem with $n = 1000, \mu = 1.0$.



(a) $r = 4$ (b) $r = 10$

Fig. 9 Convergence of function value on Sparse PCA problem with $n = 1500, \mu = 1.0$.

Table 4 Comparison on Sparse PCA problem, different $n = \{100, 200, 500, 800, 1000, 1500\}$ with $r = 5$ and $\mu = 0.8$.

$n = 100$	Iter	$F(X^*)$	sparsity	CPU time	# line-search	SSN iters
ManPG	271.02	-2.285	0.89	0.0267	1.86	1.20
ManPG-Ada	126.40	-2.285	0.89	0.0158	0.38	1.49
NLS-ManPG	60.96	-2.285	0.89	0.0125	0.00	2.34
ManPQN	57.64	-2.274	0.89	0.0193	23.94	2.68
$n = 200$						
ManPG	412.32	-5.449	0.75	0.0434	1.94	1.07
ManPG-Ada	160.38	-5.449	0.75	0.0212	7.22	1.33
NLS-ManPG	112.48	-5.449	0.75	0.0180	0.00	1.50
ManPQN	58.72	-5.428	0.75	0.0138	37.92	2.03
$n = 500$						
ManPG	667.08	-20.30	0.56	0.1017	0.00	1.02
ManPG-Ada	232.50	-20.30	0.56	0.0451	48.86	1.22
NLS-ManPG	199.50	-20.30	0.56	0.0432	0.00	1.38
ManPQN	66.34	-20.21	0.55	0.0226	41.76	1.96
$n = 800$						
ManPG	778.82	-39.16	0.47	0.2204	0.00	1.02
ManPG-Ada	272.06	-39.16	0.47	0.0979	70.66	1.19
NLS-ManPG	251.44	-39.16	0.47	0.0978	0.00	1.30
ManPQN	77.32	-39.07	0.46	0.0439	50.14	1.95
$n = 1000$						
ManPG	947.26	-53.23	0.42	0.3370	0.00	1.01
ManPG-Ada	325.62	-53.23	0.42	0.1323	102.98	1.17
NLS-ManPG	320.90	-53.23	0.42	0.1354	0.00	1.24
ManPQN	92.50	-53.17	0.42	0.0754	68.24	1.92
$n = 1500$						
ManPG	1142.60	-89.60	0.36	0.4757	19.04	1.01
ManPG-Ada	394.10	-89.60	0.36	0.1929	150.80	1.12
NLS-ManPG	393.40	-89.60	0.36	0.1981	0.00	1.20
ManPQN	137.94	-89.50	0.36	0.1238	171.00	1.87

5.2.2 Sparse PCA problem from "UF Sparse Matrix Collection"

To further evaluate the performance of ManPQN and ManPG related algorithms, we apply them to (5.2), where $A \in \mathbb{R}^{m\times n}$ is chosen from real matrices in [8]. We choose four matrices named "lpi_klein1", "bcsstk22", "lp_fit1d" and "fidap003" whose dimensions are $(m, n) = (54, 108)$, $(m, n) = (138, 138)$, $(m, n) = (24, 1049)$ and $(m, n) = (1821, 1821)$ respectively. We set $r = 4$, $\mu = 0.2$. Experiments are repeated for 50 times with different random initial point. The numerical results are given in Table 7.

Table 5 Comparison on Sparse PCA problem, different $r = \{1, 2, 4, 8, 10\}$ with $n = 800$ and $\mu = 0.6$.

$r = 1$	Iter	$F(X^*)$	sparsity	CPU time	# line-search	SSN iters
ManPG	149.56	-12.21	0.32	0.0163	0.00	0.78
ManPG-Ada	87.18	-12.21	0.32	0.0117	2.00	0.93
NLS-ManPG	55.10	-12.21	0.33	0.0109	0.00	1.03
ManPQN	35.16	-12.21	0.32	0.0134	34.04	1.26
$r = 2$						
ManPG	286.48	-24.01	0.33	0.0352	93.48	0.99
ManPG-Ada	133.80	-24.01	0.33	0.0185	14.92	1.06
NLS-ManPG	96.58	-24.01	0.33	0.0166	0.02	1.11
ManPQN	42.04	-23.98	0.33	0.0160	39.32	1.63
$r = 4$						
ManPG	564.46	-45.83	0.35	0.1357	4.52	1.02
ManPG-Ada	225.32	-45.83	0.35	0.0647	56.36	1.07
NLS-ManPG	198.54	-45.83	0.35	0.0652	0.00	1.16
ManPQN	44.10	-45.77	0.34	0.0260	31.94	1.83
$r = 6$						
ManPG	941.32	-66.18	0.37	0.3209	0.00	1.01
ManPG-Ada	345.08	-66.18	0.37	0.1451	120.30	1.19
NLS-ManPG	304.30	-66.18	0.37	0.1285	0.00	1.28
ManPQN	59.18	-66.03	0.36	0.0448	39.76	1.96
$r = 8$						
ManPG	1219.04	-85.57	0.38	0.5664	0.00	1.01
ManPG-Ada	425.80	-85.57	0.38	0.2508	171.44	1.32
NLS-ManPG	385.58	-85.57	0.38	0.2298	0.00	1.37
ManPQN	65.56	-85.39	0.36	0.0594	40.82	2.01
$r = 10$						
ManPG	1412.26	-103.59	0.39	0.8113	0.00	1.02
ManPG-Ada	486.46	-103.59	0.39	0.3737	206.70	1.44
NLS-ManPG	447.34	-103.59	0.39	0.3418	0.00	1.41
ManPQN	68.52	-103.40	0.38	0.0799	42.38	2.10

Since $A^T A$ generated by these four matrices are ill-conditioned, all algorithms need more iteration numbers and CPU time to converge. In Table 7, we can observe that ManPQN and NLS-ManPG show much better performance than ManPG and ManPG-Ada in terms of iteration numbers and CPU time. Only for matrix "lpi_klein1", ManPQN needs slightly more CPU time than NLS-ManPG. For the other three matrices, ManPQN outperforms NLS-ManPG.

Table 6 Comparison on Sparse PCA problem, different $\mu = \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ with $n = 500$ and $r = 5$.

$\mu = 0.5$	Iter	$F(X^*)$	sparsity	CPU time	# line-search	SSN iters
ManPG	504.12	-39.41	0.37	0.0843	0.00	1.02
ManPG-Ada	209.14	-39.41	0.37	0.0401	43.44	1.13
NLS-ManPG	160.62	-39.41	0.37	0.0381	0.00	1.27
ManPQN	41.24	-39.34	0.35	0.0168	26.60	1.87
$\mu = 0.6$						
ManPG	563.02	-32.69	0.43	0.0912	0.00	1.02
ManPG-Ada	219.42	-32.69	0.43	0.0431	49.30	1.15
NLS-ManPG	198.58	-32.69	0.43	0.0431	0.00	1.25
ManPQN	53.16	-32.63	0.41	0.0212	33.72	1.90
$\mu = 0.7$						
ManPG	475.74	-26.30	0.50	0.0779	0.00	1.02
ManPG-Ada	189.52	-26.30	0.50	0.0375	25.36	1.14
NLS-ManPG	154.96	-26.30	0.50	0.0391	0.00	1.28
ManPQN	55.26	-26.26	0.48	0.0229	34.32	1.96
$\mu = 0.8$						
ManPG	602.94	-20.36	0.56	0.0952	1.40	1.02
ManPG-Ada	216.12	-20.36	0.56	0.0433	39.86	1.24
NLS-ManPG	180.76	–20.36	0.56	0.0438	0.00	1.40
ManPQN	77.40	-20.32	0.56	0.0259	53.34	1.96
$\mu = 0.9$						
ManPG	677.48	-14.63	0.62	0.1026	0.14	1.03
ManPG-Ada	224.90	-14.63	0.62	0.0432	38.54	1.27
NLS-ManPG	201.88	-14.63	0.62	0.0431	0.00	1.41
ManPQN	104.62	-14.59	0.62	0.0324	69.30	1.96
$\mu = 1.0$						
ManPG	706.12	-9.525	0.69	0.1053	0.28	1.04
ManPG-Ada	225.10	-9.525	0.69	0.0426	32.26	1.30
NLS-ManPG	209.84	-9.525	0.69	0.0440	0.02	1.39
ManPQN	116.16	-9.485	0.69	0.0351	109.80	1.93

5.3 Joint Diagonalization Problem with a Regularization Term

The joint diagonalization problem with a regularization term on the Stiefel manifold [34] can be written in the following formulation:

$$\min_{X \in \mathcal{M}} -\sum_{l=1}^{N} \|\mathrm{diag}(X^T A_l X)\|_F^2 + \mu \|X\|_1, \tag{5.3}$$

where $A_1, A_2, \ldots, A_N \in \mathbb{R}^{n \times n}$ are N real symmetric matrices.

Table 7 Comparison on Sparse PCA problem with different matrices from [8].

"lpi_klein1"	Iter	$F(X^*)$	sparsity	CPU time	# line-search	SSN iters
ManPG	2252.00	-38.77	0.94	0.1684	0.00	1.01
ManPG-Ada	919.00	-38.77	0.94	0.0840	533.00	1.10
NLS-ManPG	223.00	-38.77	0.94	0.0305	170.00	1.74
ManPQN	137.00	-38.76	0.93	0.0318	58.00	1.27
"bcsstk20"	Iter	$F(X^*)$	sparsity	CPU time	# line-search	SSN iters
ManPG	11276.00	-4823.74	0.82	0.8836	157.06	1.00
ManPG-Ada	4685.58	-4823.74	0.82	0.4489	2985.08	1.11
NLS-ManPG	830.22	-4823.74	0.82	0.1073	833.80	1.59
ManPQN	137.18	-4823.71	0.80	0.0322	0.00	1.00
"lp_fit1d"						
ManPG	9596.26	-2480.94	0.08	2.9966	0.00	1.45
ManPG-Ada	4182.68	-2480.94	0.08	1.7876	2488.80	1.97
NLS-ManPG	1024.54	-2480.94	0.08	0.5133	850.58	2.31
ManPQN	210.16	-2480.64	0.05	0.1115	7.22	2.17
"fidap003"						
ManPG	4986.52	-6764.09	0.89	2.1449	0.00	1.06
ManPG-Ada	2169.84	-6764.09	0.89	1.1988	1249.16	1.19
NLS-ManPG	478.88	-6764.09	0.89	0.4152	449.62	1.93
ManPQN	95.58	-6764.08	0.87	0.1705	1.14	3.07

Let $f(X) := -\sum_{l=1}^{N} \|\mathrm{diag}(X^T A_l X)\|_F^2$. We can deduce that the Euclidean gradient of f is

$$\nabla f(X) = -4 \sum_{l=1}^{N} A_l X \mathrm{diag}(X^T A_l X),$$

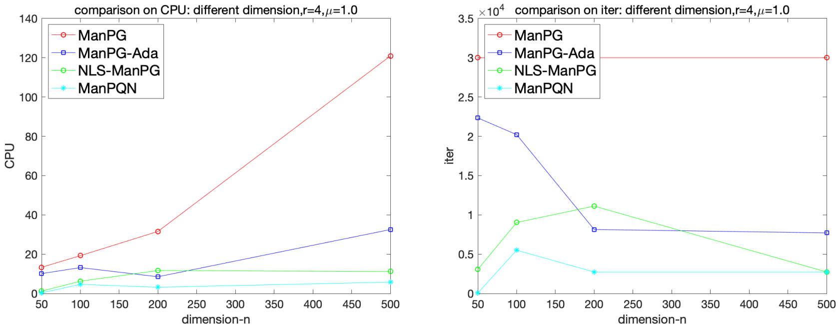
and the Riemannian gradient of f is

$$\mathrm{grad} f(X) = -4 \sum_{l=1}^{N} (A_l X \mathrm{diag}(X^T A_l X) - X \mathrm{sym}(X^T A_l X \mathrm{diag}(X^T A_l X))),$$

where $\mathrm{sym}(Y) := (Y^T + Y)/2$.

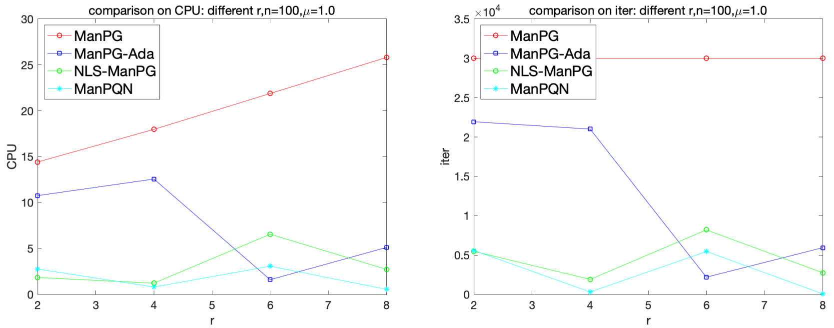
For numerical experiments of (5.3), we generate N randomly chosen $n \times n$ diagonal matrices $\Lambda_1, \Lambda_2, \ldots, \Lambda_N$ and a randomly chosen $n \times n$ orthogonal matrix P. Then, let $A_1, A_2, \ldots, A_N$ be computed by $A_i = P^T \Lambda_i P$ for $i = 1, 2, \ldots, N$. We set $N = 5$ in the following experiments.

The numerical results of each algorithm are displayed in Figures 10-12 and Tables 8-10. Since all the algorithms need large number of line search steps, we report the average iteration numbers of line search in Tables 8-10. Figures 10-12 show that ManPQN needs much less iterations and time than ManPG and ManPG-Ada. In most cases, ManPQN outperforms NLS-ManPG in terms of CPU time, especially when n and r are large.



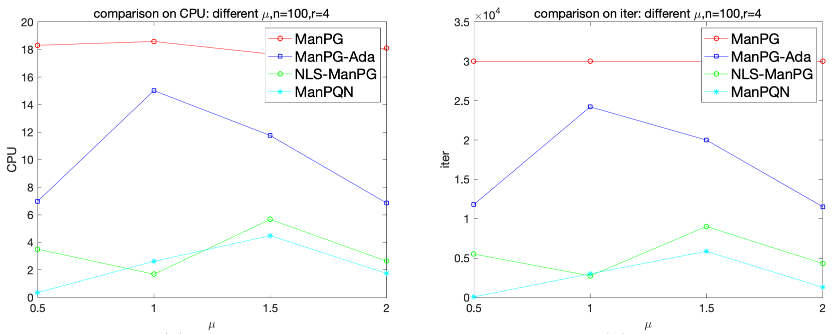
(a) CPU (b) Iter

Fig. 10 Comparison on problem (5.3), different $n = \{50, 100, 200, 500\}$ with $r = 4$ and $\mu = 1.0$.



(a) CPU (b) Iter

Fig. 11 Comparison on problem (5.3), different $r = \{2, 4, 6, 8\}$ with $n = 100$ and $\mu = 1.0$.



(a) CPU (b) Iter

Fig. 12 Comparison on problem (5.3), different $\mu = \{0.5, 1.0, 1.5, 2.0\}$ with $n = 100$ and $r = 4$.

Table 8 Comparison on problem (5.3), different $n = \{50, 100, 200, 500\}$ with $r = 4$ and $\mu = 1.0$.

$n = 50$	Iter	$F(X^*)$	CPU time	line-search	SSN iters
ManPG	30001.00	-72.804	13.3257	13.99	1.00
ManPG-Ada	22335.45	-72.784	10.1168	13.98	1.01
NLS-ManPG	3073.09	-72.821	1.2158	12.09	2.91
ManPQN	68.82	-72.825	0.3008	9.48	12.54
$n = 100$					
ManPG	30001.00	-126.85	19.2554	13.99	1.01
ManPG-Ada	20215.64	-126.85	13.1823	13.98	1.68
NLS-ManPG	9032.55	-126.87	6.2430	13.97	4.04
ManPQN	5509.45	-126.81	4.6449	12.73	14.62
$n = 200$					
ManPG	30001.00	-148.11	31.5747	13.99	1.00
ManPG-Ada	8143.82	-148.11	8.5102	13.97	2.44
NLS-ManPG	11119.36	-148.11	11.6534	13.9	3.22
ManPQN	2730.18	-148.17	3.1700	13.5	13.32
$n = 500$					
ManPG	30001.00	-222.75	120.9160	13.99	1.00
ManPG-Ada	7709.36	-222.77	32.5464	13.98	3.50
NLS-ManPG	2736.82	-222.79	11.1977	13.95	4.70
ManPQN	2729.18	-222.93	5.7707	5.54	22.69

6 Conclusion and Future Work

In this paper, we present a proximal quasi-Newton algorithm, named ManPQN, for the composite optimization problem (1.1) over the Stiefel manifold. The ManPQN method finds the descent direction V_k by solving a subproblem, which is formed by replacing the term $\|V\|^2/2t$ in the subproblem of ManPG by $\frac{1}{2}\|V\|^2_{\mathcal{B}_k}$, where $\mathcal{B}_k$ is a symmetric linear operator on $\mathrm{T}_{X_k}\mathcal{M}$. We also use several techniques to accelerate the speed of the ManPQN algorithm. The most important technique is that we use the linear operator $\mathbf{B}_k$, defined by (3.7), to approximate the linear operator $\mathcal{B}_k$. To guarantee the positive definiteness of $\mathbf{B}_k$, a damped LBFGS method is used to update $\mathbf{B}_k$. Moreover, we use a nonmonotone line search technique to improve the performance of the ManPQN method. Numerical results demonstrate that the ManPQN method is an effective method. Under some mild conditions, we establish the global convergence of ManPQN. If the Hessian operator of the objective function is positive definite at the local minimum, the local linear convergence of ManPQN is also proved.

The main cost of the ManPQN lies in the step of solving the subproblem (3.1). Stimulated by the work in [6], we use the adaptive semismooth Newton (ASSN) method to get the solution of (3.1). But the total cost of the ASSN method is excessive for large n and r. How to reduce the computational cost of the ASSN method will be one of the topics of our future work. We will also investigate other techniques to accelerate the ManPG method. This is another topic of our future work.

Table 9 Comparison on problem (5.3), different $r = \{2, 4, 6, 8\}$ with $n = 100$ and $\mu = 1.0$.

$r = 2$	Iter	$F(X^*)$	CPU time	line-search	SSN iters
ManPG	30001.00	-69.79	14.4263	13.99	0.92
ManPG-Ada	21941.64	-69.79	10.7613	13.98	1.22
NLS-ManPG	5474.55	-69.80	1.8400	8.81	3.24
ManPQN	5580.91	-69.80	2.7605	8.73	16.92
$r = 4$					
ManPG	30001.00	-119.37	17.9799	13.99	1.00
ManPG-Ada	21010.00	-119.38	12.5684	13.98	1.80
NLS-ManPG	1909.36	-119.39	1.2302	12.56	4.35
ManPQN	310.09	-119.42	0.8054	12.44	18.71
$r = 6$					
ManPG	30001.00	-122.92	21.8858	13.99	1.00
ManPG-Ada	2190.82	-122.92	1.6054	13.98	1.47
NLS-ManPG	8198.64	-122.91	6.5546	13.95	4.10
ManPQN	5456.36	-123.00	3.0761	5.89	10.95
$r = 8$					
ManPG	30001.00	-142.23	25.8129	13.99	1.00
ManPG-Ada	5973.36	-142.29	5.1177	13.97	2.67
NLS-ManPG	2747.73	-142.30	2.7289	13.94	5.02
ManPQN	70.00	-142.54	0.5549	9.25	17.60

Acknowledgments. The work of Wei Hong Yang was supported by the National Natural Science Foundation of China grant 11971118. The authors are grateful to the associate editor and the two anonymous referees for their valuable comments and suggestions.

References

[1] Absil, P.-A., Hosseini, S.: A collection of nonsmooth Riemannian optimization problems. International Series of Numerical Mathematics (2017). https://doi.org/10.1007/978-3-030-11370-4_1

[2] Absil, P.-A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, Princeton, NJ (2008)

[3] Beck, A., Teboulle, M.: A fast iterative shrinkage-threshold algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**, 183–202 (2009). https://doi.org/10.1137/080716542

[4] Bento, G., Cruz Neto, J., Oliveira, P.: A new approach to the proximal point method: Convergence on general Riemannian manifolds. Journal of Optimization Theory and Applications **168** (2016). https://doi.org/10.1007/s10957-015-0861-2

[5] Boumal, N., Absil, P.-A., Cartis, C.: Global rates of convergence for nonconvex optimization on manifolds. IMA Journal of Numerical Analysis **39** (2016). https:

Table 10 Comparison on problem (5.3), different $\mu = \{0.5, 1.0, 1.5, 2.0\}$ with $n = 100$ and $r = 4$.

$\mu = 0.5$	Iter	$F(X^*)$	CPU time	line-search	SSN iters
ManPG	30001.00	-134.85	18.3098	13.99	1.00
ManPG-Ada	11813.55	-134.86	6.9803	13.96	1.84
NLS-ManPG	5509.45	-134.86	3.5001	13.92	3.74
ManPQN	74.27	-134.87	0.3515	8.62	13.36
$\mu = 1.0$					
ManPG	30001.00	-129.57	18.5839	13.99	0.98
ManPG-Ada	24221.09	-129.57	15.0166	13.99	1.24
NLS-ManPG	2745.73	-129.59	1.6866	13.90	3.84
ManPQN	3003.18	-129.58	2.6224	13.76	12.40
$\mu = 1.5$					
ManPG	30001.00	-112.51	17.6801	13.99	1.00
ManPG-Ada	19994.13	-112.51	11.7601	13.98	1.11
NLS-ManPG	9018.80	-112.51	5.6677	13.88	3.62
ManPQN	5854.17	-112.53	4.4855	13.5	12.66
$\mu = 2.0$					
ManPG	30001.00	-76.63	18.0976	13.99	1.10
ManPG-Ada	11534.86	-76.64	6.8606	13.98	1.35
NLS-ManPG	4306.00	-76.65	2.6373	13.91	4.01
ManPQN	1292.71	-76.71	1.7506	12.65	16.84

//doi.org/10.1093/imanum/drx080

[6] Chen, S., Ma, S., So, A.M.-C., Zhang, T.: Proximal gradient method for nonsmooth optimization over the Stiefel manifold. SIAM Journal on Optimization **30**(1), 210–239 (2020). https://doi.org/10.1137/18M122457X

[7] Dai, Y.-H.: A nonmonotone conjugate gradient algorithm for unconstrained optimization. Journal of Systems Science and Complexity **15** (2002)

[8] Davis, T.A., Hu, Y.: The university of florida sparse matrix collection. ACM Transactions on Mathematical Software (TOMS) **38.1**, 1–25 (2011)

[9] Ferreira, O., Oliveira, P.: Subgradient algorithm on Riemannian manifolds. Journal of Optimization Theory and Applications **97**, 93–104 (1998). https://doi.org/10.1023/A:1022675100677

[10] Ferreira, O., Oliveira, P.: Proximal point algorithm on Riemannian manifolds. Optimization **51**, 257–270 (2002). https://doi.org/10.1080/02331930290019413

[11] Fukushima, M., Qi, L.: A globally and superlinearly convergent algorithm for nonsmooth convex minimization. SIAM Journal on Optimization **6**(4), 1106–1120 (1996). https://doi.org/10.1137/S1052623494278839

[12] Gao, B., Liu, X., Yuan, Y.-X.: Parallelizable algorithms for optimization

problems with orthogonality constraints. SIAM Journal on Scientific Computing **41**(3), 1949–1983 (2019) https://arxiv.org/abs/https://doi.org/10.1137/18M1221679. https://doi.org/10.1137/18M1221679

[13] Grippo, L., Lampariello, F., Lucidi, S.: A nonmonotone line search technique for Newton's method. SIAM Journal on Numerical Analysis **23**(4), 707–716 (1986). https://doi.org/10.1137/0723046

[14] Grohs, P., Hosseini, S.: ε-subgradient algorithms for locally Lipschitz functions on Riemannian manifolds. Advances in Computational Mathematics **42** (2015). https://doi.org/10.1007/s10444-015-9426-z

[15] Hosseini, S., Grohs, P.: Nonsmooth trust region algorithms for locally Lipschitz functions on Riemannian manifolds. IMA Journal of Numerical Analysis **36** (2015). https://doi.org/10.1093/imanum/drv043

[16] Hu, J., Milzarek, A., Wen, Z., Yuan, Y.-X.: Adaptive quadratically regularized newton method for Riemannian optimization. SIAM Journal on Matrix Analysis and Applications **39**(3), 1181–1207 (2018). https://doi.org/10.1137/17M1142478

[17] Huang, W., Wei, K.: Riemannian proximal gradient methods. Mathematical Programming (2021). https://doi.org/10.1007/s10107-021-01632-3

[18] Huang, W., Wei, K.: An extension of fast iterative shrinkage-thresholding to Riemannian optimization for sparse principal component analysis. Numerical Linear Algebra with Applications **29:e2409** (2022). https://doi.org/10.1002/nla.2409

[19] Huang, Y., Liu, H.: On the rate of convergence of projected Barzilai–Borwein methods. Optimization Methods and Software **30**, 1–13 (2015). https://doi.org/10.1080/10556788.2015.1004064

[20] Lee, J.D., Sun, Y., Saunders, M.A.: Proximal Newton-type methods for minimizing composite functions. SIAM Journal on Optimization **24**(3), 1420–1443 (2014). https://doi.org/10.1137/130921428

[21] Li, D., Wang, X., Huang, J.: Diagonal BFGS updates and applications to the limited memory BFGS method. Computational Optimization and Applications **81**, 1–28 (2022). https://doi.org/10.1007/s10589-022-00353-3

[22] Liu, H., Wu, W., So, A.M.-C.: Quadratic optimization with orthogonality constraints: Explicit Łojasiewicz exponent and linear convergence of retraction-based line-search and stochastic variance-reduced gradient methods. Mathematical Programming **178**, 215–262 (2019). https://doi.org/10.1007/s10107-018-1285-1

[23] Mordukhovich, B., Yuan, X., Zeng, S., Zhang, J.: A globally convergent proximal Newton-type method in nonsmooth convex optimization. Mathematical Programming (2022). https://doi.org/10.1007/s10107-022-01797-5

[24] Moré, J.J., Sorensen, D.C.: Computing a trust region step. SIAM Journal on Scientific and Statistical Computing **4**(3), 553–572 (1983). https://doi.org/10.1137/0904038

[25] Nakayama, S., Narushima, Y., Yabe, H.: Inexact proximal memoryless quasi-Newton methods based on the Broyden family for minimizing composite functions. Computational Optimization and Applications **79**, 1–28 (2021). https://doi.org/10.1007/s10589-021-00264-9

[26] Nesterov, Y.: Gradient methods for minimizing composite functions. Math. Program. Ser. B **140**, 125–161 (2013). https://doi.org/10.1007/s10107-012-0629-5

[27] Nesterov, Y.: Lectures on Convex Optimization, 2nd edn. Springer, Cham (2018)

[28] Nocedal, J.: Updating quasi-Newton matrices with limited storage. Mathematics of Computation **35**, 773–782 (1980). https://doi.org/10.2307/2006193

[29] Oviedo, H.: Proximal point algorithm on the stiefel manifold. Preprint in Optimization–Online: http://www.optimization-online.org/DB_FILE/2021/05/8401.pdf (2021)

[30] Ozoliņš, V., Lai, R., Caflisch, R., Osher, S.: Compressed modes for variational problems in mathematics and physics. Proceedings of the National Academy of Sciences of the United States of America **110**(46), 18368–18373 (2013). https://doi.org/10.1073/pnas.1318679110

[31] Park, Y., Dhar, S., Boyd, S., Shah, M.: Variable metric proximal gradient method with diagonal Barzilai-Borwein stepsize. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3597–3601 (2020). https://doi.org/10.1109/ICASSP40776.2020.9054193

[32] Powell, M.J.: Algorithms for nonlinear constraints that use Lagrangian functions. Math. Program. **14**(1), 224–248 (1978). https://doi.org/10.1007/BF01588967

[33] Ring, W., Wirth, B.: Optimization methods on Riemannian manifolds and their application to shape space. SIAM Journal on Optimization **22**(2), 596–627 (2012). https://doi.org/10.1137/11082885X

[34] Sato, H.: Riemannian newton-type methods for joint diagonalization on the stiefel manifold with application to independent component analysis (2014) https://arxiv.org/abs/arXiv:1403.8064. https://doi.org/10.1080/02331934.2017.1359592

[35] Sherman, J., Morrison, W.J.: Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. Annals of Mathematical Statistics **21**, 124–127 (1950)

[36] Shor, N.Z.: Application of generalized gradient descent in block programming. Cybernetics **3**(3), 43–45 (1967). https://doi.org/10.1007/BF01120005

[37] Wang, X., Ma, S., Goldfarb, D., Liu, W.: Stochastic quasi-Newton methods for nonconvex stochastic optimization. SIAM Journal on Optimization **27**(2), 927–956 (2017). https://doi.org/10.1137/15M1053141

[38] Wen, Z., Yin, W.: A feasible method for optimization with orthogonality constraints. Mathematical Programming **142** (2010). https://doi.org/10.1007/s10107-012-0584-1

[39] Wright, S., Nowak, R., Figueiredo, M.: Sparse reconstruction by separable approximation, vol. 57, pp. 3373–3376 (2008). https://doi.org/10.1109/TSP.2009.2016892

[40] Xiao, X., Li, Y., Wen, Z., Zhang, L.W.: A regularized semi-smooth Newton method with projection steps for composite convex programs. Journal of Scientific Computing **76** (2018). https://doi.org/10.1007/s10915-017-0624-3

[41] Yang, W.H., Zhang, L., Song, R.: Optimality conditions for the nonlinear programming problems on Riemannian manifolds. Pacific Journal of Optimization (2013)

[42] Zhang, C., Chen, X., Ma, S.: A riemannian smoothing steepest descent method for non-lipschitz optimization on submanifolds. ArXiv **abs/2104.04199** (2021)