Journal of Computational Mathematics
Vol.xx, No.x, 2023, 1–33.
http://www.global-sci.org/jcm
doi:10.4208/jcm.2211-m2021-0043

A TRUST-REGION METHOD FOR SOLVING TRUNCATED COMPLEX SINGULAR VALUE DECOMPOSITION*

Jiaofen Li
School of Mathematics and Computing Science,
Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation,
Guangxi Key Laboratory of Automatic Detecting Technology and Instruments,
Guilin University of Electronic Technology, Guilin 541004, China
Center for Applied Mathematics of Guangxi (GUET), Guilin 541004, China
Lingchang Kong
School of Mathematical Sciences, South China Normal University, Guangzhou 510000, China
School of Mathematics and Computing Science,
Guilin University of Electronic Technology, Guilin 541004, China
Xuefeng Duan[1)]
School of Mathematics and Computing Science,
Guangxi Colleges and Universities Key Laboratory of Data Analysis and Computation,
Guilin University of Electronic Technology, Guilin 541004, China
Center for Applied Mathematics of Guangxi (GUET), Guilin 541004, China
Email: guidian520@126.com
Xuelin Zhou
School of Mathematics and Statistics, Yunan University, Kunming 650000, China
School of Mathematics and Computing Science,
Guilin University of Electronic Technology, Guilin 541004, China
Email: zhouxuelin0309@163.com
Qilun Luo
School of Mathematical Sciences, South China Normal University, Guangzhou 510000, China

Abstract

The truncated singular value decomposition has been widely used in many areas of science including engineering, and statistics, etc. In this paper, the original truncated complex singular value decomposition problem is formulated as a Riemannian optimization problem on a product of two complex Stiefel manifolds, a practical algorithm based on the generic Riemannian trust-region method of Absil et al. is presented to solve the underlying problem, which enjoys the global convergence and local superlinear convergence rate. Numerical experiments are provided to illustrate the efficiency of the proposed method. Comparisons with some classical Riemannian gradient-type methods, the existing Riemannian version of limited-memory BFGS algorithms in the MATLAB toolbox Manopt and the Riemannian manifold optimization library ROPTLIB, and some latest infeasible methods for solving manifold optimization problems, are also provided to show the merits of the proposed approach.

Mathematics subject classification: 15A24, 15A57, 65F10, 65F30.
Key words: Truncated singular value decomposition, Riemannian optimization, Trust-region method.

* Received February 9, 2021 / Revised version received September 27, 2021 / Accepted November 3, 2022 / Published online June 6, 2023 /

1) Corresponding author

1. Introduction

In linear algebra, the singular value decomposition (SVD) of a complex matrix $A \in \mathbb{C}^{m\times n}$ is a factorization of the form $U\Sigma V^H$, where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices, $\Sigma \in \mathbb{C}^{m\times n}$ is a rectangular diagonal matrix with non-negative real numbers on the diagonal. The diagonal entries $\sigma_i = \Sigma_{ii}$ of Σ are known as the singular values of A. The number of non-zero singular values is equal to the rank of A. The columns of U and the columns of V are called the left-singular vectors and right-singular vectors of A, respectively. The truncated singular value decomposition (TSVD) is a kind of reduced SVDs, in which only the p ($p \le \operatorname{rank}(A)$) column vectors of U and p row vectors of V^H corresponding to the p largest singular values Σ_p are calculated. The rest of the matrix is discarded. This can be much quicker and more economical than the compact SVD if $p \ll \operatorname{rank}(A)$. The matrix U_p is thus $m \times p$, Σ_p is $p \times p$ diagonal, and V_p is $n \times p$. Of course the truncated SVD is no longer an exact decomposition of the original matrix A, but the approximate matrix $\tilde{A} = U_p\Sigma_p V_p^H$ is in a very useful sense the closest approximation to A that can be achieved by a matrix of rank p. To find p ($\le \operatorname{rank}(A)$) left and right singular vectors associated with the p largest singular values of a complex matrix $A \in \mathbb{C}^{m\times n}$, a closely related problem is to solve the following optimization problem [3,27,29–31]:

$$\begin{aligned} &\text{maximize:} \quad \operatorname{Tr}(U^H A V \Theta), \\ &\text{subject to:} \quad U \in \mathbb{C}^{m\times p}, \quad V \in \mathbb{C}^{n\times p}, \quad U^H U = I_p, \quad V^H V = I_p, \end{aligned} \tag{1.1}$$

where $\Theta = \operatorname{diag}(\mu_1, \cdots, \mu_p)$, with $\mu_1 > \cdots > \mu_p > 0$ arbitrary and $1 \le p \le \operatorname{rank}(A)$, and where I_p is the identity matrix of order p. Here, the arbitrary diagonal matrix Θ with positive diagonal elements in descending order plays a role in appropriately ordering the columns of U and V ([29, Chapter 5] and [30]). In (1.1), the notation $\operatorname{Tr}(\cdot)$ denotes the real part of the trace of a given complex square matrix, which corresponds to the inner product for complex vector space, i.e., in $\mathbb{C}^{p\times q}$, we define the inner product

$$\langle M, N\rangle := \operatorname{Tr}(M^H N) = \operatorname{trace}\left(\operatorname{Re}(M)^T \operatorname{Re}(N)\right) + \operatorname{trace}\left(\operatorname{Im}(M)^T \operatorname{Im}(N)\right)$$

for all $M, N \in \mathbb{C}^{p\times q}$, where $\operatorname{Re}(M)$ and $\operatorname{Im}(M)$ denote the real and imaginary parts of M, respectively. Then $\mathbb{C}^{p\times q}$ is a Hilbert inner product space and the norm of a complex matrix M generated by this inner product is the matrix Frobenius norm given by $\|M\|^2 = \operatorname{Tr}(M^H M)$. A global optimal solution to the problem (1.1) provides a collection of p dominant left and right singular vectors of A. Let (U^*, V^*) be an optimal solution to the problem (1.1). Then, the j-th columns of U^* and V^* are the left and right singular vectors of A associated with the j-th dominant singular value, respectively [3, 27, 29–31]. In addition, the p largest singular values $\sigma_1 \ge \cdots \ge \sigma_p$ can be calculated through the formula ${U^*}^H A V^* = \operatorname{diag}(\sigma_1, \cdots, \sigma_p)$.

Mathematical applications of the TSVD include computing the matrix low-rank approximation [11], analysing the ill-posed models [19] and solving the ill-conditioned system of algebraic equation [12]. In the ill-posed models, the ill conditioned of the model is mainly reflected in the amplification of the small singular values of the coefficient matrix on the parameters and their variances. The basic idea of the truncated singular value method is to cut off these small singular values and reconstruct the coefficient matrix to weaken the ill-posedness of the model. The TSVD is also extremely useful in all areas of science, engineering, and statistics, such as bioluminescence tomography (BLT) [40] and through-the-wall microwave imaging (TWI) [16]. In BLT, TSVD regularization method is applied to solving BLT inverse problem with the source permissible region as a priori knowledge. The numerical simulation shows that the TSVD is feasible and efficient for BLT reconstruction and compared with Tikhonov regularization, TSVD method has better noise-resistance, less time cost and more convenience for the determination of regularization parameter. In TWI application, TSVD becomes a suitable inversion technique when certain data are needed to be neglected. For the sake of example, in TWI scenarios, the wall itself is the trouble or in medical imaging scenarios, the skin stays out of the region of interest, therefore, these unwanted outcomes must be truncated from scattering measurements. In particular, TSVD is applied to make an analysis of breast [34] and brain [33] imaging, and it is also used for imaging a target in multi path environments [18]. In [14], the generalised cross-validation method is combined with TSVD to stabilise the inversion. In another study [36], TSVD is combined with compressive sensing techniques to reduce the number of spatial and frequency measurements while maintaining the quality of the results. It is also shown that TSVD is a promising algorithm to detect targets behind the wall experimentally [35,37].

For solving the optimization problem (1.1), Sato *et al.* [27,29–31] presented serval numerical approaches from the perspective of Riemannian optimizations, by noting that the feasible set, $\mathrm{St}(m,p,\mathbb{C}) = \{Y \in \mathbb{C}^{m\times p} \,|\, Y^H Y = I_p\}$, of (1.1) can be viewed as the complex Stiefel manifold, and then the original problem can be formulated (1.1) as the following Riemannian optimization problem on the complex product manifold $\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$:

$$\begin{aligned} &\text{minimize:} && -\operatorname{Tr}\left(U^H A V \Theta\right), \\ &\text{subject to:} && (U,V) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C}). \end{aligned} \tag{1.2}$$

Sato [27] developed an Riemannian nonlinear conjugate gradient method, in which the differentiated retraction as a vector transport is used, for solving the problem (1.2), by rewriting (1.2) as an equivalent problem on the product of two real manifolds, with each of them being an intersection of the real Stiefel manifold $\mathrm{St}(2m,2p,\mathbb{R})$ ($\mathrm{St}(2n,2p,\mathbb{R})$) and a quasi-symplectic set $\mathcal{SP}(m,p)$ ($\mathcal{SP}(n,p)$). However, as we know, gradient-type algorithms often perform reasonably well but might converge slowly when the generated iterates are close to an optimal solution. Usually, fast local convergence cannot be expected if only the gradient information is used. In the Euclidean space $\mathbb{R}^n$, it is well known that higher rates of convergence can be achieved by using second-order information on the cost function. The classical choice is Newton's method and trust-region method. Actually, in [31], by realizing $\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ as the intersection of the real Stiefel manifold and the quasi-symplectic set, Sato and Iwai have further proposed a Riemannian Newton's method for the problem (1.2). A hybrid method, by combining the conjugate gradient method with Newton's method is also provided for speeding up the convergence. However, this algorithm is not easy to implement in its original form because the Newton's equation is expressed by a system of matrix equations which is difficult to solve directly. In [31], Newton's equation are divided into a collection of subequations by putting $p = 1$ and treating the equation on $\mathrm{St}(m,1,\mathbb{R}) \times \mathrm{St}(n,1,\mathbb{R})$. This makes Newton's equation into a vector equation which is easy to solve. In addition, if A is real, U and $V^T = V^H$ are real orthogonal matrices, Sato and Iwai [30] also developed the steepest descent, the conjugate gradient, and Newton's methods for the truncated real singular value decomposition. While in the Newton's method, the problem of solving Newton's equation with $p \neq 1$ is also divided into a set of the problems with $p = 1$, if suitable initial data are given. Specifically, for the generated Newton's equation of Newton's method on the truncated real singular value decomposition, Aihara and Sato [5] proposed an effective implementation by rewriting the Newton's equation into a symmetric linear system. And then a matrix-free Krylov subspace method is used to solve this symmetric linear system, in which matrix-vector multiplications can be computed without explicitly using the coefficient matrix. Numerical experiments demonstrate that the proposed method is effective for the truncated real singular value decomposition problem.

As we know, in the Euclidean space $\mathbb{R}^n$, the pure Newton method converges only locally, and it cannot distinguish between local minima, local maxima and saddle points. Compared to the pure Newton-type algorithm, the advantage of the trust-region algorithm is its more stable behavior [1], and convergence of the trust-region algorithm to stationary points is guaranteed for all initial points. Since optimization over the Stiefel manifold can be viewed as a general nonlinear optimization problem with constraints, trust-region algorithm in the Euclidean space can be generalized to manifold setting directly, and have been explored and successfully applied to various applications [1,2,7,9,20,24,28,32,39,41]. Recently, Sato [28] developed a Riemannian trust-region (RTR) algorithm to the joint singular value decomposition of multiple rectangular matrices, which is formulated as a Riemannian optimization problem on the product of two Stiefel manifolds. Ishteva *et al.* [24] proposed a RTR scheme for the best rank-(R_1, R_2, R_3) approximation of third-order tensors, which is expressed as a minimization of a cost function on a product of three Grassmann manifolds. Sato and Sato [32] proposed a RTR method for H^2 optimal model reduction problems of linear systems, which is formulated as a nonlinear optimization problem on the product manifold of the manifold of symmetric positive-definite matrices and two Euclidean spaces. Yang *et al.* [39] presented an RTR algorithm for H_2 model reduction of bilinear systems, where the H_2 error norm is treated as a cost function on the Stiefel manifold. Motivated by the aforementioned works, in the present paper, we are interested in extending the Riemannian trust-region method to the underlying truncated complex singular value decomposition problem (1.2). We first derive the specific expression of Riemannian gradient and Hessian of the objective function of the problem (1.2). By utilizing the Taylor expansion on the product manifold, RTR algorithm constructs a trust-region subproblem on the tangent space, based on our expression of the Riemannian Hessian of the objective function. The trust-region subproblem can be solved by the classical truncated conjugate gradient method, which is most popular due to its good properties and relatively cheap computational cost. Numerical results show that the proposed algorithm is quite efficient for solving the problem (1.2). Detail numerical comparisons with some classical Riemannian gradient-type methods and the existing Riemannian version of limited-memory BFGS algorithms in the MATLAB toolbox Manopt and the Riemannian manifold optimization library ROPTLIB are presented. Comparisons with some latest infeasible methods for solving manifold optimization problems, which simplify the constrained problem (1.2) by relaxing the constraints and iteratively diminish the degree of infeasibility, and the resulting intermediate points of the generated sequence may not satisfy the orthogonality constraints, are also provided to show the merits of the proposed approach.

This paper is organized as follows. In Section 2, after presenting some basic geometric properties of the product manifold in question, we derive the representation matrix formula of Riemannian gradient and Hessian of the objective function, and then develop the corresponding RTR method to solve the transformed Riemannian optimization problem on a product manifold. Numerical examples and numerical comparisons are provided in Section 3. Finally, the paper ends with concluding remarks in Section 4.

2. Framework of Riemannian Trust-Region Method

We first recall some notation, definitions and basic properties of Riemannian manifolds used throughout the paper. The tangent space at x on a manifold $\mathcal{M}$ is denoted by $\mathbf{T}_x\mathcal{M}$. For manifolds $\mathcal{M}$ and $\mathcal{N}$ and a mapping $f : \mathcal{M} \to \mathcal{N}$, the differential of f at $x \in \mathcal{M}$ is denoted by $\mathcal{D}f(x)$, which is a mapping from $\mathbf{T}_x\mathcal{M}$ to $\mathbf{T}_{f(x)}\mathcal{N}$. Given a smooth function f on a manifold $\mathcal{M} \subset \mathbb{R}^{m\times p}$, the symbol $\bar{f}$ is the extension of f to the ambient Euclidean space $\mathbb{R}^{m\times p}$. The symbols ∇ and grad denote the Euclidean and Riemannian gradients, respectively; i.e., given a smooth function f on a manifold $\mathcal{M} \subset \mathbb{R}^{m\times p}$, ∇ and grad act on $\bar{f}$ and f, respectively. The symbol Hess denotes the Riemannian Hessian. The concept of a retraction, which is a smooth map from the tangent bundle of $\mathcal{M}$ into $\mathcal{M}$ that approximates the exponential map to the first order, is given as follows.

Definition 2.1 ([3, Definition 4.1.1]). *A retraction on a manifold $\mathcal{M}$ is a smooth mapping $\mathcal{R}$ from the tangent bundle $\mathbf{T}\mathcal{M} := \bigcup_{v\in\mathcal{M}} \mathbf{T}_v\mathcal{M}$ onto $\mathcal{M}$ with the following properties. Let $\mathcal{R}_v$ denote the restriction of $\mathcal{R}$ to $\mathbf{T}_v\mathcal{M}$.*

1. *$\mathcal{R}_v(0_v) = v$, where 0_v denotes the zero element of $\mathbf{T}_v\mathcal{M}$.*
2. *With the canonical identification $\mathbf{T}_{0_v}\mathbf{T}_v\mathcal{M} \simeq \mathbf{T}_v\mathcal{M}$, $\mathcal{R}_v$ satisfies*

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

where $\mathrm{id}_{\mathbf{T}_v\mathcal{M}}$ denotes the identity mapping on $\mathbf{T}_v\mathcal{M}$.

Given a retraction $\mathcal{R}$ and a smooth manifold $\mathcal{M}$, the general feasible algorithm framework on the manifold can be expressed as

$$x_{k+1} = \mathcal{R}_{x_k}(t_k\xi_k), \tag{2.1}$$

where t_k is the step size at the k-th iterate x_k, and $\xi_k \in \mathbf{T}_{x_k}\mathcal{M}$ is a tangent vector.

2.1. Riemannian geometry of $\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$

We next review the geometry of complex product manifold $\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$. Notice that the dimension of the complex manifold $\mathrm{St}(m,p,\mathbb{C})$ and $\mathrm{St}(n,p,\mathbb{C})$ are given by [3, 17]

$$\dim \mathrm{St}(m,p,\mathbb{C}) = 2mp - p^2, \quad \dim \mathrm{St}(n,p,\mathbb{C}) = 2np - p^2.$$

Then we have

$$\dim\big(\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})\big) := \dim \mathrm{St}(m,p,\mathbb{C}) + \dim \mathrm{St}(n,p,\mathbb{C}) = 2p(m+n-p). \tag{2.2}$$

Because the product manifold $\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ is an embedded submanifold of $\mathbb{C}^{m\times p}$ $\times\mathbb{C}^{n\times p}$, one may equip with $\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ the induced Riemannian metric

$$g_{(V,P)}\big((\xi,\eta),(\xi',\eta')\big) := \mathrm{Tr}(\xi^H\xi') + \mathrm{Tr}(\eta^H\eta') \tag{2.3}$$

for any $(U,V) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ and $(\xi,\eta),(\xi',\eta') \in \mathbf{T}_{(V,P)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$, and its induced norm $\|\cdot\|$. The tangent space $\mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ at $(U,V) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ is written as

$$\begin{aligned}
&\mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C}) \\
&\cong \mathbf{T}_U\mathrm{St}(m,p,\mathbb{C}) \times \mathbf{T}_V\mathrm{St}(n,p,\mathbb{C}) \\
&= \big\{(\xi,\eta) \in \mathbb{C}^{m\times p} \times \mathbb{C}^{n\times p} \,|\, \xi^HU + U^H\xi = \eta^HV + V^H\eta = 0\big\},
\end{aligned} \tag{2.4}$$

where "$\cong$" means the identification of two sets. In what follows, we denote by $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ the Riemannian metric and its induced norm on $\mathrm{St}(m,p,\mathbb{C})\times\mathrm{St}(n,p,\mathbb{C})$, respectively. Under the Riemannian metric (2.3), the orthogonal projection map onto the tangent space at (U,V) acts on $(B,C)\in\mathbb{C}^{m\times p}\times\mathbb{C}^{n\times p}$ as

$$\mathbf{P}_{(U,V)}(B,C)=\big(\mathbf{P}_U(B),\mathbf{P}_V(C)\big), \tag{2.5}$$

where

$$\mathbf{P}_U(B)=B-U\,\mathrm{her}(U^HB),\quad \mathbf{P}_V(C)=C-V\,\mathrm{her}(V^HC) \tag{2.6}$$

are orthogonal projections onto $\mathbf{T}_U\mathrm{St}(m,p,\mathbb{C})$ and $\mathbf{T}_V\mathrm{St}(n,p,\mathbb{C})$, respectively, and where $\mathrm{her}(W):=(W+W^H)/2$ denotes the Hermitian part of a square matrix W. In this paper we adapt the well-known Cayley transform for our retraction [38]. We next introduce this retraction on the product manifold at the current point (U,V) for descent direction (ξ,η), one can refer to [38, 42] for more details. For all $(\xi,\eta)\in\mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C})\times\mathrm{St}(n,p,\mathbb{C})$, it is easy to prove that (ξ,η) can be expressed as

$$(\xi,\eta)=(W_\xi U,\,W_\eta V),$$

where

$$\begin{aligned}W_\xi&=\Pi_U\xi U^H-U\xi^H\Pi_U,\quad \Pi_U=I_m-\frac{1}{2}UU^H,\\ W_\eta&=\Pi_V\eta V^H-V\eta^H\Pi_V,\quad \Pi_V=I_n-\frac{1}{2}VV^H.\end{aligned}$$

This leads to a retraction on the product manifold

$$\mathcal{R}_{(U,V)}(\xi,\eta):=\big(\mathcal{R}_U(\xi),\,\mathcal{R}_V(\eta)\big), \tag{2.7}$$

where

$$\mathcal{R}_U(\xi)=\left(I_m-\frac{1}{2}W_\xi\right)^{-1}\left(I_m+\frac{1}{2}W_\xi\right)U, \tag{2.8}$$

$$\mathcal{R}_V(\eta)=\left(I_n-\frac{1}{2}W_\eta\right)^{-1}\left(I_n+\frac{1}{2}W_\eta\right)V, \tag{2.9}$$

which are the so-called Cayley transforms. Wen and Yin [38] proposed a refinement for (2.8) or (2.9) in the case of low-rank matrices. Rewriting W_ξ as $W_\xi=P_\xi Q_\xi^H\in\mathbb{C}^{m\times m}$, where $P_\xi=[\Pi_U\xi U]\in\mathbb{C}^{m\times 2p}$ and $Q_\xi=[U-\Pi_U\xi]\in\mathbb{C}^{m\times 2p}$, then apply the Sherman-Morrison-Woodbury (SMW) formula

$$\left(I_m-\frac{1}{2}W_\xi\right)^{-1}=\left(I_m-\frac{1}{2}P_\xi Q_\xi^H\right)^{-1}=I_m+\frac{1}{2}P_\xi\left(I_m-\frac{1}{2}Q_\xi^HP_\xi\right)^{-1}Q_\xi^H, \tag{2.10}$$

we can derive the following refinement scheme:

$$\mathcal{R}_U(\xi)=U+P_\xi\left(I_m-\frac{1}{2}Q_\xi^HP_\xi\right)^{-1}Q_\xi^HU. \tag{2.11}$$

Similarly, we can obtain

$$\mathcal{R}_V(\eta)=V+P_\eta\left(I_n-\frac{1}{2}Q_\eta^HP_\eta\right)^{-1}Q_\eta^HV, \tag{2.12}$$

where $P_\eta = [\Pi_V \eta V] \in \mathbb{C}^{n\times 2p}, Q_\eta = [V - \Pi_V \eta] \in \mathbb{C}^{n\times 2p}$. Apparently, if $p \ll m/2$ ($p \ll n/2$), inverting $I_m - Q_\xi^H P_\xi/2 \in \mathbb{C}^{2p\times 2p}$ ($I_n - Q_\eta^H P_\eta/2 \in \mathbb{C}^{2p\times 2p}$) is much easier than inverting $I_m - W_\xi/2 \in \mathbb{C}^{m\times m}$ ($I_n - W_\eta/2 \in \mathbb{C}^{n\times n}$), hence, (2.11) ((2.12)) should be used to compute $\mathcal{R}_U(\xi)$ ($\mathcal{R}_V(\eta)$). However, if $p \geq m/2$ ($p \geq n/2$), then (2.11) ((2.12)) has no advantage over (2.8) ((2.9)), if this is the case, we still use (2.8) ((2.9)) to compute $\mathcal{R}_U(\xi)$ ($\mathcal{R}_V(\eta)$).

2.2. Riemannian gradient and Hessian

The Riemannian gradient and Hessian of an objective function are basic concepts in Riemannian optimization, we next derive the representation matrix formulas of Riemannian gradient and Hessian of the objective function in the problem (1.2). The Riemannian gradient, $\mathrm{grad}f(U,V)$, of an objective function f at $(U,V) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ is defined to be a unique tangent vector which satisfies

$$\langle \mathrm{grad}f(U,V), (\xi,\eta)\rangle = \mathcal{D}f(U,V)[(\xi,\eta)], \quad (\xi,\eta) \in \mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C}).$$

The Hessian, $\mathrm{Hess}f(U,V)$, of f at (U,V) is defined to be a linear transformation of the tangent space $\mathbf{T}_{(U,V)} \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ through the covariant derivative $\Xi_\xi \mathrm{grad}f$ of $\mathrm{grad}f$ evaluated at (U,V)

$$\mathrm{Hess}f(U,V)[(\xi,\eta)] := \Xi_{(\xi,\eta)}\mathrm{grad}f, \quad (\xi,\eta) \in \mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C}),$$

where the covariant derivative is defined through the Levi-Civita connection Ξ on $\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ ([3, Definition 5.5.1]).

In what follows, let f be the objective function of the problem (1.2), i.e.,

$$f(U,V) = -\,\mathrm{Tr}(U^H AV\Theta), \quad (U,V) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C}), \tag{2.13}$$

and let $\overline{f}$ be a function with the same form as f defined in $\mathbb{C}^{m\times p} \times \mathbb{C}^{n\times p}$, that is

$$\overline{f}(U,V) = -\,\mathrm{Tr}(U^H AV\Theta), \quad (U,V) \in \mathbb{C}^{m\times p} \times \mathbb{C}^{n\times p}. \tag{2.14}$$

Thus, the objective function $f(U,V)$ are the restrictions of $\overline{f}$ onto $\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$, i.e., $f = \overline{f}|_{\mathrm{St}(m,p,\mathbb{C})\times\mathrm{St}(n,p,\mathbb{C})}$. Let $(U,V) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$, expression for the Riemannian gradient is given in [30] as

$$\mathrm{grad}f(U,V) = \mathbf{P}_{(U,V)}\big(\nabla\overline{f}(U,V)\big), \tag{2.15}$$

where $\nabla\overline{f}(U,V)$ is the Euclidean gradient of $\overline{f}$ on $\mathbb{C}^{m\times p} \times \mathbb{C}^{n\times p}$, which is computed as

$$\nabla\overline{f}(U,V) := \big(\nabla_U\overline{f}(U,V), \nabla_V\overline{f}(U,V)\big) \tag{2.16}$$

with

$$\nabla_U\bar{f}(U,V) = -AV\Theta, \quad \nabla_V\bar{f}(U,V) = -A^H U\Theta. \tag{2.17}$$

Let (ξ,η) be a tangent vector at $(U,V) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$, by choosing the covariant derivative $\Xi_{(\xi,\eta)}\zeta := \mathbf{P}_{(U,V)}(\mathcal{D}\zeta(U,V)[(\xi,\eta)])$ ([4, §5.3.3]), expression for the Riemannian Hessian is given in [4, 28] as

$$\begin{aligned}\mathrm{Hess}f(U,V)[(\xi,\eta)] &= \mathbf{P}_{(U,V)}\big(\mathcal{D}(\mathrm{grad}f)(U,V)[(\xi,\eta)]\big)\\ &= \mathbf{P}_{(U,V)}\Big(\mathcal{D}(\nabla\overline{f})(U,V)[(\xi,\eta)] + \mathcal{D}\mathbf{P}_{(U,V)}[(\xi,\eta)]\big(\nabla\overline{f}(U,V)\big)\Big),\end{aligned} \tag{2.18}$$

where the Frechét derivative $\mathcal{D}(\nabla\overline{f})(U,V)[(\xi,\eta)]$ is written as

$$\mathcal{D}(\nabla\overline{f})(U,V)[(\xi,\eta)] = (-A\eta\Theta, -A^H\xi\Theta), \tag{2.19}$$

and where

$$\mathcal{D}\mathbf{P}_{(U,V)}[(\xi,\eta)]\big(\nabla\overline{f}(U,V)\big) = \big(\xi\,\mathrm{her}(U^HAV\Theta), \eta\,\mathrm{her}(V^HA^HU\Theta)\big). \tag{2.20}$$

Thus, we obtain a more concrete expression for $\mathrm{Hess}f(U,V)[(\xi,\eta)]$ acts on (ξ,η) as

$$\mathrm{Hess}f(U,V)[(\xi,\eta)] = \Big(\mathbf{P}_U\big(\xi\,\mathrm{her}(U^HAV\Theta) - A\eta\Theta\big), \mathbf{P}_V\big(\eta\,\mathrm{her}(V^HA^HU\Theta) - A^H\xi\Theta\big)\Big). \tag{2.21}$$

We can derive an alternative representation of the Hessian $\mathrm{Hess}f$ by the following lemma.

Lemma 2.1 ([3, p. 42])**.** *An equivalent form of the tangent space to* $\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ *at* $(U,V) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ *is given by*

$$\begin{aligned}&\mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})\\ &= \big\{(UE + U_\perp F, VM + V_\perp N) \mid E, M \in \mathbb{AH}^{p\times p}, F \in \mathbb{C}^{(m-p)\times p}, N \in \mathbb{C}^{(n-p)\times p}\big\},\end{aligned}$$

where $U_\perp$ *is an* $m\times(m-p)$ *orthogonal matrix such that* $UU^H + U_\perp U_\perp^H = I_m$, $V_\perp$ *is an* $n\times(n-p)$ *orthogonal matrix such that* $VV^H + V_\perp V_\perp^H = I_n$ *and* $\mathbb{AH}^{p\times p}$ *denotes the set of all* $p\times p$ *skew-Hermitian matrices.*

Note that for any $M = \mathrm{Re}(M) + i\,\mathrm{Im}(M) \in \mathbb{AH}^{p\times p}$, by $M^H = -M$, we have $\mathrm{Re}(M)^T = -\mathrm{Re}(M)$ and $\mathrm{Im}(M)^T = \mathrm{Im}(M)$. From Lemma 2.1, one can easy to check that the vector space of all tangent vectors has a dimension of

$$2\left(\frac{p(p-1)}{2} + \frac{p(p+1)}{2}\right) + 2(m-p)p + 2(n-p)p = 2p(m+n-p). \tag{2.22}$$

Let $(\xi,\eta) \in \mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ be the search direction at $(U,V) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$. From Lemma 2.1, (ξ,η) can be expressed by

$$(\xi,\eta) = (UE + U_\perp F, VM + V_\perp N), \tag{2.23}$$

where $E, M \in \mathbb{AH}^{p\times p}, F \in \mathbb{C}^{(m-p)\times p}$ and $N \in \mathbb{C}^{(n-p)\times p}$. Since $\mathrm{Hess}f(U,V)[(\xi,\eta)]$ is also determined to be a unique tangent vector in $\mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$, there exists unique matrices $E_H, M_H \in \mathbb{AH}^{p\times p}, F_H \in \mathbb{C}^{(m-p)\times p}, N_H \in \mathbb{C}^{(n-p)\times p}$, which satisfy

$$\mathrm{Hess}f(U,V)[(\xi,\eta)] = (UE_H + U_\perp F_H, VM_H + V_\perp N_H). \tag{2.24}$$

The following proposition shows that we can write E_H, M_H, F_H, N_H by using E, M, F, N, whose analytic framework can be regarded as an extension of the real case discussion in [5] to the complex case. To make the paper self-contained, we briefly write the proof here.

Proposition 2.1. *Let* $(U,V) \in \mathrm{St}(m,p,\mathbb{C})\times\mathrm{St}(n,p,\mathbb{C}), U_\perp \in \mathrm{St}(m-p,p,\mathbb{C})$ *satisfy* $U^HU_\perp = 0$, *and* $V_\perp \in \mathrm{St}(n-p,p,\mathbb{C})$ *satisfy* $V^HV_\perp = 0$. *If a tangent vector* $(\xi,\eta) \in \mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C})$ $\times\mathrm{St}(n,p,\mathbb{C})$ *is expressed as* (2.23)*, then the Hessian* $\mathrm{Hess}f(U,V)$ *of the objective function acts on* (ξ,η) *as in* (2.24) *with*

$$E_H = \mathrm{skewH}\big(E\,\mathrm{her}(U^HAV\Theta) - U^HAVM\Theta - U^HAV_\perp N\Theta\big), \tag{2.25}$$

$$F_H = F\,\mathrm{her}(U^HAV\Theta) - U_\perp^HAVM\Theta - U_\perp^HAV_\perp N\Theta, \tag{2.26}$$

$$M_H = \mathrm{skewH}\big(M\,\mathrm{her}(V^HA^HU\Theta) - V^HA^HUE\Theta - V^HA^HU_\perp F\Theta\big), \tag{2.27}$$

$$N_H = N\,\mathrm{her}(V^HA^HU\Theta) - V_\perp^HA^HUE\Theta - V_\perp^HA^HV_\perp F\Theta, \tag{2.28}$$

where $\mathrm{skewH}(W) := (W - W^H)/2$ *denotes the skew-Hermitian parts of a square matrix* W.

Proof. From (2.21) and (2.24), we have the following equality:

$$UE_H + U_\perp F_H = \mathbf{P}_U\big(\xi\,\mathrm{her}(U^H AV\Theta) - A\eta\Theta\big), \tag{2.29}$$
$$VM_H + V_\perp N_H = \mathbf{P}_V\big(\eta\,\mathrm{her}(V^H A^H U\Theta) - A^H\xi\Theta\big). \tag{2.30}$$

Note that $U^H\mathbf{P}_U(W) = \mathrm{skewH}\left(U^H W\right), U_\perp^H\mathbf{P}_U(W) = U_\perp^T W$ for any $W \in \mathbb{C}^{n\times p}$. Multiplying Eq. (2.29) by U^H from the left and using the relations $U^H U = I_p$ and $U^H U_\perp = 0$ yields

$$E_H = U^H\mathbf{P}_U\big(\xi\,\mathrm{her}(U^H AV\Theta) - A\eta\Theta\big) = \mathrm{skewH}\left(U^H\xi\,\mathrm{her}(U^H AV\Theta) - U^H A\eta\Theta\right). \tag{2.31}$$

By substituting (2.23) into (2.31), and using the relation $U^H\xi = U^H(UE + U_\perp F) = E$, we obtain the expression of E_H in (2.25), by using E and F. Similarly, we multiply (2.30) by $U_\perp^H$ from the left to obtain

$$F_H = U_\perp^H\mathbf{P}_U\big(\xi\,\mathrm{her}(U^H AV\Theta) - A\eta\Theta\big) = U_\perp^H\xi\,\mathrm{her}(U^H AV\Theta) - U_\perp^H A\eta\Theta. \tag{2.32}$$

By substituting (2.23) into (2.32), and using the relation $U_\perp^H\xi = U_\perp^T(UE + U_\perp F) = F$, we obtain the expression of F_H in (2.26). Based on the same analogy as used for the derivation of (2.31) and (2.32), we also obtain the following equations using (2.30):

$$M_H = \mathrm{skewH}\left(V^H\eta\,\mathrm{her}(V^H A^H U\Theta) - V^H A^H\xi\Theta\right), \tag{2.33}$$
$$N_H = V_\perp^H\eta\,\mathrm{her}(V^H A^H U\Theta) - V_\perp^H A^H\xi\Theta. \tag{2.34}$$

By substituting (2.23) into (2.33) and (2.34), and using the relation $V^H\eta = M$ and $V_\perp^H\eta = N$, we obtain (2.27) and (2.28). □

2.3. Riemannian trust-region method for the problem (1.2)

Since we have already obtained the matrix expressions of grad$f$ and Hess$f$, and some other requisites for Riemannian optimization algorithms, we next develop the Riemannian trust-region method for the problem (1.2). The trust-region method is an iterative method for minimizing a cost function. At each iteration step a quadratic model of the cost function is obtained. This model is assumed to be suitable in a region (the trust-region) around the current iterate. Then an update is computed as the minimizer of the model in the trust region. The quality of the trial update is evaluated; it is consequently accepted or rejected, and the trust-region radius is adjusted.

In a Euclidean space $\mathbb{E}$, if $f : \mathbb{E} \to \mathbb{R}$ and $\langle\cdot,\cdot\rangle$ is the inner product in $\mathbb{E}$, the trust-region subproblem for finding the update $\xi \in \mathbb{E}$ for the current iterate $x_k \in \mathbb{E}$ is given by

$$\begin{aligned}&\underset{\xi\in\mathbb{E}}{\mathrm{minimize}}\, m(\xi) = f(x_k) + Df(x_k)[\xi] + \frac{1}{2}D^2 f(x_k)[\xi,\xi],\\ &\text{subject to :}\quad \langle\xi,\xi\rangle \le \Delta_k^2,\end{aligned}$$

where $Df(x_k)[z]$ denotes the directional derivative of the function f at x_k in the direction of z, Δ_k is the trust-region radius. The quality of the model m is evaluated by means of the quotient

$$\rho_k = \frac{f(x_k) - f(x_k + \xi)}{m(0) - m(\xi)}.$$

If ρ is close to 0 or negative, then the model is very inaccurate, i.e., the step must be rejected, and the trust-region radius must be reduced. If ρ is larger but still small, the step is accepted, and the trust-region radius is reduced. Finally, if ρ is close to 1, then there is a good correspondence between the model and the cost function; the step is accepted, and the trust-region radius can be increased.

In the Riemannian trust-region method, at the k-th iteration $(U^{(k)}, V^{(k)})$, by utilizing the Taylor expansion on the product manifold, we consider the following trust-region subproblem on the tangent space:

$$\begin{aligned} &\text{minimize}\, M_{(U^{(k)},V^{(k)})}(\xi,\eta) \\ &\text{subject to :}\quad (\xi,\eta) \in \mathbf{T}_{(U^{(k)},V^{(k)})}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C}), \\ &\qquad\qquad\quad \|(\xi,\eta)\| = \sqrt{\langle(\xi,\eta),(\xi,\eta)\rangle} \le \Delta^{(k)}, \end{aligned} \tag{2.35}$$

where

$$\begin{aligned} M_{(U^{(k)},V^{(k)})}(\xi,\eta) = f(U^{(k)},V^{(k)}) &+ \left\langle \mathrm{grad}f(U^{(k)},V^{(k)}),(\xi,\eta)\right\rangle \\ &+ \frac{1}{2}\left\langle \mathrm{Hess}f(U^{(k)},V^{(k)})[(\xi,\eta)],(\xi,\eta)\right\rangle. \end{aligned}$$

From (2.23) and (2.24) we have

$$\begin{aligned} &\left\langle \mathrm{Hess}f(U^{(k)},V^{(k)})[(\xi,\eta)],(\xi,\eta)\right\rangle \\ &= \left\langle \left(U^{(k)}E_H^{(k)} + U_\perp^{(k)}F_H^{(k)},\ V^{(k)}M_H^{(k)} + V_\perp^{(k)}N_H^{(k)}\right), \left(U^{(k)}E + U_\perp^{(k)}F,\ V^{(k)}M + V_\perp^{(k)}N\right)\right\rangle \\ &= \mathrm{Tr}\left({E_H^{(k)}}^H E + {F_H^{(k)}}^H F + {M_H^{(k)}}^H M + {N_H^{(k)}}^H N \right), \end{aligned}$$

where the matrices E_H, F_H, M_H, N_H are given in Proposition 2.1. On the other hand, since $\mathrm{grad}f(U^{(k)},V^{(k)}) \in \mathbf{T}_{(U^{(k)},V^{(k)})}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$, then from Lemma 2.1, there exists unique matrices $E_g^{(k)}$, $M_g^{(k)} \in \mathbb{AH}^{p\times p}$, $F_g^{(k)} \in \mathbb{C}^{(m-p)\times p}$ and $N_g^{(k)} \in \mathbb{C}^{(n-p)\times p}$ such that $\mathrm{grad}f(U^{(k)},V^{(k)})$ can also be expressed as

$$\mathrm{grad}f(U^{(k)},V^{(k)}) = \left(U^{(k)}E_g^{(k)} + U_\perp^{(k)}F_g^{(k)}, V^{(k)}M_g^{(k)} + V_\perp^{(k)}N_g^{(k)}\right).$$

Similar to the proof of Proposition 2.1, the matrices E_g, F_g, M_g, N_g can be computed as follows:

$$\begin{aligned} E_g &= -\,\mathrm{skewH}(U^H A V\Theta), & F_g &= -U_\perp^H A V\Theta, \\ M_g &= -\,\mathrm{skewH}(V^H A^H U\Theta), & N_g &= -V_\perp^H A^H U\Theta. \end{aligned}$$

Then we have

$$\begin{aligned} &\left\langle \mathrm{grad}f(U^{(k)},V^{(k)}),(\xi,\eta)\right\rangle \\ &= \left\langle \left(U^{(k)}E_g^{(k)} + U_\perp^{(k)}F_g^{(k)},\ V^{(k)}M_g^{(k)} + V_\perp^{(k)}N_g^{(k)}\right), \left(U^{(k)}E + U_\perp^{(k)}F, V^{(k)}M + V_\perp^{(k)}N\right)\right\rangle \\ &= \mathrm{Tr}\left({E_g^{(k)}}^H E + {F_g^{(k)}}^H F + {M_g^{(k)}}^H M + {N_g^{(k)}}^H N \right). \end{aligned}$$

Therefore, the Riemannian trust-region subproblem (2.35) can be rewritten as

$$\begin{aligned} &\text{minimize}\, M_{(U^{(k)},V^{(k)})}(E,F,M,N) \\ &\text{subject to :}\quad E, M \in \mathbb{AH}^{p\times p},\quad F \in \mathbb{C}^{(m-p)\times p},\quad N \in \mathbb{C}^{(n-p)\times p}, \\ &\qquad\qquad\quad \mathrm{Tr}(E^H E + F^H F + M^H M + N^H N) \le {\Delta^{(k)}}^2, \end{aligned} \tag{2.36}$$

where

$$M_{(U^{(k)},V^{(k)})}(E,F,M,N) = f(U^{(k)},V^{(k)}) + \operatorname{Tr}\left({E_t^{(k)}}^H E + {F_t^{(k)}}^H F + {M_t^{(k)}}^H M + {N_t^{(k)}}^H N\right)$$

with

$$\begin{aligned} E_t^{(k)} &= E_g^{(k)} + \frac{1}{2}E_H^{(k)}, & F_t^{(k)} &= F_g^{(k)} + \frac{1}{2}F_H^{(k)}, \\ M_t^{(k)} &= M_g^{(k)} + \frac{1}{2}M_H^{(k)}, & N_t^{(k)} &= N_g^{(k)} + \frac{1}{2}N_H^{(k)}. \end{aligned}$$

In order to provide a guidance for selecting the new trust-region radius $\Delta^{(k)}$, we introduce the quotient

$$\rho^{(k)} = \frac{f(U^{(k)},V^{(k)}) - f\big(\mathcal{R}_{(U^{(k)},V^{(k)})}(\xi^{(k)},\eta^{(k)})\big)}{M_{(U^{(k)},V^{(k)})}(0_{(U^{(k)},V^{(k)})}) - M_{(U^{(k)},V^{(k)})}(\xi^{(k)},\eta^{(k)})},$$

which is also used to judge the acceptance or rejection of the candidate $\mathcal{R}_{(U^{(k)},V^{(k)})}(\xi^{(k)},\eta^{(k)})$ with

$$\big(\xi^{(k)},\eta^{(k)}\big) = \left(U^{(k)}E^{(k)} + U_\perp^{(k)}F^{(k)}, V^{(k)}M^{(k)} + V_\perp^{(k)}N^{(k)}\right)$$

and $(E^{(k)},F^{(k)},M^{(k)},N^{(k)})$ is a solution of (2.36). Due to the fact that $M_{(U^{(k)},V^{(k)})}(0_{(U^{(k)},V^{(k)})})$ $= f(U^{(k)},V^{(k)})$, $\rho^{(k)}$ can be given by

$$\rho^{(k)} = \frac{f\big(\mathcal{R}_{(U^{(k)},V^{(k)})}(\xi^{(k)},\eta^{(k)})\big) - f(U^{(k)},V^{(k)})}{\operatorname{Tr}\left({E_t^{(k)}}^H E^{(k)} + {F_t^{(k)}}^H F^{(k)} + {M_t^{(k)}}^H M^{(k)} + {N_t^{(k)}}^H N^{(k)}\right)}. \tag{2.37}$$

As in the Euclidean space setting, the constants 1/4 and 3/4 are compared with the ratio $\rho^{(k)}$ and the result determines the trust-region radius in the next iteration. Except that, the constant $\rho' \in [0,1/4)$ is used to measure $\rho^{(k)}$. The trust-region step will be taken as the next iteration if $\rho^{(k)} > \rho'$, and rejected, otherwise. Specifically, we present the Riemannian trust-region method for the problem (1.2) as follows.

The trust-region subproblem (2.36) is solved iteratively and forms the inner iteration of Algorithm 2.1. Note that (2.36) is a minimization problem in $\mathbb{AH}^{p\times p}\times\mathbb{C}^{(m-p)\times p}\times\mathbb{AH}^{p\times p}\times\mathbb{C}^{(n-p)\times p}$ and there are varieties of iterative routines to approach the exact minimum to arbitrarily high precision. Of particularly efficient and appropriate is the truncated conjugate-gradient (tCG) method due to the following reasons:

1. The tCG is a Krylov subspace based solver in which if the initial guess $E^0 \in \mathbb{AH}^{p\times p}$, $F^0 \in \mathbb{C}^{(m-p)\times p}$, $M^0 \in \mathbb{AH}^{p\times p}$, $N^0 \in \mathbb{C}^{(n-p)\times p}$ then the sequence $\{E^j\},\{F^j\},\{M^j\},\{N^j\}$ generated by tCG are always satisfy $E^j \in \mathbb{AH}^{p\times p}, F^j \in \mathbb{C}^{(m-p)\times p}, M^j \in \mathbb{AH}^{p\times p}$ and $N^j \in \mathbb{C}^{(n-p)\times p}$, and hence the approximation

$$(\xi^j,\eta^j) = \left(U^{(k)}E^j + U_\perp^{(k)}F^j, V^{(k)}M^j + V_\perp^{(k)}N^j\right) \in \mathbf{T}_{(U^{(k)},V^{(k)})}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C}).$$

2. The sequence $\{M_{(U^{(k)},V^{(k)})}(E^j,F^j,M^j,N^j)\}$ is strictly decreasing while

$$\operatorname{Tr}\left({E^j}^H E^j + {F^j}^H F^j + {M^j}^H M^j + {N^j}^H N^j\right)$$

is strictly increasing, which makes the RTR a descent approach and guarantees the global convergence (see [3, Proposition 7.3.2.]).

Algorithm 2.1: Riemannian trust-region method for the problem (1.2).

```
1  Choose parameters: $\bar{\Delta} > 0$, $\Delta_0 \in (0, \bar{\Delta})$, $\rho' \in [0, 1/4)$, and an initial point
   $(V^{(0)}, P^{(0)}) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$.
2  for $k = 0, 1, 2, \ldots$ do
3      Obtain $(\xi^{(k)}, \eta^{(k)}) = (U^{(k)}E^{(k)} + U_\perp^{(k)}F^{(k)},\ V^{(k)}M^{(k)} + V_\perp^{(k)}N^{(k)})$ by solving the
        trust-region subproblem (2.36). An approximate solution is given by the tCG
        algorithm (Algorithm 2.2).
4      Evaluate $\rho^{(k)}$ from (2.37).
5      if $\rho^{(k)} < 1/4$ then
6          $\Delta^{(k+1)} = \Delta^{(k)}/4$
7      else
8          if $\rho^{(k)} > 3/4$ and $\|(\xi^{(k)}, \eta^{(k)})\| = \Delta^{(k)}$ then
9              $\Delta^{(k+1)} = \min(2\Delta^{(k)}, \bar{\Delta})$
10         else
11             $\Delta^{(k+1)} = \Delta^{(k)}$
12         end if
13     end if
14     if $\rho^{(k)} > \rho'$ then
15         $(U^{(k+1)}, V^{(k+1)}) = \mathcal{R}_{(U^{(k)}, V^{(k)})}(\xi^{(k)}, \eta^{(k)})$
16     else
17         $(U^{(k+1)}, V^{(k+1)}) = (U^{(k)}, V^{(k)})$
18     end if
19 end for
```

The pseudo-code of tCG is presented in Algorithm 2.2 for completeness. Note that we use indices in superscript without round brackets to denote the evolution of E, F, M, N within the inner iteration, while superscripts with round brackets are used in the outer iteration. We should point out that finding a nearly exact solution to the subproblem is neither necessary for the overall convergence nor favourable in terms of computation speed. Actually, one can reduce the computational cost of the trust-region algorithm without losing its fast local convergence rate. This can be done by choosing a stopping criterion for the inner iteration (Algorithm 2.2) based on the gradient of the cost function as in [2]. In this case, few inner tCG steps are taken when the outer iterate is far away from the solution, i.e., when the gradient is large, and more inner tCG steps are taken close to the solution (this behavior is illustrated in Fig. 2.1). As suggested by Absil *et al.* [1–3], the algorithm can stop in either after a fixed number of iterations, or by the criterion

$$\sqrt{\mathrm{Tr}\left(R_{E,j}^H R_{E,j} + R_{F,j}^H R_{F,j} + S_{M,j}^H S_{M,j} + S_{N,j}^H S_{N,j}\right)} \le \max\left(\|R_0\| \min(\|R_0\|^\theta, \kappa)\right), \tag{2.38}$$

where $\kappa, \theta > 0$ are real parameters and

$$\|R_0\| = \sqrt{\mathrm{Tr}\left(R_{E,0}^H R_{E,0} + R_{F,0}^H R_{F,0} + S_{M,0}^H S_{M,0} + S_{N,0}^H S_{N,0}\right)}.$$

In our numerical testing, we set $\kappa = 0.1$ and $\theta = 1$.

Remark 2.1. Fix k, and let $(\xi^j, \eta^j) \in \mathbf{T}_{(U^{(k)}, V^{(k)})}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ be the j-th iterate of the tCG method for the subproblem expressed by (2.35). In the existing RTR method, which

Algorithm 2.2: The truncated CG (tCG) method for the trust-region subproblem (2.36).

```
1  Initialization: set
       $E^0 = \mathbf{0}$,  $F^0 = \mathbf{0}$,  $M^0 = \mathbf{0}$,  $N^0 = \mathbf{0}$;
       $R_{E,0} = E_g^{(k)}$,  $R_{F,0} = F_g^{(k)}$  $S_{M,0} = M_g^{(k)}$,  $S_{N,0} = N_g^{(k)}$;
       $\delta_{E,0} = -R_{E,0}$,  $\delta_{F,0} = -R_{F,0}$,  $\phi_{M,0} = -S_{M,0}$,  $\phi_{N,0} = -S_{N,0}$.
2  for $j = 0, 1, 2, \ldots$ do
3      if a stopping criterion is satisfied then
4          Return $E, F, M, N$.
5      else
6          if $\mathrm{Tr}(\delta_{E,j}^H E_{\delta,j} + \delta_{F,j}^H F_{\delta,j} + \phi_{M,j}^H M_{\phi,j} + \phi_{N,j}^H N_{\phi,j}) \leq 0$ (negative curvature) then
7              Compute $\tau$ such that
                   $E = E^j + \tau\delta_{E,j}, \quad F = F^j + \tau\delta_{F,j}, \quad M = M^j + \tau\phi_{M,j}, \quad N = N^j + \tau\phi_{N,j}$
               minimize $M_{(U^{(k)},V^{(k)})}(E, F, M, N)$ in (2.36) and satisfies
               $\mathrm{Tr}(E^H E + F^H F + M^H M + N^H N) = {\Delta^{(k)}}^2$.
8              Return $E, F, M, N$.
9          else
10             Set
                   $\alpha_j = \dfrac{\mathrm{Tr}\left(R_{E,j}^H R_{E,j} + R_{F,j}^H R_{F,j} + S_{M,j}^H S_{M,j} + S_{N,j}^H S_{N,j}\right)}{\mathrm{Tr}\left(\delta_{E,j}^H E_{\delta,j} + \delta_{F,j}^H F_{\delta,j} + \phi_{M,j}^H M_{\phi,j} + \phi_{N,j}^H N_{\phi,j}\right)}$.
11             Set $E^{j+1} = E^j + \alpha_j\delta_{E,j}$, $F^{j+1} = F^j + \alpha_j\delta_{F,j}$, $M^{j+1} = M^j + \alpha_j\phi_{M,j}$,
               $N^{j+1} = N^j + \alpha_j\phi_{N,j}$.
12             if $\mathrm{Tr}({E^{j+1}}^H E^{j+1} + {F^{j+1}}^H F^{j+1} + {M^{j+1}}^H M^{j+1} + {N^{j+1}}^H N^{j+1}) \geq {\Delta^{(k)}}^2$
               (exceeded trust region) then
13                 Compute $\tau \geq 0$ such that
                       $E = E^j + \tau\delta_{E,j}, \quad F = F^j + \tau\delta_{F,j}, \quad M = M^j + \tau\phi_{M,j}, \quad N = N^j + \tau\phi_{N,j}$
                   satisfies $\mathrm{Tr}(E^H E + F^H F + M^H M + N^H N) = {\Delta^{(k)}}^2$.
14                 Return $E, F, M, N$.
15             else
16                 Set $R_{E,j+1} = R_{E,j} + \alpha_j E_{\delta,j}$, $R_{F,j+1} = R_{F,j} + \alpha_j F_{\delta,j}$, $S_{M,j+1} = S_{M,j} + \alpha_j M_{\phi,j}$,
                   $S_{N,j+1} = S_{N,j} + \alpha_j N_{\phi,j}$
17                 Set
                       $\beta_{j+1} = \dfrac{\mathrm{Tr}\left(R_{E,j+1}^H R_{E,j+1} + R_{F,j+1}^H R_{F,j+1} + S_{M,j+1}^H S_{M,j+1} + S_{N,j+1}^H S_{N,j+1}\right)}{\mathrm{Tr}\left(R_{E,j}^H R_{E,j} + R_{F,j}^H R_{F,j} + S_{M,j}^H S_{M,j} + S_{N,j}^H S_{N,j}\right)}$.
                   Set $\delta_{E,j+1} = -R_{E,j+1} + \beta_{j+1}\delta_{E,j}$, $\delta_{F,j+1} = -R_{F,j+1} + \beta_{j+1}\delta_{F,j}$,
                   $\phi_{M,j+1} = -S_{M,j+1} + \beta_{j+1}\phi_{M,j}$, $\phi_{N,j+1} = -S_{N,j+1} + \beta_{j+1}\phi_{N,j}$.
18             end if
19         end if
20     end if
21 end for
```

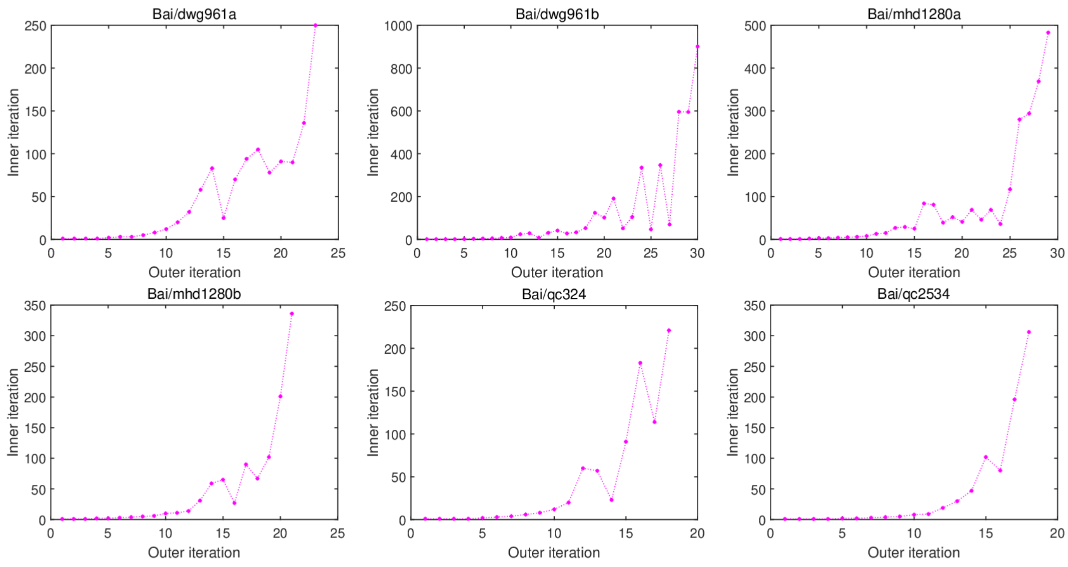

Fig. 2.1. Number of inner tCG steps per trust-region step. The date matrix has elements taken from Bai/qc324, Bai/dwg961a, Bai/dwg961b, Bai/mhd1280a, Bai/mhd1280b and Bai/qc2534, respectively. Few tCG iterations are performed in the first trust-region steps and more in the neighborhood of the solution.

directly solves the subproblem (2.35), (ξ^j, η^j) is updated as a tangent vector in $\mathbf{T}_{(U^{(k)},V^{(k)})}\mathrm{St}(m,p,\mathbb{C})\times\mathrm{St}(n,p,\mathbb{C})$. While in Algorithm 2.2, we update $E^j \in \mathbb{AH}^{p\times p}$, $F^j \in \mathbb{R}^{(m-p)\times p}$, $M^j \in \mathbb{AH}^{p\times p}$ and $N^j \in \mathbb{R}^{(n-p)\times p}$, and where $\xi^j = U^{(k)}E^j + U_\perp^{(k)}F^j$ and $\eta^j = V^{(k)}M^j + V_\perp^{(k)}N^j$. That is, if the tCG method for (2.35) terminates with E^j, F^j, M^j, N^j, we only have to compute $\xi^j = U^{(k)}E^j + U_\perp^{(k)}F^j$ and $\eta^j = V^{(k)}M^j + V_\perp^{(k)}N^j$. We do not have to compute ξ^i and η^i for $i < j$. Furthermore, in the existing RTR method, $\mathrm{Hess}f((U^{(k)},V^{(k)}))[(\delta_j,\phi_j)]$ for some

$$(\delta_j,\phi_j) = \left(U^{(k)}\delta_{E,j} + U_\perp^{(k)}\delta_{F,j}, V^{(k)}\phi_{M,j} + V_\perp^{(k)}\phi_{N,j}\right) \in \mathbf{T}_{(U^{(k)},V^{(k)})}\mathrm{St}(m,p,\mathbb{C})\times\mathrm{St}(n,p,\mathbb{C})$$

must be computed at each iteration of the tCG method. However, the proposed Algorithm 2.2 needs only $E_{\delta,j} \in \mathbb{AH}^{p\times p}$, $F_{\delta,j} \in \mathbb{C}^{(m-p)\times p}$, $M_{\phi,j} \in \mathbb{AH}^{p\times p}$, $N_{\phi,j} \in \mathbb{C}^{(n-p)\times p}$, and where

$$\mathrm{Hess}f\left(U^{(k)},V^{(k)}\right)[(\delta_j,\phi_j)] = \left(U^{(k)}E_{\delta,j} + U_\perp^{(k)}F_{\delta,j}, V^{(k)}M_{\phi,j} + V_\perp^{(k)}N_{\phi,j}\right).$$

Therefore, if the number of iterations in the inner tCG method needed for solving the trust-region subproblems is sufficiently large, the proposed Algorithm 2.2 may have a shorter total computational time than the existing RTR method which directly solves the subproblem (2.35). These facts imply that our proposed method can reduce the computational cost.

Remark 2.2. Similar to unconstrained optimization in the Euclidean space,

$$\|\mathrm{grad}f(U^{(k)},V^{(k)})\| < \varepsilon$$

for some constant $\varepsilon > 0$ is a reasonable stopping criterion for Riemannian optimization approaches to the problem (1.2). In fact, the Lagrangian of the problem (1.2) is

$$\mathcal{L} := \mathcal{L}(U,V,\Lambda,\Omega) = \overline{f}(U,V) - \frac{1}{2}\mathrm{Tr}\left(\Lambda(U^HU - I_p)\right) - \frac{1}{2}\mathrm{Tr}\left(\Omega(V^HV - I_p)\right),$$

where Λ, Ω are two Hermitian matrices representing the Lagrange multipliers. Then the first-order optimality conditions in the Euclidean sense [38] are

$$\begin{aligned} U^H U = I_p, \quad & \frac{\partial}{\partial U}\mathcal{L} = \nabla_U \overline{f} - U \nabla_U \overline{f}^H U = 0, \\ V^H V = I_p, \quad & \frac{\partial}{\partial V}\mathcal{L} = \nabla_V \overline{f} - V \nabla_V \overline{f}^H V = 0 \end{aligned}$$

with $\Lambda = \nabla_U \overline{f}^H U, \Omega = \nabla_V \overline{f}^H V$. Under the conditions $U^H U = I_p$ and $V^H V = I_p$, $\partial \mathcal{L}/\partial U = 0$ and $\partial \mathcal{L}/\partial V = 0$ are equivalent to grad$f(U, V) = 0$, since it follows from (2.15), (2.5) and (2.6) that:

$$\begin{aligned} \mathbf{P}_U(\nabla_U \overline{f}) &= \left(I_m - \frac{1}{2}UU^H\right)\left(\nabla_U \overline{f} - U \nabla_U \overline{f}^H U\right) = \left(I_m - \frac{1}{2}UU^H\right)\frac{\partial}{\partial U}\mathcal{L}, \\ \mathbf{P}_V(\nabla_V \overline{f}) &= \left(I_n - \frac{1}{2}VV^H\right)\left(\nabla_V \overline{f} - V \nabla_V \overline{f}^H V\right) = \left(I_n - \frac{1}{2}VV^H\right)\frac{\partial}{\partial V}\mathcal{L}. \end{aligned}$$

Thus, first-order critical points in the Euclidean sense can be interpreted as stationary points in the Riemannian sense.

The convergence of the general RTR-tCG on a smooth Riemannian manifold has been extensively analyzed in [2,3] where both the global convergence and local convergence rate have been established under appropriate assumptions. Therefore, to understand the performance of the RTR-tCG method for solving the problem (1.2), we only need to check these assumptions, which leads to the following conclusions.

Theorem 2.1 (Global Convergence). *Let $\{(U^{(k)}, V^{(k)})\}$ be a sequence of iterates generated by Algorithms* 2.1-2.2*, and let an iterate be accepted if $\rho^k > \rho'$ with $\rho' \in (0, 1/4)$. Then*

$$\lim_{k\to\infty} \operatorname{grad} f(U^{(k)}, V^{(k)}) = 0.$$

In fact, this conclusion is a straightforward result of [2, Theorem 4.4 and Corollary 4.6], based on the fact that the cost function $f(U, V)$ and the adopted retraction as described in Section 2 are smooth, and the involved product manifold $\mathrm{St}(m, p, \mathbb{C}) \times \mathrm{St}(n, p, \mathbb{C})$ is a smooth and compact Riemannian product manifold.

Theorem 2.2 (Local Convergence Speed). *Consider Algorithms* 2.1-2.2 *with retraction $\mathcal{R}$ as in Section* 2 *and stopping criterion in Algorithm* 2.2 *as in* (2.38)*. Let $(U^*, V^*) \in$ $\mathrm{St}(m, p, \mathbb{C}) \times \mathrm{St}(n, p, \mathbb{C})$ be a nondegenerate local minimum of f, (i.e.,* grad$f(U^*, V^*) = 0$ *and* Hess$f(U^*, V^*)$ *is positive definite). Then there exists $c > 0$ such that for all sequences $\{(U^{(k)}, V^{(k)})\}$ generated by the algorithm converging to (U^*, V^*) and exists $K > 0$ such that for all $k > K$,*

$$\operatorname{dist}\big((U^{(k+1)}, V^{(k+1)}), (U^*, V^*)\big) \le c\left\{\operatorname{dist}\big((U^{(k)}, V^{(k)}), (U^*, V^*)\big)\right\}^{\min\{\theta+1,2\}}$$

with $\theta>0$ as in (2.38)*, and* dist *defines the Riemannian distance on product manifold* $\mathrm{St}(m, p, \mathbb{C})$ $\times \mathrm{St}(n, p, \mathbb{C})$ [3, p. 46].

Theorem 2.2 indicates that superlinear convergence takes place when the convergent point is a nondegenerate local solution, which, in general, is not easy to identify in advance. However, in some special cases, the positive definiteness of the Hessian operator can be ensured at a global solution of the problem (1.2). Actually, Sato *et al.* [30] have described in detail what will happen if the Hessian of f is degenerate at a critical point in the case of A is real, U and $V^T = V^H$ are real orthogonal matrices, they proved that degenerate optimal solutions form a submanifold diffeomorphic to the product of orthogonal groups and Stiefel manifolds of smaller dimension. It then turns out that, according to whether the singular values are distinct or degenerate, the optimal solution set is a discrete finite set or a disconnected submanifold. For more details, we refer the reader to the [30, Propositions 6.1, 6.2] and their proofs. Especially, they pointed out that if the largest p singular values $\sigma_1, \cdots, \sigma_p$ of the given matrix A in question are positive and all distinct, that is, $\sigma_1 > \cdots > \sigma_p > 0$, then, there are 2^p global optimal solutions (U^*, V^*) to the problem (1.2) and the Hessian $\text{Hess}f(U^*, V^*)$ is positive definite on each $\mathbf{T}_{(U^*,V^*)}\text{St}(m,p,\mathbb{R}) \times \text{St}(n,p,\mathbb{R})$. However, we should point out that, for real problems, if the largest p singular values $\sigma_1, \cdots, \sigma_p$ are distinct, then the global optimal solutions are isolated. The equivalent minimizers are given by changing the signs of columns of the solution matrix (U, V). Therefore, they are disconnected. However, for complex problems, the equivalent minimizers are given by (UD, VD), with D being the diagonal matrix with all diagonal entries being norm 1. Therefore, the set of the equivalent minimizers is connected. Thus, the global minimizers are not isolated.

3. Numerical Experiments

In this section, we report the numerical performance of the proposed algorithm for the problem (1.2). All the numerical experiments were completed on a personal computer with a Intel(R) Core(TM)2 Quad of 2.33 GHz CPU and 3.00 GB of RAM equipped with MATLAB R2019b. To illustrate the efficiency of our Algorithm 2.1, we first test it on the problem (1.2) with large sparse data matrix A taken from the UF Sparse Matrix Collection [15], and then report numerical comparison to show advantage of the proposed algorithm over MATLAB's built-in function `svd` or `svds`. We also compare our proposals with some classical Riemannian gradient-type methods which are all applicable to the problem (1.2) with necessary modifications and the existing Riemannian version of limited-memory BFGS algorithms in the MATLAB toolbox Manopt and the Riemannian manifold optimization ROPTLIB. Comparison with some latest infeasible methods for solving manifold optimization are also given to show the merits of the proposed approach.

In the implementation of Algorithm 2.1, we choose the standard parameters in the Manopt Toolbox [10] for handling the Riemannian trust-region scheme, i.e., $\overline{\Delta} = \dim(\mathcal{M}_r)$, where $\mathcal{M}_r$ denotes the underlying product manifold, $\Delta_0 = \overline{\Delta}/8$ and $\rho' = 0.1$. The stopping criteria of Algorithm 2.1 is set to be

$$\|\text{grad}f(U^{(k)}, V^{(k)})\| < \varepsilon$$

with $\varepsilon = 10^{-6}$. All starting points $U^{(0)}, V^{(0)}$ were feasible and generated randomly by means of $U^{(0)} = \texttt{orth}(\texttt{randn}(m,p) + i\texttt{randn}(m,p))$ and $V^{(0)} = \texttt{orth}(\texttt{randn}(n,p) + i\texttt{randn}(n,p))$. Here, the notation follows MATLAB conventions.

3.1. Numerical performance of Algorithm 2.1 for large sparse matrices

In this subsection, we test the performance of our Algorithm 2.1 on the problem (1.2) with the date matrix A using the 26 instances of large sparse matrices taken from the UF Sparse

Table 3.1: Numerical results of 26 instances in the UF Sparse Matrix Collection.

Name	Type	m, n	p	CT.	IT.	T-INIT.	Grad.	Obj.
Bai/qc324	complex symmetric	324, 324	10	7.96	18	808	1.05×10^{-9}	-62.27
Bai/dwg961a	complex symmetric	961, 961	10	8.24	23	1169	4.55×10^{-8}	-26648780.46
Bai/dwg961b	complex symmetric	961, 961	10	24.48	26	3746	1.25×10^{-9}	-30210.82
Bai/mhd1280a	complex unsymmetric	1280, 1280	10	23.33	30	2230	4.23×10^{-9}	-2257843.18
Bai/mhd1280b	complex Hermitian	1280, 1280	10	9.03	23	781	1.86×10^{-10}	-1946.30
Bai/qc2534	complex symmetric	2534, 2534	10	59.94	19	1131	2.93×10^{-11}	-155.37
Bai/bfwa62	real symmetric	62, 62	10	1.44	18	644	1.76×10^{-8}	-450.06
Bai/bfwa398	real symmetric	398, 398	10	10.69	20	2050	2.18×10^{-9}	-553.44
Bai/ck400	real unsymmetric	400, 400	10	14.28	22	2797	7.33×10^{-9}	-165.93
Bai/ck656	real unsymmetric	656, 656	10	18.32	23	2818	1.39×10^{-11}	-165.93
Bai/bfwa782	real unsymmetric	782, 782	10	12.96	22	2122	4.18×10^{-8}	-666.70
Bai/cdde6	real unsymmetric	961, 961	10	43.33	24	7774	8.31×10^{-8}	-424.53
Bai/dw2048	real unsymmetric	2048, 2048	10	378.11	28	27404	2.33×10^{-7}	-52.94
Bai/cryg2500	real unsymmetric	2500, 2500	10	14.57	23	1191	7.34×10^{-10}	-435754.46
Bai/mhd3200a	real unsymmetric	3200, 3200	10	14.61	27	880	1.48×10^{-8}	-2938754.06
DRIVCAV/cavity08	real unsymmetric	1182, 1182	10	6.58	21	712	3.43×10^{-7}	-1723.67
DRIVCAV/cavity10	real unsymmetric	2597, 2597	10	110.31	23	24819	2.10×10^{-7}	-717.00
Boeing/bcsstk34	real symmetric	588, 588	10	18.46	33	2878	9.15×10^{-7}	-1983593969.04
Boeing/msc01050	real symmetric	1050, 1050	10	4.79	24	576	5.93×10^{-7}	-786358036.90
Boeing/msc01440	real symmetric	1440, 1440	10	22.34	23	2022	2.42×10^{-7}	-72290559.82
Boeing/nasa1824	real symmetric	1824, 1824	10	15.26	28	1451	9.79×10^{-7}	-891998058.97
Grund/b_dyn	real unsymmetric	1089, 1089	10	92.60	57	18967	9.37×10^{-7}	-292549.28
Grund/b2_ss	real unsymmetric	1089, 1089	10	17.39	24	1686	1.50×10^{-8}	-10687460.45
Grund/meg1	real unsymmetric	2904 2904	10	29.94	34	2280	4.97×10^{-7}	-29050454.76
Grund/poli	real unsymmetric	4008 4008	10	29.84	20	1250	4.43×10^{-11}	-271.28
Grund/meg4	real symmetric	5860 5860	10	5.16	17	268	8.13×10^{-7}	-11691510.64

Matrix Collection [15]. In addition, we fix $p = 10$ and $\Theta = \operatorname{diag}\{10, 9, \ldots, 1\}$. Table 3.1 reports the numerical results corresponding to this experiment, in which the term "CT." denotes the total computational times, "IT." denotes the outer iteration numbers of Algorithm 2.1, "T-INIT." denotes the total number of inner iterations of Algorithm 2.2, "Grad." denotes the norm of Riemannian gradient $\|\operatorname{grad} f(U^{(k)}, V^{(k)})\|$ and "Obj." means the objective function value $-\operatorname{Tr}({U^{(k)}}^H A V^{(k)} \Theta)$ at the final iterate by implementing the proposed algorithm. From Table 3.1, we see that our Algorithm 2.1 work very efficiently on solving the problem (1.2) for different large sparse data matrix A with different problem sizes, and all the tested problems achieve the required accuracy in terms of the gradient norm.

3.2. Numerical comparison of Algorithm 2.1 with MATLAB's build-in function `svd` or `svds`

In this subsection, we compared the singular value decomposition obtained by MATLAB's build-in `svd` or `svds` function with that obtained by the proposed Algorithm 2.1 to check the accuracy of numerical solutions. We generated the matrix $A \in \mathbb{C}^{m\times n}$ in question of the form

$$A = U_r \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n) V_r^T$$

together with $\Theta = \mathrm{diag}(p, \ldots, 2, 1)$ and fixed $p = 5$, and where $U_r \in \mathbb{C}^{m\times n}$ and $V_r \in \mathbb{C}^{n\times n}$ are orthonormal matrices with randomly chosen elements. Singular values $\sigma_1 \geq \cdots \geq \sigma_n$ are also chosen randomly from the interval $[0, 100]$ under the condition that A has distinct singular values among the largest p singular values of A. In this setting, the optimal solutions of the problem (1.2) are given by $U_{opt} := U_r I_{n,p}$ and $V_{opt} := V_r I_{n,p}$ with

$$I_{n,p} = \begin{pmatrix} I_p \\ 0 \end{pmatrix} \in \mathbb{C}^{n\times p}.$$

Suppose we obtain factors $\tilde{U}, \tilde{\Sigma}$ and $\tilde{V}$ of the truncated singular value decomposition of A by applying MATLAB's `svd` function.

For different parameters m, n and the generated matrix A, let U^* and V^* be matrices obtained by performing the proposed Algorithm 2.1. To see the degree of accuracy, for different stopping criteria ε of Algorithm 2.1, we compare the Frobenius norms

$$\mathrm{norm1} := \left\|\tilde{U}^H A\tilde{V} - U_{opt}^H AV_{opt}\right\|, \quad \mathrm{norm2} = \left\|U^{*H} AV^* - U_{opt}^H AV_{opt}\right\|.$$

The comparison results are reported in Table 3.2, where "CT." denotes the actual total computation times. We can see from Table 3.2 that, the proposed Algorithm 2.1 can generate the

Table 3.2: Comparison results on accuracy of numerical solutions between Algorithm 2.1 and MATLAB's `svd` function.

m, n	p	`svd`		Algorithm 2.1		
		norm1	CT.	ε	norm2	CT.
2000, 10	5	1.57×10^{-13}	0.157	10^{-2}	3.36×10^{-4}	0.423
				10^{-4}	5.40×10^{-8}	0.640
				10^{-6}	1.05×10^{-13}	0.719
				10^{-8}	4.52×10^{-12}	1.010
4000, 20	5	2.22×10^{-13}	0.996	10^{-2}	5.0×10^{-4}	0.821
				10^{-4}	3.46×10^{-6}	1.150
				10^{-6}	5.02×10^{-8}	1.184
				10^{-8}	5.74×10^{-12}	1.801
8000, 30	5	2.14×10^{-13}	6.402	10^{-2}	1.91×10^{-3}	1.216
				10^{-4}	3.38×10^{-10}	1.894
				10^{-6}	5.47×10^{-8}	2.423
				10^{-8}	1.35×10^{-13}	2.858
12000, 40	5	9.41×10^{-14}	10.062	10^{-2}	4.83×10^{-5}	3.212
				10^{-4}	2.52×10^{-8}	4.083
				10^{-6}	3.52×10^{-11}	4.862
				10^{-8}	2.76×10^{-12}	5.703
15000, 50	5	2.39×10^{-13}	35.299	10^{-2}	4.56×10^{-5}	6.871
				10^{-4}	8.82×10^{-10}	5.278
				10^{-6}	1.40×10^{-10}	6.508
				10^{-8}	1.34×10^{-12}	6.804

Table 3.3: Errors of the largest p singular values generated by MATLAB's `svds` function and Algorithm 2.1 of 10 instances in the UF Sparse Matrix Collection.

Name	Type	m, n	p	Error
Bai/qc324	complex symmetric	324, 324	10	1.01×10^{-12}
Bai/dwg961a	complex symmetric	961, 961	10	2.08×10^{-9}
Bai/dwg961b	complex symmetric	961, 961	10	3.80×10^{-10}
Bai/mhd1280a	complex unsymmetric	1280, 1280	10	2.99×10^{-10}
Bai/mhd1280b	complex Hermitian	1280, 1280	10	7.33×10^{-13}
Bai/qc2534	complex symmetric	2534, 2534	10	1.78×10^{-13}
Bai/bfwa62	real symmetric	62, 62	10	5.72×10^{-12}
Bai/bfwa398	real symmetric	398, 398	10	2.66×10^{-11}
Bai/ck400	real unsymmetric	400, 400	10	1.28×10^{-10}
Bai/ck656	real unsymmetric	656, 656	10	1.09×10^{-11}

largest p singular values of each A with comparable accuracy as those generated by MATLAB's `svd` function, and the performance of Algorithm 2.1 slightly better than that of `svd` in terms of computational time, this phenomenon becomes more pronounced as the problem sizes increase. Table 3.3 reported the errors of the largest p singular values of each A generated by Algorithm 2.1 and MATLAB's `svds` function, which is best suited for finding a few singular values of a large sparse matrix. In Table 3.3, the given large sparse data matrices A taken from the UF Sparse Matrix Collection, together with $\Theta = \mathrm{diag}(p, \ldots, 2, 1)$ and fixed $p = 10$.

3.3. Numerical comparison of Algorithm 2.1 with Riemannian gradient-type methods and limited-memory Riemannian BFGS algorithms

To show the efficiency of Algorithm 2.1, in this subsection, we compare it with the existing Riemannian version of limited-memory BFGS algorithms in the MATLAB toolbox Manopt [10] and the Riemannian manifold optimization library ROPTLIB [21–23] (denoted respectively by RLBFGS-Manopt and LRBFGS-ROPTLIB), and some Riemannian gradient-based methods, including OptStiefelGBB, which is a state-of-art algorithm proposed by Wen and Yin [38], and Riemannian Dai's nonmonotone conjugate gradient method adopted in [26], which is denoted by RCG. We first introduce the iterative framework of the two compared Riemannian gradient-based methods for solving the problem (1.2).

The OptStiefelGBB method [38] designed to solve the problem (1.2) use the Cayley transformation (2.7)-(2.9) (or (2.7), (2.11)-(2.12) if $p \ll m/2$ and $p \ll n/2$) to construct the retraction, and update the iterated by using the following scheme starting in $(U^{(0)}, V^{(0)}) \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ with

$$(U^{(k+1)}, V^{(k+1)}) = \mathcal{R}_{(U^{(k)},V^{(k)})}\big(t_k(\xi^{(k)}, \eta^{(k)})\big), \tag{3.1}$$

where

$$\big(\xi^{(k)}, \eta^{(k)}\big) = -\mathrm{grad} f\big(U^{(k)}, V^{(k)}\big) \in \mathbf{T}_{(U^{(k)},V^{(k)})}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$$

is the search direction and t_k is the step size at the current iterate $(U^{(k)}, V^{(k)})$. Similar to Euclidean line search methods, t_k is determined by inexact curvilinear search condition. Given

$\rho, \varrho, \delta \in (0,1)$, the nonmonotone Armijo rules try to find the smallest integer h satisfying

$$f\big(\mathcal{R}_{(U^{(k)},V^{(k)})}\big(t_k(\xi^{(k)},\eta^{(k)})\big)\big) \le C_k + \rho t_k \big\langle \mathrm{grad} f(U^{(k)},V^{(k)}), (\xi^{(k)},\eta^{(k)})\big\rangle, \tag{3.2}$$

where $t_k = \gamma_k \delta^h$ and γ_k is an initial step size. Here, the reference value C_{k+1} is a convex combination of C_k and $f(U^{(k+1)},V^{(k+1)})$ and is calculated via

$$C_{k+1} = \frac{1}{Q_{k+1}}\big(\varrho Q_k C_k + f(U^{(k+1)},V^{(k+1)})\big),$$

where $C_0 = f(U^{(0)},V^{(0)}), Q_{k+1} = \varrho Q_k + 1$ and $Q_0 = 1$. It is well known that an initial step size computed by the Barzilai-Borwein (BB) method often speeds up the convergence in Euclidean optimization. Similarly and as in [38], we consider the following initial step sizes:

$$\gamma_k^{(1)} = \frac{\mathrm{Tr}(S_{u,k-1}^H S_{u,k-1}) + \mathrm{Tr}(S_{v,k-1}^H S_{v,k-1})}{\big|\,\mathrm{Tr}(Z_{u,k-1}^H S_{u,k-1}) + \mathrm{Tr}(Z_{v,k-1}^H S_{v,k-1})\big|} \tag{3.3a}$$

or

$$\gamma_k^{(2)} = \frac{\big|\,\mathrm{Tr}(S_{u,k-1}^H Z_{u,k-1}) + \mathrm{Tr}(S_{v,k-1}^H Z_{v,k-1})\big|}{\mathrm{Tr}(Z_{u,k-1}^H Z_{u,k-1}) + \mathrm{Tr}(Z_{v,k-1}^H Z_{v,k-1})}, \tag{3.3b}$$

where we take

$$\begin{aligned}
(S_{u,k-1}, S_{v,k-1}) &= (U^{(k)} - U^{(k-1)}, V^{(k)} - V^{(k-1)}),\\
(Z_{u,k-1}, Z_{v,k-1}) &= \mathrm{grad} f(U^{(k)},V^{(k)}) - \mathrm{grad} f(U^{(k-1)},V^{(k-1)}).
\end{aligned}$$

The Riemannian case of Dai's nonlinear CG method (RCG) designed to solve the problem (1.2) use the same Cayley transformation as in OptStiefelGBB algorithm to construct the retraction, and the same update scheme (3.1) starting in $(U^{(0)},\ V^{(0)}) \in \mathrm{St}(m,p,\mathbb{C})\times\mathrm{St}(n,p,\mathbb{C})$. The search direction

$$\begin{aligned}
&(\xi^{(0)},\eta^{(0)}) = -\mathrm{grad} f(U^{(0)},V^{(0)}),\\
&(\xi^{(k)},\eta^{(k)}) \in \mathbf{T}_{(U^{(k)},V^{(k)})} \in \mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})
\end{aligned}$$

with $k \ge 1$ is determined by

$$(\xi^{(k+1)},\eta^{(k+1)}) = -\mathrm{grad} f(U^{(k+1)},V^{(k+1)}) + \beta_{k+1}\mathcal{T}_{t_k(\xi^{(k)},\eta^{(k)})}(\xi^{(k)},\eta^{(k)}), \tag{3.4}$$

where t_k is the step size, β_k is a parameter that determines the property of the conjugate gradient method, and $\mathcal{T}$ is a vector transport on $\mathrm{St}(n,p)$ [3], which is a smooth mapping from the product of tangent bundles $\mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C}) \oplus \mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$ to the tangent bundle $\mathbf{T}_{(U,V)}\mathrm{St}(m,p,\mathbb{C}) \times \mathrm{St}(n,p,\mathbb{C})$, where $\oplus$ is the Whitney sum. In (3.4), the vector transport is constructed by the differentiated retraction as follows:

$$\mathcal{T}_{t(\xi,\eta)}(\xi,\eta) = \left(\left(I_m - \frac{t}{2}W_\xi\right)^{-2}\xi,\ \left(I_n - \frac{t}{2}W_\eta\right)^{-2}\eta\right),$$

where

$$W_\xi = P_\xi Q_\xi^H \in \mathbb{C}^{m\times m}, \quad W_\eta = P_\eta Q_\eta^H \in \mathbb{C}^{n\times n}$$

with

$$\begin{aligned} P_\xi &= [\Pi_U \xi, U], \quad Q_\xi = [U, -\Pi_U \xi], \quad \Pi_U = I_m - \frac{1}{2}UU^H, \\ P_\eta &= [\Pi_V \eta, V], \quad Q_\eta = [V, -\Pi_V \eta], \quad \Pi_V = I_n - \frac{1}{2}VV^H. \end{aligned}$$

If $p \ll m/2$ and $p \ll n/2$, similarly and as in [26, 42], applying the SMW format (2.10), the computation of $\mathcal{T}_{t(\xi,\eta)}(\xi,\eta)$ is replaced by its refined scheme

$$\begin{aligned} \mathcal{T}_{t(\xi,\eta)}(\xi,\eta) = \Bigg(& P_\xi \left(M_\xi^1 + \frac{t}{2} M_\xi^2 M_\xi^3 + \frac{t}{2}\left(I_{2p} - \frac{t}{2} M_\xi^2\right)^{-1} M_\xi^2 M_\xi^3 \right), \\ & P_\eta \left(N_\eta^1 + \frac{t}{2} N_\eta^2 N_\eta^3 + \frac{t}{2}\left(I_{2p} - \frac{t}{2} N_\eta^2\right)^{-1} N_\eta^2 N_\eta^3 \right) \Bigg) \end{aligned}$$

with

$$\begin{aligned} M_\xi^1 &= Q_\xi^H U, \quad M_\xi^2 = Q_\xi^H P_\xi, \quad M_\xi^3 = \left(I_{2p} - \frac{t}{2} Q_\xi^H P_\xi\right)^{-1} Q_\xi^H U, \\ N_\eta^1 &= Q_\eta^H V, \quad N_\eta^2 = Q_\eta^H P_\eta, \quad N_\eta^3 = \left(I_{2p} - \frac{t}{2} Q_\eta^H P_\eta\right)^{-1} Q_\eta^H V. \end{aligned}$$

The step size t_k is determined by the following nonmonotone line search condition instead of the Wolfe conditions:

$$\begin{aligned} & f\big(\mathcal{R}_{(U^{(k)},V^{(k)})}\big(t_k(\xi^{(k)}, \eta^{(k)})\big)\big) \\ & \le \max\big\{ f(U^{(k)}, V^{(k)}), \cdots, f(U^{(k-h_k)}, V^{(k-h_k)}) \big\} \\ & \quad + \delta t_k \big\langle \mathrm{grad} f(U^{(k)}, V^{(k)}), (\xi^{(k)}, \eta^{(k)}) \big\rangle, \end{aligned}$$

where $h_k = \min\{h-1, k\}$ and h is a positive integer. The parameter β_{k+1} is computed as $\beta_{k+1} \in [0, \beta_{k+1}^D]$ with

$$\beta_{k+1}^D = \frac{\|\mathrm{grad} f(U^{(k+1)}, V^{(k+1)})\|^2}{\max\big\{T, -\big\langle \mathrm{grad} f(U^{(k)}, V^{(k)}), (\xi^{(k)}, \eta^{(k)}) \big\rangle \big\}},$$

and

$$T = \big\langle \mathrm{grad} f(U^{(k+1)}, V^{(k+1)}), \mathcal{T}_{t_k(\xi^{(k)},\eta^{(k)})}(\xi^{(k)}, \eta^{(k)}) \big\rangle - \big\langle \mathrm{grad} f(U^{(k)}, V^{(k)}), (\xi^{(k)}, \eta^{(k)}) \big\rangle.$$

Since convergence of first-order methods can slow down as the iterates approach a stationary point, in our implementation, for the compared two gradient-based methods, we use the same stopping criterion as that in [26, 38]: we let algorithms run up to K iterations and stop it at iteration $k < K$ if $\|\mathrm{grad} f(U^{(k)}, V^{(k)})\| \le \varepsilon$, or $\mathrm{rel}_{(U,V)}^k \le \epsilon_{uv}$ and $\mathrm{rel}_f^k \le \epsilon_f$, or

$$\mathrm{mean}\left\{ \mathrm{rel}_{(U,V)}^{k-\min\{k,T\}+1}, \cdots, \mathrm{rel}_{(U,V)}^k \right\} \le 10\epsilon_{uv}, \quad \mathrm{mean}\left\{ \mathrm{rel}_f^{k-\min\{k,T\}+1}, \cdots, \mathrm{rel}_f^k \right\} \le 10\epsilon_f$$

for some constants $\epsilon, \epsilon_{uv}, \epsilon_f \in (0,1)$, and $T, K \in \mathbb{N}^+$, where

$$\begin{aligned} \mathrm{rel}_{(U,V)}^k &= \frac{\|U^{(k+1)} - U^{(k)}\|}{\sqrt{m}} + \frac{\|V^{(k+1)} - V^{(k)}\|}{\sqrt{n}}, \\ \mathrm{rel}_f^k &= \frac{|f(U^{(k+1)}, V^{(k+1)}) - f(U^{(k)}, V^{(k)})|}{|f(U^{(k)}, V^{(k)})| + 1}. \end{aligned}$$

The default values of $\epsilon_{uv}, \epsilon_f, T$ and K are $10^{-10}, 10^{-12}, 5$ and 20000, respectively. In addition, all the parameters in the compared OptStiefel and RCG methods are chosen in a standard way as in their corresponding literature. The largest number of iterations in the limited-memory Riemannian BFGS algorithms in the MATLAB toolbox Manopt and ROPTLIB are both modified to be 5000. Note that all the compared OptStiefel, RCG and RBFGS methods only use a single inner iteration in every outer iteration.

In this example, we randomly generate the data matrix A using the MATLAB command $A = \texttt{randn}(m, n) + i\texttt{randn}(m, n)$, and select by $\Theta = \text{diag}\{p, p-1, \dots, 1\}$. We compare our Algorithm 2.1 with the compared OptStiefelGBB, RCG, RLBFGS-Manopt and LRBFGS-ROPTLIB methods for solving the problem (1.2) with different problem size parameters (m, n, p). Our numerical comparison results are reported in Table 3.4, where the terms "CT.", "IT.", "T-INIT.", "Grad." and "Obj." are the same as those in Table 3.1. We see from Table 3.4 that, all the considered algorithms work very efficiently for solving the problem (1.2) for all tested problems, for most cases all the compared methods obtain same objective function values, while the performance of Algorithm 2.1 is slightly better than that of the other algorithms in terms of the number of iterations and the norm of the gradient. In addition, we present in Fig. 3.1 the evolutions of the gradient norm $\|\text{grad}f(V^{(k)}, P^{(k)})\|$ vs iteration to compare further the efficiency of five algorithms. We see that, the gradient-based algorithms converge slowly when the generated iterates are close to an optimal solution, while Algorithm 2.1 outperformed the other competitors in general. We can also see from Table 3.4 that when the problem size is large, Algorithm 2.1 uses much more computing time than that of the compared algorithms in a single iteration. This is because Algorithm 2.1 involves an inner iteration, the tCG method for the trust-region subproblem (2.36), it consumes a lot of time to get an approximate solution to the subproblem, especially when the system dimension is large. Even though Algorithm 2.1 takes the longest time per iteration among the compared algorithms, the convergence is very quick as a whole. In addition, we should point out from Table 3.4 that, the implementation of limited-memory RBFGS in Manopt takes much more computing time than that of the implementation in ROPTLIB, which is consistent with the description given in the comments of the `rlbfgs.m` in Manopt. That is because the default memory size of RBFGS in Manopt is 30 and the default vector transport is by projection. This implies Manopt's implementation needs to compute the vector transport by projection 30 times in every iteration, which can dominate the computational cost. While the implementation of limited-memory RBFGS in ROPTLIB uses the vector transport by parallelization which is much more efficient.

3.4. Numerical comparison of Algorithm 2.1 with infeasible methods for solving manifold optimization

In this subsection, we compare the performance of Algorithm 2.1 with some latest infeasible methods for solving manifold optimization which are all applicable to the problem (1.2), including the splitting of orthogonality constraints (denoted by SOC) method [25], the proximal alternating minimized augmented Lagrangian (denoted by PAMAL) method [6, 13] and the extended proximal alternating linearized minimized augmented Lagrangian (denoted by EPALMAL) method [8, 43]. We first describe briefly how to apply these compared infeasible methods for solving manifold optimization problems to the problem (1.2). For all infeasible methods, we use indices in superscript without round brackets to denote the evolution of variables in the outer iteration.

Table 3.4: Comparison results of Algorithm 2.1 with OptStiefelGBB, RCG and Riemannian version of limited-memory BFGS methods.

	m, n, p	CT.	IT.	T-INIT.	Grad.	Obj.
Algorithm 2.1		0.90	18	445	1.82×10^{-11}	-233.86
OptStifelGBB		0.07	384	384	2.35×10^{-5}	-233.86
RCG	100,50,5	0.10	437	437	1.77×10^{-6}	-233.86
RLBFGS-Manopt		6.03	265	265	8.29×10^{-7}	-233.86
LRBFGS-ROPTLIB		0.12	332	332	7.67×10^{-5}	-233.86
Algorithm 2.1		4.14	20	1464	1.61×10^{-8}	-2691.31
OptStifelGBB		1.79	1874	1874	3.79×10^{-4}	-2691.31
RCG	300, 50, 15	3.49	2258	2258	6.96×10^{-5}	-2691.31
RLBFGS-Manopt		39.17	1121	1121	5.14×10^{-6}	-2691.31
LRBFGS-ROPTLIB		1.33	900	900	6.42×10^{-4}	-2691.31
Algorithm 2.1		4.02	20	1408	1.97×10^{-8}	-3034.01
OptStifelGBB		1.32	1346	1346	4.02×10^{-4}	-3034.01
RCG	300, 100, 15	3.34	1837	1837	2.18×10^{-5}	-3034.01
RLBFGS-Manopt		43.79	1205	1205	6.41×10^{-6}	-3034.01
LRBFGS-ROPTLIB		1.45	880	880	6.03×10^{-4}	-3034.01
Algorithm 2.1		13.77	22	2792	1.90×10^{-7}	-7160.04
OptStifelGBB		9.55	3728	3728	1.51×10^{-3}	-7160.04
RCG	500, 200, 20	30.40	6697	6697	5.75×10^{-5}	-7160.04
RLBFGS-Manopt		133.24	2237	2237	2.10×10^{-5}	-7160.04
LRBFGS-ROPTLIB		7.01	2144	2144	1.30×10^{-3}	-7160.04
Algorithm 2.1		44.24	25	6280	1.30×10^{-8}	-41109.87
OptStifelGBB		53.29	9526	9526	4.37×10^{-3}	-41109.87
RCG	500, 200, 50	210.72	20000	20000	9.25×10^{-2}	-41109.87
RLBFGS-Manopt		343.20	3571	3571	3.70×10^{-5}	-41109.87
LRBFGS-ROPTLIB		18.62	2526	2526	5.12×10^{-3}	-41109.87
Algorithm 2.1		25.64	20	2281	3.37×10^{-8}	-10432.26
OptStifelGBB		23.23	2406	2406	1.65×10^{-3}	-10432.26
RCG	1200, 300, 20	56.54	3435	3435	1.52×10^{-4}	-10432.26
RLBFGS-Manopt		153.13	1952	1952	1.84×10^{-5}	-10432.26
LRBFGS-ROPTLIB		18.53	1510	1510	1.70×10^{-3}	-10432.26
Algorithm 2.1		105.83	28	7006	2.33×10^{-10}	-24183.40
OptStifelGBB		68.10	4868	4868	4.51×10^{-3}	-24183.40
RCG	1200, 400, 30	337.18	13915	13915	1.48×10^{-4}	-24183.40
RLBFGS-Manopt		414.50	3694	3694	5.42×10^{-5}	-24183.40
LRBFGS-ROPTLIB		43.57	2729	2729	3.70×10^{-3}	-24183.40
Algorithm 2.1		118.90	23	6561	9.70×10^{-8}	-44322.81
OptStifelGBB		141.67	7202	7202	7.63×10^{-3}	-44322.81
RCG	1200, 500, 40	240.12	12270	12270	6.02×10^{-2}	-44322.81
RLBFGS-Manopt		559.57	4132	4132	5.35×10^{-5}	-44322.81
LRBFGS-ROPTLIB		109.38	5257	5257	4.02×10^{-3}	-44322.81
Algorithm 2.1		40.41	21	3154	4.95×10^{-7}	-11364.50
OptStifelGBB		66.59	5906	5906	2.87×10^{-3}	-11364.50
RCG	1500, 300, 20	142.72	7333	7333	1.12×10^{-4}	-11364.50
RLBFGS-Manopt		185.12	2123	2123	3.31×10^{-5}	-11364.50
LRBFGS-ROPTLIB		27.20	2035	2035	2.13×10^{-3}	-11364.50
Algorithm 2.1		117.20	24	6869	2.74×10^{-9}	-26041.22
OptStifelGBB		154.80	9604	9604	5.10×10^{-3}	-26041.22
RCG	1500, 400, 30	571.32	20000	20000	2.57×10^{-2}	-26041.20
RLBFGS-Manopt		500.43	4111	4111	4.21×10^{-5}	-26041.22
LRBFGS-ROPTLIB		58.38	3199	3199	3.74×10^{-3}	-26041.22
Algorithm 2.1		105.11	25	5057	5.20×10^{-7}	-47618.65
OptStifelGBB		177.86	8250	8250	8.56×10^{-3}	-47618.65
RCG	1500, 500, 40	682.66	18038	18038	3.37×10^{-4}	-47618.65
RLBFGS-Manopt		614.62	4063	4063	3.95×10^{-5}	-47618.65
LRBFGS-ROPTLIB		68.65	2879	2879	6.52×10^{-3}	-47618.65

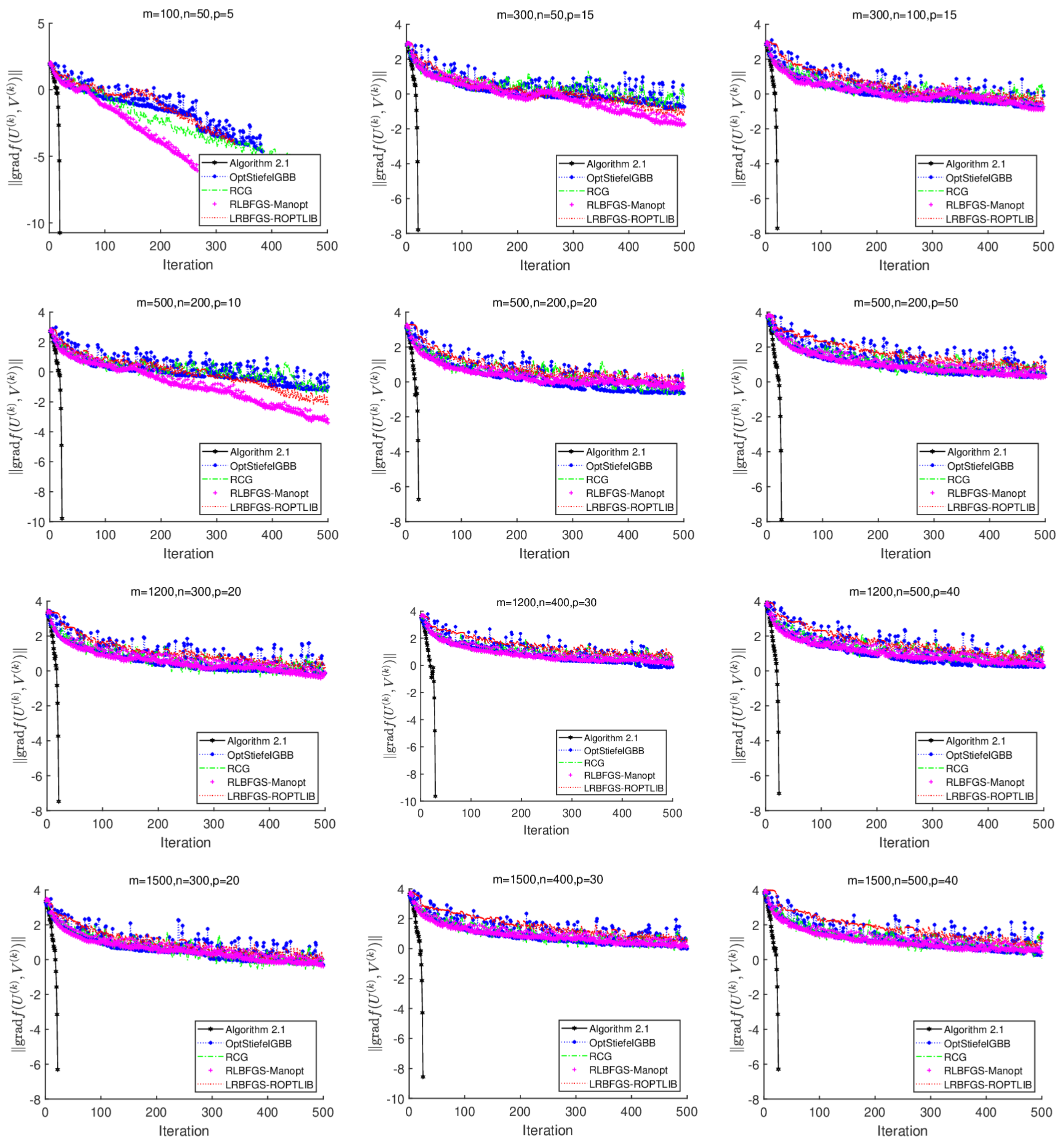

Fig. 3.1. Comparison among the performances of the five algorithms for different problem sizes. The horizontal axis represents the iteration number k and the vertical axis represents the corresponding norm $\log_{10}\|\mathrm{grad} f(V^{(k)}, P^{(k)})\|$.

By introducing auxiliary variables X and Y, the SOC method presented in [25] solves the following reformulation of (1.2):

$$\begin{aligned} &\text{minimize:} && -\operatorname{Tr}(U^H AV\Theta),\\ &\text{subject to:} && U = X, \quad V = Y, \\ & && X \in \mathrm{St}(m,p,\mathbb{C}), \quad Y \in \mathrm{St}(n,p,\mathbb{C}). \end{aligned} \tag{3.5}$$

Using the ideas of alternating direction method of multipliers and the split Bregman method, the SOC method solves (3.5) by alternately updating the variables $(U, V, X, Y, \Lambda_1, \Lambda_2)$

$$\begin{aligned}
&U^k \in \arg\min_U - \operatorname{Tr}(U^H A V^{k-1}\Theta) + \frac{r}{2}\left\|U - X^{k-1} + \Lambda_1^{k-1}\right\|^2,\\
&V^k \in \arg\min_V - \operatorname{Tr}\left({U^k}^H A V\Theta\right) + \frac{r}{2}\left\|V - Y^{k-1} + \Lambda_2^{k-1}\right\|^2,\\
&X^k \in \arg\min_X \frac{r}{2}\left\|X - (U^k + \Lambda_1^{k-1})\right\|^2, \quad \text{subject to} \quad X \in \operatorname{St}(m,p,\mathbb{C}),\\
&Y^k \in \arg\min_Y \frac{r}{2}\left\|Y - (V^k + \Lambda_2^{k-1})\right\|^2, \quad \text{subject to} \quad Y \in \operatorname{St}(n,p,\mathbb{C}),\\
&\Lambda_1^k = \Lambda_1^{k-1} + U^k - X^k,\\
&\Lambda_2^k = \Lambda_2^{k-1} + V^k - Y^k.
\end{aligned}$$

Note that the X-subproblem and Y-subproblem correspond to the projection onto $\operatorname{St}(m,p,\mathbb{C})$ and $\operatorname{St}(n,p,\mathbb{C})$, respectively, whose closed-form solutions can be derived by the singular value decomposition (SVD) [25]. The U-subproblem and V-subproblem are both convex optimization problem without constraints, which can be solved efficiently by deriving the KKT conditions. Specially, we get the following updating of (U^k, V^k, X^k, Y^k) of any fixed $k \in \mathbb{N}$:

$$\begin{aligned}
U^k &= X^{k-1} - \Lambda_1^{k-1} + \frac{1}{r} A V^{k-1}\Theta,\\
V^k &= Y^{k-1} - \Lambda_2^{k-1} + \frac{1}{r} A^H U^k \Theta,\\
X^k &= \arg\max_{X^H X = I_l} \left\langle X, U^k + \Lambda_1^{k-1}\right\rangle = \tilde{P}_1 \tilde{Q}_1^T,\\
Y^k &= \arg\max_{Y^H Y = I_l} \left\langle Y, V^k + \Lambda_2^{k-1}\right\rangle = \tilde{P}_2 \tilde{Q}_2^T,
\end{aligned}$$

where $U^k + \Lambda_1^{k-1} = \tilde{P}_1\tilde{\Sigma}_1\tilde{Q}_1^T$ and $V^k + \Lambda_2^{k-1} = \tilde{P}_2\tilde{\Sigma}_2\tilde{Q}_2^T$ are the reduced SVDs of $U^k + \Lambda_1^{k-1}$ and $V^k + \Lambda_2^{k-1}$, respectively. Note that the SOC method only uses a single inner iteration in every outer iteration.

The PAMAL method presented in [13] can be viewed as a method that hybridizes the augmented Lagrangian method and the proximal alternating minimization (denoted by PAM) techniques [6]. By denoting $\Lambda := (\Lambda_1^T, \Lambda_2^T)^T \in \mathbb{C}^{(n+2l)\times l}$, the classical augmented Lagrangian function associated with (3.5) is given by

$$\begin{aligned}
L(U,V,X,Y;\Lambda,\rho) = &- \operatorname{Tr}(U^H A V\Theta) + \delta_{\operatorname{St}(m,p,\mathbb{C})}(X) + \langle \Lambda_1, U - X\rangle + \frac{\rho}{2}\|U - X\|^2\\
&+ \delta_{\operatorname{St}(n,p,\mathbb{C})}(Y) + \langle \Lambda_2, V - Y\rangle + \frac{\rho}{2}\|V - Y\|^2, \qquad (3.6)
\end{aligned}$$

where ρ is a positive penalty parameter, $\delta_{\operatorname{St}(m,p,\mathbb{C})}$ and $\delta_{\operatorname{St}(n,p,\mathbb{C})}$ are two indicator functions associated with the two complex Stiefel manifolds $\operatorname{St}(m,p,\mathbb{C})$ and $\operatorname{St}(n,p,\mathbb{C})$, respectively. The augmented Lagrangian method for solving (3.6) is then given by

$$\begin{aligned}
&(U^k, V^k, X^k, Y^k) := \arg\min_{U,V,X,Y} L(U,V,X,Y;\bar{\Lambda}^k, \rho^k),\\
&\Lambda_1^{k+1} = \bar{\Lambda}_1^k + \rho^k(U^k - X^k),\\
&\Lambda_2^{k+1} = \bar{\Lambda}_2^k + \rho^k(V^k - Y^k),
\end{aligned} \qquad (3.7)$$

where $\bar{\Lambda}_t^{k+1}$ is the projection of Λ_t^{k+1} on $\{\Lambda_t : \Lambda_{t,\min} \le \Lambda_t \le \Lambda_{t,\max}\}, t = 1,2$. The penalty parameter is updated as

$$\rho^{k+1} := \begin{cases} \rho^k, & \text{if } \|R_i^k\|_\infty \le \tau \|R_i^{k-1}\|_\infty, \quad i = 1,2,\\ \mu\rho^k, & \text{otherwise,} \end{cases}$$

where $R_1^k = U^k - X^k, R_2^k = V^k - Y^k$. Similar to [13], at the k-th outer iteration, the PAM method is applied to solve the subproblem in (3.7) inexactly as follows, which can be viewed as a proximal regularization of a three-block Gauss-Seidel method

$$\begin{aligned}
U^{k,j} &\in \arg\min_U L(U, V^{k,j-1}, X^{k,j-1}, Y^{k,j-1}; \bar{\Lambda}^k, \rho^k) + \frac{c_1^{k,j-1}}{2}\|U - U^{k,j-1}\|^2, \\
V^{k,j} &\in \arg\min_V L(U^{k,j}, V, X^{k,j-1}, Y^{k,j-1}; \bar{\Lambda}^k, \rho^k) + \frac{c_2^{k,j-1}}{2}\|V - V^{k,j-1}\|^2, \\
X^{k,j} &\in \arg\min_X L(U^{k,j}, V^{k,j}, X, Y^{k,j-1}; \bar{\Lambda}^k, \rho^k) + \frac{c_3^{k,j-1}}{2}\|X - X^{k,j-1}\|^2, \\
Y^{k,j} &\in \arg\min_Y L(U^{k,j}, V^{k,j}, X^{k,j}, Y; \bar{\Lambda}^k, \rho^k) + \frac{c_4^{k,j-1}}{2}\|Y - Y^{k,j-1}\|^2,
\end{aligned} \tag{3.8}$$

where the proximal parameters, $c_i^{k,j-1}$, can be arbitrarily chosen as long as they satisfy $0 < \underline{c} \le c_i^{k,j-1} \le \bar{c} < \infty$, $k, j \in \mathbb{N}, i = 1, \ldots, 4$, for some predetermined positive constants $\underline{c}$ and $\bar{c}$. We get the following updating of $(V^{k,j}, P^{k,j}, X^{k,j}, Y^{k,j})$ of any fixed $k \in \mathbb{N}$ by KKT conditions:

$$\begin{aligned}
U^{k,j} &= \frac{1}{\rho^k + c_1^{k,j-1}}\big(\rho^k X^{k,j-1} - \bar{\Lambda}_1^k + c_1^{k,j-1} U^{k,j-1} + AV^{k,j-1}\Theta\big), \\
V^{k,j} &= \frac{1}{\rho^k + c_2^{k,j-1}}\big(\rho^k Y^{k,j-1} - \bar{\Lambda}_2^k + c_2^{k,j-1} V^{k,j-1} + A^H U^{k,j}\Theta\big), \\
X^{k,j} &= \arg\max_{X^H X = I_p}\left\langle X, \frac{1}{\rho^k + c_3^{k,j-1}}\big(\rho^k U^{k,j} + \bar{\Lambda}_1^k + c_3^{k,j-1} X^{k,j-1}\big)\right\rangle = \tilde{P}_3 \tilde{Q}_3^T, \\
Y^{k,j} &= \arg\max_{Y^H Y = I_p}\left\langle Y, \frac{1}{\rho^k + c_4^{k,j-1}}\big(\rho^k V^{k,j} + \bar{\Lambda}_2^k + c_4^{k,j-1} Y^{k,j-1}\big)\right\rangle = \tilde{P}_4 \tilde{Q}_4^T,
\end{aligned}$$

where

$$\begin{aligned}
\tilde{P}_3 \tilde{\Sigma}_3 \tilde{Q}_3^T &= \frac{1}{\rho^k + c_3^{k,j-1}}\big(\rho^k U^{k,j} + \bar{\Lambda}_1^k + c_3^{k,j-1} X^{k,j-1}\big), \\
\tilde{P}_4 \tilde{\Sigma}_4 \tilde{Q}_4^T &= \frac{1}{\rho^k + c_4^{k,j-1}}\big(\rho^k V^{k,j} + \bar{\Lambda}_2^k + c_4^{k,j-1} Y^{k,j-1}\big)
\end{aligned}$$

are the reduced SVDs. The inner iteration (3.8) is terminated when there exists $\Theta^{k,j} \in \partial L(U^{k,j}, V^{k,j}, X^{k,j}, Y^{k,j}; \bar{\Lambda}^k, \rho^k)$ satisfying [13]

$$\|\Theta^{k,j}\|_\infty \le \epsilon^k, \quad {U^{k,j}}^H U^{k,j} = I_p, \quad {V^{k,j}}^H V^{k,j} = I_p,$$

where the term $\Theta^{k,j} := (\Theta_U^{k,j}, \Theta_V^{k,j}, \Theta_X^{k,j}, \Theta_Y^{k,j})$ has the explicit expression as follows:

$$\begin{aligned}
\Theta_U^{k,j} &= -\nabla_U L(U^{k,j}, V^{k,j-1}, X^{k,j-1}, Y^{k,j-1}; \bar{\Lambda}^k, \rho^k) - c_1^{k,j-1}(U^{k,j} - U^{k,j-1}) \\
&\quad + \nabla_U L(U^{k,j}, V^{k,j}, X^{k,j}, Y^{k,j}; \bar{\Lambda}^k, \rho^k) \\
&= A(V^{k,j-1} - V^{k,j})\Theta + \rho^k(X^{k,j-1} - X^{k,j}) + c_1^{k,j-1}(U^{k,j-1} - U^{k,j}), \\
\Theta_V^{k,j} &= -\nabla_V L(U^{k,j}, V^{k,j}, X^{k,j-1}, Y^{k,j-1}; \bar{\Lambda}^k, \rho^k) - c_2^{k,j-1}(V^{k,j} - V^{k,j-1}) \\
&\quad + \nabla_V L(U^{k,j}, V^{k,j}, X^{k,j}, Y^{k,j}; \bar{\Lambda}^k, \rho^k) \\
&= \rho^k(Y^{k,j-1} - Y^{k,j}) + c_2^{k,j-1}(V^{k,j-1} - V^{k,j}),
\end{aligned}$$

$$\Theta_X^{k,j} = c_3^{k,j-1}(X^{k,j-1} - X^{k,j}),$$
$$\Theta_Y^{k,j} = c_4^{k,j-1}(Y^{k,j-1} - Y^{k,j}).$$

Zhu *et al.* [43] studied another algorithm called EPALMAL for nonconvex and nonsmooth optimization with generalized orthogonality, which is also applicable to the problem (1.2). The EPALMAL method is based on the augmented Lagrangian method and the proximal alternating linearized minimization (denoted by PALM) method [8]. 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 [8], while PAMAL uses PAM [6]. By denoting $\Lambda := (\Lambda_1^T, \Lambda_2^T)^T \in \mathbb{C}^{(m+n)\times p}$, as suggested in [43], we consider the scaled form of the augmented Lagrangian function associated with (3.5) as follows:

$$L(U,V,X,Y;\bar{\Lambda},\rho) = \frac{1}{\rho}\delta_{\mathrm{St}(m,p,\mathbb{C})}(X) + \frac{1}{\rho}\delta_{\mathrm{St}(n,p,\mathbb{C})}(Y) + H(U,V,X,Y),$$

where

$$\begin{aligned} H(U,V,X,Y) = {} & -\frac{1}{\rho}\,\mathrm{Tr}(U^H AV\Theta) + \left\langle \frac{\bar{\Lambda}_1}{\rho}, U - X \right\rangle \\ & + \frac{1}{2}\|U - X\|^2 + \left\langle \frac{\bar{\Lambda}_2}{\rho}, V - Y \right\rangle + \frac{1}{2}\|V - Y\|^2. \end{aligned}$$

Similar to [43], at the k-th outer iteration, the PALM method is employed to find an approximate minimizer of the scaled augmented Lagrangian function $L(U,V,X,Y;\bar{\Lambda}^k,\rho^k)$

$$\begin{aligned} U^{k,j} &\in \arg\min_U \langle U - U^{k,j-1}, \nabla_U H_k(U^{k,j-1}, V^{k,j-1}, X^{k,j-1}, Y^{k,j-1})\rangle + \frac{1}{2}\|U - U^{k,j-1}\|^2_{B_1^{k,j-1}}, \\ V^{k,j} &\in \arg\min_V \langle V - V^{k,j-1}, \nabla_V H_k(U^{k,j}, V^{k,j-1}, X^{k,j-1}, Y^{k,j-1})\rangle + \frac{1}{2}\|V - V^{k,j-1}\|^2_{B_2^{k,j-1}}, \\ X^{k,j} &\in \arg\min_X \frac{1}{\rho_k}\delta_{\mathrm{St}(m,p,\mathbb{C})}(X) + H_k(U^{k,j}, V^{k,j}, X, Y^{k,j-1}) + \frac{1}{2}\|X - X^{k,j-1}\|^2_{B_3^k}, \\ Y^{k,j} &\in \arg\min_Y \frac{1}{\rho_k}\delta_{\mathrm{St}(n,p,\mathbb{C})}(Y) + H_k(U^{k,j}, V^{k,j}, X^{k,j}, Y) + \frac{1}{2}\|Y - Y^{k,j-1}\|^2_{B_4^k}, \end{aligned} \tag{3.9}$$

where

$$\begin{cases} B_1^{k,j-1} = \gamma_1 L_1^{k,j-1} I_m, & \gamma_1 > 1, \\ B_2^{k,j-1} = \gamma_2 L_2^{k,j-1} I_n, & \gamma_2 > 1, \\ B_3^k = \gamma_3 I_m, & \gamma_3 > 1, \\ B_4^k = \gamma_4 I_n, & \gamma_4 > 1. \end{cases}$$

Here, $L_1^{k,j-1}$ and $L_2^{k,j-1}$ are the two global Lipschitz constants of $\nabla_U H_k(U, V^{k,j-1}, X^{k,j-1}, Y^{k,j-1})$ and $\nabla_V H_k(U^{k,j}, V, X^{k,j-1}, Y^{k,j-1})$, respectively. By simple calculation we have $L_1^{k,j-1} = L_2^{k,j-1} = 1$. For the subproblems regarding X and Y, we use the PAM method rather than the PALM method since $H_k(U^{k,j}, V^{k,j}, X, Y^{k,j-1})$ and $H_k(U^{k,j}, V^{k,j}, X^{k,j}, Y)$ are both quadratic on X and Y. The updating schemes of $(U^{k,j}, V^{k,j}, X^{k,j}, Y^{k,j})$ can be get by following:

$$\begin{aligned} U^{k,j} &= U^{k,j-1} - \frac{1}{\gamma_1 L_1^{k,j-1}}\nabla_U H_k(U^{k,j-1}, V^{k,j-1}, X^{k,j-1}, Y^{k,j-1}) \\ &= U^{k,j-1} - \frac{1}{\gamma_1}\left(\frac{1}{\rho^k}\bar{\Lambda}_1^k + U^{k,j-1} - X^{k,j-1} - \frac{1}{\rho^k}AV^{k,j-1}\Theta\right), \end{aligned}$$

$$\begin{aligned}
V^{k,j} &= V^{k,j-1} - \frac{1}{\gamma_2 L_2^{k,j-1}} \nabla_V H_k(U^{k,j}, V^{k,j-1}, X^{k,j-1}, Y^{k,j-1}) \\
&= V^{k,j-1} - \frac{1}{\gamma_2}\left(\frac{1}{\rho^k}\bar{\Lambda}_2^k + V^{k,j-1} - Y^{k,j-1} - \frac{1}{\rho^k}A^H U^{k,j}\Theta\right), \\
X^{k,j} &= \underset{X^H X = I_p}{\arg\max} \left\langle X, \frac{1}{1+\gamma_3}\left(U^{k,j} + \frac{1}{\rho^k}\bar{\Lambda}_1^k + \gamma_3 X^{k,j-1}\right)\right\rangle = \tilde{P}_5 \tilde{Q}_5^T, \\
Y^{k,j} &= \underset{Y^H Y = I_p}{\arg\max} \left\langle Y, \frac{1}{1+\gamma_4}\left(V^{k,j} + \frac{1}{\rho^k}\bar{\Lambda}_2^k + \gamma_4 Y^{k,j-1}\right)\right\rangle = \tilde{P}_6 \tilde{Q}_6^T,
\end{aligned}$$

where

$$\begin{aligned}
\tilde{P}_5 \tilde{\Sigma}_5 \tilde{Q}_5^T &= \frac{1}{1+\gamma_3}\left(U^{k,j} + \frac{\bar{\Lambda}_1^k}{\rho^k} + \gamma_3 X^{k,j-1}\right), \\
\tilde{P}_6 \tilde{\Sigma}_6 \tilde{Q}_6^T &= \frac{1}{1+\gamma_4}\left(V^{k,j} + \frac{\bar{\Lambda}_2^k}{\rho^k} + \gamma_4 Y^{k,j-1}\right)
\end{aligned}$$

are the reduced SVDs. The inner iteration (3.9) is terminated when there exists $\Theta^{k,j} \in \partial L(U^{k,j}, V^{k,j}, X^{k,j}, Y^{k,j}; \bar{\Lambda}^k, \rho^k)$ satisfying [43]

$$\|\Theta^{k,j}\|_\infty \le \frac{\epsilon^k}{\rho^k}, \quad {U^{k,j}}^H U^{k,j} = I_p, \quad {V^{k,j}}^H V^{k,j} = I_p,$$

where $\Theta^{k,j} = (\Theta_U^{k,j}, \Theta_V^{k,j}, \Theta_X^{k,j}, \Theta_Y^{k,j})$ is given by

$$\begin{aligned}
\Theta_U^{k,j} &= -\nabla_U H_k(U^{k,j-1}, V^{k,j-1}, X^{k,j-1}, Y^{k,j-1}) - B_1^{k,j-1}(U^{k,j} - U^{k,j-1}) \\
&\quad + \nabla_U H_k(U^{k,j}, V^{k,j}, X^{k,j}, Y^{k,j}) \\
&= \frac{1}{\rho^k} A (V^{k,j-1} - V^{k,j})\Theta + (\gamma_1 - 1)(U^{k,j-1} - U^{k,j}) + (X^{k,j-1} - X^{k,j}), \\
\Theta_V^{k,j} &= -\nabla_V H_k(U^{k,j}, V^{k,j-1}, X^{k,j-1}, Y^{k,j-1}) - B_2^{k,j-1}(V^{k,j} - V^{k,j-1}) \\
&\quad + \nabla_V H_k(U^{k,j}, V^{k,j}, X^{k,j}, Y^{k,j}) \\
&= (\gamma_2 - 1)(V^{k,j-1} - V^{k,j}) + (Y^{k,j-1} - Y^{k,j}), \\
\Theta_X^{k,j} &= B_3^k (X^{k,j-1} - X^{k,j}) = \gamma_3 (X^{k,j-1} - X^{k,j}), \\
\Theta_Y^{k,j} &= B_4^k (Y^{k,j-1} - Y^{k,j}) = \gamma_4 (Y^{k,j-1} - Y^{k,j}).
\end{aligned}$$

In our experiments, the parameters of the three infeasible methods are set as follows: in the SOC method, we choose $\Lambda_1^0 = 0_{n\times l}, \Lambda_2^0 = 0_{2l\times l}$ and the penalty parameter $r = 300$ as recommended in [25]. Our numerical experiments show that the convergence rate of the SOC method is not very sensitive to this parameter as long as it is greater than a certain value. In the PAMAL method, parameters are set to be the same as in [13, p.B587], i.e., $\epsilon^k = (0.999)^k, k \in \mathbb{N}, \bar{\Lambda}_1^1 = 0_{n\times l}, \bar{\Lambda}_2^1 = 0_{2l\times l}, \bar{\Lambda}_{t,\min} = -100, \bar{\Lambda}_{t,\max} = 100, t = 1, 2, \tau = 0.99, \mu = 1.01$ and $\underline{c} = c_i^{k,j} = \bar{c} = 0.5$ for all k, j, i. In the EPALMAL method, as recommended in [43], $\gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 = 1.01$, while parameters $\epsilon^k, \bar{\Lambda}_1^1, \bar{\Lambda}_2^1, \bar{\Lambda}_{p,\min}, \bar{\Lambda}_{p,\max}, \tau$ and μ are set to be the same as in the PAMAL method. In particular, in both the PAMAL and EPALMAL methods, we set $\rho^1 = 50n$ empirically, which works quite well in most cases for the tested problems with different problem sizes. In addition, the relative errors of the objective function does not depend on the values of ρ^1 once the two methods converge, while the iteration numbers of the two methods do depend on the parameter ρ^1. For all three methods, we use the same random orthogonal initial points as those in Algorithm 2.1 and terminate the outer iteration when

$$\frac{|f(U^k,V^k)-f(U^{k-1},V^{k-1})|}{\max\left\{1,|f(U^{k-1},V^{k-1})|\right\}} < 10^{-5} \quad \text{and} \quad \max\left\{\|U^k-X^k\|,\|V^k-Y^k\|\right\} < 10^{-2}.$$

In this example, the data matrices A and Θ in (1.2) are generated as same as in that in the last example. The numerical comparison results of Algorithm 2.1 and the three compared infeasible methods are reported in Table 3.5, where the terms "CT.", "IT.", "T-INIT.", "Grad." and "Obj." are the same as those in Table 3.4, the terms "Feasi$_U$" and "Feasi$_V$" mean respectively the feasibility $\|{U^k}^H U^k - I_p\|$ and $\|{V^k}^H V^k - I_p\|$ at the final iterate by implementing the proposed algorithms. We also plot the evolutions of the objective function value $f(U^{(k)},V^{(k)})$, vs. iterations for different problem sizes in Fig. 3.2. Based on the results reported in Table 3.5 and Fig. 3.2 and many other performed unreported tests which show similar patterns, we see that our proposed Algorithm 2.1 outperformed all the compared infeasible methods in most cases in terms of the computational time, the total numbers of iteration and the quality of the solutions. In addition, it can be seen that for most cases all the compared methods obtain similar objective function values. Notice that in many cases EPALMAL needs more inner iterations than PAMAL, especially with increase of k. This is mainly because EPALMAL uses linearization in solving inner subproblems which usually needs more inner iterations to reach certain accuracy.

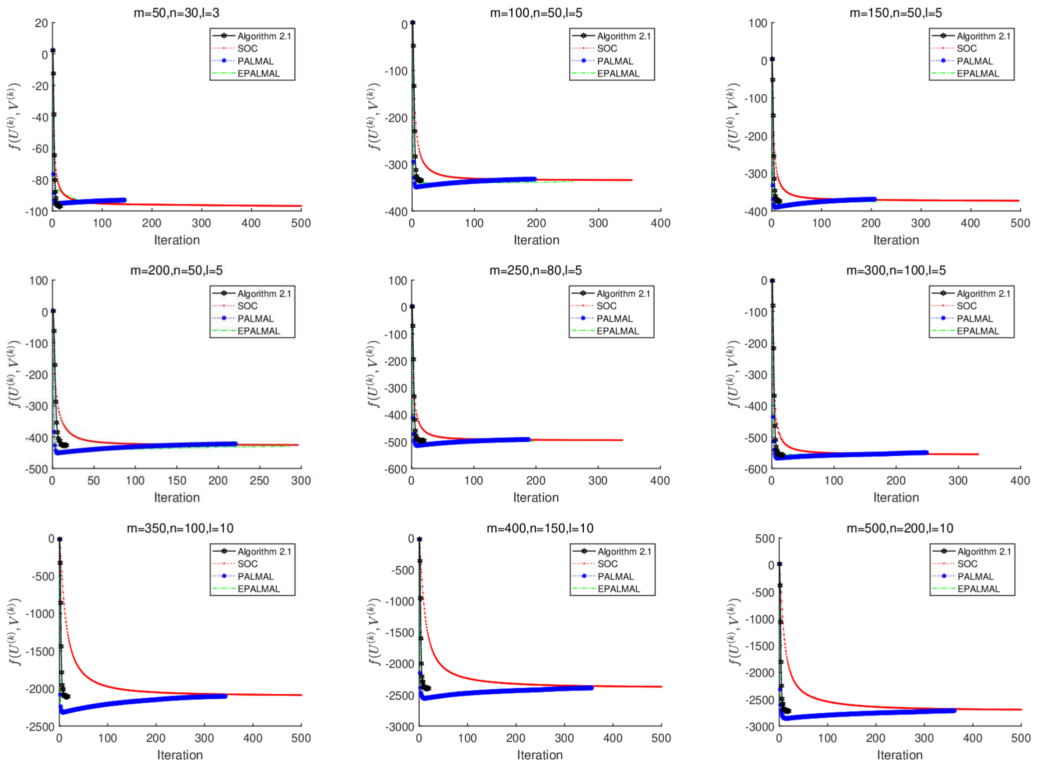

Fig. 3.2. Comparison among the performances of the four algorithms for different problem sizes. The horizontal axis represents the iteration number k and the vertical axis represents the corresponding function value $f(V^{(k)},P^{(k)})$.

Table 3.5: Comparison results of Algorithm 2.1 with EPALMAL, PALMAL and SOC methods.

	m, n, p	CT.	IT.	T-INIT.	Grad.	Obj.	Feasi$_{\mathrm{U}}$	Feasi$_{\mathrm{V}}$
Algorithm 2.1	50 30 3	0.375	15	204	2.79×10^{-7}	-96.929	5.56×10^{-16}	7.47×10^{-16}
SOC		0.141	538	538	3.22×10^{-1}	-96.828	1.75×10^{-4}	1.75×10^{-4}
PALMAL		0.084	145	549	4.40×10^{0}	-93.051	1.15×10^{-2}	1.15×10^{-2}
EPALMAL		0.439	172	2094	2.47×10^{0}	-95.852	1.14×10^{-2}	1.15×10^{-2}
Algorithm 2.1	100 50 5	1.266	14	350	1.24×10^{-8}	-334.583	1.24×10^{-15}	8.45×10^{-16}
SOC		0.658	353	353	1.02×10^{0}	-333.911	7.13×10^{-5}	7.11×10^{-5}
PALMAL		0.485	197	660	6.95×10^{0}	-331.712	8.95×10^{-3}	8.95×10^{-3}
EPALMAL		6.361	259	8768	3.00×10^{0}	-337.086	8.93×10^{-3}	8.93×10^{-3}
Algorithm 2.1	150 50 5	1.461	15	472	2.34×10^{-8}	-373.689	8.43×10^{-16}	1.12×10^{-15}
SOC		0.458	496	496	1.09×10^{0}	-372.667	1.04×10^{-4}	1.04×10^{-4}
PALMAL		0.537	207	684	7.20×10^{0}	-368.687	8.96×10^{-3}	8.96×10^{-3}
EPALMAL		6.411	220	7199	3.59×10^{0}	-374.841	8.94×10^{-3}	8.95×10^{-3}
Algorithm 2.1	200 50 5	1.323	16	407	3.61×10^{-9}	-425.972	7.46×10^{-16}	7.77×10^{-16}
SOC		0.352	296	296	1.16×10^{0}	-424.884	1.34×10^{-4}	1.34×10^{-4}
PALMAL		0.572	221	634	8.66×10^{0}	-421.537	8.94×10^{-3}	8.94×10^{-3}
EPALMAL		11.373	286	11240	3.63×10^{0}	-429.298	8.93×10^{-3}	8.93×10^{-3}
Algorithm 2.1	250 80 5	2.575	19	734	4.66×10^{-11}	-496.580	7.92×10^{-16}	5.91×10^{-16}
SOC		0.586	339	339	1.26×10^{0}	-494.739	1.37×10^{-4}	1.37×10^{-4}
PALMAL		1.039	188	998	8.47×10^{0}	-492.137	8.92×10^{-3}	8.92×10^{-3}
EPALMAL		10.210	197	6588	4.77×10^{0}	-498.010	8.89×10^{-3}	8.89×10^{-3}
Algorithm 2.1	300 100 5	2.171	17	574	1.02×10^{-11}	-556.929	5.35×10^{-16}	6.29×10^{-16}
SOC		0.704	332	332	1.33×10^{0}	-554.737	1.75×10^{-4}	1.75×10^{-4}
PALMAL		1.546	249	1189	1.03×10^{1}	-549.478	8.89×10^{-3}	8.89×10^{-3}
EPALMAL		12.975	191	6260	5.45×10^{0}	-557.435	8.95×10^{-3}	8.95×10^{-3}
Algorithm 2.1	350 100 10	4.312	17	797	1.81×10^{-7}	-2107.214	1.25×10^{-15}	1.37×10^{-15}
SOC		2.585	691	691	8.00×10^{0}	-2093.519	1.77×10^{-5}	1.77×10^{-5}
PALMAL		3.819	343	1508	1.44×10^{1}	-2103.491	6.31×10^{-3}	6.31×10^{-3}
EPALMAL		51.657	297	15879	9.44×10^{0}	-2117.996	6.31×10^{-3}	6.31×10^{-3}
Algorithm 2.1	400 150 10	6.195	19	954	4.28×10^{-8}	-2397.133	7.09×10^{-16}	1.17×10^{-15}
SOC		3.869	721	721	8.55×10^{0}	-2380.523	2.21×10^{-5}	2.21×10^{-5}
PALMAL		6.661	356	2079	1.61×10^{1}	-2390.650	6.33×10^{-3}	6.33×10^{-3}
EPALMAL		65.248	279	15391	1.09×10^{1}	-2407.595	6.31×10^{-3}	6.31×10^{-3}
Algorithm 2.1	500 200 10	15.279	19	1553	8.18×10^{-9}	-2717.906	1.71×10^{-15}	8.72×10^{-16}
SOC		7.756	695	695	9.12×10^{0}	-2701.082	3.08×10^{-5}	3.08×10^{-5}
PALMAL		11.479	362	2851	1.70×10^{1}	-2713.819	6.29×10^{-3}	6.29×10^{-3}
EPALMAL		130.314	254	14076	1.26×10^{1}	-2729.258	6.29×10^{-3}	6.29×10^{-3}

4. Conclusion

In this paper, we have developed a Riemannian trust-region method to solve the truncated complex singular value decomposition problem, which can be transformed to a Riemannian optimization problem on a complex product Stiefel manifold. Under our proposed framework, a new trust-region subproblem is constructed based on our expressions of the Hessian of the objection function, a truncated conjugate gradient method has been proposed to solve the new trust-region subproblem. Moreover, the developed algorithm converges globally with a fast rate, which is important for data matrices with large sizes. In the numerical experiments part, numerical comparisons with some other methods are provided to illustrate the performance of the proposed algorithm, including some classical gradient-based algorithms, the Riemannian version of limited-memory BFGS methods and some latest infeasible methods for solving manifold optimization.

Acknowledgements. The authors would like to thank both reviewers for their very constructive suggestions that lead to the improvement of this manuscript.

Research supported by the National Natural Science Foundation of China (Grant Nos. 12261026, 11961012, 12201149), by the Natural Science Foundation of Guangxi Province (Grant Nos. 2016GXNSFAA380074, 2023GXNSFAA026067), by the Innovation Project of GUET Graduate Education (Grant No. 2022YXW01), by the GUET Graduate Innovation Project (Grant No. 2022YCXS142), by the Guangxi Key Laboratory of Automatic Detecting Technology and Instruments (Grant Nos. YQ23103, YQ21103, YQ22106), by the Special Fund for Science and Technological Bases and Talents of Guangxi (Grant No. 2021AC06001) and by the Guizhou Science and Technology Program of Projects (Grant No. ZK2021G339).

References

[1] P.A. Absil, C.G. Baker, and K.A. Gallivan, Trust-region methods on Riemannian manifolds with applications in numerical linear algebra, in: *Proceedings of the 16th International Symposium on Mathematical Theory of Networks and Systems*, Belgium, (2004).

[2] P.A. Absil, C.G. Baker, and K.A. Gallivan, Trust-region methods on Riemannian manifolds, *Found. Comput. Math.*, **7**:3 (2007), 303–330.

[3] P.A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*, Princeton University Press, 2009.

[4] P.A. Absil, R. Mahony, and J. Trumpf, An extrinsic look at the Riemannian Hessian, in: *International Conference on Geometric Science of Information*, 361–368, Springer, (2013).

[5] K. Aihara and H. Sato, A matrix-free implementation of Riemannian Newton's method on the Stiefel manifold, *Optim. Lett.*, **11**:8 (2017), 1729–1741.

[6] 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, *Math. Oper. Res.*, **35**:2 (2010), 438–457.

[7] C.G. Baker, P.A. Absil, and K.A. Gallivan, An implicit trust-region method on Riemannian manifolds, *IMA J. Numer. Anal.*, **28**:4 (2008), 665–689.

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

[9] N. Boumal and P.A. Absil, RTRMC: A Riemannian trust-region method for low-rank matrix completion, in: *Advances in Neural Information Processing Systems*, (2011), 406–414.

[10] N. Boumal, B. Mishra, P.A. Absil, and R. Sepulchre, Manopt, a Matlab toolbox for optimization on manifolds, *J. Mach. Learn. Res.*, **15**:1 (2014), 1455–1459.

[11] M. Brand, Fast low-rank modifications of the thin singular value decomposition, *Linear Algebra Appl.*, **415**:1 (2006), 20–30.

[12] T.F. Chan and P.C. Hansen, Computing truncated singular value decomposition least squares solutions by rank revealing QR-factorizations, *SIAM J. Sci. Comput.*, **11**:3 (1990), 519–530.

[13] W. Chen, Y. You, and H. Ji, An augented Lagrangian method for l_1-regularized optimization problems with orthogonal constraints, *SIAM J. Sci. Stat. Comput.*, **38**:4 (2016), B570–B592.

[14] Y. Chu, K. Xu, Y. Zhong, X. Ye, T. Zhou, X. Chen, and G. Wang, Fast microwave through wall imaging method with inhomogeneous background based on Levenberg-Marquardt algorithm, *IEEE Trans. Microw. Theory Tech.*, **67**:3 (2018), 1138–1147.

[15] T.A. Davis and Y. Hu, The University of Florida sparse matrix collection, *ACM Trans. Math. Softw.*, **38**:1 (2011), 1–25.

[16] S. Doğu, M.N. Akıncı, M. Çayören, and İ. Akduman, Truncated singular value decomposition for through-the-wall microwave imaging application, *IET Microw. Antennas Propag.*, **14**:4 (2019), 260–267.

[17] A. Edelman, T.A. Arias, and S.T. Smith, The geometry of algorithms with orthogonality constraints, *SIAM J. Matrix Anal. Appl.*, **20**:2 (1998), 303–353.

[18] G. Gennarelli and F. Soldovieri, A linear inverse scattering algorithm for radar imaging in multipath environments, *IEEE Geosci. Remote. Sens. Lett.*, **10**:5 (2013), 1085–1089.

[19] P.C. Hansen, Truncated singular value decomposition solutions to discrete ill-posed problems with ill-determined numerical rank, *SIAM J. Sci. Stat. Comput.*, **11**:3 (1990), 503–518.

[20] G. Heidel and V. Schulz, A Riemannian trust-region method for low-rank tensor completion, *Numer. Linear Algebra Appl.*, **25**:6 (2018), e2175.

[21] W. Huang, P.A. Absil, K.A. Gallivan, and P. Hand, ROPTLIB: An object-oriented C++ library for optimization on Riemannian manifolds, *ACM Trans. Math. Softw.*, **44**:4 (2018), 43.1–43.21.

[22] W. Huang, K.A. Gallivan, and P.A. Absil, A broyden class of quasi-Newton methods for Riemannian optimization, *SIAM J. Optim.*, **25**:3 (2015), 1660–1685.

[23] W. Huang, K.A. Gallivan, P.A. Absil, and P. Hand, ROPTLIB: Riemannian Manifold Optimization Library (User Manual), 2020.

[24] M. Ishteva, P.A. Absil, S.V. Huffel, and L.D. Lathauwer, Best low multilinear rank approximation of higher-order tensors, based on the Riemannian trust-region scheme, *SIAM J. Matrix Anal. Appl.*, **32**:1 (2011), 115–135.

[25] R. Lai and S. Osher, A splitting method for orthogonality constrained problems, *J. Sci. Comput.*, **58**:2 (2014), 431–449.

[26] J.F. Li, W. Li, S.W. Vong, Q.L. Luo, and M. Xiao, A Riemannian optimization approach for solving the generalized eigenvalue problem for nonsquare matrix pencils, *J. Sci. Comput.*, **82**:3 (2020), 1–43.

[27] H. Sato, Riemannian conjugate gradient method for complex singular value decomposition problem, in: *53rd IEEE Conference on Decision and Control*, IEEE, (2014), 5849–5854.

[28] H. Sato, Joint singular value decomposition algorithm based on the Riemannian trust-region method, *JSIAM Lett.*, **7** (2015), 13–16.

[29] H. Sato, *Riemannian Optimization and Its Applications*, Springer Cham, 2021.

[30] H. Sato and T. Iwai, A Riemannian optimization approach to the matrix singular value decomposition, *SIAM J. Optim.*, **23**:1 (2013), 188–212.

[31] H. Sato, and T. Iwai, A complex singular value decomposition algorithm based on the Riemannian Newton method, in: *52nd IEEE Conference on Decision and Control*, 2972–2978. IEEE, (2013).

[32] K. Sato and H. Sato, Structure-preserving H^2 optimal model reduction based on the Riemannian trust-region method, *IEEE Trans. Automat. Contr.*, **63**:2 (2018), 505–512.

[33] R. Scapaticci, O. Bucci, I. Catapano, and L. Crocco, Differential microwave imaging for brain stroke followup, *Int. J. Antennas Propag.*, **2014** (2014), 1–11.

[34] J.D. Shea, B.D.V. Veen, and S.C. Hagness, A TSVD analysis of microwave inverse scattering for breast imaging, *IEEE Trans. Biomed. Eng.*, **59**:4 (2011), 936–945.

[35] F. Soldovieri, R. Solimene, and R. Pierri, A simple strategy to detect changes in through the wall imaging, *Prog. Electromagn. Res. M*, **7**:3 (2009), 1–13.

[36] R. Solimene, F. Ahmad, and F. Soldovieri, A novel CS-TSVD strategy to perform data reduction

in linear inverse scattering problems, *IEEE Geosci. Remote Sens. Lett.*, **9**:5 (2012), 881–885.

[37] R. Solimene, F. Soldovieri, A. Baratonia, and R. Pierri, Experimental validation of a linear inverse scattering TWI algorithm by a SF-CW radar, *IEEE Antennas Wirel. Propag. Lett.*, **9** (2010), 506–509.

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

[39] P. Yang, Y.L. Jiang, and K.L. Xu, A trust-region method for H_2 model reduction of bilinear systems on the Stiefel manifold, *J. Franklin Inst.*, **356**:4 (2019), 2258–2273.

[40] J. Yu, F. Liu, L. Jiao, and X. He, Research on the reconstruction for bioluminescence tomography using truncated singular value decomposition method, *J. Northwest Univ. Nat. Sci.*, **39**:5 (2009), 755–760.

[41] L.H. Zhang, Riemannian trust-region method for the maximal correlation problem, *Numer. Funct. Anal. Optim.*, **33**:3 (2012), 338–362.

[42] X. Zhu, A Riemannian conjugate gradient method for optimization on the Stiefel manifold, *Comput. Optim. Appl.*, **67**:1 (2017), 73–110.

[43] H. Zhu, X. Zhang, D. Chu, and L.Z. Liao, Nonconvex and nonsmooth optimization with generalized orthogonality constraints: An approximate augmented Lagrangian method, *J. Sci. Comput.*, **72**:1 (2017), 331–372.