Convergence of gradient-based block coordinate descent algorithms for non-orthogonal joint approximate diagonalization of matrices

Jianze Li, Konstantin Usevich, Pierre Comon

CONVERGENCE OF GRADIENT-BASED BLOCK COORDINATE DESCENT ALGORITHMS FOR NON-ORTHOGONAL JOINT APPROXIMATE DIAGONALIZATION OF MATRICES*

JIANZE LI†, KONSTANTIN USEVICH‡, AND PIERRE COMON§

Abstract. In this paper, we propose a gradient-based block coordinate descent (BCD-G) framework to solve the joint approximate diagonalization of matrices defined on the product of the complex Stiefel manifold and the special linear group. Instead of the cyclic fashion, we choose a block optimization based on the Riemannian gradient. To update the first block variable in the complex Stiefel manifold, we use the well-known line search descent method. To update the second block variable in the special linear group, based on four kinds of different elementary transformations, we construct three classes: GLU, GQU and GU, and then get three BCD-G algorithms: BCD-GLU, BCD-GQU and BCD-GU. We establish the global and weak convergence of these three algorithms using the Łojasiewicz gradient inequality under the assumption that the iterates are bounded. We also propose a gradient-based Jacobi-type framework to solve the joint approximate diagonalization of matrices defined on the special linear group. As in the BCD-G case, using the GLU and GQU classes of elementary transformations, we focus on the Jacobi-GLU and Jacobi-GQU algorithms and establish their global and weak convergence. All the algorithms and convergence results described in this paper also apply to the real case.

Key words. blind source separation, joint approximate diagonalization of matrices, block coordinate descent, Jacobi-G algorithm, convergence analysis, manifold optimization

AMS subject classifications. 49M30, 65F99, 90C30, 15A23

1. Introduction.

Let $1 \leq m \leq n$. Given a complex matrix $\boldsymbol{Z} \in \mathbb{C}^{n\times m}$, we denote by $\boldsymbol{Z}^{\mathsf{T}}$, $\boldsymbol{Z}^*$ and $\boldsymbol{Z}^{\mathsf{H}}$ its *transpose, conjugate* and *conjugate transpose,* respectively. We shall also use $(\cdot)^{\blacklozenge}$ to denote either $(\cdot)^{\mathsf{T}}$ or $(\cdot)^{\mathsf{H}}$. A complex matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ is called *Hermitian* if $\boldsymbol{A}^{\mathsf{H}} = \boldsymbol{A}$. It is called *complex symmetric* if $\boldsymbol{A}^{\mathsf{T}} = \boldsymbol{A}$. Let $\{\boldsymbol{A}^{(\ell)}\}_{1\leq\ell\leq \mathrm{L}} \subseteq \mathbb{C}^{n\times n}$ be a set of complex matrices. The well-known *blind source separation* (BSS) problem [17, 18, 36, 43] can be formulated as finding a full column rank matrix $\boldsymbol{Z} \in \mathbb{C}^{n\times m}$ to make the matrices $\boldsymbol{W}^{(\ell)} = \boldsymbol{Z}^{\blacklozenge}\boldsymbol{A}^{(\ell)}\boldsymbol{Z} \in \mathbb{C}^{m\times m}$ simultaneously as diagonal as possible. A natural idea is to solve the *joint approximate diagonalization of matrices* (JADM) problem, which consists in minimizing

$$f(\boldsymbol{Z}) = \sum_{\ell=1}^{\mathrm{L}} \|\operatorname{offdiag}\{\boldsymbol{W}^{(\ell)}\}\|^2, \tag{1.1}$$

where $\boldsymbol{Z} \in \mathbb{C}^{n\times m}$ is a full column rank matrix, and $\operatorname{offdiag}\{\cdot\}$ is the *zero diagonal* operator, setting all the diagonal elements of a square matrix in $\mathbb{C}^{m\times m}$ to zero.

Note that, the set of full-column rank matrices is not closed (the limit of a sequence of full column rank matrices can be rank deficient), and therefore problem (1.1) is ill-posed. For example, for a full column rank matrix $\boldsymbol{Z} \in \mathbb{C}^{n\times m}$ and nonzero $\lambda \in \mathbb{C}$, we have $\lim_{\lambda\to 0} f(\lambda\boldsymbol{Z}) = \lim_{\lambda\to 0} |\lambda|^4 f(\boldsymbol{Z}) = 0$. To tackle this issue, it is first necessary to

*Submitted to the editors on Nov. 3, 2021; revised July 1st, 2022; revised Nov. 25, 2022.

Funding: This work was supported in part by the National Natural Science Foundation of China (No. 11601371), the Guangdong Basic and Applied Basic Research Foundation (No. 2021A1515010232), and Agence Nationale de Recherche (ANR-19-CE23-0021).

†Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen, China (lijianze@gmail.com).

‡Université de Lorraine, CNRS, CRAN, Nancy, France (konstantin.usevich@cnrs.fr).

§Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-Lab, France (pierre.comon@gipsa-lab.fr).

use scale- and permutation-invariant cost functions [4, 49]. Second, the set of matrices $\boldsymbol{Z}$ must be restricted to a smaller *closed* subset. Several possibilities can be envisaged, *e.g.*, a restriction to the special linear group $\mathbf{SL}_m(\mathbb{C})$ in the square case $m = n$. In this paper, we follow the latter approach as it will be discussed later.

Problem (1.1) has been widely used in BSS and *Independent component analysis* (ICA) [14, 18, 6, 7], and has the following well-known special cases:

- *joint approximate diagonalization of Hermitian matrices* (JADM-H) [42, 36]: $(\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{H}}$, $\boldsymbol{A}^{(\ell)} \in \mathbb{C}^{n\times n}$ is Hermitian for $1 \leq \ell \leq \mathrm{L}$;
- *joint approximate diagonalization of complex symmetric matrices* (JADM-CS) [36]: $(\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{T}}$, $\boldsymbol{A}^{(\ell)} \in \mathbb{C}^{n\times n}$ is complex symmetric for $1 \leq \ell \leq \mathrm{L}$;
- *joint approximate diagonalization of real symmetric matrices* (JADM-RS) [4, 5]: over real $\boldsymbol{Z}$, $(\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{T}}$, $\boldsymbol{A}^{(\ell)} \in \mathbb{R}^{n\times n}$ is real symmetric for $1 \leq \ell \leq \mathrm{L}$.

Many classic approaches use prewhitening to reduce the problem (1.1) to orthogonal (and square) diagonalization case [12, 13, 17, 25, 26, 27, 28, 45]. This, however, results in a two-step procedure, which may not be optimal in the statistical sense and may suffer more from noise. Therefore, the non-orthogonal joint diagonalization attracted considerable interest in the literature. In particular, to solve the JADM-RS problem, Jacobi-type algorithms were introduced based on the LU and QR decompositions in [5], and on the Givens transformations, hyperbolic transformations, and diagonal transformations in [43, Eq. (9)]. To solve the JADM-H problem, Jacobi-type algorithms were proposed based on the LU decomposition in [36, 37], and based on the QL decomposition in [42]. To solve the JADM-CS problem, a Jacobi-type algorithm was proposed based on the LU decomposition in [35, 36]. However, to our knowledge, there was no theoretical result about the convergence of these Jacobi-type algorithms in the literature. In addition, mostly the square ($m = n$) case was considered.

In this paper, we consider the general rectangular case of (1.1), with $\boldsymbol{Z}$ restricted to a $\mathbf{SL}_m(\mathbb{C})$-like subset. By using a reformulation of the problem, we develop optimization algorithms on manifolds, and provide convergence results. An overview of the contributions is provided in the rest of the section.

1.1. Search space and reformulations of the problem. Let $\mathbf{GL}_m(\mathbb{C}) \stackrel{\mathsf{def}}{=} \{\boldsymbol{X} \in \mathbb{C}^{m\times m}, \det(\boldsymbol{X}) \neq 0\}$ (resp. $\mathbf{SL}_m(\mathbb{C}) \stackrel{\mathsf{def}}{=} \{\boldsymbol{X} \in \mathbf{GL}_m(\mathbb{C}), \det(\boldsymbol{X}) = 1\}$) be the *general* (resp. *special*) *linear group*. We define the *rectangular special linear set* as

$$\mathbf{RSL}(m, n, \mathbb{C}) \stackrel{\mathsf{def}}{=} \{\boldsymbol{Z} \in \mathbb{C}^{n\times m}, \boldsymbol{Z}^{\mathsf{H}}\boldsymbol{Z} \in \mathbf{SL}_m(\mathbb{C})\}. \tag{1.2}$$

Every matrix in $\mathbf{RSL}(m, n, \mathbb{C})$ is of full column rank, and, moreover this set is closed. Thus the problem of rank deficiency or trivial solution at $\mathbf{0}$ does not appear when optimizing (1.1) over $\mathbf{RSL}(m, n, \mathbb{C})$, since $\lambda\boldsymbol{Z} \notin \mathbf{RSL}(m, n, \mathbb{C})$ if $|\lambda| \neq 1$. Still, this remains a difficult optimization problem, since the feasible region $\mathbf{RSL}(m, n, \mathbb{C})$ is neither convex nor compact, and the function $f(\boldsymbol{Z})$ is a quartic polynomial. In what follows, we provide a reformulation of the problem for two scenarios.

- *General (rectangular) case.* Let $\mathbf{St}(m, n, \mathbb{C}) \stackrel{\mathsf{def}}{=} \{\boldsymbol{Y} \in \mathbb{C}^{n\times m}, \boldsymbol{Y}^{\mathsf{H}}\boldsymbol{Y} = \boldsymbol{I}_m\}$ be the *complex Stiefel manifold*. We have the following simple result:

  LEMMA 1.1. *A complex matrix $\boldsymbol{Z} \in \boldsymbol{RSL}(m, n, \mathbb{C})$ if and only if there exist $\boldsymbol{Y} \in \mathbf{St}(m, n, \mathbb{C})$ and $\boldsymbol{X} \in \mathbf{SL}_m(\mathbb{C})$ such that $\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{X}$.*

  By Lemma 1.1, problem (1.1) over $\mathbf{RSL}(m, n, \mathbb{C})$ is equivalent to minimizing

$$f : \mathbf{St}(m, n, \mathbb{C}) \times \mathbf{SL}_m(\mathbb{C}) \to \mathbb{R}^+, \quad (\boldsymbol{Y}, \boldsymbol{X}) \mapsto \sum_{\ell=1}^{\mathrm{L}} \|\operatorname{offdiag}\{\boldsymbol{W}^{(\ell)}\}\|^2, \tag{1.3}$$

where $\boldsymbol{W}^{(\ell)} = (\boldsymbol{Y}\boldsymbol{X})^{\blacklozenge}\boldsymbol{A}^{(\ell)}(\boldsymbol{Y}\boldsymbol{X}) \in \mathbb{C}^{m\times m}$.

- *Square case (second reformulation).* This is a special case of (1.3), when we assume $\boldsymbol{Y}_* \in \mathbf{St}(m,n,\mathbb{C})$ to be fixed (for example, it is found in advance by some other method, *e.g.*, PCA [16, 17, 18], which is a common procedure for dimensionality and noise reduction). Denote $\boldsymbol{B}^{(\ell)} = \boldsymbol{Y}_*^{\blacklozenge}\boldsymbol{A}^{(\ell)}\boldsymbol{Y}_*$ for $1 \le \ell \le \mathrm{L}$. Then the cost function (1.3) becomes

$$g : \mathbf{SL}_m(\mathbb{C}) \to \mathbb{R}^+, \quad \boldsymbol{X} \mapsto g(\boldsymbol{X}) = \sum_{\ell=1}^{\mathrm{L}} \|\operatorname{offdiag}\{\boldsymbol{W}^{(\ell)}\}\|^2, \tag{1.4}$$

where $\boldsymbol{W}^{(\ell)} = \boldsymbol{X}^{\blacklozenge}\boldsymbol{B}^{(\ell)}\boldsymbol{X} \in \mathbb{C}^{m\times m}$. Alternatively, this case may appear when $m = n$ in (1.1). Indeed, $\mathbf{RSL}(m,m,\mathbb{C}) = \{\boldsymbol{Z} \in \mathbb{C}^{m\times m}, |\det(\boldsymbol{Z})| = 1\}$, and since (1.1) is invariant with respect to multiplication by a unimodular scalar, we can optimize it over $\mathbf{SL}_m(\mathbb{C})$ instead.

1.2. Contributions. In this paper, to solve problem (1.3), which is defined on the product of $\mathbf{St}(m,n,\mathbb{C})$ and $\mathbf{SL}_m(\mathbb{C})$, the *gradient-based block coordinate descent* (BCD-G) algorithms (Algorithm 1) will be proposed in Subsection 2.1 (more detailedly in Subsection 5.2), which chooses a block optimization based on the Riemannian gradient. This is similar to the gradient-based way of choosing index pairs in the Jacobi-G algorithms on the orthogonal group [21, 25] or unitary group [45]. Then their *global convergence*[1] and *weak convergence*[2] will be established in Section 8 using the *Łojasiewicz gradient inequality* [24, 31, 2, 44], under the assumption that the iterates $\boldsymbol{\omega}_k$ are bounded, that is, there exists a universal positive constant $\mathrm{M}_\omega > 0$ such that

$$\|\boldsymbol{\omega}_k\| \le \mathrm{M}_\omega \tag{1.5}$$

always holds for all $k \ge 1$.

To solve problem (1.4), which is defined on the special linear group $\mathbf{SL}_m(\mathbb{C})$, the *gradient-based Jacobi-type* (Jacobi-G) algorithms will be proposed in Subsection 2.1 (more detailedly in Subsection 5.2), which can be seen as non-orthogonal analogues of the Jacobi-G algorithms on orthogonal group [21, 25] or unitary group [45]. Then their global and weak convergence will be established in Section 8 using the Łojasiewicz gradient inequality, under the assumption that the iterates $\boldsymbol{X}_k$ are bounded, that is, there exists a universal positive constant $\mathrm{M}_{\boldsymbol{X}} > 0$ such that

$$\|\boldsymbol{X}_k\| \le \mathrm{M}_{\boldsymbol{X}} \tag{1.6}$$

always holds for all $k \ge 1$. To our knowledge, this is the first time that the theoretical convergence is established for the Jacobi-type algorithms on $\mathbf{SL}_m(\mathbb{C})$.

1.3. Organization. The paper is organized as follows. In Section 2, we present the BCD-G and Jacobi-G algorithms, define four kinds of elementary transformations and give a summary of the main results. In Section 3, we recall the basics of first-order geometries on the Stiefel manifold $\mathbf{St}(m,n,\mathbb{C})$ and special linear group $\mathbf{SL}_m(\mathbb{C})$, as well as the convergence results related to Łojasiewicz inequality. In Section 4, we show the details of how to use the line search descent method to update the first block variable in $\mathbf{St}(m,n,\mathbb{C})$. In Section 5, we define four kinds of elementary functions and

[1] For any starting point, the iterates converge to a limit point as a whole sequence.

[2] Every accumulation point is a stationary point, *i.e.*, the Riemannian gradient is equal to 0.

present the details of three subalgorithms. In Section 6 and Section 7, we present the details of four kinds of elementary transformations for JADM problem. In Section 8, we prove our main results about the global and weak convergence of BCD-G and Jacobi-G algorithms. In Section 9, some experiments are conducted to compare the proposed algorithms. Section 10 concludes this paper with some final remarks and possible future work.

2. Gradient-based algorithmic framework and a summary of results.

2.1. BCD-G and Jacobi-G algorithms. Suppose that $\{\mathcal{M}_i\}_{1\leq i\leq d}$ are smooth manifolds. To minimize a smooth function

$$\tilde{f}:\ \mathcal{M}_1\times\mathcal{M}_2\times\cdots\times\mathcal{M}_d\longrightarrow\mathbb{R}^+, \tag{2.1}$$

a popular approach is the *block coordinate descent* (BCD) algorithm [9, 32, 33, 47, 48, 29]. In this method, only one block variable is updated at each iteration, while other block variables are fixed; in other words, the problem (2.1) is decomposed into a sequence of lower-dimensional optimization problems. In the BCD algorithm, there are different ways to choose blocks for optimization, including the *essentially cyclic, cyclic*, *random* fashions [47, 48] and the so-called *maximum block improvement* (MBI) method [15, 30].

If $d=2$, $\mathcal{M}_1=\mathbf{St}(m,n,\mathbb{C})$ and $\mathcal{M}_2=\mathbf{SL}_m(\mathbb{C})$, then problem (2.1) reduces to our cost function (1.3). For $\boldsymbol{\omega}=(\boldsymbol{Y},\boldsymbol{X})\in\mathbf{St}(m,n,\mathbb{C})\times\mathbf{SL}_m(\mathbb{C})$, we denote

$$f_{1,\boldsymbol{X}}:\boldsymbol{Y}\mapsto f(\boldsymbol{Y},\boldsymbol{X}),\quad f_{2,\boldsymbol{Y}}:\boldsymbol{X}\mapsto f(\boldsymbol{Y},\boldsymbol{X}), \tag{2.2}$$

as the two restricted functions, which are defined on $\mathbf{St}(m,n,\mathbb{C})$ and $\mathbf{SL}_m(\mathbb{C})$, respectively. For simplicity, we denote their *Riemannian gradients*[3] as $\operatorname{grad} f_1(\boldsymbol{\omega})\overset{\text{def}}{=}\operatorname{grad} f_{1,\boldsymbol{X}}(\boldsymbol{Y})$ and $\operatorname{grad} f_2(\boldsymbol{\omega})\overset{\text{def}}{=}\operatorname{grad} f_{2,\boldsymbol{Y}}(\boldsymbol{X})$, and the Riemannian gradient of f in (1.3) at $\boldsymbol{\omega}$ as $\operatorname{grad} f(\boldsymbol{\omega})$. To minimize the function (1.3), we now propose the following *gradient-based block coordinate descent* (BCD-G) algorithm in Algorithm 1.

Algorithm 1: BCD-G algorithm

Input: A starting point $\boldsymbol{\omega}_0=(\boldsymbol{Y}_0,\boldsymbol{X}_0)$, a positive constant $0<\upsilon<\sqrt{2}/2$.
Output: Sequence of iterates $\boldsymbol{\omega}_k=(\boldsymbol{Y}_k,\boldsymbol{X}_k)$.
for $k=1,2,\cdots$, **do**
  Choose $t_k=1$ or 2 such that the Riemannian gradients satisfy

$$\|\operatorname{grad} f_{t_k}(\boldsymbol{\omega}_{k-1})\|\geq\upsilon\|\operatorname{grad} f(\boldsymbol{\omega}_{k-1})\|; \tag{2.3}$$

  if $t_k=1$ **then**
    Update $\boldsymbol{Y}_k$ using the line search descent method (cf. Subsection 4.2);
    Set $\boldsymbol{X}_k=\boldsymbol{X}_{k-1}$;
  else
    Set $\boldsymbol{Y}_k=\boldsymbol{Y}_{k-1}$;
    Update $\boldsymbol{X}_k$ using elementary transformations (cf. Subalgorithm 1a to 1c).
  end if
end for

[3] See [3, Section 3.6] and Section 3 for a detailed definition.

In each iteration of Algorithm 1, instead of the frequently used cyclic or random fashion to choose the block for optimization, we choose the block $t_k = 1$ or 2 satisfying the inequality[4] (2.3). Since the Riemannian gradients are related as

$$\operatorname{grad} f(\boldsymbol{\omega}) = (\operatorname{grad} f_1(\boldsymbol{\omega}),\ \operatorname{grad} f_2(\boldsymbol{\omega})), \tag{2.4}$$

we have that $\|\operatorname{grad} f(\boldsymbol{\omega})\|^2 = \|\operatorname{grad} f_1(\boldsymbol{\omega})\|^2 + \|\operatorname{grad} f_2(\boldsymbol{\omega})\|^2$. Therefore, in each iteration, if $0 < \upsilon < \sqrt{2}/2$, we can always choose $t_k = 1$ or 2 such that the inequality (2.3) is satisfied, and thus Algorithm 1 is well defined.

In Algorithm 1, to update $\boldsymbol{Y}_k$, we choose the *line search descent* method [2, 3, 38, 39, 40], which will be detailedly presented in Section 4. To update $\boldsymbol{X}_k$, as in Jacobi-type methods, we use four kinds of elementary transformations (will be detailed introduced in Subsection 2.3), including the *Givens plane, plane upper triangular, plane lower triangular* and *plane diagonal transformations*[5]. We group these elementary transformations into three classes (GLU, GQU and GU) motivated by well-known matrix decompositions, which give rise to three different variants of Algorithm 1 (BCD-GLU, BCD-GQU and BCD-GU). We recall the matrix decompositions and related Lie groups in Subsection 2.2, before introducing the elementary transformations and their classes in Subsection 2.3.

Similarly to Algorithm 1, we propose optimization algorithms for minimization of the cost function (1.4) for the square case (second reformulation on $\mathbf{SL}_m(\mathbb{C})$). In these algorithms, $\boldsymbol{X}_k$ is updated with four elementary transformations, and therefore they are Jacobi-type algorithms. We summarize these *gradient-based Jacobi-type* (Jacobi-G) algorithms in Algorithm 2.

Algorithm 2: Jacobi-G algorithm

1: **Input:** A starting point $\boldsymbol{X}_0$.
2: **Output:** Sequence of iterates $\{\boldsymbol{X}_k\}_{k\geq 1}$.
3: **for** $k = 1, 2, \cdots$, **do**
4: Update $\boldsymbol{X}_k$ using elementary transformations (cf. Subalgorithm 1a to 1b).
5: **end for**

Algorithm 2 can be seen as a non-orthogonal analogue of the Jacobi-G algorithm in [21, 25, 45]. As with BCD-G, two types of Jacobi-G exist: Jacobi-GLU and Jacobi-GQU, based on GLU and GQU classes of elementary transformations, respectively. Roughly speaking, these algorithms are variants of Algorithm 1, where only $\boldsymbol{X}_k$ is updated.

2.2. Matrix decompositions and matrix groups. A matrix $\boldsymbol{X} \in \mathbb{C}^{m\times m}$ is said to be *upper triangular* if $\mathrm{X}_{ij} = 0$ for $i > j$. Let $\mathbf{UT}_m(\mathbb{C}) \subseteq \mathbf{GL}_m(\mathbb{C})$ be the *upper triangular subgroup*. Let $\mathbf{EUT}_m(\mathbb{C}) = \mathbf{UT}_m(\mathbb{C}) \cap \mathbf{SL}_m(\mathbb{C})$, *i.e.*, the set of upper triangular matrices with determinant equal to 1. Similarly, we let $\mathbf{LT}_m(\mathbb{C}) \subseteq \mathbb{C}^{m\times m}$ be the *lower triangular subgroup* and $\mathbf{ELT}_m(\mathbb{C}) = \mathbf{LT}_m(\mathbb{C}) \cap \mathbf{SL}_m(\mathbb{C})$. Let $\mathbf{U}_m(\mathbb{C}) \subseteq \mathbb{C}^{m\times m}$ be the *unitary group*, and $\mathbf{SU}_m(\mathbb{C}) \subseteq \mathbf{U}_m(\mathbb{C})$ be the *special unitary group*.

We first discuss the matrix decompositions of $\mathbf{SL}_m(\mathbb{C})$.

[4] The inequality (2.3) can be seen as a block coordinate analogue of [21, Eq. (3.3)] and [25, Eq. (10)].

[5] The reason why we use plane diagonal transformations will be shown in Section 5.

- Any matrix $\boldsymbol{X} \in \mathbf{SL}_m(\mathbb{C})$ has the *LU decomposition* [19] $\boldsymbol{X} = \boldsymbol{LU}$ with $\boldsymbol{L} \in \mathbf{LT}_m(\mathbb{C})$ and $\boldsymbol{U} \in \mathbf{UT}_m(\mathbb{C})$. We use the shorthand notation

$$\mathbf{SL}_m(\mathbb{C}) = \mathbf{ELT}_m(\mathbb{C}) \bullet \mathbf{EUT}_m(\mathbb{C}), \tag{2.5}$$

  where $\mathcal{A} \bullet \mathcal{B}$ denotes the set of all matrix product for matrices coming from two matrix sets $\mathcal{A}$ and $\mathcal{B}$. The decomposition (2.5) motivates the GLU class, which includes the plane lower triangular, plane upper triangular and plane diagonal transformations (Subalgorithm 1a), and is used in BCD-GLU and Jacobi-GLU algorithms.
- Any matrix $\boldsymbol{X} \in \mathbf{SL}_m(\mathbb{C})$ has the *QU decomposition*[6] $\boldsymbol{X} = \boldsymbol{QU}$ with $\boldsymbol{Q} \in \mathbf{SU}_m(\mathbb{C})$ and $\boldsymbol{U} \in \mathbf{UT}_m(\mathbb{C})$, which can be compactly written as

$$\mathbf{SL}_m(\mathbb{C}) = \mathbf{SU}_m(\mathbb{C}) \bullet \mathbf{EUT}_m(\mathbb{C}). \tag{2.6}$$

  The decomposition (2.6) motivates the GQU class, which includes the Givens plane, plane upper triangular and plane diagonal transformations (Subalgorithm 1b), and is used in BCD-GQU and Jacobi-GQU algorithms.

The decompositions mentioned above can be used to parameterize $\mathbf{RSL}(m, n, \mathbb{C})$. Indeed, Lemma 1.1 in the compact notation can be written as

$$\mathbf{RSL}(m, n, \mathbb{C}) = \mathbf{St}(m, n, \mathbb{C}) \bullet \mathbf{SL}_m(\mathbb{C}),$$

which gives rise to LU- and QU-based decompositions of $\mathbf{RSL}(m, n, \mathbb{C})$:

$$\mathbf{RSL}(m, n, \mathbb{C}) = \mathbf{St}(m, n, \mathbb{C}) \bullet \mathbf{ELT}_m(\mathbb{C}) \bullet \mathbf{EUT}_m(\mathbb{C}), \tag{2.7}$$

$$\mathbf{RSL}(m, n, \mathbb{C}) = \mathbf{St}(m, n, \mathbb{C}) \bullet \mathbf{SU}_m(\mathbb{C}) \bullet \mathbf{EUT}_m(\mathbb{C}). \tag{2.8}$$

Moreover, for $\mathbf{RSL}(m, n, \mathbb{C})$, a third decomposition is possible, using the fact that $\mathbf{St}(m, n, \mathbb{C}) \bullet \mathbf{SU}_m(\mathbb{C}) = \mathbf{St}(m, n, \mathbb{C})$. Then the equation (2.8) can be simplified as

$$\mathbf{RSL}(m, n, \mathbb{C}) = \mathbf{St}(m, n, \mathbb{C}) \bullet \mathbf{EUT}_m(\mathbb{C}), \tag{2.9}$$

which can also be interpreted as applying the QU decomposition to a rectangular matrix from $\mathbf{RSL}(m, n, \mathbb{C})$. This gives rise to the third class GU, which only includes the plane upper triangular and plane diagonal transformations (Subalgorithm 1c), and is used in BCD-GU.

2.3. Elementary transformations. Let us introduce a few more matrix groups. An upper triangular matrix $\boldsymbol{X}$ is said to be *unipotent* if it satisfies $\mathrm{X}_{ii} = 1$ for $1 \leq i \leq m$. Let $\mathbf{SUT}_m(\mathbb{C}) \subseteq \mathbf{EUT}_m(\mathbb{C})$ be the *upper unipotent subgroup* of unipotent upper triangular matrices. Similarly, we let $\mathbf{SLT}_m(\mathbb{C}) \subseteq \mathbf{ELT}_m(\mathbb{C})$ be the *lower unipotent subgroup*. Finally, a diagonal matrix $\boldsymbol{X} \in \mathbb{C}^{m \times m}$ is said to be a *diagonal transformation* if the product of all the diagonal elements is equal to 1. Let $\mathbf{D}_m(\mathbb{C}) \subseteq \mathbf{GL}_m(\mathbb{C})$ be the set of diagonal transformation matrices.

The elementary transformations are based on the following 2×2 matrices:

$$\mathbf{SUT}_2(\mathbb{C}) = \left\{ \begin{bmatrix} 1 & z \\ 0 & 1 \end{bmatrix}, z \in \mathbb{C} \right\}, \ \mathbf{SLT}_2(\mathbb{C}) = \left\{ \begin{bmatrix} 1 & 0 \\ z & 1 \end{bmatrix}, z \in \mathbb{C} \right\}, \ \mathbf{D}_2(\mathbb{C}) = \left\{ \begin{bmatrix} z & 0 \\ 0 & \frac{1}{z} \end{bmatrix}, z \in \mathbb{C}_* \right\},$$

as well as the 2×2 matrices from $\mathbf{SU}_2(\mathbb{C})$.

[6] This is also called QR decomposition in the literature.

Let (i,j) be a pair of indices satisfying $1 \leq i < j \leq m$. We introduce an operator $\mathcal{E}_{i,j} : \mathbb{C}^{2\times 2} \to \mathbb{C}^{m\times m}$ sending Ψ to $\boldsymbol{X} \in \mathbb{C}^{m\times m}$ satisfying

$$\begin{bmatrix} \mathrm{X}_{ii} & \mathrm{X}_{ij} \\ \mathrm{X}_{ji} & \mathrm{X}_{jj} \end{bmatrix} = \begin{bmatrix} \Psi_{11} & \Psi_{12} \\ \Psi_{21} & \Psi_{22} \end{bmatrix}, \quad \begin{cases} \mathrm{X}_{\ell\ell} = 1, & \text{if } \ell \notin \{i,j\}, \\ \mathrm{X}_{k\ell} = 0, & \text{otherwise.} \end{cases}$$

Now we define the following four elementary transformations on $\mathbf{SL}_m(\mathbb{C})$:

- $\boldsymbol{Q}^{(i,j,\Psi)} \stackrel{\mathsf{def}}{=} \mathcal{E}_{i,j}(\boldsymbol{\Psi})$: *Givens plane transformation* for $\boldsymbol{\Psi} \in \mathbf{SU}_2(\mathbb{C})$;
- $\boldsymbol{U}^{(i,j,\Psi)} \stackrel{\mathsf{def}}{=} \mathcal{E}_{i,j}(\boldsymbol{\Psi})$: *plane upper triangular transformation* for $\boldsymbol{\Psi} \in \mathbf{SUT}_2(\mathbb{C})$;
- $\boldsymbol{L}^{(i,j,\Psi)} \stackrel{\mathsf{def}}{=} \mathcal{E}_{i,j}(\boldsymbol{\Psi})$: *plane lower triangular transformation* for $\boldsymbol{\Psi} \in \mathbf{SLT}_2(\mathbb{C})$;
- $\boldsymbol{D}^{(i,j,\Psi)} \stackrel{\mathsf{def}}{=} \mathcal{E}_{i,j}(\boldsymbol{\Psi})$: *plane diagonal transformation* for $\boldsymbol{\Psi} \in \mathbf{D}_2(\mathbb{C})$.

Remark 2.1. These elementary transformations have all been used in the literature. The Givens transformations $\boldsymbol{Q}^{(i,j,\Psi)}$ were used very often in the Jacobi-type algorithms for joint approximate diagonalization of matrices or tensors by orthogonal or non-orthogonal transformations [18, 25, 45, 6, 5, 42]. Triangular transformations $\boldsymbol{U}^{(i,j,\Psi)}$ and $\boldsymbol{L}^{(i,j,\Psi)}$ also appeared many times in the Jacobi-type algorithms on $\mathbf{SL}_m(\mathbb{C})$ or $\mathbf{SL}_m(\mathbb{R})$ [4, 5, 35, 36, 37]. In the real case, the diagonal transformation $\boldsymbol{D}^{(i,j,\Psi)}$ was once used in [43].

The iterates $\boldsymbol{X}_k$ in Algorithm 1 and Algorithm 2 are updated multiplicatively as $\boldsymbol{X}_k = \boldsymbol{X}_{k-1}\boldsymbol{P}_k$, where $\boldsymbol{P}_k$ is an elementary transformation for a pair of indices (i_k, j_k) belonging to one of the following three classes. These three classes are inspired by equations (2.7), (2.8), (2.9) and by a similar idea as in [21, 25, 45]. We call them the *GLU* (based on LU decomposition), *GQU* (based on QU decomposition) and GU transformations, respectively.

- GLU class: $\boldsymbol{P}_k = \boldsymbol{L}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$, $\boldsymbol{U}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$ or $\boldsymbol{D}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$;
- GQU class: $\boldsymbol{P}_k = \boldsymbol{Q}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$, $\boldsymbol{U}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$ or $\boldsymbol{D}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$;
- GU class: $\boldsymbol{P}_k = \boldsymbol{U}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$ or $\boldsymbol{D}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$.

The choice of the pair (i_k, j_k), the matrix $\boldsymbol{\Psi}_k^*$ and the particular type of transformations in each class will be given in Subalgorithm 1a, Subalgorithm 1b and Subalgorithm 1c. The algorithms and their convergence results are summarized in Table 1.

TABLE 1
A summary of the proposed algorithms

Model	Proposed algorithms	Location	Elementary transformations	Global convergence	Weak convergence
First reformulation (1.3) on $\mathbf{St}(m,n,\mathbb{C}) \times \mathbf{SL}_m(\mathbb{C})$	BCD-GLU	Algorithm 1 & Subalgorithm 1a	$\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{D}$	Theorem 8.1	Theorem 8.2
	BCD-GQU	Algorithm 1 & Subalgorithm 1b	$\boldsymbol{Q}, \boldsymbol{U}, \boldsymbol{D}$		
	BCD-GU	Algorithm 1 & Subalgorithm 1c	$\boldsymbol{U}, \boldsymbol{D}$		
Second reformulation (1.4) on $\mathbf{SL}_m(\mathbb{C})$	Jacobi-GLU	Algorithm 2 & Subalgorithm 1a	$\boldsymbol{L}, \boldsymbol{U}, \boldsymbol{D}$	Theorem 8.3	Theorem 8.4
	Jacobi-GQU	Algorithm 2 & Subalgorithm 1b	$\boldsymbol{Q}, \boldsymbol{U}, \boldsymbol{D}$		

Remark 2.2. While the algorithms and convergence results described in this paper are provided for complex matrices, complex Stiefel manifold $\mathbf{St}(m,n,\mathbb{C})$ and complex special linear group $\mathbf{SL}_m(\mathbb{C})$, they also remain valid in the real case.

3. Geometries on $\mathbf{St}(m,n,\mathbb{C})$ and $\mathbf{SL}_m(\mathbb{C})$.

3.1. Notations. Let $1 \leq m \leq n$. For a complex matrix $\boldsymbol{Z} \in \mathbb{C}^{n\times m}$ and a complex number $z \in \mathbb{C}$, we write the real and imaginary parts as $\boldsymbol{Z} = \boldsymbol{Z}^{\Re} + i\boldsymbol{Z}^{\Im}$ and $z = \Re(z) + i\Im(z)$, respectively. For complex matrices $\boldsymbol{Z}_1, \boldsymbol{Z}_2 \in \mathbb{C}^{n\times m}$, we introduce the following real-valued inner product

$$\langle \boldsymbol{Z}_1, \boldsymbol{Z}_2\rangle_{\Re} \stackrel{\text{def}}{=} \langle \boldsymbol{Z}_1^{\Re}, \boldsymbol{Z}_2^{\Re}\rangle + \langle \boldsymbol{Z}_1^{\Im}, \boldsymbol{Z}_2^{\Im}\rangle = \Re\left(\operatorname{tr}(\boldsymbol{Z}_1^{\mathsf{H}}\boldsymbol{Z}_2)\right), \tag{3.1}$$

which makes $\mathbb{C}^{n\times m}$ a real Euclidean space of dimension $2nm$. Let $h : \mathbb{C}^{n\times m} \to \mathbb{R}$ be a differentiable function and $\boldsymbol{Z} \in \mathbb{C}^{n\times m}$. We denote by $\frac{\partial h}{\partial \boldsymbol{Z}^{\Re}}, \frac{\partial h}{\partial \boldsymbol{Z}^{\Im}} \in \mathbb{R}^{n\times m}$ the matrix Euclidean derivatives of h with respect to real and imaginary parts of $\boldsymbol{Z}$. The *Wirtinger derivatives* [1, 11, 23] are defined as

$$\frac{\partial h}{\partial \boldsymbol{Z}^*} \stackrel{\text{def}}{=} \frac{1}{2}\left(\frac{\partial h}{\partial \boldsymbol{Z}^{\Re}} + i\frac{\partial h}{\partial \boldsymbol{Z}^{\Im}}\right), \quad \frac{\partial h}{\partial \boldsymbol{Z}} \stackrel{\text{def}}{=} \frac{1}{2}\left(\frac{\partial h}{\partial \boldsymbol{Z}^{\Re}} - i\frac{\partial h}{\partial \boldsymbol{Z}^{\Im}}\right).$$

Then the Euclidean gradient of h with respect to the inner product (3.1) becomes

$$\nabla h(\boldsymbol{Z}) = \frac{\partial h}{\partial \boldsymbol{Z}^{\Re}} + i\frac{\partial h}{\partial \boldsymbol{Z}^{\Im}} = 2\frac{\partial h}{\partial \boldsymbol{Z}^*}. \tag{3.2}$$

For real matrices $\boldsymbol{Z}_1, \boldsymbol{Z}_2 \in \mathbb{R}^{n\times m}$, we see that (3.1) becomes the standard inner product, and (3.2) becomes the standard Euclidean gradient. We denote by $\mathbb{S}_2 \subseteq \mathbb{R}^3$ the unit sphere, and $\mathbb{C}_* = \mathbb{C}\backslash\{0\}$.

3.2. Riemannian gradient on $\mathbf{St}(m,n,\mathbb{C})$. For a matrix $\boldsymbol{C} \in \mathbb{C}^{m\times m}$, we denote $\operatorname{sym}(\boldsymbol{C}) \stackrel{\text{def}}{=} \frac{1}{2}(\boldsymbol{C} + \boldsymbol{C}^{\mathsf{H}})$ and $\operatorname{skew}(\boldsymbol{C}) \stackrel{\text{def}}{=} \frac{1}{2}(\boldsymbol{C} - \boldsymbol{C}^{\mathsf{H}})$. Let $\mathbf{T}_{\boldsymbol{Y}}\mathbf{St}(m,n,\mathbb{C})$ be the *tangent space* to $\mathbf{St}(m,n,\mathbb{C})$ at a point $\boldsymbol{Y} \in \mathbf{St}(m,n,\mathbb{C})$. Let $\boldsymbol{Y}_\perp \in \mathbb{C}^{n\times(n-m)}$ be an orthogonal complement of $\boldsymbol{Y}$, that is, $[\boldsymbol{Y}, \boldsymbol{Y}_\perp] \in \mathbb{C}^{n\times n}$ is a unitary matrix. By [34, Definition 6], we know that

$$\mathbf{T}_{\boldsymbol{Y}}\mathbf{St}(m,n,\mathbb{C}) = \{\boldsymbol{V} \in \mathbb{C}^{n\times m}, \boldsymbol{V} = \boldsymbol{Y}\boldsymbol{C} + \boldsymbol{Y}_\perp\boldsymbol{B}, \boldsymbol{C} \in \mathbb{C}^{m\times m}, \boldsymbol{C}^{\mathsf{H}} + \boldsymbol{C} = 0, \boldsymbol{B} \in \mathbb{C}^{(n-m)\times m}\},$$

which is a $(2nm - m^2)$-dimensional vector space. The orthogonal projection of a matrix $\xi \in \mathbb{C}^{n\times m}$ onto $\mathbf{T}_{\boldsymbol{Y}}\mathbf{St}(m,n,\mathbb{C})$ is

$$\operatorname{Proj}_{\boldsymbol{Y}}\xi = (\boldsymbol{I}_n - \boldsymbol{Y}\boldsymbol{Y}^{\mathsf{H}})\xi + \boldsymbol{Y}\operatorname{skew}(\boldsymbol{Y}^{\mathsf{H}}\xi) = \xi - \boldsymbol{Y}\operatorname{sym}(\boldsymbol{Y}^{\mathsf{H}}\xi). \tag{3.3}$$

We denote $\operatorname{Proj}^{\perp}_{\boldsymbol{Y}}\xi \stackrel{\text{def}}{=} \xi - \operatorname{Proj}_{\boldsymbol{Y}}\xi$. Let $p : \mathbf{St}(m,n,\mathbb{C}) \to \mathbb{R}$ be a differentiable function, and $\boldsymbol{Y} \in \mathbf{St}(m,n,\mathbb{C})$. Note that $\mathbf{St}(m,n,\mathbb{C})$ is an embedded submanifold of the Euclidean space $\mathbb{C}^{n\times m}$. By equation (3.3), we have the Riemannian gradient of p at $\boldsymbol{Y}$ as:

$$\operatorname{grad} p(\boldsymbol{Y}) = \operatorname{Proj}_{\boldsymbol{Y}}\nabla p(\boldsymbol{Y}) = \nabla p(\boldsymbol{Y}) - \boldsymbol{Y}\operatorname{sym}(\boldsymbol{Y}^{\mathsf{H}}\nabla p(\boldsymbol{Y})). \tag{3.4}$$

By [3, Example 5.4.2], the *exponential map* at $\boldsymbol{Y}$ is defined as

$$\begin{aligned}\operatorname{Exp}_{\boldsymbol{Y}} : \mathbf{T}_{\boldsymbol{Y}}\mathbf{St}(m,n,\mathbb{C}) &\longrightarrow \mathbf{St}(m,n,\mathbb{C})\\ \boldsymbol{V} &\longmapsto [\boldsymbol{Y}, \boldsymbol{V}]\exp\left(\begin{bmatrix}\boldsymbol{Y}^H\boldsymbol{V} & -\boldsymbol{V}^{\mathsf{H}}\boldsymbol{V}\\ \boldsymbol{I}_m & \boldsymbol{Y}^H\boldsymbol{V}\end{bmatrix}\right)\begin{bmatrix}\exp\left(-\boldsymbol{Y}^H\boldsymbol{V}\right)\\ \boldsymbol{0}_{m\times m}\end{bmatrix},\end{aligned} \tag{3.5}$$

where $\exp(\cdot)$ is the matrix exponential function [3, 8, 20].

3.3. Riemannian gradient on $\mathbf{SL}_m(\mathbb{C})$. Let $\mathfrak{sl}_m(\mathbb{C}) \stackrel{\text{def}}{=} \{\boldsymbol{X} \in \mathbb{C}^{m\times m}, \mathrm{tr}(\boldsymbol{X}) = 0\}$ be the *Lie algebra* [8] of the complex special linear group $\mathbf{SL}_m(\mathbb{C})$. Then the tangent space to $\mathbf{SL}_m(\mathbb{C})$ at a point $\boldsymbol{X} \in \mathbf{SL}_m(\mathbb{C})$ can be constructed [8, Eq. (3.7),(3.8)] by

$$\mathbf{T}_{\boldsymbol{X}}\mathbf{SL}_m(\mathbb{C}) = \{\boldsymbol{X}\Omega, \Omega \in \mathfrak{sl}_m(\mathbb{C})\}. \tag{3.6}$$

Let $\mathfrak{su}_m(\mathbb{C}) \stackrel{\text{def}}{=} \{\boldsymbol{X} \in \mathbb{C}^{m\times m}, \boldsymbol{X}^{\mathsf{H}} = -\boldsymbol{X}, \mathrm{tr}(\boldsymbol{X}) = 0\}$ be the Lie algebra of the special unitary group $\mathbf{SU}_m(\mathbb{C})$. Then the tangent space to $\mathbf{SU}_m(\mathbb{C})$ at a point $\boldsymbol{X} \in \mathbf{SU}_m(\mathbb{C})$ can be constructed [8, Eq. (3.15)] by $\mathbf{T}_{\boldsymbol{X}}\mathbf{SU}_m(\mathbb{C}) = \{\boldsymbol{X}\Omega, \Omega \in \mathfrak{sl}_m(\mathbb{C})\}$.

Let $\mathbf{T}_{\boldsymbol{X}}\mathbf{SL}_m(\mathbb{C})$ be the tangent space to $\mathbf{SL}_m(\mathbb{C})$ at a point $\boldsymbol{X} \in \mathbf{SL}_m(\mathbb{C})$ as in (3.6). For tangent matrices $\boldsymbol{V}_1, \boldsymbol{V}_2 \in \mathbf{T}_{\boldsymbol{X}}\mathbf{SL}_m(\mathbb{C})$, we use the *left invariant* [3], [4, Eq. (6.2)] Riemannian metric

$$\langle \boldsymbol{V}_1, \boldsymbol{V}_2\rangle_{\boldsymbol{X}} \stackrel{\text{def}}{=} \left\langle \boldsymbol{X}^{-1}\boldsymbol{V}_1, \boldsymbol{X}^{-1}\boldsymbol{V}_2\right\rangle_{\Re} = \Re\left(\mathrm{tr}(\boldsymbol{V}_1^{\mathsf{H}}(\boldsymbol{X}\boldsymbol{X}^{\mathsf{H}})^{-1}\boldsymbol{V}_2)\right).$$

Let $g : \mathbf{SL}_m(\mathbb{C}) \longrightarrow \mathbb{R}^+$ be a differentiable function, and $\boldsymbol{X} \in \mathbf{SL}_m(\mathbb{C})$. Then the Riemannian gradient of g at $\boldsymbol{X}$ is the orthogonal projection [4, Lemma 6.2] of its Euclidean gradient $\nabla g(\boldsymbol{X})$ to $\mathbf{T}_{\boldsymbol{X}}\mathbf{SL}_m(\mathbb{C})$, that is,

$$\mathrm{grad}\, g(\boldsymbol{X}) = \boldsymbol{X}\left(\boldsymbol{X}^{\mathsf{H}}\nabla g(\boldsymbol{X}) - \frac{\mathrm{tr}(\boldsymbol{X}^{\mathsf{H}}\nabla g(\boldsymbol{X}))}{n}\boldsymbol{I}_n\right). \tag{3.7}$$

We denote $\boldsymbol{\Lambda}(\boldsymbol{X}) \stackrel{\text{def}}{=} \boldsymbol{X}^{-1}\,\mathrm{grad}\, g(\boldsymbol{X}) \in \mathfrak{sl}_m(\mathbb{C})$ for $\boldsymbol{X} \in \mathbf{SL}_m(\mathbb{C})$, which will be frequently used in this paper.

In what follows, we will use the following exponential map

$$\mathrm{Exp}_{\boldsymbol{X}} : \mathbf{T}_{\boldsymbol{X}}\mathbf{SL}_m(\mathbb{C}) \to \mathbf{SL}_m(\mathbb{C}), \quad \boldsymbol{X}\Omega \mapsto \boldsymbol{X}\exp(\Omega), \tag{3.8}$$

where $\exp(\cdot)$ is the matrix exponential function [3, 8, 20]. For any tangent matrix $\boldsymbol{V} \in \mathbf{T}_{\boldsymbol{X}}\mathbf{SL}_m(\mathbb{C})$, we have the following relationship between $\mathrm{Exp}_{\boldsymbol{X}}$ in (3.8) and the Riemannian gradient [3, Eq. (3.31)]:

$$\langle \boldsymbol{V}, \mathrm{grad}\, g(\boldsymbol{X})\rangle_{\boldsymbol{X}} = \left(\frac{d}{dt}g(\mathrm{Exp}_{\boldsymbol{X}}(t\boldsymbol{V}))\right)\Big|_{t=0}, \tag{3.9}$$

which will be used in the proof of Lemma 5.1.

3.4. Tangent spaces to other matrix groups. A matrix $\boldsymbol{X} \in \mathbb{C}^{m\times m}$ is said to be *strictly upper triangular* if $\mathrm{X}_{ij} = 0$ for $i \geq j$. Let $\mathfrak{sut}_m(\mathbb{C}) \subseteq \mathbb{C}^{m\times m}$ be the set of strictly upper triangular matrices. Then the tangent space to $\mathbf{SUT}_m(\mathbb{C})$ at a point $\boldsymbol{X} \in \mathbf{SUT}_m(\mathbb{C})$ can be constructed [8, Eq. (3.11)], [4, Section 6.4] by $\mathbf{T}_{\boldsymbol{X}}\mathbf{SUT}_m(\mathbb{C}) = \{\boldsymbol{X}\Omega, \Omega \in \mathfrak{sut}_m(\mathbb{C})\}$. Similar as above, we let $\mathfrak{slt}_n(\mathbb{C}) \subseteq \mathbb{C}^{m\times m}$ be the set of *strictly lower triangular* matrices. Then the tangent space to $\mathbf{SLT}_m(\mathbb{C})$ at a point $\boldsymbol{X} \in \mathbf{SLT}_m(\mathbb{C})$ can be constructed by $\mathbf{T}_{\boldsymbol{X}}\mathbf{SLT}_m(\mathbb{C}) = \{\boldsymbol{X}\Omega, \Omega \in \mathfrak{slt}_m(\mathbb{C})\}$. Let $\mathfrak{d}_m(\mathbb{C}) \subseteq \mathbb{C}^{m\times m}$ be the set of diagonal traceless matrices. Then the tangent space to $\mathbf{D}_m(\mathbb{C})$ at a point $\boldsymbol{X} \in \mathbf{D}_m(\mathbb{C})$ can be constructed by $\mathbf{T}_{\boldsymbol{X}}\mathbf{D}_m(\mathbb{C}) = \{\boldsymbol{X}\Omega, \Omega \in \mathfrak{d}_m(\mathbb{C})\}$. In particular, for the case $m = 2$, we have

$$\mathfrak{sut}_2(\mathbb{C}) = \left\{\begin{bmatrix}0 & z\\ 0 & 0\end{bmatrix}, z \in \mathbb{C}\right\}, \quad \mathfrak{slt}_2(\mathbb{C}) = \left\{\begin{bmatrix}0 & 0\\ z & 0\end{bmatrix}, z \in \mathbb{C}\right\}, \quad \mathfrak{d}_2(\mathbb{C}) = \left\{\begin{bmatrix}z & 0\\ 0 & -z\end{bmatrix}, z \in \mathbb{C}\right\}.$$

3.5. Inequalities for convergence analysis. We recall some definitions and results about the Łojasiewicz gradient inequality [24, 31, 2, 44]. These results were used in [26, 45] to prove the global convergence of Jacobi-G algorithms on the orthogonal and unitary groups, and will be used in this paper as well.

DEFINITION 3.1 ([41, Definition 2.1]). *Let $\mathcal{M} \subseteq \mathbb{R}^d$ be a Riemannian submanifold, and $\varphi : \mathcal{M} \to \mathbb{R}$ be a differentiable function. The function $\varphi : \mathcal{M} \to \mathbb{R}$ is said to satisfy a* Łojasiewicz gradient inequality *at $\boldsymbol{x} \in \mathcal{M}$, if there exist $\delta > 0$, $\zeta \in (0, \frac{1}{2}]$ and a neighborhood $\mathcal{U}$ in $\mathcal{M}$ of $\boldsymbol{x}$ such that for all $\boldsymbol{y} \in \mathcal{U}$, it follows that*

$$|\varphi(\boldsymbol{y}) - \varphi(\boldsymbol{x})|^{1-\zeta} \leq \delta \| \operatorname{grad} \varphi(\boldsymbol{y}) \|. \tag{3.10}$$

LEMMA 3.2 ([41, Proposition 2.2]). *Let $\mathcal{M} \subseteq \mathbb{R}^d$ be an analytic submanifold[7] and $\varphi : \mathcal{M} \to \mathbb{R}$ be a real analytic function. Then φ satisfies a Łojasiewicz gradient inequality (3.10) at any $\boldsymbol{x} \in \mathcal{M}$.*

THEOREM 3.3 ([41, Theorem 2.3]). *Let $\mathcal{M} \subseteq \mathbb{R}^d$ be an analytic submanifold and $\{\boldsymbol{x}_k\}_{k\geq 1} \subseteq \mathcal{M}$. Suppose that φ is real analytic and, for large enough k,*
(i) there exists $\sigma > 0$ such that

$$\varphi(\boldsymbol{x}_k) - \varphi(\boldsymbol{x}_{k+1}) \geq \sigma \| \operatorname{grad} \varphi(\boldsymbol{x}_k) \| \| \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \|;$$

(ii) $\operatorname{grad} \varphi(\boldsymbol{x}_k) = 0$ implies that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$.
Then, if $\boldsymbol{x}_*$ is an accumulation point of $\{\boldsymbol{x}_k\}_{k\geq 1}$, it is the limit point.

Since the special linear group $\mathbf{SL}_m(\mathbb{C})$ is not compact, the iterates $\{\boldsymbol{\omega}_k\}_{k\geq 1}$ in Algorithm 1 for cost function (1.3) may have no accumulation point. However, if there exists an accumulation point, we have the following result about its global convergence, which is a direct consequence of Theorem 3.3 and inequality (2.3).

LEMMA 3.4. *Suppose that, in Algorithm 1 for cost function (1.3), the iterates $\{\boldsymbol{\omega}_k\}_{k\geq 1}$ satisfy that, for large enough k,*
(i) there exists $\sigma > 0$ such that

$$f(\boldsymbol{\omega}_{k-1}) - f(\boldsymbol{\omega}_k) \geq \sigma \| \operatorname{grad} f_{t_k}(\boldsymbol{\omega}_{k-1}) \| \| \boldsymbol{\omega}_k - \boldsymbol{\omega}_{k-1} \|; \tag{3.11}$$

(ii) $\operatorname{grad} f_{t_k}(\boldsymbol{\omega}_{k-1}) = 0$ implies that $\boldsymbol{\omega}_k = \boldsymbol{\omega}_{k-1}$.
Then, if $\boldsymbol{\omega}_*$ is an accumulation point of the iterates $\{\boldsymbol{\omega}_k\}_{k\geq 1}$, it is the limit point.

We also have the following result about its weak convergence, which can be proved easily by inequality (2.3) and the fact that $f(\omega) \geq 0$.

LEMMA 3.5. *In Algorithm 1 for cost function (1.3), if there exists $\eta > 0$ such that*

$$f(\boldsymbol{\omega}_{k-1}) - f(\boldsymbol{\omega}_k) \geq \eta \| \operatorname{grad} f_{t_k}(\boldsymbol{\omega}_{k-1}) \|^2 \tag{3.12}$$

always holds, then $\lim_{k\to\infty} \operatorname{grad} f(\boldsymbol{\omega}_{k-1}) = 0$. In particular, if $\boldsymbol{\omega}_*$ is an accumulation point of the iterates $\{\boldsymbol{\omega}_k\}_{k\geq 1}$, then $\boldsymbol{\omega}_*$ is a stationary point of f.

4. Line search descent method on $\mathbf{St}(m, n, \mathbb{C})$. Let f be the cost function (1.3) defined on the product of $\mathbf{St}(m, n, \mathbb{C})$ and $\mathbf{SL}_m(\mathbb{C})$. Let $\boldsymbol{\omega}_{k-1} = (\boldsymbol{Y}_{k-1}, \boldsymbol{X}_{k-1})$ and $p = f_{1,\boldsymbol{X}_{k-1}}$ be the first restricted function. Denote $\boldsymbol{X} = \boldsymbol{X}_{k-1}$ for simplicity. Then the restricted function p can be expressed as

$$p : \mathbf{St}(m, n, \mathbb{C}) \to \mathbb{R}^+, \quad \boldsymbol{Y} \mapsto \sum_{\ell=1}^{\mathrm{L}} \| \operatorname{offdiag}\{\boldsymbol{W}^{(\ell)}\} \|^2, \tag{4.1}$$

[7] See [22, Definition 2.7.1] or [26, Definition 5.1] for a definition of an analytic submanifold.

where $\boldsymbol{W}^{(\ell)} = \boldsymbol{X}^{\blacklozenge}\boldsymbol{Y}^{\blacklozenge}\boldsymbol{A}^{(\ell)}\boldsymbol{Y}\boldsymbol{X}$, and $(\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{T}}$ or $(\cdot)^{\mathsf{H}}$. In this section, we adopt the *line search descent* [2, 3, 38, 39, 40] method on $\mathbf{St}(m, n, \mathbb{C})$ to find the next iterate $\boldsymbol{Y}_k$ for the restricted function p in (4.1).

4.1. Riemannian gradient. We first present a lemma, which can be obtained by direct calculations. This result will help us to obtain the Riemannian gradient of the restricted function p in (4.1).

LEMMA 4.1. *Let $\boldsymbol{A} \in \mathbb{C}^{n\times n}$ and the function $\tilde{p}$ be defined as*

$$\tilde{p} : \mathbb{C}^{n\times m} \to \mathbb{R}^+, \quad \boldsymbol{Z} \mapsto \|\operatorname{offdiag}\{\boldsymbol{W}\}\|^2,$$

where $\boldsymbol{W} = \boldsymbol{Z}^{\blacklozenge}\boldsymbol{A}\boldsymbol{Z}$. Denote $\boldsymbol{V} = \boldsymbol{A}\boldsymbol{Z} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_m] \in \mathbb{C}^{n\times m}$ and $\overline{\boldsymbol{V}} = \boldsymbol{A}^{\blacklozenge}\boldsymbol{Z} = [\bar{\boldsymbol{v}}_1, \cdots, \bar{\boldsymbol{v}}_m] \in \mathbb{C}^{n\times m}$. Denote $\boldsymbol{Z} = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_m]$. Then the Euclidean gradient is

$$\nabla\tilde{p}(\boldsymbol{Z}) = 2\left(\sum_{j\neq 1}\boldsymbol{v}_j\boldsymbol{v}_j^{\mathsf{H}}\boldsymbol{z}_1, \cdots, \sum_{j\neq m}\boldsymbol{v}_j\boldsymbol{v}_j^{\mathsf{H}}\boldsymbol{z}_m\right) + 2\left(\sum_{j\neq 1}\bar{\boldsymbol{v}}_j(\bar{\boldsymbol{v}}_j)^{\mathsf{H}}\boldsymbol{z}_1, \cdots, \sum_{j\neq m}\bar{\boldsymbol{v}}_j(\bar{\boldsymbol{v}}_j)^{\mathsf{H}}\boldsymbol{z}_m\right).$$

In particular, it satisfies

$$\boldsymbol{Z}\boldsymbol{Z}^{\mathsf{H}}\nabla\tilde{p}(\boldsymbol{Z}) = 2\boldsymbol{Z}\Upsilon(\boldsymbol{W}), \tag{4.2}$$

where $\Upsilon(\boldsymbol{W}) \in \mathbb{C}^{m\times m}$ is defined as

$$\Upsilon(\boldsymbol{W}) \overset{\text{def}}{=} \begin{cases} \boldsymbol{W}\operatorname{offdiag}\{\boldsymbol{W}\}^{\mathsf{H}} + \boldsymbol{W}^{H}\operatorname{offdiag}\{\boldsymbol{W}\}, & \text{if } (\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{H}}; \\ \boldsymbol{W}^{*}\operatorname{offdiag}\{\boldsymbol{W}\}^{\mathsf{T}} + \boldsymbol{W}^{H}\operatorname{offdiag}\{\boldsymbol{W}\}, & \text{if } (\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{T}}. \end{cases} \tag{4.3}$$

LEMMA 4.2. *Let $\boldsymbol{W}^{(\ell)}$ and the function p be as in (4.1). Then the Euclidean gradient satisfies*

$$\boldsymbol{Y}^{\mathsf{H}}\nabla p(\boldsymbol{Y}) = 2(\boldsymbol{X}^{\mathsf{H}})^{-1}\sum_{\ell=1}^{\mathrm{L}}\Upsilon(\boldsymbol{W}^{(\ell)})\boldsymbol{X}^{\mathsf{H}}, \tag{4.4}$$

where $\Upsilon(\boldsymbol{W}^{(\ell)})$ is as in (4.3).

Proof. By the product rule, we see that $\nabla p(\boldsymbol{Y}) = \nabla\tilde{p}(\boldsymbol{Y}\boldsymbol{X})\boldsymbol{X}^{\mathsf{H}}$. Then, by equation (4.2), we have that

$$\boldsymbol{X}\boldsymbol{X}^{\mathsf{H}}\boldsymbol{Y}^{\mathsf{H}}\nabla p(\boldsymbol{Y}) = \boldsymbol{Y}^{\mathsf{H}}\boldsymbol{Y}\boldsymbol{X}(\boldsymbol{Y}\boldsymbol{X})^{\mathsf{H}}\nabla\tilde{p}(\boldsymbol{Y}\boldsymbol{X})\boldsymbol{X}^{\mathsf{H}} = 2\boldsymbol{Y}^{\mathsf{H}}\boldsymbol{Y}\boldsymbol{X}\sum_{\ell=1}^{\mathrm{L}}\Upsilon(\boldsymbol{W}^{(\ell)})\boldsymbol{X}^{\mathsf{H}}.$$

Note that $\boldsymbol{X}$ is invertible. The proof is complete. ∎

Now, by equations (4.4) and (3.4), we see that the Riemannian gradient of the function p in (4.1) satisfies

$$\boldsymbol{Y}^{\mathsf{H}}\operatorname{grad} p(\boldsymbol{Y}) = (\boldsymbol{X}^{\mathsf{H}})^{-1}\sum_{\ell=1}^{\mathrm{L}}\Upsilon(\boldsymbol{W}^{(\ell)})\boldsymbol{X}^{\mathsf{H}} - \boldsymbol{X}\sum_{\ell=1}^{\mathrm{L}}\Upsilon(\boldsymbol{W}^{(\ell)})^{\mathsf{H}}\boldsymbol{X}^{-1}. \tag{4.5}$$

4.2. Line search descent method. We now begin to present more details about the line search descent method [2, 3, 38, 39, 40] on $\mathbf{St}(m,n,\mathbb{C})$. In this method, we choose the next iteration as

$$\boldsymbol{Y}_k = \mathrm{Exp}_{\boldsymbol{Y}_{k-1}}(t_{k-1}\boldsymbol{V}_{k-1}), \tag{4.6}$$

where $\boldsymbol{V}_{k-1}$ is the *search direction*, t_{k-1} is the *step size* and $\mathrm{Exp}_{\boldsymbol{Y}_{k-1}}$ is the exponential map defined in (3.5). We always choose the search direction $\boldsymbol{V}_{k-1}$ such that

$$\langle \mathrm{grad}\, p(\boldsymbol{Y}_{k-1}), \boldsymbol{V}_{k-1}\rangle_{\boldsymbol{Y}_{k-1}} \leq -\delta_s \|\mathrm{grad}\, p(\boldsymbol{Y}_{k-1})\|\|\boldsymbol{V}_{k-1}\|, \tag{4.7}$$

where $0<\delta_s<1$ is a fixed positive constant. We say that the step size t_{k-1} satisfies the *Armijo condition*[8], if

$$p(\boldsymbol{Y}_k) \leq p(\boldsymbol{Y}_{k-1}) + \delta_w t_{k-1}\langle \mathrm{grad}\, p(\boldsymbol{Y}_{k-1}), \boldsymbol{V}_{k-1}\rangle_{\boldsymbol{Y}_{k-1}}, \tag{4.8}$$

where $0<\delta_w<1$ is a fixed positive constant. We say that the step size t_{k-1} satisfies the *curvature condition*, if

$$\langle \mathrm{grad}\, p(\boldsymbol{Y}_k), \mathbf{D}\mathrm{Exp}_{\boldsymbol{Y}_{k-1}}(t_{k-1}\boldsymbol{V}_{k-1})[\boldsymbol{V}_{k-1}]\rangle_{\boldsymbol{Y}_k} \geq \delta_c \langle \mathrm{grad}\, p(\boldsymbol{Y}_{k-1}), \boldsymbol{V}_{k-1}\rangle_{\boldsymbol{Y}_{k-1}}, \tag{4.9}$$

where $\delta_w<\delta_c<1$ is a fixed positive constant. The conditions (4.8) and (4.9) are known collectively as the *Wolfe conditions*. As in the Euclidean space case [38, Lemma 3.1], it was shown [39, 40] that we can always choose the step size t_{k-1} such that the conditions (4.8) and (4.9) are both satisfied. It is not difficult to see that there exists $\mathrm{M}_e>0$ such that

$$\|\mathrm{Exp}_{\boldsymbol{Y}}(\boldsymbol{V}_1) - \mathrm{Exp}_{\boldsymbol{Y}}(\boldsymbol{V}_2)\| \leq \mathrm{M}_e\|\boldsymbol{V}_1 - \boldsymbol{V}_2\|,$$

for any $\boldsymbol{Y}\in\mathbf{St}(m,n,\mathbb{C})$ and $\boldsymbol{V}_1,\boldsymbol{V}_2\in\mathbf{T}_{\boldsymbol{Y}}\mathbf{St}(m,n,\mathbb{C})$. Then the next result follows directly.

LEMMA 4.3. *If we choose the next iterate $\boldsymbol{Y}_k$ as in (4.6) such that the conditions (4.7) and (4.8) are both satisfied, then we have*

$$p(\boldsymbol{Y}_{k-1}) - p(\boldsymbol{Y}_k) \geq \delta_s\delta_w\|\mathrm{grad}\, p(\boldsymbol{Y}_{k-1})\|\|t_{k-1}\boldsymbol{V}_{k-1}\| \geq \sigma_p\|\mathrm{grad}\, p(\boldsymbol{Y}_{k-1})\|\|\boldsymbol{Y}_k - \boldsymbol{Y}_{k-1}\|,$$

where $\sigma_p = (\delta_s\delta_w)/\mathrm{M}_e$.

We also have the next result, a simple corollary of the proof in [39, Theorem 2].

LEMMA 4.4. *If we choose the next iterate $\boldsymbol{Y}_k$ as in (4.6) such that the conditions (4.7), (4.8) and (4.9) are all satisfied, then we have*

$$p(\boldsymbol{Y}_{k-1}) - p(\boldsymbol{Y}_k) \geq \eta_p\|\mathrm{grad}\, p(\boldsymbol{Y}_{k-1})\|^2, \tag{4.10}$$

where $\eta_p>0$ is a fixed positive constant.

5. Elementary functions and three subalgorithms. In this section, we define four kinds of elementary functions and present the details of three subalgorithms.

[8] It is also known as the *first Wolfe condition* in the literature.

5.1. Elementary functions and their derivatives. Let $g : \mathbf{SL}_m(\mathbb{C}) \to \mathbb{R}^+$ be a differentiable function, $\boldsymbol{X} \in \mathbf{SL}_m(\mathbb{C})$ and $z = x + iy$. Corresponding to the four elementary transformations, we define the following four *elementary functions*:

$$
\begin{aligned}
h^{(U)}_{(i,j),\boldsymbol{X}}(x,y) = h^{(U)}_{(i,j),\boldsymbol{X}}(\boldsymbol{\Psi}) &\stackrel{\mathsf{def}}{=} g(\boldsymbol{X}\boldsymbol{U}^{(i,j,\boldsymbol{\Psi})}), \quad \boldsymbol{\Psi} \in \mathbf{SUT}_2(\mathbb{C}),\ z \in \mathbb{C};\\
h^{(L)}_{(i,j),\boldsymbol{X}}(x,y) = h^{(L)}_{(i,j),\boldsymbol{X}}(\boldsymbol{\Psi}) &\stackrel{\mathsf{def}}{=} g(\boldsymbol{X}\boldsymbol{L}^{(i,j,\boldsymbol{\Psi})}), \quad \boldsymbol{\Psi} \in \mathbf{SLT}_2(\mathbb{C}),\ z \in \mathbb{C};\\
h^{(D)}_{(i,j),\boldsymbol{X}}(x,y) = h^{(D)}_{(i,j),\boldsymbol{X}}(\boldsymbol{\Psi}) &\stackrel{\mathsf{def}}{=} g(\boldsymbol{X}\boldsymbol{D}^{(i,j,\boldsymbol{\Psi})}), \quad \boldsymbol{\Psi} \in \mathbf{D}_2(\mathbb{C}),\ z \in \mathbb{C}_*;\\
h^{(Q)}_{(i,j),\boldsymbol{X}}(c,s_1,s_2) = h^{(Q)}_{(i,j),\boldsymbol{X}}(\theta,\phi) &= h^{(Q)}_{(i,j),\boldsymbol{X}}(\boldsymbol{\Psi})\\
&\stackrel{\mathsf{def}}{=} g(\boldsymbol{X}\boldsymbol{Q}^{(i,j,\boldsymbol{\Psi})}), \boldsymbol{\Psi} \in \mathbf{SU}_2(\mathbb{C}),\ (c,s_1,s_2) \in \mathbb{S}_2,\ (\theta,\phi) \in \mathbb{R}^2.
\end{aligned}
\tag{5.1}
$$

In the above last equation, as in [18, 45], we parameterize $\boldsymbol{\Psi} \in \mathbf{SU}_2(\mathbb{C})$ as

$$
\boldsymbol{\Psi} = \boldsymbol{\Psi}(c,s_1,s_2) = \begin{bmatrix} c & -s \\ s^* & c \end{bmatrix} = \begin{bmatrix} c & -(s_1+is_2) \\ s_1 - is_2 & c \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \mathrm{e}^{i\phi} \\ \sin\theta \mathrm{e}^{-i\phi} & \cos\theta \end{bmatrix},
$$

where $(c,s_1,s_2) \in \mathbb{S}_2$ and $(\theta,\phi) \in \mathbb{R}^2$.

Recall that $\boldsymbol{\Lambda}(\boldsymbol{X}) \stackrel{\mathsf{def}}{=} \boldsymbol{X}^{-1}\operatorname{grad} g(\boldsymbol{X}) \in \mathfrak{sl}_m(\mathbb{C})$ in Subsection 3.3, and denote $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}(\boldsymbol{X})$ for simplicity. We now show the relationships between the Riemannian gradients of the four elementary functions defined in (5.1) and the Riemannian gradient of the function g at $\boldsymbol{X} \in \mathbf{SL}_m(\mathbb{C})$. The proof is postponed to Appendix A.

LEMMA 5.1. *The Riemannian gradients of the elementary functions defined in (5.1) at the identity matrix $\boldsymbol{I}_2$ can be expressed as follows:*

$$
(i)\ \operatorname{grad} h^{(Q)}_{(i,j),\boldsymbol{X}}(\boldsymbol{I}_2) = \begin{bmatrix} \frac{i}{2}\Im(\boldsymbol{\Lambda}_{ii}-\boldsymbol{\Lambda}_{jj}) & \frac{1}{2}\Re(\boldsymbol{\Lambda}_{ij}-\boldsymbol{\Lambda}_{ji}) + \frac{i}{2}\Im(\boldsymbol{\Lambda}_{ij}+\boldsymbol{\Lambda}_{ji}) \\ -\frac{1}{2}\Re(\boldsymbol{\Lambda}_{ij}-\boldsymbol{\Lambda}_{ji}) + \frac{i}{2}\Im(\boldsymbol{\Lambda}_{ij}+\boldsymbol{\Lambda}_{ji}) & -\frac{i}{2}\Im(\boldsymbol{\Lambda}_{ii}-\boldsymbol{\Lambda}_{jj}) \end{bmatrix};
$$

$$
(ii)\ \operatorname{grad} h^{(U)}_{(i,j),\boldsymbol{X}}(\boldsymbol{I}_2) = \begin{bmatrix} 0 & \boldsymbol{\Lambda}_{ij} \\ 0 & 0 \end{bmatrix}; \quad (iii)\ \operatorname{grad} h^{(L)}_{(i,j),\boldsymbol{X}}(\boldsymbol{I}_2) = \begin{bmatrix} 0 & 0 \\ \boldsymbol{\Lambda}_{ji} & 0 \end{bmatrix};
$$

$$
(iv)\ \operatorname{grad} h^{(D)}_{(i,j),\boldsymbol{X}}(\boldsymbol{I}_2) = \begin{bmatrix} \Re(\boldsymbol{\Lambda}_{ii}-\boldsymbol{\Lambda}_{jj}) & 0 \\ 0 & \Im(\boldsymbol{\Lambda}_{ii}-\boldsymbol{\Lambda}_{jj}) \end{bmatrix}.
$$

The following lemma can be easily obtained from Lemma 5.1.

LEMMA 5.2. *The partial derivatives of the elementary functions defined in (5.1) satisfy*

$$
\begin{aligned}
&(i)\ \partial h^{(Q)}_{(i,j),\boldsymbol{X}}(\boldsymbol{I}_2) \stackrel{\mathsf{def}}{=} \partial h^{(Q)}_{(i,j),\boldsymbol{X}}(1,0,0) = [0,\ -\Re(\boldsymbol{\Lambda}_{ij}-\boldsymbol{\Lambda}_{ji}),\ -\Im(\boldsymbol{\Lambda}_{ij}+\boldsymbol{\Lambda}_{ji})]^{\mathsf{T}};\\
&(ii)\ \partial h^{(U)}_{(i,j),\boldsymbol{X}}(\boldsymbol{I}_2) \stackrel{\mathsf{def}}{=} \partial h^{(U)}_{(i,j),\boldsymbol{X}}(0,0) = [\Re(\boldsymbol{\Lambda}_{ij}),\ \Im(\boldsymbol{\Lambda}_{ij})]^{\mathsf{T}};\\
&(iii)\ \partial h^{(L)}_{(i,j),\boldsymbol{X}}(\boldsymbol{I}_2) \stackrel{\mathsf{def}}{=} \partial h^{(L)}_{(i,j),\boldsymbol{X}}(0,0) = [\Re(\boldsymbol{\Lambda}_{ji}),\ \Im(\boldsymbol{\Lambda}_{ji})]^{\mathsf{T}};\\
&(iv)\ \partial h^{(D)}_{(i,j),\boldsymbol{X}}(\boldsymbol{I}_2) \stackrel{\mathsf{def}}{=} \partial h^{(D)}_{(i,j),\boldsymbol{X}}(0,0) = [\Re(\boldsymbol{\Lambda}_{ii}-\boldsymbol{\Lambda}_{jj}),\ \Im(\boldsymbol{\Lambda}_{ii}-\boldsymbol{\Lambda}_{jj})]^{\mathsf{T}}.
\end{aligned}
$$

5.2. Three subalgorithms. Let f be the cost function (1.3). Let $\boldsymbol{\omega}_{k-1} = (\boldsymbol{Y}_{k-1}, \boldsymbol{X}_{k-1})$ be the $(k-1)$-th iterate produced by Algorithm 1, and $g : \mathbf{SL}_m(\mathbb{C}) \to \mathbb{R}$ be the restricted function $f_{2,\boldsymbol{Y}_{k-1}}$ defined as in (2.2). Let (i_k, j_k) be a pair of indices satisfying $1 \le i_k < j_k \le m$. For simplicity, we denote

$$
\begin{aligned}
h^{(Q)}_k &= h^{(Q)}_{(i_k,j_k),\boldsymbol{X}_{k-1}}, \quad h^{(U)}_k = h^{(U)}_{(i_k,j_k),\boldsymbol{X}_{k-1}},\\
h^{(L)}_k &= h^{(L)}_{(i_k,j_k),\boldsymbol{X}_{k-1}}, \quad h^{(D)}_k = h^{(D)}_{(i_k,j_k),\boldsymbol{X}_{k-1}}.
\end{aligned}
\tag{5.2}
$$

Based on the three classes of elementary transformations, the subalgorithms to update $\boldsymbol{X}_k$ in Algorithm 1 are summarized in Subalgorithm 1a, Subalgorithm 1b and Subalgorithm 1c, respectively. In these three cases, as in Subsection 2.1, we call Algorithm 1 the *BCD-GLU*, *BCD-GQU* and *BCD-GU* algorithms, respectively.

Subalgorithm 1a: The subalgorithm to update $\boldsymbol{X}_k$ based on GLU class

1: **Input:** Current iterate $\boldsymbol{X}_{k-1}$, a fixed positive constant $0 < \varepsilon < \sqrt{\frac{2}{3m(m-1)}}$.
2: **Output:** New iterate $\boldsymbol{X}_k$.
3: Choose an index pair (i_k, j_k) and an elementary function h_k such that[9]

$$\|\partial h_k(\boldsymbol{I}_2)\| \geq \varepsilon \|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|, \tag{5.3}$$

where $h_k = h_k^{(U)}, h_k^{(L)}$ or $h_k^{(D)}$;
4: Compute $\boldsymbol{\Psi}_k^*$ that minimizes the elementary function h_k, satisfying Update rule 6.4 and 6.5 (will be shown in Section 6);
5: Update $\boldsymbol{X}_k = \boldsymbol{X}_{k-1}\boldsymbol{P}_k$, where $\boldsymbol{P}_k = \boldsymbol{U}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}, \boldsymbol{L}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$ or $\boldsymbol{D}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$.

Subalgorithm 1b: The subalgorithm to update $\boldsymbol{X}_k$ based on GQU class

1: **Input:** Current iterate $\boldsymbol{X}_{k-1}$, a fixed positive constant $0 < \varepsilon < \sqrt{\frac{3-\sqrt{5}}{3m(m-1)}}$.
2: **Output:** New iterate $\boldsymbol{X}_k$.
3: Choose an index pair (i_k, j_k) and an elementary function h_k satisfying the inequality (5.3), where $h_k = h_k^{(Q)}, h_k^{(U)}$ or $h_k^{(D)}$;
4: Compute $\boldsymbol{\Psi}_k^*$ that minimizes the elementary function h_k, satisfying Update rule 6.4, 6.5 and 7.3 (will be shown in Sections 6 and 7);
5: Update $\boldsymbol{X}_k = \boldsymbol{X}_{k-1}\boldsymbol{P}_k$, where $\boldsymbol{P}_k = \boldsymbol{Q}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}, \boldsymbol{U}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$ or $\boldsymbol{D}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$.

Subalgorithm 1c: The subalgorithm to update $\boldsymbol{X}_k$ based on GU class

1: **Input:** Current iterate $\boldsymbol{X}_{k-1}$, a fixed positive constant $0 < \varepsilon < \sqrt{\frac{1}{m(m-1)}}$.
2: **Output:** New iterate $\boldsymbol{X}_k$.
3: Choose an index pair (i_k, j_k) and an elementary function h_k satisfying the inequality (5.3), where $h_k = h_k^{(U)}$ or $h_k^{(D)}$;
4: Compute $\boldsymbol{\Psi}_k^*$ that minimizes the elementary function h_k, satisfying Update rule 6.4 and 6.5 (will be shown in Section 6);
5: Update $\boldsymbol{X}_k = \boldsymbol{X}_{k-1}\boldsymbol{P}_k$, where $\boldsymbol{P}_k = \boldsymbol{U}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$ or $\boldsymbol{D}^{(i_k,j_k,\boldsymbol{\Psi}_k^*)}$.

In the following result, we will show that Subalgorithm 1a and Subalgorithm 1b are both well-defined. The proof is postponed to Appendix A.

PROPOSITION 5.3. *(i) In Subalgorithm 1a, we can always choose an index pair (i_k, j_k) and an elementary function $h_k = h_k^{(U)}, h_k^{(L)}$ or $h_k^{(D)}$ such that the inequality (5.3) is satisfied.*
(ii) In Subalgorithm 1b, we can always choose an index pair (i_k, j_k) and an elementary function $h_k = h_k^{(Q)}, h_k^{(U)}$ or $h_k^{(D)}$ such that the inequality (5.3) is satisfied.

In BCD-GU algorithm, we always choose a starting point $\boldsymbol{X}_0 \in \mathbf{EUT}_m(\mathbb{C})$. Let $\mathfrak{eut}_m(\mathbb{C}) \subseteq \mathbb{C}^{m\times m}$ be the set of upper triangular matrices with the trace equal to 0. Then the tangent space to $\mathbf{EUT}_m(\mathbb{C})$ at a point $\boldsymbol{X} \in \mathbf{EUT}_m(\mathbb{C})$ can be constructed [4, 8] by $\mathbf{T}_{\boldsymbol{X}}\mathbf{EUT}_m(\mathbb{C}) = \{\boldsymbol{X}\Omega, \Omega \in \mathfrak{eut}_m(\mathbb{C})\}$, which is useful to the proof of the following result. The proof is postponed to Appendix A.

PROPOSITION 5.4. *In Subalgorithm 1c, we can always choose an index pair (i_k, j_k) and an elementary function $h_k = h_k^{(U)}$ or $h_k^{(D)}$ such that the inequality (5.3) is satisfied.*

6. Plane triangular and diagonal transformations for JADM problem. Let f be the cost function (1.3). Let $\boldsymbol{\omega}_{k-1} = (\boldsymbol{Y}_{k-1}, \boldsymbol{X}_{k-1})$ and $g : \mathbf{SL}_m(\mathbb{C}) \to \mathbb{R}$ be the restricted function $f_{2,\boldsymbol{Y}_{k-1}}$ as in Subsection 5.2. Denote $\boldsymbol{B}^{(\ell)} = \boldsymbol{Y}_{k-1}^{\blacklozenge}\boldsymbol{A}^{(\ell)}\boldsymbol{Y}_{k-1}$ for $1 \le \ell \le \mathrm{L}$, where $(\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{T}}$ or $(\cdot)^{\mathsf{H}}$. Then g can be expressed as

$$g : \mathbf{SL}_m(\mathbb{C}) \to \mathbb{R}^+, \quad \boldsymbol{X} \mapsto \sum_{\ell=1}^{\mathrm{L}} \|\operatorname{offdiag}\{\boldsymbol{W}^{(\ell)}\}\|^2, \tag{6.1}$$

where $\boldsymbol{W}^{(\ell)} = \boldsymbol{X}^{\blacklozenge}\boldsymbol{B}^{(\ell)}\boldsymbol{X}$ for $1 \le \ell \le \mathrm{L}$. In this section, we will first calculate the Riemannian gradient of g in (6.1), and the partial derivatives of elementary functions $h_k^{(U)}$, $h_k^{(L)}$ and $h_k^{(D)}$ in (5.2). Then, we will prove that inequalities (3.11) and (3.12) are both satisfied in the plane triangular and diagonal transformations.

6.1. Riemannian gradient. Let g and $\boldsymbol{W}^{(\ell)}$ be as in (6.1). Then, by equations (4.2) and (3.7), we have the Euclidean gradient and Riemannian gradient of g at $\boldsymbol{X} \in \mathbf{SL}_m(\mathbb{C})$ as follows:

$$\nabla g(\boldsymbol{X}) = 2(\boldsymbol{X}^{\mathsf{H}})^{-1}\sum_{\ell=1}^{\mathrm{L}} \Upsilon(\boldsymbol{W}^{(\ell)}), \tag{6.2}$$

$$\operatorname{grad} g(\boldsymbol{X}) = 2\boldsymbol{X}\sum_{\ell=1}^{\mathrm{L}} \left(\Upsilon(\boldsymbol{W}^{(\ell)}) - \frac{\operatorname{tr}(\Upsilon(\boldsymbol{W}^{(\ell)}))}{n}\boldsymbol{I}_n\right), \tag{6.3}$$

where $\Upsilon(\boldsymbol{W}^{(\ell)})$ is defined as in equation (4.3).

Remark 6.1. In the real case, the Euclidean gradient in (6.2) was earlier derived in [4, Eq. (6.3)] and [10, Section 2.3]. In this paper, we extend it to problem (6.1) in the complex case and calculate the Riemannian gradient (6.3) as well.

6.2. Elementary functions. Let $\boldsymbol{W}^{(\ell)} = \boldsymbol{X}_{k-1}^{\blacklozenge}\boldsymbol{B}^{(\ell)}\boldsymbol{X}_{k-1}$ for $1 \le \ell \le \mathrm{L}$. Let

$$\varrho \overset{\mathsf{def}}{=} \begin{cases} 1, & \text{if } (\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{H}}; \\ -1, & \text{if } (\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{T}}. \end{cases} \tag{6.4}$$

Denote $(i, j) = (i_k, j_k)$ for simplicity. Now we use the following notations:

- $\alpha_1 \overset{\mathsf{def}}{=} \sum_{\ell=1}^{\mathrm{L}}\sum_{p\neq j}\left(|\mathrm{W}_{ip}^{(\ell)}|^2 + |\mathrm{W}_{pi}^{(\ell)}|^2\right),$

  $\alpha_2 \overset{\mathsf{def}}{=} \sum_{\ell=1}^{\mathrm{L}}\sum_{p\neq j}\left(\mathrm{W}_{ip}^{(\ell,\Re)}\mathrm{W}_{jp}^{(\ell,\Re)} + \mathrm{W}_{ip}^{(\ell,\Im)}\mathrm{W}_{jp}^{(\ell,\Im)} + \mathrm{W}_{pi}^{(\ell,\Re)}\mathrm{W}_{pj}^{(\ell,\Re)} + \mathrm{W}_{pi}^{(\ell,\Im)}\mathrm{W}_{pj}^{(\ell,\Im)}\right),$

  $\alpha_3 \overset{\mathsf{def}}{=} \sum_{\ell=1}^{\mathrm{L}}\sum_{p\neq j}\left(\varrho\left(\mathrm{W}_{ip}^{(\ell,\Im)}\mathrm{W}_{jp}^{(\ell,\Re)} - \mathrm{W}_{ip}^{(\ell,\Re)}\mathrm{W}_{jp}^{(\ell,\Im)}\right) + \mathrm{W}_{pi}^{(\ell,\Re)}\mathrm{W}_{pj}^{(\ell,\Im)} - \mathrm{W}_{pi}^{(\ell,\Im)}\mathrm{W}_{pj}^{(\ell,\Re)}\right).$

- $\beta_1 \stackrel{\text{def}}{=} \sum_{\ell=1}^{\mathrm{L}} \sum_{p\neq i} \left(|\mathrm{W}_{jp}^{(\ell)}|^2 + |\mathrm{W}_{pj}^{(\ell)}|^2 \right),$

  $\beta_2 \stackrel{\text{def}}{=} \sum_{\ell=1}^{\mathrm{L}} \sum_{p\neq i} \left(\mathrm{W}_{ip}^{(\ell,\Re)} \mathrm{W}_{jp}^{(\ell,\Re)} + \mathrm{W}_{ip}^{(\ell,\Im)} \mathrm{W}_{jp}^{(\ell,\Im)} + \mathrm{W}_{pi}^{(\ell,\Re)} \mathrm{W}_{pj}^{(\ell,\Re)} + \mathrm{W}_{pi}^{(\ell,\Im)} \mathrm{W}_{pj}^{(\ell,\Im)} \right),$

  $\beta_3 \stackrel{\text{def}}{=} \sum_{\ell=1}^{\mathrm{L}} \sum_{p\neq i} \left(\varrho \left(\mathrm{W}_{ip}^{(\ell,\Re)} \mathrm{W}_{jp}^{(\ell,\Im)} - \mathrm{W}_{ip}^{(\ell,\Im)} \mathrm{W}_{jp}^{(\ell,\Re)} \right) + \mathrm{W}_{pi}^{(\ell,\Im)} \mathrm{W}_{pj}^{(\ell,\Re)} - \mathrm{W}_{pi}^{(\ell,\Re)} \mathrm{W}_{pj}^{(\ell,\Im)} \right).$

- $\gamma_1 \stackrel{\text{def}}{=} \sum_{\ell=1}^{\mathrm{L}} \sum_{p\neq i,j} \left(|\mathrm{W}_{ip}^{(\ell)}|^2 + |\mathrm{W}_{pi}^{(\ell)}|^2 \right), \quad \gamma_2 \stackrel{\text{def}}{=} \sum_{\ell=1}^{\mathrm{L}} \sum_{p\neq i,j} \left(|\mathrm{W}_{jp}^{(\ell)}|^2 + |\mathrm{W}_{pj}^{(\ell)}|^2 \right).$

Then we can get the following results by direct calculations.

LEMMA 6.2. *Let the function g be as in (6.1). Then*
(i) the elementary function $h_k^{(U)}$ in (5.2) and its optimal solution (x_k^*, y_k^*) satisfy

$$\begin{aligned} h_k^{(U)}(x,y) - h_k^{(U)}(0,0) &= \alpha_1 x^2 + 2\alpha_2 x + \alpha_1 y^2 + 2\alpha_3 y, \\ h_k^{(U)}(x_k^*, y_k^*) - h_k^{(U)}(0,0) &= -\frac{1}{\alpha_1} \left(\alpha_2^2 + \alpha_3^2 \right), \\ \partial h_k^{(U)}(0,0) &= 2[\alpha_2,\ \alpha_3]^{\mathsf{T}}. \end{aligned} \tag{6.5}$$

(ii) the elementary function $h_k^{(L)}$ in (5.2) and its optimal solution (x_k^*, y_k^*) satisfy

$$\begin{aligned} h_k^{(L)}(x,y) - h_k^{(L)}(0,0) &= \beta_1 x^2 + 2\beta_2 x + \beta_1 y^2 + 2\beta_3 y, \\ h_k^{(L)}(x_k^*, y_k^*) - h_k^{(L)}(0,0) &= -\frac{1}{\beta_1} \left(\beta_2^2 + \beta_3^2 \right), \\ \partial h_k^{(L)}(0,0) &= 2[\beta_2,\ \beta_3]^{\mathsf{T}}. \end{aligned}$$

(iii) the elementary function $h_k^{(D)}$ in (5.2) and its optimal solution (x_k^*, y_k^*) satisfy

$$\begin{aligned} h_k^{(D)}(x,y) - h_k^{(D)}(1,0) &= \gamma_1 (x^2 + y^2) + \gamma_2 \frac{1}{x^2 + y^2} - \gamma_1 - \gamma_2, \\ h_k^{(D)}(x_k^*, y_k^*) - h_k^{(D)}(1,0) &= -\left(\sqrt{\gamma_1} - \sqrt{\gamma_2} \right)^2, \\ \partial h_k^{(D)}(1,0) &= 2[\gamma_1 - \gamma_2,\ 0]^{\mathsf{T}}. \end{aligned}$$

Remark 6.3. In the real case, the solution x_k^* in (6.5) was earlier derived in [5, Eq. (7)]. In the complex case, the solution $z_k^* = x_k^* + i y_k^*$ in (6.5) was earlier derived in [46, Eq. (8)].

Update rule 6.4. In Algorithm 1 for cost function (1.3), when the elementary function $h_k = h_k^{(U)}$, we see that $x_k^* = 0$ if $\alpha_1 \neq 0$ and $\alpha_2 = 0$. It is not possible that $\alpha_1 = 0$ and $\alpha_2 \neq 0$. If $\alpha_1 = \alpha_2 = 0$, we set $x_k^* = 0$. In the case of $h_k = h_k^{(L)}$, we make the similar update rules for the value of y_k^*.

Update rule 6.5. Let $0 < \varsigma_D < \frac{1}{4}$ be a small positive constant. In Algorithm 1 for cost function (1.3), if $h_k = h_k^{(D)}$, we always set $y_k^* = 0$. Moreover, we determine x_k^* based on the following rules.

- If $\gamma_1 = \gamma_2 = 0$, we set $x_k^* = 0$.

- Let $\varpi \overset{\text{def}}{=} \frac{\gamma_2}{\gamma_1}$. If $\varpi \in [0, \varsigma_D)$, we set $x_k^* = \frac{1}{2}$. If $\varpi \in (\frac{1}{\varsigma_D}, +\infty]$, we set $x_k^* = 2$.
- Otherwise, if $\varpi \in [\varsigma_D, \frac{1}{\varsigma_D}]$, we set $x_k^* = \sqrt[4]{\varpi}$, which is the minimum point.

6.3. Inequalities for global convergence. It will be seen that $f(\boldsymbol{\omega}_k) \leq f(\boldsymbol{\omega}_{k-1})$ always holds in Algorithm 1. We denote $\mathrm{M}_0 \overset{\text{def}}{=} f(\boldsymbol{\omega}_0)$ in Algorithm 1 for cost function (1.3). Then we have that $\gamma_1 + \gamma_2 \leq \mathrm{M}_0 = f(\boldsymbol{\omega}_0)$. In the following result, we will show an inequality, which is helpful to establish inequality (3.11) when elementary function $h_k = h_k^{(D)}$. The proof is postponed to Appendix B.

LEMMA 6.6. *In Algorithm 1 for cost function (1.3), there exists $\iota_D > 0$ such that*

$$g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) \geq \iota_D \|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\| \|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\|, \tag{6.6}$$

whenever the elementary function $h_k = h_k^{(D)}$.

Note that

$$\|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\| \leq \|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\| \|\boldsymbol{X}_{k-1}\|, \tag{6.7}$$

$$\|\operatorname{grad} g(\boldsymbol{X}_{k-1})\| \leq \|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\| \|\boldsymbol{X}_{k-1}\|. \tag{6.8}$$

Let ι_D be as in (6.6), and M_ω be as in the condition (1.5). Let $\sigma_D = \iota_D / \mathrm{M}_\omega^2 > 0$. Then the next result follows directly from Lemma 6.6, inequalities (6.7) and (6.8).

COROLLARY 6.7. *In Algorithm 1 for cost function (1.3), if the iterates remain bounded, i.e., the condition (1.5) is satisfied, then*

$$g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) \geq \sigma_D \|\operatorname{grad} g(\boldsymbol{X}_{k-1})\| \|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|, \tag{6.9}$$

whenever the elementary function $h_k = h_k^{(D)}$.

As for inequality (6.6), we now show a similar result for the cases of elementary functions $h_k^{(L)}$ and $h_k^{(U)}$. The proof is also postponed to Appendix B.

LEMMA 6.8. *In Algorithm 1 for cost function (1.3), there exists $\iota_{LU} > 0$ such that*

$$g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) \geq \iota_{LU} \|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\| \|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\|, \tag{6.10}$$

whenever the elementary function $h_k = h_k^{(L)}$ or $h_k^{(U)}$.

Let ι_{LU} be as in (6.10) and $\sigma_{LU} = \iota_{LU} / \mathrm{M}_\omega^2 > 0$. Similar as for Corollary 6.7, the next result follows directly from Lemma 6.8, inequalities (6.7) and (6.8).

COROLLARY 6.9. *In Algorithm 1 for cost function (1.3), if the iterates remain bounded, i.e., the condition (1.5) is satisfied, then*

$$g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) \geq \sigma_{LU} \|\operatorname{grad} g(\boldsymbol{X}_{k-1})\| \|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|,$$

whenever the elementary function $h_k = h_k^{(L)}$ or $h_k^{(U)}$.

6.4. Inequalities for weak convergence. In this subsection, we show an inequality, which will be helpful to establish inequality (3.12). The proof is postponed to Appendix B.

LEMMA 6.10. *In Algorithm 1 for cost function (1.3), if the iterates remain bounded, i.e., the condition (1.5) is satisfied, then there exists $\kappa > 0$ such that*

$$\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\| \geq \kappa \|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|,$$

whenever the elementary function $h_k = h_k^{(D)}$, $h_k^{(L)}$ or $h_k^{(U)}$.

By Lemma 6.6, Lemma 6.8 and Lemma 6.10, we can easily get the following results by setting $\eta_D = (\kappa \iota_D)/\mathrm{M}_\omega^2$ and $\eta_{LU} = (\kappa \iota_{LU})/\mathrm{M}_\omega^2$.

COROLLARY 6.11. *In Algorithm 1 for cost function (1.3), if the iterates remain bounded, i.e., the condition (1.5) is satisfied, then*

$$g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) \geq \min(\eta_D, \eta_{LU}) \| \operatorname{grad} g(\boldsymbol{X}_{k-1}) \|^2, \tag{6.11}$$

whenever the elementary function $h_k = h_k^{(D)}$, $h_k^{(L)}$ or $h_k^{(U)}$.

7. Givens plane transformations for JADM problem. Let the function g be as in (6.1). Let $\boldsymbol{W}^{(\ell)} = \boldsymbol{X}_{k-1}^{\blacklozenge} \boldsymbol{B}^{(\ell)} \boldsymbol{X}_{k-1}$ for $1 \leq \ell \leq \mathrm{L}$ as in Subsection 6.2, where $(\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{T}}$ or $(\cdot)^{\mathsf{H}}$. Let ϱ be as in (6.4). Denote $(i,j) = (i_k, j_k)$ for simplicity. Define

$$\boldsymbol{\Gamma}^{(i,j,\boldsymbol{X}_{k-1})} \stackrel{\mathsf{def}}{=} \frac{\varrho}{2} \sum_{\ell=1}^{L} \Re \left(\boldsymbol{z}_{i,j}(\boldsymbol{W}^{(\ell)}) \boldsymbol{z}_{i,j}^{\mathsf{H}}(\boldsymbol{W}^{(\ell)}) \right) \in \mathbb{R}^{3\times 3}, \tag{7.1}$$

where

$$\boldsymbol{z}_{i,j}(\boldsymbol{W}) \stackrel{\mathsf{def}}{=} \begin{cases} \left[\mathrm{W}_{jj} - \mathrm{W}_{ii}, \quad \mathrm{W}_{ij} + \mathrm{W}_{ji}, \quad -i(\mathrm{W}_{ij} - \mathrm{W}_{ji}) \right]^{\mathsf{T}}, & \text{if } (\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{H}}; \\ \left[\mathrm{W}_{ij} + \mathrm{W}_{ji}, \quad \mathrm{W}_{ii} - \mathrm{W}_{jj}, \quad i(\mathrm{W}_{ii} + \mathrm{W}_{jj}) \right]^{\mathsf{T}}, & \text{if } (\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{T}}. \end{cases}$$

Denote

$$c_0 \stackrel{\mathsf{def}}{=} \begin{cases} \frac{1}{2} \sum_{\ell=1}^{\mathrm{L}} \left| \mathrm{W}_{jj}^{(\ell)} - \mathrm{W}_{ii}^{(\ell)} \right|^2, & \text{if } (\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{H}}; \\ -\frac{1}{2} \sum_{\ell=1}^{\mathrm{L}} \left| \mathrm{W}_{ij}^{(\ell)} + \mathrm{W}_{ji}^{(\ell)} \right|^2, & \text{if } (\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{T}}. \end{cases}$$

7.1. Elementary function. As in [45, Eq. (4.4)], we denote the unit vector

$$\boldsymbol{r} \stackrel{\mathsf{def}}{=} \left[2c^2 - 1, \ -2cs_1, \ -2cs_2 \right]^{\mathsf{T}} = \left[\cos 2\theta, \ -\sin 2\theta \cos\phi, \ -\sin 2\theta \sin\phi \right]^{\mathsf{T}}, \tag{7.2}$$

where $c \in \mathbb{R}^+$, $s = s_1 + i s_2 \in \mathbb{C}$, $c^2 + |s|^2 = 1$, and $\theta, \phi \in \mathbb{R}$ are two angles. Then we can get the following result[10] by direct calculations.

LEMMA 7.1. *In Algorithm 1 for cost function (1.3), the elementary function $h_k^{(Q)}$ satisfies*

$$h_k^{(Q)}(c, s_1, s_2) - h_k^{(Q)}(1, 0, 0) = -\left(\boldsymbol{r}^{\mathsf{T}} \boldsymbol{\Gamma}^{(i,j,\boldsymbol{X}_{k-1})} \boldsymbol{r} - c_0 \right), \tag{7.3}$$

where $\boldsymbol{\Gamma}^{(i,j,\boldsymbol{X}_{k-1})} \in \mathbb{R}^{3\times 3}$ is as in equation (7.1).

[10] In the $(\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{H}}$ case, this expression was first formulated in [13].

Denote $\mathbf{\Gamma} = \mathbf{\Gamma}^{(i,j,\boldsymbol{X}_{k-1})}$ for simplicity. It follows by equations (7.2) and (7.3) that

$$h_k^{(Q)}(c, s_1, s_2) - h_k^{(Q)}(1, 0, 0) = -\left(q(\theta, \phi) - c_0\right), \tag{7.4}$$

where

$$q(\theta,\phi) \stackrel{\mathsf{def}}{=} \frac{1}{2}\left(\Gamma_{11} - \Gamma_{22}\cos^2\phi - \Gamma_{33}\sin^2\phi - \Gamma_{23}\sin(2\phi)\right)\cos(4\theta)$$
$$-\left(\Gamma_{12}\cos\phi + \Gamma_{13}\sin\phi\right)\sin(4\theta) + \frac{1}{2}\left(\Gamma_{11} + \Gamma_{22}\cos^2\phi + \Gamma_{33}\sin^2\phi + \Gamma_{23}\sin(2\phi)\right). \tag{7.5}$$

Note that, by Lemma 7.1 and equation (7.2), we have

$$\partial h_k^{(Q)}(\boldsymbol{I}_2) = -4[0,\ \Gamma_{12},\ \Gamma_{13}]^{\mathsf{T}}. \tag{7.6}$$

Remark 7.2. By equation (7.4), we see that $h_k^{(Q)}(\theta + \pi/2, \phi) = h_k^{(Q)}(\theta, \phi)$ for any $\theta, \phi \in \mathbb{R}$. Therefore, we can always choose $\theta_* \in [-\pi/4, \pi/4]$.

Update rule 7.3. In Algorithm 1 for cost function (1.3), we set a positive constant $\varsigma_Q > 0$. If $h_k = h_k^{(Q)}$, we find the eigenvector $\boldsymbol{u}$ of $\mathbf{\Gamma}$ corresponding to the largest eigenvalue. Define two vectors $\boldsymbol{v}_{i,j} \stackrel{\mathsf{def}}{=} [\Gamma_{12},\ \Gamma_{13}]^{\mathsf{T}} \in \mathbb{R}^2$ and $\boldsymbol{w}_{i,j} \stackrel{\mathsf{def}}{=} [u_2,\ u_3]^{\mathsf{T}} \in \mathbb{R}^2$.

- If it holds that
$$|\langle \boldsymbol{v}_{i,j}, \boldsymbol{w}_{i,j}\rangle| \geq \varsigma_Q \|\boldsymbol{v}_{i,j}\|\|\boldsymbol{w}_{i,j}\|, \tag{7.7}$$
then we find ϕ_* and θ_* by setting $\boldsymbol{r} = \boldsymbol{u}$, and $\mathbf{\Psi}_k^* = \mathbf{\Psi}(\theta_*, \phi_*)$;
- Otherwise, we set $[\cos\phi_*,\ \sin\phi_*]^{\mathsf{T}} = \boldsymbol{v}_{i,j}/\|\boldsymbol{v}_{i,j}\|$, and then calculate θ_*, which maximizes the restricted function $q(\theta, \phi_*)$.

7.2. Inequalities for global convergence. We first present a lemma, which will help us to prove Lemma 7.5.

LEMMA 7.4. *Let $\alpha, \beta \in \mathbb{R}$ be two constants. For $\theta \in [-\frac{\pi}{4}, \frac{\pi}{4}]$, we define a function $p(\theta) \stackrel{\mathsf{def}}{=} \alpha\cos(4\theta) + \beta\sin(4\theta)$. If $\theta_* \in [-\frac{\pi}{4}, \frac{\pi}{4}]$ satisfies $p(\theta_*) = \max p(\theta)$, then we have*

$$p(\theta_*) - p(0) \geq 2\sqrt{2}|\beta|\left|\sin(\frac{\theta_*}{2})\right|.$$

LEMMA 7.5. *Let the function $q(\theta, \phi)$ be as in equation (7.5). Suppose that ϕ_* and θ_* are determined as in Update rule 7.3. Then we have*

$$q(\theta_*, \phi_*) - q(0, 0) \geq 2\sqrt{2}\varsigma_Q\left|\sin(\frac{\theta_*}{2})\right|\|\boldsymbol{v}_{i,j}\|,$$

where ς_Q is the positive constant defined in Update rule 7.3.

Proof. By Update rule 7.3, we see that

$$|\langle \boldsymbol{v}_{i,j}, [\cos\phi_*\ \ \sin\phi_*]^{\mathsf{T}}\rangle| \geq \varsigma_Q\|\boldsymbol{v}_{i,j}\| \tag{7.8}$$

always holds. By Lemma 7.4 and the above inequality (7.8), we get that

$$q(\theta_*, \phi_*) - q(0, 0) = q(\theta_*, \phi_*) - q(0, \phi_*) \geq 2\sqrt{2}\left|\sin(\frac{\theta_*}{2})\right||\Gamma_{12}\cos\phi_* + \Gamma_{13}\sin\phi_*|$$
$$\geq 2\sqrt{2}\varsigma_Q\left|\sin(\frac{\theta_*}{2})\right|\|\boldsymbol{v}_{i,j}\|.$$

The proof is complete. ∎

As for inequality (6.6), we now show a similar result for the case of elementary functions $h_k^{(Q)}$, which will be helpful to establish inequality (3.11).

LEMMA 7.6. *In Algorithm 1 for cost function (1.3), there exists $\iota_Q > 0$ such that*

$$g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) \geq \iota_Q \|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\| \|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\|, \tag{7.9}$$

whenever the elementary function $h_k = h_k^{(Q)}$.

Proof. We only prove the case $(\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{H}}$, the other case being similar. By Lemma 7.5 and equation (7.6), we get that

$$\begin{aligned} h_k^{(Q)}(0,0) - h_k^{(Q)}(\theta_*, \phi_*) &\geq 2\sqrt{2}\varsigma_Q \left|\sin(\frac{\theta_*}{2})\right| \|\boldsymbol{v}_{i,j}\| = \frac{\varsigma_Q}{4} 2\sqrt{2} \left|\sin(\frac{\theta_*}{2})\right| \|\partial h_k^{(Q)}(\boldsymbol{I}_2)\| \\ &\geq \frac{\varsigma_Q \varepsilon}{4} \|\boldsymbol{Q}^{(i,j,\boldsymbol{\Psi}_k^*)} - \boldsymbol{I}_m\| \|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|. \end{aligned}$$

We can set $\iota_Q = (\varsigma_Q \varepsilon)/4$. The proof is complete. ∎

Let ι_Q be as in (7.9) and M_ω be as in condition (1.5). Let $\sigma_Q = \iota_Q / \mathrm{M}_\omega^2 > 0$. As for Corollary 6.7, the next result follows directly from Lemma 7.6, inequalities (6.7) and (6.8).

COROLLARY 7.7. *In Algorithm 1 for cost function (1.3), if the iterates remain bounded, i.e., the condition (1.5) is satisfied, then*

$$g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) \geq \sigma_Q \|\operatorname{grad} g(\boldsymbol{X}_{k-1})\| \|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|,$$

whenever the elementary function $h_k = h_k^{(Q)}$.

7.3. Inequalities for weak convergence. If the condition (1.5) is satisfied, it is easy to see that there exists a positive constant $\mathrm{M}_\Gamma > 0$ such that $\|\boldsymbol{\Gamma}^{(i,j,\boldsymbol{X}_{k-1})}\| \leq \mathrm{M}_\Gamma$ always holds in Algorithm 1. In this subsection, we first show an inequality, which will be helpful to establish inequality (3.12).

LEMMA 7.8. *In Algorithm 1 for cost function (1.3), if the iterates remain bounded, i.e., the condition (1.5) is satisfied, then there exists $\kappa_Q > 0$ such that*

$$\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\| \geq \kappa_Q \|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|, \tag{7.10}$$

whenever the elementary function $h_k = h_k^{(Q)}$.

Proof. If $\boldsymbol{v}_{i,j}$ and $\boldsymbol{w}_{i,j}$ satisfy inequality (7.7), then inequality (7.10) can be proved by a similar method as for [45, Lemma 7.2]. Otherwise, if we set $[\cos\phi_* \ \ \sin\phi_*]^{\mathsf{T}} = \boldsymbol{v}_{i,j}/\|\boldsymbol{v}_{i,j}\|$ and find θ_* based on ϕ_*, then

$$\begin{aligned} |\sin(4\theta_*)| &= \frac{|\Gamma_{12}\cos\phi_* + \Gamma_{13}\sin\phi_*|}{\sqrt{(\Gamma_{12}\cos\phi_* + \Gamma_{13}\sin\phi_*)^2 + \frac{1}{4}\left(\Gamma_{11} - \Gamma_{22}\cos^2\phi_* - \Gamma_{33}\sin^2\phi_* - \Gamma_{23}\sin(2\phi_*)\right)^2}} \\ &\geq \frac{\sqrt{\Gamma_{12}^2 + \Gamma_{13}^2}}{2\sqrt{5}\mathrm{M}_\Gamma} = \frac{\|\partial h_k^{(Q)}(\boldsymbol{I}_2)\|}{8\sqrt{5}\mathrm{M}_\Gamma} \geq \frac{\varepsilon}{8\sqrt{5}\mathrm{M}_\Gamma} \|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|. \end{aligned}$$

Note that

$$\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\| = 2\sqrt{2}\left|\sin(\frac{\theta_*}{2})\right| \geq \frac{\sqrt{2}}{4}|\sin(4\theta_*)|.$$

We only need to set $\kappa_Q = \frac{\sqrt{2}\varepsilon}{32\sqrt{5}\mathrm{M}_\Gamma}$ in this case. The proof is complete. ∎

By Lemma 7.6, Lemma 7.8, inequalities (6.7) and (6.8), we can now easily get the following result by setting $\eta_Q = (\kappa_Q \iota_Q)/\mathrm{M}_\omega^2$.

COROLLARY 7.9. *In Algorithm 1 for cost function (1.3), if the iterates remain bounded, i.e., the condition (1.5) is satisfied, then*

$$g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) \geq \eta_Q \| \operatorname{grad} g(\boldsymbol{X}_{k-1}) \|^2,$$

whenever the elementary function $h_k = h_k^{(Q)}$.

8. Convergence analysis. In this section, based on the inequalities derived in Sections 6 and 7, we will prove our main results about the global and weak convergence of the BCD-G and Jacobi-G algorithms formulated in Subsection 2.1.

8.1. Convergence analysis of BCD-G algorithms. We now prove the following results about the global and weak convergence of BCD-G algorithms.

THEOREM 8.1. *In BCD-GLU, BCD-GQU and BCD-GU algorithms for cost function (1.3), if the iterates remain bounded, i.e., the condition (1.5) is satisfied, then the iterates* $\{\boldsymbol{\omega}_k\}_{k\geq 1}$ *converge to a point* $\boldsymbol{\omega}_*$.

Proof. We first prove the case of BCD-GLU algorithm. By Corollaries 6.7 and 6.9, we see that

$$g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) \geq \min(\sigma_D, \sigma_{LU}) \| \operatorname{grad} g(\boldsymbol{X}_{k-1}) \| \| \boldsymbol{X}_k - \boldsymbol{X}_{k-1} \|, \tag{8.1}$$

whenever the elementary function $h_k = h_k^{(D)}, h_k^{(U)}$ or $h_k^{(L)}$. By the above inequality (8.1) and Lemma 4.3, we have that

$$f(\boldsymbol{\omega}_{k-1}) - f(\boldsymbol{\omega}_k) \geq \min(\sigma_D, \sigma_{LU}, \sigma_p) \| \operatorname{grad} f_{t_k}(\boldsymbol{\omega}_{k-1}) \| \| \boldsymbol{\omega}_k - \boldsymbol{\omega}_{k-1} \| \tag{8.2}$$

always holds in BCD-GLU algorithm, which is the inequality (3.11) in Lemma 3.4, if we set $\sigma = \min(\sigma_D, \sigma_{LU}, \sigma_p)$. Therefore, if $\boldsymbol{\omega}_*$ is an accumulation point of the iterates $\{\boldsymbol{\omega}_k\}_{k\geq 1}$ produced by BCD-GLU algorithms, it is the limit point. Note that the iterates $\{\boldsymbol{\omega}_k\}_{k\geq 1}$ remain bounded by condition (1.5). There exists an accumulation point $\boldsymbol{\omega}_*$ such that the iterates $\{\boldsymbol{\omega}_k\}_{k\geq 1}$ converge to $\boldsymbol{\omega}_*$. For other two cases of BCD-GQU and BCD-GU algorithms, by Lemma 4.3, Corollary 6.7, Corollary 6.9 and Corollary 7.7, we can similarly prove that the inequality (3.11) is always satisfied, if we set $\sigma = \min(\sigma_D, \sigma_{LU}, \sigma_Q, \sigma_p)$ and $\min(\sigma_D, \sigma_{LU}, \sigma_p)$, respectively. The proof is complete. ∎

To help the readers better understand the proof of Theorem 8.1, we now summarize in Figure 1 the proof structure of Theorem 8.1 for BCD-GLU algorithm. Other two cases of BCD-GQU and BCD-GU algorithms are similar.

THEOREM 8.2. *In BCD-GLU, BCD-GQU and BCD-GU algorithms for cost function (1.3), if the iterates remain bounded, i.e., the condition (1.5) is satisfied, and* $\boldsymbol{\omega}_*$ *is an accumulation point of the iterates* $\{\boldsymbol{\omega}_k\}_{k\geq 1}$, *then* $\boldsymbol{\omega}_*$ *is a stationary point of the cost function (1.3).*

Proof. We first prove the case of BCD-GLU algorithm. By Corollary 6.11, we see that

$$g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) \geq \min(\eta_D, \eta_{LU}) \| \operatorname{grad} g(\boldsymbol{X}_{k-1}) \|^2, \tag{8.3}$$

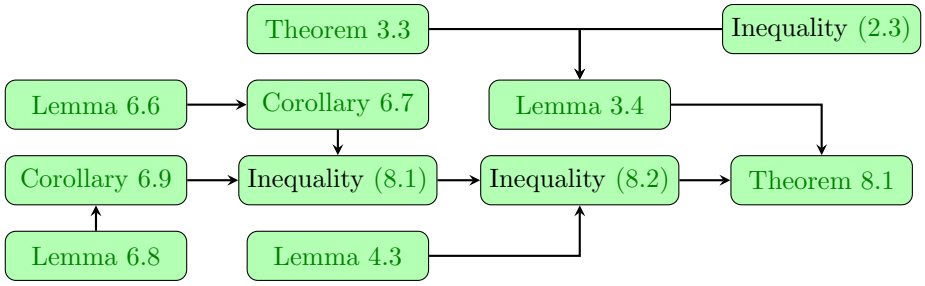

FIG. 1. *Proof structure of Theorem 8.1 for BCD-GLU algorithm.*

whenever the elementary function $h_k = h_k^{(D)}, h_k^{(U)}$ or $h_k^{(L)}$. By the above inequality (8.3) and Lemma 4.4, we have that

$$f(\boldsymbol{\omega}_{k-1}) - f(\boldsymbol{\omega}_k) \geq \min(\eta_D, \eta_{LU}, \eta_p) \| \operatorname{grad} f_{t_k}(\boldsymbol{\omega}_{k-1}) \|^2 \tag{8.4}$$

always holds in BCD-GLU algorithm, which is the inequality (3.12) in Lemma 3.5, if we set $\eta = \min(\eta_D, \eta_{LU}, \eta_p)$. Therefore, if $\boldsymbol{\omega}_*$ is an accumulation point of the iterates $\{\boldsymbol{\omega}_k\}_{k\geq 1}$ produced by BCD-GLU algorithm, then $\boldsymbol{\omega}_*$ is a stationary point. For other two cases of BCD-GQU and BCD-GU algorithms, by Corollary 6.11 and Corollary 7.9, we can similarly prove that the inequality (3.12) is always satisfied, if we set $\eta = \min(\eta_D, \eta_{LU}, \eta_Q, \eta_p)$ and $\min(\eta_D, \eta_{LU}, \eta_p)$, respectively. The proof is complete. ∎

To help the readers better understand the proof of Theorem 8.2, we now summarize in Figure 2 the proof structure of Theorem 8.2 for BCD-GLU algorithm. Other two cases of BCD-GQU and BCD-GU algorithms are similar.

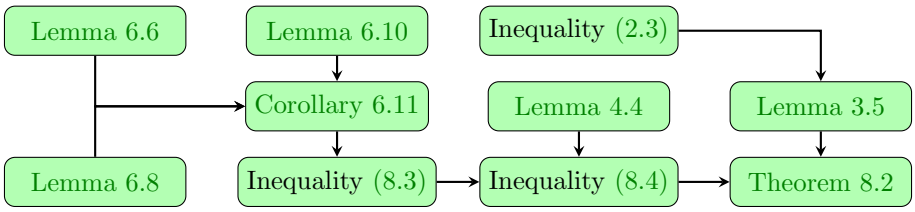

FIG. 2. *Proof structure of Theorem 8.2 for BCD-GLU algorithm.*

8.2. Convergence analysis of Jacobi-G algorithms. Similar as in Subsection 8.1, we have the following results about the global and weak convergence of Jacobi-G algorithms. We omit the detailed proofs here.

THEOREM 8.3. *In Jacobi-GLU and Jacobi-GQU algorithms for cost function (1.4), if the iterates remain bounded, i.e., the condition (1.6) is satisfied, then the iterates* $\{\boldsymbol{X}_k\}_{k\geq 1}$ *converge to a point* $\boldsymbol{X}_*$.

THEOREM 8.4. *In Jacobi-GLU and Jacobi-GQU algorithms for cost function (1.4), if the iterates remain bounded, i.e., the condition (1.6) is satisfied, and* $\boldsymbol{X}_*$ *is an accumulation point of the iterates* $\{\boldsymbol{X}_k\}_{k\geq 1}$, *then* $\boldsymbol{X}_*$ *is a stationary point of the cost function (1.4).*

Remark 8.5. We propose two natural variants of Jacobi-GLU and Jacobi-GQU algorithms, which will be called *Jacobi-GLU-M* and *Jacobi-GQU-M* algorithms, respectively. In these two algorithms, in each iteration, among all the index pairs (i_k, j_k) and elementary functions h_k satisfying inequality (5.3), we choose (i_k, j_k) and h_k such that the cost function obtains the largest reduction. It is clear that Theorems 8.3 and 8.4 also apply to these two new variants.

Remark 8.6. In Jacobi-GLU and Jacobi-GQU algorithms, a more natural way of choosing the index pair (i_k, j_k) is according to a *cyclic* ordering. In fact, this cyclic way has often been used in the literature [36, 42, 46]. In this case, we call them the *Jacobi-CLU* and *Jacobi-CQU* algorithms, respectively.

9. Numerical experiments. In the BCD-G and Jacobi-G algorithms of this paper, there exist several parameters to be adjusted, including the positive constant υ in inequality (2.3), the stepsize t_{k-1} in equation (4.6), the positive constant ε in inequality (5.3), $\varsigma_D > 0$ in Update rule 6.5, and $\varsigma_Q > 0$ in Update rule 7.3. In this section, we choose different values for the positive constant ε in inequality (5.3), while fixing other parameters as small positive constants. We set $(\cdot)^{\blacklozenge} = (\cdot)^{\mathsf{H}}$ in both the cost functions (1.3) and (1.4). All the algorithms run at most 1000 iterations. All the randomly generated complex matrices are uniformly distributed. All the computations are done using MATLAB R2019a. The numerical experiments are conducted on a PC with an Intel® Core™ i5 CPU at 2.11 GHz and 8.00 GB of RAM in 64bt Windows operation system.

Example 9.1. For the following sets of complex matrices, we run BCD-GLU and BCD-GQU algorithms to minimize the cost function (1.3). The values of cost function (1.3) in the iterations are shown in Figure 3. The positive constant υ in inequality (2.3) is fixed to 0.001. For the positive constant ε in inequality (5.3), we choose different values. For example, *BCD-GLU 0.5* means the BCD-GLU algorithm with $\varepsilon = 0.5\sqrt{\frac{2}{3m(m-1)}}$. If $\varepsilon = 0$, we denote the BCD-GLU and BCD-GQU algorithms by BCD-CLU and BCD-CQU, respectively. The starting point is $\boldsymbol{\omega}_0 = (\boldsymbol{I}_{n\times m}, \boldsymbol{I}_m)$.
(i) We set $n = 5$, $m = 3$, and randomly generate complex matrices $\{\boldsymbol{A}_\ell\}_{1\le\ell\le 3} \subseteq \mathbb{C}^{5\times 5}$.
(ii) We set $n = 10$, $m = 8$, randomly generate a complex matrix $\boldsymbol{X} \in \mathbb{C}^{10\times 10}$, and set $\boldsymbol{A}^{(\ell)} = \boldsymbol{X}^{\mathsf{H}}(\boldsymbol{I}_{10} + \boldsymbol{e}_\ell^{\mathsf{T}}\boldsymbol{e}_\ell)\boldsymbol{X}$ for $1 \le \ell \le 5$.
(iii) We set $n = 10$, $m = 8$, randomly generate a complex upper triangular matrix $\boldsymbol{X} \in \mathbf{UT}_{10}(\mathbb{C})$, complex diagonal matrices $\{\boldsymbol{D}_\ell\}_{1\le\ell\le 5} \subseteq \mathbb{C}^{10\times 10}$, and set $\boldsymbol{A}^{(\ell)} = \boldsymbol{X}^{\mathsf{H}}\boldsymbol{D}_\ell\boldsymbol{X}$ for $1 \le \ell \le 5$.
(iv) We set $n = 10$, $m = 8$, randomly generate a complex nonsingular matrix $\boldsymbol{X} \in \mathbf{SL}_{10}(\mathbb{C})$, complex diagonal matrices $\{\boldsymbol{D}_\ell\}_{1\le\ell\le 5} \subseteq \mathbb{C}^{10\times 10}$, and set $\boldsymbol{A}^{(\ell)} = \boldsymbol{X}^{\mathsf{H}}\boldsymbol{D}_\ell\boldsymbol{X}$ for $1 \le \ell \le 5$.

Example 9.2. For the following sets of complex matrices, we run eight Jacobi-type algorithms to minimize the cost function (1.4). Here, we denote by Jacobi-GQ the gradient-based Jacobi-type algorithm on the unitary group proposed in [45], and by Jacobi-CQ the Jacobi-type algorithm on the unitary group with a cyclic ordering. Note that Jacobi-GQ and Jacobi-CQ find the iterates only in $\mathbf{U}_m(\mathbb{C})$, not in $\mathbf{SL}_m(\mathbb{C})$. The values of cost function (1.4) in the iterations are shown in Figure 4. We choose the starting point $\boldsymbol{X}_0 = \boldsymbol{I}_m$.
(i) We randomly generate two complex matrices $\{\boldsymbol{A}_\ell\}_{1\le\ell\le 2} \subseteq \mathbb{C}^{5\times 5}$.
(ii) We randomly generate a complex matrix $\boldsymbol{X} \in \mathbb{C}^{10\times 10}$, and set $\boldsymbol{A}^{(\ell)} = \boldsymbol{X}^{\mathsf{H}}(\boldsymbol{I}_{10} + \boldsymbol{e}_\ell^{\mathsf{T}}\boldsymbol{e}_\ell)\boldsymbol{X}$ for $1 \le \ell \le 10$.

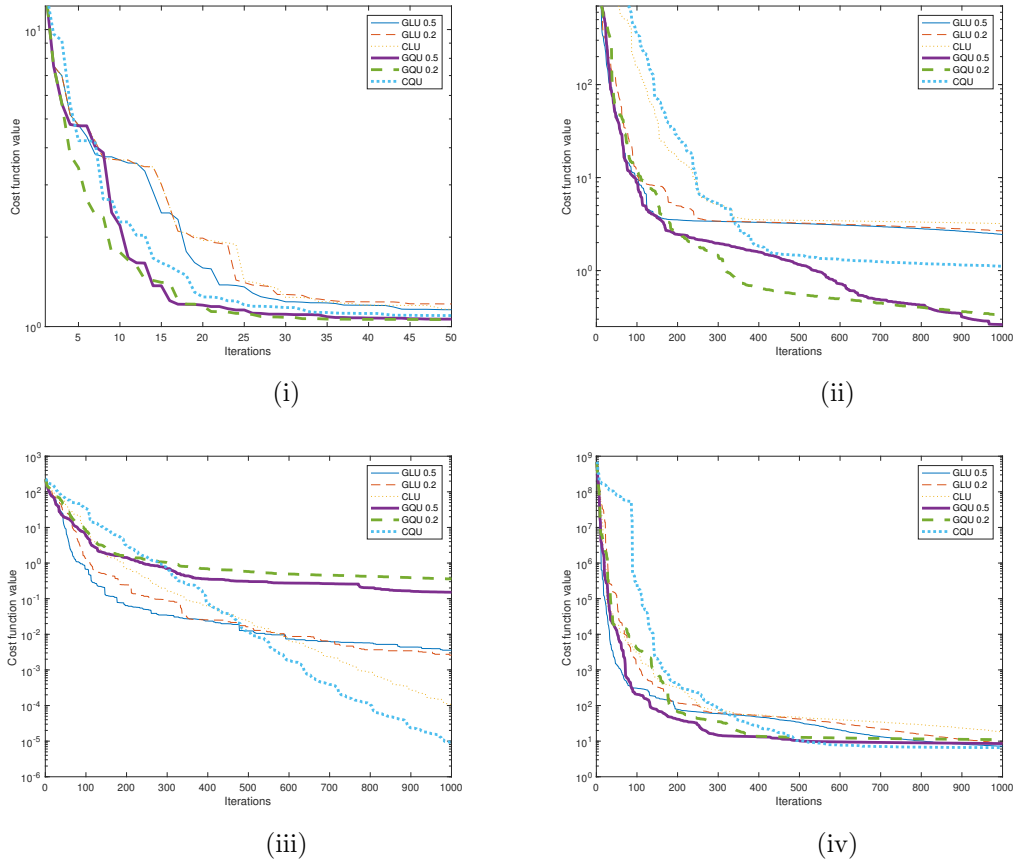

FIG. 3. *Experimental results for BCD-G algorithms in Example 9.1.*

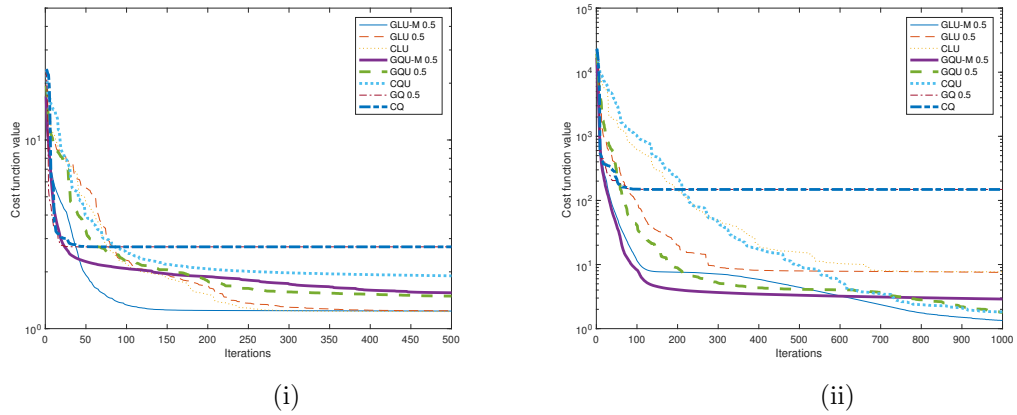

FIG. 4. *Experimental results for Jacobi-type algorithms in Example 9.2.*

From the above numerical experiments, we can see that: (i) in Example 9.1, compared with BCD-GLU algorithms, the BCD-GQU algorithms generally have better performances; (ii) in Example 9.2, compared with Jacobi-GQU algorithms, the Jacobi-GLU algorithms generally have better performances; (iii) in Example 9.2, compared with the Jacobi-GQ and Jacobi-CQ algorithms on $\mathbf{U}_m(\mathbb{C})$, the Jacobi-type algorithms on $\mathbf{SL}_m(\mathbb{C})$ considered in this paper always obtain much smaller cost function values, and they also need more iterations to attain steady state values of cost functions.

10. Conclusions. In this paper, to solve JADM problem (1.1), which is important in BSS problem, we formulate two different equivalent formulations, *i.e.*, problem (1.3) defined on $\mathbf{St}(m,n,\mathbb{C})\times\mathbf{SL}_m(\mathbb{C})$, and problem (1.4) defined on $\mathbf{SL}_m(\mathbb{C})$. Then, for these two approaches, based on the Riemannian gradients, we propose three BCD-G algorithms and two Jacobi-G algorithms, and establish their global and weak convergence, under the condition that the iterates are bounded. An interesting question is, in the BCD-G and Jacobi-G algorithms, whether one can find a method to guarantee both the boundedness of the iterates and the inequalities (3.11) and (3.12) for global and weak convergence. If so, then one can get rid of the dependence of convergence results on the condition that the iterates are bounded.

Appendix A. Proofs in Section 5. Before the proof of Lemma 5.1, we need to show a lemma, which is similar as equation (3.9) and can be directly obtained from [3, Eq. (3.31)].

LEMMA A.1. *Let* $\exp:\mathbb{C}^{2\times2}\to\mathbf{GL}_2(\mathbb{C})$ *be the matrix exponential function [3, 8, 20] sending* $\boldsymbol{\Delta}$ *to* $\exp(\boldsymbol{\Delta})$.
(i) If $h:\mathbf{SU}_2(\mathbb{C})\to\mathbb{R}$ *is a differentiable function and* $\boldsymbol{\Delta}\in\mathfrak{su}_2(\mathbb{C})=\mathbf{T}_{\boldsymbol{I}_2}\mathbf{SU}_2(\mathbb{C})$, *we have that*

$$\langle\boldsymbol{\Delta},\operatorname{grad}h(\boldsymbol{I}_2)\rangle_{\boldsymbol{I}_2}=\left(\frac{d}{dt}h(\exp(t\boldsymbol{\Delta}))\right)\Big|_{t=0}. \tag{A.1}$$

(ii) If $h:\mathbf{SUT}_2(\mathbb{C})\to\mathbb{R}$ *is a differentiable function and* $\boldsymbol{\Delta}\in\mathfrak{sut}_2(\mathbb{C})=\mathbf{T}_{\boldsymbol{I}_2}\mathbf{SUT}_2(\mathbb{C})$, *we have the relationship* (A.1).
(iii) If $h:\mathbf{SLT}_2(\mathbb{C})\to\mathbb{R}$ *is a differentiable function and* $\boldsymbol{\Delta}\in\mathfrak{slt}_2(\mathbb{C})=\mathbf{T}_{\boldsymbol{I}_2}\mathbf{SLT}_2(\mathbb{C})$, *we have the relationship* (A.1).
(iv) If $h:\mathbf{D}_2(\mathbb{C})\to\mathbb{R}$ *is a differentiable function and* $\boldsymbol{\Delta}\in\mathfrak{d}_2(\mathbb{C})=\mathbf{T}_{\boldsymbol{I}_2}\mathbf{D}_2(\mathbb{C})$, *we have the relationship* (A.1).

Proof of Lemma 5.1. Define a projection operator $\mathcal{P}_{i,j}:\mathbb{C}^{m\times m}\to\mathbb{C}^{2\times2}$ extracting a submatrix of $\boldsymbol{X}\in\mathbb{C}^{m\times m}$ as in [45, Eq. (3.7)], and $\mathcal{P}_{i,j}^{\mathsf{T}}:\mathbb{C}^{2\times2}\to\mathbb{C}^{m\times m}$ the conjugate operator. For the elementary function $h^{(Q)}_{(i,j),\boldsymbol{X}}$ defined in (5.1), if $\boldsymbol{\Delta}\in\mathfrak{su}_2(\mathbb{C})=\mathbf{T}_{\boldsymbol{I}_2}\mathbf{SU}_2(\mathbb{C})$, we have that

$$\begin{aligned}
&\langle\boldsymbol{\Delta},\operatorname{grad}h^{(Q)}_{(i,j),\boldsymbol{X}}(\boldsymbol{I}_2)\rangle_{\boldsymbol{I}_2}=\left(\frac{d}{dt}h^{(Q)}_{(i,j),\boldsymbol{X}}(\exp(t\boldsymbol{\Delta}))\right)\Big|_{t=0}\quad\text{(by Lemma A.1(i))}\\
&=\left(\frac{d}{dt}g(\boldsymbol{X}\boldsymbol{Q}^{(i,j,\exp(t\boldsymbol{\Delta}))})\right)\Big|_{t=0}=\left(\frac{d}{dt}g(\operatorname{Exp}_{\boldsymbol{X}}(\boldsymbol{X}\mathcal{P}_{i,j}^{\mathsf{T}}(\boldsymbol{\Delta})t))\right)\Big|_{t=0}\\
&=\langle\boldsymbol{X}\mathcal{P}_{i,j}^{\mathsf{T}}(\boldsymbol{\Delta}),\operatorname{grad}g(\boldsymbol{X})\rangle_{\boldsymbol{X}}\ \text{(by equation (3.9))}=\langle\boldsymbol{\Delta},\mathcal{P}_{i,j}(\boldsymbol{\Lambda})\rangle_{\boldsymbol{I}_2},
\end{aligned}$$

where $\operatorname{Exp}_{\boldsymbol{X}}$ is the map defined in (3.8). Note that $\boldsymbol{\Delta}\in\mathfrak{su}_2(\mathbb{C})$ and $\operatorname{grad}h^{(Q)}_{(i,j),\boldsymbol{X}}(\boldsymbol{I}_2)\in\mathfrak{su}_2(\mathbb{C})$. The result can be obtained by direct calculations. For other three elementary functions $h^{(U)}_{(i,j),\boldsymbol{X}}$, $h^{(L)}_{(i,j),\boldsymbol{X}}$ and $h^{(D)}_{(i,j),\boldsymbol{X}}$, similar as the above case, we can obtain the

results by Lemma A.1(ii), Lemma A.1(iii) and Lemma A.1(iv), respectively. The proof is complete. ∎

We need a simple lemma before the proofs of Proposition 5.3 and Proposition 5.4.

LEMMA A.2. *(i) If $z_1, z_2 \in \mathbb{C}$, then*

$$|z_1 - z_2|^2 + |z_2|^2 \geq \frac{3-\sqrt{5}}{2}(|z_1|^2 + |z_2|^2).$$

(ii) If $\{z_i\}_{1\leq i\leq m} \subseteq \mathbb{C}$ satisfy $\sum_{1\leq i\leq m} z_i = 0$, then

$$\sum_{1\leq i<j\leq m} |z_i - z_j|^2 = m \sum_{1\leq i\leq m} |z_i|^2.$$

Proof of Proposition 5.3. (i) We first prove the existence of such an index pair (i_k, j_k) and an elementary function h_k in Subalgorithm 1a. By Lemma 5.2, Lemma A.2 and $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}(\boldsymbol{X}_{k-1}) \in \mathfrak{sl}_m(\mathbb{C})$, we have that

$$\begin{aligned}&\sum_{1\leq i_k<j_k\leq m} \left(\|\partial h_k^{(U)}(\boldsymbol{I}_2)\|^2 + \|\partial h_k^{(L)}(\boldsymbol{I}_2)\|^2 + \|\partial h_k^{(D)}(\boldsymbol{I}_2)\|^2\right)\\ &= \sum_{1\leq i_k<j_k\leq m} \left(|\Lambda_{i_kj_k}|^2 + |\Lambda_{j_ki_k}|^2\right) + m \sum_{1\leq i_k\leq m} |\Lambda_{i_ki_k}|^2 \geq \|\boldsymbol{\Lambda}\|^2.\end{aligned}$$

Therefore, there exist an index pair (i_k, j_k) and an elementary function $h_k = h_k^{(U)}, h_k^{(L)}$ or $h_k^{(D)}$ such that $\frac{3}{2}m(m-1)\|\partial h_k(\boldsymbol{I}_2)\|^2 \geq \|\boldsymbol{\Lambda}\|^2$.
(ii) We now prove the existence in Subalgorithm 1b. Similar as above, we get that

$$\begin{aligned}&\sum_{1\leq i_k<j_k\leq m} \left(\|\partial h_k^{(Q)}(\boldsymbol{I}_2)\|^2 + \|\partial h_k^{(U)}(\boldsymbol{I}_2)\|^2 + \|\partial h_k^{(D)}(\boldsymbol{I}_2)\|^2\right)\\ &= \sum_{1\leq i_k<j_k\leq m} \left(|\Lambda^*_{i_kj_k} - \Lambda_{j_ki_k}|^2 + |\Lambda_{i_kj_k}|^2 + |\Lambda_{i_ki_k} - \Lambda_{j_kj_k}|^2\right)\\ &\geq \frac{3-\sqrt{5}}{2} \sum_{1\leq i_k<j_k\leq m} \left(|\Lambda_{i_kj_k}|^2 + |\Lambda_{j_ki_k}|^2\right) + m \sum_{1\leq i_k\leq m} |\Lambda_{i_ki_k}|^2 \geq \frac{3-\sqrt{5}}{2}\|\boldsymbol{\Lambda}\|^2.\end{aligned}$$

Therefore, there exists an index pair (i_k, j_k) and an elementary function $h_k = h_k^{(Q)}, h_k^{(U)}$ or $h_k^{(D)}$ such that $\frac{3}{3-\sqrt{5}}m(m-1)\|\partial h_k(\boldsymbol{I}_2)\|^2 \geq \|\boldsymbol{\Lambda}\|^2$. The proof is complete. ∎

Proof of Proposition 5.4. Note that the starting point $\boldsymbol{X}_0 \in \mathbf{EUT}_m(\mathbb{C})$ in Subalgorithm 1c. We see that $\boldsymbol{X}_k \in \mathbf{EUT}_m(\mathbb{C})$ for all $k \in \mathbb{N}$. By Lemma 5.2, Lemma A.2 and $\boldsymbol{\Lambda} = \boldsymbol{\Lambda}(\boldsymbol{X}_{k-1}) \in \mathfrak{eut}_m(\mathbb{C})$, we get that

$$\begin{aligned}\sum_{1\leq i_k<j_k\leq m} \left(\|\partial h_k^{(U)}(\boldsymbol{I}_2)\|^2 + \|\partial h_k^{(D)}(\boldsymbol{I}_2)\|^2\right) &= \sum_{1\leq i_k<j_k\leq m} \left(|\Lambda_{i_kj_k}|^2 + |\Lambda_{i_ki_k} - \Lambda_{j_kj_k}|^2\right)\\ &= \sum_{1\leq i_k<j_k\leq m} |\Lambda_{i_kj_k}|^2 + m \sum_{1\leq i_k\leq m} |\Lambda_{i_ki_k}|^2 \geq \|\boldsymbol{\Lambda}\|^2.\end{aligned}$$

Therefore, there exist an index pair (i_k, j_k) and an elementary function $h_k = h_k^{(U)}$ or $h_k^{(D)}$ such that $m(m-1)\|\partial h_k(\boldsymbol{I}_2)\|^2 \geq \|\boldsymbol{\Lambda}\|^2$. The proof is complete. ∎

Appendix B. Proofs in Section 6.

Proof of Lemma 6.6. We now prove the inequality (6.6) by Lemma 6.2(iii) in three different cases shown in Update rule 6.5.

- If $\gamma_1 = \gamma_2 = 0$, it is clear that the inequality (6.6) is satisfied for any $\iota_D > 0$.

- If $\varpi \in [0, \varsigma_D)$, we get that

$$\begin{aligned} &h_k^{(D)}(1,0) - h_k^{(D)}(x_k^*, y_k^*) = \frac{3}{4}\gamma_1(1-4\varpi) = \frac{3(1-4\varpi)}{8(1-\varpi)}|\partial h_k^{(D)}(1,0)| \\ &\geq \frac{3(1-4\varpi)\varepsilon}{8(1-\varpi)}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\| = \frac{3(1-4\varpi)\varepsilon}{4\sqrt{5}(1-\varpi)}\frac{\sqrt{5}}{2}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\| \\ &\geq \frac{3(1-4\varpi)\varepsilon}{4\sqrt{5}(1-\varpi)}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\| \geq \frac{3(1-4\varsigma_D)\varepsilon}{4\sqrt{5}}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\|. \end{aligned}$$

- If $\varpi \in (\frac{1}{\varsigma_D}, +\infty]$, we similarly get the above inequality.
- If $\varpi \in [\varsigma_D, \frac{1}{\varsigma_D}]$, it is easy to verify that

$$\sqrt[4]{\gamma_1\gamma_2} \geq \frac{\sqrt[4]{\varsigma_D}}{2}\left(\sqrt{\gamma_1} + \sqrt{\gamma_2}\right). \tag{B.1}$$

Then, we get that

$$\begin{aligned} &h_k^{(D)}(1,0) - h_k^{(D)}(x_k^*, y_k^*) = \left(\sqrt{\gamma_1} - \sqrt{\gamma_2}\right)^2 = \frac{1}{2}|\partial h_k^{(D)}(1,0)|\frac{\left|\sqrt{\gamma_1} - \sqrt{\gamma_2}\right|}{\sqrt{\gamma_1} + \sqrt{\gamma_2}} \\ &\geq \frac{\varepsilon\sqrt[4]{\varsigma_D}}{4}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|\frac{\left|\sqrt{\gamma_1} - \sqrt{\gamma_2}\right|}{\sqrt[4]{\gamma_1\gamma_2}} \quad \text{(by equation (B.1))} \\ &\geq \frac{\varepsilon\sqrt[4]{\varsigma_D}}{4}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|\frac{\left|\sqrt[4]{\gamma_1} - \sqrt[4]{\gamma_2}\right|\left(\sqrt{\gamma_1} + \sqrt{\gamma_2}\right)^{1/2}}{\sqrt[4]{\gamma_1\gamma_2}} \\ &\geq \frac{\varepsilon\sqrt[4]{\varsigma_D}}{4}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\||x_k^* - 1|\sqrt{1 + \frac{1}{{x_k^*}^2}} \geq \frac{\varepsilon\sqrt[4]{\varsigma_D}}{4}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\|. \end{aligned}$$

Now we set $\iota_D = \min(\frac{3(1-4\varsigma_D)\varepsilon}{4\sqrt{5}}, \frac{\varepsilon\sqrt[4]{\varsigma_D}}{4})$. The proof is complete. ∎

Proof of Lemma 6.8. We prove that the inequality (6.10) is satisfied in two cases.

- If $h_k = h_k^{(U)}$, by Lemma 6.2(i), we see that

$$\begin{aligned} g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) &= h_k^{(U)}(0,0) - h_k^{(U)}(x_k^*, y_k^*) = \frac{1}{\alpha_1}(\alpha_2^2 + \alpha_3^2) \\ &= \frac{1}{2}\|\partial h_k^{(U)}(0,0)\|\|(x_k^*, y_k^*)\| \geq \frac{\varepsilon}{2}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\|. \end{aligned}$$

- If $h_k = h_k^{(L)}$, by Lemma 6.2(ii), we see that

$$\begin{aligned} g(\boldsymbol{X}_{k-1}) - g(\boldsymbol{X}_k) &= h_k^{(L)}(0,0) - h_k^{(L)}(x_k^*, y_k^*) = \frac{1}{\beta_1}(\beta_2^2 + \beta_3^2) \\ &= \frac{1}{2}\|\partial h_k^{(L)}(0,0)\|\|(x_k^*, y_k^*)\| \geq \frac{\varepsilon}{2}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\|. \end{aligned}$$

Now we set $\iota_{LU} = \frac{\varepsilon}{2}$. The proof is complete. ∎

Proof of Lemma 6.10. If $h_k = h_k^{(U)}$, we have

$$\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\|^2 = \frac{\alpha_2^2 + \alpha_3^2}{\alpha_1^2} \geq \frac{\|\partial h_k^{(U)}(0,0)\|^2}{4\mathrm{M}_\alpha^2} \geq \frac{\varepsilon^2}{4\mathrm{M}_\alpha^2}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|^2,$$

where M_α is a fixed positive constant always satisfying $|\alpha_1| \leq \mathrm{M}_\alpha$. The case $h_k = h_k^{(L)}$ is similar. Now we prove the $h_k = h_k^{(D)}$ case. If $\varpi \in [0, \varsigma_D)$, we have that

$$\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\|^2 = \frac{5}{4} \geq \frac{5}{4}\frac{1}{4(\gamma_1^2+\gamma_2^2)}\|\partial h_k^{(D)}(1,0)\|^2 \geq \frac{5}{4}\frac{\varepsilon^2}{4\mathrm{M}_0^2}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|^2.$$

The case $\varpi \in (\frac{1}{\varsigma_D}, +\infty]$ is similar. If $\varpi \in [\varsigma_D, \frac{1}{\varsigma_D}]$, we have

$$\|\boldsymbol{\Psi}_k^* - \boldsymbol{I}_2\|^2 \geq (1 - x_k^*)^2 \geq \frac{\gamma_1^2(1-\varpi)^2}{\mathrm{M}_0^2\mathrm{M}_\varpi^2} = \frac{\|\partial h_k^{(D)}(1,0)\|^2}{4\mathrm{M}_0^2\mathrm{M}_\varpi^2} \geq \frac{\varepsilon^2}{4\mathrm{M}_0^2\mathrm{M}_\varpi^2}\|\boldsymbol{\Lambda}(\boldsymbol{X}_{k-1})\|^2,$$

where M_ϖ is a fixed positive constant always satisfying $(1+\sqrt{\varpi})(1+\sqrt[4]{\varpi}) \leq \mathrm{M}_\varpi$. We only need to set κ^2 to be the minimum of all the above corresponding positive constants. The proof is complete. ∎

Acknowledgment. The authors would like to thank the three anonymous reviewers and the editor for their helpful suggestions and comments, which significantly improved the presentation of the article.

REFERENCES

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

[2] P.-A. Absil, R. Mahony, and B. Andrews, *Convergence of the iterates of descent methods for analytic cost functions*, SIAM Journal on Optimization, 16 (2005), pp. 531–547.

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

[4] B. Afsari, *Gradient flow-based matrix joint diagonalization for independent component analysis*, University of Maryland, College Park, 2004. Master's thesis.

[5] B. Afsari, *Simple LU and QR based non-orthogonal matrix joint diagonalization*, in International Conference on Independent Component Analysis and Signal Separation, Springer, 2006, pp. 1–7.

[6] B. Afsari, *What can make joint diagonalization difficult?*, in ICASSP, vol. III, Honolulu, Apr. 2007, pp. 1377–1380.

[7] R. André, X. Luciani, and E. Moreau, *A new class of block coordinate algorithms for the joint eigenvalue decomposition of complex matrices*, Signal Processing, 145 (2018), pp. 78–90.

[8] A. Baker, *Matrix groups: An introduction to Lie group theory*, Springer Science & Business Media, 2012.

[9] D. P. Bertsekas, *Nonlinear programming*, Athena Scientific, second ed., 1999.

[10] F. Bouchard, B. Afsari, J. Malick, and M. Congedo, *Approximate joint diagonalization with Riemannian optimization on the general linear group*, SIAM Journal on Matrix Analysis and Applications, 41 (2020), pp. 152–170.

[11] D. Brandwood, *A complex gradient operator and its application in adaptive array theory*, IEE Proceedings H-Microwaves, Optics and Antennas, 130 (1983), pp. 11–16.

[12] J. Cardoso and A. Souloumiac, *Blind beamforming for non-gaussian signals*, IEE Proceedings F (Radar and Signal Processing), 6 (1993), pp. 362–370.

[13] J.-F. Cardoso and A. Souloumiac, *Jacobi angles for simultaneous diagonalization*, SIAM Journal on Matrix Analysis and Applications, 17 (1996), pp. 161–164.

[14] G. Chabriel, M. Kleinsteuber, E. Moreau, H. Shen, P. Tichavsky, and A. Yeredor, *Joint matrices decompositions and blind source separation: A survey of methods, identification, and applications*, IEEE Signal Processing Magazine, 31 (2014), pp. 34–43.

[15] B. Chen, S. He, Z. Li, and S. Zhang, *Maximum block improvement and polynomial optimization*, SIAM Journal on Optimization, 22 (2012), pp. 87–107.
[16] P. Comon, *Independent Component Analysis*, in Higher Order Statistics, J.-L. Lacoume, ed., Elsevier, Amsterdam, London, 1992, pp. 29–38.
[17] P. Comon, *Independent component analysis, a new concept?*, Signal Processing, 36 (1994), pp. 287–314.
[18] P. Comon and C. Jutten, eds., *Handbook of Blind Source Separation*, Academic Press, Oxford, 2010.
[19] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, third ed., 1996.
[20] B. Hall, *Lie groups, Lie algebras, and representations: an elementary introduction*, vol. 222, Springer, 2015.
[21] M. Ishteva, P.-A. Absil, and P. Van Dooren, *Jacobi algorithm for the best low multilinear rank approximation of symmetric tensors*, SIAM Journal on Matrix Analysis and Applications, 2 (2013), pp. 651–672.
[22] S. Krantz and H. Parks, *A Primer of Real Analytic Functions*, Birkhäuser Boston, 2002.
[23] S. G. Krantz, *Function theory of several complex variables*, vol. 340, American Mathematical Soc., 2001.
[24] S. law Lojasiewicz, *Ensembles semi-analytiques*, IHES notes, (1965).
[25] J. Li, K. Usevich, and P. Comon, *Globally convergent Jacobi-type algorithms for simultaneous orthogonal symmetric tensor diagonalization*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 1–22.
[26] J. Li, K. Usevich, and P. Comon, *On approximate diagonalization of third order symmetric tensors by orthogonal transformations*, Linear Algebra and its Applications, 576 (2019), pp. 324–351.
[27] J. Li, K. Usevich, and P. Comon, *On the convergence of jacobi-type algorithms for independent component analysis*, in 2020 IEEE 11th Sensor Array and Multichannel Signal Processing Workshop (SAM), IEEE, 2020, pp. 1–5.
[28] J. Li, K. Usevich, and P. Comon, *Jacobi-type algorithm for low rank orthogonal approximation of symmetric tensors and its convergence analysis*, Pacific Journal of Optimization, 17 (2021), pp. 357–379.
[29] J. Li and S. Zhang, *Polar decomposition based algorithms on the product of Stiefel manifolds with applications in tensor approximation*, Journal of the Operations Research Society of China, (2023).
[30] Z. Li, A. Uschmajew, and S. Zhang, *On convergence of the maximum block improvement method*, SIAM Journal on Optimization, 25 (2015), pp. 210–233.
[31] S. Łojasiewicz, *Sur la géométrie semi- et sous-analytique*, Annales de l'institut Fourier, 43 (1993), pp. 1575–1595.
[32] Z.-Q. Luo and P. Tseng, *On the convergence of the coordinate descent method for convex differentiable minimization*, Journal of Optimization Theory and Applications, 72 (1992), pp. 7–35.
[33] Z.-Q. Luo and P. Tseng, *Error bounds and convergence analysis of feasible descent methods: a general approach*, Annals of Operations Research, 46 (1993), pp. 157–178.
[34] J. H. Manton, *Modified steepest descent and Newton algorithms for orthogonally constrained optimisation. part i. the complex Stiefel manifold*, in Proceedings of the Sixth International Symposium on Signal Processing and its Applications, vol. 1, IEEE, 2001, pp. 80–83.
[35] V. Maurandi, C. De Luigi, and E. Moreau, *Fast jacobi like algorithms for joint diagonalization of complex symmetric matrices*, in 21st European Signal Processing Conference (EUSIPCO 2013), IEEE, 2013, pp. 1–5.
[36] V. Maurandi and E. Moreau, *A decoupled Jacobi-like algorithm for non-unitary joint diagonalization of complex-valued matrices*, IEEE Signal Processing Letters, 21 (2014), pp. 1453–1456.
[37] V. Maurandi, E. Moreau, and C. De Luigi, *Jacobi like algorithm for non-orthogonal joint diagonalization of hermitian matrices*, in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2014, pp. 6196–6200.
[38] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.
[39] W. Ring and B. Wirth, *Optimization methods on Riemannian manifolds and their application to shape space*, SIAM Journal on Optimization, 22 (2012), pp. 596–627.
[40] H. Sato and T. Iwai, *A new, globally convergent Riemannian conjugate gradient method*, Optimization, 64 (2015), pp. 1011–1031.
[41] R. Schneider and A. Uschmajew, *Convergence results for projected line-search methods on*

varieties of low-rank matrices via łojasiewicz inequality, SIAM Journal on Optimization, 25 (2015), pp. 622–646.

[42] M. Sørensen, P. Comon, S. Icart, and L. Deneire, *Approximate tensor diagonalization by invertible transforms*, in 2009 17th European Signal Processing Conference, Eurasip, 2009, pp. 500–504.

[43] A. Souloumiac, *Nonorthogonal joint diagonalization by combining givens and hyperbolic rotations*, IEEE Transactions on Signal Processing, 57 (2009), pp. 2222–2231.

[44] A. Uschmajew, *A new convergence proof for the higher-order power method and generalizations*, Pacific Journal of Optimization, 11 (2015), pp. 309–321.

[45] K. Usevich, J. Li, and P. Comon, *Approximate matrix and tensor diagonalization by unitary transformations: convergence of jacobi-type algorithms*, SIAM Journal on Optimization, 30 (2020), pp. 2998–3028.

[46] K. Wang, X.-F. Gong, and Q.-H. Lin, *Complex non-orthogonal joint diagonalization based on LU and LQ decompositions*, in International Conference on Latent Variable Analysis and Signal Separation, Springer, 2012, pp. 50–57.

[47] S. J. Wright, *Coordinate descent algorithms*, Mathematical Programming, 151 (2015), pp. 3–34.

[48] Y. Xu and W. Yin, *A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 1758–1789.

[49] A. Yeredor, *Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation*, IEEE Transactions on Signal Processing, 50 (2002), pp. 1545–1553.