Numerische Mathematik (2022) 152:587–629
https://doi.org/10.1007/s00211-022-01322-y

Numerische Mathematik

Maximizing sum of coupled traces with applications

Li Wang[1] · **Lei-Hong Zhang**[2] · **Ren-Cang Li**[1,3]

Received: 27 November 2021 / Revised: 29 May 2022 / Accepted: 3 October 2022 /
Published online: 14 October 2022

Abstract
This paper concerns maximizing the sum of coupled traces of quadratic and linear matrix forms. The coupling comes from requiring the matrix variables in the quadratic and linear matrix forms to be packed together to have orthonormal columns. At a maximum, the KKT condition becomes a nonlinear polar decomposition (NPD) of a matrix-valued function with dependency on the orthogonal polar factor. A self-consistent-field iteration, along with a locally optimal conjugate gradient (LOCG) acceleration, are proposed to compute the NPD. It is proved that both methods are convergent and it is demonstrated numerically that the LOCG acceleration is very effective. As applications, we demonstrate our methods on the MAXBET subproblem and the multi-view partially shared subspace learning (MvPS) subproblem, both of which sit at the computational kernels of two multi-view subspace learning models. In particular, we also demonstrate MvPS on several real world data sets.

Keywords Coupled traces · Stiefel manifold · Nonlinear polar decomposition · NPD · SCF · Multi-view subspace learning

Mathematics Subject Classification 58C40 · 65F30 · 65H17 · 65K05 · 90C26 · 90C32

✉ Ren-Cang Li
rcli@uta.edu

Li Wang
li.wang@uta.edu

Lei-Hong Zhang
longzlh@suda.edu.cn

1 Department of Mathematics, University of Texas at Arlington, P.O. Box 19408, Arlington, TX 76019-0408, USA

2 School of Mathematical Sciences, Soochow University, Suzhou, Jiangsu 215006, China

3 Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong

1 Introduction

In this paper, we investigate the following maximization problem for the sum of coupled traces:

$$\max_{X^{\mathrm{T}}X=I_k} \left\{ f(X) := \sum_{j=1}^{\ell} \left[\operatorname{tr}(X_j^{\mathrm{T}} A_j X_j) + 2\operatorname{tr}(X_j^{\mathrm{T}} D_j) \right] \right\}, \tag{1.1a}$$

where $A_j \in \mathbb{R}^{n\times n}$ for $1 \le j \le \ell$ are symmetric and $D_j \in \mathbb{R}^{n\times k_j}$ for $1 \le j \le \ell$, the optimizing variable $X \in \mathbb{R}^{n\times k}$ is partitioned as $X = [X_1, X_2, \ldots, X_\ell]$ with $X_j \in \mathbb{R}^{n\times k_j}$ for $1 \le j \le \ell$, and $k = \sum_{j=1}^{\ell} k_j$. The coupledness in (1.1) is caused by the constraint that X composed of X_j for $1 \le j \le \ell$ associated with different traces has orthonormal columns. Necessarily $1 \le k \le n$, but usually $k \ll n$ in applications.

Without loss of generality, in the rest of this paper, we will assume

$$\boxed{A_j \succeq 0, \text{ i.e., positive semi-definite, for } 1 \le j \le \ell.} \tag{1.1b}$$

Otherwise, we may consider maximizing

$$f_\alpha(X) := \sum_{j=1}^{\ell} \left[\operatorname{tr}(X_j^{\mathrm{T}}[A_j + \alpha_j I_n]X_j) + 2\operatorname{tr}(X_j^{\mathrm{T}} D_j) \right] = f(X) + \sum_{j=1}^{\ell} k_j \alpha_j,$$

instead, where α_j for $1 \le j \le \ell$ are constants such that $A_j + \alpha_j I_n \succeq 0$. Clearly, any $\alpha_j \ge -\lambda_{\min}(A_j)$, the opposite of the smallest eigenvalue of A_j, suffices. It can be seen that the same maximizers optimize both $f(X)$ and $f_\alpha(X)$. Numerically, α_j can be estimated cheaply [66].

Compactly, the objective function of (1.1) can be written as

$$f(X) = \operatorname{tr}(X^{\mathrm{T}}\mathscr{A}(X)) + 2\operatorname{tr}(X^{\mathrm{T}} D), \tag{1.2a}$$

where, for $X \in \mathbb{R}^{n\times k}$ partitioned as above,

$$\mathscr{A}(X) := [A_1X_1, \ldots, A_\ell X_\ell] \in \mathbb{R}^{n\times k}, \quad D := [D_1, D_2, \ldots, D_\ell] \in \mathbb{R}^{n\times k}. \tag{1.2b}$$

There are two well-known special cases. The first special case is that all A_j are the same, say $A_j = A = A^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ for all j, and $D = 0$. Then (1.1) becomes

$$\max_{X^{\mathrm{T}}X=I_k} \operatorname{tr}(X^{\mathrm{T}} A X), \tag{1.3}$$

which admits an analytic solution in terms of the eigendecomposition of A, and is known as Ky Fan's trace maximization principle [16] [21, p.248]. Namely, the optimal objective value is the sum of the first k largest eigenvalues of A, and a corresponding global maximizer X can be any orthonormal eigenbasis matrix of A associated with its

k largest eigenvalues. There are infinitely many maximizers, but their column spaces are the same if the kth largest eigenvalue of A is strictly bigger than its $(k+1)$st largest eigenvalue. There are many efficient numerical linear algebra techniques, and software packages as well, to solve (1.3) [2, 3, 13, 33, 44, 47], regardless of its scale – small, medium, or large. The second special case is that still all A_j are the same, say $A_j = A = A^{\mathrm{T}} \in \mathbb{R}^{n\times n}$ for all j, but $D \neq 0$. Then (1.1) becomes the MAXBET subproblem

$$\max_{X^{\mathrm{T}}X=I_k} \operatorname{tr}(X^{\mathrm{T}}AX) + 2\operatorname{tr}(X^{\mathrm{T}}D). \tag{1.4}$$

MAXBET is a statistical term that appeared first in [53] in 1984 and then in [52], and more detail can be found in Sect. 5.1 later. Simply because $D \neq 0$, problem (1.4) does not admit analytic solutions any more. Problem (1.4) is a fundamental one in optimization and applied statistics [11, 14]. The unbalanced Procrustes problem [11, 14, 15, 19, 23, 64]

$$\min_{X\in\mathbb{O}^{n\times k}} \|CX - B\|_{\mathrm{F}}^2 \tag{1.5}$$

can be equivalently turned into (1.4) with $A = -C^{\mathrm{T}}C$ and $D = C^{\mathrm{T}}B$. Numerically solving the MAXBET problem [35, 53] via an alternating scheme has (1.4) at its core as subproblems. Both (1.4) and (1.5) can also be found in many real world applications including the orthogonal least squares regression (OLSR) for feature extraction [39, 65], the multidimensional similarity structure analysis (SSA) [8, Chapter 19]. Recently in [63], for its numerical solution (1.4) is turned into an equivalent nonlinear eigenvalue problem with eigenvector dependency (NEPv), a term that was first coined in [10].

Another not-so-common case is when $\ell = k$, all $k_j = 1$, and all $D_j = 0$. Adopting the convention of using lower-case letters for vectors, we rewrite (1.1) for this case as

$$\max_{X^{\mathrm{T}}X=I_k} \sum_{j=1}^{k} \boldsymbol{x}_j^{\mathrm{T}} A_j \boldsymbol{x}_j. \tag{1.6}$$

Its objective was called in [7] the *sum of heterogeneous quadratic form* [7]. It does not admit analytic solutions in general either. In [7], the authors established the KKT condition for (1.6) and presented an iterative method. More recently, in [5, 6, 46] problem (1.6) was used as an illustrative example for optimization on Stiefel manifolds.

Optimization problem (1.1) is one of many over the Stiefel manifold

$$\mathbb{O}^{n\times k} = \{X \in \mathbb{R}^{n\times k} \,:\, X^{\mathrm{T}}X = I_k\}.$$

There is enormous interest in optimization on manifolds, especially over the last two decades. Generic constrained optimization methods [41] can be readily applied [12], but they often perform not so well as their variations adapted particularly for optimization on Riemannian manifolds (see [1, 9, 14, 17, 54, 56] and references therein). On the other hand, customized optimization methods through taking advantage of structures in objective functions such as $f(X)$ in (1.1) and leveraging mature numerical linear algebra techniques and software packages can gain even more efficiency (see [29, 35, 60–63] and references therein). Our goal in this paper is to design customized

optimization methods that can solve (1.1) much more efficiently than existing adapted methods. In particular, we will provide numerical comparisons among our methods and solvers from `manopt` [9] and `STOP` [17, 54], two widely used MALAB toolboxes written by experts for optimization on manifolds, to demonstrate superiority of our customized optimization methods.

The rest of this paper is organized as follows. In Sect. 2, we derive the KKT condition, its associated nonlinear polar decomposition (NPD), and necessary conditions for a global maximizer beyond the standard first order conditions. In Sect. 3, we propose our SCF iteration for problem (1.1) and conduct a detailed convergence analysis of the method. In Sect. 4, we propose an acceleration technique to speedup the SCF iteration. Two applications of (1.1), namely the MAXBET subproblem and the MvPS subproblem, are discussed in Sect. 5 with numerical demonstrations and applications to multi-view subspace learning. Finally, we draw our conclusions in Sect. 6.

Notation. $\mathbb{R}^{m\times n}$ is the set of $m \times n$ real matrices, and $\mathbb{R}^n = \mathbb{R}^{n\times 1}$ and $\mathbb{R} = \mathbb{R}^1$. $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix. B^{T} stands for the transpose of a matrix/vector. $\mathcal{R}(B)$ is the column subspace of B, spanned by its columns. For $B \in \mathbb{R}^{m\times n}$, unless otherwise explicitly stated, its SVD is referred to the one $B = U\Sigma V^{\mathrm{T}}$ with

$$\Sigma = \mathrm{diag}(\sigma_1(B), \sigma_2(B), \ldots, \sigma_s(B)) \in \mathbb{R}^{s\times s},\ U \in \mathbb{O}^{m\times s},\ V \in \mathbb{O}^{n\times s},$$

where $s = \min\{m, n\}$, and the singular values $\sigma_j(B)$ are always arranged decreasingly as

$$\sigma_1(B) \geq \sigma_2(B) \geq \cdots \geq \sigma_s(B) \geq 0.$$

$\|B\|_2$, $\|B\|_{\mathrm{F}}$, and $\|B\|_{\mathrm{tr}}$ are its spectral, Frobenius, and trace norms:

$$\|B\|_2 = \sigma_1(B),\ \ \|B\|_{\mathrm{F}} = \Big(\sum_{i=1}^{s}[\sigma_i(B)]^2\Big)^{1/2},\ \ \|B\|_{\mathrm{tr}} = \sum_{i=1}^{s}\sigma_i(B),$$

respectively. For a symmetric matrix $A \in \mathbb{R}^{n\times n}$, $\mathrm{eig}(A) = \{\lambda_i(A)\}_{i=1}^n$ denotes the set of its eigenvalues (counted by multiplicities) arranged in the decreasing order as well. A matrix $A \succ 0$ ($\succeq 0$) means that it is symmetric and positive definite (semi-definite), and accordingly $A \prec 0$ ($\preceq 0$) if $-A \succ 0$ ($\succeq 0$).

2 KKT condition and nonlinear polar decomposition

In the rest of this paper, as a convention and often without explicitly stating it, any matrix $X \in \mathbb{R}^{n\times k}$ may be partitioned into ℓ column blocks, with the jth column block denoted by $X_j \in \mathbb{R}^{n\times k_j}$, in accordance with the way the objective function of (1.1) is defined.

It can be seen that the partial derivative of f with respect to $X \in \mathbb{R}^{n\times k}$ is

$$\frac{\partial f(X)}{\partial X} = 2\mathscr{A}(X) + 2D =: 2\,\mathscr{B}(X), \tag{2.1}$$

where $\mathscr{A}(X)$ is as defined in (1.2b). Let

$$\Pi_X(Z) := Z - X\,\mathrm{sym}(X^{\mathrm{T}}Z) \text{ for } Z \in \mathbb{R}^{n\times k},$$

where $\mathrm{sym}(X^{\mathrm{T}}Z) = (X^{\mathrm{T}}Z + Z^{\mathrm{T}}X)/2$. The gradient of f with respect to the Stiefel manifold $\mathbb{O}^{n\times k}$ at X is then given by [1, (3.35)]

$$\begin{aligned}\mathrm{grad}\, f_{|\mathbb{O}^{n\times k}}(X) &= \Pi_X\left(\frac{\partial f(X)}{\partial X}\right) = \frac{\partial f(X)}{\partial X} - X\,\mathrm{sym}\left(X^{\mathrm{T}}\frac{\partial f(X)}{\partial X}\right)\\ &= 2\mathscr{A}(X) + 2D - 2X\Lambda\\ &= 2\big[\mathscr{B}(X) - X\Lambda\big],\end{aligned} \tag{2.2}$$

where $\Lambda \in \mathbb{R}^{k\times k}$ is symmetric and its explicit form, although available, is not important to us. Finally, the KKT condition, also known as the first order optimality condition, is $\mathrm{grad}\, f_{|\mathbb{O}^{n\times k}}(X) = 0$, or equivalently,

$$\mathscr{B}(X) = X\Lambda, \quad X \in \mathbb{O}^{n\times k}, \tag{2.3a}$$

$$\Lambda^{\mathrm{T}} = \Lambda \in \mathbb{R}^{k\times k}. \tag{2.3b}$$

Pre-multiply the first equation in (2.3a) by X^{T} to get $\Lambda = X^{\mathrm{T}}\mathscr{B}(X)$ in terms of X and $\mathscr{B}(X)$. Also (2.3a) implies $\mathcal{R}(\mathscr{B}(X)) \subseteq \mathcal{R}(X)$.

In what follows, we will establish some necessary conditions for a global maximizer, keyond the KKT condition (2.3). Recall assumption (1.1b) that $A_j \succeq 0$ for $1 \le j \le \ell$.

Lemma 2.1 ([62, Lemma 3]) *For $H \in \mathbb{R}^{m\times m}$, we have $|\mathrm{tr}(H)| \le \|H\|_{\mathrm{tr}}$. If $|\mathrm{tr}(H)| = \|H\|_{\mathrm{tr}}$, then H is symmetric and is either positive semi-definite when $\mathrm{tr}(H) \ge 0$, or negative semi-definite when $\mathrm{tr}(H) \le 0$.*

Remark 2.1 As a corollary of Lemma 2.1, for any $H \in \mathbb{R}^{m\times m}$, if $H \not\succeq 0$, then $\mathrm{tr}(H) < \|H\|_{\mathrm{tr}}$. Now let $H = U\Sigma V^{\mathrm{T}}$ be the SVD of H [18], and set $Q = UV^{\mathrm{T}}$. Then $Q^{\mathrm{T}}H = V\Sigma V^{\mathrm{T}} \succeq 0$ and $\mathrm{tr}(Q^{\mathrm{T}}H) = \|H\|_{\mathrm{tr}} > \mathrm{tr}(H)$.

Lemma 2.2 *For $X \equiv [X_1, X_2 \ldots, X_\ell] \in \mathbb{O}^{n\times k}$ and $Y \equiv [Y_1, Y_2 \ldots, Y_\ell] \in \mathbb{O}^{n\times k}$, we have*

$$\sum_{j=1}^{\ell} \mathrm{tr}(Y_j^{\mathrm{T}}A_jX_j) \le \frac{1}{2}\sum_{j=1}^{\ell}\Big[\mathrm{tr}(Y_j^{\mathrm{T}}A_jY_j) + \mathrm{tr}(X_j^{\mathrm{T}}A_jX_j)\Big]. \tag{2.4}$$

Proof Inequality (2.4) is unlikely new. It is implied in [7] (for all $k_j = 1$). For completeness, we provide a proof here. It suffices to prove for each j,

$$\mathrm{tr}(Y_j^{\mathrm{T}}A_jX_j) \le \frac{1}{2}\Big[\mathrm{tr}(Y_j^{\mathrm{T}}A_jY_j) + \mathrm{tr}(X_j^{\mathrm{T}}A_jX_j)\Big]. \tag{2.5}$$

Recall $A_j \succeq 0$. Without loss of generality we may assume $A_j \succ 0$; otherwise we shift A_j to $A_j + \epsilon I_n \succ 0$ for some $\epsilon > 0$ and then letting $\epsilon \to 0$. To prove (2.5) for the case

$A_j \succ 0$, we note that $\mathrm{tr}(Y_j^{\mathrm{T}} A_j X_j)$ can be regarded as an inner-product on the matrix space $\mathbb{R}^{n\times k_j}$ between X_j and Y_j, and hence, by the Cauchy–Bunyakovsky-Schwarz inequality, we have

$$\mathrm{tr}(Y_j^{\mathrm{T}} A_j X_j) \le \sqrt{\mathrm{tr}(Y_j^{\mathrm{T}} A_j Y_j)}\sqrt{\mathrm{tr}(X_j^{\mathrm{T}} A_j X_j)} \le \frac{1}{2}\Big[\mathrm{tr}(Y_j^{\mathrm{T}} A_j Y_j) + \mathrm{tr}(X_j^{\mathrm{T}} A_j X_j)\Big],$$

as expected. □

Lemma 2.3 *Given $X \in \mathbb{O}^{n\times k}$, let $X^{\mathrm{T}}\mathscr{B}(X) = W\Sigma V^{\mathrm{T}}$ be its SVD, and let $\widetilde{X} = XWV^{\mathrm{T}} \in \mathbb{O}^{n\times k}$. Then*

$$f(\widetilde{X}) \ge f(X) + 2\Big[\|X^{\mathrm{T}}\mathscr{B}(X)\|_{\mathrm{tr}} - \mathrm{tr}(X^{\mathrm{T}}\mathscr{B}(X))\Big]. \tag{2.6}$$

Furthermore, $f(\widetilde{X}) > f(X)$, unless $X^{\mathrm{T}}\mathscr{B}(X) \succeq 0$, for which case $\widetilde{X} = X$.

Proof We note that

$$\widetilde{X}^{\mathrm{T}}\mathscr{B}(X) = VW^{\mathrm{T}}X^{\mathrm{T}}\mathscr{B}(X) = V\Sigma V^{\mathrm{T}} \succeq 0.$$

Hence $\mathrm{tr}(\widetilde{X}^{\mathrm{T}}\mathscr{B}(X)) = \|X^{\mathrm{T}}\mathscr{B}(X)\|_{\mathrm{tr}} \ge \mathrm{tr}(X^{\mathrm{T}}\mathscr{B}(X))$. We have, by Lemma 2.2,

$$\begin{aligned}
2\|X^{\mathrm{T}}\mathscr{B}(X)\|_{\mathrm{tr}} &= 2\,\mathrm{tr}(\widetilde{X}^{\mathrm{T}}\mathscr{B}(X)) \\
&= 2\sum_{j=1}^{\ell}\mathrm{tr}(\widetilde{X}_j^{\mathrm{T}} A_j X_j) + 2\,\mathrm{tr}(\widetilde{X}^{\mathrm{T}} D) \\
&\le \sum_{j=1}^{\ell}\mathrm{tr}(\widetilde{X}_j^{\mathrm{T}} A_j \widetilde{X}_j) + \sum_{j=1}^{\ell}\mathrm{tr}(X_j^{\mathrm{T}} A_j X_j) + 2\,\mathrm{tr}(\widetilde{X}^{\mathrm{T}} D) \\
&\le f(\widetilde{X}) + f(X) - 2\,\mathrm{tr}(X^{\mathrm{T}} D),
\end{aligned} \tag{2.7}$$

and also

$$2\,\mathrm{tr}(X^{\mathrm{T}}\mathscr{B}(X)) = 2\,\mathrm{tr}(X^{\mathrm{T}}\mathscr{A}(X)) + 2\,\mathrm{tr}(X^{\mathrm{T}} D) = 2f(X) - 2\,\mathrm{tr}(X^{\mathrm{T}} D). \tag{2.8}$$

Subtracting equation (2.8) from the inequality (2.7) yields (2.6).

By Lemma 2.1, $\|X^{\mathrm{T}}\mathscr{B}(X)\|_{\mathrm{tr}} - \mathrm{tr}(X^{\mathrm{T}}\mathscr{B}(X)) \ge 0$ always, and it is equal to 0 if and only if $X^{\mathrm{T}}\mathscr{B}(X) \succeq 0$, for which case $X^{\mathrm{T}}\mathscr{B}(X) = V\Sigma V^{\mathrm{T}}$ and hence $\widetilde{X} = X$. □

In the theorem below, we present some necessary conditions for a global maximizer beyond the KKT condition (2.3).

Theorem 2.1 *If $X_* = [X_{*1}, \ldots, X_{*\ell}] \in \mathbb{O}^{n\times k}$ is a global maximizer of (1.1), then*

$$\mathscr{B}(X_*) = X_*[X_*^{\mathrm{T}}\mathscr{B}(X_*)], \quad X_*^{\mathrm{T}}\mathscr{B}(X_*) \succeq 0, \quad X_{*j}^{\mathrm{T}} D_j \succeq 0 \text{ for } 1 \le j \le \ell. \tag{2.9}$$

Proof X_* must be a KKT point and hence it satisfies (2.3). We have to show $\Lambda_* := X_*^{\mathrm{T}}\mathscr{B}(X_*) \succeq 0$ and with all $X_{*j}^{\mathrm{T}} D_j \succeq 0$ for $1 \le j \le \ell$.

Suppose, to the contrary, that $X_{*i}^{\mathrm{T}} D_i \not\succeq 0$ for some i. Let $Q_i \in \mathbb{O}^{k_i \times k_i}$ such that $Q_i^{\mathrm{T}} X_{*i}^{\mathrm{T}} D_i \succeq 0$. We have

$$\mathrm{tr}(Q_i^{\mathrm{T}} X_{*i}^{\mathrm{T}} D_i) = \|X_{*i}^{\mathrm{T}} D_i\|_{\mathrm{tr}} > \mathrm{tr}(X_{*i}^{\mathrm{T}} D_i)$$

by Lemma 2.1 (see Remark 2.1). Hence $f(Y) > f(X_*)$ for $Y \equiv [Y_1, Y_2 \ldots, Y_\ell] \in \mathbb{O}^{n\times k}$ with $Y_j = X_{*j}$ for $j \neq i$ and $Y_i = X_{*i} Q_i$, a contradiction. Therefore $X_{*j}^{\mathrm{T}} D_j \succeq 0$ for $1 \le j \le \ell$.

Suppose, to the contrary, that $\Lambda_* \not\succeq 0$. Then there exists $Q \in \mathbb{O}^{k\times k}$ such that $Q^{\mathrm{T}}\Lambda_* \succeq 0$ and

$$\mathrm{tr}(X_*^{\mathrm{T}}\mathscr{B}(X_*)) = \mathrm{tr}(\Lambda_*) < \|\Lambda_*\|_{\mathrm{tr}} = \mathrm{tr}(Q^{\mathrm{T}}\Lambda_*) = \mathrm{tr}(Q^{\mathrm{T}} X_*^{\mathrm{T}}\mathscr{B}(X_*)).$$

Now use Lemma 2.3 to conclude $f(X_*Q) > f(X_*)$, contradicting that X_* is a global maximizer. □

As a result of this theorem, at a global maximizer X_*, $X_*\Lambda_*$ is a polar decomposition of $\mathscr{B}(X_*)$ and the decomposition is unique if $\mathrm{rank}(\mathscr{B}(X_*)) = k$ [30–32]. We call (2.3) with $\Lambda \succeq 0$ a *nonlinear polar decomposition* (NPD) of $\mathscr{B}(\cdot)$ which is a matrix-valued function of its orthogonal polar factor.

The necessary conditions in (2.9) demand much more on X_* than being just a KKT point of optimization problem (1.1), as given by (2.3), in that: 1) $\Lambda \succeq 0$, and 2) $X_j^{\mathrm{T}} D_j \succeq 0$ for $1 \le j \le \ell$. These extra conditions turn out to be powerful ones that numerical solutions by our later methods satisfy and compare favorably to existing general optimization methods on manifolds such as the solvers from `manopt` [9] and `STOP` [17, 54]. For example, as shown in Table 1, a better maximizer in terms of larger objective value is found by our methods than the ones obtained by solvers from `manopt` and `STOP`. Also in Remark 3.1(vi) later, we use a simple example to illustrate the importance of having $X_j^{\mathrm{T}} D_j \succeq 0$.

3 SCF for nonlinear polar decomposition

Theorem 2.1 provides us with important necessary conditions for a global maximizer of (1.1). In this section, we will first present a self-consistent-field (SCF) iteration for solving (2.3) with post-processing to maximally increase the objective function value and then perform a convergence analysis of the method. For that purpose, we first investigate how to drive up the objective value of (1.1) and by at least how much.

Lemma 3.1 *For $X,\ Y \in \mathbb{O}^{n\times k}$, we have*

$$f(Y) \ge f(X) + 2\eta(Y), \tag{3.1}$$

where

$$\eta(Y) := \mathrm{tr}(Y^{\mathrm{T}}\mathscr{B}(X)) - \mathrm{tr}(X^{\mathrm{T}}\mathscr{B}(X)). \tag{3.2}$$

In particular, $f(Y) \geq f(X)$ *if* $\eta(Y) \geq 0$.

Proof Recall $\mathscr{B}(X) = \mathscr{A}(X) + D$. We have, by Lemma 2.2,

$$\begin{aligned} \operatorname{tr}(X^{\mathrm{T}}\mathscr{B}(X)) &= \operatorname{tr}(X^{\mathrm{T}}D) + \operatorname{tr}(X^{\mathrm{T}}\mathscr{A}(X)), \qquad (3.3)\\ \operatorname{tr}(Y^{\mathrm{T}}\mathscr{B}(X)) &= \operatorname{tr}(Y^{\mathrm{T}}D) + \operatorname{tr}(Y^{\mathrm{T}}\mathscr{A}(X))\\ &= \operatorname{tr}(Y^{\mathrm{T}}D) + \sum_{j=1}^{\ell} \operatorname{tr}(Y_j^{\mathrm{T}}A_jX_j)\\ &\leq \operatorname{tr}(Y^{\mathrm{T}}D) + \frac{1}{2}\sum_{j=1}^{\ell}\Big[\operatorname{tr}(Y_j^{\mathrm{T}}A_jY_j) + \operatorname{tr}(X_j^{\mathrm{T}}A_jX_j)\Big]\\ &= \operatorname{tr}(Y^{\mathrm{T}}D) + \frac{1}{2}\operatorname{tr}(Y^{\mathrm{T}}\mathscr{A}(X)) + \frac{1}{2}\operatorname{tr}(X^{\mathrm{T}}\mathscr{A}(X)). \qquad (3.4) \end{aligned}$$

Hence by (3.2)

$$\begin{aligned} \operatorname{tr}(Y^{\mathrm{T}}D) + \frac{1}{2}\operatorname{tr}(Y^{\mathrm{T}}\mathscr{A}(X)) + \frac{1}{2}\operatorname{tr}(X^{\mathrm{T}}\mathscr{A}(X)) &\geq \operatorname{tr}(X^{\mathrm{T}}\mathscr{B}(X)) + \eta(Y)\\ &= \operatorname{tr}(X^{\mathrm{T}}D) + \operatorname{tr}(X^{\mathrm{T}}\mathscr{A}(X)) + \eta(Y), \end{aligned}$$

which yields (3.1) after rearrangement and upon noting (1.2a). □

What this theorem says is that to increase the objective value $f(X)$, we will need to find some $Y \in \mathbb{O}^{n\times k}$ such that $\eta(Y) > 0$, but how?

For any $Y \in \mathbb{O}^{n\times k}$, let $Y_\perp \in \mathbb{O}^{n\times(n-k)}$ such that $[Y, Y_\perp]$ is orthogonal. Then

$$[Y, Y_\perp]^{\mathrm{T}}\mathscr{B}(X) = \begin{bmatrix} Y^{\mathrm{T}}\mathscr{B}(X) \\ Y_\perp^{\mathrm{T}}\mathscr{B}(X) \end{bmatrix} =: \begin{bmatrix} B_1(Y) \\ B_2(Y) \end{bmatrix}.$$

Hence by Lemma 2.1

$$\begin{aligned} \operatorname{tr}(Y^{\mathrm{T}}\mathscr{B}(X)) = \operatorname{tr}(B_1(Y)) &\leq \sum_{i=1}^{k}\sigma_i(B_1(Y)) \qquad (3.5)\\ &\leq \sum_{i=1}^{k}\sigma_i([Y, Y_\perp]^{\mathrm{T}}\mathscr{B}(X)) \qquad (3.6)\\ &= \sum_{i=1}^{k}\sigma_i(\mathscr{B}(X)) = \|\mathscr{B}(X)\|_{\mathrm{tr}}, \qquad (3.7) \end{aligned}$$

where (3.5) is an equality if and only if $B_1(Y) = Y^{\mathrm{T}}\mathscr{B}(X) \succeq 0$, and (3.6) is an equality if and only if $B_2(Y) = Y_\perp^{\mathrm{T}}\mathscr{B}(X) = 0$. As a corollary of (3.5), (3.6), and (3.7), we know $\operatorname{tr}(X^{\mathrm{T}}\mathscr{B}(X)) \leq \|\mathscr{B}(X)\|_{\mathrm{tr}}$ upon letting $Y = X$, and also the largest $\eta(Y)$ in (3.2) is $\|\mathscr{B}(X)\|_{\mathrm{tr}} - \operatorname{tr}(X^{\mathrm{T}}\mathscr{B}(X)) \geq 0$, achieved by any $Y \in \mathbb{O}^{n\times k}$ such that

$\mathcal{R}(\mathscr{B}(X)) \subseteq \mathcal{R}(Y)$ (which ensures $B_2(Y) = Y_\perp^{\mathrm{T}} \mathscr{B}(X) = 0$) and $Y^{\mathrm{T}} \mathscr{B}(X) \succeq 0$. One such Y is given by $Y = UV^{\mathrm{T}}$, where $\mathscr{B}(X) = U\Sigma V^{\mathrm{T}}$ is the SVD of $\mathscr{B}(X)$. Hence, we have the following lemma.

Lemma 3.2 *Let $\eta(Y)$ be defined as in Lemma 3.1. Then*

$$\max_{Y \in \mathbb{O}^{n\times k}} \eta(Y) = \|\mathscr{B}(X)\|_{\mathrm{tr}} - \mathrm{tr}(X^{\mathrm{T}} \mathscr{B}(X)) \geq 0,$$

and the maximum is attained by $Y_{\mathrm{opt}} = UV^{\mathrm{T}}$, where U and V come from the SVD $\mathscr{B}(X) = U\Sigma V^{\mathrm{T}}$. Furthermore the maximum is strictly positive, unless $X^{\mathrm{T}}\mathscr{B}(X) \succeq 0$ and $\mathcal{R}(\mathscr{B}(X)) \subseteq \mathcal{R}(X)$, under which it is 0.

3.1 SCF iteration

Lemmas 3.1 and 3.2 point to a way to maximally increase the objective function value at a point $X \in \mathbb{O}^{n\times k}$. Namely, compute SVD $\mathscr{B}(X) = U\Sigma V^{\mathrm{T}}$ and let $Y = UV^{\mathrm{T}}$. Then we have

$$\mathrm{tr}(Y^{\mathrm{T}}\mathscr{B}(X)) = \mathrm{tr}(\Sigma) = \|\mathscr{B}(X)\|_{\mathrm{tr}} \geq \mathrm{tr}(X^{\mathrm{T}}\mathscr{B}(X)),$$

where the last inequality is strict if $X^{\mathrm{T}}\mathscr{B}(X) \not\succeq 0$ or if $\mathcal{R}(\mathscr{B}(X)) \not\subseteq \mathcal{R}(X)$. As a result, we have $\eta(Y) \geq 0$ in (3.1) and (3.2). A further improvement on $Y \equiv [Y_1, Y_2, \ldots, Y_\ell]$, made possible by Lemma 2.1 (see Remark 2.1), is as follows: Let $Y_j^{\mathrm{T}} D_j = U_j \Sigma_j V_j^{\mathrm{T}}$ be the SVD of $Y_j^{\mathrm{T}} D_j$, and set $X^{(\mathrm{new})} \equiv [X_1^{(\mathrm{new})}, X_2^{(\mathrm{new})}, \ldots, X_\ell^{(\mathrm{new})}] \in \mathbb{O}^{n\times k}$ with $X_j^{(\mathrm{new})} = Y_j(U_j V_j^{\mathrm{T}})$ for $1 \leq j \leq \ell$. We will then have

$$f(X^{(\mathrm{new})}) = f(Y) + 2\sum_{j=1}^{\ell} \Big[\|Y_j^{\mathrm{T}} D_j\|_{\mathrm{tr}} - \mathrm{tr}(Y_j^{\mathrm{T}} D_j)\Big] \tag{3.8}$$

$$\geq f(X) + 2\Big[\|\mathscr{B}(X)\|_{\mathrm{tr}} - \mathrm{tr}(X^{\mathrm{T}}\mathscr{B}(X))\Big] + 2\sum_{j=1}^{\ell} \Big[\|Y_j^{\mathrm{T}} D_j\|_{\mathrm{tr}} - \mathrm{tr}(Y_j^{\mathrm{T}} D_j)\Big]. \tag{3.9}$$

The idea we discussed so far is similar to [7], where there are no D_j-terms. Formally, we summarize these into Algorithm 1, whose overall complexity is[1] $O(n^2k + nk^2 + k^3) = O(n^2k)$ per SCF iterative step.

Remark 3.1 Regarding Algorithm 1, there are a few comments in order.

(i) The purpose of Line 1 is to make sure $(X_j^{(0)})^{\mathrm{T}} D_j \succeq 0$ initially. Thus it can be dropped if it is known $(X_j^{(0)})^{\mathrm{T}} D_j \succeq 0$ already upon entry. With Lines 5 and 6, we always make sure $(X_j^{(i)})^{\mathrm{T}} D_j \succeq 0$ at the end of each for-loop.

[1] Computing the SVD of an $n \times k$ matrix takes $6nk^2 + 20k^3$ flops [18, p.493].

Algorithm 1 NPDvSCF: NPD (2.3) solved by SCF

Input: $A_j \succeq 0$ and D_j for $1 \le j \le \ell$, and $X^{(0)} \equiv [X_1^{(0)}, X_2^{(0)}, \ldots, X_\ell^{(0)}] \in \mathbb{O}^{n\times k}$;
Output: a maximizer of (1.1) by solving NPD (2.3).
1: for $1 \le j \le \ell$, compute SVD $(X_j^{(0)})^{\mathrm{T}} D_j = U_j \Sigma_j V_j^{\mathrm{T}}$ and reset $X_j^{(0)}$ to $X_j^{(0)}(U_j V_j^{\mathrm{T}})$;
2: **for** $i = 0, 1, \ldots$ until convergence **do**
3: compute $B_i = \mathscr{B}(X^{(i)}) \in \mathbb{R}^{n\times k}$ and its SVD $B_i = U_i \Sigma_i V_i^{\mathrm{T}}$;
4: $Y^{(i)} \equiv [Y_1^{(i)}, Y_2^{(i)}, \ldots, Y_\ell^{(i)}] = U_i V_i^{\mathrm{T}} \in \mathbb{O}^{n\times k}$;
5: compute SVD $(Y_j^{(i)})^{\mathrm{T}} D_j = W_j \Sigma_j Q_j^{\mathrm{T}}$ for $1 \le j \le \ell$;
6: compute $X^{(i+1)} \equiv [X_1^{(i+1)}, X_2^{(i+1)}, \ldots, X_\ell^{(i+1)}]$ with $X_j^{(i+1)} = Y_j^{(i)}(W_j Q_j^{\mathrm{T}})$ for $1 \le j \le \ell$;
7: **end for**
8: **return** the last $X^{(i)}$.

(ii) For each for-loop, (3.8) and (3.9) hold with $X = X^{(i)}$, $Y = Y^{(i)}$, and $X^{(\mathrm{new})} = X^{(i+1)}$.

(iii) Lines 5 and 6 are to ensure that one of the necessary conditions: $X_{*j}^{\mathrm{T}} D_j \succeq 0$ for $1 \le j \le \ell$ is satisfied at convergence.

(iv) A reasonable stopping criterion at Line 2 is

$$\varepsilon_{\mathrm{KKT}} + \varepsilon_{\mathrm{sym}} := \frac{\|\mathscr{B}(X) - X[X^{\mathrm{T}}\mathscr{B}(X)]\|_{\mathrm{F}}}{\beta} + \frac{\|[X^{\mathrm{T}}\mathscr{B}(X)] - [X^{\mathrm{T}}\mathscr{B}(X)]^{\mathrm{T}}\|_{\mathrm{F}}}{\beta} \le \epsilon, \tag{3.10a}$$

where ϵ is a given tolerance, and

$$\beta = \sum_{j=1}^{\ell} \Big(\|A_j\|_{\mathrm{F}} + 2\|D_j\|_{\mathrm{F}} \Big). \tag{3.10b}$$

The significance of both $\varepsilon_{\mathrm{KKT}}$ and $\varepsilon_{\mathrm{sym}}$ is rather self-explanatory. In fact, we will call $\varepsilon_{\mathrm{KKT}}$ and $\varepsilon_{\mathrm{sym}}$ the *normalized residual for the KKT equation* (2.3a) and the *normalized residual for the symmetry equation* (2.3b) in $\Lambda = X^{\mathrm{T}}\mathscr{B}(X)$, respectively.

(v) It is interesting to look at whether $f(X^{(i_0+1)}) = f(X^{(i_0)})$ can possibly happen for some i_0 in Algorithm 1. By the inequality in (3.9), if $f(X^{(i_0+1)}) = f(X^{(i_0)})$ happens, then $X_* := X^{(i_0)}$ satisfies (2.9), i.e., $X^{(i_0)}$ is a KKT point and satisfies the neceaary coniditons for a global maximizer in Theorem 2.1, and hence $X^{(i_0)}$ should be and is usually accepted as a solution to (1.1). This in general is too good a case and almost never happens in actual computations.

(vi) Algorithm 1, taken out Lines 5 and 6 and then applied to the special case (1.4), appeared in [35, 39]. However, Lines 5 and 6 can be critical in steering away from some stationary points that are not global maximizers. Here is a simple example:

$$A = \begin{bmatrix} 4 & 0 \\ 0 & 3 \end{bmatrix}, \quad D = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}.$$

For any $X \in \mathbb{O}^{2\times 2}$, it can be seen that $\operatorname{tr}(X^{\mathrm{T}}AX) = 7$ and $\operatorname{tr}(X^{\mathrm{T}}D) \le \|D\|_{\mathrm{tr}} = 2$, attainable at $X = -I_2$, and hence

$$\max_{X\in\mathbb{O}^{2\times 2}} \operatorname{tr}(X^{\mathrm{T}}AX) + 2\operatorname{tr}(X^{\mathrm{T}}D) = 7 + 4 = 11$$

with a global maximizer $X_{\mathrm{opt}} = -I_2$. However, if with $X^{(0)} = I_2$, Algorithm 1, without Lines 5 and 6 (i.e., simply let $X^{(i+1)} = Y^{(i)}$), will find that $B_0 = \operatorname{diag}(3, 2)$, $Y^{(0)} = I_2$ and hence $X^{(1)} = I_2$. Inductively, $X^{(i)} = X^{(0)}$ for $i \ge 1$ and

$$\operatorname{tr}((X^{(i)})^{\mathrm{T}}AX^{(i)}) + 2\operatorname{tr}((X^{(i)})^{\mathrm{T}}D) = 7 - 4 = 3.$$

On the other hand, with Lines 5 and 6, i.e., Algorithm 1 as stated, will find $X^{(1)} = -I_2$ and $X^{(i)} = -I_2$ thereafter, reaching the global maximizer in just one iterative step! Note that for both I_2 and $-I_2$,

$$\begin{aligned} A \cdot I_2 + D = I_2 \cdot \Lambda \equiv I_2 \cdot \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}, \quad A \cdot (-I_2) + D \\ = (-I_2) \cdot \Lambda \equiv (-I_2) \cdot \begin{bmatrix} 5 & 0 \\ 0 & 4 \end{bmatrix} \end{aligned}$$

satisfying the KKT condition (2.3), and the necessary condition $X_*^{\mathrm{T}}\mathscr{B}(X_*) \succeq 0$ in Theorem 2.1, except that the last necessary condition $X_*^{\mathrm{T}}D \succeq 0$ is violated by I_2: $I_2^{\mathrm{T}} \cdot D \prec 0$, but satisfied by $-I_2$: $(-I_2)^{\mathrm{T}} \cdot D \succ 0$. Hence Lines 5 and 6 of Algorithm 1 can be critical, not to mention that the objective value gains from them, as shown in (3.8).

3.2 Convergence analysis

For the first special case (1.3), Algorithm 1 is equivalent to subspace iteration (also known as the power method). It converges to the eigenspace of A associated with its k largest eigenvalues if $A \succeq 0$, and even if A is indefinite, it still converges but to the eigenspace of A associated with its k largest eigenvalues in magnitude [13, 49].

In this subsection, we will perform a convergence analysis on Algorithm 1 in its generality under the assumption (1.1b). In view of Remark 3.1(v), we will exclude the case $f(X^{(i_0+1)}) = f(X^{(i_0)})$ for some i_0 from consideration.

Theorem 3.1 *Let the sequence $\{X^{(i)}\}_{i=0}^{\infty}$ be generated by Algorithm 1, and let $Y^{(i)}$ be an orthogonal polar factor of $\mathscr{B}(X^{(i)})$ as defined in the algorithm. Suppose that $f(X^{(i+1)}) \ne f(X^{(i)})$ for all $i \ge 0$. The following statements hold.*

(a) *For* $i \geq 0$,

$$f(X^{(i+1)}) - f(X^{(i)}) \geq 2\Big[\big\|\mathscr{B}(X^{(i)})\big\|_{\mathrm{tr}} - \mathrm{tr}((X^{(i)})^{\mathrm{T}}\mathscr{B}(X^{(i)}))\Big] + 2\sum_{j=1}^{\ell}\Big[\|(Y_j^{(i)})^{\mathrm{T}}D_j\|_{\mathrm{tr}} - \mathrm{tr}((Y_j^{(i)})^{\mathrm{T}}D_j)\Big] \geq 0. \tag{3.11}$$

The right-hand side of (3.11) *is strictly positive if*

$$\mathcal{R}(\mathscr{B}(X^{(i)})) \nsubseteq \mathcal{R}(X^{(i)}),\ \text{or}\ (X^{(i)})^{\mathrm{T}}\mathscr{B}(X^{(i)}) \ngeq 0,\ \text{or}\ (Y_j^{(i)})^{\mathrm{T}}D_j \ngeq 0\ \text{for some}\ j. \tag{3.12}$$

(b) *The sequence* $\{f(X^{(i)})\}_{i=0}^{\infty}$ *is monotonically increasing and convergent.*

(c) *If*

$$\mathrm{rank}(\mathscr{B}(X^{(i)})) = k,\ \text{and}\ \mathrm{rank}((Y_j^{(i)})^{\mathrm{T}}D_j) = k_j\ \text{for}\ 1 \leq j \leq \ell, \tag{3.13}$$

then $X^{(i+1)}$ *is uniquely determined by the procedure of Algorithm* 1.

(d) *Any accumulation point* X_* *of the sequence* $\{X^{(i)}\}_{i=0}^{\infty}$ *satisfies the necessary conditions in* (2.9) *for a global maximizer.*

Proof Item (a) is a corollary of (3.9). Hence the sequence $\{f(X^{(i)})\}_{i=0}^{\infty}$ is monotonically increasing. It is also bounded from above because $f(X) \leq \sum_{j=1}^{\ell}\big[\|A_j\|_{\mathrm{tr}} + \|D_j\|_{\mathrm{tr}}\big]$ for any $X \in \mathbb{O}^{n\times k}$. Therefore the sequence $\{f(X^{(i)})\}_{i=0}^{\infty}$ is convergent. This proves item (b).

For item (c), since $\mathrm{rank}(\mathscr{B}(X^{(i)})) = k$, $Y^{(i)}$ at Line 4 of Algorithm 1 as the orthogonal polar factor of $\mathscr{B}(X^{(i)})$ is unique [30]. For the same reason, $W_j Q_j^{\mathrm{T}}$ at Line 6 as the orthogonal polar factor of $(Y_j^{(i)})^{\mathrm{T}}D_j$ is unique. Hence $X^{(i+1)}$ is uniquely determined.

We now prove item (d). There is a subsequence $\{X^{(i)}\}_{i\in\mathbb{I}}$ that converges to X_*, i.e.,

$$\lim_{\mathbb{I}\ni i\to\infty} \|X^{(i)} - X_*\|_{\mathrm{F}} = 0, \tag{3.14}$$

where $\mathbb{I}$ is an infinite subset of $\{1, 2, \ldots\}$. Since $(X_j^{(i)})^{\mathrm{T}}D_j \succeq 0$ for $1 \leq j \leq \ell$ and $i \geq 0$, we conclude that $X_{*j}^{\mathrm{T}}D_j \succeq 0$ for $1 \leq j \leq \ell$ by letting $\mathbb{I} \ni i \to \infty$. It remains to show $\mathscr{B}(X_*) = X_*\Lambda_*$ and $X_*^{\mathrm{T}}\mathscr{B}(X_*) \succeq 0$, or, equivalently, $\mathcal{R}(\mathscr{B}(X_*)) \subseteq \mathcal{R}(X_*)$ and $X_*^{\mathrm{T}}\mathscr{B}(X_*) \succeq 0$. Assume, to the contrary, that either $\mathcal{R}(\mathscr{B}(X_*)) \nsubseteq \mathcal{R}(X_*)$ or $X_*^{\mathrm{T}}\mathscr{B}(X_*) \ngeq 0$ (or both). Then

$$\begin{aligned}\delta &:= \|\mathscr{B}(X_*)\|_{\mathrm{tr}} - \mathrm{tr}(X_*^{\mathrm{T}}\mathscr{B}(X_*))\\ &= \big(\|\mathscr{B}(X_*)\|_{\mathrm{tr}} - \|X_*^{\mathrm{T}}\mathscr{B}(X_*)\|_{\mathrm{tr}}\big) + \big(\|X_*^{\mathrm{T}}\mathscr{B}(X_*)\|_{\mathrm{tr}} - \mathrm{tr}(X_*^{\mathrm{T}}\mathscr{B}(X_*))\big)\\ &> 0,\end{aligned}$$

because $\|\mathscr{B}(X_*)\|_{\rm tr} - \|X_*^{\rm T}\mathscr{B}(X_*)\|_{\rm tr} > 0$ if $\mathcal{R}(\mathscr{B}(X_*)) \not\subseteq \mathcal{R}(X_*)$, or $\|X_*^{\rm T}\mathscr{B}(X_*)\|_{\rm tr} - {\rm tr}(X_*^{\rm T}\mathscr{B}(X_*)) > 0$ if $X_*^{\rm T}\mathscr{B}(X_*) \not\succeq 0$. Since $\|\mathscr{B}(X)\|_{\rm tr}$, ${\rm tr}(X^{\rm T}\mathscr{B}(X))$, and $f(X)$ are continuous in $X \in \mathbb{O}^{n\times k}$, it follows from (3.14) that there is an $i_0 \in \mathbb{I}$ such that

$$\left|\|\mathscr{B}(X_*)\|_{\rm tr} - \|\mathscr{B}(X^{(i_0)})\|_{\rm tr}\right| < \delta/3, \tag{3.15a}$$

$$\left|{\rm tr}((X^{(i_0)})^{\rm T}\mathscr{B}(X^{(i_0)})) - {\rm tr}(X_*^{\rm T}\mathscr{B}(X_*))\right| < \delta/3, \tag{3.15b}$$

$$f(X_*) - \delta/3 < f(X^{(i_0)}) \le f(X_*). \tag{3.15c}$$

Using (3.11) and (3.15), we have

$$\begin{aligned} f(X^{(i_0+1)}) &\ge f(X^{(i_0)}) + 2\Big[\|\mathscr{B}(X^{(i_0)})\|_{\rm tr} - {\rm tr}((X^{(i_0)})^{\rm T}\mathscr{B}(X^{(i_0)}))\Big] \\ &> f(X_*) - \frac{\delta}{3} + 2\Big[\|\mathscr{B}(X_*)\|_{\rm tr} - \frac{\delta}{3} - {\rm tr}(X_*^{\rm T}\mathscr{B}(X_*)) - \frac{\delta}{3}\Big] \\ &= f(X_*) + \frac{\delta}{3} > f(X_*), \end{aligned}$$

contradicting $f(X^{(i)}) \le \lim_{j\to\infty} f(X^{(j)}) = f(X_*)$ for all i. □

It is interesting to check what happens if

$$\mathcal{R}(\mathscr{B}(X^{(i)})) \subseteq \mathcal{R}(X^{(i)}),\ \ (X^{(i)})^{\rm T}\mathscr{B}(X^{(i)}) \succ 0,\ \ (X_j^{(i)})^{\rm T}D_j \succ 0 \text{ for } 1 \le j \le \ell, \tag{3.16}$$

which is not exactly the opposite of (3.12), however. The first two conditions in (3.16) imply that $\mathscr{B}(X^{(i)}) = X^{(i)}[(X^{(i)})^{\rm T}\mathscr{B}(X^{(i)})]$ is a polar decomposition of $\mathscr{B}(X^{(i)})$, and the decomposition is also unique [30] because of ${\rm rank}(\mathscr{B}(X^{(i)})) = k$. Hence $Y^{(i)} = X^{(i)}$. By the last condition in (3.16), we conclude that $W_j Q_j^{\rm T} = I_{k_j}$ for all j at Line 6, implying $X^{(i+1)} = X^{(i)}$. By induction, $X^{(t)} = X^{(i)}$ for all $t \ge i+1$, implying $f(X^{(i+1)}) = f(X^{(i)})$. Hence the case (3.16) cannot happen under the assumption $f(X^{(i+1)}) \ne f(X^{(i)})$ for all $i \ge 0$ in Theorem 3.1.

As a corollary of Theorem 3.1, we establish a sufficient condition for NPD (2.3) to have a solution.

Corollary 3.1 *Under the condition* (1.1b)*, NPD* (2.3) *is solvable, i.e., there exists* $X \in \mathbb{O}^{n\times k}$ *such that* $\Lambda = X^{\rm T}\mathscr{B}(X) \succeq 0$ *and* (2.3) *holds.*

To further analyze the convergence of the sequence $\{X^{(i)}\}_{i=0}^{\infty}$, we now introduce a distance measure on Grassmann manifold $\mathscr{G}_k(\mathbb{R}^n)$, the collection of all k-dimensional subspaces in $\mathbb{R}^n$. Let $\mathcal{X} = \mathcal{R}(X)$ and $\mathcal{Y} = \mathcal{R}(Y)$ be two points in $\mathscr{G}_k(\mathbb{R}^n)$, where $X,\ Y \in \mathbb{O}^{n\times k}$. The canonical angles $\theta_1(\mathcal{X},\mathcal{Y}) \ge \cdots \ge \theta_k(\mathcal{X},\mathcal{Y})$ between $\mathcal{X}$ and $\mathcal{Y}$ are defined by

$$0 \le \theta_i(\mathcal{X},\mathcal{Y}) := \arccos\sigma_i(X^{\rm T}Y) \le \frac{\pi}{2} \quad \text{for } 1 \le i \le k,$$

and accordingly, $\Theta(\mathcal{X}, \mathcal{Y}) = \text{diag}(\theta_1(\mathcal{X}, \mathcal{Y}), \ldots, \theta_k(\mathcal{X}, \mathcal{Y}))$. It is known that

$$\text{dist}_2(\mathcal{X}, \mathcal{Y}) := \| \sin \Theta(\mathcal{X}, \mathcal{Y}) \|_2 \tag{3.17}$$

is a unitarily invariant metric on Grassmann manifold $\mathcal{G}_k(\mathbb{R}^n)$ [50, p.99].

The following lemma is an equivalent restatement of [38, Lemma 4.10] (see also [25, Proposition 7]) in the context of a metric space.

Lemma 3.3 ([38, Lemma 4.10]) *Let $\mathcal{G}$ be a metric space with metric* dist$(\cdot, \cdot)$*, and let $\{\boldsymbol{x}_i\}_{i=0}^{\infty}$ be a sequence in $\mathcal{G}$. If $\boldsymbol{x}_* \in \mathcal{G}$ is an isolated accumulation point of the sequence such that, for every subsequence $\{\boldsymbol{x}_i\}_{i\in\mathbb{I}}$ converging to $\boldsymbol{x}_*$, there is an infinite subset $\widehat{\mathbb{I}} \subseteq \mathbb{I}$ satisfying* dist$(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \to 0$ *as $\widehat{\mathbb{I}} \ni i \to \infty$, then the entire sequence $\{\boldsymbol{x}_i\}_{i=0}^{\infty}$ converges to $\boldsymbol{x}_*$.*

In applying this lemma, on Grassmann manifold $\mathcal{G}_k(\mathbb{R}^n)$, we will use the unitarily invariant metric in (3.17), and, on matrix space $\mathbb{R}^{n\times k}$, we will use $\|X - Y\|_2$ as the metric.

Theorem 3.2 *Let the sequence $\{X^{(i)}\}_{i=0}^{\infty}$ be generated by Algorithm 1, and let X_* be an accumulation point of the sequence.*

(a) *$\mathcal{R}(X_*)$ is an accumulation point of the sequence $\{\mathcal{R}(X^{(i)})\}_{i=0}^{\infty}$.*
(b) *If $\mathcal{R}(X_*)$ is an isolated accumulation point of $\{\mathcal{R}(X^{(i)})\}_{i=0}^{\infty}$ and if* rank$(\mathscr{B}(X_*)) = k$*, then the sequence $\{\mathcal{R}(X^{(i)})\}_{i=0}^{\infty}$ converges to $\mathcal{R}(X_*)$.*
(c) *If X_* is an isolated accumulation point of $\{X^{(i)}\}_{i=0}^{\infty}$ and if* rank$(\mathscr{B}(X_*)) = k$ *and* rank$(X_{*j}^{\mathrm{T}} D_j) = k_j$ *for $1 \le j \le \ell$, then the sequence $\{X^{(i)}\}_{i=0}^{\infty}$ converges to X_*.*

Proof Let $\{X^{(i)}\}_{i\in\mathbb{I}}$ be a subsequence that converges to X_*. Then it can be seen that [50, pp.125-127]

$$0 \le \text{dist}_2(\mathcal{R}(X^{(i)}), \mathcal{R}(X_*)) \le \|X^{(i)} - X_*\|_2 \to 0 \text{ as } \mathbb{I} \ni i \to \infty, \tag{3.18}$$

i.e., $\mathcal{R}(X_*)$ is an accumulation point of the sequence $\{\mathcal{R}(X^{(i)})\}_{i=0}^{\infty}$. This proves item (a).

Recall the associated sequence $\{Y^{(i)}\}_{i=0}^{\infty}$. Consider $\{Y^{(i)}\}_{i\in\mathbb{I}}$ for which there is a convergent subsequence $\{Y^{(i)}\}_{i\in\widehat{\mathbb{I}}}$, converging to Y_*, where $\widehat{\mathbb{I}} \subseteq \mathbb{I}$. Further, $\{X^{(i+1)}\}_{i\in\widehat{\mathbb{I}}}$ has a convergent subsequence $\{X^{(i+1)}\}_{i\in\widetilde{\mathbb{I}}}$, converging to $\widetilde{X}_*$, where $\widetilde{\mathbb{I}} \subseteq \widehat{\mathbb{I}} \subseteq \mathbb{I}$. In particular,

$$\lim_{\widetilde{\mathbb{I}}\ni i\to\infty} \text{dist}_2(\mathcal{R}(X^{(i+1)}, \mathcal{R}(\widetilde{X}_*)) = 0. \tag{3.19}$$

We claim that

$$Y_*^{\mathrm{T}} \mathscr{B}(X_*) \succeq 0, \quad \text{tr}(Y_*^{\mathrm{T}} \mathscr{B}(X_*)) = \|\mathscr{B}(X_*)\|_{\text{tr}}, \tag{3.20a}$$

$$\mathcal{R}(\widetilde{X}_*) = \mathcal{R}(Y_*). \tag{3.20b}$$

Since $\text{tr}((Y^{(i)})^{\mathrm{T}} \mathscr{B}(X^{(i)})) = \|\mathscr{B}(X^{(i)}))\|_{\text{tr}}$ and $(Y^{(i)})^{\mathrm{T}} \mathscr{B}(X^{(i)}) \succeq 0$, letting $\widehat{\mathbb{I}} \ni i \to \infty$ yields (3.20a). Since $\mathcal{R}(X^{(i+1)}) = \mathcal{R}(Y^{(i)})$, letting $\widetilde{\mathbb{I}} \ni i \to \infty$ yields (3.20b).

It follows from (3.20a) that $\mathscr{B}(X_*) = Y_*(Y_*^{\mathrm{T}}\mathscr{B}(X_*))$ is a polar decomposition of $\mathscr{B}(X_*)$. Next, by Theorem 3.1(d), we know $\mathscr{B}(X_*) = X_*[X_*^{\mathrm{T}}\mathscr{B}(X_*)]$ and $X_*^{\mathrm{T}}\mathscr{B}(X_*) \succeq 0$. This also gives a polar decomposition of $\mathscr{B}(X_*)$. Because $\mathrm{rank}(\mathscr{B}(X_*)) = k$, $\mathscr{B}(X_*)$ has a unique polar decomposition [30] and hence $X_* = Y_*$, which, together with (3.20b), lead to $\mathcal{R}(X_*) = \mathcal{R}(\widetilde{X}_*)$. Therefore, as $\widetilde{\mathbb{I}} \ni i \to \infty$, by (3.19) we have

$$\mathrm{dist}_2(\mathcal{R}(X^{(i)}, \mathcal{R}(X^{(i+1)}) \le \mathrm{dist}_2(\mathcal{R}(X^{(i)}, \mathcal{R}(X_*)) + \mathrm{dist}_2(\mathcal{R}(\widetilde{X}_*), \mathcal{R}(X^{(i+1)}) \to 0.$$

Now use Lemma 3.3 to conclude that the entire sequence $\{\mathcal{R}(X^{(i)})\}_{i=0}^{\infty}$ converges to $\mathcal{R}(X_*)$. This completes the proof of item (b).

Consider now item (c). All the arguments in the proof above for item (b) remain valid as the condition that $\mathcal{R}(X_*)$ is an isolated accumulation point of $\{\mathcal{R}(X^{(i)})\}_{i=0}^{\infty}$ is only used at the very end when Lemma 3.3 is invoked. For the current case, it suffices to prove $\widetilde{X}_* = X_*$. By design, $X_j^{(i+1)} = Y_j^{(i)} Z_j^{(i)}$ where $Z_j^{(i)} \in \mathbb{O}^{k_j \times k_j}$ is the orthogonal polar factor of $(Y_j^{(i)})^{\mathrm{T}} D_j$. By assumption $\mathrm{rank}(X_{*j}^{\mathrm{T}} D_j) = k_j$ and hence $X_{*j}^{\mathrm{T}} D_j \succ 0$. Note that

$$(Y_j^{(i)})^{\mathrm{T}} D_j \to Y_{*j}^{\mathrm{T}} D_j = X_{*j}^{\mathrm{T}} D_j \text{ as } \widetilde{\mathbb{I}} \ni i \to \infty,$$

since $Y_* = X_*$ has already been proved. By the continuity of the orthogonal polar factor [30, 31], the orthogonal polar factor $Z_j^{(i)}$ of $(Y_j^{(i)})^{\mathrm{T}} D_j$ converges to the one of $X_{*j}^{\mathrm{T}} D_j \succ 0$, i.e., I_{k_j} as $\widetilde{\mathbb{I}} \ni i \to \infty$. Hence taking limit on $X_j^{(i+1)} = Y_j^{(i)} Z_j^{(i)}$ as $\widetilde{\mathbb{I}} \ni i \to \infty$ yields $\widetilde{X}_{*j} = Y_{*j}$ which is the same as X_{*j} for $1 \le j \le \ell$, i.e., $\widetilde{X}_* = X_*$, as expected. □

Item (a) of the theorem is very much a trivial conclusion, but it is remarkable that each of item (b) and item (c) starts with an isolated accumulation point and ends up with the conclusion that the isolated accumulation point is the limit of the entire sequence. Still there is a technical detail to address, namely how to verify that an accumulation point is an isolated accumulation point in a simple manner. At this point, we do not have an answer to it. In item (c), it is also assumed that each $X_{*j}^{\mathrm{T}} D_j \succ 0$, which fails when $\mathrm{rank}(D_j) < k_j$. In particular, if $D_j = 0$ then the objective value $f(X_*)$ is invariant with respect to changing X_{*j} to $X_{*j} Q_j$ for any $Q_j \in \mathbb{O}^{k_j \times k_j}$, suggesting that in practice it is unlikely the sequence $\{X^{(i)}\}_{i=0}^{\infty}$ converges in $\mathbb{R}^{n \times k}$ but $\{\mathcal{R}(X^{(i)})\}_{i=0}^{\infty}$ may as stated in item (b). These questions warrant continuing investigations.

Remark 3.2 Theorem 3.2 essentially says that Algorithm 1 always converges, but it says little quantitatively on its rate of convergence. Our rough discussion in what follows provides some insights, but further investigation is needed. Using perturbation results on the polar decomposition [32, p.21.9] applied to

$$\mathscr{B}(X^{(i)}) = Y^{(i)} \Lambda_i, \quad \mathscr{B}(X_*) = X_* \Lambda_*,$$

where $\Lambda_i \succeq 0$ and $\Lambda_* \succeq 0$, we have

$$\|Y^{(i)} - X_*\|_{\mathrm{F}} \leq \frac{\|\mathscr{B}(X^{(i)}) - \mathscr{B}(X_*)\|_{\mathrm{F}}}{\sigma_k(\mathscr{B}(X^{(i)})) + \sigma_k(\mathscr{B}(X_*))} \tag{3.21}$$

$$\leq \frac{\left(\sum_{j=1}^{\ell} \|A_j\|_2^2\right)^{1/2}}{\sigma_k(\mathscr{B}(X^{(i)})) + \sigma_k(\mathscr{B}(X_*))} \|X^{(i)} - X_*\|_{\mathrm{F}}. \tag{3.22}$$

To proceed, we will have to relate $Y^{(i)}$ to $X^{(i+1)}$. For that purpose, we noted $X_j^{(i+1)} = Y_j^{(i)} Z_j^{(i)}$ where $Z_j^{(i)} \in \mathbb{O}^{k_j \times k_j}$ is the orthogonal polar factor of $(Y_j^{(i)})^{\mathrm{T}} D_j$. We expect $Z_j^{(i)}$ to be close to I_j under certain conditions, as we argued in the proof of Theorem 3.2, but bounding $Z_j^{(i)} - I_j$ in general could be messy because we have to deal with cases: 1) $D_j = 0$; 2) $(Y_j^{(i)})^{\mathrm{T}} D_j$ has full rank; 3) $(Y_j^{(i)})^{\mathrm{T}} D_j$ is rank-deficient. But still it is reasonable to conjecture a local linear rate of convergence that is no bigger than $\left(\sum_{j=1}^{\ell} \|A_j\|_2^2\right)^{1/2} \Big/ [2\sigma_k(\mathscr{B}(X_*))]$.

4 Acceleration by LOCG

Although Algorithm 1, an SCF iteration for solving NPD (2.3), is proved always convergent to KKT points, it may take many SCF iterations to converge to a solution with desired accuracy and that can be costly for large scale problems. In this section, we will explain an idea to speed things up. The idea is inspired by locally optimal conjugate gradient methods (LOCG) from optimization [45, 51] and has been increasingly used in numerical linear algebra for linear systems and eigenvalue problems [24, 26, 33, 58].

4.1 A variant of LOCG for acceleration

Without loss of generality, let $X^{(-1)} \in \mathbb{O}^{n \times k}$ be the approximate maximizer of (1.1) from the very previous iterative step, and $X \in \mathbb{O}^{n \times k}$ the current approximate maximizer. We are now looking for the next approximate maximizer $X^{(1)}$, along the line of LOCG, according to

$$X^{(1)} = \arg \max_{Y \in \mathbb{O}^{n \times k}} f(Y), \ \text{ s.t. } \mathcal{R}(Y) \subseteq \mathcal{R}([X, \mathscr{R}(X), X^{(-1)}]), \tag{4.1}$$

where

$$\mathscr{R}(X) := \frac{1}{2} \operatorname{grad} f_{|\mathbb{O}^{n \times k}}(X) = \mathscr{B}(X) - X \cdot \frac{1}{2}\Big[X^{\mathrm{T}} \mathscr{B}(X) + \mathscr{B}(X)^{\mathrm{T}} X\Big] \tag{4.2}$$

by (2.2) with $\mathscr{B}(X)$ as in (2.1).

Algorithm 2 LOCGvNPD: LOCG with (4.3) solved by Algorithm 1

Input: $A_j \succeq 0$ and D_j for $1 \le j \le \ell$, and $X^{(0)} \equiv [X_1^{(0)}, X_2^{(0)}, \ldots, X_\ell^{(0)}] \in \mathbb{O}^{n\times k}$;
Output: a maximizer of (1.1).
1: for $1 \le j \le \ell$, compute SVD $(X_j^{(0)})^{\mathrm{T}} D_j = U_j \Sigma_j V_j^{\mathrm{T}}$ and reset $X_j^{(0)}$ to $X_j^{(0)}(U_j V_j^{\mathrm{T}})$;
2: $X^{(-1)} = [\,]$; % null matrix
3: **for** $i = 0, 1, \ldots$ until convergence **do**
4: compute $W \in \mathbb{O}^{n\times m}$ such that $\mathcal{R}(W) = \mathcal{R}([X^{(i)}, \mathscr{R}(X^{(i)}), X^{(i-1)}])$;
5: compute $\widetilde{A}_j$, $\widetilde{D}_j$ according to (4.3b);
6: solve (4.3c) for Z_{opt} by Algorithm 1 with initially $Z^{(0)}$ being the first k columns of I_m;
7: $X^{(i+1)} = W Z_{\mathrm{opt}}$;
8: **end for**
9: **return** the last $X^{(i)}$.

Initially for the first iteration, we don't have $X^{(-1)} \in \mathbb{O}^{n\times k}$ and it is understood that $X^{(-1)}$ is absent from (4.1), i.e., simply $\mathcal{R}(Y) \subseteq \mathcal{R}([X, \mathscr{R}(X)])$.

We have to numerically solve (4.1). For that purpose, let $W \in \mathbb{O}^{n\times m}$ be an orthonormal basis matrix of subspace $\mathcal{R}([X, \mathscr{R}(X), X^{(-1)}])$. Generically, $m = 3k$ but $m < 3k$ can happen. It can be implemented by the Gram-Schmit orthogonalization process, starting with the columns of $\mathscr{R}(X)$ against X since $X \in \mathbb{O}^{n\times k}$ already. In MATLAB, to fully take advantage of its optimized functions, we simply set $W = [\mathscr{R}(X), X^{(-1)}]$ (or $W = \mathscr{R}(X)$ for the first iteration) and then do

```
W=W-X*(X'*W); W=orth(W); W=W-X*(X'*W); W=orth(W);
W=[X,W];
```

where the first line performs the classical Gram-Schmit orthonormalization twice to ensure that the resulting columns of W are fully orthogonal to the columns of X at the end of the first line, and `orth` is a MATLAB function for orthogonalization. It is important to note that the first k columns of the final W are the same as those of X.

Now it follows from $\mathcal{R}(Y) \subseteq \mathcal{R}([X, \mathscr{R}(X), X^{(-1)}]) = \mathcal{R}(W)$ that in (4.1)

$$Y = WZ \quad \text{for } Z \in \mathbb{O}^{m\times k}. \tag{4.3a}$$

Partition $Z \equiv [Z_1, Z_2, \ldots, Z_\ell]$ with $Z_j \in \mathbb{R}^{m\times k_j}$, similarly to X as in (1.1), and let

$$\widetilde{A}_j = W^{\mathrm{T}} A_j W \succeq 0, \quad \widetilde{D}_j = W^{\mathrm{T}} D_j \text{ for } 1 \le j \le \ell. \tag{4.3b}$$

Problem (4.1) becomes

$$Z_{\mathrm{opt}} = \arg \max_{Z\in\mathbb{O}^{m\times k}} \left\{ \widetilde{f}(Z) := f(WZ) = \sum_{j=1}^{\ell} \left[\mathrm{tr}(Z_j^{\mathrm{T}} \widetilde{A}_j Z_j) + 2\,\mathrm{tr}(Z_j^{\mathrm{T}} \widetilde{D}_j) \right] \right\}, \tag{4.3c}$$

and $X^{(1)} = WZ_{\mathrm{opt}}$. This is of the same type as (1.1), except its dimensionality n is drastically reduced to m. Algorithm 1 can be simply applied to solve (4.3c), and we denote the resulting method, outlined in Algorithm 2, by LOCGvNPD.

Algorithm 2 outlines an overall inner-outer iterative scheme for (1.1), where at Line 6 any other method, if known, can be inserted to replace Algorithm 1 to solve (4.3c). Later when we consider the MAXBET subproblem (1.4), we will test Algorithm 2 with a method different from Algorithm 1.

Remark 4.1 There are a few comments in order, regarding Algorithm 2.

(i) It is important to compute W at Line 4 in such a way, as explained moments ago, that its first k columns are exactly the same as those of $X^{(i)}$. This is because as $X^{(i)}$ converges, $X^{(i+1)}$ changes little from $X^{(i)}$ and hence $Z_{\rm opt}$ is increasingly close to the first k columns of I_m. This explains the choice of $Z^{(0)}$ at Line 6.

(ii) At Line 5, portions of $\widetilde{A}_j$ and $\widetilde{D}_j$ have already been computed. For example, $A_jX^{(i)}$ and $[X^{(i)}]^{\rm T}A_jX^{(i)}$ are needed, and, after Line 7, we may use

$$A_jX^{(i+1)} = (A_jW)Z_{\rm opt}, \quad [X^{(i+1)}]^{\rm T}A_jX^{(i+1)} = Z_{\rm opt}^{\rm T}(W^{\rm T}A_jW)Z_{\rm opt}$$

to compute the next $A_jX^{(i+1)}$ and $[X^{(i+1)}]^{\rm T}A_jX^{(i+1)}$ at the costs $O(nk^2)$ and $O(k^3)$, respectively, instead of $O(n^2k)$ by reusing (A_jW) and $(W^{\rm T}A_jW)$.

(iii) At Line 6, Algorithm 1 may take many SCF iterative steps to solve (4.3c). But since $m \le 3k$, the cost of each iterative step is still $O(k^3)$.

(iv) Another area of improvement is to solve (4.3c) with an accuracy, comparable but fractionally better than the current $X^{(i)}$ as an approximate solution of (1.1). Specifically, if we use (3.10) at Line 3 to stop the for-loop: Lines 3–8, with tolerance ϵ, then instead of using the same ϵ for Algorithm 1 at Line 6, we can use a fraction, say 1/4, of $\varepsilon_{KKT} + \varepsilon_{\rm sym}$ evaluated at the current approximation $X = X^{(i)}$ as stopping tolerance.

The overall complexity of Algorithm 2 per outer iterative step is still $O(n^2k)$, the same as Algorithm 1 per SCF iterative step, assuming $k \ll n$. But Algorithm 2 may take many fewer outer iterative steps than the number of SCF iterative steps required by Algorithm 1 if applied directly to solve (1.1). Example 4.1 below presents a random example to illustrate this point. We also ran three general-purpose solvers for optimization on manifold: steepest-descent (SD) and trust-regions (RTR) methods from `manopt`[2] [9] and the multiplier correction method (FOForth) from `STOP`[3] [17, 54] for a brief comparison to demonstrate that specially designed solvers like ours can outperform general-purpose ones. It is noted that, except RTR, all methods are first-order methods. More details can be found online at the web sites of `manopt` and `STOP`. Usually second-order methods converge faster and are more efficient than first-order methods for problems of modest scales, but for problems of large scales, second-order methods may suffer too high complexity per iterative step. Hence, in general methods of the same order are compared against each other for fairness. Another general-purpose solver is `OPtM` [56] which was extensively compared against `STOP` in [17] where it was concluded that `STOP` holds an edge in performance.

[2] A MATLAB toolbox for OPTimization on MANifolds available online at https://www.manopt.org/.

[3] A toolbox for STiefel manifold OPtimization available online at https://stmopt.gitee.io/.

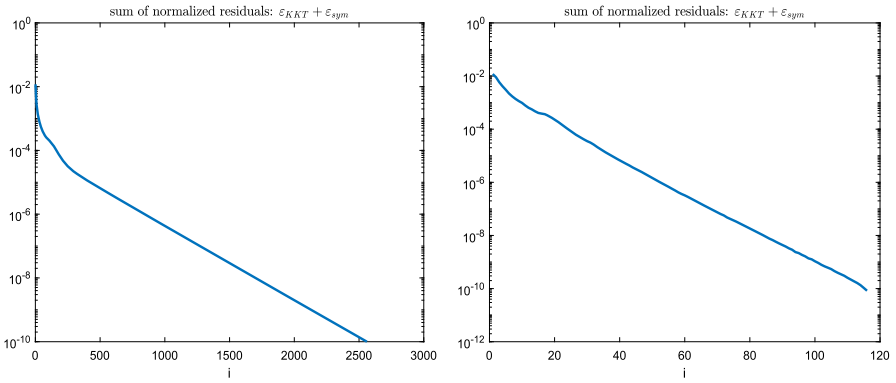

Fig. 1 Example 4.1: convergence history of Algorithms 1 (left) and Algorithm 2 (right). Various performance statistics by both methods, along with solvers from `manopt` and `STOP`, are listed in Table 1. We see tremendous speedups by Algorithm 2

Example 4.1 This is a random example in MATLAB: $n = 1000$, $\ell = 10$, all $k_j = 2$ and hence $k = 2\ell = 20$, and A_j and D_j are generated as

$$C_j = \texttt{randn}(n),\ \ A_j = C_j C_j^{\mathrm{T}},\ \ D_j = \texttt{randn}(n, k_j). \tag{4.4}$$

With tolerance $\epsilon = 10^{-10}$ in (3.10), Fig. 1 plots the normalized residuals $\varepsilon_{\mathrm{KKT}} + \varepsilon_{\mathrm{sym}}$ in (3.10) evaluated at each approximation $X^{(i)}$, by Algorithm 1 and Algorithm 2, respectively, while various important quantities measuring the performance of each method at convergence are displayed in Table 1, where "# itn" and "CPU"[4] refer to the number of outer-iterative steps and the CPU time (in seconds); "sol. err." and "obj. err" refer to the relative error in the computed maximizer and the objective value by any other algorithm against the ones returned by Algorithm 1 as

$$\frac{\|X_{\text{computed}} - X_{\text{alg. 1}}\|_{\mathrm{F}}}{\|X_{\text{alg. 1}}\|_{\mathrm{F}}},\quad \frac{f_{\text{computed}} - f_{\text{alg. 1}}}{f_{\text{alg. 1}}};$$

and "residual" is for the sum of normalized residuals $\varepsilon_{\mathrm{KKT}} + \varepsilon_{\mathrm{sym}}$ at computed X by an algorithm.

In Table 1, we also include performance statistics for SD and RTR from `manopt` [9] and FOForth from `STOP` both with 10000 as the maximally allowed number of iterations and tolerance 10^{-10} on the gradient norm. Both SD and FOForth were forced to stop at iteration 10000. We highlight the following observations:

1. The last row for "residual" determines the quality of computed solutions as KKT points. Judging from it, we find that all methods compute some KKT points but the one by `manopt`'s SD is least accurate while the second-order method RTR is

[4] All numerical demonstrations in this paper were done in MATLAB on a laptop with Intel(R) Core(TM) i7-1165G7 CPU 2.80 GHz and 32GB memory, except those in Sect. 5.2.2 which were performed on an EXXACT workstation (www.exxactcorp.com).

Table 1 Performance statistics (Example 4.1)

	Algorithm 1	Algorithm 2	`manopt` [9]		`STOP`
	NPDvSCF	LOCGvNPD	SD	RTR	FOForth
# itn	2560	115	10^4	30	10^4
CPU	9.0711	2.0115	668.94	43.783	250.58
sol. err.		2.15×10^{-7}	1.09	1.09	1.09
obj. err.		-7.98×10^{-15}	-1.67×10^{-4}	-1.67×10^{-4}	-1.67×10^{-4}
Residual	9.96×10^{-10}	8.31×10^{-11}	7.23×10^{-7}	7.45×10^{-17}	3.96×10^{-10}

most accurate. The KKT points by the other three methods have about the same accuracy.

2. The row for "sol. err." is calculated with the solution by NPDvSCF as the reference. Hence we will see a large error if a method computes a different KKT point. We find that NPDvSCF and LOCGvNPD compute the same KKT point, which is surely different from the ones by SD and RTR from `manopt` and by FOForth from `STOP`.
3. The row for "obj. err." is calculated with the objective value by NPDvSCF as the reference. A negative entry implies a smaller objective value returned by the corresponding method, and hence a worse maximizer. NPDvSCF and LOCGvNPD produce essentially the same objective value that is larger (i.e., better) than the one by any of the other three methods.
4. Lastly, LOCGvNPD is fastest, about 4.5 times faster than the second best NPDvSCF in terms of CPU time. The acceleration by LOCG really helps, reducing the number of iterations from 2560 by NPDvSCF to 115 by LOCGvNPD. Both NPDvSCF and LOCGvNPD decidedly outperform the other three solvers, even the second-order RTR, by wide margins.

In summary, LOCGvNPD is the best method measured in terms of speed, accuracy in KKT points, and quality in maximizer.

Example 4.2 In this example, we investigate how the CPU time changes when k or n is varied. Specifically, we let $\ell = 10$, all $k_j = \hat{k}$ and hence $k = \hat{k}\ell$, and generate A_j and D_j as in (4.4). For our methods, we use tolerance $\epsilon = 10^{-6}$ in (3.10), and also use 10^{-6} on the gradient norm for the solvers from `manopt` and `STOP`. The maximally allowed number of iterations are set at 10,000 for all methods. We checked that, with these tolerances, residuals as defined in (3.10) at the returned solutions by all methods are about 10^{-6} to 10^{-7}, except for `manopt`'s RTR whose scaled residuals are about 10^{-12} or smaller as one might expect from a second-order method.

For each random problem, the same initial guess is used by all methods.

Table 2 displays the CPU time consumed by each method for a typical run for $n = 2000$ with $k_j = \hat{k} \in \{1, 3, 5, 7, 9\}$. In this particular run, `manopt`'s SD was forced to stop at iteration 10000 each time, except for $k = 10$ for which it only took 1566 iterations. It seems that this particular random problem for $k = 10$ is an easy one. In general (as we observed from many runs), LOCGvNPD is the best, and NPDvSCF and LOCGvNPD easily outperform the solvers from `manopt` and `STOP`.

Table 2 CPU time for $n = 2000$ while k varies (Example 4.2)

k	Algorithm 1	Algorithm 2	`manopt` [9]		`STOP`
	NPDvSCF	LOCGvNPD	SD	RTR	FOForth
10	11.530	1.6620	417.98	28.400	14.444
30	15.412	4.6623	2871.2	209.21	196.87
50	35.621	13.655	2914.5	313.75	203.74
70	35.140	15.377	3014.4	319.44	205.84
90	58.568	21.710	3060.4	448.52	326.54

Table 3 CPU time for $k = 20$ while n varies (Example 4.2)

n	Algorithm 1	Algorithm 2	`manopt` [9]		`STOP`
	NPDvSCF	LOCGvNPD	SD	RTR	FOForth
1000	5.2726	1.1855	764.47	37.365	26.200
1200	21.528	4.7167	1083.8	45.735	35.630
1400	5.8807	1.5276	1384.3	79.075	92.361
1600	12.602	2.5251	1760.0	89.325	85.393
1800	13.501	3.1630	2242.4	110.09	113.95
2000	14.255	2.8914	2621.5	325.10	199.70

Table 3 displays the CPU time consumed by each method for a typical run for fixed $k_j = 2$ giving $k = 20$ as n varies from 1000 to 2000. Once again, `manopt`'s SD was forced to stop at iteration 10000. LOCGvNPD is indisputably the best method and together with NPDvSCF, they are the top 2 performers. We point out that the CPU times in both Tables 2 and 3 vary from one run to another as they are for randomly generated problems, but the overall behavior remains the same. This comment applies to all our later experiments for the MAXBET subproblem in Sect. 5.1 and the MvPS subproblem in Sect. 5.2.

4.2 Convergence analysis

In this subsection, we will perform a convergence analysis of Algorithm 2. We first consider an ideal situation that at its Line 6, Z_{opt} is computed to be an exact maximizer of (4.3) for simplicity, and then discuss situations when Z_{opt} is computed approximately. We point out that the seemingly ideal situation is not completely unrealistic. In actual computation, as we explained in Remark 4.1(iv), the computed Z_{opt} should be sufficiently more accurate as an approximation solution for (4.3) than $X^{(i)}$ as an approximation one for the original problem (1.1) at that moment.

Before we state our main results, let us again look at whether $f(X^{(i_0+1)}) = f(X^{(i_0)})$ can possibly happen for some i_0 in Algorithm 2. Suppose that is case. Then there will be no increase in the objective value at Line 6 for (4.3). By (3.8) and (3.9), we conclude that, for $i = i_0$ at Line 6,

$$\widetilde{\mathscr{B}}(WZ^{(0)}) = Z^{(0)}\Lambda \quad \text{for some } \Lambda \succeq 0.$$

Equivalently, $0 = W^{\mathrm{T}}\big[\mathscr{B}(WZ^{(0)}) - WZ^{(0)}\Lambda\big] = W^{\mathrm{T}}\mathscr{R}(X^{(i_0)})$ because $WZ^{(0)} = X^{(i_0)}$. Since $\mathcal{R}(W) \supseteq \mathcal{R}(\mathscr{R}(X^{(i_0)}))$, we have $\mathscr{R}(X^{(i_0)}) = 0$. This, together with $(X_j^{(i_0)})^{\mathrm{T}} D_j \succeq 0$ for $1 \le j \le \ell$ already, implying $X^{(i_0)}$ is a KKT point of (1.1) and satisfies the necessary conditions in (2.9) for a global maximizer. This in general is too good a scenario and almost never happens in actual computation. For that reason, in our analysis below, we will exclude such a situation from consideration.

Theorem 4.1 *Let the sequence $\{X^{(i)}\}_{i=0}^{\infty}$ be generated by Algorithm 2 in which, it is assumed that Z_{opt} is an exact maximizer of (4.3). Suppose that $f(X^{(i+1)}) \neq f(X^{(i)})$ for all $i \ge 0$. The following statements hold.*

(a) $(X^{(i)})^{\mathrm{T}}\mathscr{B}(X^{(i)}) \succeq 0$ *for* $i \ge 1$, *and* $(X_j^{(i)})^{\mathrm{T}} D_j \succeq 0$ *for* $1 \le j \le \ell$ *and* $i \ge 1$.

(b) *The sequence $\{f(X^{(i)})\}_{i=0}^{\infty}$ is monotonically increasing and convergent.*

(c) *Any accumulation point X_* of the sequence $\{X^{(i)}\}_{i=0}^{\infty}$ is a KKT point of (1.1) and satisfies the necessary conditions in (2.9) for a global maximizer.*

Proof Similarly to (1.2b) and (2.1), introduce for (4.3)

$$\widetilde{\mathscr{A}}(Z) := [\widetilde{A}_1 Z_1, \ldots, \widetilde{A}_\ell Z_\ell] \in \mathbb{R}^{m\times k}, \quad \widetilde{D} := [\widetilde{D}_1, \widetilde{D}_2, \ldots, \widetilde{D}_\ell] \in \mathbb{R}^{m\times k},$$
$$\widetilde{\mathscr{B}}(Z) := \widetilde{\mathscr{A}}(Z) + \widetilde{D} \in \mathbb{R}^{m\times k}.$$

It can be seen that $\widetilde{\mathscr{A}}(Z) = W^{\mathrm{T}}\mathscr{A}(WZ)$ and $\widetilde{\mathscr{B}}(Z) = W^{\mathrm{T}}\mathscr{B}(WZ)$. Hence, by the assumption that Z_{opt} is an exact maximizer of (4.3), we have at the end of Line 6

$$\begin{aligned} 0 &\preceq Z_{\mathrm{opt}}^{\mathrm{T}}\widetilde{\mathscr{B}}(Z_{\mathrm{opt}}) = Z_{\mathrm{opt}}^{\mathrm{T}} W^{\mathrm{T}}\mathscr{B}(WZ_{\mathrm{opt}}) = (X^{(i+1)})^{\mathrm{T}}\mathscr{B}(X^{(i+1)}), \\ 0 &\preceq Z_{\mathrm{opt},j}^{\mathrm{T}}\widetilde{D}_j = Z_{\mathrm{opt},j}^{\mathrm{T}} W^{\mathrm{T}} D_j = (X_j^{(i+1)})^{\mathrm{T}} D_j, \end{aligned} \tag{4.5}$$

proving item (a). Let Z be the first k columns of I_m. Then $\widetilde{f}(Z) = f(X^{(i)})$ in (4.3), and thus

$$f(X^{(i+1)}) = f(WZ_{\mathrm{opt}}) = \widetilde{f}(Z_{\mathrm{opt}}) \ge \widetilde{f}(Z) = f(X^{(i)}).$$

This proves item (b).

Next, we prove item (c). Let $\{X^{(i)}\}_{i\in\mathbb{I}}$ be a subsequence that converges to $X_* \equiv [X_{*1}, \ldots, X_{*\ell}]$. Letting $\mathbb{I} \ni i \to \infty$ in the inequalities in item (a) immediately yields $X_*^{\mathrm{T}}\mathscr{B}(X_*) \succeq 0$ and $X_{*j}^{\mathrm{T}} D_j \succeq 0$ for $1 \le j \le \ell$.

It remains to show $\mathscr{B}(X_*) = X_*\Lambda_*$, where $\Lambda_* = X_*^{\mathrm{T}}\mathscr{B}(X_*)$. Suppose, to the contrary, that $\mathscr{B}(X_*) \neq X_*\Lambda_*$. Since $\mathscr{R}(\cdot)$ of (4.2) is continuous, we have

$$R_* := \lim_{\mathbb{I}\ni i\to\infty} \mathscr{R}(X^{(i)}) = \mathscr{R}(X_*) = \mathscr{B}(X_*) - X_*\Lambda_* \neq 0, \tag{4.6}$$

where $R_* \neq 0$ is due to the assumption $\mathscr{B}(X_*) \neq X_*\Lambda_*$ we just made.

In this paragraph, we will establish some general results for $X \in \mathbb{O}^{n\times k}$ sufficiently close to X_* and they will be used later with X set to some particular $X^{(i)}$ to complete our proof. It can be seen that there exists $\gamma_1 > 0$ such that for any $X \in \mathbb{O}^{n\times k}$

$$\|\mathscr{R}(X) - \mathscr{R}(X_*)\|_{\mathrm{F}} \le \gamma_1 \|X - X_*\|_{\mathrm{F}},$$

where γ_1 is independent of X. Given $X \in \mathbb{O}^{n\times k}$, let for $t \in \mathbb{R}$

$$G(t) = X + t\mathscr{R}(X) = X_* + tR_* + \left\{X - X_* + t[\mathscr{R}(X) - R_*]\right\} \tag{4.7a}$$

from which we have

$$\|G(t) - X_*\|_{\mathrm{F}} \le |t|(\|R_*\|_{\mathrm{F}} + \gamma_1) + \|X - X_*\|_{\mathrm{F}} \quad \text{for } \|X - X_*\|_{\mathrm{F}} \le 1. \tag{4.7b}$$

It is not too hard to verify that if

$$|t| < \min\left\{1, \frac{1}{2(\|R_*\|_{\mathrm{F}} + \gamma_1)}\right\} =: \tau_1, \quad \|X - X_*\|_{\mathrm{F}} < \frac{1}{2} =: \delta_1, \tag{4.8}$$

then $\|G(t) - X_*\|_{\mathrm{F}} < 1$. Hence under (4.8), $G(t)$ has full column rank and thus

$$Y(t) = G(t)[G(t)^{\mathrm{T}}G(t)]^{-1/2}$$

is well-defined. Again because of (4.7), there exist τ_2 and δ_2, satisfying $0 < \tau_2 \le \tau_1$ and $0 < \delta_2 \le \delta_1$, such that, whenever $|t| < \tau_2$ and $\|X - X_*\|_{\mathrm{F}} < \delta_2$,

$$Y(t) = X_* + tR_* + \left\{X - X_* + t[\mathscr{R}(X) - R_*]\right\} + h_1(t, X), \tag{4.9a}$$

$$\|h_1(t, X)\|_{\mathrm{F}} \le \gamma_2(\|X - X_*\|_{\mathrm{F}}^2 + t^2), \tag{4.9b}$$

where γ_2 is a constant, independent of X and t. On the other hand $f(X)$ is differentiable with respect to $X \in \mathbb{O}^{n\times k}$. Therefore there exists δ_3, satisfying $0 < \delta_3 \le \delta_2$ and independent of X, such that, whenever $\|X - X_*\|_{\mathrm{F}} < \delta_3$,

$$f(X) = f(X_*) + \frac{1}{2}\operatorname{tr}((X - X_*)^{\mathrm{T}}\mathscr{R}(X_*)) + h_2(X), \quad |h_2(X)| \le \gamma_3\|X - X_*\|_{\mathrm{F}}^2, \tag{4.10}$$

where constant γ_3 is independent of X. We claim that if

$$\|X - X_*\|_{\mathrm{F}} < \delta_4 := \frac{\delta_3}{2(1 + \gamma_2)} < \delta_3, \quad |t| < \tau_4 := \min\left\{1, \frac{\delta_3}{2(\|R_*\|_{\mathrm{F}} + \gamma_1 + \gamma_2)}\right\}, \tag{4.11}$$

then $\|Y(t) - X_*\|_{\mathrm{F}} < \delta_3$. In fact, it follows from (4.7) and (4.9) that under (4.11)

$$\begin{aligned}\|Y(t) - X_*\|_{\mathrm{F}} &\le |t|\|R_*\|_{\mathrm{F}} + \|X - X_*\|_{\mathrm{F}} + |t|\gamma_1\|X - X_*\|_{\mathrm{F}} + \gamma_2(\|X - X_*\|_{\mathrm{F}}^2 + t^2)\\ &\le |t|(\|R_*\|_{\mathrm{F}} + \gamma_1 + \gamma_2) + \|X - X_*\|_{\mathrm{F}}(1 + \gamma_2)\\ &< \delta_3,\end{aligned} \tag{4.12}$$

as expected. Let

$$\tau = \min\left\{\tau_4, \frac{\operatorname{tr}(R_*^{\mathrm{T}}R_*)}{32\gamma_3(\|R_*\|_{\mathrm{F}}+\gamma_1+\gamma_2)^2}, \frac{\operatorname{tr}(R_*^{\mathrm{T}}R_*)}{8\gamma_2\|R_*\|_{\mathrm{F}}}\right\} > 0, \tag{4.13a}$$

$$\epsilon = \min\left\{\delta_4, \sqrt{\frac{\tau\operatorname{tr}(R_*^{\mathrm{T}}R_*)}{32\gamma_3(1+\gamma_2)^2}}, \frac{\tau\operatorname{tr}(R_*^{\mathrm{T}}R_*)}{8\|R_*\|_{\mathrm{F}}(1+\tau\gamma_1+\gamma_2)}\right\} > 0, \tag{4.13b}$$

where $\tau > 0$ is due to the assumption $R_* \neq 0$ in (4.6).

Since subsequence $\{X^{(i)}\}_{i\in\mathbb{I}}$ converges to X_*, there is an $i_0 \in \mathbb{I}$ such that $\|X^{(i_0)} - X_*\|_{\mathrm{F}} < \epsilon$. Now let X in the previous paragraph be $X^{(i_0)}$ and $t = \tau$. We have

$$\mathcal{R}(Y(\tau)) \subseteq \mathcal{R}([X^{(i_0)}, \mathscr{R}(X^{(i_0)})]) \subseteq \mathcal{R}([X^{(i_0)}, \mathscr{R}(X^{(i_0)}), X^{(i_0-1)}]),$$

and hence by (4.10)

$$f(X^{(i_0+1)}) \geq f(Y(\tau)) = f(X_*) + \frac{1}{2}\operatorname{tr}((Y(\tau)-X_*)^{\mathrm{T}}R_*) + h_2(Y(\tau)). \tag{4.14}$$

We now estimate each of the last two terms in (4.14). By (4.12) we have

$$\begin{aligned} |h_2(Y(\tau))| &\leq \gamma_3\Big[\tau(\|R_*\|_{\mathrm{F}}+\gamma_1+\gamma_2) + \|X^{(i_0)}-X_*\|_{\mathrm{F}}(1+\gamma_2)\Big]^2 \\ &\leq 2\gamma_3\Big[\tau^2(\|R_*\|_{\mathrm{F}}+\gamma_1+\gamma_2)^2 + \epsilon^2(1+\gamma_2)^2\Big] \\ &\leq \frac{1}{8}\tau\operatorname{tr}(R_*^{\mathrm{T}}R_*), \end{aligned} \tag{4.15}$$

where we have used (4.13) to derive the last inequality. Next we estimate $\operatorname{tr}((Y(\tau)-X_*)^{\mathrm{T}}R_*)$ which, upon using (4.9) and then (4.7), is

$$\begin{aligned} &\frac{1}{2}\operatorname{tr}\left(\Big[\tau R_* + (X^{(i_0)}-X_*) + \tau(\mathscr{R}(X^{(i_0)})-R_*) + h_1(\tau, X^{(i_0)})\Big]^{\mathrm{T}}R_*\right) \\ &\quad\geq \frac{1}{2}\tau\operatorname{tr}(R_*^{\mathrm{T}}R_*) - \frac{1}{2}\|X^{(i_0)}-X_*\|_{\mathrm{F}}(1+\tau\gamma_1+\gamma_2)\|R_*\|_{\mathrm{F}} - \frac{1}{2}\gamma_2\tau^2\|R_*\|_{\mathrm{F}} \\ &\quad\geq \frac{1}{2}\tau\operatorname{tr}(R_*^{\mathrm{T}}R_*) - \frac{1}{8}\tau\operatorname{tr}(R_*^{\mathrm{T}}R_*), \end{aligned} \tag{4.16}$$

where we again have used (4.13) to derive the last inequality. Combine (4.14), (4.15), and (4.16) to get

$$\begin{aligned} f(X^{(i_0+1)}) &\geq f(X_*) + \frac{1}{2}\tau\operatorname{tr}(R_*^{\mathrm{T}}R_*) - \frac{1}{8}\tau\operatorname{tr}(R_*^{\mathrm{T}}R_*) - \frac{1}{8}\tau\operatorname{tr}(R_*^{\mathrm{T}}R_*) \\ &= f(X_*) + \frac{1}{4}\tau\operatorname{tr}(R_*^{\mathrm{T}}R_*) > f(X_*), \end{aligned}$$

contradicting $f(X_*) \geq f(X^{(i)})$ for all i because $f(X_*)$ is the limit of the monotonically increasing sequence $\{f(X^{(i)})\}_{i=0}^{\infty}$. Hence $R_* = 0$, or equivalently, $\mathscr{B}(X_*) = X_* \Lambda_*$, as was to be shown. □

One of the consequences of requiring that Z_{opt} is an exact maximizer of (4.3) in Theorem 4.1 is the first claim in its item (a): $(X^{(i)})^{\mathrm{T}} \mathscr{B}(X^{(i)}) \succeq 0$ for $i \geq 1$. It is a sensible thing to do because the reduced problem (4.3) is of a much smaller scale than the original problem (1.1), but our current implementation for computing Z_{opt} approximately follows Remark 4.1(iv). Theoretically, for obtaining a convergence theorem similar to Theorem 4.1, we can use a (crude) approximate solution Z'_{opt} to (4.3) at Line 6 and let $X^{(i+1)} = W Z'_{\text{opt}}$, replacing Line 7 of Algorithm 2, so long as that Z'_{opt} is obtained by executing at least one for-loop iteration in Algorithm 1 on (4.3) that increases the value of $\widetilde{f}$, with an additional safe-guard step (Line 7a below). Specifically, we can replace Lines 6 and 7 of Algorithm 2 with

6: compute a (crude) approximate solution Z'_{opt} to (4.3c) by Algorithm 1 with initially $Z^{(0)}$ being the first k columns of I_m in such a way that at least one for-loop iteration in Algorithm 1 is executed;
7: $X^{(i+1)} = W Z'_{\text{opt}}$;
7a: if $f(X^{(i+1)}) = f(X^{(i)})$, then execute one for-loop iteration in Algorithm 1 on (1.1) with $X^{(i)}$ as the initial and overwrite $X^{(i+1)}$ by the output.

For such modified Algorithm 2, we claim that all conclusions of Theorem 4.1, except $(X^{(i)})^{\mathrm{T}} \mathscr{B}(X^{(i)}) \succeq 0$ for $i \geq 1$, remain valid. In fact, except (4.5) which is no longer true, the proof of Theorem 4.1 above remains valid. Additionally, we need to take care of the case when the test in Line 7a above holds true. Then for the overwritten $X^{(i+1)}$, either $f(X^{(i+1)}) > f(X^{(i)})$ or still $f(X^{(i+1)}) = f(X^{(i)})$. For the former, the execution of the modified Algorithm 2 continues, and for the latter we fall back to Remark 3.1(v) to conclude the whole iteration.

5 Applications

In this section, we describe two applications where special cases of (1.1) arise naturally as computational subproblems that have to be solved repeatedly.

5.1 MAXBET subproblem

Without going into too much detail, we may state compactly the MAXBET problem [52, 53] arising from applied multivariate statistical analysis and data mining as

$$\max_{Q_i \in \mathbb{O}^{n_i \times k}\, \forall i} \left\{ g(\{Q_i\}_{i=1}^{v}) := \sum_{i,j=1}^{v} \operatorname{tr}(Q_i^{\mathrm{T}} \Phi_{ij} Q_j) \right\}, \tag{5.1}$$

where $\Phi_{ij} \in \mathbb{R}^{n_i \times n_j}$ such that $\Phi = [\Phi_{ij}] \in \mathbb{R}^{N \times N}$ is symmetric, and $N = \sum_{i=1}^{v} n_i$. Necessarily, $k \leq n_i$ for all i and usually $k \ll N$. Without loss of generality, we may assume

$$\Phi_{ii} \succeq 0 \quad \text{for } 1 \le i \le v;$$

otherwise each Φ_{ii} may be shifted to $\Phi_{ii} + \alpha_i I_{n_i}$ by some constant $\alpha_i \ge -\lambda_{\min}(\Phi_{ii})$ without affecting the maximizers of (5.1). Numerically, α_i can be estimated cheaply [66].

The MAXBET problem includes a generalization of the classical canonical correlation analysis (CCA) [12, 20, 22] for the multi-view (also known as multi-set) analysis, and the special case $k = 1$ is referred to as *the maximal correlation problem* (MCP) in the literature. There are a few numerical methods designed to solve (5.1) but no single one is better than any other with guarantee. Generic optimization methods on the product of Stiefel manifolds such as the Riemannian trust-region method [1, 59] are too slow to be practical for large N [36]. Recently, Liu, Wang, and Wang [35] investigated a few interesting ideas but its most ambitious one, which aims to find a global maximizer of (5.1), is again too expensive and still without any guarantee.

Often a general framework for solving (5.1) is an inner-outer iterative scheme to repeatedly update $\{Q_i\}_{i=1}^v$ in a manner similar to either the Jacobi or Gauss-Seidel updating scheme for a linear system of equations. By either scheme, it has to solve repeatedly, in its computational kernel, a problem of the form (1.4), i.e.,

$$\max_{X \in \mathbb{O}^{n\times k}} \operatorname{tr}(X^{\mathrm{T}} A X) + 2\operatorname{tr}(X^{\mathrm{T}} D), \tag{(1.4)}$$

a special case of (1.1): all $A_j = A \in \mathbb{R}^{n\times n}$ are symmetric positive semidefinite. For that reason, (1.4) is often called the *MAXBET subproblem*. According to the theory in Sect. 2, the KKT condition for (1.4) is

$$AX + D = X\Lambda, \quad X \in \mathbb{O}^{n\times k}, \quad \Lambda^{\mathrm{T}} = \Lambda \in \mathbb{R}^{k\times k}. \tag{5.2}$$

Our previous developments in Sects. 2–4 are immediately applicable. For example, as a corollary of Theorem 2.1, we have

Theorem 5.1 *Any global maximizer $X_* \in \mathbb{O}^{n\times k}$ of (1.4) satisfies (5.2) and $X_*^{\mathrm{T}} D \succeq 0$.*

Proof The theorem is a consequence of Theorem 2.1 and that $X^{\mathrm{T}} D \succeq 0$ implies $X^{\mathrm{T}}(AX + D) \succeq 0$ because $A \succeq 0$. □

Algorithm 1 (SCF for NPD (2.3)) straightforwardly applies. Possibly it may take many SCF iterative steps to reach a prescribed tolerance as before, and again Algorithm 2 comes to rescue. For MAXBET subproblem (1.4),

$$\mathscr{B}(X) \equiv \mathscr{A}(X) + D = AX + D. \tag{5.3}$$

The authors in [63] showed that its KKT condition (5.2) is equivalent to the following NEPv (nonlinear eigenvalue problem with eigenvector dependency):

$$\mathscr{H}(X)X := \Big[A + (D\,X^{\mathrm{T}} + X\,D^{\mathrm{T}})\Big]X = X\Omega, \; X \in \mathbb{O}^{n\times k}, \tag{5.4}$$

Algorithm 3 MBSvSCF: MAXBET subproblem solved by SCF [63]

Input: $A \succeq 0$ and D, and $X^{(0)} \in \mathbb{O}^{n\times k}$;
Output: a maximizer of (1.4) by solving NEPv (5.4).
1: compute SVD $(X^{(0)})^{\mathrm{T}} D = W\Sigma V^{\mathrm{T}}$ and reset $X^{(0)}$ to $X^{(0)}(WV^{\mathrm{T}})$;
2: **for** $i = 0, 1, \ldots$ until convergence **do**
3: compute a partial eigendecomposition: $\mathscr{H}(X^{(i)})\widehat{X}^{(i+1)} = \widehat{X}^{(i+1)}\Omega_{i+1}$, where $\widehat{X}^{(i+1)} \in \mathbb{O}^{n\times k}$ and $\mathrm{eig}(\Omega_{i+1})$ consists of the k largest eigenvalues of $\mathscr{H}(X^{(i)})$;
4: compute SVD $(\widehat{X}^{(i+1)})^{\mathrm{T}} D = W_{i+1}\Sigma_{i+1}V_{i+1}^{\mathrm{T}}$, and let $X^{(i+1)} = \widehat{X}^{(i+1)}(W_{i+1}V_{i+1}^{\mathrm{T}})$;
5: **end for**
6: **return** the last $X^{(i)}$.

and devised an SCF iteration, as outlined in Algorithm 3, to compute a maximizer.

By now, we have several ways to solve MAXBET subproblem (1.4): Algorithms 1, 2, 3 [63], and Algorithm 2/3 (by which we mean, at Line 6 of Algorithm 2, Algorithm 1 is replaced by Algorithm 3). It follows from (5.3) that

$$X^{\mathrm{T}}\mathscr{B}(X) = X^{\mathrm{T}}AX + X^{\mathrm{T}}D$$

and hence it is symmetric positive semidefinite if $X^{\mathrm{T}}D \succeq 0$, which is guaranteed for any of the approximations by all methods just mentioned. In order to get a sense how these methods perform against each other, we performed two experiments on random problems in Examples 5.1 and 5.2.

Example 5.1 This is a random MAXBET subproblem (1.4) experimented in MATLAB: $n = 1000$, $k = 10$, and

$$C = \texttt{randn}(n),\ \ A = CC^{\mathrm{T}},\ \ D = 10.0 \times \texttt{randn}(n, k). \tag{5.5}$$

This random problem is first generated and saved with an initial guess for reproducibility. With tolerance $\epsilon = 10^{-10}$ in (3.10), Fig. 2 plots the sum of normalized residual $\varepsilon_{\mathrm{KKT}} + \varepsilon_{\mathrm{sym}}$ in (3.10) evaluated at each approximation $X^{(i)}$, by Algorithms 1, 2, 3, and Algorithm 2/3. Table 4 displays important performance statistics at convergence, including those by SD and RTR from `manopt` [9], FOForth from `STOP` (both with 10,000 as the maximally allowed number of iterations and tolerance 10^{-10} on the gradient norm), and by the generalized power iteration (GPI) of [39] which is essentially Algorithm 1 with its Lines 5 and 6 simply replaced by $X^{(i+1)} = Y^{(i)}$ (recalling $\ell = 1$ for the MAXBET subproblem (1.4)). We highlight the following observations:

1. We can tell from the last row for "residual" in Table 4 that all methods compute some KKT points but the one by GPI is much worse than the others.
2. In terms of objective value, MBSvSCF yields the largest one, and the second largest value comes from NPDvSCF, LOCGvNPD, MBSvLOCG, and `manopt`'s RTR and `STOP`'s FOForth.
   Despite of taking 10,000 iterations, GPI [39] still produces the smallest objective value among all and that is not competitive at all.
   For GPI, Fig. 3 plots its iterative history in objective value and normalized residuals $\varepsilon_{\mathrm{KKT}}$ and $\varepsilon_{\mathrm{sym}}$. Although the curve for objective value signals convergence, it

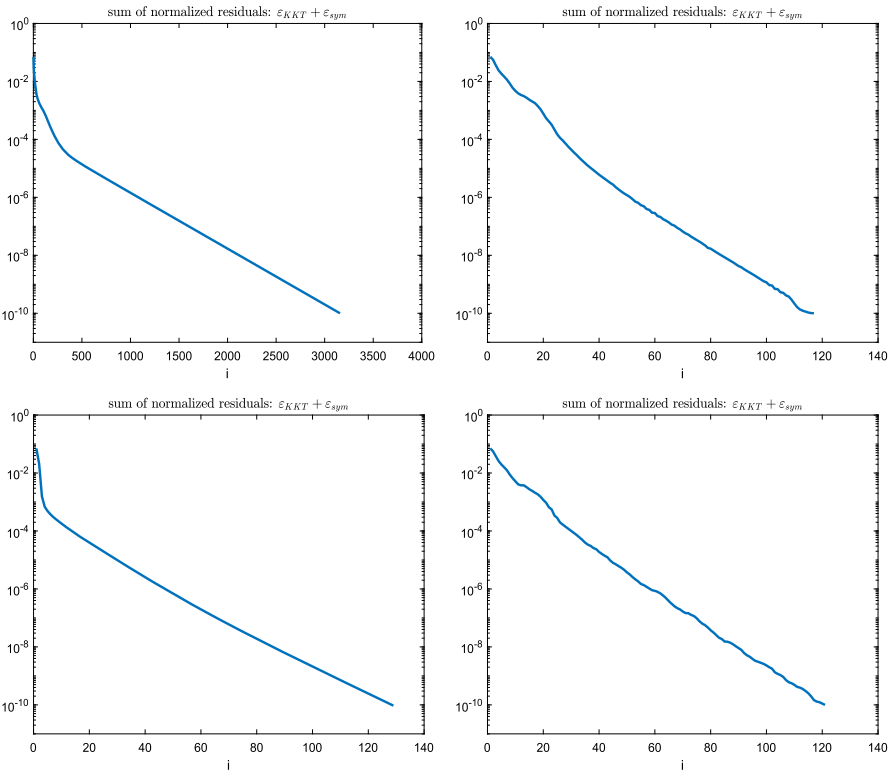

Fig. 2 Example 5.1: convergence history of Algorithm 1 (top left), Algorithm 2 (top right), Algorithm 3 (bottom left), and Algorithm 2/3 (bottom right). Various performance statistics by all methods, along with solvers from `manopt` and `STOP`, are listed in Table 4. We see tremendous speedups by Algorithm 2 and Algorithm 2/3

Table 4 Performance statistics (Example 5.1)

	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 2/3	`manopt` [9]		`STOP`	GPI [39]
	NPDvSCF	LOCGvNPD	MBSvSCF	MBSvLOCG	SD	RTR	FOForth	
# itn	3157	115	128	119	10^4	31	1144	10^4
CPU	1.52	0.38	9.56	0.26	79.8	2.95	3.28	4.72
sol. err.		9.4×10^{-8}	6.2×10^{-1}	1.3×10^{-7}	1.7×10^{-5}	2.2×10^{-7}	2.5×10^{-7}	2.0×10^{-2}
obj. err.		7.7×10^{-15}	2.7×10^{-4}	-2.1×10^{-15}	-1.5×10^{-13}	1.9×10^{-16}	3.7×10^{-16}	-1.9×10^{-2}
Residual	1.0×10^{-10}	8.9×10^{-11}	9.4×10^{-11}	9.5×10^{-11}	9.7×10^{-9}	1.2×10^{-15}	1.1×10^{-11}	5.7×10^{-6}

is misleading because both normalized residuals ε_{KKT} and ε_{sym} are stuck at an elevated level compared to the first four methods (see Fig. 2).

3. The row for "sol. err." is calculated with the solution by NPDvSCF as the reference. Hence we will see a large error if a method computes a different KKT point. We find that NPDvSCF, LOCGvNPD, MBSvLOCG, and `manopt`'s RTR and `STOP`'s

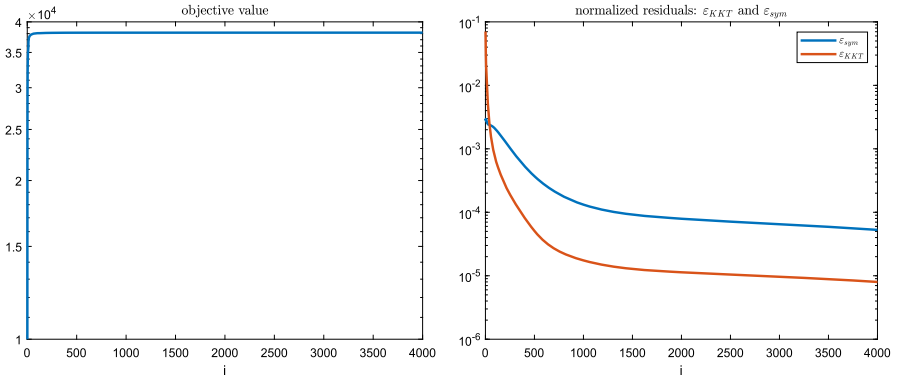

Fig. 3 Example 5.1: convergence history for GPI [39] (Algorithm 1 with its Lines 5 and 6 simply replaced by $X^{(i+1)} = Y^{(i)}$) for objective value (left) and normalized residuals ε_{KKT} and ε_{sym} (right). Both are stuck at an elevated level at iteration 4000

FOForth compute the same KKT point, which is surely different from the one by MBSvSCF. GPI may be too far from convergence to draw any conclusion.

4. Lastly, MBSvLOCG is fastest but LOCGvNPD is not too far behind in terms of CPU time. The acceleration by LOCG again really helps, reducing the number of iterations from 3157 by NPDvSCF to 115 by LOCGvNPD,
and in the case of the NEPv approach (Algorithm 3 [63]), LOCG barely reduces the number of iterations, but it dramatically reduces the sizes of the involved NEPv and saves the CPU time from 9.56 (seconds) required by MBSvSCF to 0.26 (seconds) required by MBSvLOCG. We point out that the long CPU time required by MBSvSCF can be reduced by using an iterative method instead of MATLAB's `eig` in our current implementation.

Overall, LOCGvNPD and MBSvLOCG are the best methods measured in terms of speed, accuracy in KKT points, and quality in maximizer.

Example 5.2 In this example, we investigate how the CPU time changes when k or n is varied. Random problems are created as in (5.5). For our methods, we use tolerance $\epsilon = 10^{-6}$ in (3.10), and use 10^{-6} on the gradient norm for the solvers from `manopt` and from `STOP`. The maximally allowed number of iterations is set at 10^4. We checked that, with these tolerances, the scaled residuals as defined in (3.10) at the returned solutions are about $O(10^{-5})$ for SD and GPI, $O(10^{-12})$ or smaller for RTR (once again due to the fact that it is a second-order method), and $O(10^{-6})$ for all other methods.

For each random problem, the same initial guess is used by all methods.

Table 5 displays the CPU time consumed by each method for a typical run for $n = 2000$ with k varying from 10 to 90. It can be observed that MBSvLOCG is the best while LOCGvNPD and MBSvSCF compete to be the second best. Both SD and GPI were forced to stop at iteration 10,000.

Table 6 displays the CPU time consumed by each method for a typical run for $k = 20$ with n varying from 1000 to 2000. MBSvLOCG is still the best while LOCGvNPD is the second best, unlike in Table 5. NPDvSCF comes in third place. Noticeably, RTR

Table 5 CPU time for $n = 2000$ while k varies (Example 5.2)

k	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 2/3	manopt [9]		STOP	GPI [39]
	NPDvSCF	LOCGvNPD	MBSvSCF	MBSvLOCG	SD	RTR	FOForth	
10	3.1705	0.3672	3.5994	0.2234	289.11	20.259	31.817	18.033
30	3.3152	0.9879	5.0598	0.6347	339.09	37.109	43.386	41.324
50	13.941	4.0063	5.3259	1.5063	400.77	67.389	59.789	71.602
70	58.808	14.890	9.7731	3.8846	518.82	88.315	112.50	118.17
90	37.621	10.948	6.4539	3.9183	572.31	124.18	74.650	157.37

Table 6 CPU time for $k = 20$ while n varies (Example 5.2)

n	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 2/3	manopt [9]		STOP	GPI [39]
	NPDvSCF	LOCGvNPD	MBSvSCF	MBSvLOCG	SD	RTR	FOForth	
1000	1.7730	0.6110	28.490	0.4212	94.862	6.0573	4.4660	8.9887
1200	2.4189	0.7503	7.4116	0.3949	68.245	3.1151	2.4801	10.185
1400	1.9557	0.5044	9.5028	0.5643	165.76	9.7379	6.6982	17.459
1600	5.2386	0.6107	29.796	0.5643	188.46	7.1918	4.9768	20.347
1800	2.5541	0.6439	49.740	0.4864	258.72	9.3432	5.8275	23.124
2000	5.5881	1.1475	129.09	0.7447	294.55	14.195	9.3191	27.479

and FOForth perform reasonably well in this experiment, but still fall far behind the first four methods in the table.

5.2 Multi-view subspace learning with both view-specific and shared projections

Today more and more data sets in real-world applications are collected with multiple heterogeneous features from different perspectives (views). Each view represents a different aspect of the same object, and thus can and will possess its own characteristics. Multi-view learning is a methodology to make good use of all views for effective learning [34]. Most multi-view subspace learning methods aim to obtain a latent subspace shared by multiple views with an assumption that all views are generated from the latent subspace. Recently in [37], the authors proposed a novel multi-view partially shared subspace learning (MvPS) model that allows the co-existence of both commonality among views and individuality within each view. The idea of the co-existence is rather natural, of course, but the question is how to exploit it. Without going into too much detail, we may state the general MvPS model [37] as

$$\max\ g(\{Q_i,\ P_i\}_{i=1}^{v}) := \sum_{i,j=1}^{v} \mathrm{tr}(Q_i^{\mathrm{T}} \Phi_{ij} Q_j) + \sum_{i=1}^{v} \mathrm{tr}(P_i^{\mathrm{T}} \Psi_{ii} P_i) \tag{5.6a}$$

$$\text{s.t.}\ [Q_i,\ P_i] \in \mathbb{O}^{n_i \times (\kappa_0 + \kappa_i)}\ \forall i, \tag{5.6b}$$

where $Q_i \in \mathbb{R}^{n_i \times \kappa_0}$, $P_i \in \mathbb{R}^{n_i \times \kappa_i}$ for $1 \le i \le v$, and

$$\Phi = \begin{array}{c} \\ n_1 \\ n_2 \\ \\ n_v \end{array} \begin{array}{c} \begin{array}{ccc} n_1 & n_2 & n_v \end{array} \\ \begin{bmatrix} \Phi_{11} & \Phi_{12} & \dots & \Phi_{1v} \\ \Phi_{21} & \Phi_{22} & \dots & \Phi_{2v} \\ \vdots & \vdots & & \vdots \\ \Phi_{v1} & \Phi_{v2} & \dots & \Phi_{vv} \end{bmatrix} \end{array}, \quad \Psi = \begin{array}{c} \\ n_1 \\ n_2 \\ \\ n_v \end{array} \begin{array}{c} \begin{array}{ccc} n_1 & n_2 & n_v \end{array} \\ \begin{bmatrix} \Psi_{11} & & & \\ & \Psi_{22} & & \\ & & \ddots & \\ & & & \Psi_{vv} \end{bmatrix} \end{array}$$

are of $N \times N$ and symmetric, and $N = \sum_{i=1}^{v} n_i$. Necessarily, $\kappa_0 + \kappa_i \le n_i$ for all i. This model determines projection matrices $[Q_i, P_i]$ that will be used to project any instance $\{\boldsymbol{x}^{(i)}\}_{i=1}^{v}$ of v data points $\boldsymbol{x}^{(i)} \in \mathbb{R}^{n_i}$ for $1 \le i \le v$, one from each view, to $(Q_i^{\mathrm{T}}\boldsymbol{x}^{(i)}, P_i^{\mathrm{T}}\boldsymbol{x}^{(i)}) \in \mathbb{R}^{\kappa_0} \times \mathbb{R}^{\kappa_i}$, where $Q_i^{\mathrm{T}}\boldsymbol{x}^{(i)}$ represents the commonality among all $\boldsymbol{x}^{(i)}$ while $P_i^{\mathrm{T}}\boldsymbol{x}^{(i)}$ the individuality particular to view i.

Without loss of generality, we may assume

$$\Phi_{ii} \succeq 0, \quad \Psi_{ii} \succeq 0 \quad \text{for } 1 \le i \le v;$$

otherwise each Φ_{ii} and Ψ_{ii} may be shifted to $\Phi_{ii} + \alpha_i I_{n_i}$ and $\Psi_{ii} + \beta_i I_{n_i}$, respectively, by some constants $\alpha_i \ge -\lambda_{\min}(\Phi_{ii})$ and $\beta_i \ge -\lambda_{\min}(\Psi_{ii})$ without affecting the maximizers of (5.6). Numerically, α_i and β_i can be estimated cheaply [66].

5.2.1 MvPS subproblem

Similarly to MAXBET (5.1), MvPS (5.6) can be solved by the same general framework, an inner-outer iterative scheme, to repeatedly update $\{[Q_i, P_i]\}_{i=1}^{v}$ in a manner similar to either the Jacobi or Gauss-Seidel updating scheme. Either scheme relies on the following computational kernel

$$\max_{X \in \mathbb{O}^{n \times k}} \sum_{i=1}^{2} \operatorname{tr}(X_i^{\mathrm{T}} A_i X_i) + 2 \operatorname{tr}(X_1^{\mathrm{T}} D_1), \tag{5.7}$$

which is again a special case of (1.1): $\ell = 2$ and $D_2 = 0$, where $X = [X_1, X_2] \in \mathbb{O}^{n \times k}$ with $X_i \in \mathbb{R}^{n \times k_i}$ and $k = k_1 + k_2$. For that reason, we will call (5.7) the *MvPS subproblem*. According to the theory in Sect. 2, the KKT condition for (5.7) is

$$\mathscr{B}(X) := [A_1 X_1 + D_1, A_2 X_2] = X\Lambda, \ \ X \in \mathbb{O}^{n \times k}, \tag{5.8a}$$

$$\Lambda^{\mathrm{T}} = \Lambda \in \mathbb{R}^{k \times k}. \tag{5.8b}$$

Our previous developments in Sects. 2–4 are immediately applicable.

Next we use random MvPS subproblems to demonstrate the performances by Algorithm 1 and Algorithm 2, compared with solvers from `manopt` and `STOP`.

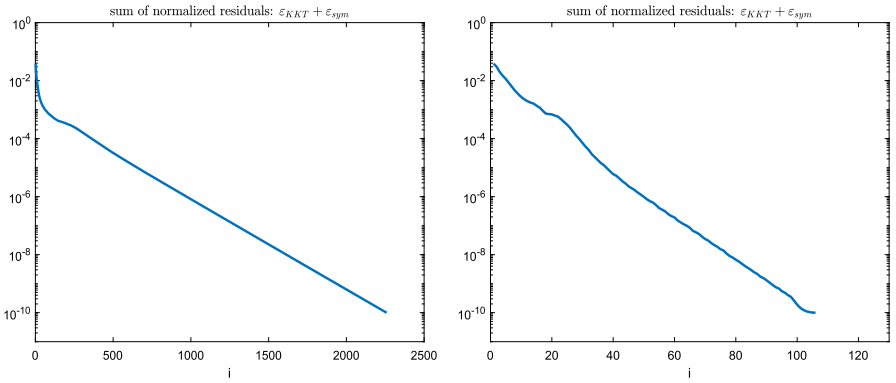

Fig. 4 Example 5.3: convergence history of Algorithms 1 (left) and Algorithm 2 (right). Various performance statistics by both methods, along with solvers from `manopt` and `STOP`, are listed in Table 7. We see tremendous speedups by Algorithm 2

Table 7 Performance statistics (Example 5.3)

	Algorithm 1	Algorithm 2	`manopt` [9]		`STOP`
	NPDvSCF	LOCGvNPD	SD	RTR	FOForth
# itn	2258	103	10^4	23	3614
CPU	1.8008	0.3709	169.81	7.2650	21.511
sol. err.		6.8×10^{-8}	8.2×10^{-1}	8.2×10^{-1}	8.2×10^{-1}
obj. err.		3.4×10^{-15}	-4.3×10^{-5}	-4.3×10^{-5}	-4.3×10^{-5}
residual	9.9×10^{-11}	1.0×10^{-10}	3.0×10^{-6}	3.9×10^{-16}	5.2×10^{-12}

Example 5.3 This is a random MvPS subproblem (5.7) experimented in MATLAB: $n = 1000$, $k_1 = 6$, $k_2 = 4$ and hence $k = 10$, and each A_j and D_1 are generated as

$$C_j = \texttt{randn}(n),\ \ A_j = C_j C_j^{\mathrm{T}},\ \ D_1 = \texttt{randn}(n, k_1). \tag{5.9}$$

This random problem is first generated and saved with an initial guess for reproducibility. With tolerance $\epsilon = 10^{-10}$ in (3.10), Fig. 4 plots the sum of normalized residuals $\varepsilon_{\mathrm{KKT}} + \varepsilon_{\mathrm{sym}}$ in (3.10) evaluated at each approximation $X^{(i)}$, by Algorithm 1 and Algorithm 2. Table 7 displays important performance statistics at convergence, including those by SD and RTR from `manopt` [9] and FOForth from `STOP` (both with 10,000 as the maximally allowed number of iterations and tolerance 10^{-10} on the gradient norm).

Notice that X_2 in (5.7) is clearly not uniquely determined because the objective function there is invariant under the transformation $X_2 \to X_2 Q$ for any $Q \in \mathbb{O}^{n\times k_2}$, while X_1 is unique under the assumption that $X_1^{\mathrm{T}} D_1 \succ 0$. For that reason, we calculate the difference in computed maximizers, from the one returned by Algorithm 1, as

Table 8 CPU time for $n = 2000$ while k varies (Example 5.4)

k	Algorithm 1	Algorithm 2	manopt [9]		STOP
	NPDvSCF	LOCGvNPD	SD	RTR	FOForth
10	2.5783	0.4711	583.59	21.997	21.539
30	11.367	2.0754	598.63	43.845	54.214
50	10.252	3.5555	698.51	59.922	26.979
70	52.790	27.816	790.08	155.90	53.707
90	52.202	16.610	873.49	78.870	53.388

Table 9 CPU time for $k_1 = k_2 = 10$ while n varies (Example 5.4)

n	Algorithm 1	Algorithm 2	manopt [9]		STOP
	NPDvSCF	LOCGvNPD	SD	RTR	FOForth
1000	1.4528	0.3342	177.64	11.089	5.2429
1200	1.5477	0.4710	245.54	15.153	12.913
1400	3.1492	0.6224	314.78	20.760	23.827
1600	6.0870	0.8025	393.49	37.752	33.479
1800	3.7702	0.7494	491.62	47.288	51.752
2000	3.8998	0.7376	611.88	73.058	81.513

$$\frac{\|X_1 - X_{1;\text{alg. }1}\|_{\mathrm{F}}}{\|X_{1;\text{alg. }1}\|_{\mathrm{F}}} + \frac{\|X_2 - X_{2;\text{alg. }1}(X_{2;\text{alg. }1}^{\mathrm{T}} X_2)\|_{\mathrm{F}}}{\|X_{2;\text{alg. }1}\|_{\mathrm{F}}}.$$

Algorithm 2 (LOCGvNPD) significantly outperforms Algorithms 1 (NPDvSCF) and all other methods. Similar observations to those for Example 4.1 can be made here.

Example 5.4 In this example, we investigate how the CPU time changes when k or n is varied. Random problems are created as in (5.9). For our methods, we use tolerance $\epsilon = 10^{-6}$ in (3.10), and also use 10^{-6} on the gradient norm for the solvers from `manopt` and `STOP`. The maximally allowed number of iterations are set at 10,000 for all methods. We checked that, with these tolerances, residuals as defined in (3.10) at the returned solutions by all methods are about 10^{-6}, except for `manopt`'s RTR whose scaled residuals are about 10^{-11} or smaller. For each random problem, the same initial guess is used by all methods.

Table 8 displays the CPU time consumed by each method for a typical run for $n = 2000$ with k varying from 10 to 90 and $k_1 = k/2$, including performance statistics for SD and RTR from `manopt` [9] and FOForth from `STOP`. SD was forced to stop at iteration 10,000 each time. It can be observed that LOCGvNPD is the best while NPDvSCF is the second best.

Table 9 displays the CPU time consumed by each method for a typical run for $k = 20$ with n varying from 1000 to 2000. Again LOCGvNPD is the best and NPDvSCF is the second best.

5.2.2 An MvPS model

Given a multi-view data set $\{\boldsymbol{x}_i^{(1)}, \ldots, \boldsymbol{x}_i^{(v)}, y_i\}_{i=1}^M$ with v views and M instances, where each $\boldsymbol{x}_i^{(s)} \in \mathbb{R}^{n_s}$ resides in an n_s-dimensional space and $y_i \in \{1, \ldots, c\}$ is the class label of c classes. MvPS (5.6) will learn orthonormal projection matrices $[Q_s, P_s]$ where $Q_s \in \mathbb{O}^{n_s \times \kappa_0}$ and $P_s \in \mathbb{O}^{n_s \times \kappa_s}$, which project data point $\boldsymbol{x}^{(s)}$ of view s into $(Q_s^{\mathrm{T}}\boldsymbol{x}^{(s)}, P_s^{\mathrm{T}}\boldsymbol{x}^{(s)}) \in \mathbb{R}^{\kappa_0} \times \mathbb{R}^{\kappa_s}$ of shared subspace $\mathbb{R}^{\kappa_0}$ and view-specific subspace $\mathbb{R}^{\kappa_s}$. In our experiments, for simplicity, we fix the total dimension of the shared and view-specific spaces as k: $\kappa_0 = m$ and $\kappa_s = k - m$ for $1 \le s \le v$.

Let[5] $X_s = [\boldsymbol{x}_1^{(s)}, \ldots, \boldsymbol{x}_n^{(s)}] \in \mathbb{R}^{n_s \times M}$, the data matrix of view s. The shared embedding of X_s is $Z_{\text{share}}^{(s)} = Q_s^{\mathrm{T}} X_s$ while the view-specific embedding is $Z_{\text{view}}^{(s)} = P_s^{\mathrm{T}} X_s$. Finally, the combined embedding for view s and that of all views are given by

$$Z^{(s)} = \begin{bmatrix} Z_{\text{view}}^{(s)} \\ Z_{\text{share}}^{(s)} \end{bmatrix} \in \mathbb{R}^{k \times M} \ \forall s, \quad Z = \begin{bmatrix} Z_{\text{view}}^{(1)} \\ \vdots \\ Z_{\text{view}}^{(v)} \\ \frac{1}{v}\sum_{s=1}^{v} Z_{\text{share}}^{(s)} \end{bmatrix} \in \mathbb{R}^{[v(k-m)+m] \times M}, \quad (5.10)$$

respectively.

The general MvPS model (5.6) can be versatile and instantiated according to needs. Here for demonstration of our algorithms only, we will use the one in [37] in the spirit of PCA, i.e., for $1 \le s,\ t \le v$,

$$\Phi_{st} = \frac{1}{v} X_s H_M X_t^{\mathrm{T}}, \quad \Psi_{ss} = X_s H_M X_s^{\mathrm{T}}, \quad (5.11)$$

where $H_M = I_M - \frac{1}{M}\mathbf{1}_M\mathbf{1}_M^{\mathrm{T}}$ is the centering matrix. The resulting MvPS (5.6) will then be solved by an alternating updating: either the Jacobi-style or Gauss-Seidel-style updating, with each subproblem in the form of (5.7) solved by Algorithm 2, as outlined in Algorithm 4. For references, we name the methods as MvPS-J and MvPS-G, respectively.

We evaluate the MvPS model solved by MvPS-J and MvPS-G on four real-world data sets. Two image data sets, Caltech101 [28] and Scene15 [27], are used. Six views are extracted from each data set according to the feature descriptors: CENTRIST (CENT) [57], GIST [43], LBP [42], histogram of oriented gradient (HOG), CH, and SIFT-SPM (SS) [27]. For Caltech101, two subsets are used: one contains 7 labels (Caltech101-7) and the other contains 20 labels (Caltech101-20). Note that CH is not included for Scene15 due to its gray-level images. A text data set Internet Advertisements (Ads) is used, which is for predicting whether or not a given hyperlink (associated with an image) is an advertisement, and has three views: features based on the terms in the images URL, caption, and alt text (UAC), features based on the terms in the URL of the current site (origurl), and features based on the terms in the anchor URL (ancurl). The statistics of the four data sets are shown in Table 10.

[5] Up to this point, X_j as in (1.1) has been used for an orthonormal projection matrix. For the rest of this section, we will use them as data matrices as is done conventionally. Hopefully, no confusion will arise.

Table 10 Data sets (inside the bracket is the number of features for the view)

Data set	M	c	View 1	View 2	View 3	View 4	View 5	View 6
Caltech101-7	1474	7	CENT (254)	GIST (512)	LBP (1180)	HOG (1008)	CH (64)	SS (1000)
Caltech101-20	2386	20	CENT (254)	GIST (512)	LBP (1180)	HOG (1008)	CH (64)	SS (1000)
Scene15	4310	15	CENT (254)	GIST (512)	LBP (531)	HOG (360)	SS (1000)	–
Ads	3279	2	UAC (588)	origurl (495)	ancurl (472)	–	–	–

Algorithm 4 MvPS (5.6) solved by NPD with LOCG acceleration

Input: Φ and Ψ as in (5.11), $1 \le \kappa_0 + \kappa_s \le \min_s n_s$, and tolerance ϵ;
Output: $\{[Q_s, P_s] \in \mathbb{O}^{n_s \times (\kappa_0+\kappa_s)}\}_{s=1}^v$ that approximately solves (5.6).

1: pick initial guess $\{[Q_s^{(0)}, P_s^{(0)}] \in \mathbb{O}^{n_s \times (\kappa_0+\kappa_s)}\}_{s=1}^v$;
2: $i = 0$, and evaluate the objective function of (5.6) at $\{[Q_s^{(0)}, P_s^{(0)}]\}_{s=1}^v$ to f;
3: **repeat**
4: **for** $s = 1$ to v **do**
5: $A_1 = \Phi_{ss}$, $A_2 = \Psi_{ss}$;
6: $D_1 = \sum_{t \ne s} \Phi_{st} Q_t^{(i)}$ for the Jacobi-style updating, or $D_1 = \sum_{t<s} \Phi_{st} Q_t^{(i+1)} + \sum_{t>s} \Phi_{st} Q_t^{(i)}$ for the Gauss-Seidel-style updating;
7: solve (5.7) by Algorithm 2 (with $[Q_s^{(i)}, P_s^{(i)}]$ as an initial guess) for its maximizer $[Q_s^{(i+1)}, P_s^{(i+1)}]$;
8: **end for**
9: $f_0 = f$, and evaluate the objective function of (5.6) at $\{[Q_s^{(i+1)}, P_s^{(i+1)}]\}_{s=1}^v$ to f;
10: $i = i + 1$;
11: **until** $|f - f_0| \le \epsilon f$;
12: **return** the last $\{[Q_s^{(i)}, P_s^{(i)}] \in \mathbb{O}^{n_s \times (\kappa_0+\kappa_s)}\}_{s=1}^v$.

We conduct two sets of experiments to demonstrate different aspects of our MvPS model. First, we show that the shared subspace learned by MvPS can boost the view-specific classification performance on each view. Second, we show that MvPS can be more effective than methods either without view-specific subspace or with only view-specific subspace in terms of multi-view feature extraction.

The importance of shared embedding MvPS (5.6) instantiated with (5.11) extends PCA from single-view learning to multi-view learning through introducing a subspace shared by all views in addition to individual view-specific subspaces. Therefore, it is important to show that the shared subspace can improve learning performance on each view individually.

To empirically validate the above hypothesis, we set up our experiments with the following settings for a fair comparison. We will use PCA as the baseline, for which, we first apply it to each view and get its embedding in the k-dimensional space, and then use the one-nearest-neighbor classifier to evaluate the performance of the embedding for the view. For our MvPS, there are two embeddings for each view: shared embedding $Z_{\text{share}}^{(s)}$ in $\mathbb{R}^m$ and view-specific embedding $Z_{\text{view}}^{(s)}$ in $\mathbb{R}^{(k-m)}$. Their concatenation $Z^{(s)}$ in (5.10) leads to a k-dimensional embedding for the view, which is analogous to that of PCA. Hence, view-wise, the same evaluation approach as PCA can be applied to MvPS. To obtain quantitative evaluation, we first randomly split the input data into 30% for training and 70% for testing and then learn the projection matrices of PCA and MvPS from the training data. With the learned projection matrices, the low-dimensional embeddings are obtained for both training and testing sets. The one-nearest-neighbor classifier is trained on the low-dimensional embeddings of the training data for each view and evaluated on that of the testing data of the view. We repeat the experiment 10 times and report the average classification accuracy with standard deviation. In those experiments, we fix $k = 20$ and vary $m \in [0, 20]$ for MvPS.

View-wise classification accuracy with standard deviation obtained by the three methods is shown in Table 11. We have the following observations: (i) MvPS sig-

Table 11 View-specific classification accuracy with standard deviation

Method	View 1	View 2	View 3	View 4	View 5	View 6
Caltech101-7 ($k = 20$, 30% training data)						
PCA	75.87 ± 4.89	51.82 ± 17.68	47.96 ± 8.96	52.68 ± 21.09	50.76 ± 6.93	34.47 ± 31.49
MvPS-J	92.91 ± 0.55	94.43 ± 0.59	94.61 ± 0.59	94.91 ± 0.48	79.55 ± 1.14	96.24 ± 0.50
MvPS-G	92.93 ± 0.56	94.34 ± 0.60	94.61 ± 0.47	95.05 ± 0.52	79.46 ± 1.03	96.25 ± 0.42
Caltech101-20 ($k = 20$, 30% training data)						
PCA	44.75 ± 18.83	38.25 ± 23.75	26.36 ± 16.04	15.49 ± 12.96	28.96 ± 11.45	24.31 ± 16.48
MvPS-J	78.77 ± 0.62	82.49 ± 0.66	83.44 ± 0.76	82.27 ± 0.65	62.24 ± 1.25	87.59 ± 1.16
MvPS-G	78.66 ± 0.82	82.43 ± 0.55	83.38 ± 0.63	82.08 ± 0.50	62.30 ± 0.91	87.68 ± 0.94

Method	View 1	View 2	View 3
Ads ($k = 20$, 30% training data)			
PCA	79.24 ± 4.26	69.55 ± 17.54	67.24 ± 27.09
MvPS-J	93.51 ± 0.55	91.37 ± 0.29	95.28 ± 0.46
MvPS-G	93.51 ± 0.49	91.37 ± 0.23	95.34 ± 0.51

Method	vView 1	View 2	View 3	View 4	View 5
Scene15 ($k = 20$, 30% training data)					
PCA	9.53 ± 4.69	14.61 ± 9.93	24.78 ± 15.03	20.49 ± 7.82	13.48 ± 4.43
MvPS-J	58.76 ± 1.29	55.05 ± 0.75	57.38 ± 0.50	47.38 ± 0.74	83.84 ± 0.78
MvPS-G	58.96 ± 0.84	55.01 ± 0.50	57.00 ± 0.70	47.35 ± 0.69	83.84 ± 0.67

nificantly outperforms PCA on all individual views; (ii) MvPS-J and MvPS-G show similar performances; (iii) PCA shows large variance among the 10 random experiments, while the MvPS methods demonstrate robust results with small variances. These observations demonstrate that it is absolutely beneficial to include a shared subspace even for view-specific classification, noting that PCA is always view-specific and does not take any shared information among views into consideration.

Combining shared and view-specific embeddings for feature extraction Moments ago, we demonstrated that shared embeddings can boost the accuracy of view-specific classification. Next, we will show that the combination of shared embeddings over all views and view-specific embeddings can boost the performance of multi-view feature extraction at very low training over testing ratios.

In terms of unsupervised feature extraction, we will compare our methods MvPS-J and MvPS-G against five existing methods: GMPCA [48], PLS (partial least-squares), MCCA [40], MvPCA (which averages the embeddings by PCA applied to each view independently), and UMvPLS [55]. These methods will be used to transform the input data of each view to latent spaces of dimension $k \in \{3, 5, 10, 15, 20\}$. Since these five methods do not actually produce view-specific embeddings, the average of projected embeddings of all views in the latent space is used as the fused embedding for classification. But for MvPS-J and MvPS-G, the embeddings in the k-dimensional space are given as Z in (5.10) with shared dimension m being tuned in the range of $[1, k]$. For each data set, we randomly draw 10% for training and leave the rest for testing and perform one experiment. We repeat the experiment 10 times, and report the average accuracy with standard deviation by each method in Table 12. Against MvPCA, it can be seen that including shared embedding by MvPS can significantly increase the classification accuracy for multi-view feature extraction from 2–27%. Overall, MvPS that also takes advantage of view-specific embeddings, compared with the best outcomes by the five existing methods with only shared embedding, always improves the classification accuracy, e.g., by 2% on Caltech101-20 and 10% on Scene15. These observations demonstrate that the combination of view-specific and shared embeddings is worth doing for better multi-view feature extraction.

6 Concluding remarks

We have developed a self-consistent-field (SCF) iteration to solve the nonlinear polar decomposition (NPD) arising from the first order optimality condition, combined with some critical maximizer necessary conditions, for the following maximization problem of the sum of coupled traces over the Stiefel manifold:

$$\max_{X^{\mathrm{T}}X=I_k} \sum_{j=1}^{\ell} \Big[\operatorname{tr}(X_j^{\mathrm{T}} A_j X_j) + 2 \operatorname{tr}(X_j^{\mathrm{T}} D_j) \Big]. \tag{6.1}$$

In the extreme special case: all $A_j = A$ and $D_j = 0$, the SCF iteration degenerates to the usual power method and hence it may converge slowly. To overcome possible slowness in convergence, we introduce a local optimal CG (LOCG) acceleration technique

Table 12 Classification accuracy with standard deviation (10% training, 90% testing)

Data	GMPCA	PLS	MCCA	MvPCA	UMvPLS	MvPS-J	MvPS-G
Ads	92.84 ± 1.01	92.23 ± 1.06	83.30 ± 1.98	91.82 ± 0.81	92.31 ± 0.93	93.54 ± 0.78	93.48 ± 0.82
Caltech101-7	93.66 ± 0.88	92.88 ± 0.62	78.88 ± 3.59	84.80 ± 4.94	92.56 ± 0.69	94.11 ± 0.58	94.08 ± 0.50
Caltech101-20	81.33 ± 0.93	81.33 ± 0.93	50.92 ± 2.43	70.96 ± 1.83	81.02 ± 0.73	83.62 ± 1.08	83.64 ± 1.02
Scene15	61.94 ± 1.38	61.94 ± 1.38	30.72 ± 1.91	44.82 ± 2.47	59.76 ± 2.30	72.13 ± 1.44	72.12 ± 1.40

to speed up the iteration. Numerical experiments indeed show significant speedups. It is proved that the SCF iteration with or without LOCG is always convergent. Numerical results indicate that both the SCF iteration and its accelerated version via LOCG converge linearly, but their precise rates of convergence remains to be investigated further. Perhaps, something along the lines of the recent paper [4] could be done.

Numerical comparisons with two solvers from `manopt` [9] and one from `STOP` [17, 54], two widely used MATLAB toolboxes for optimization on manifolds, are presented to demonstrate the effectiveness of our customized methods compared to general-purpose solvers that are carefully adapted from constrained optimization and well written by experts. The two solvers from `manopt` are the steepest-descent (SD) and trust-regions (RTR) methods, and the one from `STOP` is the multiplier correction method (FOForth), where SD and FOForth are first-order methods like ours, while RTR is a second-order method. It is observed that our customized methods are far superior to these general-purpose ones (even the second-order RTR).

Two machine learning applications, MAXBET [35, 53] and MvPS [37], both of which have their optimization subproblems as special cases of (6.1) in their computational kernels, are investigated. In particular, we experiment with MvPS on four real-world data sets and show that properly exploiting both shared latent space and view-specific spaces pays dividends for multi-view learning.

In studying the optimization problem (6.1), we have limited ourselves to real symmetric matrices A_j and real matrices D_j, in part motivated by targeted applications. Mathematically, our development in this paper works equally well for Hermitian matrices A_j and complex matrices D_j, upon straightforward modifications, namely replacing all transposes of vectors/matrices by their complex conjugate transposes.

Acknowledgements The authors wish to thank the two anonymous referees for their constructive suggestions that greatly improved the presentation of this paper. They are indebted to Prof. M. Overton of New York University for his numerous minor but important corrections across the manuscript. Wang was supported in part by NSF DMS-2009689; Zhang was supported in part by the National Natural Science Foundation of China NSFC-12071332; Li was supported in part by NSF DMS-1719620 and DMS-2009689.

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, Princeton (2008)
2. Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Croz, J.D., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S., Sorensen, D.: LAPACK Users' Guide, 3rd edn. SIAM, Philadelphia (1999)
3. Bai, Z., Demmel, J., Dongarra, J., Ruhe, A., van der Vorst, H. (eds.): Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. SIAM, Philadelphia (2000)
4. Bai, Z., Li, R.C., Lu, D.: Sharp estimation of convergence rate for self-consistent field iteration to solve eigenvector-dependent nonlinear eigenvalue problems. SIAM J. Matrix Anal. Appl. **43**(1), 301–327 (2022)
5. Balogh, J., Csendes, T., Rapcsá, T.: Some global optimization problems on Stiefel manifolds. J. Glob. Optim. **30**, 91–101 (2004)
6. Birtea, P., Caşu, I., Comănescu, D.: First order optimality conditions and steepest descent algorithm on orthogonal Stiefel manifolds. Opt. Lett. **13**, 1773–1791 (2019)

7. Bolla, M., Michaletzky, G., Tusnády, G., Ziermann, M.: Extrema of sums of heterogeneous quadratic forms. Linear Algebra Appl. **269**(1), 331–365 (1998). https://doi.org/10.1016/S0024-3795(97)00230-9
8. Borg, I., Lingoes, J.: Multidimensional Similarity Structure Analysis. Springer-Verlag, New York (1987)
9. Boumal, N., Mishra, B., Absil, P.A., Sepulchre, R.: Manopt, a Matlab toolbox for optimization on manifolds. J. Mach. Learn. Res. **15**(42), 1455–1459 (2014)
10. Cai, Y., Zhang, L.H., Bai, Z., Li, R.C.: On an eigenvector-dependent nonlinear eigenvalue problem. SIAM J. Matrix Anal. Appl. **39**(3), 1360–1382 (2018)
11. Chu, M.T., Trendafilov, N.T.: The orthogonally constrained regression revisited. J. Comput. Graph. Stat. **10**(4), 746–771 (2001)
12. Cunningham, J.P., Ghahramani, Z.: Linear dimensionality reduction: survey, insights, and generalizations. J. Mach. Learn. Res. **16**, 2859–2900 (2015)
13. Demmel, J.: Applied Numerical Linear Algebra. SIAM, Philadelphia (1997)
14. Edelman, A., Arias, T.A., Smith, S.T.: The geometry of algorithms with orthogonality constraints. SIAM J. Matrix Anal. Appl. **20**(2), 303–353 (1999)
15. Eldén, L., Park, H.: A procrustes problem on the Stiefel manifold. Numer. Math. **82**, 599–619 (1999)
16. Fan, K.: On a theorem of Weyl concerning eigenvalues of linear transformations. I. Proc. Natl. Acad. Sci. USA **35**(11), 652–655 (1949)
17. Gao, B., Liu, X., Chen, X., Yuan, Y.X.: A new first-order algorithmic framework for optimization problems with orthogonality constraints. SIAM J. Optim. **28**(1), 302–332 (2018). https://doi.org/10.1137/16M1098759
18. Golub, G.H., Van Loan, C.F.: Matrix Computations, 4th edn. Johns Hopkins University Press, Baltimore (2013)
19. Gower, J.C., Dijksterhuis, G.B.: Procrustes Problems. Oxford University Press, New York (2004)
20. Hardoon, D.R., Szedmak, S., Shawe-Taylor, J.: Canonical correlation analysis: an overview with application to learning methods. Neural Comput. **16**, 2639–2664 (2004)
21. Horn, R.A., Johnson, C.R.: Matrix Analysis, 2nd edn. Cambridge University Press, New York (2013)
22. Hotelling, H.: Relations between two sets of variates. Biometrika **28**(3–4), 321–377 (1936)
23. Hurley, J.R., Cattell, R.B.: The Procrustes program: producing direct rotation to test a hypothesized factor structure. Comput. Behav. Sci. **7**, 258–262 (1962)
24. Imakura, A., Li, R.C., Zhang, S.L.: Locally optimal and heavy ball GMRES methods. Jpn. J. Ind. Appl. Math. **33**, 471–499 (2016)
25. Kanzow, C., Qi, H.D.: A QP-free constrained Newton-type method for variational inequality problems. Math. Program. **85**, 81–106 (1999)
26. Knyazev, A.V.: Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method. SIAM J. Sci. Comput. **23**(2), 517–541 (2001)
27. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 2, pp. 2169–2178. IEEE (2006)
28. Li, F.F., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. Comput. Vis. Image Underst. **106**(1), 59–70 (2007)
29. Li, L., Zhang, Z.: Semi-supervised domain adaptation by covariance matching. IEEE Trans. Pattern Anal. Mach. Intell. **41**(11), 2724–2739 (2019). https://doi.org/10.1109/TPAMI.2018.2866846
30. Li, R.C.: New perturbation bounds for the unitary polar factor. SIAM J. Matrix Anal. Appl. **16**, 327–332 (1995)
31. Li, R.C.: Relative perturbation bounds for the unitary polar factor. BIT **37**, 67–75 (1997)
32. Li, R.C.: Matrix perturbation theory. In: Hogben, L., Brualdi, R., Stewart, G.W. (eds.) Handbook of Linear Algebra, 2nd edn, Chapter 21. CRC Press, Boca Raton (2014)
33. Li, R.C.: Rayleigh quotient based optimization methods for eigenvalue problems. In: Bai, Z., Gao, W., Su, Y. (eds.) Matrix Functions and Matrix Equations, Series in Contemporary Applied Mathematics, vol. 19, pp. 76–108. World Scientific, Singapore (2015)
34. Li, Y., Yang, M., Zhang, Z.: A survey of multi-view representation learning. IEEE Trans. Knowl. Data Eng. **31**(10), 1863–1883 (2018)
35. Liu, X.G., Wang, X.F., Wang, W.G.: Maximization of matrix trace function of product Stiefel manifolds. SIAM J. Matrix Anal. Appl. **36**(4), 1489–1506 (2015)

36. Ma, X., Shen, C., Wang, L., Zhang, L.H., Li, R.C.: A self-consistent-field iteration for MAXBET with an application to multi-view feature extraction. Adv. Comput. Math. **48**, 13 (2022)
37. Ma, X., Wang, L., Zhang, L.H., Shen, C., Li, R.C.: Multi-view partially shared subspace learning (2021). Submitted
38. Moré, J., Sorensen, D.: Computing a trust region step. SIAM J. Sci. Stat. Comput. **4**(3), 553–572 (1983)
39. Nie, F., Zhang, R., Li, X.: A generalized power iteration method for solving quadratic problem on the Stiefel manifold. Sci. China Inf. Sci. **60**, 1–10 (2017)
40. Nielsen, A.A.: Multiset canonical correlations analysis and multispectral, truly multitemporal remote sensing data. IEEE Trans. Image Process. **11**(3), 293–305 (2002)
41. Nocedal, J., Wright, S.: Numerical Optimization, 2nd edn. Springer (2006)
42. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
43. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. Int. J. Comput. Vis. **42**(3), 145–175 (2001)
44. Parlett, B.N.: The Symmetric Eigenvalue Problem. SIAM, Philadelphia,: This SIAM edition is an unabridged, corrected reproduction of the work first published by Prentice-Hall Inc, p. 1980. Englewood Cliffs, New Jersey (1998)
45. Polyak, B.T.: Introduction to Optimization. Optimization Software, New York (1987)
46. Rapcsák, T.: On minimization on Stiefel manifolds. Eur. J. Oper. Res. **143**(2), 365–376 (2002)
47. Saad, Y.: Numerical Methods for Large Eigenvalue Problems. Manchester University Press, Manchester (1992)
48. Sharma, A., Kumar, A., Daume, H., Jacobs, D.W.: Generalized multiview analysis: a discriminative latent space. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2160–2167 (2012)
49. Stewart, G.W.: Matrix Algorithms, Eigensystems, vol. II. SIAM, Philadelphia (2001)
50. Sun, J.G.: Matrix Perturbation Analysis. Graduate Texts (Academia, Sinica), 2nd edn. Science Publisher, Beijing (2001). (**in Chinese**)
51. Takahashi, I.: A note on the conjugate gradient method. Inf. Process. Jpn. **5**, 45–49 (1965)
52. Ten Berge, J.M.F.: Generalized approaches to the MAXBET problem and the MAXDIFF problem, with applications to canonical correlations. Psychometrika **53**(4), 487–494 (1984)
53. Van de Geer, J.P.: Linear relations among k sets of variables. Psychometrika **49**(1), 70–94 (1984)
54. Wang, L., Gao, B., Liu, X.: Multipliers correction methods for optimization problems over the Stiefel manifold. CSIAM Trans. Appl. Math. **2**(3), 508–531 (2021). https://doi.org/10.4208/csiam-am.SO-2020-0008
55. Wang, L., Li, R.C.: A scalable algorithm for large-scale unsupervised multi-view partial least squares. IEEE Trans. Big Data (2020). https://doi.org/10.1109/TBDATA.2020.3014937
56. Wen, Z., Yin, W.: A feasible method for optimization with orthogonality constraints. Math. Program. **142**(1–2), 397–434 (2013)
57. Wu, J., Rehg, J.M.: Where am I: place instance and category recognition using spatial pact. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE (2008)
58. Yang, M., Li, R.C.: Heavy ball flexible GMRES method for nonsymmetric linear systems. J. Comput. Math. (2021). To appear
59. Zhang, L.H.: Riemannian trust-region method for the maximal correlation problem. Numer. Funct. Anal. Optim. **33**(3), 338–362 (2012)
60. Zhang, L.H., Li, R.C.: Maximization of the sum of the trace ratio on the Stiefel manifold, I: theory. Sci. China Math. **57**(12), 2495–2508 (2014)
61. Zhang, L.H., Li, R.C.: Maximization of the sum of the trace ratio on the Stiefel manifold, II: computation. Sci. China Math. **58**(7), 1549–1566 (2015)
62. Zhang, L.H., Wang, L., Bai, Z., Li, R.C.: A self-consistent-field iteration for orthogonal canonical correlation analysis. IEEE Trans. Pattern Anal. Mach. Intell. **44**(2), 890–904 (2022). https://doi.org/10.1109/TPAMI.2020.3012541
63. Zhang, L.H., Yang, W.H., Shen, C., Ying, J.: An eigenvalue-based method for the unbalanced Procrustes problem. SIAM J. Matrix Anal. Appl. **41**(3), 957–983 (2020)
64. Zhang, Z., Du, K.: Successive projection method for solving the unbalanced procrustes problem. Sci. China Math. **49**(7), 971–986 (2006)
65. Zhao, H., Wang, Z., Nie, F.: Orthogonal least squares regression for feature extraction. Neurocomputing **216**, 200–207 (2016)

66. Zhou, Y., Li, R.C.: Bounding the spectrum of large Hermitian matrices. Linear Algebra Appl. **435**, 480–493 (2011)