A New Method for Completely Positive Matrix Factorization

LAI ZHIJIAN

(Master's Program in Policy and Planning Sciences)

Advised by Akiko Yoshise

Submitted to the Graduate School of
Systems and Information Engineering
in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Policy and Planning Sciences
at the
University of Tsukuba

March 2021

Abstract

A real symmetric matrix A is called completely positive if there is an entrywise nonnegative matrix B such that $A = BB^T$. The set $\mathcal{CP}_n$ consisting of all $n \times n$ completely positive matrices is called the completely positive cone.

Chapter 1 covers some basic facts about complete positivity in terms of applications, open problems, and mathematical properties. One of the open problems is how to check whether a given matrix is completely positive. As shown by Dickinson and Gijben in 2014, this problem is, unfortunately, NP-hard in general. Even for a given matrix $A \in \mathcal{CP}_n$, it is still hard to find a completely positive (CP) factorization $B \in \mathbb{R}_+^{n\times r}$, so as to provide a completely positive criterion. This thesis is devoted to the CP factorization for a given completely positive matrix.

We propose a new numerical method of CP factorization, which stems from the idea presented by Groetzner and Dür in 2020, wherein the CP factorization problem can be reformulated as a feasibility problem. We describe it in Chapter 2.

As the core of the new method, in Chapter 3, we introduce the outstanding curvilinear search method proposed by Wen and Yin in 2013, which is designed for general orthogonality optimization problem. In Chapter 4, we begin to solve the feasibility problem through the use of curvilinear search method. To be able to apply this method, we use a smooth approximation function named LogSumExp. Furthermore, some improvements are proposed here.

Chapter 5 is a collection of our numerical experiments. It shows that our method is much faster than most of the other CP factorization algorithms. It also is reliable for most completely positive matrices. In the end, in Chapter 6, we summarize this thesis, and then briefly introduce the advantages and disadvantages of our method and new directions that can be explored.

Keywords: completely positive matrix, matrix factorization, orthogonality constrained optimization, smooth approximation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we introduce the copositive and the completely positive matrix cone, which are closely related to many familiar matrix cones. We also show some applications and open problems on the completely positive cone. The purpose of the thesis is to find a new CP factorization method for a given completely positive matrix. To this end, we introduce the notation of the cp-(plus)-rank to obtain more characteristics about the completely positive cone. We will also introduce the existing methods of CP factorization.

1.1 The Copositive and the Completely Positive Cone

Most of the symbols used in this thesis are standard. $\mathbb{R}^n$ is the usual Euclidean space, and $\mathbb{R}^n_+$ is the nonnegative orthant. We consider the space of $m \times n$ real matrices, $\mathbb{R}^{m\times n}$, with inner product defined as $\langle X, Y\rangle := \operatorname{tr}\left(X^T Y\right) = \sum_{i,j=1}^{n} X_{ij}Y_{ij}$ and Frobenius norm defined as $\|X\| := \sqrt{\langle X, X\rangle}$ for all $X, Y \in \mathbb{R}^{m\times n}$. The definitions also reduce to the space of $n \times n$ real symmetric matrices, $\mathcal{S}_n$. A matrix $X \geqslant 0 (X > 0)$ means that every element is nonnegative (positive). A square matrix $Q \in \mathbb{R}^{n\times n}$ is defined to be *orthogonal* if $Q^T Q = I$ or $QQ^T = I$, where I is the identity matrix. For a set $K \subseteq \mathcal{S}_n$, we define:

- K is a *cone* if λX belongs to K for all $X \in K$ and scalar $\lambda \geqslant 0$. Additionally, it is *convex* if $(1-\lambda)X + \lambda Y$ belongs to K for all $X, Y \in K$ and $\lambda \in [0, 1]$. It is a convex cone if both conditions are met.
- A convex cone K is *proper* if it is (i) closed, (ii) pointed (i.e., if both $X, -X \in K$, we get $X = 0$), and (iii) has a nonempty interior, i.e., $\operatorname{int} K \neq \varnothing$.
- The *dual* of K, $K^* := \{X \in \mathcal{S}_n \mid \langle X, Y\rangle \geqslant 0 \text{ for all } Y \in K\}$.
- The *convex hull* of K, $\operatorname{conv}(K) := \{\sum_{i=1}^{n} \lambda_i X_i \mid n \in \mathbb{N}, \forall \lambda_i \geqslant 0, \sum_{i=1}^{n} \lambda_i = 1, \forall X_i \in K\}$.

With the above definitions in hand, we can now introduce the fundamental concepts of this thesis.

1.1.1 Copositive and Completely Positive Matrices

Definition 1.1.1. *A matrix $A \in \mathcal{S}_n$ is called completely positive if for some $r \in \mathbb{N}$, there exists $B \in \mathbb{R}^{n\times r}, B \geqslant 0$, such that $A = BB^T$.*

In this case, $A = BB^T$ is called a CP factorization of A. In this thesis, for convenience we call such B a CP factor of A. We refer to $\mathcal{CP}_n$ as the set of all $n \times n$ completely positive matrices, that is,

$$\mathcal{CP}_n := \left\{A \in \mathcal{S}_n | A = BB^T \text{ where } B \in \mathbb{R}^{n\times r}, B \geqslant 0\right\},$$

or equivalently

$$\mathcal{CP}_n = \left\{BB^T \in \mathcal{S}_n \mid B \text{ is a nonnegative matrix } \right\} = \operatorname{conv}\left\{\mathbf{x}\mathbf{x}^T \mid \mathbf{x} \in \mathbb{R}^n_+\right\},$$

where $\operatorname{conv}(\cdot)$ is the convex hull of a given set. The dual of the set $\mathcal{CP}_n$ gives rise to the set of copositive matrices.

Definition 1.1.2. *A matrix $A \in \mathcal{S}_n$ is called copositive if the quadratic form $\boldsymbol{x}^T A\boldsymbol{x}$ is nonnegative for all nonnegative vectors $\boldsymbol{x}$. We denote the set of all $n \times n$ copositive matrices by*

$$\mathcal{COP}_n := \left\{A \in \mathcal{S}_n \mid \boldsymbol{x}^T A\boldsymbol{x} \geqslant 0 \text{ for all } \boldsymbol{x} \in \mathbb{R}^n_+\right\}.$$

Theorem 1.1.3. *$\mathcal{CP}_n$ and $\mathcal{COP}_n$ are dual to each other in the space $\mathcal{S}_n$, i.e., $\mathcal{CP}_n^* = \mathcal{COP}_n$ and $\mathcal{COP}_n^* = \mathcal{CP}_n$.*

The proof can be found in [29, Theorem 2.6]. This property motivates their joint study. From the definitions, it is immediately apparent that both $\mathcal{CP}_n$ and $\mathcal{COP}_n$ are convex cones. Moreover, both of them are proper; a proof can be found in [18, Chapter 5]. For a comprehensive monograph, the book [1] is recommended.

1.1.2 Relationship with Other Important Matrix Cones

In order to show that these matrix cones are also closely related to other important matrix cones in $\mathcal{S}_n$, consider the following definitions.

Definition 1.1.4. *1. The cone of symmetric nonnegative matrices is defined by*

$$\mathcal{N}_n := \{A \in \mathcal{S}_n \mid A_{ij} \geqslant 0 \text{ for all } i, j = 1, \ldots, n\}.$$

2. The cone of symmetric positive semidefinite matrices is defined by

$$\mathcal{S}_n^+ := \left\{A \in \mathcal{S}_n \mid \boldsymbol{x}^T A\boldsymbol{x} \geqslant 0 \text{ for all } \boldsymbol{x} \in \mathbb{R}^n\right\}.$$

3. A matrix is called doubly nonnegative if it is nonnegative and positive semidefinite at the same time. We denote the set of all such matrices by

$$\mathcal{DNN}_n := \mathcal{S}_n^+ \cap \mathcal{N}_n.$$

4. The Minkowski sum of $\mathcal{S}_n^+$ and $\mathcal{N}_n$ is defined by

$$\mathcal{S}_n^+ + \mathcal{N}_n := \{A + B \in \mathcal{S}_n \mid A \in \mathcal{S}_n^+, B \in \mathcal{N}_n\}.$$

Obviously, every $A \in \mathcal{S}_n^+$ is also copositive by definition. $A \in \mathcal{N}_n$ gives $\mathbf{x}^T A\mathbf{x} \geqslant 0$ for every nonnegative vector $\mathbf{x}$. A trivial fact is that BB^T is positive semidefinite for any matrix B, so every $X \in \mathcal{CP}_n$ is doubly nonnegative. Finally, we have the following inclusion relationship among those cones (see also [16, 35]).

Theorem 1.1.5. *For any positive integer n,*

$$\mathcal{CP}_n \subseteq \mathcal{DNN}_n \subseteq \mathcal{S}_n^+ \subseteq \mathcal{S}_n^+ + \mathcal{N}_n \subseteq \mathcal{COP}_n.$$

In particular, if $n \leqslant 4$, the following holds:

$$\mathcal{CP}_n = \mathcal{DNN}_n \subseteq \mathcal{S}_n^+ \subseteq \mathcal{S}_n^+ + \mathcal{N}_n = \mathcal{COP}_n.$$

To show that equality $\mathcal{S}_n^+ + \mathcal{N}_n = \mathcal{COP}_n$ does not hold for $n \geqslant 5$, consider the Horn matrix,

$$H = \begin{pmatrix} 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 & 1 \end{pmatrix} \in \mathcal{COP}_5 \backslash (\mathcal{S}_5^+ + \mathcal{N}_5),$$

which cannot be decomposed into the sum of a positive semidefinite and a nonnegative matrix, cf [30]. To see that H is copositive, write

$$\begin{aligned} \mathbf{x}^T H\mathbf{x} &= (x_1 - x_2 + x_3 + x_4 - x_5)^2 + 4x_2x_4 + 4x_3(x_5 - x_4) \\ &= (x_1 - x_2 + x_3 - x_4 + x_5)^2 + 4x_2x_5 + 4x_1(x_4 - x_5). \end{aligned}$$

The first expression shows that $\mathbf{x}^T H\mathbf{x} \geqslant 0$ for nonnegative $\mathbf{x}$ with $x_5 \geqslant x_4$, while the second expression shows $\mathbf{x}^T H\mathbf{x} \geqslant 0$ for nonnegative $\mathbf{x}$ with $x_5 < x_4$. Similarly, to show that $\mathcal{DNN}_n = \mathcal{CP}_n$ does not hold for $n \geqslant 5$, a counterexample is

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 1 & 0 & 0 & 1 & 6 \end{pmatrix} \in \mathcal{DNN}_5 \backslash \mathcal{CP}_5.$$

which is not completely positive by using a criterion based on graph theory, cf. [1, Theorem 2.8].

It is easy to show that $\mathcal{S}_n^+ + \mathcal{N}_n$ and $\mathcal{DNN}_n$ are also a pair of dual matrix cones in the space $\mathcal{S}_n$. The dual cone of $\mathcal{S}_n^+$ is itself, leading to a nice property called "self-dual". Clearly, neither $\mathcal{COP}_n$ nor $\mathcal{CP}_n$ have the self-dual property. Fig. 1.1 illustrates the inclusion and duality relationship among those cones.

1.2 Applications and Open Problems

Conic optimization is a subfield of convex optimization that studies the minimization of linear functions over proper cones. A very important concept in the field of conic optimization is duality. From the previous results, we can consider a conic optimization over $\mathcal{COP}_n$ and $\mathcal{CP}_n$.

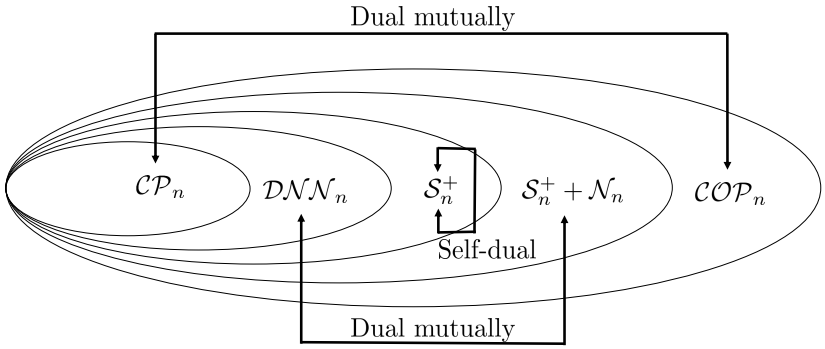

Fig. 1.1. Inclusion and duality relationship among the main matrix cones in the space $\mathcal{S}_n$.

1.2.1 Applications

For some given matrix $C, A_i \in \mathcal{S}_n$ and scalar $b_i \in \mathbb{R}$ for $i = 1, \ldots, m$. A *copositive programming* has the following form:

$$\begin{array}{ll} \min\limits_{X} & \langle C, X \rangle \\ \text{s.t.} & \langle A_i, X \rangle = b_i \quad (i = 1, \ldots, m) \\ & X \in \mathcal{CP}_n. \end{array} \tag{Primal}$$

Interpreting this as the primal problem, one can associate a corresponding dual problem that is a maximization problem over the dual cone. The corresponding dual problem is easily found to be

$$\begin{array}{ll} \max\limits_{y_i} & \sum_{i=1}^m b_i y_i \\ \text{s.t.} & C - \sum_{i=1}^m y_i A_i \in \mathcal{COP}_n. \end{array} \tag{Dual}$$

Copositive programming is nothing but a special case of conic optimization. As is other classical conic programming, namely linear/semidefinite/second-order cone programming, copositive programming covers various problems. In particular, copositive programming is closely related to many nonconvex, NP-hard quadratic and combinatorial optimizations. A short list of examples is:

- Quadratic Problems: standard quadratic problem, quadratic binary problem, fractional quadratic problem, quadratic assignment problem, etc.
- Combinatorial Problems: maximum clique problem, maximum independent set problem, chromatic number, etc.

Since we are mainly interested in the cone of completely positive matrices in this thesis, the corresponding results for $\mathcal{COP}_n$ are omitted here. For surveys on applications of copositive programming, see [9, 12, 14, 22].

Bomze et al. [11] were the first to establish an equivalent completely positive formulation for

the so-called standard quadratic optimization (STQP), i.e.,

$$\begin{array}{ll} \min & \mathbf{x}^T M \mathbf{x} \\ \text{s.t.} & \mathbf{e}^T \mathbf{x} = 1 \\ & \mathbf{x} \in \mathbb{R}^n_+, \end{array} \tag{STQP}$$

where $M \in \mathcal{S}_n$ is possibly not positive semidefinite, and $\mathbf{e}$ denotes the all-ones vector in $\mathbb{R}^n$. They gave the following completely positive reformulation:

$$\begin{array}{ll} \min & \langle M, X \rangle \\ \text{s.t.} & \langle E, X \rangle = 1 \\ & X \in \mathcal{CP}_n. \end{array}$$

Here, $E = \mathbf{e}\mathbf{e}^T$ is the all-ones matrix. Equivalence holds because if we let X be an optimal solution of the completely positive reformulation, then $\mathbf{x}$ is an optimal solution of (STQP) when $\mathrm{rank}(X) = 1$, by $X = \mathbf{x}\mathbf{x}^T$. When $\mathrm{rank}(X) > 1$, i.e., X can be factorized as $X = \sum_{i=1}^r \mathbf{x}_i \mathbf{x}_i^T$, and an appropriately scaled version of each $\mathbf{x}_i$ is an optimal solution of (STQP).

Burer [13] reported a more general result where any nonconvex quadratic problem with binary and continuous variables can be rewritten as a linear program over $\mathcal{CP}_n$. More precisely, it is possible to derive a completely positive reformulation of the following quadratic problem:

$$\begin{array}{ll} \min & \mathbf{x}^T M \mathbf{x} + 2\mathbf{c}^T \mathbf{x} \\ \text{s.t.} & \mathbf{a}_i^T \mathbf{x} = b_i \quad (i = 1, \ldots, m) \\ & \mathbf{x} \in \mathbb{R}^n_+ \\ & x_j \in \{0, 1\} \quad (j \in B), \end{array}$$

where $M \in \mathcal{S}_n$ is possibly not positive semidefinite, and B represents the index subset of binary variables. This question can be equivalently formulated as follows:

$$\begin{array}{ll} \min & \langle M, X \rangle + 2\mathbf{c}^T \mathbf{x} \\ \text{s.t.} & \mathbf{a}_i^T \mathbf{x} = b_i \quad (i = 1, \ldots, m) \\ & \langle \mathbf{a}_i \mathbf{a}_i^T, X \rangle = b_i^2 \quad (i = 1, \ldots, m) \\ & x_j = X_{jj} \quad (j \in B) \\ & \begin{pmatrix} 1 & \mathbf{x}^T \\ \mathbf{x} & X \end{pmatrix} \in \mathcal{CP}_n. \end{array}$$

As an application to combinatorial problems, consider the problem of computing the independence number $\alpha(G)$ of a graph G with n nodes. Recall that an independent set is a set of vertices where no two vertices are adjacent, and the independence number $\alpha(G)$ is the cardinality of the largest possible independent set of a graph; see Fig. 1.2. De Klerk and Pasechnik [15] showed that $\alpha(G)$ is the solution of a maximization problem over $\mathcal{CP}_n$:

$$\alpha(G) = \max \left\{ \langle E, X \rangle \mid \langle A + I, X \rangle = 1, X \in \mathcal{CP}_n \right\},$$

where A is the adjacency matrix of G, and E is the all-ones matrix.

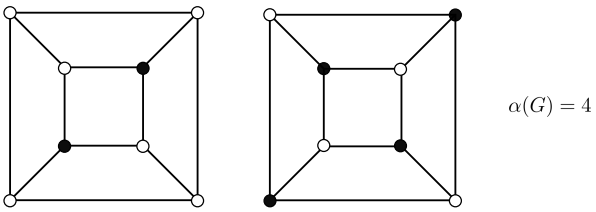

Fig. 1.2. Example of an independent set and independence number $\alpha(G)$.

1.2.2 Open Problems

We have known that many NP-hard problems can reduce to the completely positive formulations. It is easy to see that the difficulty of the original problems lies entirely in the cone constraint since all the others are linear constraints. Note that neither $\mathcal{COP}_n$ nor $\mathcal{CP}_n$ is self-dual. The primal-dual interior point methodology for conic optimization does not work as it is. Besides this fact, there are many fundamental open problems in completely positive cones. Here is a list of these open problems, from [7]:

1. Checking membership in $\mathcal{CP}_n$.
2. Determining geometry of $\mathcal{CP}_n$.
3. Factoring completely positive matrices.
   (a) Finding a factorization of a matrix in $\mathcal{CP}_n$.
   (b) Computing the cp-rank*.
4. Finding cutting planes for completely positive optimization problems.

A typical open problem is the checking membership in $\mathcal{CP}_n$, which was shown to be NP-hard by [20]. For a more detailed discussion of the other unresolved issues, we refer the reader to [7, 22]. One can modify the membership check in $\mathcal{CP}_n$ by minimizing $\langle A, X\rangle$ over $\mathcal{COP}_n$ as $A \in \mathcal{CP}_n$ if and only if $\langle A, X\rangle \geqslant 0$ for all $X \in \mathcal{COP}_n$. Another way is that $A \in \mathcal{CP}_n$ if the optimal value of the following problem is equal to zero.

$$\min_{B \in \mathbb{R}_+^{n\times k}} \|A - BB^T\|^2,$$

where k is large enough. More exactly, $k \geqslant \mathrm{cp}(A)$ (see the next section). Clearly, neither of them is easy to deal with. Here we focus on another issue closely related to the membership problem.
Our goal. *In this thesis, we focus on a fundamental problem of completely positive matrices — finding a CP factorization for the given $A \in \mathcal{CP}_n$.*

*The concept of cp-rank will be presented in the next section

Such a CP factorization offers a natural criterion for whether a matrix is completely positive. Sometimes, in fact, some matrices can be shown to be completely positive through duality, but no CP factorizations can be found for them. On the other hand, it is easy to create an arbitrary completely positive matrix. For example, we can obtain $A := BB^T$ by using a nonnegative matrix B; meanwhile, we naturally have a trivial CP factor B. Nonetheless, finding an extra CP factor is still a challenging task.

1.3 The cp-rank and Interior of $\mathcal{CP}_n$

Now let us establish more characteristics on the factorization of a completely positive matrix.

Lemma 1.3.1. *Let $A \in \mathbb{R}^{m\times n}$; then,*

$$\operatorname{rank}\left(A^{\mathrm{T}}A\right) = \operatorname{rank}\left(AA^{\mathrm{T}}\right) = \operatorname{rank}(A).$$

Proof. We show that the singular values of A^TA and AA^T are nothing but the squares of the singular values of A. Let r be the rank of A. We then have the singular value decomposition of A,

$$A_{m\times n} = U_{m\times r}\Sigma_{r\times r}V^T_{r\times n}.$$

Here, the subscripts mean the number of rows and columns. This gives A^TA and AA^T:

$$A^TA = V\Sigma(U^TU)\Sigma V^T = V_{n\times r}\Sigma^2_{r\times r}V^T_{r\times n},$$
$$AA^T = U\Sigma(V^TV)\Sigma U^T = U_{m\times r}\Sigma^2_{r\times r}U^T_{r\times m}.$$

It follows that A^TA and AA^T also have rank r. □

The next lemma tell us that every CP factor of $A \in \mathcal{CP}_n$ is of the same rank as A.

Lemma 1.3.2. *Given $A \in \mathcal{CP}_n$ with any CP factor B of A, then*

$$\operatorname{rank}(A) = \operatorname{rank}(B).$$

Proof. This is clear from Lemma 1.3.1. □

Example 1.3.3. *Consider the matrix $A \in \mathcal{CP}_3$,*

$$A = \begin{pmatrix} 18 & 9 & 9 \\ 9 & 18 & 9 \\ 9 & 9 & 18 \end{pmatrix}.$$

$A = B_iB_i^T$ for each of the following matrices.

$$B_1 := \begin{pmatrix} 4 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 4 \end{pmatrix}, \quad B_2 := \begin{pmatrix} 3 & 3 & 0 & 0 \\ 3 & 0 & 3 & 0 \\ 3 & 0 & 0 & 3 \end{pmatrix}, B_3 := \begin{pmatrix} 3 & 3 & 0 \\ 3 & 0 & 3 \\ 0 & 3 & 3 \end{pmatrix},$$

$$B_4 := \begin{pmatrix} 3 & 3 & 0 & 0 \\ 3 & 0 & 3 & 0 \\ 0 & 3 & 3 & 0 \end{pmatrix}, \quad B_5 := \begin{pmatrix} 3 & 3 & 0 & 0 & 0 \\ 3 & 0 & 3 & 0 & 0 \\ 0 & 3 & 3 & 0 & 0 \end{pmatrix}.$$

From the example above, generally, one can have many CP factorizations even if the numbers of columns are distinct. Of course, the numbers of rows must consistently be n under the rules of multiplication of matrices. This gives rise to the following definitions.

Definition 1.3.4. *We define the cp-rank of $A \in \mathcal{S}_n$ as the minimum number of columns for all CP factors of A, i.e.,*

$$\operatorname{cp}(A) := \min_B \left\{ r \in \mathbb{N} | \exists B \in \mathbb{R}^{n\times r}, B \geqslant 0, A = BB^T \right\}.$$

Notice that $\operatorname{cp}(A) := \infty$ if $A \notin \mathcal{CP}_n$. We define the cp-plus-rank of $A \in \mathcal{S}_n$ as

$$\operatorname{cp}^+(A) := \min_B \left\{ r \in \mathbb{N} | \exists B \in \mathbb{R}^{n\times r}, B > 0, A = BB^T \right\}.$$

Immediately, we obtain the following inequalities about rank, cp-rank and cp-plus-rank.

Proposition 1.3.5. *For all $A \in \mathcal{S}_n$, we have*

$$\operatorname{rank}(A) \leqslant \operatorname{cp}(A) \leqslant \operatorname{cp}^+(A).$$

Proof. It is clear that $\operatorname{cp}(A) \leqslant \operatorname{cp}^+(A)$ holds by definition. We will show that $\operatorname{rank}(A) \leqslant \operatorname{cp}(A)$. If $A \notin \mathcal{CP}_n$, then $\operatorname{cp}(A) = \infty > \operatorname{rank}(A)$. If $A \in \mathcal{CP}_n$, then according to lemma 1.3.2, we have

$$\operatorname{rank}(A) = \operatorname{rank}(B) \leqslant \text{ the number of columns of } B,$$

for each CP factor B of A. Thus, $\operatorname{rank}(A) \leqslant \operatorname{cp}(A)$. □

It is an open problem to compute the cp-rank or the cp-plus-rank of a matrix. Nevertheless, there are results on the upper bound for the cp-rank of a completely positive matrix. For instance, the following result is the best one we currently have, cf. [10, Theorem 4.1].

Theorem 1.3.6. *For all $A \in \mathcal{CP}_n$, we have*

$$\operatorname{cp}(A) \leqslant \operatorname{cp}_n := \begin{cases} n & \text{for } n \in \{2,3,4\} \\ \frac{1}{2}n(n+1) - 4 & \text{for } n \geqslant 5. \end{cases}$$

Sometimes, we need to distinguish completely positive matrices in either the interior or on the boundary of $\mathcal{CP}^n$. The result below, from [17, Theorem 3.8], is very useful.

Theorem 1.3.7. *We have*

$$\begin{aligned} \operatorname{int} \mathcal{CP}_n &= \left\{ A \in \mathcal{S}_n \mid \operatorname{rank}(A) = n, \operatorname{cp}^+(A) < \infty \right\}, \\ &= \left\{ A \in \mathcal{S}_n \mid \operatorname{rank}(A) = n, A > 0 \right\}, \\ &= \{ A \in \mathcal{S}_n \mid \operatorname{rank}(A) = n, A = BB^T, B \in \mathbb{R}^{n\times r}, B \geqslant 0, \\ &\qquad \boldsymbol{b}_j > 0 \text{ for at least one column } \boldsymbol{b}_j \}. \end{aligned}$$

1.4 Related Work

Many different methods of solving CP factorization problems have been studied. Here, we categorize the relevant literature according to their practical applicability and complexity:

Some methods work well for matrices with specific properties. Dickinson and Dür [19] deal with decomposition of special sparse matrices in $\mathcal{CP}_n$ and show that it can be done in linear time. Sikirić, Schürmann, and Vallentin [41] developed a simplex-like method for a rational CP factorization if the input matrix allows a rational CP factor. Anstreicher, Burer, and Dickinson [18, Section 3.3] used the ellipsoid method; their method works for the matrices of $\operatorname{int}\mathcal{CP}_n$ admitting a rational CP factorization. Bomze [14] showed that a factorization of an $n \times n$ matrix can be constructed if the decomposition of the $(n-1) \times (n-1)$ principal submatrix is known.

Some methods are numerically expensive, i.e., impractical for reasonably big input matrices, although they can compute CP factorizations for general matrices. Expensive subproblems are usually included in their methods. Jarre and Schmallowsky [33] stated a criterion of complete positivity, based on the augmented primal dual method. They intended to solve certain second-order cone problems, where the Lyapunov equations need be solved to get a CP factorization. Nie [36] dealt with the CP factorization problem as a case of an $\mathcal{A}$-truncated K-moment problem, for which Nie developed an algorithm that solves a series of semidefinite optimization problems. Sponsel and Dür [43] considered the problem of projecting a matrix onto the cones of copositive and completely positive matrices by using polyhedral approximations of the cones. With these projections, they devised a method to compute the factorizations of completely positive matrices.

In 2020, Groetzner and Dür [28] applied the alternating projection method to the CP factorization problem under a feasibility form. They aimed to compute a singular value decomposition and solve a second-order cone problem alternatively at every iteration. The numerical performance of their method is higher than that of the previous method.

As will be seen in the next chapter, we propose a new method that is based on the core idea from Groetzner and Dür [28].

Chapter 2

CP factorization as a Feasibility Problem

This chapter consists of two parts. First, we introduce an idea about how to reformulate the CP factorization problem as a feasibility problem. This idea was first mentioned in [31] and was formally established for CP factorization by Groetzner and Dür in [28]. Second, we briefly introduce how Groetzner and Dür deal with that feasibility problem (an approach called alternating projections) and then analyze its drawbacks.

2.1 Rewrite via Feasibility Problem

2.1.1 Some Lemmas

The following two lemmas are simple but necessary.

Lemma 2.1.1. *Given $A \in \mathcal{CP}_n$. If we have a CP factor $B \in \mathbb{R}^{n\times r}$ of A, then for any positive integer $r' \geqslant r$, there is another CP factor $\widehat{B} \in \mathbb{R}^{n\times r'}$ of A.*

Proof. The simplest way to construct an $n \times r'$ matrix $\widehat{B}$ with $A = \widehat{B}\widehat{B}^T$ is to append $k := r' - r$ zero columns to B, i.e.,

$$\widehat{B} := [B, O_{n\times k}] \geqslant 0.$$

For instance, see B_3, B_4, B_5 in Example 1.3.3. Another way to extend B to r' columns is column replication, i.e.,

$$\widehat{B} := [\mathbf{b}_1, \ldots, \mathbf{b}_{n-1}, \underbrace{\frac{1}{\sqrt{m}}\mathbf{b}_n, \ldots, \frac{1}{\sqrt{m}}\mathbf{b}_n}_{m:=r'-n+1 \text{ columns}}], \tag{2.1}$$

where $\mathbf{b}_i$ denotes the i-th column of B. It is easy to verify that $\widehat{B}\widehat{B}^T = BB^T = A$. □

Lemma 2.1.2. *Suppose that $A \in \mathcal{S}_n$, $r \in \mathbb{N}$. Then, $r \geqslant \mathrm{cp}(A)$ if and only if A has a CP factor B with r columns.*

Proof. The "if" part is trivial. The "only if" part follows from Lemma 2.1.1. □

Corollary 2.1.2.1. *For any $A \in \mathcal{CP}_n$, there exists an $n \times \mathrm{cp}_n$ CP factor B, where the constant cp_n is defined in Theorem 1.3.6.*

The following lemma will be used throughout this thesis. We will prove just the "only if" part, since the "if" part is trivial. Unlike other authors who proved the theoretical existence of an orthogonal matrix X (see, e.g., [46, Lemma 1] and [28, Lemma 2.6]), we provide a specific X here.

Lemma 2.1.3. *Let $\mathcal{O}_r$ denote the set of $r \times r$ orthogonal matrices. Suppose that $B, C \in \mathbb{R}^{n\times r}$. Then, $BB^T = CC^T$ if and only if $\exists X \in \mathcal{O}_r$ such that $BX = C$.*

Proof. Suppose that $BB^T = CC^T$. Let k be $\mathrm{rank}(B) = \mathrm{rank}(C)$ because of Lemma 1.3.1.

Case 1. Consider the first special case that $k = n$, i.e., B and C are of full row rank, and $n \leqslant r$. We will find two $(r-n)\times r$ matrices B' and C' such that $BB'^T = O, CC'^T = O$. Let $\mathrm{Null}(B)$ resp. $\mathrm{Null}(C)$ denote the null space of B resp. C. It is clear that $\dim \mathrm{Null}(B) = \dim \mathrm{Null}(C) = r - n$. For B, we construct a matrix B' as below,

$$B' = \begin{pmatrix} \mathbf{b}'_1 \\ \vdots \\ \mathbf{b}'_{r-n} \end{pmatrix},$$

where the row vectors $\mathbf{b}'_j$'s are an arbitrary orthogonal basis of $\mathrm{Null}(B)$. This gives

$$BB'^T = \begin{pmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_n \end{pmatrix} \begin{pmatrix} \mathbf{b}'_1 \\ \vdots \\ \mathbf{b}'_{r-n} \end{pmatrix}^T = \begin{pmatrix} \mathbf{b}_1\mathbf{b}_1'^T & \cdots & \mathbf{b}_1\mathbf{b}_{r-n}'^T \\ \vdots & & \vdots \\ \mathbf{b}_n\mathbf{b}_1'^T & \cdots & \mathbf{b}_n\mathbf{b}_{r-n}'^T \end{pmatrix} = O,$$

where $\mathbf{b}_i$ denotes the i-th row of B for every $i \in \{1, \ldots, n\}$. The same operation can be applied to C, we obtain C' such that $CC'^T = O$. Moreover, it is clear that $B'B'^T = C'C'^T = I$ by their construction. Now, let us construct the desired orthogonal matrix X by

$$X := B^T(BB^T)^{-1}C + B'^T(B'B'^T)^{-1}C'.$$

Because B, B' are of full row rank, $BB^T, B'B'^T$ are invertible and X is well-defined. It is clear that

$$\begin{aligned} BX &= BB^T(BB^T)^{-1}C + BB'^T(B'B'^T)^{-1}C' \\ &= C + O = C. \end{aligned}$$

Hence, it remains to prove that X is an orthogonal matrix. We observe that

$$\begin{aligned} XX^T &= \left[B^T(BB^T)^{-1}C + B'^T(B'B'^T)^{-1}C'\right]\left[C^T(BB^T)^{-1}B + C'^T(B'B'^T)^{-1}B'\right] \\ &= B^T(BB^T)^{-1}CC^T(BB^T)^{-1}B + B^T(BB^T)^{-1}CC'^T(B'B'^T)^{-1}B' + \\ &\quad B'^T(B'B'^T)^{-1}C'C^T(BB^T)^{-1}B + B'^T(B'B'^T)^{-1}C'C'^T(B'B'^T)^{-1}B' \\ &= B^T(BB^T)^{-1}B + B'^T(B'B'^T)^{-1}B'. \end{aligned}$$

Let the singular value decomposition of B and B' be

$$\begin{aligned} B_{n\times r} &= U_{n\times n}\Sigma_{n\times n}V^T_{n\times r}, \\ B'_{(r-n)\times r} &= U'_{(r-n)\times(r-n)}\Sigma'_{(r-n)\times(r-n)}V'^T_{(r-n)\times r}, \end{aligned}$$

where U and U' are orthogonal matrices, V and V' have orthonormal columns. Then, $BB^T = U\Sigma^2U^T$ and $(BB^T)^{-1} = U(\Sigma^{-1})^2U^T$, which in turn give

$$B^T(BB^T)^{-1}B = V\Sigma U^T U(\Sigma^{-1})^2U^T U\Sigma V^T = VV^T.$$

Similarly,

$$B'^T(B'B'^T)^{-1}B' = V'V'^T.$$

Finally, we get

$$XX^T = VV^T + V'V'^T = \begin{pmatrix} V & V' \end{pmatrix}\begin{pmatrix} V & V' \end{pmatrix}^T.$$

We complete part (1) by showing $r \times r$ square matrix $[V_{r\times n}\ V'_{r\times(r-n)}]$ is orthogonal. Notice that $BB'^T = (U\Sigma)V^TV'(\Sigma'U'^T) = O$. Since both $U\Sigma$ and $\Sigma'U'^T$ are invertible, the above equation holds if and only if $V^TV' = O$. We find that

$$\begin{pmatrix} V & V' \end{pmatrix}^T\begin{pmatrix} V & V' \end{pmatrix} = \begin{pmatrix} V^TV & V^TV' \\ V'^TV & V'^TV' \end{pmatrix} = \begin{pmatrix} I_n & O \\ O & I_{r-n} \end{pmatrix}.$$

Case 2. In the case that $k < n$, let

$$\bar{B} := PB = \begin{pmatrix} B^*_{k\times r} \\ B'_{(n-k)\times r} \end{pmatrix},$$

where P is a product of a series of elementary row permutation matrices (hence orthogonal) such that the first k rows of $\bar{B}$ are linearly independent; i.e., B^* is of full row rank. B' represents the remaining rows. For the same permutation P, let

$$\bar{C} := PC = \begin{pmatrix} C^*_{k\times r} \\ C'_{(n-k)\times r} \end{pmatrix},$$

where C^* denotes the first k rows of $\bar{C}$ and C' represents the remaining rows. Next, we show that C^* is also full row rank. Note that

$$\bar{B}\bar{B}^T = PBB^TP^T = PCC^TP^T = \bar{C}\bar{C}^T,$$

and

$$\bar{B}\bar{B}^T = \begin{pmatrix} B^* \\ B' \end{pmatrix}\begin{pmatrix} B^{*T} & B'^T \end{pmatrix} = \begin{pmatrix} B^*B^{*T} & B^*B'^T \\ B'B^{*T} & B'B'^T \end{pmatrix},$$
$$\bar{C}\bar{C}^T = \begin{pmatrix} C^* \\ C' \end{pmatrix}\begin{pmatrix} C^{*T} & C'^T \end{pmatrix} = \begin{pmatrix} C^*C^{*T} & C^*C'^T \\ C'C^{*T} & C'C'^T \end{pmatrix}.$$

This implies that $B^*B^{*T} = C^{*T}\ C^{*T}$; hence, C^* is also full row rank by Lemma 1.3.1. Since B^*, C^* have rank k, then $\exists X, Y \in \mathbb{R}^{r\times k}$ such that $B' = XB^*, C' = YC^*$. In fact, all the rows of B' are linearly dependent of the rows of B^* (similarly for C' and C^*). We claim that $X = Y$. Notice that

$$C'C^{*T} = YC^*C^{*T} = YB^*B^{*T},$$

and

$$B'B^{*T} = XB^*B^{*T}.$$

Because of $C'C^{*T} = B'B^{*T}$, we have

$$\begin{aligned} XB^*B^{*T}(B^*B^{*T})^{-1} &= YB^*B^{*T}(B^*B^{*T})^{-1} \\ X &= Y. \end{aligned}$$

We rewrite the matrices

$$\bar{B} = \begin{pmatrix} B^* \\ XB^* \end{pmatrix} = \begin{pmatrix} I \\ X \end{pmatrix} B^* \text{ and } \bar{C} = \begin{pmatrix} C^* \\ XC^* \end{pmatrix} = \begin{pmatrix} I \\ X \end{pmatrix} C^*.$$

For the full row rank matrices B^*, C^*, from part (1), there exits an $r \times r$ orthogonal matrix Q such that $B^*Q = C^*$. We will show that $BQ = C$ holds for the same Q.

$$\bar{B}Q = \begin{pmatrix} I \\ X \end{pmatrix} B^*Q = \begin{pmatrix} I \\ X \end{pmatrix} C^* = \bar{C}.$$

Hence,

$$\begin{aligned} (P^{-1}\bar{B})Q &= (P^{-1}\bar{C}) \\ BQ &= C. \end{aligned}$$

□

2.1.2 CP factorization as a Feasibility Problem

The next proposition puts all the previous results together to explain why we can rewrite the original question.

Proposition 2.1.4. *Let $A \in \mathcal{CP}_n, r \geqslant \mathrm{cp}(A), A = BB^T$, where $B \in \mathbb{R}^{n\times r}$ is not nonnegative. Then there exists an orthogonal matrix $X \in \mathcal{O}_r$ such that $BX \geqslant 0$, $A = (BX)(BX)^T$.*

Proof. We have

$$\begin{aligned} r \geqslant \mathrm{cp}(A) &\iff \exists B' \in \mathbb{R}^{n\times r}, B' \geqslant 0, A = B'B'^T. \\ &\implies A = B'B'^T = BB^T. \\ &\implies \exists X \in \mathcal{O}_r \text{ such that } BX = B' \geqslant 0. \end{aligned}$$

□

This lemma tells us that one can find an orthogonal matrix X which can take a "bad" factorization into a "good" factorization. Thus, the task of finding a CP factorization of A can be formulated as the following feasibility problem:

$$\begin{array}{ll} \text{find} & X \\ \text{s.t.} & BX \geqslant 0 \\ & X \in \mathcal{O}_r, \end{array} \tag{FeasP}$$

where $r \geqslant \mathrm{cp}(A)$, $B \in \mathbb{R}^{n\times r}$ is an arbitrary initial factorization $A = BB^T$ and not nonnegative.

We must notice that the condition $r \geqslant \mathrm{cp}(A)$ is necessary; otherwise, (FeasP) has no solution, although $A \in \mathcal{CP}_n$. Regardless of $\mathrm{cp}(A)$, one can use p_n or other upper bound. Moreover, finding an initial matrix B is very easy. One can use Cholesky decomposition or spectral decomposition and then extend its columns to r columns by using the technique in Lemma 2.1.1.

The following theorem shows that the feasibility of (FeasP) is precisely a criterion for complete positivity.

Theorem 2.1.5. *We have*

$$A \in \mathcal{CP}_n \Longleftrightarrow (FeasP) \textit{ is feasible.}$$

In this case, for any feasible solution X, we have $A = (BX)(BX)^T$ with $BX \geqslant 0$.

Proof. The "if" part is obvious. The "only if" part is derived using Lemma 2.1.4. □

2.2 Alternating Projections to CP factorization

Now, solving (FeasP) is the key to finding a CP factorization. Groetzner and Dür [28] applied the so-called alternating projection method. This method obtains points in the intersection of two or more sets. For an introduction, the reader is referred to [5, 21].

2.2.1 Applying Alternating Projections

Groetzner added the polyhedral cone,

$$\mathcal{P} := \left\{X \in \mathbb{R}^{r\times r} | BX \geqslant 0\right\},$$

and rewrote (FeasP) as

$$\begin{array}{ll} \text{find} & X \\ \text{s.t.} & X \in \mathcal{P} \cap \mathcal{O}_r. \end{array}$$

Here, the alternating projections method to solve it is as follows: choose a starting point $X_0 \in \mathcal{O}_r$; then compute $P_0 = \mathrm{proj}_{\mathcal{P}}(X_0)$, compute $X_1 = \mathrm{proj}_{\mathcal{O}_r}(P_0)$, and iterate this process (see Fig 2.1). In order to apply this method, we need to project onto the sets $\mathcal{P}$ and $\mathcal{O}_r$.

Since $\mathcal{P}$ is a polyhedral cone and hence convex, the projection of a matrix $M \in \mathbb{R}^{r\times r}$ onto $\mathcal{P}$ is unique, and computing it amounts to solving a second-order cone problem (SOCP). We have

$$\begin{array}{ll} \min & \|X - M\| \\ \text{s.t.} & BX \geqslant 0, \end{array}$$

which can be transformed as

$$\begin{array}{ll} \min & t \\ \text{s.t.} & BY \geqslant -BM \\ & (t, \mathrm{vec}(Y)) \in SOC_{r^2+1}, \end{array} \qquad \text{(SOCP)}$$

where $Y := X - M$, $\mathrm{vec}(Y)$ is the $r^2 + 1$ column vector obtained by stacking the columns of the matrix Y on top of one another, and the second-order cone of order n is defined as $SOC_n := \left\{(t, \mathbf{x}) \in \mathbb{R} \times \mathbb{R}^{n-1} \mid t \geqslant \|\mathbf{x}\|_2\right\}$.

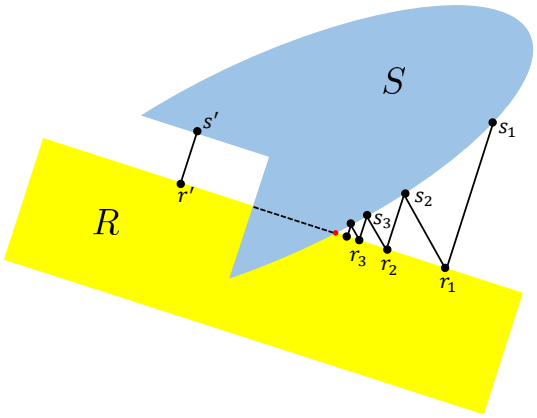

Fig. 2.1. Alternating projections between a closed convex set and a compact but nonconvex set.

On the other hand, the projection of a matrix M onto $\mathcal{O}_r$ always exists since $\mathcal{O}_r$ is compact; however it may not be unique because of the nonconvexity* of $\mathcal{O}_r$. $\mathrm{proj}_{\mathcal{O}_r}(M)$ can be computed through the following lemma, a proof of which can be found in [8, Corollary 5.6.4 and Fact 9.9.42].

Lemma 2.2.1. *Let $M \in \mathbb{R}^{r\times r}$. Then there exists the so-called polar decomposition of M; i.e., there exist a positive semidefinite matrix $T \in \mathbb{R}^{r\times r}$ and an orthogonal matrix $Q \in \mathbb{R}^{r\times r}$ such that $M = TQ$. We have*

$$\|M - Q\| \leqslant \|M - U\| \text{ for all } U \in \mathcal{O}_r.$$

We take this Q as $\mathrm{proj}_{\mathcal{O}_r}(M)$ and compute $Q := UV^T$ from the singular value decomposition of M, since $M = U\Sigma V^T = \left(U\Sigma U^T\right)\left(UV^T\right) = TQ$. Finally, the alternating projection to compute the factorization of a completely positive matrix now reads as:

Algorithm 1: CP factorization by alternating projection

Given $A = BB^T$ with $B \in \mathbb{R}^{n\times r}$ and $r \geqslant \mathrm{cp}(A)$; initial matrix $Q_0 \in \mathcal{O}_r$, $k \leftarrow 0$.
while $BQ_k \ngeqslant 0$ **do**
  $P_k \leftarrow \mathrm{proj}_{\mathcal{P}}(Q_k)$ *(solve an SOCP)*;
  $Q_{k+1} \leftarrow \mathrm{proj}_{\mathcal{O}_r}(P_k)$ *(compute an SVD decomposition)*;
  $k \leftarrow k + 1$;
end

Local convergence can be ensured by the following theorem. Global convergence fails; for example, see s', r' in Fig 2.1.

Theorem 2.2.2 ([28, Theorem 4.2])**.** *Let $A \in \mathcal{CP}_n$, $A = BB^T$ be any initial factorization with $B \in \mathbb{R}^{n\times r}$ and $r \geqslant \mathrm{cp}(A)$. If one starts at a point Q_0 sufficiently close to $\mathcal{P} \cap \mathcal{O}_r$, Algorithm 1 converges to a point $Q^* \in \mathcal{P} \cap \mathcal{O}_r$. In this case, $A = (BQ^*)(BQ^*)^T$ is a completely positive factorization of A.*

*To see nonconvexity of $\mathcal{O}_r$, note that the zero matrix $O = \frac{1}{2}X + \frac{1}{2}(-X)$ is not in $\mathcal{O}_r$ for a matrix $X \in \mathcal{O}_r$.

2.2.2 Modifying Alternating Projections

Although this algorithm has the local convergence property, in practice it usually fails because it is difficult to find a starting point close enough to a solution. For some hard instances, it seems impossible to find any starting point, and thus, the algorithm fails in those instances.

Moreover, we need to solve a second-order cone problem alternately at every iteration. Although solving an SOCP can be done in polynomial time, it is still very costly overall. Clearly, if subproblems have to be solved in each iteration, algorithm will always be slow. After all, the subproblem itself typically needs an iterative method rather than a closed-form computation by arithmetical operations, for example computing an SVD decomposition. For this reason, ultimately, they provided a modified alternating projections for CP factorization. Instead of computing the projection $\mathrm{proj}_{\mathcal{P}}(Q)$ of Q onto $\mathcal{P}$, they take $\widehat{P}$ as an approximation of $\mathrm{proj}_{\mathcal{P}}(Q)$. Define

$$\widehat{P} := B^{+}D + \left(I - B^{+}B\right)Q,$$

where $D \in \mathbb{R}^{n\times r}$ is defined through $D_{ij} := \max\{(BQ)_{ij}, 0\}$, and B^{+} denotes the Moore Penrose inverse of B. The numerical experiments show that Algorithm 2 is often faster than Algorithm 1, but the local convergence property was lost as a result.

Algorithm 2: CP factorization by modified alternating projection

Given $A = BB^{T}$ with $B \in \mathbb{R}^{n\times r}$ and $r \geqslant \mathrm{cp}(A)$; initial matrix $Q_0 \in \mathcal{O}_r$, $k \leftarrow 0$.
while $BQ_k \not\geqslant 0$ **do**
  $D \leftarrow \max\{BQ_k, 0\}$ entrywise ;
  $\widehat{P_k} \leftarrow B^{+}D + (I - B^{+}B)\,Q_k$ *(compute Moore Penrose inverse B^{+})* ;
  $Q_{k+1} \leftarrow \mathrm{proj}_{\mathcal{O}_r}(\widehat{P_k})$ *(compute an SVD decomposition)*;
  $k \leftarrow k + 1$;
end

Recall that our goal is solving (FeasP); that is the key to finding a CP factorization. Naturally, we would wonder: is there any other way to solve (FeasP), instead of alternating projections? This leads to the topic of the next chapter — a curvilinear search method that we want to apply to (FeasP).

Chapter 3

Curvilinear Search on the Stiefel Manifold

In this chapter, we introduce the curvilinear search method proposed by Wen and Yin [45], which is designed for general optimization with orthogonality constraints:

$$\min_{X \in \mathcal{M}_n^p} \mathcal{F}(X), \tag{StOp}$$

where $\mathcal{F}(X) : \mathbb{R}^{n\times p} \to \mathbb{R}$ is continuously differentiable, and the feasible set

$$\mathcal{M}_n^p := \left\{X \in \mathbb{R}^{n\times p} : X^T X = I\right\}$$

is often called the *Stiefel manifold*, on which optimization problems have a variety of applications, including matrix rank minimization, polynomial optimization, sparse principal component analysis, eigenvalue problems, p-harmonic flows, combinatorial optimization, etc.

To keep the thesis self-contained, we provide exhaustive proofs throughout this chapter.

3.1 Optimality Conditions and Update Scheme

We start with two auxiliary lemmas. The following lemma states the most important property of the feasible set $\mathcal{M}_n^p$. Note that $\mathcal{M}_n^p$ is just the set of orthogonal matrices if $p = n$ and is the unit sphere of $\mathbb{R}^n$ if $p = 1$.

Lemma 3.1.1. *The feasible set* $\mathcal{M}_n^p := \left\{X \in \mathbb{R}^{n\times p} : X^T X = I\right\}$ *is a compact subset in the space* $\mathbb{R}^{n\times p}$.

Proof. Define $f : \mathbb{R}^{n\times p} \to \mathcal{S}_p, X \mapsto X^T X$. Then, $\mathcal{M}_n^p = f^{-1}(\{I\})$, where I is the identity matrix. Continuity follows from that the every entry of $f(X)$ is a polynomial of the entries of X. Since f is continuous, $\mathcal{M}_n^p$ is closed as a preimage of a singleton.

In particular, $X^T X = I$ if and only if all columns of X are orthonormal, so $\|X\| = \sqrt{r}$. Let $X, Y \in \mathbb{R}^{n\times p}$; then, $\|X - Y\| \leqslant \|X\| + \|Y\| = 2\sqrt{r}$, which means boundedness. In the space $\mathbb{R}^{n\times p}$, a subset K is compact if and only if it is closed and bounded. □

A Cayley transformation can map *skew-symmetric* matrices, i.e., $W^T = -W$, to orthogonal matrices, which is a well-known property in matrix analysis (see, for instance, [26, P2.1.9]).

Lemma 3.1.2 (Cayley transformation)**.** *Given any skew-symmetric matrix $W \in \mathbb{R}^{n\times n}$, the matrix $Q := (I + W)^{-1}(I - W)$ is well-defined and orthogonal.*

Proof. Note that $\mathbf{x}^T W\mathbf{x} = 0$ holds for any $\mathbf{x} \in \mathbb{R}^n$, which follows from

$$\mathbf{x}^T W\mathbf{x} = \mathbf{x}^T W^T\mathbf{x} = -\mathbf{x}^T W\mathbf{x} \Rightarrow 2(\mathbf{x}^T W\mathbf{x}) = 0.$$

Then, $\mathbf{x}^T(I + W)\mathbf{x} = \|\mathbf{x}\|_2^2 > 0$ holds for any nonzero $\mathbf{x} \in \mathbb{R}^n$. Suppose that there exists a nonzero $\mathbf{x} \in \mathbb{R}^n$ such that $(I + W)\mathbf{x} = 0$, then $\mathbf{x}^T(I + W)\mathbf{x} = 0$, which is a contradiction. Thus, $(I + W)$ is invertible and Q is well-defined. On the other hand, if W is skew-symmetric, so is $-W$; thus, $(I - W)$ is invertible too.

Since $(I + B)(I - B) = (I - B)(I + B)$ for any matrix B, if we replace B with W and take inverse of both sides, we have $(I - W)^{-1}(I + W)^{-1} = (I + W)^{-1}(I - W)^{-1}$. Hence,

$$\begin{aligned} Q^T Q &= (I - W)^T(I + W)^{-T}(I + W)^{-1}(I - W) \\ &= (I + W)(I - W)^{-1}(I + W)^{-1}(I - W) \\ &= (I + W)(I + W)^{-1}(I - W)^{-1}(I - W) = I. \end{aligned}$$

□

3.1.1 First-order Optimality Conditions

The Lagrangian function of (StOp) is

$$\mathcal{L}(X, \Lambda) = \mathcal{F}(X) - \frac{1}{2}\operatorname{tr}\left(\Lambda\left(X^T X - I\right)\right),$$

where $\Lambda \in \mathcal{S}_p$ is a Lagrange multiplier and $\operatorname{tr}(\cdot)$ denotes the trace of matrix. Note that since $X^T X$ is symmetric, the Lagrange multiplier Λ corresponding to $X^T X = I$ is a symmetric matrix. The second term $\operatorname{tr}\left(\Lambda\left(X^T X - I\right)\right)$ comes from $\langle\Lambda, X^T X - I\rangle$.

Let the gradient of $\mathcal{F}(X)$ at point X be $G := \mathcal{D}\mathcal{F}(X) = \left(\frac{\partial\mathcal{F}(X)}{\partial X_{i,j}}\right)$, which is continuous in X. Differentiating the Lagrangian function with respect to X and Λ yields

$$\begin{aligned} \mathcal{D}_X\mathcal{L}(X, \Lambda) &= \mathcal{D}\mathcal{F}(X) - \frac{1}{2}\left(\frac{\partial\operatorname{tr}(\Lambda X^T X)}{\partial X} - \frac{\partial\operatorname{tr}(\Lambda)}{\partial X}\right) = G - \frac{1}{2}(X\Lambda^T + X\Lambda) \\ &= G - X\Lambda, \end{aligned}$$

and

$$\mathcal{D}_\Lambda\mathcal{L}(X, \Lambda) = X^T X - I,$$

where we use the fact $\frac{\partial\operatorname{tr}(BX^T X)}{\partial X} = XB^T + XB$, cf. [38, (112) Page 13]. Imitating the equality constraints problem of vector form, we give the first-order optimality conditions of (StOp) below.

Lemma 3.1.3 (First-order optimality conditions [45, Lemma 1.])**.**

- *Suppose that X is a local minimizer of (StOp). Then, X satisfies the first-order optimality conditions*
$$G - XG^TX = 0 \text{ and } X^TX = I,$$
with the associated Lagrange multiplier $\Lambda = G^TX$.

- *Define*
$$\nabla\mathcal{F}(X) := G - XG^TX \text{ and } A := GX^T - XG^T.$$
Then, $\nabla\mathcal{F}(X) = AX$. Moreover, $\nabla\mathcal{F}(X) = 0$ if and only if $A = 0$.

Proof. It follows from $X^TX = I$ that the linear independence constraint qualification is satisfied. Hence, there exists a Lagrange multiplier Λ such that

$$\begin{aligned} \mathcal{D}_X\mathcal{L}(X,\Lambda) =& G - X\Lambda = 0, \\ \mathcal{D}_\Lambda\mathcal{L}(X,\Lambda) =& X^TX - I = 0. \end{aligned} \tag{3.1}$$

Left multiplying both sides of (3.1) by X^T and using $X^TX = I$, we have $\Lambda = X^TG$. Since Λ must be a symmetric matrix, we obtain $\Lambda = G^TX$ and $\mathcal{D}_X\mathcal{L}(X,\Lambda) = G - XG^TX = 0$. It is easy to verify the last two statements. □

In a word, it is follows from the optimality conditions that we aim to find a feasible point X such that X satisfies $\nabla\mathcal{F}(X) := G - XG^TX = 0$ or equivalently $A := GX^T - XG^T = 0$. Remember that G is the gradient of $\mathcal{F}(X)$ at point X, depending on X.

3.1.2 Update Scheme

Next, we review the update scheme. Recall that a *smooth curve* on a set S is a smooth mapping: $\mathbb{R} \to S$, the real argument is often called the *step size*.

At a point X on the Stiefel manifold $\mathcal{M}_n^p$, we construct a curve $Y(\tau)$ on $\mathcal{M}_n^p$ starting from X. This means that the curve goes through X, and we obtain X at zero step size; in addition, the curve maintains orthogonality with an arbitrary step size; i.e., the image of $Y(\tau)$ is contained in $\mathcal{M}_n^p$.

Simultaneously, as long as the point X is not a local minimizer of (StOp), the objective value can become smaller along this curve at a certain step size; i.e., we are able to find an suitable $\bar{\tau}$ such that

$$\mathcal{F}\left(Y\left(\bar{\tau}\right)\right) < \mathcal{F}\left(Y(0)\right).$$

In fact, the composition map $(\mathcal{F} \circ Y)(\tau) = \mathcal{F}\left(Y\left(\tau\right)\right)$ is just a real-valued function on $\mathbb{R}$. If $(\mathcal{F} \circ Y)'(0) := \left.\frac{d\mathcal{F}(Y(\tau))}{d\tau}\right|_{\tau=0} < 0$ holds, then a positive $\bar{\tau}$ that satisfies the condition exists by basic calculus.

Obviously, the above framework is precisely the classical gradient descent method of unconstrained optimization. Let us consider the unconstrained optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}), \tag{UnOp}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuously differentiable. We want to produce a sequence $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$, where $\mathbf{x}_{k+1}$ is generated from $\mathbf{x}_k$, the current direction $\mathbf{d}_k$, and the step size $\alpha_k > 0$ by the rule,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k.$$

In general, α_k is chosen so that $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$. Fig. 3.1 illustrates the process of the classical gradient descent for some real-valued function defined on $\mathbb{R}^2$. Likewise, Fig. 3.2 shows the process of the curve search for some real-valued function defined on the unit sphere $\mathcal{M}_3^1 \subseteq \mathbb{R}^3$. The difference is that one searches along a straight line, the other along a curve. Whether it is a straight line or a curve, they all search within the feasible region, and each iteration finds a better result. Hence, we can recast the constrained optimization as an unconstrained one. An interesting insight is that $f(\mathbf{x})$ in (UnOp) can be regarded as an optimization constrained on $\mathbb{R}^n$, or conversely, $\mathcal{F}(X)$ in (StOp) as one unconstrained on $\mathcal{M}_n^p$.

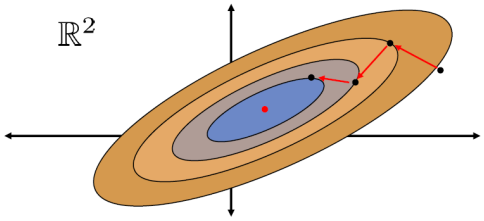

Fig. 3.1. Illustration of classical gradient descent on a contour plot of $\mathbb{R}^2$ plane.

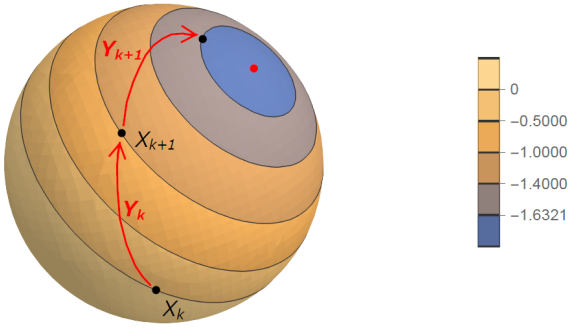

Fig. 3.2. Illustration of curvilinear search on a contour plot of the unit sphere $\mathcal{M}_3^1$.

The next lemma gives us a good way to construct such a curve on $\mathcal{M}_n^p$.

Lemma 3.1.4 (Update scheme [45, Lemma 3.])**.**

1. Let X be a feasible point. Given any skew-symmetric matrix $W \in \mathbb{R}^{n\times n}$, $Y(\tau) : \mathbb{R} \to \mathbb{R}^{n\times p}$, defined below, satisfies $Y(\tau)^T Y(\tau) = X^T X$ for any τ and $Y(0) = X$,

$$Y(\tau) := \left(I + \frac{\tau}{2}W\right)^{-1}\left(I - \frac{\tau}{2}W\right)X, \tag{3.2}$$

or in an equivalent implicit form,

$$Y(\tau) = X - \tau W\left(\frac{X + Y(\tau)}{2}\right). \tag{3.3}$$

It is a smooth curve on the Stiefel manifold $\mathcal{M}_n^p$, and its derivative with respect to τ is

$$Y'(\tau) = -\left(I + \frac{\tau}{2}W\right)^{-1} W\left(\frac{X + Y(\tau)}{2}\right). \tag{3.4}$$

In particular, $Y'(0) = -WX$.

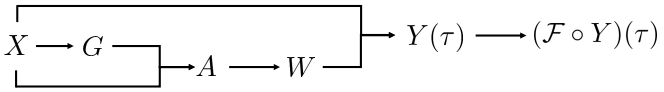

Fig. 3.3. Diagram of update scheme at each iteration.

2. *Let* $W = A := GX^T - XG^T$. *Then,* $Y(\tau)$ *is a descent curve at* $\tau = 0$, *that is,*

$$(\mathcal{F} \circ Y)'(0) := \left.\frac{d\mathcal{F}(Y(\tau))}{d\tau}\right|_{\tau=0} = -\frac{1}{2}\|A\|^2.$$

Moreover, as long as X *is not yet a local minimizer,* $(\mathcal{F} \circ Y)'(0) < 0$ *holds.*

Proof. **Part 1.** Since $\frac{\tau}{2}W$ is skew-symmetric for any real τ, Lemma 3.1.2 gives $Y(\tau)^T Y(\tau) = X^T X$. In particular, $Y(\tau)^T Y(\tau) = I$ for any real τ; thus, $Y(\tau)$ is a curve in $\mathcal{M}_n^p$. It can be seen from (3.2) that it is smooth with respect to τ. It is easy to verify that (3.2) and (3.3) are equivalent and $Y(0) = X$. Differentiating both sides of (3.3) with respect to τ, we obtain (3.4).

Part 2. It is easy to see $A := GX^T - XG^T$ is skew-symmetric. Using the chain rule, we obtain

$$(\mathcal{F} \circ Y)'(\tau) = \langle \mathcal{D}\mathcal{F}(Y(\tau)), Y'(\tau) \rangle = \operatorname{tr}\left(\mathcal{D}\mathcal{F}(Y(\tau))^T Y'(\tau)\right).$$

At $\tau = 0, \mathcal{D}\mathcal{F}(Y(0)) = \mathcal{D}\mathcal{F}(X) = G$ and $Y'(0) = -WX = -\left(GX^T - XG^T\right)X$. Thus, we have

$$(\mathcal{F} \circ Y)'(0) = -\operatorname{tr}\left(G^T\left(GX^T - XG^T\right)X\right) = \operatorname{tr}\left(G^T X G^T X - G^T G\right)$$

and

$$\begin{aligned} -\frac{1}{2}\|A\|^2 = -\frac{1}{2}\operatorname{tr}(A^T A) &= -\frac{1}{2}\operatorname{tr}\{(XG^T - GX^T)(GX^T - XG^T)\} \\ &= \operatorname{tr}\left(G^T X G^T X - G^T G\right). \end{aligned}$$

Finally, we obtain $(\mathcal{F} \circ Y)'(0) = -\frac{1}{2}\|A\|^2$. Since $\nabla\mathcal{F}(X) = 0$ if and only if $A = 0$, so if X is not yet a local minimizer, then $A \neq 0$ and $(\mathcal{F} \circ Y)'(0) < 0$.

□

From the lemma above, for any step size τ, the matrix $Y(\tau)$ keeps orthogonality. We should notice that the curve $Y(\tau)$, defined by an arbitrary skew-symmetric matrix $W \in \mathbb{R}^{n\times n}$, has this property. But if the set $W = A := GX^T - XG^T$, then $Y(\tau)$ is a descent curve at $\tau = 0$. (We may well wonder if there are other Ws such that $(\mathcal{F} \circ Y)'(0) < 0$, but this is not the topic of this thesis.)

Notice that by setting $W = A$, the curve $Y(\tau)$ is completely determined by the current point X (see Fig. 3.3). If X_k denotes the current point at the k-th iteration, the subscript k is used for $Y_k(\tau)$, and similarly for G_k,A_k,W_k,$Y_k(\tau)$, and $(\mathcal{F} \circ Y_k)(\tau)$.

3.2 Curvilinear Search Algorithm

3.2.1 Monotone Curvilinear Search

At iteration k, one can choose the step size by finding a $\tau_k > 0$ satisfying the Armijo-Wolfe conditions:

$$(\mathcal{F} \circ Y_k)(\tau_k) \leqslant (\mathcal{F} \circ Y_k)(0) + c_1 \tau_k (\mathcal{F} \circ Y_k)'(0), \tag{3.5a}$$

$$(\mathcal{F} \circ Y_k)'(\tau_k) \geqslant c_2 (\mathcal{F} \circ Y_k)'(0), \tag{3.5b}$$

where $0 < c_1 < c_2 < 1$ are two parameters. The proof of the existence of τ_k is exactly the same as in the classical line search, cf. [37, Lemma 3.1]. To find such τ_k in practice, we refer the reader to algorithms 3.5 and 3.6 in [37].

Lemma 3.2.1 (Existence of Armijo-Wolfe Step [45, Lemma 6.]). *If $0 < c_1 < c_2 < 1$ and $(\mathcal{F} \circ Y_k)'(0) < 0$, there exist nonempty intervals of step lengths satisfying the Armijo-Wolfe conditions.*

Every iteration of Algorithm 3 is well defined. Note that if rewrite (3.5a) as

$$\mathcal{F}(X_{k+1}) \leqslant \mathcal{F}(X_k) + c_1 \tau_k (\mathcal{F} \circ Y_k)'(0),$$

where $(\mathcal{F} \circ Y_k)'(0) \leqslant 0$ for all k by Lemma 3.1.4, the generated sequence $\{\mathcal{F}(X_k)\}$ is monotonically decreasing.

Algorithm 3: Monotone Curvilinear Search

```
Initialization: Set 0 < c_1 < c_2 < 1, ε > 0, k ← 0, an initial point X_0 ∈ O_r;
while ‖∇F(X_k)‖ > ε do
    Generate A_k ← G_k X_k^T − X_k G_k^T, W_k ← A_k ;
    Find a step size τ_k > 0 that satisfies the Armijo-Wolfe conditions (3.5a) and (3.5b);
    Set X_{k+1} ← Y_k (τ_k);
    k ← k + 1 and continue;
end
```

To prove the global convergence of Algorithm 3, we need an assumption about the skew-symmetric matrix W that yields the curve $Y(\tau)$:

Condition 1. *The matrix W in (3.3) is continuous in X and satisfies*

$$(\mathcal{F} \circ Y)'(0) \leqslant -\sigma \|A\|^2,$$

where $\sigma > 0$ is a constant.

It is clear that condition 1 is satisfied since $W(X) = G(X)X^T - XG(X)^T$ is continuous with respect to X and $\sigma = \frac{1}{2}$ by Lemma 3.1.4.

3.2.2 Global Convergence

Here, we give a detailed proof of convergence. For the original content, we refer the reader to [44]. For given arbitrary starting point $X_0 \in \mathcal{M}_n^p$, let us define the level set:

$$\Theta = \{X \mid \mathcal{F}(X) \leqslant \mathcal{F}(X_0), X \in \mathcal{M}_n^p\},$$

which is compact. To see this, the per-image of the closed set $(-\infty, \mathcal{F}(X_0)]$ under a continuous map $\mathcal{F}(X)$ is also closed, and $\mathcal{M}_n^p$ is compact by Lemma 3.1.1. A closed subset of a compact set is compact.

Starting from $X_0 \in \mathcal{M}_n^p$, the sequence $\{X_k\}$ generated by Algorithm 3 lies in Θ since $\mathcal{F}(X_k)$ decreases monotonically and $X_k \in \mathcal{M}_n^p$ for all k. We start with an auxiliary lemma as follows.

Lemma 3.2.2. *Suppose that Condition 1 is satisfied and $\{X_k\}_{k\in K}$ is an infinite subsequence generated by Algorithm 3. If $\lim_{k\in K} \tau_k = 0$, then the sequence $\{Y_k(\tau_k)\}_{k\in K}$ satisfies*

$$\lim_{k\in K} \|Y_k(\tau_k) - Y_k(0)\| = 0 \text{ and } \lim_{k\in K} \left\|Y_k'(\tau_k) - Y_k'(0)\right\| = 0.$$

Proof. **Part 1.** We first have $Y_k(\tau_k) - Y_k(0) = -\tau_k \left(I + \frac{\tau_k}{2} W_k\right)^{-1} W_k X_k$ for each k. To see this, omitting subscripts for simplicity, we have

$$\begin{aligned}
Y(\tau) - Y(0) &= -\frac{\tau}{2} W(X + Y(\tau)) \\
&= -\frac{\tau}{2} W\left(X + \left(I + \frac{\tau}{2}W\right)^{-1}\left(I - \frac{\tau}{2}W\right)X\right) \\
&= -\frac{\tau}{2} W\left(I + \left(I + \frac{\tau}{2}W\right)^{-1}\left(I - \frac{\tau}{2}W\right)\right)X \\
&= -\frac{\tau}{2} W\left(\left(I + \frac{\tau}{2}W\right)^{-1}\left(I + \frac{\tau}{2}W\right) + \left(I + \frac{\tau}{2}W\right)^{-1}\left(I - \frac{\tau}{2}W\right)\right)X \\
&= -\frac{\tau}{2} W\left(I + \frac{\tau}{2}W\right)^{-1}\left(\left(I + \frac{\tau}{2}W\right) + \left(I - \frac{\tau}{2}W\right)\right)X \\
&= -\tau\left(I + \frac{\tau}{2}W\right)^{-1} WX.
\end{aligned}$$

Hence, we observe that

$$\|Y_k(\tau_k) - Y_k(0)\| \leqslant \tau_k \left\|\left(I + \frac{\tau_k}{2}W_k\right)^{-1}\right\| \|W_k X_k\|.$$

Since $W(X)$ is a continuous function with respect to X, which stays in the compact set Θ, the sequences $\|W_k X_k\|$ [*] and $\left\|\left(I + \frac{\tau_k}{2}W_k\right)^{-1}\right\|$ are bounded. Hence, we obtain $\lim_{k\in K} \|Y_k(\tau_k) - Y_k(0)\| = 0$.

Part 2. On the other hand, we have

$$Y_k'(\tau_k) - Y_k'(0) = -\left(I + \frac{\tau_k}{2}W_k\right)^{-1}\left(\frac{1}{2}W_k(Y_k(\tau_k) - X_k) - \frac{\tau_k}{2}W_k^2 X_k\right),$$

[*] Do not regard W_k as a different function for each k; actually, W_k means $W(X_k)$.

which follows from (omitting subscripts)

$$\begin{aligned} Y'(\tau) - Y'(0) &= WX - \left(I + \frac{\tau}{2}W\right)^{-1} W\left(\frac{X + Y(\tau)}{2}\right) \\ &= \left(I + \frac{\tau}{2}W\right)^{-1}\left(\left(I + \frac{\tau}{2}W\right)WX - W\left(\frac{X + Y(\tau)}{2}\right)\right) \\ &= \left(I + \frac{\tau}{2}W\right)^{-1}\left(\frac{1}{2}WX - \frac{1}{2}WY(\tau) + \frac{\tau}{2}W^2X\right) \\ &= -\left(I + \frac{\tau}{2}W\right)^{-1}\left(\frac{1}{2}W\left(Y(\tau) - X\right) - \frac{\tau}{2}W^2X\right). \end{aligned}$$

Therefore, we obtain

$$\begin{aligned} \left\|Y_k'(\tau_k) - Y_k'(0)\right\| &= \left\|\left(I + \frac{\tau_k}{2}W_k\right)^{-1}\left(\frac{1}{2}W_k\left(Y_k(\tau_k) - X_k\right) - \frac{\tau_k}{2}W_k^2X_k\right)\right\| \\ &\leqslant \left\|\left(I + \frac{\tau_k}{2}W_k\right)^{-1}\right\|\left(\frac{1}{2}\left\|W_k\right\|\left\|Y_k(\tau_k) - X_k\right\| + \left\|\frac{\tau_k}{2}W_k^2X_k\right\|\right). \end{aligned}$$

Using Part 1, one can similarly show $\lim_{k\in K}\left\|Y_k'(\tau_k) - Y_k'(0)\right\| = 0$.

□

Theorem 3.2.3. *Suppose Condition 1 is satisfied for the sequence $\{X_k\}$ generated by Algorithm 3. Then,*

$$\lim_{k\to\infty}\left\|\nabla\mathcal{F}(X_k)\right\| = 0.$$

Proof. **Part 1.** It suffices to prove $\lim_{k\to\infty}\|A_k\| = 0$ by Lemma 3.1.3. For a proof by contradiction, we suppose that $\lim_{k\to\infty}\|\nabla A_k\| \neq 0$; then there exists a constant $\epsilon > 0$ and an infinite index set K such that

$$\|A_k\| > \epsilon, \text{ for all } k \in K. \tag{3.6}$$

Part 2. From Condition 1,

$$\sigma\|A_k\|^2 \leqslant -(\mathcal{F}\circ Y_k)'(0) \tag{3.7}$$

and (3.5a), we have

$$\sigma c_1\tau_k\|A_k\|^2 \leqslant -c_1\tau_k(\mathcal{F}\circ Y_k)'(0) \leqslant \mathcal{F}(X_k) - \mathcal{F}(X_{k+1})$$

and finally get

$$\sigma c_1\tau_k\|A_k\|^2 \leqslant \mathcal{F}(X_k) - \mathcal{F}(X_{k+1}).$$

Observe that

$$\begin{aligned} \sigma c_1\tau_0\|A_0\|^2 &\leqslant \mathcal{F}(X_0) - \mathcal{F}(X_1), \\ \sigma c_1\tau_1\|A_1\|^2 &\leqslant \mathcal{F}(X_1) - \mathcal{F}(X_2), \\ &\vdots \end{aligned}$$

Summing up the above inequalities gives

$$\sum_{k=0}^{\infty} \sigma\mu_1\tau_k \left\|A_k\right\|^2 \leqslant \mathcal{F}\left(X_0\right) - \lim_{k\to\infty} \mathcal{F}\left(X_k\right).$$

Here, $\lim_{k\to\infty} \mathcal{F}\left(X_k\right)$ exists. To see this, the continuity of $\mathcal{F}(X)$ and compactness of Θ imply that there is a $X^* \in \Theta$ such that $\mathcal{F}(X^*) \leqslant \mathcal{F}(X_k)$ for each k. Hence, $\{\mathcal{F}(X_k)\}$ is lower bounded. The descent property of $\mathcal{F}\left(X_k\right)$ leads to the existence of $\lim_{k\to\infty} \mathcal{F}\left(X_k\right)$. Finally, we have $\tau_k \left\|A_k\right\|^2 \to 0$, which implies that $\tau_k \to 0$ for $k \in K$ in light of (3.6). Now, we can use Lemma 3.2.2.

Part 3. From (3.5a) and (3.7), we have

$$(\mathcal{F}\circ Y_k)'(\tau_k) - (\mathcal{F}\circ Y_k)'(0) \geqslant (c_2-1)(\mathcal{F}\circ Y_k)'(0) \geqslant (1-\mu_2)\,\sigma\left\|A_k\right\|^2,$$

and finally, we find that, for all $k \in K$,

$$(1-\mu_2)\,\sigma\left\|A_k\right\|^2 \leqslant (\mathcal{F}\circ Y_k)'(\tau_k) - (\mathcal{F}\circ Y_k)'(0).$$

On the other hand, it follows from the chain rule that

$$\begin{aligned}
&(\mathcal{F}\circ Y_k)'(\tau_k) - (\mathcal{F}\circ Y_k)'(0) \\
&= \operatorname{tr}\left(\mathcal{D}\mathcal{F}(Y_k\left(\tau_k\right))^T Y_k'(\tau_k)\right) - \left(\mathcal{D}\mathcal{F}(Y_k\left(0\right))^T Y_k'(0)\right) \\
&= \operatorname{tr}\left(\mathcal{D}\mathcal{F}\left(Y_k\left(\tau_k\right)\right)^T \left[Y_k'\left(\tau_k\right) - Y_k'(0)\right]\right) + \operatorname{tr}\left(\left[\mathcal{D}\mathcal{F}\left(Y_k\left(\tau_k\right)\right) - \mathcal{D}\mathcal{F}\left(Y_k(0)\right)\right]^T Y_k'(0)\right)
\end{aligned}$$

Moreover, the Cauchy-Schwartz inequality implies that

$$\begin{aligned}
(1-\mu_2)\,\sigma\left\|A_k\right\|^2 \leqslant{}& \left\|\mathcal{D}\mathcal{F}\left(Y_k\left(\tau_k\right)\right)\right\| \left\|Y_k'\left(\tau_k\right) - Y_k'(0)\right\| \\
&+ \left\|\mathcal{D}\mathcal{F}\left(Y_k\left(\tau_k\right)\right) - \mathcal{D}\mathcal{F}\left(Y_k(0)\right)\right\| \left\|Y_k'(0)\right\|.
\end{aligned} \tag{3.8}$$

Part 4. We easily have the following facts:

1. $\left\|\mathcal{D}\mathcal{F}\left(Y_k\left(\tau_k\right)\right)\right\| = \left\|\mathcal{D}\mathcal{F}\left(X_{k+1}\right)\right\|$ is bounded by the continuity of $\mathcal{D}\mathcal{F}(X)$.
2. $\left\|Y_k'(0)\right\| = \left\|W_k X_k\right\|$ and $\left\|W_k X_k\right\|$ is bounded.
3. Recalling $\lim_{k\in K} \left\|Y_k\left(\tau_k\right) - Y_k(0)\right\| = 0$ in Lemma 3.2.2, we have

$$\lim_{k\in K} \left\|\mathcal{D}\mathcal{F}\left(Y_k\left(\tau_k\right)\right) - \mathcal{D}\mathcal{F}\left(Y_k(0)\right)\right\| = 0.$$

4. $\lim_{k\in K} \left\|Y_k'\left(\tau_k\right) - Y_k'(0)\right\| = 0$ by Lemma 3.2.2.

These facts together imply that the right-hand side of (3.8) converges to zero as $k \in K$ goes to ∞. This implies that $\lim_{k\in K} \left\|A_k\right\| = 0$, which contradicts (3.6).

□

Chapter 4

CP factorization via Curvilinear Search

A new method of solving the feasibility problem (FeasP) is proposed in this chapter. Here, $\max x_i$ ($\min x_i$) denotes the largest (smallest) entry of $\mathbf{x} \in \mathbb{R}^n$. As well, $\max(\cdot)_{ij}$ ($\min(\cdot)_{ij}$) represents the largest (smallest) entry of the given matrix. We say these are *hard* maximum or minimum functions because they are not differentiable. Notice that $-\min(x_i) = \max(-x_i)$.

$$\begin{array}{ll} \text{find} & X \\ \text{s.t.} & BX \geqslant 0 \\ & X \in \mathcal{O}_r. \end{array} \tag{FeasP}$$

First, we will establish the connection between (FeasP) and the following optimization problem:

$$\begin{array}{ll} \max & \min(BX)_{ij} \\ \text{s.t.} & X \in \mathcal{O}_r. \end{array}$$

For consistency of notation, we will turn the maximization problem into a minimization:

$$\begin{array}{ll} \min & -\min(BX)_{ij} \\ \text{s.t.} & X \in \mathcal{O}_r, \end{array}$$

which is equivalent to

$$\begin{array}{ll} \min & \max(-BX)_{ij} \\ \text{s.t.} & X \in \mathcal{O}_r. \end{array} \tag{OptP}$$

Apparently, (FeasP) is feasible if and only if $\exists X \in \mathcal{O}_r$ such that $\min(BX)_{ij} \geqslant 0$ or $\max(-BX)_{ij} \leqslant 0$. In this case, we find a CP factorization $A = (BX)(BX)^T$ with $BX \geqslant 0$.

Furthermore, as a special case of the Stiefel manifold, the feasible set of (OptP), $\mathcal{O}_r$, is known to be compact and $\max(-BX)_{ij} : \mathbb{R}^{r\times r} \to \mathbb{R}$ is continuous. Following the well-known extreme value theorem, (OptP) can obtain the global minimum, say t, with a global minimizer X^*. Then, $A \in \mathcal{CP}_n$ if and only if $t \leqslant 0$. Summarizing these results with Theorem 2.1.5 gives following theorem.

Theorem 4.0.1. *The following statements are equivalent:*

1. $A \in \mathcal{CP}_n$.

2. *(FeasP) is feasible.*

3. *In (OptP), there exists a feasible solution X such that $\max\,(-BX)_{ij} \leqslant 0$; alternatively, $\min\,(BX)_{ij} \geqslant 0$.*

4. *In (OptP), the global minimum $t \leqslant 0$.*

Our goal has changed from (FeasP) to solving the optimization problem (OptP) with orthogonality constraints. Due to the NP-hardness of checking membership, we cannot expect to compute the global minimum easily. However, we still struggle to minimize the value $\max\,(-BX)_{ij}$. Our strategy for (OptP) is roughly as follows:

1. Introduce a smooth approximation to replace the hard maximum function.

2. Utilize the curvilinear search method on the approximation problem.

Note that the reason for using the approximate function is that the differentiability of the objective function is required by the curvilinear search. Below, we review some facts about the approximate function, namely the LogSumExp function.

4.1 LogSumExp: Smooth Approximation to Maximum Function

The logarithm of the sum of exponentials function, i.e., LogSumExp, is defined as $LSE_\mu(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$,

$$LSE_\mu(\mathbf{x}) = \mu \log\left(\sum_{i=1}^n \exp\left(x_i/\mu\right)\right).$$

$LSE_\mu(\mathbf{x})$ is able to approximate the maximum (resp., minimum) function if the parameter $\mu > 0$ (resp., $\mu < 0$). In addition, it is a convex (resp., concave) function on $\mathbb{R}^n$ if $\mu > 0$ (resp., $\mu < 0$), cf. [25, Proposition 2]. For a summary including many other results on LogSumExp and its gradient (i.e., softmax function), we refer readers to [25]. Similar to the vector form defined above, LogSumExp with a matrix argument can be simply derived from entrywise operation:

$$LSE_\mu(X) = \mu \log\left(\sum_{i,j} \exp\left(X_{ij}/\mu\right)\right).$$

For simplicity, we will employ the vector argument to build the following crucial lemmas. Here, we concentrate on the case $\mu > 0$ and $\mu \to 0$.

4.1.1 Vector Argument

First, we shall show an auxiliary lemma, which is important for avoiding numerical overflow and underflow when computing $LSE_\mu\left(\mathbf{x}\right)$.

Lemma 4.1.1. *Suppose that $\mu > 0$. Let $\boldsymbol{1}$ be a vector whose entries are all ones. Then, we have*

$$LSE_\mu\left(\boldsymbol{x}\right) = LSE_\mu\left(\boldsymbol{x} - c\boldsymbol{1}\right) + c,$$

for any $c \in \mathbb{R}$. Alternately,

$$LSE_\mu\left(\boldsymbol{x}\right) = \mu \log\left(\sum_{i=1}^n \exp\left((x_i - c)/\mu\right)\right) + c.$$

Proof. The statement of the theorem follows from

$$\begin{aligned}\mu \log\left(\sum_{i=1}^{n} \exp\left((x_i - c)/\mu\right)\right) + c =& \mu \log\left(\exp(-c/\mu)\sum_{i=1}^{n} \exp\left(x_i/\mu\right)\right) + c \\ =& \mu \log\left(\exp(-c/\mu)\right) + \mu \log\left(\sum_{i=1}^{n} \exp\left(x_i/\mu\right)\right) + c \\ =& \mu \log\left(\sum_{i=1}^{n} \exp\left(x_i/\mu\right)\right).\end{aligned}$$

□

Consider the two-variable case with $\mu = 1$, $LSE_1(x, y) = \log(e^x + e^y)$. The overflow problem occurs in practice if x or y is large. For example, if $x = 1000$ and $y = 2$, the value of e^{1000} is too big to be represented as a floating point number, and so it is treated as infinity in a computer. Finally, we get the false result: $\log(\inf + e^2) = \log(\inf) = \inf$. In opposite, the underflow problem occurs if both x and y are small. For example, if $x = -1000$ and $y = -2000$, then e^{-1000} and e^{-2000} are both too small, and so they are treated as zeros, which leads to: $\log(0 + 0) = -\inf$.

$$\log(e^{1000} + e^2) = \log(e^{1000-1000} + e^{2-1000}) + 1000 \approx \log(1 + 0) + 1000 = 1000,$$
$$\log(e^{-1000} + e^{-2000}) = \log(e^{-1000+1000} + e^{-2000+1000}) - 1000 \approx \log(1 + 0) - 1000 = -1000.$$

From Lemma 4.1.1, we can shift every argument by a constant $c := \max(x, y)$, so we can avoid overflow and underflow. The multivariate situation is similar, where let $c := \max x_i$.

Lemma 4.1.2. *The gradient of $LSE_\mu(\boldsymbol{x})$ is named the softmax function, which is given by $\sigma : \mathbb{R}^n \to \operatorname{int} \Delta^{n-1}$,*

$$\sigma(\boldsymbol{x}) := \frac{1}{\sum_{j=1}^{n} \exp\left(x_j/\mu\right)} \begin{bmatrix} \exp\left(x_1/\mu\right) \\ \vdots \\ \exp\left(x_n/\mu\right) \end{bmatrix},$$

where $\operatorname{int} \Delta^{n-1} := \{x \in \mathbb{R}^n \mid \sum_{i=1} x_i = 1, x_i > 0\}$ is the interior of the unit simplex.

Proof.

$$\frac{\partial LSE_\mu}{\partial x_i}(\mathbf{x}) = \frac{\mu}{\sum_{j=1}^{n} \exp\left(x_j/\mu\right)} \frac{\exp(x_i/\mu)}{\mu} = \frac{\exp(x_i/\mu)}{\sum_{j=1}^{n} \exp\left(x_j/\mu\right)} = \sigma_i(\mathbf{x}).$$

□

Rewrite the right-hand part by using the equality $\sum_{i=1}^{n} \exp\left(x_i/\mu\right) = \exp\{LSE_\mu\left(\mathbf{x}\right)/\mu\}$. Then, the components of the softmax function can be rewritten as

$$\sigma_i(\mathbf{x}) = \frac{\partial LSE_\mu}{\partial x_i}(\mathbf{x}) = \exp\{(x_i - LSE_\mu(\mathbf{x}))/\mu\}.$$

We will use this formula for $\sigma(\mathbf{x})$ in the numerical experiments for avoiding overflow and underflow.

The next theorem indicates that LogSumExp is a pretty good approximation to the maximum function, as it uniformly converges to the maximum function as $\mu \to 0$. Analogous results hold for $\mu < 0$ with regard to the minimum function. Here are the 3-D graphics of $\max(x, y)$ and $\log(e^x + e^y)$ on $\mathbb{R}^2$.

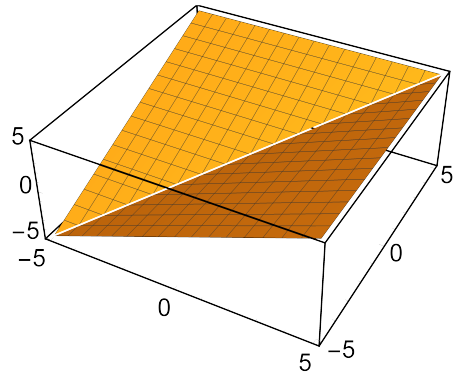

Fig. 4.1. Graph of $\max(x, y)$.

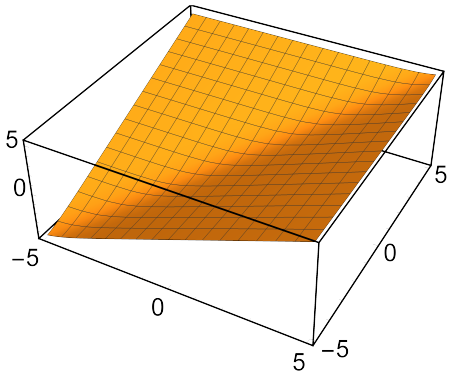

Fig. 4.2. Graph of $\log(e^x + e^y)$.

Theorem 4.1.3 (Approximation theorem of LogSumExp)**.** *Suppose that* $\mu > 0$*, and* $\max x_i$ *denotes the minimum entry of* $\boldsymbol{x}$*. For all* $\boldsymbol{x} \in \mathbb{R}^n$*, we have*

1. $\max x_i < LSE_\mu(\boldsymbol{x}) \leqslant \max x_i + \mu \log(n)$*, hence* $|\max x_i - LSE_\mu(\boldsymbol{x})| \leqslant \mu \log(n)$.

2. $\lim_{\mu \to 0^+} LSE_\mu(\boldsymbol{x}) = \max x_i$.

3. *if* $0 < \mu_2 < \mu_1$*, then* $LSE_{\mu_2}(\boldsymbol{x}) < LSE_{\mu_1}(\boldsymbol{x})$.

Proof. 1. In particular, if let $c = \max x_i$, say x_j, then by Lemma 4.1.1 we have

$$LSE_\mu(\mathbf{x}) = \mu \log\left(1 + \textstyle\sum_{i \neq j}^{n} \exp\left((x_i - x_j)/\mu\right)\right) + x_j.$$

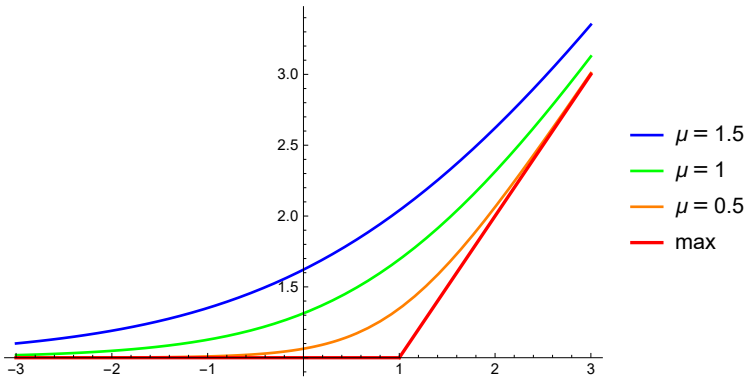

Fig. 4.3. Slices through the line $y = 1$ of $\max(x, y)$ and some $\mu \log(e^{x/\mu} + e^{y/\mu})$ on $\mathbb{R}^2$.

Table 4.1. Example of approximation effect with different parameters μ.

$n = 4$	$\mu = 1$	$\mu = 1/2$	$\mu = 1/4$	$\mu = 1/8$
$\mathbf{x}_1 = (2, 5, -1, 3)$	5.1719	5.0103	5.0001	5.0000
$\mathbf{x}_2 = (5, 5, 5, 5)$	6.3863	5.6931	5.3466	5.1733
$\epsilon_\mu = \mu \log(n)$	1.3863	0.6931	0.3466	0.1733

For every $i \neq j$, $\mu(x_i - x_j) \leqslant 0$ implies $1 < 1 + \sum_{i \neq j}^{n} \exp\left((x_i - x_j)/\mu\right) \leqslant n$. Then, taking the logarithm gives

$$0 < \mu \log\left(1 + \sum_{i \neq j}^{n} \exp\left((x_i - x_j)/\mu\right)\right) \leqslant \mu \log(n),$$

which means that $0 < LSE_\mu(\mathbf{x}) - x_j \leqslant \mu \log(n)$.

2. It directly follows from above.

3. We claim that for any fixed $\mathbf{x} \in \mathbb{R}^n$, if we regard $LSE_\mu(\mathbf{x})$ as the function corresponding to single variable $\mu \in (0, \infty)$, denoted by $LSE_{\mathbf{x}}(\mu)$, then for all $\mu \in (-\infty, 0)$,

$$\frac{dLSE_{\mathbf{x}}}{d\mu}(\mu) > 0.$$

For simplicity, we replace $LSE_{\mathbf{x}}(\mu)$ or $LSE_\mu(\mathbf{x})$ with LSE next. We have

$$\begin{aligned}
\frac{dLSE_{\mathbf{x}}}{d\mu}(\mu) =& \log\left(\sum_{i=1}^{n} \exp\left(x_i/\mu\right)\right) - \frac{\sum_{i=1}^{n} x_i \exp\left(x_i/\mu\right)}{\mu \sum_{i=1}^{n} \exp\left(x_i/\mu\right)} = LSE/\mu - \frac{\sum_{i=1}^{n} x_i \exp\left(x_i/\mu\right)}{\mu \exp\{LSE/\mu\}} \\
=& (LSE - \textstyle\sum_{i=1}^{n} x_i \exp\{(x_i - LSE)/\mu\})/\mu = (LSE - \textstyle\sum_{i=1}^{n} x_i \sigma_i)/\mu \\
=& \left(LSE - \mathbf{x}^T \sigma\right)/\mu > 0.
\end{aligned}$$

For the last inequality, we observe from Lemma 4.1.2 that $\sum_{i=1}^{n} \sigma_i = 1$ and every entry $\sigma_i > 0$; hence, the term $\mathbf{x}^T \sigma$ is a convex combination of all entries of $\mathbf{x}$, which implies that $\mathbf{x}^T \sigma \leqslant \max x_i < LSE$.

□

Observe that the parameter μ plays the role of a controller. The third statement of the above theorem is interpreted in Fig. 4.3. The upper bound of the error $\epsilon_\mu := \mu \log(n)$, which is completely determined by μ, vanishes as $\mu \to 0$. An example is shown in Table 4.1. Rows 2 and 3 consist of the values of LSE corresponding to $\mathbf{x}$ and μ. The worst approximation appears if all entries of $\mathbf{x}$ are the same. However, a sufficiently small μ can eliminate this concern.

4.1.2 Matrix Argument

Recall that we want to approximate $\max(-BX)_{ij} : \mathbb{R}^{r \times r} \to \mathbb{R}$. The following theorem and lemma for the matrix argument directly comes from the resulting vector form and chain rule.

Lemma 4.1.4. *Given $B \in \mathbb{R}^{n\times r}$, the gradient of $\mathcal{K}(X) := LSE_\mu(-BX) : \mathbb{R}^{r\times r} \to \mathbb{R}$ at X is given by*

$$\mathcal{D}\mathcal{K}(X) = -B^T \mathcal{D}LSE_\mu(-BX),$$

where the gradient of the matrix-value function $LSE_\mu(Y)$ is denoted by $\mathcal{D}LSE_\mu(Y)$, whose ij-th entry is $\exp\{(Y_{ij} - LSE_\mu(Y))/\mu\}$.

Theorem 4.1.5. *Suppose that $\mu > 0, B \in \mathbb{R}^{n\times r}$, and $\max(-BX)_{ij}$ denotes the minimum entry of $-BX$. For all $X \in \mathbb{R}^{r\times r}$, we have*

1. $\max(-BX)_{ij} < LSE_\mu(-BX) \leqslant \max(-BX)_{ij} + \mu\log(nr)$, *hence,* $|\max(-BX)_{ij} - LSE_\mu(-BX)| \leqslant \mu\log(nr)$.
2. $\lim\limits_{\mu\to 0^+} LSE_\mu(-BX) = \max(-BX)_{ij}$.
3. *if* $0 < \mu_2 < \mu_1$, *then* $LSE_{\mu_2}(-BX) < LSE_{\mu_1}(-BX)$.

4.2 CP factorization via Monotone Curvilinear Search

Now, we can approximate problem (OptP) as problem (LseP):

$$\begin{array}{ll} \min & LSE_\mu(-BX) \\ \text{s.t.} & X \in \mathcal{O}_r. \end{array} \tag{LseP}$$

Let us investigate some facts about these problems. Similar to $\max(-BX)_{ij}$, considering the continuity of $LSE_\mu(-BX) : \mathbb{R}^{r\times r} \to \mathbb{R}$, (LseP) can obtain the global minimum, say t_μ, with a global minimizer X^*_μ. The following results show that as $\mu \to 0$, the two global minimums almost coincide.

Proposition 4.2.1. *We have $0 < t_\mu - t \leqslant \epsilon_\mu$, where $\epsilon_\mu := \mu\log(nr) > 0$.*

Proof. By Lemma 4.1.5, we have

$$LSE_\mu(-BX^*) \geqslant LSE_\mu(-BX^*_\mu) = t_\mu > \max(-BX^*_\mu)_{ij} \geqslant \max(-BX^*)_{ij} = t.$$

Thus, $0 < t_\mu - t$. And

$$t_\mu - t = LSE_\mu(-BX^*_\mu) - \max(-BX^*)_{ij} - \leqslant LSE_\mu(-BX^*) - \max(-BX^*)_{ij} \leqslant \epsilon_\mu.$$

□

If problem (LseP) has a feasible solution X such that $LSE_\mu(-BX) \leqslant 0$, then for the same X, $\max(-BX)_{ij} < 0$ for (OptP). Hence, we have found a CP factorization $(BX)(BX)^T = A$ with $BX > 0$.

We can directly use Algorithm 1 for solving (LseP). Here, $\mathcal{F}(X) := LSE_\mu(-BX)$, $p = n := r$. Finally, we get to optimize a convex function over the compact but nonconvex feasible set. There are some adjustments: if A is full rank, we use Cholesky decomposition to obtain an initial B for reason of precision. Algorithms 3.5 and 3.6 in [37] are used to compute the step size. Since our

main goal is to find a feasible solution X such that $\max(-BX)_{ij} \leqslant 0$, rather than a minimizer, we set $\max(-BX_{k+1})_{ij} \leqslant 0$ or $\min(BX_{k+1})_{ij} \geqslant 0$ in the place of $\|\nabla \mathcal{F}_{k+1}\| \leqslant \epsilon$ as the stopping condition. In addition, the algorithm terminates unsuccessfully if the maximum number of iterations (here 5000) is reached.

Algorithm 4: CP factorization via Monotone Curvilinear Search

Given $A \in \mathcal{CP}_n, r \geqslant \mathrm{cp}(A)$, an initial decomposition $B \in \mathbb{R}^{n\times r}$;
Initialization: Set $0 < c_1 < c_2 < 1, k \leftarrow 0, \mu > 0$, an initial point $X_0 \in \mathcal{O}_r$;
while *true* **do**
    Generate $G_k \leftarrow -B^T \mathcal{D}LSE_\mu(-BX_k)$, $A_k \leftarrow G_k X_k^T - X_k G_k^T, W_k \leftarrow A_k$;
    Find a step size $\tau_k > 0$ that satisfies the Armijo-Wolfe conditions (3.5a),(3.5b);
    Set $X_{k+1} \leftarrow Y(\tau_k)$;
    if $\min(BX_{k+1})_{ij} \geqslant 0$ *or* $k = 5000$ **then**
        STOP;
    end
    $k \leftarrow k + 1$ and continue;
end

The inversion of $I + \frac{\tau}{2}W$ on the curve $Y(\tau)$ dominates the computation of this method. It is numerically cheaper than calculating subproblems at each iteration, as other methods do. Thus, we expect that its numerical performance will be excellent.

4.3 CP factorization via Nonmonotone Curvilinear Search

Instead of the Armijo-Wolfe rules, the well-known Barzilai-Borwein (BB) step size can usually be used to speed up the gradient method without the extra cost of solving unconstrained optimization problems on $\mathbb{R}^n$, cf. [4]. Likewise for Stiefel manifold optimization, we can set τ_{k+1} to either

$$\tau_{k+1,1} = \frac{\langle S_k, S_k\rangle}{|\langle S_k, Y_k\rangle|} \text{ or } \tau_{k+1,2} = \frac{|\langle S_k, Y_k\rangle|}{\langle Y_k, Y_k\rangle},$$

where $S_k = X_{k+1} - X_k$ and $Y_k = \nabla \mathcal{F}(X_{k+1}) - \nabla \mathcal{F}(X_k)$. However, a poorly chosen BB step size may rule out convergence.

To ensure global convergence, Wen and Yin suggested a non-monotone line search method based on a strategy in [47]. In [47], this non-monotone technique guarantees global convergence of the BB step size for unconstrained optimization on $\mathbb{R}^n$. Although Wen and Yin suggested this adaption, they did not discuss it in detail. Fortunately, the convergence of the adaption to Stiefel manifold optimization was proved by [32].

Algorithm 5 with a BB step size is summarized below. The only difference from Algorithm 4 is how the convergence-guaranteed step size is chosen. The next section provides a heuristic improvement that has a more significant effect.

Algorithm 5: CP factorization with BB step

Given $A \in \mathcal{CP}_n, r \geqslant \mathrm{cp}(A)$, an initial decomposition $B \in \mathbb{R}^{n\times r}$;
Initialization: Set $\tau > 0, \rho, \delta, \eta \in (0,1), \mu > 0, c_0 \leftarrow \mathcal{F}(X_0), q_0 \leftarrow 1, k \leftarrow 0$, an initial point $X_0 \in \mathcal{O}_r$;
while *true* **do**
  Generate $G_k \leftarrow -B^T \mathcal{D}LSE_\mu\left(-BX_k\right), A_k \leftarrow G_k X_k^T - X_k G_k^T, W_k \leftarrow A_k$;
  while $(\mathcal{F} \circ Y_k)(\tau) \geqslant c_k + \rho\tau(\mathcal{F} \circ Y_k)'(0)$ **do**
    $\tau \leftarrow \delta\tau$
  end
  Set $X_{k+1} \leftarrow Y_k(\tau)$;
  if $\min\left(BX_{k+1}\right)_{ij} \geqslant 0$ *or* $k = 5000$ **then**
    STOP;
  end
  $q_{k+1} \leftarrow \eta q_k + 1$, $c_{k+1} \leftarrow \left(\eta q_k c_k + \mathcal{F}\left(X_{k+1}\right)\right)/q_{k+1}$;
  $\tau \leftarrow \tau_{k+1,1}$ or $\tau_{k+1,2}$;
  $k \leftarrow k+1$ and continue;
end

4.4 Heuristic Extension: Decreasing Parameter μ

In practice, the parameter μ is not as small as possible. Although a smaller μ can give a more rigorous approximation, the speed of convergence is usually slower. In contrast, too large a μ is meaningless. It is unlikely that the best parameter μ is known for every matrix.

Fig. 4.4 shows the detailed iterative processes of two different fixed values of μ, 0.01 and 1, for the same instance. Here, $A = HH^T$, where $H \in \mathbb{R}^{20\times 20}$ with entries randomly generated in $\{5, 6, \ldots, 10\}$ and $r = 20$. The solid and dashed lines respectively represent the values of $-LSE(-BX_k)$ and $\min(BX_k)$. In particular, the blue solid line and the blue dashed line overlap, because μ is so small that it has a high degree of fitness.

It can be seen that for a relatively larger μ, the value grows faster in the early stages of the iteration; it, however, converges to a smaller number that may be less than zero, meaning that we cannot obtain a CP factorization. For a smaller μ, it behaves in the opposite way. This observation gives us the idea that we should use a large μ at the beginning and then reduce μ as the iteration progresses. So we should use a reduction scheme for μ. A natural choice is the reciprocal function, $y = 1/x$, which yields Algorithm 6 below.

Algorithm 6: CP factorization with BB steps and decreasing μ

The steps are the same as in Algorithm 5 except:
Set the initial values $\mu_0 > 0$, $\mu_{step} > 0$, and $\mu_k \leftarrow \mu_0/(1 + k\mu_{step})$ after every iteration.

In Fig. 4.4, the red lines represent a reduction strategy, where $\mu_0 = 10$ and $\mu_{step} = 1$. As expected, Algorithm 6 increases rapidly in the early iterations and converges to a larger number in the later iterations; thereby, it takes fewer iterations for it to reach zero. In the next chapter, we will compare Algorithms 4, 5, and 6.

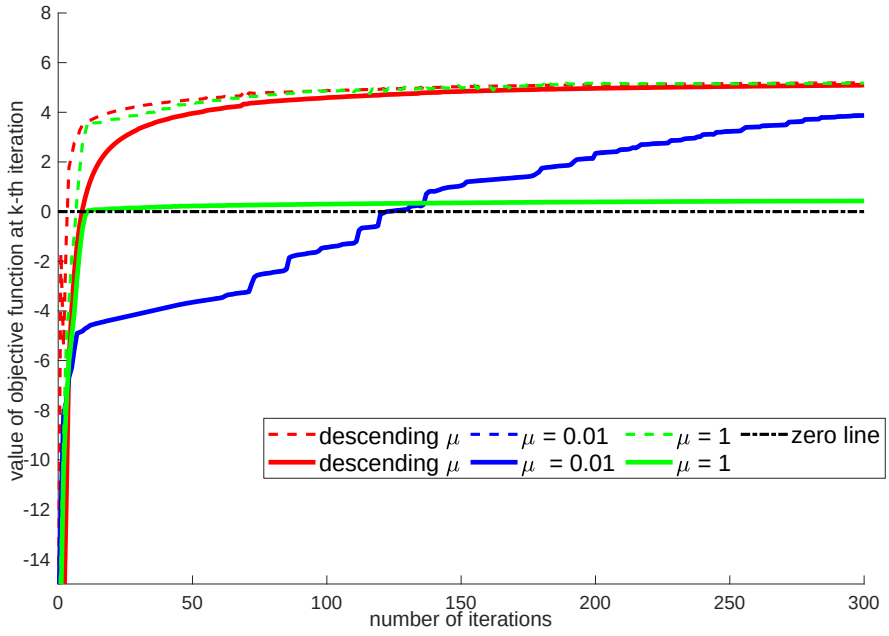

Fig. 4.4. Detailed iterative processes of two different values μ for the same instance under Algorithm 5.

4.5 A Family of Smooth Approximations to the Maximum Function

Notice that LogSumExp is nothing but a smooth approximation to the maximum function. Are there are other smooth approximation functions besides LogSumExp? What if we choose them for the CP factorization problem? Unfortunately, there is not much relevant research on smooth approximations to the maximum.

Hence, as a useful supplement, we propose here a method for constructing a smooth approximation to the maximum. Briefly, it shows that every smooth approximation to the absolute value function leads to the maximum function. Nevertheless, due to the recursive definitions, calculating the gradient of them is more expensive than that of LogSumExp. Hence, we did not run them in the experiments.

4.5.1 2-dimensional Case

First, we try to construct a smooth approximation to the maximum function on $\mathbb{R}^2$, i.e., $\max(x_1, x_2)$. Observe that

$$\max(x_1, x_2) = \frac{1}{2}(x_1 + x_2 + |x_1 - x_2|). \tag{4.1}$$

Following this hint, it is enough to produce a smooth approximation to the absolute value function. In fact, there are many parametric smooth approximations to $|x|$. Most of the following results come from [3], [2]. We treat μ as a parameter in the function $h(\cdot, \mu)$, and h' means the derivative of h with respect to the first variable for a fixed μ.

1. $h_1(x,\mu) = \sqrt{x^2+\mu^2}$.

$$||x|-h_1(x,\mu)| \leqslant \mu, \quad h_1'(x,\mu) = \frac{x}{\sqrt{x^2+\mu^2}}.$$

2. $h_2(x,\mu) = x\tanh\left(\frac{x}{\mu}\right)$, where $\tanh(z)$ is hyperbolic tangent function.

$$||x|-h_2(x,\mu)| \leqslant \mu, \quad h_2'(x,\mu) = \frac{x}{\mu}\operatorname{sech}^2\left(\frac{x}{\mu}\right) + \tanh\left(\frac{x}{\mu}\right).$$

3. $h_3(x,\mu) = 2\mu\log\left(1+e^{\frac{x}{\mu}}\right) - x = 2\mu\log\left(1+e^{-\frac{x}{\mu}}\right) + x$.

$$||x|-h_3(x,\mu)| \leqslant 2\log(2)\mu, \quad h_3'(x,\mu) = 1 - \frac{2}{1+e^{\frac{x}{\mu}}} = \frac{2}{1+e^{-\frac{x}{\mu}}} - 1.$$

4. $h_4(x,\mu) = x\,\mathrm{erf}(\frac{x}{\mu})$, where $\mathrm{erf}(z) := \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt$ is the Gauss error function.

$$||x|-h_4(x,\mu)| \leqslant \frac{2}{e\sqrt{\pi}}\mu, \quad h_5'(x,\mu) = \mathrm{erf}\left(\frac{x}{\mu}\right) + \left(\frac{x}{\mu}\right)\frac{2}{\sqrt{\pi}}e^{-\left(\frac{x}{\mu}\right)^2}.$$

5. Others:

   (a) $(2/\pi)xgd(x/\mu)$, where $gd(z) := \int_0^z \frac{1}{\cosh(t)}dt$ is the Gudermannian function.

   (b) $(2/\pi)x\tan^{-1}(x/\mu)$.

   (c) $x^2\left(x^2+\mu^2\right)^{-1/2}$.

All of them satisfy the following condition.

Condition 2. *Suppose that $h(x,\mu)$ is a smooth approximation to $|x|$ on $\mathbb{R}$, satisfying:*

1. *$h(x,\mu)$ is parameterized by $\mu > 0$, and it converges uniformly to $|x|$ on $\mathbb{R}$ as $\mu \to 0$; i.e., there exists a constant κ such that*

$$|h(x,\mu) - |x|| \leqslant \kappa\mu, \quad \forall x \in \mathbb{R}.$$

2. *$h(x,\mu)$ is smooth on $\mathbb{R}$ for any $\mu > 0$.*

Accordingly, we use $h(x,\mu)$ to create a smooth approximation to $\max(x_1,x_2)$, namely, $g : \mathbb{R}^2 \to \mathbb{R}$, defined as

$$g(x_1,x_2) := \frac{1}{2}(x_1 + x_2 + h(x_1 - x_2,\mu)). \tag{4.2}$$

We will refer to $h(x,\mu)$ as $h(x)$ when μ is clear from the context. Immediately, we get similar properties like $h(x,\mu)$.

Lemma 4.5.1. *1. g is parameterized by $\mu > 0$, and it converges uniformly to $\max(x_1, x_2)$ on $\mathbb{R}^2$ as $\mu \to 0$, since*

$$|g(x_1, x_2) - \max(x_1, x_2)| \leqslant \frac{1}{2}\kappa\mu, \quad \forall (x_1, x_2) \in \mathbb{R}^2.$$

2. g is smooth on $\mathbb{R}^2$, and

$$\frac{\partial g}{\partial x_1}(x_1, x_2) = \frac{1}{2}(1 + h'(x_1 - x_2)), \frac{\partial g}{\partial x_2}(x_1, x_2) = \frac{1}{2}(1 - h'(x_1 - x_2)). \tag{4.3}$$

Proof. Using (4.1) and the properties of $h(x, \mu)$, we get $\frac{1}{2}|h(x_1 - x_2, \mu) - |x_1 - x_2|| \leqslant \frac{1}{2}\kappa\mu$. □

4.5.2 n-dimensional Case

Thus, we have constructed a smooth approximation to the maximum function on $\mathbb{R}^2$. Can this idea be extended to a general dimension n? Yes is the answer. Inspired by the equivalent formula of the maximum function on $\mathbb{R}^n$, i.e.,

$$\begin{aligned}
&\max(x_1, \ldots, x_{n-1}, x_n) \\
=&\max(\max(x_1, \ldots, x_{n-1}), x_n) \\
&\vdots \\
=&\max(\max(\ldots \max(\max(x_1, x_2), x_3) \ldots \ldots, x_n),
\end{aligned}$$

we can be assured that the general maximum function can be reduced to a multiple composition of $\max(x_1, x_2)$ on $\mathbb{R}^2$. Thus, we will explore the multiple composition formula for $g(x_1, x_2)$.

Theorem 4.5.2. *Suppose that $g : \mathbb{R}^2 \to \mathbb{R}$ is given by (4.2) in terms of $h(x, \mu)$ satisfying Condition 2. If we recursively define $g^{n-1} : \mathbb{R}^n \to \mathbb{R}$ for all $n \in \mathbb{N}$,*

- $g^0 : \mathbb{R} \to \mathbb{R}, \quad g^0(x_1) := x_1$;
- $g^1 : \mathbb{R}^2 \to \mathbb{R}, \quad g^1(x_1, x_2) := g(x_1, x_2) = g(g^0(x_1), x_2)$;
- $g^{n-1} : \mathbb{R}^n \to \mathbb{R}, \quad g^{n-1}(x_1, \ldots, x_{n-1}, x_n) := g(g^{n-2}(x_1, \ldots, x_{n-1}), x_n)$,

we have, for all $n \in \mathbb{N}$,

1. g^{n-1} is parameterized by $\mu > 0$, and it converges uniformly to $\max(x_1, \ldots, x_n)$ on $\mathbb{R}^n$ as $\mu \to 0$, since

$$|g^{n-1}(x_1, \ldots, x_n) - \max(x_1, \ldots, x_n)| \leqslant \frac{n-1}{2}\kappa\mu, \quad \forall (x_1, \ldots, x_n) \in \mathbb{R}^n.$$

2. g^{n-1} is smooth on $\mathbb{R}^n$, and we have for $i = 1, \ldots, n$,

$$\frac{\partial g^{n-1}}{\partial x_i}(x_1, \ldots, x_n) = A_i B_i,$$

where

$$B_1 := 1, B_i := \frac{\partial g}{\partial x_2}(g^{i-2}, x_i) \text{ for } i = 2, \ldots, n,$$
$$\text{and } A_n := 1, A_i := A_{i+1}(1 - B_{i+1}) \text{ for } i = n-1, \ldots, 1.$$

Proof. 1. The proof is by induction. The statement trivially holds for $n = 0$, and the previous Lemma 4.5.1 proves the case for $n = 1$. Suppose that $|g^{n-2}(x_1, \ldots, x_{n-1}) - \max(x_1, \ldots, x_{n-1})| \leqslant \frac{n-2}{2}\kappa\mu$ holds. For simplicity, we will refer to $g^{n-2}(x_1, \ldots, x_n)$ as g^{n-2} and $\max(x_1, \ldots, x_{n-1})$ as $*$. Then, we have

$$
\begin{aligned}
&|g^{n-1}(x_1, \ldots, x_n) - \max(x_1, \ldots x_n)| = |g(g^{n-2}, x_n) - \max(*, x_n)| \\
=&\frac{1}{2}|g^{n-2} + x_n + h(g^{n-2} - x_n) - * - x_n - |* - x_n||| \\
=&\frac{1}{2}|(g^{n-2} - *) + h(g^{n-2} - x_n) - |g^{n-2} - x_n| + |g^{n-2} - x_n| - |* - x_n|| \\
\leqslant&\frac{1}{2}|g^{n-2} - *| + \frac{1}{2}|h(g^{n-2} - x_n) - |g^{n-2} - x_n|| + \frac{1}{2}||g^{n-2} - x_n| - |* - x_n|| \\
\leqslant&\frac{1}{2}|g^{n-2} - *| + \frac{1}{2}|h(g^{n-2} - x_n) - |g^{n-2} - x_n|| + \frac{1}{2}|(g^{n-2} - x_n) - (* - x_n)| \\
\leqslant&|g^{n-2} - *| + \frac{1}{2}|h(g^{n-2} - x_n) - |g^{n-2} - x_n|| \\
\leqslant&\frac{n-2}{2}\kappa\mu + \frac{1}{2}\kappa\mu = \frac{n-1}{2}\kappa\mu.
\end{aligned}
$$

2. From the definition $g^{n-1} = g\left(g^{n-2}, x_n\right)$ and the chain rule, we have

$$
\begin{aligned}
\frac{\partial g^{n-1}}{\partial x_1}(x_1, \ldots, x_n) &= \frac{\partial g}{\partial x_1}\left(g^{n-2}, x_n\right)\frac{\partial g^{n-2}}{\partial x_1}(x_1, \ldots, x_{n-1}), \\
&\vdots \\
\frac{\partial g^{n-1}}{\partial x_{n-1}}(x_1, \ldots, x_n) &= \frac{\partial g}{\partial x_1}\left(g^{n-2}, x_n\right)\frac{\partial g^{n-2}}{\partial x_{n-1}}(x_n, \ldots, x_{n-1}), \\
\frac{\partial g^{n-1}}{\partial x_n}(x_1, \ldots, x_n) &= \frac{\partial g}{\partial x_2}\left(g^{n-2}, x_n\right).
\end{aligned}
$$

Further, we rearrange them to get

$$
\begin{aligned}
\frac{\partial g^{n-1}}{\partial x_1}(x_1, \ldots, x_n) &= \prod_{k=1}^{n-1}\frac{\partial g}{\partial x_1}(g^{k-1}, x_{k+1}), \\
\frac{\partial g^{n-1}}{\partial x_i}(x_1, \ldots, x_n) &= \frac{\partial g}{\partial x_2}\left(g^{i-2}, x_i\right)\prod_{k=i}^{n-1}\frac{\partial g}{\partial x_1}(g^{k-1}, x_{k+1}) \quad \text{for } i = 2, \ldots, n-1, \\
\frac{\partial g^{n-1}}{\partial x_n}(x_1, \ldots, x_n) &= \frac{\partial g}{\partial x_2}\left(g^{n-2}, x_n\right).
\end{aligned}
$$

Each term $\frac{\partial g^{n-1}}{\partial x_i}$ consists of only $\frac{\partial g}{\partial x_1}$ and $\frac{\partial g}{\partial x_2}$. However, the formulas above are very expensive

to calculate. If we expand and enumerate them in reverse order, we find that

$$\frac{\partial g^{n-1}}{\partial x_n}(x_1,\ldots,x_n) = \underbrace{\frac{\partial g}{\partial x_2}\big(g^{n-2},x_n\big)}_{B_n},$$

$$\frac{\partial g^{n-1}}{\partial x_{n-1}}(x_1,\ldots,x_n) = \underbrace{\frac{\partial g}{\partial x_1}\big(g^{n-2},x_n\big)}_{A_{n-1}:=1-B_n}\underbrace{\frac{\partial g}{\partial x_2}\big(g^{n-3},x_{n-1}\big)}_{B_{n-1}},$$

$$\frac{\partial g^{n-1}}{\partial x_{n-2}}(x_1,\ldots,x_n) = \underbrace{\frac{\partial g}{\partial x_1}\big(g^{n-2},x_n\big)\frac{\partial g}{\partial x_1}\big(g^{n-3},x_{n-1}\big)}_{A_{n-2}:=A_{n-1}(1-B_{n-1})}\underbrace{\frac{\partial g}{\partial x_2}\big(g^{n-4},x_{n-2}\big)}_{B_{n-2}},$$

$$\frac{\partial g^{n-1}}{\partial x_{n-3}}(x_1,\ldots,x_n) = \underbrace{\frac{\partial g}{\partial x_1}\big(g^{n-2},x_n\big)\frac{\partial g}{\partial x_1}\big(g^{n-3},x_{n-1}\big)\frac{\partial g}{\partial x_1}\big(g^{n-4},x_{n-2}\big)}_{A_{n-3}:=A_{n-2}(1-B_{n-2})}\underbrace{\frac{\partial g}{\partial x_2}\big(g^{n-5},x_{n-3}\big)}_{B_{n-3}},$$

$$\vdots$$

Note that $\frac{\partial g}{\partial x_1} + \frac{\partial g}{\partial x_2} = 1$ from (4.3). Thus, we have arrived at an easier form to calculate.

□

Chapter 5

Numerical Results

Not only is our proposal inspired by Groetzner's idea stated in Chapter 2, but Groetzner's method (specifically, Algorithm 2) is by far the best practical method of CP factorization; thus, we choose it as the main object of comparison. Thanks Groetzner for his code is available on the website [27]. Our experiments used many of the same instances given in Groetzner's paper [28]. The numerical experiments were performed on a computer equipped with Intel Core i7-4770 3.40 GHz and 16GB Ram. The algorithms were implemented in MatlabR2020a. The details of the experiments are as follows.

If A is full rank, for accuracy reasons, we obtain the initial B by using Cholesky decomposition. Otherwise, it is obtained by spectral decomposition. Then, we extend B to r columns by using column replication; see (2.1). We use this manner of extension throughout, same as Groetzner's experiments. Here, $r = \mathrm{cp}(A)$ if $\mathrm{cp}(A)$ is known, or r is sufficiently larger than $\mathrm{cp}(A)$, e.g. cp_n. We use `RandOrthMat.m` [40] to generate the random starting point X_0 based on the Gram–Schmidt process.

In Algorithm 4, we set $c_1 = 10^{-4}, c_2 = 0.9$ and use algorithm 3.5 and 3.6 in [37] to compute the step size. We set $\tau = 0.5, c_1 = 10^{-4}, \delta = 0.5, \eta = 0.5$ in Algorithm 5 and the same setting together with the initial setting of $\mu_0 = 10$ and $\mu_{step} = 1$ for Algorithm 6. We set fixed $\mu = 0.1$ in Algorithm 4 and 5. Algorithm 6 is usually used for most experimental scenarios if there are no special instructions.

The termination conditions are exactly same as Groetzner's experiments. All the algorithms terminate successfully at iteration k if $\min(BX_k)_{ij} \geqslant -10^{-15}$, or a maximum number of iterations (5000) is reached. In other words, it is considered successful as long as it terminates within 5000 iterations.

5.1 A Specifically Structured Example in Different Dimensions

Example 5.1.1 ([28, Example 7.1])**.** *Let $\boldsymbol{e}_n$ denote the all-ones vector in $\mathbb{R}^n$ and consider the matrix,*

$$A_n = \begin{pmatrix} 0 & \boldsymbol{e}_{n-1}^T \\ \boldsymbol{e}_{n-1} & I_{n-1} \end{pmatrix}^T \begin{pmatrix} 0 & \boldsymbol{e}_{n-1}^T \\ \boldsymbol{e}_{n-1} & I_{n-1} \end{pmatrix} \in \mathcal{CP}_n.$$

It has been shown in [39, Example 7.4] that $A_n \in \operatorname{int} \mathcal{CP}_n$ for every $n \geqslant 2$. By construction, it

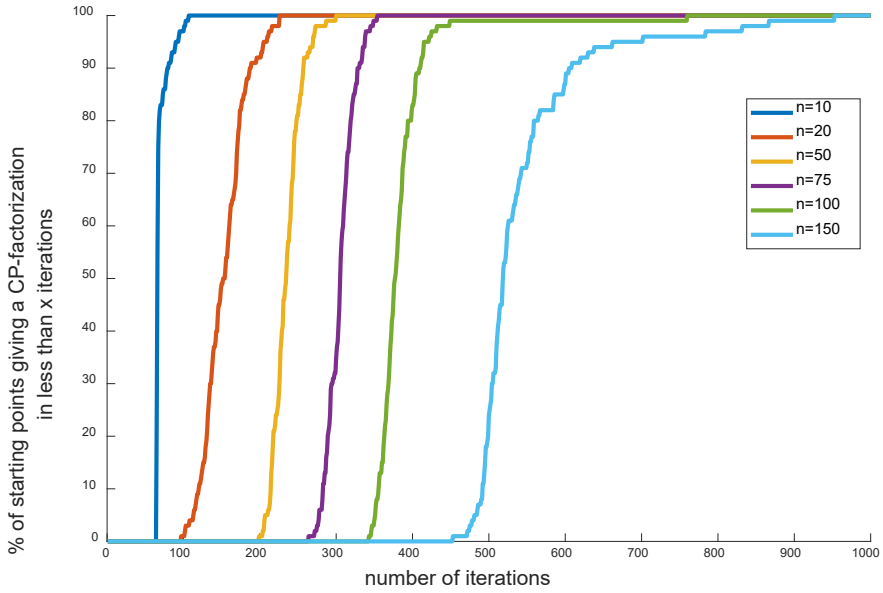

Fig. 5.1. Performance of Algorithm 6 for A_n from Example 5.1.1 with different dimensions n.

is obvious that $\mathrm{cp}(A_n) = n$. As in Groetzner's experimental setting, we try to factorize A_n for the values $n \in \{10, 20, 50, 75, 100, 150\}$. For each A_n, using $r = \mathrm{cp}(A_n) = n$ we tested 100 starting points and performed at most 5000 iterations for each starting point.

In contrast to [28, Fig. 1] where the success rate of Groetzner's method decreases to 15% as n increases to 150, it can be seen from Fig. 5.1 that Algorithm 6 succeeds for all dimensions n and all starting points. It should be noted that Algorithm 6 succeeds within 1000 iterations for each instance and does not require 5000 iterations in all cases.

5.2 Comparison of Algorithms 4, 5 and 6

The next experiment is based on Example 5.1.1, and it compares the performances of Algorithms 4, 5, and 6. All algorithms were run on the same experiment as in Example 5.1.1. For each algorithm and each A_n, we tested 100 starting points and performed at most 5000 iterations for each starting point, $r = \mathrm{cp}(A_n) = n$.

Table 5.1 demonstrates that each algorithm succeeds in all dimensions n and all starting points within 5000 iterations. As mentioned earlier, the BB step often accelerates the gradient method. We can see that Algorithm 5 does indeed take less time in each dimension n compared with Algorithm 4. Although Algorithm 5 may entail more iterations, the cost of a single iteration is much smaller.

Moreover, a decreasing μ leads to a remarkable improvement. From Table 5.1, Algorithm 6 takes much less time and much fewer iterations than either Algorithm 4 or 5. Although the best initial μ_0 and μ_{step} are not yet clear, a heuristic setting $\mu_0 = 10$ and $\mu_{step} = 1$ can reduce the computational time and iterations.

Table 5.1. Comparison of Algorithm 4, 5 and 6 for Example 5.1.1 with different dimensions n.

		av. no. of iterations		
n	$r = n$	Alg 4	Alg 5	Alg 6
10	10	34	40	72
20	20	152	176	147
50	50	554	899	237
75	75	545	1574	310
100	100	1771	2289	386
150	150	1584	3947	543

		av. time (sec.)		
n	$r = n$	Alg 4	Alg 5	Alg 6
10	10	0.007	0.007	0.009
20	20	0.069	0.044	0.038
50	50	1.7	1.0	0.3
75	75	3.6	3.5	0.6
100	100	19.7	8.6	1.4
150	150	47.4	36.1	4.2

5.3 Comparison of Algorithm 6 and Groetzner's method

In this section, we compare our approach and Groetzner's method directly. We use Algorithm 6 and Groetzner's method (Algorithm 2) to factorize $A := HH^T$, where $H \in \mathbb{R}^{n\times n}$ with entries randomly generated in $\{1, 2, \ldots, 10\}$ for the values $n \in \{10, 15, 20, 25, 30\}$. Hence, $\mathrm{cp}(A) = n$, We take $r = tn$ for $t \in \{1, 1.5, 2, 3\}$. For each pair of n and r, we generated 100 instances to examine. For each instance, both methods have up to 100 initial point opportunities. If it is not successful after 5000 iterations, then the next initial point will be tried. As long as one succeeds, no other initial points are tried. The time for a single instance is calculated from the first initial point to the last successful one.

It can be seen from Table 5.2 that our approach only took a very short time to succeed for all 100 instances in any pair of n and r. However, no matter time and success rate, Groetzner's method is worse than ours in any case of n and r. In particular, the number of successes is falling as r approaches to n.

5.4 On the Boundary of $\mathcal{CP}_n$

Now, let us examine how well our method works for a matrix on the boundary of $\mathcal{CP}_n$.

Example 5.4.1. *Consider the following matrix from [42, Example 2.7].*

$$A = \begin{pmatrix} 41 & 43 & 80 & 56 & 50 \\ 43 & 62 & 89 & 78 & 51 \\ 80 & 89 & 162 & 120 & 93 \\ 56 & 78 & 120 & 104 & 62 \\ 50 & 51 & 93 & 62 & 65 \end{pmatrix}.$$

The sufficient condition from [42, Theorem 2.5] ensures that this matrix is completely positive and $\mathrm{cp}(A) = \mathrm{rank}(A) = 3$. Theorem 1.3.7 tells that $A \in \mathrm{bd}\,\mathcal{CP}_5$, since $\mathrm{rank}(A) \neq 5$.

Referring to the results of [28, Example 7.2], we find that both Groetzner's method and Algorithm 6 can easily factorize this matrix, which implies that our method can also factorize a matrix on the boundary of $\mathcal{CP}_n$. However, Algorithm 6 returns the resulting factorization with the smallest entry as large as possible if we don't stop it as soon as it reaches zero. For example, our approach

Table 5.2. A direct comparison of Algorithm 6 and Groetzner's method.

$N = 100$		Algorithm 6		Groetzner's method (Algorithm 2)	
n	$r = n$	no. of successful cases	av. time (sec.) for successful cases	no. of successful cases	av. time (sec.) for successful cases
10	10	100	0.0010	95	3.17
15	15	100	0.0016	77	13.98
20	20	100	0.0025	2	26.76
25	25	100	0.0043	0	-
30	30	100	0.0056	0	-
n	$r \approx 1.5n$	no. of successful cases	av. time (sec.) for successful cases	no. of successful cases	av. time (sec.) for successful cases
10	15	100	0.0014	100	0.39
15	23	100	0.0027	100	3.78
20	30	100	0.0044	93	18.33
25	38	100	0.0089	65	55.10
30	45	100	0.0121	36	122.37
n	$r = 2n$	no. of successful cases	av. time (sec.) for successful cases	no. of successful cases	av. time (sec.) for successful cases
10	20	100	0.0031	100	0.41
15	30	100	0.0041	100	1.94
20	40	100	0.0076	99	9.56
25	50	100	0.0127	82	40.82
30	60	100	0.0166	50	64.87
n	$r = 3n$	no. of successful cases	av. time (sec.) for successful cases	no. of successful cases	av. time (sec.) for successful cases
10	30	100	0.0036	100	0.40
15	45	100	0.0080	100	2.80
20	60	100	0.0131	100	12.27
25	75	100	0.0222	92	48.91
30	90	100	0.0315	70	90.74

Table 5.3. Performance of Algorithm 6 for Example 5.4.2 with $n = 2$ and $n = 3$.

n	r	av. success rate (%)	av. time (sec.)	n	r	av. success rate (%)	av. time (sec.)
2	3	0	0.262	3	8	0	0.376
2	4	100	0.016	3	9	83	0.272
2	5	0	0.288	3	10	0	0.438

gives the following CP factor B whose the smallest entry is around 2.8573.

$$A = BB^T, \text{ where } B \approx \begin{pmatrix} 3.5771 & 4.4766 & \mathbf{2.8573} \\ 2.8574 & 3.0682 & 6.6650 \\ 8.3822 & 7.0001 & 6.5374 \\ 5.7515 & 2.8574 & 7.9219 \\ 2.8574 & 6.7741 & 3.3085 \end{pmatrix}.$$

Example 5.4.2. *Consider matrices of the type,*

$$\begin{pmatrix} nI_n & E_n \\ E_n & nI_n \end{pmatrix} \in \mathcal{CP}_{2n},$$

where $E_n \in \mathbb{R}^{n\times n}$ denotes the all-ones matrix.

These matrices do not have full rank and are therefore on the boundary. Contrary to Groetzner's failure to return a CP factor for these instances, from Table 5.3, we see that our method succeeds with $n = 2$ and $n = 3$ under a loose stopping condition that $\min (BX_k)_{ij} \geqslant -10^{-14}$. For each pair n and r, we used 100 starting points and a maximum of 5000 iterations per starting point. The value of r obviously affects the success rate for $n = 2$ and $n = 3$.

Example 5.4.3. *Consider the following matrix on the boundary taken from [23].*

$$A = \begin{pmatrix} 8 & 5 & 1 & 1 & 5 \\ 5 & 8 & 5 & 1 & 1 \\ 1 & 5 & 8 & 5 & 1 \\ 1 & 1 & 5 & 8 & 5 \\ 5 & 1 & 1 & 5 & 8 \end{pmatrix} \in \operatorname{bd}\mathcal{CP}_5.$$

Since A is full rank, then $\operatorname{cp}^+(A) = \infty$; i.e., there is no strictly positive CP factor for A. Hence, for A, the global minimum of (OptP), $t = 0$ is clear. Neither Groetzner's method nor our method can decompose this matrix. However, Algorithm 6 returns a decomposition whose the smallest entry is -10^{-5} that is close to zero. This result demonstrates that for (OptP), the difference between a local minimum and the global minimum is less than 10^{-5} in this case.

As in [28, Example 7.3], we will investigate slight perturbations of this matrix. We test the convex combinations of A and the following $C \in \operatorname{int}\mathcal{CP}_5$.

$$C = MM^T, \text{ where } M = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Table 5.4 shows the performance of Algorithm 6 in factorizing $A_\lambda := \lambda A + (1-\lambda)C$ for different values $\lambda \in [0, 1]$, using $r = 12 > \operatorname{cp}_5 = 11$. For each A_λ, we used 100 starting points and a maximum of 5000 iterations per starting point. The second column shows the total computation time for all 100 starting points.

In contrast with [28, Table 1] showing that Groetzner's method decreases as it approaches the boundary, the success rate of our method is always 100%. Moreover, the total running time of all 100 starting points is much less for each λ.

Table 5.4. Performance of Algorithm 6 for slight perturbations A_λ with different values of $\lambda \in [0, 1]$.

λ	time (sec.)	success rate (%)
0	0.64	100
0.1	0.77	100
0.2	0.75	100
0.3	0.74	100
0.4	0.76	100
0.5	0.87	100
0.6	1.11	100
0.7	1.58	100

λ	time (sec.)	success rate (%)
0.8	2.37	100
0.9	4.18	100
0.95	8.02	100
0.97	12.99	100
0.976	16.06	100
0.977	16.62	100
0.9774	17.08	100
0.9775	17.38	100

Table 5.5. Performance of Algorithm 6 for randomly generated matrices A of high order.

n	r	av. success rate (%)	av. no. of iterations	av. time (sec.)
50	51	100	51	0.06
50	151	100	63	0.27
100	151	100	70	0.42
100	301	100	79	1.41
150	201	100	84	0.89
200	301	100	99	2.31
1000	1500	100	390	641

5.5 Randomly Generated Examples of High Order

Finally, we examine the case of randomly generated matrices of high order to investigate how Algorithm 6 is affected by the order. The instances were generated in the same way as [28, Section 7.7]: We computed C by setting $C_{ij} := |B_{ij}|$ for all i, j, where B is a random $n \times k$ matrix based on the Matlab command `randn`, and we took $A = CC^T$ to be factorized. Here, $k = 2n$. We tested 100 starting points for $n \leqslant 50$ and 10 starting points for $n > 50$. In each case, we set a maximum of 5000 iterations per starting point. The numbers in columns 3–5 of Table 5.5 represent averages of 100 randomly generated instances.

Comparing [28, Table 3] and Table 5.5, we find that, for all randomly generated instances, the success rate of algorithm 6 reaches 100% and it is significantly faster than Groetzner's even though the specifications of our computer are inferior to those of Groetzner's, that is equipped with 88 Intel Xenon ES-2699 cores (2.2 GHz each) and a total of 0.792 TB Ram. Due to our inferior computer, unfortunately, we cannot finish the same experiments with Groetzner's method within a reasonable time, thus making direct comparisons impossible.

Chapter 6

Conclusion and Further Remarks

Now, let us come to a conclusion, then briefly mention the advantages and disadvantages of our method and the new directions that can be explored.

This thesis was devoted to the CP factorization B of a given completely positive matrix. We proposed a new numerical method, which stems from the idea that the CP factorization problem can be reformulated as a feasibility problem. Its solution can be regarded as an optimization problem with orthogonality constraints and a maximum function as the objective. To be able to apply a curvilinear search, we made a smooth approximation to the maximum function, called LogSumExp. We found that this method is much faster than most of the other CP factorization algorithms. The reason why our method is so fast is that only the inversion of an $n \times n$ matrix dominates the computation at each iteration, not other subproblems like SOCP. Compared with the existing CP factorization methods mentioned in Section 1.4, speed is our biggest advantage.

We know that LogSumExp can approximate the maximum function infinitely and globally. Let LSE_{μ_1} denote the LogSumExp function parameterized by positive μ_1. Because of global convergence, the curvilinear search for LSE_{μ_1} yields a local minimizer of LSE_{μ_1}, say X_1^*. If we continue to optimize with a higher degree approximation $LSE_{\mu_2}(\mu_2 < \mu_1)$ from the point X_1^*, the curvilinear search also yields a local minimizer, say X_2^*. Intuitively, they converge to a local minimizer of the maximum function itself. Unfortunately, we have not been able to prove this assertion yet.

The limitation is that there is no guarantee that Algorithm 4 (hence 5,6) will successfully find a CP factorization for any $A \in \mathcal{CP}^n$. Let us assume our method guarantees a local minimum of (OptP). Recall that $A \in \mathcal{CP}_n$ if and only if the global minimum of (OptP), say t, is such that $t \leqslant 0$. Since checking membership is NP-hard, we cannot expect to compute t in general, or estimate t relying on a local minimum. For example, it may be that the local minimum is greater than zero, but the global minimum t is equal to zero, which means $A \in \mathcal{CP}^n$. Here, we can refer back to Example 5.4.3, where the algorithm failed to terminate. However, in most cases, we can succeed by finding a local maximum of (OptP) greater than zero.

Finally, let us focus on Stiefel manifold optimization. In fact, it is included in a larger research topic — Riemannian manifold optimization, which is a kind of optimization on matrix manifolds. Intuitively, a manifold is locally like Euclidean space $\mathbb{R}^n$ near each point. This roughly explains why we can use a gradient method on the Stiefel manifold, just like in unconstrained optimization on $\mathbb{R}^n$. Therefore, to a certain extent, the mature methods developed for nonsmooth optimization on

$\mathbb{R}^n$ (such as the subgradient method and ADMM) can be applied to manifold optimization. Actually, many studies have been reported, cf. [24, 6, 34]. Thus, we can apply those nonsmooth techniques to the Stiefel manifold to solve (OptP) directly without a smooth approximation. This will also be a future work.

Acknowledgements

This thesis would never have been possible without the support and guidance of various people at the Graduate School for Policy and Planning Sciences, University of Tsukuba. Firstly, I would like to thank Akiko Yoshise for giving me the wonderful opportunity to complete my master's thesis under her supervision; it is truly an honor. Thank you for all the advice, ideas, moral support, and patience in guiding me through this thesis. Your wealth of knowledge in the field of mathematical optimization is inspiring. Thank you for giving me the opportunity to grow in this field of research. I would also like to express my appreciation to my classmates — Zhang Kai, Wang Yuzhu, and Shinichi Kanoh, who provided advice, insight, and expertise that greatly assisted the research. Lastly, I would like to thank my mom and dad, Dai Fendi and Lai Liangao. Your devotion, unconditional love, support, patience, optimism, and advice was more valuable than you could ever imagine.

Bibliography

[1] B. ABRAHAM AND S.-M. NAOMI, *Completely Positive Matrices*, World Scientific, 2003.

[2] Y. J. BAGUL AND C. CHESNEAU, *Sigmoid functions for the smooth approximation to* $|x|$. Preprints 2019, 2019030140 (doi: 10.20944/preprints201903.0140.v1).

[3] Y. J. BAGUL AND B. K. KHAIRNAR, *A note on smooth transcendental approximation to* $|x|$. Preprints 2019, 2019020190 (doi: 10.20944/preprints201902.0190.v1).

[4] J. BARZILAI AND J. M. BORWEIN, *Two-point step size gradient methods*, IMA journal of numerical analysis, 8 (1988), pp. 141–148.

[5] H. H. BAUSCHKE AND J. M. BORWEIN, *On the convergence of von neumann's alternating projection algorithm for two sets*, Set-Valued Analysis, 1 (1993), pp. 185–212.

[6] G. C. BENTO AND J. G. MELO, *Subgradient method for convex feasibility on riemannian manifolds*, Journal of Optimization Theory and Applications, 152 (2012), pp. 773–785.

[7] A. BERMAN, M. DÜR, AND N. SHAKED-MONDERER, *Open problems in the theory of completely positive and copositive matrices*, Electronic Journal of Linear Algebra, 29 (2015), pp. 46–58.

[8] D. S. BERNSTEIN, *Matrix Mathematics: Theory, Facts, and Formulas*, Princeton university press, 2009.

[9] I. M. BOMZE, *Copositive optimization–recent developments and applications*, European Journal of Operational Research, 216 (2012), pp. 509–520.

[10] I. M. BOMZE, P. J. DICKINSON, AND G. STILL, *The structure of completely positive matrices according to their cp-rank and cp-plus-rank*, Linear algebra and its applications, 482 (2015), pp. 191–206.

[11] I. M. BOMZE, M. DÜR, E. DE KLERK, C. ROOS, A. J. QUIST, AND T. TERLAKY, *On copositive programming and standard quadratic optimization problems*, Journal of Global Optimization, 18 (2000), pp. 301–320.

[12] I. M. BOMZE, W. SCHACHINGER, AND G. UCHIDA, *Think co (mpletely) positive! matrix properties, examples and a clustered bibliography on copositive optimization*, Journal of Global Optimization, 52 (2012), pp. 423–445.

[13] S. BURER, *On the copositive representation of binary and continuous nonconvex quadratic programs*, Mathematical Programming, 120 (2009), pp. 479–495.

[14] S. BURER, *A gentle, geometric introduction to copositive optimization*, Mathematical Programming, 151 (2015), pp. 89–116.

[15] E. DE KLERK AND D. V. PASECHNIK, *Approximation of the stability number of a graph via copositive programming*, SIAM Journal on Optimization, 12 (2002), pp. 875–892.

[16] P. H. DIANANDA, *On non-negative forms in real variables some or all of which are non-negative*, in Mathematical Proceedings of the Cambridge Philosophical Society, vol. 58, Cambridge University Press, 1962, pp. 17–25.

[17] P. J. DICKINSON, *An improved characterisation of the interior of the completely positive cone*, Electronic Journal of Linear Algebra, 20 (2010), pp. 723–729.

[18] P. J. DICKINSON, *The Copositive Cone, the Completely Positive Cone and Their Generalisations*, PhD thesis, University of Groningen, 2013.

[19] P. J. DICKINSON AND M. DÜR, *Linear-time complete positivity detection and decomposition of sparse matrices*, SIAM Journal on Matrix Analysis and Applications, 33 (2012), pp. 701–720.

[20] P. J. DICKINSON AND L. GIJBEN, *On the computational complexity of membership problems for the completely positive cone and its dual*, Computational optimization and applications, 57 (2014), pp. 403–415.

[21] D. DRUSVYATSKIY, A. D. IOFFE, AND A. S. LEWIS, *Transversality and alternating projections for nonconvex sets*, Foundations of Computational Mathematics, 15 (2015), pp. 1637–1651.

[22] M. DÜR, *Copositive programming–a survey*, in Recent advances in optimization and its applications in engineering, Springer, 2010, pp. 3–20.

[23] M. DÜR AND G. STILL, *Interior points of the completely positive cone*, The Electronic Journal of Linear Algebra, 17 (2008).

[24] O. FERREIRA AND P. OLIVEIRA, *Subgradient algorithm on riemannian manifolds*, Journal of Optimization Theory and Applications, 97 (1998), pp. 93–104.

[25] B. GAO AND L. PAVEL, *On the properties of the softmax function with application in game theory and reinforcement learning*, 2018, `https://arxiv.org/abs/1704.00805`.

[26] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, 2013.

[27] P. GROETZNER, *Matlab code for cp-factorization* `https://sites.google.com/site/groetznerpatrick/codes-and-files`, 2018.

[28] P. GROETZNER AND M. DÜR, *A factorization method for completely positive matrices*, Linear Algebra and its Applications, 591 (2020), pp. 1–24.

[29] P. H. GROETZNER, *A Method for Completely Positive and Nonnegative Matrix Factorization*, PhD thesis, University of Trier, 2018.

[30] M. HALL AND M. NEWMAN, *Copositive and completely positive quadratic forms*, in Mathematical Proceedings of the Cambridge Philosophical Society, vol. 59, Cambridge University Press, 1963, pp. 329–339.

[31] J. HANNAH AND T. J. LAFFEY, *Nonnegative factorization of completely positive matrices*, Linear algebra and its applications, 55 (1983), pp. 1–9.

[32] B. IANNAZZO AND M. PORCELLI, *The riemannian barzilai–borwein method with nonmonotone line search and the matrix geometric mean computation*, IMA Journal of Numerical Analysis, 38 (2018), pp. 495–517.

[33] F. JARRE AND K. SCHMALLOWSKY, *On the computation of C^* certificates*, Journal of Global Optimization, 45 (2009), p. 281.

[34] A. KOVNATSKY, K. GLASHOFF, AND M. M. BRONSTEIN, *MADMM: a generic algorithm for non-smooth optimization on manifolds*, in European Conference on Computer Vision, Springer, 2016, pp. 680–696.

[35] J. E. MAXFIELD AND H. MINC, *On the matrix equation $X^T X = A$*, Proceedings of the Edinburgh Mathematical Society, 13 (1962), pp. 125–129.

[36] J. NIE, *The $\mathcal{A}$-truncated K-moment problem*, Foundations of Computational Mathematics, 14 (2014), pp. 1243–1276.

[37] J. NOCEDAL AND S. WRIGHT, *Numerical Optimization*, Springer Science & Business Media, 2006.

[38] K. B. PETERSEN AND M. S. PEDERSEN, *The matrix cookbook (version: November 15, 2012)*, 2012.

[39] B. SHADER, N. SHAKED-MONDERER, AND D. B. SZYLD, *Nearly positive matrices*, Linear Algebra and its Applications, 449 (2014), pp. 520–544.

[40] O. SHILON, *Randorthmat* `https://www.mathworks.com/matlabcentral/fileexchange/11783-randorthmat`. MATLAB Central File Exchange. Retrieved January 23, 2021.

[41] M. D. SIKIRIĆ, A. SCHÜRMANN, AND F. VALLENTIN, *A simplex algorithm for rational cp-factorization*, Mathematical Programming, (2020), pp. 1–21.

[42] W. SO AND C. XU, *A simple sufficient condition for complete positivity*, Operators and Matrices, 9 (2015), pp. 233–239.

[43] J. SPONSEL AND M. DÜR, *Factorization and cutting planes for completely positive matrices by copositive projection*, Mathematical Programming, 143 (2014), pp. 211–229.

[44] Z. WEN AND W. YIN, *A feasible method for optimization with orthogonality constraints*, tech. report, Rice University, 2010.

[45] Z. WEN AND W. YIN, *A feasible method for optimization with orthogonality constraints*, Mathematical Programming, 142 (2013), pp. 397–434.

[46] C. XU, *Completely positive matrices*, Linear algebra and its applications, 379 (2004), pp. 319–327.

[47] H. ZHANG AND W. W. HAGER, *A nonmonotone line search technique and its application to unconstrained optimization*, SIAM journal on Optimization, 14 (2004), pp. 1043–1056.