Low-Rank Matrix Splitting and ADMM for Scalable and Efficient Semidefinite Programming

Chris Junchi Li[◇]

Department of Electrical Engineering and Computer Sciences[◇]
University of California, Berkeley

October 7, 2024

Abstract

This paper presents a novel low-rank Alternating Direction Method of Multipliers (ADMM) framework for solving large-scale Semidefinite Programming (SDP) problems. We introduce a new matrix splitting technique, which reformulates the SDP problem into a bilinear constrained and quadratically penalized optimization task. By incorporating low-rank factorization into the ADMM structure, the proposed algorithm reduces memory overhead while maintaining computational efficiency. Theoretical convergence guarantees are provided, and extensive numerical experiments demonstrate the superior performance of the method on various SDP benchmarks. This approach proves particularly effective in handling large-scale problems, achieving both accuracy and scalability where traditional methods falter.

Keywords: Semidefinite Programming (SDP), ADMM, Low-Rank Factorization, Matrix Splitting, Optimization Algorithms.

1 Introduction

Semidefinite programming (SDP) plays a fundamental role in various fields such as control theory, combinatorial optimization, and machine learning, offering a robust framework for solving convex optimization problems. However, solving large-scale SDP problems is often computationally challenging, primarily due to the quadratic memory and time complexities associated with traditional methods like interior point techniques. In recent years, first-order methods, including Augmented Lagrangian and ADMM-based algorithms, have emerged as promising alternatives, particularly for large-scale instances. These methods circumvent the expensive computations of interior point methods by leveraging structure and sparsity.

Despite these advancements, a significant challenge remains in handling large SDP problems efficiently due to the increasing size of matrix variables and the associated memory constraints. Low-rank approaches, which exploit the fact that SDP solutions often exhibit low-rank structures, offer a potential solution. By factorizing the matrix variable, these methods aim to reduce the memory and computational overhead. However, many existing low-rank approaches rely on non-convex formulations, complicating the convergence analysis.

In this paper, we propose a new ADMM-based approach that integrates matrix splitting with low-rank factorization for solving general SDP problems. Our method recasts the original SDP problem into a bilinear form, introducing penalties that facilitate the use of ADMM while retaining the problem's convexity. The resulting algorithm, named LoRADS (Low-Rank ADMM Splitting), demonstrates superior scalability and efficiency in addressing large-scale SDPs. By incorporating a dynamic rank reduction strategy, the algorithm further optimizes performance, dynamically adjusting the rank during iterations based on problem complexity.

Mathematically, our goal is to address the general semidefinite program (SDP) in the following form:

$$\min_{X\in\mathbb{S}^n} \quad f(X) \quad \text{s.t.} \quad \mathcal{A}(X) = b, \quad X \succeq 0 \tag{1}$$

where $\mathbb{S}^n$ represents the space of all the $n \times n$ symmetric matrices, $f(X)$ denotes a convex and continuously differentiable function, and the linear map $\mathcal{A}(\cdot) : \mathbb{S}^n \to \mathbb{R}^m$ is defined as $\mathcal{A}(X) = (\langle A_1, X\rangle, \ldots, \langle A_m, X\rangle)^\top$ with each $A_i \in \mathbb{S}^n$ for $i = 1, 2, \ldots, m$. Specifically, when $f(X)$ has the linear form, we have the linear SDP problem:

$$\min_{X\in\mathbb{S}^n} \quad \langle C, X\rangle \quad \text{s.t.} \quad \mathcal{A}(X) = b, \quad X \succeq 0 \tag{2}$$

where $C \in \mathbb{S}^n$. Throughout this paper, we conduct theoretical analyses on the general SDP form (1) while focusing on the linear form (2) for computational explorations.

SDPs have a wide range of applications in practice, such as the MaxCut problem [GW95], optimal power flow [LL11], combinatorial optimization [BV97] and sensor network localization [SY07], to name a few. Traditionally, interior point methods are the go-to solution for solving SDP problems. It has been well known that for SDP, interior point methods can achieve high accuracy in polynomial time. See [Ali95, NN94, Ali91]. Despite their effectiveness, traditional interior point methods struggle with scalability for large-scale problems due to the necessity of solving Newton systems repeatedly, which is computationally intensive. To bypass the Newton system and enhance scalability, more recent research has shifted towards the first-order methods [WGY10, LLM11, OCPB16]. A significant area of interest has been the use of penalty methods, such as the augmented Lagrangian method and alternating direction method of multipliers (ADMM), for solving SDPs [WGY10, ZST10, YST15, WDLW23]. Specifically, [ZST10] introduced SDPNAL, which employs the augmented Lagrangian method on dual SDP and uses an inexact semi-smooth Newton-CG method to solve the inner subproblem. The subsequent work [YST15] presented an enhanced algorithm SDPNAL+, which achieves improved efficiency and robustness and adequately addresses the numerical difficulty of SDPNAL when solving degenerate SDPs. [OCPB16] proposes a general splitting conic solver (SCS) for linear and conic programming, which applies ADMM to solve the homogeneous self-dual embedding associated with SDP. For large-scale SDP, a critical issue is the quadratically growing ($\mathcal{O}(n^2)$) memory cost in maintaining the matrix variable. For a class of rank-constrained matrix problems, [YUTC17] apply the conditional gradient method to their convex relaxation problems. To reduce the storage overheads, they employ random sketching to keep track of a low-rank approximation of the matrix variable. In a follow-up work by [YTF+21], random sketching is further incorporated in the augmented Lagrangian-based method to deal with more general trace-constrained SDPs.

In addition to the mentioned convex approaches, another important research direction for large-scale SDP is to exploit the low-rank structure of SDP solutions directly through nonconvex reformulation. In the work of [Pat98, Bar95], the authors have shown that standard SDP admits an optimal solution with rank r satisfying $r(r+1)/2 \le m$ for specific $r > 0$. Due to this low-rank structure, [BM03] propose to replace the semidefinite matrix X with the factorized form $X = UU^\top$, where $U \in \mathbb{R}^{n\times r}$. This factorization gives rise to the following nonconvex formulation of (2):

$$\min_{U\in\mathbb{R}^{n\times r}} \quad \langle C, UU^\top\rangle \quad \text{s.t.} \quad \mathcal{A}(UU^\top) = b \tag{3}$$

[BM03, BM05] propose an effective limited memory BFGS augmented Lagrangian algorithm to handle the low-rank formulation. Another popular low-rank approach for SDP is the Riemannian

optimization methods [JBAS10, WY13, Bou15]. These methods are built upon the assumption that the feasible set of (3) is a Riemannian manifold. Within the low-rank formulation, [TT23] proposes a rank-support adaptive Riemannian optimization method for solving the general convex SDP problems and gives the convergence rate analysis. There are also several different ways to solve the low-rank formulation of SDP. [BGP21] propose a relaxed variant of the interior point method, which exploits the low-rank structure of the optimal solution. [EOPV22] apply the block-coordinate ascent method to solve the low-rank reformulation of SDP with diagonal constraints and establish global convergence towards the first-order stationary point. In the BM formulation, a critical step is determining the rank of the matrix variable. The rank r is chosen to ensure that either an optimal solution for (3) is attainable or a satisfactory approximate low-rank solution is obtained. In particular, [SYZ08] demonstrate the existence of a solution of logarithmic rank order that still maintains good approximation property. This discovery forms the basis of an empirical tactic to initiate the algorithm with a solution of rank $\mathcal{O}(\log(m))$, which will be employed in our approach.

Our paper's contributions are summarized as follows: First, we introduce a novel ADMM algorithm tailored for general SDPs that accommodate nonlinear and convex objectives. By introducing a new matrix splitting approach, we recast the SDP into a bilinearly constrained and quadratically penalized nonconvex problem, much different from the Burer-Monteiro approach. This formulation offers two primary benefits for ADMM implementation: 1) For a linear objective, enabled by the splitting technique, both the alternating steps in ADMM result in a convex and well-conditioned quadratic problem, and hence, can be efficiently solved by either direct methods or indirect ones such as the conjugate gradient descent method. This simplification provides a substantial advantage over the nonconvex quartic subproblems inherent in the BM formulation; 2) This splitting provides an enlarged search area, which could potentially improve the algorithm's performance on some hard instances. Under mild conditions, we prove that the ADMM algorithm achieves a sublinear rate of convergence to the KKT solutions. Additionally, when the Łojasiewicz inequality is satisfied with an exponent of $1/2$, we enhance the convergence result to a linear rate, offering a significant improvement in the algorithm's efficiency.

Second, we present a new SDP solver, which employs low-rank factorization and ADMM through matrix splitting (LoRADS). This solver integrates the proposed ADMM along with several new implementation strategies that significantly improve the practical efficiency. Our empirical evaluations highlight two critical heuristics: 1) a warm-start procedure employing the Burer-Monteiro algorithm, and 2) aggressive rank reduction to logarithmic order with a dynamic rank selection strategy ensuring the stability. By leveraging these enhancements, we compare LoRADS with the leading SDP solvers on a wide range of problems, including MaxCut, matrix completion, and general SDP benchmarks in SDPLIB and Mittlemann's dataset. Extensive numerical studies demonstrate that LoRADS is highly competitive and often outperforms the popular SDP solvers. Furthermore, LoRADS demonstrates notable efficacy in addressing super large-scale problems, an area where current SDP solvers show limitations. A case in point is LoRADS's ability to solve a matrix completion problem with $15,694,167$ constraints and a 40000×40000 matrix variable within approximately 350 seconds.

A closely related work is [CG23], which also propose an ADMM-based method modified from the Burer-Monteiro factorization. Similar to our work, they employ matrix splitting to generate more manageable subproblems. However, their focus is restricted to a particular subset of SDP problems where the linear constraint is only applied to the diagonal elements. Due to the diagonally-constrained structure, when converted to the BM formulation, the closed-form solution of the

projection can be efficiently computed. In stark contrast, our study encompasses a much wider range of SDP problem types, incorporating general linear constraints alongside convex nonlinear objectives. Given the challenges associated with handling general SDP, we adopt a substantially different approach in our convergence analysis to achieve convergence and complexity results.

Contributions: Our main contributions are summarized as follows:

- We introduce a novel ADMM algorithm tailored for general SDPs, utilizing a low-rank matrix splitting technique that preserves convexity and improves scalability.
- We provide theoretical convergence guarantees, showing sublinear convergence to the Karush-Kuhn-Tucker (KKT) points.
- We present an efficient solver, LoRADS, which leverages low-rank factorization, ADMM, and dynamic rank selection to handle large-scale SDP problems effectively.
- Extensive empirical evaluations demonstrate the superior performance of LoRADS compared to existing state-of-the-art SDP solvers, particularly on large-scale problems.

Organization: The rest of the paper is organized as follows. Section 2 introduces the proposed ADMM algorithm and matrix splitting formulation for semidefinite programming. In Section 3, we present the convergence analysis, establishing both sublinear and linear convergence rates under specific conditions. Section 4 details the implementation of LoRADS, including practical strategies and heuristics to improve computational efficiency. Section 5 provides numerical experiments and comparative results across a range of SDP problems. Finally, Section 6 concludes the paper and outlines future research directions.

Notations. Let $\|\cdot\|_2$ denote the Euclidean norm for vectors and operator norm for matrices, and $\|\cdot\|_F$ denote the Frobenius norm for matrices. For a given matrix X and radius δ, we use $\mathcal{B}_\delta(X)$ to denote the ball of radius δ around X, i.e. $\mathcal{B}_\delta(X) = \{Y | \|X - Y\|_F \leq \delta\}$. For a nonempty and compact set $\mathcal{S}$, we reuse the notation $\mathcal{B}_\delta(\mathcal{S})$ to denote the neighborhood of $\mathcal{S}$, which is defined as $\mathcal{B}_\delta(\mathcal{S}) = \{Y | \inf_{Z\in\mathcal{S}} \|Y - Z\|_F \leq \delta\}$. For any matrix X, let $\sigma_{\min}(X)$ denote the minimal singular value of X. Let $\|\mathcal{A}\|_2$ denote the operator norm of the linear map $\mathcal{A}$ and $s_A = \sum_{i=1}^m \|A_i\|_F^2$. Given a sequence of matrices $\{X^k\}_{k=0}^{K-1}$ and a sequence of vectors $\{x^k\}_{k=0}^{K-1}$, we use ΔX_k to denote $X^k - X^{k-1}$ and Δx^k to denote $x^k - x^{k-1}$, respectively.

2 The Low-Rank Bilinear Decomposition and the ADMM Approach

In this section, we present a general low-rank ADMM method with matrix splitting for solving the SDP problem. While we shall develop a more practical SDP solver for the standard linear SDP (2) in Section 4, the ADMM algorithm and the subsequent analysis are adaptable to accommodate a general convex objective beyond the linear function. For this reason, this and the next sections will focus on the general convex SDP (1).

2.1 Bilinear Reformulations and The Penalized Problem

The Burer-Monteiro factorization for the problem is expressed as follows:

$$\min_{U\in\mathbb{R}^{n\times r}} \quad f(UU^\top) \quad \text{s.t.} \quad \mathcal{A}(UU^\top) = b \tag{4}$$

To address the challenges posed by large-scale SDP instances and to thoroughly explore the low-rank domain, we propose a different and bilinear factorization technique for problem (1), yielding the following formulation:

$$\min_{U,V\in\mathbb{R}^{n\times r}} \quad f(UV^\top) \quad \text{s.t.} \quad \mathcal{A}(UV^\top) = b, \quad U = V \tag{5}$$

We further consider the penalized version of (5) by treating $U-V$ as a penalty term in the objective

$$\min_{U,V\in\mathbb{R}^{n\times r}} \quad f(UV^\top) + \frac{\gamma}{2}\|U-V\|_F^2 \quad \text{s.t.} \quad \mathcal{A}(UV^\top) = b \tag{6}$$

To elucidate the relationship between model (4) and (6) with a suitably large parameter γ, we introduce the following assumption.

Assumption 1. *The set collected by the KKT points of* (6)*, denoted by $\mathcal{W}$, is nonempty. The function f is symmetric in $\mathbb{R}^{n\times n}$, i.e., $f(X) = f(X^\top)$ for any matrix $X \in \mathbb{R}^{n\times n}$, and ∇f and $\nabla^2 f$ are Lipschitz continuous on any convex compact set.*

For any given $\sigma > 0$ and $X \in \mathcal{B}_{\sigma^2}\big(0^{n\times n}\big)$, there exist constants L_f^σ and L^σ such that $\|\nabla f(X)\|_F \le L_f^\sigma$ and ∇f is L^σ-Lipschitz continuous. Moreover, $\nabla^2 f$ is L_0^σ-Lipschitz continuous on $\mathcal{B}_{\sigma^2}\big(0^{n\times n}\big)$, i.e.

$$\|\nabla^2 f(X)[H] - \nabla^2 f(Y)[H]\|_F \le L_0^\sigma \|H\|_F \|X-Y\|_F, \quad \forall X, Y \in \mathcal{B}_{\sigma^2}\big(0^{n\times n}\big), H \in \mathbb{R}^{n\times n}$$

where $\nabla^2 f(X)[H]$ denotes the tensor product of $\nabla^2 f(X)$ and H. From the symmetry of f, it is easy to verify that the partial gradients satisfy

$$\frac{\partial f(UV^\top)}{\partial U} = \nabla f(UV^\top)V \quad \text{and} \quad \frac{\partial f(UV^\top)}{\partial V} = \nabla f(VU^\top)U$$

With this preparation, we are ready to give the definitions of ϵ-KKT point of problems (4) and (6).

Definition 1. *We say that (U^*, λ^*) is an ϵ-KKT point of problem* (4) *if*

$$\begin{aligned} \Big\|2\nabla f(U^*(U^*)^\top)U^* + \sum_{i=1}^m \lambda_i^* A_i U^*\Big\|_F &\le \epsilon, \\ \big\|\mathcal{A}(U^*(U^*)^\top) - b\big\|_2 &\le \epsilon \end{aligned} \tag{7}$$

and $(\overline{U}, \overline{V}, \overline{\lambda})$ is an ϵ-KKT point of problem (6) *if it satisfies*

$$\begin{aligned} \Big\|\nabla f(\overline{U}\,\overline{V}^\top)\overline{V} + \gamma(\overline{U}-\overline{V}) + \sum_{i=1}^m \overline{\lambda}_i A_i \overline{V}\Big\|_F &\le \epsilon, \\ \Big\|\nabla f(\overline{V}\,\overline{U}^\top)\overline{U} + \gamma(\overline{V}-\overline{U}) + \sum_{i=1}^m \overline{\lambda}_i A_i \overline{U}\Big\|_F &\le \epsilon, \\ \big\|\mathcal{A}(\overline{U}\,\overline{V}^\top) - b\big\|_2 &\le \epsilon \end{aligned} \tag{8}$$

When $\epsilon = 0$, the definition of ϵ-KKT points reduces to that of KKT points.

The following theorem states that the solution produced by our algorithm for (6) can be suitably adapted to address (4) approximately.

Theorem 1. *Let $(\overline{U}, \overline{V}, \overline{\lambda})$ be any ϵ-KKT point of problem (6) in $\mathcal{B}_\sigma\big(0^{n\times r}, 0^{n\times r}, 0^m\big)$ and $\widehat{U} = \frac{\overline{U}+\overline{V}}{2}$. Suppose Assumption 1 holds, then the following statement s valid.*

$$\begin{aligned} &\left\|\mathcal{A}(\widehat{U}\widehat{U}^\top) - b\right\|_2 \leq \epsilon + \frac{\|\mathcal{A}\|_2(\epsilon + L_f^\sigma\sigma + \sqrt{s_A}\sigma^2)^2}{4\gamma^2}, \\ &\left\|2\nabla f(\widehat{U}\widehat{U}^\top)\widehat{U} + 2\sum_{i=1}^m \overline{\lambda}_i A_i\widehat{U}\right\|_F \leq 2\epsilon + \frac{(2L_0^\sigma\sigma^3 + 3L^\sigma\sigma)(\epsilon + L_f^\sigma\sigma + \sqrt{s_A}\sigma^2)^2}{2\gamma^2} \end{aligned} \tag{9}$$

Moreover, suppose that we choose the penalty parameter γ such that

$$\gamma > \max\left\{\frac{\sqrt{\|\mathcal{A}\|_2}(\epsilon + L_f^\sigma\sigma + \sqrt{s_A}\sigma^2)}{2\sqrt{2\epsilon}}, \frac{\sqrt{2L_0^\sigma\sigma^3 + 3L^\sigma\sigma}(\epsilon + L_f^\sigma\sigma + \sqrt{s_A}\sigma^2)}{\sqrt{2\epsilon}}\right\}$$

then $(\widehat{U}, 2\overline{\lambda})$ is a 3ϵ-KKT point of problem (4).

Theorem 1 further indicates that if we have an ϵ-KKT point of the penalty problem with $\gamma = \Omega(1/\sqrt{\epsilon})$, then we are able to obtain an ϵ-KKT point of problem (4). As we will see later, this theoretical insight corroborates the empirical evidence observed in our study.

2.2 The ADMM Approach

We describe ADMM for solving the penalty problem (6) in Algorithm 1. In this algorithm, the augmented Lagrangian function of (6) is defined by

$$\mathcal{L}_\rho(U, V, \lambda) = f(UV^\top) + \frac{\gamma}{2}\|U - V\|_F^2 + \big\langle\lambda, \mathcal{A}(UV^\top) - b\big\rangle + \frac{\rho}{2}\|\mathcal{A}(UV^\top) - b\|_2^2 \tag{10}$$

By leveraging the bilinear factorization, ADMM alternates between optimizing two proximal subproblems associated with U and V. Since f is convex, both the subproblems are γ-strongly convex with their own variables. The optimality conditions of the subproblems can be written as

$$\begin{aligned} 0 =& \nabla f(U^{k+1}(V^k)^\top)V^k + \gamma(U^{k+1} - V^k) + \sum_{i=1}^m \lambda_i^k A_i V^k \\ &+ \rho\sum_{i=1}^m \Big(\big\langle A_i V^k, U^{k+1}\big\rangle - b_i\Big) A_i V^k \end{aligned} \tag{11}$$

and

$$\begin{aligned} 0 =& \nabla f(V^{k+1}(U^{k+1})^\top)U^{k+1} - \gamma(U^{k+1} - V^{k+1}) + \sum_{i=1}^m \lambda_i^k A_i U^{k+1} \\ &+ \rho\sum_{i=1}^m \big(\big\langle A_i U^{k+1}, V^{k+1}\big\rangle - b_i\big) A_i U^{k+1} \end{aligned} \tag{12}$$

For linear SDPs, it is noteworthy to point out that due to the γ penalty term from the splitting, both the subproblems (11) and (12) reduce to well-conditioned linear systems, which can be solved

Algorithm 1: LRADMM

Initialize U^0, V^0, λ^0 and penalty parameter ρ.
Set $k = 0$
while *the termination criteria is not satisfied* **do**
    $U^{k+1} \leftarrow \operatorname{argmin}_U \mathcal{L}_\rho(U, V^k, \lambda^k)$
    $V^{k+1} \leftarrow \operatorname{argmin}_V \mathcal{L}_\rho(U^{k+1}, V, \lambda^k)$
    $\lambda^{k+1} = \lambda^k + \rho\big(\mathcal{A}(U^{k+1}(V^{k+1})^\top) - b\big)$
    Set $k \leftarrow k + 1$
end

effectively by the conjugate gradient (CG) method. Numerical results supporting this view can be found in Section 4.4.

For the stopping criterion of ADMM, we terminate the algorithm if

$$\max\left\{\frac{\|\mathcal{A}(U^{k+1}(V^{k+1})^\top) - b\|_2}{1 + \|b\|_\infty}, \frac{\gamma\|V^{k+1} - V^k\|_F}{1 + \|V^k\|_F}\right\} < \epsilon \tag{13}$$

is met, where $||\cdot||_\infty$ is the infinity norm of a vector. From (11) and (12), we can easily derive that

$$\begin{aligned}
&\nabla f(U^{k+1}(V^{k+1})^\top)V^{k+1} + \gamma(U^{k+1} - V^{k+1}) + \sum_{i=1}^m \lambda_i^{k+1} A_i V^{k+1} \\
&= \nabla f(U^{k+1}(V^{k+1})^\top)(V^{k+1} - V^k) + \Big(\nabla f(U^{k+1}(V^{k+1})^\top) - \nabla f(U^{k+1}(V^k)^\top)\Big) V^k \\
&\quad + \gamma(V^k - V^{k+1}) + \rho\sum_{i=1}^m \big\langle A_i(V^{k+1} - V^k), U^{k+1}\big\rangle A_i V^k + \sum_{i=1}^m \lambda_i^{k+1} A_i(V^{k+1} - V^k)
\end{aligned} \tag{14}$$

and

$$\nabla f(V^{k+1}(U^{k+1})^\top)V^{k+1} + \gamma(V^{k+1} - U^{k+1}) + \sum_{i=1}^m \lambda_i^{k+1} A_i V^{k+1} = 0 \tag{15}$$

Assume $\|V^k\|_F \le \sigma$ and $(U^{k+1}, V^{k+1}, \lambda^{k+1}) \in \big(\mathcal{B}_\sigma(0^{n\times r})\big)^2 \times \mathcal{B}_\sigma(0^m)$. By taking the Frobenius norm on both sides of (14) and using triangle inequality, it is not difficult to show

$$\begin{aligned}
&\|\nabla f(U^{k+1}(V^{k+1})^\top)V^{k+1} + \gamma(U^{k+1} - V^{k+1}) + \sum_{i=1}^m \lambda_i^{k+1} A_i V^{k+1}\|_F \\
&\le \big(L_f^\sigma + L^\sigma\sigma^2 + \rho s_A\sigma^2 + \sqrt{s_A}\sigma + \gamma\big)\|V^{k+1} - V^k\|_F \\
&\le \left(\frac{L_f^\sigma + L^\sigma\sigma^2 + \rho s_A\sigma^2 + \sqrt{s_A}\sigma}{\gamma} + 1\right)(1+\sigma)\epsilon \triangleq c\epsilon
\end{aligned} \tag{16}$$

This implies that the k-th iterate $(U^{k+1}, V^{k+1}, \lambda^{k+1})$ is a $(\max\{c, 1 + \|b\|_\infty\} \cdot \epsilon)$-KKT point of problem (6) and thus $\big(\frac{U^{k+1}+V^{k+1}}{2}, 2\lambda^{k+1}\big)$ is an $O(\epsilon)$-KKT point of (4) by taking $\gamma = \Omega(1/\sqrt{\epsilon})$.

2.3 The Power of ADMM

The intuition for constructing the bilinear factorization stems from the observation that within the iterative trajectory of LRADMM, U and V do not need to be identical. This splitting affords an enlarged search area, which sometimes improves the algorithm performance on hard instances. Figure 1 depicts a scenario in which the Burer-Monteiro approach is unable to solve the SDP relaxation of a sensor network localization (SNL) problem, whereas LRADMM demonstrates success. Given the location of anchors observed distances information between sensor-sensor and sensor-anchor pairs, the goal of SNL is to determine the sensor's true position, represented by the blue star. In a more formal setting, for a d-dimensional graph $G(V, E)$, we differentiate between the sensor set $V_x = \{x_1, x_2, \cdots, x_n\}$ and the anchor set $V_a = \{a_1, a_2, \cdots, a_m\}$ being a partition of V, where these sets collectively partition V and $m \geq d + 1$. Observable edges are represented by two sets: $N_x = \{(i, j) \mid i, j \in V_x\}$ for sensor-sensor pairs and $N_a = \{(i, j) \mid i \in V_a, j \in V_x\}$ for sensor-anchor pairs. The lengths of these edges are denoted as $d_{ij} = \|x_i - x_j\|_2\,,\ \forall (i, j) \in N_x$ for sensor-sensor edges and $\widetilde{d}_{kj} = \|a_k - x_j\|_2\,,\ \forall (k, j) \in N_a$ for sensor-anchor edges. The SNL problem is formulated as follows:

$$\min_x \sum_{(i,j)\in N_x} \left| \|x_i - x_j\|_2^2 - d_{ij}^2 \right| + \sum_{(k,j)\in N_a} \left| \|a_k - x_j\|_2^2 - \widetilde{d}_{kj}^2 \right|$$

Let $e_i \in \mathbb{R}^n$ be the vector with 1 at i-th position and 0 at others, and $\mathbf{0} \in \mathbb{R}^d$ be the all-zeros vector. The SDP relaxation is then given by

$$\begin{array}{ll} \max & 0 \\ \text{s.t.} & Z_{(1:d,1:d)} = I_d \\ & \left\langle (\mathbf{0}; e_i - e_j)(\mathbf{0}; e_i - e_j)^\top, Z \right\rangle = d_{ij}^2, \quad \forall (i, j) \in N_x \\ & \left\langle (a_k; -e_j)(a_k; -e_j)^\top, Z \right\rangle = \widetilde{d}_{kj}^2, \quad \forall (k, j) \in N_a \\ & Z \succeq 0 \end{array}$$

where the solution $Z \in \mathbb{R}^{(d+n)\times(d+n)}$ comprises submatrices as:

$$Z = \begin{bmatrix} I & X \\ X^\top & Y \end{bmatrix}$$

If $Y = X^\top X$, then Z is a feasible solution for the above SDP problem, and $X = [x_1\ x_2\ \cdots\ x_n]$ gives a set of feasible locations of the sensors. The original SNL case under consideration involves a single sensor and three anchors, with their true positions illustrated in Figure 1. We claim that the original SNL problem possesses a local minimum near the coordinates $(0.5, -0.9)$, posing potential difficulties for solving the SDP relaxation. The Burer-Monteiro and LRADMM methods' iterations, when mapped back to the original SNL context, indicate the sensor's coordinates in a two-dimensional space, as depicted in the figures. According to the left part of Figure 1, the Burer-Monteiro method becomes stuck at the local minimum of the original problem, thereby failing to place the sensor. For comparison, we initialize ADMM from the point where the Burer-Monteiro method gets stuck. As shown on the right part of Figure 1, ADMM successfully navigates away from this region and converges to the true position. The SNL instance is constructed as described in [LZY23], while the SDP relaxation of SNL is detailed in [BY04, BLWY06, WZYB08, STZY12].

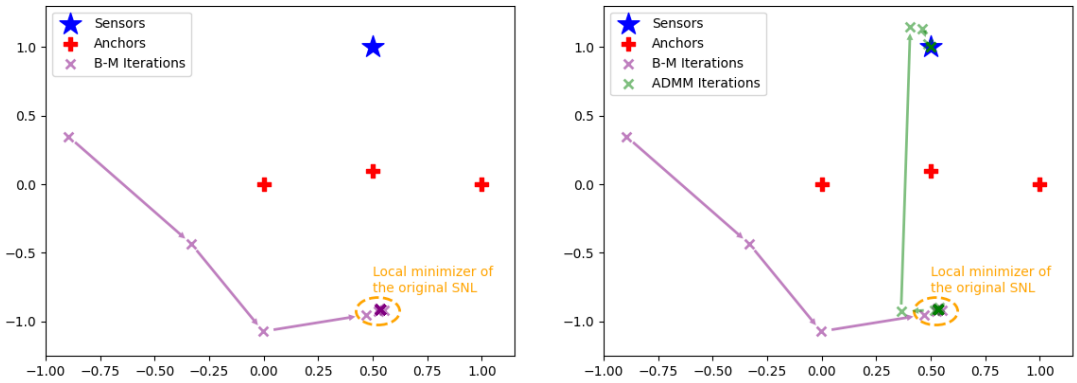

Figure 1: Comparison of ADMM and Burer-Monteiro Approaches on the SNL Instance

3 Convergence Analysis of LRADMM

In this section, we present the main theoretical results of our proposed Algorithm 1. We will prove the convergence of LRADMM for solving the bilinear factorized problem and further improve the convergence rate under the assumption of Łojasiewicz inequality.

3.1 Descent Property of LRADMM

Our initial emphasis is on examining the change in the augmented Lagrangian function's value after each iteration. In the subsequent proposition, we demonstrate precisely how the updates of U^{k+1}, V^{k+1}, and λ^{k+1} affect the augmented Lagrangian function's value.

Proposition 1. *Suppose that Assumption 1 holds. Let* $\{(U^k, V^k, \lambda^k)\}_{k=0}^{K-1}$ *be the sequence generated by Algorithm 1, then for any* $k \geq 0$*, we have*

$$\mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^{k+1}) - \mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^k) = \frac{1}{\rho}\|\lambda^{k+1} - \lambda^k\|_2^2, \tag{17}$$

$$\mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^k) - \mathcal{L}_\rho(U^k, V^k, \lambda^k) \leq -\frac{\gamma}{2}(\|V^{k+1} - V^k\|_F^2 + \|U^{k+1} - U^k\|_F^2) \tag{18}$$

It is worth noting that the update of the dual variable λ leads to an increase in the value of the augmented Lagrangian function. Therefore, bounding $\|\lambda^{k+1} - \lambda^k\|_2^2$ is crucial to ensure a sufficient decrease in each iteration. To this end, we need some boundedness and nonsingularity conditions.

Assumption 2. *There exists a* $\rho_0 > 0$ *such that for any given* β*, the level set* $S_\beta := \{(U, V) \mid f(UV^\top) + \frac{\rho_0}{2}\|U - V\|_F^2 + \frac{\rho_0}{2}\|\mathcal{A}(UV^\top) - b\|_2^2 \leq \beta\}$ *is either compact or empty.*

We remark that Assumption 2, while seemingly stringent, is applicable in a broad range of scenarios. The subsequent proposition demonstrates the widespread applicability of Assumption 2 across various contexts.

Proposition 2. *Assumption 2 holds for some* $\rho_0 > 0$ *if either of the following conditions is satisfied:*

(i) $f(X) + \frac{\rho_0}{2}\|\mathcal{A}(X) - b\|_2^2$ *strongly convex with respect to* $X \in \mathbb{R}^{n\times n}$.

(ii) $f(X) = \langle C, X\rangle$ *with* $C \in \mathcal{S}^n$*, and the dual problem of* (2) *has a strictly feasible solution* y^**. Define* $\lambda_{\min}$ *and* $\lambda_{\max}$ *as the smallest and largest eigenvalues of* $C - \sum_{i=1}^m y_i^* A_i$*, respectively, and let* $\kappa_c = \frac{\lambda_{\max}}{\lambda_{\min}}$ *represent its condition number. Set* $\rho_0 \geq 1 + \lambda_{\max}\kappa_c$*.*

According to Assumption 1, the nonlinear system $\mathcal{A}(UV^T) = b$ is feasible. Hence, we can locate an initial point (U^0, V^0, λ^0) near the solution of $\mathcal{A}(UV^T) = b$ such that

$$\mathcal{L}_\rho(U^0, V^0, \lambda^0) + \frac{1}{\rho}\|\lambda^0\|_2^2 \leq \beta^0 < \widehat{\beta} \tag{19}$$

where β^0 and $\widehat{\beta}$ are constants independent of ρ. Using Assumption 2, we establish that $S_{\widehat{\beta}}$ is bounded, and there exists a constant $\delta > 0$ such that $\|U\|_F \leq \delta$ and $\|V\|_F \leq \delta$.

We also need the following assumption, which assumes that the constraints $\mathcal{A}(UU^T) = b$ are linearly independent within the region $S_{\widehat{\beta}}$. More previously, let the function $\mathcal{C} : \mathbb{R}^{n\times r} \to \mathbb{R}^{nr\times m}$ denote the map from any point U to the matrix consisting of the vectorized gradients of these constraints $\mathcal{A}(UU^T) = b$, i.e., $\mathcal{C}(U) = [\mathrm{vec}\,(A_1U), \cdots, \mathrm{vec}\,(A_mU)]$. We assume that $\mathcal{C}(U)$ is of full rank for any point (U, V) in the region $S_{\widehat{\beta}}$.

Assumption 3. *There exists a constant* $\eta > 0$ *such that*

$$\min_{(U,V)\in S_{\widehat{\beta}}} \sigma_{\min}\big(\mathcal{C}(U)\big) \geq \eta$$

where and $\sigma_{\min}$ *denotes the least singular value of a matrix.*

Assumption 3 is called the linear independence constraint qualification (LICQ), which is standard in the convergence analysis of ADMM-type methods. For example, see [LMZ15, WGY10, XW21].

With these foundations in place, we are now prepared to establish the bound of $\|\lambda^{k+1} - \lambda^k\|_2^2$ and hence the descent property of our proposed Algorithm 1.

Lemma 1. *Suppose the initial point* (U^0, V^0, λ^0) *is chosen according to* (19)*, and Assumptions 1–3 hold. Let* ρ *and* γ *be selected in Algorithm 1 such that*

$$\rho > \max\left\{2\rho_0, \frac{2c_1}{\gamma}, \frac{2c_2}{\gamma}, \frac{1}{\widehat{\beta} - \beta^0}\left(\frac{L_f^\delta\delta + 2\gamma\delta}{\eta}\right)^2\right\},\ \gamma > \rho_0. \tag{20}$$

Then, for any $k \geq 0$*, we have*

(i) $\mathcal{L}_\rho(U^k, V^k, \lambda^k) \leq \beta^0$ *and* $\mathcal{L}_\rho(U^k, V^k, \lambda^k) + \frac{3}{4\rho}\|\lambda^k\|_2^2 < \widehat{\beta}$*;*

(ii) $(U^k, V^k) \in S_{\widehat{\beta}}$*;*

(iii) $\|\lambda^{k+1} - \lambda^k\|_2^2 \leq c_1\|\Delta U^{k+1}\|_F^2 + c_2\|\Delta V^{k+1}\|_F^2$*, where* $c_1 = \frac{2}{\eta^2}\big(L^\delta\delta^2 + L_f^\delta + \gamma + \frac{1}{\eta}L_f^\delta\delta\sqrt{s_A} + \frac{2}{\eta}\gamma\delta\sqrt{s_A}\big)^2$ *and* $c_2 = \frac{2}{\eta^2}\big(L^\delta\delta^2 + \gamma\big)^2$*;*

(iv) $\mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^{k+1}) - \mathcal{L}_\rho(U^k, V^k, \lambda^k) \leq -(\frac{\gamma}{2} - \frac{c_1}{\rho})\|\Delta U^{k+1}\|_F^2 - (\frac{\gamma}{2} - \frac{c_2}{\rho})\|\Delta V^{k+1}\|_F^2$.

3.2 Convergence and Convergence Rate of ADMM

Next, we proceed to establish the convergence of the whole sequence generated by Algorithm 1. To accomplish this, it is necessary to impose certain conditions on the augmented Lagrangian function, such as the Łojasiewicz property. Denote the set collected by the KKT points of (6) as $\mathcal{W}$ and the description of Łojasiewicz inequality is shown as follows.

Assumption 4 (Łojasiewicz Inequality)**.** *The augmented Lagrangian function $\mathcal{L}_\rho(U,V,\lambda)$ satisfies the Łojasiewicz inequality with exponent $\alpha \in [0,1)$ at any KKT point of problem (6), i.e. there exist constants $c, \delta_c > 0$ such that for any $(U,V,\lambda) \in \mathbb{R}^{n\times r}\times\mathbb{R}^{n\times r}\times\mathbb{R}^m$ and $(\overline{U},\overline{V},\overline{\lambda}) \in \mathcal{W}$ satisfying* $\operatorname{dist}\big((U,V,\lambda),(\overline{U},\overline{V},\overline{\lambda})\big) \le \delta_c$,

$$|\mathcal{L}_\rho(U,V,\lambda) - \mathcal{L}_\rho(\overline{U},\overline{V},\overline{\lambda})|^\alpha \le c\|\nabla\mathcal{L}_\rho(U,V,\lambda)\|_F. \tag{21}$$

Assumption 4 is non-restrictive as it remains valid in various instances. For example, when f is a lower-semicontinuous semi-algebraic or tame function [ABS13], the augmented Lagrangian function $\mathcal{L}_\rho$ adheres to this condition. Furthermore, the Łojasiewicz inequality inequality is satisfied for $\mathcal{L}_\rho$ at every interior point of its domain. With this assumption, we are ready to state the convergence of the sequence.

Theorem 2. *Suppose Assumptions 1– 4 hold. Let the initial point (U^0,V^0,λ^0) and ρ,γ satisfy the conditions of Lemma 1. Then, the sequence of iterates $\{(U^k,V^k,\lambda^k)\}_{k\ge0}^{K-1}$ converges to a KKT point of problem (6), $(\overline{U},\overline{V},\overline{\lambda})$. Moreover, the total iteration complexity to achieve an ϵ-KKT point is $\mathcal{O}(\epsilon^{-1})$.*

Therefore, we have established the $O(1/k)$ convergence rate of our ADMM method. However, as observed in our numerical experiments, Algorithm 1 shows an even faster convergence rate in most cases. This is achievable if we could precisely characterize the exponent α of the Łojasiewicz inequality.

Theorem 3. *Let the initial point (U^0,V^0,λ^0) and ρ,γ satisfy the conditions of Lemma 1. Suppose that Assumption 1– 4 hold. Then the following conditions hold*

(i) If $\alpha = 0$, then the sequence $\{(U^k,V^k,\lambda^k)\}_{k=0}^{\infty}$ converges in a finite number of steps.

(ii) If $\alpha \in (0,\frac{1}{2}]$, then there exists $k_0 \ge 0$, $\xi > 0$ and $\mu \in [0,1)$ such that for any $k > k_0$

$$\operatorname{dist}((U^k,V^k,\lambda^k),\mathcal{W}) \le \xi\mu^{k-k_0}.$$

(iii) If $\alpha \in (\frac{1}{2},1)$, then there exists $k_0 \ge 0$, $\xi > 0$ such that for any $k > k_0$

$$\operatorname{dist}((U^k,V^k,\lambda^k),\mathcal{W}) \le \xi(k-k_0)^{-\frac{1-\alpha}{2\alpha-1}}.$$

4 An Enhanced Approach

In this section, we introduce LoRADS, an SDP solver based on the Low-rank ADMM Splitting approach, for effectively solving large-scale SDP problems. First, we give an outline of LoRADS in Section 4.1. Then, in Section 4.2, we show the motivation of adding a warm-start phase that universally employing LRADMM does not guarantee stability. Subsequently, we propose a logarithmic dynamic rank selection strategy in Section 4.3, which significantly improves the performance of LoRADS. Finally, we detail the warm-started CG method used to solve the ADMM subproblems and show its good performance.

Algorithm 2: LoRADS

1 Initialize $\widetilde{U}^0, \widetilde{\lambda}^0$ and penalty parameter ρ
2 **Phase I (the warm-start phase):** Call LRALM to solve problem (4) with initialization $(\widetilde{U}^0, \widetilde{\lambda}^0, \rho)$, and output a solution pair $(\widetilde{U}^*, \widetilde{\lambda}^*)$ once the **warm start** termination criteria is satisfied;
3 **Switching Phases:** Set $U^0 = V^0 = \widetilde{U}^*, \lambda^0 = \frac{1}{2}\widetilde{\lambda}^*$;
4 **Phase II (the solving phase):** Call LRADMM to solve problem (6) with initialization $(U^0, V^0, \lambda^0, \rho)$, and output a solution pair $(\overline{U}, \overline{V})$ when the **final** termination criteria is satisfied;
5 Set $\widehat{U} = \frac{\overline{U}+\overline{V}}{2}, \widehat{X} = \widehat{U}\widehat{U}^\top$ and output $\widehat{X}$ as a solution for (1);

4.1 LoRADS

LoRADS is designed to derive a satisfactory low-rank approximate solution for the linear SDP problem (2). The approach unfolds in two phases: Phase I centers on the Burer-Monteiro factorization problem (4). To solve this problem, we employ the augmented Lagrangian algorithm [BM03], initiating with a random starting point and employing the L-BFGS method to enhance algorithmic efficacy.

In the ensuing discussion, we refer to the augmented Lagrangian algorithm applied to solve problem (4) as LRALM. When problem (4) is solved to a certain accuracy, we stop the augmented Lagrangian algorithm and take the secured solution $(\widetilde{U}^*, \widetilde{\lambda}^*)$. Then, we switch to Phase II, focusing on solving the penalized bilinear decomposed problem (6) with LRADMM. We split the matrix variable U and initiate LRADMM by setting $U^0 = V^0 = \widetilde{U}^*, \lambda^0 = \frac{1}{2}\widetilde{\lambda}^*$, where (U^0, V^0, λ^0) is expected to be a satisfactory initial point as stated in Lemma 1. The LRADMM is then executed to derive an approximate solution $(\overline{U}, \overline{V})$ for problem (6) satisfying the final tolerance. By setting $\widehat{U} = \frac{\overline{U}+\overline{V}}{2}$, we obtain an intermediate solution for problem (4). Finally, with $\widehat{X} = \widehat{U}\widehat{U}^\top$, we get a solution to the general SDP problem (1). The complete pseudocode for the LoRADS is presented in Algorithm 2.

4.2 The Motivation of the Two-phases Structure

Although we aim at developing an approach based on the previously proposed LRADMM, our exploration reveals that directly applying Algorithm 1 to solve (6) might not always ensure stability. Empirically, initiating from arbitrary starting points occasionally leads to suboptimal performance. In view of our theoretical result (Theorem 2), Algorithm 1 exhibits promising convergence properties when the initial point (U^0, V^0, λ^0) aligns with the conditions specified in Lemma 1. These conditions imply that U^0 and V^0 should be close to the optimal solutions U^* and V^* to ensure fast convergence. To address this, we propose a warm-start approach, which initializes ADMM by the Burer-Monteiro method [BM03], as BM keeps the low-rank factorization and directly yields a factorized solution $X = UU^\top$. To illustrate the ideas discussed above, we compare the warm-start approach against applying LRADMM throughout and applying LRALM throughout on a set of representative problems selected[1]. In this experiment, we maintain a constant rank of $\sqrt{2m}$ and employ SDPLR as the solver for LRALM. Other implementation details and experiment settings follow Section 5.1 and 5.2.

[1]They are selected from the problems tested in Section 5.

Table 1: The effectiveness of The Warm-start Approach (Solving Times)

Problem	warm-started LRADMM	LRADMM	LRALM (SDPLR)
G60	3.47	6.61	7.66
MC_3000	154	t	819
G40_mb	29.76	32.62	59.95
H3O	19.37	f	199
p_auss2_3.0	56.85	158	83.95
qap7	1.41	2.14	2.18
qap10	6.47	21.51	10.19
theta12	62.26	22.76	127

[1] We denote "timeout" by "t" and "fail" by "f", and the solving times are capped at 1800 seconds.
[2] For LRADMM, we use an initial $\rho = 100$.

From Table 1, we observe that while LRADMM outperforms the LRALM in solving problems such as G60, G40_mb and theta12, it can be notably slower and may even fail in the other cases. However, the warm-started LRADMM exhibits a stable performance across the problems tested. Therefore, to enhance ADMM's stability, we integrate a warm-start phase using LRALM. More empirical results presented in Section 5 indicate that using LRALM for the warm-start phase is adequately effective, though exploring more effective warm-start strategies presents an intriguing direction for future research.

4.3 The Dynamic Rank Selection Strategy

To improve the computation and memory efficiency, it is often desirable to apply a smaller rank while maintaining the algorithm's stability. Most literature on low-rank SDP algorithms typically recommends a rank of $\sqrt{2m}$ to maintain favorable properties of local minima. However, a theoretical result by [SYZ08] suggests that an approximate solution of satisfactory quality can be efficiently achieved by employing a rank of $O(\log m)$. The theorem underscoring this idea is stated as follows:

> **Theorem 1.1 of [SYZ08]** Let $A_1, \ldots, A_m \in \mathbb{R}^{n\times n}$ be symmetric positive semidefinite matrices, and let $b_1, \ldots, b_m \geq 0$. Suppose that there exists an $X \succeq 0$ such that $\langle A_i, X\rangle = b_i$ for $i = 1, 2, \ldots, m$. Let $r = \min\{\sqrt{2m}, n\}$. Then, for any $d \geq 1$, there exists an $X_0 \succeq 0$ with $\operatorname{rank}(X_0) \leq d$ such that:
>
> $$\beta(m, n, d) \cdot b_i \leq \langle A_i, X_0\rangle \leq \alpha(m, n, d) \cdot b_i \quad \text{for } i = 1, \ldots, m$$
>
> where:
>
> $$\alpha(m, n, d) = \begin{cases} 1 + \frac{12\log(4mr)}{d} & \text{for } 1 \leq d \leq 12\log(4mr) \\ 1 + \sqrt{\frac{12\log(4mr)}{d}} & \text{for } d > 12\log(4mr) \end{cases}$$
>
> and
>
> $$\beta(m, n, d) = \begin{cases} \frac{1}{e(2m)^{2/d}} & \text{for } 1 \leq d \leq 4\log(2m) \\ \max\left\{\frac{1}{e(2m)^{2/d}}, 1 - \sqrt{\frac{4\log(2m)}{d}}\right\} & \text{for } d > 4\log(2m) \end{cases}$$
>
> Moreover, there exists an efficient randomized algorithm for finding such an X_0.

Table 2: Solving Times Comparison of Different Rank Strategy

Problem			$2\log(m)$		$\sqrt{2m}$		Dynamic Rank	
Name	n	m	Rank	Time	Rank	Time	Final Rank	Time
G60	7000	7000	18	0.70	118	3.47	18	0.66
MC_3000	3000	930328	27	9.94	1364	154	27	10.34
G40_mb	2001	2001	15	24.86	63	29.76	15	19.89
H3O	3163	2964	16	10.14	77	19.37	16	10.12
p_auss2_3.0	9116	9115	18	4.45	135	56.85	18	4.42
qap7	50	358	12	9.03	27	1.41	26	1.02
qap10	101	1021	14	f	45	6.47	45	2.97
theta12	601	90020	23	f	424	62.26	77	86.23

This theoretical insight motivates us to aggressively adopt an $O(\log(m))$ rank to further reduce storage requirements and computational costs. Nevertheless, employing a rank r that does not satisfy $r \geq \sqrt{2m}$ may make the problem more challenging. In some scenarios, fixing a rank of $O(\log(m))$ may result in slow convergence or even failure to find a solution. Our strategy is simple: start with a small rank order (of $O(\log(m))$), increasing it only when necessary. We observed that adjusting the rank solely during the warm-start phase suffices, as LRADMM is empirically more capable of handling low-rank instances than LRALM. To determine when to increase the rank, we employ a heuristic to identify when a problem becomes "difficult" to solve, reflected by the number of inner iterations of the ALM subproblems. Upon detecting the difficulty, we halt the current subproblem's iterative cycle, increase the rank by a factor α, and then proceed with the algorithm's iterations. The results presented in Table 2 compare the performance of LoRADS when applying a consistent rank of $r = 2\log(m)$, $r = \sqrt{2m}$, or employing the dynamic rank selection strategy, where the rank is multiplied by a factor of 1.5 each time. We continue to use the selected problems in the previous subsection to illustrate the concept, with the experiment settings following the descriptions in Section 5.1 and Section 5.2. From Table 2, it can be seen that using a rank of $2\log(m)$ significantly reduces solving times compared to $\sqrt{2m}$ in many cases, while, for some other cases, a rank of $2\log(m)$ makes the problem slower to solve (e.g., qap7) or even unsolvable (e.g., qap10, theta12). On the other hand, the dynamic rank strategy can leverage the advantages of a low-rank structure while bypassing its potential challenges.

4.4 Solving the ADMM Subproblems with the Warm-started CG

Since we consider the linear SDP problems, where the subproblems (11) and (12) of LRADMM with respect to (U^{k+1}, V^{k+1}) can be reformulated into the following linear systems with dimension $nr \times nr$:

$$\begin{aligned} &\left(\rho \sum_{i=1}^{m} \mathrm{vec}(A_i V^k)\mathrm{vec}(A_i V^k)^\top + \gamma I_{nr\times nr}\right) \mathrm{vec}(U^{k+1}) \\ =& -\mathrm{vec}\left(CV^k - \gamma V^k + \sum_{i=1}^{m} \lambda_i^k A_i V^k - \rho \sum_{i=1}^{m} b_i A_i V^k\right) \end{aligned} \tag{22}$$

Table 3: Statistics of the Number of CG Iterations

Problem	$nr = n\sqrt{2m}$	ADMM Iterations	Total CG Iters	Avg CG Iters
G60	828251	100	845	8.45
MC_3000	4092176	17	9626	566.24
G40_mb	126586	104	2252	21.65
p_auss2_3.0	1230829	1656	11152	6.73
qap7	1338	1829	91128	49.82
qap10	4564	29452	226521	7.69
theta12	255011	1261	6938	5.50

[1] H3O is not included here since it fails to be solved using only LRADMM (see Table 1).

and

$$\begin{aligned} &\left(\rho\sum_{i=1}^{m}\mathrm{vec}(A_iU^{k+1})\mathrm{vec}(A_iU^{k+1})^\top + \gamma I_{nr\times nr}\right)\mathrm{vec}(V^{k+1}) \\ =&-\mathrm{vec}\left(CU^{k+1} - \gamma U^{k+1} + \sum_{i=1}^{m}\lambda_i^k A_iU^{k+1} - \rho\sum_{i=1}^{m} b_iA_iU^{k+1}\right) \end{aligned} \tag{23}$$

where $\mathrm{vec}(\cdot)$ is the vectorization of a matrix.

To address the linear systems discussed, we employ a warm-started CG method. In this approach, we initialize the algorithm with V^k (or U^k) to efficiently solve the linear system for U^{k+1} (or V^{k+1}). It is important to highlight that the inclusion of the γ penalty term from the splitting significantly improves the conditioning of the linear systems (22) and (23). Consequently, when γ is sufficiently large, these systems are well-conditioned and can be effectively solved using the CG method without any preconditioning. To demonstrate the efficacy of this method, we present in Table 3 the number of ADMM iterations, the total CG iterations required to solve the problem, and the average number of CG iterations per ADMM iteration (involving two subproblems) across the selection of representative problems. The problems are solved solely using LRADMM without Burer-Monteiro warm-start, and the ranks are set to be $\sqrt{2m}$. The result clearly shows that the average number of CG iterations per subproblem is extremely smaller than the worst-case scenario of nr iterations.

5 Numerical Experiments

In this section, we provide a detailed empirical evaluation of LoRADS applied to a diverse set of SDP problems. We begin by outlining the key implementation details and the experiment setup in Section 5.1. Subsequently, in Section 5.2, we assess the performance of our method on three sets of SDP problems: MaxCut, Matrix Completion, and a collection of general SDP benchmark problems.

5.1 Implementation Details

We implement LoRADS[2] in ANSI C with open-source numerical linear algebra libraries, Lapack [ABB+99] and Blas [BPP+02]. To solve the subproblems of LRADMM, we use the warm-started conjugate gradient method to solve the linear systems (22) and (23), where $f(UV^\top) =$

[2] LoRADS is available at `https://github.com/COPT-Public/LoRADS`

$\langle C, UV^\top \rangle$. For the dynamic rank adjustment strategy discussed previously, we initiate the rank at $2\log(m)$ and increment the rank by a factor of 1.5 with each rank update. Especially, given the extensive scale of matrix completion problems from MC_10000 to MC_40000, a modified approach is adopted where the initial rank is set at $\log(m)$. Regarding the timing for variable splitting, we switch to Phase II upon meeting a switching tolerance of 1e-3 in the majority of instances tested. However, for all MaxCut problems (all problems in Section 5.3 and problems identified by starting with 'mcp' in Section 5.5), we apply a more lenient switching tolerance of 1e-2. Our code is capable of automatically selecting the phase-switching tolerance by analyzing the structural characteristics of the problem. The error used as the indicator for terminating Phase I is primal infeasibility, consistent with the final termination indicator outlined in Section 5.2. Beyond these specific settings, our investigation suggests that the algorithm's effectiveness relies on applying certain techniques and implementation strategies. In the ensuing subsection, we delve into these aspects in detail.

The control of penalty factor. The efficacy of ADMM is significantly influenced by the choice of the penalty parameter, ρ. A well-known adaptive method for selecting this parameter is the residual balancing technique, introduced by [HYW00, BPC+11], whose idea is to balance primal and dual residuals. However, our experiments indicate that this method does not consistently provide satisfactory performance, primarily due to the challenge of selecting appropriate values for the related hyper-parameters across different problem instances. To avoid the need for problem-specific manual adjustment of the penalty parameter, we use a simple heuristic strategy that incrementally increases the penalty parameter within our approach. Specifically, within LRADMM, we increase ρ by 1.2 times every five iterations until it reaches a predetermined maximum value, $\rho_{\max} = 5,000$. The initial value of ρ for LRADMM inherits from the warm start phase. In the warm-start phase, the update strategy for ρ follows from [BC06].

The heuristic factor. The appropriate value of ρ for ADMM and ALM may differ when transitioning between phases. To address this, we introduce a heuristic factor, h, to adjust ρ appropriately during phase transitions. When switching from LRALM to LRADMM, we multiply ρ by h instead of carrying it over directly. In most cases, directly inheriting ρ from the warm-start phase exhibits sufficiently robust performance. However, for certain types of problems, selecting an appropriate heuristic factor h can significantly improve the performance. In subsequent experiments, we applied the heuristic factor as follows: $h = 10$ for all MaxCut, MaxCut MinBisection, graph partition, and quadratic assignment problems[3]; in matrix completion problems, we set $h = 5$ for MC_1000 to MC_8000, and $h = 2.5$ for MC_10000 to MC_40000; for all other problems, we set $h = 1$ to indicate no use of the heuristic factor.

The implementation of LRALM. To improve the performance of the LRALM during the warm-start phase, we incorporate enhancements suggested in [BC06]. However, we do not apply the rank-updating method introduced in their work. Our implementation generally follows that of SDPLR v1.03[4].

[3]The problems are recognized by: "G" and "mcp" for MaxCut; G40_mb and G48_mb for MaxCut MinBisection; "gpp" for graph partition; and "qap" for quadratic assignment problems.

[4]For the source code, see `https://yalmip.github.io/solver/sdplr/`. For a brief guide, see `https://sburer.github.io/files/SDPLR-1.03-beta-usrguide.pdf`.

5.2 Experiment Setting

We compare LoRADS with the following widely used SDP solvers:

- SDPLR [BC06]. An open-source solver that utilizes the Burer-Monteiro (BM) low-rank factorization method.
- SDPNAL+ [STYZ19]. An open-source solver employing ALM, designed for tackling large-scale SDP problems.
- COPT [GHW+22]. A commercial solver based on the interior point method. COPT is currently recognized as the top-performing solver in Hans Mittelmann's benchmarks for SDP problems[5]. This solver is included as a benchmark for the interior point method.

Given the diverse termination settings employed by different solvers—some of which are deeply integrated into their code—we have tried aligning their termination criteria while recognizing inherent differences. These solvers terminate when certain error metrics fall below a specified tolerance ϵ:

- LoRADS and SDPLR terminate when $\frac{\|\mathcal{A}(X)-b\|_2}{1+\|b\|_\infty} \leq \epsilon$.
- Denote the slack variable of the dual of (1) by S and the metric projection of $(X-S)$ onto S^n_+ by $\Pi_{S^n_+}(X-S)$, SDPNAL+ terminates when $\max\left\{\frac{\|\mathcal{A}(X)-b\|}{1+\|b\|}, \frac{\|\mathcal{A}^*(y)+S-C\|}{1+\|C\|}, \frac{1}{5}\frac{\left\|X-\Pi_{S^n_+}(X-S)\right\|}{1+\|X\|+\|S\|}\right\} \leq \epsilon$.
- COPT terminates when $\max\left\{\frac{\|\mathcal{A}(X)-b\|_2}{1+\|b\|_1}, \frac{|\min\{0,\sigma_{\min}(C-\mathcal{A}^*(\lambda))\}|}{1+\|c\|_1}, \frac{\langle C,X\rangle-\lambda^\top b}{1+|\langle C,X\rangle|+|\lambda^\top b|}\right\} < \epsilon$, where c is the flatten vector of matrix C.

We have chosen to align the termination criterion of LoRADS with that of SDPLR rather than directly employing the condition (13). This decision is informed by the observation that the second term in (13), $\frac{\gamma\|V^{k+1}-V^k\|_F}{1+\|V^k\|F}$, typically meets the required standards once the primal infeasibility gap is addressed. Notably, the primal infeasibility is measured as a value normalized using the infinity norm $\|b\|_\infty$, making this approach significantly more rigorous than utilizing either $\|b\|_2$ or $\|b\|_1$, particularly in scenarios involving a large number of constraints.

In the following experiments in this section, we set ϵ to be 1e-5, and the time limit to be 10,000 seconds[6]. The iteration cap for the LRADMM algorithm is set at 5,000. All the experiments are performed on a Macbook Pro with a 12 Core M3 Pro chip and 18 GB RAM. For clarity, this section only reports the results of solving times for each experiment. A report of properly defined and aligned solving errors for all solvers in all the experiments can be found in Appendix B.

5.3 The MaxCut Problems

Given a weighted graph G with weights w, the MaxCut problem aims to partition the vertices into two disjoint subsets such that the total weight of the edges connecting vertices across the subsets is maximized. The SDP formulation of the MaxCut problem can be expressed as follows:

$$\max_{X\in\mathbb{S}^{n\times n}} \ \left\langle \frac{1}{4}L(G,w), X\right\rangle$$

[5] `https://plato.asu.edu/ftp/sparse_sdp.html`.

[6] The instances that failed to be solved in 10,000 seconds are marked with "t"

$$\begin{aligned} \text{s.t.} \quad & X_{ii} = 1 \quad , \forall i \in [n] \\ & X \succeq 0, \end{aligned}$$

where the Laplacian matrix $L(G, w)$ is defined by

$$L(G,w)_{ij} := \begin{cases} -w_{ij} & \text{if } (i,j) \in E \\ \sum_k w_{ik} & \text{if } i = j \\ 0 & \text{o.w.} \end{cases}$$

We evaluated all previously discussed solvers using the Gset benchmark[7] [Ye03], and report the test results in Table 4. Our findings suggest that SDPNAL+ struggles with this specific set of SDP challenges, and LoRADS outperforms the other solvers in almost all cases. For problems with $n, m < 5000$, LoRADS is approximately 1 to 2 times faster than both SDPLR and COPT. However, for larger instances where $n, m \geq 5000$, the performance gap widens significantly, with LoRADS being 10 to 20 times faster than SDPLR and COPT. This trend highlights that the advantages of LoRADS become more pronounced in larger-scale problems.

5.4 The Matrix Completion Problem

The Matrix Completion Problem, which involves estimating the unobserved entries of a partially observed matrix, finds wide application in areas such as collaborative filtering, image inpainting, and system identification, among others. SDP is a well-known method for addressing matrix completion problems. However, the use of conventional SDP solvers in matrix completion, particularly in large-scale instances, is hindered by computational inefficiencies [LHSZ19]. Despite these limitations, recent progress in SDP algorithms highlighted by [WH23], has demonstrated significant improvements over traditional SDP solvers. Additionally, recent studies by [YMLS23] have shown that the Burer-Monteiro factorization method is particularly advantageous in scenarios with limited observations and a small number of measurements. Motivated by these efforts, we test our method on matrix completion problems and demonstrate its effectiveness.

Given a matrix $M \in \mathbb{R}^{n \times m}$ with observed entries in $\Omega \subseteq \{1, \ldots, n\} \times \{1, \ldots, m\}$, the problem is to find Y such that $Y_{ij} = M_{ij}$ for any $(i, j) \in \Omega$, while $\text{rank}(Y)$ being minimized. The convex optimization problem can be formulated as follows:

$$\min_{Y \in \mathbb{R}^{m \times n}} \quad \|Y\|_* \quad \text{s.t.} \quad Y_{ij} = M_{ij}, \quad \forall (i,j) \in \Omega,$$

where $\|Y\|_*$ is the nuclear norm of Y. Further, the nuclear norm minimization problem above can be formulated into a semidefinite programming problem as follows:

$$\min_{Y \in \mathbb{R}^{m \times n}} \quad \text{trace}\big(W_1\big) + \text{trace}\big(W_2\big) \quad \text{s.t.} \quad Y_{ij} = M_{ij}, \quad \forall (i,j) \in \Omega, \begin{bmatrix} W_1 & Y^\top \\ Y & W_2 \end{bmatrix} \succeq 0,$$

which is equivalent to

$$\min_{X \in \mathbb{S}^{(m+n)\times(m+n)}} \quad \big\langle I, X \big\rangle, \quad \text{s.t.} \quad \left\langle \begin{bmatrix} 0_{m\times m} & E_{ij}^\top \\ E_{ij} & 0_{n\times n} \end{bmatrix}, X \right\rangle = 2M_{ij}, \quad \forall (i,j) \in \Omega, X \succeq 0,$$

where $X = \begin{bmatrix} W_1 & Y^\top \\ Y & W_2 \end{bmatrix}$, and E_{ij} is the ij-th adjacency matrix with $(E_{ij})_{ij} = 1$ and 0 elsewhere.

[7] For problem files, see `https://web.stanford.edu/~yyye/yyye/Gset/`

Table 4: Solving Times on Max-Cut Problems

Problem	n, m	LoRADS	SDPLR	SDPNAL+	COPT	Problem	n, m	LoRADS	SDPLR	SDPNAL+	COPT
G1	800	**0.18**	0.75	33.43	0.82	G35	2000	**0.23**	1.60	723	3.49
G2	800	**0.17**	0.68	36.49	0.78	G36	2000	**0.24**	1.41	828	2.99
G3	800	**0.18**	0.81	31.69	0.90	G37	2000	**0.25**	1.84	381	3.52
G4	800	**0.17**	0.69	29.59	0.81	G38	2000	**0.25**	1.85	414	3.25
G5	800	**0.15**	0.66	25.41	0.72	G39	2000	**0.38**	1.76	352	3.29
G6	800	**0.23**	0.67	30.18	0.61	G40	2000	**0.57**	2.82	428	3.54
G7	800	**0.22**	0.60	29.54	0.56	G41	2000	**0.52**	1.99	444	3.20
G8	800	**0.20**	0.69	34.49	0.61	G42	2000	**0.54**	1.67	360	3.19
G9	800	**0.20**	0.54	28.22	0.54	G43	1000	**0.09**	0.44	18.20	0.97
G10	800	**0.20**	0.64	36.14	0.81	G44	1000	**0.11**	0.37	18.17	0.74
G11	800	**0.05**	0.47	49.13	0.25	G45	1000	**0.10**	0.40	53.57	1.00
G12	800	**0.05**	0.34	58.34	0.26	G46	1000	**0.10**	0.56	50.16	0.85
G13	800	**0.05**	0.17	58.37	0.28	G47	1000	**0.10**	0.43	55.18	1.10
G14	800	**0.07**	0.28	45.07	0.50	G48	3000	**0.12**	0.67	1436	2.09
G15	800	**0.07**	0.34	55.99	0.59	G49	3000	**0.14**	0.76	462	2.00
G16	800	**0.07**	0.33	56.73	0.58	G50	3000	**0.13**	0.96	1433	1.88
G17	800	**0.06**	0.34	39.46	0.53	G51	1000	**0.09**	0.45	51.64	0.83
G18	800	**0.12**	0.37	30.17	0.37	G52	1000	**0.09**	0.38	58.41	0.88
G19	800	**0.12**	0.41	33.84	0.54	G53	1000	**0.09**	0.43	19.48	0.75
G20	800	**0.12**	0.32	34.12	0.48	G54	1000	**0.08**	0.48	49.06	0.66
G21	800	**0.11**	0.35	27.20	0.50	G55	5000	**0.38**	3.45	5587	18.04
G22	2000	**0.28**	1.66	277	3.42	G56	5000	**0.42**	4.03	4664	18.18
G23	2000	**0.31**	1.31	483	3.67	G57	5000	**0.54**	6.55	7123	11.34
G24	2000	**0.27**	1.91	274	3.50	G58	5000	**0.87**	14.76	8426	24.52
G25	2000	**0.27**	1.26	276	3.26	G59	5000	**2.42**	26.48	7318	25.16
G26	2000	**0.27**	1.72	286	3.76	G60	7000	**0.66**	7.66	t	40.20
G27	2000	**0.40**	1.66	525	3.49	G61	7000	**0.86**	6.69	t	41.16
G28	2000	**0.50**	1.99	418	4.09	G62	7000	**1.02**	16.25	t	23.59
G29	2000	**0.33**	2.37	272	4.27	G63	7000	**1.92**	23.69	t	59.32
G30	2000	**0.31**	1.37	553	4.54	G64	7000	**5.60**	61.82	t	62.78
G31	2000	**0.41**	2.03	321	3.86	G65	8000	**1.30**	24.80	t	33.35
G32	2000	**0.18**	1.36	645	1.50	G66	9000	**1.44**	29.39	t	43.09
G33	2000	**0.17**	1.21	475	1.44	G67	10000	**1.58**	42.73	t	57.18
G34	2000	**0.17**	1.17	579	1.61						

We benchmark our algorithm on a set of randomly generated matrix completion problems, adopting the generating process described by [WH23]. The results, as depicted in Table 5, demonstrate that LoRADS consistently outperforms other solvers across a range of instance sizes. For instances larger than MC_10000, further evaluations of the compared solvers were not conducted due to time and computation resource limits. Notably, LoRADS successfully solves an exceptionally large-scale instance with $15,694,167$ constraints and a 40000×40000 matrix variable in less than 6 minutes.

Table 5: Solving Times on Matrix Completion Problems

Problem	n	m	LoRADS	SDPLR	SDPNAL+	COPT
MC_1000	1000	199424	**1.41**	36.96	6.21	14.84
MC_2000	2000	550536	**4.68**	220	60.65	106
MC_3000	3000	930328	**10.09**	819	343	416
MC_4000	4000	1318563	**18.78**	2401	326	1239
MC_5000	5000	1711980	**24.49**	4547	636	2502
MC_6000	6000	2107303	**32.15**	6390	2347	5006
MC_8000	8000	2900179	**57.21**	t	2921	t
MC_10000	10000	3695929	**27.33**	t	5879	t
MC_12000	12000	4493420	**49.40**	-	-	-
MC_14000	14000	5291481	**42.62**	-	-	-
MC_16000	16000	6089963	**74.66**	-	-	-
MC_18000	18000	6889768	**71.60**	-	-	-
MC_20000	20000	7688309	**83.83**	-	-	-
MC_40000	40000	15684167	**351**	-	-	-

[1] For instances beyond MC_10000, we only evaluate LoRADS and use "-" for other solvers as placeholders.

5.5 General SDP Benchmark Problems

In this section, we evaluate our method across a wider range of SDP problems from SDPLIB[8] and Hans Mittelmann's SDP benchmarks[9]. SDPLIB offers relatively small-scale SDP instances, while Hans Mittelmann's SDP benchmarks provide larger and more challenging SDP instances.

The comparative performance of the four solvers is detailed in Table 6. Our method exhibits considerable improvements over SDPLR across most tested instances. In comparison with SDPNAL+, our solver falls short in only a few cases. Against COPT, our solver shows competitive performance, surpassing COPT in several instances, albeit falling short in others. We need to remark that the results of LoRADS are based on a very elementary parameter tuning detailed in Section 5.1. For example, the problem "hand" can be solved in 5.81 seconds if a more suitable ρ_{max} and heuristic factor h is applied, potentially reducing the solving time by a factor of 20.

To further benchmark the performance of LoRADS, we employ the scaled shifted geometric means (SSGM) to analyze solving times for problems sourced from both SDPLIB and Hans Mittelmann's benchmarks respectively. The SSGMs calculated for SDPLIB illustrate LoRADS's proficiency in solving small-scale SDP problems, whereas the SSGMs for Hans Mittelmann's benchmarks assess the efficacy of LoRADS in addressing larger SDP challenges. For an observation $y \in \mathbb{R}^n_+$, the shifted geometric mean is calculated as follows:

$$SGM = \exp\left[\sum_{i=1}^{n} \frac{1}{n} \log(\max\{1, y_i + s\})\right] - s,$$

where s represents the shift parameter, set to 10 in this case. To compare the solving times of different solvers, we first calculate the shifted geometric mean for each solver's solving time. Then, we derive the scaled shifted geometric mean by normalizing these values against the smallest one. The SSGMs on SDPLIB and Hans Mittelmann's SDP benchmarks are shown in Table 7.

In the evaluation of the solvers' performance on the selected problems from the SDPLIB using SSGMs, COPT emerges as the top performer, closely followed by LoRADS. Compared to SDPLR,

[8] For the problem files and a detailed description, see `http://euler.nmt.edu/~brian/sdplib/`.

[9] For the problem files and a detailed description, see `https://plato.asu.edu/ftp/sparse_sdp.html`.

Table 6: Solving Times on Problems from SDPLIB and Mittelmann's Set

Problem	n	m	LoRADS	SDPLR	SDPNAL+	COPT	Problem	n	m	LoRADS	SDPLR	SDPNAL+	COPT
AlH*	5991	7230	**63.21**	2491	943	1208	mcp250-1*	250	250	**0.01**	**0.01**	7.57	0.86
BH2*	2167	1743	**7.80**	124	48.31	35.65	mcp250-2*	250	250	**0.01**	**0.01**	5.42	0.88
cancer_100*	570	10470	513.88	t	728	**83.36**	mcp250-3*	250	250	**0.01**	0.02	5.49	0.79
CH2*	2167	1743	**6.74**	51.46	88.84	19.06	mcp250-4*	250	250	**0.02**	0.04	5.25	0.90
checker_1.5*	3971	3971	**0.95**	40.45	6201	20.3	mcp500-1*	500	500	**0.01**	0.03	21.36	0.20
G40_mb*	2001	2001	19.89	59.95	4645	**12.00**	mcp500-2*	500	500	**0.02**	0.04	14.59	0.20
G48_mb*	3001	3001	**17.21**	25.45	3019	17.50	mcp500-3*	500	500	**0.03**	0.07	21.87	0.29
gpp100*	100	101	**0.05**	0.14	3.76	0.14	mcp500-4*	500	500	**0.04**	0.11	23.22	0.40
gpp124-1*	124	125	**0.04**	0.26	6.14	0.24	NH2*	2047	1743	24.77	19.03	**10.81**	18.37
gpp124-2*	124	125	**0.07**	0.11	4.99	0.23	NH3*	3163	2964	**9.57**	177	55.38	63.81
gpp124-3*	124	125	**0.06**	0.10	7.42	0.26	p_auss2_3.0*	9116	9115	**4.42**	83.95	t	289
gpp124-4*	124	125	**0.11**	0.32	4.57	0.23	qap5*	26	136	0.18	0.09	0.48	**0.01**
gpp250-1*	250	250	**0.15**	0.71	27.93	1.79	qap6*	37	229	0.63	1.90	1.69	**0.03**
gpp250-2*	250	250	**0.14**	0.49	14.52	1.63	qap7*	50	358	1.02	2.18	1.65	**0.06**
gpp250-3*	250	250	**0.16**	1.59	16.46	2.03	qap8*	65	529	1.75	4.33	8.89	**0.15**
gpp250-4*	250	250	**0.20**	0.61	13.94	1.84	qap9*	82	748	2.50	4.89	5.34	**0.26**
gpp500-1*	500	501	**0.71**	1.84	31.15	1.08	qap10*	101	1021	2.97	10.19	6.14	**0.43**
gpp500-2*	500	501	**0.72**	1.02	106	0.76	sensor_500*	4029	3540	94.13	73.41	121	**12.3**
gpp500-3*	500	501	**0.62**	2.64	35.98	0.67	shmup4*	800	4962	**53.59**	70.77	3068	115
gpp500-4*	500	501	**0.65**	1.30	36.37	0.76	theta1*	50	104	0.22	0.32	0.32	**0.02**
H3O*	3163	2964	**10.12**	199	295	101	theta2*	100	498	0.57	0.72	1.65	**0.10**
hand*	1297	1297	116	**24.22**	628	92.76	theta3*	150	1106	1.49	2.29	1.57	**0.30**
ice_2.0*	8114	8113	**4.91**	41.97	t	269	theta4*	200	1949	3.09	6.01	3.74	**0.82**
mcp100*	100	100	**<0.005**	**<0.005**	1.19	0.07	theta5*	250	3028	4.67	9.04	3.05	**1.89**
mcp124-1*	124	124	**<0.005**	0.01	2.05	0.12	theta6*	300	4375	6.77	19.49	4.22	**3.90**
mcp124-2*	124	124	**<0.005**	**<0.005**	1.90	0.13	theta12*	601	17979	86.23	127	11.69	**9.85**
mcp124-3*	124	124	**0.01**	**0.01**	1.83	0.12	theta102*	501	37467	71.44	141	6.36	**4.57**
mcp124-4*	124	124	**0.01**	**0.01**	1.56	0.13	theta123*	601	90020	206	369	10.60	**8.92**

[1] Problems from SDPLIB are marked with "*", and problems from Hans Mittelmann's SDP benchmarks are marked with "*"

Table 7: SSGMs of Different Solvers

Problem Set	LoRADS	SDPLR	SDPNAL+	COPT
SDPLIB	1.13	2.46	13.59	1
Mittelmann's Benchmarks	1	3.68	10.81	1.44

LoRADS demonstrates approximately 1.5 times the speed and outpaces SDPNAL+ by a factor of about 9. For selected problems from Hans Mittelmann's SDP Benchmarks, LoRADS outperforms all competitors, achieving a significant speed advantage over SDPLR by a factor of 3.68 and surpassing SDPNAL+ by 10.81 times. LoRADS also marginally exceeds COPT by 1.44 times. These observations suggest that LoRADS has a more pronounced advantage in handling larger-scale SDP problems.

6 Conclusion

In this paper, we introduced a new first-order method for solving semidefinite programming (SDP) problems, based on an ADMM framework with a matrix-splitting technique. Our approach improves upon traditional methods, such as the Burer-Monteiro factorization, by involving quadratically regularized subproblems that are easier to solve and better conditioned. For linear objectives, these subproblems reduce to well-conditioned quadratic programs, which can be efficiently handled

using conjugate gradient methods. We demonstrated that the ADMM algorithm achieves sublinear or linear convergence rates to the KKT solutions, depending on certain conditions like the satisfaction of the Lojasiewicz inequality.

Building on this theoretical foundation, we developed LoRADS, a novel solver for linear SDP based on a Low-Rank ADMM Splitting approach. LoRADS incorporates key strategies, including a warm-start phase leveraging the Burer-Monteiro method, and a dynamic rank selection mechanism motivated by low-rank SDP theory. Extensive numerical evaluations highlight LoRADS' efficiency and scalability, including its ability to solve a large matrix completion problem with over 15 million constraints and a matrix size of 40,000 $\times$ 40,000 in just 351 seconds.

Future improvements to LoRADS, such as adaptive step sizes, enhanced warm-start strategies, phase-switching automation, and additional heuristics, are expected to further boost its performance. Given its first-order nature, LoRADS is well-suited for GPU-based implementations, which could unlock its potential for handling even larger-scale SDP problems.

References

[AB09] Hedy Attouch and Jérôme Bolte. On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. *Mathematical Programming*, 116:5–16, 2009.

[ABB+99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[ABS13] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2013.

[Ali91] Farid Alizadeh. Combinatorial optimization with interior point methods and semi-definite matrices. *Ph. D. thesis, University of Minnesota*, 1991.

[Ali95] Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM journal on Optimization*, 5(1):13–51, 1995.

[Bar95] Alexander I. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete & Computational Geometry*, 13:189–202, 1995.

[BC06] Samuel Burer and Changhui Choi. Computational enhancements in low-rank semidefinite programming. *Optimization Methods and Software*, 21(3):493–512, 2006.

[BGP21] Stefania Bellavia, Jacek Gondzio, and Margherita Porcelli. A relaxed interior point method for low-rank semidefinite programming problems with applications to matrix completion. *Journal of Scientific Computing*, 89(2):46, 2021.

[BLWY06] Pratik Biswas, Tzu-Chen Lian, Ta-Chung Wang, and Yinyu Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Trans. Sen. Netw.*, 2(2):188–220, may 2006.

[BM03] Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[BM05] Samuel Burer and Renato DC Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical programming*, 103(3):427–444, 2005.

[Bou15] Nicolas Boumal. A riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints. *arXiv preprint arXiv:1506.00575*, 2015.

[BPC+11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 01 2011.

[BPP+02] L Susan Blackford, Antoine Petitet, Roldan Pozo, Karin Remington, R Clint Whaley, James Demmel, Jack Dongarra, Iain Duff, Sven Hammarling, Greg Henry, et al. An updated set of basic linear algebra subprograms (blas). *ACM Transactions on Mathematical Software*, 28(2):135–151, 2002.

[BV97] Stephen Boyd and Lieven Vandenberghe. Semidefinite programming relaxations of non-convex problems in control and combinatorial optimization. In *Communications, Computation, Control, and Signal Processing: a tribute to Thomas Kailath*, pages 279–287. Springer, 1997.

[BY04] Pratik Biswas and Yinyu Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, IPSN '04, page 46–54, New York, NY, USA, 2004. Association for Computing Machinery.

[CG23] Yuwen Chen and Paul Goulart. Burer-monteiro admm for large-scale sdps. *arXiv preprint arXiv:2302.04016*, 2023.

[EOPV22] Murat A Erdogdu, Asuman Ozdaglar, Pablo A Parrilo, and Nuri Denizcan Vanli. Convergence rate of block-coordinate maximization burer–monteiro method for solving large sdps. *Mathematical Programming*, 195(1-2):243–281, 2022.

[GGY22] Wenzhi Gao, Dongdong Ge, and Yinyu Ye. HDSDP: Software for semidefinite programming. *arXiv preprint arXiv:2207.13862*, 2022.

[GHW+22] Dongdong Ge, Qi Huangfu, Zizhuo Wang, Jian Wu, and Yinyu Ye. Cardinal optimizer (COPT) user guide, 2022.

[GW95] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[HYW00] Bing-Sheng He, Hai Yang, and SL Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and applications*, 106:337–356, 2000.

[JBAS10] Michel Journée, Francis Bach, P-A Absil, and Rodolphe Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

[LHSZ19] Xiao Peng Li, Lei Huang, Hing Cheung So, and Bo Zhao. A survey on matrix completion: Perspective of signal processing, 2019.

[LL11] Javad Lavaei and Steven H Low. Zero duality gap in optimal power flow problem. *IEEE Transactions on Power systems*, 27(1):92–107, 2011.

[LLM11] Guanghui Lan, Zhaosong Lu, and Renato DC Monteiro. Primal-dual first-order methods with iteration-complexity for cone programming. *Mathematical Programming*, 126(1):1–29, 2011.

[LMZ15] Tianyi Lin, Shiqian Ma, and Shuzhong Zhang. On the global linear convergence of the admm with multiblock variables. *SIAM Journal on Optimization*, 25(3):1478–1497, 2015.

[LZY23] Mingyu Lei, Jiayu Zhang, and Yinyu Ye. Blessing of high-order dimensionality: from non-convex to convex optimization for sensor network localization, 2023.

[NN94] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[OCPB16] Brendan O'donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169:1042–1068, 2016.

[OR00] James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[Pat98] Gábor Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of operations research*, 23(2):339–358, 1998.

[STYZ19] Defeng Sun, Kim-Chuan Toh, Yancheng Yuan, and Xin-Yuan Zhao. Sdpnal+: A matlab software for semidefinite programming with bound constraints (version 1.0), 2019.

[STZY12] Davood Shamsi, Nicole Taheri, Zhisu Zhu, and Yinyu Ye. On sensor network localization using sdp relaxation, 2012.

[SY07] Anthony Man-Cho So and Yinyu Ye. Theory of semidefinite programming for sensor network localization. *Mathematical Programming*, 109(2-3):367–384, 2007.

[SYZ08] Anthony Man-Cho So, Yinyu Ye, and Jiawei Zhang. A unified theorem on sdp rank reduction. *Mathematics of Operations Research*, 33(4):910–920, 2008.

[TT23] Tianyun Tang and Kim-Chuan Toh. A feasible method for general convex low-rank sdp problems. *arXiv preprint arXiv:2312.07908*, 2023.

[WDLW23] Yifei Wang, Kangkang Deng, Haoyang Liu, and Zaiwen Wen. A decomposition augmented lagrangian method for low-rank semidefinite programming. *SIAM Journal on Optimization*, 33(3):1361–1390, 2023.

[WGY10] Zaiwen Wen, Donald Goldfarb, and Wotao Yin. Alternating direction augmented lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, 2(3-4):203–230, 2010.

[WH23] Jie Wang and Liangbing Hu. Solving low-rank semidefinite programs via manifold optimization, 2023.

[WY13] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[WZYB08] Zizhuo Wang, Song Zheng, Yinyu Ye, and Stephen Boyd. Further relaxations of the semidefinite programming approach to sensor network localization. *SIAM Journal on Optimization*, 19(2):655–673, 2008.

[XW21] Yue Xie and Stephen J Wright. Complexity of proximal augmented lagrangian for nonconvex optimization with nonlinear equality constraints. *Journal of Scientific Computing*, 86:1–30, 2021.

[Ye03] Yinyu Ye. Gset. `https://web.stanford.edu/~yyye/yyye/Gset/`, 2003.

[YMLS23] Baturalp Yalçin, Ziye Ma, Javad Lavaei, and Somayeh Sojoudi. Semidefinite programming versus burer-monteiro factorization for matrix sensing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10702–10710, 2023.

[YST15] Liuqin Yang, Defeng Sun, and Kim-Chuan Toh. Sdpnal+: a majorized semismooth newton-cg augmented lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.

[YTF+21] Alp Yurtsever, Joel A Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021.

[YUTC17] Alp Yurtsever, Madeleine Udell, Joel Tropp, and Volkan Cevher. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *Artificial intelligence and statistics*, pages 1188–1196. PMLR, 2017.

[ZST10] Xin-Yuan Zhao, Defeng Sun, and Kim-Chuan Toh. A newton-cg augmented lagrangian method for semidefinite programming. *SIAM Journal on Optimization*, 20(4):1737–1765, 2010.

A Complete Theoretical Analysis

A.1 Proof of Theorem 1

Proof. Proof For notation simplicity, we will drop the superscript σ in the constants L_f^σ, L^σ and L_0^σ in the proof. Firstly, using the ϵ-KKT condition (8) of (4), we have

$$\begin{aligned}\gamma\|\overline{U}-\overline{V}\|_F \leq & \epsilon + \left\|\nabla f(\overline{V}\overline{U}^\top)\overline{U} + \sum_{i=1}^m \overline{\lambda}_i A_i \overline{U}\right\|_F \\ \leq & \epsilon + L_f\|\overline{U}\|_F + \sqrt{s_A}\|\overline{\lambda}\|_2 \cdot \|\overline{U}\|_F \\ \leq & \epsilon + L_f\sigma + \sqrt{s_A}\sigma^2\end{aligned} \tag{24}$$

where the inequalities holds because $(\overline{U},\overline{V},\overline{\lambda})$ lies in the bounded set $\mathcal{B}_\sigma\big(0^{n\times r}, 0^{n\times r}, 0^m\big)$, $\|\nabla f\|_F$ is bounded above by L_f on $B_{\sigma^2}\big(0^{n\times n}\big)$ and

$$\left\|\sum_{i=1}^m \overline{\lambda}_i A_i \overline{U}\right\|_F \leq \sum_{i=1}^m |\overline{\lambda}_i| \cdot \|A_i\|_2 \cdot \|\overline{U}\|_F \leq \sqrt{s_A}\|\overline{\lambda}\|_2 \cdot \|\overline{U}\|_F$$

where the first inequality holds because of triangle inequality and the second inequality because of the Cauchy–Schwarz inequality. Next, adding up the first two equations in (8) and using the triangle inequality yield

$$\left\|\nabla f(\overline{U}\overline{V}^\top)\overline{V} + \nabla f(\overline{V}\overline{U}^\top)\overline{U} + 2\sum_{i=1}^m \overline{\lambda}_i A_i \widehat{U}\right\|_F \leq 2\epsilon \tag{25}$$

Moreover, it is easily verified that

$$\begin{aligned}& \nabla f(\overline{U}\overline{V}^\top)\overline{V} + \nabla f(\overline{V}\overline{U}^\top)\overline{U} - 2\nabla f(\widehat{U}\widehat{U}^\top)\widehat{U} \\ = & \left(\nabla f(\overline{U}\overline{V}^\top) + \nabla f(\overline{V}\overline{U}^\top) - 2\nabla f\big(\frac{\overline{U}\overline{V}^\top + \overline{V}\overline{U}^\top}{2}\big)\right)\widehat{U} \\ & + 2\left(\nabla f\big(\frac{\overline{U}\overline{V}^\top + \overline{V}\overline{U}^\top}{2}\big) - \nabla f(\widehat{U}\widehat{U}^\top)\right)\widehat{U} \\ & + \nabla f(\overline{U}\overline{V}^\top)(\overline{V}-\widehat{U}) + \nabla f(\overline{V}\overline{U}^\top)(\overline{U}-\widehat{U})\end{aligned} \tag{26}$$

Using [OR00, Proposition 3.2.12] and the Lipschitz continuity of $\nabla^2 f$, we know

$$\|\nabla f(X) - \nabla f(Y) - \nabla^2 f(Y)(X-Y)\|_F \leq \frac{L_0}{2}\|X-Y\|_F^2$$

for $X, Y \in \mathcal{B}_{\sigma^2}\big(0^{n\times n}\big)$. Then, by applying this inequality to the pairs $\Big(\overline{U}\overline{V}^\top, \frac{\overline{U}\overline{V}^\top + \overline{V}\overline{U}^\top}{2}\Big)$ and $\Big(\overline{V}\overline{U}^\top, \frac{\overline{U}\overline{V}^\top + \overline{V}\overline{U}^\top}{2}\Big)$, and then summing up these resulting inequalities, one can easily show that

$$\begin{aligned}& \left\|\nabla f(\overline{U}\overline{V}^\top) + \nabla f(\overline{V}\overline{U}^\top) - 2\nabla f\left(\frac{\overline{U}\overline{V}^\top + \overline{V}\overline{U}^\top}{2}\right)\right\|_F \\ \leq & \frac{L_0}{4}\|\overline{U}\overline{V}^\top - \overline{V}\overline{U}^\top\|_F^2 \\ = & \frac{L_0}{4}\|\overline{U}(\overline{V}^\top - \overline{U}^\top) + (\overline{U}-\overline{V})\overline{U}^\top\|_F^2 \\ \leq & L_0\sigma^2\|\overline{U}-\overline{V}\|_F^2\end{aligned} \tag{27}$$

where the last inequality holds because $\overline{U}$ and $\overline{V}$ are bounded by σ. Besides, since ∇f is Lipschitz continuous, we have

$$\begin{aligned}\left\|\nabla f\left(\frac{\overline{U}\overline{V}^\top+\overline{V}\overline{U}^\top}{2}\right)-\nabla f(\widehat{U}\widehat{U}^\top)\right\|_F &\le L\left\|\frac{\overline{U}\overline{V}^\top+\overline{V}\overline{U}^\top}{2}-\widehat{U}\widehat{U}^\top\right\|_F \\ &= \frac{L}{4}\|(\overline{U}-\overline{V})(\overline{U}-\overline{V})^\top\|_F \\ &\le \frac{L}{4}\|\overline{U}-\overline{V}\|_F^2\end{aligned} \tag{28}$$

For the third term in (26), we can derive that

$$\nabla f(\overline{U}\overline{V}^\top)(\overline{V}-\widehat{U})+\nabla f(\overline{V}\overline{U}^\top)(\overline{U}-\widehat{U})=\frac{1}{2}\left(\nabla f(\overline{U}\overline{V}^\top)-\nabla f(\overline{V}\overline{U}^\top)\right)(\overline{V}-\overline{U})$$

which implies that

$$\begin{aligned}\|\nabla f(\overline{U}\overline{V}^\top)(\overline{V}-\widehat{U})+\nabla f(\overline{V}\overline{U}^\top)(\overline{U}-\widehat{U})\|_F &\le \frac{1}{2}\|\nabla f(\overline{U}\overline{V}^\top)-\nabla f(\overline{V}\overline{U}^\top)\|_F\|\overline{V}-\overline{U}\|_F \\ &\le \frac{L}{2}\|\overline{U}\overline{V}^\top-\overline{V}\overline{U}^\top\|_F\|\overline{V}-\overline{U}\|_F \\ &\le L\sigma\|\overline{V}-\overline{U}\|_F^2\end{aligned} \tag{29}$$

Substituting (27), (28) and (29) into (26), we get

$$\|\nabla f(\overline{U}\overline{V}^\top)\overline{V}+\nabla f(\overline{V}\overline{U}^\top)\overline{U}-2\nabla f(\widehat{U}\widehat{U}^\top)\widehat{U}\|_F\le(L_0\sigma^3+\frac{3L\sigma}{2})\|\overline{U}-\overline{V}\|_F^2,$$

which together with (24) and (25) yields

$$\left\|2\nabla f(\widehat{U}\widehat{U}^\top)\widehat{U}+2\sum_{i=1}^m\overline{\lambda}_iA_i\widehat{U}\right\|_F\le 2\epsilon+\frac{(2L_0\sigma^3+3L\sigma)(\epsilon+L_f\sigma+\sqrt{s_A}\sigma^2)^2}{2\gamma^2}.$$

Similarly, since $\|\mathcal{A}(\overline{U}\overline{V}^\top)-b\|_2=\|\mathcal{A}(\overline{V}\overline{U}^\top)-b\|_2\le\epsilon$ and thus it holds

$$\begin{aligned}\left\|\mathcal{A}(\widehat{U}\widehat{U}^\top)-b\right\|_2 &\le \epsilon+\frac{1}{2}\left\|2\mathcal{A}(\widehat{U}\widehat{U}^\top)-\mathcal{A}(\overline{U}\overline{V}^\top)-\mathcal{A}(\overline{V}\overline{U}^\top)\right\|_2 \\ &\le \epsilon+\frac{\|\mathcal{A}\|_2}{2}\cdot\left\|2\widehat{U}\widehat{U}^\top-\overline{U}\overline{V}^\top-\overline{V}\overline{U}^\top\right\|_F \\ &\le \epsilon+\frac{\|\mathcal{A}\|_2(\epsilon+L_f\sigma+\sqrt{s_A}\sigma^2)^2}{4\gamma^2}.\end{aligned}$$

□

A.2 Proof of Proposition 1

Proof. Proof Note that

$$\begin{aligned}
&\mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^{k+1}) - \mathcal{L}_\rho(U^k, V^k, \lambda^k) \\
=&\underbrace{\mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^{k+1}) - \mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^k)}_{\text{(I)}} \\
&+ \underbrace{\mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^k) - \mathcal{L}_\rho(U^{k+1}, V^k, \lambda^k)}_{\text{(II)}} \\
&+ \underbrace{\mathcal{L}_\rho(U^{k+1}, V^k, \lambda^k) - \mathcal{L}_\rho(U^k, V^k, \lambda^k)}_{\text{(III)}}
\end{aligned}$$

Now, we bound term (I) − (III) separately,

$$\begin{aligned}
\text{(I)} =&\mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^{k+1}) - \mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^k) \\
=&(\lambda^{k+1} - \lambda^k)^\top \big(\mathcal{A}(U^{k+1}(V^{k+1})^\top) - b\big) \\
=&\rho\|\mathcal{A}(U^{k+1}(V^{k+1})^\top) - b\|_2^2 \\
=&\frac{1}{\rho}\|\lambda^{k+1} - \lambda^k\|_2^2
\end{aligned}$$

Because of the strongly convexity of subproblems, we have

$$\mathcal{L}_\rho(U^{k+1}, V^k, \lambda^k) + \frac{\gamma}{2}\|U^{k+1} - U^k\|_F^2 \le \mathcal{L}_\rho(U^k, V^k, \lambda^k),$$

and

$$\mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^k) + \frac{\gamma}{2}\|V^{k+1} - V^k\|_F^2 \le \mathcal{L}_\rho(U^{k+1}, V^k, \lambda^k)$$

We can then bound term (II) and (III)

$$\begin{aligned}
\text{(II)} =&\mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^k) - \mathcal{L}_\rho(U^{k+1}, V^k, \lambda^k) \\
\le& -\frac{\gamma}{2}\|V^{k+1} - V^k\|_F^2
\end{aligned}$$

Similarly, we have

$$\text{(III)} \le -\frac{\gamma}{2}\|U^{k+1} - U^k\|_F^2.$$

By combining the above three inequalities we complete the proof. □

A.3 Proof of Proposition 2

Proof. Proof For any given $\beta > 0$, S_β is closed because f is a smooth convex function. Suppose S_β is not empty.
(i) Let μ be the strongly convex modulus of $f(X) + \frac{\rho_0}{2}\|\mathcal{A}(X) - b\|_2^2$ and X^* be its minimum point. It follows from the strongly convexity that

$$f(UV^\top) + \frac{\rho_0}{2}\|\mathcal{A}(UV^\top) - b\|_2^2 \ge f(X^*) + \frac{\rho_0}{2}\|\mathcal{A}(X^*) - b\|_2^2 + \frac{\mu}{2}\|UV^\top - X^*\|_F^2$$

which implies $\{UV^\top \,|\, (U,V) \in S_\beta\}$ is bounded. On the other hand, $\|U - V\|_F$ is bounded for any $(U,V) \in S_\beta$. Note that

$$\|U\|_F^2 + \|V\|_F^2 = \|U - V\|_F^2 + 2\big\langle U, V \big\rangle$$

We can deduce that S_β is bounded. This completes the proof of the first part.
(ii) Given the strictly feasible solution λ^*, we have

$$\big\langle C, UV^\top \big\rangle = \big\langle C - \sum_{i=1}^m y_i^* A_i, UV^\top \big\rangle + \sum_{i=1}^m y_i^* \big\langle A_i, UV^\top \big\rangle = \big\langle C - \sum_{i=1}^m y_i^* A_i, UV^\top \big\rangle + \big\langle y^*, \mathcal{A}(UV^\top) \big\rangle. \tag{30}$$

For the first term in (30), we have

$$\begin{aligned}
\big\langle C - \sum_{i=1}^m y_i^* A_i, UV^\top \big\rangle =& \big\langle C - \sum_{i=1}^m y_i^* A_i, UU^\top \big\rangle + \big\langle C - \sum_{i=1}^m y_i^* A_i, U(V-U)^\top \big\rangle \\
\geq & \lambda_{\min} \|U\|_F^2 - \big\langle (C - \sum_{i=1}^m y_i^* A_i) U, V - U \big\rangle \\
\geq & \lambda_{\min} \|U\|_F^2 - \left\| \left(C - \sum_{i=1}^m y_i^* A_i \right) U \right\|_F \|V - U\|_F \\
\geq & \lambda_{\min} \|U\|_F^2 - \lambda_{\max} \|U\|_F \|V - U\|_F \\
\geq & \frac{\lambda_{\min}}{2} \|U\|_F^2 - \frac{\lambda_{\max} \kappa_c}{2} \|V - U\|_F^2,
\end{aligned}$$

where the second inequality holds because the fact $\big\langle B_1, B_2 \big\rangle \geq -\|B_1\|_F \|B_2\|_F$ for any B_1, B_2, the third inequality holds because of $\|B_1 B_2\|_F \leq \|B_1\|_F \|B_2\|_F$, and the last inequlity holds because $\lambda_{\max} \|U\|_F \|V - U\|_F \leq \frac{\lambda_{\min}}{2} \|U\|_F^2 + \frac{\lambda_{\max}^2}{2\lambda_{\min}} \|V - U\|_F^2$ and $\kappa_c = \frac{\lambda_{\max}}{\lambda_{\min}}$. For the second term in (30), we have

$$\big\langle y^*, \mathcal{A}(UV^\top) \big\rangle = \big\langle y^*, \mathcal{A}(UV^\top) - b \big\rangle + \big\langle y^*, b \big\rangle \geq -\frac{1}{2} \|y^*\|_2^2 - \frac{1}{2} \|\mathcal{A}(UV^\top) - b\|_2^2 + \big\langle y^*, b \big\rangle.$$

By choosing $\rho_0 \geq 1 + \lambda_{\max} \kappa_c$, we have

$$\begin{aligned}
& f(UV^\top) + \frac{\rho_0}{2} \|U - V\|_F^2 + \frac{\rho_0}{2} \|\mathcal{A}(UV^\top) - b\|_2^2 \\
\geq & \frac{\lambda_{\min}}{2} \|U\|_F^2 + \left(\frac{\rho_0}{2} - \frac{\lambda_{\max} \kappa_c}{2} \right) \|V - U\|_F^2 + \left(\frac{\rho_0}{2} - \frac{1}{2} \right) \|\mathcal{A}(UV^\top) - b\|_2^2 - \frac{1}{2} \|y^*\|_2^2 + \big\langle y^*, b \big\rangle \\
\geq & \frac{\lambda_{\min}}{2} \|U\|_F^2 - \frac{1}{2} \|y^*\|_2^2 + \big\langle y^*, b \big\rangle.
\end{aligned}$$

We can deduce from the above inequality that

$$\|U\|_F^2 \leq \frac{1}{\lambda_{\min}} \left(\|y^*\|_2^2 - 2\big\langle y^*, b \big\rangle + 2\beta \right). \tag{31}$$

By exchanging U and V, it is easy to show that (31) also holds for V. Hence, S_β is bounded and thus compact. □

A.4 Proof of Lemma 1

Proof. Proof We establish part (i) of this lemma by mathematical induction while proving part (ii)-(iv) automatically in the process. As (U^0, V^0, λ^0) is selected according to (19), part (i) naturally holds for $k = 0$. Now, assuming part (i) holds for k-th iterate, we proceed to establish its validity for $k + 1$. By this assumption and Proposition 1, we obtain

$$
\begin{aligned}
\widehat{\beta} &> L_\rho(U^k, V^k, \lambda^k) + \frac{1}{\rho}\|\lambda^k\|_2^2 \\
&\geq L_\rho(U^{k+1}, V^{k+1}, \lambda^k) + \frac{1}{\rho}\|\lambda^k\|_2^2 \\
&= f(U^{k+1}(V^{k+1})^T) + \frac{\gamma}{2}\|U^{k+1} - V^{k+1}\|_F^2 + \left\langle \lambda^k, \mathcal{A}(U^{k+1}V^{k+1\top}) - b \right\rangle + \frac{\rho}{2}\|\mathcal{A}(U^{k+1}V^{k+1\top}) - b\|_2^2 + \frac{1}{\rho}\|\lambda^k\|_2^2 \\
&\geq f(U^{k+1}(V^{k+1})^T) + \frac{\gamma}{2}\|U^{k+1} - V^{k+1}\|_F^2 - \frac{\|\lambda^k\|_2^2}{2(\rho - \rho_0)} - \frac{(\rho - \rho_0)}{2}\|\mathcal{A}(U^{k+1}V^{k+1\top}) - b\|_2^2 + \frac{1}{\rho}\|\lambda^k\|_2^2 \\
&\quad + \frac{\rho}{2}\|\mathcal{A}(U^{k+1}V^{k+1\top}) - b\|_2^2 \\
&\geq f(U^{k+1}(V^{k+1})^T) + \frac{\rho_0}{2}\|U^{k+1} - V^{k+1}\|_F^2 + \frac{\rho_0}{2}\|\mathcal{A}(U^{k+1}V^{k+1\top}) - b\|_2^2 + \left(\frac{1}{\rho} - \frac{1}{2(\rho - \rho_0)}\right)\|\lambda^k\|_2^2 \\
&\geq f(U^{k+1}(V^{k+1})^T) + \frac{\rho_0}{2}\|U^{k+1} - V^{k+1}\|_F^2 + \frac{\rho_0}{2}\|\mathcal{A}(U^{k+1}V^{k+1\top}) - b\|_2^2
\end{aligned}
$$

where the last inequality follows from $\rho > 2\rho_0$. This implies $(U^{k+1}, V^{k+1}) \in S_{\widehat{\beta}}$ and it thus holds from Assumption 2 that $\|U^{k+1}\|_F \leq \delta$ and $\|V^{k+1}\|_F \leq \delta$. Analogous to the proof above, we can also establish

$$
f(U^k(V^k)^T) + \frac{\rho_0}{2}\|U^k - V^k\|_F^2 + \frac{\rho_0}{2}\|\mathcal{A}(U^kV^{k\top}) - b\|_2^2 \leq \widehat{\beta}
$$

It therefore holds $(U^{k+1}, V^{k+1}) \in S_{\widehat{\beta}}$ and $\|U^k\|_F \leq \delta, \|V^k\|_F \leq \delta$.

Next, we bound the dual update λ^k and $\lambda^{k+1} - \lambda^k$. To simplify notation, we will omit the superscript δ from the constants L_f^δ and L^δ throughout the proof. According to the optimality condition (12) and the update rule of $\boldsymbol{\lambda}^{k+1}$, we have

$$
\nabla f(V^{k+1}(U^{k+1})^\top)U^{k+1} + \sum_{i=1}^m \lambda_i^{k+1} A_i U^{k+1} - \gamma(U^{k+1} - V^{k+1}) = 0 \tag{32}
$$

Note that $(U^{k+1}, V^{k+1}) \in S_{\widehat{\beta}}$ and it follows from Assumption 3 that

$$
\sigma_{\min}\left(\mathcal{C}(U^{k+1})\right) \geq \eta. \tag{33}
$$

Let $B_{k+1} = \nabla f(V^{k+1}(U^{k+1})^\top)U^{k+1} - \gamma(U^{k+1} - V^{k+1})$, and $b_{k+1} = \mathrm{vec}(B_{k+1}) \in \mathbb{R}^{nr}$. Let $C^{k+1} = \mathcal{C}(U^{k+1}) = [\mathrm{vec}\,(A_1U^{k+1}), \cdots, \mathrm{vec}\,(A_mU^{k+1})] \in \mathbb{R}^{nr\times m}$. Then we can rewrite equation (32) as

$$
C^{k+1}\lambda^{k+1} = -b^{k+1}. \tag{34}
$$

Combining (33) and (34), one can easily show

$$
\begin{aligned}
\eta\|\lambda^{k+1}\|_2 \leq& \|b^{k+1}\|_2 = \|B^{k+1}\|_F \\
\leq& \|\nabla f(V^{k+1}(U^{k+1})^\top)U^{k+1}\|_F + \gamma\|U^{k+1} - V^{k+1}\|_F \\
\leq& L_f\delta + 2\gamma\delta
\end{aligned} \tag{35}
$$

Similarly, we have

$$C^k\lambda^k = -b^k$$

and it therefore holds

$$C^{k+1}\lambda^{k+1} - C^k\lambda^k = C^k\Delta\lambda^{k+1} + (C^{k+1} - C^k)\lambda^{k+1} = -(b^{k+1} - b^k)$$

Then we can bound $\|\Delta\lambda^{k+1}\|_2$ by the following inequality

$$\eta\|\Delta\lambda^{k+1}\|_2 \leq \|C^k\Delta\lambda^{k+1}\|_2 \leq \|b^{k+1} - b^k\|_2 + \|C^{k+1} - C^k\|_F\|\lambda^{k+1}\|_2 \tag{36}$$

In what follows, we bound $\|b^{k+1} - b^k\|_2$ and $\|C^{k+1} - C^k\|_F$ separately

$$\begin{aligned}\|b^{k+1} - b^k\|_2 =& \|B^{k+1} - B^k\|_F \\ \leq& \|\nabla f\big(V^{k+1}(U^{k+1})^T\big)U^{k+1} - \nabla f\big(V^k(U^k)^T\big)U^k\|_F + \gamma\|\Delta U^{k+1}\|_F + \gamma\|\Delta V^{k+1}\|_F \\ \leq& \big(L\delta^2 + L_f + \gamma\big)\|\Delta U^{k+1}\|_F + \big(L\delta^2 + \gamma\big)\|\Delta V^{k+1}\|_F\end{aligned} \tag{37}$$

where the first inequality is because of the definition of B^k and B^{k+1}, and the last inequality is due to Assumption 1. Moreover, according to the definition of C^k, we have

$$\|C^{k+1} - C^k\|_F^2 = \sum_{i=1}^m \|A_i\big(U^{k+1} - U^k\big)\|_F^2 \leq \sum_{i=1}^m \|A_i\|_F^2\|U^{k+1} - U^k\|_F^2 = s_A\|\Delta U^{k+1}\|_F^2 \tag{38}$$

which together with (35) yields

$$\|C^{k+1} - C^k\|_F\|\lambda^k\|_2 \leq \frac{1}{\eta}\big(L_f\delta\sqrt{s_A} + 2\gamma\delta\sqrt{s_A}\big)\|\Delta U^{k+1}\|_F \tag{39}$$

Combining (36), (37) and (39), we have

$$\begin{aligned}\eta\|\lambda^{k+1} - \lambda^k\|_2 \leq& \big(L\delta^2 + L_f + \gamma + \frac{1}{\eta}\big(L_f\delta\sqrt{s_A} + 2\gamma\delta\sqrt{s_A}\big)\big)\|\Delta U^{k+1}\|_F \\ &+ \big(L\delta^2 + \gamma\big)\|\Delta V^{k+1}\|_F\end{aligned}$$

Therefore, we have

$$\begin{aligned}\|\lambda^{k+1} - \lambda^k\|_2^2 \leq& \frac{2}{\eta^2}\big(L\delta^2 + L_f + \gamma + \frac{1}{\eta}L_f\delta\sqrt{s_A} + \frac{2}{\eta}\gamma\eta\sqrt{s_A}\big)^2\|\Delta U^{k+1}\|_F^2 \\ &+ \frac{2}{\eta^2}\big(L\delta^2 + \gamma\big)^2\|\Delta V^{k+1}\|_F^2 \\ \triangleq& c_1\|\Delta U^{k+1}\|_F^2 + c_2\|\Delta V^{k+1}\|_F^2\end{aligned} \tag{40}$$

Finally, we establish the descent of the augmented Lagrangian function and justify the validation of part (i) for iterate $k+1$. By invoking proposition (1) and using (40), it holds that

$$\begin{aligned}&\mathcal{L}_\rho(U^{k+1}, V^{k+1}, \lambda^{k+1}) - \mathcal{L}_\rho(U^k, V^k, \lambda^k) \\ \leq& \frac{1}{\rho}\|\Delta\lambda^{k+1}\|_2^2 - \frac{\gamma}{2}\big(\|\Delta U^{k+1}\|_F^2 + \|\Delta V^{k+1}\|^2\big) \\ \leq& -\big(\frac{\gamma}{2} - \frac{c_1}{\rho}\big)\|\Delta U^{k+1}\|_F^2 - \big(\frac{\gamma}{2} - \frac{c_2}{\rho}\big)\|\Delta V^{k+1}\|_F^2 \\ \leq& 0\end{aligned}$$

,

Since $\rho > \frac{1}{\widehat{\beta}-\beta^0}\left(\frac{L_f\delta+2\gamma\delta}{\eta}\right)^2$, we know

$$\begin{aligned} L_\rho(U^{k+1},V^{k+1},\lambda^{k+1})+\frac{1}{\rho}\|\lambda^{k+1}\|_2^2 &\le L_\rho(U^k,V^k,\lambda^k)+\frac{1}{\rho}\|\lambda^{k+1}\|_2^2 \\ &\le \beta^0+\frac{1}{\rho}\left(\frac{L_f\delta+2\gamma\delta}{\eta}\right)^2<\widehat{\beta} \end{aligned}$$

This completes the proof.

□

A.5 Proof of Theorem 2

Proof. Proof Based on the developments outlined in Lemma 1, we deduce the existence of a constant κ such that

$$\mathcal{L}_\rho(U^{k+1},V^{k+1},\lambda^{k+1})-\mathcal{L}_\rho(U^k,V^k,\lambda^k)\le-\kappa\|\Delta U^{k+1}\|_F^2-\kappa\|\Delta V^{k+1}\|_F^2 \tag{41}$$

On the other hand, we know from (15) and (16) that

$$\|\nabla\mathcal{L}(U^{k+1},V^{k+1},\lambda^{k+1})\|_F^2\le\left(L_f^\delta+L^\delta\delta^2+\rho s_A\delta^2+\|\mathcal{A}\|_2\delta+\gamma\right)^2\|\Delta V^{k+1}\|_F^2+\frac{1}{\rho^2}\|\Delta\lambda^{k+1}\|^2$$

where $\mathcal{L}$ is the Lagrangian function of (6). Note that

$$\mathcal{L}_\rho(U,V,\lambda)=\mathcal{L}(U,V,\lambda)+\frac{\rho}{2}\|\mathcal{A}(UV^T)-b\|_2^2$$

there must exist a constant $\zeta>0$ such that

$$\|\nabla\mathcal{L}_\rho(U^{k+1},V^{k+1},\lambda^{k+1})\|_F\le\zeta(\|\Delta U^{k+1}\|_F+\|\Delta V^{k+1}\|_F+\|\Delta\lambda^{k+1}\|_2). \tag{42}$$

Therefore, by using (42), (41) and Assumption 4, we can invoke [ABS13, Theorem 2.9] that the sequence $\{(U^k,V^k,\lambda^k)\}_{k=0}^\infty$ converges to a critical point of $L_\rho(U,V,\lambda)$, which corresponds to a KKT point of problem (6). We further have the sequence $\{(U^k,V^k,\lambda^k)\}_{k=0}^\infty$ has a finite length, i.e.

$$\sum_{j=1}^\infty\|\Delta U^{j+1}\|_F+\|\Delta V^{j+1}\|_F+\|\Delta\lambda^{j+1}\|_2<\infty$$

By elementary calculus, one can show that

$$\min_{1\le j\le K}\|\Delta U^{j+1}\|_F+\|\Delta V^{j+1}\|_F+\|\Delta\lambda^{j+1}\|_2=O(\frac{1}{K})$$

This, together with (15) and (16), implies that

$$\begin{aligned} \min_{1\le j\le K}\Bigg\{&\|\nabla f(U^{k+1}(V^{k+1})^\top)V^{k+1}+\gamma(U^{k+1}-V^{k+1})+\sum_{i=1}^m\lambda_i^{k+1}A_iV^{k+1}\|_F \\ &+\|\nabla f(V^{k+1}(U^{k+1})^\top)V^{k+1}+\gamma(V^{k+1}-U^{k+1})+\sum_{i=1}^m\lambda_i^{k+1}A_iV^{k+1}\|_F \\ &+\|\mathcal{A}(U^{k+1}(V^{k+1})^T)-b\|_2\Bigg\}=O\left(\frac{1}{K}\right) \end{aligned}$$

In other words, Algorithm 1 requires $O(1/\epsilon)$ iterations to produce an ϵ-KKT point of (6). □

A.6 Proof of Theorem 3

Proof. Proof Because of the convergence of the sequence $\{(U^k, V^k, \lambda^k)\}_{k=0}^{\infty}$, the iteration point will enter the local area around the limit point and satisfies the Łojasiewicz inequality after a constant number of steps. For the simplicity of notation, we assume $(U^k, V^k, \lambda^k) \in \mathcal{B}_{\delta_c}(\overline{U}, \overline{V}, \overline{\lambda})$ for any $k \geq 0$ in the following analysis.

First, consider the case where $\alpha = 0$. Assume that the sequence does not converge in finite steps and hence $\mathcal{L}_\rho(U^k, V^k, \lambda^k) - \mathcal{L}_\rho(\overline{U}, \overline{V}, \overline{\lambda}) > 0$ holds for any $k \geq 0$. According to the Łojasiewicz inequality, we then have

$$c\|\nabla \mathcal{L}_\rho(U^k, V^k, \lambda^k)\|_F \geq 1.$$

This contradicts to the result that the sequence converges to a KKT point. Hence, the sequence must converge in finite steps when $\alpha = 0$.

Next, we consider the remaining cases where $\alpha \in (0, 1)$. Define $\phi(t) = \frac{c}{1-\alpha} t^{1-\alpha}$. We can then rewrite the Łojasiewicz inequality (21) as

$$\phi'(\mathcal{L}_\rho(U, V, \lambda) - \mathcal{L}_\rho(\overline{U}, \overline{V}, \overline{\lambda})) \cdot \|\nabla \mathcal{L}_\rho(U, V, \lambda)\|_F \geq 1. \tag{43}$$

Let $l^k = \mathcal{L}_\rho(U^k, V^k, \lambda^k) - \mathcal{L}_\rho(\overline{U}, \overline{V}, \overline{\lambda})$. Then according to the Łojasiewicz inequality, we have

$$\phi'(l^k) \cdot \|\nabla \mathcal{L}_\rho(U^k, V^k, \lambda^k)\|_F \geq 1.$$

Because $l^k - l^{k+1} \geq 0$, by multiplying it to both side we get the following inequality

$$\phi'(l^k)(l^k - l^{k+1}) \cdot \|\nabla \mathcal{L}_\rho(U^k, V^k, \lambda^k)\|_F \geq l^k - l^{k+1}.$$

Note that ϕ is concave, we further have

$$\big(\phi(l^k) - \phi(l^{k+1})\big) \cdot \|\nabla \mathcal{L}_\rho(U^k, V^k, \lambda^k)\|_F \geq l^k - l^{k+1}.$$

According to Lemma 1 and (42), we can bound $l^k - l^{k+1}$ and $\|\nabla \mathcal{L}_\rho(U^k, V^k, \lambda^k)\|_F$ respectively

$$\begin{aligned} l^k - l^{k+1} &\geq \kappa\big(\|\Delta U^{k+1}\|_F^2 + \|\Delta V^{k+1}\|_F^2\big), \\ \|\nabla \mathcal{L}_\rho(U^k, V^k, \lambda^k)\|_F &\leq \zeta\big(\|\Delta U^k\|_F + \|\Delta V^k\|_F + \|\Delta \lambda^k\|_2\big). \end{aligned}$$

Note that $\|\Delta \lambda^{k+1}\|_2^2$ can be upper bounded by $\|\Delta U^{k+1}\|_F^2$ and $\|\Delta V^{k+1}\|_F^2$, hence there exists some constant $\widetilde{\kappa} > 0$ such that

$$l^k - l^{k+1} \geq \widetilde{\kappa}\big(\|\Delta U^{k+1}\|_F^2 + \|\Delta V^{k+1}\|_F^2 + \|\Delta \lambda^{k+1}\|_2^2\big).$$

Combining the above inequalities, the following inequality follows naturally

$$\frac{\zeta}{\widetilde{\kappa}}\big(\phi(l^k) - \phi(l^{k+1})\big)\big(\|\Delta U^k\|_F + \|\Delta V^k\|_F + \|\Delta \lambda^k\|_2\big) \geq \big(\|\Delta U^{k+1}\|_F^2 + \|\Delta V^{k+1}\|_F^2 + \|\Delta \lambda^{k+1}\|_2^2\big).$$

Recall the inequality that $2\sqrt{ab} \leq a + b$, we can further get

$$\frac{2\zeta}{\widetilde{\kappa}}\big(\phi(l^k) - \phi(l^{k+1})\big) + \big(\|\Delta U^k\|_F + \|\Delta V^k\|_F + \|\Delta \lambda^k\|_2\big) \geq 2\big(\|\Delta U^{k+1}\|_F + \|\Delta V^{k+1}\|_F + \|\Delta \lambda^{k+1}\|_2\big).$$

By adding the above inequality from k to K, we have

$$\frac{2\zeta}{\widetilde{\kappa}}\big(\phi(l^k)-\phi(l^{K+1})\big)+\big(\|\Delta U^k\|_F+\|\Delta V^k\|_F+\|\Delta\lambda^k\|_2\big)\geq\sum_{t=k}^{K}\big(\|\Delta U^{t+1}\|_F+\|\Delta V^{t+1}\|_F+\|\Delta\lambda^{t+1}\|_2\big)$$
$$+\big(\|\Delta U^{K+1}\|_F+\|\Delta V^{K+1}\|_F+\|\Delta\lambda^{K+1}\|_2\big).$$

Because $\mathcal{L}_\rho(U^k,V^k,\lambda^k)$ is non-increasing, we have $l^{K+1}>0$ and hence $\phi(l^k)-\phi(l^{K+1})\leq\phi(l^k)$. Taking K to ∞, we can derive the following inequality

$$\frac{2\zeta}{\widetilde{\kappa}}\phi(l^k)+\big(\|\Delta U^k\|_F+\|\Delta V^k\|_F+\|\Delta\lambda^k\|_2\big)\geq\sum_{t=k}^{\infty}\big(\|\Delta U^{t+1}\|_F+\|\Delta V^{t+1}\|_F+\|\Delta\lambda^{t+1}\|_2\big).\quad(44)$$

Let $S^k=\sum_{t=k}^{\infty}\big(\|\Delta U^{t+1}\|_F+\|\Delta V^{t+1}\|_F+\|\Delta\lambda^{t+1}\|_2\big)$. Then we can rewrite the above inequality as follows

$$\frac{2\zeta}{\widetilde{\kappa}}\phi(l^k)+S^{k-1}-S^k\geq S^k.\quad(45)$$

According to the definition of ϕ and the Łojasiewicz inequality, we have

$$c(l^k)^{-\alpha}\cdot\|\nabla\mathcal{L}_\rho(U^k,V^k,\lambda^k)\|_F\geq 1,$$

and hence there exists a constant $\widetilde{c}>0$ such that

$$\phi(l^k)=\frac{c}{1-\alpha}(l^k)^{1-\alpha}\leq\widetilde{c}\big(\|\Delta U^k\|_F+\|\Delta V^k\|_F+\|\Delta\lambda^k\|_2\big)^{\frac{1-\alpha}{\alpha}}=\widetilde{c}(S^{k-1}-S^k)^{\frac{1-\alpha}{\alpha}}.$$

Combining with (45), we have

$$\frac{2\zeta\widetilde{c}}{\widetilde{\kappa}}(S^{k-1}-S^k)^{\frac{1-\alpha}{\alpha}}+S^{k-1}-S^k\geq S^k.$$

The convergence property of such sequence has been studied in Theorem 2 of [AB09], which gives the analysis of convergence rate.

□

B Additional Numerical Results

In this section, we provide the solving errors for the instances tested in Section 5. Since each solver terminates with distinct criteria, we present three commonly defined errors for the solutions of each solver including:

Primal Infeasibility:

$$\frac{\|\mathcal{A}(X)-b\|_2}{1+\|b\|_1},$$

Dual Infeasibility:

$$\frac{|\min\{0,\sigma_{\min}(C-\mathcal{A}^*(\lambda))\}|}{1+\|c\|_1},$$

Primal Dual Gap:

$$\frac{\langle C,X\rangle-\lambda^\top b}{1+|\langle C,X\rangle|+|\lambda^\top b|}.$$

The three error measures are chosen for ranking in Hans Mittelmann's benchmarks, and it is also widely used in many existing solvers, like COPT [GHW+22] and HDSDP [GGY22]. The reports of errors in all three experiments are shown in the following tables.

Table 8: Solving Errors on Max-cut Problems

	LoRADS			SDPLR			SDPNAL+			COPT		
Problem	P Infeasi	D Infeasi	PD Gap	P Infeasi	D Infeasi	PD Gap	P Infeasi	D Infeasi	PD Gap	P Infeasi	D Infeasi	PD Gap
G1	1.19E-08	2.51E-07	4.10E-09	1.84E-08	4.51E-09	1.59E-06	8.11E-08	2.27E-08	2.64E-07	6.54E-13	0.00E+00	2.37E-09
G2	1.18E-08	2.36E-07	4.90E-09	2.44E-08	6.09E-09	1.77E-06	1.53E-07	2.76E-08	1.35E-06	1.60E-13	0.00E+00	3.21E-09
G3	1.16E-08	5.62E-07	3.21E-09	1.28E-08	2.94E-08	3.81E-07	2.59E-07	3.03E-08	4.90E-06	1.00E-13	0.00E+00	3.93E-09
G4	1.35E-08	4.58E-07	3.68E-09	6.88E-09	1.54E-08	4.12E-07	1.95E-07	1.91E-08	1.37E-06	1.85E-13	0.00E+00	2.25E-09
G5	1.33E-08	3.21E-07	6.70E-09	2.29E-08	7.51E-09	2.58E-07	2.77E-07	8.80E-08	6.39E-06	5.84E-14	0.00E+00	3.38E-09
G6	5.97E-09	4.57E-07	8.81E-09	1.62E-08	8.56E-09	9.76E-08	6.95E-08	7.13E-09	9.55E-07	5.20E-14	0.00E+00	2.39E-08
G7	8.77E-09	1.62E-07	2.17E-08	2.46E-08	5.77E-10	6.35E-06	6.43E-08	6.53E-09	1.06E-06	4.00E-12	0.00E+00	1.22E-08
G8	6.89E-09	6.61E-07	1.29E-08	2.03E-08	1.02E-08	3.64E-06	5.13E-08	2.87E-09	5.39E-07	7.10E-13	0.00E+00	1.11E-08
G9	5.21E-09	1.00E-07	9.17E-09	1.77E-08	4.50E-09	1.48E-06	2.05E-09	3.51E-08	1.38E-05	2.60E-13	0.00E+00	1.15E-08
G10	3.94E-09	8.37E-08	8.66E-09	1.44E-08	5.20E-09	1.92E-06	6.09E-08	2.00E-08	1.04E-05	1.80E-13	0.00E+00	9.39E-09
G11	1.58E-08	2.95E-07	2.38E-08	2.25E-08	1.04E-08	9.26E-07	2.53E-08	5.69E-08	1.19E-05	1.53E-13	0.00E+00	6.44E-08
G12	1.88E-08	4.21E-07	3.48E-08	2.40E-08	1.56E-08	9.59E-07	2.23E-07	5.84E-08	1.61E-05	1.34E-13	0.00E+00	2.78E-08
G13	1.87E-08	1.64E-07	3.11E-08	2.04E-08	1.43E-08	3.31E-06	1.04E-15	7.66E-08	1.26E-05	1.90E-14	0.00E+00	4.09E-08
G14	2.46E-08	2.00E-06	1.73E-07	1.61E-08	8.89E-09	3.72E-07	4.27E-08	8.15E-08	2.83E-06	2.29E-14	0.00E+00	2.45E-08
G15	1.17E-08	1.24E-06	9.35E-08	2.11E-08	1.42E-08	1.36E-06	1.22E-07	2.59E-08	8.42E-07	1.67E-12	0.00E+00	6.14E-09
G16	6.75E-09	1.00E-06	1.08E-08	2.18E-08	1.23E-08	1.24E-06	1.61E-07	1.02E-07	7.55E-06	1.51E-13	0.00E+00	1.01E-08
G17	1.72E-08	1.17E-06	9.45E-08	2.32E-08	2.11E-08	3.54E-06	5.46E-08	2.09E-08	8.83E-07	2.87E-13	0.00E+00	7.50E-09
G18	1.80E-08	2.95E-07	1.41E-07	1.62E-08	1.16E-08	1.96E-06	3.56E-07	2.51E-09	3.78E-06	6.69E-13	0.00E+00	3.36E-08
G19	3.04E-09	2.55E-07	3.37E-09	1.81E-08	2.27E-08	4.63E-06	4.27E-08	7.12E-10	1.98E-07	4.40E-13	0.00E+00	6.31E-08
G20	2.49E-08	3.67E-07	1.45E-07	1.75E-08	2.58E-07	1.48E-06	1.03E-07	1.53E-08	2.73E-07	3.54E-13	0.00E+00	4.74E-08
G21	2.32E-08	1.26E-07	1.73E-07	1.97E-08	6.01E-09	5.85E-07	2.28E-08	5.29E-08	8.87E-06	7.33E-13	0.00E+00	4.10E-08
G22	2.33E-09	3.68E-07	2.58E-09	7.46E-09	9.26E-09	1.12E-06	1.93E-07	4.47E-08	9.03E-06	3.24E-14	0.00E+00	4.80E-09
G23	2.09E-09	4.76E-07	1.07E-09	5.03E-09	4.28E-09	6.25E-07	1.27E-08	3.63E-08	9.80E-06	3.28E-14	0.00E+00	7.53E-09
G24	2.35E-09	2.29E-07	2.96E-09	9.46E-09	7.39E-09	1.25E-06	1.90E-08	6.64E-08	1.28E-05	1.73E-13	0.00E+00	1.79E-09
G25	2.47E-09	2.95E-07	2.95E-09	6.63E-09	8.38E-09	9.15E-07	1.88E-07	5.96E-08	1.04E-05	3.29E-14	0.00E+00	5.88E-09
G26	2.39E-09	2.87E-07	3.27E-09	9.27E-09	1.78E-09	1.28E-06	2.04E-07	4.47E-08	1.00E-05	6.83E-15	0.00E+00	8.11E-09
G27	8.27E-09	9.04E-07	1.29E-08	6.69E-09	1.49E-09	9.32E-07	7.83E-08	3.89E-09	1.83E-06	8.81E-14	0.00E+00	7.47E-09
G28	8.76E-09	7.36E-07	5.41E-08	5.30E-09	6.75E-09	3.86E-07	7.45E-08	1.38E-08	5.95E-06	2.19E-14	0.00E+00	4.63E-09
G29	9.94E-10	6.00E-07	1.80E-09	4.24E-09	0.00E+00	3.25E-06	4.18E-08	2.55E-08	9.31E-06	2.38E-13	0.00E+00	4.80E-09
G30	1.11E-09	6.00E-07	1.96E-09	8.81E-09	1.15E-08	9.01E-07	6.79E-08	3.16E-09	1.57E-06	2.48E-12	0.00E+00	9.40E-09
G31	7.24E-09	1.03E-06	2.80E-08	6.52E-09	8.20E-09	9.62E-07	3.02E-16	2.51E-08	1.05E-05	8.41E-14	0.00E+00	7.05E-09
G32	1.07E-09	2.86E-08	2.95E-10	8.51E-09	6.59E-10	1.94E-06	7.41E-08	3.47E-08	2.03E-05	2.80E-14	0.00E+00	4.15E-08
G33	1.04E-09	2.84E-08	3.43E-09	9.34E-09	3.66E-09	1.98E-07	1.34E-15	3.69E-08	1.90E-05	6.40E-14	0.00E+00	4.54E-08
G34	9.31E-10	1.63E-08	1.95E-09	5.07E-09	1.10E-09	8.99E-07	6.08E-09	3.81E-08	1.58E-05	2.79E-14	0.00E+00	1.10E-08
G35	9.53E-09	6.24E-07	9.16E-08	8.67E-09	6.81E-09	2.24E-06	4.90E-16	7.34E-08	2.72E-05	1.49E-13	0.00E+00	6.46E-09
G36	4.29E-09	5.41E-07	5.29E-08	8.23E-09	4.77E-09	5.69E-07	9.81E-09	2.73E-08	1.11E-05	1.78E-13	0.00E+00	7.59E-09
G37	8.49E-09	7.48E-07	8.77E-08	8.63E-09	0.00E+00	6.44E-06	8.40E-08	3.55E-08	4.64E-06	8.52E-13	0.00E+00	5.63E-09
G38	8.52E-09	4.37E-07	1.25E-07	9.13E-09	3.75E-09	3.85E-07	4.50E-08	4.98E-08	1.08E-05	6.95E-13	0.00E+00	4.93E-09
G39	1.74E-09	5.79E-08	8.31E-09	9.05E-09	2.03E-09	1.63E-06	2.45E-08	4.39E-08	1.45E-05	6.48E-13	0.00E+00	6.79E-09
G40	3.25E-09	1.13E-07	3.04E-08	6.87E-09	6.19E-09	5.75E-07	3.59E-16	4.55E-08	1.07E-05	9.76E-13	0.00E+00	6.76E-09
G41	3.03E-09	1.93E-07	3.23E-08	5.55E-09	9.54E-09	1.62E-06	1.35E-07	1.37E-08	4.93E-06	1.86E-14	0.00E+00	2.88E-08
G42	2.71E-09	5.15E-08	2.03E-08	9.77E-09	1.93E-08	1.70E-06	2.80E-16	4.38E-08	1.62E-05	7.22E-15	0.00E+00	7.04E-09
G43	5.24E-09	2.87E-07	7.00E-09	1.60E-08	5.26E-09	1.01E-06	3.76E-16	5.98E-06	1.45E-03	8.51E-14	0.00E+00	8.73E-09
G44	6.01E-09	2.94E-07	7.80E-10	1.38E-08	3.04E-09	5.46E-07	3.03E-16	6.18E-06	1.38E-03	8.90E-14	0.00E+00	8.15E-09
G45	3.93E-09	3.14E-07	2.50E-09	1.67E-08	4.54E-09	1.40E-06	1.36E-08	7.65E-09	3.66E-07	1.71E-13	0.00E+00	6.09E-09
G46	4.53E-09	2.43E-07	4.14E-09	7.87E-09	3.52E-09	4.46E-07	2.77E-07	2.23E-09	1.65E-06	3.92E-12	0.00E+00	7.03E-09
G47	5.61E-09	2.71E-07	4.12E-09	1.10E-08	1.35E-08	4.29E-07	1.18E-07	5.26E-09	1.35E-07	8.55E-13	0.00E+00	3.92E-09
G48	2.72E-09	5.76E-09	1.48E-08	5.90E-09	1.39E-10	1.88E-07	1.26E-14	3.36E-07	3.95E-03	1.65E-17	0.00E+00	5.28E-09
G49	3.33E-09	5.77E-09	1.30E-08	6.11E-09	3.26E-10	5.35E-07	6.03E-14	3.28E-07	5.26E-04	1.71E-17	0.00E+00	5.28E-09
G50	2.38E-09	5.57E-08	1.54E-08	5.24E-09	0.00E+00	2.26E-07	8.06E-15	2.28E-07	1.85E-04	6.43E-17	0.00E+00	1.26E-08
G51	1.83E-08	1.01E-06	7.32E-08	1.24E-08	1.86E-08	6.79E-07	2.14E-07	3.54E-08	7.63E-07	1.30E-13	0.00E+00	9.00E-09
G52	1.81E-08	1.17E-06	1.37E-07	1.80E-08	2.19E-08	1.69E-06	6.27E-08	2.05E-08	6.22E-07	2.90E-14	0.00E+00	9.60E-09
G53	1.62E-08	5.05E-07	7.00E-09	1.58E-08	1.34E-08	1.02E-06	1.28E-15	7.01E-06	1.61E-03	2.77E-14	0.00E+00	4.73E-09
G54	1.35E-08	1.05E-06	6.46E-08	1.58E-08	0.00E+00	4.91E-06	1.60E-07	5.27E-08	1.66E-06	5.65E-13	0.00E+00	5.02E-09
G55	1.66E-09	1.35E-07	1.08E-08	3.12E-09	1.04E-09	4.29E-07	1.81E-08	2.66E-08	1.44E-06	9.48E-15	0.00E+00	7.23E-09
G56	3.65E-09	2.49E-07	2.94E-08	1.76E-09	1.71E-09	8.54E-07	4.88E-09	1.31E-08	7.14E-06	6.56E-15	0.00E+00	5.04E-09
G57	9.60E-10	9.40E-09	4.69E-09	3.12E-09	8.73E-10	3.24E-07	9.90E-10	2.02E-08	2.79E-05	2.19E-13	0.00E+00	1.12E-08
G58	1.76E-09	3.31E-07	2.14E-08	1.97E-09	9.71E-11	3.70E-06	5.60E-08	4.08E-08	4.05E-05	7.96E-14	0.00E+00	6.47E-09
G59	1.92E-09	1.21E-07	2.87E-08	3.74E-09	7.40E-10	1.04E-06	2.66E-08	1.40E-08	1.28E-05	2.79E-13	0.00E+00	8.42E-09
G60	9.24E-10	1.63E-07	4.42E-09	2.00E-09	5.05E-10	6.38E-07	t	t	t	1.25E-15	0.00E+00	4.15E-09
G61	1.22E-09	2.26E-07	4.63E-09	2.79E-09	6.91E-10	2.16E-06	t	t	t	1.16E-15	0.00E+00	8.51E-09
G62	7.77E-10	3.72E-09	4.36E-08	1.14E-09	2.15E-10	1.42E-07	t	t	t	4.89E-14	0.00E+00	6.76E-09
G63	8.78E-10	2.02E-07	5.25E-09	2.52E-09	1.26E-09	3.67E-07	t	t	t	1.45E-14	0.00E+00	4.03E-09
G64	2.19E-09	1.92E-07	4.28E-08	1.75E-09	3.87E-09	6.00E-07	t	t	t	4.54E-13	0.00E+00	2.99E-09
G65	5.79E-10	3.74E-09	9.13E-09	1.40E-09	3.05E-11	5.35E-07	t	t	t	6.44E-14	0.00E+00	5.59E-09
G66	4.45E-10	2.76E-09	1.00E-08	3.37E-10	2.06E-10	5.27E-08	t	t	t	5.78E-14	0.00E+00	9.57E-09
G67	6.68E-10	2.98E-09	2.04E-08	1.19E-09	0.00E+00	9.29E-07	t	t	t	1.20E-13	0.00E+00	8.31E-09

Table 9: Solving Errors on Matrix Completion Problems

	LoRADS			SDPLR			SDPNAL+			COPT		
Problem	P Infeasi	D Infeasi	PD Gap	P Infeasi	D Infeasi	PD Gap	P Infeasi	D Infeasi	PD Gap	P Infeasi	D Infeasi	PD Gap
MC_1000	1.45E-10	4.04E-08	3.84E-06	3.00E-10	3.54E-07	2.26E-05	1.71E-18	3.69E-09	1.49E-06	1.30E-06	0.00E+00	1.96E-08
MC_2000	1.45E-10	2.18E-08	1.90E-06	1.20E-10	2.28E-08	2.39E-06	1.87E-09	2.25E-09	2.00E-06	2.02E-06	0.00E+00	6.99E-09
MC_3000	5.49E-11	4.22E-10	9.30E-07	4.17E-11	3.68E-08	1.60E-05	1.30E-09	3.88E-10	1.45E-06	4.76E-06	0.00E+00	4.85E-09
MC_4000	4.58E-11	2.54E-09	5.78E-06	5.40E-11	6.14E-08	4.07E-06	1.62E-09	1.23E-09	2.98E-06	6.77E-06	0.00E+00	3.11E-09
MC_5000	3.18E-11	1.46E-09	1.24E-06	2.07E-11	4.39E-08	1.88E-06	6.60E-10	2.81E-10	2.11E-07	6.73E-06	0.00E+00	1.34E-08
MC_6000	3.74E-11	3.89E-09	2.24E-06	4.19E-11	1.51E-08	1.05E-05	2.87E-10	0.00E+00	1.81E-06	5.06E-06	0.00E+00	9.80E-09
MC_8000	2.57E-11	3.75E-09	7.26E-07	t	t	t	8.84E-11	1.29E-08	6.39E-06	t	t	t
MC_10000	2.13E-11	3.05E-09	1.20E-06	t	t	t	2.79E-09	3.49E-11	1.37E-05	t	t	t
MC_12000	1.76E-11	4.15E-09	2.25E-06	-	-	-	-	-	-	-	-	-
MC_14000	1.26E-11	4.52E-10	1.48E-06	-	-	-	-	-	-	-	-	-
MC_16000	1.20E-11	5.25E-09	5.23E-06	-	-	-	-	-	-	-	-	-
MC_18000	1.17E-11	2.06E-09	4.33E-06	-	-	-	-	-	-	-	-	-
MC_20000	1.09E-11	1.53E-09	9.54E-07	-	-	-	-	-	-	-	-	-
MC_40000	5.17E-12	8.41E-11	1.37E-06	-	-	-	-	-	-	-	-	-

Table 10: Solving Errors on Problems from SDPLIB and Mittelmann's Benchmark

	LoRADS			SDPLR			SDPNAL+			COPT		
Problem	P Infeasi	D Infeasi	PD Gap	P Infeasi	D Infeasi	PD Gap	P Infeasi	D Infeasi	PD Gap	P Infeasi	D Infeasi	PD Gap
ALH	1.29E-06	1.46E-04	1.23E-03	1.36E-06	2.85E-05	4.87E-04	7.24E-07	1.85E-07	6.70E-06	7.30E-10	0.00E+00	2.51E-09
BH2	6.07E-07	1.82E-02	5.30E-03	1.04E-06	5.07E-05	1.24E-04	2.72E-07	9.39E-08	1.78E-06	6.21E-10	0.00E+00	2.54E-09
cancer_100	2.86E-09	3.28E-04	1.77E-05	t	t	t	2.61E-08	9.24E-10	4.43E-05	1.97E-05	0.00E+00	4.54E-09
CH2	8.95E-07	2.09E-03	6.09E-04	1.13E-06	1.99E-04	2.49E-04	7.92E-09	2.60E-07	3.48E-04	6.36E-10	0.00E+00	3.56E-09
checker_1.5	4.58E-09	5.29E-08	1.54E-07	4.11E-09	1.83E-08	1.36E-04	9.12E-08	2.27E-07	1.25E-03	2.70E-13	0.00E+00	4.53E-08
G40_mb	9.72E-09	1.29E-07	9.23E-09	9.89E-09	8.57E-09	1.07E-06	9.06E-10	0.00E+00	3.33E-07	6.01E-07	0.00E+00	1.07E-05
G48_mb	4.03E-09	8.63E-10	1.37E-07	4.05E-09	0.00E+00	2.96E-04	1.14E-11	1.00E-07	7.15E-02	1.92E-04	0.00E+00	4.85E-03
gpp100	1.96E-07	1.41E-06	1.75E-06	1.98E-07	1.56E-08	2.62E-06	1.65E-12	5.20E-06	7.97E-04	1.67E-08	0.00E+00	6.59E-09
gpp124-1	1.58E-07	3.02E-06	8.02E-06	1.60E-07	5.62E-06	2.81E-03	1.10E-06	0.00E+00	3.47E-06	3.46E-07	0.00E+00	4.77E-08
gpp124-2	1.59E-07	8.72E-07	1.54E-06	1.54E-07	2.81E-08	1.34E-06	9.19E-07	0.00E+00	4.22E-06	4.20E-08	0.00E+00	2.09E-09
gpp124-3	1.58E-07	2.52E-06	6.50E-07	1.57E-07	1.41E-08	6.04E-07	3.32E-12	3.12E-06	4.59E-04	4.96E-08	0.00E+00	6.00E-09
gpp124-4	1.60E-07	2.44E-07	9.99E-07	1.58E-07	1.11E-06	4.81E-05	6.48E-09	0.00E+00	3.73E-06	6.50E-08	0.00E+00	3.19E-09
gpp250-1	7.87E-08	4.75E-07	2.64E-06	7.41E-08	1.11E-06	3.17E-05	6.32E-07	0.00E+00	9.06E-06	7.89E-08	0.00E+00	2.93E-08
gpp250-2	7.75E-08	2.90E-07	1.20E-06	7.42E-08	1.78E-09	7.79E-07	1.79E-11	1.05E-05	4.93E-03	4.53E-08	0.00E+00	7.70E-09
gpp250-3	7.84E-08	9.26E-08	3.28E-07	7.97E-08	1.73E-07	4.45E-05	6.35E-07	0.00E+00	6.11E-06	7.52E-08	0.00E+00	5.34E-09
gpp250-4	7.89E-08	5.64E-08	3.48E-07	7.92E-08	6.03E-08	1.37E-05	3.30E-12	8.06E-06	1.28E-03	1.21E-07	0.00E+00	3.61E-09
gpp500-1	3.98E-08	2.00E-07	7.84E-07	3.88E-08	2.37E-07	2.11E-04	8.89E-07	1.15E-11	5.76E-06	1.22E-06	0.00E+00	1.02E-05
gpp500-2	3.97E-08	1.08E-07	5.30E-07	3.94E-08	1.02E-09	3.64E-07	1.51E-06	7.64E-11	5.08E-07	1.37E-07	0.00E+00	1.30E-06
gpp500-3	3.93E-08	3.45E-07	2.24E-08	3.98E-08	2.38E-08	1.18E-05	1.29E-10	5.03E-06	4.10E-03	1.02E-07	0.00E+00	1.01E-08
gpp500-4	3.98E-08	1.17E-07	3.41E-08	3.99E-08	0.00E+00	2.34E-05	4.75E-13	3.95E-10	6.99E-07	3.23E-07	0.00E+00	2.47E-06
H3O	1.13E-06	1.29E-04	2.47E-04	1.18E-06	2.13E-05	7.14E-05	6.84E-08	9.82E-08	5.39E-06	7.36E-10	0.00E+00	3.49E-09
hand	7.62E-08	2.27E-09	2.71E-08	1.54E-08	4.34E-09	1.08E-07	8.87E-09	3.97E-12	2.14E-07	2.57E-06	0.00E+00	4.09E-05
ice_2.0	1.41E-09	1.63E-07	2.60E-06	5.17E-10	1.63E-07	1.01E-05	t	t	t	1.02E-11	0.00E+00	1.98E-08
mcp100	1.03E-07	5.71E-06	2.60E-07	5.96E-08	7.03E-08	4.04E-06	4.90E-16	8.26E-06	3.55E-05	1.13E-11	0.00E+00	1.72E-09
mcp124-1	5.04E-08	9.37E-06	2.52E-07	9.96E-08	3.21E-08	1.47E-05	2.80E-07	6.27E-08	8.10E-07	1.42E-11	0.00E+00	5.65E-10
mcp124-2	1.47E-07	1.03E-05	3.15E-07	1.56E-07	2.43E-07	1.12E-06	4.28E-16	1.45E-05	3.17E-04	5.39E-12	0.00E+00	6.00E-10
mcp124-3	1.55E-07	4.64E-06	7.67E-08	1.44E-07	3.99E-08	4.80E-06	3.66E-16	9.71E-06	3.94E-04	1.74E-12	0.00E+00	4.61E-09
mcp124-4	1.46E-07	1.94E-06	6.70E-08	3.49E-08	8.71E-08	2.79E-07	1.88E-15	1.02E-05	5.01E-04	1.44E-11	0.00E+00	7.56E-10
mcp250-1	4.91E-08	2.62E-06	2.01E-07	3.89E-08	4.57E-08	5.62E-06	2.30E-07	1.68E-08	1.10E-07	3.11E-12	0.00E+00	4.52E-09
mcp250-2	6.07E-08	5.33E-06	1.53E-07	7.23E-08	2.70E-09	1.29E-05	2.01E-15	9.92E-06	1.09E-04	3.97E-12	0.00E+00	1.26E-09
mcp250-3	7.96E-08	1.59E-06	6.79E-08	7.63E-08	3.10E-08	3.09E-06	3.92E-07	3.76E-10	9.89E-08	1.72E-10	0.00E+00	4.77E-09
mcp250-4	3.11E-08	1.49E-06	3.13E-08	7.20E-08	8.55E-08	4.10E-06	3.65E-07	1.99E-10	1.89E-07	1.03E-11	0.00E+00	9.63E-10
mcp500-1	2.71E-08	1.63E-06	1.16E-07	3.25E-08	8.11E-08	4.40E-06	1.25E-07	9.61E-08	4.60E-06	1.19E-15	0.00E+00	2.23E-07
mcp500-2	2.37E-09	5.15E-07	8.19E-09	2.88E-08	1.25E-07	7.21E-06	3.52E-14	1.00E-07	2.80E-06	4.12E-14	0.00E+00	5.99E-08
mcp500-3	6.45E-09	1.01E-06	1.18E-08	2.41E-08	6.80E-08	8.77E-06	2.75E-07	2.41E-08	1.61E-06	9.74E-14	0.00E+00	2.28E-08
mcp500-4	1.97E-08	6.44E-07	2.43E-08	2.54E-08	1.18E-08	1.29E-06	3.10E-16	8.22E-06	1.04E-03	2.00E-14	0.00E+00	1.10E-08
NH2	1.26E-06	3.10E-02	1.68E-02	1.17E-06	5.52E-05	8.48E-05	3.85E-15	2.11E-06	9.10E-06	1.84E-09	0.00E+00	2.51E-09
NH3	8.59E-07	1.60E-04	9.01E-05	1.07E-06	3.96E-05	2.42E-04	1.55E-08	1.05E-07	2.65E-05	5.06E-10	0.00E+00	2.98E-09
p_auss2_3.0	1.40E-09	1.35E-09	9.42E-07	1.80E-09	1.98E-09	3.14E-05	t	t	t	1.31E-13	0.00E+00	4.74E-06
qap5	2.53E-06	1.07E-03	4.60E-05	2.44E-06	0.00E+00	3.08E-03	5.52E-07	0.00E+00	7.87E-09	7.91E-08	0.00E+00	1.89E-09
qap6	2.51E-06	2.34E-04	1.38E-03	2.52E-06	1.32E-07	9.92E-04	9.87E-06	1.34E-06	7.16E-04	4.10E-08	0.00E+00	1.08E-07
qap7	2.49E-06	1.47E-04	2.77E-04	2.50E-06	6.07E-08	9.73E-04	8.58E-06	4.14E-07	5.16E-04	3.56E-08	0.00E+00	6.62E-08
qap8	2.48E-06	3.24E-04	3.59E-04	2.49E-06	4.89E-08	1.32E-03	2.96E-06	1.15E-07	3.29E-04	4.88E-08	0.00E+00	4.96E-08
qap9	2.39E-06	1.08E-04	7.38E-03	2.48E-06	1.82E-08	9.18E-04	3.48E-06	4.03E-08	1.76E-04	1.49E-07	0.00E+00	2.91E-08
qap10	2.18E-06	1.85E-04	1.57E-02	2.48E-06	2.24E-07	1.51E-03	6.65E-06	2.42E-08	3.41E-04	1.05E-07	0.00E+00	3.48E-08
sensor_500	1.92E-07	1.02E-03	2.06E-04	1.93E-07	3.22E-04	1.77E-06	2.94E-07	3.50E-09	7.60E-06	5.85E-08	0.00E+00	4.41E-10
shmup4	2.16E-08	1.00E-06	6.89E-03	2.46E-08	5.16E-08	2.19E-05	2.36E-08	3.03E-06	3.19E-01	3.59E-09	0.00E+00	4.48E-06
theta1	9.96E-06	1.51E-04	6.33E-06	9.99E-06	8.87E-07	1.59E-05	7.71E-06	4.39E-08	5.09E-06	2.83E-08	0.00E+00	1.60E-09
theta2	1.00E-05	1.64E-04	6.94E-06	9.97E-06	2.48E-08	2.23E-05	2.02E-13	2.25E-07	9.09E-06	7.41E-09	0.00E+00	4.11E-10
theta3	9.97E-06	5.24E-04	6.30E-06	9.95E-06	5.08E-09	2.98E-05	2.30E-06	2.82E-08	1.46E-06	2.88E-08	0.00E+00	9.03E-10
theta4	1.00E-05	2.57E-04	9.27E-07	9.94E-06	1.94E-09	2.35E-05	6.13E-13	1.54E-08	1.12E-04	1.01E-08	0.00E+00	3.56E-10
theta5	9.99E-06	2.65E-04	5.10E-06	9.99E-06	3.32E-09	2.97E-05	3.20E-13	5.96E-08	1.06E-03	7.99E-09	0.00E+00	2.74E-10
theta6	9.99E-06	4.24E-04	9.51E-06	9.96E-06	1.11E-09	2.73E-05	6.65E-13	4.83E-08	7.14E-04	7.71E-09	0.00E+00	2.78E-10
theta12	3.09E-05	1.80E-03	4.12E-05	1.00E-05	1.39E-09	1.83E-05	5.02E-06	1.15E-09	5.31E-06	1.74E-08	0.00E+00	2.12E-09
theta102	1.08E-04	3.02E-03	6.78E-04	9.98E-06	4.08E-09	2.53E-05	1.12E-06	7.03E-10	9.80E-08	2.80E-09	0.00E+00	2.70E-09
theta123	2.24E-05	8.26E-04	2.04E-05	9.95E-06	2.22E-09	3.50E-05	3.86E-07	6.21E-09	5.29E-08	1.90E-08	0.00E+00	2.78E-09