An Enhanced ADMM-based Interior Point Method for Linear and Conic Optimization

Qi Deng\textsuperscript{1} Qing Feng\textsuperscript{2} Wenzhi Gao\textsuperscript{3} Dongdong Ge\textsuperscript{1} Bo Jiang\textsuperscript{4}\textsuperscript{*} Yuntian Jiang\textsuperscript{4} Jingsong Liu\textsuperscript{4} Tianhao Liu\textsuperscript{4} Chenyu Xue\textsuperscript{4} Yinyu Ye\textsuperscript{3} Chuwen Zhang\textsuperscript{4}

\textsuperscript{1}Shanghai JiaoTong University, \textsuperscript{2}Cornell University, \textsuperscript{3}Stanford University, \textsuperscript{4}Shanghai University of Finance and Economics

April 9, 2024

Abstract

The ADMM-based interior point (ABIP, Lin et al. 2021) method is a hybrid algorithm that effectively combines interior point method (IPM) and first-order methods to achieve a performance boost in large-scale linear optimization. Different from traditional IPM that relies on computationally intensive Newton steps, the ABIP method applies the alternating direction method of multipliers (ADMM) to approximately solve the barrier penalized problem. However, similar to other first-order methods, this technique remains sensitive to condition number and inverse precision. In this paper, we provide an enhanced ABIP method with multiple improvements. Firstly, we develop an ABIP method to solve the general linear conic optimization and establish the associated iteration complexity. Secondly, inspired by some existing methods, we develop different implementation strategies for ABIP method, which substantially improve its performance in linear optimization. Finally, we conduct extensive numerical experiments in both synthetic and real-world datasets to demonstrate the empirical advantage of our developments. In particular, the enhanced ABIP method achieves a 5.8x reduction in the geometric mean of run time on 105 selected LP instances from Netlib, and it exhibits advantages in certain structured problems such as SVM and PageRank. However, the enhanced ABIP method still falls behind commercial solvers in many benchmarks, especially when high accuracy is desired. We posit that it can serve as a complementary tool alongside well-established solvers.

Keywords: Linear optimization; Conic optimization; ADMM; Interior point method; Implementation improvement; Iteration complexity

1 Introduction

In this paper, we consider the following linear conic optimization problem with standard primal (P) and dual (D) forms:

\[
\begin{align*}
\min_{x} & \quad c^T x \\
\text{s.t.} & \quad A x = b \\
\end{align*}
\]

\[
\begin{align*}
\max_{y} & \quad b^T y \\
\text{s.t.} & \quad A^T y + s = c \\
& \quad x \in K \\
& \quad s \in K^* 
\end{align*}
\]

\textsuperscript{*}Correspondence to jiang.bo@mail.shufe.edu.cn.
where \( c \in \mathbb{R}^n, \ b \in \mathbb{R}^m, \ A \in \mathbb{R}^{m \times n}, \ K \) is a nonempty, closed, and convex cone with its dual cone defined as \( K^+ = \{ y : x^T y \geq 0, \forall x \in K \} \). In fact, this problem has strong modeling power, which includes linear optimization (LP), second-order cone optimization (SOCP), and semidefinite optimization (SDP) as special cases. Moreover, many important problems, such as quadratic optimization (QP) and quadratically constrained QP (QCQP), can be translated to equivalent conic formulations described by (1) (Alizadeh and Goldfarb, 2003).

Due to the wide applications in engineering and data science, developing efficient and accurate algorithms for linear and conic optimization has been a central topic in the optimization field in the past few decades. The traditional method for solving problem (1) is interior point method (IPM), which resorts to a sequence of log-barrier penalty subproblems and requires one step of Newton’s method to solve each subproblem (Nesterov and Nemirovskii, 1994). Open-source and commercial solvers based on IPM, such as SeDuMi (Sturm, 1999), SDPT3 (Toh et al., 1999), MOSEK (ApS, Mosek, 2019), Gurobi (Gurobi Optimization, LLC, 2022), COPT (Ge et al., 2022), are well-developed and have received great success in practice. Recently, new variants of IPM have been proposed (Pougkakiotis and Gondzio, 2022; Cipolla and Gondzio, 2023). These algorithms show promising improvements compared to the traditional IPM, and their algorithm frameworks still fall in the scope of solving regularized Newton equations. However, despite the fact that IPM can achieve fast convergence to high-accuracy solution, the computational cost can be a major concern, as solving a sequence of Newton equations can be highly expensive for large-scale or high-dimensional problems.

Compared to IPM, first-order methods are considered to be more scalable due to the low per-iteration cost and avoidance of solving Newton’s equation. Initial attempts in this direction start by replacing the matrix decomposition with iterative methods, and it is shown that convergence of IPM can be preserved even if the Newton system is not accurately solved (Zhou and Toh, 2004; Bellavia and Pieraccini, 2004; Lu et al., 2006; Al-Jeiroudi and Gondzio, 2009; Zanetti and Gondzio, 2023). Lately, there has been a growing interest in developing first-order methods for solving large-scale LP or conic optimization (Yang et al., 2015; O’Donoghue et al., 2016; Applegate et al., 2021a). In particular, O’Donoghue et al. (2016) develop the Splitting Conic Solver (SCS) for general conic LP, which applies the alternating direction method of multipliers (ADMM) (Boyd et al., 2011) to solve the homogeneous self-dual (HSD) reformulation (Ye et al., 1994) of the conic problem (1). The numerical results in O’Donoghue et al. (2016) showcase the superior performance of SCS over traditional IPM for several large-scale conic problems. Sopasakis et al. (2019) present a new Douglas-Rachford splitting method for solving the HSD system, which uses the quasi-Newton directions, such as restarted Broyden directions, and Anderson’s acceleration to further improve the convergence performance.

For LP problems, Lin et al. (2021) propose the ADMM-based Interior Point (ABIP) method, which can be viewed as a hybrid algorithm of the path-following IPM and ADMM. More specifically, it constructs a sequence of HSD reformulation of the LP problem with diminishing log-barrier parameters and uses ADMM to approximately solve each subproblem associated with a fixed log-barrier parameter. Therefore, it is expected to inherit some merits from both methods. For ease of exposition, we use “ABIP” and “the ABIP method” interchangeably in the rest of the paper. Very recently, Applegate et al. (2021a) apply the primal-dual hybrid gradient (PDHG) (Chambolle and Pock, 2011) method to solve LP based on its saddle point formulation. Applegate et al. (2021b) further propose a practical first-order method for LP (PDLP), which is an enhanced version of PDHG by combining several advanced implementation techniques. It is shown in Applegate et al. (2021b) that PDLP even outperforms a commercial LP solver in a large-scale application, i.e., the PageRank problem.

While earlier papers consider linear optimization, some recent effort has been made in developing first-order methods for QP. Stellato et al. (2020) present OSQP, a convex quadratic opti-
mization solver based on ADMM, which is competitive against commercial solvers. Garstka et al. (2021a) propose a new conic operator splitting method (COSMO) solver, which extends ADMM to deal with more general conic-constrained QP. By using chordal decomposition and some new clique merging techniques, they significantly improve the algorithm performance on solving some large-scale SDP. O’Donoghue (2021) extend the SCS framework (O’Donoghue et al., 2016) to solve quadratic conic optimization based on a more general formulation of the linear complementarity problem. The new implementation (O’Donoghue et al., 2023) has shown a solid empirical advantage in the infeasible problem while preserving great efficiency in feasible problems.

Contributions. In this paper, we continue the development of ABIP in Lin et al. (2021) along several new directions. Our contributions can be summarized as follows.

First, we present an important extension of ABIP such that the new solver can directly handle more general conic constraints. Theoretically, we show that the extended ABIP obtains an $O((1/\epsilon) \log (1/\epsilon))$ complexity bound, thus generalizing the complexity results previously known for LP (Lin et al., 2021) to a more general conic setting. For practical implementation, we show that the proximal operator associated with the log-barrier subproblem in ABIP can be efficiently computed. For several important applications in machine learning, such as LASSO and SVM, we develop customized linear system solvers to further accelerate ABIP for specific large-scale problems. We use extensive experiments on both synthetic and real-world datasets to show that enhanced ABIP compares favorably against many popular open-source and commercial solvers.

Second, we significantly improve the practical performance of ABIP for LP by developing several acceleration strategies. Those strategies were mostly motivated and adapted from the techniques used in the previous literature, including the adaptive strategy for choosing barrier parameter $\mu$, restart scheme, new inner loop convergence criteria, half update in ADMM, Presolve and preconditioning, and a tailored acceleration strategy for null objective problems. We further propose a decision tree-based approach to efficiently integrate those new strategies. With all the mentioned acceleration techniques, our numerical experiments show that the enhanced ABIP achieves a 5.8x reduction in the geometric mean of run time on 105 selected problem instances from Netlib. In large-scale applications such as the (staircase) PageRank problem, the enhanced ABIP is highly competitive and often outperforms PDLP (Applegate et al., 2021b). It is worth noting that the enhanced ABIP still falls behind commercial solvers across a variety of benchmarks, especially when a highly accurate solution is desired. This limitation seems inherent to all first-order methods, and addressing it remains a subject for future research.

Organizing the paper. This paper proceeds as follows. Section 2 generalizes ABIP to solve the convex conic optimization. Section 3 develops new strategies to further accelerate the practical performance of ABIP for LP. Section 4 conducts a detailed and extensive experimental study to demonstrate the empirical advantage of ABIP.

Notation and terminology. We use bold-face letters to denote matrices (i.e., $A$, $Q$) and vectors (i.e., $x$, $y$). Let $0$ be a vector of zeros, with its dimensionality unspecified, whenever it is clear from the context. Let $\mathbb{R}^n$ be the $n$-dimensional Euclidean space. We use $x \succeq y$ to denote the element-wise inequality $x_i \geq y_i$ for all $i$. The nonnegative orthant is defined by $\mathbb{R}^n_+ = \{x \mid x \succeq 0, x \in \mathbb{R}^n\}$. The regular second-order cone and the rotated second-order cone are denoted by $\text{SOC}_{1+n} = \{(t,y) \in \mathbb{R}_+ \times \mathbb{R}^n : \|y\| \leq t \}$ and $\text{RSOC}_{2+n} = \{(t,s,y) \in \mathbb{R}_+^2 \times \mathbb{R}^n : ts \geq \frac{1}{2}\|y\|_2^2 \}$, respectively.
2 ABIP for conic optimization

The original ABIP is designed for LP only. In this section, we extend it to solve general linear conic optimization problems where $\mathcal{K}$ can be represented by the Cartesian product of different general convex cones.

Motivating problems. We consider the following convex quadratic problem:

$$\begin{align*}
\min_z & \quad \frac{1}{2} z^T P z + q^T z \\
\text{s.t.} & \quad \bar{A} z = \bar{b} \\
& \quad z \geq 0
\end{align*}$$

(2)

where $\bar{A} \in \mathbb{R}^{m \times n}$, and $P \in \mathbb{R}^{n \times n}$ is a positive semi-definite matrix. It has extensive applications in machine learning and data science, such as SVM (Cortes and Vapnik, 1995), LASSO (Tibshirani, 1996), and portfolio optimization (Markowitz, 1991). Due to the existence of the quadratic term in the objective function, problem (2) cannot be directly solved by ABIP. However, it can be reformulated into a linear conic optimization problem.

Let $P = \Lambda^T \Lambda$ be the factorization where $\Lambda \in \mathbb{R}^{r \times n}$, problem (2) can be expressed as the following problem:

$$\begin{align*}
\min & \quad \frac{1}{2} \bar{x}^T \bar{x} + q^T \bar{z} \\
\text{s.t.} & \quad \bar{A} \bar{z} = \bar{b} \\
& \quad \bar{x} = \Lambda \bar{z} \\
& \quad \bar{z} \geq 0
\end{align*}$$

which can be further transformed into the conic form:

$$\begin{align*}
\min_{\eta, \nu, \bar{x}, \bar{z}} & \quad \nu + q^T \bar{z} \\
\text{s.t.} & \quad \eta = 1 \\
& \quad \bar{A} \bar{z} = \bar{b} \\
& \quad \bar{x} = \Lambda \bar{z} \\
& \quad \eta \nu \geq \frac{1}{2} \bar{x}^T \bar{x} \\
& \quad \bar{z} \geq 0
\end{align*}$$

$$(\eta, \nu, \bar{x}) \in \text{RSOC}_{2^+ \times r}, \bar{z} \in \mathbb{R}^n_+$$

where $I_r$ is $r$-dimensional identity matrix.

It should be noted that, although we use convex quadratic optimization as a motivating example, the conic formulation (1), as well as our subsequent algorithmic development and convergence analysis, still hold for general convex conic optimization such as SDP. We further provide an extensive numerical study under the SOCP setting in Section 4, while a more sophisticated empirical discussion beyond this setting will be left for future work.

HSD embedding for conic optimization. Before extending ABIP to linear conic optimization, we briefly describe the HSD embedding technique for convex conic optimization (Luo et al., 1997; Zhang, 2004). Given an initial solution $x^0 \in \text{int} \mathcal{K}$, $s^0 \in \text{int} \mathcal{K}^*$ and $y^0 \in \mathbb{R}^m$, we can derive
the following HSD embedding formulation:

$$
\begin{align*}
\min & \quad \beta((x^0)^T s^0 + 1) \theta \\
\text{s.t.} & \quad Ax - b + r_p \theta = 0 \\
& \quad -A^T y + c^T r_d + r_g \theta = s \\
& \quad b^T y - c^T x + r_d \theta = \kappa \\
& \quad -r_p^T y - r_d^T x - r_g \theta = -(x^0)^T s^0 - 1 \\
& \quad x \in K, s \in K^*, \tau, \kappa \geq 0
\end{align*}
$$

where $r_p = b - Ax^0$, $r_d = s^0 - c + A^T y^0$, $r_g = 1 + c^T x^0 - b^T y^0$. It is easy to check that the constraint matrix of (3) is skew-symmetric, and hence the above problem is self-dual. The conic HSD embedding formulation (3) enjoys several attractive properties (Zhang, 2004; Luo et al., 1997), which we list without proof as follows.

**Proposition 1.** Problem (3) is strictly feasible. Specifically, there is a strictly feasible solution $(y, x, \tau, \theta, s, \kappa)$ such that $y = y^0$, $x = x^0$, $\tau = 1$, $\theta = 1$, $s = s^0$, and $\kappa = 1$.

**Proposition 2.** Problem (3) has a maximally complementary optimal solution, denoted by $(y^*, x^*, \tau^*, \theta^*, s^*, \kappa^*)$, such that $\theta^* = 0$ and $(x^*)^T s^* + \tau^* \kappa^* = 0$. Moreover, the following statements hold:

- If $\tau^* > 0$, then $x^*/\tau^*$ is an optimal solution for (P), and $(y^*/\tau^*, s^*/\tau^*)$ is an optimal solution for (D).
- If $\kappa^* > 0$, then either $c^T x^* < 0$ or $b^T y^* > 0$. In the former case (D) is infeasible, and in the latter case (P) is infeasible.

The first proposition shows that one can construct a strictly feasible solution to problem (3) based on a feasible solution to (1). The second proposition shows that, one can obtain the optimal solution to (1) by solving (3). Meanwhile, the HSD embedding technique can detect the infeasibility of a convex conic optimization problem, which is also inherited by ABIP. We remark that there are other first-order algorithms that can handle infeasibility (Banjac et al., 2019; Applegate et al., 2021), which go beyond our scope in this paper since we mainly focus on how to solve the feasible problem efficiently.

### 2.1 Algorithmic development

In this subsection, we present the algorithm designed for linear conic optimization. Inspired by ABIP, we first rewrite problem (3) as:

$$
\begin{align*}
\min & \quad \beta((x^0)^T s^0 + 1) \theta + 1(r = 0) + 1(\xi = -(x^0)^T s^0 - 1) \\
\text{s.t.} & \quad Qu = v \\
& \quad (x, s, \tau, \kappa) \in K \times K^* \times \mathbb{R}_+ \times \mathbb{R}_+, \ y, \theta \text{ free},
\end{align*}
$$

where $Q$, $u$, and $v$ are defined by,

$$
Q = \begin{bmatrix}
0_{n \times m} & A & -b & r_p \\
-A^T & 0_{n \times m} & c & r_d \\
b^T & -c^T & 0 & r_g \\
-r_p^T & r_d^T & -r_g & 0
\end{bmatrix},
$$

$u = \begin{bmatrix} y \\ x \\ \tau \\ \theta \end{bmatrix}$, and $v = \begin{bmatrix} r \\ s \\ \kappa \end{bmatrix}$.

Then, we follow the framework of IPM and introduce the log-barrier penalty with parameter $\mu > 0$ to the conic constraints, and move it to the objective function. Specifically, let $F(x)$ and $G(s)$ be
the log-barrier functions associated with \( x \in K \) and \( s \in K^* \), respectively, and define

\[
B(u, v, \mu) = \beta((x^0)^T s^0 + 1)\theta + 1(r = 0) + 1(\xi = -(x^0)^T s^0 - 1) \\
+ \mu F(x) + \mu G(s) - \mu \log \tau - \mu \log \kappa.
\]  

(5)

Following Lin et al. (2021), our enhanced ABIP has a double-loop structure. The details of the outer loop and the inner loop are discussed as follows.

**Outer loop.** At each outer iteration \( k \), ABIP constructs the following log-barrier penalized problem with respect to \( \mu^{(k)} \):

\[
\min B \left( u, v, \mu^{(k)} \right) \\
\text{s.t.} \ Qu = v.
\]  

(6)

Here, the sequence of \( \{\mu^{(k)}\} \) is a diminishing sequence, i.e., \( \lim_{k \to \infty} \mu^{(k)} = 0 \). In theoretical analysis, we let \( \mu^{(k+1)} = \phi \mu^{(k)} \) with \( \phi \in (0, 1) \).

**Inner loop.** In the inner loop associated with \( \mu^{(k)} \), ABIP use ADMM to approximately solves problem (6). With variable splitting, (6) is equivalent to the following problem:

\[
\min 1(Q\tilde{u} = \tilde{v}) + B \left( u, v, \mu^{(k)} \right) \\
\text{s.t.} \ (\tilde{u}, \tilde{v}) = (u, v).
\]  

(7)

Denote the augmented Lagrangian function by

\[
\mathcal{L}_\beta(\tilde{u}, \tilde{v}, u, v, \mu^{(k)}, p, q) = 1(Q\tilde{u} = \tilde{v}) + B \left( u, v, \mu^{(k)} \right) \\
- \langle \beta(p, q), (\tilde{u}, \tilde{v}) - (u, v) \rangle + \frac{\beta}{2} ||(\tilde{u}, \tilde{v}) - (u, v)||^2,
\]

where \( \beta > 0 \) is the same parameter as the one in (4), and \( p, q \) are the Lagrangian multipliers associated with the linear constraints. Then, in the \( i \)-th iteration, the update rule of ADMM is as follows:

\[
(\tilde{u}^{(k)}_{i+1}, \tilde{v}^{(k)}_{i+1}) = \text{argmin}_{\tilde{u}, \tilde{v}} \mathcal{L}_\beta(\tilde{u}, \tilde{v}, u, v, \mu^{(k)}, p^{(k)}, q^{(k)}) \\
= \Pi_{Q\tilde{u} = \tilde{v}}(p^{(k)}_{i} + p^{(k)}_{i}, v^{(k)}_{i} + q^{(k)}_{i}),
\]  

(8)

\[
(u^{(k)}_{i+1}, v^{(k)}_{i+1}) = \text{argmin}_{u, v} \mathcal{L}_\beta(u^{(k)}_{i+1}, v^{(k)}_{i+1}, u, v, \mu^{(k)}, p^{(k)}_{i}, q^{(k)}_{i})
\]  

(9)

\[
(p^{(k)}_{i+1}, q^{(k)}_{i+1}) = (p^{(k)}_{i}, q^{(k)}_{i}) - (u^{(k)}_{i+1}, v^{(k)}_{i+1}) + (u^{(k)}_{i}, v^{(k)}_{i}),
\]  

(10)

where \( \Pi_S(x) \) denotes the Euclidean projection of \( x \) onto the set \( S \). The above update rule seems to be complicated at the first glance. However, Lin et al. (2021) show that, for LP, the ADMM iteration can be substantially simplified by a customized initialization technique. We prove in the following theorem that this attractive feature still holds for linear conic optimization. The proof can be easily extended from Lin et al. (2021), and we provide it in Appendix A for self-completeness.

**Theorem 1.** At the \( k \)-th outer iteration of ABIP, suppose that we initialize \( p^{(k)}_0 = v^{(k)}_0 \) and \( q^{(k)}_0 = u^{(k)}_0 \). Then, it holds that \( p^{(k)}_i = v^{(k)}_i \) and \( q^{(k)}_i = u^{(k)}_i \), for all \( i \geq 0 \).

Theorem 1 implies that, by carefully choosing the initial point at the beginning of the inner loop, we can eliminate the dual variables \( p^{(k)}, q^{(k)} \) safely. Moreover, following a similar argument of Lin et al. (2021), one can use the skew-symmetry property of \( Q \) to obtain a simpler update for
the update criterion is satisfied:

\[ \bar{u}^{(k+1)} = (\mathbf{I} + \mathbf{Q})^{-1}(u_i^{(k)} + v_i^{(k)}), \]

\[ u_i^{(k+1)} = \text{prox}_{\frac{B}{\mu}}(u_i^{(k)} - v_i^{(k)}), \]

\[ v_i^{(k+1)} = v_i^{(k)} - \bar{u}^{(k)} + u_i^{(k)}, \]

where \( \bar{B}_\mu(\cdot) \) is the log-barrier function \( B(\mathbf{u}, \mathbf{v}, \mu) \) restricted to the \( \mathbf{u} \) part, that is,

\[ \bar{B}_\mu(\mathbf{u}) = \beta((\mathbf{x})^T s^0 + 1)\theta + \mathbb{I}(\xi = -((\mathbf{x})^T s^0 - 1) + \mu F(\mathbf{x}) - \mu \log \tau \]

The proximal operator is defined by \( \text{prox}_f(\mathbf{x}) = \arg\min_u \{ f(\mathbf{u}) + \frac{1}{2}\| \mathbf{u} - \mathbf{x} \|^2 \}. \)

It is clear that, ABIP needs to the inverse of \( I + \mathbf{Q} \), which can be computed before the algorithm proceed. The remaining computational bottleneck is to compute the proximal operator. In Section 2.4, we show that the proximal operator associated with several widely-used convex cones can be solved efficiently.

**Termination criteria.** For the inner loop, we run ADMM iteration until the following termination criterion is satisfied:

\[ \|\mathbf{Q} u_i^{(k)} - v_i^{(k)}\|^2 \leq \mu^{(k)}. \]

(14)

For the outer loop, we theoretically terminate the algorithm when \( \mu^{(k)} < \epsilon \). In the implementation, however, a different termination criterion is employed to align with other solvers, which will be introduced later.

### 2.2 Convergence analysis

In this subsection, we develop the iteration complexity of ABIP, for which we generalize the analysis of Lin et al. (2021) to linear conic optimization. To begin with, we outline the key ingredients of the analysis. First, we prove that the iterates generated by ADMM in each inner loop are uniformly bounded above. Then, based on this property, we show that ABIP exhibits a linear convergence to the optimal solution associated with each subproblem. Finally, we derive the total iteration complexity of ABIP under the path-following framework.

Let \( (u_i^*, v_i^*) \) be the optimal solution to subproblem (7). In the following proposition, we show that the iterates generated by ADMM will converge to \( (u_i^*, v_i^*) \). The proof can be extended from Lin et al. (2021), and thus is omitted in the paper.

**Proposition 3.** For a fixed \( k \in \mathbb{N}_+ \), the sequence \( \{\|u_i^{(k)} - u_i^*\|^2 + \|v_i^{(k)} - v_i^*\|^2\}_{i \geq 0} \) is monotonically decreasing and converges to 0.

In the next proposition, we further prove that the sequence \( \{\|u_i^{(k)} - u_i^*\|^2 + \|v_i^{(k)} - v_i^*\|^2\}_{i \geq 0} \) is uniformly bounded. With Proposition 3, it suffices to show that the initial point \( \{\|u_i^{(0)} - u_i^*\|^2 + \|v_i^{(0)} - v_i^*\|^2\} \) is bounded.

**Proposition 4.** The sequence \( \{\|u_i^{(k)} - u_i^*\|^2 + \|v_i^{(k)} - v_i^*\|^2\} \) is uniformly bounded.

**Proof.** We first recall an important fact that the set of central path points \( \{(u_i^*, v_i^*)\}_{k \geq 1} \) is uniformly bounded. That is, there exists a constant \( C_1 \) such that

\[ \|u_i^*\|^2 + \|v_i^*\|^2 \leq C_1. \]
Let \((\mathbf{u}^*, \mathbf{v}^*)\) be the limit point of \(\{(\mathbf{u}_k^i, \mathbf{v}_k^i)\}_{k \geq 1}\), then \((\mathbf{u}^*, \mathbf{v}^*)\) also satisfies \(||\mathbf{u}^*||^2 + ||\mathbf{v}^*||^2 \leq C_1\).

Let \(N_k\) denote the number of inner iterations in the \(k\)-th outer loop. With Proposition 3 and the inner loop termination criterion (14), we have \(N_k < \infty\) for any fixed \(k\).

Now we claim that when \(k \to \infty, ||u_{N_k}^{(k)} - u^*||^2 + ||v_{N_k}^{(k)} - v^*||^2 \to 0\). Otherwise, there exists \(\delta > 0\) and a subsequence \(\{k_i\}_{i \in \mathbb{N}^+}\) with \(k_i \uparrow \infty\) as \(t \to \infty\) such that \(||u_{N_k}^{(k_i)} - u^*||^2 + ||v_{N_k}^{(k_i)} - v^*|| > \delta\) for all \(t \in \mathbb{N}^+\), and there exists an integer \(T\) such that \(||u_{N_k}^{(k_i)} - u^*||^2 + ||v_{N_k}^{(k_i)} - v^*|| > \delta/2\) for all \(t > T\), which contradicts with (14). Therefore, \(||u_{N_k}^{(k)} - u^*||^2 + ||v_{N_k}^{(k)} - v^*|| \to 0\) when \(k \to \infty\), which also implies that there exists \(C_2 > 0\) such that

\[||u_{N_k}^{(k)}||^2 + ||v_{N_k}^{(k)}||^2 \leq C_2.\]

Since \(u_{N_k}^{(k)} = u_0^{(k+1)}\) and \(v_{N_k}^{(k)} = v_0^{(k+1)}\), we have proven the boundedness of the sequence \(\{v_0^{(k)}\}\). Moreover, by the results of Proposition 3, we have that \(\{(u_0^{(k)}, v_0^{(k)})\}\) is uniformly bounded. \(\square\)

With the above results, we give an upper bound on the number of ADMM iterations for each inner loop in the next proposition, and present the total iteration complexity of ABIP in Theorem 2.

**Proposition 5.** The number of ADMM iterations in each inner loop associated with \(\mu^{(k)}\), denoted by \(N_k\), satisfies

\[N_k \leq \log \left(\frac{4C_1(1 + ||\mathbf{Q}||^2)}{\mu^{(k)}}\right) \left[\log \left(1 + \min \left\{\frac{1}{C_3}, 2C_D C_3 \beta\right\}\right)\right]^{-1},\]

where

\[C_3 = \left[1 + \frac{12 \lambda_{\max}(\mathbf{A}^T \mathbf{A})}{\lambda_{\min}(\mathbf{A}^T \mathbf{A})} \max\{1, ||\mathbf{c}||^2, ||\mathbf{r}_d||^2\}\right] \cdot \left[1 + \frac{6}{||\mathbf{r}_p||^2} \max\{||\mathbf{b}||^2, ||\mathbf{A}||^2\}\right].\]

and \(1/C_D > 0\) is the strong convexity parameter of the log-barrier function.

The proof can be found in Appendix B. Now we are ready to develop the total iteration complexity of ABIP for solving linear conic optimization problems.

**Theorem 2.** Suppose that ABIP is terminated when \(\mu^{(k)} \leq \epsilon\), where \(\epsilon > 0\) is a pre-specified tolerance. It requires a total number of \(T_1 = \mathcal{O}\left(\log(1/\epsilon)\right)\) outer loops and \(T_2 = \mathcal{O}\left(\kappa_A^2 ||\mathbf{Q}||^2/\epsilon \cdot \log(1/\epsilon)\right)\) ADMM iterations, where \(\kappa_A := \lambda_{\max}(\mathbf{A}^T \mathbf{A})/\lambda_{\min}(\mathbf{A}^T \mathbf{A})\).

**Proof.** Proof. Note that ABIP is a double-loop algorithm. The outer loop is terminated when \(\mu^{(k)} \leq \epsilon\), where \(\epsilon > 0\) is a pre-specified tolerance level. It is easy to see that the number of outer iterations is

\[T_{\text{IPM}} = \left[\log(\mu^{(0)}/\epsilon)\right]/\log(1/\epsilon).\]

For the total number of ADMM iterations, we have the following bound:

\[T_{\text{ADMM}} = \sum_{k=1}^{T_1} N_k \leq \sum_{k=1}^{T_1} \log \left(\frac{4C_1(1 + ||\mathbf{Q}||^2)}{\mu^{(k)}}\right) \left[\log \left(1 + \min \left\{\frac{1}{C_3}, 2C_D C_3 \beta\right\}\right)\right]^{-1},\]

\[= \sum_{k=1}^{T_2} \log \left(\frac{4C_1(1 + ||\mathbf{Q}||^2)}{\mu^{(0)}/\epsilon}\right) \left[\log \left(1 + \min \left\{\frac{1}{C_3}, 2C_D C_3 \beta\right\}\right)\right]^{-1},\]

\[= \mathcal{O}\left(\kappa_A^2 ||\mathbf{Q}||^2/\epsilon \cdot \log(1/\epsilon)\right).\]

\(\square\)
With Theorem 2, we now make a comparison of iteration complexity between ABIP and PDLP in Table 1. It shows that ABIP is more sensitive to the condition number of the coefficient matrix compared with PDLP, which also motivates us to improve the practical performance of ABIP in the next section. It is important to note, however, that our result is established for general linear conic optimization, whereas PDLP’s analysis is specifically tailored for linear optimization.

To conclude this subsection, we further give the total arithmetic operations of ABIP, that is, $O(n^3 + (n^2 \kappa_A^3 \|Q\|^2/\epsilon) \log (1/\epsilon))$. The first term $O(n^3)$ comes from the matrix decomposition, and the second term is from the $O(n^2)$ arithmetic operations required by each ADMM iteration and the total number of ADMM iterations, that is, $O((\kappa_A^2 \|Q\|^2/\epsilon) \log (1/\epsilon))$.

| Algorithm | Problem            | Complexity                                      |
|-----------|--------------------|-------------------------------------------------|
| ABIP      | Conic Optimization | $O(\kappa_A^2 \|Q\|^2/\epsilon \cdot \log (1/\epsilon))$ |
| PDLP (Applegate et al., 2021a) | Linear Optimization | $O(\sigma_{\max}(A)/\sigma_{\min}(A) \cdot \log (1/\epsilon))$ |

Table 1: Iteration Complexity of ABIP and PDLP. Note that $\kappa_A$, $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ represent the condition number of $A$, the minimum nonzero singular value of $A$, and the maximum singular value of $A$, respectively. The matrix $Q$ is the coefficient matrix of the HSD formulation in (4).

2.3 Solving the linear system

In this subsection, we discuss how to efficiently solve the linear system of $(I + Q)\tilde{u} = w$ for some given $w$. In particular, we want to solve

$$
\begin{bmatrix}
M & h \\
-h^T & 1
\end{bmatrix}
\tilde{u} = w,
$$

where

$$
M = \begin{bmatrix}
I_m & A \\
-A^T & I_n
\end{bmatrix}
$$

and $h = \begin{bmatrix}
-b \\
c
\end{bmatrix}$.

To further simplify (15), one can follow Lin et al. (2021) and only consider a smaller linear system restricted to the $(\tilde{y}, \tilde{x})$ block variables of $\tilde{u}$:

$$
\begin{bmatrix}
\tilde{y} \\
\tilde{x}
\end{bmatrix} = M^{-1} \begin{bmatrix}
y + r \\
x + s
\end{bmatrix}.
$$

To solve this, the first approach, or the so-called direct method, is to perform a sparse permuted LDL$^T$ factorization of $M$ before the first iteration, and store the factors $L$ and $D$. Note that the factorization only needs to be computed one time, and the subsequent ADMM iterations can be carried out by solving much easier triangular and diagonal linear systems with the cached factors $L$ and $D$. The second approach, or the indirect method, is to approximately solve the system with conjugate gradient method (Wright et al., 1999). When the time or memory cost of factorizing $M$ is expensive, the indirect method is more favorable due to its simple and efficient update rule. Note that similar strategies have been employed in many other popular open-source solvers, such as OSQP (Stellato et al., 2020) and SCS (O’Donoghue et al., 2016).

Although the general-purpose approaches designed for solving linear systems are convenient, they may still have limitations since the inner structure of the problem is hardly exploited. This further prohibits the use of ABIP for large-scale problems in machine learning and data science. Therefore, to accelerate the performance of ABIP for such applications, it is desirable to use a more specialized linear system solver that can be adaptive to the problem structure. For some important problems in machine learning, such as LASSO and SVM, we present novel implemen-
tation strategies to solve linear systems, which significantly improve the scalability of ABIP for large-scale problems. The more details are presented in Appendix D.

2.4 Solving the proximal problem

In this subsection, we discuss how to solve the proximal problem in (12). Lin et al. (2021) have shown that, the proximal problem associated with the nonnegative orthant exhibits a closed-form solution. As a complement, we show that the proximal problems associated with other convex cones, including second-order cone and semidefinite cone, can also be computed efficiently.

2.4.1 Second-order cone

We consider the following proximal problem:

$$x^* = \text{argmin}_{x \in \text{int}K} \lambda F(x) + \frac{1}{2} \|x - \zeta\|^2,$$  \hspace{1cm} (17)

where $\lambda > 0$ and $\zeta$ is a column vector. In this part, we show that the proximal problems associated with standard second-order cone (SOC) and rotated second-order cone (RSOC) have closed-form solutions. We present the main results here and leave the technical details to Appendix C.

**Proximal problem with standard SOC.** For $x = (t, \bar{x}) \in \text{SOC}$, the log-barrier function is defined by $F(x) = -\log(t^2 - \bar{x}^T \bar{x})$. For convenience, we let $\zeta = (\zeta_t, \zeta_{\bar{x}})^T$ and discuss the following two situations. When $\zeta_t = 0$, we have

$$t^* = \sqrt{2\lambda + \frac{1}{4} \zeta_t^T \zeta_{\bar{x}}}, \quad x^* = \frac{1}{2} \zeta_{\bar{x}}.$$

When $\zeta_t \neq 0$, we define $\rho_1$ and $\rho_2$, respectively, by

$$\rho_1 = \frac{\gamma - \sqrt{\gamma^2 - 16}}{2} < 2, \quad \rho_2 = \frac{\gamma + \sqrt{\gamma^2 - 16}}{2} > 2,$$

where

$$\gamma = \frac{\zeta_t^2 - \zeta_{\bar{x}}^2}{\lambda} + \sqrt{\left(\frac{\zeta_t^2 - \zeta_{\bar{x}}^2}{\lambda}\right)^2 + 4 \left(\frac{4\zeta_t^2 + 4\zeta_{\bar{x}}^2}{\lambda}\right) + 16}.$$

Then, we can obtain the solution

$$t^* = \frac{\rho}{\rho - 2} \zeta_t, \quad x^* = \frac{\rho}{\rho + 2} \zeta_{\bar{x}},$$

where $\rho = \rho_1$ if $\zeta_t < 0$ and $\rho = \rho_2$ otherwise.

**Proximal problem with standard RSOC.** Let $x = (\eta, \nu, \bar{x}) \in \text{RSOC}$ and define the log-barrier function by $F(x) = -\log(\eta \nu - \frac{1}{2} \bar{x}^T \bar{x})$. For convenience, we let $\zeta = (\zeta_\eta, \zeta_\nu, \zeta_{\bar{x}})^T$ and discuss the following two situations. If $\zeta_\eta + \zeta_\nu = 0$, we have

$$\eta^* = \frac{\zeta_\eta + \sqrt{\zeta_\eta^2 + 4 \left(\lambda + \frac{1}{2} \zeta_\nu^T \zeta_{\bar{x}}\right)}}{2}, \quad \nu^* = \frac{-\zeta_\nu + \sqrt{\zeta_\nu^2 + 4 \left(\lambda + \frac{1}{2} \zeta_\eta^T \zeta_{\bar{x}}\right)}}{2}, \quad x^* = \frac{1}{2} \zeta_{\bar{x}}.$$

When $\zeta_\eta + \zeta_\nu \neq 0$, we define $\rho_1$ and $\rho_2$ as

$$\rho_1 = \frac{\gamma - \sqrt{\gamma^2 - 4}}{2} < 1, \quad \rho_2 = \frac{\gamma + \sqrt{\gamma^2 - 4}}{2} > 1,$$
where
\[ \gamma = \frac{2\zeta_\eta - \zeta_\nu \zeta_\nu}{2\lambda} + \sqrt{\left(\frac{2\zeta_\eta - \zeta_\nu \zeta_\nu}{2\lambda}\right)^2 + 4 \left(\frac{\zeta_\eta^2 + \zeta_\nu^2 + \zeta_\nu}{\lambda}\right) + 4}. \]

Then, we have
\[ \eta^* = \frac{\rho^2 \zeta_\eta + \rho \zeta_\nu}{(\rho + 1)(\rho - 1)}, \quad \nu^* = \frac{\rho \zeta_\eta + \rho^2 \zeta_\nu}{(\rho + 1)(\rho - 1)} \quad \text{and} \quad \bar{x}^* = \frac{\rho \zeta_\bar{x}}{\rho + 1}, \]
where \( \rho = \rho_1 \) if \( \zeta_\eta + \zeta_\nu < 0 \) and \( \rho = \rho_2 \) otherwise.

### 2.4.2 Semidefinite cone

In this part, we consider the proximal problem associated with the semidefinite cone (SDC), which can be represented by \( \mathcal{K} = \{ X : X \in \mathbb{R}^{n \times n}, X \succeq 0, X = X^T \} \). The log-barrier function is defined by \( F(X) = -\log \det(X) \).

We consider the following proximal subproblem:
\[ X^* = \arg\min_{X \succ 0, X = X^T} -\lambda \log \det(X) + \frac{1}{2} \|X - A\|^2, \tag{18} \]

where \( \lambda > 0, A \in \mathbb{R}^{n \times n} \) is a symmetric matrix, and \( \| \cdot \| \) represents matrix Frobenius norm, i.e., \( \|A\| = \sqrt{\text{tr}(A^T A)} \). By the optimality condition, problem (18) is equivalent to finding the root of the following equation:
\[ -\lambda(X)^{-1} + X - A = 0, \quad X \succ 0. \tag{19} \]

Since \( A \) is real symmetric, we have its spectral decomposition \( A = Q^T D Q \), where \( D = \text{diag}(d_i)_{i=1}^n \) is a diagonal matrix with eigenvalues \( d_i \), and \( Q \) is an orthogonal matrix. Then, we define a diagonal matrix
\[ E = \text{diag}(e_i)_{i=1}^n, \quad \text{where} \quad e_i = \frac{d_i + \sqrt{d_i^2 + 4 \lambda}}{2}, \quad \forall i = 1, 2, \ldots, n. \]

Moreover, it is easy to verify that
\[ \lambda E^{-1} + E - D = 0, \quad E \succ 0, \]
which further implies
\[ Q^T (\lambda E^{-1} + E - D) Q = \lambda Q^T E^{-1} Q + Q^T EQ - A = 0. \]

In other words, we have shown that \( Q^T EQ \) is a solution to equation (19). Due to the strong convexity, the optimal solution \( X^* \) to the proximal problem (18) must be unique, which is exactly given by \( X^* = Q^T EQ \).

### 3 Newly implemented strategies of ABIP for linear optimization

In this section, we introduce several new strategies that further improve ABIP’s performance for LP. Specifically, the LP problem has the following form:
\[ \begin{align*}
\min & \quad c^T x \\
s.t. & \quad Ax = b, \\
& \quad x \geq 0.
\end{align*} \tag{20} \]
We remark that, although these strategies are applicable to general conic optimization, our preliminary experiments show that their benefits are considerably more pronounced for LP. Consequently, our subsequent discussion will focus on the enhancements within the realm of LP. A comprehensive empirical study is deferred to Section 4.

3.1 An adaptive strategy for the barrier parameter \( \mu \)

Under the framework of ABIP, the barrier parameter \( \mu^{(k)} \) plays a critical role in balancing the subproblem’s convexity and global optimality of the obtained ADMM iterates. Moreover, a proper path driving \( \mu^{(k)} \) down to 0 benefits the convergence of ABIP by reducing the total number of both IPM iterations and ADMM iterations. In particular, we consider the following two popular adaptive methods for the barrier parameter update.

The aggressive strategy (Wächter and Biegler, 2006). Given \( 0 < \zeta < 1 \) and \( \eta > 1 \), the aggressive strategy updates by
\[
\mu^{(k+1)} = \min\{\zeta \mu^{(k)}, (\mu^{(k)})^\eta\}.
\]
This is often used at the beginning of the algorithm when the subproblem is well-behaved and allows an aggressive decrease in the barrier parameter. However, as ABIP progresses, this aggressive strategy will result in more but useless ADMM iterations to solve the subproblem since \( \mu \) will be quite small. Hence, under this case, we will switch to the following strategy to gradually decrease \( \mu \).

The LOQO strategy (Vanderbei, 1999). Given parameter \( 0 < \alpha < 1 \), the LOQO strategy updates by
\[
\mu^{(k+1)} = \mu^{(k)} \cdot \max \left\{ 0.1 \cdot \min \left\{ 0.05 \cdot \frac{1 - \phi}{\phi}, 2 \right\}^3, \alpha \right\},
\]
where \( \phi := (n+2) \cdot \min_{0 \leq i \leq n} \{ x_i s_i \}, \tau \kappa, \theta \xi \}/(x^T s + \kappa \tau + \theta \xi) \). This method is adapted from the well-known update rule in the quadratic optimization solver LOQO, where \( \phi \) serves as a measure of centrality and \( \mu^{(k+1)} \) is lower-bounded by \( \alpha \mu^{(k)} \) to avoid over-aggressive updates.

In our implementation, the two methods are integrated into a hybrid strategy that automatically switches between them. To be more specific, we switch from the aggressive strategy to the LOQO strategy when the barrier parameter \( \mu^{(k)} \leq 10^3 \cdot \epsilon \), where \( \epsilon \) is the pre-determined tolerance level.

3.2 Restart scheme

The restart scheme is a useful technique to improve the practical and theoretical performance of an optimization algorithm (Pokutta, 2020). Motivated by the acceleration performance of the restart scheme for solving large-scale linear optimization (Applegate et al., 2021b,a), we implement a restart scheme in the inner loop of ABIP.

In particular, let \( M \) be the accumulated number of ADMM iterations performed by ABIP so far, and \( M_k \) be the number of ADMM iterations for solving the inner problem (7) at the \( k \)-th outer loop. We observe that the restart scheme tends to be less effective in the initial stage of ABIP. Therefore, we employ this strategy only when the accumulated number of iterations \( M \) is sufficiently large, such as surpassing a pre-determined threshold \( M_T \). Once this scheme is triggered, the ADMM algorithm in the inner loop will be restarted with a fixed period \( F \). Namely, we restart the algorithm if
\[
M \geq M_T \quad \text{and} \quad (M_k + 1) \mod F = 0.
\]
Then, we use the average of historical iterates in the current restart cycle as the initial point of the next restart cycle. Details of the restart scheme are presented in Algorithm 1.

**Algorithm 1: ABIP with fixed frequency restart**

1. Initialize $u_0^{(0)}$ and $v_0^{(0)}$;
2. Set ADMM iteration counter $M ← 0$;
3. Set restart threshold $M_T$ and restart period $F$;
4. for $k = 0, 1, 2, \cdots$ do
5. \hspace{1em} $u_{\text{avg}}^{(k)} ← u_0^{(k)}$, $v_{\text{avg}}^{(k)} ← v_0^{(k)}$;
6. \hspace{1em} $M_k ← 0$;
7. \hspace{2em} while True do
8. \hspace{3em} if the inner termination condition is satisfied then
9. \hspace{4em} break;
10. \hspace{3em} end
11. \hspace{3em} $u_{M_k+1}^{(k)}, v_{M_k+1}^{(k)} ← \text{ADMM Step}(u_{M_k}^{(k)}, v_{M_k}^{(k)})$;
12. \hspace{3em} $u_{\text{avg}}^{(k)} ← u_{\text{avg}}^{(k)} + u_{M_k+1}^{(k)}$;
13. \hspace{3em} $v_{\text{avg}}^{(k)} ← v_{\text{avg}}^{(k)} + v_{M_k+1}^{(k)}$;
14. \hspace{3em} $M_k ← M_k + 1$;
15. \hspace{3em} $M ← M + 1$;
16. \hspace{3em} if $M ≥ M_T$ and $M_k \mod F = 0$ then
17. \hspace{4em} $u_{M_k+1}^{(k)} ← \frac{u_{\text{avg}}^{(k)}}{F}$, $v_{M_k+1}^{(k)} ← \frac{v_{\text{avg}}^{(k)}}{F}$;
18. \hspace{4em} $u_{\text{avg}}^{(k)} ← u_{M_k+1}^{(k)}$, $v_{\text{avg}}^{(k)} ← v_{M_k+1}^{(k)}$;
19. \hspace{3em} end
20. \hspace{3em} if the final termination criterion is satisfied then
21. \hspace{4em} return;
22. \hspace{3em} end
23. $u_0^{(k+1)} ← u_{M_k}^{(k)}$, $v_0^{(k+1)} ← v_{M_k}^{(k)}$;
24. end
25. end

To illustrate the acceleration performance of the restart strategy, we visualize the trajectory of ABIP for solving the SC50B instance from Netlib as an example in Figure 1. For ease of exposition, we limit our visualization to the first three dimensions of the candidate solutions. Specifically, Figure 1(a) shows that the original ABIP has a spiral trajectory, leading to a long time to converge to an optimal solution. However, ABIP progresses more aggressively after restart and is closer to the optimal solution than the original ABIP due to averaging the previous iterates, as shown in Figure 1(b). In fact, the restart strategy contributes to reducing almost 70% total ADMM iterations on the SC50B instance; see Table 2. Preliminary experiments also show that the restart strategy is more efficient when one hopes to obtain a high-accuracy solution, for example, $\epsilon = 10^{-6}$.

| # of ADMM iterations | ABIP | ABIP with restart |
|----------------------|------|-------------------|
| 309175               |      | 93313             |

Table 2: # of ADMM iterations for SC50B of ABIP and ABIP with restart strategy, $\epsilon = 10^{-6}$, direct method

### 3.3 New inner loop termination criteria

The original ABIP adopts the criterion (14) to terminate the inner loop. In its enhanced implementation, motivated by the restart strategy, we also compute the average of previous iterates $\bar{u}_i^{(k)} = \sum_{i=1}^{k} u_i^{(k)}/k$ and $\bar{v}_i^{(k)} = \sum_{i=1}^{k} v_i^{(k)}/k$, and adopt $\left\| Q u_i^{(k)} - \bar{v}_i^{(k)} \right\|_2^2 ≤ \mu^{(k)}$ as another stopping criterion when solving the $k$-th subproblem. When one of the two criteria is satisfied, we
terminate the inner loop and let the returned solution be the initial point of ADMM to solve the subproblem in the form of (7) corresponding to \( \mu^{(k+1)} \).

As mentioned before, we use a different criterion in the implementation to check the global convergence. Specifically, we stop the algorithm when the following condition is satisfied:

\[
\max (\epsilon'_\text{pres}, \epsilon'_\text{dres}, \epsilon'_\text{dgap}) \leq \epsilon,
\]

where \( \epsilon'_\text{pres}, \epsilon'_\text{dres}, \) and \( \epsilon'_\text{dgap} \) represent the primal residual, dual residual, and duality gap, respectively. For linear optimization, we compute \( \epsilon'_\text{pres}, \epsilon'_\text{dres}, \epsilon'_\text{dgap} \) by

\begin{align*}
\epsilon'_\text{pres} &= \frac{\left\| \frac{A}{\tau} X - b \right\|}{\left( 1 + \|b\| \right)}, \\
\epsilon'_\text{dres} &= \frac{\left\| \frac{A^T}{\tau} Y + \frac{s}{\tau} - c \right\|}{\left( 1 + \|c\| \right)}, \\
\epsilon'_\text{dgap} &= \frac{\left\| \frac{c^T}{\tau} X - \frac{b^T}{\tau} Y \right\|}{\left( 1 + \left\| \frac{c^T}{\tau} X \right\| + \left\| \frac{b^T}{\tau} Y \right\| \right)}.
\end{align*}

In the inner loop, we use (22) to check the global convergence. This is an early-stop strategy to jump out of the inner loop, which is particularly useful when some ADMM iterates do not meet the inner loop convergence criterion but already satisfy the global convergence criterion. We remark that when to activate this strategy should be chosen carefully. If we check the global convergence criterion too early, the algorithm will waste a considerable amount of time to verify those ADMM iterates that are far from the global optimality. In the implementation, we employ this strategy when the barrier parameter \( \mu^{(k)} \) satisfies \( \mu^{(k)} < \epsilon \). Although ABIP will terminate in theory when such condition is satisfied, the obtained iterate does not satisfy (22) in practice, and thus the algorithm still proceeds.

### 3.4 Half update

As mentioned before, Lin et al. (2021) propose a customized initialization of \((p_0^{(k)}, q_0^{(k)})\) for the inner problem (7) to eliminate the variables \( p, q \) and \( \tilde{v} \) in the update rule, and then sequentially update \( \tilde{u}_i^{(k+1)}, u_i^{(k+1)}, v_i^{(k+1)} \) in each ADMM iteration. Here, \( v_i^{(k+1)} \) can be seen as the role of dual variable. This further motivates us to introduce the half update strategy in ADMM-type algorithm by updating \( v_i^{(k)} \) twice. In particular, we first update the primal variable \( \tilde{u}_i^{(k+1)} \), followed by the update of dual variable \( v_i^{(k)} \) with stepsize \( \alpha_1 \in (0, 2) \). Then, we update the primal variable \( u_i^{(k+1)} \), followed by
the update of dual variable $v_{i+1}^{(k)}$ with stepsize $\alpha_2 \in (0, 2)$. Therefore, the update rule for $u_{i+1}^{(k)}$ is slightly changed as

$$u_{i+1}^{(k)} = \arg\min_u \left[ B(u, \mu^{(k)}) + \frac{\beta}{2} \left\| u - u_{i+\frac{1}{2}}^{(k)} + v_{i+\frac{1}{2}}^{(k)} \right\|^2 \right],$$  

(23)

where $B(u, \mu^{(k)}) = \beta(n+1)\theta - \mu^{(k)} \log(x) - \mu^{(k)} \log \tau$. Meanwhile, problem (23) admits closed-form solutions given by

$$y_{i+1}^{(k)} = \arg\min_y \frac{\beta}{2} \left\| y - y_{i+\frac{1}{2}}^{(k)} + r_{i+\frac{1}{2}}^{(k)} \right\|^2 = \tilde{y}_{i+1}^{(k)} - r_{i+\frac{1}{2}}^{(k)}$$  

(24)

$$x_{i+1}^{(k)} = \arg\min_x \left[ \frac{\mu^{(k)}}{\beta} \log(x) + \frac{1}{2} \left\| x - \tilde{x}_{i+\frac{1}{2}}^{(k)} + s_{i+\frac{1}{2}}^{(k)} \right\|^2 \right] = \frac{1}{2} \left[ \frac{\mu^{(k)}}{\beta} \log(x) + \frac{1}{2} \left\| x - \tilde{x}_{i+\frac{1}{2}}^{(k)} + s_{i+\frac{1}{2}}^{(k)} \right\|^2 \right] + \frac{4\mu^{(k)}}{\beta}$$  

(25)

$$\tau_{i+1}^{(k)} = \arg\min_\tau \frac{\mu^{(k)}}{\beta} \log \tau + \frac{1}{2} \left\| \tau - \tilde{\tau}_{i+\frac{1}{2}}^{(k)} + \kappa_{i+\frac{1}{2}}^{(k)} \right\|^2 = \frac{1}{2} \left[ \frac{\mu^{(k)}}{\beta} \log(x) + \frac{1}{2} \left\| x - \tilde{x}_{i+\frac{1}{2}}^{(k)} + s_{i+\frac{1}{2}}^{(k)} \right\|^2 \right] + \frac{4\mu^{(k)}}{\beta}$$  

(26)

$$\theta_{i+1}^{(k)} = \beta_{i+1}^{(k)} - \xi_{i+\frac{1}{2}}^{(k)} - (n+1)$$  

(27)

The details of the half-update strategy are presented in Algorithm 2. We observe that this strategy can reduce the number of ADMM iterations on some specific datasets.

---

**Algorithm 2: ABIP with half update**

1. Set $\mu^0 = \beta > 0$, $\alpha, \gamma \in (0, 1)$;
2. Set $r_0^0 = y_0^0 = 0$, $(x_0^0, s_0^0, \kappa_0^0) = (e, 1, e, 1) > 0$, $\theta_0^0 = 1$, $\xi_0^0 = -n - 1$ with $x_0^0 \circ s_0^0 = \mu^0 \cdot e$, and $\tau_0^0 = \frac{\theta_0^0}{\gamma}$;
3. for $k = 0, 1, 2, \cdots$ do
   4.  for $i = 0, 1, 2, \cdots$ do
      5.  if the inner termination condition is satisfied then
         6.    break;
      7.  end
      8.  Update $\bar{u}_{i+1}^{(k)}$ by (11);
      9.  Update $v_{i+\frac{1}{2}}^{(k)} = \alpha u_{i+1}^{(k)} + v_{i+\frac{1}{2}}^{(k)} - \alpha u_{i+1}^{(k)}$;
     10. Update $u_{i+1}^{(k)}$ by (24), (25), (26) and (27);
     11. Update $v_{i+\frac{1}{2}}^{(k)} = \gamma (\tau_{i+\frac{1}{2}}^{(k)} - \xi_{i+\frac{1}{2}}^{(k)})$;
     12. if the final termination criterion is satisfied then
         13.    return;
     14. end
     15. end
     16. Set $\mu^{(k+1)} = \gamma \mu^{(k)}$;
     17. Set $r_0^{(k+1)} = 0$, $\epsilon_0^{(k+1)} = -n - 1$ and
         $(y_0^{(k+1)}, x_0^{(k+1)}, s_0^{(k+1)}, \tau_0^{(k+1)}, \xi_0^{(k+1)}, \theta_0^{(k+1)}) = \sqrt{\gamma} \cdot ( y_{i+1}^{(k)}, x_{i+1}^{(k)}, s_{i+1}^{(k)}, \tau_{i+1}^{(k)}, \xi_{i+1}^{(k)}, \theta_{i+1}^{(k)} )$;  
18. end
3.5 Presolve and preconditioning

The presolve procedure plays an essential role in modern commercial LP solvers. It analyzes the LP problem before submitting it to the optimization algorithm. With the procedure, we can detect and remove empty rows, empty columns, singleton rows and fixed variables, together with removing all the linearly dependent rows. We integrate PaPILO (Gleixner et al., 2022), an open-source presolve package for linear optimization, into ABIP in the enhanced implementation.

After Presolve, we transform (20) to the following reduced LP problem:

\[
\begin{align*}
\min & \quad \bar{c}^T \bar{x} \\
\text{s.t.} & \quad \bar{A} \bar{x} = \bar{b}, \\
& \quad \bar{x} \geq 0.
\end{align*}
\]

Then, similar to PDLP (Applegate et al., 2021b), we perform diagonal preconditioning, which is a popular heuristic to improve the convergence of optimization algorithms. We rescale the constraint matrix \( \bar{A} \in \mathbb{R}^{m \times n} \) to \( \bar{A} = D_1^{-1} \bar{A} D_2^{-1} \) with positive diagonal matrices \( D_1 \) and \( D_2 \). Such preconditioning creates a new LP instance that replaces \( \bar{A}, \bar{c}, \bar{x} \) and \( \bar{b} \) with \( \tilde{A}, \tilde{c} = D_2^{-1} \bar{c}, \tilde{x} = D_2 \bar{x} \) and \( \tilde{b} = D_1^{-1} \bar{b} \), respectively. In particular, we consider the following two rescaling methods.

Pock-Chambolle rescaling (Pock and Chambolle, 2011). This rescaling method is parameterized by \( \alpha > 0 \). The diagonal matrices are defined by \( (D_1)_{jj} = \sqrt{\|A_j, \cdot \|_2} - \alpha \) for \( j = 1, 2, \ldots, m \), and \( (D_2)_{ii} = \sqrt{\|A, i, \cdot \|_\infty} \) for \( i = 1, 2, \ldots, n \).

Ruiz rescaling (Ruiz, 2001). In each iteration of Ruiz rescaling, the diagonal matrices are defined by \( (D_1)_{jj} = \sqrt{\|A_j, \cdot \|_2} \) for \( j = 1, 2, \ldots, m \), and \( (D_2)_{ii} = \sqrt{\|A, i, \cdot \|_\infty} \) for \( i = 1, 2, \ldots, n \). When this rescaling method is applied iteratively, the infinity norm of each row and each column will finally converge to 1.

In the implementation, we first take a few iterations of Ruiz rescaling and then apply the Pock-Chambolle rescaling with \( \alpha = 1 \). Then, we solve the new LP instance with ABIP, and transform its optimal solution with stored \( D_1 \) and \( D_2 \) to obtain the true one.

3.6 Ignoring dual feasibility for null objective problems

In this subsection, we consider the following null objective linear optimization problem:

\[
\begin{align*}
\min & \quad 0 \\
\text{s.t.} & \quad Ax = b, \\
& \quad x \geq 0,
\end{align*}
\]

which is essentially a feasibility problem. This class of problem has wide applications in real life, such as PageRank. It is used by Google to order search engine results, and aims to find the maximal right eigenvector of a stochastic matrix. Nesterov (2014) shows that, it can be reformulated as a linear optimization problem with the null objective,

\[
\begin{align*}
\min & \quad 0 \\
\text{s.t.} & \quad Sx \leq x, \\
& \quad 1^T x = 1, \\
& \quad x \geq 0.
\end{align*}
\]

When ABIP is invoked to solve the above feasibility problem, the dual is homogeneous and
inherently admits a trivial feasible solution \((y, s) = (0, 0)\). Therefore, we can ignore dual feasibility checks during iterations. Although the above observation cannot be used to identify the support of the optimal \(s^*\), it implies that we do not need to check dual infeasibility when the algorithm iterates to obtain a near-feasible primal-dual pair in ABIP.

### 3.7 Integrating the enhancements

As mentioned above, we propose various new strategies to improve the empirical performance of ABIP. Each strategy has some parameters to choose from, and thus one critical problem is how to determine the combination of parameters in these strategies to obtain a better empirical performance. To address this challenge, a decision tree is embedded in ABIP to help identify the rules for choosing a proper combination of strategies. More specifically, for each LP instance, we collect the sparsity and dimension-related statistics of the problem as features, as shown in Figure 2, and record the runtime of ABIP under different combinations of strategies. Then, the combination of strategies with the least runtime is chosen to be the label of the LP instance.

To avoid overfitting, we only consider 3 different combinations of strategies and construct the training set based on different LP datasets including MIPLIB 2017 (Gleixner et al., 2021) and Netlib LP (Koch, 2003). In addition, after the decision tree is obtained from the training procedure, we prune the tree till the rules left are general enough to be coded by the solver. The computational experiments in the following section suggest that the decision tree is often able to capture features for different LP problems, and the derived decision rules prove effective in practice.

![Figure 2: Decision tree for ABIP](image)

### 3.8 Predictor-corrector method

In practice, we observe that the sublinear convergence rate of ABIP often results in a “tailing-off” effect, preventing the algorithm from achieving satisfactory tolerance levels. This is particularly noticeable in medium-sized problems, where 10 to 100 ADMM iterations can be as computationally expensive as a full Newton step in traditional IPM. To partially address this, we incorporate a primal-dual IPM with Mehrotra’s corrector step into ABIP. Specifically, with a pre-specified tolerance level, if ABIP observes the significant tailing-off effect and the tolerance level is not reached, it will perform several interior point steps to improve solution accuracy. Since each interior point step involves a matrix decomposition, we only consecutively use it at most five times and employ Mehrotra’s corrector step to reuse the decomposition.

### 4 Numerical experiments

In this section, we present a comprehensive numerical study on the enhanced version of ABIP, referred to as ABIP+. ABIP+ is now implemented in C language with a MATLAB interface. As
mentioned before, it uses two different methods to solve the linear system. The direct method uses a sparse permuted LDL$^\top$ factorization provided by the SuiteSparse package (Davis et al., 2013), while the indirect ABIP+ uses a preconditioned conjugate gradient method.

For LP, we comprehensively evaluate the strategies of ABIP+ using the Netlib LP benchmark dataset. Then, we test the performance of ABIP+ on 240 instances from MIP2017, 48 instances from the Mittlemann LP barrier dataset, and randomly generated PagaRank instances. We also compare ABIP+ with PDLP (Applegate et al., 2021b), a state-of-the-art first-order LP solver implemented in C++, and COPT, a general-purpose commercial solver with leading performance on LP.

To further demonstrate the performance of ABIP+ on conic optimization, we first randomly generate several LASSO problems and choose 6 large SVM problems from the LIBSVM dataset (Chang and Lin, 2011). For each problem, we maintain its two equivalent formulations. The first is the SOCP formulation. The second is the QP formulation with the quadratic objective function and conic constraints. We provide the details in Appendix D. Although both LASSO and SVM problems can be solved by customized algorithms developed in machine learning, it is a standard practice (Stellato et al., 2020; O’Donoghue et al., 2016) to use them to examine the scalability and efficiency of conic optimization and QP solvers. For ABIP+, despite that we only discuss its extension to linear conic optimization before, it can be further developed to solve the following quadratic conic optimization (QCP) and thus handle the QP formulation of LASSO and SVM problems:

$$\min \frac{1}{2} x^T P x + c^T x$$

s.t. $A x = b,$

$x \in \mathcal{K}.$

To be succinct, ABIP+ can maintain a similar update rule as (11), (12), and (13) by considering the monotone linear complementarity problem and homogeneous embedding (Andersen and Ye, 1999; O’Donoghue, 2021), and we omit the details in this paper. We use the state-of-the-art open-source conic optimization solver SCS (O’Donoghue et al., 2016) and the general-purpose commercial solver GUROBI (Gurobi Optimization, LLC, 2022) as the benchmarks. We do not compare with OSQP (Stellato et al., 2020), since it uses a weaker termination criterion and has an inferior performance to SCS as shown in O’Donoghue (2021). Finally, we assess the performance of ABIP+ on standard SOCP benchmark CBLIB (Friberg, 2016), and compare with SCS and another recently popular first-order solver COSMO (Garstka et al., 2021a).

Let $\epsilon > 0$ be the tolerance level of the solution accuracy. We use (21) as the global convergence criterion. For linear optimization, we compute $\epsilon_{\text{pres}}$, $\epsilon_{\text{dres}}$, and $\epsilon_{\text{gap}}$ by (22). While for general conic optimization, they are computed by

$$\epsilon_{\text{pres}} = \|A^{\frac{\tau}{\tau}} x - b\|_\infty / \left(1 + \max \left(\|A^{\frac{\tau}{\tau}} x\|_\infty, \|b\|_\infty\right)\right),$$

$$\epsilon_{\text{dres}} = \|A^{\frac{\tau}{\tau}} y^{\frac{\tau}{\tau}} + s - c\|_\infty / (1 + \|c\|_\infty),$$

$$\epsilon_{\text{gap}} = \|c^T x^{\frac{\tau}{\tau}} - b^T y^{\frac{\tau}{\tau}}\|_\infty / \left(1 + \max \left(\|c^T x^{\frac{\tau}{\tau}}\|_\infty, \|b^T y^{\frac{\tau}{\tau}}\|_\infty\right)\right).$$

Typically, we set $\epsilon = 10^{-4}$ or $10^{-6}$. For inner loop in conic optimization, we use the following termination criteria: $||Qu^{(k)} - v_i^{(k)}|| \leq (\mu^{(k)})^\alpha (1 + ||u_i^{(k)}, v_i^{(k)}||)$ with $\alpha \in [0.25, 2]$, which ensures the satisfactory convergence behavior of ABIP+.

When testing the Netlib LP benchmark dataset, we limit the number of ADMM iterations to $10^6$. For other LP benchmark datasets, we set the runtime limit to be 3600s. For conic optimization,
we set the runtime limit to be 2000s for LASSO, 3000s for SVM, and 100s for SOCP in CBLIB. Finally, we remark that we do not apply any of the novel strategies developed in Section 3 in the general conic optimization since the acceleration performance is marginal in the experiments.

Unless otherwise mentioned, we use the shifted geometric mean (SGM) of runtimes to evaluate the algorithm’s performance over a dataset. Specifically, we define the SGM by

$$\text{SGM} = \left( \prod_{i=1}^{n} (t_i + \Delta) \right)^{1/n} - \Delta,$$

where $t_i$ is the runtime for the $i$-th instance, and $\Delta = 10$ is a time-shift value to alleviate the effect of runtimes that are almost 0. When the algorithm fails to solve a certain instance, we set $t_i = 15000$. We normalize all the SGMs by setting the smallest SGM to be 1.

### 4.1 Numerical results on Netlib LP benchmark dataset

In this subsection, we showcase the improvements of ABIP+ over the original ABIP on the Netlib LP benchmark dataset. For the restart strategy, we set the restart threshold $M_T = 10^5$ and the fixed restart period $F = 10^3$. For the rescale strategy, we do Pock-Chambolle rescaling first and then Ruiz rescaling 10 times. For the hybrid $\mu$ strategy, we use the aggressive strategy when the barrier parameter $\mu(k) < 10^3 \cdot \epsilon$. Otherwise, we use the LOQO method.

Table 3 shows the performance of ABIP with different strategies. The first row gives the results of the original ABIP, and the second row shows the performance of ABIP with restart strategy. Similarly, the third and the fourth row correspond to ABIP with both restart and rescale strategies and ABIP with the whole three strategies, respectively. The second column gives the number of instances each algorithm solved within $10^6$ ADMM iterations. Moreover, the last three columns give the average number of IPM iterations, the average number of ADMM iterations, and the average computational time over the solved instances, respectively. It can be seen that each added strategy improves the performance of ABIP. Among 105 LP instances selected from the Netlib database, ABIP solves only 65 instances to $10^{-6}$ relative accuracy given a limit of $10^6$ ADMM iterations per problem. On the contrary, ABIP+ solves 86 instance under the same setting. Moreover, ABIP+ reduces more than 70% ADMM iterations and more than 80% runtime on average. Therefore, the acceleration performance of these strategies is significant.

| Method               | # Solved | Avg. # IPM | Avg. # ADMM | Avg.Time (s) |
|----------------------|----------|------------|-------------|--------------|
| ABIP                 | 65       | 74         | 265418      | 87.07        |
| + restart            | 68       | 74         | 88257       | 23.63        |
| + rescale            | 84       | 72         | 77925       | 20.44        |
| + hybrid $\mu$ (=ABIP+) | **86**  | **22**     | **73738**   | **14.97**    |

Table 3: Performance of ABIP with different strategies on Netlib, $\epsilon = 10^{-6}$.

### 4.2 Numerical results on MIP2017 LP benchmark dataset

In this subsection, we evaluate the performance of ABIP+, PDLP and COPT, on the MIP2017 LP benchmark dataset. For a fair comparison, all tested instances for PDLP, ABIP, and ABIP+ are presolved by PaPILO (Gleixner et al., 2022). We also include different tolerance levels to terminate the first-order LP solvers. Specifically, we let $\epsilon = 10^{-4}$ and $10^{-6}$ for PDLP and ABIP+. Since ABIP+ inherits the framework of IPM, we also test the effectiveness of the predictor-corrector method. Specifically, after the ADMM iterates reach the tolerance level $\epsilon = 10^{-3}$, we invoke the predictor-corrector method to further reach $\epsilon = 10^{-6}$. These results are indicated with a † symbol presented in Table 4 and Table 5.
On one hand, Table 4 shows that the first-order algorithms generally lag behind the IPM implemented in COPT. On the other hand, it demonstrates that ABIP+ significantly improves the performance of original ABIP. It is also comparable to PDLP on the MIP2017 dataset when $\epsilon = 10^{-6}$ and the predictor-corrector method is used. While PDLP may slightly outperform ABIP+ in terms of overall performance, Table 5 gives specific instances where ABIP+ has a clear advantage over PDLP in terms of computational time. This suggests that ABIP+ and PDLP may each be better suited to different types of problems, which may require independent investigation.

| Method | $\epsilon$ | # Solved | SGM |
|--------|------------|----------|-----|
| COPT   | $10^{-6}$  | 240      | 1   |
| ABIP+  | $10^{-4}$  | 192      | 34.8|
| ABIP+† | $10^{-4}$  | 220      | 21.3|
|        | $10^{-6}$† | 213      | 35.4|
|        | $10^{-6}$  | 220      | 14  |
| PDLP   | $10^{-4}$  | 226      | 6.7 |
|        | $10^{-6}$  | 221      | 12.9|

Table 4: Numerical results on MIP2017. The symbol † means that we utilize the predictor-corrector method.

| Instance       | ABIP+ | ABIP+† | PDLP  | COPT  |
|----------------|-------|--------|-------|-------|
| app1-2         | 49.65 | 8.07   | 118.83| **1.18** |
| buildingenergy | 508.34| 74.98  | 998.21| **18.09** |
| map16715-04    | 638.13| 17.57  | 1241.83| **3.08** |
| neos-4647030-tutaki | 403.03 | 266.87 | f     | **4.71** |
| neos-4763324-toguru | 1336.10 | 997.77 | f     | **6.06** |
| unitcal_7      | 27.12 | 32.39  | 282.34| **3.61** |

Table 5: The runtime (in sec.) of ABIP+, ABIP+†, PDLP and COPT on selected instances from MIP2017. For ABIP+ and PDLP, we set $\epsilon = 10^{-4}$. For ABIP+† and COPT, we set $\epsilon = 10^{-6}$. The symbol † means that we utilize the predictor-corrector method. Note that “f” in the table means that the algorithm fails to solve the instance within the runtime limit.

### 4.3 Numerical results on Mittlemann LP barrier benchmark dataset

To further test the robustness of different solvers studied in the last subsection, we compare their performance on the Mittlemann LP barrier dataset. This dataset is commonly used for benchmarking IPM-based LP solvers. As before, we include the results of ABIP+ utilizing the predictor-corrector method, marked by the symbol †, at a tolerance level of $\epsilon = 10^{-6}$. Similarly, the barrier IPM in COPT is also terminated at the same accuracy. Table 6 shows that, although our ABIP+ is slightly inferior to PDLP in terms of computational time, it solves more instances at $\epsilon = 10^{-6}$. Meanwhile, ABIP+ with the predictor-corrector method also shows its effectiveness in this dataset. Table 7 lists some instances that ABIP+ solves faster than PDLP.

### 4.4 Numerical results on PageRank instances

In this subsection, we apply ABIP+ to solve the PageRank problem via its LP formulation, as introduced in Section 3.6. Due to a special sparse pattern in the coefficient matrix, the traditional IPM solvers, such as GUROBI, fail to solve most of these instances, as shown in Applegate et al. (2021b). The reason behind is that, this special sparse pattern takes much time in the ordering stage and consumes a massive amount of memory (typically above 30GBytes). However, since
| Method | $\epsilon$ | # Solved | SGM |
|--------|------------|----------|-----|
| COPT   | $10^{-6}$  | 50       | 1   |
| ABIP   | $10^{-4}$  | 22       | 57.40 |
| ABIP+  | $10^{-4}$  | 29       | 27.22 |
|        | $10^{-6}$  | 22       | 49.32 |
|        | $10^{-6^+}$| 32       | 17.22 |
| PDLP   | $10^{-4}$  | 39       | 9.57 |
|        | $10^{-6}$  | 19       | 34.54 |

Table 6: Numerical results on Mittelmann LP barrier dataset.

| Instance | ABIP+ | PDLP | COPT |
|----------|-------|------|------|
| cont1    | 8.93  | 49.17| 3.35 |
| irish-e  | 31.61 | 536.14| 24.81|
| ns1688926| 1200.62 | f    | 19.06|
| rmine15  | 713.36 | 2782.00| 87.99|

Table 7: The runtime (in sec.) of ABIP+, PDLP and COPT on selected instances from the Mittelmann LP barrier dataset. For ABIP+ and PDLP, we set $\epsilon = 10^{-4}$.

ABIP+ can utilize the conjugate gradient method instead of direct factorization for these instances, together with the strategies introduced in Section 3, our ABIP+ manages to solve these PageRank instances efficiently.

The PageRank instances are first benchmarked by PDLP (Applegate et al., 2021b). Following the instructions in Applegate et al. (2021b), we generate 115 PageRank instances from several sparse matrix datasets, including DIMACS10, Gleich, Newman, and SNAP. These matrices can be found from the SuiteSparse matrix collection. Besides a comparison over runtime, we also compare the number of matrix-vector products used in these methods. Table 9 provides some specific instances where ABIP+ can be significantly faster than PDLP.

| Method | # Solved | SGM     | Scaled Mean Matrix-Vector Products |
|--------|----------|---------|-----------------------------------|
| ABIP+  | 107      | 1       | 1528.72                           |
| PDLP   | 101      | 3.91    | 6510.48                           |

Table 8: ABIP+ v.s. PDLP on randomly generated PageRank dataset, $\epsilon = 10^{-4}$.

Another interesting class of PageRank instances where ABIP+ has a clear advantage is the staircase PageRank instances. When one generates the PageRank instance by the code provided in Applegate et al. (2021b), the coefficient matrix will exhibit a staircase form if the number of nodes is set to equal to the number of edges, see Figure 3. In Table 10, we list several such instances and compare ABIP+ with PDLP. It is notable that, as the matrix size increases, the runtime gap between PDLP and ABIP+ becomes more prominent.

4.5 Numerical results on LASSO instances

From this subsection, we evaluate the performance of ABIP+ on conic optimization. In particular, we apply ABIP+ to solve LASSO, SVM and standard SOCP problems. In this section, we focus on LASSO problem, which is a well-known regression model obtained by adding an $\ell_1$-norm regularization term in the objective to promote sparsity. It can be formulated as

$$
\min_x \|Ax - b\|_2^2 + \lambda\|x\|_1,
$$

2For details, see https://sparse.tamu.edu/about.
Table 9: The runtime (in sec.) of selected instances from randomly generated PageRank dataset, $\epsilon = 10^{-4}$.

| Instance                  | # nodes | PDLP  | ABIP+ |
|---------------------------|---------|-------|-------|
| usroads                   | 129164  | 51.66 | \textbf{7.16} |
| vsp_bcsstk30_500sep_10in_1Kout | 58348   | 252.86| \textbf{10.72} |
| web-BerkStan              | 685230  | 3497.32| \textbf{552.44} |

Figure 3: An illustration of the coefficient matrix of the staircase PageRank instance with 10 nodes.

where $A \in \mathbb{R}^{m \times n}$ is the data matrix and $\lambda$ is the $\ell_1$-penalty weight. Here, $m$ is the number of samples and $n$ is the number of features. As mentioned before, we solve the problem via its SOCP and QCP formulations. For ABIP+, we also utilize the customized linear system solver by exploiting the intrinsic structure of the problem.

We randomly generate 9 LASSO problems, which span a wide range of ratios between $m$ and $n$. We choose $m \in \{1000, 2000, 5000\}, n \in \{5000, 10000, 15000\}$. In all the experiments, we let $\lambda = \|A^T b\|_\infty/5$, the tolerance level $\epsilon = 10^{-3}$, and set the runtime limit to be 2000 seconds. We compare the performance of ABIP+ with SCS and GUROBI, and provide the results in Table 11. It shows that ABIP+ with SOCP formulation outperforms SCS and GUROBI with both formulations over all the instances, suggesting the empirical advantage of ABIP+ over other solvers. Moreover, it is interesting to observe that GUROBI almost has the lowest iteration number among all the three solvers. However, its overall runtime performance is still inferior to ABIP+ due to the expensive computation cost when repetitively solving the Newton equation.

4.6 Numerical results on SVM instances

In this subsection, we compare the performance of ABIP+ with SCS and GUROBI on SVM problem. Specifically, we consider the binary classification problem and let $\{x_i, y_i\}_{1 \leq i \leq m}$ be the training data, where $x_i \in \mathbb{R}^n$ is the feature vector and $y_i \in \{1, -1\}$ is the corresponding label.

| # nodes | PDLP  | ABIP+ |
|---------|-------|-------|
| $10^4$  | 8.60  | \textbf{0.93} |
| $10^5$  | 135.67| \textbf{10.36} |
| $10^6$  | 2248.40| \textbf{60.32} |

Table 10: Staircase PageRank instances, $\epsilon = 10^{-6}$. 
Table 11: Comparison of runtime (in sec.) and iteration numbers on the LASSO problems.

| m    | n    | ABIP+ SOCP | ABIP+ QCP | SCS SOCP | SCS QCP | GUROBI SOCP | GUROBI QCP |
|------|------|------------|-----------|----------|---------|-------------|------------|
| 1000 | 5000 | 1.11       | 67        | 4.31     | 6.50    | 1.19        | 8          |
| 1000 | 10000| 1.95       | 61        | 10.09    | 16.95   | 2.76        | 9          |
| 1000 | 15000| 3.62       | 60        | 23.09    | 23.92   | 4.23        | 10         |
| 2000 | 5000 | 1.92       | 35        | 12.79    | 32.64   | 6.77        | 11         |
| 2000 | 10000| 4.17       | 35        | 29.78    | 32.64   | 7.77        | 11         |
| 2000 | 15000| 6.52       | 39        | 43.23    | 48.20   | 9.53        | 9          |
| 5000 | 5000 | 21.15      | 161       | 134.17   | 140.50  | 10.95       | 7          |
| 5000 | 10000| 19.80      | 32        | 176.02   | 193.27  | 49.40       | 12         |
| 5000 | 15000| 26.80      | 31        | 208.23   | 231.44  | 42.25       | 12         |
| SGM  |      | 1.00       | 4.83      | 3.05     | 1.14    |             |            |

Table 12: Comparison of runtime (in sec.) on the selected SVM problems.

SVM aims to classify the two groups by a linear model, and solves the following quadratic problem:

$$\min_{(w, b) \in \mathbb{R}^{n+1}} \frac{\lambda}{2} w^T w + \frac{1}{m} \sum_{i=1}^{m} \xi_i$$

s.t. $$y_i (x_i^T w + b) \geq 1 - \xi_i, \quad \forall i = 1, 2, ..., m,$$

$$\xi_i \geq 0, \quad \forall i = 1, 2, ..., m.$$ 

We choose 6 large SVM instances from LIBSVM (Chang and Lin, 2011), which cover both scenarios of high dimensionality and large sample size. In all the experiments, we choose $$\lambda = 10^{-3}$$, let $$\epsilon = 10^{-3}$$, and set the runtime limit to be 3000s. We also solve the problem via its SOCP and QCP formulations, and present the results in Table 12. It demonstrates that ABIP+ has the most robust performance among all the compared solvers. It successfully solves all the problems in both QCP and SOCP formulations, while SCS and GUROBI are unable to solve all instances in each formulation. Specifically, GUROBI is unable to solve the real-sim instance in both formulations. SCS exhibits inferior performance on large-scale problems, failing on news20 and real-sim instances, and consuming significant runtime for the rcv1_train instance in the SOCP formulation. Furthermore, it fails on real-sim and skin_nonskin instances and experiences a large runtime for the news20 instance in the QCP formulation.

| Dataset  | m    | n    | ABIP+ SOCP | ABIP+ QCP | SCS SOCP | SCS QCP | GUROBI SOCP | GUROBI QCP |
|----------|------|------|------------|-----------|----------|---------|-------------|------------|
| s        | 581012 | 54    | 31.85      | 270.92    | 134.98   | 67.97   | 1513.57     | 1711.41    |
| ijcnn1   | 49990 | 22    | 2.76       | 4.00      | 6.45     | 2.98    | 13.4        | 13.79      |
| news20   | 19996 | 1355191 | 449.23    | 209.35    | 2877.66  | 138.19  | 147.7       |            |
| rcv1_train | 20242 | 44504 | 225.42    | 168.41    | 2352.65  | 962.9   | 91.35       | 138.97     |
| real-sim | 72309 | 20958 | 353.85    | 232.54    | timeout  | timeout | timeout     |            |
| skin_nonskin | 245057 | 3     | 9.27      | 67.72     | 132.22   | 91.48   | 91.71       |            |

Table 12: Comparison of runtime (in sec.) on the selected SVM problems.

### 4.7 Numerical results on SOCP instances of CBLIB

In this subsection, we compare the performance of ABIP+ with the other two first-order solvers SCS (O’Donoghue et al., 2016) and COSMO (Garstka et al., 2021b) on 1405 selected SOCP instances from the CBLIB (Friberg, 2016). All the instances are presolved by COPT. We set $$\epsilon = 10^{-4}$$ and set the runtime limit to be 100 seconds. When conducting experiments, we find that the first-order algorithms sometimes obtain inaccurate solutions even though their empirical stopping
criteria are satisfied. For a fair comparison, when an algorithm solves an instance within the run-time limit, we count the instance as a solved instance only if its relative error to the optimal value returned by MOSEK is less than a pre-specified threshold $\epsilon_{\text{tol}}$. The relative error is defined by $(|f - f^*|)/\max\{|f^*|, 1\}$, where $f^*$ is the optimal value returned by MOSEK, and we set $\epsilon_{\text{tol}} = 0.01$. The detailed results are presented in Table 13. It can be seen that MOSEK and COPT achieve better performance than the first-order solvers, since second-order methods have the advantage of high accuracy and efficiency when solving small-scale problems. When comparing between the first-order solvers, SCS has a better performance than ABIP+ on runtime, but only solves 3 more instances. Hence, our ABIP+ has a comparable performance with SCS. Meanwhile, we find that both SCS and ABIP+ outperform COSMO in terms of solving more instances and requiring less runtime.

| # Solved | MOSEK | COPT | SCS | ABIP+ | COSMO |
|----------|-------|------|-----|-------|-------|
| SGM      | 1405  | 1387 | 1337| 1334  | 1308  |
|          | 1.00  | 1.13 | 3.26| 3.52  | 4.64  |

Table 13: Numerical results on CBLIB dataset.

5 Conclusion

In this paper, we continue the development of the ADMM-based interior point method (Lin et al., 2021). We generalize ABIP to deal with general conic constraints, theoretically justify that ABIP converges at a $\tilde{O}(1/\epsilon)$ rate for the general linear conic problems, and provide efficient implementation to further accelerate the empirical performance on some important SOCP problems. Moreover, we provide several implementation techniques that are inspired by some existing methods and substantially improve the performance of ABIP for large-scale LP. Despite its sensitivity to the condition number and inverse precision, the enhanced ABIP solver is highly competitive for certain structured challenging problems such as PageRank and some machine learning problems. We believe that ABIP+ is complementary to the state-of-the-art open-source solvers and exhibits a strong potential for future improvement. For example, it would be interesting to extend ABIP+ to the distributed and asynchronous environment. Our code is open-source and available at https://github.com/leavesgrp/ABIP.

References

Farid Alizadeh and Donald Goldfarb. Second-order cone programming. Mathematical programming, 95(1):3–51, 2003.

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

Jos F Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. Optimization methods and software, 11(1-4):625–653, 1999.

Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. Sdpt3—a matlab software package for semidefinite programming, version 1.3. Optimization methods and software, 11(1-4):545–581, 1999.

ApS, Mosek. Mosek optimization toolbox for matlab, 2019.
Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL https://www.gurobi.com.

Dongdong Ge, Qi Huangfu, Zizhuo Wang, Jian Wu, and Yinyu Ye. Cardinal optimizer (copt) user guide. arXiv preprint arXiv:2208.14314, 2022.

Spyridon Pougkakiotis and Jacek Gondzio. An interior point-proximal method of multipliers for linear positive semi-definite programming. Journal of Optimization Theory and Applications, pages 1–33, 2022.

Stefano Cipolla and Jacek Gondzio. Proximal stabilized interior point methods and low-frequency-update preconditioning techniques. Journal of Optimization Theory and Applications, 197(3):1061–1103, 2023.

Guanglu Zhou and Kim-Chuan Toh. Polynomiality of an inexact infeasible interior point algorithm for semidefinite programming. Mathematical programming, 99:261–282, 2004.

Stefania Bellavia and Sandra Pieraccini. Convergence analysis of an inexact infeasible interior point method for semidefinite programming. Computational Optimization and Applications, 29:289–313, 2004.

Zhaosong Lu, Renato DC Monteiro, and Jerome W O’Neal. An iterative solver-based infeasible primal-dual path-following algorithm for convex quadratic programming. SIAM Journal on optimization, 17(1):287–310, 2006.

Ghussoun Al-Jeiroudi and Jacek Gondzio. Convergence analysis of the inexact infeasible interior-point method for linear optimization. Journal of Optimization Theory and Applications, 141:231–247, 2009.

Filippo Zanetti and Jacek Gondzio. A new stopping criterion for krylov solvers applied in interior point methods. SIAM Journal on Scientific Computing, 45(2):A703–A728, 2023.

Liuqin Yang, Defeng Sun, and Kim-Chuan Toh. SDPNAL + : a majorized semismooth newton-cg augmented lagrangian method for semidefinite programming with nonnegative constraints. Math. Program. Comput., 7(3):331–366, 2015. doi: 10.1007/s12532-015-0082-6. URL https://doi.org/10.1007/s12532-015-0082-6.

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(3):1042–1068, 2016.

David Applegate, Oliver Hinder, Haihao Lu, and Miles Lubin. Faster first-order primal-dual methods for linear programming using restarts and sharpness. arXiv preprint arXiv:2105.12715, 2021a.

Stephen P. Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn., 3(1):1–122, 2011. doi: 10.1561/2200000016. URL https://doi.org/10.1561/2200000016.

Yinyu Ye, Michael J Todd, and Shinji Mizuno. An O (√nL)-iteration homogeneous and self-dual linear programming algorithm. Mathematics of operations research, 19(1):53–67, 1994.

Pantelis Sopasakis, Krina Menounou, and Panagiotis Patrinos. Superscs: fast and accurate large-scale conic optimization. In 2019 18th European Control Conference (ECC), pages 1500–1505. IEEE, 2019.
Tianyi Lin, Shiqian Ma, Yinyu Ye, and Shuzhong Zhang. An admm-based interior-point method for large-scale linear programming. Optimization Methods and Software, 36(2-3):389–424, 2021. doi: 10.1080/10556788.2020.1821200.

Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. J. Math. Imaging Vis., 40(1):120–145, 2011. doi: 10.1007/s10851-010-0251-1. URL https://doi.org/10.1007/s10851-010-0251-1.

David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O’Donoghue, and Warren Schudy. Practical large-scale linear programming using primal-dual hybrid gradient. Advances in Neural Information Processing Systems, 34:20243–20257, 2021b.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. Mathematical Programming Computation, 12(4):637–672, 2020.

Michael Garstka, Mark Cannon, and Paul Goulart. Cosmo: A conic operator splitting method for convex conic problems. Journal of Optimization Theory and Applications, 190(3):779–810, 2021a.

Brendan O’Donoghue. Operator splitting for a homogeneous embedding of the linear complementarity problem. SIAM Journal on Optimization, 31(3):1999–2023, 2021.

Brendan O’Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. SCS: Splitting conic solver, version 3.2.4. https://github.com/cvxgrp/scs, November 2023.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. Machine learning, 20(3):273–297, 1995.

Robert Tibshirani. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1):267–288, 1996.

Harry M Markowitz. Foundations of portfolio theory. The journal of finance, 46(2):469–477, 1991.

Z-Q. Luo, J.F. Sturm, and Shuzhong Zhang. Duality results for conic convex programming. Econometric Institute Research Papers EI 9719/A, Erasmus University Rotterdam, Erasmus School of Economics (ESE), Econometric Institute, 1997.

Shuzhong Zhang. A new self-dual embedding method for convex programming. Journal of Global Optimization, 29(4):479–496, 2004.

Goran Banjac, Paul Goulart, Bartolomeo Stellato, and Stephen Boyd. Infeasibility detection in the alternating direction method of multipliers for convex optimization. Journal of Optimization Theory and Applications, 183:490–519, 2019.

David Applegate, Mateo Díaz, Haihao Lu, and Miles Lubin. Infeasibility detection with primal-dual hybrid gradient for large-scale linear programming. arXiv preprint arXiv:2102.04592, 2021c.

Stephen Wright, Jorge Nocedal, et al. Numerical optimization. Springer Science, 35(67-68):7, 1999.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming, 106(1):25–57, 2006.

Robert J Vanderbei. Loqo: An interior point code for quadratic programming. Optimization methods and software, 11(1-4):451–484, 1999.
Sebastian Pokutta. Restarting algorithms: sometimes there is free lunch. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 22–38. Springer, 2020.

Ambros Gleixner, Leona Gottwald, and Alexander Hoen. Papilo: A parallel presolving library for integer and linear programming with multiprecision support. *arXiv preprint arXiv:2206.10709*, 2022.

Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 International Conference on Computer Vision*, pages 1762–1769. IEEE, 2011.

Daniel Ruiz. A scaling algorithm to equilibrate both rows and columns norms in matrices. Technical report, CM-P00040415, 2001.

Yu Nesterov. Subgradient methods for huge-scale optimization problems. *Mathematical Programming*, 146(1):275–297, 2014.

Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarrk, Thorsten Koch, Jeff Linderoth, et al. Miplib 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, 13(3):443–490, 2021.

Thorsten Koch. The final netlib-lp results. 2003.

Tim Davis, I Duff, P Amestoy, J Gilbert, S Larimore, E Palamadai Natarajan, Y Chen, W Hager, and S Rajamanickam. Suitesparse: A suite of sparse matrix packages. https://github.com/DrTimothyAldenDavis/SuiteSparse, 2013. Accessed: March 15, 2024.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

Erling D Andersen and Yinyu Ye. On a homogeneous algorithm for the monotone complementarity problem. *Mathematical Programming*, 84(2):375–399, 1999.

Henrik A Friberg. Chlib 2014: a benchmark library for conic mixed-integer and continuous optimization. *Mathematical Programming Computation*, 8(2):191–214, 2016.

Michael Garstka, Mark Cannon, and Paul Goulart. COSMO: A conic operator splitting method for convex conic problems. *Journal of Optimization Theory and Applications*, 190(3):779–810, 2021b. doi: 10.1007/s10957-021-01896-x. URL https://doi.org/10.1007/s10957-021-01896-x.

Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale modeling & simulation*, 4(4):1168–1200, 2005.
A Proof of Theorem 1

We shall prove the result by induction. (i) At iteration \( j = 0 \), the result holds true from our definition. (ii) Assuming it holds true for iteration \( j = i \), we will prove that the result still holds true for iteration \( j = i + 1 \) in two steps:

**Step 1:** We claim that

\[
\| \mathbf{u}_i^{(k)} + \mathbf{v}_i^{(k)} \| = \| \mathbf{u}_{i+1}^{(k)} + \mathbf{v}_{i+1}^{(k)} \|. \tag{29}
\]

Indeed, (8) can be rewritten as

\[
(\mathbf{u}_{i+1}^{(k)}, \mathbf{v}_{i+1}^{(k)}) = \Pi_{\mathcal{P}}(\mathbf{u}_i^{(k)} + \mathbf{v}_i^{(k)}, \mathbf{u}_i^{(k)} + \mathbf{v}_i^{(k)}), \tag{30}
\]

where \( \mathcal{P} = \{ (\mathbf{u}, \mathbf{v}) : \mathbf{Q}\mathbf{u} = \mathbf{v} \} \). Moreover, since \( \mathbf{Q} \) is skew-symmetric, the orthogonal complement of \( \mathcal{P} \) is \( \mathcal{P}^\perp = \{ (\mathbf{v}, \mathbf{u}) : \mathbf{Q}\mathbf{u} = \mathbf{v} \} \). Therefore, we conclude that

\[(\mathbf{u}, \mathbf{v}) = \Pi_{\mathcal{P}}(\mathbf{z}, \mathbf{z}) \text{ if and only if } (\mathbf{v}, \mathbf{u}) = \Pi_{\mathcal{P}^\perp}(\mathbf{z}, \mathbf{z}),\]

because the two projections are identical for reversed output arguments. This implies that

\[(\mathbf{v}_i^{(k)}, \mathbf{u}_i^{(k)}) = \Pi_{\mathcal{P}^\perp}(\mathbf{u}_i^{(k)} + \mathbf{v}_i^{(k)}, \mathbf{u}_i^{(k)} + \mathbf{v}_i^{(k)}). \tag{31}\]

Then we combine (30) and (31) to get the desired result.

**Step 2:** Given \( p_i^{(k)} = v_i^{(k)}, q_i^{(k)} = u_i^{(k)} \) and \( \xi_i^{(k)} = -x_0^T s_0 - 1 \), our goal is to show that

\[ p_{i+1}^{(k)} = v_{i+1}^{(k)}, \quad q_{i+1}^{(k)} = u_{i+1}^{(k)}. \]

Note that (29) can be rewritten as

\[ \mathbf{u}_i^{(k)} - \mathbf{v}_i^{(k)} = - (\mathbf{v}_{i+1}^{(k)} - \mathbf{u}_{i+1}^{(k)}). \]

We denote \( \sigma = \mathbf{u}_{i+1}^{(k)} - \mathbf{v}_{i+1}^{(k)} \). It is easy to verify that functions in (5) form pairs of the function \( h(t) \) and \( w(t) \) satisfying \( w(t) = h^*(t) \), where \( h^* \) denotes the conjugate function of \( h \). From Moreau’s decomposition (see Section 2.5 in Combettes and Wajs (2005) for more details), for such pairs of functions \( (h(t), w(t)) \) in \( B \), we have

\[ \mathbf{u}_{i+1}^{(k)} - \mathbf{v}_{i+1}^{(k)} = \text{prox}_h(\sigma) - \text{prox}_w(-\sigma) = \sigma. \]

Then from (10) we have

\[
p_{i+1}^{(k)} = v_{i+1}^{(k)} + u_{i+1}^{(k)} - u_{i+1}^{(k)} = u_{i+1}^{(k)} - \sigma = v_{i+1}^{(k)},
\]

\[
q_{i+1}^{(k)} = u_{i+1}^{(k)} + v_{i+1}^{(k)} - v_{i+1}^{(k)} = v_{i+1}^{(k)} + \sigma = u_{i+1}^{(k)}.
\]

This completes the proof. \( \square \)
B  Proof of Proposition 5

Observing that

\[ \nabla^2 B(u, v, \mu^{(k)}) \succeq \frac{\mu^{(k)}}{C_D} I \quad (32) \]
\[ \nabla^2 B(u, v, \mu^{(k)}) \succeq \frac{\mu^{(k)}}{C_D} I \quad (33) \]
\[ \nabla^2 B(u, v, \mu^{(k)}) \geq \frac{\mu^{(k)}}{D} \quad (34) \]
\[ \nabla^2 B(u, v, \mu^{(k)}) \geq \frac{\mu^{(k)}}{D} \quad (35) \]

For convenience, we assume that \( C_D \geq D \). The optimality condition of problem (7) implies that

\[ \beta(p_k^*, q_k^*) \in \partial I(Qu = v)[u_k^*, v_k^*], \quad -\beta(p_k^*, q_k^*) \in \partial B(u_k^*, v_k^*, \mu^{(k)}). \]

Using the convexity of the two functions and (32)(33)(34)(35) we have

\[ 0 \leq \beta(u_k^* - \tilde{u}_{i+1}^{(k)}, v_k^* - \tilde{v}_{i+1}^{(k)})^T (p_k^*, q_k^*) = \beta(u_k^* - \tilde{u}_{i+1}^{(k)}, v_k^* - \tilde{v}_{i+1}^{(k)})^T (v_k^*, u_k^*) \quad (36) \]
\[ B(u_k^*, v_k^*, \mu^{(k)}) - B(u_{i+1}^{(k)} - v_{i+1}^{(k)}, \mu^{(k)}) + \frac{\mu^{(k)}}{2CD} (||x_{i+1}^{(k)} - x_k^*||^2 + ||s_{i+1}^{(k)} - s_k^*||^2 + (\tau_{i+1}^{(k)} - \tau_k^* )^2 + (\kappa_{i+1}^{(k)} - \kappa_k^* )^2) \]
\[ \leq - \beta(u_k^* - u_{i+1}^{(k)}, v_k^* - v_{i+1}^{(k)})^T (p_k^*, q_k^*) = - \beta(u_k^* - u_{i+1}^{(k)}, v_k^* - v_{i+1}^{(k)})^T (v_k^*, u_k^*). \]

(37)

Summing up (36)(37) we have

\[ B(\tilde{u}_{i+1}^{(k)}, \tilde{v}_{i+1}^{(k)}, \mu^{(k)}) - B(u_{i+1}^{(k)} - v_{i+1}^{(k)}, \mu^{(k)}) + \frac{\mu^{(k)}}{2CD} (||x_{i+1}^{(k)} - x_k^*||^2 + ||s_{i+1}^{(k)} - s_k^*||^2 + (\tau_{i+1}^{(k)} - \tau_k^* )^2 + (\kappa_{i+1}^{(k)} - \kappa_k^* )^2) \]
\[ \leq \beta(u_{i+1}^{(k)} - \tilde{u}_{i+1}^{(k)}, v_{i+1}^{(k)} - \tilde{v}_{i+1}^{(k)})^T (v_k^*, u_k^*) \quad (38) \]

With the definition of \( u_{i+1}^{(k)}, v_{i+1}^{(k)}, \tilde{u}_{i+1}^{(k)}, \tilde{v}_{i+1}^{(k)} \), we have

\[ \beta(p_{i+1}^{(k)}, q_{i+1}^{(k)}) - \beta(u_{i+1}^{(k)}, v_{i+1}^{(k)}) + \beta(u_{i+1}^{(k)}, v_{i+1}^{(k)}) \leq \partial I(Qu = v)[\tilde{u}_{i+1}^{(k)}, \tilde{v}_{i+1}^{(k)}] \]
\[ - \beta(p_{i+1}^{(k)}, q_{i+1}^{(k)}) \in \partial B(u_{i+1}^{(k)}, v_{i+1}^{(k)}, \mu^{(k)}) \]

Therefore

\[ 0 \leq \beta(u_{i+1}^{(k)} - u_k^*, \tilde{v}_{i+1}^{(k)} - v_k^*)^T (p_{i+1}^{(k)} - u_{i+1}^{(k)} + u_k^*, q_{i+1}^{(k)} - v_{i+1}^{(k)} + v_k^*) \]
\[ = \beta(u_{i+1}^{(k)} - u_k^*, \tilde{v}_{i+1}^{(k)} - v_k^*)^T (p_{i+1}^{(k)}, q_{i+1}^{(k)}) + \frac{\beta}{2} (||u_{i+1}^{(k)} - u_k^*||^2 + ||v_{i+1}^{(k)} - v_k^*||^2 - ||u_k^* - \tilde{u}_{i+1}^{(k)}||^2 - ||v_k^* - \tilde{v}_{i+1}^{(k)}||^2) \]
\[ - \frac{\beta}{2} (||u_{i+1}^{(k)} - u_k^*||^2 + ||v_{i+1}^{(k)} - v_k^*||^2 - ||u_{i+1}^{(k)} - \tilde{u}_{i+1}^{(k)}||^2 - ||v_{i+1}^{(k)} - \tilde{v}_{i+1}^{(k)}||^2) \]
\[ \leq - \beta(u_{i+1}^{(k)} - u_k^*, \tilde{v}_{i+1}^{(k)} - v_k^*)^T (p_{i+1}^{(k)}, q_{i+1}^{(k)}). \]
Summing up (39) and (40) gives

\[
B(u_{i+1}^{(k)}, v_{i+1}^{(k)}, \mu^{(k)}) - B(u_{i+1}^{*}, v_{i+1}^{*}, \mu^{(k)}) + \frac{\mu^{(k)}}{4D}(||y_{i+1}^{(k)} - x_{i+1}^{*}||^2 + ||s_{i+1}^{(k)} - s_{i+1}^{*}||^2 + (\tau_{i+1}^{(k)} - \tau_{i+1}^{*})^2 + (\kappa_{i+1}^{(k)} - \kappa_{i+1}^{*})^2)
\leq \beta(u_{i+1}^{(k)} - u_{i+1}^{*}) + \frac{\beta}{2}(||u_{i+1}^{(k)} - u_{i+1}^{*}||^2 + ||v_{i+1}^{(k)} - v_{i+1}^{*}||^2) - \beta \left(\frac{1}{2}||u_{i+1}^{(k)} - u_{i+1}^{*}||^2 + ||v_{i+1}^{(k)} - v_{i+1}^{*}||^2\right).
\]

Observing that

\[
(u_{i+1}^{(k)} - \tilde{u}_{i+1}^{(k)})^\top \left(v_{i+1}^{(k)} - \tilde{v}_{i+1}^{(k)}\right) = (u_{i+1}^{(k)} - u_{i+1}^{*})^\top \left(v_{i+1}^{(k)} - v_{i+1}^{*}\right)
\]

\[
= (u_{i+1}^{(k)} - u_{i+1}^{*})^\top \left(v_{i+1}^{*} - v_{i+1}^{(k)}\right) + (u_{i+1}^{(k)} - u_{i+1}^{*})^\top \left(v_{i+1}^{(k)} - v_{i+1}^{*}\right)
\]

\[
= (p_{i+1}^{(k)} - q_{i+1}^{(k)})\left(v_{i+1}^{*} - v_{i+1}^{(k)}\right) + (u_{i+1}^{(k)} - u_{i+1}^{*})^\top \left(v_{i+1}^{(k)} - u_{i+1}^{*}\right)
\]

\[
= \frac{1}{2}(||u_{i+1}^{(k)} - u_{i+1}^{*}||^2 + ||v_{i+1}^{(k)} - v_{i+1}^{*}||^2) - \frac{1}{2}(||u_{i+1}^{(k)} - u_{i+1}^{*}||^2 + ||v_{i+1}^{(k)} - v_{i+1}^{*}||^2).
\]

Summing (38), (41), and using (42), we have

\[
\frac{1}{2}(||u_{i+1}^{*} - u_{i+1}||^2 + ||v_{i+1}^{*} - v_{i+1}||^2) + \frac{\mu^{(k)}}{2C_{D}^2} \left(||y_{i+1}^{(k)} - x_{i+1}^{*}||^2 + ||s_{i+1}^{(k)} - s_{i+1}^{*}||^2 + (\tau_{i+1}^{(k)} - \tau_{i+1}^{*})^2 + (\kappa_{i+1}^{(k)} - \kappa_{i+1}^{*})^2\right)
\leq \frac{1}{2}(||u_{i+1}^{(k)} - u_{i+1}^{*}||^2 + ||v_{i+1}^{(k)} - v_{i+1}^{*}||^2) - \frac{1}{2}(||u_{i+1}^{(k)} - u_{i+1}^{*}||^2 + ||v_{i+1}^{(k)} - v_{i+1}^{*}||^2).
\]

Using the optimality condition of (9) we have

\[
0 \leq B(u, v, \mu^{(k)}) - B(u_{i+1}^{(k)}, v_{i+1}^{(k)}, \mu^{(k)}) + \beta(u - u_{i+1}^{(k)}, v - v_{i+1}^{(k)})^\top (p_{i+1}^{(k)}, q_{i+1}^{(k)})
\]

and

\[
0 \leq B(u, v, \mu^{(k)}) - B(u_{i+1}^{(k)}, v_{i+1}^{(k)}, \mu^{(k)}) + \beta(u - u_{i+1}^{(k)}, v - v_{i+1}^{(k)})^\top (p_{i+1}^{(k)}, q_{i+1}^{(k)}).
\]

Let \((u, v) = (u_{i+1}^{(k)}, v_{i+1}^{(k)})\) in (44) and \((u, v) = (u_{i+1}^{(k)}, v_{i+1}^{(k)})\) in (45), and sum up the two inequalities we have

\[
0 \leq -\left(u_{i+1}^{(k)} - u_{i+1}^{*}, v_{i+1}^{(k)} - v_{i+1}^{*}\right)^\top \left(p_{i+1}^{(k)} - p_{i+1}^{(k)}, q_{i+1}^{(k)} - q_{i+1}^{(k)}\right)
\]

\[
= -\left(u_{i+1}^{(k)} - u_{i+1}^{*}, v_{i+1}^{(k)} - v_{i+1}^{*}\right)^\top \left(u_{i+1}^{(k)} - u_{i+1}^{*}, v_{i+1}^{(k)} - v_{i+1}^{*}\right)
\]

\[
= \frac{1}{2} \left(||u_{i+1}^{(k)} - u_{i+1}^{*}||^2 + ||v_{i+1}^{(k)} - v_{i+1}^{*}||^2\right) - \frac{1}{2}(||u_{i+1}^{(k)} - u_{i+1}^{*}||^2 + ||v_{i+1}^{(k)} - v_{i+1}^{*}||^2),
\]

which implies that

\[
||u_{i+1}^{(k)} - u_{i+1}^{*}||^2 + ||v_{i+1}^{(k)} - v_{i+1}^{*}||^2 \leq ||u_{i+1}^{(k)} - u_{i+1}^{*}||^2 + ||v_{i+1}^{(k)} - v_{i+1}^{*}||^2.
\]
Summing up (43) and (47) we have

\[
\frac{1}{2} \left( \|u_{i+1}^{(k)} - \bar{u}_{i+1}^{(k)}\|^2 + \|v_{i+1}^{(k)} - \bar{v}_{i+1}^{(k)}\|^2 \right) \\
+ \frac{\mu^{(k)}}{2CD^2} \left( \|x_{i+1}^{(k)} - x_i^*\|^2 + \|s_{i+1}^{(k)} - s_i^*\|^2 + (\tau_{i+1}^{(k)} - \tau_i^*)^2 + (\kappa_{i+1}^{(k)} - \kappa_i^*)^2 \right) \\
\leq \|u_i^{(k)} - u_i^*\|^2 + \|v_i^{(k)} - v_i^*\|^2 - \|u_{i+1}^{(k)} - u_{i+1}^*\|^2 - \|v_{i+1}^{(k)} - v_{i+1}^*\|^2. 
\]

(48)

Using the linear constraint of (7) we have (by denoting \(C_4 = 6\lambda_{\text{max}}(A^T A) \max\{1, \|c\|^2, \|r_d\|^2\}/\lambda_{\text{min}}^2(A^T A))

\[
\|y_{i+1}^{(k)} - y_i^*\|^2 \\
\leq 2 \left( \|y_{i+1}^{(k)} - \tilde{y}_{i+1}^{(k)}\|^2 + \|y_i^* - \bar{y}_i^*\|^2 \right) \\
= 2\|y_{i+1}^{(k)} - \tilde{y}_{i+1}^{(k)}\|^2 + 2\|(A A^T)^{-1} A (A^T y_{i+1}^{(k)} - A^T y_i^*)\|^2 \\
\leq 2\|y_{i+1}^{(k)} - \tilde{y}_{i+1}^{(k)}\|^2 + C_4 \left( \|y_{i+1}^{(k)} - s_i^*\|^2 + (\tilde{y}_{i+1}^{(k)} - \tau_i^*)^2 + (\bar{\theta}_{i+1}^{(k)} - \theta_i^*)^2 \right) \\
\leq 2C_4 \left( \|s_i^* - \tilde{s}_i^*\|^2 + \|s_i^* - s_i^*\|^2 + \|\tilde{y}_{i+1}^{(k)} - \tau_{i+1}^*\|^2 + (\bar{\theta}_{i+1}^{(k)} - \theta_i^*)^2 \right) \\
+ C_4 (\bar{\theta}_{i+1}^{(k)} - \theta_i^*)^2 + 2\|y_{i+1}^{(k)} - \tilde{y}_{i+1}^{(k)}\|^2. 
\]

(49)

and (by denoting \(C_5 = 3\max\{\|b\|^2, \|A\|^2/\|r_p\|^2\})

\[
(\theta_{i+1}^{(k)} - \theta_i^*)^2 \\
= (\bar{\theta}_{i+1}^{(k)} - \theta_i^*)^2 = \frac{1}{\|r_p\|^2} \|b y_{i+1}^{(k)} - A s_{i+1}^{(k)} + \bar{r}_{i+1}^{(k)} - b r_i^* + A x_i^* - r_i^*\|^2 \\
\leq C_5 (\|y_{i+1}^{(k)} - s_i^*\|^2 + \|y_i^* - \tau_i^*\|^2 + \|\bar{r}_{i+1}^{(k)} - \bar{r}_i^*\|^2) \\
\leq 2C_5 (\|s_i^* - \tilde{s}_i^*\|^2 + \|x_i^* - \tilde{x}_i^*\|^2 + (\tau_{i+1}^* - \tilde{y}_{i+1}^{(k)})^2 + (\bar{\theta}_{i+1}^{(k)} - \theta_i^*)^2) + C_5 \|r_i^* - \bar{r}_i^*\|^2. 
\]

(50)

Summing up (49) and (50) we have

\[
\|y_{i+1}^{(k)} - y_i^*\|^2 + (\theta_{i+1}^{(k)} - \theta_i^*)^2 \\
\leq 2(C_4 + C_5) \left( \|u_{i+1}^{(k)} - \bar{u}_{i+1}^{(k)}\|^2 + \|v_{i+1}^{(k)} - \bar{v}_{i+1}^{(k)}\|^2 \right) \\
+ 2C_4 \left( \|y_{i+1}^{(k)} - \tilde{y}_{i+1}^{(k)}\|^2 + \|y_i^* - \tilde{y}_i^*\|^2 \right) \\
+ 2C_5 \left( \|x_{i+1}^{(k)} - \tilde{x}_{i+1}^{(k)}\|^2 + \|x_i^* - \tilde{x}_i^*\|^2 \right) \\
\leq 2(C_4 + C_5) \left( \|u_{i+1}^{(k)} - \bar{u}_{i+1}^{(k)}\|^2 + \|v_{i+1}^{(k)} - \bar{v}_{i+1}^{(k)}\|^2 \right) \\
+ 2C_4 \left( \|y_{i+1}^{(k)} - \tilde{y}_{i+1}^{(k)}\|^2 + \|y_i^* - \tilde{y}_i^*\|^2 \right) \\
+ 4C_4 C_5 (\|s_i^* - \tilde{s}_i^*\|^2 + \|\bar{r}_{i+1}^{(k)} - \bar{r}_i^*\|^2) \\
+ 4C_4 C_5 (\|x_i^* - \tilde{x}_i^*\|^2 + \|x_i^* - \tilde{x}_i^*\|^2 + (\tau_{i+1}^* - \tilde{y}_{i+1}^{(k)})^2 + (\bar{\theta}_{i+1}^{(k)} - \theta_i^*)^2) \\
+ 2C_4 C_5 \|\bar{r}_{i+1}^{(k)} - \bar{r}_i^*\|^2 \\
\leq C_\delta (\|u_{i+1}^{(k)} - \bar{u}_{i+1}^{(k)}\|^2) + C_\delta (\|x_{i+1}^{(k)} - \tilde{x}_{i+1}^{(k)}\|^2 + \|s_i^* - \tilde{s}_i^*\|^2 + (\tau_{i+1}^* - \tilde{y}_{i+1}^{(k)})^2). 
\]

(51)
Finally, we have

\[
\min \left\{ \frac{1}{C_3}, \frac{\mu^{(k)}}{2CD_3\beta} \right\} \left( ||\mathbf{u}_{i+1}^{(k)} - \mathbf{u}_i^*||^2 + ||\mathbf{v}_{i+1}^{(k)} - \mathbf{v}_i^*||^2 \right) \\
\leq \frac{\mu^{(k)}}{2CD_3\beta} \left( ||\mathbf{x}_{i+1}^{(k)} - \mathbf{x}_i^*||^2 + ||\mathbf{s}_{i+1}^{(k)} - \mathbf{s}_i^*||^2 + (\tau_{i+1}^{(k)} - \tau_k^*)^2 + (\kappa_{i+1}^{(k)} - \kappa_k^*)^2 \right) \\
+ \min \left\{ \frac{1}{C_3}, \frac{\mu^{(k)}}{2CD_3\beta} \right\} \left( ||\mathbf{y}_{i+1}^{(k)} - \mathbf{y}_i^*||^2 + (\theta_{i+1}^{(k)} - \theta_k^*)^2 \right) \\
\leq \frac{\mu^{(k)}}{2CD_3\beta} \left( ||\mathbf{x}_{i+1}^{(k)} - \mathbf{x}_i^*||^2 + ||\mathbf{s}_{i+1}^{(k)} - \mathbf{s}_i^*||^2 + (\tau_{i+1}^{(k)} - \tau_k^*)^2 + (\kappa_{i+1}^{(k)} - \kappa_k^*)^2 \right) \\
+ \frac{1}{2} \left( ||\mathbf{u}_{i+1}^{(k)} - \mathbf{u}_i^{(k)}||^2 + ||\mathbf{v}_{i+1}^{(k)} - \mathbf{v}_i^{(k)}||^2 \right) \\
\leq \frac{\mu^{(k)}}{2CD_3\beta} \left( ||\mathbf{x}_{i+1}^{(k)} - \mathbf{x}_i^*||^2 + ||\mathbf{s}_{i+1}^{(k)} - \mathbf{s}_i^*||^2 + (\tau_{i+1}^{(k)} - \tau_k^*)^2 + (\kappa_{i+1}^{(k)} - \kappa_k^*)^2 \right) \\
+ \frac{1}{2} \left( ||\mathbf{u}_{i+1}^{(k)} - \mathbf{u}_i^{(k)}||^2 + ||\mathbf{v}_{i+1}^{(k)} - \mathbf{v}_i^{(k)}||^2 \right) \\
\leq ||\mathbf{u}_{i+1}^{(k)} - \mathbf{u}_i^{(k)}||^2 + ||\mathbf{v}_{i+1}^{(k)} - \mathbf{v}_i^{(k)}||^2.
\]

Therefore, we have

\[
||\mathbf{u}_{Nk}^{(k)} - \mathbf{u}_k^*||^2 + ||\mathbf{v}_{Nk}^{(k)} - \mathbf{v}_k^*||^2 \leq \left( 1 + \min \left\{ \frac{1}{C_3}, \frac{\mu^{(k)}}{2CD_3\beta} \right\} \right)^{-N_k} \left( ||\mathbf{u}_0^{(k)} - \mathbf{u}_i^*||^2 + ||\mathbf{v}_0^{(k)} - \mathbf{v}_i^*||^2 \right) \\
\leq 2C_1 \left( 1 + \min \left\{ \frac{1}{C_3}, \frac{\mu^{(k)}}{2CD_3\beta} \right\} \right)^{-N_k}.
\]

Combining the termination criteria and the following inequality

\[
||\mathbf{Q}\mathbf{u}_{Nk}^{(k)} - \mathbf{v}_{Nk}^{(k)}||^2 \leq (2 + 2||\mathbf{Q}||^2)(||\mathbf{u}_{Nk}^{(k)} - \mathbf{u}_k^*||^2 + ||\mathbf{v}_{Nk}^{(k)} - \mathbf{v}_k^*||^2),
\]

we have

\[
4C_1(1 + ||\mathbf{Q}||^2) \left( 1 + \min \left\{ \frac{1}{C_3}, \frac{\mu^{(k)}}{2CD_3\beta} \right\} \right)^{-N_k} \geq \mu^{(k)},
\]

which completes our proof.

**C  Solve the proximal subproblem of second-order cone**

We shall solve (12) in two cases of second-order cone \(\mathbb{K}\): SOC and RSOC. The subproblem (12) is

\[
\mathbf{x}^* = \arg\min_{\mathbf{x} \in \text{int} \mathbb{K}} \lambda F(\mathbf{x}) + \frac{1}{2}||\mathbf{x} - \mathbf{z}||_2^2
\]

where \(\lambda > 0\), \(\mathbf{z}\) is a column vector and \(F(\mathbf{x})\) is the barrier function of \(\mathbb{K}\). The optimality condition is

\[
\lambda \nabla F(\mathbf{x}) + \mathbf{x} - \mathbf{z} = 0
\]

**C.1  Regular second-order cone**

Let \(\mathbf{x} = (t, \mathbf{x}) \in \mathbb{R} \times \mathbb{R}^n\) and \(\mathbb{K} = \left\{ \mathbf{x} : t \geq \sqrt{\mathbf{x}^T \mathbf{x}} \right\}\). Notice that \(F(\mathbf{x}) = -\log \left( t^2 - \mathbf{x}^T \mathbf{x} \right)\).
In the view of the optimality condition (57), we need to solve

\[- \frac{\lambda}{t^2 - \mathbf{x}^T \mathbf{x}} \begin{bmatrix} 2t \\ -2 \end{bmatrix} + \begin{bmatrix} t \\ -\mathbf{x} \end{bmatrix} - \frac{\zeta_t}{\zeta_x} \mathbf{x} = 0.\]

For brevity, let us denote

\[\delta = t^2 - \mathbf{x}^T \mathbf{x} > 0\] (58)

we have

\[(-2\lambda + \delta)t = \delta \zeta_t,\] (59)

\[(2\lambda + \delta)\mathbf{x} = \delta \zeta_x.\] (60)

In view of the relation (59), we consider two cases.

1) When \(\delta = 2\lambda\), from (59) and (60) we have \(\zeta_t = 0, \mathbf{x}^* = \frac{1}{2} \zeta_x\). Plugging the value of \(\mathbf{x}^*\) in (58) and combining it with \(\delta = 2\lambda\), we have

\[2\lambda = t^2 - \frac{1}{4} \zeta_x^T \zeta_x \Rightarrow t^* = \sqrt{2\lambda + \frac{1}{4} \zeta_x^T \zeta_x}.\]

2) When \(\delta \neq 2\lambda\), from (59) and (60) we have

\[\begin{cases} t = \frac{\delta}{\delta - 2\lambda} \zeta_t \\ \mathbf{x} = \frac{\delta}{\delta - 2\lambda} \zeta_x \end{cases}.\] (61)

Substitute \(t, \mathbf{x}\) in (58), we have the equation of \(\delta\)

\[\delta = \frac{\delta^2}{(\delta - 2\lambda)^2} \zeta_t^2 - \frac{\delta^2}{(\delta + 2\lambda)^2} \zeta_x^T \zeta_x\]

\[\Rightarrow (\delta + 2\lambda)^2(\delta - 2\lambda)^2 = \delta(\delta + 2\lambda)^2 \zeta_t^2 - \delta(\delta - 2\lambda)^2 \zeta_x^T \zeta_x.\]

Let \(\rho = \frac{\delta}{\lambda} \in (0, 2) \cup (2, +\infty)\), then we have the equation of \(\rho\)

\[(\rho + 2)^2(\rho - 2)^2 = \frac{\rho^2}{\lambda}(\rho + 2)^2 \zeta_t^2 - \frac{\rho^2}{\lambda}(\rho - 2)^2 \zeta_x^T \zeta_x\]

\[\Rightarrow \rho^4 - 8\rho^2 + 16 = \frac{\zeta_t^2 - \zeta_x^T \zeta_x}{\lambda} \rho^3 + \frac{4\zeta_t^2 + 4\zeta_x^T \zeta_x}{\lambda} \rho^2 + \frac{4\zeta_t^2 - 4\zeta_x^T \zeta_x}{\lambda} \rho\]

\[\Rightarrow \left(\rho^4 + \frac{16}{\rho^2}\right) - \frac{\zeta_t^2 - \zeta_x^T \zeta_x}{\lambda} \rho^3 - \frac{4\zeta_t^2 + 4\zeta_x^T \zeta_x}{\lambda} \rho^2 - \frac{4\zeta_t^2 - 4\zeta_x^T \zeta_x}{\lambda} \rho = 8 = 0\]

\[\Rightarrow \left(\rho^4 + \frac{16}{\rho^2}\right) - \frac{\zeta_t^2 - \zeta_x^T \zeta_x}{\lambda} \rho^3 - \frac{4\zeta_t^2 + 4\zeta_x^T \zeta_x}{\lambda} \rho^2 - \frac{4\zeta_t^2 - 4\zeta_x^T \zeta_x}{\lambda} \rho = 0 = 16.\]

Let \(\gamma = \rho + \frac{2}{\rho} > 4\), then we have the equation of \(\gamma\)

\[\gamma^2 - \frac{\zeta_t^2 - \zeta_x^T \zeta_x}{\lambda} \gamma - \frac{4\zeta_t^2 + 4\zeta_x^T \zeta_x}{\lambda} = 16 = 0.\] (62)

It is easy to prove that the equation above must have and only have one solution no less than 4, so we choose the larger solution

\[\gamma = \frac{\zeta_t^2 - \zeta_x^T \zeta_x}{\lambda} + \sqrt{\left(\frac{\zeta_t^2 - \zeta_x^T \zeta_x}{\lambda}\right)^2 + 4 \left(\frac{4\zeta_t^2 + 4\zeta_x^T \zeta_x}{\lambda} + 16\right)} \]

33
From (66) we have this value of $x$

\[ \rho_1 = \frac{\gamma - \sqrt{\gamma^2 - 16}}{2} < 2, \quad \rho_2 = \frac{\gamma + \sqrt{\gamma^2 - 16}}{2} > 2. \]

In order to make $t \geq 0$, from (61), $\rho$ should satisfy

\[ (\rho - 2)\zeta_t \geq 0. \]

Thus if $\zeta_t > 0$, we choose $\rho_2$; if $\zeta_t < 0$, we choose $\rho_1$. When $\zeta_t \neq 0$, from (61), we have

\[ t^* = \frac{\rho}{\rho_2 - 2} \zeta_t, \quad x^* = \frac{\rho}{\rho_2} x. \]

### C.2 Rotated second-order cone

Let $x = (\eta, \nu, \bar{x}) \in \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}^n$ and $K = \{ x : \eta \nu \geq \frac{1}{2} \bar{x}^T \bar{x}, \eta \geq 0, \nu \geq 0 \}$. Notice that $F(x) = -\log \left( \eta \nu - \frac{1}{2} \bar{x}^T \bar{x} \right)$.

In view of the optimality condition (57), we have

\[ -\frac{\lambda}{\eta \nu - \frac{1}{2} \bar{x}^T \bar{x}} \begin{bmatrix} \nu \\ \eta \\ -\bar{x} \end{bmatrix} = \begin{bmatrix} \zeta_x \\ \zeta_\eta \\ \zeta_\nu \end{bmatrix} = 0. \]

For brevity, let us denote

\[ \delta = \eta \nu - \frac{1}{2} \bar{x}^T \bar{x} > 0, \]

we have

\[ -\lambda \nu + \delta \eta = \delta \zeta_\eta, \]

\[ -\lambda \eta + \delta \nu = \delta \zeta_\nu, \]

\[ (\lambda + \delta) \bar{x} = \delta \zeta_x. \]

Due to the relation (64) and (65) we consider two cases.

1) When $\delta = \lambda$, from (64) and (65) we have $\zeta_\eta + \zeta_\nu = 0$. From (66) we have $x^* = \frac{1}{2} \zeta_x$. Plugging this value of $x^*$ in (63) and combining it with $\delta = \lambda$ and (64) gives

\[ \begin{cases} \eta - \nu = \zeta_\eta \\ \lambda = \eta \nu - \frac{1}{2} \zeta_x^T \zeta_x \end{cases} \Rightarrow \begin{cases} \eta^* = \frac{\zeta_\eta + \sqrt{\zeta_\eta^2 + 4(\lambda + \frac{1}{2} \zeta_x^T \zeta_x)}}{2} \\ \nu^* = \frac{-\zeta_\eta + \sqrt{\zeta_\eta^2 + 4(\lambda + \frac{1}{2} \zeta_x^T \zeta_x)}}{2} \end{cases}. \]

2) When $\delta \neq \lambda$, from (64) and (65) we have

\[ \begin{cases} \eta + \nu = \frac{\lambda}{\delta - \lambda} (\zeta_\eta + \zeta_\nu) \\ \eta - \nu = \frac{\lambda}{\delta + \lambda} (\zeta_\eta - \zeta_\nu) \end{cases} \Rightarrow \begin{cases} \eta = \frac{\delta^2 \zeta_\eta + 2 \lambda \zeta_\nu}{(\delta + \lambda)(\delta - \lambda)} \\ \nu = \frac{2 \lambda \zeta_\eta + \delta^2 \zeta_\nu}{(\delta + \lambda)(\delta - \lambda)} \end{cases}. \]

From (66) we have $\bar{x} = \frac{\delta \zeta_x}{\delta - \lambda \bar{x}}$. Substitute $\eta, \nu, \bar{x}$ in (63), we have the equation of $\delta$

\[ \delta = \frac{\delta^2 \zeta_\eta + 2 \lambda \zeta_\nu}{(\delta + \lambda)(\delta - \lambda)} (\zeta_\eta + \zeta_\nu) - \frac{1}{2} \delta^2 \frac{\delta^2 \zeta_\eta + 2 \lambda \zeta_\nu}{(\delta + \lambda)(\delta - \lambda)} \bar{x}^T \zeta_x \]

\[ = 2(\delta + \lambda)^2 (\delta - \lambda)^2 = 2 \delta (\delta \zeta_\eta + \lambda \zeta_\nu)(\lambda \zeta_\eta + \delta \zeta_\nu) - (\delta - \lambda)^2 \bar{x}^T \zeta_x. \]

34
Let $\rho = \frac{\delta}{\lambda} \in (0, 1) \cup (1, +\infty)$, then we have the equation of $\rho$

$$2(\rho + 1)^2(\rho - 1)^2 = 2\rho^2(\zeta_{\rho} + \zeta_\nu)(\zeta_{\rho} + \zeta_\nu\rho) - \rho^2(\rho - 1)^2$$
$$\Rightarrow 2(\rho^2 - 2\rho^2 + 1) = \frac{2\rho^2 - \zeta_{\rho} + \zeta_\nu^T \zeta_{\rho}}{\lambda} \rho^2 + \frac{2\rho^2 + 2\zeta_{\rho}^T \zeta_{\rho} \rho^2 + 2\zeta_{\rho} \zeta_\nu - \zeta_{\rho}^T \zeta_{\rho}}{\rho}$$
$$\Rightarrow 2\left(\rho + \frac{1}{\rho}\right)^2 - \frac{2\rho\zeta_{\rho} - \zeta_{\rho}^T \zeta_{\rho}}{\lambda} \left(\rho + \frac{1}{\rho}\right) - \frac{2\rho^2 + 2\zeta_{\rho}^T \zeta_{\rho}}{\rho} = 4 = 0$$
$$\Rightarrow 2\left(\rho + \frac{1}{\rho}\right)^2 - \frac{2\rho\zeta_{\rho} - \zeta_{\rho}^T \zeta_{\rho}}{\lambda} \left(\rho + \frac{1}{\rho}\right) - \frac{2\rho^2 + 2\zeta_{\rho}^T \zeta_{\rho}}{\rho} = 8 = 0.$$ 

Let $\gamma = \rho + \frac{1}{\rho} > 2$, then we have the equation of $\gamma$

$$\gamma^2 - \frac{2\rho\zeta_{\rho} - \zeta_{\rho}^T \zeta_{\rho}}{2\lambda} \gamma - \frac{\zeta_{\rho}^2 + \zeta_\nu^2 + \zeta_{\rho}^T \zeta_{\rho}}{\lambda} = 4 = 0. \quad (68)$$

It is easy to prove that the equation above must have and only have one solution no less than 2, so we choose the larger solution

$$\gamma = \frac{2\rho\zeta_{\rho} - \zeta_{\rho}^T \zeta_{\rho}}{2\lambda} \sqrt{\left(\frac{2\rho\zeta_{\rho} - \zeta_{\rho}^T \zeta_{\rho}}{2\lambda}\right)^2 + 4 \left(\frac{\zeta_{\rho}^2 + \zeta_\nu^2 + \zeta_{\rho}^T \zeta_{\rho}}{\lambda}\right)^2}$$

and then we can get the possible values of $\rho$

$$\rho_1 = \frac{\gamma - \sqrt{\gamma^2 - 4}}{2} < 1, \quad \rho_2 = \frac{\gamma + \sqrt{\gamma^2 - 4}}{2} > 1.$$ 

In order to ensure $\eta \geq 0$ and $\nu \geq 0$, from (67) $\rho$ should satisfy

$$(\rho - 1)(\rho \zeta_{\rho} + \zeta_\nu) \geq 0, \quad \text{and} \quad (\rho - 1)(\zeta_{\rho} + \rho \zeta_\nu) \geq 0.$$ 

Therefore, if $\zeta_{\rho} + \zeta_\nu > 0$, we set $\rho = \rho_2$; if $\zeta_{\rho} + \zeta_\nu < 0$, we set $\rho = \rho_1$. When $\zeta_{\rho} + \zeta_\nu \neq 0$, from (67) we have

$$\eta^* = \frac{\rho \zeta_{\rho} + \rho^2 \zeta_\nu}{(\rho + 1)(\rho - 1)}, \quad \nu^* = \frac{\rho \zeta_{\rho} + \rho^2 \zeta_\nu}{(\rho + 1)(\rho - 1)}, \quad x^* = \frac{\rho \zeta_{\rho}}{\rho + 1}.$$ 

D Specialized linear system solvers and two equivalent formulations for LASSO and SVM

To further improve the performance of ABIP on several important applications, we provide specialized linear system solvers that can effectively exploit the problem structure.

D.1 LASSO

Consider the LASSO problem:

$$\min_x \|A x - b\|^2 + \lambda \|x\|_1$$

where $A \in \mathbb{R}^{m \times n}$. 

35
**SOCP** reformulation for ABIP is:

\[
\begin{aligned}
\min \quad & \begin{bmatrix}
0 & 2 \\
\alpha \cdot 1_{2n \times 1} & \end{bmatrix}^T \begin{bmatrix}
w \\
z \\
y \\
x^+ \\
x^-
\end{bmatrix} \\
\text{s.t.} \quad & \begin{bmatrix}
1 & 0 & 0_{1 \times (m+2n)} \\
0_{m \times 1} & 0_{m \times 1} & I_m \\
0_{m \times 1} & 0_{m \times 1} & \tilde{A} - \bar{A}
\end{bmatrix} \begin{bmatrix}
w \\
z \\
y \\
x^+ \\
x^-
\end{bmatrix} = \begin{bmatrix}1 \\
b
\end{bmatrix}
\end{aligned}
\]

\( (w, z, y)^T \in \text{RSOC}_{2+m} \)

\( x^+, x^- \geq 0 \)

As discussed above, our aim is to solve \((I + AA^T)x = b\), and for LASSO,

\[ AA^T = \begin{bmatrix}
1 & 0_{1 \times m} \\
0_{m \times 1} & I_m + 2\tilde{A}\tilde{A}^T
\end{bmatrix}, \]

so we need to solve \((I_m + 2\tilde{A}\tilde{A}^T)x = b\), then we may reduce the dimension of factorization.

1. If \(n > m\), we directly perform cholesky or LDL factorization to \(I_m + 2\tilde{A}\tilde{A}^T\). In this case, we reduce the dimension of factorization from \(2m + 2n + 3\) to \(m\).

2. If \(n \leq m\), we apply the Sherman-Morrison-Woodbury formula again to \(I_m + 2\tilde{A}\tilde{A}^T\), we can get:

\[
(I_m + 2\tilde{A}\tilde{A}^T)^{-1} = I_m - 2\tilde{A}(I_m + 2\tilde{A}^T\tilde{A})^{-1}\tilde{A}^T \\
= I_m - \tilde{A}(0.5I_m + \tilde{A}^T\tilde{A})^{-1}\tilde{A}^T
\]

Then, we only need to perform Cholesky or LDL factorization to \(0.5I_m + \tilde{A}^T\tilde{A}\). In this case, we reduce the dimension of factorization from \(2m + 2n + 3\) to \(n\).

In a word, we only need to perform matrix factorization in the dimension of \(O(\min\{m, n\})\) instead of \(O(m + n)\).

**QP** reformulation for ABIP is:

\[
\min \quad \hat{x}^T \hat{A}^T \hat{A}\hat{x} + (\lambda \cdot 1_{2m} - 2\hat{A}^T b)^T \hat{x}
\]

where \( \hat{x} = \begin{bmatrix} x^+ \\
x^-
\end{bmatrix} \geq 0, \ x = x^+ - x^- \), \( \hat{A} = [\tilde{A} - \bar{A}] \)

**D.2 Support vector machines**

Consider the Support Vector Machine problem with training data \((x_i, y_i)_{1 \leq i \leq n}\):

\[
\begin{aligned}
\min \quad & \frac{1}{2} \|\hat{w}\|^2 + C \sum_{i=1}^{m} \xi_i \\
\text{s.t.} \quad & y_i \hat{x}_i^T \hat{w} + y_i \hat{b} + \xi_i \geq 1 \quad i = 1, \ldots, m \\
& \xi_i \geq 0 \quad i = 1, \ldots, m
\end{aligned}
\]
where each $x_i \in \mathbb{R}^n$. For brevity, let $X = [x_1, x_2, \ldots, x_m]^T \in \mathbb{R}^{m \times n}$ and $y = [y_1, y_2, \ldots, y_n]^T \in \mathbb{R}^n$ represent the feature matrix and the label vector, respectively.

**SOCP reformulation of SVM (69) for ABIP** is:

$$
\begin{align*}
\min \quad & \nu + C \sum_{i=1}^{l} \xi_i \\
\text{s.t.} \quad & \eta = 1 \\
& \tilde{A} \tilde{w}^+ - \tilde{A} \tilde{w}^- + \tilde{b}^+ y - \tilde{b}^- y + \xi - t = e_m \\
& \tilde{w} = \tilde{w}^+ - \tilde{w}^- \\
& \xi, \tilde{w}^+, \tilde{w}^-, \tilde{b}^+, \tilde{b}^-, t \geq 0 \\
& \eta \nu - \frac{1}{2} \| \tilde{w} \|^2 \geq 0,
\end{align*}
$$

where $e_m$ is the vector that all elements are one, and $\tilde{A} = \text{diag}(y) \cdot X$. Our formulation introduces new variables $\tilde{w}^+$ and $\tilde{w}^-$, which puts the matrix $\tilde{A}$ into the columns of positive orthant. An empirical advantage of such formulation is that ABIP can scale each column independently. Note that, however, such scaling cannot be accomplished for the initial formulation, as the matrix $\tilde{A}$ is in the columns of the RSOC and can only be scaled as a whole.

Consequently, we reformulate SVM to align with the ABIP standard input

$$
\begin{align*}
\min \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
& x \in K
\end{align*}
$$

where

$$A = \begin{bmatrix}
1 & 0 & \tilde{A} & -\tilde{A} & -y & I_m & -I_m \\
I_n & -I_n & I_n \\
\end{bmatrix},$$

$$x = \begin{bmatrix}
\eta \\
\nu \\
\tilde{w}^+ \\
\tilde{w}^- \\
\xi \\
t
\end{bmatrix},
$$

and $b = \begin{bmatrix}
e_m \\
0
\end{bmatrix} \in \mathbb{R}^{2n+2m+2}$. Therefore, we need to perform the LDL factorization of the following matrix:

$$K = \begin{bmatrix}
I_{m+n+1} & A \\
A^T & -I_{2m+3n+4}
\end{bmatrix},$$

where

$$A = \begin{bmatrix}
1 & \tilde{A} & -\tilde{A} & -y & I_m & -I_m \\
I_n & -I_n & I_n
\end{bmatrix}.$$
Note that there is an obvious factorization form of $K$:

\[
K = \begin{bmatrix}
I_{m+n+1} & -A \\
A^T & I_{2m+3n+4}
\end{bmatrix}
\begin{bmatrix}
I_{m+n+1} + AA^T & -I_{2m+3n+4} \\
-I_{2m+3n+4} & -A^T
\end{bmatrix}
\]

Then, it suffices to factorize the matrix:

\[
I_{m+n+1} + AA^T
= \begin{bmatrix}
2 & 2\tilde{A}^T + 2yy^T + 3I_{m} & -2\tilde{A} \\
2\tilde{A}^T + 2yy^T + F & X^T & G
\end{bmatrix}
\]

\[
= \begin{bmatrix}
1 & I_{m}XG^{-1} \\
1 & I_{n}
\end{bmatrix}
\begin{bmatrix}
2 & 2\tilde{A}^T + 2yy^T + F - XG^{-1}X^T & G \\
2 & 2\tilde{A}^T + 2yy^T + F - 4\tilde{A}E^{-1}E\tilde{A}^T & G
\end{bmatrix}
\begin{bmatrix}
1 & I_{m}G^{-1}X^T \\
1 & I_{n}
\end{bmatrix}
\]

where $X = -2\tilde{A}$, $F = 3I_{m}$, $G = 4I_{n}$. Let

\[
H = \begin{bmatrix}
2I_{n} - 4G^{-1} & 2
\end{bmatrix}
\quad \text{and} \quad
M = \begin{bmatrix}
\tilde{A} & y
\end{bmatrix}.
\]

It follows that $H$ is a diagonal matrix and

\[
2\tilde{A}\tilde{A}^T + 2yy^T + F - 4\tilde{A}E^{-1}E\tilde{A}^T = F + MHM^T.
\]

We consider two cases for factorization. 1) When $m \leq n + 1$, we directly factorize $F + MHM^T$. 2) When $m > n + 1$, we apply the Sherman-Morrison-Woodbury formula to obtain:

\[
(F + MHM^T)^{-1} = F^{-1} - F^{-1}M(H^{-1} + M^TF^{-1}M)^{-1}M^TF^{-1}.
\]

In the above case, we factorize $H^{-1} + M^TF^{-1}M$. Therefore, we reduce the dimension of matrix factorization from $3m + 4n + 5$ to $\min\{m, n + 1\}$.

**QP reformulation of SVM (69) for ABIP** is:

\[
\min \frac{\lambda}{2} \tilde{x}^T \begin{bmatrix}
I_{n} \\
0_{2m+1}
\end{bmatrix} \tilde{x} + \frac{1}{m} \begin{bmatrix}
0_{n+1} \\
I_{m} \\
0_{m}
\end{bmatrix}^T \tilde{x}
\]

s.t. \[
\begin{bmatrix}
\text{diag}(y) \\
X \\
I_{m} - I_{n}
\end{bmatrix} \tilde{x} = 1_{m}
\]

\[
\tilde{x} = \begin{bmatrix}
w \\
b \\
\xi \\
s
\end{bmatrix}, \quad \xi, s \geq 0.
\]