Optimization Methods and Software

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/goms20

An ADMM-based interior-point method for large-scale linear programming

Tianyi Lin , Shiqian Ma , Yinyu Ye & Shuzhong Zhang

To cite this article: Tianyi Lin , Shiqian Ma , Yinyu Ye & Shuzhong Zhang (2020): An ADMM-based interior-point method for large-scale linear programming, Optimization Methods and Software, DOI: 10.1080/10556788.2020.1821200

To link to this article: https://doi.org/10.1080/10556788.2020.1821200

Published online: 24 Sep 2020.

OPTIMIZATION METHODS & SOFTWARE
https://doi.org/10.1080/10556788.2020.1821200

Taylor & Francis
Taylor & Francis Group

An ADMM-based interior-point method for large-scale linear programming

Tianyi Lin[a], Shiqian Ma[b], Yinyu Ye[c] and Shuzhong Zhang[d,e]

[a]Department of Electrical Engineering and Computer Science, UC Berkeley, Berkeley, CA, USA; [b]Department of Mathematics, University of California, Davis, CA, USA; [c]Department of Management Science and Engineering, Stanford University, Stanford, CA, USA; [d]Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, MN, USA; [e]Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen, People's Republic of China

ABSTRACT

In this paper, we propose a new framework to implement interior point method (IPM) in order to solve some very large-scale linear programs (LPs). Traditional IPMs typically use Newton's method to approximately solve a subproblem that aims to minimize a log-barrier penalty function at each iteration. Due its connection to Newton's method, IPM is often classified as *second-order method* – a genre that is attached with stability and accuracy at the expense of scalability. Indeed, computing a Newton step amounts to solving a large system of linear equations, which can be efficiently implemented if the input data are reasonably sized and/or sparse and/or well-structured. However, in case the above premises fail, then the challenge still stands on the way for a traditional IPM. To deal with this challenge, one approach is to apply the iterative procedure, such as preconditioned conjugate gradient method, to solve the system of linear equations. Since the linear system is different in each iteration, it is difficult to find good pre-conditioner to achieve the overall solution efficiency. In this paper, an alternative approach is proposed. Instead of applying Newton's method, we resort to the alternating direction method of multipliers (ADMM) to approximately minimize the log-barrier penalty function at each iteration, under the framework of primal–dual path-following for a homogeneous self-dual embedded LP model. The resulting algorithm is an ADMM-Based Interior Point Method, abbreviated as ABIP in this paper. The new method inherits stability from IPM and scalability from ADMM. Because of its self-dual embedding structure, ABIP is set to solve any LP without requiring prior knowledge about its feasibility. We conduct extensive numerical experiments testing ABIP with large-scale LPs from NETLIB and machine learning applications. The results demonstrate that ABIP compares favourably with other LP solvers including SDPT3, MOSEK, DSDP-CG and SCS.

ARTICLE HISTORY
Received 21 August 2019
Accepted 5 September 2020

KEYWORDS
Linear programming; homogeneous self-dual embedding; interior-point method; central path following; ADMM; iteration complexity

CONTACT Shuzhong Zhang zhangs@umn.edu Department of Industrial and Systems Engineering, University of Minnesota, Minneapolis, MN 55455, USA; Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen, People's Republic of China

This paper is dedicated to the special issue in memory of Professor Masao Iri

Supplemental data for this article can be accessed here. https://doi.org/10.1080/10556788.2020.1821200

1. Introduction

By and large, traditional interior point method (IPM) for linear program (LP) is based on solving a sequence of log-barrier penalty subproblems using Newton's method [29,35,40,53]. It turns out that with a suitable penalty-parameter choice scheme, one step of Newton's method usually yields a very good initial solution for the next log-barrier penalty subproblem. As a result, the crux of IPMs boils down to the computation of Newton steps, which requires to solve system of linear equations (e.g.[20,29,40,54]). On the surface of it, computing Newton's direction then amounts to the Cholesky decomposition of a large (often ill-conditioned) matrix or even computing its inverse, which can be done when the dimensions are not exceedingly high, or the input data are sparse and/or well-structured. In general, however, computing Newton's direction is computationally expensive. Because of this connection to Newton's method, IPM is often classified as a second-order approach. Typical to the second-order methods, IPM is known to be stable and accurate, but computing Newton's direction remains a challenge when the problem is dense and large scale. On the other hand, recently there has been considerable research attention paid on the so-called first-order methods, in that no Newton direction is needed. One very successful first-order method is called alternating direction method of multipliers (ADMM), which is essentially a gradient-based algorithm for the dual of a linearly constrained optimization model; see, for example, [16,19,21,22,31] and the recent survey papers [7,15]. It turns out that the ADMM is highly scalable; however, it may suffer from numerical instability and it may take overly many iterations to compute an accurate solution. A natural question thus arises: Can we combine the benefits from both campuses? This paper aims to provide an affirmative answer to the afore question.

Before moving further to that question, let us first mention a beautiful technique which resolved a difficulty that baffled researchers in the early days of the IPM. The difficulty is that an IPM requires an interior feasible solution to begin with, but an LP may not even be feasible let alone availability of an interior feasible solution. To tackle this, Ye *et al.* [55] proposed a homogeneous self-dual (HSD) reformulation of the original LP, which contains all the information that one may possibly care to obtain: an optimal solution if it exists; or, if the problem is infeasible or unbounded, a certificate that proves the case. Interestingly, the HSD has a ready interior feasible solution, while an optimal solution is guaranteed to exist. A central path following algorithm was subsequently proposed in [55] to solve the HSD. Later, Xu *et al.* [48] proposed a simplified homogeneous self-dual model. However, in either cases computing the Newton directions remains to be the chore. In this paper, we propose to apply the ADMM to approximately solve the log-barrier penalty subproblems from the path-following scheme for the HSD. The resulting algorithm, ADMM-based IPM (abbreviated as ABIP henceforth), inherits advantages of both IPM and ADMM. It turns out that ABIP is robust, highly scalable and achieves high accuracy. Moreover, as a benefit inherited from HSD, ABIP finds both primal and dual optimal solutions if they exist; otherwise it finds a certificate proving primal or dual infeasibility.

There have been lots of efforts to efficiently compute Newton's step in IPMs. Existing approaches adopted in IPM for computing Newton's step include sparse matrix factorization, Krylov subspace method and preconditioned conjugate gradient method (cf. [6,10,13,18,25,26]). Very recently, Cui *et al.* [10] proposed a novel inner-iteration preconditioned Krylov subspace method which overcomes the severe ill-conditioning of

linear equations solved in the final phase of interior-point iterations. Despite several progresses along this direction, the performance of these methods still highly depends on the structure of the problem and the input data and suffers from the curse of dimensionality. For example, the system of linear equation data of the IPM is changing every iteration and becomes more and more ill-conditioned, so that it is typically hard to find a good preconditioner for the CG method. Moreover, often we have to solve more than one system of linear equations with different right-hand-side vectors. By investigating the structure of the problem in HSD, we discover that ADMM can be used to approximately solve the log-barrier penalty problems very efficiently. Though connected with the classical operator splitting methods in [16,17,19], renaissance of ADMM in recent years was due to its success in signal processing [9,50], image processing [23,51] and machine learning problems; see a recent survey paper [7] and the references therein. An interesting and positive discovery through our study reported in this paper is twofold: (1) we find that the overall IPM strategy such as central-path following greatly improve the stability and robustness of the variable-splitting approach; (2) it turns out that the log-barrier penalty subproblem under the central path following scheme for the HSD is well-structured and can be efficiently solved by ADMM.

Recently, there are several attempts on designing first-order methods for solving LPs (and conic programs); see [45,47,52,56]. Without an HSD framework, such methods are not suited for primal or dual infeasible problems. More recently, the ADMM-based solvers for LP have been explored prior to our method, e.g. a split conic solver (SCS) developed by O'Donoghue *et al.* [36], which applies ADMM to solve the homogeneous self-dual embedding of the original LP. Therefore, ABIP is similar to SCS in that they are both based on ADMM and HSD. However, conceptually ABIP is built within the IPM framework, so that it can be used as an optional solver when the data is large and dense for any existing IPM solver. It is our hope that, by combining ADMM with interior-point strength, the solver would have an additional machinery to improve its solution robustness. For example, the scheme of ABIP can easily incorporate the notion of step-sizes and wide neighbourhoods of the central path (cf. [43]). The efficacy of ABIP is confirmed by our extensive numerical tests on large-scale LPs from NETLIB and machine learning applications. Finally, we remark that extending ABIP to a more general convex conic optimization setting is straightforward.

1.1. Notation and organization

Throughout this paper, we denote vectors by bold lower case letters, e.g. $\mathbf{x}$, and matrices by regular upper case letters, e.g. X. The transpose of a real vector $\mathbf{x}$ is denoted as $\mathbf{x}^\top$. For a vector $\mathbf{x}$, and a matrix X, $\|\mathbf{x}\|$ and $\|X\|$ denote the ℓ_2 norm and the matrix spectral norm, respectively. For two symmetric matrices A and B, $A \succeq B$ indicates that $A-B$ is symmetric positive semi-definite. The superscript, e.g. $\mathbf{x}^t$, denotes iteration counter. $\log(\mathbf{x})$ denotes the natural logarithm of $\mathbf{x}$. $\mathbf{e}$ denotes the vector of all ones. e_j denotes the coordinate vector with jth entry being 1. $\mathbb{I}$ is an identity matrix with appropriate dimension. For two vectors $\mathbf{x}$ and $\mathbf{y}$, the Hadamard product is denoted as $\mathbf{x} \circ \mathbf{y} = (x_1 y_1, \ldots, x_n y_n)$.

The rest of the paper is organized as follows. In Section 2, we discuss some background of homogeneous self-dual embedding. In Section 3, we propose our ABIP method for solving the homogeneous self-dual embedding with log barrier functions. We also discuss how ABIP can be simplified and reduced to a matrix-free algorithm. The iteration complexity

of ABIP is also analysed. In Section 4, we propose several techniques that help improve the performance of ABIP in practice. In Section 5, we present extensive numerical results on large-scale LPs from NETLIB and machine learning applications and compare with several existing LP solvers. We make some concluding remarks in Section 6.

2. Homogeneous and self-dual linear programming

We are interested in solving the following *primal-dual* pair of linear programs (LP):

$$\begin{array}{lllll} & \min & \mathbf{c}^\top \mathbf{x} & & \max \quad \mathbf{b}^\top \mathbf{y} \\ (P) & \text{s.t.} & A\mathbf{x} = \mathbf{b}, & (D) & \text{s.t.} \quad A^\top \mathbf{y} + \mathbf{s} = \mathbf{c}, \\ & & \mathbf{x} \geq 0, & & \mathbf{s} \geq 0, \end{array} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the primal variable, $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{s} \in \mathbb{R}^n$ are the dual variables, the problem data are $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^n$ with $m \leq n$, and without loss of generality, we assume that A is of full row rank. The primal and dual optimal objective values are denoted as p^* and d^* respectively.

In addition to the celebrated simplex method of Dantzig [11] in 1940s, the interior point method (IPM), which was pioneered by Karmarkar [30] and intensively developed by many researchers in the 1980s and 1990s, has been a standard approach to solve linear program (1). In the early years of IPM, initial feasible interior solutions were assumed to be available at hand. Clearly, this assumption can be restrictive. To address this issue specifically, Ye *et al.* [55] proposed to solve the following homogeneous and self-dual linear programming with arbitrary initial points $\mathbf{x}^0 > 0$, $\mathbf{s}^0 > 0$ and $\mathbf{y}^0$:

$$\begin{array}{lll} \text{(HSD)} \quad \min & ((\mathbf{x}^0)^\top \mathbf{s}^0 + 1)\theta & \\ \text{s.t.} & A\mathbf{x} - \mathbf{b}\tau + \bar{\mathbf{b}}\theta = 0, & \\ & -A^\top \mathbf{y} + \mathbf{c}\tau - \mathbf{s} - \bar{\mathbf{c}}\theta = 0, & \\ & \mathbf{b}^\top \mathbf{y} - \mathbf{c}^\top \mathbf{x} - \kappa + \bar{\mathbf{z}}\theta = 0, & \\ & -\bar{\mathbf{b}}^\top \mathbf{y} + \bar{\mathbf{c}}^\top \mathbf{x} - \bar{\mathbf{z}}\tau = -(\mathbf{x}^0)^\top \mathbf{s}^0 - 1, & \\ & \mathbf{y} \text{ free}, \ \mathbf{x} \geq 0, \ \tau \geq 0, \ \mathbf{s} \geq 0, \ \kappa \geq 0, \ \theta \text{ free}, & \end{array} \tag{2}$$

where

$$\bar{\mathbf{b}} = \mathbf{b} - A\mathbf{x}^0, \quad \bar{\mathbf{c}} = \mathbf{c} - A^\top \mathbf{y}^0 - \mathbf{s}^0, \quad \bar{\mathbf{z}} = \mathbf{c}^\top \mathbf{x}^0 + 1 - \mathbf{b}^\top \mathbf{y}^0. \tag{3}$$

The HSD (2) has many nice properties. In the following, we give a partial list (cf. [55]).

Theorem 2.1 (Theorem 2 in [55]): *The following holds for* (2):

(i) *The optimal value of* (2) *is zero, and for any feasible point* $(\mathbf{y}, \mathbf{x}, \tau, \theta, \mathbf{s}, \kappa)$, *it holds*

$$((\mathbf{x}^0)^\top \mathbf{s}^0 + 1) \cdot \theta = \mathbf{x}^\top \mathbf{s} + \tau\kappa.$$

(ii) *There is a feasible solution* $(\mathbf{y}, \mathbf{x}, \tau, \theta, \mathbf{s}, \kappa)$ *to* (2) *such that*

$$\mathbf{y} = \mathbf{y}^0, \quad \mathbf{x} = \mathbf{x}^0, \quad \tau = 1, \quad \theta = 1, \quad \mathbf{s} = \mathbf{s}^0, \quad \kappa = 1.$$

(iii) *There is an optimal solution* $(\mathbf{y}^*, \mathbf{x}^*, \tau^*, \theta^* = 0, \mathbf{s}^*, \kappa^*)$ *such that* $\mathbf{x}^* + \mathbf{s}^* > 0$ *and* $\tau^* + \kappa^* > 0$, *which is called a strictly complementary solution.*

Theorem 2.2 (Theorem 3 in [55]): *Let* $(\mathbf{y}^*, \mathbf{x}^*, \tau^*, \theta^* = 0, \mathbf{s}^*, \kappa^*)$ *be a strictly complementary solution for* (2). *Then*:

(i) *(P) has a solution (feasible and bounded) if and only if* $\tau^* > 0$. *In this case,* $\mathbf{x}^*/\tau^*$ *is an optimal solution for (P) and* $(\mathbf{y}^*/\tau^*, \mathbf{s}^*/\tau^*)$ *is an optimal solution for (D).*
(ii) *If* $\tau^* = 0$, *then* $\kappa^* > 0$, *which implies that* $\mathbf{c}^\top \mathbf{x}^* - \mathbf{b}^\top \mathbf{y}^* < 0$, *i.e. at least one of* $\mathbf{c}^\top \mathbf{x}^*$ *and* $-\mathbf{b}^\top \mathbf{y}^*$ *is strictly less than zero. If* $\mathbf{c}^\top \mathbf{x}^* < 0$, *then (D) is infeasible*; *if* $-\mathbf{b}^\top \mathbf{y}^* < 0$, *then (P) is infeasible*; *and if both* $\mathbf{c}^\top \mathbf{x}^* < 0$ *and* $-\mathbf{b}^\top \mathbf{y}^*$, *then both (P) and (D) are infeasible.*

Theorem 2.3 (Corollary 4 in [55]): *Let* $(\bar{\mathbf{y}}, \bar{\mathbf{x}}, \bar{\tau}, \bar{\theta} = 0, \bar{\mathbf{s}}, \bar{\kappa})$ *be any optimal solution for* (2). *If* $\bar{\kappa} > 0$, *then either (P) or (D) is infeasible.*

3. An ADMM-based interior-point method

In [55], Ye *et al.* proposed an $O(\sqrt{n}\log(\frac{1}{\varepsilon}))$-iteration and $O(n^3 \log(\frac{1}{\varepsilon}))$-arithmetic operation interior point algorithm to solve (2). However, like all interior-point methods, it requires solving a linear system at each iteration and therefore does not scale well for dense data. In this section, we propose our ABIP method which uses ADMM to solve the log-barrier penalty subproblems for HSD, and we show that the procedures of ABIP can be simplified. In this section, we also provide an iteration complexity analysis for ABIP.

3.1. The ABIP method

For ease of presentation, we choose $\mathbf{y}^0 = 0, \mathbf{x}^0 = \mathbf{e}$ and $\mathbf{s}^0 = \mathbf{e}$, where $\mathbf{e}$ denotes the vector of all ones. By introducing a constant parameter $\beta > 0$ and constant variables $\mathbf{r} = 0$ and $\xi = -(\mathbf{x}^0)^\top \mathbf{s}^0 - 1 = -n - 1$, (2) can be rewritten as

$$\begin{array}{ll} \min & \beta(n+1)\theta + \mathbb{1}(\mathbf{r} = 0) + \mathbb{1}(\xi = -n-1) \\ \text{s.t.} & Q\mathbf{u} = \mathbf{v}, \\ & \mathbf{y} \text{ free}, \ \mathbf{x} \geq 0, \ \tau \geq 0, \ \theta \text{ free}, \ \mathbf{s} \geq 0, \ \kappa \geq 0, \end{array} \tag{4}$$

where

$$Q = \begin{bmatrix} 0 & A & -\mathbf{b} & \bar{\mathbf{b}} \\ -A^\top & 0 & \mathbf{c} & -\bar{\mathbf{c}} \\ \mathbf{b}^\top & -\mathbf{c}^\top & 0 & \bar{\mathbf{z}} \\ -\bar{\mathbf{b}}^\top & \bar{\mathbf{c}}^\top & -\bar{\mathbf{z}} & 0 \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \\ \tau \\ \theta \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \mathbf{r} \\ \mathbf{s} \\ \kappa \\ \xi \end{bmatrix},$$

$$\bar{\mathbf{b}} = \mathbf{b} - A\mathbf{e}, \quad \bar{\mathbf{c}} = \mathbf{c} - \mathbf{e}, \quad \bar{\mathbf{z}} = \mathbf{c}^\top \mathbf{e} + 1, \tag{5}$$

and the indicator function $\mathbb{1}(\mathcal{C})$ equals zero if the constraint $\mathcal{C}$ is satisfied, and equals $+\infty$ otherwise. The reason that we introduce a parameter β in the objective is completely for ease of presentation. It does not change the solution of the problem.

One classical way to solve (4) is to use log-barrier penalty to penalize the variables with non-negativity constraints, which results in a primal–dual interior-point method. The new formulation with log-barrier penalty is

$$\begin{array}{ll} \min & B(\mathbf{u},\mathbf{v},\mu), \\ \text{s.t.} & Q\mathbf{u}=\mathbf{v}, \end{array} \tag{6}$$

where $B(\mathbf{u},\mathbf{v},\mu)$ is a barrier function defined as follows:

$$\begin{aligned} B(\mathbf{u},\mathbf{v},\mu) = \beta(n+1)\theta + \mathbb{1}(\mathbf{r}=0) + \mathbb{1}(\xi=-n-1) - \mu\sum_i \log(\mathbf{x}_i) \\ - \mu\sum_i \log(\mathbf{s}_i) - \mu\log(\tau) - \mu\log(\kappa), \end{aligned} \tag{7}$$

and $\mu > 0$ is the penalty parameter. In the kth iteration of IPM, one uses Newton's method to solve the KKT system of (6) with $\mu = \mu^k$. One then reduces μ^k to μ^{k+1} for the next iteration. When $\mu^k \to 0$, the solution of (6) approaches that of (4). The computational bottleneck of IPM is that one has to assemble a Newton's direction, which can be expensive when the problem is large and data are dense.

Observing the structure of (6), we propose to use the Alternating Direction Method of Multipliers (ADMM) to solve it inexactly. To do so, we first rewrite (6) as the following problem by introducing auxiliary variables $(\tilde{\mathbf{u}},\tilde{\mathbf{v}})$:

$$\begin{array}{ll} \min & \mathbb{1}(Q\tilde{\mathbf{u}}=\tilde{\mathbf{v}}) + B(\mathbf{u},\mathbf{v},\mu^k), \\ \text{s.t.} & (\tilde{\mathbf{u}},\tilde{\mathbf{v}}) = (\mathbf{u},\mathbf{v}). \end{array} \tag{8}$$

By associating (scaled) Lagrange multipliers $\mathbf{p}$ to constraint $\tilde{\mathbf{u}}=\mathbf{u}$ and $\mathbf{q}$ to constraint $\tilde{\mathbf{v}}=\mathbf{v}$, the augmented Lagrangian function for (8) can be written as

$$\begin{aligned} \mathcal{L}_\beta(\tilde{\mathbf{u}},\tilde{\mathbf{v}},\mathbf{u},\mathbf{v},\mu^k,\mathbf{p},\mathbf{q}) := \mathbb{1}(Q\tilde{\mathbf{u}}=\tilde{\mathbf{v}}) + B(\mathbf{u},\mathbf{v},\mu^k) - \langle\beta(\mathbf{p},\mathbf{q}),(\tilde{\mathbf{u}},\tilde{\mathbf{v}})-(\mathbf{u},\mathbf{v})\rangle \\ + \frac{\beta}{2}\|(\tilde{\mathbf{u}},\tilde{\mathbf{v}})-(\mathbf{u},\mathbf{v})\|^2, \end{aligned}$$

where $\beta > 0$ is the same parameter as in (4). The ith iteration of ADMM for solving (8) is as follows:

$$(\tilde{\mathbf{u}}_{i+1}^k,\tilde{\mathbf{v}}_{i+1}^k) = \operatorname*{argmin}_{\tilde{\mathbf{u}},\tilde{\mathbf{v}}} \mathcal{L}_\beta(\tilde{\mathbf{u}},\tilde{\mathbf{v}},\mathbf{u}_i^k,\mathbf{v}_i^k,\mu^k,\mathbf{p}_i^k,\mathbf{q}_i^k) = \prod_{Q\mathbf{u}=\mathbf{v}} (\mathbf{u}_i^k+\mathbf{p}_i^k,\mathbf{v}_i^k+\mathbf{q}_i^k), \tag{9}$$

$$(\mathbf{u}_{i+1}^k,\mathbf{v}_{i+1}^k) = \operatorname*{argmin}_{\mathbf{u},\mathbf{v}} \mathcal{L}_\beta(\tilde{\mathbf{u}}_{i+1}^k,\tilde{\mathbf{v}}_{i+1}^k,\mathbf{u},\mathbf{v},\mu^k,\mathbf{p}_i^k,\mathbf{q}_i^k), \tag{10}$$

$$(\mathbf{p}_{i+1}^k,\mathbf{q}_{i+1}^k) = (\mathbf{p}_i^k,\mathbf{q}_i^k) - (\tilde{\mathbf{u}}_{i+1}^k,\tilde{\mathbf{v}}_{i+1}^k) + (\mathbf{u}_{i+1}^k,\mathbf{v}_{i+1}^k), \tag{11}$$

where $\prod_{\mathcal{S}}(\mathbf{x})$ denotes the Euclidean projection of $\mathbf{x}$ onto the set $\mathcal{S}$. A generic description of our proposed approach – ABIP – is sketched as Algorithm 1.

3.2. Implementing ABIP

In this section, we discuss the detailed implementation of ABIP. In particular, we show that the dual variables $\mathbf{p}$ and $\mathbf{q}$ in (9)–(11) can be eliminated using a proper initialization.

Algorithm 1 The Basic Algorithmic Framework of ABIP for Linear Programming

Given parameters $\beta > 0$ and $\gamma \in (0,1)$. Set initial points $(\mathbf{u}_0^0, \mathbf{v}_0^0)$, $(\mathbf{p}_0^0, \mathbf{q}_0^0)$ and $\mu^0 > 0$.
for $k = 0, 1, 2, \ldots$ **do**
  for $i = 0, 1, 2, \ldots$ **do**
    if the termination criterion is satisfied **then**
      break.
    end if
    Update $(\tilde{\mathbf{u}}_{i+1}^k, \tilde{\mathbf{v}}_{i+1}^k)$ by (9);
    Update $(\mathbf{u}_{i+1}^k, \mathbf{v}_{i+1}^k)$ by (10);
    Update $(\mathbf{p}_{i+1}^k, \mathbf{q}_{i+1}^k)$ by (11).
  end for
  Set $(\mathbf{u}_0^{k+1}, \mathbf{v}_0^{k+1}) = (\mathbf{u}_{i+1}^k, \mathbf{v}_{i+1}^k)$ and $(\mathbf{p}_0^{k+1}, \mathbf{q}_0^{k+1}) = (\mathbf{p}_{i+1}^k, \mathbf{q}_{i+1}^k)$;
  Set $\mu^{k+1} = \gamma \mu^k$.
end for

The framework of our analysis is similar to the one in [36], but the techniques we use are quite different because the *Moreau decomposition* cannot be directly applied to $(\mathbf{u}, \mathbf{v})$ when the log-barrier penalty function is used. The main technical result is summarized in the following theorem.

Theorem 3.1: *For the kth outer iteration of Algorithm* 1, *we initialize* $\mathbf{p}_0^k = \mathbf{v}_0^k$ *and* $\mathbf{q}_0^k = \mathbf{u}_0^k$ *with*

$$\mathbf{x}_0^k \circ \mathbf{s}_0^k = \frac{\mu^k}{\beta}\mathbf{e}, \quad \tau_0^k \kappa_0^k = \frac{\mu^k}{\beta}, \quad \mathbf{r}_0^k = 0, \quad \xi_0^k = -n - 1.$$

It then holds, for all iterations $i \geq 0$, *that*

$$\mathbf{p}_i^k = \mathbf{v}_i^k, \quad \mathbf{q}_i^k = \mathbf{u}_i^k, \quad \mathbf{x}_i^k \circ \mathbf{s}_i^k = \frac{\mu^k}{\beta}\mathbf{e}, \quad \tau_i^k \kappa_i^k = \frac{\mu^k}{\beta}, \quad \mathbf{r}_i^k = 0, \quad \xi_i^k = -n - 1. \tag{12}$$

Proof: We shall prove the result by induction. Indeed, the proof is based on the following steps: (i) Iteration $j = 0$: the result holds true since we can initialize the variables accordingly. (ii) The result holds true for iteration $j = i + 1$ given that it holds true for iteration $j = i$. We prove the desired result in two steps:

Step 1: We claim that

$$\mathbf{u}_i^k + \mathbf{v}_i^k = \tilde{\mathbf{u}}_{i+1}^k + \tilde{\mathbf{v}}_{i+1}^k. \tag{13}$$

Indeed, we rewrite (9) as

$$(\tilde{\mathbf{u}}_{i+1}^k, \tilde{\mathbf{v}}_{i+1}^k) = \prod_{\mathcal{Q}} (\mathbf{u}_i^k + \mathbf{v}_i^k, \mathbf{u}_i^k + \mathbf{v}_i^k), \tag{14}$$

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

$$(\mathbf{v}, \mathbf{u}) = \prod_{\mathcal{Q}^\perp} (\mathbf{z}, \mathbf{z}), \quad \text{if } (\mathbf{u}, \mathbf{v}) = \prod_{\mathcal{Q}} (\mathbf{z}, \mathbf{z}),$$

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

$$(\tilde{\mathbf{v}}_{i+1}^k, \tilde{\mathbf{u}}_{i+1}^k) = \prod_{\mathcal{Q}^\perp} (\mathbf{u}_i^k + \mathbf{v}_i^k, \mathbf{u}_i^k + \mathbf{v}_i^k). \tag{15}$$

Therefore, combining (14) and (15) yields the desired result.

Step 2: We proceed to proving that

$$\mathbf{p}_{i+1}^k = \mathbf{v}_{i+1}^k, \quad \mathbf{q}_{i+1}^k = \mathbf{u}_{i+1}^k, \quad \mathbf{x}_{i+1}^k \circ \mathbf{s}_{i+1}^k = \frac{\mu^k}{\beta}\mathbf{e}, \quad \tau_{i+1}^k \kappa_{i+1}^k = \frac{\mu^k}{\beta}, \quad \xi_{i+1}^k = -n-1,$$

given

$$\mathbf{p}_i^k = \mathbf{v}_i^k, \quad \mathbf{q}_i^k = \mathbf{u}_i^k \tag{16}$$

and

$$\mathbf{x}_i^k \circ \mathbf{s}_i^k = \frac{\mu^k}{\beta}\mathbf{e}, \quad \tau_i^k \kappa_i^k = \frac{\mu^k}{\beta}, \quad \xi_i^k = -n-1. \tag{17}$$

Indeed, we partition $\mathbf{p}$ and $\mathbf{q}$ as

$$\mathbf{p} = \begin{bmatrix} \mathbf{p_y} \\ \mathbf{p_x} \\ \mathbf{p}_\tau \\ \mathbf{p}_\theta \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} \mathbf{q_r} \\ \mathbf{q_s} \\ \mathbf{q}_\kappa \\ \mathbf{q}_\xi \end{bmatrix},$$

and the optimality conditions of (10) are given by

$$0 = \mathbf{y}_{i+1}^k - \tilde{\mathbf{y}}_{i+1}^k + (\mathbf{p_y})_i^k, \tag{18}$$

$$0 = -\frac{\mu^k}{\beta} \cdot \frac{1}{\mathbf{x_{i+1}^k}} + \mathbf{x}_{i+1}^k - \tilde{\mathbf{x}}_{i+1}^k + (\mathbf{p_x})_i^k, \tag{19}$$

$$0 = -\frac{\mu^k}{\beta} \cdot \frac{1}{\tau_{i+1}^k} + \tau_{i+1}^k - \tilde{\tau}_{i+1}^k + (\mathbf{p}_\tau)_i^k, \tag{20}$$

$$0 = (n+1) + \theta_{i+1}^k - \tilde{\theta}_{i+1}^k + (\mathbf{p}_\theta)_i^k, \tag{21}$$

$$0 = \mathbf{r}_{i+1}^k, \tag{22}$$

$$0 = -\frac{\mu^k}{\beta} \cdot \frac{1}{\mathbf{s}_{i+1}^k} + \mathbf{s}_{i+1}^k - \tilde{\mathbf{s}}_{i+1}^k + (\mathbf{q_s})_i^k, \tag{23}$$

$$0 = -\frac{\mu^k}{\beta} \cdot \frac{1}{\kappa_{i+1}^k} + \kappa_{i+1}^k - \tilde{\kappa}_{i+1}^k + (\mathbf{q}_\kappa)_i^k, \tag{24}$$

$$0 = \xi_{i+1}^k + n + 1. \tag{25}$$

First, we show that

$$\mathbf{r}_{i+1}^k = (\mathbf{p}_\mathbf{y})_{i+1}^k = 0, \quad \mathbf{y}_{i+1}^k = (\mathbf{q}_\mathbf{r})_{i+1}^k.$$

Indeed, from (11), (18) and (22) we have that

$$(\mathbf{p}_\mathbf{y})_{i+1}^k = (\mathbf{p}_\mathbf{y})_i^k - \tilde{\mathbf{y}}_{i+1}^k + \mathbf{y}_{i+1}^k = 0 = \mathbf{r}_{i+1}^k.$$

Furthermore, we have

$$\begin{aligned}(\mathbf{q}_\mathbf{r})_{i+1}^k &\overset{(11)}{=} (\mathbf{q}_\mathbf{r})_i^k - \tilde{\mathbf{r}}_{i+1}^k + \mathbf{r}_{i+1}^k \overset{(22)}{=} (\mathbf{q}_\mathbf{r})_i^k - \tilde{\mathbf{r}}_{i+1}^k \overset{(13)}{=} (\mathbf{q}_\mathbf{r})_i^k - (\mathbf{r}_i^k + \mathbf{y}_i^k - \tilde{\mathbf{y}}_{i+1}^k) \\ &\overset{(18)}{=} (\mathbf{q}_\mathbf{r})_i^k - (\mathbf{r}_i^k + \mathbf{y}_i^k - \mathbf{y}_{i+1}^k - (\mathbf{p}_\mathbf{y})_i^k) \overset{(16)}{=} \mathbf{y}_i^k - (\mathbf{r}_i^k + \mathbf{y}_i^k - \mathbf{y}_{i+1}^k - \mathbf{r}_i^k) = \mathbf{y}_{i+1}^k.\end{aligned}$$

Second, we shall prove

$$\mathbf{x}_{i+1}^k = (\mathbf{q}_\mathbf{s})_{i+1}^k, \quad \mathbf{s}_{i+1}^k = (\mathbf{p}_\mathbf{x})_{i+1}^k, \quad \mathbf{x}_{i+1}^k \circ \mathbf{s}_{i+1}^k = \frac{\mu^k}{\beta} \cdot \mathbf{e}.$$

Indeed, from (11) and (16) we have

$$(\mathbf{p}_\mathbf{x})_{i+1}^k = (\mathbf{p}_\mathbf{x})_i^k - \tilde{\mathbf{x}}_{i+1}^k + \mathbf{x}_{i+1}^k = \mathbf{s}_i^k - \tilde{\mathbf{x}}_{i+1}^k + \mathbf{x}_{i+1}^k,$$

and

$$(\mathbf{q}_\mathbf{s})_{i+1}^k = (\mathbf{q}_\mathbf{s})_i^k - \tilde{\mathbf{s}}_{i+1}^k + \mathbf{s}_{i+1}^k = \mathbf{x}_i^k - \tilde{\mathbf{s}}_{i+1}^k + \mathbf{s}_{i+1}^k.$$

Combining the above two equations with (13) yields

$$(\mathbf{p}_\mathbf{x})_{i+1}^k + (\mathbf{q}_\mathbf{s})_{i+1}^k = \mathbf{x}_{i+1}^k + \mathbf{s}_{i+1}^k. \tag{26}$$

Besides, from (19) and (11) we have

$$\frac{\mu^k}{\beta} \cdot \mathbf{e} = \mathbf{x}_{i+1}^k \circ (\mathbf{x}_{i+1}^k - \tilde{\mathbf{x}}_{i+1}^k + (\mathbf{p}_\mathbf{x})_i^k) = \mathbf{x}_{i+1}^k \circ (\mathbf{p}_\mathbf{x})_{i+1}^k, \tag{27}$$

and from (23) and (11) we have

$$\frac{\mu^k}{\beta} \cdot \mathbf{e} = \mathbf{s}_{i+1}^k \circ (\mathbf{s}_{i+1}^k - \tilde{\mathbf{s}}_{i+1}^k + (\mathbf{q}_\mathbf{s})_i^k) = \mathbf{s}_{i+1}^k \circ (\mathbf{q}_\mathbf{s})_{i+1}^k. \tag{28}$$

Therefore, we obtain

$$\begin{aligned}0 &\overset{(27),(28)}{=} \mathbf{x}_{i+1}^k \circ (\mathbf{p}_\mathbf{x})_{i+1}^k - \mathbf{s}_{i+1}^k \circ (\mathbf{q}_\mathbf{s})_{i+1}^k \\ &\overset{(26)}{=} \mathbf{x}_{i+1}^k \circ (\mathbf{x}_{i+1}^k + \mathbf{s}_{i+1}^k - (\mathbf{q}_\mathbf{s})_{i+1}^k) - \mathbf{s}_{i+1}^k \circ (\mathbf{q}_\mathbf{s})_{i+1}^k \\ &= (\mathbf{x}_{i+1}^k - (\mathbf{q}_\mathbf{s})_{i+1}^k) \circ (\mathbf{x}_{i+1}^k + \mathbf{s}_{i+1}^k).\end{aligned}$$

Since $\mathbf{x}_{i+1}^k + \mathbf{s}_{i+1}^k > 0$, we conclude that $\mathbf{x}_{i+1}^k = (\mathbf{q}_\mathbf{s})_{i+1}^k$ which, combining with (28), leads to

$$\frac{\mu^k}{\beta} \cdot \mathbf{e} = \mathbf{x}_{i+1}^k \circ \mathbf{s}_{i+1}^k.$$

It also directly follows from (26) that $\mathbf{s}_{i+1}^k = (\mathbf{p}_\mathbf{x})_{i+1}^k$. We use the same arguments to conclude

$$\tau_{i+1}^k = (\mathbf{q}_\kappa)_{i+1}^k, \quad \kappa_{i+1}^k = (\mathbf{p}_\tau)_{i+1}^k, \quad \tau_{i+1}^k \kappa_{i+1}^k = \frac{\mu^k}{\beta}.$$

Finally, we show that

$$\xi_{i+1}^k = (\mathbf{p}_\theta)_{i+1}^k = -n - 1, \quad \theta_{i+1}^k = (\mathbf{q}_\xi)_{i+1}^k = \frac{\mu^k}{\beta}.$$

Indeed, from (11), (21) and (25) we have

$$(\mathbf{p}_\theta)_{i+1}^k = (\mathbf{p}_\theta)_i^k - \tilde{\theta}_{i+1}^k + \theta_{i+1}^k = -n - 1 = \xi_{i+1}^k.$$

Furthermore, combining (16) and (17) we have

$$(\mathbf{p}_\theta)_i^k = \xi_i^k = -n - 1,$$

which implies that

$$\theta_{i+1}^k - \tilde{\theta}_{i+1}^k = (\mathbf{p}_\theta)_{i+1}^k - (\mathbf{p}_\theta)_i^k = 0. \tag{29}$$

Therefore, we conclude that

$$\begin{aligned}(\mathbf{q}_\xi)_{i+1}^k &\overset{(11)}{=} (\mathbf{q}_\xi)_i^k - \tilde{\xi}_{i+1}^k + \xi_{i+1}^k \overset{(13)}{=} (\mathbf{q}_\xi)_i^k - \theta_i^k - \xi_i^k + \tilde{\theta}_{i+1}^k + \xi_{i+1}^k \\ &\overset{(17),(25)}{=} (\mathbf{q}_\xi)_i^k - \theta_i^k + \tilde{\theta}_{i+1}^k \overset{(29)}{=} (\mathbf{q}_\xi)_i^k - \theta_i^k + \theta_{i+1}^k \overset{(16)}{=} \theta_{i+1}^k.\end{aligned}$$

This completes the proof. ■

Observe that Theorem 3.1 simplifies Algorithm 1 by eliminating the dual variables $\mathbf{p}$ and $\mathbf{q}$. They are replaced by $\mathbf{v}$ and $\mathbf{u}$, respectively. Moreover, note that $\mathbf{u}$ and $\mathbf{v}$ are separable in (10). As a result, we can update $\mathbf{u}$ by

$$\mathbf{u}_{i+1}^k = \operatorname*{argmin}_\mathbf{u} \left[\bar{B}(\mathbf{u}, \mu^k) + \frac{\beta}{2} \left\| \mathbf{u} - \tilde{\mathbf{u}}_{i+1}^k + \mathbf{v}_i^k \right\|^2 \right], \tag{30}$$

where

$$\bar{B}(\mathbf{u}, \mu) := \beta(n+1)\theta - \mu \log(\mathbf{x}) - \mu \log(\tau), \tag{31}$$

and update $\mathbf{v}$ by

$$\mathbf{v}_{i+1}^k = \mathbf{v}_i^k - \tilde{\mathbf{u}}_{i+1}^k + \mathbf{u}_{i+1}^k, \tag{32}$$

which follows from the update for $\mathbf{p}_{i+1}^k$. Note that $\tilde{\mathbf{v}}_i^k$ can now be eliminated from the algorithm. Problem (30) admits closed-form solutions given by

$$\mathbf{y}_{i+1}^k = \operatorname*{argmin}_\mathbf{y} \left[\frac{1}{2} \|\mathbf{y} - \tilde{\mathbf{y}}_{i+1}^k + \mathbf{r}_i^k\|^2 \right] = \tilde{\mathbf{y}}_{i+1}^k, \tag{33}$$

$$\mathbf{x}_{i+1}^k = \operatorname*{argmin}_{\mathbf{x}} \left[-\frac{\mu^k}{\beta} \log(\mathbf{x}) + \frac{1}{2} \|\mathbf{x} - \tilde{\mathbf{x}}_{i+1}^k + \mathbf{s}_i^k\|^2 \right]$$
$$= \frac{1}{2} \left[(\tilde{\mathbf{x}}_{i+1}^k - \mathbf{s}_i^k) + \sqrt{(\tilde{\mathbf{x}}_{i+1}^k - \mathbf{s}_i^k) \circ (\tilde{\mathbf{x}}_{i+1}^k - \mathbf{s}_i^k) + \frac{4\mu^k}{\beta}} \right], \tag{34}$$

$$\tau_{i+1}^k = \operatorname*{argmin}_{\tau} \left[-\frac{\mu^k}{\beta} \log(\tau) + \frac{1}{2} \|\tau - \tilde{\tau}_{i+1}^k + \kappa_i^k\|^2 \right]$$
$$= \frac{1}{2} \left[(\tilde{\tau}_{i+1}^k - \kappa_i^k) + \sqrt{(\tilde{\tau}_{i+1}^k - \kappa_i^k) \circ (\tilde{\tau}_{i+1}^k - \kappa_i^k) + \frac{4\mu^k}{\beta}} \right], \tag{35}$$

$$\theta_{i+1}^k = \tilde{\theta}_{i+1}^k, \tag{36}$$

where the last step is from (29). By eliminating $\mathbf{p}_i^k$ and $\mathbf{q}_i^k$, (9) reduces to

$$(\tilde{\mathbf{u}}_{i+1}^k, \tilde{\mathbf{v}}_{i+1}^k) = \prod_{Q\mathbf{u}=\mathbf{v}} (\mathbf{u}_i^k + \mathbf{v}_i^k, \mathbf{u}_i^k + \mathbf{v}_i^k).$$

It is easy to show (by the KKT condition) that the solution is given by

$$\tilde{\mathbf{u}}_{i+1}^k = (I + Q^\top Q)^{-1}(I - Q)(\mathbf{u}_i^k + \mathbf{v}_i^k) = (I + Q)^{-1}(\mathbf{u}_i^k + \mathbf{v}_i^k), \tag{37}$$

because matrix Q is skew-symmetric. Moreover, we only need to invert (or factorize) $I + Q$ once at the beginning of the algorithm. In this sense, ABIP is matrix inversion free.

Therefore, we have shown that (9), (10) and (11) can be simply implemented by means of (37), (30) and (32) respectively, and the solutions of (30) are given by (33), (34), (35) and (36).

We use the following criterion to terminate the inner loop of Algorithm 1:

$$\|Q\mathbf{u}_i^k - \mathbf{v}_i^k\|^2 \le \mu^k. \tag{38}$$

Finally, we present this specific implementation ABIP as Algorithm 2.

Remark 3.2: We denote $(\mathbf{u}_k^*, \mathbf{v}_k^*)$ as the optimal solution to (6) when $\mu = \mu^k$, which also satisfies the following optimality conditions of (6):

$$\begin{aligned} Q\mathbf{u} - \mathbf{v} &= 0, \\ \mathbf{x} \circ \mathbf{s} &= \frac{\mu^k}{\beta}\mathbf{e}, \\ \tau\kappa &= \frac{\mu^k}{\beta}, \\ \theta &= \frac{\mu^k}{\beta}, \\ \mathbf{r} &= 0, \\ \xi &= -n - 1, \\ (\mathbf{x}, \mathbf{s}, \tau, \kappa) &> 0. \end{aligned} \tag{40}$$

Algorithm 2 The detailed implementation of ABIP

Set $\mu^0 = \beta > 0$ and $\gamma \in (0, 1)$.
Set $\mathbf{r}_0^0 = \mathbf{y}_0^0 = 0$, $(\mathbf{x}_0^0, \tau_0^0, \mathbf{s}_0^0, \kappa_0^0) = (\mathbf{e}, 1, \mathbf{e}, 1) > 0$, $\theta_0^0 = 1$, and $\xi_0^0 = -n - 1$ with $\mathbf{x}_0^0 \circ \mathbf{s}_0^0 = \frac{\mu^0}{\beta}\mathbf{e}$, and $\tau_0^0\kappa_0^0 = \frac{\mu^0}{\beta}$.
for $k = 0, 1, 2, \ldots$ **do**
  for $i = 0, 1, 2, \ldots$ **do**
    if the **inner** termination criterion (38) is satisfied **then**
      break.
    end if
    Update $\tilde{\mathbf{u}}_{i+1}^k$ by using (37);
    Update $\mathbf{u}_{i+1}^k$ by using (33), (34), (35) and (36);
    Update $\mathbf{v}_{i+1}^k$ by using (32).
    if the **final** termination criterion is satisfied **then**
      return.
    end if
  end for
  Set $\mu^{k+1} = \gamma \cdot \mu^k$;
  Set $\mathbf{r}_0^{k+1} = 0$, $\xi_0^{k+1} = -n - 1$ and

$$(\mathbf{y}_0^{k+1}, \mathbf{x}_0^{k+1}, \mathbf{s}_0^{k+1}, \tau_0^{k+1}, \kappa_0^{k+1}, \theta_0^{k+1}) = \sqrt{\gamma} \cdot (\mathbf{y}_{i+1}^k, \mathbf{x}_{i+1}^k, \mathbf{s}_{i+1}^k, \tau_{i+1}^k, \kappa_{i+1}^k, \theta_{i+1}^k). \tag{39}$$

end for

Moreover, $(\mathbf{u}_k^*, \mathbf{v}_k^*)$ is uniquely defined. In fact, (40) defines a central path (cf. [5,32,40,42]) of the homogeneous self-dual embedded model [55].

From Theorem 3.1, we have

$$\begin{aligned} Q\tilde{\mathbf{u}}_i^k - \tilde{\mathbf{v}}_i^k &= 0, \\ \mathbf{x}_i^k \circ \mathbf{s}_i^k &= \frac{\mu^k}{\beta}\mathbf{e}, \\ \tau_i^k \kappa_i^k &= \frac{\mu^k}{\beta}, \\ \mathbf{r}_i^k &= 0, \\ \xi_i^k &= -n - 1, \\ \left(\mathbf{x}_i^k, \mathbf{s}_i^k, \tau_i^k, \kappa_i^k\right) &> 0, \end{aligned}$$

and

$$\theta_i^k = \tilde{\theta}_i^k = \frac{(\tilde{\mathbf{x}}_i^k)^\top \tilde{\mathbf{s}}_i^k + \tilde{\tau}_i^k \tilde{\kappa}_i^k + (\tilde{\mathbf{y}}_i^k)^\top \tilde{\mathbf{r}}_i^k}{-\tilde{\xi}_i^k}.$$

Together with the feasibility condition that $\|(\tilde{\mathbf{u}}_i^k, \tilde{\mathbf{v}}_i^k) - (\mathbf{u}_i^k, \mathbf{v}_i^k)\| \to 0$ when $i \to +\infty$ (see Lemma 3.4), we conclude that the optimal solution to problem (8) is on the central path.

This implies that ABIP is indeed a central path following algorithm, in view of the classical primal–dual central path following scheme.

Corollary 3.3: *Following a similar argument as in Theorem 3.1, it is easy to prove:*

$$(\mathbf{p}_k^*, \mathbf{q}_k^*) = (\mathbf{v}_k^*, \mathbf{u}_k^*), \tag{41}$$

where $(\mathbf{p}_k^*, \mathbf{q}_k^*)$ *denotes the optimal dual solution of (8).*

3.3. Iteration complexity analysis

In this section, we analyse the iteration complexity of ABIP. The following identity will be frequently used in our analysis:

$$\begin{aligned}&(a_1 - a_2)^\top (a_3 - a_4)\\ &\quad = \frac{1}{2}(\|a_4 + a_2\|^2 - \|a_4 + a_1\|^2 + \|a_3 + a_1\|^2 - \|a_3 + a_2\|^2), \forall a_1, a_2, a_3, a_4.\end{aligned} \tag{42}$$

To prove the main result, we need several technical lemmas.

Lemma 3.4: *Given* $k \geq 1$, *the sequence* $\{\|\mathbf{u}_i^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_k^*\|^2\}_{i\geq 0}$ *is monotonically decreasing and converges to 0.*

Proof: We observe from the optimality condition of problem (8) that

$$\beta(\mathbf{p}_k^*, \mathbf{q}_k^*) \in \partial \mathbb{1}(Q\mathbf{u} = \mathbf{v})[\mathbf{u}_k^*, \mathbf{v}_k^*], \quad -\beta(\mathbf{p}_k^*, \mathbf{q}_k^*) \in \partial B(\mathbf{u}_k^*, \mathbf{v}_k^*, \mu^k).$$

By using the convexity of $\mathbb{1}(Q\mathbf{u} = \mathbf{v})$ and (41) we have

$$0 \leq \beta(\mathbf{u}_k^* - \tilde{\mathbf{u}}_{i+1}^k, \mathbf{v}_k^* - \tilde{\mathbf{v}}_{i+1}^k)^\top (\mathbf{p}_k^*, \mathbf{q}_k^*) = \beta(\mathbf{u}_k^* - \tilde{\mathbf{u}}_{i+1}^k, \mathbf{v}_k^* - \tilde{\mathbf{v}}_{i+1}^k)^\top (\mathbf{v}_k^*, \mathbf{u}_k^*),$$

and using the convexity of $B(\mathbf{u}, \mathbf{v}, \mu)$ with respect to $(\mathbf{u}, \mathbf{v})$ and (41) we have

$$\begin{aligned}\beta(n+1)(\theta_k^* - \theta_{i+1}^k) &\leq -\beta(\mathbf{u}_k^* - \mathbf{u}_{i+1}^k, \mathbf{v}_k^* - \mathbf{v}_{i+1}^k)^\top (\mathbf{p}_k^*, \mathbf{q}_k^*)\\ &= -\beta(\mathbf{u}_k^* - \mathbf{u}_{i+1}^k, \mathbf{v}_k^* - \mathbf{v}_{i+1}^k)^\top (\mathbf{v}_k^*, \mathbf{u}_k^*),\end{aligned}$$

where we have used the fact $\beta(n+1)(\theta_{i+1}^k - \theta_k^*) = B(\mathbf{u}_{i+1}^k, \mathbf{v}_{i+1}^k, \mu^k) - B(\mathbf{u}_k^*, \mathbf{v}_k^*, \mu^k)$ that follows from the complementarity conditions $\mathbf{x}_i^k \circ \mathbf{s}_i^k = \frac{\mu^k}{\beta}\mathbf{e}$ and $\tau_i^k \kappa_i^k = \frac{\mu^k}{\beta}$. Summing up the above two inequalities leads to

$$\beta(n+1)(\theta_k^* - \theta_{i+1}^k) \leq \beta(\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k, \mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k)^\top (\mathbf{v}_k^*, \mathbf{u}_k^*). \tag{43}$$

Combining (11) and the optimality conditions of (9) and (10) yield

$$(\mathbf{p}_{i+1}^k, \mathbf{q}_{i+1}^k) - (\mathbf{u}_{i+1}^k, \mathbf{v}_{i+1}^k) + (\mathbf{u}_i^k, \mathbf{v}_i^k) \in \partial \mathbb{1}(Q\mathbf{u} = \mathbf{v})[\tilde{\mathbf{u}}_{i+1}^k, \tilde{\mathbf{v}}_{i+1}^k], \tag{44}$$

$$-\beta(\mathbf{p}_{i+1}^k, \mathbf{q}_{i+1}^k) \in \partial B(\mathbf{u}_{i+1}^k, \mathbf{v}_{i+1}^k, \mu^k). \tag{45}$$

From the convexity of $\mathbb{1}(Q\mathbf{u}=\mathbf{v})$, we have

$$\begin{aligned}
0 &\overset{(44)}{\leq} (\tilde{\mathbf{u}}_{i+1}^k-\mathbf{u}_k^*,\tilde{\mathbf{v}}_{i+1}^k-\mathbf{v}_k^*)^\top(\mathbf{p}_{i+1}^k-\mathbf{u}_{i+1}^k+\mathbf{u}_i^k,\mathbf{q}_{i+1}^k-\mathbf{v}_{i+1}^k+\mathbf{v}_i^k)\\
&\overset{(42)}{=} (\tilde{\mathbf{u}}_{i+1}^k-\mathbf{u}_k^*,\tilde{\mathbf{v}}_{i+1}^k-\mathbf{v}_k^*)^\top(\mathbf{p}_{i+1}^k,\mathbf{q}_{i+1}^k)+\frac{1}{2}(\|\mathbf{u}_i^k-\mathbf{u}_k^*\|^2+\|\mathbf{v}_i^k-\mathbf{v}_k^*\|^2-\|\mathbf{u}_i^k-\tilde{\mathbf{u}}_{i+1}^k\|^2\\
&\quad-\|\mathbf{v}_i^k-\tilde{\mathbf{v}}_{i+1}^k\|^2)+\frac{1}{2}(\|\mathbf{u}_{i+1}^k-\tilde{\mathbf{u}}_{i+1}^k\|^2+\|\mathbf{v}_{i+1}^k-\tilde{\mathbf{v}}_{i+1}^k\|^2-\|\mathbf{u}_{i+1}^k-\mathbf{u}_k^*\|^2\\
&\quad-\|\mathbf{v}_{i+1}^k-\mathbf{v}_k^*\|^2). \qquad (46)
\end{aligned}$$

From the convexity of $B(\mathbf{u},\mathbf{v},\mu^k)$ with respect to $(\mathbf{u},\mathbf{v})$, we have

$$\begin{aligned}
\beta(n+1)(\theta_{i+1}^k-\theta_k^*) &= B(\mathbf{u}_{i+1}^k,\mathbf{v}_{i+1}^k,\mu^k)-B(\mathbf{u}_k^*,\mathbf{v}_k^*,\mu^k)\\
&\overset{(45)}{\leq} -\beta(\mathbf{u}_{i+1}^k-\mathbf{u}_k^*,\mathbf{v}_{i+1}^k-\mathbf{v}_k^*)^\top(\mathbf{p}_{i+1}^k,\mathbf{q}_{i+1}^k), \qquad (47)
\end{aligned}$$

where the equality again follows from the complementarity conditions $\mathbf{x}_i^k\circ\mathbf{s}_i^k=\frac{\mu^k}{\beta}\mathbf{e}$ and $\tau_i^k\kappa_i^k=\frac{\mu^k}{\beta}$. Adding (46) and (47) and using (12) yields

$$\begin{aligned}
\beta(n+1)(\theta_{i+1}^k-\theta_k^*) &\leq \beta(\tilde{\mathbf{u}}_{i+1}^k-\mathbf{u}_{i+1}^k,\tilde{\mathbf{v}}_{i+1}^k-\mathbf{v}_{i+1}^k)^\top(\mathbf{v}_{i+1}^k,\mathbf{u}_{i+1}^k)\\
&\quad+\frac{\beta}{2}(\|\mathbf{u}_i^k-\mathbf{u}_k^*\|^2+\|\mathbf{v}_i^k-\mathbf{v}_k^*\|^2-\|\mathbf{u}_i^k-\tilde{\mathbf{u}}_{i+1}^k\|^2-\|\mathbf{v}_i^k-\tilde{\mathbf{v}}_{i+1}^k\|^2)\\
&\quad+\frac{\beta}{2}(\|\mathbf{u}_{i+1}^k-\tilde{\mathbf{u}}_{i+1}^k\|^2+\|\mathbf{v}_{i+1}^k-\tilde{\mathbf{v}}_{i+1}^k\|^2-\|\mathbf{u}_{i+1}^k-\mathbf{u}_k^*\|^2-\|\mathbf{v}_{i+1}^k-\mathbf{v}_k^*\|^2). \qquad (48)
\end{aligned}$$

Further adding (43) and (48), we have

$$\begin{aligned}
0 &\leq (\tilde{\mathbf{u}}_{i+1}^k-\mathbf{u}_{i+1}^k,\tilde{\mathbf{v}}_{i+1}^k-\mathbf{v}_{i+1}^k)^\top(\mathbf{v}_{i+1}^k-\mathbf{v}_k^*,\mathbf{u}_{i+1}^k-\mathbf{u}_k^*)\\
&\quad+\frac{1}{2}(\|\mathbf{u}_i^k-\mathbf{u}_k^*\|^2+\|\mathbf{v}_i^k-\mathbf{v}_k^*\|^2-\|\mathbf{u}_i^k-\tilde{\mathbf{u}}_{i+1}^k\|^2-\|\mathbf{v}_i^k-\tilde{\mathbf{v}}_{i+1}^k\|^2)\\
&\quad+\frac{1}{2}(\|\mathbf{u}_{i+1}^k-\tilde{\mathbf{u}}_{i+1}^k\|^2+\|\mathbf{v}_{i+1}^k-\tilde{\mathbf{v}}_{i+1}^k\|^2-\|\mathbf{u}_{i+1}^k-\mathbf{u}_k^*\|^2-\|\mathbf{v}_{i+1}^k-\mathbf{v}_k^*\|^2)\\
&\overset{(11),(12)}{=} (\mathbf{v}_i^k-\mathbf{v}_{i+1}^k,\mathbf{u}_i^k-\mathbf{u}_{i+1}^k)^\top(\mathbf{v}_{i+1}^k-\mathbf{v}_k^*,\mathbf{u}_{i+1}^k-\mathbf{u}_k^*)\\
&\quad+\frac{1}{2}(\|\mathbf{u}_i^k-\mathbf{u}_k^*\|^2+\|\mathbf{v}_i^k-\mathbf{v}_k^*\|^2-\|\mathbf{u}_i^k-\tilde{\mathbf{u}}_{i+1}^k\|^2-\|\mathbf{v}_i^k-\tilde{\mathbf{v}}_{i+1}^k\|^2)\\
&\quad+\frac{1}{2}(\|\mathbf{u}_{i+1}^k-\tilde{\mathbf{u}}_{i+1}^k\|^2+\|\mathbf{v}_{i+1}^k-\tilde{\mathbf{v}}_{i+1}^k\|^2-\|\mathbf{u}_{i+1}^k-\mathbf{u}_k^*\|^2-\|\mathbf{v}_{i+1}^k-\mathbf{v}_k^*\|^2)\\
&\overset{(42)}{=} \frac{1}{2}(\|\mathbf{u}_i^k-\mathbf{u}_k^*\|^2+\|\mathbf{v}_i^k-\mathbf{v}_k^*\|^2-\|\mathbf{u}_{i+1}^k-\mathbf{u}_k^*\|^2-\|\mathbf{v}_{i+1}^k-\mathbf{v}_k^*\|^2)-\frac{1}{2}(\|\mathbf{u}_i^k-\mathbf{u}_{i+1}^k\|^2\\
&\quad+\|\mathbf{v}_i^k-\mathbf{v}_{i+1}^k\|^2)+\frac{1}{2}(\|\mathbf{u}_i^k-\mathbf{u}_k^*\|^2+\|\mathbf{v}_i^k-\mathbf{v}_k^*\|^2-\|\mathbf{u}_i^k-\tilde{\mathbf{u}}_{i+1}^k\|^2-\|\mathbf{v}_i^k-\tilde{\mathbf{v}}_{i+1}^k\|^2)\\
&\quad+\frac{1}{2}(\|\mathbf{u}_{i+1}^k-\tilde{\mathbf{u}}_{i+1}^k\|^2+\|\mathbf{v}_{i+1}^k-\tilde{\mathbf{v}}_{i+1}^k\|^2-\|\mathbf{u}_{i+1}^k-\mathbf{u}_k^*\|^2-\|\mathbf{v}_{i+1}^k-\mathbf{v}_k^*\|^2). \qquad (49)
\end{aligned}$$

Recall that (11) and (12) imply

$$\begin{aligned}\|\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k\|^2 &= \|\mathbf{p}_i^k - \mathbf{p}_{i+1}^k\|^2 + \|\mathbf{q}_i^k - \mathbf{q}_{i+1}^k\|^2 \\ &= \|\mathbf{u}_i^k - \mathbf{u}_{i+1}^k\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_{i+1}^k\|^2. \end{aligned} \tag{50}$$

Now, we use (50) and (49) to obtain

$$\begin{aligned}\frac{1}{2}(\|\mathbf{u}_i^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_i^k - \tilde{\mathbf{v}}_{i+1}^k\|^2) \le \|\mathbf{u}_i^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_k^*\|^2 - \|\mathbf{u}_{i+1}^k \\ - \mathbf{u}_k^*\|^2 - \|\mathbf{v}_{i+1}^k - \mathbf{v}_k^*\|^2. \end{aligned} \tag{51}$$

Therefore, we conclude that $\{\|\mathbf{u}_i^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_k^*\|^2\}_{i\ge 0}$ is a monotonically decreasing sequence. Note that $\{(\mathbf{u}_i^k, \mathbf{v}_i^k)\}_{i\ge 0}$ is a bounded sequence, there must exist a subsequence $\{(\mathbf{u}_{i_j}^k, \mathbf{v}_{i_j}^k)\}_{j\ge 0}$ that converges to a limit point $(\bar{\mathbf{u}}, \bar{\mathbf{v}})$. Since

$$-\beta(\mathbf{v}_{i_j}^k, \mathbf{u}_{i_j}^k) = -\beta(\mathbf{p}_{i_j}^k, \mathbf{q}_{i_j}^k) \in \partial B(\mathbf{u}_{i_j}^k, \mathbf{v}_{i_j}^k, \mu^k),$$

by letting $j \to +\infty$ we have

$$\begin{aligned}\bar{\mathbf{x}} \circ \bar{\mathbf{s}} &= (\mu^k/\beta) \cdot \mathbf{e}, \\ \bar{\tau}\bar{\kappa} &= \mu^k/\beta, \\ \bar{\mathbf{r}} &= 0, \\ \bar{\xi} &= -n - 1, \\ (\bar{\mathbf{x}}, \bar{\mathbf{s}}, \bar{\tau}, \bar{\kappa}) &> 0. \end{aligned}$$

Furthermore, from (51) we have

$$\|\mathbf{u}_{i_j}^k - \tilde{\mathbf{u}}_{i_j+1}^k\|^2 + \|\mathbf{v}_{i_j}^k - \tilde{\mathbf{v}}_{i_j+1}^k\|^2 \to 0,$$

which implies that $(\tilde{\mathbf{u}}_{i_j+1}^k, \tilde{\mathbf{v}}_{i_j+1}^k)$ converges to $(\bar{\mathbf{u}}, \bar{\mathbf{v}})$. Therefore, we have $Q\bar{\mathbf{u}} - \bar{\mathbf{v}} = 0$ and

$$\bar{\theta} = \frac{(\bar{\mathbf{x}})^\top \bar{\mathbf{s}} + \bar{\tau}\bar{\kappa} + (\bar{\mathbf{y}})^\top \bar{\mathbf{r}}}{-\bar{\xi}} = \frac{\mu^k}{\beta}.$$

Due to the uniqueness of the central path solution, we have $(\bar{\mathbf{u}}, \bar{\mathbf{v}}) = (\mathbf{u}_k^*, \mathbf{v}_k^*)$, which implies that

$$\|\mathbf{u}_{i_j}^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_{i_j}^k - \mathbf{v}_k^*\|^2 \longrightarrow 0, \quad \text{as } j \to +\infty.$$

Therefore, we conclude that

$$\|\mathbf{u}_i^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_k^*\|^2 \longrightarrow 0, \quad \text{as } i \to +\infty.$$

This completes the proof. ■

Lemma 3.5: *The sequence* $\{\|\mathbf{u}_0^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_0^k - \mathbf{v}_k^*\|^2\}_{k\geq 0}$ *is uniformly bounded, i.e. there exists a constant* $C > 0$ *that does not depend on* k *or* μ^k *such that*

$$\|\mathbf{u}_0^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_0^k - \mathbf{v}_k^*\|^2 \leq C. \tag{52}$$

Moreover, the iterates $\{(\mathbf{u}_i^k, \mathbf{v}_i^k)\}$, *for* $k \geq 1, i = 0, 1, \ldots, N_k$, *are uniformly bounded, i.e. there exists a constant* $D > 0$ *that does not depend on* k *or* μ^k *such that*

$$\|\mathbf{u}_i^k\|^2 + \|\mathbf{v}_i^k\|^2 \leq D, \quad \forall i = 0, 1, \ldots, N_k, \tag{53}$$

where N_k *denotes the number of inner iterations in the kth outer loop.*

Proof: We recall an important fact (see, e.g.[55]) that the set of the central path points $\{(\mathbf{u}_k^*, \mathbf{v}_k^*)\}$ with μ^k/β, i.e. the solution of (40), is uniformly bounded, where $0 < \mu^k \leq \mu^0$. That is, there exists a constant C_1 that does not depend on k or μ^k such that

$$\|\mathbf{u}_k^*\|^2 + \|\mathbf{v}_k^*\|^2 \leq C_1, \quad \forall k \geq 1. \tag{54}$$

This also implies that the following inequality holds for the limiting point $(\mathbf{u}^*, \mathbf{v}^*)$ of the central path points $\{(\mathbf{u}_k^*, \mathbf{v}_k^*)\}$, namely $\|\mathbf{u}^*\|^2 + \|\mathbf{v}^*\|^2 \leq C_1$. Using Lemma 3.4 and (38) with $\mu^k > 0$, we know that $N_k < +\infty$ is well-defined for any fixed $k \geq 0$. Now we claim that

$$\|\mathbf{u}_{N_k}^k - \mathbf{u}^*\|^2 + \|\mathbf{v}_{N_k}^k - \mathbf{v}^*\|^2 \to 0, \quad \text{as } k \to +\infty. \tag{55}$$

Indeed, if the claim does not hold, then there exists $\delta > 0$, and a subsequence $\{k_\ell \mid \ell = 1, 2, \ldots\}$ with $k_\ell \uparrow \infty$ as $\ell \to \infty$ and $N' > 0$ such that

$$\|\mathbf{u}_{N_{k_\ell}}^{k_\ell} - \mathbf{u}^*\|^2 + \|\mathbf{v}_{N_{k_\ell}}^{k_\ell} - \mathbf{v}^*\|^2 \geq \delta, \quad \text{for all } \ell > N'.$$

Using the property of central path, we have $\|\mathbf{u}_k^* - \mathbf{u}^*\|^2 + \|\mathbf{v}_k^* - \mathbf{v}^*\|^2 \to 0$ as $k \to +\infty$. Thus there exists $N'' > 0$ such that

$$\|\mathbf{u}_{N_{k_\ell}}^{k_\ell} - \mathbf{u}_{k_\ell}^*\|^2 + \|\mathbf{v}_{N_{k_\ell}}^{k_\ell} - \mathbf{v}_{k_\ell}^*\|^2 \geq \delta/2, \quad \text{for all } \ell > N''.$$

This contradicts with (38). Thus (55) holds and there exists a constant $D_1 > 0$ that does not depend on k or μ^k such that

$$\|\mathbf{u}_{N_k}^k\|^2 + \|\mathbf{v}_{N_k}^k\|^2 \leq D_1, \quad \text{for all } k \geq 0. \tag{56}$$

Now we prove (52). For $k = 0$, since we choose $\mu^0 = \beta$, the initial point we choose in Algorithm 2 satisfies (40) automatically, i.e. $(\mathbf{u}_0^*, \mathbf{v}_0^*) = (\mathbf{u}_0^0, \mathbf{v}_0^0)$, and (38), i.e. $\|Q\mathbf{u}_0^0 - \mathbf{v}_0^0\|^2 \leq \mu^0$, and $N_0 = 0$. Thus we have

$$\|\mathbf{u}_0^0 - \mathbf{u}_0^*\|^2 + \|\mathbf{v}_0^0 - \mathbf{v}_0^*\|^2 = 0.$$

For $k \geq 0$, by the definition of $\mathbf{u}$ and $\mathbf{v}$, we have

$$\|\mathbf{u}_0^{k+1} - \mathbf{u}_{k+1}^*\|^2 + \|\mathbf{v}_0^{k+1} - \mathbf{v}_{k+1}^*\|^2$$

$$= \|\mathbf{y}_0^{k+1} - \mathbf{y}_{k+1}^*\|^2 + \|\mathbf{x}_0^{k+1} - \mathbf{x}_{k+1}^*\|^2 + (\tau_0^{k+1} - \tau_{k+1}^*)^2 + (\theta_0^{k+1} - \theta_{k+1}^*)^2 + \|\mathbf{s}_0^{k+1} - \mathbf{s}_{k+1}^*\|^2 + (\kappa_0^{k+1} - \kappa_{k+1}^*)^2.$$

Recall that (cf. (39)) the following equation holds true,

$$(\mathbf{y}_0^{k+1}, \mathbf{x}_0^{k+1}, \mathbf{s}_0^{k+1}, \tau_0^{k+1}, \kappa_0^{k+1}, \theta_0^{k+1}) = \sqrt{\gamma}(\mathbf{y}_{N_k}^k, \mathbf{x}_{N_k}^k, \mathbf{s}_{N_k}^k, \tau_{N_k}^k, \kappa_{N_k}^k, \theta_{N_k}^k).$$

This implies that

$$\|\mathbf{y}_0^{k+1}\|^2 + \|\mathbf{x}_0^{k+1}\|^2 + (\tau_0^{k+1})^2 + (\theta_0^{k+1})^2 + \|\mathbf{s}_0^{k+1}\|^2 + (\kappa_0^{k+1})^2$$
$$= \gamma(\|\mathbf{y}_{N_k}^k\|^2 + \|\mathbf{x}_{N_k}^k\|^2 + (\tau_{N_k}^k)^2 + (\theta_{N_k}^k)^2 + \|\mathbf{s}_{N_k}^k\|^2 + (\kappa_{N_k}^k)^2) \overset{(56)}{\leq} \gamma D_1.$$

Putting these pieces together yields that

$$\|\mathbf{u}_0^{k+1} - \mathbf{u}_{k+1}^*\|^2 + \|\mathbf{v}_0^{k+1} - \mathbf{v}_{k+1}^*\|^2 \leq 2\gamma D_1 + 2C_1.$$

Therefore, letting $C = 2\gamma D_1 + 2C_1$ proves (52).

Now we prove (53). Note that (52) leads to (53) immediately. To see this, note that combining (52) and Lemma 3.4 yields

$$\|\mathbf{u}_i^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_k^*\|^2 \leq C, \quad \forall i = 0, 1, \ldots, N_k$$

which together with (54) implies

$$\|\mathbf{u}_i^k\|^2 + \|\mathbf{v}_i^k\|^2 \leq 2C + 2C_1, \quad \forall i = 0, 1, \ldots, N_k$$

proving (53) with $D = 2C + 2C_1$. ■

Lemma 3.6: *The number of iterations (denoted by N_k) needed in the inner loop of Algorithm 2 is*

$$N_k \leq \log\left(\frac{2C(1 + \|Q\|^2)}{\mu^k}\right)\left[\log\left(1 + \min\left\{\frac{1}{C_3}, \frac{\mu^k}{4DC_3\beta}\right\}\right)\right]^{-1}, \tag{57}$$

where C_3 is defined as

$$C_3 = \left[1 + \frac{12\lambda_{\max}(A^\top A)}{\lambda_{\min}^2(AA^\top)} \cdot \max\{1, \|\mathbf{c}\|^2, \|\bar{\mathbf{c}}\|^2\}\right] \cdot \left[1 + \frac{6}{\|\bar{\mathbf{b}}\|^2} \cdot \max\{\|\mathbf{b}\|^2, \|A\|^2\}\right] > 1. \tag{58}$$

Proof: It follows from Lemma 3.5 that $B(\mathbf{u}_i^k, \mathbf{v}_i^k, \mu^k)$ is strongly convex with respect to $(\mathbf{x}, \mathbf{s}, \tau, \kappa)$. More specifically, we have

$$\nabla_{\mathbf{x}}^2 B(\mathbf{u}_i^k, \mathbf{v}_i^k, \mu^k) = \mathrm{Diag}\left(\frac{\mu^k}{\mathbf{x_i^k} . * \mathbf{x_i^k}}\right) \succeq \frac{\mu^k}{D} \cdot \mathbb{I},$$

$$\nabla_{\mathbf{s}}^2 B(\mathbf{u}_i^k, \mathbf{v}_i^k, \mu^k) = \mathrm{Diag}\left(\frac{\mu^k}{\mathbf{s_i^k} . * \mathbf{s_i^k}}\right) \succeq \frac{\mu^k}{D} \cdot \mathbb{I},$$

$$\nabla_\tau^2 B(\mathbf{u}_i^k, \mathbf{v}_i^k, \mu^k) = \frac{\mu^k}{(\tau_i^k)^2} \succeq \frac{\mu^k}{D},$$
$$\nabla_\kappa^2 B(\mathbf{u}_i^k, \mathbf{v}_i^k, \mu^k) = \frac{\mu^k}{(\kappa_i^k)^2} \succeq \frac{\mu^k}{D}.$$

Therefore, (48) is changed to

$$\begin{aligned}
&\beta(n+1)(\theta_{i+1}^k - \theta_k^*) + \frac{\mu^k}{2D}(\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + \|\mathbf{s}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2 + (\kappa_{i+1}^k - \kappa_k^*)^2) \\
&\quad \le \beta(\tilde{\mathbf{u}}_{i+1}^k - \mathbf{u}_{i+1}^k, \tilde{\mathbf{v}}_{i+1}^k - \mathbf{v}_{i+1}^k)^\top(\mathbf{v}_{i+1}^k, \mathbf{u}_{i+1}^k) + \frac{\beta}{2}(\|\mathbf{u}_i^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_k^*\|^2 \\
&\qquad - \|\mathbf{u}_i^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 - \|\mathbf{v}_i^k - \tilde{\mathbf{v}}_{i+1}^k\|^2) \\
&\qquad + \frac{\beta}{2}(\|\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k\|^2 - \|\mathbf{u}_{i+1}^k - \mathbf{u}_k^*\|^2 - \|\mathbf{v}_{i+1}^k - \mathbf{v}_k^*\|^2). \qquad (59)
\end{aligned}$$

By summing (43) and (59), using (50) and observing

$$\begin{aligned}
&(\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k, \mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k)^\top(\mathbf{v}_k^*, \mathbf{u}_k^*) + (\tilde{\mathbf{u}}_{i+1}^k - \mathbf{u}_{i+1}^k, \tilde{\mathbf{v}}_{i+1}^k - \mathbf{v}_{i+1}^k)^\top(\mathbf{v}_{i+1}^k, \mathbf{u}_{i+1}^k) \\
&\quad = (\tilde{\mathbf{u}}_{i+1}^k - \mathbf{u}_{i+1}^k, \tilde{\mathbf{v}}_{i+1}^k - \mathbf{v}_{i+1}^k)^\top(\mathbf{v}_{i+1}^k - \mathbf{v}_k^*, \mathbf{u}_{i+1}^k - \mathbf{u}_k^*) \\
&\quad = (\mathbf{p}_i^k - \mathbf{p}_{i+1}^k, \mathbf{q}_i^k - \mathbf{q}_{i+1}^k)^\top(\mathbf{v}_{i+1}^k - \mathbf{v}_k^*, \mathbf{u}_{i+1}^k - \mathbf{u}_k^*) \\
&\quad = (\mathbf{v}_i^k - \mathbf{v}_{i+1}^k, \mathbf{u}_i^k - \mathbf{u}_{i+1}^k)^\top(\mathbf{v}_{i+1}^k - \mathbf{v}_k^*, \mathbf{u}_{i+1}^k - \mathbf{u}_k^*) \\
&\quad = \tfrac{1}{2}(\|\mathbf{u}_i^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_k^*\|^2 - \|\mathbf{u}_{i+1}^k - \mathbf{u}_k^*\|^2 - \|\mathbf{v}_{i+1}^k - \mathbf{v}_k^*\|^2) \\
&\qquad - \tfrac{1}{2}(\|\mathbf{u}_i^k - \mathbf{u}_{i+1}^k\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_{i+1}^k\|^2),
\end{aligned}$$

we obtain

$$\begin{aligned}
&\frac{\mu^k}{2D\beta}(\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + \|\mathbf{s}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2 + (\kappa_{i+1}^k - \kappa_k^*)^2) \\
&\qquad + \frac{1}{2}(\|\mathbf{u}_i^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_i^k - \tilde{\mathbf{v}}_{i+1}^k\|^2) \\
&\qquad\quad \le \|\mathbf{u}_i^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_k^*\|^2 - \|\mathbf{u}_{i+1}^k - \mathbf{u}_k^*\|^2 - \|\mathbf{v}_{i+1}^k - \mathbf{v}_k^*\|^2. \qquad (60)
\end{aligned}$$

Moreover, from (45) we have

$$0 \le B(\mathbf{u}, \mathbf{v}, \mu^k) - B(\mathbf{u}_{i+1}^k, \mathbf{v}_{i+1}^k, \mu^k) + \beta(\mathbf{u} - \mathbf{u}_{i+1}^k, \mathbf{v} - \mathbf{v}_{i+1}^k)^\top(\mathbf{p}_{i+1}^k, \mathbf{q}_{i+1}^k) \qquad (61)$$

and

$$0 \le B(\mathbf{u}, \mathbf{v}, \mu^k) - B(\mathbf{u}_i^k, \mathbf{v}_i^k, \mu^k) + \beta(\mathbf{u} - \mathbf{u}_i^k, \mathbf{v} - \mathbf{v}_i^k)^\top(\mathbf{p}_i^k, \mathbf{q}_i^k). \qquad (62)$$

Letting $(\mathbf{u}, \mathbf{v}) = (\mathbf{u}_i^k, \mathbf{v}_i^k)$ in (61) and $(\mathbf{u}, \mathbf{v}) = (\mathbf{u}_{i+1}^k, \mathbf{v}_{i+1}^k)$ in (62), and adding them up lead to

$$0 \le -(\mathbf{u}_i^k - \mathbf{u}_{i+1}^k, \mathbf{v}_i^k - \mathbf{v}_{i+1}^k)^\top(\mathbf{p}_i^k - \mathbf{p}_{i+1}^k, \mathbf{q}_i^k - \mathbf{q}_{i+1}^k)$$

$$
\begin{aligned}
&= (\mathbf{u}_i^k - \mathbf{u}_{i+1}^k, \mathbf{v}_i^k - \mathbf{v}_{i+1}^k)^\top (\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k, \mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k) \\
&= \frac{1}{2}(\|\mathbf{u}_i^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 - \|\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 - \|\mathbf{u}_i^k - \mathbf{u}_{i+1}^k\|^2) \\
&\quad + \frac{1}{2}(\|\mathbf{v}_i^k - \tilde{\mathbf{v}}_{i+1}^k\|^2 - \|\mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k\|^2 - \|\mathbf{v}_i^k - \mathbf{v}_{i+1}^k\|^2),
\end{aligned}
$$

which implies that

$$
\|\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k\|^2 \le \|\mathbf{u}_i^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_i^k - \tilde{\mathbf{v}}_{i+1}^k\|^2. \tag{63}
$$

Combining (60) and (63) yields

$$
\begin{aligned}
&\frac{1}{2}(\|\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k\|^2) \\
&\quad + \frac{\mu^k}{2D\beta}(\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + \|\mathbf{s}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2 + (\kappa_{i+1}^k - \kappa_k^*)^2) \\
&\quad \le \|\mathbf{u}_i^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_k^*\|^2 - \|\mathbf{u}_{i+1}^k - \mathbf{u}_k^*\|^2 - \|\mathbf{v}_{i+1}^k - \mathbf{v}_k^*\|^2.
\end{aligned} \tag{64}
$$

Furthermore, we have (by denoting $C_4 := 6\lambda_{\max}(A^\top A)\max\{1, \|\mathbf{c}\|^2, \|\bar{\mathbf{c}}\|^2\}/\lambda_{\min}^2(AA^\top)$)

$$
\begin{aligned}
\|\mathbf{y}_{i+1}^k - \mathbf{y}_k^*\|^2 &\le 2(\|\mathbf{y}_{i+1}^k - \tilde{\mathbf{y}}_{i+1}^k\|^2 + \|\tilde{\mathbf{y}}_{i+1}^k - \mathbf{y}_k^*\|^2) \\
&= 2\|\mathbf{y}_{i+1}^k - \tilde{\mathbf{y}}_{i+1}^k\|^2 + 2\|(AA^\top)^{-1}A(A^\top \tilde{\mathbf{y}}_{i+1}^k - A^\top \mathbf{y}_k^*)\|^2 \\
&= 2\|\mathbf{y}_{i+1}^k - \tilde{\mathbf{y}}_{i+1}^k\|^2 + 2\|(AA^\top)^{-1}A(\mathbf{c}\tilde{\tau}_{i+1}^k - \tilde{\mathbf{s}}_{i+1}^k \\
&\quad - \bar{\mathbf{c}}\tilde{\theta}_{i+1}^k - \mathbf{c}\tau_k^* + \mathbf{s}_k^* + \bar{\mathbf{c}}\theta_k^*)\|^2 \\
&\le C_4(\|\tilde{\mathbf{s}}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tilde{\tau}_{i+1}^k - \tau_k^*)^2 + (\tilde{\theta}_{i+1}^k - \theta_k^*)^2) + 2\|\mathbf{y}_{i+1}^k - \tilde{\mathbf{y}}_{i+1}^k\|^2 \\
&\le 2C_4(\|\mathbf{s}_{i+1}^k - \tilde{\mathbf{s}}_{i+1}^k\|^2 + (\tau_{i+1}^k - \tilde{\tau}_{i+1}^k)^2 + \|\mathbf{s}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2) \\
&\quad + C_4(\theta_{i+1}^k - \theta_k^*)^2 + 2\|\mathbf{y}_{i+1}^k - \tilde{\mathbf{y}}_{i+1}^k\|^2,
\end{aligned} \tag{65}
$$

and (by denoting $C_5 := 3\max\{\|\mathbf{b}\|^2, \|A\|^2\}/\|\bar{\mathbf{b}}\|^2$)

$$
\begin{aligned}
(\theta_{i+1}^k - \theta_k^*)^2 = (\tilde{\theta}_{i+1}^k - \theta_k^*)^2 &= \frac{1}{\|\bar{\mathbf{b}}\|^2}\|\mathbf{b}\tilde{\tau}_{i+1}^k - A\tilde{\mathbf{x}}_{i+1}^k + \tilde{\mathbf{r}}_{i+1}^k - \mathbf{b}\tau_k^* + A\mathbf{x}_k^* - \mathbf{r}_k^*\|^2 \\
&\le C_5(\|\tilde{\mathbf{x}}_{i+1}^k - \mathbf{x}_k^*\|^2 + (\tilde{\tau}_{i+1}^k - \tau_k^*)^2 + \|\tilde{\mathbf{r}}_{i+1}^k - \mathbf{r}_k^*\|^2) \\
&= C_5(\|\tilde{\mathbf{x}}_{i+1}^k - \mathbf{x}_k^*\|^2 + (\tilde{\tau}_{i+1}^k - \tau_k^*)^2 + \|\tilde{\mathbf{r}}_{i+1}^k - \mathbf{r}_{i+1}^k\|^2) \\
&\le 2C_5(\|\mathbf{x}_{i+1}^k - \tilde{\mathbf{x}}_{i+1}^k\|^2 + (\tau_{i+1}^k - \tilde{\tau}_{i+1}^k)^2 + \|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 \\
&\quad + (\tau_{i+1}^k - \tau_k^*)^2) + C_5\|\tilde{\mathbf{r}}_{i+1}^k - \mathbf{r}_{i+1}^k\|^2.
\end{aligned} \tag{66}
$$

By summing up (65) and (66), we have

$$
\begin{aligned}
&\|\mathbf{y}_{i+1}^k - \mathbf{y}_k^*\|^2 + (\theta_{i+1}^k - \theta_k^*)^2 \\
&\quad \le 2(C_4 + C_5)(\|\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k\|^2)
\end{aligned}
$$

$$
\begin{aligned}
&+ 2C_4(\|\mathbf{s}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2 + (\theta_{i+1}^k - \theta_k^*)^2) \\
&+ 2C_5(\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2) \\
&\overset{(66)}{\leq} 2(C_4 + C_5)(\|\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k\|^2) \\
&+ 2C_4(\|\mathbf{s}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2) \\
&\quad + 2C_5(\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2) + 4C_4C_5(\|\mathbf{x}_{i+1}^k - \tilde{\mathbf{x}}_{i+1}^k\|^2 + (\tau_{i+1}^k - \tilde{\tau}_{i+1}^k)^2) \\
&\quad + 4C_4C_5(\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2) + 2C_4C_5\|\tilde{\mathbf{r}}_{i+1}^k - \mathbf{r}_{i+1}^k\|^2 \\
&\overset{(58)}{\leq} C_3(\|\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k\|^2) + C_3(\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + \|\mathbf{s}_{i+1}^k \\
&- \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2). \qquad (67)
\end{aligned}
$$

Noting that $\|\mathbf{r}_{i+1}^k - \mathbf{r}_k^*\|^2 = (\xi_{i+1}^k - \xi_k^*)^2 = 0$, we have

$$
\begin{aligned}
&\min\left\{\frac{1}{2C_3}, \frac{\mu^k}{4DC_3\beta}\right\}(\|\mathbf{u}_{i+1}^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_{i+1}^k - \mathbf{v}_k^*\|^2) \\
&\quad \leq \frac{\mu^k}{4DC_3\beta}(\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + \|\mathbf{s}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2 + (\kappa_{i+1}^k - \kappa_k^*)^2) \\
&\qquad + \min\left\{\frac{1}{2C_3}, \frac{\mu^k}{4DC_3\beta}\right\}(\|\mathbf{y}_{i+1}^k - \mathbf{y}_k^*\|^2 + (\theta_{i+1}^k - \theta_k^*)^2) \\
&\quad \leq \frac{\mu^k}{4D\beta}(\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + \|\mathbf{s}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2 + (\kappa_{i+1}^k - \kappa_k^*)^2) \\
&\qquad + \frac{1}{2}(\|\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k\|^2) + \frac{\mu^k}{4D\beta} \\
&\quad \cdot (\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + \|\mathbf{s}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2) \\
&\quad \leq \frac{1}{2}(\|\mathbf{u}_{i+1}^k - \tilde{\mathbf{u}}_{i+1}^k\|^2 + \|\mathbf{v}_{i+1}^k - \tilde{\mathbf{v}}_{i+1}^k\|^2) \\
&\qquad + \frac{\mu^k}{2D\beta}(\|\mathbf{x}_{i+1}^k - \mathbf{x}_k^*\|^2 + \|\mathbf{s}_{i+1}^k - \mathbf{s}_k^*\|^2 + (\tau_{i+1}^k - \tau_k^*)^2 + (\kappa_{i+1}^k - \kappa_k^*)^2) \\
&\quad \overset{(64)}{\leq} \|\mathbf{u}_i^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_i^k - \mathbf{v}_k^*\|^2 - \|\mathbf{u}_{i+1}^k - \mathbf{u}_k^*\|^2 - \|\mathbf{v}_{i+1}^k - \mathbf{v}_k^*\|^2,
\end{aligned}
$$

where the second inequality is due to $C_3 > 1$ and (67). Therefore, we obtain that

$$
\begin{aligned}
&\|\mathbf{u}_{N_k}^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_{N_k}^k - \mathbf{v}_k^*\|^2 \\
&\quad \leq \left(1 + \min\left\{\frac{1}{2C_3}, \frac{\mu^k}{4DC_3\beta}\right\}\right)^{-N_k}(\|\mathbf{u}_0^k - \mathbf{u}_k^*\|^2 + \|\mathbf{v}_0^k - \mathbf{v}_k^*\|^2) \\
&\quad \leq C\left(1 + \min\left\{\frac{1}{2C_3}, \frac{\mu^k}{4DC_3\beta}\right\}\right)^{-N_k}.
\end{aligned}
$$

On the other hand, we have

$$\|Q\mathbf{u}_{N_k}^k - \mathbf{v}_{N_k}^k\|^2 = \|Q(\mathbf{u}_{N_k}^k - \mathbf{u}_k^*) - (\mathbf{v}_{N_k}^k - \mathbf{v}_k^*)\|^2 \le 2\|Q\|^2\|\mathbf{u}_{N_k}^k - \mathbf{u}_k^*\|^2 + 2\|\mathbf{v}_{N_k}^k - \mathbf{v}_k^*\|^2.$$

Therefore, the number of iterations (denoted by N_k) needed in the inner loop of Algorithm 2 should satisfy that

$$2C(1+\|Q\|^2)\left(1+\min\left\{\frac{1}{2C_3}, \frac{\mu^k}{4DC_3\beta}\right\}\right)^{-N_k} \le \mu^k,$$

which proves (57). ■

Now we are ready to present the main result of the iteration complexity of Algorithm 2.

Theorem 3.7: *Suppose that the* ABIP *is terminated when* $\mu^k < \varepsilon$, *where* $\varepsilon > 0$ *is a pre-given tolerance. The total* IPM *and* ADMM *iteration complexities of* ABIP *are respectively*

$$T_{\text{IPM}} = O\left(\log\left(\frac{1}{\varepsilon}\right)\right), \quad T_{\text{ADMM}} = O\left(\frac{\kappa_A^2\|Q\|^2}{\varepsilon}\log\left(\frac{1}{\varepsilon}\right)\right),$$

where $\kappa_A := \lambda_{\max}(A^\top A)/\lambda_{\min}(AA^\top)$.

Proof: Note that ABIP consists of two types of loops: inner loops and outer loops. In the outer loop, a log-barrier penalty problem is formed with a penalty parameter μ^k, and in the inner loop, this log-barrier penalty problem is solved by a two-block ADMM. The outer loop is terminated when $\mu^k < \varepsilon$, with a pre-given tolerance $\varepsilon > 0$. It is then easy to see that the number of outer loops, i.e. the number of interior point iterations, is

$$T_{\text{IPM}} = \left\lceil \frac{\log(\mu^0/\varepsilon)}{\log(1/\gamma)} \right\rceil.$$

For the *total* number of ADMM steps (rather than the ADMM steps between two IPM outer loops), we have the following estimate:

$$\begin{aligned} T_{\text{ADMM}} = \sum_{k=1}^{T_{\text{IPM}}} N_k &= \sum_{k=1}^{T_{\text{IPM}}} \log\left(\frac{2C(1+\|Q\|^2)}{\mu^k}\right)\left[\log\left(1+\min\left\{\frac{1}{C_3}, \frac{\mu^k}{4DC_3\beta}\right\}\right)\right]^{-1} \\ &= \sum_{k=1}^{T_{\text{IPM}}} \log\left(\frac{2C(1+\|Q\|^2)/\mu^0}{\gamma^k}\right)\left[\log\left(1+\min\left\{\frac{1}{C_3}, \frac{\mu^0\gamma^k}{4DC_3\beta}\right\}\right)\right]^{-1} \\ &= O\left(\frac{\kappa_A^2\|Q\|^2}{\varepsilon}\log\left(\frac{1}{\varepsilon}\right)\right). \end{aligned}$$

This completes the proof. ■

Corollary 3.8: *Assume that $m = O(n)$, the total arithmetic operation complexity of* ABIP *is*

$$T = O\left(n^3 + \frac{n^2\kappa_A^2\|Q\|^2}{\varepsilon}\log\left(\frac{1}{\varepsilon}\right)\right).$$

Proof: ABIP can be divided into two parts. In the first part, we decompose the matrix which requires $O(n^3)$ arithmetic operations. In the second part, we perform several matrix–vector and vector–vector operations where each ADMM iteration requires $O(n^2)$ arithmetic operations. So Theorem 3.7 implies the desired result. This completes the proof. ■

4. Implementation details

4.1. Termination criteria

So far we have not discussed how to terminate the outer loop of ABIP yet. In our implementation, we chose the one that is used in SCS [36]. Specifically, we run the algorithm until it finds a primal–dual optimal solution or a certificate of primal or dual infeasibility of the original LP pair (1), up to some tolerance. The detailed procedure is as follows. If $\tau_i^k > 0$ in $\mathbf{u}_i^k$, then let

$$\left(\frac{\mathbf{x_i^k}}{\tau_i^k}, \frac{\mathbf{s_i^k}}{\tau_i^k}, \frac{\mathbf{y_i^k}}{\tau_i^k}\right)$$

be the candidate solution which is guaranteed to satisfy the feasibility condition. It thus suffices to check if the residuals

$$\mathsf{pres}_i^k = \frac{A\mathbf{x}_i^k}{\tau_i^k} - \mathbf{b}, \quad \mathsf{dres}_i^k = \frac{A^\top\mathbf{y}_i^k}{\tau_i^k} + \frac{\mathbf{s_i^k}}{\tau_i^k} - \mathbf{c}, \quad \mathsf{dgap}_i^k = \frac{\mathbf{c}^\top\mathbf{x_i^k}}{\tau_i^k} - \frac{\mathbf{b}^\top\mathbf{y_i^k}}{\tau_i^k},$$

are small. More specifically, we terminate the algorithm if

$$\|\mathsf{pres}_i^k\| \le \varepsilon_{\mathsf{pres}}(1 + \|\mathbf{b}\|), \quad \|\mathsf{dres}_i^k\| \le \varepsilon_{\mathsf{dres}}(1 + \|\mathbf{c}\|),$$
$$\|\mathsf{dgap}_i^k\| \le \varepsilon_{\mathsf{dgap}}(1 + |\mathbf{c}^\top\mathbf{x}| + |\mathbf{b}^\top\mathbf{y}|),$$

are met. The quantities $\varepsilon_{\mathsf{pres}}$, $\varepsilon_{\mathsf{dres}}$ and $\varepsilon_{\mathsf{dgap}}$ are the primal residual, dual residual and duality gap tolerances, respectively.

On the other hand, if the current iterates satisfy that

$$\|A^\top\mathbf{y}_i^k + \mathbf{s}_i^k\| \le \varepsilon_{\mathsf{pinfeas}}\left(\frac{\mathbf{b}^\top\mathbf{y_i^k}}{\|\mathbf{b}\|}\right), \tag{68}$$

then $\frac{\mathbf{y_i^k}}{\mathbf{b}^\top\mathbf{y_i^k}}$ is an approximate certificate for the primal infeasibility with the tolerance $\varepsilon_{\mathsf{pinfeas}}$; or if the current iterates satisfy that

$$\|A\mathbf{x}_i^k\| \le \varepsilon_{\mathsf{dinfeas}}\left(\frac{-\mathbf{c}^\top\mathbf{x}_i^k}{\|\mathbf{c}\|}\right), \tag{69}$$

then $-\frac{\mathbf{x_i^k}}{\mathbf{c}^\top\mathbf{x_i^k}}$ is an approximate certificate for the dual infeasibility with the tolerance $\varepsilon_{\mathsf{dinfeas}}$.

4.2. Over relaxation

In practice, we implemented some techniques that can accelerate the algorithm. One of them is to incorporate a relaxation parameter in the ADMM [16]. Specifically, when applied to Algorithm 2, we replace all $\tilde{\mathbf{u}}_{i+1}^k$ in the $\mathbf{u}$- and $\mathbf{v}$-updates with

$$\alpha\tilde{\mathbf{u}}_{i+1}^k + (1-\alpha)\mathbf{u}_i^k,$$

where $\alpha \in [0,2]$ is a relaxation parameter. In that case, (32), (33), (34), (35) and (36) become

$$\mathbf{v}_{i+1}^k = \mathbf{v}_i^k - \alpha\tilde{\mathbf{u}}_{i+1}^k - (1-\alpha)\mathbf{u}_i^k + \mathbf{u}_{i+1}^k$$

and

$$\mathbf{y}_{i+1}^k = \alpha\tilde{\mathbf{y}}_{i+1}^k + (1-\alpha)\mathbf{y}_i^k,$$

$$\mathbf{x}_{i+1}^k = \frac{1}{2}\left[(\alpha\tilde{\mathbf{x}}_{i+1}^k + (1-\alpha)\mathbf{x}_i^k - \mathbf{s}_i^k) + \sqrt{\begin{array}{c}(\alpha\tilde{\mathbf{x}}_{i+1}^k + (1-\alpha)\mathbf{x}_i^k - \mathbf{s}_i^k)\circ(\alpha\tilde{\mathbf{x}}_{i+1}^k \\ +(1-\alpha)\mathbf{x}_i^k - \mathbf{s}_i^k) + \frac{4\mu^k}{\beta}\end{array}}\right],$$

$$\tau_{i+1}^k = \frac{1}{2}\left[(\alpha\tilde{\tau}_{i+1}^k + (1-\alpha)\tau_i^k - \kappa_i^k) + \sqrt{\begin{array}{c}(\alpha\tilde{\tau}_{i+1}^k + (1-\alpha)\tau_i^k - \kappa_i^k)\circ(\alpha\tilde{\tau}_{i+1}^k \\ +(1-\alpha)\tau_i^k - \kappa_i^k) + \frac{4\mu^k}{\beta}\end{array}}\right],$$

$$\theta_{i+1}^k = \alpha\tilde{\theta}_{i+1} + (1-\alpha)\theta_i^k.$$

When $\alpha = 1$, this reduces to the corresponding update in Algorithm 2 given above; when $\alpha > 1$, this is known as *over-relaxation*; when $\alpha < 1$, this is known as *under-relaxation*. Previous works [14,37] suggest that the performance of ADMM can be improved significantly if one sets $\alpha \approx 1.5$.

4.3. Barzilai–Borwein spectral method for selecting β

The performance of ADMM highly depends on the choice of β. One way to accelerate ADMM is to adaptively adjust β (see also [28,47]). In practice, we generated a sequence of $\{\beta^k\}_{k\geq 0}$, where β^k is only used in the kth outer iteration. Intuitively, the speed of travelling along the central path is determined by μ^k and β, implying that adjusting β in each outer iteration based on the iterates is equivalent to a predictor–corrector method [33]. The way we adaptively adjust β is based on the Barzilai–Borwein spectral method [4,49], which is proven to be superior than the residue balancing approach [47]. Indeed, for each $k \geq 0$, we select β^k using **spectral stepsize estimation** and **safeguarding** at the beginning of the kth outer iteration.

Spectral stepsize estimation: We calculate the first three iterates, i.e.$(\tilde{\mathbf{u}}_i^k, \mathbf{u}_i^k, \mathbf{v}_i^k)_{i=0}^{i=2}$, using a fixed $\beta^k > 0$ and an initial point $(\mathbf{y}_0^k, \mathbf{x}_0^k, \mathbf{s}_0^k, \tau_0^k, \kappa_0^k, \theta_0^k)$ obtained by (39) and $\mathbf{r}_0^k = 0$, $\xi_0^k = -n-1$. Then we estimate the curvature, i.e.

$$\hat{\mathbf{v}}_1^k = \mathbf{v}_0^k - \alpha\tilde{\mathbf{u}}_1^k - (1-\alpha)\mathbf{u}_0^k + \alpha\mathbf{u}_1^k, \quad \hat{\mathbf{v}}_2^k = \mathbf{v}_1^k - \alpha\tilde{\mathbf{u}}_2^k - (1-\alpha)\mathbf{u}_1^k + \alpha\mathbf{u}_2^k$$

and

$$\Delta\hat{\mathbf{v}}^k = \hat{\mathbf{v}}_2^k - \hat{\mathbf{v}}_1^k, \quad \Delta\tilde{\mathbf{u}}^k = \alpha(\tilde{\mathbf{u}}_2^k - \tilde{\mathbf{u}}_1^k) + (1-\alpha)(\mathbf{u}_1^k - \mathbf{u}_0^k).$$

As is typical in Barzilai–Borwein stepsize gradient methods [4], the spectral stepsizes φ^k_{SD} and φ^k_{MG} have the closed-form solutions as

$$\varphi^k_{\mathsf{SD}} = \frac{\langle \Delta\hat{\mathbf{v}}^k, \Delta\hat{\mathbf{v}}^k\rangle}{\langle \Delta\hat{\mathbf{v}}^k, \Delta\tilde{\mathbf{u}}^k\rangle}, \quad \varphi^k_{\mathsf{MG}} = \frac{\langle \Delta\hat{\mathbf{v}}^k, \Delta\tilde{\mathbf{u}}^k\rangle}{\langle \Delta\tilde{\mathbf{u}}^k, \Delta\tilde{\mathbf{u}}^k\rangle},$$

where SD and MG refer to *steepest descent* and *minimum gradient*, respectively, and $\varphi^k_{\mathsf{SD}} \geq \varphi^k_{\mathsf{MG}}$ due to the Cauchy–Schwarz inequality. We then consider the hybrid stepsize rule proposed in [57],

$$\varphi^k = \begin{cases} \varphi^k_{\mathsf{MG}}, & \text{if } 2\varphi^k_{\mathsf{MG}} > \varphi^k_{\mathsf{SD}}, \\ \varphi^k_{\mathsf{SD}} - \varphi^k_{\mathsf{MG}}/2, & \text{otherwise.} \end{cases} \tag{70}$$

Similarly, we calculate

$$\Delta\mathbf{u}^k = -(\mathbf{u}^k_2 - \mathbf{u}^k_1),$$

and the spectral stepsizes ψ^k_{SD} and ψ^k_{MG} as

$$\psi^k_{\mathsf{SD}} = \frac{\langle \Delta\hat{\mathbf{v}}^k, \Delta\hat{\mathbf{v}}^k\rangle}{\langle \Delta\hat{\mathbf{v}}^k, \Delta\mathbf{u}^k\rangle}, \quad \psi^k_{\mathsf{MG}} = \frac{\langle \Delta\hat{\mathbf{v}}^k, \Delta\mathbf{u}^k\rangle}{\langle \Delta\mathbf{u}^k, \Delta\mathbf{u}^k\rangle},$$

and consider the hybrid stepsize rule again,

$$\psi^k = \begin{cases} \psi^k_{\mathsf{MG}}, & \text{if } 2\psi^k_{\mathsf{MG}} > \psi^k_{\mathsf{SD}}, \\ \psi^k_{\mathsf{SD}} - \psi^k_{\mathsf{MG}}/2, & \text{otherwise.} \end{cases} \tag{71}$$

Safeguarding: We suggest a safeguarding heuristic by accessing the quality of the curvature estimates, i.e. only update the stepsize if the curvature estimates satisfy a reliability criterion. More specifically, we consider the following quantities defined in [49]

$$\varphi^k_{\mathsf{cor}} = \frac{\langle \Delta\hat{\mathbf{v}}^k, \Delta\tilde{\mathbf{u}}^k\rangle}{\|\Delta\hat{\mathbf{v}}^k\|\|\Delta\tilde{\mathbf{u}}^k\|}, \quad \psi^k_{\mathsf{cor}} = \frac{\langle \Delta\hat{\mathbf{v}}^k, \Delta\mathbf{u}^k\rangle}{\|\Delta\hat{\mathbf{v}}^k\|\|\Delta\mathbf{u}^k\|}.$$

The spectral stepsizes are updated only if the estimation is sufficiently reliable, i.e.

$$\hat{\beta}^k = \begin{cases} \sqrt{\varphi^k\psi^k}, & \text{if } \varphi^k_{\mathsf{cor}} > \varepsilon_{\mathsf{cor}} \text{ and } \psi^k_{\mathsf{cor}} > \varepsilon_{\mathsf{cor}}, \\ \varphi^k, & \text{if } \varphi^k_{\mathsf{cor}} > \varepsilon_{\mathsf{cor}} \text{ and } \psi^k_{\mathsf{cor}} \leq \varepsilon_{\mathsf{cor}}, \\ \psi^k, & \text{if } \varphi^k_{\mathsf{cor}} \leq \varepsilon_{\mathsf{cor}} \text{ and } \psi^k_{\mathsf{cor}} > \varepsilon_{\mathsf{cor}}, \\ \beta^k, & \text{otherwise,} \end{cases} \tag{72}$$

where $\varepsilon_{\mathsf{cor}} > 0$ is a quality threshold for the curvature estimates. Notice that $\hat{\beta}^k = \beta^k$ when both curvature estimates are deemed inaccurate while $\hat{\beta}^k \neq \beta^k$ but $\hat{\beta}^k \approx \beta^k$ implies that $\hat{\beta}^k$ and β^k are both suitable to be used in the kth outer iteration.

A heuristic selection: We select a threshold $\varepsilon_{\mathsf{penalty}} > 0$ and set

(1) If $\hat{\beta}^k \neq \beta^k$ and $|\hat{\beta}^k - \beta^k| \leq \varepsilon_{\mathsf{penalty}}$, then $\frac{\beta^k + \hat{\beta}^k}{2}$ will be used in the kth outer iteration.

(2) If $\hat{\beta}^k \neq \beta^k$ and $|\hat{\beta}^k - \beta^k| > \varepsilon_{\text{penalty}}$, then $\beta^k = \hat{\beta}^k$ and we redo the spectral step-size estimation and safeguarding with the same initial point.
(3) If $\hat{\beta}^k = \beta^k$, then the spectral step-size estimation and safeguarding will be continued based on the subsequent iterates.

4.4. Linear system solver

In this section, we discuss how to efficiently compute the projection onto the subspace $\mathcal{Q} = \{(\mathbf{u}, \mathbf{v}) : \ Q\mathbf{u} = \mathbf{v}\}$ in (37). This requires solving the linear system $(I + Q)\mathbf{u} = \mathbf{w}$ for some $\mathbf{w} \in \mathbb{R}^{m+n+2}$, i.e.

$$\begin{bmatrix} \mathbf{w}_\mathbf{y} \\ \mathbf{w}_\mathbf{x} \\ \mathbf{w}_\tau \end{bmatrix} = \begin{bmatrix} M & \mathbf{h} \\ -\mathbf{h}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_\mathbf{y} \\ \mathbf{u}_\mathbf{x} \\ \mathbf{u}_\tau \end{bmatrix},$$

where

$$M = \begin{bmatrix} I & A \\ -A^\top & I \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} -\mathbf{b} \\ \mathbf{c} \end{bmatrix}.$$

This implies that

$$\mathbf{u}_\tau = \mathbf{w}_\tau + \mathbf{h}^\top \begin{bmatrix} \mathbf{u}_\mathbf{y} \\ \mathbf{u}_\mathbf{x} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{u}_\mathbf{y} \\ \mathbf{u}_\mathbf{x} \end{bmatrix} = (M + \mathbf{h}\mathbf{h}^\top)^{-1} \left(\begin{bmatrix} \mathbf{w}_\mathbf{y} \\ \mathbf{w}_\mathbf{x} \end{bmatrix} - \mathbf{w}_\tau \mathbf{h} \right),$$

where $M + \mathbf{h}\mathbf{h}^\top$ is the Schur complement of the lower right block 1 in $I + Q$. Therefore, we can apply the Sherman–Morrison–Woodbury formula [39] to $(M + \mathbf{h}\mathbf{h}^\top)^{-1}$ and obtain

$$\begin{bmatrix} \mathbf{u}_\mathbf{y} \\ \mathbf{u}_\mathbf{x} \end{bmatrix} = \left(M^{-1} - \frac{M^{-1}\mathbf{h}\mathbf{h}^\top M^{-1}}{1 + \mathbf{h}^\top M^{-1}\mathbf{h}} \right) \left(\begin{bmatrix} \mathbf{w}_\mathbf{y} \\ \mathbf{w}_\mathbf{x} \end{bmatrix} - \mathbf{w}_\tau \mathbf{h} \right)$$

and

$$\mathbf{u}_\tau = \mathbf{w}_\tau + \mathbf{c}^\top \mathbf{u}_\mathbf{x} - \mathbf{b}^\top \mathbf{u}_\mathbf{y}.$$

Therefore, (37) reduces to

$$\begin{bmatrix} \tilde{\mathbf{y}}_{i+1}^k \\ \tilde{\mathbf{x}}_{i+1}^k \end{bmatrix} = \left(M^{-1} - \frac{M^{-1}\mathbf{h}\mathbf{h}^\top M^{-1}}{1 + \mathbf{h}^\top M^{-1}\mathbf{h}} \right) \left(\begin{bmatrix} \mathbf{y}_i^k + \mathbf{r}_i^k \\ \mathbf{x}_i^k + \mathbf{s}_i^k \end{bmatrix} - (\tau_i^k + \kappa_i^k)\mathbf{h} \right) \tag{73}$$

and

$$\tilde{\tau}_{i+1}^k = \tau_i^k + \kappa_i^k + \mathbf{c}^\top \tilde{\mathbf{x}}_{i+1}^k - \mathbf{b}^\top \tilde{\mathbf{y}}_{i+1}^k. \tag{74}$$

Remark 4.1: In view of practical implementation, we can calculate and cache $M^{-1}\mathbf{h}$ before the first iteration. In the subsequent iterations, the main computational cost lies in the calculation of

$$\begin{bmatrix} \mathbf{z}_\mathbf{y} \\ \mathbf{z}_\mathbf{x} \end{bmatrix} = M^{-1} \begin{bmatrix} \mathbf{y}_i^k + \mathbf{r}_i^k \\ \mathbf{x}_i^k + \mathbf{s}_i^k \end{bmatrix},$$

while the other simple vector operations using cached quantities are cheap.

To solve the linear system of the form

$$\begin{bmatrix} I & -A \\ -A^\top & -I \end{bmatrix} \begin{bmatrix} \mathbf{z}_\mathbf{y} \\ -\mathbf{z}_\mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_i^k + \mathbf{r}_i^k \\ \mathbf{x}_i^k + \mathbf{s}_i^k \end{bmatrix}, \tag{75}$$

we follow [36] and propose two approaches. The first approach is to compute a sparse permuted LDL$^\top$ factorization [13] of the matrix in (75) before the first iteration and use this cached factorization to solve the system in the subsequent steps. When the factorization cost dominates, this technique is very effective since only one factorization is necessary and all iterations after that can be carried out quickly. The second approach is to approximately solve the system, involves first reformulate (75) and then solve it with the conjugate gradient method (CG) [41,46]. Each iteration of conjugate gradient requires multiplying once by A and once by $A^\top$, each of which can be parallelized. If A is very sparse, then these multiplications can be performed especially quickly; when A is dense, it may be better to first form $G = I + AA^\top$ in the setup phase. We terminate the CG iterations using the standard same criterion; see [36] for the details.

4.5. Presolving and postsolving

Now we discuss issues in analysing large and sparse LPs prior to solving them with ABIP. First, we remove several computational expensive subprocedures, e.g.forcing, dominated and duplicate rows and columns procedures, as used in [3,24,27,34]. Second, we use Dulmage–Mendelsohn decomposition [38] to remove all the dependent rows in A and reformulate the original problem in the form of problem (1).

We consider LPs formulated in the following form:

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{l} \le \mathbf{x} \le \mathbf{w}, \end{aligned}$$

where A has some linearly dependent rows. Before solving them with ABIP, we run a simple presolve procedure. More specifically, we detect and remove empty rows, singleton rows, fixed variables and empty columns, together with removing all the linearly dependent rows, i.e.

(P1) An empty row: $\exists i:\ a_{ij} = 0,\ \forall j$. Either the ith constraint is redundant or infeasible.
(P2) An empty column: $\exists j:\ a_{ij} = 0,\ \forall i$. x_j is fixed at one of its bounds or the problem is unbounded.
(P3) An infeasible variable: $\exists j:\ l_j > w_j$. The problem is trivially infeasible.
(P4) A fixed variable: $\exists j:\ l_j = w_j$. x_j can be substituted out of the problem.
(P5) A free variable: $\exists j:\ l_j = -\infty,\ w_j = \infty$. x_j can be substituted by two nonnegative variables x_j^+ and x_j^-.
(P6) A singleton row: $\exists (i,k):\ a_{ij} = 0,\ \forall j \neq k,\ a_{jk} \neq 0$. The ith constraint fixes variable $x_j = \frac{b_i}{a_{ik}}$.
(P7) Dulmage–Mendelsohn decomposition. All the linearly dependent rows are detected and removed.

Then we have a reduced LP problem as follows:

$$\begin{aligned} \min \quad & \tilde{\mathbf{c}}^\top \tilde{\mathbf{x}} \\ \text{s.t.} \quad & \tilde{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}, \\ & \tilde{\mathbf{l}} \le \tilde{\mathbf{x}} \le \tilde{\mathbf{w}}, \end{aligned}$$

where $\tilde{\mathbf{x}}, \tilde{\mathbf{l}}, \tilde{\mathbf{w}} \in \mathbb{R}^{\tilde{n}}$ and $\tilde{A} \in \mathbb{R}^{\tilde{m}+\tilde{n}}$ have full row rank. The last step is to reformulate the above problem as in the form of problem (1). After the presolving, it is guaranteed that $\tilde{\mathbf{l}} > -\infty$. Now we can define $U = \{j : \tilde{\mathbf{w}}_j < +\infty\}$ and $\bar{\mathbf{x}} = \tilde{\mathbf{x}} - \tilde{\mathbf{l}}$, and obtain the desired LP problem,

$$\begin{aligned} \min \quad & \tilde{\mathbf{c}}^\top \bar{\mathbf{x}} \\ \text{s.t.} \quad & \tilde{A}\bar{\mathbf{x}} = \tilde{\mathbf{b}} - \tilde{A}\tilde{\mathbf{l}}, \\ & \tilde{\mathbf{x}}_U + \tilde{\mathbf{s}} = \tilde{\mathbf{w}}_U - \tilde{\mathbf{l}}_U, \\ & \tilde{\mathbf{x}} \ge 0,\ \tilde{\mathbf{s}} \ge 0. \end{aligned}$$

After the presolve procedure, the reduced problem is ready to be solved by ABIP. A post-solve procedure is used to convert the solution to the reduced problem back to the solution to the original problem.

Remark 4.2: Since our algorithm is a purely first-order algorithm, we also conduct the scale procedure after the presolve procedure to make the problems more well-conditioned. We refer to [36] for more details.

5. Numerical experiments

In this section, we report experimental results of ABIP on solving randomly generated LPs and 6 instances from UCI collection.[1] We also report the results on 114 instances from NETLIB collection[2] in the online supplementary materials due to the space limits. To make the comparison fair, we compare the direct and indirect ABIP with different group of state-of-the-art solvers. More specifically, we compare the direct ABIP with SDPT3 [44], MOSEK [2] and the direct SCS [36] and the indirect ABIP with DSDP-CG [6] and the indirect SCS [36].

Our ABIP code is written in C with a MATLAB interface. It has multi-threaded and single-threaded versions. The direct ABIP computes the (approximate) projections onto the subspace using a direct method based on a single-threaded sparse permuted $\mathrm{LDL}^\top$ decomposition from the SuiteSparse package [1,12,13], and the indirect ABIP using a pre-conditioned conjugate gradient method (denoted by ABIP-CG). For a fair comparison, we compare ABIP with SCS (v2.0.0) where the direct solver is single-threaded.

In the experimental results reported below, we use the termination criteria for ABIP in Section 4.1 with default values

$$\epsilon_{\text{pres}} = \epsilon_{\text{dres}} = \epsilon_{\text{dgap}} = \epsilon_{\text{pinfeas}} = \epsilon_{\text{dinfeas}} = 10^{-3} \text{or} 10^{-5}, \tag{76}$$

and that for Barzilai–Borwein spectral method in Section 4.3 with default values

$$\epsilon_{\text{cor}} = 0.2, \quad \epsilon_{\text{penalty}} = 0.1.$$

For the scaling update in (39), a better heuristic update for $\mathbf{x}$, $\mathbf{s}$, τ and κ is available in practice and we used it in our software package.[3] Indeed, we used

$$((\mathbf{x}_0^{k+1})_j, (\mathbf{s}_0^{k+1})_j) = \begin{cases} ((\mathbf{x}_{i+1}^k)_j, \gamma \cdot (\mathbf{s}_{i+1}^k)_j), & \text{if } (\mathbf{x}_{i+1}^k)_j \geq (\mathbf{s}_{i+1}^k)_j, \\ (\gamma \cdot (\mathbf{x}_{i+1}^k)_j, (\mathbf{s}_{i+1}^k)_j), & \text{if } (\mathbf{s}_{i+1}^k)_j \geq (\mathbf{x}_{i+1}^k)_j. \end{cases} \quad j = 1, 2, \ldots, n,$$

and

$$((\tau_0^{k+1}), (\kappa_0^{k+1})) = \begin{cases} (\tau_{i+1}^k, \gamma \cdot \kappa_{i+1}^k), & \text{if } \tau_{i+1}^k \geq \kappa_{i+1}^k, \\ (\gamma \cdot \tau_{i+1}^k, \kappa_{i+1}^k), & \text{if } \kappa_{i+1}^k \geq \tau_{i+1}^k. \end{cases}$$

The over-relaxation parameter is set to $\alpha = 1.8$. Moreover, we set the maximum number of ADMM steps of ABIP to 10^6. If the target accuracy in (76) is not achieved in 10^6 ADMM steps, we terminate the code and claim that ABIP fails to solve this instance and use '–' in the table to indicate the failure.

The decreasing ratio γ is adjusted according to the value of the barrier parameter μ, primal/dual feasibility violations and the duality gap. More specifically, we increase γ as the iterate approaches the optimal solution. The objective value reported for all methods in the experiments below is the one after postsolving. The time required to do any preprocessing, i.e. presolving, postsolving, do/undo scaling and matrix factorization are included in the total solve times. All the experiments were carried out on a laptop with Linux system and 8 2.60GHz cores and 16Gb of RAM.

For other four solvers, we use the following stopping criteria. For SCS, we change the default α from 1.5 to 1.8, use the tolerance in the stopping tolerance as 10^{-3} or 10^{-5}, and set the maximum number ADMM steps to 10^6. These changes are made to ensure a more fair comparison, because we found that the default parameter setting needs much longer time to converge. Moreover, we use SCS-CG to denote SCS with a preconditiond conjugate gradient method computing the projections onto the subspaces. For SDPT3 and DSDP-CG, the maximum number of interior point steps are both set as the default value 100. For MOSEK, we use the default settings. For all these solvers, we claim that they fail to solve an instance (denoted by '–' in the tables) if after the codes terminate, the target accuracy in (76) is not achieved.

5.1. Random LP instances

In this section, we test the five solvers on randomly generated dense LPs. First, we generate a Gaussian random vector $\mathbf{x} \in \mathbb{R}^n$ and split its entries randomly into three groups. The first group consists 60% of entries and their values are set to zero. The second group consists of 10% entries and their values are zoom in for 10 times larger. The rest of the entries are in the third group and their values are zoomed out for 10 times smaller. We then generate vector $\mathbf{s} \in \mathbb{R}^n$ such that $\mathbf{x}_i \mathbf{s}_i = 0$ for all i, and nonzero entries of $\mathbf{s}$ follow standard normal distribution. This ensures the complementary slackness and zero duality gap. We generate the data matrix $A = U^\top DV \in \mathbb{R}^{m \times n}$ where $D = \text{diag}(\text{randn}(m, 1))$, $U = \text{sprandn}(m, m, 0.2)$ and $V = \text{sprandn}(m, n, 0.2)$. We generate the dual solution $\mathbf{y} \in \mathbb{R}^m$ with entries following standard normal distribution. Finally, we set $\mathbf{b} = A\mathbf{x}$ and $\mathbf{c} = A^\top \mathbf{y} + \mathbf{s}$, which ensures primal and dual feasibility. The solution to the problem is not necessarily unique, but the optimal value is given by $\mathbf{c}^\top \mathbf{x} = \mathbf{b}^\top \mathbf{y}$.

Table 1. Performance of four direct solvers on randomly generated LPs. CPU times are in seconds.

(m, n)	Solver	obj	pres	dres	dgap	time
(500, 5000)	ABIP	1.01e+04	1.10e−04	6.82e−04	9.56e−04	1.06e+01
	SCS	1.01e+04	5.98e−04	4.28e−04	3.06e−04	1.42e+01
	SDPT3	1.01e+04	1.22e−08	2.72e−05	2.20e−05	4.74e+01
	MOSEK	1.01e+04	1.70e−12	4.51e−14	1.15e−12	1.22e+00
(500, 10000)	ABIP	−2.65e+04	6.88e−05	3.84e−04	9.60e−04	2.57e+01
	SCS	−2.65e+04	8.71e−04	4.37e−04	3.47e−04	2.21e+02
	SDPT3	−2.65e+04	1.83e−08	4.81e−05	3.99e−05	8.22e+01
	MOSEK	−2.65e+04	1.03e−09	1.31e−09	2.07e−12	2.36e+00
(500, 20000)	ABIP	3.41e+04	7.99e−05	2.28e−04	9.75e−04	6.16e+01
	SCS	3.41e+04	7.91e−04	3.24e−04	3.10e−04	1.19e+02
	SDPT3	3.41e+04	1.15e−08	9.55e−05	5.62e−05	1.73e+02
	MOSEK	3.41e+04	4.55e−12	2.76e−14	6.15e−12	5.97e+00
(1000, 5000)	ABIP	−1.35e+04	7.12e−05	4.28e−04	8.83e−04	3.99e+01
	SCS	−1.35e+04	9.57e−04	9.11e−04	6.36e−04	1.51e+02
	SDPT3	−1.35e+04	4.02e−09	7.22e−06	6.31e−06	2.62e+02
	MOSEK	−1.35e+04	6.00e−10	4.97e−11	8.72e−10	3.90e+00
(1000, 10000)	ABIP	1.57e+05	9.27e−05	6.73e−04	9.73e−04	5.60e+01
	SCS	1.57e+05	9.97e−04	7.49e−04	2.15e−05	2.49e+02
	SDPT3	1.57e+05	6.34e−08	3.80e−05	3.15e−05	3.33e+02
	MOSEK	1.57e+05	5.01e−10	2.05e−13	1.03e−10	7.04e+00
(1000, 20000)	ABIP	1.47e+05	5.96e−05	4.56e−04	9.97e−04	1.22e+02
	SCS	1.46e+05	9.68e−04	5.10e−04	3.36e−05	5.01e+02
	SDPT3	1.46e+05	5.91e−08	1.10e−04	6.97e−05	8.34e+02
	MOSEK	1.46e+05	2.54e−11	1.23e−13	6.39e−12	1.40e+01
(2000, 10000)	ABIP	−5.32e+04	3.76e−05	2.64e−04	9.85e−04	8.89e+02
	SCS	−5.32e+04	9.02e−04	8.35e−04	1.62e−04	9.94e+02
	SDPT3	−5.34e+04	1.03e−08	5.81e−06	5.69e−06	2.29e+03
	MOSEK	−5.34e+04	5.52e−11	5.74e−11	7.22e−12	2.14e+01
(2000, 20000)	ABIP	−7.40e+04	3.36e−05	2.15e−04	9.92e−04	4.04e+03
	SCS	−7.40e+04	9.18e−04	7.06e−04	5.26e−04	5.47e+03
	SDPT3	−7.42e+04	4.93e−08	1.79e−04	2.22e−04	3.37e+03
	MOSEK	−7.42e+04	1.97e−10	7.70e−12	1.14e−10	3.62e+01
(4000, 20000)	ABIP	−3.50e+05	2.88e−05	2.54e−04	1.00e−03	5.67e+03
	SCS	−3.50e+05	9.55e−04	8.84e−04	4.09e−04	6.72e+03
	SDPT3	−3.50e+05	1.09e−08	9.06e−06	9.75e−06	1.64e+04
	MOSEK	−3.50e+05	5.04e−10	2.92e−10	1.95e−11	9.06e+01

Results: We report the comparison results of the four direct solvers in Table 1 and the three indirect solvers in Table 2. The results in Table 1 show that ABIP compares favourably to SCS and SDPT3, though it is inferior to the commercial solver MOSEK. For results in Table 2, our ABIP-CG compares favourably to SCS-CG and DSDP-CG.

5.2. Sparse inverse covariance estimation

In this section, we compare the five solvers on solving the following problem which arises from machine learning applications:

$$\begin{aligned} \min_{\Omega\in\mathbb{R}^{d\times d}} \quad & \|\Omega\|_1 \\ \text{s.t.} \quad & \|\Sigma\Omega - I_d\|_\infty \le \lambda, \end{aligned} \tag{77}$$

where $\Sigma \in \mathbb{R}^{d\times d}$ denotes a sample covariance matrix, and $\lambda > 0$ denotes some noisy tolerance. This problem, known as sparse inverse covariance estimation (SICE), is widely studied in high-dimensional statistics and machine learning [8]. For given Σ, SICE (77)

Table 2. Performance of three indirect solvers on randomly generated LPs. CPU times are in seconds.

(m, n)	Solver	obj	pres	dres	dgap	time
(500, 5000)	ABIP-CG	−9.32e+02	1.75e−05	2.01e−04	9.66e−04	1.34e+03
	SCS-CG	−9.33e+02	6.00e−04	5.13e−04	9.22e−04	3.45e+03
	DSDP-CG	−9.32e+02	1.03e−10	–	2.45e−04	6.53e+01
(500, 10000)	ABIP-CG	1.08e+04	4.70e−05	3.21e−04	9.95e−04	8.73e+02
	SCS-CG	1.07e+04	8.38e−04	4.95e−04	3.85e−04	2.43e+03
	DSDP-CG	1.07e+04	9.88e−11	–	1.25e−04	7.60e+01
(500, 20000)	ABIP-CG	1.93e+04	2.82e−05	1.22e−04	9.94e−04	2.44e+03
	SCS-CG	1.93e+04	8.66e−04	3.62e−04	9.99e−04	3.05e+03
	DSDP-CG	1.93e+04	2.13e−11	–	5.32e−05	1.52e+02
(1000, 5000)	ABIP-CG	7.73e+04	1.23e−04	7.69e−04	9.99e−04	8.53e+02
	SCS-CG	7.71e+04	9.00e−04	8.12e−04	1.65e−04	4.59e+03
	DSDP-CG	7.71e+04	1.24e−09	–	4.45e−04	3.28e+02
(1000, 10000)	ABIP-CG	−5.41e+04	6.38e−05	2.83e−04	9.93e−04	2.80e+03
	SCS-CG	−5.44e+04	9.97e−04	8.01e−04	4.20e−05	8.82e+03
	DSDP-CG	−5.43e+04	2.51e−11	–	3.68e−05	4.72e+02
(1000, 20000)	ABIP-CG	−1.70e+05	7.15e−05	3.32e−04	9.77e−04	4.53e+03
	SCS-CG	−1.71e+05	8.78e−04	7.34e−04	1.10e−04	3.34e+04
	DSDP-CG	−1.71e+05	6.39e−12	–	2.17e−04	5.86e+02
(2000, 10000)	ABIP-CG	1.16e+05	4.91e−05	3.70e−04	9.86e−04	9.47e+03
	SCS-CG	1.15e+05	9.38e−04	8.16e−04	8.01e−05	3.19e+04
	DSDP-CG	1.15e+05	1.19e−09	–	4.43e−04	3.01e+03

Note: DSDP-CG does not provide the dual residuals.

aims to find a perturbed inverse convariance matrix which is also sparse. Note that SICE (77) is separable for columns of $\Omega = (\beta_1, \beta_2, \ldots, \beta_d)$, and thus can be decomposed to d problems as follows:

$$\begin{aligned} \min_{\beta_j \in \mathbb{R}^d} \quad & \|\beta_j\|_1 \\ \text{s.t.} \quad & \|\Sigma\beta_j - e_j\|_\infty \le \lambda. \end{aligned} \tag{78}$$

Problem (78) can be written as a standard LP as follows:

$$\begin{aligned} \min \quad & \mathbf{e}^\top \beta^+ + \mathbf{e}^\top \beta^- \\ \text{s.t.} \quad & \Sigma\beta^+ - \Sigma\beta^- + w^+ = \lambda\mathbf{e} + e_j, \\ & w^+ + w^- = 2\lambda\mathbf{e}, \\ & \beta^+,\ \beta^-,\ w^+,\ w^- \ge 0, \end{aligned} \tag{79}$$

where the number of variables is $n = 4d$ and the number of constraints is $m = 2d$. In our experiment, we set $\lambda = \frac{3}{2}\sqrt{\frac{\log(d)}{N}}$ where N is the number of sampled data in the original data.

Problem instances: We obtained Σ from the UCI Machine Learning Repository.[4] The statistics of the six selected instances is summarized in Table 3.

Results: Detailed numerical results are reported in Tables 4 and 5. From Table 4, we see that MOSEK is the best among all the direct solvers possibly because of its preprocessing procedure of detecting dependent columns. ABIP and SCS are comparable and the speedup over SDPT3 is more significant as the problem size increases. ABIP is more robust than SCS and SDPT3 as SCS fails on ucihapt and SDPT3 fail on gisette. From Table 5, ABIP-CG is more robust than SCS-CG and DSDP-CG as SCS-CG fails on ucihapt and ucihar and DSDP-CG fail on gisette.

Table 3. Statistics of six instances in UCI collection.

	Problem statistics						
Name	Samples (N)	Variables (d)	Threshold (λ)	Rows (m)	Cols (n)	Nonzeros	Sparsity
gisette	13,500	5000	0.0377	10,000	20,000	50,015,000	0.2501
isolet	7797	618	0.0431	1236	2472	765,702	0.2506
sEMG	1800	3000	0.1000	6000	12,000	18,009,000	0.2501
sEMGday	3600	2500	0.0699	5000	10,000	12,507,500	0.2501
ucihapt	10,929	561	0.0361	1122	2244	631,125	0.2507
ucihar	10,299	561	0.0372	1122	2244	631,125	0.2507

Table 4. Performance of four direct solvers on UCI. CPU times are in seconds.

Name	Method	obj	pres	dres	dgap	time
gisette	ABIP	1.25e−05	1.00e−03	1.16e−04	6.47e−06	1.03e+03
	SCS	1.25e−05	8.89e−04	9.54e−04	4.13e−08	2.31e+03
	SDPT3	–	–	–	–	–
	MOSEK	1.25e−05	7.11e−13	4.63e−15	2.89e−13	7.46e+01
isolet	ABIP	6.37e+02	1.56e−04	9.99e−04	2.77e−08	1.89e+02
	SCS	6.37e+02	7.34e−04	8.85e−04	1.28e−05	8.68e+02
	SDPT3	6.37e+02	1.67e−08	1.45e−06	4.62e−08	2.89e+01
	MOSEK	6.37e+02	2.89e−08	1.92e−10	2.43e−10	6.95e−01
sEMG	ABIP	1.06e+02	9.68e−04	6.21e−04	3.19e−04	2.61e+02
	SCS	1.07e+02	9.48e−04	8.61e−04	5.27e−05	2.85e+02
	SDPT3	1.07e+02	2.35e−08	5.38e−06	4.06e−06	3.52e+03
	MOSEK	1.07e+02	3.05e−09	5.76e−15	9.95e-10	2.99e+01
sEMGday	ABIP	1.89e+02	4.04e−04	9.93e-04	4.55e−05	1.67e+02
	SCS	1.89e+02	4.18e−04	9.99e−04	1.70e−05	2.43e+02
	SDPT3	1.89e+02	1.23e−08	2.43e−06	1.21e−06	2.04e+03
	MOSEK	1.89e+02	1.85e−09	1.10e−13	1.03e−09	1.76e+01
ucihapt	ABIP	3.64e+03	2.23e−05	1.00e−03	4.65e−07	8.44e+02
	SCS	–	–	–	–	–
	SDPT3	3.63e+03	7.73e−07	3.79e-07	4.25e−08	4.20e+01
	MOSEK	3.63e+03	1.06e−10	1.56e−11	3.17e−14	7.46e−01
ucihar	ABIP	2.05e+03	1.51e−04	1.00e−03	6.30e−07	1.42e+02
	SCS	2.05e+03	3.55e−04	9.99e−04	1.88e−06	1.00e+03
	SDPT3	2.05e+03	3.99e−08	1.67e−09	1.08e−13	4.65e+01
	MOSEK	2.05e+03	2.16e−08	6.57e−10	5.77e−11	5.87e−01

5.3. NETLIB LP collections

In this section, we report the performance of all five solvers on 114 feasible instances from NETLIB collection.[5] NETLIB is a collection of LPs from real applications. It has been recognized as the standard testing data set for LP. Due to the space constraint, the detailed numerical results on NETLIB are given in the online supplementary materials. Here we give a brief discussion on our observations from the numerical results.

We observe that neither SCS nor SCS-CG is as robust as other solvers. To be more specific, SCS only successfully solved 89 and 63 problems when the target accuracy is set to 10^{-3} and 10^{-5} respectively, while SCS-CG only successfully solved 86 problems. MOSEK is the most robust one while ABIP, ABIP-CG, SDPT3 and DSDP-CG are comparable, and they all significantly outperform SCS and SCS-CG. This phenomenon can be explained by the superior robustness of the interior-point methods over the pure first-order methods. Furthermore, ABIP and ABIP-CG are also more efficient than SCS and SCS-CG on the collection of problem instances that can be solved by all the solvers. Promising performances

Table 5. Performance of three indirect solvers on UCI. CPU times are in seconds.

Name	Method	obj	pres	dres	dgap	time
gisette	ABIP-CG	1.23e−05	9.87e−04	1.21e−04	4.48e−06	1.42e+04
	SCS-CG	1.23e−05	9.93e−04	6.35e−04	1.85e−07	6.11e+04
	DSDP-CG	–	–	–	–	–
isolet	ABIP-CG	6.37e+02	1.55e−04	9.98e−04	3.78e−07	2.18e+03
	SCS-CG	6.37e+02	9.36e−04	9.52e−04	1.30e−06	1.20e+04
	DSDP-CG	6.37e+02	3.70e−10	–	1.65e−04	4.24e+01
sEMG	ABIP-CG	1.06e+02	9.95e−04	6.66e−04	2.41e−04	1.80e+03
	SCS-CG	1.07e+02	9.56e−04	7.92e−04	1.07e−04	2.81e+03
	DSDP-CG	1.07e+02	2.38e−12	–	3.18e−04	6.15e+03
sEMGday	ABIP-CG	1.89e+02	4.07e−04	9.98e−04	3.76e−05	1.13e+03
	SCS-CG	1.89e+02	4.72e−04	9.62e−04	3.49e−05	2.10e+03
	DSDP-CG	1.89e+02	6.63e−11	–	2.76e−04	3.32e+03
ucihapt	ABIP-CG	3.64e+03	2.36e−05	1.00e−03	4.73e−07	1.12e+04
	SCS-CG	–	–	–	–	–
	DSDP-CG	3.63e+03	1.58e−07	–	1.82e−04	3.57e+01
ucihar	ABIP-CG	2.05e+03	1.46e−04	1.00e−03	5.68e−07	2.24e+03
	SCS-CG	–	–	–	–	–
	DSDP-CG	2.05e+03	4.42e−08	–	3.95e−04	3.27e+01

of ABIP and ABIP-CG strongly support the use of the first-order interior-point method on very large LP problems.

Remark 5.1: From the numerical results presented here and the ones in the online appendix, we have the following observations. (i) The second-order IPM is still the best option either when the dataset is of small or medium size, or when solution with high accuracy is desired. (ii) The first-order IPM (ABIP) is better than the vanilla first-order method (ADMM) when the dataset is extremely large. (iii) We only focused on LP in this paper; how to design general first-order IPM for other convex optimization problems remains a future research topic.

6. Conclusion

In this paper, we present a novel implementation of the primal–dual interior point method to solve linear programs via self-dual embedding. In our approach, we use the ADMM to track the central path. Therefore, the new approach is an implementation of first-order interior point method (IPM). As such, it inherits intrinsic properties of the IPM. We present a theoretical analysis showing that the overall complexity of ADMM steps is $O(\frac{1}{\varepsilon}\log(\frac{1}{\varepsilon}))$, and the extensive numerical experiments demonstrate that the new algorithm is stable in performance and scalable in size. For the future research, we are interested in improving the numerical stability by incorporating advanced iterative methods, and the numerical efficiency by incorporating distributed computing environment.

Notes

1. https://archive.ics.uci.edu/ml/datasets.html
2. http://users.clas.ufl.edu/hager/coap/format.html
3. We sincerely thank an anonymous reviewer for suggesting this scaling strategy, which significantly improved the efficiency of the algorithm.
4. https://archive.ics.uci.edu/ml/datasets.html
5. http://www.netlib.org/lp/

Acknowledgments

We would like to thank two anonymous referees for their insightful and constructive suggestions that greatly improved the presentation of this paper. We would like to thank Zaiwen Wen for fruitful discussions regarding this project and many helps on the codes. The research of Shiqian Ma was supported in part by NSF grants DMS-1953210 and CCF-2007797, and UC Davis CeDAR (Center for Data Science and Artificial Intelligence Research) Innovative Data Science Seed Funding Program.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

The research of Shiqian Ma was supported in part by NSF grants Division of Mathematical Sciences PDMS-1953210 and Division of Computing and Communication Foundations CCF-2007797, and UC Davis CeDAR (Center for Data Science and Artificial Intelligence Research) Innovative Data Science Seed Funding Program.

Notes on contributors

Tianyi Lin is a Ph.D. student in the Department of Electrical Engineering and Computer Science at UC Berkeley. His research interests include algorithms, optimal transport and game-theoretical learning arising from machine learning. He earned a BS in Mathematics Science, majoring in Pure Mathematics and Applied Mathematics from the Nanjing University, and a MS in Pure Mathematics and Statistics from the University of Cambridge.

Shiqian Ma received his PhD degree in Industrial Engineering and Operations Research from Columbia University in 2011. He is currently an associate professor in the Department of Mathematics at University of California, Davis.

Yinyu Ye is currently the K.T. Li Chair Professor of Engineering at Department of Management Science and Engineering and Institute of Computational and Mathematical Engineering, Stanford University. He received the B.S. degree in System Engineering from the Huazhong University of Science and Technology, China, and the M.S. and Ph.D. degrees in Engineering-Economic Systems and Operations Research from Stanford University. He received several academic awards including the 2009 John von Neumann Theory Prize for fundamental sustained contributions to theory in Operations Research and the Management Sciences. He has also supervised numerous doctoral students at Stanford who received the Prizes of INFORMS Nicholson Paper Competition.

Shuzhong Zhang is Professor at the Department of Industrial and System Engineering, University of Minnesota. He received a B.Sc. degree in Applied Mathematics from Fudan University in 1984, and a Ph.D degree in Operations Research and Econometrics from the Tinbergen Institute, Erasmus University, in 1991. He had held faculty positions at University of Groningen, Erasmus University, and The Chinese University of Hong Kong. Currently, he is under a joint faculty appointment scheme between University of Minnesota and the Chinese University of Hong Kong, Shenzhen. He received the Erasmus University Research Prize in 1999, the CUHK Vice-Chancellor Exemplary Teaching Award in 2001, the SIAM Outstanding Paper Prize in 2003, the IEEE Signal Processing Society Best Paper Award in 2010, and the 2015 SPS Signal Processing Magazine Best Paper Award. Dr. Zhang was an elected Council Member at Large of the MPS (Mathematical Programming Society) (2006–2009), and served as Vice-President of the Operations Research Society of China (ORSC) (2008–2012).

References

[1] P.R. Amestoy, T.A. Davis, and I.S. Duff, *Algorithm 837: AMD, an approximate minimum degree ordering algorithm*, ACM Trans. Math. Softw. 30(3) (2004), pp. 381–388.

[2] E.D. Andersen, *The homogeneous and self-dual model and algorithm for linear optimization*, Technical Report TR-1-2009, MOSEK ApS., 2009.

[3] E.D. Andersen and K.D. Andersen, *Presolving in linear programming*, Math. Program. 71(2) (1995), pp. 221–245.

[4] J. Barzilai and J.M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal. 8(1) (1988), pp. 141–148.

[5] D.A. Bayer and J.C. Lagarias, *The nonlinear geometry of linear programming. I. Affine and projective scaling trajectories*, Trans. Am. Math. Soc. 314(2) (1989), pp. 499–526.

[6] S.J. Benson, Y. Ye, and X. Zhang, *Solving large-scale sparse semidefinite programs for combinatorial optimization*, SIAM J. Optim. 10(2) (2000), pp. 443–461.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends Mach. Learn. 3(1) (2011), pp. 1–122.

[8] T. Cai, W. Liu, and X. Luo, *A constrained ℓ_1 minimization approach to sparse precision matrix estimation*, J. Am. Stat. Assoc. 106(494) (2011), pp. 594–607.

[9] P.L. Combettes and J.-C. Pesquet, *A Douglas–Rachford splitting approach to nonsmooth convex variational signal recovery*, IEEE J. Sel. Top. Signal Process. 1(4) (2007), pp. 564–574.

[10] Y. Cui, K. Morikuni, T. Tsuchiya, and K. Hayami, *Implementation of interior-point methods for LP based on Krylov subspace iterative solvers with inner-iteration preconditioning*, Comput. Optim. Appl. 74(1) (2019), pp. 143-176.

[11] G.B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, 1963.

[12] T.A. Davis, *Algorithm 849: A concise sparse Cholesky factorization package*, ACM Trans. Math. Softw. 31(4) (2005), pp. 587–591.

[13] T.A. Davis, *Direct Methods for Sparse Linear Systems*, SIAM Fundamentals of Algorithms, SIAM, Philadephia, 2006.

[14] J. Eckstein, *Parallel alternating direction multiplier decomposition of convex programs*, J. Optim. Theory Appl. 80(1) (1994), pp. 39–62.

[15] J. Eckstein, *Augmented Lagrangian and alternating direction methods for convex optimization: a tutorial and some illustrative computational results*, preprint (2012).

[16] J. Eckstein and D.P. Bertsekas, *On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Program. 55(1) (1992), pp. 293–318.

[17] M. Fortin and R. Glowinski, *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*, North-Holland Pub. Co., Amsterdam, 1983.

[18] K. Fountoulakis, J. Gondzio, and P. Zhlobich, *Matrix-free interior point method for compressed sensing problems*, Math. Program. Comput. 6 (2014), pp. 1–31.

[19] D. Gabay, *Applications of the method of multipliers to variational inequalities*, in *Augmented Lagrangian Methods: Applications to the Solution of Boundary Value Problems*, M. Fortin and R. Glowinski, eds., North-Holland, Amsterdam, 1983.

[20] P.E. Gill, W. Murry, M.A. Saunders, J.A. Tomlin, and M.H. Wright, *On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method*, Math. Program. 36(2) (1986), pp. 183–209.

[21] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and Operator-Splitting Methods in Nonlinear Mechanics*, SIAM, Philadelphia, PA, 1989.

[22] R. Glowinski and A. Marrocco, *Sur l'approximation par èlèments finis et la rèsolution par pènalisation-dualitè d'une classe de problèmes de dirichlet non linèaires*, RAIRO R-2 (1975), pp. 41–76.

[23] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Imaging Sci. 2 (2009), pp. 323–343.

[24] J. Gondzio, *Presolve analysis of linear programs prior to applying an interior point method*, Informs J. Comput. 9(1) (1997), pp. 73–91.
[25] J. Gondzio, *Interior point methods 25 years later*, Eur. J. Oper. Res. 218(3) (2012), pp. 587–601.
[26] J. Gondzio, *Matrix-free interior point method*, Comput. Optim. Appl. 51 (2012), pp. 457–480.
[27] N. Gould and P.L. Toint, *Preprocessing for quadratic programming*, Math. Program. 100(1) (2004), pp. 95–132.
[28] B.S. He, H. Yang, and S.L. Wang, *Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities*, J. Optim. Theory Appl. 106(2) (2000), pp. 337–356.
[29] M. Iri and H. Imai, *A multiplicative barrier function method for linear programming*, Algorithmica 1(1–4) (1986), pp. 455–482.
[30] N.K. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica 4 (1984), pp. 373–395.
[31] P.L. Lions and B. Mercier, *Splitting algorithms for the sum of two nonlinear operators*, SIAM J. Numer. Anal. 16 (1979), pp. 964–979.
[32] N. Megiddo, *Progress in Mathematical Programming*, Springer, New York, NY, 1989.
[33] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM J. Optim. 2(4) (1992), pp. 575–601.
[34] C. Mészáros and U.H. Suhl, *Advanced preprocessing techniques for linear and quadratic programming*, OR Spectrum 25(4) (2003), pp. 575–595.
[35] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*, SIAM, Philadelphia, PA, 1994.
[36] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, *Conic optimization via operator splitting and homogeneous self-dual embedding*, J. Optim. Theory Appl. 169(3) (2016), pp. 1042–1068.
[37] B. O'Donoghue, G. Stathopoulos, and S. Boyd, *A splitting method for optimal control*, IEEE Trans. Control Syst. Technol. 21(6) (2013), pp. 2432–2442.
[38] A. Pothen and C.-J. Fan, *Computing the block triangular form of a sparse matrix*, ACM Trans. Math. Softw. 16(4) (1990), pp. 303–324.
[39] W.H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, New York, NY, 2007.
[40] J. Renegar, *A polynomial-time algorithm, based on Newton's method, for linear programming*, Math. Program. 40(1) (1988), pp. 59–93.
[41] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Vol. 82. SIAM, Philadelphia, PA, 2003.
[42] G. Sonnevend, *An "analytical centre" for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming*, in *System Modelling and Optimization. Lecture Notes in Control and Information Sciences, vol 84*, A. Prékopa, J. Szelezsáan, and B. Strazicky, eds., Springer, Berlin, Heidelberg, 1986, pp. 866–875.
[43] J.F. Sturm and S. Zhang, *On a wide region of centers and primal-dual interior point algorithms for linear programming*, Math. Oper. Res. 22 (1997), pp. 408–431.
[44] R.H. Tütüncü, K.-C. Toh, and M.J. Todd, *Solving semidefinite-quadratic-linear programs using SDPT3*, Math. Program. 95(2) (2003), pp. 189–217.
[45] S. Wang and N. Shroff, *A new alternating direction method for linear programming*, in *NIPS*, 2017, pp. 1479–1487.
[46] D.S. Watkins, *Fundamentals of Matrix Computations*, Vol. 64. John Wiley & Sons, New York, NY, 2004.
[47] Z. Wen, D. Goldfarb, and W. Yin, *Alternating direction augmented Lagrangian methods for semidefinite programming*, Math. Programm. Comput. 2(3) (2010), pp. 203–230.
[48] X. Xu, P.-F. Hung, and Y. Ye, *A simplified homogeneous and self-dual linear programming algorithm and its implementation*, Ann. Oper. Res. 62(1) (1996), pp. 151–171.
[49] Z. Xu, M. Figueiredo, and T. Goldstein, *Adaptive ADMM with spectral penalty parameter selection*, in *AISTATS*, 2017, pp. 718–727.
[50] J. Yang and Y. Zhang, *Alternating direction algorithms for l_1-problems in compressive sensing*, SIAM J. Sci. Comput. 33(1) (2011), pp. 250–278.

[51] J. Yang, Y. Zhang, and W. Yin, *A fast alternating direction method for TVL1-L2 signal reconstruction from partial fourier data*, IEEE J. Sel. Topics Signal Process. Special Issue on Compressed Sensing 4(2) (2010), pp. 288–297.
[52] L. Yang, D. Sun, and K-C. Toh, *SDPNAL+: A majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints*, Math. Program. Comput. 7(3) (2015), pp. 331–366.
[53] Y. Ye, *Interior Point Algorithms: Theory and Analysis*, Wiley & Sons, New York, 1997.
[54] Y. Ye and M. Kojima, *Recovering optimal dual solutions in Karmarkar's polynomial time algorithm for linear programming*, Math. Program. 39(3) (1987), pp. 305–317.
[55] Y. Ye, M.J. Todd, and S. Mizuno, *An $\mathcal{O}(\sqrt{nL})$-iteration homogeneous and self-dual linear programming algorithm*, Math. Oper. Res. 19(1) (1994), pp. 53–67.
[56] I.E-H. Yen, K. Zhong, C-J. Hsieh, P.K. Ravikumar, and I.S. Dhillon, *Sparse linear programming via primal and dual augmented coordinate descent*, in *NIPS*, 2015, pp. 2368–2376.
[57] B. Zhou, L. Gao, and Y-H. Dai, *Gradient methods with adaptive step-sizes*, Comput. Optim. Appl. 35(1) (2006), pp. 69–86.