Computational Optimization and Applications (2019) 74:143–176
https://doi.org/10.1007/s10589-019-00103-y

Implementation of interior-point methods for LP based on Krylov subspace iterative solvers with inner-iteration preconditioning

Yiran Cui[1] · Keiichi Morikuni[2] · Takashi Tsuchiya[3] · Ken Hayami[4]

Received: 10 July 2017 / Published online: 6 June 2019

Abstract

We apply novel inner-iteration preconditioned Krylov subspace methods to the interior-point algorithm for linear programming (LP). Inner-iteration preconditioners recently proposed by Morikuni and Hayami enable us to overcome the severe ill-conditioning of linear equations solved in the final phase of interior-point iterations. The Krylov subspace methods do not suffer from rank-deficiency and therefore no preprocessing is necessary even if rows of the constraint matrix are not linearly independent. By means of these methods, a new interior-point recurrence is proposed in order to omit one matrix-vector product at each step. Extensive numerical experiments are conducted over diverse instances of 140 LP problems including the Netlib, QAPLIB, Mittelmann and Atomizer Basis Pursuit collections. The largest problem has 434,580 unknowns. It turns out that our implementation is more robust than the standard public domain solvers SeDuMi (Self-Dual Minimization), SDPT3 (Semidefinite Programming Toh-Todd-Tütüncü) and the LSMR iterative solver in PDCO (Primal-Dual Barrier Method for Convex Objectives) without increasing CPU time. The proposed interior-point method based on iterative solvers succeeds in solving a fairly large number of LP instances from benchmark libraries under the standard stopping criteria. The work also presents a fairly extensive benchmark test for several renowned solvers including direct and iterative solvers.

Keywords Linear programming problems · Interior-point methods · Inner-iteration preconditioning · Krylov subspace methods

The preliminary version of this paper appeared as [16]. K. Morikuni was supported in part by JSPS KAKENHI Grant Number 16K17639. T. Tsuchiya was supported in part by JSPS KAKENHI Grant Number 15H02968. K. Hayami was supported in part by JSPS KAKENHI Grant Numbers 15K04768 and 15H02968.

✉ Yiran Cui
y.cui.12@ucl.ac.uk

Extended author information available on the last page of the article

1 Introduction

Consider the linear programming (LP) problem in the standard primal-dual formulation

$$\min_{\boldsymbol{x}} \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \quad \text{subject to} \quad A\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \geq \boldsymbol{0}, \tag{1a}$$

$$\max_{\boldsymbol{y},\boldsymbol{s}} \boldsymbol{b}^\mathsf{T}\boldsymbol{y} \quad \text{subject to} \quad A^\mathsf{T}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}, \quad \boldsymbol{s} \geq \boldsymbol{0}, \tag{1b}$$

where $A \in \mathbb{R}^{m\times n}$, $m \leq n$, and we assume the existence of an optimal solution. In this paper, we describe an implementation of the interior-point method for LP based on iterative solvers. The main computational task in one iteration of the interior-point method is the solution of a system of linear equations to compute the search direction.

For this task, direct solvers are usually used. But some solvers also employ iterative solvers. Iterative solvers are advantageous when the systems are large and sparse, or even when they are large and dense but the product of the coefficient matrix and a vector can be approximated cheaply, as in [11,65]. The difficulty with iterative solvers is that the linear system becomes notoriously ill-conditioned towards the end of interior-point iterations. One approach is to precondition the mathematically equivalent indefinite augmented system [as in Eq. (5)] as in HOPDM (Higher Order Primal-Dual Method) [29] and also [2,3,6,7,12,26,27,33,58,61]. The other approach is to precondition the equivalent normal equations [as in Eq. (6)] [9,14,28,40,42,44,45,48,60,70].

In this paper, we treat the normal equations and apply novel inner-iteration preconditioned Krylov subspace methods to them. The inner-iteration preconditioners recently proposed by Morikuni and Hayami [54,55] enable us to deal with the severe ill-conditioning of the normal equations. Furthermore, the proposed Krylov subspace methods do not suffer from singularity and therefore no preprocessing is necessary even if A is rank-deficient.

The main contribution of the present paper is that we actually show that the use of the inner-iteration preconditioner enables the efficient interior-point solution of wide-ranging LP problems. We further proposed combining the row-scaling scheme with the inner-outer iteration methods, where the row norm appears in the successive overrelaxation (SOR) inner-iterations, to improve the condition of the system at each interior-point step. The linear systems are solved with a gradually tightened stopping tolerance. We proposed a new recurrence in order to omit one matrix-vector product at each interior-point step. These techniques reduce the CPU time.

Extensive numerical experiments were conducted over diverse instances of 127 LP problems taken from the standard benchmark libraries Netlib, QAPLIB, and Mittelmann collections. The largest problem has 434,580 unknowns. The proposed interior-point method is entirely based on iterative solvers and yet succeeds in solving a fairly large number of standard LP instances from the benchmark libraries with standard stopping criteria. We could not find any other analogous result where this level of LP instances were solved just relying on iterative solvers.

We compared our interior-point LP solvers based on AB-GMRES (right-preconditioned generalized minimal residual method) [37,55], CGNE, and MRNE (preconditioned CG and MINRES applied to the normal equations of the second kind) [13,55] with the following well-known interior-point LP solvers:

1. SeDuMi (Self-Dual Minimization) [66], (public-domain, direct solver),
2. SDPT3 (Semidefinite Programming Toh-Todd-Tütüncü) [68,69], (public-domain, direct solver),
3. PDCO (Primal-Dual Barrier Method for Convex Objectives) [65],
   (a) PDCO-Direct (public-domain, direct solver),
   (b) PDCO-LSMR (public-domain, LSMR iterative solver),
4. MOSEK [57] (commercial, direct solver).

SeDuMi and SDPT3 are solvers for conic linear programming including semidefinite programming (SDP) and second-order cone programming (SOCP). PDCO is for LP and convex quadratic programming (QP) and has options to solve the system of linear equations with Krylov subspace iterative method LSMR in addition to the direct method. MOSEK is considered as one of the state-of-the-art solvers for LP.

As summarized in Table 1, our implementation was able to solve most instances, which is clearly superior to SeDuMi, SDPT3, PDCO-Direct, and PDCO-LSMR with comparable computation time, though it is still slower than MOSEK.

We also tested our solvers on different problems which arise in basis pursuit [11] where the coefficient matrix is much denser than the aforementioned standard benchmark problems.

We emphasize that there are many interesting topics to be further worked out based on this paper. There is still room for improvement regarding the iterative solvers as well as using more sophisticated methods for the interior-point iterations.

In the following, we introduce the interior-point method and review the iterative solvers previously used. We employ an infeasible primal-dual predictor-corrector interior-point method, one of the methods that evolved from the original primal-dual interior-point method [41,49,67,71] incorporating several innovative ideas, e.g., [45,73].

An optimal solution $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s}$ to problem (1) must satisfy the Karush-Kuhn-Tucker (KKT) conditions

$$A^{\mathsf{T}}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}, \tag{2a}$$

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{2b}$$

$$XS\boldsymbol{e} = \boldsymbol{0}, \tag{2c}$$

$$\boldsymbol{x} \geq \boldsymbol{0}, \quad \boldsymbol{s} \geq \boldsymbol{0}, \tag{2d}$$

where $X := \mathrm{diag}(x_1, x_2, \ldots, x_n)$, $S := \mathrm{diag}(s_1, s_2, \ldots, s_n)$, and $\boldsymbol{e} := [1, 1, \ldots, 1]^{\mathsf{T}}$. The complementarity condition (2c) implies that at an optimal solution, one of the elements x_i or s_i must be zero for $i = 1, 2, \ldots, n$.

The following system is obtained by relaxing (2c) to $XS\boldsymbol{e} = \mu\boldsymbol{e}$ with $\mu > 0$:

$$XS\boldsymbol{e} = \mu\boldsymbol{e}, \quad A\boldsymbol{x} = \boldsymbol{b}, \quad A^{\mathsf{T}}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}, \quad \boldsymbol{x} \geq \boldsymbol{0}, \quad \boldsymbol{s} \geq \boldsymbol{0}. \tag{3}$$

The interior-point method solves the problem (1) by generating solutions to (3), with μ decreasing towards zero, so that (2) is satisfied within some tolerance level at the solution point. The search direction at each infeasible interior-point step is obtained by solving the Newton equations

$$\begin{bmatrix} \mathbf{0} & A^\mathsf{T} & I \\ A & \mathbf{0} & \mathbf{0} \\ S & \mathbf{0} & X \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{y} \\ \Delta \boldsymbol{s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_\mathrm{d} \\ \boldsymbol{r}_\mathrm{p} \\ \boldsymbol{r}_\mathrm{c} \end{bmatrix}, \tag{4}$$

where $\boldsymbol{r}_\mathrm{d} := \boldsymbol{c} - A^\mathsf{T}\boldsymbol{y} - \boldsymbol{s} \in \mathbb{R}^n$ is the residual of the dual problem, $\boldsymbol{r}_\mathrm{p} := \boldsymbol{b} - A\boldsymbol{x} \in \mathbb{R}^m$ is the residual of the primal problem, $\boldsymbol{r}_\mathrm{c} := -XS\boldsymbol{e} + \sigma\mu\boldsymbol{e}$, $\mu := \boldsymbol{x}^\mathsf{T}\boldsymbol{s}/n$ is the duality measure, and $\sigma \in [0, 1)$ is the centering parameter, which is dynamically chosen to govern the progress of the interior-point method. Once the kth iterate $(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{s}^{(k)})$ is given and (4) is solved, we define the next iterate as $(\boldsymbol{x}^{(k+1)}, \boldsymbol{y}^{(k+1)}, \boldsymbol{s}^{(k+1)}) := (\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{s}^{(k)}) + \alpha(\Delta\boldsymbol{x}, \Delta\boldsymbol{y}, \Delta\boldsymbol{s})$, where $\alpha \in (0, 1]$ is a step length to ensure the positivity of $\boldsymbol{x}$ and $\boldsymbol{s}$, and then reduce μ to $\sigma\mu$ before solving (4) again.

At each iteration, the solution of (4) dominates the total CPU time. The choice of linear solvers depends on the way of arranging the matrix of (4). Aside from solving the $(m + 2n) \times (m + 2n)$ system (4), one can solve its reduced equivalent form of size $(m + n) \times (m + n)$

$$\begin{bmatrix} A & 0 \\ S & -XA^\mathsf{T} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_\mathrm{p} \\ \boldsymbol{r}_\mathrm{c} - X\boldsymbol{r}_\mathrm{d} \end{bmatrix}, \tag{5}$$

or a more condensed equivalent form of size $m \times m$

$$AXS^{-1}A^\mathsf{T}\Delta\boldsymbol{y} = \boldsymbol{r}_\mathrm{p} - AS^{-1}(\boldsymbol{r}_\mathrm{c} - X\boldsymbol{r}_\mathrm{d}), \tag{6}$$

both of which are obtained by performing block Gaussian eliminations on (4). We are concerned in this paper with solving the third equivalent form (6).

It is known that the matrix of (6) is semidefinite when any of the following cases is encountered. First, when A is rank-deficient, system (6) is singular. There exist presolving techniques that address this problem, see, e.g., [4,31]. However, they do not guarantee to detect all dependent rows in A. Second, in late interior-point iterations, the diagonal matrix XS^{-1} has very tiny and very large diagonal values as a result of convergence. Thus, the matrix may become positive semidefinite. In particular, the situation becomes severe when primal degeneracy occurs at an optimal solution. One can refer to [34,74] for more detailed explanations.

Thus, when direct methods such as Cholesky decomposition are applied to (6), some diagonal pivots encountered during decomposition can be zero or negative, causing the algorithm to break down. Many direct methods adopt a strategy of replacing the problematic pivot with a very large number. See, e.g., [74] for the Cholesky-Infinity factorization, which is specially designed to solve (6) when it is positive semidefinite but not definite. Numerical experience [1,5,17,25,43,44,72] indicates that direct methods provide sufficiently accurate solutions for interior-point methods to converge regardless of the ill-conditioning of the matrix. However, as the LP problems become larger, the significant fill-ins in decompositions make direct methods prohibitively expensive. It is stated in [32] that the fill-ins are observed even for very sparse matrices. Moreover, the matrix can be dense, as in QP in support vector machine training [24] or linear programming in basis pursuit [11], and even when A is sparse, $AXS^{-1}A^{\mathsf{T}}$ can be dense or have a pattern of nonzero elements that renders the system difficult for direct methods. The expensive solution of the KKT systems is a usual disadvantage of second-order methods including interior-point methods.

These drawbacks of direct methods and the progress in preconditioning techniques motivate researchers to develop stable iterative methods for solving (6) or alternatively (5). The major problem is that as the interior-point iterations proceed, the condition number of the term XS^{-1} increases, making the system of linear equations intractable. One way to deal with this is to employ suitable preconditioners. Since our main focus is on solving (6), we explain preconditioners for (6) in detail in the following. We mention [2,3,6,7,12,26,27,58,61] as literature related to preconditioners for (5).

For the iterative solution of (6), the conjugate gradient (CG) method [38] has been applied with diagonal scaling preconditioners [9,42,60] or incomplete Cholesky preconditioners [12,40,45,48]. LSQR with a preconditioner was used in [28]. A matrix-free method of using CG for least squares (CGLS) preconditioned by a partial Cholesky decomposition was proposed in [33]. In [14], a preconditioner based on Greville's method [15] for generalized minimal residual (GMRES) method was applied. Suitable preconditioners were also introduced for particular fields such as the minimum-cost network flow problem in [39,50,51,62]. One may refer to [18] for a review on the application of numerical linear algebra algorithms to the solutions of KKT systems in the optimization context.

In this paper, we propose to solve (6) using Krylov subspace methods preconditioned by stationary inner-iterations recently proposed for least squares problems in [37,54,55]. In Sect. 2, we briefly describe the framework of Mehrotra's predictor-corrector interior-point algorithm we implemented and the normal equations arising from this algorithm. In Sect. 3, we specify the application of our method to the normal equations. In Sect. 4, we present numerical results comparing our method with a modified sparse Cholesky method, three direct solvers in CVX, a major public package for specifying and solving convex programs [35,36], and direct and iterative solvers in PDCO [65]. The testing problems include the typical LP problems from the Netlib, Qaplib and Mittelmann collections in [20] and basis pursuit problems from the package Atomizer [10]. In Sect. 5, we conclude the paper.

Throughout, we use bold lower case letters for column vectors. We denote quantities related to the kth interior-point iteration by using a superscript with round brackets, e.g., $\boldsymbol{x}^{(k)}$, the kth iteration of Krylov subspace methods by using a subscript without brackets, e.g., $\boldsymbol{x}_k$, and the kth inner iteration by using a superscript with angle brackets, e.g., $\boldsymbol{x}^{\langle k\rangle}$. $\mathcal{R}(A)$ denotes the range space of a matrix A. $\kappa(A)$ denotes the condition number $\kappa(A) = \sigma_1(A)/\sigma_r(A)$, where $\sigma_1(A)$ and $\sigma_r(A)$ denote the maximum and minimum nonzero singular values of A, respectively. $\mathcal{K}_k(A, \boldsymbol{b}) = \operatorname{span}\{\boldsymbol{b}, A\boldsymbol{b}, \dots, A^{k-1}\boldsymbol{b}\}$ denotes the Krylov subspace of order k.

2 Interior-point algorithm and the normal equations

We implement an infeasible version of Mehrotra's predictor-corrector method [46], which has been established as a standard in this area [43,44,47,71]. Note that our method can be applied to other interior-point methods (see, e.g., [71] for more interior-point methods) whose directions are computed via the normal equations (6).

2.1 Mehrotra's predictor-corrector algorithm

In this method, the centering parameter σ is determined by dividing each step into two stages.

In the first stage, we solve for the affine direction $(\Delta \boldsymbol{x}_{\mathrm{af}}, \Delta \boldsymbol{y}_{\mathrm{af}}, \Delta \boldsymbol{s}_{\mathrm{af}})$

$$\begin{bmatrix} \mathbf{0} & A^\mathsf{T} & I \\ A & \mathbf{0} & \mathbf{0} \\ S & \mathbf{0} & X \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x}_{\mathrm{af}} \\ \Delta \boldsymbol{y}_{\mathrm{af}} \\ \Delta \boldsymbol{s}_{\mathrm{af}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_{\mathrm{d}} \\ \boldsymbol{r}_{\mathrm{p}} \\ -XS\boldsymbol{e} \end{bmatrix}, \tag{7}$$

and measure its progress in reducing μ. If the affine direction makes large enough progress without violating the nonnegative boundary (2d), then σ is assigned a small value. Otherwise, σ is assigned a larger value to steer the iterate to be more centered in the strictly positive region.

In the second stage, we solve for the corrector direction $(\Delta \boldsymbol{x}_{\mathrm{cc}}, \Delta \boldsymbol{y}_{\mathrm{cc}}, \Delta \boldsymbol{s}_{\mathrm{cc}})$

$$\begin{bmatrix} \mathbf{0} & A^\mathsf{T} & I \\ A & \mathbf{0} & \mathbf{0} \\ S & \mathbf{0} & X \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x}_{\mathrm{cc}} \\ \Delta \boldsymbol{y}_{\mathrm{cc}} \\ \Delta \boldsymbol{s}_{\mathrm{cc}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ -\Delta X_{\mathrm{af}} \Delta S_{\mathrm{af}} \boldsymbol{e} + \sigma \mu_{\mathrm{af}} \boldsymbol{e} \end{bmatrix}, \tag{8}$$

where $\Delta X_{\mathrm{af}} = \mathrm{diag}(\Delta \boldsymbol{x}_{\mathrm{af}})$, $\Delta S_{\mathrm{af}} = \mathrm{diag}(\Delta \boldsymbol{s}_{\mathrm{af}})$ and σ is determined according to the solution in the first stage. Finally, we update the current iterate along the linear combination of the two directions.

In our implementation of the interior-point method, we adopt Mehrotra's predictor-corrector algorithm as follows.

In line 5 in Algorithm 1, the step lengths $\alpha_{\mathrm{p}},\ \alpha_{\mathrm{d}}$ are computed by

$$\alpha_{\mathrm{p}} = \min\left(1, \eta \min_{i:\Delta x_i < 0}\left(-\frac{x_i}{\Delta x_i}\right)\right), \quad \alpha_{\mathrm{d}} = \min\left(1, \eta \min_{i:\Delta s_i < 0}\left(-\frac{s_i}{\Delta s_i}\right)\right), \tag{9}$$

where $(\Delta \boldsymbol{x}, \Delta \boldsymbol{s}) = (\Delta \boldsymbol{x}_{\mathrm{af}}, \Delta \boldsymbol{s}_{\mathrm{af}})$, $\eta \in [0.9, 1)$.

In line 9, the quantity μ_{af} is computed by

$$\mu_{\mathrm{af}} = (\boldsymbol{x}^{(k)} + \alpha_{\mathrm{p}} \Delta \boldsymbol{x}_{\mathrm{af}})^\mathsf{T} (\boldsymbol{s}^{(k)} + \alpha_{\mathrm{d}} \Delta \boldsymbol{s}_{\mathrm{af}})/n.$$

In the same line, the parameter σ is chosen as $\sigma = \min(0.208, (\mu_{\mathrm{af}}/\mu^{(k)})^2)$ in the early phase of the interior-point iterations. The value 0.208 and the range [0.9, 1) for η are adopted by the LIPSOL package [74]. In the late phase of the interior-point iterations, σ is chosen as approximately 10 times the error measure $\Gamma^{(k)}$ which is defined as:

Algorithm 1 Mehrotra's predictor-corrector algorithm.

```
1: Given $(\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}, \boldsymbol{s}^{(0)})$ with $(\boldsymbol{x}^{(0)}, \boldsymbol{s}^{(0)}) > \boldsymbol{0}$.
2: for $k = 0, 1, 2, \ldots$ until convergence, do
3:    $\mu^{(k)} := \boldsymbol{x}^{(k)\mathsf{T}} \boldsymbol{s}^{(k)}/n$ {the predictor stage}
4:    Solve (7) for the affine direction $(\Delta \boldsymbol{x}_{\mathrm{af}}, \Delta \boldsymbol{y}_{\mathrm{af}}, \Delta \boldsymbol{s}_{\mathrm{af}})$.
5:    Compute $\alpha_{\mathrm{p}},\ \alpha_{\mathrm{d}}$.
6:    if min $(\alpha_{\mathrm{p}},\ \alpha_{\mathrm{d}}) \geq 1$ then
7:       $\sigma := 0, \left(\Delta \boldsymbol{x}^{(k)}, \Delta \boldsymbol{y}^{(k)}, \Delta \boldsymbol{s}^{(k)}\right) := \left(\Delta \boldsymbol{x}_{\mathrm{af}}, \Delta \boldsymbol{y}_{\mathrm{af}}, \Delta \boldsymbol{s}_{\mathrm{af}}\right)$
8:    else
9:       Set $\mu_{\mathrm{af}}$ and $\sigma :=$ a small value, e.g., 0.208. {the corrector stage}
10:      Solve (8) for the corrector direction $(\Delta \boldsymbol{x}_{\mathrm{cc}}, \Delta \boldsymbol{y}_{\mathrm{cc}}, \Delta \boldsymbol{s}_{\mathrm{cc}})$.
11:      $\left(\Delta \boldsymbol{x}^{(k)}, \Delta \boldsymbol{y}^{(k)}, \Delta \boldsymbol{s}^{(k)}\right) := (\Delta \boldsymbol{x}_{\mathrm{af}}, \Delta \boldsymbol{y}_{\mathrm{af}}, \Delta \boldsymbol{s}_{\mathrm{af}}) + (\Delta \boldsymbol{x}_{\mathrm{cc}}, \Delta \boldsymbol{y}_{\mathrm{cc}}, \Delta \boldsymbol{s}_{\mathrm{cc}})$
12:   end if
13:   Compute $\hat{\alpha}_{\mathrm{p}},\ \hat{\alpha}_{\mathrm{d}}$.
14:   $\boldsymbol{x}^{(k+1)} := \boldsymbol{x}^{(k)} + \hat{\alpha}_{\mathrm{p}} \Delta \boldsymbol{x}^{(k)}, \left(\boldsymbol{y}^{(k+1)}, \boldsymbol{s}^{(k+1)}\right) := \left(\boldsymbol{y}^{(k)}, \boldsymbol{s}^{(k)}\right) + \hat{\alpha}_{\mathrm{d}} \left(\Delta \boldsymbol{y}^{(k)}, \Delta \boldsymbol{s}^{(k)}\right)$
15: end for
```

$$\Gamma^{(k)} := \max \left\{ \mu^{(k)}, \frac{\|\boldsymbol{b} - A\boldsymbol{x}^{(k)}\|_2}{\max\{\|\boldsymbol{b}\|_2, 1\}}, \frac{\|\boldsymbol{c} - \boldsymbol{s}^{(k)} - A^{\mathsf{T}}\boldsymbol{y}^{(k)}\|_2}{\max\{\|\boldsymbol{c}\|_2, 1\}} \right\}. \tag{10}$$

Here the distinction between *early* and *late* phases is when $\Gamma^{(k)}$ is more or less than 10^{-3}.

In line 13, we first compute trial step lengths $\alpha_{\mathrm{p}}, \alpha_{\mathrm{d}}$ using equations (9) with $(\Delta \boldsymbol{x}, \Delta \boldsymbol{s}) = (\Delta \boldsymbol{x}^{(k)}, \Delta \boldsymbol{s}^{(k)})$. Then, we gradually reduce $\alpha_{\mathrm{p}}, \alpha_{\mathrm{d}}$ to find the largest step lengths that can ensure the centrality of the updated iterates, i.e., to find the maximum $\hat{\alpha}_{\mathrm{p}},\ \hat{\alpha}_{\mathrm{d}}$ that satisfy

$$\min_i (x_i + \hat{\alpha}_{\mathrm{p}} \Delta x_i)(s_i + \hat{\alpha}_{\mathrm{d}} \Delta s_i) \geq \phi (\boldsymbol{x} + \hat{\alpha}_{\mathrm{p}} \Delta \boldsymbol{x})^{\mathsf{T}} (\boldsymbol{s} + \hat{\alpha}_{\mathrm{d}} \Delta \boldsymbol{s})/n,$$

where ϕ is typically chosen as 10^{-5}.

2.2 The normal equations in the interior-point algorithm

We consider modifying Algorithm 1 so that it is not necessary to update $\boldsymbol{y}^{(k)}$. Since we assume the existence of an optimal solution to problem (1), we have $\boldsymbol{b} \in \mathcal{R}(A)$. Let $D := S^{-1/2} X^{1/2}$ and $\mathcal{A} := AD$. Problem (6) with $\Delta \boldsymbol{w} = \mathcal{A}^{\mathsf{T}} \Delta \boldsymbol{y}$ (the normal equations of the second kind) is equivalent to

$$\min \|\Delta \boldsymbol{w}\|_2 \quad \text{subject to} \quad \mathcal{A} \Delta \boldsymbol{w} = \boldsymbol{f}, \tag{11}$$

where $\boldsymbol{f} := \boldsymbol{r}_{\mathrm{p}} - AS^{-1}(\boldsymbol{r}_{\mathrm{c}} - X\boldsymbol{r}_{\mathrm{d}})$.

In the predictor stage, the problem (7) is equivalent to first solving (11) for $\Delta \boldsymbol{w}_{\mathrm{af}}$ with $\Delta \boldsymbol{w} = \Delta \boldsymbol{w}_{\mathrm{af}},\ \boldsymbol{f} = \boldsymbol{f}_{\mathrm{af}} := \boldsymbol{b} + AS^{-1}X\boldsymbol{r}_{\mathrm{d}}$, and then updating the remaining unknowns by

$$\Delta \boldsymbol{s}_{\mathrm{af}} = \boldsymbol{r}_{\mathrm{d}} - D^{-1}\Delta \boldsymbol{w}_{\mathrm{af}}, \tag{12a}$$

$$\Delta \boldsymbol{x}_{\mathrm{af}} = -D^{2}\Delta \boldsymbol{s}_{\mathrm{af}} - \boldsymbol{x}. \tag{12b}$$

In the corrector stage, the problem (8) is equivalent to first solving (11) for $\Delta \boldsymbol{w}_{\mathrm{cc}}$ with $\Delta \boldsymbol{w} = \Delta \boldsymbol{w}_{\mathrm{cc}}$, $\boldsymbol{f} = \boldsymbol{f}_{\mathrm{cc}} := AS^{-1}\Delta X_{\mathrm{af}}\Delta S_{\mathrm{af}}\boldsymbol{e} - \sigma\mu AS^{-1}\boldsymbol{e}$, and then updating the remaining unknowns by

$$\Delta \boldsymbol{s}_{\mathrm{cc}} = -D^{-1}\Delta \boldsymbol{w}_{\mathrm{cc}}, \tag{13a}$$

$$\Delta \boldsymbol{x}_{\mathrm{cc}} = -D^{2}\Delta \boldsymbol{s}_{\mathrm{cc}} - S^{-1}\Delta X_{\mathrm{af}}\Delta \boldsymbol{s}_{\mathrm{af}} + \sigma\mu S^{-1}\boldsymbol{e}. \tag{13b}$$

By solving (11) for $\Delta \boldsymbol{w}$ instead of solving (6) for $\Delta \boldsymbol{y}$, we can compute $\Delta \boldsymbol{s}_{\mathrm{af}}$, $\Delta \boldsymbol{x}_{\mathrm{af}}$, $\Delta \boldsymbol{s}_{\mathrm{cc}}$, and $\Delta \boldsymbol{x}_{\mathrm{cc}}$ and can save 1MV in (12a) and another in (13a) if a predictor step is performed per interior-point iteration. Here, MV denotes the computational cost required for one matrix-vector multiplication.

Remark 1 For solving an interior-point step from the condensed step equation (6) using a suited Krylov subspace method, updating $(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s})$ rather than $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{s})$ can save 1MV each interior-point iteration.

Note that in the predictor and corrector stages, problem (11) has the same matrix but different right-hand sides. We introduce methods for solving it in the next section.

3 Application of inner-iteration preconditioned Krylov subspace methods

In lines 4 and 10 of Algorithm 1, the linear system (11) needs to be solved, with its matrix becoming increasingly ill-conditioned as the interior-point iterations proceed. In this section, we focus on applying inner-iteration preconditioned Krylov subspace methods to (11) because they are advantageous in dealing with ill-conditioned sparse matrices. The methods to be discussed are the preconditioned CG and MINRES methods [38,59] applied to the normal equations of the second kind ((P)CGNE and (P)MRNE, respectively) [13,55], and the right-preconditioned generalized minimal residual method (AB-GMRES) [37,55].

Consider solving linear system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} \in \mathbf{R}^{n\times n}$. First, the conjugate gradient (CG) method [38] is an iterative method for such problems when $\mathbf{A}$ is a symmetric and positive (semi)definite matrix and $\mathbf{b} \in \mathcal{R}(\mathbf{A})$. CG starts with an initial approximate solution $\mathbf{x}_0 \in \mathbb{R}^n$ and determines the kth iterate $\mathbf{x}_k \in \mathbb{R}^n$ by minimizing $\|\mathbf{x}_k - \mathbf{x}_*\|_{\mathbf{A}}^2$ over the space $\mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$, where $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$, $\mathbf{x}_*$ is a solution of $\mathbf{Ax} = \mathbf{b}$, and $\|\mathbf{x}_k - \mathbf{x}_*\|_{\mathbf{A}}^2 := (\mathbf{x}_k - \mathbf{x}_*)^{\mathsf{T}}\mathbf{A}(\mathbf{x}_k - \mathbf{x}_*)$.

MINRES [59] is another iterative method for solving such problems but only requires $\mathbf{A}$ to be symmetric. MINRES with $\mathbf{x}_0$ determines the kth iterate $\mathbf{x}_k$ by minimizing $\|\mathbf{b} - \mathbf{Ax}\|_2$ over the same space as CG.

Third, the generalized minimal residual (GMRES) method [64] only requires $\mathbf{A}$ to be square. GMRES with $\mathbf{x}_0$ determines the kth iterate $\mathbf{x}_k$ by minimizing $\|\mathbf{b} - \mathbf{Ax}\|_2$ over $\boldsymbol{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$.

3.1 Application of inner-iteration preconditioned CGNE and MRNE methods

We first introduce CGNE and MRNE. Let $\mathbf{A} = \mathcal{A}\mathcal{A}^{\mathsf{T}}$, $\mathbf{x} = \Delta \boldsymbol{y}_{\mathrm{af}}$, $\mathbf{b} = \boldsymbol{f}_{\mathrm{af}}$, and $\Delta \boldsymbol{w}_{\mathrm{af}} = \mathcal{A}^{\mathsf{T}} \Delta \boldsymbol{y}_{\mathrm{af}}$ for the predictor stage, and similarly, let $\mathbf{A} = \mathcal{A}\mathcal{A}^{\mathsf{T}}$, $\mathbf{x} = \Delta \boldsymbol{y}_{\mathrm{cc}}$, $\mathbf{b} = \boldsymbol{f}_{\mathrm{cc}}$, and $\Delta \boldsymbol{w}_{\mathrm{cc}} = \mathcal{A}^{\mathsf{T}} \Delta \boldsymbol{y}_{\mathrm{cc}}$ for the corrector stage. CG and MINRES applied to systems $\mathbf{A}\mathbf{x} = \mathbf{b}$ are CGNE and MRNE, respectively. With these settings, let the initial solution $\Delta \boldsymbol{w}_0 \in \mathcal{R}(\mathcal{A}^{\mathsf{T}})$ in both stages, and denote the initial residual by $\boldsymbol{g}_0 := \boldsymbol{f} - \mathcal{A}\Delta \boldsymbol{w}_0$. CGNE and MRNE can solve (11) without forming $\mathcal{A}\mathcal{A}^{\mathsf{T}}$ explicitly.

Concretely, CGNE gives the kth iterate $\Delta \boldsymbol{w}_k$ such that $\|\Delta \boldsymbol{w}_k - \Delta \boldsymbol{w}_*\|_2 = \min_{\Delta \boldsymbol{w} \in \Delta \boldsymbol{w}_0 + \mathcal{K}_k(\mathcal{A}^{\mathsf{T}}\mathcal{A}, \mathcal{A}^{\mathsf{T}}\boldsymbol{g}_0)} \|\Delta \boldsymbol{w} - \Delta \boldsymbol{w}_*\|_2$, where $\Delta \boldsymbol{w}_*$ is the minimum-norm solution of $\mathcal{A}\Delta \boldsymbol{w} = \boldsymbol{f}$ for $\Delta \boldsymbol{w}_0 \in \mathcal{R}(\mathcal{A}^{\mathsf{T}})$ and $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$. MRNE gives the kth iterate $\Delta \boldsymbol{w}_k$ such that $\|\boldsymbol{f} - \mathcal{A}\Delta \boldsymbol{w}_k\|_2 = \min_{\Delta \boldsymbol{w} \in \Delta \boldsymbol{w}_0 + \mathcal{K}_k(\mathcal{A}^{\mathsf{T}}\mathcal{A}, \mathcal{A}^{\mathsf{T}}\boldsymbol{g}_0)} \|\boldsymbol{f} - \mathcal{A}\Delta \boldsymbol{w}\|_2$.

We use inner-iteration preconditioning for CGNE and MRNE methods. The following is a brief summary of the part of [55] where the inner-outer iteration method is analyzed. We give the expressions for the inner-iteration preconditioning and preconditioned matrices to state the conditions under which the former is SPD. Let M be a symmetric nonsingular splitting matrix of $\mathcal{A}\mathcal{A}^{\mathsf{T}}$ such that $\mathcal{A}\mathcal{A}^{\mathsf{T}} = M - N$. Denote the inner-iteration matrix by $H = M^{-1}N$. The inner-iteration preconditioning and preconditioned matrices are $C^{\langle \ell \rangle} = \sum_{i=0}^{\ell-1} H^i M^{-1}$ and $\mathcal{A}\mathcal{A}^{\mathsf{T}} C^{\langle \ell \rangle} = M \sum_{i=0}^{\ell-1} (I - H) H^i M^{-1} = M(I - H^{\ell})M^{-1}$, respectively. If $C^{\langle \ell \rangle}$ is nonsingular, then $\mathcal{A}\mathcal{A}^{\mathsf{T}} C^{\langle \ell \rangle} \boldsymbol{u} = \boldsymbol{f}$, $\boldsymbol{z} = C^{\langle \ell \rangle}\boldsymbol{u}$ is equivalent to $\mathcal{A}\mathcal{A}^{\mathsf{T}}\boldsymbol{z} = \boldsymbol{f}$ for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$. For ℓ odd, $C^{\langle \ell \rangle}$ is symmetric and positive definite (SPD) if and only if the inner-iteration M is SPD; for ℓ even, $C^{\langle \ell \rangle}$ is SPD if and only if $M + N$ is SPD [52,53, Theorem 2.8]. We give Algorithms 2, 3 for CGNE and MRNE preconditioned by inner iterations [55, Algorithms E.3, E.4].

Algorithm 2 CGNE method preconditioned by inner iterations.

```
1: Let $\Delta \boldsymbol{w}_0$ be the initial approximate solution, and $\boldsymbol{g}_0 := \boldsymbol{f} - \mathcal{A}\Delta \boldsymbol{w}_0$.
2: Apply $\ell$ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^{\mathsf{T}}\boldsymbol{z} = \boldsymbol{g}_0$, $\boldsymbol{u} = \mathcal{A}^{\mathsf{T}}\boldsymbol{z}$ to obtain $\boldsymbol{z}_0 := \mathcal{C}^{\langle \ell \rangle}\boldsymbol{g}_0$ and
   $\boldsymbol{u}_0 := \mathcal{A}^{\mathsf{T}}\boldsymbol{z}_0$.
3: $\boldsymbol{q}_0 := \boldsymbol{u}_0$, $\gamma_0 := (\boldsymbol{g}_0, \boldsymbol{z}_0)$
4: for $k = 0, 1, 2, \ldots$ until convergence, do
5:    $\alpha_k := \gamma_k / (\boldsymbol{q}_k, \boldsymbol{q}_k)$,  $\Delta \boldsymbol{w}_{k+1} := \Delta \boldsymbol{w}_k + \alpha \boldsymbol{q}_k$,  $\boldsymbol{g}_{k+1} := \boldsymbol{g}_k - \alpha_k \mathcal{A}\boldsymbol{q}_k$
6:    Apply $\ell$ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^{\mathsf{T}}\boldsymbol{z} = \boldsymbol{g}_{k+1}$ to obtain $\boldsymbol{z}_{k+1} := \mathcal{C}^{\langle \ell \rangle}\boldsymbol{g}_{k+1}$ and
      $\boldsymbol{u}_{k+1} := \mathcal{A}^{\mathsf{T}}\boldsymbol{z}_{k+1}$.
7:    $\gamma_{k+1} := (\boldsymbol{g}_{k+1}, \boldsymbol{z}_{k+1})$,  $\beta_k := \gamma_{k+1}/\gamma_k$,  $\boldsymbol{q}_{k+1} := \boldsymbol{u}_{k+1} + \beta_k \boldsymbol{q}_k$
8: end for
```

3.2 Application of inner-iteration preconditioned AB-GMRES method

Next, we introduce AB-GMRES. GMRES can solve a square linear system transformed from the rectangular system $\mathcal{A}\Delta \boldsymbol{w}_{\mathrm{af}} = \boldsymbol{f}_{\mathrm{af}}$ in the predictor stage and $\mathcal{A}\Delta \boldsymbol{w}_{\mathrm{cc}} = \boldsymbol{f}_{\mathrm{cc}}$ in the corrector stage by using a rectangular right-preconditioning matrix that does not necessarily have to be $\mathcal{A}^{\mathsf{T}}$. Let $\mathcal{B} \in \mathbb{R}^{n \times m}$ be a preconditioning matrix for $\mathcal{A}$. Then, AB-GMRES corresponds to GMRES [64] applied to

Algorithm 3 MRNE method preconditioned by inner iterations.

```
1: Let $\Delta\boldsymbol{w}_0$ be the initial approximate solution, and $\boldsymbol{g}_0 := \boldsymbol{f} - \mathcal{A}\Delta\boldsymbol{w}_0$.
2: Apply $\ell$ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{u} = \boldsymbol{g}_0,\ \boldsymbol{q} = \mathcal{A}^\mathsf{T}\boldsymbol{u}$ to obtain $\boldsymbol{q}_0 := \mathcal{A}^\mathsf{T}\mathcal{C}^{\langle\ell\rangle}\boldsymbol{g}_0$.
3: $\boldsymbol{p}_0 := \boldsymbol{q}_0,\ \gamma_0 := \|\boldsymbol{q}_0\|_2^2$
4: for $k = 1, 2, \dots$ until convergence, do
5:    $\boldsymbol{t}_k := \mathcal{A}\boldsymbol{p}_k$
6:    Apply $\ell$ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{u} = \boldsymbol{t}_k,\ \boldsymbol{v} = \mathcal{A}^\mathsf{T}\boldsymbol{u}$ to obtain $\boldsymbol{v}_k := \mathcal{A}^\mathsf{T}\mathcal{C}^{\langle\ell\rangle}\boldsymbol{t}_k$.
7:    $\alpha_k := \gamma_k/(\boldsymbol{v}_k, \boldsymbol{p}_k),\ \Delta\boldsymbol{w}_k := \Delta\boldsymbol{w}_k + \alpha_k\boldsymbol{p}_k,\ \boldsymbol{g}_{k+1} := \boldsymbol{g}_k - \alpha_k\boldsymbol{t}_k, \boldsymbol{q}_{k+1} := \boldsymbol{q}_k - \alpha_k\boldsymbol{v}_k$
8:    $\gamma_k := \|\boldsymbol{q}_{k+1}\|_2^2,\ \beta_k := \gamma_{k+1}/\gamma_k,\ \boldsymbol{p}_{k+1} := \boldsymbol{q}_k + \beta_k\boldsymbol{p}_k$
9: end for
```

$$\mathcal{A}\mathcal{B}\boldsymbol{z} = \boldsymbol{f}, \quad \Delta\boldsymbol{w} = \mathcal{B}\boldsymbol{z},$$

which is equivalent to the minimum-norm solution to the problem (11), for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ if $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^\mathsf{T})$ [55, Theorem 5.2], where $\Delta\boldsymbol{w} = \Delta\boldsymbol{w}_{\mathrm{af}}$ or $\Delta\boldsymbol{w}_{\mathrm{cc}}$, $\boldsymbol{f} = \boldsymbol{f}_{\mathrm{af}}$ or $\boldsymbol{f}_{\mathrm{cc}}$, respectively. AB-GMRES gives the kth iterate $\Delta\boldsymbol{w}_k = \mathcal{B}\boldsymbol{z}_k$ such that $\boldsymbol{z}_k = \mathrm{argmin}_{\boldsymbol{z}\in\boldsymbol{z}_0+\mathcal{K}_k(\mathcal{A}\mathcal{B},\boldsymbol{g}_0)} \|\boldsymbol{f} - \mathcal{A}\mathcal{B}\boldsymbol{z}\|_2$, where $\boldsymbol{z}_0$ is the initial iterate and $\boldsymbol{g}_0 = \boldsymbol{f} - \mathcal{A}\mathcal{B}\boldsymbol{z}_0$.

Specifically, we apply AB-GMRES preconditioned by inner iterations [54,55] to (11). This method was shown to outperform previous methods on ill-conditioned and rank-deficient problems. We give expressions for the inner-iteration preconditioning and preconditioned matrices. Let M be a nonsingular splitting matrix such that $\mathcal{A}\mathcal{A}^\mathsf{T} = M - N$. Denote the inner-iteration matrix by $H = M^{-1}N$. With $C^{\langle\ell\rangle} = \sum_{i=0}^{\ell-1} H^i M^{-1}$, the inner-iteration preconditioning and preconditioned matrices are $\mathcal{B}^{\langle\ell\rangle} = \mathcal{A}^\mathsf{T}C^{\langle\ell\rangle}$ and $\mathcal{A}\mathcal{B}^{\langle\ell\rangle} = \sum_{i=0}^{\ell-1}(I - H)H^i = M(I - H^\ell)M^{-1}$, respectively. If the inner-iteration matrix H is semiconvergent, i.e., $\lim_{i\to\infty} H^i$ exists, then AB-GMRES preconditioned by the inner-iterations determines the minimum-norm solution of $\mathcal{A}\Delta\boldsymbol{w} = \boldsymbol{f}$ without breakdown for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ and for all $\Delta\boldsymbol{w}_0 \in \mathcal{R}(\mathcal{A}^\mathsf{T})$ [55, Theorem 5.5]. The inner-iteration preconditioning matrix $\mathcal{B}^{\langle\ell\rangle}$ works on $\mathcal{A}$ in AB-GMRES as in Algorithm 4 [55, Algorithm 5.1].

Algorithm 4 AB-GMRES method preconditioned by inner iterations.

```
1: Let $\Delta\boldsymbol{w}_0 \in \mathbb{R}^n$ be the initial approximate solution, and $\boldsymbol{g}_0 := \boldsymbol{f} - \mathcal{A}\Delta\boldsymbol{w}_0$.
2: $\beta := \|\boldsymbol{g}_0\|_2,\ \boldsymbol{v}_1 := \boldsymbol{r}_0/\beta$
3: for $k = 1, 2, \dots$ until convergence, do
4:    Apply $\ell$ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{p} = \boldsymbol{v}_k,\ \boldsymbol{z} = \mathcal{A}^\mathsf{T}\boldsymbol{p}$ to obtain $\boldsymbol{z}_k := \mathcal{B}^{\langle\ell\rangle}\boldsymbol{v}_k$.
5:    $\boldsymbol{u}_k := \mathcal{A}\boldsymbol{z}_k$
6:    for $i = 1, 2, \dots, k$, do
7:       $h_{i,k} := (\boldsymbol{u}_k, \boldsymbol{v}_i),\ \boldsymbol{u}_k := \boldsymbol{u}_k - h_{i,k}\boldsymbol{v}_i$
8:    end for
9:    $h_{k+1,k} := \|\boldsymbol{u}_k\|_2,\ \boldsymbol{v}_{k+1} := \boldsymbol{u}_k/h_{k+1,k}$
10: end for
11: $\boldsymbol{p}_k := \arg\min_{\boldsymbol{p}\in R^k} \|\beta\boldsymbol{e}_1 - \bar{H}_k\boldsymbol{p}\|_2,\ \boldsymbol{q}_k = [\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_k]\boldsymbol{p}_k$
12: Apply $\ell$ steps of a stationary iterative method to $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{p} = \boldsymbol{q}_k,\ \boldsymbol{z} = \mathcal{A}^\mathsf{T}\boldsymbol{p}$ to obtain $\boldsymbol{z}' := \mathcal{B}^{\langle\ell\rangle}\boldsymbol{q}_k$.
13: $\Delta\boldsymbol{w}_k := \Delta\boldsymbol{w}_0 + \boldsymbol{z}'$
```

Here, $\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_k$ are orthonormal, $\boldsymbol{e}_1$ is the first column of the identity matrix, and $\bar{H}_k = \{h_{i,j}\} \in \mathbb{R}^{(k+1)\times k}$.

Note that the left-preconditioned generalized minimal residual method (BA-GMRES) [37,54,55] can be applied to solve the corrector stage problem, which can be written as the normal equations of the first kind

$$\mathcal{A}\mathcal{A}^{\mathsf{T}}\Delta \boldsymbol{y}_{\mathrm{cc}} = \mathcal{A}(SX)^{-1/2}\left(\Delta X_{\mathrm{af}}\Delta S_{\mathrm{af}}\boldsymbol{e} - \sigma\mu\boldsymbol{e}\right),$$

or equivalently

$$\min_{\Delta \boldsymbol{y}_{\mathrm{cc}}} \|\mathcal{A}^{\mathsf{T}}\Delta \boldsymbol{y}_{\mathrm{cc}} - (SX)^{-1/2}\left(\Delta X_{\mathrm{af}}\Delta S_{\mathrm{af}}\boldsymbol{e} - \sigma\mu\boldsymbol{e}\right)\|_2. \tag{14}$$

In fact, this formulation was adopted in [32] and solved by the CGLS method preconditioned by partial Cholesky decomposition that works in m-dimen-sional space. The BA-GMRES also works in m-dimensional space.

The advantage of the inner-iteration preconditioning methods is that we can avoid explicitly computing and storing the preconditioning matrices for $\mathcal{A}$ in (11). We present efficient algorithms for specific inner iterations in the next section.

3.3 SSOR inner iterations for preconditioning the CGNE and MRNE methods

The inner-iteration preconditioned CGNE and MRNE methods require a symmetric preconditioning matrix. This is achieved by the SSOR inner-iteration preconditioning, which works on the normal equations of the second kind $\mathcal{A}\mathcal{A}^{\mathsf{T}}\boldsymbol{z} = \boldsymbol{g}$, $\boldsymbol{u} = \mathcal{A}^{\mathsf{T}}\boldsymbol{z}$, and its preconditioning matrix $C^{\langle \ell \rangle}$ is SPD for ℓ odd for $\omega \in (0, 2)$ [52,53, Theorem 2.8]. This method exploits a symmetric splitting matrix by the forward updates, $i = 1, 2, \dots, m$ in lines 3–6 in Algorithm 6 and the reverse updates, $i = m, m-1, \dots, 1$, and can be efficiently implemented as the NE-SSOR method [63], [55, Algorithm D.8]. See [8] where SSOR preconditioning for CGNE with $\ell = 1$ is proposed. Let $\boldsymbol{\alpha}_i^{\mathsf{T}}$ be the ith row vector of $\mathcal{A}$. Algorithm 5 shows the NE-SSOR method.

Algorithm 5 NE-SSOR method.

```
1: Let $z^{\langle 0 \rangle} = \mathbf{0}$ and $\boldsymbol{u}^{\langle 0 \rangle} = \mathbf{0}$.
2: for $k = 1, 2, \dots, \ell$, do
3:    for $i = 1, 2, \dots, m$, do
4:       $d_i^{\langle k-\frac{1}{2} \rangle} := \omega[g_i - (\boldsymbol{\alpha}_i, \boldsymbol{u}^{\langle k-1 \rangle})]/\|\boldsymbol{\alpha}_i\|_2^2$
5:       $z_i^{\langle k-\frac{1}{2} \rangle} := z_i^{\langle k-1 \rangle} + d_i^{\langle k-\frac{1}{2} \rangle}$, $\boldsymbol{u}^{\langle k-1 \rangle} := \boldsymbol{u}^{\langle k-1 \rangle} + d_i^{\langle k-\frac{1}{2} \rangle}\boldsymbol{\alpha}_i$
6:    end for
7:    for $i = m, m-1, \dots, 1$, do
8:       $d_i^{\langle k \rangle} := \omega[g_i - (\boldsymbol{\alpha}_i, \boldsymbol{u}^{\langle k-1 \rangle})]/\|\boldsymbol{\alpha}_i\|_2^2$
9:       $z_i^{\langle k \rangle} := z_i^{\langle k-\frac{1}{2} \rangle} + d_i^{\langle k \rangle}$, $\boldsymbol{u}^{\langle k-1 \rangle} := \boldsymbol{u}^{\langle k-1 \rangle} + d_i^{\langle k \rangle}\boldsymbol{\alpha}_i$
10:   end for
11:   $\boldsymbol{u}^{\langle k \rangle} := \boldsymbol{u}^{\langle k-1 \rangle}$
12: end for
```

When Algorithm 5 is applied to lines 2 and 6 of Algorithm 2 and lines 2 and 6 of Algorithm 3, the normal equations of the second kind are solved approximately.

3.4 SOR inner iterations for preconditioning the AB-GMRES method

Next, we introduce the SOR method applied to the normal equations of the second kind $\mathcal{A}\mathcal{A}^\mathsf{T}\boldsymbol{p} = \boldsymbol{g}$, $\boldsymbol{z} = \mathcal{A}^\mathsf{T}\boldsymbol{p}$ with $\boldsymbol{g} = \boldsymbol{v}_k$ or $\boldsymbol{q}_k$ as used in Algorithm 4. If the relaxation parameter ω satisfies $\omega \in (0, 2)$, then the iteration matrix H of this method is semiconvergent, i.e., $\lim_{i\to\infty} H^i$ exists [21]. An efficient algorithm for this method is called NE-SOR and is given as follows [63], [55, Algorithm D.7].

Algorithm 6 NE-SOR method.

```
1: Let $z^{\langle 0\rangle} = \mathbf{0}$.
2: for $k = 1, 2, \dots, \ell$, do
3:   for $i = 1, 2, \dots, m$, do
4:     $d_i^{\langle k\rangle} := \omega[g_i - (\boldsymbol{\alpha}_i,\ z^{\langle k-1\rangle})]/\|\boldsymbol{\alpha}_i\|_2^2,\ z^{\langle k-1\rangle} := z^{\langle k-1\rangle} + d_i^{\langle k\rangle}\boldsymbol{\alpha}_i$
5:   end for
6:   $z^{\langle k\rangle} := z^{\langle k-1\rangle}$
7: end for
```

When Algorithm 6 is applied to lines 4 and 12 of Algorithm 4, the normal equations of the second kind are solved approximately.

Since the rows of A are required in the NE-(S)SOR iterations, it would be more efficient if A is stored row-wise.

3.5 Row-scaling of $\mathcal{A}$

Let $\mathcal{D}$ be a diagonal matrix whose diagonal elements are positive. Then, problem (11) is equivalent to

$$\min \|\Delta\boldsymbol{w}\|_2 \quad \text{subject to} \quad \mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}. \tag{15}$$

Denote $\hat{\mathcal{A}} := \mathcal{D}^{-1}\mathcal{A}$ and $\hat{\boldsymbol{f}} := \mathcal{D}^{-1}\boldsymbol{f}$. Then, the scaled problem (15) is

$$\min \|\Delta\boldsymbol{w}\|_2 \quad \text{subject to} \quad \hat{\mathcal{A}}\Delta\boldsymbol{w} = \hat{\boldsymbol{f}}. \tag{16}$$

If $\hat{\mathcal{B}} \in \mathbb{R}^{n\times m}$ satisfies $\mathcal{R}(\hat{\mathcal{B}}) = \mathcal{R}(\hat{\mathcal{A}}^\mathsf{T})$, then (16) is equivalent to

$$\hat{\mathcal{A}}\hat{\mathcal{B}}\hat{z} = \hat{\boldsymbol{f}}, \qquad \Delta\boldsymbol{w} = \hat{\mathcal{B}}\hat{z} \tag{17}$$

for all $\hat{\boldsymbol{f}} \in \mathcal{R}(\hat{\mathcal{A}})$. The methods discussed earlier can be applied to (17). In the NE-(S)SOR inner iterations, one has to compute $\|\hat{\boldsymbol{\alpha}}_i\|_2$, the norm of the ith row of $\hat{\mathcal{A}}$. However, this can be omitted if the ith diagonal element of $\mathcal{D}$ is chosen as the norm of the ith row of $\mathcal{A}$, that is, $\mathcal{D}(i,i) := \|\boldsymbol{\alpha}_i\|_2,\ i = 1, \dots, m$. With this choice, the matrix $\hat{\mathcal{A}}$ has unit row norm $\|\hat{\boldsymbol{\alpha}}_i\|_2 = 1,\ i = 1, \dots, m$. Hence, we do not have to compute the norms $\|\hat{\boldsymbol{\alpha}}_i\|_2$ inside the NE-(S)SOR inner iterations if we compute the norms $\|\boldsymbol{\alpha}_i\|_2$ for the construction of the scaling matrix $\mathcal{D}$. The row-scaling scheme does not incur extra CPU time. We observe in the numerical results that this scheme improves the convergence of the Krylov subspace methods.

CGNE and MRNE preconditioned by inner iterations applied to a scaled linear system $\mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ are equivalent to CG and MINRES applied to $\mathcal{D}^{-1}\mathcal{A}\mathcal{A}^\mathsf{T}C^{\langle\ell\rangle}\mathcal{D}\boldsymbol{v} = \boldsymbol{f}$, $\Delta\boldsymbol{w} = \mathcal{A}^\mathsf{T}C^{\langle\ell\rangle}\mathcal{D}\boldsymbol{v}$, respectively, and hence determine the minimum-norm solution of $\mathcal{A}\Delta\boldsymbol{w} = \boldsymbol{f}$ for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ and for all $\Delta\boldsymbol{w}_0 \in \mathbb{R}^n$ if C^ℓ is SPD. Now we give conditions under which AB-GMRES preconditioned by inner iterations applied to a scaled linear system $\mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ determines the minimum-norm solution of the unscaled one $\mathcal{A}\Delta\boldsymbol{w} = \boldsymbol{f}$.

Lemma 1 *If $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^\mathsf{T})$ and $\mathcal{D} \in \mathbb{R}^{m\times m}$ is nonsingular, then AB-GMRES applied to $\mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ determines the solution of $\min \|\Delta\boldsymbol{w}\|_2$, subject to $\mathcal{A}\Delta\boldsymbol{w} = \boldsymbol{f}$ without breakdown for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ and for all $\Delta\boldsymbol{w}_0 \in \mathbb{R}^n$ if and only if $\mathcal{N}(\mathcal{B}) \cap \mathcal{R}(\mathcal{D}^{-1}\mathcal{A}) = \{\mathbf{0}\}$.*

Proof Since $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^\mathsf{T})$ gives $\mathcal{R}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) = \mathcal{R}(\mathcal{D}^{-1}\mathcal{A}\mathcal{A}^\mathsf{T}) = \mathcal{R}(\mathcal{D}^{-1}\mathcal{A})$, the equality $\min_{\boldsymbol{u}\in\mathbb{R}^m} \|\mathcal{D}^{-1}(\boldsymbol{f} - \mathcal{A}\mathcal{B}\boldsymbol{u})\|_2 = \min_{\Delta\boldsymbol{w}\in\mathbb{R}^n} \|\mathcal{D}^{-1}(\boldsymbol{f} - \mathcal{A}\Delta\boldsymbol{w})\|_2$ holds for all $\boldsymbol{f} \in \mathbb{R}^m$ [37, Theorem 3.1]. AB-GMRES applied to $\mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ determines the kth iterate $\Delta\boldsymbol{w}_k$ by minimizing $\|\mathcal{D}(\boldsymbol{f} - \mathcal{A}\Delta\boldsymbol{w})\|_2$ over the space $\Delta\boldsymbol{w}_0 + \mathcal{K}_k(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}, \mathcal{D}^{-1}\boldsymbol{g}_0)$, and thus determines the solution of $\min \|\Delta\boldsymbol{w}\|_2$, subject to $\mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ without breakdown for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ and for all $\Delta\boldsymbol{w}_0 \in \mathbb{R}^n$ if and only if $\mathcal{N}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) \cap \mathcal{R}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) = \{\mathbf{0}\}$ [55, Theorem 5.2], which reduces to $\mathcal{R}(\mathcal{D}^{-1}\mathcal{A}) \cap \mathcal{N}(\mathcal{B}) = \{\mathbf{0}\}$ from $\mathcal{N}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) = \mathcal{R}(\mathcal{B}^\mathsf{T}\mathcal{A}^\mathsf{T}\mathcal{D}^{-\mathsf{T}})^\perp = \mathcal{R}(\mathcal{B}^\mathsf{T}\mathcal{A}^\mathsf{T})^\perp = \mathcal{R}(\mathcal{B}^\mathsf{T}\mathcal{B})^\perp = \mathcal{R}(\mathcal{B}^\mathsf{T})^\perp = \mathcal{N}(\mathcal{B})$. □

Theorem 1 *If $\mathcal{D} \in \mathbb{R}^{m\times m}$ is nonsingular and the inner-iteration matrix is semiconvergent, then AB-GMRES preconditioned by the inner iterations applied to $\mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}$ determines the solution of $\min \|\Delta\boldsymbol{w}\|_2$, subject to $\mathcal{A}\Delta\boldsymbol{w} = \boldsymbol{f}$ without breakdown for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ and for all $\Delta\boldsymbol{w}_0 \in \mathbb{R}^n$.*

Proof From Lemma 1, it is sufficient to show that $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^\mathsf{T})$ and $\mathcal{N}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) \cap \mathcal{R}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) = \{\mathbf{0}\}$. Since $\mathcal{D}^{-1}M\mathcal{D}^{-\mathsf{T}} = \mathcal{D}^{-1}(\mathcal{A}\mathcal{A}^\mathsf{T} - N)\mathcal{D}^{-\mathsf{T}}$ is the splitting matrix of $\mathcal{D}^{-1}\mathcal{A}\mathcal{A}^\mathsf{T}\mathcal{D}^{-\mathsf{T}}$ for the inner iterations, the inner-iteration matrix is $\mathcal{D}^\mathsf{T}H\mathcal{D}^{-\mathsf{T}}$. Hence, the inner-iteration preconditioning matrix $\mathcal{B} = \mathcal{A}^\mathsf{T}C^{\langle\ell\rangle}\mathcal{D}$ satisfies $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^\mathsf{T})$ [55, Lemma 4.5]. On the other hand, $\mathcal{D}^{-1}\mathcal{A}\mathcal{B} = \mathcal{D}^{-1}M(I - H^\ell)(\mathcal{D}^{-1}M)^{-1}$ satisfies $\mathcal{N}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) \cap \mathcal{R}(\mathcal{D}^{-1}\mathcal{A}\mathcal{B}) = \{\mathbf{0}\}$ [55, Lemmas 4.3, 4.4]. □

4 Numerical experiments

In this section, we compare the performance of the interior-point method based on the iterative solvers with the standard interior-point programs. We also developed an efficient direct solver coded in C to compare with the iterative solvers. For the sake of completeness, we briefly describe our direct solver first.

4.1 Direct solver for the normal equations

To deal with the rank-deficiency, we used a strategy that is similar to the Cholesky-Infinity modification scheme introduced in the LIPSOL solver [74]. However, instead of penalizing the elements that are close to zero, we removed them and solved the reduced system. We implemented this modification by an LDLT decomposition. We used the MATLAB built-in function `chol` to detect whether the matrix is symmetric positive definite. We used the `ldlchol` from CSPARSE package version 3.1.0 [19] when the matrix was symmetric positive definite, and we turned to the MATLAB built-in solver `ldl` for the semidefinite cases which uses MA57 [23].

We explain the implementation by an example where $\mathcal{A}\mathcal{A}^\mathsf{T} \in \mathbb{R}^{3\times 3}$. For matrix $\mathcal{A}\mathcal{A}^\mathsf{T}$, LDLT decomposition gives

$$\mathcal{A}\mathcal{A}^\mathsf{T} = LGL^\mathsf{T} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} g_1 & 0 & 0 \\ 0 & g_2 & 0 \\ 0 & 0 & g_3 \end{bmatrix} \begin{bmatrix} 1 & l_{21} & l_{31} \\ 0 & 1 & l_{32} \\ 0 & 0 & 1 \end{bmatrix}.$$

Correspondingly, we partition $\Delta\boldsymbol{y} = [\Delta y_1, \Delta y_2, \Delta y_3]^\mathsf{T}$ and $\boldsymbol{f} = [f_1, f_2, f_3]^\mathsf{T}$. Assuming that the diagonal element g_2 is close to zero, we let $\tilde{L} := \left[\begin{smallmatrix} 1 & 0 \\ l_{31} & 1 \end{smallmatrix}\right]$, $\tilde{G} := \left[\begin{smallmatrix} g_1 & 0 \\ 0 & g_3 \end{smallmatrix}\right]$, $\tilde{\boldsymbol{f}} = [f_1, f_3]^\mathsf{T}$, $\tilde{\Delta\boldsymbol{y}} = [\Delta y_1, \Delta y_3]^\mathsf{T}$, and solve

$$\tilde{L}\tilde{G}^{1/2}\left((\tilde{L}\tilde{G}^{1/2})^\mathsf{T}\tilde{\Delta\boldsymbol{y}}\right) = \tilde{\boldsymbol{f}},$$

using forward and backward substitutions. The solution is then given by $\Delta\boldsymbol{y} = [\Delta y_1, 0, \Delta y_3]^\mathsf{T}$.

4.2 Implementation specifications

In this section, we describe our numerical experiments.

The initial solution for the interior-point method was set using the method described in LIPSOL solver [74]. The initial solution for the Krylov subspace iterations and the inner iterations was set to zero.

We set the maximum number of the interior-point iterations as 99 and the stopping criterion regarding the error measure as

$$\Gamma^{(k)} \le \epsilon_{\text{out}} = 10^{-8}, \tag{18}$$

where $\Gamma^{(k)}$ is defined by (10).

For the iterative solver for the linear system (11), we set the maximum number of iterations for CGNE, MRNE and AB-GMRES as m, and relaxed it to 40,000 for some difficult problems for CGNE and MRNE. We set the stopping criterion for the scaled residual as

$$\|\hat{\boldsymbol{f}} - \hat{A}\Delta\boldsymbol{w}^{(k)}\|_2 \le \epsilon_{\text{in}}\|\hat{\boldsymbol{f}}\|_2,$$

where ϵ_{in} is initially 10^{-6} and is kept in the range $[10^{-14}, 10^{-4}]$ during the process. We adjusted ϵ_{in} according to the progress of the interior-point iterations. We truncated the iterative solving prematurely in the early interior-point iterations, and pursued a more precise direction as the LP solution was approached. The progress was measured by the error measure $\Gamma^{(k)}$. Concretely, we adjusted ϵ_{in} as

$$\epsilon_{\text{in}}^{(k)} = \begin{cases} \epsilon_{\text{in}}^{(k-1)} \times 0.75 & \boldsymbol{if} \ \log_{10} \Gamma^{(k)} \in (-3, 1], \\ \epsilon_{\text{in}}^{(k-1)} \times 0.375 & \boldsymbol{if} \ \log_{10} \Gamma^{(k)} \in (-\infty, -3]. \end{cases}$$

For steps where iterative solvers failed to converge within the maximum number of iterations, we adopted the iterative solution with the minimum residual norm and slightly increased the value of ϵ_{in} by multiplying by 1.5 which would be used in the next interior-point step.

Note that preliminary experiments were conducted with the tolerance being fixed for all the problems. However, further experiments showed that adjusting the parameter ϵ_{in} with the progress towards an optimal solution worked better. This is also another advantage of using iterative solvers rather than direct solvers.

We adopt the implementation of AB-GMRES preconditioned by NE-SOR inner-iterations [56] with the additional row-scaling scheme (Sect. 3.5). No restarts were used for the AB-GMRES method. The non-breakdown conditions discussed in Sects. 3.1, 3.2 are satisfied.

For the direct solver, the tolerance for dropping pivot elements close to zero was 10^{-16} for most of the problems; for some problems this tolerance has to be increased to 10^{-6} to overcome breakdown.

The experiment was conducted on a MacBook Pro with a 2.6 GHz Intel Core i5 processor with 8 GB of random-access memory, OS X El Capitan version 10.11.2. The interior-point method was coded in Matlab R2014b and the iterative solvers including AB-GMRES (NE-SOR), CGNE (NE-SSOR), and MRNE (NE-SSOR), were coded in C and compiled as Matlab Executable (MEX) files accelerated with Basic Linear Algebra Subprograms (BLAS).

We compared our implementation with PDCO version 2013 [65] and three solvers available in CVX [35,36]: SDPT3 version 4.0 [68,69], SeDuMi version 1.34 [68] and MOSEK version 7.1.0.12 [57], with the default interior-point stopping criterion (18). Note that SDPT3, SeDuMi, and PDCO are non-commercial public domain solvers, whereas MOSEK is a commercial solver known as one of the state-of-the-art solvers. PDCO provides several choices for the solvers for the interior-point steps, among which we chose the direct (Cholesky) method and the LSMR method. Although MINRES solver is another iterative solver available in PDCO, its homepage [65] suggests that LSMR performs better in general. Thus, we tested with LSMR. For PDCO parameters, we chose to suppress scaling for the original problem. The other solvers were implemented with the CVX Matlab interface, and we recorded the CPU time reported in the screen output of each solver. However, it usually took a longer time for the CVX to finish the whole process. The larger the problem was, the more apparent this extra CPU time became. For example, for problem `ken_18`, the screen output of SeDuMi was 765.3 s while the total processing time was 7615.2 s.

Table 1 Overall performance of the solvers on 127 testing problems

Status	Solved	Failed	Expensive
AB-GMRES (NE-SOR)	123	2	2
CGNE (NE-SSOR)	124	3	0
MRNE (NE-SSOR)	125	2	0
Modified Cholesky	117	10	0
SDPT3	76	46	5
SeDuMi	104	23	0
MOSEK	127	0	0
PDCO (Direct)	110	17	0
PDCO (LSMR)	88	35	4

We tested on two classes of LP problems: 127 typical problems from the benchmark libraries and 13 problems arising from basis pursuit. The results are described in Sects. 4.3 and 4.4, respectively.

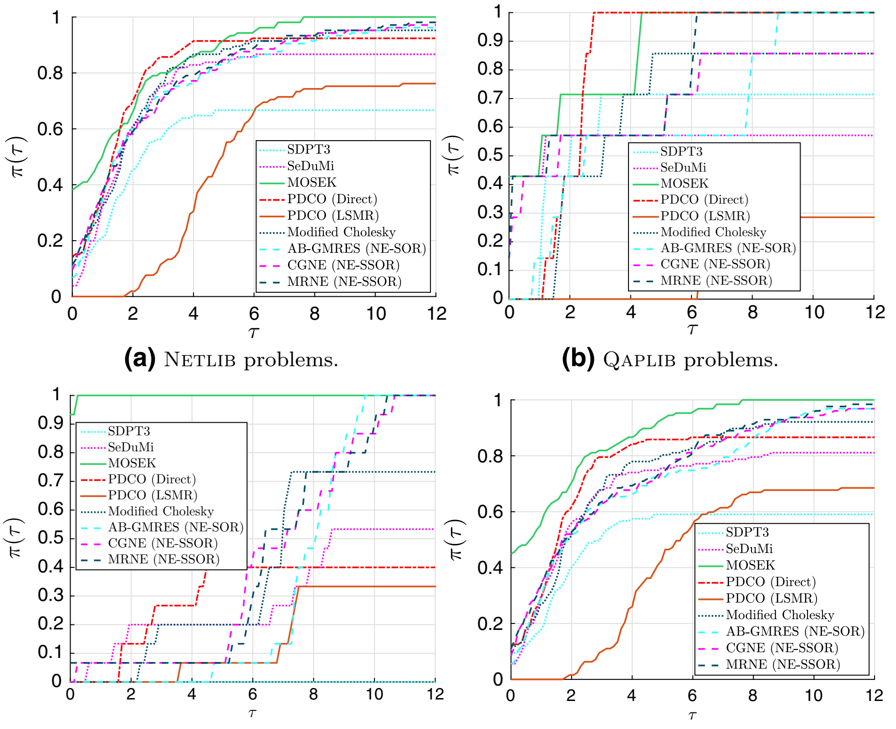
4.3 Typical LP problems: sparse and ill-conditioned problems

We tested 127 typical LP problems from the NETLIB, QAPLIB and MITTELMANN collections in [20]. Most of the problems have sparse and full-rank constraint matrix A (except problems `bore3d` and `cycle`). For the problems with $\boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u},\ \boldsymbol{l} \ne \boldsymbol{0},\ \boldsymbol{u} \ne \infty$, we transform them using the approach in LIPSOL [74].

The overall summary of numerical experiments on the 127 typical problems is given in Table 1. The counts in column "Failed" include the case where a problem was solved at a relaxed tolerance (phrased as "inaccurately solved" in CVX). Column "Expensive" refers to the case where the interior-point iterations took more than a time limit of 20 hours.

MOSEK was most stable in the sense that it solved all 127 problems, and MRNE (NE-SSOR) came next with only two failures with the NETLIB problems `greanbea` and `greanbeb`. CGNE (NE-SSOR) method solved almost all the problems that MRNE (NE-SSOR) solved, except for the largest QAPLIB problem, which was solved to a slightly larger tolerance level of 10^{-7}. AB-GMRES (NE-SOR) was also very stable and solved the problems accurately enough. However, it took longer than 20 hours for two problems that have 105,127 and 16,675 equations, respectively, although it succeeded in solving larger problems such as `pds-80`. The other solvers were less stable. The modified Cholesky solver and PDCO (Direct) solved 92% and 87% of the problems, respectively, although they were faster than the other solvers for the problems that they could successfully solve. PDCO (LSMR) solved 69% problems and was slower than the proposed solvers. The reason could be that it does not use preconditioners. SDPT3 solved 60% and SeDuMi 82% of the problems. Here we should mention that SeDuMi and SDPT3 are designed for LP, SDP, and SOCP, while our code is (currently) tuned solely for LP.

Note that MOSEK solver uses a multi-corrector interior-point method [30] while our implementation is a single corrector (i.e., predictor-corrector) method. This led



(a) NETLIB problems. **(b)** QAPLIB problems.

(c) MITTELMANN problems. **(d)** All the problems.

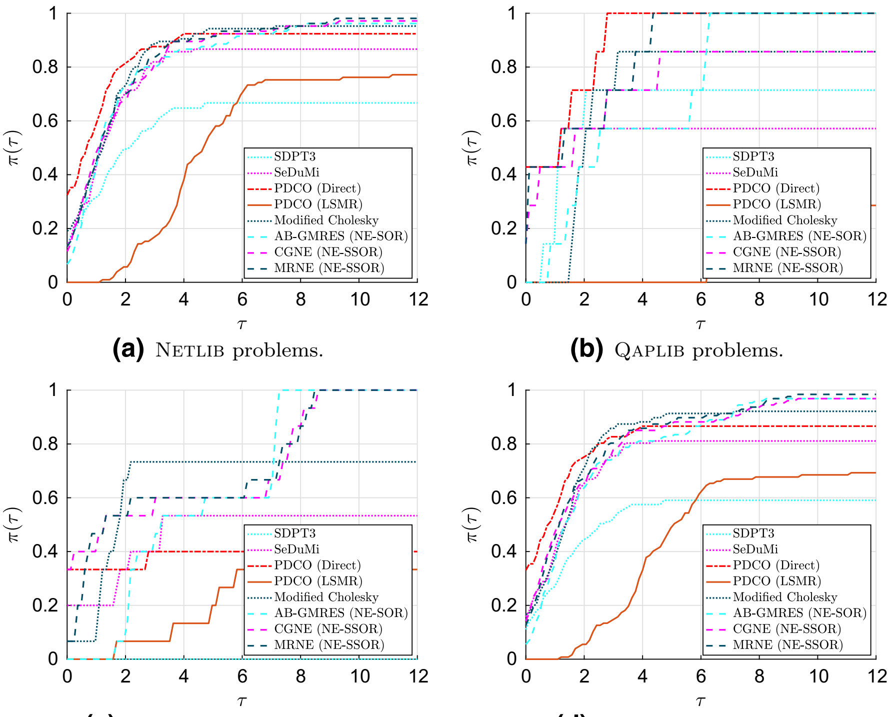
Fig. 1 Dolan–Moré profiles comparing the CPU time costs for the proposed solvers, public domain and commercial solvers

to different numbers of interior-point iterations as shown in the tables. Thus, there is still room for improvement in the efficiency of our solver based on iterative solvers if a more elaborately tuned interior-point framework such as the one in MOSEK is adopted.

In order to show the trends of performance, we use the Dolan–Moré performance profiles [22] in Figs. 1 and 2, with $\pi(\tau) := P(\log_2 r_{ps} \leq \tau)$ the proportion of problems for which $\log_2$-scaled performance ratio is at most τ, where $r_{ps} := t_{ps}/t_{\mathrm{p}}^*$, t_{ps} is the CPU time for solver s to solve problem p, and t_{p}^* is the minimal CPU time for problem p. Figure 1 includes the commercial solver MOSEK while Fig. 2 does not. Note that the generation of Fig. 2 is not by simply removing the curve of MOSEK from Fig. 1, but rather removing the profile of MOSEK from the comparison dataset and thus changing the minimum CPU time cost for each problem. The comparison indicates that the iterative solvers, although slower than the commercial solver MOSEK in some cases, were often able to solve the problems to the designated accuracy.

In Tables 2, 3, and 4, we give the following information:

1. the name of the problem and the size (m, n) of the constraint matrix,
2. the number of interior-point iterations required for convergence,



(a) Netlib problems. **(b)** Qaplib problems. **(c)** Mittelmann problems. **(d)** All the problems.

Fig. 2 Dolan–Moré profiles comparing the CPU time costs for the proposed solvers and public domain solvers

3. CPU time for the entire computation in seconds. For the cases shorter than 3000 s, CPU time is taken as an average over 10 measurements. In each row, we indicate in boldface and italic the fastest and second fastest solvers in CPU time, respectively.

Besides the statistics, we also use the following notation:

†inaccurately solved, i.e., the value of ϵ_{out} was relaxed to a larger level. In the column "Iter", we provide extra information $\dagger_a$ at the stopping point: for our solvers, $a = \lfloor \log_{10} \Gamma^{(k)} \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function; for CVX solvers, $a = \lfloor \log_{10} \mu \rfloor$ as provided in the CVX output; PDCO solvers do not provide this information, thus they are not given;
f the interior-point iterations diverged;
t the iterations took longer than 20 hours.

Note that all zero rows and columns of the constraint matrix A were removed beforehand. The problems marked with # are with rank-deficient A even after this preprocessing. For these problems we put rank(A) in brackets after m, which is computed using the MATLAB function `sprank`.

In order to give an idea of the typical differences between methods, we present the interior-point convergence curves for problem `ken_13`. The problem has a constraint matrix $A \in \mathbb{R}^{28,632 \times 42,659}$ with full row rank and 97, 246 nonzero elements.

Different aspects of the performance of the four solvers are displayed in Fig. 3. The red dotted line with diamond markers represents the quantity related to AB-GMRES (NE-SOR), the blue with downward-pointing triangle CGNE (NE-SSOR), the yellow with asterisk MRNE (NE-SSOR), and the dark green with plus sign the modified Cholesky solver. Note that for this problem `ken_13`, the modified Cholesky solver became numerically inaccurate at the last step and it broke down if the default dropping tolerance was used. Thus, we increased it to 10^{-6}.

Figure 3a shows $\kappa(\mathcal{A}\mathcal{A}^\mathsf{T})$ in $\log_{10}$ scale. It verifies the claim that the least squares problem becomes increasingly ill-conditioned at the final steps in the interior-point process: $\kappa(\mathcal{A}\mathcal{A}^\mathsf{T})$ started from around 10^{20} and increased to 10^{80} at the last 3–5 steps. Figure 3b shows the convergence curve of the duality measure μ in $\log_{10}$ scale. The μ drops below the tolerance and the stopping criterion is satisfied. Although it is not shown in the figure, we found that the interior-point method with modified Cholesky with the default value of the dropping tolerance 10^{-16} stagnated for $\mu \simeq 10^{-4}$. Comparing with Fig. 3a, it is observed that the solvers started to behave differently as $\kappa(\mathcal{A}\mathcal{A}^\mathsf{T})$ increased sharply.

Figure 3c, d show the relative residual norm $\|\boldsymbol{f}_{\mathrm{af}} - \mathcal{A}\mathcal{A}^\mathsf{T}\Delta\boldsymbol{y}_{\mathrm{af}}\|_2/\|\boldsymbol{f}_{\mathrm{af}}\|_2$ in the predictor stage and $\|\boldsymbol{f}_{\mathrm{cc}} - \mathcal{A}\mathcal{A}^\mathsf{T}\Delta\boldsymbol{y}_{\mathrm{cc}}\|_2/\|\boldsymbol{f}_{\mathrm{cc}}\|_2$ in the corrector stage, respectively. The quantities are in $\log_{10}$ scale. The relative residual norm for modified Cholesky tended to increase with the interior-point iterations and sharply increased in the final phase when it lost accuracy in solving the normal equations for the steps. We observed similar trends for other test problems and, in the worst cases, the inaccuracy in the solutions prevented interior-point convergence. Among the iterative solvers, AB-GMRES (NE-SOR) and MRNE (NE-SSOR) were the most stable in keeping the accuracy of solutions to the normal equations; CGNE (NE-SSOR) performed similarly but lost numerical accuracy at the last few interior-point steps.

Figure 3e, f show the CPU time and number of iterations of the Krylov methods for each interior-point step, respectively. It was observed that the CPU time of the modified Cholesky solver was more evenly distributed in the whole process while that of the iterative solvers tended to be less in the beginning and ending phases. At the final stage, AB-GMRES (NE-SOR) required the fewest number of iterations but cost much more CPU time than the other two iterative solvers. This can be explained as follows: AB-GMRES (NE-SOR) requires increasingly more CPU time and memory with the number of iterations because it has to store the orthonormal vectors in the modified Gram-Schmidt process as well as the Hessenberg matrix. In contrast, CGNE (NE-SSOR) and MRNE (NE-SSOR) based methods require constant memory. CGNE (NE-SSOR) took more iterations and CPU time than MRNE (NE-SSOR). Other than $\mathcal{A}$ and the preconditioner, the memory required for k iterations of AB-GMRES is $\mathcal{O}(k^2 + km + n)$ and that for CGNE and MRNE iterations is $\mathcal{O}(m + n)$ [37,55]. This explains why AB-GMRES (NE-SOR), although requiring fewer iterations, usually takes longer to obtain the solution at each interior-point step. We also did experiments on restarting AB-GMRES for a few problems. However, the performance was not competitive compared to the non-restarted version.

On the other hand, the motivation for using AB-GMRES (NE-SOR) is that GMRES is more robust for ill-conditioned problems than the symmetric solvers CG and MINRES. This is because GMRES uses a modified Gram-Schmidt process to orthogonalize

Table 2 Experiments on NETLIB problems

Problem	m	n	AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK		PDCO Direct		PDCO LSMR	
			Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
25fv47	821	1876	25	4.62	25	5.00	25	4.60	25	3.67	59	2.50	29	*2.30*	26	3.90	45	**1.92**	48	20.60
adlittle	56	138	12	**0.03**	13	**0.03**	13	*0.05*	12	0.09	16	0.16	14	0.10	10	1.98	25	0.15	25	1.60
afiro	27	51	8	*0.02*	8	**0.01**	8	**0.01**	8	0.03	11	0.11	7	0.10	9	1.91	18	0.11	18	0.48
agg	488	615	21	*0.72*	21	0.88	24	0.79	21	1.49	34	**0.61**	32	0.90	18	2.24	34	0.82	34	3.25
agg2	516	758	21	0.64	21	*0.56*	23	**0.53**	21	1.55	32	1.28	23	1.00	13	2.12	33	1.11	33	2.78
agg3	516	758	19	0.68	19	**0.52**	21	*0.58*	19	1.38	32	1.24	22	1.10	12	2.06	32	1.45	32	2.52
bandm	305	472	18	0.73	19	0.62	19	0.74	17	0.90	42	1.52	20	*0.50*	15	2.17	35	**0.44**	37	5.96
beaconfd	173	295	13	**0.07**	13	**0.07**	13	**0.07**	12	0.41	15	0.22	10	*0.20*	8	1.97	26	0.30	26	0.97
blend	74	114	12	**0.06**	14	*0.07*	13	0.08	12	0.11	15	0.16	11	0.10	9	1.98	20	0.30	20	1.94
bnl1	643	1586	25	2.53	25	4.66	25	4.92	25	*1.95*	$\dagger_{-5}$	†	64	2.50	20	2.51	52	**1.06**	59	27.03
bnl2	2324	4486	32	44.98	32	23.37	32	27.63	32	12.41	$\dagger_{-4}$	†	38	5.80	25	**2.66**	63	*5.42*	63	22.86
bore3d#	233 (232)	334	19	0.35	19	*0.23*	19	**0.21**	19	0.63	35	1.92	18	1.50	19	3.00	31	0.38	31	2.76
brandy	220	303	17	*0.43*	18	0.86	18	0.86	17	0.59	46	1.02	19	**0.40**	12	2.04	34	*0.43*	34	4.85
capri	271	482	19	*0.80*	19	0.88	19	0.91	19	1.04	47	3.22	33	1.60	14	2.63	45	**0.59**	50	16.49
cre_a	3516	7248	30	186.77	30	48.43	31	35.79	31	105.60	$\dagger_{-7}$	†	28	8.70	20	**2.69**	61	*4.19*	61	66.84
cre_b	9648	77,137	43	787.95	42	611.11	42	*455.04*	53	1143.90	$\dagger_{-6}$	†	$\dagger_{-7}$	†	19	**3.63**	£	£	£	£
cre_c	3068	6411	30	268.84	32	47.92	33	46.12	33	79.67	£	£	28	*7.70*	17	**2.56**	70	4.20	70	85.35
cre_d	8926	73,948	37	387.17	37	316.81	37	213.69	37	847.00	£	£	34	*42.10*	16	**3.06**	£	£	£	£
cycle#	1903 (1875)	3371	30	61.87	31	50.44	61	185.12	£	£	$\dagger_{-6}$	†	30	*5.30*	20	**2.76**	82	7.18	£	£
czprob	929	3562	39	**1.51**	38	*1.60*	39	1.73	39	10.45	$\dagger_{-5}$	†	39	2.80	27	2.91	68	1.96	68	5.24
d2q06c	2171	5831	32	132.75	33	581.83	36	750.06	32	24.09	84	6.43	29	*4.10*	21	**2.85**	63	6.66	£	£
d6cube	415	6184	23	3.77	24	7.41	23	7.12	26	2.68	34	**1.65**	£	£	11	*2.50*	76	6.60	76	21.61

Table 2 continued

Problem	m	n	AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK		PDCO Direct		PDCO LSMR	
			Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
degen2	444	757	15	1.26	16	1.13	16	1.18	21	2.27	17	*0.41*	13	**0.40**	8	2.12	29	1.10	f	f
degen3	1503	2604	19	27.30	21	13.26	21	13.38	19	27.52	†$_{-6}$	†	15	**2.00**	12	*2.18*	38	4.04	f	f
dfl001	6071	12,230	48	4336.35	50	2044.54	55	2205.16	91	3131.77	f	f	†$_{-5}$	†	22	**7.46**	72	*23.01*	87	1389.78
e226	223	472	21	0.64	20	0.61	21	0.82	20	*0.59*	61	1.17	22	0.60	14	1.97	38	**0.45**	39	6.13
etamacro	400	816	30	*1.23*	31	1.58	31	1.43	30	2.30	f	f	30	2.80	20	2.82	38	**0.87**	38	3.88
fffff800	524	1028	32	4.11	30	6.29	33	6.39	32	3.31	44	**0.86**	46	*1.60*	22	2.55	61	1.98	82	35.60
fit1d	24	1049	21	0.78	21	*0.45*	21	0.49	19	**0.38**	36	2.11	18	0.80	13	0.67	32	0.58	38	1.41
fit1p	627	1677	16	4.01	16	5.31	16	5.14	16	3.56	25	*1.78*	53	2.00	17	**0.73**	32	3.43	34	5.36
fit2d	25	10,524	20	3.40	21	3.54	21	3.72	20	*2.40*	41	3.10	15	2.60	18	**0.79**	44	3.78	49	5.45
fit2p	3000	13,525	19	1103.13	32	1755.13	32	1831.13	19	102.02	27	*3.69*	40	8.90	17	**0.82**	33	47.12	39	249.45
ganges	1309	1706	18	8.21	18	27.73	21	33.06	18	3.80	22	**0.90**	26	1.60	15	*0.91*	30	1.18	30	15.29
gfrd_pnc	616	1160	21	1.15	22	1.04	21	0.88	21	0.98	27	*0.85*	20	1.00	29	0.90	39	**0.56**	38	4.29
greenbea	2392	5598	f	f	f	f	f	f	f	f	†$_{-7}$	†	f	f	35	*20.52*	69	**8.20**	69	227.30
greenbeb	2392	5598	†$_{-8}$	†	f	f	f	f	f	f	†$_{-6}$	†	69	30.57	30	*18.86*	88	**8.50**	90	220.00
grow15	300	645	19	0.43	19	**0.35**	20	*0.37*	17	0.40	21	0.80	25	1.00	13	0.89	f	f	f	f
grow22	440	946	20	0.68	20	*0.59*	22	*0.59*	18	**0.53**	22	0.93	26	1.40	14	0.95	f	f	f	f
grow7	140	301	18	**0.12**	18	*0.16*	18	**0.12**	16	*0.16*	19	0.66	19	0.70	12	0.69	f	f	f	f
israel	174	316	24	0.99	27	0.94	27	1.06	25	1.12	34	**0.51**	20	*0.60*	15	2.14	45	0.95	40	2.45
kb2	43	68	16	*0.09*	17	**0.08**	17	**0.08**	15	0.11	26	0.71	15	0.50	16	0.75	33	0.20	35	2.21
ken_07	2426	3602	17	4.14	18	2.39	17	2.24	16	*1.07*	33	1.74	18	1.80	15	**0.79**	26	*1.07*	27	51.26
ken_11	14,694	21,349	22	636.24	23	123.23	23	85.95	22	7.83	†$_{-4}$	†	38	10.60	31	**1.87**	43	*4.84*	56	3352.37
ken_13	28,632	42,659	27	2633.00	28	365.15	29	348.51	27	23.90	f	f	43	29.50	20	**2.83**	57	*11.26*	72	25,533.36

Table 2 continued

Problem	*m*	*n*	AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK		PDCO Direct		PDCO LSMR	
			Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
ken_18	105,127	154,699	t	t	38	12,893.63	46	21,315.47	38	324.89	f	f	59	765.30	20	**24.98**	82	*74.94*	t	t
lotfi	153	366	16	*0.28*	16	**0.24**	16	0.32	16	0.39	37	1.14	20	1.20	15	2.47	32	0.36	38	5.18
maros_r7	3136	9408	15	57.78	15	29.69	15	31.68	15	11.14	21	5.39	15	*4.80*	12	**3.29**	23	6.14	23	22.77
modszk1	687	1620	23	2.70	23	3.60	23	3.48	22	2.54	29	*0.85*	23	1.00	22	0.92	36	**0.73**	36	8.68
osa_07	1118	25,067	34	12.35	32	6.26	36	8.51	27	5.85	31	*3.90*	31	4.90	14	**2.55**	64	7.75	65	11.93
osa_14	2337	54,797	38	11.41	32	9.11	37	11.81	37	16.07	37	7.65	36	*7.30*	18	**3.03**	76	13.29	76	21.64
osa_30	4350	104,374	39	22.69	41	19.08	38	17.16	36	28.98	37	12.49	40	*11.50*	17	**3.36**	f	f	f	f
osa_60	10,280	243,246	30	48.25	40	40.12	33	37.26	34	67.90	39	26.73	41	*21.70*	17	**5.10**	f	f	f	f
pds_02	2953	7716	29	4.43	29	3.43	29	4.16	29	*3.16*	$\dagger_{-5}$	†	30	6.90	18	**0.82**	36	3.31	36	16.03
pds_06	9881	29,351	48	49.77	48	44.17	51	45.85	48	44.65	f	f	51	61.50	23	**1.45**	51	*9.90*	59	238.85
pds_10	16,558	49,932	51	91.60	52	87.75	50	79.22	52	130.17	f	f	74	157.20	28	**2.54**	65	*26.43*	77	406.33
pds_20	33,874	108,175	61	1365.98	64	1155.95	62	683.72	62	665.05	f	f	$\dagger_{-7}$	†	34	**11.02**	86	*147.67*	f	f
perold	625	1506	36	4.71	36	6.71	36	6.97	37	2.82	$\dagger_{-6}$	†	$\dagger_{-7}$	†	24	**0.87**	65	*2.54*	87	65.11
pilot	1441	4860	33	31.54	33	51.15	33	49.36	33	16.18	f	f	81	19.70	39	**1.73**	79	*9.26*	79	57.65
pilot4	410	1123	30	2.11	30	2.12	30	2.29	30	2.26	$\dagger_{-7}$	†	f	f	27	**0.78**	66	*1.57*	f	f
pilot87	2030	6680	39	55.59	39	105.77	39	102.58	39	33.13	88	11.54	76	12.60	38	**2.45**	74	*10.86*	74	77.94
pilot_ja	940	2267	35	13.02	37	19.51	36	14.79	37	4.84	f	f	f	f	29	**0.71**	74	*4.71*	f	f
pilot_we	722	2928	35	5.67	39	8.58	38	7.62	35	*2.42*	$\dagger_{-7}$	†	44	4.90	31	**0.71**	73	2.47	f	f
pilotnov	975	2446	24	5.70	25	5.02	27	4.07	22	*2.90*	f	f	f	f	17	**0.73**	71	4.34	84	13.17
qap12	3192	8856	19	758.92	21	144.74	20	99.35	19	50.45	26	21.78	$\dagger_{-7}$	†	17	**6.09**	31	*14.68*	f	f
qap15	6330	22,275	23	5530.52	25	789.81	24	581.25	24	335.83	52	330.31	$\dagger_{-7}$	†	17	**21.11**	42	*103.91*	f	f
qap8	912	1632	11	1.73	12	*1.09*	11	**0.98**	10	2.75	13	1.25	8	1.10	7	2.16	19	2.19	f	f

Table 2 continued

Problem	*m*	*n*	AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK		PDCO Direct		PDCO LSMR	
			Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
sc105	105	163	10	0.05	10	*0.04*	10	*0.04*	10	**0.02**	20	0.50	10	0.20	8	2.13	22	0.24	22	1.83
sc205	205	317	11	0.17	11	0.09	11	*0.07*	10	**0.05**	18	0.61	12	0.30	10	2.16	24	0.23	24	2.14
sc50a	50	78	10	0.03	10	**0.00**	6	*0.02*	10	*0.02*	12	0.17	8	0.20	8	2.13	20	0.28	20	1.18
sc50b	50	78	7	**0.01**	7	*0.02*	7	*0.02*	7	0.03	11	0.26	7	0.20	6	1.94	18	0.15	18	1.02
scagr25	471	671	18	0.93	18	0.69	18	0.71	17	**0.20**	35	0.84	21	0.70	21	2.63	29	*0.34*	29	6.06
scagr7	129	185	14	0.15	15	0.11	15	*0.11*	14	**0.07**	33	0.71	17	0.50	19	2.52	27	0.25	27	2.35
scfxm1	330	600	18	1.03	19	1.05	20	1.14	18	*0.70*	52	1.40	20	0.80	15	2.42	39	**0.56**	38	9.27
scfxm2	660	1200	21	2.44	22	4.73	23	4.71	21	1.35	58	1.59	24	*1.30*	18	2.56	43	**0.71**	51	34.91
scfxm3	990	1800	22	5.94	23	12.64	24	12.10	22	1.64	59	1.79	25	*1.50*	16	2.53	43	**1.20**	79	78.55
scorpion	388	466	15	0.28	16	*0.23*	16	0.26	15	**0.20**	17	0.39	11	0.30	11	2.21	27	0.33	30	11.73
scrs8	490	1275	25	0.91	26	0.78	25	*0.77*	25	**0.61**	37	1.06	35	1.70	14	2.41	48	0.81	51	6.81
scsd1	77	760	9	0.06	9	0.05	9	**0.03**	9	*0.04*	12	0.23	8	0.20	8	2.02	20	0.22	20	1.45
scsd6	147	1350	11	0.17	12	*0.12*	11	0.13	11	**0.07**	15	0.32	11	0.40	10	2.06	24	0.39	24	2.03
scsd8	397	2750	12	0.76	12	0.71	12	0.64	11	**0.16**	13	*0.32*	10	0.60	7	1.93	22	0.51	22	3.05
sctap1	300	660	17	*0.31*	19	0.38	19	0.36	17	**0.12**	20	0.46	20	0.50	11	2.15	36	0.39	38	5.06
sctap2	1090	2500	20	1.36	20	1.21	21	1.04	19	1.75	21	**0.48**	1	*0.60*	9	2.05	33	0.85	35	6.75
sctap3	1480	3340	19	1.33	19	1.14	20	1.22	18	2.31	23	*0.94*	13	**0.40**	9	2.11	35	1.12	39	6.99
share1b	117	253	23	0.50	24	0.51	24	0.48	23	**0.16**	27	0.52	22	0.50	23	2.74	34	*0.33*	55	11.25
share2b	96	162	12	*0.16*	14	0.20	16	0.21	12	**0.09**	26	0.60	19	0.30	15	2.47	26	0.22	39	5.83
shell	536	1777	19	0.61	19	*0.57*	19	0.58	19	1.68	f	f	31	1.10	22	**0.56**	43	0.70	43	5.78
ship04l	402	2166	14	**0.26**	14	**0.26**	14	**0.26**	15	1.00	20	*0.74*	17	0.80	10	1.86	40	*0.74*	40	3.75
ship04s	402	1506	15	0.78	15	*0.30*	15	**0.21**	14	1.14	20	0.67	17	0.70	11	0.48	36	0.51	36	3.28

Table 2 continued

Problem	m	n	AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK		PDCO Direct		PDCO LSMR	
			Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
ship08l	778	4363	16	*0.82*	17	1.33	17	1.28	16	2.47	21	**0.51**	18	0.90	11	1.93	43	1.31	43	6.45
ship08s	778	2467	15	0.44	16	0.46	16	0.60	15	1.82	20	**0.32**	16	*0.40*	10	1.88	38	0.80	38	5.52
ship12l	1151	5533	20	1.48	19	2.21	20	2.01	19	4.66	22	**0.65**	23	1.90	14	2.04	42	*1.63*	47	10.85
ship12s	1151	5533	17	*0.90*	19	1.00	19	0.94	17	2.66	22	**0.41**	17	*0.90*	12	1.96	35	0.94	37	6.66
sierra	1227	2735	17	1.28	19	1.37	19	*1.05*	21	1.60	f	f	29	3.50	16	**0.59**	34	1.71	35	9.58
stair	356	614	22	1.43	22	1.63	22	1.87	22	0.96	$\dagger_{-6}$	$\dagger$	18	*0.70*	15	**0.52**	30	0.85	30	6.11
standata	359	1274	18	0.63	17	**0.34**	17	*0.38*	17	0.86	f	f	19	0.70	9	0.48	54	0.74	55	4.89
standgub	361	1383	17	*0.35*	17	**0.30**	17	0.37	17	0.91	f	f	19	0.70	9	0.51	53	0.77	55	5.00
standmps	467	1274	25	0.81	24	*0.68*	25	0.82	24	1.71	f	f	15	0.70	17	**0.53**	61	0.88	62	6.43
stocfor1	117	165	19	0.13	21	0.13	20	0.20	19	**0.09**	30	0.71	17	0.50	11	2.21	26	*0.12*	29	3.41
stocfor2	2157	3045	23	37.36	24	18.00	24	17.59	21	13.43	53	*1.95*	$\dagger_{-4}$	$\dagger$	17	2.54	42	**1.64**	f	f
stocfor3	16,675	23,541	t	t	38	4590.71	37	4071.37	$\dagger_{-7}$	$\dagger$	80	11.05	f	f	26	**3.37**	62	*7.41*	f	f
truss	1000	8806	19	6.62	21	10.22	22	10.59	19	3.29	21	**1.12**	19	*1.90*	12	2.27	41	3.59	f	f
tuff	333	628	21	1.63	22	1.27	24	2.03	21	1.39	$\dagger_{-7}$	$\dagger$	21	*0.80*	18	**0.60**	45	0.95	69	12.60
vtp_base	198	346	24	0.69	24	*0.52*	24	0.61	24	0.77	39	1.26	42	1.30	12	0.69	32	**0.33**	36	3.93
wood1p	244	2595	17	1.75	17	*1.34*	17	**1.19**	f	f	38	2.75	19	2.00	10	2.17	f	f	f	f
woodw	1098	8418	25	5.12	27	6.72	28	7.34	22	3.73	f	f	33	*3.20*	17	**2.47**	81	7.76	f	f

In each row, boldface and italic denote the fastest and second fastest solvers in CPU time, respectively

Table 3 Experiments on QAPLIB problems

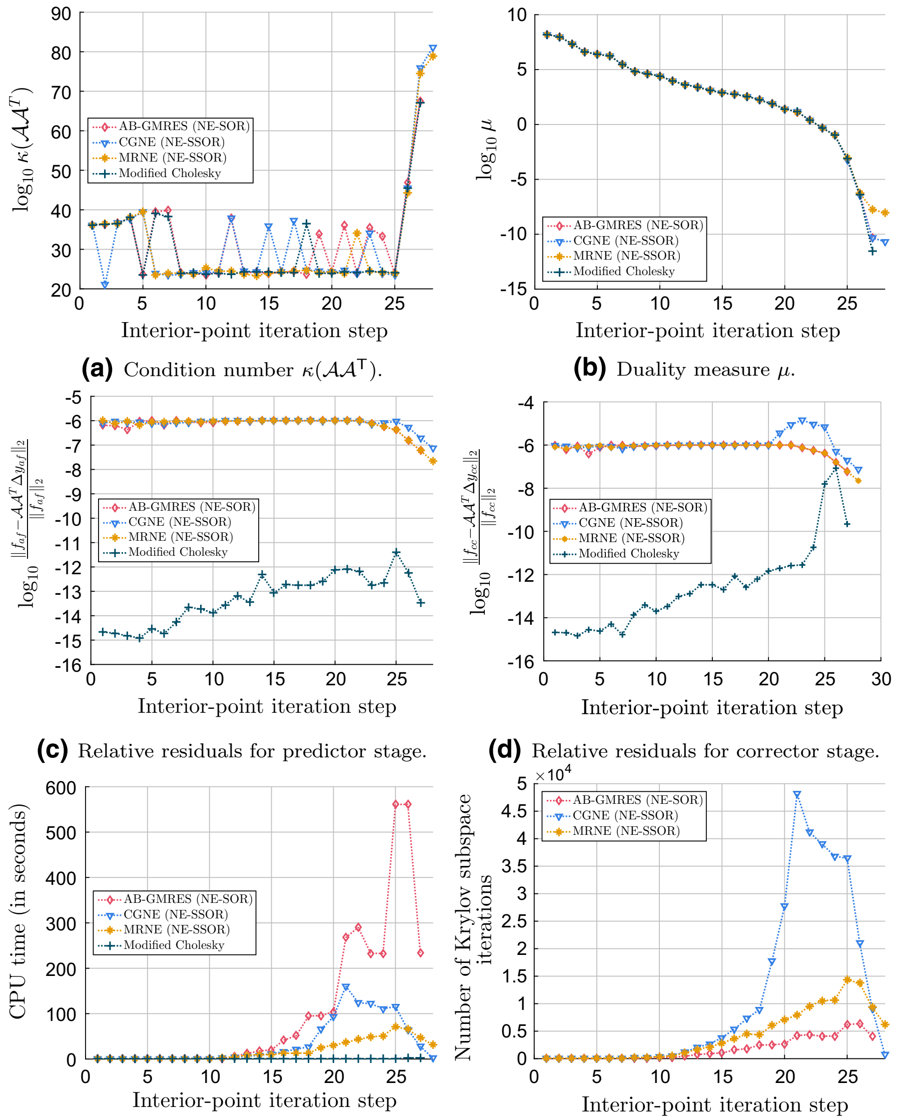
Problem	m	n	AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK		PDCO Direct		PDCO LSMR	
			Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
nug05	201	225	7	*0.16*	7	**0.09**	7	**0.09**	7	0.27	12	0.36	5	0.20	5	1.81	17	0.48	17	6.60
nug06	372	486	10	0.56	10	0.31	10	0.25	8	0.83	11	*0.22*	6	**0.10**	6	1.84	18	0.65	28	22.59
nug07	602	931	12	1.83	13	0.96	12	*0.72*	12	2.02	18	1.48	10	**0.70**	8	2.09	25	1.98	f	f
nug08	912	1632	10	3.27	11	*1.03*	12	1.06	10	3.26	16	2.08	8	**1.00**	7	1.96	19	2.21	f	f
nug12	3192	8856	19	1287.19	20	427.16	19	355.36	20	73.13	$\dagger_{-7}$	†	$\dagger_{-7}$	†	17	**5.57**	31	*18.07*	f	f
nug15	6330	22,275	23	9521.25	25	809.23	24	773.55	23	559.88	33	171.64	$\dagger_{-5}$	†	17	**22.13**	42	*121.43*	f	f
nug20	15,240	72,600	25	60,223.29	$\dagger_{-7}$	†	33	16,650.52	$\dagger_{-7}$	†	$\dagger_{-7}$	†	$\dagger_{-5}$	†	19	**243.71**	44	*1260.90*	t	t

In each row, boldface and italic denote the fastest and second fastest solvers in CPU time, respectively

Table 4 Experiments on MITTELMANN problems

Problem	m	n	AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK		PDCO Direct		PDCO LSMR	
			Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
fome11	12,142	24,460	47	6900.09	48	14,156.31	53	12,270.84	f	f	f	f	$\dagger_{-5}$	†	23	**8.97**	72	*51.99*	69	1577.24
fome12	24,284	48,920	48	12,568.26	48	38,138.98	52	28,159.85	f	f	f	f	$\dagger_{-7}$	†	21	**33.17**	72	*101.01*	69	3736.93
fome13	48,568	97,840	47	25,726.58	50	37,625.03	54	63,301.06	f	f	f	f	$\dagger_{-7}$	†	24	**61.01**	72	*186.56*	71	10,247.41
fome20	33,874	108,175	61	1510.85	64	1240.23	62	689.71	62	692.71	f	f	$\dagger_{-7}$	†	34	**8.96**	86	*164.60*	t	t
fome21	67,748	216,350	74	12,671.62	74	3185.03	84	3822.02	75	*1617.71*	f	f	$\dagger_{-6}$	†	39	**18.47**	f	f	t	t
nug08-3rd	19,728	29,856	12	5833.97	11	259.01	10	**237.02**	f	f	f	f	f	f	7	*257.82*	25	1604.5	30	2864.40
pds-30	49,944	158,489	69	1964.48	72	1105.42	70	788.98	69	1659.21	f	f	103	2014.70	34	**19.93**	98	*446.15*	f	f
pds-40	66,844	217,531	66	4878.49	68	*1551.30*	77	1904.76	67	4012.71	t	t	105	4832.20	34	**31.15**	f	f	100	4777.11
pds-50	83,060	275,814	73	13,860.17	73	*3274.74*	80	3960.55	73	7196.51	t	t	111	11,433.90	38	**49.74**	f	f	f	f
pds-60	99,431	336,421	72	25,592.33	75	*5024.43*	83	7535.99	72	11,609.01	t	t	$\dagger_{-7}$	†	36	**94.28**	f	f	f	f
pds-70	114,944	390,005	80	22,564.32	82	*4980.04*	85	7405.50	84	17,575.97	t	t	126	44,946.8	46	**136.50**	f	f	f	f
pds-80	129,181	434,580	80	25,752.26	83	*6279.08*	86	9853.86	85	21,077.53	t	t	119	58,286.40	42	**157.64**	f	f	f	f
rail507	507	63,516	43	1039.09	51	1138.80	51	475.47	48	14.98	$\dagger_{-7}$	†	34	*7.10*	17	**2.69**	f	f	f	f
rail516	516	47,827	39	496.60	43	700.58	39	536.36	38	11.82	$\dagger_{-7}$	†	19	*3.70*	11	**2.48**	f	f	f	f
rail582	582	56,097	44	1296.56	46	971.35	47	1422.62	41	17.52	$\dagger_{-7}$	†	40	*8.60*	16	**2.43**	f	f	f	f

In each row, boldface and italic denote the fastest and second fastest solvers in CPU time, respectively



(a) Condition number $\kappa(\mathcal{A}\mathcal{A}^{\mathsf{T}})$.

(b) Duality measure μ.

(c) Relative residuals for predictor stage.

(d) Relative residuals for corrector stage.

(e) CPU time for each interior-point step.

(f) Krylov iteration for each interior-point step.

Fig. 3 Numerical results for problem ken_13

the vectors explicitly; CG and MINRES rely on short recurrences, where orthogonality of vectors may be lost due to rounding error. Moreover, GMRES allows using non-symmetric preconditioning while the symmetric solvers require symmetric preconditioning. For example, using SOR preconditioner is cheaper than SSOR for one iteration because the latter goes forwards and backwards. SOR requires 2MV + 3m operations per inner iteration, while SSOR requires 4MV + 6m. In this sense, the GMRES method has more freedom for choosing preconditioners.

From Fig. 3, we may draw a few conclusions. For most problems, the direct solver gave the most efficient result in terms of CPU time. However, for some problems, the direct solver tended to lose accuracy as interior-point iterations proceeded and, in the worst cases, this would inhibit convergence. For problems where the direct method broke down, the proposed inner-iteration preconditioned Krylov subspace methods worked until convergence. With the iterative solvers, it is acceptable to solve (7) and (8) to a moderate level of accuracy in the early phase of the interior-point iterations, and then increase the level of accuracy in the late phase.

4.4 Basis pursuit problems

Most of the problems tested in the last section have a sparse constraint matrix A. The average nonzero density is 2.55%, 0.62%, and 0.45% for the problems in NETLIB, QAPLIB, and MITTELMANN, respectively. However, the matrix can be large and dense for problems such as QP in support vector machine training and linear programming in basis pursuit [11]. The package Atomizer [10] gives such matrices.

In this section, we enrich the experiment by adding problems arising from basis pursuit [11]. We reproduced the ℓ_1-norm optimization problems from the package Atomizer [10], and reformulated them in the standard form of linear programming. The connection between basis pursuit and LP can be found therein. The problems tested in this section have constraint matrices with average nonzero density 48.33% and are usually very well-conditioned, with condition number in the range of (1, 18.54]. The results are shown in Table 5.

The notations have the same meaning as explained in the previous section. Although PDCO's direct solver may be fast for the problems in Table 5, if the problems are given without explicit constraint matrices, one has to use the iterative solver (e.g., LSMR) version. The result shows that only AB-GMRES (NE-SOR) and MOSEK succeeded in solving all the problems. Among these two methods, AB-GMRES (NE-SOR) was faster than MOSEK for the problems bpfig22, bpfig23, bpfig31, and bpfig51.

5 Conclusions

We proposed a new way of preconditioning the normal equations of the second kind arising within interior-point methods for LP problems (11). The resulting interior-point solver is composed of three nested iteration schemes. The outer-most layer is the predictor-corrector interior-point method; the middle layer is the Krylov subspace method for least squares problems, where we may use AB-GMRES, CGNE or MRNE; on top of that, we use a row-scaling scheme that does not incur extra CPU time but helps improving the condition of the system at each interior-point step; the inner-most layer, serving as a preconditioner for the middle

Table 5 Experiments on basis pursuit problems

Problem	m	n	AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		MOSEK		PDCO Direct		PDCO LSMR	
			Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
bpfig22	512	10,240	8	141.24	9	121.68	9	117.46	6	686.02	*14*	*33.19*	**14**	**30.50**
bpfig23	256	4608	*7*	*5.40*	7	10.86	7	9.44	5	120.41	**9**	**3.44**	79	629.77
bpfig24	256	2048	24	124.83	$\dagger_{-2}$	†	$\dagger_{-7}$	†	*18*	*39.85*	**28**	**13.03**	28	42.20
bpfig26	1024	14,336	19	3138.24	$\dagger_{-4}$	†	$\dagger_{-5}$	†	*16*	*1731.40*	**35**	**222.88**	87	14,226.49
bpfig31	512	10,240	8	136.35	9	112.73	9	118.93	6	632.10	*14*	*35.95*	**14**	**31.61**
bpfig32	1024	14,336	20	2016.11	$\dagger_{-4}$	†	$\dagger_{-4}$	†	*20*	*1162.40*	**40**	**227.77**	£	£
bpfig33	1024	22,528	23	2507.16	$\dagger_{-5}$	†	$\dagger_{-5}$	†	*26*	*1846.84*	**41**	**231.69**	£	£
bpfig34	1024	14,336	20	1876.94	$\dagger_{-4}$	†	$\dagger_{-4}$	†	*20*	*1174.63*	**40**	**250.12**	£	£
bpfig41	256	4096	20	391.63	$\dagger_{-4}$	†	$\dagger_{-4}$	†	*24*	*121.53*	**32**	**11.63**	£	£
bpfig51	1024	22,528	*20*	*1048.55*	$\dagger_{-5}$	†	$\dagger_{-5}$	†	16	1741.09	**35**	**219.62**	35	2969.34
bpfig52	256	2048	16	77.93	$\dagger_{-3}$	†	$\dagger_{-2}$	†	*13*	*38.39*	**28**	**9.06**	28	105.90
bpfig53	1024	4096	24	1447.58	$\dagger_{-4}$	†	$\dagger_{-4}$	†	*21*	*156.18*	**41**	**65.68**	£	£
bpfig54	1024	4096	*22*	*1830.62*	$\dagger_{-5}$	†	$\dagger_{-6}$	†	**28**	**168.20**	$\dagger_{-6}$	†	£	£

In each row, boldface and italic denote the fastest and second fastest solvers in CPU time, respectively

layer, is the stationary inner iterations. Among the three layers, only the outer-most one runs towards the required accuracy and the other two are terminated prematurely. The linear systems are solved with a gradually tightened stopping tolerance. We also proposed a new recurrence regarding $\Delta \boldsymbol{w}$ in place of $\Delta \boldsymbol{y}$ to omit one matrix-vector product at each interior-point step. We showed that the use of inner-iteration preconditioners in combination with these techniques enables the efficient interior-point solution of wide-ranging LP problems. We also presented a fairly extensive benchmark test for several renowned solvers including direct and iterative solvers.

The advantage of our method is that it does not break down, even when the matrices become ill-conditioned or (nearly) singular. The method is competitive for large and sparse problems and may also be well-suited to problems in which matrices are too large and dense for direct approaches to work. Extensive numerical experiments showed that our method outperforms the open-source solvers SDPT3, SeDuMi, and PDCO regarding stability and efficiency.

There are several aspects of our method that could be improved. The current implementation of the interior-point method does not use a preprocessing step except for eliminating empty rows and columns. Its efficiency may be improved by adopting some existing preprocessing procedure such as presolve to detect and remove linear dependencies of rows and columns in the constraint matrix. Also, the proposed method could be used in conjunction with more advanced interior-point frameworks such as the multi-corrector interior-point method. In terms of the linear solver, future work is to try reorthogonalization for CG and MINRES and the Householder orthogonalization for GMRES. It is also important to develop preconditioners that only require the action of the operator on a vector, as in huge basis pursuit problems.

It would also be worthwhile to extend our method to problems such as convex QP and SDP.

Acknowledgements We would like to thank the editor and referees for their valuable comments.

References

1. Adler, I., Resende, M.G.C., Veiga, G., Karmarkar, N.: An implementation of Karmarkar's algorithm for linear programming. Math. Program. **44**(1–3), 297–335 (1989). https://doi.org/10.1007/BF01587095
2. Al-Jeiroudi, G., Gondzio, J.: Convergence analysis of the inexact infeasible interior-point method for linear optimization. J. Optim. Theory Appl. **141**(2), 231–247 (2009). https://doi.org/10.1007/s10957-008-9500-5
3. Al-Jeiroudi, G., Gondzio, J., Hall, J.: Preconditioning indefinite systems in interior point methods for large scale linear optimisation. Optim. Method Softw. **23**(3), 345–363 (2008). https://doi.org/10.1080/10556780701535910

4. Andersen, E.D., Andersen, K.D.: Presolving in linear programming. Math. Program. **71**(2), 221–245 (1995). https://doi.org/10.1007/BF01586000
5. Andersen, E.D., Gondzio, J., Mészáros, C., Xu, X.: Implementation of interior-point methods for large scale linear programs. In: Pardalos, P.M., Hearn, D. (eds.) Interior Point Methods of Mathematical Programming, App. Optim., vol. 5. Kluwer Academic Publishers, Dordrecht (1996). https://doi.org/10.1007/978-1-4613-3449-1_6
6. Bergamaschi, L., Gondzio, J., Venturin, M., Zilli, G.: Inexact constraint preconditioners for linear systems arising in interior point methods. Comput. Optim. Appl. **36**(2–3), 137–147 (2007). https://doi.org/10.1007/s10589-006-9001-0
7. Bergamaschi, L., Gondzio, J., Zilli, G.: Preconditioning indefinite systems in interior point methods for optimization. Comput. Optim. Appl. **28**(2), 149–171 (2004). https://doi.org/10.1023/B:COAP.0000026882.34332.1b
8. Björck, Å., Elfving, T.: Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. BIT **19**, 145–163 (1979). https://doi.org/10.1007/BF01930845
9. Carpenter, T.J., Shanno, D.F.: An interior point method for quadratic programs based on conjugate projected gradients. Comput. Optim. Appl. **2**(1), 5–28 (1993). https://doi.org/10.1007/BF01299140
10. Chen, S., Donoho, D., Saunders, M.: About atomizer (2000). http://sparselab.stanford.edu/atomizer/. Accessed 27 May 2019
11. Chen, S., Donoho, D., Saunders, M.: Atomic decomposition by basis pursuit. SIAM Rev. **43**(1), 129–159 (2001). https://doi.org/10.1137/S003614450037906X
12. Chin, P., Vannelli, A.: PCG techniques for interior point algorithms. In: Proceedings of the 36th Midwest Symposium on Circuits and Systems, pp. 200–203. IEEE (1994). https://doi.org/10.1109/MWSCAS.1993.343095
13. Craig, E.J.: The N-step iteration procedures. J. Math. Phys. **34**, 64–73 (1955). https://doi.org/10.1002/sapm195534164
14. Cui, X.: Approximate Generalized Inverse Preconditioning Methods for Least Squares Problems. Ph.D. thesis, The Graduate University for Advanced Studies, Japan (2009). http://id.nii.ac.jp/1013/00001492/
15. Cui, X., Hayami, K., Yin, J.F.: Greville's method for preconditioning least squares problems. Adv. Comput. Math. **35**, 243–269 (2011). https://doi.org/10.1007/s10444-011-9171-x
16. Cui, Y., Morikuni, K., Tsuchiya, T., Hayami, K.: Implementation of interior-point methods for LP based on Krylov subspace iterative solvers with inner-iteration preconditioning, pp. 1–30. arXiv prepr. arXiv:1604.07491 (2016)
17. Czyzyk, J., Mehrotra, S., Wagner, M., Wright, S.J.: PCx: An interior-point code for linear programming. Optim. Methods Softw. **11**(1–4), 397–430 (1999). https://doi.org/10.1080/10556789908805757
18. D'Apuzzo, M., De Simone, V., Di Serafino, D.: On mutual impact of numerical linear algebra and large-scale optimization with focus on interior point methods. Comput. Optim. Appl. **45**(2), 283–310 (2010). https://doi.org/10.1007/s10589-008-9226-1
19. Davis, T.A.: CSparse: A concise sparse matrix package. Version 3.1.4 (2014). http://www.suitesparse.com. Accessed 27 May 2019
20. Davis, T.A., Hu, Y.: The university of Florida sparse matrix collection. ACM Trans. Math. Softw. **38**(1), 1:1–1:25 (2011). https://doi.org/10.1145/2049662.2049663
21. Dax, A.: The convergence of linear stationary iterative processes for solving singular unstructured systems of linear equations. SIAM Rev. **32**, 611–635 (1990). https://doi.org/10.1137/1032122
22. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002). https://doi.org/10.1007/s101070100263
23. Duff, I.S.: MA57—A new code for the solution of sparse symmetric definite systems. ACM Trans. Math. Softw. **30**(2), 118–144 (2004). https://doi.org/10.1145/992200.992202
24. Ferris, M.C., Munson, T.S.: Interior-point methods for massive support vector machines. SIAM J. Optim. **13**(3), 783–804 (2002). https://doi.org/10.1137/S1052623400374379
25. Fourer, R., Mehrotra, S.: Solving symmetric indefinite systems in an interior-point method for linear programming. Math. Program. **62**(1–3), 15–39 (1993). https://doi.org/10.1007/BF01585158
26. Freund, R.W., Jarre, F.: A QMR-based interior-point algorithm for solving linear programs. Math. Program. **76**(1), 183–210 (1997). https://doi.org/10.1007/BF02614383
27. Freund, R.W., Jarre, F., Mizuno, S.: Convergence of a class of inexact interior-point algorithms for linear programs. Math. Oper. Res. **24**(1), 50–71 (1999). https://doi.org/10.1287/moor.24.1.50

28. Gill, P.E., Murray, W., Saunders, M.A., Tomlin, J.A., Wright, M.H.: On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. Math. Program. **36**(2), 183–209 (1986). https://doi.org/10.1007/BF02592025
29. Gondzio, J.: HOPDM (version 2.12)—a fast LP solver based on a primal-dual interior point method. Eur. J. Oper. Res. **85**(1), 221–225 (1995). https://doi.org/10.1016/0377-2217(95)00163-K
30. Gondzio, J.: Multiple centrality corrections in a primal-dual method for linear programming. Comput. Optim. Appl. **6**(2), 137–156 (1996). https://doi.org/10.1007/BF00249643
31. Gondzio, J.: Presolve analysis of linear programs prior to applying an interior point method. INFORMS J. Comput. **9**(1), 73–91 (1997). https://doi.org/10.1287/ijoc.9.1.73
32. Gondzio, J.: Interior point methods 25 years later. Eur. J. Oper. Res. **218**(3), 587–601 (2012). https://doi.org/10.1016/j.ejor.2011.09.017
33. Gondzio, J.: Matrix-free interior point method. Comput. Optim. Appl. **51**(2), 457–480 (2012). https://doi.org/10.1007/s10589-010-9361-3
34. Gondzio, J., Terlaky, T.: A computational view of interior point methods. In: Beasley, J.E. (ed.) Advances in Linear and Integer Programming, pp. 103–144. Oxford University Press, Oxford (1996)
35. Grant, M., Boyd, S.: CVX: Matlab software for Disciplined Convex Programming. Version 2.1. (2014). http://cvxr.com/cvx. Accessed 27 May 2019
36. Grant, M.C., Boyd, S.P.: Graph implementations for nonsmooth convex programs. In: Blondel, V., Boyd, S., Kimura, H. (eds.) Recent Advances in Learning and Control, Lecture Notes in Control and Information Sciences, pp. 95–110. Springer, Berlin (2008). https://doi.org/10.1007/978-1-84800-155-8_7
37. Hayami, K., Yin, J.F., Ito, T.: GMRES methods for least squares problems. SIAM J. Matrix Anal. Appl. **31**, 2400–2430 (2010). https://doi.org/10.1137/070696313
38. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. **49**(6), 409–436 (1952). https://doi.org/10.6028/jres.049.044
39. Júdice, J.J., Patricio, J., Portugal, L.F., Resende, M.G.C., Veiga, G.: A study of preconditioners for network interior point methods. Comput. Optim. Appl. **24**(1), 5–35 (2003). https://doi.org/10.1023/A:1021882330897
40. Karmarkar, N., Ramakrishnan, K.: Computational results of an interior point algorithm for large scale linear programming. Math. Program. **52**(1–3), 555–586 (1991). https://doi.org/10.1007/BF01582905
41. Kojima, M., Mizuno, S., Yoshise, A.: A polynomial-time algorithm for a class of linear complementarity problems. Math. Program. **4**(1–3), 1–26 (1989). https://doi.org/10.1007/BF01587074
42. Korzak, J.: Convergence analysis of inexact infeasible-interior-point algorithms for solving linear programming problems. SIAM J. Optim. **11**(1), 133–148 (2000). https://doi.org/10.1137/S1052623497329993
43. Lustig, I.J., Marsten, R.E., Shanno, D.F.: On implementing Mehrotra's predictor-corrector interior-point method for linear programming. SIAM J. Optim. **2**(3), 435–449 (1992). https://doi.org/10.1137/0802022
44. Lustig, I.J., Marsten, R.E., Shanno, D.F.: Interior point methods for linear programming: computational state of the art. ORSA J. Comput. **6**(1), 1–14 (1994). https://doi.org/10.1287/ijoc.6.1.1
45. Mehrotra, S.: Implementations of affine scaling methods: approximate solutions of systems of linear equations using preconditioned conjugate gradient methods. ORSA J. Comput. **4**(2), 103–118 (1992). https://doi.org/10.1287/ijoc.4.2.103
46. Mehrotra, S.: On the implementation of a primal-dual interior point method. SIAM J. Optim. **2**(4), 575–601 (1992). https://doi.org/10.1137/0802028
47. Mehrotra, S., Li, Z.: Convergence conditions and Krylov subspace-based corrections for primalual interior-point method. SIAM J. Optim. **15**(3), 635–653 (2005). https://doi.org/10.1137/S1052623403431494
48. Mehrotra, S., Wang, J.: Conjugate gradient based implementation of interior point methods for network flow problems. In: Adams, L., Nazareth, J. (eds.) Linear and nonlinear conjugate gradient-related methods, pp. 124–142. SIAM, Philadelphia (1996)
49. Monteiro, R.D.C., Adler, I.: Interior path following primal-dual algorithms. Part I: linear programming. Math. Program. **44**(1–3), 27–41 (1989). https://doi.org/10.1007/BF01587075
50. Monteiro, R.D.C., O'Neal, J.W.: Convergence analysis of a long-step primal-dual infeasible interior-point LP algorithm based on iterative linear solvers. Technical report, Georgia Institute of Technology (2003). http://www.optimization-online.org/DB_FILE/2003/10/768.pdf

51. Monteiro, R.D.C., O'Neal, J.W., Tsuchiya, T.: Uniform boundedness of a preconditioned normal matrix used in interior-point methods. SIAM J. Optim. **15**(1), 96–100 (2004). https://doi.org/10.1137/S1052623403426398
52. Morikuni, K.: Symmetric inner-iteration preconditioning for rank-deficient least squares problems, pp. 1–15. arXiv prepr. arXiv:1504.00889 (2015)
53. Morikuni, K.: Inner-iteration preconditioning with symmetric splitting matrices for symmetric singular linear systems. Trans. JSIAM **29**(1), 62–77 (2019). https://doi.org/10.11540/jsiamt.29.1_62
54. Morikuni, K., Hayami, K.: Inner-iteration Krylov subspace methods for least squares problems. SIAM J. Matrix Anal. Appl. **34**(1), 1–22 (2013). https://doi.org/10.1137/110828472
55. Morikuni, K., Hayami, K.: Convergence of inner-iteration GMRES methods for rank-deficient least squares problems. SIAM J. Matrix Anal. Appl. **36**(1), 225–250 (2015). https://doi.org/10.1137/130946009
56. Morikuni, K., Hayami, K.: Matlab-MEX Codes of the AB-GMRES Method Preconditioned by NE-SOR Inner Iterations. http://researchmap.jp/KeiichiMorikuni/Implementations/. Accessed 27 May 2019
57. MOSEK ApS: The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 63). (2015). https://docs.mosek.com/7.1/toolbox.pdf. Accessed 27 May 2019
58. Oliveira, A.R.L., Sorensen, D.C.: A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. Linear Algebra Appl. **394**, 1–24 (2005). https://doi.org/10.1016/j.laa.2004.08.019
59. Paige, C.C., Saunders, M.A.: Solution of sparse indefinite systems of linear equations. SIAM J. Numer. Anal. **12**, 617–629 (1975). https://doi.org/10.1137/0712047
60. Portugal, L.F., Resende, M.G.C., Veiga, G., Júdice, J.J.: A truncated primal-infeasible dual-feasible network interior point method. Networks **35**(2), 91–108 (2000). https://doi.org/10.1002/(SICI)1097-0037(200003)35:2h91::AID-NET1i3.0.CO;2-T
61. Rees, T., Greif, C.: A preconditioner for linear systems arising from interior point optimization methods. SIAM J. Sci. Comput. **29**(5), 1992–2007 (2007). https://doi.org/10.1137/060661673
62. Resende, M.G.C., Veiga, G.: An implementation of the dual affine scaling algorithm for minimum-cost flow on bipartite uncapacitated networks. SIAM J. Optim. **3**(3), 516–537 (1993). https://doi.org/10.1137/0803025
63. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. SIAM, Philadelphia (2003). https://doi.org/10.1137/1.9780898718003
64. Saad, Y., Schultz, M.H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986). https://doi.org/10.1137/0907058
65. Saunders, M.A., Kim, B., Maes, C., Akle, S., Zahr, M.: PDCO: Primal-dual interior method for convex objectives (2013). https://web.stanford.edu/group/SOL/software/pdco/. Accessed 27 May 2019
66. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. **11**, 625–633 (1999). https://doi.org/10.1080/10556789908805766
67. Tanabe, K.: Centered Newton method for mathematical programming. In: System Modeling and Optimization, Lecture Notes in Control and Information Sciences, Vol. 113, pp. 197–206. Springer, Berlin (1988). https://doi.org/10.1007/BFb0042787
68. Toh, K.C., Todd, M.J., Tütüncü, R.H.: SDPT3—a Matlab software package for semidefinite programming. Optim. Methods Softw. **11**, 545–581 (1999). https://doi.org/10.1080/10556789908805762
69. Tütüncü, R.H., Toh, K.C., Todd, M.J.: Solving semidefinite-quadratic-linear programs using SDPT3. Math. Program. Ser. B **95**, 189–217 (2003). https://doi.org/10.1007/s10107-002-0347-5
70. Wang, W., O'leary, D.P.: Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming. Numer. Algorithms **25**(1–4), 387–406 (2000). https://doi.org/10.1023/A:1016614603137
71. Wright, S.J.: Primal-Dual Interior-Point Methods. SIAM, Philadelphia (1997). https://doi.org/10.1137/1.9781611971453
72. Wright, S.J.: Modified Cholesky factorizations in interior-point algorithms for linear programming. SIAM J. Optim. **9**(4), 1159–1191 (1999). https://doi.org/10.1137/S1052623496304712
73. Zhang, Y.: On the convergence of a class of infeasible interior-point methods for the horizontal linear complementary problem. SIAM J. Optim. **4**(1), 208–227 (1994). https://doi.org/10.1137/0804012
74. Zhang, Y.: Solving large-scale linear programs by interior-point methods under the Matlab environment. Optim. Methods Softw. **10**(1), 1–31 (1998). https://doi.org/10.1080/10556789808805699

Affiliations

Yiran Cui[1] · Keiichi Morikuni[2] · Takashi Tsuchiya[3] · Ken Hayami[4]

Keiichi Morikuni
morikuni@cs.tsukuba.ac.jp

Takashi Tsuchiya
tsuchiya@grips.ac.jp

Ken Hayami
hayami@nii.ac.jp

1 Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK

2 Division of Information Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-0006, Japan

3 National Graduate Institute for Policy Studies, 7-22-1 Roppongi, Minato-ku, Tokyo 106-8677, Japan

4 National Institute of Informatics and SOKENDAI (The Graduate University for Advanced Studies), 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan