ISSN 1346-5597

National Institute of Informatics

NII Technical Report

Implementation of Interior-Point Methods for LP Based on Krylov Subspace Iterative Solvers with Inner-Iteration Preconditioning

Yiran Cui, Keiichi Morikuni, Takashi Tsuchiya, and Ken Hayami

NII-2016-003E
May 2016

URL: http://www.nii.ac.jp/ Tokyo 101-8430, JAPAN +81-3-4212-2000 FAX: +81-3-3556-1916

IMPLEMENTATION OF INTERIOR-POINT METHODS FOR LP BASED ON KRYLOV SUBSPACE ITERATIVE SOLVERS WITH INNER-ITERATION PRECONDITIONING*

YIRAN CUI[†], KEIICHI MORIKUNI[‡], TAKASHI TSUCHIYA[§], AND KEN HAYAMI[¶]

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 employed 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. Extensive numerical experiments are conducted over diverse instances of 125 LP problems including Netlib, QAPLIB, and Mittelmann's collections. The number of variables of the largest problem is 434,580. It turns out that our implementation is more stable and robust than the standard public domain solvers SeDuMi (Self-Dual Minimization) and SDPT3 (Semidefinite Programming Toh-Todd-Tütüncü) without increasing CPU time. As far as we know, this is the first result that an interior-point method entirely based on iterative solvers succeed in solving a fairly large number of standard LP instances from benchmark libraries under the standard stopping criteria.

Key words. linear programming problems, interior-point methods, inner-iteration preconditioning, Krylov subspace methods

AMS subject classifications. 90C51, 90C05, 65F10

1. Introduction.

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

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

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

where $A \in \mathbb{R}^{m\times n}$, $m \leq n$, and we assume the existence of an optimal solution to this problem. In this paper, we deal with 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 the system of linear equations to compute the search direction. In spite of the fact that there are two known approaches for this, i.e., the direct method and the iterative method, the direct method is the only choice so far and we do not find any known implementation solely depending on an iterative method. This is because the system of linear equations to be solved becomes notoriously ill-conditioned towards the end of interior-point iterations and no iterative solver has managed to resolve this ill-conditioning problem.

*May 9, 2016

[†]Department of Computer Science, University College London, Gower Street, London WC1E 6BT, United Kingdom (y.cui.12@ucl.ac.uk).

[‡]Division of Information Engineering, University of Tsukuba, Tenoudai 1-1-1, Tsukuba, Ibaraki 305-8573, Japan (morikuni@cs.tsukuba.ac.jp).

[§]National Graduate Institute for Policy Studies, 7-22-1 Roppongi, Minato, Tokyo 106-8677, Japan (tsuchiya@grips.ac.jp). This author is supported in part by the Grant-in-Aid for Scientific Research (B) 15H02968 from the Japan Society for the Promotion of Sciences.

[¶]National Institute of Informatics, SOKENDAI (The Graduate University for Advanced Studies), 2-1-2 Hitotsubashi, Chiyoda, Tokyo 101-0003, Japan (hayami@nii.ac.jp). This author is supported in part by the Grant-in-Aid for Scientific Research (C) 15K04768 from the Japan Society for the Promotion of Sciences.

To overcome this difficulty, we apply novel inner-iteration preconditioned Krylov subspace methods for least squares problems. The inner-iteration preconditioners recently proposed by Morikuni and Hayami [47, 48] enable us to deal with the severe ill-conditioning of the system of linear equations. Furthermore, the employed Krylov subspace methods do not suffer from rank-deficiency and therefore no preprocessing is necessary even if rows of A are not linearly independent.

Extensive numerical experiments were conducted over diverse instances of 125 LP problems taken from the standard benchmark libraries including Netlib, QAPLIB, and Mittelmann's collections. The number of variables of the largest problem is 434,580. It turns out that our implementation is more stable and robust than the standard public domain solvers SeDuMi (Self-Dual Minimization) [56] and SDPT3 (Semidefinite Programming Toh-Todd-Tütüncü) [58, 59] without increasing CPU time. As far as the authors know, this is the first result where an interior-point method entirely based on iterative solvers succeed in solving a fairly large number of standard LP instances from the benchmark libraries with standard stopping criteria. Our implementation is yet considerably slower than MOSEK [49], one of the state-of-the-art commercial interior-point LP solvers, though it is competitive in robustness and stability. On the other hand, we observed that our implementation is able to solve ill-conditioned dense problems with severe rank deficiency which the interior-point solver of MOSEK can not solve.

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 previous iterative solvers for the interior-point method. The interior-point method that we deal with is an infeasible primal-dual predictor-corrector interior-point method. This is one of the state-of-the-art interior-point methods which evolved from the original primal-dual interior-point method [57, 35, 43, 60] incorporating several innovative ideas, e.g., [62, 39].

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

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

$$A\boldsymbol{x}^{(*)} = \boldsymbol{b}, \tag{2b}$$

$$X^{(*)} S^{(*)} \boldsymbol{e} = \boldsymbol{0}, \tag{2c}$$

$$\boldsymbol{x}^{(*)} \geq \boldsymbol{0}, \quad \boldsymbol{s}^{(*)} \geq \boldsymbol{0}, \tag{2d}$$

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

The infeasible primal-dual interior-point method gives the following system 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 approximate solutions to (3) repeatedly. Along with the progress of iterations, μ is decreased to zero. In this way (2) is satisfied within some tolerance level at the solution point. Therefore, the search direction at each interior-point step is obtained by solving the Newton's equations

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

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, $\mu := \boldsymbol{x}^\mathsf{T}\boldsymbol{s}/n$ 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 Newton's equations (4) are 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 proper step length to ensure the nonnegtivity requirement (2d), and then reduce μ by σ to seek for the next solution by solving (4) again.

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

$$
\begin{bmatrix} X^{-1}S & A^\mathsf{T} \\ A & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ -\Delta \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{c} + A^\mathsf{T}\boldsymbol{y} + \sigma\mu X^{-1}\boldsymbol{e} \\ \boldsymbol{r}_\mathrm{p} \end{bmatrix}, \tag{5}
$$

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

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

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 coefficient matrix of the normal equations (6) is not symmetric positive definite when any of the following cases is encountered. Firstly, when the constraint matrix A is rank-deficient, the coefficient matrix of (6) is singular. There exist presolving techniques that can detect and remove the linear dependent rows in A, see, e.g., [2, 24]. Secondly, in the late phase of the interior-point iterations, the diagonal matrix XS^{-1} has both very tiny and very large diagonal values as a result of convergence. Thus, the coefficient matrix may become positive semidefinite, or even slightly indefinite, due to rounding error. In particular, the situation becomes harsh when primal degeneracy occurs at the optimal solution. One can refer to [27, 63] for more detailed explanations.

Thus, when direct methods that compute symmetric factorization, such as the Cholesky decomposition, are applied to (6), some diagonal pivots encountered during decomposition can be zero or negative, causing the algorithm to break down. Most direct methods adopt a strategy of replacing the problematic pivot with a very large number. See, e.g., [63] for the Cholesky-Infinity factorization which is specially designed to solve (6) when it has a positive semidefinite, but not definite, coefficient matrix. Numerical experience (see, e.g., [1, 37, 19, 38, 3, 61, 12]) indicates that direct methods provide sufficiently accurate solutions for the interior-point methods to converge regardless of the ill-conditioning of the coefficient matrix. However, as the LP problems become larger, the significant fill-ins in decompositions make direct methods prohibitively expensive. It is stated in [25] that the fill-ins are observed even for very sparse matrices. Moreover, the coefficient matrix can be dense such as in quadratic programs arising in support vector machine training [18] or linear programs arising in basis pursuit [7], and even when A is given as a sparse matrix, $AXS^{-1}A^\mathsf{T}$ 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 problems including interior-point methods.

These drawbacks of the direct methods and the progress in preconditioning techniques motivate researchers to develop stable iterative methods for solving (6) or alternatively (5). The major problem in iteratively solving (6) or (5) is that as the interior-point iterations proceed, the condition number of the term XS^{-1} increases and their solution tends to be expensive. The ill-conditioning property makes 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 [8, 20, 21, 4, 50] as literature related to preconditioners for (5).

For the iterative solution of the normal equations (6), the conjugate gradient (CG) method [31] was applied together with diagonal scaling preconditioners [6, 52, 36], or incomplete Cholesky factorization preconditioners [39, 34, 8, 42]. The LSQR method preconditioned by an approximation to the coefficient matrix was used in [22]. A matrix-free method of using the CG for least squares (CGLS) preconditioned by a partial Cholesky decomposition was proposed in [26]. In [10], a preconditioner based on Greville's method [11] 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 [53, 32, 44, 45]. One may refer to [13] 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) by using Krylov subspace methods preconditioned by stationary inner-iterations recently proposed for least squares problems in [30, 47, 48]. In section 2, we briefly describe the framework of Mehrotra's predictor-corrector interior-point algorithm and the normal equations arising from this algorithm. In section 3, we specify the application of our method to the normal equations. In section 4, we present numerical results in comparison with a modified sparse Cholesky method and three direct solvers in CVX, a major public package for specifying and solving convex programs [29, 28]. In section 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., $\boldsymbol{x}^{\langle k\rangle}$. $\mathcal{R}(\cdot)$ denotes the range space of a matrix. $\kappa(A)$ denotes the condition number of matrix A, i.e., $\kappa(A) = \sigma_1(A)/\sigma_r(A)$ where $\sigma_1(A)$ and $\sigma_r(A)$ denote the maximum and minimum nonzero singular values of A. $\mathcal{K}_k(A, \boldsymbol{b}) = \operatorname{span}\{\boldsymbol{b}, A\boldsymbol{b}, \ldots, A^{k-1}\boldsymbol{b}\}$ 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 [40] which has been established as a standard in this area [37, 38, 60, 41]. Note that our method is not limited to this version of interior-point method and can also be applied to other interior-point methods (see, e.g., [60] for more interior-point methods) whose directions are computed via the normal equations (6).

2.1. Mehrotra's predictor-corrector algorithm.

In Mehrotra's predictor-corrector 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 to a small value. Otherwise, σ is assigned to 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 $(\Delta \boldsymbol{x}_{\mathrm{cc}}, \Delta \boldsymbol{y}_{\mathrm{cc}}, \Delta \boldsymbol{s}_{\mathrm{cc}})$

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

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.

Algorithm 1 Mehrotra's predictor-corrector algorithm.

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

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

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

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

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

In the same line, the parameter σ is chosen as $\sigma = \min\left(0.208, \left(\mu_{\mathrm{af}} / \mu^{(k)}\right)^2\right)$ in the early phase of the interior-point iterations. In the late phase of the interior-point iterations, the parameter σ is chosen approximately as 10 times the order of the error measure which is defined in (19). Here the distinction between *early* and *late* phases is in the error measure 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 $\hat{\alpha}_{\mathrm{p}}$, $\hat{\alpha}_{\mathrm{d}}$ which 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 $\boldsymbol{y}^{(k)}$. Let $\mathcal{A} := AS^{-1/2}X^{1/2}$. Since we assume the existence of an optimal solution to problem (1), we have $\boldsymbol{b} \in \mathcal{R}(A)$. Then, the problem in the predictor stage (7) is equivalent to

$$\mathcal{A}\mathcal{A}^{\mathsf{T}} \Delta \boldsymbol{y}_{\mathrm{af}} = \boldsymbol{b} + AS^{-1}X\boldsymbol{r}_{\mathrm{d}}, \tag{10a}$$

$$\Delta \boldsymbol{s}_{\mathrm{af}} = \boldsymbol{r}_{\mathrm{d}} - A^{\mathsf{T}} \Delta \boldsymbol{y}_{\mathrm{af}}, \tag{10b}$$

$$\Delta \boldsymbol{x}_{\mathrm{af}} = -S^{-1}X\Delta \boldsymbol{s}_{\mathrm{af}} - \boldsymbol{x}. \tag{10c}$$

The equations (10a) with $\Delta \boldsymbol{w}_{\mathrm{af}} := \mathcal{A}^{\mathsf{T}} \Delta \boldsymbol{y}_{\mathrm{af}}$ (the normal equations of the second kind) are equivalent to

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

where $\boldsymbol{f}_{\mathrm{af}} := \boldsymbol{b} + AS^{-1}X\boldsymbol{r}_{\mathrm{d}}$. Then, (10b) can be computed by $\Delta \boldsymbol{s}_{\mathrm{af}} = \boldsymbol{r}_{\mathrm{d}} - S^{1/2}X^{-1/2}\Delta \boldsymbol{w}_{\mathrm{af}}$.

On the other hand, the problem in the corrector stage (8) is equivalent to

$$\mathcal{A}\mathcal{A}^{\mathsf{T}} \Delta \boldsymbol{y}_{\mathrm{cc}} = AS^{-1}\Delta X_{\mathrm{af}} \Delta S_{\mathrm{af}} \boldsymbol{e} - \sigma\mu AS^{-1}\boldsymbol{e}, \tag{12a}$$

$$\Delta \boldsymbol{s}_{\mathrm{cc}} = -A^{\mathsf{T}} \Delta \boldsymbol{y}_{\mathrm{cc}}, \tag{12b}$$

$$\Delta \boldsymbol{x}_{\mathrm{cc}} = -S^{-1}X\Delta \boldsymbol{s}_{\mathrm{cc}} - S^{-1}\Delta X_{\mathrm{af}} \Delta S_{\mathrm{af}} + \sigma\mu S^{-1}\boldsymbol{e}. \tag{12c}$$

The equations (12a) with $\Delta \boldsymbol{w}_{\mathrm{cc}} := \mathcal{A}^{\mathsf{T}} \Delta \boldsymbol{y}_{\mathrm{cc}}$ are equivalent to

$$\min \|\Delta \boldsymbol{w}_{\mathrm{cc}}\|_2, \quad \text{subject to} \quad \mathcal{A}\Delta \boldsymbol{w}_{\mathrm{cc}} = \boldsymbol{f}_{\mathrm{cc}}, \tag{13}$$

where $\boldsymbol{f}_{\mathrm{cc}} = AS^{-1}\Delta X_{\mathrm{af}} \Delta S_{\mathrm{af}} \boldsymbol{e} - \sigma\mu AS^{-1}\boldsymbol{e}$. Then, (12b) can be computed by $\Delta \boldsymbol{s}_{\mathrm{cc}} = -S^{1/2}X^{-1/2}\Delta \boldsymbol{w}_{\mathrm{cc}}$.

Thus, by solving (11) and (13) instead of (10a) and (12a), 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 one matrix-vector product in (10b) and another in (12b) if a predictor step is performed per interior-point iteration. Note that in the predictor and corrector stages, the problems (11) and (13) have the same coefficient matrix but different right-hand sides. We will introduce methods for solving (11) and (13) in the next section.

3. Application of inner-iteration preconditioned Krylov subspace methods. In lines 4 and 10 in Algorithm 1, the linear systems (10a) and (12a), and hence (11) and (13) need to be solved. The coefficient matrices of (11) and (13) become increasingly ill-conditioned as the interior-point iterations proceed. In this section, we focus on applying inner-iteration preconditioned Krylov subspace methods to (11) and (13), since they are advantageous in dealing with ill-conditioned sparse matrices. The methods to be discussed are the preconditioned CG and MINRES methods [31, 51] applied to the normal equations of the second kind ((P)CGNE and (P)MRNE, respectively) [9, 48], and the right-preconditioned generalized minimal residual method (AB-GMRES) [30, 48].

Firstly, the conjugate gradient (CG) method [31] is an iterative method for solving linear systems of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} \in \mathbf{R}^{n\times n}$ is a symmetric and positive (semi)definite coefficient 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 $\|\mathbf{x}_k - \mathbf{x}_*\|^2_{\mathbf{A}}$ over the space $\mathbf{x}_0 + \mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$, where $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{x}_*$ is a solution of $\mathbf{A}\mathbf{x} = \mathbf{b}$, and $\|\mathbf{x}_k - \mathbf{x}_*\|^2_{\mathbf{A}} := (\mathbf{x}_k - \mathbf{x}_*)^\mathsf{T}\mathbf{A}(\mathbf{x}_k - \mathbf{x}_*)$.

Secondly, the MINRES method [51] is another iterative method for solving linear systems of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} \in \mathbf{R}^{n\times n}$ is a symmetric coefficient matrix. MINRES with $\mathbf{x}_0$ determines the kth iterate $\mathbf{x}_k$ by minimizing $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2$ over the same space as CG.

Thirdly, the generalized minimal residual (GMRES) method [55] is an iterative method for solving linear systems of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} \in \mathbf{R}^{n\times n}$ is a square matrix. GMRES with $\mathbf{x}_0$ determines the kth iterate $\mathbf{x}_k$ by minimizing $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2$ 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 these systems 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) and (13) 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. We give the expressions for the inner-iteration preconditioning and preconditioned matrices. 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 $C^{(\ell)} = \sum_{i=0}^{\ell-1} H^i M^{-1}$ and $\mathcal{A}\mathcal{A}^\mathsf{T}C^{(\ell)} = M\sum_{i=0}^{\ell-1}(I - H)H^iM^{-1} = M(I - H^\ell)M^{-1}$, respectively. If $C^{(\ell)}$ is nonsingular, then $\mathcal{A}\mathcal{A}^\mathsf{T}C^{(\ell)}\boldsymbol{u} = \boldsymbol{f}$, $\boldsymbol{z} = C^{(\ell)}\boldsymbol{u}$ 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, $C^{(\ell)}$ is symmetric and positive definite (SPD) if and only if the inner-iteration splitting maxrit M is SPD [46, Theorem 2.8]. For ℓ even, $C^{(\ell)}$ is SPD if and only if the inner-iteration splitting matrix $M + N$ is SPD [46, Theorem 2.8]. We give algorithms for CGNE and MRNE preconditioned by inner iterations [48, Algorithms E.3, E.4].

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 $\mathcal{A}\Delta\boldsymbol{w}_{\mathrm{af}} = \mathbf{f}_{\mathrm{af}}$ 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 which does not necessarily have to be $\mathcal{A}^\mathsf{T}$. Let $\mathcal{B} \in \mathbb{R}^{n\times m}$ be a preconditioning

Algorithm 2 CGNE method preconditioned by inner iterations.

Let $\Delta \boldsymbol{w}_0$ be the initial approximate solution, and $\boldsymbol{g}_0 := \boldsymbol{f} - \mathcal{A}\Delta \boldsymbol{w}_0$.
Apply ℓ 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 $\boldsymbol{u}_0 := \mathcal{A}^\mathsf{T}\boldsymbol{z}_0$.
$\boldsymbol{s}_0 := \boldsymbol{u}_0, \gamma_0 := (\boldsymbol{g}_0, \boldsymbol{z}_0)$
for $k = 0, 1, 2, \dots$ until convergence, **do**
  $\alpha_k := \gamma_k/(\boldsymbol{s}_k, \boldsymbol{s}_k),\ \ \Delta \boldsymbol{w}_{k+1} := \Delta \boldsymbol{w}_k + \alpha \boldsymbol{s}_k,\ \ \boldsymbol{g}_{k+1} := \boldsymbol{g}_k - \alpha_k \mathcal{A}\boldsymbol{s}_k$
  Apply ℓ 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 $\boldsymbol{u}_{k+1} := \mathcal{A}^\mathsf{T}\boldsymbol{z}_{k+1}$.
  $\gamma_{k+1} := (\boldsymbol{g}_{k+1}, \boldsymbol{z}_{k+1}),\ \ \beta_k := \gamma_{k+1}/\gamma_k,\ \ \boldsymbol{s}_{k+1} := \boldsymbol{u}_{k+1} + \beta_k \boldsymbol{s}_k$
end for

Algorithm 3 MRNE method preconditioned by inner iterations.

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

matrix for $\mathcal{A}$. Then, AB-GMRES corresponds to GMRES [55] applied to

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

which is equivalent to the minimum-norm solution to the problem

$$\min \|\Delta \boldsymbol{w}\|_2, \quad \text{subject to} \quad \mathcal{A}\Delta \boldsymbol{w} = \boldsymbol{f} \tag{14}$$

for all $\boldsymbol{f} \in \mathcal{R}(\mathcal{A})$ if $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A}^\mathsf{T})$ [48, 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 $\boldsymbol{z}_k = \operatorname{argmin}_{\boldsymbol{z} \in \boldsymbol{z}_0 + \mathcal{K}_k(\mathcal{A}\mathcal{B}, \boldsymbol{g}_0)} \|\boldsymbol{f} - \mathcal{A}\mathcal{B}\boldsymbol{z}\|_2$, 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 [47, 48] to (14). This method was shown to outperform previous methods when dealing with ill-conditioned and rank-deficient problems. We give the expressions for the inner-iteration preconditioning and preconditioned matrices. Let M be a 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 $H = M^{-1}N$. Letting $C^{(\ell)} = \sum_{i=0}^{\ell-1} H^i M^{-1}$, the inner-iteration preconditioning and preconditioned matrices are $\mathcal{B}^{(\ell)} = \mathcal{A}^\mathsf{T} C^{(\ell)}$ and $\mathcal{A}\mathcal{B}^{(\ell)} = \sum_{i=0}^{\ell-1} (I - H) H^i = M(I - H^\ell)M^{-1}$, respectively. If the inner-iteration matrix H is semiconvergent, i.e., $\lim_{i \to \infty} H^i$ exists, then AB-GMRES 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 $\Delta \boldsymbol{w}_0 \in \mathcal{R}(\mathcal{A}^\mathsf{T})$ [48, Theorem 5.5]. The inner-iteration preconditioning matrix $\mathcal{B}^{(\ell)}$ works on $\mathcal{A}$ in AB-GMRES as in the following algorithm [48, Algorithm 5.1].

Algorithm 4 AB-GMRES method preconditioned by inner iterations.

Let $\Delta \boldsymbol{w}_0 \in \mathbb{R}^n$ be the initial approximate solution, and $\boldsymbol{g}_0 := \boldsymbol{f} - \mathcal{A}\Delta \boldsymbol{w}_0$.
$\beta := \|\boldsymbol{g}_0\|_2, \;\; \boldsymbol{v}_1 := \boldsymbol{r}_0/\beta$
for $k = 1, 2, \dots$ until convergence, **do**
  Apply ℓ 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 $\boldsymbol{z}_k := \mathcal{B}^{\langle \ell \rangle}\boldsymbol{v}_k$.
  $\boldsymbol{u}_k := \mathcal{A}\boldsymbol{z}_k$
  for $i = 1, 2, \dots, k$, **do**
    $h_{i,k} := (\boldsymbol{u}_k, \boldsymbol{v}_i), \; \boldsymbol{u}_k := \boldsymbol{u}_k - h_{i,k}\boldsymbol{v}_i$
  end for
  $h_{k+1,k} := \|\boldsymbol{u}_k\|_2, \; \boldsymbol{v}_{k+1} := \boldsymbol{u}_k / h_{k+1,k}$
end for
$\boldsymbol{p}_k := \arg\min_{\boldsymbol{p}\in R^k} \|\beta \boldsymbol{e_1} - \bar{H}_k \boldsymbol{p}\|_2, \; \boldsymbol{q}_k = [\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_k]\boldsymbol{p}_k$
Apply ℓ 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 $\boldsymbol{z}' := \mathcal{B}^{\langle \ell \rangle}\boldsymbol{q}_k$.
$\Delta \boldsymbol{w}_k := \Delta \boldsymbol{w}_0 + \boldsymbol{z}'$

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) [30, 47, 48] can be applied to solve the corrector stage problem (12a) 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

$$\min_{\Delta \boldsymbol{y}_{\mathrm{cc}}} \|\mathcal{A}^\mathsf{T}\Delta \boldsymbol{y}_{\mathrm{cc}} - (SX)^{-1/2}\left(\Delta X_{\mathrm{af}}\Delta S_{\mathrm{af}}\boldsymbol{e} - \sigma\mu\boldsymbol{e}\right)\|_2. \tag{15}$$

In fact, this formulation was adopted in [26] and solved by the CGLS method preconditioned by partial Cholesky decomposition that works in m-dimensional 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 the coefficient matrices of the constraints of (11) and (13). 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 $C^{(\ell)}$ is SPD for ℓ odd for $\omega \in (0, 2)$ [46, 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 [54], [48, Algorithm D.8]. See [5] where SSOR preconditioning for CGNE with $\ell = 1$ is proposed.

By applying Algorithm 5 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.

Algorithm 5 NE-SSOR method.

Let $\boldsymbol{z}^{\langle 0\rangle} = \boldsymbol{0}$ and $\boldsymbol{u}^{\langle 0\rangle} = \boldsymbol{0}$.
for $k = 1, 2, \dots, \ell$, **do**
  for $i = 1, 2, \dots, m$, **do**
    $d_i^{\langle k-\frac{1}{2}\rangle} := \omega[g_i - (\boldsymbol{\alpha}_i, \boldsymbol{u}^{\langle k-1\rangle})]/\|\boldsymbol{\alpha}_i\|_2^2$
    $z_i^{\langle k-\frac{1}{2}\rangle} := z_i^{\langle k-1\rangle} + d_i^{\langle k-\frac{1}{2}\rangle}, \boldsymbol{u}^{\langle k-1\rangle} := \boldsymbol{u}^{\langle k-1\rangle} + d_i^{\langle k-\frac{1}{2}\rangle}\boldsymbol{\alpha}_i$
  end for
  for $i = m, m-1, \dots, 1$, **do**
    $d_i^{\langle k\rangle} := \omega[g_i - (\boldsymbol{\alpha}_i, \boldsymbol{u}^{\langle k-1\rangle})]/\|\boldsymbol{\alpha}_i\|_2^2$
    $z_i^{\langle k\rangle} := z_i^{\langle k-\frac{1}{2}\rangle} + d_i^{\langle k\rangle}, \boldsymbol{u}^{\langle k-1\rangle} := \boldsymbol{u}^{\langle k-1\rangle} + d_i^{\langle k\rangle}\boldsymbol{\alpha}_i$
  end for
  $\boldsymbol{u}^{\langle k\rangle} := \boldsymbol{u}^{\langle k-1\rangle}$
end for

3.4. SOR inner iterations for preconditioning the AB-GMRES method. Next, we introduce the successive overrelaxation (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 $\boldsymbol{q}_k$ which is 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 [16]. An efficient algorithm for this method is called NE-SOR and is given as follows [54], [48, Algorithm D.7]. Let $\boldsymbol{\alpha}_i$ be the ith row of $\mathcal{A}$.

Algorithm 6 NE-SOR method.

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

By applying Algorithm 6 to lines 4 and 12 of Algorithm 4, the normal equations of the second kind are solved approximately.

3.5. Row-scaling of the coefficient matrix. Let $\mathcal{D}$ be a diagonal matrix whose diagonal elements are positive. Then, the problem (14) is equivalent to

$$\min \|\Delta\boldsymbol{w}\|_2, \quad \text{subject to} \quad \mathcal{D}^{-1}\mathcal{A}\Delta\boldsymbol{w} = \mathcal{D}^{-1}\boldsymbol{f}. \tag{16}$$

Denote $\hat{\mathcal{A}} := \mathcal{D}^{-1}\mathcal{A}$ and $\hat{f} := \mathcal{D}^{-1}\boldsymbol{f}$. Then, the scaled problem (16) can be written as

$$\min \|\Delta\boldsymbol{w}\|_2, \quad \text{subject to} \quad \hat{\mathcal{A}}\Delta\boldsymbol{w} = \hat{\boldsymbol{f}}. \tag{17}$$

If $\hat{\mathcal{B}} \in \mathbb{R}^{n\times m}$ satisfies $\mathcal{R}(\hat{\mathcal{B}}) = \mathcal{R}(\hat{\mathcal{A}}^\mathsf{T})$, then (17) is equivalent to

$$\hat{\mathcal{A}}\hat{\mathcal{B}}\hat{\boldsymbol{z}} = \hat{\boldsymbol{f}}, \qquad \Delta\boldsymbol{w} = \hat{\mathcal{B}}\hat{\boldsymbol{z}} \tag{18}$$

for all $\hat{\boldsymbol{f}} \in \mathcal{R}(\hat{\mathcal{A}})$. The methods discussed earlier can be applied to solve (18). In the NE-(S)SOR inner iterations, one has to compute $\|\hat{\boldsymbol{\alpha}}_i\|_2$, the norm of the ith row of the coefficient matrix $\hat{\mathcal{A}}$. 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.

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 softwares. We also developed an efficient direct solver coded in C to compare with the iterative solvers. Therefore, 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 [63]. 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 coefficient matrix is symmetric positive definite. We used the `ldlchol` from CSPARSE package version 3.1.0 [14] when the coefficient matrix was symmetric positive definite, and we turned to the MATLAB built-in solver `ldl` for the indefinite cases.

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 [63]. 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

$$\Gamma \le \epsilon_{\text{out}} = 10^{-8}, \qquad \Gamma := \max\left\{\mu, \frac{\|\boldsymbol{b} - A\boldsymbol{x}\|_2}{\max\{\|\boldsymbol{b}\|_2, 1\}}, \frac{\|\boldsymbol{c} - \boldsymbol{s} - A^{\mathsf{T}}\boldsymbol{y}\|_2}{\max\{\|\boldsymbol{c}\|_2, 1\}}\right\}. \tag{19}$$

For the iterative solver for the normal equations (10a) and (12a), we set the maximum number of iterations for CGNE, MRNE and AB-GMRES as m, and relax it to a larger number for some difficult problems for CGNE and MRNE. We set the stopping criterion for the scaled residual as

$$\|\hat{\boldsymbol{f}} - \hat{\mathcal{A}}\Delta\boldsymbol{w}\|_2 \le \epsilon_{\text{in}}\|\hat{\boldsymbol{f}}\|_2,$$

where the value of ϵ_{in} is chosen to start from 10^{-6} and is kept to be in the range $[10^{-14}, 10^{-4}]$ during the process. We adjusted the value of ϵ_{in} according to the progress of the interior-point iterations. We truncated the iterative solving prematurely in the early phase of the interior-point process, and pursued a more precise direction when approaching the solution to the LP problem. The progress was measured by the error measure Γ. Concretely, we adjusted ϵ_{in} as

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

As there were steps where iterative solvers failed to converge within the maximum number of iterations, we slightly increased the value of ϵ_{in} by multiplying by 1.5 for the next step in such cases.

We adopt the implementation of AB-GMRES preconditioned by NE-SOR inner-iterations [33] with the additional row-scaling scheme (subsection 3.5). No restarts were used for the AB-GMRES method.

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 the standard solvers available in CVX [29, 28]: SDPT3 version 4.0 [58, 59], SeDuMi version 1.34 [56], and MOSEK version 7.1.0.12 [49], with the default interior-point stopping criterion (19). Note that SDPT3 and SeDuMi are non-commercial public domain solvers, whereas MOSEK is a commercial solver known as one of the state-of-the-art solvers. These solvers were implemented with the CVX MATLAB interface, and we recorded the CPU time as 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 seconds while the total processing time was 7,615.2 seconds.

4.3. Typical LP problems: sparse and full-rank problems. We tested on 125 problems which are a subset of the typical LP problem collections: NETLIB, QAPLIB and MITTELMANN, which can be found in [15]. These problems usually have sparse and full-rank constraint matrix A (except problems `bore3d` and `cycle`).

The overall numerical experience is summarized in Table 1. MRNE (NE-SSOR) and MOSEK were the stablest in the sense that they solved all the 125 problems. CGNE (NE-SSOR) method solved all the problems 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 it solved the problems accurately enough. However, it took longer than 20 hours for two problems that have 154,699 and 23,541 unknowns, respectively, although it succeeded in solving larger problems such as `pds-80`. The other solvers were less stable. The modified Cholesky solver solved only 93% of the problems due to numerical instabilities, although it was fast for the problems that

TABLE 1
Overall performance of the solvers on 125 testing problems.

Status	Solved	Solved†	Unsolved	Expensive
AB-GMRES (NE-SOR)	123	0	0	2
CGNE (NE-SSOR)	124	1	0	0
MRNE (NE-SSOR)	125	0	0	0
Modified Cholesky	117	2	6	0
SDPT3	76	19	25	5
SeDuMi	103	16	6	0
MOSEK	125	0	0	0

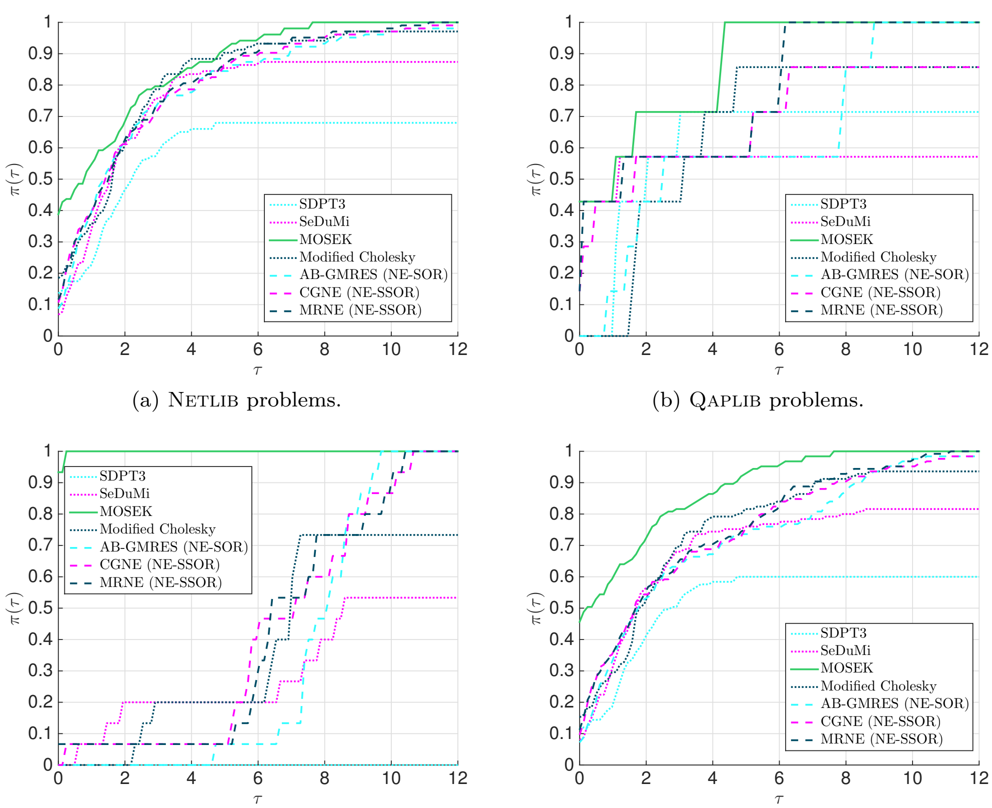
it could successfully solve. SDPT3 solved 61% and SeDuMi 82% of the problems. Here we should mention that SeDuMi and SDPT3 are designed to solve LP, semidefinite programming (SDP), and second-order convex programming (SOCP), while our code is (currently) tuned solely for LP.

Notice that the MOSEK solver is embedded in a multi-corrector interior-point method [23] while our implementation is a single corrector (i.e., predictor-corrector) method. This led to different numbers of interior-point iterations as given 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 [17] in Figures 1 and 2, with the notations: $\pi(\tau) := P(\log_2 r_{ps} \leq \tau)$ 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 the solver s to solve the problem p, t^*_{p} is the minimal CPU time for the problem p. 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 and 4, we give the following information:

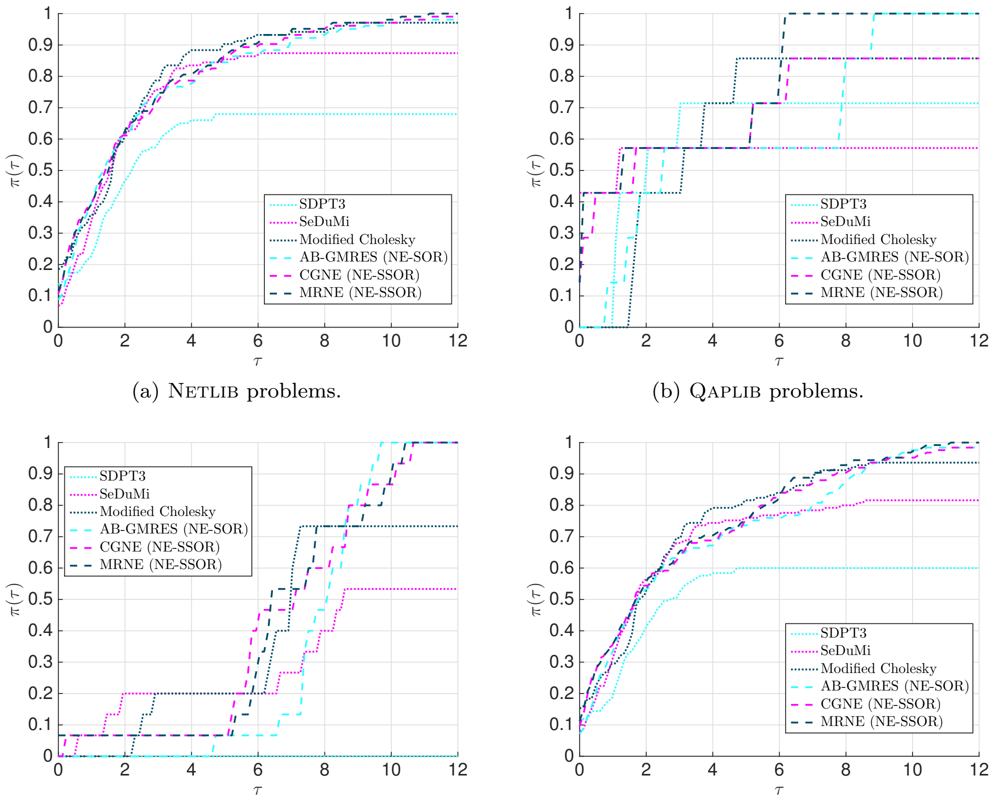
1. the name of the problem and the size (m, n) of the constraint matrix,



(a) Netlib problems. (b) Qaplib problems.

(c) Mittelmann problems. (d) All the problems.

Fig. 1. *Dolan-Moré profiles for the proposed solvers, public domain and commercial solvers.*



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

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

FIG. 2. *Dolan-Moré profiles for the proposed solvers and public domain solvers.*

2. the number of interior-point iterations required for convergence,
3. CPU time for the entire computation in seconds. For the cases shorter than $3,000$ seconds, CPU time is taken as an average over 10 measurements. We indicate in red boldface and blue underline the fastest and second fastest solvers in CPU time, respectively.

Besides the statistics, we also use the following notations:

- † inaccurately solved, i.e., the value of ϵ_{out} was relaxed to a larger level. For our solvers, we provide extra information at the stopping point: $\dagger_a$, $a = \lfloor \log_{10} \Gamma \rfloor$ in the iter column, and $\dagger_b$, $b = \lfloor \log_{10} \kappa(\mathcal{A}\mathcal{A}^\mathsf{T}) \rfloor$ in the time column, where $\lfloor \cdot \rfloor$ is the floor function; the CVX solvers do not provide the condition number but only the relative duality gap,
- \- the iterations diverged due to numerical instabilities,
- ⋄ the iterations took longer than 20 hours.

Note that all zero rows and columns of the constrained matrix A were removed beforehand. The problems marked with # are with rank-deficient A even after this preprocessing. For these problems we put the rank(A) in the bracket after m.

TABLE 2
Experiments on NETLIB *problems.*

			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
25fv47	821	1,876	25	4.62	25	5.00	25	4.60	25	3.67	59	2.50	29	**2.30**	26	3.90
adlittle	56	138	12	**0.03**	13	**0.03**	13	0.05	12	0.09	16	0.16	14	0.10	10	1.98
afiro	27	51	8	0.02	8	**0.01**	8	**0.01**	8	0.03	11	0.11	7	0.10	9	1.91
agg	488	615	21	0.72	21	0.88	24	0.79	21	1.49	34	**0.61**	32	0.90	18	2.24
agg2	516	758	21	0.64	21	0.56	23	**0.53**	21	1.55	32	1.28	23	1.00	13	2.12
agg3	516	758	19	0.68	19	**0.52**	21	0.58	19	1.38	32	1.24	22	1.10	12	2.06
bandm	305	472	18	0.73	19	0.62	19	0.74	17	0.90	42	1.52	20	**0.50**	15	2.17
beaconfd	173	295	13	**0.07**	13	**0.07**	13	**0.07**	12	0.41	15	0.22	10	0.20	8	1.97
blend	74	114	12	**0.06**	14	0.07	13	0.08	12	0.11	15	0.16	11	0.10	9	1.98
bnl1	643	1,586	25	2.53	25	4.66	25	4.92	25	**1.95**	$\dagger_{-5}$	†	64	2.50	20	2.51
bnl2	2,324	4,486	32	44.98	32	23.37	32	27.63	32	12.41	$\dagger_{-4}$	†	38	5.80	25	**2.66**
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
brandy	220	303	17	0.43	18	0.86	18	0.86	17	0.59	46	1.02	19	**0.40**	12	2.04
capri	271	482	19	**0.80**	19	0.88	19	0.91	19	1.04	47	3.22	33	1.60	14	2.63
cre_a	3,516	7,248	30	186.77	30	48.43	31	35.79	31	105.60	$\dagger_{-7}$	†	28	8.70	20	**2.69**
cre_b	9,648	77,137	43	787.95	42	611.11	42	455.04	53	1,143.90	$\dagger_{-6}$	†	$\dagger_{-7}$	†	19	**3.63**
cre_c	3,068	6,411	30	268.84	32	47.92	33	46.12	33	79.67	-	-	28	7.70	17	**2.56**
cre_d	8,926	73,948	37	387.17	37	316.81	37	213.69	37	847.00	-	-	34	42.10	16	**3.06**
cycle#	1,903 (1,875)	3,371	30	61.87	31	50.44	61	185.12	-	-	$\dagger_{-6}$	†	30	5.30	20	**2.76**
czprob	929	3,562	39	**1.51**	38	1.60	39	1.73	39	10.45	$\dagger_{-5}$	†	39	2.80	27	2.91
d2q06c	2,171	5,831	32	132.75	33	581.83	36	750.06	32	24.09	84	6.43	29	4.10	21	**2.85**
d6cube	415	6,184	23	3.77	24	7.41	23	7.12	26	2.68	34	**1.65**	-	-	11	2.50
degen2	444	757	15	1.26	16	1.13	16	1.18	21	2.27	17	0.41	13	**0.40**	8	2.12
degen3	1,503	2,604	19	27.30	21	13.26	21	13.38	19	27.52	$\dagger_{-6}$	†	15	**2.00**	12	2.18
dfl001	6,071	12,230	48	4,336.35	50	2,044.54	55	2,205.16	91	3,131.77	-	-	$\dagger_{-5}$	†	22	**7.46**
e226	223	472	21	0.64	20	0.61	21	0.82	20	**0.59**	61	1.17	22	0.60	14	1.97

Table 2
(cont.) Experiments on Netlib *problems.*

			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
etamacro	400	816	30	**1.23**	31	1.58	31	1.43	30	2.30	-	-	30	2.80	20	2.82
fffff800	524	1,028	32	4.11	30	6.29	33	6.39	32	3.31	44	**0.86**	46	1.60	22	2.55
fit1d	24	1,049	21	0.78	21	0.45	21	0.49	19	**0.38**	36	2.11	18	0.80	13	0.67
fit1p	627	1,677	16	4.01	16	5.31	16	5.14	16	3.56	25	1.78	53	2.00	17	**0.73**
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**
fit2p	3,000	13,525	19	1,103.13	32	1,755.13	32	1,831.13	19	102.02	27	3.69	40	8.90	17	**0.82**
ganges	1,309	1,706	18	8.21	18	27.73	21	33.06	18	3.80	22	**0.90**	26	1.60	15	0.91
gfrd_pnc	616	1,160	21	1.15	22	1.04	21	0.88	21	0.98	27	**0.85**	20	1.00	29	0.90
grow15	300	645	19	0.43	19	**0.35**	20	0.37	17	0.40	21	0.80	25	1.00	13	0.89
grow22	440	946	20	0.68	20	0.59	22	0.59	18	**0.53**	22	0.93	26	1.40	14	0.95
grow7	140	301	18	**0.12**	18	0.16	18	**0.12**	16	0.16	19	0.66	19	0.70	12	0.69
israel	174	316	24	0.99	27	0.94	27	1.06	25	1.12	34	**0.51**	20	0.60	15	2.14
kb2	43	68	16	0.09	17	**0.08**	17	**0.08**	15	0.11	26	0.71	15	0.50	16	0.75
ken_07	2,426	3,602	17	4.14	18	2.39	17	2.24	16	1.07	33	1.74	18	1.80	15	**0.79**
ken_11	14,694	21,349	22	636.24	23	123.23	23	85.95	22	7.83	$\dagger_{-4}$	†	38	10.60	31	**1.87**
ken_13	28,632	42,659	27	2,633.00	28	365.15	29	348.51	27	23.90	-	-	43	29.50	20	**2.83**
ken_18	105,127	154,699	⋄	⋄	38	12,893.63	46	21,315.47	38	324.89	-	-	59	765.30	20	**24.98**
lotfi	153	366	16	0.28	16	**0.24**	16	0.32	16	0.39	37	1.14	20	1.20	15	2.47
maros_r7	3,136	9,408	15	57.78	15	29.69	15	31.68	15	11.14	21	5.39	15	4.80	12	**3.29**
modszk1	687	1,620	23	2.70	23	3.60	23	3.48	22	2.54	29	**0.85**	23	1.00	22	0.92
osa_07	1,118	25,067	34	12.35	32	6.26	36	8.51	27	5.85	31	3.90	31	4.90	14	**2.55**
osa_14	2,337	54,797	38	11.41	32	9.11	37	11.81	37	16.07	37	7.65	36	7.30	18	**3.03**
osa_30	4,350	104,374	39	22.69	41	19.08	38	17.16	36	28.98	37	12.49	40	11.50	17	**3.36**
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**
pds_02	2,953	7,716	29	4.43	29	3.43	29	4.16	29	3.16	$\dagger_{-5}$	†	30	6.90	18	**0.82**
pds_06	9,881	29,351	48	49.77	48	44.17	51	45.85	48	44.65	-	-	51	61.50	23	**1.45**

TABLE 2
(cont.) Experiments on NETLIB *problems.*

			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
pds_10	16,558	49,932	51	91.60	52	87.75	50	79.22	52	130.17	-	-	74	157.20	28	**2.54**
pds_20	33,874	108,175	61	1,365.98	64	1,155.95	62	683.72	62	665.05	-	-	$\dagger_{-7}$	†	34	**11.02**
perold	625	1,506	36	4.71	36	6.71	36	6.97	37	2.82	$\dagger_{-6}$	†	$\dagger_{-7}$	†	24	**0.87**
pilot	1,441	4,860	33	31.54	33	51.15	33	49.36	33	16.18	-	-	81	19.70	39	**1.73**
pilot4	410	1,123	30	2.11	30	2.12	30	2.29	30	2.26	$\dagger_{-7}$	†	-	-	27	**0.78**
pilot87	2,030	6,680	39	55.59	39	105.77	39	102.58	39	33.13	88	11.54	76	12.60	38	**2.45**
pilot_ja	940	2,267	35	13.02	37	19.51	36	14.79	37	4.84	-	-	-	-	29	**0.71**
pilot_we	722	2,928	35	5.67	39	8.58	38	7.62	35	2.42	$\dagger_{-7}$	†	44	4.90	31	**0.71**
pilotnov	975	2,446	24	5.70	25	5.02	27	4.07	22	2.90	-	-	-	-	17	**0.73**
qap12	3,192	8,856	19	758.92	21	144.74	20	99.35	19	50.45	26	21.78	$\dagger_{-7}$	†	17	**6.09**
qap15	6,330	22,275	23	5,530.52	25	789.81	24	581.25	24	335.83	52	330.31	$\dagger_{-7}$	†	17	**21.11**
qap8	912	1,632	11	1.73	12	1.09	11	**0.98**	10	2.75	13	1.25	8	1.10	7	2.16
sc105	105	163	10	0.05	10	0.04	10	0.04	10	**0.02**	20	0.50	10	0.20	8	2.13
sc205	205	317	11	0.17	11	0.09	11	0.07	10	**0.05**	18	0.61	12	0.30	10	2.16
sc50a	50	78	10	0.03	10	**0.00**	6	0.02	10	0.02	12	0.17	8	0.20	8	2.13
sc50b	50	78	7	**0.01**	7	0.02	7	0.02	7	0.03	11	0.26	7	0.20	6	1.94
scagr25	471	671	18	0.93	18	0.69	18	0.71	17	**0.20**	35	0.84	21	0.70	21	2.63
scagr7	129	185	14	0.15	15	0.11	15	0.11	14	**0.07**	33	0.71	17	0.50	19	2.52
scfxm1	330	600	18	1.03	19	1.05	20	1.14	18	**0.70**	52	1.40	20	0.80	15	2.42
scfxm2	660	1,200	21	2.44	22	4.73	23	4.71	21	1.35	58	1.59	24	**1.30**	18	2.56
scfxm3	990	1,800	22	5.94	23	12.64	24	12.10	22	1.64	59	1.79	25	**1.50**	16	2.53
scorpion	388	466	15	0.28	16	0.23	16	0.26	15	**0.20**	17	0.39	11	0.30	11	2.21
scrs8	490	1,275	25	0.91	26	0.78	25	0.77	25	**0.61**	37	1.06	35	1.70	14	2.41
scsd1	77	760	9	0.06	9	0.05	9	**0.03**	9	0.04	12	0.23	8	0.20	8	2.02
scsd6	147	1,350	11	0.17	12	0.12	11	0.13	11	**0.07**	15	0.32	11	0.40	10	2.06
scsd8	397	2,750	12	0.76	12	0.71	12	0.64	11	**0.16**	13	0.32	10	0.60	7	1.93

Table 2
(cont.) Experiments on Netlib *problems.*

Problem	m	n	AB-GMRES (NE-SOR) Iter	AB-GMRES (NE-SOR) Time	CGNE (NE-SSOR) Iter	CGNE (NE-SSOR) Time	MRNE (NE-SSOR) Iter	MRNE (NE-SSOR) Time	Modified Cholesky Iter	Modified Cholesky Time	SDPT3 Iter	SDPT3 Time	SeDuMi Iter	SeDuMi Time	MOSEK Iter	MOSEK Time
sctap1	300	660	17	0.31	19	0.38	19	0.36	17	**0.12**	20	0.46	20	0.50	11	2.15
sctap2	1,090	2,500	20	1.36	20	1.21	21	1.04	19	1.75	21	**0.48**	12	0.60	9	2.05
sctap3	1,480	3,340	19	1.33	19	1.14	20	1.22	18	2.31	23	0.94	13	**0.40**	9	2.11
share1b	117	253	23	0.50	24	0.51	24	0.48	23	**0.16**	27	0.52	22	0.50	23	2.74
share2b	96	162	12	0.16	14	0.20	16	0.21	12	**0.09**	26	0.60	19	0.30	15	2.47
shell	536	1,777	19	0.61	19	0.57	19	0.58	19	1.68	-	-	31	1.10	22	**0.56**
ship04l	402	2,166	14	**0.26**	14	**0.26**	14	**0.26**	15	1.00	20	0.74	17	0.80	10	1.86
ship04s	402	1,506	15	0.78	15	0.30	15	**0.21**	14	1.14	20	0.67	17	0.70	11	0.48
ship08l	778	4,363	16	0.82	17	1.33	17	1.28	16	2.47	21	**0.51**	18	0.90	11	1.93
ship08s	778	2,467	15	0.44	16	0.46	16	0.60	15	1.82	20	**0.32**	16	0.40	10	1.88
ship12l	1,151	5,533	20	1.48	19	2.21	20	2.01	19	4.66	22	**0.65**	23	1.90	14	2.04
ship12s	1,151	5,533	17	0.90	19	1.00	19	0.94	17	2.66	22	**0.41**	17	0.90	12	1.96
sierra	1,227	2,735	17	1.28	19	1.37	19	1.05	21	1.60	-	-	29	3.50	16	**0.59**
stair	356	614	22	1.43	22	1.63	22	1.87	22	0.96	$\dagger_{-6}$	$\dagger$	18	0.70	15	**0.52**
standata	359	1,274	18	0.63	17	**0.34**	17	0.38	17	0.86	-	-	19	0.70	9	0.48
standgub	361	1,383	17	0.35	17	**0.30**	17	0.37	17	0.91	-	-	19	0.70	9	0.51
standmps	467	1,274	25	0.81	24	0.68	25	0.82	24	1.71	-	-	15	0.70	17	**0.53**
stocfor1	117	165	19	0.13	21	0.13	20	0.20	19	**0.09**	30	0.71	17	0.50	11	2.21
stocfor2	2,157	3,045	23	37.36	24	18.00	24	17.59	21	13.43	53	**1.95**	$\dagger_{-4}$	$\dagger$	17	2.54
stocfor3	16,675	23,541	$\diamond$	$\diamond$	38	4,590.71	37	4,071.37	$\dagger_{-7}$	$\dagger_{32}$	80	11.05	-	-	26	**3.37**
truss	1,000	8,806	19	6.62	21	10.22	22	10.59	19	3.29	21	**1.12**	19	1.90	12	2.27
tuff	333	628	21	1.63	22	1.27	24	2.03	21	1.39	$\dagger_{-7}$	$\dagger$	21	0.80	18	**0.60**
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
wood1p	244	2,595	17	1.75	17	1.34	17	**1.19**	-	-	38	2.75	19	2.00	10	2.17
woodw	1,098	8,418	25	5.12	27	6.72	28	7.34	22	3.73	-	-	33	3.20	17	**2.47**

TABLE 3
Experiments on QAPLIB *problems.*

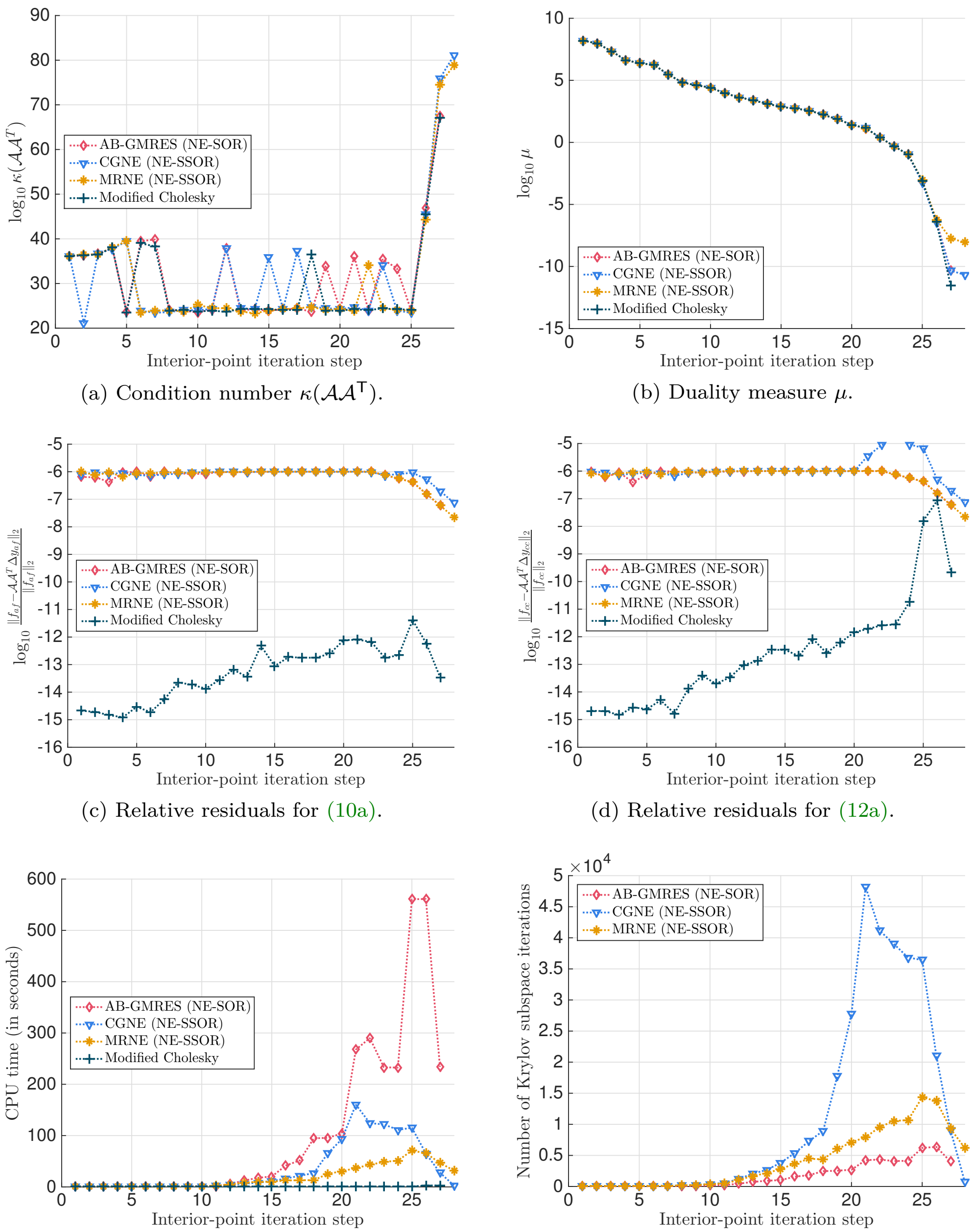
			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
nug05	201	225	7	0.16	7	**0.09**	7	**0.09**	7	0.27	12	0.36	5	0.20	5	1.81
nug06	372	486	10	0.56	10	0.31	10	0.25	8	0.83	11	0.22	6	**0.10**	6	1.84
nug07	602	931	12	1.83	13	0.96	12	0.72	12	2.02	18	1.48	10	**0.70**	8	2.09
nug08	912	1,632	10	3.27	11	1.03	12	1.06	10	3.26	16	2.08	8	**1.00**	7	1.96
nug12	3,192	8,856	19	1,287.19	20	427.16	19	355.36	20	73.13	$\dagger_{-7}$	$\dagger$	$\dagger_{-7}$	$\dagger$	17	**5.57**
nug15	6,330	22,275	23	9,521.25	25	809.23	24	773.55	23	559.88	33	171.64	$\dagger_{-5}$	$\dagger$	17	**22.13**
nug20	15,240	72,600	25	60,223.29	$\dagger_{-7}$	$\dagger_{28}$	33	16,650.52	$\dagger_{-7}$	$\dagger_{28}$	$\dagger_{-7}$	$\dagger$	$\dagger_{-5}$	$\dagger$	19	**243.71**

TABLE 4
Experiments on MITTELMANN *problems.*

			AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		SDPT3		SeDuMi		MOSEK	
Problem	m	n	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
fome11	12,142	24,460	47	6,900.09	48	14,156.31	53	12,270.84	-	-	-	-	$\dagger_{-5}$	$\dagger$	23	**8.97**
fome12	24,284	48,920	48	12,568.26	48	38,138.98	52	28,159.85	-	-	-	-	$\dagger_{-7}$	$\dagger$	21	**33.17**
fome13	48,568	97,840	47	25,726.58	50	37,625.03	54	63,301.06	-	-	-	-	$\dagger_{-7}$	$\dagger$	24	**61.01**
fome20	33,874	108,175	61	1,510.85	64	1,240.23	62	689.71	62	692.71	-	-	$\dagger_{-7}$	$\dagger$	34	**8.96**
fome21	67,748	216,350	74	12,671.62	74	3,185.03	84	3,822.02	75	1,617.71	-	-	$\dagger_{-6}$	$\dagger$	39	**18.47**
nug08-3rd	19,728	29,856	12	5,833.97	11	259.01	10	**237.02**	-	-	-	-	-	-	7	257.82
pds-30	49,944	158,489	69	1,964.48	72	1,105.42	70	788.98	69	1,659.21	-	-	103	2,014.70	34	**19.93**
pds-40	66,844	217,531	66	4,878.49	68	1,551.30	77	1,904.76	67	4,012.71	$\diamond$	$\diamond$	105	4,832.20	34	**31.15**
pds-50	83,060	275,814	73	13,860.17	73	3,274.74	80	3,960.55	73	7,196.51	$\diamond$	$\diamond$	111	11,433.90	38	**49.74**
pds-60	99,431	336,421	72	25,592.33	75	5,024.43	83	7,535.99	72	11,609.01	$\diamond$	$\diamond$	$\dagger_{-7}$	$\dagger$	36	**94.28**
pds-70	114,944	390,005	80	22,564.32	82	4,980.04	85	7,405.50	84	17,575.97	$\diamond$	$\diamond$	126	44,946.8	46	**136.50**
pds-80	129,181	434,580	80	25,752.26	83	6,279.08	86	9,853.86	85	21,077.53	$\diamond$	$\diamond$	119	58,286.40	42	**157.64**
rail507	507	63,516	43	1,039.09	51	1,138.80	51	475.47	48	14.98	$\dagger_{-7}$	$\dagger$	34	7.10	17	**2.69**
rail516	516	47,827	39	496.60	43	700.58	39	536.36	38	11.82	$\dagger_{-7}$	$\dagger$	19	3.70	11	**2.48**
rail582	582	56,097	44	1,296.56	46	971.35	47	1,422.62	41	17.52	$\dagger_{-7}$	$\dagger$	40	8.60	16	**2.43**

In order to give an idea of the typical differences between the methods, we present the interior-point convergence curves for the 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 Figure 3. The red dotted line with diamond markers represents the quantity related to AB-



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

(b) Duality measure μ.

(c) Relative residuals for (10a).

(d) Relative residuals for (12a).

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

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

FIG. 3. *Numerical result for problem ken_13.*

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^{37} and increased to 10^{70} at the last 3-5 steps. Figure 3b shows the convergence curve of the duality measure μ in $\log_{10}$ scale. The value of μ successfully attained the stopping criterion (19) for all the solvers. 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 stagnated for $\mu \simeq 10^{-4}$. Comparing with Figure 3a, it is observed that the solvers started to behave differently as $\kappa(\mathcal{A}\mathcal{A}^\mathsf{T})$ increased sharply.

Figures 3c and 3d show the relative residual norm for the normal equations $\|\boldsymbol{f}_{\mathrm{af}} - \mathcal{A}\mathcal{A}^\mathsf{T}\Delta\boldsymbol{y}_{\mathrm{af}}\|_2/\|\boldsymbol{f}_{\mathrm{af}}\|_2$ 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 testing problems and, in the worst cases, the inaccuracy in the solutions prohibited 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.

Figures 3e and 3f 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 the coefficient matrix and the preconditioner, the memory required for k iterations of ABGMRES is $\mathcal{O}(k^2 + km + n)$ and that for CGNE and MRNE iterations is $\mathcal{O}(m + n)$[30, 48]. This explains why ABGMRES (NE-SOR), although requiring less iterations, usually takes longer time to obtain the solution at each interior-point step.

From Figure 3, we may draw a few conclusions. For most of the 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 the problems that the direct method broke down, the proposed inner-iteration preconditioned Krylov subspace methods worked until convergence. It is acceptable to iteratively solve for an approximate step in the early phase of the interior-point method and then increase the level of accuracy in the late phase.

4.4. Rank-deficient problems. Most of the problems tested in the last section have a sparse and full-rank constraint matrix A. In this section, we enrich the experiment by adding artificial problems with a dense, rank-deficient and ill-conditioned constraint matrix, which challenge some of the solvers.

We first present an experiment to investigate the effect of rank-deficiency on CPU time. Since MOSEK was the most efficient and stable standard solver as presented in the previous section, here we only compare our solvers with MOSEK. We randomly generated a set of constraint matrix A whose rank ranged from 50 to 100 with a step of 5. The elements of $\boldsymbol{x}$ and $\boldsymbol{c}$ were uniformly distributed random numbers, generated by using the MATLAB function `rand`. The location of zero elements of $\boldsymbol{x}$ was also subject to the random uniform distribution. Then, $\boldsymbol{b}$ was generated as $\boldsymbol{b} = A\boldsymbol{x}$. More details are given in Table 5.

In Figure 4, we plot the CPU time required for each solver to achieve the interior-point convergence versus rank(A). In order to give an averaged information, we took an average of the CPU times for 5 different randomly generated problems for each rank, where the CPU time was taken as an average of 10 measurements for each problem.

All the solvers succeeded in solving the problems. Iterative solvers performed better than modified Cholesky as the rank decreased.

Next, we present an experiment for problems that were both rank-deficient and ill-conditioned. We randomly generated a set of problems, each had a constraint matrix A with information given in Table 6. The sparsity of A was around 50%.

TABLE 5
Information on artificial problems: completely dense with different rank.

Problem	m	n	Nonzeros	Rank	$\kappa(A)$
Artificial	100	300	30,000	$[50, 100]$	10^2

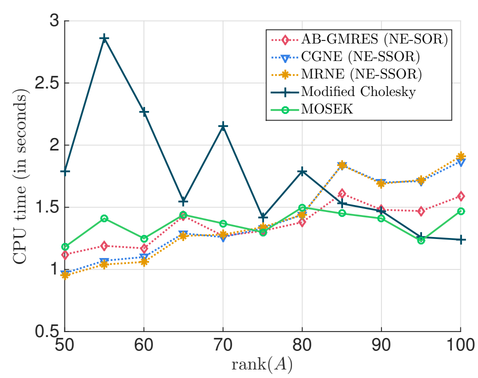


FIG. 4. *CPU time for artificial problems: completely dense with different rank.*

TABLE 6
Information on artificial problems: ill-conditioned with different rank.

Problem	m	n	Nonzeros	Rank	$\kappa(A)$
Artificial	100	300	15,000	$[50, 100]$	10^8

In Figure 5, we plot the CPU time required for each solver to achieve the interior-point convergence versus rank(A). The graphs for modified Cholesky and MOSEK are disconnected since there were failed cases. For example, MOSEK (green line with circles) failed at the point rank$(A) = 88$ and rank$(A) = 90$, and hence the points at rank$(A) = 86$ and rank$(A) = 92$ were not connected.

This result shows that, MOSEK, although fast and stable for the full-rank problems, failed for 7 out of 26 ill-conditioned rank-deficient problems and was almost always slower than the proposed solvers. The modified Cholesky solver broke down due to numerical errors for 21 problems. However, the three iterative solvers overcame this difficulty and solved all the problems.

Note that when the interior-point solver with MOSEK failed to converge, it automatically switched to a simplex method. Although this re-optimization process can usually give an optimal solution to the LP problem, we consider the interior-point method to have failed.

Similar experiments were carried out on larger problems. We tested the solvers on problems of size $1,000 \times 1,500$ with condition number 10^8 and sparsity around 50%. The result is presented in Table 7. The notations † and - have the same meaning as explained in the previous section. The table shows that only AB-GMRES (NE-SOR) succeeded in solving the problems.

5. Conclusions. We proposed a new way of preconditioning the normal equations of the second kind arising from the interior-point methods for LP problems. The resulting interior-point solver is composed of three nested iteration schemes. The

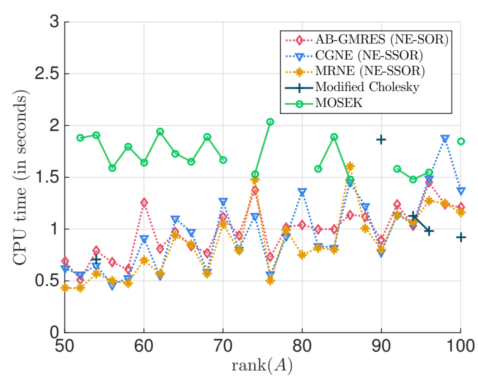


Fig. 5. *CPU time for artificial problems: ill-conditioned with different rank.*

Table 7
Experiments on artificial problems.

		AB-GMRES (NE-SOR)		CGNE (NE-SSOR)		MRNE (NE-SSOR)		Modified Cholesky		MOSEK	
Problem	Rank(A)	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
Rand1	1,000	22	120.04	$\dagger_{-4}$	$\dagger_{18}$	$\dagger_{-6}$	$\dagger_{21}$	-	-	26	6.50
Rand2	999	28	483.85	$\dagger_{-4}$	$\dagger_{18}$	$\dagger_{-6}$	$\dagger_{31}$	-	-	27	11.04
Rand3	998	21	336.19	$\dagger_{-4}$	$\dagger_{21}$	$\dagger_{-6}$	$\dagger_{20}$	-	-	-	-
Rand4	997	24	392.52	$\dagger_{-4}$	$\dagger_{18}$	$\dagger_{-6}$	$\dagger_{20}$	-	-	-	-
Rand5	996	31	441.28	$\dagger_{-4}$	$\dagger_{19}$	$\dagger_{-6}$	$\dagger_{20}$	-	-	-	-
Rand6	995	21	305.69	$\dagger_{-4}$	$\dagger_{21}$	$\dagger_{-6}$	$\dagger_{20}$	-	-	-	-

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 methods; on top of that, we use a row-scaling scheme which does not incur extra CPU time; the inner-most layer, serving as a preconditioner for the middle 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 advantage of our method is that it does not break down, even when the coefficient matrices become (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 the stability and efficiency of our method outperform the open-source solvers SDPT3 and SeDuMi, and can solve the rank-deficient and ill-conditioned problems where the MOSEK interior-point solver fails. There is still room for improvement regarding the iterative solvers as well as using more sophisticated methods for the interior-point iterations such as a multi-corrector interior-point method. It would also be worthwhile to extend our method to solve problems such as convex quadratic programming and SDP.

REFERENCES

[1] I. Adler, M. Resende, G. Veiga, and N. Karmarkar, *An implementation of Karmarkar's algorithm for linear programming*, Math. Program., 44 (1989), pp. 297–335, http://dx.doi.org/10.1007/BF01587095.

[2] E. Andersen and K. Andersen, *Presolving in linear programming*, Math. Program., 71 (1995), pp. 221–245, http://dx.doi.org/10.1007/BF01586000.

[3] E. Andersen, J. Gondzio, C. Mészáros, and X. Xu, *Implementation of interior-point methods for large scale linear programs*, in Interior Point Methods of Mathematical Programming, P. M. Pardalos and D. Hearn, eds., vol. 5 of App. Optim., Kluwer Academic Publishers, Dordrecht, 1996, http://dx.doi.org/10.1007/978-1-4613-3449-1_6.

[4] L. Bergamaschi, J. Gondzio, and G. Zilli, *Preconditioning indefinite systems in interior point methods for optimization*, Comput. Optim. Appl., 28 (2004), pp. 149–171, http://dx.doi.org/10.1023/B:COAP.0000026882.34332.1b.

[5] Å. Björck and T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT, 19 (1979), pp. 145–163, http://dx.doi.org/10.1007/BF01930845.

[6] T. Carpenter and D. Shanno, *An interior point method for quadratic programs based on conjugate projected gradients*, Comput. Optim. Appl., 2 (1993), pp. 5–28, http://dx.doi.org/10.1007/BF01299140.

[7] S. Chen, D. Donoho, and M. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., 20 (1998), pp. 33–61, http://dx.doi.org/10.1137/S003614450037906X.

[8] P. Chin and A. Vannelli, *PCG techniques for interior point algorithms*, in Proceedings of the 36th midwest symposium on circuits and systems, IEEE, 1994, pp. 200–203, http://dx.doi.org/10.1109/MWSCAS.1993.343095.

[9] E. Craig, *The N-step iteration procedures*, J. Math. and Phys., 34 (1995), pp. 64–73, http://dx.doi.org/10.1002/sapm195534164.

[10] X. Cui, *Approximate Generalized Inverse Preconditioning Methods for Least Squares Problems*, PhD thesis, Department of Informatics, School of Multidisciplinary Sciences, The Graduate University for Advanced Studies (SOKENDAI), 2009, http://id.nii.ac.jp/1013/00001492/ (accessed 2016-04-22).

[11] X. Cui, K. Hayami, and J.-F. Yin, *Greville's method for preconditioning least squares problems*, Adv. Comput. Math., 35 (2011), pp. 243 – 269, http://dx.doi.org/10.1007/s10444-011-9171-x.

[12] J. Czyzyk, S. Mehrotra, M. Wagner, and S. Wright, *PCx: An interior-point code for linear programming*, Optim. Methods Softw., 11 (1999), pp. 397–430, http://dx.doi.org/10.1080/10556789908805757.

[13] M. D'Apuzzo, V. De Simone, and D. Di Serafino, *On mutual impact of numerical linear*

algebra and large-scale optimization with focus on interior point methods, Comput. Optim. Appl., 45 (2010), pp. 283–310, http://dx.doi.org/10.1007/s10589-008-9226-1.

[14] T. Davis, *CSparse: A concise sparse matrix package*, 2014, http://people.sc.fsu.edu/~jburkardt/c_src/csparse/csparse.html (accessed 2016-04-15). Version 3.1.4.

[15] T. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011), pp. 1:1–1:25, http://www.cise.ufl.edu/research/sparse/matrices/ (accessed 2016-04-15).

[16] A. Dax, *The convergence of linear stationary iterative processes for solving singular unstructured systems of linear equations*, SIAM Rev., 32 (1990), pp. 611–635, http://dx.doi.org/10.1137/1032122.

[17] E. Dolan and J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213, http://dx.doi.org/10.1007/s101070100263.

[18] M. Ferris and T. Munson, *Interior-point methods for massive support vector machines*, SIAM J. Optim., 13 (2002), pp. 783–804, http://dx.doi.org/10.1137/S1052623400374379.

[19] R. Fourer and S. Mehrotra, *Solving symmetric indefinite systems in an interior-point method for linear programming*, Math. Program., 62 (1993), pp. 15–39, http://dx.doi.org/10.1007/BF01585158.

[20] R. Freund and F. Jarre, *A QMR-based interior-point algorithm for solving linear programs*, Math. Program., 76 (1997), pp. 183–210, http://dx.doi.org/10.1007/BF02614383.

[21] R. Freund, F. Jarre, and S. Mizuno, *Convergence of a class of inexact interior-point algorithms for linear programs*, Math. Oper. Res., 24 (1999), pp. 50–71, http://dx.doi.org/10.1287/moor.24.1.50.

[22] P. Gill, W. Murray, M. Saunders, J. Tomlin, and M. Wright, *On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method*, Math. Program., 36 (1986), pp. 183–209, http://dx.doi.org/10.1007/BF02592025.

[23] J. Gondzio, *Multiple centrality corrections in a primal-dual method for linear programming*, Comput. Optim. Appl., 6 (1996), pp. 137–156, http://dx.doi.org/10.1007/BF00249643.

[24] J. Gondzio, *Presolve analysis of linear programs prior to applying an interior point method*, INFORMS J. Comput., 9 (1997), pp. 73–91, http://dx.doi.org/10.1287/ijoc.9.1.73.

[25] J. Gondzio, *Interior point methods 25 years later*, Eur. J. Oper. Res., 218 (2012), pp. 587–601, http://dx.doi.org/10.1016/j.ejor.2011.09.017.

[26] J. Gondzio, *Matrix-free interior point method*, Comput. Optim. Appl., 51 (2012), pp. 457–480, http://dx.doi.org/10.1007/s10589-010-9361-3.

[27] J. Gondzio and T. Terlaky, *A computational view of interior point methods*, in Advances in Linear and Integer Programming, J. E. Beasley and J. E. Beasley, eds., Oxford University Press: Oxford, 1996, pp. 103–144, http://dl.acm.org/citation.cfm?id=247975.247978 (accessed 2016-04-15).

[28] M. Grant and S. Boyd, *Graph implementations for nonsmooth convex programs*, in Recent Advances in Learning and Control, V. Blondel, S. Boyd, and H. Kimura, eds., Lecture Notes in Control and Information Sciences, Springer-Verlag Limited, 2008, pp. 95–110, http://dx.doi.org/10.1007/978-1-84800-155-8_7.

[29] M. Grant and S. Boyd, *CVX: Matlab software for disciplined convex programming*, March 2014, http://cvxr.com/cvx (accessed 2016-04-15). version 2.1.

[30] K. Hayami, J.-F. Yin, and T. Ito, *GMRES methods for least squares problems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2400–2430, http://dx.doi.org/10.1137/070696313.

[31] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436, http://dx.doi.org/10.6028/jres.049.044.

[32] J. Júdice, J. Patricio, L. Portugal, M. Resende, and G. Veiga, *A study of preconditioners for network interior point methods*, Comput. Optim. Appl., 24 (2003), pp. 5–35, http://dx.doi.org/10.1023/A:1021882330897.

[33] K. Morikuni and K. Hayami, *AB-GMRES preconditioned by NE-SOR inner iterations*, 2014, http://researchmap.jp/KeiichiMorikuni/Implementations/ (accessed 2016-04-15).

[34] N. Karmarkar and K. Ramakrishnan, *Computational results of an interior point algorithm for large scale linear programming*, Math. Program., 52 (1991), pp. 555–586, http://dx.doi.org/10.1007/BF01582905.

[35] M. Kojima, S. Mizuno, and A. Yoshise, *A polynomial-time algorithm for a class of linear complementarity problems*, Math. Program., 4 (1989), pp. 1–26, http://dx.doi.org/10.1007/BF01587074.

[36] J. Korzak, *Convergence analysis of inexact infeasible-interior-point algorithms for solving linear programming problems*, SIAM J. Optim., 11 (2000), pp. 133–148, http://dx.doi.org/10.1137/S1052623497329993.

[37] I. Lustig, R. Marsten, and D. Shanno, *On implementing Mehrotra's predictor-corrector interior-point method for linear programming*, SIAM J. Optim., 2 (1992), pp. 435–449, http://dx.doi.org/10.1137/0802022.

[38] I. Lustig, R. Marsten, and D. Shanno, *Interior point methods for linear programming: Computational state of the art*, ORSA J. Comput., 6 (1994), pp. 1–14, http://dx.doi.org/10.1287/ijoc.6.1.1.

[39] S. Mehrotra, *Implementations of affine scaling methods: approximate solutions of systems of linear equations using preconditioned conjugate gradient methods*, ORSA Journal on Computing, 4 (1992), pp. 103–118, http://dx.doi.org/10.1287/ijoc.4.2.103.

[40] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM J. Optim., 2 (1992), pp. 575–601, http://dx.doi.org/10.1137/0802028.

[41] S. Mehrotra and Z. Li, *Convergence conditions and Krylov subspace-based corrections for primal-dual interior-point method*, SIAM J. Optim., 15 (2005), pp. 635–653, http://dx.doi.org/10.1137/S1052623403431494.

[42] S. Mehrotra and J. Wang, *Conjugate gradient based implementation of interior point methods for network flow problems*, in Linear and nonlinear conjugate gradient-related methods, L. Adams and J. Nazareth, eds., SIAM, Philadelphia, PA, 1996, pp. 124–142.

[43] R. Monteiro and I. Adler, *Interior path following primal-dual algorithms. part i: Linear programming*, Math. Program., 44 (1989), pp. 27–41, http://dx.doi.org/10.1007/BF01587075.

[44] R. Monteiro and J. O'Neal, *Convergence analysis of a long-step primal-dual infeasible interior-point LP algorithm based on iterative linear solvers*, tech. report, Georgia Institute of Technology, 2003, http://www.optimization-online.org/DB_FILE/2003/10/768.pdf (accessed 2016-04-15).

[45] R. Monteiro, J. O'Neal, and T. Tsuchiya, *Uniform boundedness of a preconditioned normal matrix used in interior-point methods*, SIAM J. Optim., 15 (2004), pp. 96–100, http://dx.doi.org/10.1137/S1052623403426398.

[46] K. Morikuni, *Symmetric inner-iteration preconditioning for rank-deficient least squares problems*, 2015, arXiv:1504.00889 [math.NA].

[47] K. Morikuni and K. Hayami, *Inner-iteration Krylov subspace methods for least squares problems*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1–22, http://dx.doi.org/10.1137/110828472.

[48] K. Morikuni and K. Hayami, *Convergence of inner-iteration GMRES methods for rank-deficient least squares problems*, SIAM J. Matrix Anal. Appl., 36(1) (2015), pp. 225–250, http://dx.doi.org/10.1137/130946009.

[49] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual.*, 2015, http://docs.mosek.com/7.0/toolbox/ (accessed 2016-04-15). Version 7.1 (Revision 35).

[50] A. Oliveira and D. Sorensen, *A new class of preconditioners for large-scale linear systems from interior point methods for linear programming*, Linear Algebra Appl., 394 (2005), pp. 1–24, http://dx.doi.org/10.1016/j.laa.2004.08.019.

[51] C. Paige and M. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., (1975), pp. 617–629, http://dx.doi.org/10.1137/0712047.

[52] L. Portugal, M. Resende, G. Veiga, and J. Júdice, *A truncated primal-infeasible dual-feasible network interior point method*, Networks, 35 (2000), pp. 91–108, http://dx.doi.org/10.1002/(SICI)1097-0037(200003)35:2<91::AID-NET1>3.0.CO;2-T.

[53] M. Resende and G. Veiga, *An implementation of the dual affine scaling algorithm for minimum-cost flow on bipartite uncapacitated networks*, SIAM J. Optim., 3 (1993), pp. 516–537, http://dx.doi.org/10.1137/0803025.

[54] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2nd ed., 2003, http://dx.doi.org/10.1137/1.9780898718003.

[55] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869, http://dx.doi.org/10.1137/0907058.

[56] J. Sturm, *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones.*, Optim. Methods Softw., 11-12 (1999 Special issue on Interior Point Methods), pp. 625–633, http://dx.doi.org/10.1080/10556789908805766.

[57] K. Tanabe, *Centered Newton method for mathematical programming*, in System Modeling and Optimization, vol. 113 of Lecture Notes in Control and Information Sciences, Springer-Verlag, 1988, pp. 197–206, http://dx.doi.org/10.1007/BFb0042787.

[58] K. Toh, M. Todd, and R. Tütüncü, *SDPT3 — a Matlab software package for semidefinite programming*, Optim. Methods Softw., 11 (1999), pp. 545–581, http://dx.doi.org/10.1080/10556789908805762.

[59] R. Tütüncü, K. Toh, and M. Todd, *Solving semidefinite-quadratic-linear programs us-*

ing SDPT3, Math. Program. Ser. B, 95 (2003), pp. 189–217, http://dx.doi.org/10.1007/s10107-002-0347-5.

[60] S. Wright, *Primal-Dual Interior-Point Methods*, SIAM, 1997, http://dx.doi.org/10.1137/1.9781611971453.

[61] S. Wright, *Modified Cholesky factorizations in interior-point algorithms for linear programming*, SIAM J. Optim., 9 (1999), pp. 1159–1191, http://dx.doi.org/10.1137/S1052623496304712.

[62] Y. Zhang, *On the convergence of a class of infeasible interior-point methods for the horizontal linear complementarity problem*, SIAM J. Optim., 4 (1994), pp. 208–227, http://dx.doi.org/10.1137/0804012.

[63] Y. Zhang, *Solving large-scale linear programs by interior-point methods under the Matlab environment*, Optim. Methods Softw., 10 (1998), pp. 1–31, http://dx.doi.org/10.1080/10556789808805699.