On geometrical properties of preconditioners in IPMs for classes of block-angular problems

Jordi Castro
Dept. of Stat. and Oper. Res.
Universitat Politècnica de Catalunya
Barcelona, Catalonia
jordi.castro@upc.edu

Stefano Nasini
Dept. of Marketing
IESEG School of Management
Lille, France
snasini@ieseg.fr

Research Report UPC-DEIO DR 2016-03
February 2016; updated November 2016

Report available from http://www-eio.upc.es/~jcastro

ON GEOMETRICAL PROPERTIES OF PRECONDITIONERS IN IPMS FOR CLASSES OF BLOCK-ANGULAR PROBLEMS*

JORDI CASTRO † AND STEFANO NASINI ‡

Abstract. One of the most efficient interior-point methods for some classes of block-angular structured problems solves the normal equations by a combination of Cholesky factorizations and preconditioned conjugate gradient for, respectively, the block and linking constraints. In this work we show that the choice of a good preconditioner depends on geometrical properties of the constraints structure. In particular, it is seen that the principal angles between the subspaces generated by the diagonal blocks and the linking constraints can be used to estimate ex-ante the efficiency of the preconditioner. Numerical validation is provided with some generated optimization problems. An application to the solution of multicommodity network flow problems with nodal capacities and equal flows of up to 64 million variables and up to 7.9 million constraints is also presented.

Key words. interior-point methods, structured problems, preconditioned conjugate gradient, principal angles, large-scale optimization

AMS subject classifications. 90C06, 90C08, 90C51

1. Introduction.

Many real-world optimization problems exhibit a primal block-angular structure, where decision variables and constraints can be grouped in different blocks which are dependent due to some linking constraints Applications of this class of problems can be found, for instance, in multicommodity telecommunications networks, statistical data protection, multistage control and planning, and complex networks.

Solution strategies to deal with this class of problems can be broadly classified into simplex-based methods [14, 23], decomposition methods [17, 1, 2, 27], approximation methods [5], and interior-point methods [10, 21]. One of the most efficient interior-point methods (IPMs) for some classes of block-angular problems solves normal equations by a combination of Cholesky factorizations for the block constraints and preconditioned conjugate gradient (PCG) iterations for the linking constraints [10, 11]. The spectral radius of a certain matrix in the preconditioner, which is always in $[0, 1)$ plays an important role in the efficiency of this approach. It was observed that for separable convex problems with nonzero Hessians this spectral radius is reduced, and the PCG becomes more efficient [13]. When the spectral radius approaches 1, switching to a suitable preconditioner may be an efficient alternative [7]. It is worth noting that computing approximate Newton directions by PCG does not destroy the good convergence properties of IPMs, as shown in [20]. There is an extensive literature on the use of PCG within IPMs for other types of problems (e.g., [3, 4, 9, 18, 25, 28] to mention a few).

However, it is not yet clear why for some classes of block-angular problems the above approach may be very efficient (see, for instance, the results of [13, 12]), while it may need a large number of PCG iterations in others. The spectral radius may be used to monitor the good or bad behaviour, but an ex-ante explanation has to be found in the structural information of the block-angular constraints matrix. The purpose of this paper is to provide such a explanation by considering geometrical properties of the primal block-angular matrix structure which, in addition, may be used to improve the preconditioning in some instances. It will be shown that the principal angles between the subspaces generated by the diagonal blocks and the ones generated by the linking constraints play an important role, explaining the efficiency of the approach and providing a proper choice of the preconditioner.

This paper is organized as follows. Section 2 outlines the specialized IPM for primal block-angular problems. Section 3 analyzes the quality of preconditioning techniques based on geometrical and spectral properties. This analysis is used to obtain complementary preconditioners to deal with orthogonality and collinearity of the subspaces generated by the diagonal blocks and the ones generated by the linking constraints. Section 4 provides a numerical validation of the analysis in previous section. This is based on both a simulation analysis of the effect of geometrical relations on PCG iterations—Subsection 4.1—and a real world application to multicommodity network flow problems with nodal capacities—Subsection 4.2—, where the analyzed geometrical properties can be fully exploited.

*This work has been supported by MINECO-FEDER grants MTM2012-31440 and MTM2015-65362-R.

†Dept. of Statistics and Operations Research,Universitat Politècnica de Catalunya, Jordi Girona 1–3, 08034 Barcelona, Catalonia. jordi.castro@upc.edu

‡Department of Marketing, IESEG School of Management (LEM-CNRS 9221), 3 rue de la Digue, 59000 Lille, France. s.nasini@ieseg.fr

2. Outline of the IPM for primal block-angular problems. The primal block-angular formulation dealt with by the algorithm is

$$\min \quad \sum_{i=0}^{k} (c^i)^\top x^i \tag{2.1a}$$

$$\text{subject to} \quad \begin{bmatrix} N_1 & & & & \\ & N_2 & & & \\ & & \ddots & & \\ & & & N_k & \\ L_1 & L_2 & \dots & L_k & I \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^k \\ x^0 \end{bmatrix} = \begin{bmatrix} b^1 \\ b^2 \\ \vdots \\ b^k \\ b^0 \end{bmatrix} \tag{2.1b}$$

$$0 \le x^i \le u^i \quad i = 0, \dots, k. \tag{2.1c}$$

Matrices $N_i \in \mathbb{R}^{m_i \times n_i}$ and $L_i \in \mathbb{R}^{l \times n_i}$, $i = 1, \dots, k$, respectively define the block-diagonal and linking constraints, k being the number of blocks. Vectors $x^i \in \mathbb{R}^{n_i}, i = 1, \dots, k$, are the variables for each block. $x^0 \in \mathbb{R}^l$ are the slacks of the linking constraints. $b^i \in \mathbb{R}^{m_i}, i = 1, \dots, k$, is the right-hand side vector for each block of constraints, whereas $b^0 \in \mathbb{R}^l$ is for the linking constraints. The upper bounds for each group of variables are defined by $u^i \in \mathbb{R}^{n_i}, i = 0, \dots, k$. Problems with equality linking constraints can be formulated by setting $u^0 = 0$. The total number of constraints and variables of (2.1) is thus, respectively, $\widetilde{m} = l + \sum_{i=1}^{k} m_i$ and $\widetilde{n} = l + \sum_{i=1}^{k} n_i$. Formulation (2.1) is a very general model which accommodates to several block-angular problems.

In this work we consider the specialized IPM of [10, 11]. Briefly, this approach requires the solution at each interior-point iteration of the normal equations system $A\Theta A^\top \Delta y = g$, where $A \in \mathbb{R}^{\widetilde{m} \times \widetilde{n}}$ is the constraints matrix of (2.1b); $\Theta = (W(U-X)^{-1} + ZX^{-1})^{-1} \in \mathbb{R}^{\widetilde{n}}$ is a diagonal matrix computed from the values $(x, w, z) \in \mathbb{R}^{3\widetilde{n}}$ of the current primal-dual point—w and z being the Lagrange multipliers associated to, respectively, upper and lower bounds; $\Delta y \in \mathbb{R}^{\widetilde{m}}$ is the direction of movement for the Lagrange multipliers of equality constraints y; and $g \in \mathbb{R}^{\widetilde{m}}$ is some right-hand side. A derivation of the normal equations can be found in [29].

Exploiting the block structure of A and Θ we have

$$\begin{aligned} A\Theta A^\top \Delta y &= \left[\begin{array}{ccc|c} N_1\Theta_1 N_1^\top & & & N_1\Theta_1 L_1^\top \\ & \ddots & & \vdots \\ & & N_k\Theta_k N_k^\top & N_k\Theta_k L_k^\top \\ \hline L_1\Theta_1 N_1^\top & \dots & L_k\Theta_k N_k^\top & \Theta_0 + \sum_{i=1}^{k} L_i\Theta_i L_i^\top \end{array}\right] \Delta y \\ &= \begin{bmatrix} B & C \\ C^\top & D \end{bmatrix} \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \end{aligned} \tag{2.2}$$

where Δy_1 and Δy_2 are the components of Δy associated to, respectively, block and linking constraints; $\Theta_i = (W_i(U_i - X_i)^{-1} + Z_i X_i^{-1})^{-1}$, $i = 0, \dots, k$, are the blocks of Θ; and $g = (g_1^\top g_2^\top)^\top$ is a proper partition of the right-hand side g. By eliminating Δy_1 from the first group of equations of (2.2), we obtain

$$(D - C^\top B^{-1} C)\Delta y_2 = (g_2 - C^\top B^{-1} g_1) \tag{2.3a}$$

$$B\Delta y_1 = (g_1 - C\Delta y_2). \tag{2.3b}$$

The specialized IPM for this class of problems solves (2.3) by a combination of k Cholesky factorizations, for the systems involving B and PCG for (2.3a). Indeed, matrix $D - C^\top B^{-1} C \in \mathbb{R}^{l \times l}$ of (2.3a), whose dimension is the number of linking constraints, is symmetric and positive definite, since it is the Schur complement of the normal equations (2.2), which are symmetric and positive definite. System (2.3a) can thus be solved by PCG. A good preconditioner is instrumental. $D - C^\top B^{-1} C$ is a *P-regular splitting*, i.e., it is symmetric and positive definite, D is nonsingular and $D + C^\top B^{-1} C$ is positive definite. Therefore the *P-regular splitting theorem* [26] guarantees that

$$0 < \rho(D^{-1}(C^\top B^{-1} C)) < 1, \tag{2.4}$$

where $\rho(\cdot)$ denotes the spectral radius of a matrix (i.e., the maximum absolute eigenvalue). This allows us to compute the inverse of $D - C^\top B^{-1} C$ as the following infinite power series (see [10, Prop.

4] for a proof).

$$(D - C^\top B^{-1} C)^{-1} = \left(\sum_{i=0}^{\infty} (D^{-1}(C^\top B^{-1} C))^i \right) D^{-1}. \tag{2.5}$$

A preconditioner is thus obtained by truncating the infinite power series (2.5) at some term. The more the terms included, the better the preconditioner will be, at the expense of increasing the execution time of each PCG iteration. If we only include the first term, or the first and second terms, of the infinite power series (2.5) the resulting preconditioners will be, respectively, D^{-1} or $(I + D^{-1}(C^\top B^{-1} C))D^{-1}$. As shown in [12], in most test problems D^{-1} was the best option for the tradeoff between being a good and an efficient preconditioner. Thus, in this work we will focus on the first term preconditioner D^{-1}. Although the expected performance for a general primal block-angular matrix is problem dependent, the effectiveness of the preconditioner obtained by truncating the infinite power series (2.5) is governed by the spectral radius $\rho(D^{-1}(C^\top B^{-1} C))$—the farther from 1, the better [13]—and the structure of D, as each PCG iteration requires the solution of a system with this matrix.

The spectral radius tends to approach 1 when the IPM is close to the optimal solution [13]. For this reason a hybrid preconditioner was introduced in [7]: the power series preconditioner (either D^{-1} or $(I + D^{-1}(C^\top B^{-1} C))D^{-1}$) is used in the initial IPM iterations, switching to the *splitting preconditioner* [25] when the former becomes inefficient. The above hybrid scheme is supported by the known good behaviour of the splitting preconditioner near the solution of the linear optimization problem. Briefly, given the normal equations matrix $A\Theta A^\top$, the splitting preconditioner is based on a partition of the columns of A into basic and nonbasic columns, forming the nonsingular basis matrix $A_B \in \mathbb{R}^{\tilde{m} \times \tilde{m}}$ and the nonbasic one $A_N \in \mathbb{R}^{\tilde{m} \times (\tilde{n} - \tilde{m})}$, respectively. Applying the same partition to Θ, the normal equations matrix can be rewritten as

$$A\Theta A^\top = A_B \Theta_B A_B^\top + A_N \Theta_N A_N^\top. \tag{2.6}$$

The splitting preconditioner is defined as $\Theta_B^{-1/2} A_B^{-1}$. The symmetric application of this preconditioner to $A\Theta A^\top$ gives:

$$\begin{aligned} (\Theta_B^{-1/2} A_B^{-1})(A\Theta A^\top)(\Theta_B^{-1/2} A_B^{-1})^\top &= \Theta_B^{-1/2} A_B^{-1}(A_B \Theta_B A_B^\top + A_N \Theta_N A_N^\top) A_B^{-T} \Theta_B^{-1/2} \\ &= I + (\Theta_B^{-1/2} A_B^{-1} A_N \Theta_N^{1/2})(\Theta_B^{-1/2} A_B^{-1} A_N \Theta_N^{1/2})^\top. \end{aligned} \tag{2.7}$$

Sufficiently close to an optimal solution, with a suitable choice of the columns of A_B, the diagonal entries of Θ_B^{-1} and Θ_N are very small and $(\Theta_B^{-1/2} A_B^{-1} A_N \Theta_N^{1/2})$ approaches the zero matrix. Some strategies for identifying a suitable matrix A_B are discussed in [25, 7].

This specialized IPM has been recently implemented in a software package called BlockIP [12], that will be used for the computational results of this paper.

3. Analysis of preconditioning techniques for block-angular problems. By the definition of B, C and D in (2.2), we may write

$$\begin{aligned} C^\top B^{-1} C &= \sum_{i=1}^{k} L_i \Theta_i N_i^\top \, (N_i \Theta_i N_i^\top)^{-1} \, N_i \Theta_i L_i^\top \\ &= \sum_{i=1}^{k} L_i \Theta_i^{1/2} P_i \Theta_i^{1/2} L_i^\top \end{aligned} \tag{3.1}$$

where

$$P_i = \Theta_i^{1/2} N_i^\top \, (N_i \Theta_i N_i^\top)^{-1} \, N_i \Theta_i^{1/2} \quad i = 1, \ldots, k \tag{3.2}$$

is the projection operator onto $\mathcal{R}(\Theta_i^{1/2} N_i^\top)$, the range or column space of $\Theta_i^{1/2} N_i^\top$. Let us consider the following two opposite cases:

1. If each column of $\Theta_i^{1/2} L_i^\top$ belongs to $\mathcal{R}(\Theta_i^{1/2} N_i^\top)$ for $i = 1, \ldots, k$ (which is equivalent to saying that the rows of $L_i \Theta_i^{1/2}$ might be written as a linear combination of the rows of $N_i \Theta_i^{1/2}$) then $P_i \Theta_i^{1/2} L_i^\top = \Theta_i^{1/2} L_i^\top$, as P_i is the identity operator of the subspace generated

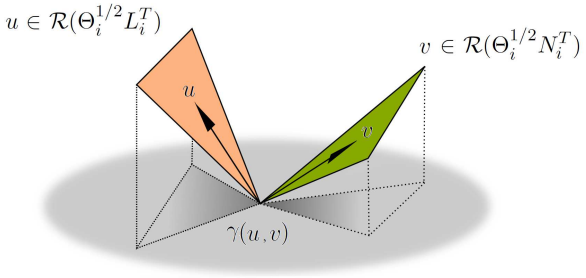

FIG. 1. *Angles between subspaces.*

by the columns of $\Theta_i^{1/2} N_i^\top$, and

$$\begin{aligned} D - C^\top B^{-1} C &= D - \sum_{i=1}^{k} L_i \Theta_i^{1/2} P_i \Theta_i^{1/2} L_i^\top \\ &= D - \sum_{i=1}^{k} L_i \Theta_i L_i^\top = \Theta_0. \end{aligned} \tag{3.3}$$

2. On the contrary, if each column of $\Theta_i^{1/2} L_i^\top$ belongs to $\mathcal{N}(N_i \Theta_i^{1/2})$, the null space of $N_i \Theta_i^{1/2}$, for $i = 1, \ldots, k$, then $P_i \Theta_i^{1/2} L_i^\top = 0$ and

$$D - C^\top B^{-1} C = D - \sum_{i=1}^{k} L_i \Theta_i^{1/2} P_i \Theta_i^{1/2} L_i^\top = D. \tag{3.4}$$

Thus Θ_0^{-1} and D^{-1} would be the inverse of $D - C^\top B^{-1} C$ (i.e., the exact preconditioners) when, for $i = 1 \ldots k$, each column of $\Theta_i^{1/2} L_i^\top$ belongs to either $\mathcal{R}(\Theta_i^{1/2} N_i^\top)$ or $\mathcal{N}(N_i \Theta_i^{1/2})$, respectively. Needless to say, these two extreme cases will rarely appear in a real problem; for instance, when $\Theta_i^{1/2} L_i^\top$ belongs to $\mathcal{R}(\Theta_i^{1/2} N_i^\top)$ the linking constraints are redundant and can be removed, obtaining a block-separable problem. However, as seen in next subsections, they allow to decide whether D^{-1} or Θ_0^{-1} will be a good preconditioner for (2.3a), and which of them will theoretically behave better.

3.1. Geometrical and spectral properties.

According to previous discussion, the goodness of the approximation of Θ_0^{-1} and D^{-1} to $(D - C^\top B^{-1} C)^{-1}$ might be measured by the *principal angles* between the range spaces of $\Theta_i^{1/2} N_i^\top$ and $\Theta_i^{1/2} L_i^\top$, for $i = 1, \ldots, k$. The principal angles provide information about the relative position of two subspaces of an inner product space. Consider two subspaces $\mathcal{L}_\Theta$ and $\mathcal{N}_\Theta$ of $\mathbb{R}^n$, with $\dim \mathcal{L}_\Theta = l$, $\dim \mathcal{N}_\Theta = m$, and $q = \min\{l, m\}$. The principal angles between $\mathcal{L}_\Theta$ and $\mathcal{N}_\Theta$, denoted as $0 \leq \gamma_1 \leq \ldots \leq \gamma_q \leq \pi/2$, are obtained by solving for $j = 1, \ldots, q$ this sequence of optimization problems [6][19, Chapter 12]:

$$\cos(\gamma_j) = \max_{u,v} \quad u^\top v \tag{3.5}$$

$$\text{subject to} \quad \|u\| = 1, \ \|v\| = 1 \tag{3.6}$$

$$u \in \mathcal{L}_\Theta, \ v \in \mathcal{N}_\Theta \tag{3.7}$$

$$v^\top v_k = 0, \ u^\top u_k = 0, \ k = 1 \ldots j-1, \tag{3.8}$$

$u_j \in \mathbb{R}^n$ and $v_j \in \mathbb{R}^n$ being the optimal vectors. In other words, for $j = 1$ the procedure finds the unit vectors $u_1 \in \mathcal{L}_\Theta$ and $v_1 \in \mathcal{N}_\Theta$ which minimize the angle between them—named γ_1. For $j > 1$, the procedure considers the orthogonal complements of $\text{span}\{u_1, \ldots, u_{j-1}\}$ in $\mathcal{L}_\Theta$ and of $\text{span}\{v_1, \ldots, v_{j-1}\}$ in $\mathcal{N}_\Theta$, and computes a new pair of vectors u_j, v_j in the above subspaces minimizing the angle γ_j. In our case, we have that $\mathcal{L}_\Theta = \mathcal{R}(\Theta^{1/2} L^\top)$ and $\mathcal{N}_\Theta = \mathcal{R}(\Theta^{1/2} N^\top)$, where $N = N_i$, $L = L_i$ and $\Theta = \Theta_i$ for some $i \in \{1, \ldots, k\}$. The vectors $\{u_1, \ldots, u_q\}$ and $\{v_1, \ldots, v_q\}$ are called *principal vectors*, associated to principal angles $\{\gamma_1, \ldots, \gamma_q\}$. The principal angles between subspaces can be graphically depicted as in Figure 1. If $q = m = l$, the distance between the equidimensional subspaces $\mathcal{L}_\Theta$ and $\mathcal{N}_\Theta$ is defined as $\sin \gamma_q = \sqrt{1 - \cos^2 \gamma_q}$ [19, Chapter 2]. The Appendix shows a procedure to compute principal angles based on the singular value decomposition. This procedure is

prohibitive for large optimization problems, but, as it will be shown later, it is not required in our optimization method.

As $\Theta_i, i = 1, \ldots, k$, are different at each interior-point iteration, the principal angles between the subspaces $\mathcal{L}_\Theta$ and $\mathcal{N}_\Theta$ also change. Accordingly, the goodness of the approximation of Θ_0^{-1} and D^{-1} to $(D - C^\top B^{-1} C)^{-1}$ dynamically changes along the interior-point iterations. Proposition 3.1 below provides a lower bound of the cosine of principal angles of $\mathcal{L}_\Theta$ and $\mathcal{N}_\Theta$ as a function of the principal angles of $\mathcal{L} = \mathcal{L}_I$ and $\mathcal{N} = \mathcal{N}_I$, that is, the subspaces $\mathcal{L} = \mathcal{R}(L_i^\top)$ and $\mathcal{N} = \mathcal{R}(N_i^\top)$ defined by matrices L_i and N_i without the scaling matrix Θ_i.

PROPOSITION 3.1. *Consider the subspaces $\mathcal{L} = \mathcal{R}(L^\top) \subseteq \mathbb{R}^n$ and $\mathcal{N} = \mathcal{R}(N^\top) \subseteq \mathbb{R}^n$, and the associated subspaces $\mathcal{L}_\Theta = \mathcal{R}(\Theta^{1/2} L^\top) \subseteq \mathbb{R}^n$ and $\mathcal{N}_\Theta = \mathcal{R}(\Theta^{1/2} N^\top) \subseteq \mathbb{R}^n$ under the linear transformation $\Theta^{1/2}$, where $L = L_i$, $N = N_i$ and $\Theta = \Theta_i$ for some $i \in \{1, \ldots, k\}$, Θ thus being a positive diagonal matrix. Let $\gamma(u, v)$ be the first principal angle between $\mathcal{L}$ and $\mathcal{N}$, associated to principal vectors u and v; and γ_Θ the first principal angle between $\mathcal{L}_\Theta$ and $\mathcal{N}_\Theta$. If $\mathcal{L}$ and $\mathcal{N}$ are not orthogonal, then*

$$\cos(\gamma_\Theta) \geq \max\left\{0, \frac{\Theta_c}{\Theta_{max}} \cos\left(\gamma(u, v)\right)\right\}, \tag{3.9}$$

where $\Theta_{\min}$ and $\Theta_{\max}$ are the smallest and largest diagonal components of Θ, and $\Theta_c \in [-\Theta_{\max}, \Theta_{max}]$. If in addition $u_i v_i \geq 0$ for all $i = 1, \ldots, n$ then we have

$$\cos(\gamma_\Theta) \geq \frac{\Theta_{\min}}{\Theta_{\max}} \cos\left(\gamma(u, v)\right) > 0. \tag{3.10}$$

Proof. Let $\gamma\left(\frac{\Theta^{1/2}u}{\|\Theta^{1/2}u\|}, \frac{\Theta^{1/2}v}{\|\Theta^{1/2}v\|}\right)$ be the angle between the unitary transformed vectors $\frac{\Theta^{1/2}u}{\|\Theta^{1/2}u\|} \in \mathcal{L}_\Theta$ and $\frac{\Theta^{1/2}v}{\|\Theta^{1/2}v\|} \in \mathcal{N}_\Theta$. From the definition of principal angles as resulting from the maximization problem (3.5)–(3.8), and since vectors $\frac{\Theta^{1/2}u}{\|\Theta^{1/2}u\|}$ and $\frac{\Theta^{1/2}v}{\|\Theta^{1/2}v\|}$ are feasible for this problem, we have

$$\cos(\gamma_\Theta) \geq \cos\left(\gamma\left(\frac{\Theta^{1/2}u}{\|\Theta^{1/2}u\|}, \frac{\Theta^{1/2}v}{\|\Theta^{1/2}v\|}\right)\right). \tag{3.11}$$

Since u, v are principal vectors, by (3.6) $\|u\| = \|v\| = 1$. Moreover, since principal angles are always in $[0, \pi/2]$, and we are assuming $\mathcal{L}$ and $\mathcal{N}$ are not orthogonal, we have

$$0 < u^\top v = \|u\| \, \|v\| \cos(\gamma(u, v)) = \cos(\gamma(u, v)) \leq 1. \tag{3.12}$$

By definition of inner product, by (3.12), using that for any square matrix M and vector w $\|Mw\| \leq \|M\| \, \|w\|$, and that $\sqrt{\Theta_{\max}} \geq \|\Theta^{1/2}\|$,

$$\begin{aligned} \cos\left(\gamma\left(\frac{\Theta^{1/2}u}{\|\Theta^{1/2}u\|}, \frac{\Theta^{1/2}v}{\|\Theta^{1/2}v\|}\right)\right) &= \frac{u^\top \Theta v}{\|\Theta^{1/2}u\| \, \|\Theta^{1/2}v\|} \geq \frac{u^\top \Theta v}{\|\Theta^{1/2}\|^2 \, \|u\| \, \|v\|} \geq \\ &\geq \frac{1}{\Theta_{\max}} \frac{u^\top \Theta v}{u^\top v} \cos(\gamma(u, v)) \geq \frac{u^\top \Theta v}{\Theta_{\max}} \cos(\gamma(u, v)). \end{aligned} \tag{3.13}$$

Defining the set $\mathcal{P} = \{i \in \{1, \ldots, n\} : u_i v_i \geq 0\}$, since $\|u\| = \|v\| = 1$ and $u^\top v \leq 1$, we have that

$$0 \leq \sum_{i \in \mathcal{P}} u_i v_i \leq 1 \qquad \text{and} \qquad -1 \leq \sum_{i \notin \mathcal{P}} u_i v_i \leq 0. \tag{3.14}$$

Then, by (3.14),

$$u^\top \Theta v = \sum_{i \in \mathcal{P}} \Theta_i u_i v_i + \sum_{i \notin \mathcal{P}} \Theta_i u_i v_i \geq \Theta_{\max} \sum_{i \notin \mathcal{P}} u_i v_i \geq -\Theta_{\max},$$

and similarly,

$$u^\top \Theta v = \sum_{i \in \mathcal{P}} \Theta_i u_i v_i + \sum_{i \notin \mathcal{P}} \Theta_i u_i v_i \leq \Theta_{\max} \sum_{i \in \mathcal{P}} u_i v_i \leq \Theta_{\max}.$$

Then, there exists some (unkwon) $\Theta_c \in [-\Theta_{\max}, \Theta_{\max}]$ such that

$$\Theta_c = u^\top \Theta v. \tag{3.15}$$

Applying (3.15) in (3.13), and since, by definition, $\cos(\gamma_\Theta) \geq 0$, inequality (3.9) is proved.

If $u_i v_i \geq 0$ for all $i = 1, \ldots, n$, we have

$$\frac{u^\top \Theta v}{u^\top v} \geq \frac{\Theta_{\min} u^\top v}{u^\top v} = \Theta_{\min}. \tag{3.16}$$

Applying (3.16) to the penultimate term of (3.13) we get (3.10). ∎

An important consequence of Proposition 3.1 is that we have no information about the principal angles in the final iterations of the IPM, as $\Theta_{\min}/\Theta_{\max}$ can approach 0 and $\Theta_c/\Theta_{\max}$ can be any number in $[-1, 1]$ when the iterative process gets close to the optimal solution. In the worst case, it may even happen that almost-collinear subspaces $\mathcal{L}$ and $\mathcal{N}$ become almost-orthogonal for some Θ.

It is worth noting that the two extreme cases have opposite behaviour:

- If $\mathcal{L}$ and $\mathcal{N}$ are orthogonal (that is, their principal angles are $\pi/2$), then, for a diagonal Θ matrix such that some component $\Theta_{i,i}$ is very large, and for other components $j \neq i$ $\Theta_{jj} \approx 0$, the principal angles of $\mathcal{L}_\Theta$ and $\mathcal{N}_\Theta$ will become close to 0. This is a limit case of (3.10). In such a situation, D^{-1} would be the best preconditioner for the first interior-point iterations, while Θ_0^{-1} should be used for the last ones. However, we empirically observed that, for the instances considered in this work, large principal angles in the first interior-point iterations may not tend to 0 as the algorithm approaches the solution. This will be illustrated in below Subsection 3.2.
- If $\mathcal{L} \subseteq \mathcal{N}$, then the principal angles are 0. This means that, for some matrix $Y \in \mathbb{R}^{m \times l}$, $L^\top = N^\top Y$, thus $(\Theta^{1/2} L^\top) = (\Theta^{1/2} N^\top) Y$ and the principal angles of $\mathcal{L}_\Theta$ and $\mathcal{N}_\Theta$ will also be 0. Although this case is not of practical interest—since if $\mathcal{L} \subseteq \mathcal{N}$ then the linking constraints are redundant and can be removed—it can explain, as we observed, why small principal angles in the initial interior-point iterations remain small near the optimal solution.

As shown below, the goodness of the dynamical approximation of Θ_0^{-1} and D^{-1} to $(D - C^\top B^{-1} C)^{-1}$ along the interior-point iterations is related to the changes in the spectral radius of matrix $D^{-1}(C^\top B^{-1} C)$—which is always in $[0, 1)$ [10, Theorem 1]. Since D^{-1} is the first term of the power series (2.5), the farther away from 1 is the spectral radius of $D^{-1}(C^\top B^{-1} C)$ the better is the quality of the approximation of the first few terms of (2.5). Although the particular behavior of the spectral radius value is problem dependent, in general, it comes closer to 1 as we approach the optimal solution, because of the ill-conditioning of the Θ matrix. Next result provides a clear relationship between the spectral radius of $D^{-1}(C^\top B^{-1} C)$ and the projection operators in the subspaces $\mathcal{L}_\Theta$ and $\mathcal{N}_\Theta$.

PROPOSITION 3.2. *Let λ be an arbitrary eigenvalue of $D^{-1}(C^\top B^{-1} C)$. If each column of $\Theta_i^{1/2} L_i^\top$ belongs to $\mathcal{R}(\Theta_i^{1/2} N_i^\top)$ then*

$$\lambda = \frac{r^\top \left(\sum_{i=1}^k L_i \Theta_i L_i^\top \right) r}{r^\top D r}, \tag{3.17}$$

where r is an eigenvector associated to λ. On the contrary, if each column of $\Theta_i^{1/2} L_i^\top$ belongs to $\mathcal{N}(N_i \Theta_i^{1/2})$, the null space of $N_i \Theta_i^{1/2}$, then

$$\lambda = 0. \tag{3.18}$$

Proof. Eigenvalue λ of $D^{-1}(C^\top B^{-1} C)$ satisfies $(C^\top B^{-1} C) r = \lambda D r$ for some eigenvector r. From the definition of C, B, D in (2.2) and projection matrices P_i in (3.2) we have

$$\left(\sum_{i=1}^k L_i \Theta_i N_i^\top \left(N_i \Theta_i N_i^\top \right)^{-1} N_i \Theta_i L_i^\top \right) r = \lambda D r$$

is equivalent to

$$(1 - \lambda) D r = \left(\Theta_0 + \sum_{i=1}^k L_i \Theta_i L_i^\top \right) r - \left(\sum_{i=1}^k L_i \Theta_i N_i^\top \left(N_i \Theta_i N_i^\top \right)^{-1} N_i \Theta_i L_i^\top \right) r,$$

which implies

$$(1 - \lambda) = \frac{r^\top \Theta_0 r + r^\top \left(\sum_{i=1}^k L_i \Theta_i^{1/2} (I - P_i) \Theta_i^{1/2} L_i^\top \right) r}{r^\top D r}. \tag{3.19}$$

From this expression and applying the definition of $W_{\mathcal{R}}$ and $W_{\mathcal{N}}$

$$W_{\mathcal{N}} = \sum_{i=1}^{k} L_i \Theta_i^{1/2} (I - P_i) \Theta_i^{1/2} L_i^\top \quad \text{and} \quad W_{\mathcal{R}} = \sum_{i=1}^{k} L_i \Theta_i^{1/2} P_i \Theta_i^{1/2} L_i^\top, \tag{3.20}$$

we find the following identities

$$\lambda = \frac{r^\top \left(\sum_{i=1}^{k} L_i \Theta_i L_i^\top \right) r - r^\top W_{\mathcal{N}} r}{r^\top D r} = \frac{r^\top W_{\mathcal{R}} r}{r^\top D r}. \tag{3.21}$$

It turns out that $r^\top W_{\mathcal{N}} r = 0$ when each column of $\Theta_i^{1/2} L_i^\top$ belongs to $\mathcal{R}(\Theta_i^{1/2} N_i^\top)$; and $r^\top W_{\mathcal{R}} r = 0$ when each column of $\Theta_i^{1/2} L_i^\top$ belongs to $\mathcal{N}(N_i \Theta_i^{1/2})$. The proof is complete. □

Note that a clear lesson from Proposition 3.2 is that the spectral radius might be small even in the case of almost collinearity between the rows of $L_i \Theta_i^{1/2}$ and their projections into $\mathcal{R}(\Theta_i^{1/2} N_i^\top)$, when $r^\top \Theta_0 r \gg r^\top (\sum_{i=1}^{k} L_i \Theta_i L_i^\top) r$. This means that, although Θ_0 is in theory a better preconditioner in case of almost collinearity, in practice it can be even outperformed by D. This is consistent with some of the computational results of Subsections 4.1–4.2, and with Theorem 1 of [13], which stated that the spectral radius ρ of $D^{-1}(C^\top B^{-1} C)$ is bounded by

$$0 \leq \rho \leq \max_{j \in \{1,\ldots,l\}} \frac{\gamma_j}{\left(\frac{u_j}{v_j}\right)^2 \Theta_{0j} + \gamma_j} < 1, \tag{3.22}$$

where u is the eigenvector (or one of the eigenvectors) of $D^{-1}(C^\top B^{-1} C)$ for ρ; γ_j, $j = 1, \ldots, l$, and $V = [V_1 \ldots V_l]$, are respectively the eigenvalues and matrix of columnwise eigenvectors of $\sum_{i=1}^{k} L_i \Theta_i {L_i}^\top$; $v = V^\top u$ (and, abusing of notation, we assume that for $v_j = 0$, $(u_j/v_j)^2 = +\infty$). Clearly, from (3.22), the larger Θ_0, the closer ρ to 0, which it is also concluded from (3.17).

3.2. Practical estimation of principal angles. The use of principal angles between subspaces in our context has two downsides:

- If the subspace spanned by linking constraints is $\mathbb{R}^{n_i}$, that is, if $\mathcal{R}(L_i^\top) = \mathbb{R}^{n_i}$, for $i = 1, \ldots, k$, then $\mathcal{N}_i \subseteq \mathcal{L}_i$. In such case the principal angles are always zero, regardless of what $\mathcal{R}(N_i^\top)$ is. This happens, for instance, in multicommodity network flow problems [10], edge-colored network problems [15], and cutting planes approaches for facility location problems [16], where $L_i = I$, for $i = 1, \ldots, k$. Our interest is on whether the linking constraints belong to $\mathcal{R}(N_i^\top)$, equivalent to saying that the rows of L_i may be written as a linear combination of the rows of N_i, whereas the principal angles between $\mathcal{R}(L_i^\top)$ and $\mathcal{R}(N_i^\top)$ might be seen as a measure of *mutual dependency*, in the sense that they go to zero even in the opposite case when the rows of N_i are linear combinations of the ones of L_i. The latter condition does not imply any favorable argument for Θ_0 as a preconditioner.
- As already stated in previous subsection, the computation of principal angles between $\mathcal{L}_\Theta$ and $\mathcal{N}_\Theta$ using the algorithm of Figure 7 in the Appendix can be totally prohibitive to be included as part of the optimization procedure.

To circumvent the above drawbacks, two sensible and more practical approaches will be described below. Both are based on the computation of principal angles between a subspace $\mathcal{L}_{\Theta_i} = \mathcal{R}(\Theta_i^{1/2} L^\top)$ of dimension 1 (that is, $L \in \mathbb{R}^{1 \times n_i}$) and $\mathcal{N}_{\Theta_i} = \mathcal{R}(\Theta_i^{1/2} N_i^\top)$ (where $N_i \in \mathbb{R}^{m_i \times n_i}$). In this case, the problem (3.5)–(3.8) is highly simplified. First, $q = \min\{1, m_i\} = 1$, and then we do not have constraints (3.8) (i.e., there is only a principal angle to compute). Second, constraints (3.6) and (3.7) impose

$$u = \Theta_i^{1/2} L^\top \alpha, \text{ for some } \alpha \in \mathbb{R}, \text{ and } 1 = u^\top u = \alpha^2 L \Theta_i L^\top, \text{ such that } \alpha = \frac{1}{\sqrt{L \Theta_i L^\top}}. \tag{3.23}$$

Similarly, we have

$$v = \Theta_i^{1/2} N_i^\top x, \text{ for some } x \in \mathbb{R}^{m_i}, \text{ and } v^\top v = x^\top N_i N_i^\top x = 1. \tag{3.24}$$

Then, using (3.23) and (3.24) the optimization problem (3.5)–(3.8) reduces to

$$\cos(\gamma) = \begin{array}{ll} \max\limits_{x \in \mathbb{R}^{m_i}} & u^\top v = c^\top x \\ \text{subject to} & x^\top N_i N_i^\top x = 1 \end{array} \qquad \text{where} \quad c = \frac{N_i \Theta_i L}{\sqrt{L \Theta_i L^\top}},$$

whose solution can be directly computed, considering a Lagrange multiplier $\nu \in \mathbb{R}$ for the constraint, as

$$x^* = (N_i\Theta_i N_i^\top)^{-1}\frac{c}{2\nu^*} \qquad \text{where} \qquad \nu^* = \frac{1}{2}\sqrt{c^\top (N_i\Theta_i N_i^\top)^{-1}c}\,. \tag{3.25}$$

Since for the principal angle we only need the optimal value of the objective function $\cos(\gamma) = c^\top x^*$, computations can be simplified by noting that

$$\cos(\gamma) = c^\top x^* = \frac{c^\top (N_i\Theta_i N_i^\top)^{-1}c}{\sqrt{c^\top (N_i\Theta_i N_i^\top)^{-1}c}} = \sqrt{c^\top (N_i\Theta_i N_i^\top)^{-1}c}\,. \tag{3.26}$$

The cost of solving (3.26) is equivalent to the solution of a system of equations with one of the blocks defining matrix B in (2.2), whose factorization is already available at each interior-point iteration.

The two practical approaches considered for computing principal angles, which rely on (3.26), are as follows:

- For each block i, $i = 1,\ldots,k$, and linking constraint j, $j = 1,\ldots,l$, we solve (3.26) for $L = L_{i,j}$, that is, we compute the principal angle—denoted as $\gamma_{i,j}$—between N_i and the j-th row of L_i. Then the reported principal angle is the average of principal angles:

$$\gamma = \frac{\sum_{i=1}^{k}\sum_{j=1}^{l}\gamma_{i,j}}{kl}. \tag{3.27}$$

  The inconvenience of this approach is that it solves kl problems (3.26). A good feature of this method is that it averages angles between real linking constraints and block matrices.
- To reduce the number of problems (3.26) to be solved, in this second method we compute, for each block i, $i = 1,\ldots,k$, the principal angle between N_i and the average of the rows of L_i, that is we solve (3.26) for $L = \bar{L}_i = (\sum_{j=1}^{l} L_{i,j})/l$. If we denote this angle as $\bar{\gamma}_i$, then the final angle provided is

$$\gamma = \frac{\sum_{i=1}^{k}\bar{\gamma}_i}{k}. \tag{3.28}$$

  This approach only requires solving k problems (3.26). However, it provides some sort of approximation of the principal angle, since it does not use the rows of L_i, but a convex combination of them. This can be justified by noting that, if the principal angle between every row $L_{i,j}$, $j = 1,\ldots,l$, and N_i is 0 (that is, $L_{i,j}$ is linearly dependent with the rows of N_i) then the principal angle between a convex combination of rows of L_i and N_i will also be 0.

The above two approaches have been implemented in the optimization package used for the computational experiments. Figure 2 shows the estimation of principal angles obtained with either (3.27) or (3.28) during the interior-point iterations of four of the instances of Tables 3–6 of Subsection 4.2. We observe that, for the first interior-point iterations, both methods provide similar values. However, when we approach the optimal solution, angles computed with (3.28) tend to go to 0, while those obtained with (3.27) in general remain close to the original values. From Table 7 of Subsection 4.2—which provides the true angles for these instances, computed using the costly algorithm in Figure 7 of the Appendix—we see that angles computed with (3.27) are usually better estimations. Unfortunately, in the huge instances of Subsection 4.2, method (3.27) was prohibitive (it resulted in very high CPU times), so we had to (successfully) apply method (3.28). The estimations with (3.27) in Figure 2 also confirm, as discussed in previous subsection, that although the scaling matrices Θ_i may change the principal angles at each interior-point iteration, in general when the angles are large at the first iterations, they remain large near the optimal solution.

There are other possible methods to reduce the computational cost of (3.27), such as, for instance, to consider just a sample or a representative subset of rows of L_i. However, we did not try these other options because, at least for the instances tested in this work, the results obtained with (3.28) were satisfactory.

According to the above discussion, if we want to predict from the beginning which preconditioner will behave better based on principal angles, we can use either (3.27) or (3.28), since both methods provide similar estimations at the first interior-point iterations. It is worth to mention that in this case (3.26) is solved before the IPM has started, and then $\Theta_i = I$ has to be used. Moreover, since Θ_i are the same for all $i = 1,\ldots,k$, if matrices N_i and L_i are also the same for all i, then the number

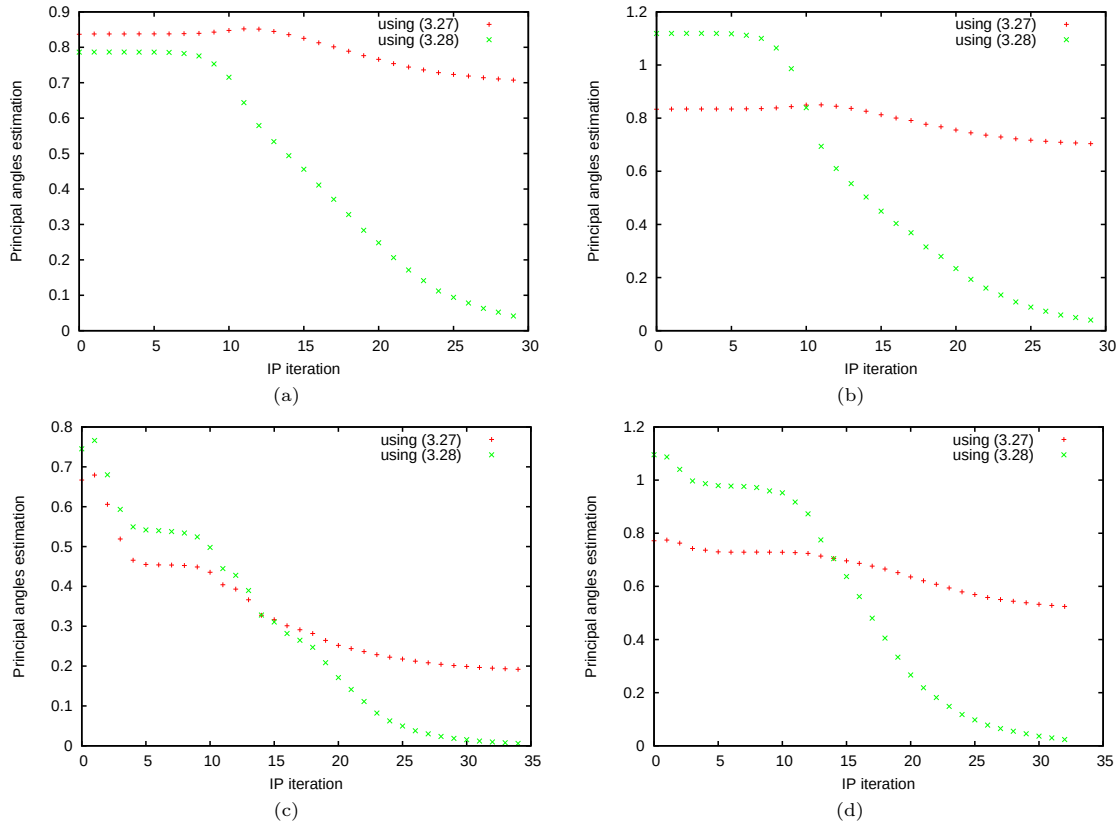

FIG. 2. *Estimation of principal angles (in vertical axes) computed with either (3.27) or (3.28) along the interior-point iterations (in horizontal axes), for four particular instances. Plots (a) and (b) correspond to the first instances of, respectively, Tables 3 and 4 of Subsection 4.2. Plots (c) and (d) were obtained by adding extra constraints to the first instances of Tables 5 and 6 of Subsection 4.2.*

of problems (3.26) to be solved drops to l and 1, for, respectively, (3.27) and (3.28). We considered the following rule for automatic selection of the preconditioner:

$$\begin{array}{l} \text{if } \gamma \geq \tau_\gamma \text{ then use } D^{-1}, \\ \text{if } \gamma < \tau_\gamma \text{ then use } \Theta_0^{-1}, \end{array} \tag{3.29}$$

where γ has been previously computed with either (3.27) or (3.28) using $\Theta_i = I$, and τ_γ is a threshold parameter provided by the user. In the computational experiments of Subsection 4.2 we used $\tau_\gamma = 0.95\pi/4$, i.e., if the angle is closer to 0 than to $\pi/2$, we choose Θ_0^{-1}, otherwise D^{-1} is selected; the 0.95 coefficient guarantees that, in case of tie, D^{-1} will be used, since, according to (2.5), it is the first term of the exact preconditioner.

Switching between preconditioners at some interior-point iteration will only be worth if principal angles significantly change with Θ_i. However, according to the above discussion, when the initial principal angles are close enough to 0, Θ_0^{-1} will be in general the best preconditioner, not only for the first, but for all the interior-point iterations. Otherwise, D^{-1} will be in general preferred. In addition, the CPU time required for computing principal angles is not negligible for huge instances. Given all these considerations, our switching criteria only computed principal angles if strictly necessary, and it focused on other indicators such as (1) the number of PCG iterations performed with the current preconditioner in the last interior-point iteration; (2) and, if the current preconditioner is D^{-1}, the spectral radius ρ (which can be easily computed using the procedure described in [7]). The implemented switching criteria, which is applied every τ_t interior-point iterations, is as follows:

$$\begin{array}{l} \text{if current preconditioner is } \Theta_0^{-1} \text{ then} \\ \quad \text{if (PCG iterations} > \tau_P \cdot l) \text{ and } (\gamma \geq \tau_\gamma) \text{ then switch to } D^{-1}, \\ \text{else if current preconditioner is } D^{-1} \text{ then} \\ \quad \text{if (PCG iterations} > \tau_P \cdot l) \text{ and } (\rho > \tau_\rho) \text{ and } (\gamma < \tau_\gamma) \text{ then switch to } \Theta_0^{-1}, \end{array} \tag{3.30}$$

where τ_ρ is a threshold value for the spectral radius, τ_P is a percentage of the number of linking constraints l (the size of the system solved by PCG), and τ_γ was already used in (3.29). The values

TABLE 1
CPU time of BlockIP using D^{-1} and Θ_0^{-1} preconditioners. The instances have $l = 100$ linking constraints and $k = 100$ equal diagonal block matrices $N \in \mathbb{R}^{10\times 50}$.

α	$\overline{r}$	CPU time		IPM iterations		average PCG iter.	
		Θ_0^{-1}	D^{-1}	Θ_0^{-1}	D^{-1}	Θ_0^{-1}	D^{-1}
$\pi/14$	2	0.99	1.47	30	43	2.84	5.87
$\pi/10$	4	1.27	1.54	38	44	4.94	6.29
$\pi/6$	6	1.30	1.58	36	46	9.00	6.05
$\pi/2$	8	1.29	1.27	35	39	8.91	5.05

TABLE 2
CPU time of BlockIP using D^{-1} and Θ_0^{-1} preconditioners. The instances have $l = 200$ linking constraints and $k = 1000$ equal diagonal block matrices $N \in \mathbb{R}^{20\times 200}$.

α	$\overline{r}$	CPU time		IPM iterations		average PCG iter.	
		Θ_0^{-1}	D^{-1}	Θ_0^{-1}	D^{-1}	Θ_0^{-1}	D^{-1}
$\pi/14$	2	191.52	547.21	28	77	1.85	7.08
$\pi/10$	4	512.61	479.12	80	61	6.40	12.05
$\pi/6$	6	496.39	574.33	64	76	17.50	11.75
$\pi/2$	8	565.52	497.19	67	70	26.29	11.85

used in the computational results of Subsection 4.2 were $\tau_t = 3$, $\tau_P = 0.2$, and $\tau_\rho = 0.95$. As it will be observed in the computational results this switching criteria is rarely applied (only in three out of 60 instances). Choosing the right preconditioner, either D^{-1} or Θ_0^{-1}, from the beginning turned out to be a more important factor in the performance of the IPM. We remark that this situation is quite different from other hybrid preconditioning approaches, such as, for instance, the one in [7]: in that approach the two different preconditioners are known in advance to behave better for, respectively, the first and last interior-point iterations, and thus (1) we know in advance which preconditioner to start with; and (2) the switching is usually performed.

4. Numerical validation. The numerical validation of the proposed preconditioning technique is based on two types of computational experiments: (1) assessing the effect of the described geometrical relations to the number of PCG iterations; (2) analyzing the global numerical performance of both preconditioning techniques, when applied to classes of multicommodity network flow problems. All the runs were carried out on a Fujitsu Primergy RX300 server with two 3.33 GHz Intel Xeon X5680 CPUs (each CPU with 12 cores) and 144 GB of RAM, under a GNU/Linux operating system (Suse 11.4), without exploitation of multithreading capabilities.

4.1. Geometrical relations and PCG iterations. Our goal here is to generate synthetic problems such that we can control the level of collinearity between their linking and block constraints. To this end, consider two full rank matrices $N \in \mathbb{R}^{m\times n}$, $n > m$, and $Y \in \mathbb{R}^{l\times m}$, $n > l$, and let $L = YN$. The rows of $L \in \mathbb{R}^{l\times n}$ are linear combinations of the rows of N and the principal angles between the subspaces generated by $L^\top$ and $N^\top$ are zero. Each vector in $\mathcal{R}(L^\top)$ can be rotated an angle α around the i-th and j-th coordinate axes by pre-multiplying $L^\top$ by the $n \times n$ rotation matrix:

$$R_{ij}(\alpha) = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & 0 & \cdots & 0 & \vdots & \ddots & \vdots \\ 0 & \cdots & \cos(\alpha) & 0 & \cdots & 0 & -\sin(\alpha) & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ldots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & \sin(\alpha) & 0 & \cdots & 0 & \cos(\alpha) & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ldots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{4.1}$$

If we consider $\mathcal{H} = \{(i,j) : 1 \le i \le n-1, i < j \le n\}$, the set of all possible pairs of coordinate axes, the $n(n-1)/2$ distinct $R_{ij}(\alpha)$ rotation matrices may be concatenated in some order to produce a new rotation matrix such as $\prod_{(i,j)\in\mathcal{S}} R_{ij}(\alpha)$, where $\mathcal{S} \subseteq \mathcal{H}$. (Rotations in three dimensions and higher do not commute, so that different orderings give different rotations.)

We are interested in showing the performance of the PCG method at each interior-point iteration when changing the geometrical relations between the diagonal block and the linking constraint matrices, in accordance with specified rotations. To do so we randomly draw $N_i \in \mathbb{R}^{m\times n}$ and

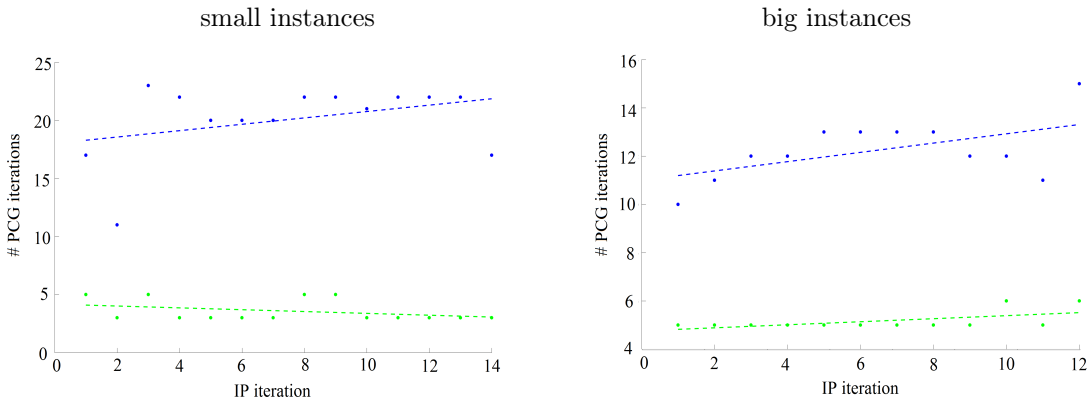

FIG. 3. *PCG iterations along the IPM iterations for $\alpha = \pi/16$, $\bar{r} = 2$, with preconditioners Θ_0^{-1} and D^{-1}. The blue points (and associated dashed regression line) correspond to the PCG iterations based on D^{-1} preconditioner, while the green points (and associated dashed regression line) correspond to the PCG iterations based on Θ_0^{-1} preconditioner.*

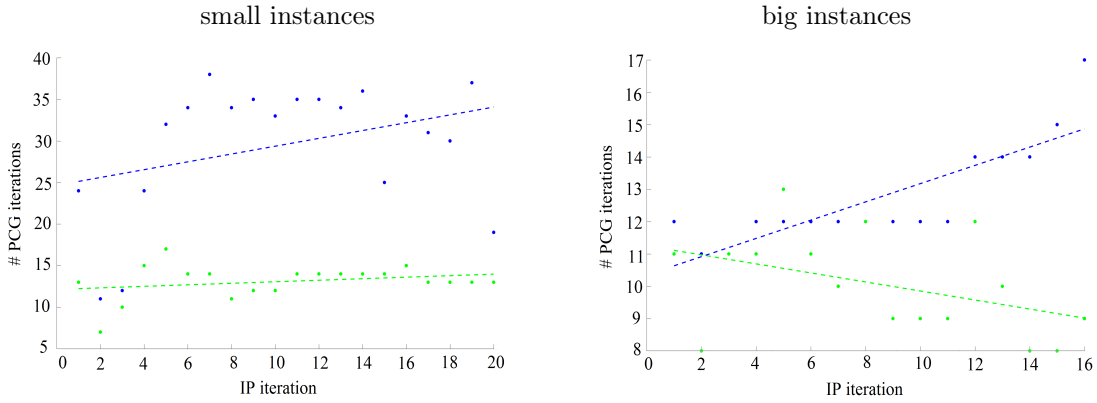

FIG. 4. *PCG iterations along the IPM iterations for $\alpha = \pi/12$, $\bar{r} = 2$, with preconditioners Θ_0^{-1} and D^{-1}. The blue points (and associated dashed regression line) correspond to the PCG iterations based on D^{-1} preconditioner, while the green points (and associated dashed regression line) correspond to the PCG iterations based on Θ_0^{-1} preconditioner.*

$Y_i \in \mathbb{R}^{l \times m}$, $i = 1, \ldots, k$, from a uniform probability distribution. Then $\mathcal{S} \subseteq \mathcal{H}$ is also randomly selected to compute the rotation matrix $\prod_{(i,j)\in\mathcal{S}} R_{ij}(\alpha)$. Finally, we obtain the linking constraint matrices $L_i = Y_i N_i \left(\prod_{(i,j)\in\mathcal{S}} R_{ij}(\alpha)\right)$, for $i = 1, \ldots, k$. This procedure relies on the number $\bar{r} = |\mathcal{S}|$ of concatenated rotation matrices and the angle of rotation α to evaluate the associated changes in the PCG iterations using Θ_0^{-1} and D^{-1} as preconditioners. It is worth to remark that α are not the principal angles between linking constraints matrices L_i and block constraints matrices N_i, but it is expected that small α values (that is, small rotations) will result in small principal angles.

The BlockIP package [12] implementing the specialized interior-point method has been extended with the Θ_0^{-1} preconditioner introduced in this work. The main features of BlockIP are: (1) it implements a long-step infeasible IPM, including both Newton and predictor-corrector directions [29]; (2) it uses the approach described in Section 2—which relies on Cholesky factorizations and PCG—for the solution of the normal equations [10, 11, 13]; (3) Cholesky systems are exactly solved; (4) PCG systems are stopped when the error (computed as $1 - \cos(Qx, d)$, for some system $Qx = d$) is below some tolerance; (5) in the computational results of this paper the initial PCG tolerance was set to $\epsilon^0 = 10^{-4}$, and it was updated at each interior-point iteration t as $\epsilon^{t+1} = \min\{0.95 \cdot \epsilon^t, 10^{-8}\}$. More details about the code—which is available for research purposes from `http://www-eio.upc.es/~jcastro/BlockIP.html`—can be found in [12].

Tables 1 and 2 show the computational results obtained with BlockIP using D^{-1} and Θ_0^{-1} preconditioners. Instances of Table 1 are "small", with $l = 100$ linking constraints, and $k = 100$ equal diagonal block matrices $N \in \mathbb{R}^{10\times 50}$. For the "big" instances of Table 2 these dimensions are $l = 200$, $k = 1000$, and $N \in \mathbb{R}^{20\times 200}$. The first columns of these tables show the angle α of each rotation,

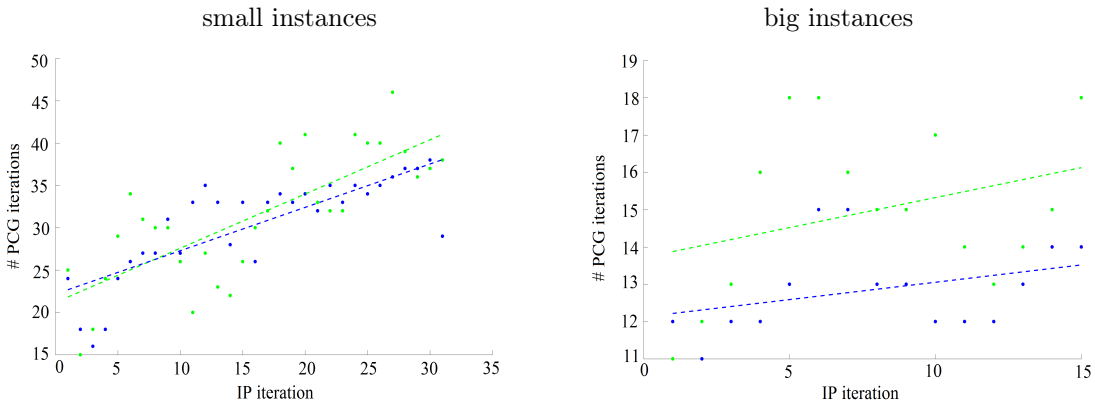

FIG. 5. *PCG iterations along the IPM iterations for $\alpha = \pi/8$, $\overline{r} = 2$, with preconditioners Θ_0^{-1} and D^{-1}. The blue points (and associated dashed regression line) correspond to the PCG iterations based on D^{-1} preconditioner, while the green points (and associated dashed regression line) correspond to the PCG iterations based on Θ_0^{-1} preconditioner.*

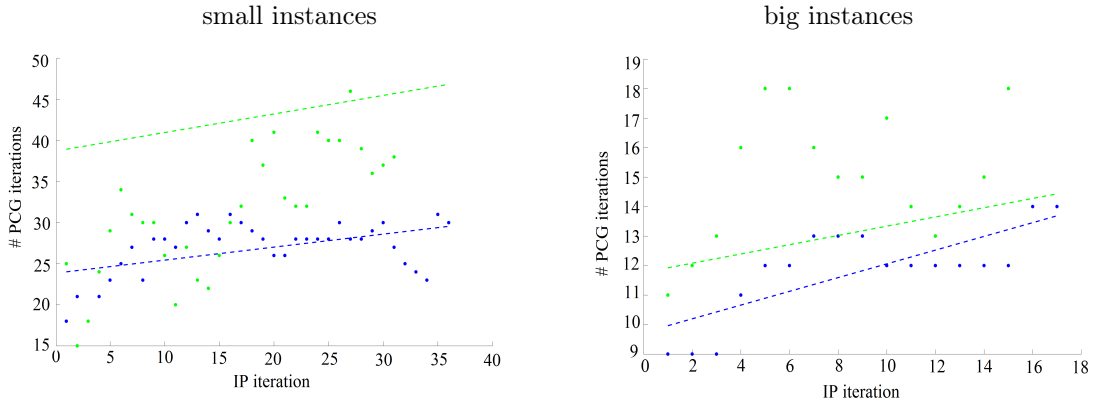

FIG. 6. *PCG iterations along the IPM iterations for $\alpha = \pi/4$, $\overline{r} = 2$, with preconditioners Θ_0^{-1} and D^{-1}. The blue points (and associated dashed regression line) correspond to the PCG iterations based on D^{-1} preconditioner, while the green points (and associated dashed regression line) correspond to the PCG iterations based on Θ_0^{-1} preconditioner.*

whereas the second columns report the number $\overline{r}$ of rotation matrices considered. The remaining columns give the CPU time, number of IPM iterations, and average number of PCG iterations per IPM iteration, for both preconditioners.

It is observed that the performance of the specialized IPM strongly relies on α for both the small and big instances. In particular, for Θ_0^{-1}, the smaller α, the better the preconditioner; and the average number of PCG iterations increases with α. On the other hand, D^{-1} is the best preconditioner when $\alpha = \pi/2$. This numerical evidence supports the previous discussion on the effect of the geometrical relations between the diagonal blocks and the linking constraint matrices on the quality of the preconditioner. In addition, the same conclusion will be provided by the numerical results of next Subsection 4.2 using a nontrivial—i.e., non-synthetic—problem.

The plots of Figures 3, 4, 5 and 6 illustrate the evolution of the number of PCG iterations (vertical axis) at each IPM iteration (horizontal axis) for both preconditioners. Small and large instances of the same dimensions as in Tables 1 and 2 have been used, but with different angles and always with $\overline{r} = 2$ rotations matrices. The blue and green dots are related to D^{-1} and Θ_0^{-1} respectively. The corresponding regression lines have been fitted to the empirical number of PCG iterations. It is observed that for small angles Figures 3 and 4), as expected by the theory, the preconditioner D^{-1} usually requires more PCG iterations than Θ_0^{-1}; this difference increases as we approach the optimal solution, for the small instances. However, for the "larger" angles of Figures 5 and 6, again in accordance with theory, preconditioner D^{-1} becomes more efficient than Θ_0^{-1} after certain IPM iteration. The message should then be that, for problems with small angles, Θ_0^{-1} should be preferred in general to D^{-1}. It is worth noting that a side benefit of Θ_0^{-1} is that it is always a diagonal preconditioner, independently of the problem, unlike D^{-1}, which may require a Cholesky

TABLE 3
CPU time and iterations (within parenthesis) of BlockIP (using D^{-1} and Θ_0^{-1} preconditioners) and CPLEX available LP methods (Primal Simplex, Dual Simplex, Barrier Method) when solving MNFPNCs. Only 20% of nodes are constrained to have an outflow capacity, i.e., $l = 0.2n$.

n	k	n. var.	n. con.	CPLEX			BlockIP	
				Primal Simplex	Dual Simplex	Barrier	Θ_0^{-1}	D^{-1}
150	200	880030	29830	57.5 (997332)	5.5 (85732)	5.8 (13)	5.7 (29)	5.6 (29)
150	400	1803230	59630	242.2 (428032)	14.0 (163391)	12.9 (14)	12.2 (30)	12.1 (30)
150	600	2601030	89430	362.2 (444258)	25.9 (249271)	19.7 (15)	19.6 (31)	18.8 (31)
150	800	3850430	119230	590.5 (575356)	38.0 (329632)	30.5 (16)	24.1 (29)	26.8 (32)
300	200	3587460	59860	459.3 (4676110)	27.5 (203873)	29.8 (15)	30.3 (30)	32.2 (32)
300	400	7022460	119660	> 3600 (8306650)	69.5 (415156)	57.2 (15)	57.4 (30)	59.1 (30)
300	600	11245260	179460	> 3600 (8453150)	112.7 (571964)	102.7 (17)	99.0 (31)	94.9 (30)
300	800	14862460	239260	> 3600 (9071240)	178.5 (784104)	140.3 (18)	123.1 (30)	133.8 (31)

TABLE 4
CPU time and iterations (within parenthesis) of BlockIP (using D^{-1} and Θ_0^{-1} preconditioners) and CPLEX available LP methods (Primal Simplex, Dual Simplex, Barrier Method) when solving MNFPNCs. All the nodes have an outflow capacity, i.e., $l = n$.

n	k	n. var.	n. con.	CPLEX			BlockIP	
				Primal Simplex	Dual Simplex	Barrier	Θ_0^{-1}	D^{-1}
150	200	880030	29950	75.1 (967177)	6.7 (86446)	11.8 (17)	5.9 (29)	5.8 (29)
150	400	1803230	59750	257.0 (607837)	18.6 (163574)	23.8 (17)	14.0 (32)	12.9 (31)
150	600	2601030	89550	426.9 (729617)	32.3 (249227)	33.6 (17)	21.9 (34)	20.0 (33)
150	800	3850430	119350	585.4 (476154)	52.8 (330616)	52.0 (19)	29.9 (33)	27.9 (32)
300	200	3587460	60100	682.9 (4310028)	28.3 (203780)	85.4 (16)	54.9 (33)	30.4 (34)
300	400	7022460	119900	> 3600 (8226752)	72.2 (413460)	120.3 (13)	78.9 (37)	66.5 (33)
300	600	11245260	179700	> 3600 (8489927)	118.5 (567590)	205.6 (15)	140.5 (32)	140.1 (33)
300	800	14862460	239500	> 3600 (8408703)	181.6 (779265)	284.7 (16)	141.2 (32)	140.8 (40)

factorization.

4.2. Multicommodity network flow problem with nodal capacities and equal flows. Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a directed graph, where $\mathcal{V}$ is a set of n vertices or nodes and $\mathcal{A}$ is a set of n' arcs, and let $\mathcal{K}$ be a set of k commodities. The multicommodity network flow problem with nodal capacities (MNFPNC from now on) looks for the minimum cost routing of the flows for all the commodities from some source to some destination nodes, imposing capacities on the total outflow at some nodes $h \in \mathcal{C} \subseteq \mathcal{V}$. This problem differs from the standard multicommodity flow problem in that capacities are imposed at nodes, instead of at arcs. The node capacity constraints are

$$\sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{V}} x_{hj}^i \leq b_h^0, \qquad h \in \mathcal{C}, \tag{4.2}$$

where b_h^0 denotes the capacity of node $h \in \mathcal{C}$. MNFPNC matches the standard formulation (2.1) of primal block-angular problems, by considering $m_i = n$, $n_i = n'$, $N_i = N \in \mathbb{R}^{(n-1)\times n'}$ for all $i = 1, \ldots, k$, where N is the node-arc incidence matrix associated to $\mathcal{G}$; and $l = |\mathcal{C}|$, $L_i = L \in \mathbb{R}^{l \times n'}$ for all $i = 1, \ldots, k$, are derived from N by considering only positive coefficients to obtain (4.2).

Tables 3 and 4 report two computational experiments associated to MNFPNCs of different sizes. The eight instances of Table 3 correspond to problems where $l = 0.2n$ (i.e., only 20% of nodes are constrained to have an outflow capacity), whereas the eight instances of Table 4 correspond to problems where $l = n$ (all the nodes have capacities). For each instance, the tables provide the number of nodes n, number of commodities k, the total number of variables ("n.var"), total number of constraints ("n.con."), and the CPU time and (within parentheses) total number of iterations for all the solvers considered: primal simplex, dual simplex and barrier for CPLEX; and BlockIP with both Θ_0^{-1} and D^{-1} preconditioners. In these runs BlockIP computed Newton directions, which is its default option when PCG is used. A time limit of 3600 seconds was considered.

It can be seen from Tables 3 and 4 that the CPU time associated to the primal simplex is always far greater than the ones of the other solvers. The dual simplex is quite competitive and in some instances outperforms CPLEX barrier method and BlockIP. BlockIP with both preconditioners is in general more efficient than CPLEX barrier, although it requires more IPM iterations because it computes Newton directions instead of second or higher order ones. The network size n does not seem to have a substantial effect on the comparative efficiency of the five solvers. Instead the increase in the number of linking constraints (either $l = 0.2n$, in Table 3 or $l = n$, in Table 4) almost doubles the CPU time of the CPLEX barrier method, whereas only slightly affects the specialized IPM.

TABLE 5

CPU time and iterations (within parenthesis) of BlockIP (using D^{-1} and Θ_0^{-1} preconditioner) and CPLEX available LP methods (Primal Simplex, Dual Simplex, Barrier Method) when solving MEFPNCs. Only 20% of nodes are constrained to have an outflow capacity, i.e., $l = 0.2n$.

n	k	n. var.	n. con.	CPLEX			BlockIP	
				Primal Simplex	Dual Simplex	Barrier	Θ_0^{-1}	D^{-1}
150	200	880030	56830	59.3 (989973)	5.9 (90369)	7.7 (16)	6.5 (30)	6.8 (32)
150	400	1803230	113630	239.0 (516211)	19.5 (222725)	17.6 (18)	11.5 (32)	11.5 (33)
150	600	2601030	170430	514.1 (563073)	33.9 (280339)	42.2 (27)	25.4 (34)	24.0 (33)
150	800	3850430	227230	635.4 (650626)	52.0 (406268)	54.1 (26)	30.9 (32)	30.5 (32)
300	200	3587460	113860	1357.0 (958439)	37.9 (232976)	44.9 (20)	39.6 (37)	34.8 (34)
300	400	7022460	227660	> 3600.0 (1325388)	85.3 (456233)	118.2 (29)	70.7 (32)	70.1 (32)
300	600	11245260	341460	> 3600.0 (1213005)	142.5 (665822)	155.7 (25)	109.2 (33)	108.7 (34)
300	800	14862460	455260	> 3600.0 (1179502)	204.8 (860729)	239.8 (30)	152.8 (34)	150.8 (34)

TABLE 6

CPU time and iterations (within parenthesis) of BlockIP (using D^{-1} and Θ_0^{-1} preconditioner) and CPLEX available LP methods (Primal Simplex, Dual Simplex, Barrier Method) when solving MEFPNCs. All the nodes have an outflow capacity, i.e., $l = n$.

n	k	n. var.	n. con.	CPLEX			BlockIP	
				Primal Simplex	Dual Simplex	Barrier	Θ_0^{-1}	D^{-1}
150	200	880030	56950	79.0 (2320414)	7.5 (86446)	11.3 (16)	5.9 (32)	5.0 (32)
150	400	1803230	113750	296.9 (964870)	26.7 (163574)	26.7 (18)	15.1 (33)	14.6 (33)
150	600	2601030	170550	619.9 (1839715)	43.0 (230553)	96.7 (27)	29.6 (34)	26.5 (33)
150	800	3850430	227350	891.7 (1061339)	74.7 (249227)	68.0 (22)	35.8 (33)	34.5 (32)
300	200	3587460	114100	820.8 (1184579)	36.9 (330616)	87.6 (18)	51.1 (36)	43.7 (34)
300	400	7022460	227900	> 3600.0 (4906436)	88.6 (203780)	232.8 (22)	98.8 (34)	80.2 (34)
300	600	11245260	341700	> 3600.0 (4701108)	177.3 (413460)	288.5 (21)	125.5 (35)	119.0 (36)
300	800	14862460	455500	> 3600.0 (4904433)	364.9 (1278588)	480.1 (25)	240.0 (35)	206.6 (34)

Let us consider a slight modification of the MNFPNCs, obtained by introducing to each commodity a set of equal flow constraints, i.e. constraints requiring that each arc in a specified set $\mathcal{R}_r$ must carry the same amount of flow, for every group of arcs $r \in R$; that is, $x^i_{a_j} = x^i_{a_h}$, for $i = 1 \dots k$, $a_j, a_h \in \mathcal{R}_r$, $r = 1 \dots R$ (where a_j, a_h denote two particular arcs in the network). These constraints arose while modeling some real-life problems, such as water resource system management [22]. We call this problem *multicommodity equal flow problem with nodal capacities* (MEFPNC from now on). Here we are considering MEFPNCs of different sizes with $R = \text{EF} \cdot n$ groups of arcs having the same flows per each commodity and $|\mathcal{R}_r| = \text{EF} \cdot 100$ number of arcs in each group $r = 1 \dots R$, where EF is a given parameter to control the number of equal flow constraints.

Tables 5 and 6 report two computational experiments associated to these MEFPNCs. As before, the eight instances of Table 5 correspond to problems with $l = 0.2n$, whereas $l = n$ in Table 6. The parameter EF was set to 0.1 in these instances. The meaning of the columns is the same as in Tables 3 and 4. It is worth mentioning that, for every set $\mathcal{R}_r = \{a_1, a_2, \dots, a_{|\mathcal{R}_r|}\}$, and $i \in \mathcal{K}$, the equal flow constraints are formulated as $x^i_{a_j} = x^i_{a_{j+1}}, j = 1, \dots, |\mathcal{R}_r| - 1$, instead of $x^i_{a_1} = x^i_{a_j}, j = 2, \dots, |\mathcal{R}_r|$. With this formulation we avoid a "dense column" for variables $x^i_{a_1}$, which makes BlockIP more efficient. CPLEX barrier is not affected by this different reformulation due to its presolving capabilities. The principal angles between linking and block constraints are the same for both formulations, since they span the same subspace (it is easy to show that given the constraints matrices for the two formulations, one can be obtained from the other by simple linear manipulations).

The inclusion of equal flow constraints negatively affects the computational performance of all the considered solvers, though in different proportions. The dual simplex and barrier almost double its CPU times in the largest instances with respect to the MNFPNCs, whereas the ones associated to the specialized IPM slightly increase. In fact, the specialized IPM becomes the most efficient algorithm for this class of problems from $k = 400$. Also for the MEFPNCs, as it was for the MNFPNCs, the network size n does not seem to have a substantial effect on the ranking of the five solvers.

As for the two preconditioners of BlockIP, results with Θ_0^{-1} and D^{-1} are very similar in solving MEFPNCs with $l = 0.2n$, as shown in Table 5. The results with D^{-1} indicate it to be a slightly better preconditioner when $l = n$. In both cases BlockIP outperforms the CPLEX available LP methods. It is worth noting that BlockIP uses an out-of-date sparse linear algebra package for the Cholesky factorization [24], while CPLEX implements highly tuned state-of-the-art factorization routines. Therefore, the performance of BlockIP could be significantly improved by the use of a more

TABLE 7
Average principal angles between the subspaces generated by the columns of $L^\top$ and $N^\top$, for each instance of multi-commodity network flow problems with nodal capacities in Tables 3, 4, 5 and 6.

n	average principal angles			
	Table 3	Table 4	Table 5	Table 6
150	0.8774	0.8319	0.7958	0.8225
300	0.8544	0.8297	0.7732	0.8247

TABLE 8
Dimensions and properties of MEFPNCs with small and dense networks.

#	n	k	l	density	NC	EF	n. var.	n. con.
1	400	2000	40	0.2	0.1	0.00	32894040	798040
2	400	2000	40	0.2	0.1	0.05	32894040	998040
3	400	2000	40	0.2	0.1	0.10	32894040	1598040
4	400	2000	40	0.2	0.1	0.15	32894040	2598040
5	400	2000	40	0.2	0.1	0.20	32894040	3998040
6	400	2000	80	0.2	0.2	0.00	32894080	798080
7	400	2000	80	0.2	0.2	0.05	32894080	998080
8	400	2000	80	0.2	0.2	0.10	32894080	1598080
9	400	2000	80	0.2	0.2	0.15	32894080	2598080
10	400	2000	80	0.2	0.2	0.20	32894080	3998080
11	400	2000	120	0.2	0.3	0.00	32894120	798120
12	400	2000	120	0.2	0.3	0.05	32894120	998120
13	400	2000	120	0.2	0.3	0.10	32894120	1598120
14	400	2000	120	0.2	0.3	0.15	32894120	2598120
15	400	2000	120	0.2	0.3	0.20	32894120	3998120
16	400	2000	40	0.4	0.1	0.00	64142040	798040
17	400	2000	40	0.4	0.1	0.05	64142040	998040
18	400	2000	40	0.4	0.1	0.10	64142040	1598040
19	400	2000	40	0.4	0.1	0.15	64142040	2598040
20	400	2000	40	0.4	0.1	0.20	64142040	3998040
21	400	2000	80	0.4	0.2	0.00	64142080	798080
22	400	2000	80	0.4	0.2	0.05	64142080	998080
23	400	2000	80	0.4	0.2	0.10	64142080	1598080
24	400	2000	80	0.4	0.2	0.15	64142080	2598080
25	400	2000	80	0.4	0.2	0.20	64142080	3998080
26	400	2000	120	0.4	0.3	0.00	64142120	798120
27	400	2000	120	0.4	0.3	0.05	64142120	998120
28	400	2000	120	0.4	0.3	0.10	64142120	1598120
29	400	2000	120	0.4	0.3	0.15	64142120	2598120
30	400	2000	120	0.4	0.3	0.20	64142120	3998120

recent Cholesky solver.

The information of the average principal angles between the subspaces generated by the columns of $L^\top$ and $N^\top$ of the instances of Tables 3–6 is reported in Table 7, differentiating for the 150 and 300 nodes instances. These true principal angles were computed with the exact algorithm of Figure 7 in the Appendix, so they are not estimations given by the two methods described in Subsection 3.2. The first observation we deduce from data collected in Table 7 is that the average principal angles are generally stable with respect to the number of nodes. We also see that average principal angles are far from 0, which may explain why preconditioner D^{-1} outperforms Θ_0^{-1}, in general, in Tables 3–6.

We finally generated and solved a set of large instances, considering only the CPLEX barrier solver, and BlockIP with the two preconditioners Θ_0^{-1} and D^{-1}, and a new variant (named "automatic" in the tables of results) that selected the best preconditioner according to (3.29), possibly switching to the other preconditioner using the criteria in (3.30). No time limit was set in these runs. The corresponding results are reported in Tables 8–9 and Tables 10–12, for small dense and big sparse networks, respectively. Tables 8 and 10 show the dimensions and characteristics of the instances: columns l provide the number of linking constraints; columns "density" provide the fraction "number of arcs/maximum number of arcs $(n(n-1)/2)$"; columns "NC" show the fraction of nodal capacity constraints, i.e., "number of node capacities/n"; and columns "EF" provide the parameter considered for the equal flow constraints. The rest of columns have the same meaning as in previous tables.

The instances of Table 8 with small dense networks were solved with CPLEX, and BlockIP using the Newton direction; Table 9 shows these results. For the instances of Table 10 with large sparse networks, we additionally used BlockIP with the predictor-corrector direction (as discussed below, this was instrumental for the efficiency of BlockIP). These results are reported in Tables 11–

TABLE 9
Results with BlockIP (using Newton direction for variants D^{-1}, Θ_0^{-1} and automatic selection of preconditioners) and CPLEX barrier for MEFPNCs with small and dense networks.

	CPLEX		BlockIP Newton direction										
	Barrier		Θ_0^{-1}			D^{-1}			Automatic				
#	it.	CPU	it.	PCG	CPU	it.	PCG	CPU	it.	PCG	CPU	Prec.	γ
1	17	428.32	32	57	424.55	33	51	432.94	33	51	441.49	D^{-1}	0.773
2	19	461.37	35	75	487.77	34	55	508.19	34	55	455.40	D^{-1}	0.771
3	31	761.24	35	88	544.80	35	61	538.53	35	61	559.44	D^{-1}	0.767
4	30	724.38	37	155	706.19	37	99	636.19	36	103	596.13	D^{-1} (4)	0.756
5	27	708.23	37	127	697.22	40	90	698.52	37	127	721.17	Θ_0^{-1}	0.740
6	18	402.91	33	60	441.18	33	50	435.00	33	50	432.44	D^{-1}	0.803
7	19	465.19	35	82	487.26	34	55	456.62	34	55	460.36	D^{-1}	0.802
8	29	705.41	35	88	542.94	36	61	556.18	36	61	540.66	D^{-1}	0.797
9	26	704.96	38	169	674.00	39	106	641.39	37	121	639.92	D^{-1}(4)	0.788
10	41	1071.90	38	187	754.78	40	119	727.62	38	137	717.11	D^{-1}(4)	0.774
11	17	422.87	33	62	445.08	33	49	440.76	33	49	452.55	D^{-1}	0.838
12	20	535.59	35	80	491.03	34	55	459.19	34	55	488.47	D^{-1}	0.837
13	37	961.37	36	90	575.95	37	59	574.79	37	59	591.68	D^{-1}	0.833
14	28	831.13	38	180	687.14	39	106	651.40	39	106	671.72	D^{-1}	0.825
15	32	1319.40	39	199	789.96	40	115	731.61	40	115	728.53	D^{-1}	0.814
16	28	901.33	33	64	678.91	33	53	661.25	33	53	663.92	D^{-1}	0.768
17	38	1315.00	34	81	712.32	34	61	688.42	34	61	689.50	D^{-1}	0.767
18	36	1385.90	35	85	752.29	35	66	731.96	35	66	743.15	D^{-1}	0.765
19	45	1697.40	36	90	818.97	36	69	826.87	36	69	807.50	D^{-1}	0.760
20	43	1843.00	37	100	939.06	37	86	920.71	37	86	886.39	D^{-1}	0.753
21	26	949.55	32	64	660.20	33	54	667.20	33	54	667.95	D^{-1}	0.794
22	30	1210.40	34	84	715.17	34	62	734.03	34	62	692.10	D^{-1}	0.793
23	50	2124.00	35	91	775.08	35	66	752.99	35	66	741.86	D^{-1}	0.791
24	36	1461.50	36	96	838.86	36	70	840.70	36	70	798.98	D^{-1}	0.787
25	35	1606.60	38	112	993.90	38	83	912.51	38	83	907.41	D^{-1}	0.782
26	38	1349.50	33	78	709.65	32	54	647.39	32	54	652.25	D^{-1}	0.835
27	28	1328.00	34	86	719.12	34	62	694.62	34	62	694.77	D^{-1}	0.834
28	41	1656.70	36	105	804.65	36	69	770.40	36	69	768.26	D^{-1}	0.832
29	41	1757.90	37	109	872.03	36	70	803.45	36	70	803.71	D^{-1}	0.828
30	30	1564.20	38	123	964.46	38	81	914.56	38	81	916.24	D^{-1}	0.823

TABLE 10
Dimensions and properties of MEFPNCs with large and sparse networks.

#	n	k	l	density	NC	EF	n. var.	n. con.
31	800	2000	80	0.02	0.1	0.00	13898080	1598080
32	800	2000	80	0.02	0.1	0.05	13898080	1998080
33	800	2000	80	0.02	0.1	0.10	13898080	3198080
34	800	2000	80	0.02	0.1	0.15	13898080	5198080
35	800	2000	80	0.02	0.1	0.20	13898080	7998080
36	800	2000	160	0.02	0.2	0.00	13898160	1598160
37	800	2000	160	0.02	0.2	0.05	13898160	1998160
38	800	2000	160	0.02	0.2	0.10	13898160	3198160
39	800	2000	160	0.02	0.2	0.15	13898160	5198160
40	800	2000	160	0.02	0.2	0.20	13898160	7998160
41	800	2000	240	0.02	0.3	0.00	13898240	1598240
42	800	2000	240	0.02	0.3	0.05	13898240	1998240
43	800	2000	240	0.02	0.3	0.10	13898240	3198240
44	800	2000	240	0.02	0.3	0.15	13898240	5198240
45	800	2000	240	0.02	0.3	0.20	13898240	7998240
46	800	2000	80	0.04	0.1	0.00	26484080	1598080
47	800	2000	80	0.04	0.1	0.05	26484080	1998080
48	800	2000	80	0.04	0.1	0.10	26484080	3198080
49	800	2000	80	0.04	0.1	0.15	26484080	5198080
50	800	2000	80	0.04	0.1	0.20	26484080	7998080
51	800	2000	160	0.04	0.2	0.00	26484160	1598160
52	800	2000	160	0.04	0.2	0.05	26484160	1998160
53	800	2000	160	0.04	0.2	0.10	26484160	3198160
54	800	2000	160	0.04	0.2	0.15	26484160	5198160
55	800	2000	160	0.04	0.2	0.20	26484160	7998160
56	800	2000	240	0.04	0.3	0.00	26484240	1598240
57	800	2000	240	0.04	0.3	0.05	26484240	1998240
58	800	2000	240	0.04	0.3	0.10	26484240	3198240
59	800	2000	240	0.04	0.3	0.15	26484240	5198240
60	800	2000	240	0.04	0.3	0.20	26484240	7998240

TABLE 11
Results with BlockIP (using predictor-corrector direction for variants D^{-1}, Θ_0^{-1} and automatic selection of preconditioners) and CPLEX barrier for MEFPNCs with large and sparse networks.

	CPLEX		BlockIP predictor-corrector direction										
	Barrier		Θ_0^{-1}			D^{-1}			Automatic				
#	it.	CPU	it.	PCG	CPU	it.	PCG	CPU	it.	PCG	CPU	Prec.	γ
31	12	495.06	22	72	677.86	24	79	764.06	22	72	691.65	Θ_0^{-1}	0.703
32	21	796.48	24	134	791.57	22	90	741.25	24	134	814.78	Θ_0^{-1}	0.694
33	13	613.00	23	126	943.45	26	197	1152.24	23	126	948.45	Θ_0^{-1}	0.649
34	32	1535.60	28	149	1543.33	31	307	2010.34	28	149	1547.49	Θ_0^{-1}	0.589
35	85	4442.30	21	98	1472.82	192	3755	18933.60	21	98	1491.84	Θ_0^{-1}	0.491
36	12	559.93	22	85	696.55	23	77	746.63	23	77	707.57	D^{-1}	0.748
37	22	1041.20	23	133	774.28	26	103	829.87	23	133	760.36	Θ_0^{-1}	0.741
38	22	1243.30	24	166	1064.07	27	220	1232.14	24	166	1037.54	Θ_0^{-1}	0.707
39	25	1611.80	23	126	1310.97	39	360	2385.75	23	126	1282.90	Θ_0^{-1}	0.647
40	45	2901.50	21	93	1504.07	31	406	2611.41	21	93	1453.75	Θ_0^{-1}	0.548
41	11	674.28	22	77	679.94	23	77	707.79	23	77	704.19	D^{-1}	0.804
42	23	1359.10	24	140	836.78	26	103	822.08	26	103	830.87	D^{-1}	0.798
43	14	1050.30	24	174	1038.12	30	232	1312.26	30	232	1322.88	D^{-1}	0.768
44	22	1597.40	22	141	1253.54	58	860	4049.23	22	141	1270.42	Θ_0^{-1}	0.708
45	85	5944.70	21	100	1463.74	24	410	2167.35	21	100	1483.09	Θ_0^{-1}	0.606
46	13	789.73	22	79	1031.86	24	66	1100.24	22	79	1046.23	Θ_0^{-1}	0.738
47	29	1708.00	23	90	1129.02	26	75	1257.89	23	90	1131.68	Θ_0^{-1}	0.733
48	32	1985.10	24	124	1348.27	28	122	1577.65	24	124	1352.66	Θ_0^{-1}	0.714
49	15	1108.80	23	154	1617.12	25	160	1804.16	23	154	1632.36	Θ_0^{-1}	0.689
50	15	1230.90	21	124	1773.00	29	250	2626.98	21	124	1743.29	Θ_0^{-1}	0.652
51	12	875.53	23	77	1087.61	25	68	1141.81	25	68	1177.41	D^{-1}	0.767
52	26	2172.00	24	104	1260.91	26	74	1244.23	26	74	1270.97	D^{-1}	0.763
53	29	2191.00	24	131	1385.90	25	99	1361.02	25	99	1365.77	D^{-1}	0.749
54	24	2146.90	24	204	1807.09	25	199	1793.25	24	204	1751.85	Θ_0^{-1}	0.723
55	18	1733.10	23	151	2020.30	26	249	2309.60	23	151	2075.58	Θ_0^{-1}	0.687
56	13	1134.40	22	82	1033.35	25	70	1146.59	25	70	1204.39	D^{-1}	0.805
57	32	2622.70	24	119	1212.28	26	84	1257.95	26	84	1263.93	D^{-1}	0.802
58	29	2505.10	25	136	1413.30	25	93	1422.24	25	93	1372.66	D^{-1}	0.789
59	27	2610.70	25	198	1891.78	25	215	1911.14	25	215	1821.69	D^{-1}	0.767
60	13	1547.80	25	197	2202.88	28	295	2693.02	25	197	2164.69	Θ_0^{-1}	0.732

12. For each run, these tables provide the number of IPM iterations (columns "it."), CPU times in seconds (columns "CPU"), overall number of PCG iterations (columns "PCG"); and , for the BlockIP "automatic" variant, the initial estimated principal angle using (3.28) (columns "γ"), and the initially selected preconditioner according to (3.29) (columns "Prec."). Column "Prec." also reports—in brackets, and in case of a switching between preconditioners—the particular IPM iteration number where the switch was performed.

Among the 30 instances of Table 9, BlockIP with some variant outperformed CPLEX barrier in 27 instances, and CPLEX barrier was only superior in the three smallest ones: #1, #6 and #11. The "automatic" variant only selected preconditioner Θ_0^{-1} for instance #5; D^{-1} was used for the remaining ones since their estimated angles were above the threshold value $\tau_\gamma = 0.95\pi/4 = 0.74613$ for the rule (3.29). The switching between preconditioners only happened in three instances, #4, #9 and #1, reducing the CPU time compared to using either Θ_0^{-1} or D^{-1} for all the IPM iterations. In the remaining instances the initially selected preconditioner was applied for all the IPM iterations; and, remarkably, the best preconditioner (the one involving less CPU time and/or less number of PCG iterations) was selected in most instances. In these cases the CPU time of the "automatic" variant incurred in some overhead due to the computation of principal angles. (In some instances the CPU time with the "automatic" variant is sligthly less than that obtained with either Θ_0^{-1} or D^{-1}; this (in theory impossible) fact was due to inacuracies of the operating system timing routines. This fact was also observed in results of other tables.)

As for the 30 instances of Table 10, the results obtained with BlockIP computing the Newton direction, reported in Table 12, were not fully satisfactory. It can be observed that CPLEX barrier (whose results are reported in Table 11) usually outperformed BlockIP using the Newton direction with either variant. Nevertheless, even in these unfavourable cases, the "automatic" variant usually selected the best option (by a significant margin in some instances, such as in #35, #40, #45, #50, #55 and #60). Results drastically changed when BlockIP computed predictor-director directions: BlockIP was competititive and outperformed CPLEX in some of the most difficult instances. These

TABLE 12
Results with BlockIP (using Newton direction for variants D^{-1}, Θ_0^{-1} and automatic selection of preconditioners) and CPLEX barrier for MEFPNCs with large and sparse networks.

	BlockIP Newton direction										
	Θ_0^{-1}			D^{-1}			Automatic				
#	it.	PCG	CPU	it.	PCG	CPU	it.	PCG	CPU	Prec.	γ
31	36	66	946.43	37	64	989.44	36	66	948.33	Θ_0^{-1}	0.703
32	37	121	1021.63	37	83	988.42	37	121	1022.78	Θ_0^{-1}	0.694
33	45	138	1558.14	86	171	2880.19	45	138	1627.43	Θ_0^{-1}	0.649
34	40	115	1883.31	53	169	2516.36	40	115	2008.19	Θ_0^{-1}	0.589
35	36	81	2153.31	86	191	5198.91	36	81	2237.70	Θ_0^{-1}	0.491
36	39	71	1043.15	39	58	1014.32	39	58	1051.49	D^{-1}	0.748
37	37	129	1030.03	38	84	1016.23	37	129	1057.02	Θ_0^{-1}	0.741
38	39	139	1388.36	91	181	3075.57	39	139	1390.82	Θ_0^{-1}	0.707
39	43	110	2037.50	160	327	7524.45	43	110	2048.50	Θ_0^{-1}	0.647
40	36	80	2157.00	99	207	6239.70	36	80	2252.81	Θ_0^{-1}	0.548
41	37	63	1015.59	42	61	1166.39	42	61	1095.37	D^{-1}	0.804
42	40	142	1128.31	38	84	1022.93	38	84	1032.06	D^{-1}	0.798
43	38	142	1376.46	52	157	1843.27	52	157	1830.28	D^{-1}	0.768
44	42	120	2141.33	97	256	4618.71	42	120	2011.55	Θ_0^{-1}	0.708
45	37	79	2297.12	93	209	5749.08	37	79	2237.28	Θ_0^{-1}	0.606
46	34	62	1434.37	33	52	1330.78	34	62	1385.14	Θ_0^{-1}	0.738
47	35	97	1546.51	36	64	1554.71	35	97	1535.42	Θ_0^{-1}	0.733
48	37	140	1843.27	40	97	1999.25	37	140	1843.39	Θ_0^{-1}	0.714
49	36	128	2105.54	38	128	2364.30	36	128	2181.78	Θ_0^{-1}	0.689
50	36	101	2527.68	55	160	4030.74	36	101	2517.88	Θ_0^{-1}	0.652
51	33	65	1404.61	33	51	1329.42	33	51	1335.45	D^{-1}	0.767
52	36	99	1545.08	37	64	1565.00	37	64	1568.63	D^{-1}	0.763
53	38	142	1888.47	38	95	1801.29	38	95	1874.87	D^{-1}	0.749
54	38	174	2284.05	50	172	2981.77	38	174	2278.19	Θ_0^{-1}	0.723
55	38	125	2709.45	62	194	4374.40	38	125	2665.91	Θ_0^{-1}	0.687
56	34	65	1378.88	35	52	1424.71	35	52	1399.58	D^{-1}	0.805
57	36	104	1544.71	37	66	1538.87	37	66	1544.03	D^{-1}	0.802
58	38	144	1856.94	40	90	1872.33	40	90	1936.99	D^{-1}	0.789
59	38	166	2265.80	49	161	2836.92	49	161	2942.40	D^{-1}	0.767
60	37	164	2691.99	69	217	4849.53	37	164	2753.61	Θ_0^{-1}	0.732

results are shown in Table 11. Although predictor-corrector directions for PCG-based IPMs are not as competitive as for Cholesky-based ones (since we have no factorization to reuse), in these instances the number of PCG iterations was small enough to make it worth solving twice the normal equations for a better direction. As in previous cases, the "automatic" variant selected in almost all cases the best preconditioners (some notable results are observed for instances #34, #35, #39, #40, #44, #45, #50). We also observe that the more (equal-flow block) constraints, the better Θ_0^{-1} is. This is consistent with theory: the more constraints in N_i, the "larger" is the subspace spanned by their columns, and then principal angles between the rows of L_i and N_i are smaller (i.e., closer to 0), which means that Θ_0^{-1} is closer to be the exact preconditioner. From Table 11, we see that BlockIP outperformed CPLEX in 19 of the 30 instances, and by a significant margin in some cases. These results could be improved if larger instances would have been tested. For instance, in a tough instance (not included in the above tables) of $n = 4000$, $k = 2000$, density$= 0.001$, NC$=0.2$, with no equal flow constraints, CPLEX barrier spent about 37000 seconds (requiring 85 Gigabytes of memory), while BlockIP found a solution in 11000 seconds (using 15 Gigabytes of memory).

It is worth remarking that the density of D may negatively affect the performance of preconditioner D^{-1}, possibly involving a costly Cholesky factorization, whereas Θ_0 is always diagonal. However, in all the MNFPNCs and MEFPNCs instances tested in this work, D is also diagonal. Therefore the different performance of preconditioners is mainly explained by the geometrical relations between the subspaces spanned by the linking and block constraints (and their principal angles). Yet, the computation of Θ_0 will always be less costly than that of D: even in the best case for D (that is, when $L_i = I$), we have $D = \Theta_0 + \sum_{i=1}^{k} \Theta_i$.

5. Conclusions. This work showed that the angles between the subspaces generated by the diagonal blocks and the linking constraints may explain the effectiveness of two complementary preconditioners, namely Θ_0^{-1} and D^{-1}, in the specialized IPM for block-angular problems. It was also shown that the evolution of principal angles along the IPM iterations rely on the diagonal Θ matrix. Two methods for the estimation of principal angles were developed and used to predict ex-ante which

Algorithm *Computation of principal angles*($L^\top \in \mathbb{R}^{n\times l}, N^\top \in \mathbb{R}^{n\times m}$)
$q = \min(m, l)$
//Compute orthonormal basis of $\mathcal{R}(L^\top)$ and $\mathcal{R}(N^\top)$ using QR factorizations
$Q_L \leftarrow QR(L^\top)$, $Q_L \in \mathbb{R}^{n\times l}$
$Q_N \leftarrow QR(N^\top)$, $Q_N \in \mathbb{R}^{n\times m}$
//Compute singular value decomposition of $Q_L^\top Q_N$: $U\Sigma V^\top = Q_L^\top Q_N$
$[U, \Sigma, V] \leftarrow SVD(Q_L^\top Q_N)$, $U \in \mathbb{R}^{l\times q}$, $V \in \mathbb{R}^{m\times q}$ orthonormal, $\Sigma \in \mathbb{R}^{q\times q}$
// Compute vector of principal angles γ
$\gamma \leftarrow \arccos(\mathrm{diag}(\Sigma))$, $\gamma \in \mathbb{R}^q$
// Compute principal vectors W_L and W_N
$W_L \leftarrow Q_L U$, $W_L \in \mathbb{R}^{n\times q}$
$W_N \leftarrow Q_N V$, $W_N \in \mathbb{R}^{n\times q}$
Return: γ, W_L, W_N
End_algorithm

FIG. 7. *Algorithm to compute principal angles between column spaces of* $L^\top$ *and* $N^\top$.

preconditioner to be used for each particular instance. Algebraical properties of the two complementary preconditioners were supported by numerical results. We also analyzed the performance of the specialized IPM with the two preconditioners in the solution of the multicommodity network flow problem with nodal capacities and equal flows, observing that it outperforms all of the available CPLEX methods in some of the largest instances.

Θ_0^{-1} and D^{-1} have shown to be the exact preconditioners when a linking constraint belongs to respectively the range and null space of the matrix of block constraints. Since any vector defining a linking constraint can be decomposed as the sum of two orthogonal vectors belonging to the range and null space of the matrix of block constraints, it might be worth to exploit this fact to see whether a *multipreconditioner* based on the two above can be obtained, following for instance [8]. This is part of the further research to be done.

Alternative more efficient and accurate methods for the estimation of the principal angles (e.g., considering a sample or representative subset of linking constraints), and applying the specialized IPM (either with Θ_0^{-1}, D^{-1} or combining both) to other problems are also part of the future research.

Acknowledgments. We would like to acknowledge the two anonymous reviewers for his valuable suggestions, that improved the results in the manuscript and fixed several issues.

Appendix. Computation of principal angles.

Principal angles between two subspaces can be computed by the singular value decomposition, as shown by next theorem from [6] (procedure also described in [19, Chapter 12]):

THEOREM A.1. *Let the columns of matrices* $Q_L \in \mathbb{R}^{n\times l}$ *and* $Q_N \in \mathbb{R}^{n\times m}$ *form orthonormal bases for the subspaces* $\mathcal{L}$ *and* $\mathcal{N}$*, correspondingly. Principal vectors* $u \in \mathbb{R}^n$ *and* $v \in \mathbb{R}^n$ *must verify* $u = Q_L\widetilde{u}$ *and* $v = Q_N\widetilde{v}$*, where* $\widetilde{u} \in \mathbb{R}^l$ *and* $\widetilde{v} \in \mathbb{R}^m$ *are left and right singular vectors of* $Q_L^\top Q_N$*, associated to the singular value* $\cos(\gamma(u, v))$*, that is to say,* $(Q_L^\top Q_N)\widetilde{v} = \cos(\gamma(u, v))\,\widetilde{u}$.

The algorithm of Figure 7 outlines how to compute the principal angles of column spaces of matrices $L^\top \in \mathbb{R}^{n\times l}$ and $N^\top \in \mathbb{R}^{n\times m}$. The algorithm is just provided for completeness: as shown in Subsection 3.2, it does not need to be used in practice in our approach. In fact, its use would be prohibitive since the required singular value decomposition is computationally very expensive for large optimization problems.

REFERENCES

[1] F. BABONNEAU, O. DU MERLE, AND J.-P. VIAL, *Solving large-scale linear multicommodity flow problems with an active set strategy and proximal-ACCPM*, Oper. Res., 54 (2006), pp. 184–197.
[2] F. BABONNEAU AND J.-P. VIAL, *ACCPM with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems*, Math. Prog., 120 (2009), pp. 179-210.
[3] S. BELLAVIA, J. GONDZIO, AND B. MORINI, B., *A matrix-free preconditioner for sparse symmetric positive definite systems and least-squares problems*, SIAM J. Sci. Comp., 35 (2013), pp. A192-A211.
[4] L. BERGAMASCHI, J. GONDZIO, J., AND G. ZILLI, *Preconditioning indefinite systems in interior point methods for optimization*, Comput. Optim. Appl., 28 (2004), pp. 149–171.
[5] D. BIENSTOCK, *Potential Function Methods for Approximately Solving Linear Programming Problems. Theory and Practice*, Kluwer, Boston, MA, 2002.
[6] Å. BJÖRCK AND G.H. GOLUB, *Numerical methods for computing angles between linear subspaces*, Math. Comp., 27 (1977), pp. 579–594.

[7] S. Bocanegra, J. Castro, and A.R.L. Oliveira, *Improving an interior-point approach for large block-angular problems by hybrid preconditioners*, Eur. J. Oper. Res., 231 (2013), pp. 263–273.
[8] R. Bridson and C. Greif, *A multipreconditioned conjugate gradient algorithm*, SIAM J Matrix Anal. Appl., 27 (2006), pp. 1056–1068.
[9] Y. Cao, C.D. Laird, and V.M. Zavala, *Clustering-based preconditioning for stochastic programs*, Comput. Optim. Appl., 64 (2016), pp. 379–406.
[10] J. Castro, *A specialized interior-point algorithm for multicommodity network flows*, SIAM J. Optim., 10 (2000), pp. 852–877.
[11] J. Castro, *An interior-point approach for primal block-angular problems*, Comput. Optim. Appl., 36 (2007), pp. 195–219.
[12] J. Castro, *Interior-point solver for convex separable block-angular problems*, Optim. Method. Softw., 31 (2016), pp. 88–109.
[13] J. Castro and J. Cuesta, *Quadratic regularizations in an interior-point method for primal block-angular problems*, Math. Prog., 130 (2011), pp. 415–445.
[14] J. Castro and N. Nabona, *An implementation of linear and nonlinear multicommodity network flows*, Eur. J. Oper. Res., 92 (1996), pp. 37–53.
[15] J. Castro and S. Nasini, *Mathematical programming approaches for classes of random network problems*, Eur. J. Oper. Res., 245 (2015), pp. 402–414.
[16] J. Castro, S. Nasini and F. Saldanha-da-Gama, *A cutting-plane approach for large-scale capacitated multi-period facility location using a specialized interior-point method*, Math. Prog., (2016), doi:10.1007/s10107-016-1067-6.
[17] A. Frangioni, and G. Gallo, *A bundle type dual-ascent approach to linear multicommodity min cost flow problems*, INFORMS J. Comput., 11 (1999), pp. 370–393.
[18] A. Frangioni and C. Gentile, *New preconditioners for KKT systems of network flow problems*, SIAM J. Optim., 14 (2004), pp. 894–913.
[19] G.H. Golub and C.F. Van Loan, *Matrix Computations, Third Ed.*, Johns Hopkins Univ. Press, Baltimore, MD, 1996.
[20] J. Gondzio, *Convergence analysis of an inexact feasible interior point method for convex quadratic programming*, SIAM J. Optim., 23 (2013), pp. 1510–1527.
[21] J. Gondzio and R. Sarkissian, *Parallel interior-point solver for structured linear programs*, Math. Prog., 96 (2003), pp. 561–584.
[22] A. Manca, G. Sechi, and P. Zuddas, *Water supply network optimisation using equal flow algorithms*, Water Resour. Manag., 24 (2010), pp. 3665–3678.
[23] R.D. McBride, *Progress made in solving the multicommodity flow problem*, SIAM J. Optim., 8 (1998), pp. 947–955.
[24] E. Ng, and B.W. Peyton, *Block sparse Cholesky algorithms on advanced uniprocessor computers*, SIAM J. Sci. Comput., 14 (1993), pp. 1034–1056.
[25] A.R.L. Oliveira, and D.C. 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.
[26] J.M. Ortega, *Introduction to Parallel and Vector Solutions of Linear Systems*, Plenum Press, New York, NY, 1988.
[27] A. Ouorou, *A proximal cutting plane method using Chebychev center for nonsmooth convex optimization*, Math. Prog., 119 (2009), pp. 239–271.
[28] M.G.C. 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.
[29] S.J. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, PA, 1996.