axioms

MDPI

Article

On Partial Cholesky Factorization and a Variant of Quasi-Newton Preconditioners for Symmetric Positive Definite Matrices

Benedetta Morini †

Dipartimento di Ingegneria Industriale, Università degli Studi di Firenze, 50134 Florence, Italy; benedetta.morini@unifi.it; Tel.: +39-055-275-8684
† Member of the INdAM Research Group GNCS.

Received: 23 April 2018; Accepted: 20 June 2018; Published: 1 July 2018

Abstract: This work studies limited memory preconditioners for linear symmetric positive definite systems of equations. Connections are established between a partial Cholesky factorization from the literature and a variant of Quasi-Newton type preconditioners. Then, a strategy for enhancing the Quasi-Newton preconditioner via available information is proposed. Numerical experiments show the behaviour of the resulting preconditioner.

Keywords: linear systems; preconditioners; Cholesky factorization; limited memory

1. Introduction

The numerical solution of linear algebraic systems with symmetric positive definite (SPD) matrix is required in a broad range of applications, see e.g., [1–6]. We consider the case where the linear systems are large and investigate their iterative solution by preconditioned Krylov subspace methods [4,7,8]. Our problem takes the form

$$Hx = b, \tag{1}$$

where $H \in \mathcal{R}^{m\times m}$ is SPD. As a particular case of interest we also consider the case where $H = A\Theta A^T$, $A \in \mathcal{R}^{m\times n}$ is a sparse full row-rank matrix and $\Theta \in \mathcal{R}^{n\times n}$ is SPD. Systems of this kind arise in many contexts, such as the solution of linear and nonlinear least-squares problems and the solution of linear programming problems, see e.g., [4,6,9]. The iterative solver employed is the Conjugate Gradient (CG) method or its variants [4] and we propose its use in combination with limited memory preconditioners.

A preconditioner is denoted as limited memory if it can be stored compactly in a few vectors of length m, and its product by a vector calls for scalar products and, possibly, sums of vectors [10]. The limited memory preconditioners studied in this work belong to both the class of Incomplete Cholesky factorizations and to the class of Quasi-Newton preconditioners. Interestingly, they are approximate inverse preconditioners, i.e., they are approximations for H^{-1}. We point out that the preconditioners proposed can also be used for solving symmetric saddle point linear systems iteratively. In fact, the application of constraint or augmented preconditioners involves the factorization of SPD matrices and a cheap approximation of such matrices or their inverses can be convenient [11,12].

Incomplete Cholesky factorizations use the entries of H and may fail for a general SPD matrix, thus requiring strategies for recovering breakdowns. Further, memory requirements are difficult to predict if a drop tolerance is used to reduce the fill-in. For this reason, some factorizations allow a limited number of fill-ins to be created and the bound on the number of nonzero entries can be prefixed column-wise, typically taking into account the number of nonzero entries of the original matrix. Clearly, Incomplete Cholesky factorizations are not suitable when matrix H is dense, which may be the case if $H = A\Theta A^T$ even though A is sparse, see e.g., [3]. A limited memory and breakdown-free

"partial" Cholesky factorization was proposed in [13,14] and used in the solution of compressed sensing, linear and quadratic programming, Lasso problems, maximum cut problems [3,14,15]. This preconditioner is built by computing a trapezoidal partial Cholesky factorization limited to a prefixed and small number of columns and by approximating the resulting Schur complement via its diagonal.

Limited memory Quasi-Newton preconditioners are a class of matrices built drawing inspiration from Quasi-Newton schemes for convex quadratic programming [10]. Given a preconditioner (first-level preconditioner), Quasi-Newton preconditioners provide its update (second-level preconditioner) by exploiting a few vectors of dimension m, i.e., information belonging to a low-dimensional subspace of $\mathcal{R}^m$. Variants of the original Quasi-Newton scheme [10] have been proposed in the literature. They differ in the choice of the low-dimensional subspace [10,16–19] and several instances convey information from the iterative solver, possibly as approximate invariant subspaces.

In this paper, we analyze the connection between the partial Cholesky factorization [13,14] and a variant of the Quasi-Newton preconditioners. We show that the partial Cholesky factorization coincides with a Quasi-Newton preconditioner where the first-level preconditioner is diagonal and the low-dimensional subspace is constituted by a subset of columns of the identity matrix of dimension m. This observation provides a way for building the partial Cholesky factorization which is alternative to the procedures in [13,14] and can offer some advantages in terms of computational effort. Due to the specific form of the low-dimensional subspace spanned by coordinate vectors in $\mathbb{R}^m$, we denote the resulting preconditioner as the Coordinate Limited Memory Preconditioner (Coordinate-LMP). Successively, we propose a strategy for enriching the low-dimensional space that generates the partial Cholesky factorization, and thus enhancing the performance of the preconditioner; such a strategy is guided by the spectral analysis of H preconditioned by the partial Cholesky factorization and it is analyzed from both the theoretical and practical point of view.

The paper is organized as follows. In Section 2 we introduce the partial Cholesky factorization. In Section 3 we show how the partial Cholesky factorization can be formulated as a Quasi-Newton type preconditioner and discuss the application of the two formulations in terms of computational effort. In Section 4 we propose a strategy for enlarging the subspace in the Quasi-Newton formulation and analyze the spectral properties of the preconditioned matrix; the numerical performance of the resulting preconditioner are shown in Section 5.

In the following, for any square matrix B, $\mathrm{diag}(B)$ is the diagonal matrix with the same diagonal entries as B. For a SPD matrix $B \in \mathbb{R}^{m\times m}$, an eigenvalue is denoted either as $\lambda(B)$ or as $\lambda_i(B)$, $1 \leq i \leq m$; the minimum and maximum eigenvalues are denoted as $\lambda_{\min}(B)$ and $\lambda_{\max}(B)$. The identity matrix of dimension q is denoted as I_q. For indicating submatrices we borrow the MATLAB notation. Preconditioned CG method is denoted as PCG.

2. A Limited Memory Partial Cholesky Preconditioner

The convergence behaviour of CG depends on the eigenvalue distribution of H and the condition number of H determines the worst-case behaviour of CG [4,8,20,21]. Further characterizations, possibly sharp, of the convergence behaviour can be gained from additional information on the eigenvalues. More specifically, it is known that if A has t distinct eigenvalues, then the CG method will converge in at most t iterations, while CG applied to a matrix with t tight eigenvalue clusters may not behave similarly as though it were applied to a matrix with t distinct eigenvalues representing the individual clusters ([21], [§5.6.5]).

A proposal for building a partial Cholesky factorization P of H was given by Gondzio in [14] and Bellavia et al., in [13]. It aims at clustering the largest eigenvalues of H at one and reducing both the value of $\lambda_{\max}(P^{-1}H)$ with respect to $\lambda_{\max}(H)$ and the condition number of $P^{-1}H$ with respect to H. The first step is based on the observation that the trace $tr(H)$ of H is such that $tr(H) = \sum_{i=1}^{m} \lambda_i(H)$ and $\lambda_{\max}(H) \leq tr(H)$ since H is symmetric and positive definite. Then, it is possible to handle the

largest eigenvalues of H by an heuristic technique where the largest k, $k \ll m$, diagonal elements of H are identified and the rank-k partial Cholesky factorization of the corresponding columns of H is performed. More specifically, suppose that H can be partitioned as

$$H = \begin{bmatrix} H_{11} & H_{21}^T \\ H_{21} & H_{22} \end{bmatrix},$$

where $H_{11} \in \mathcal{R}^{k\times k}$, $H_{22} \in \mathcal{R}^{(m-k)\times(m-k)}$ and H_{11} contains the k largest diagonal elements of H (throughout the paper, the symmetric row and column permutations required to move the k largest diagonal elements of H to the $(1,1)$ block are ignored to make the presentation simpler). To handle the large eigenvalues of H, the Cholesky factorization limited to the first k columns $\begin{bmatrix} H_{11} \\ H_{21} \end{bmatrix}$ is computed. We denote such factors as $\begin{bmatrix} L_{11} \\ L_{21} \end{bmatrix} \in \mathbb{R}^{m\times k}$, with $L_{11} \in \mathbb{R}^{k\times k}$ unit lower triangular, $L_{21} \in \mathbb{R}^{(m-k)\times k}$, and $D_1 \in \mathbb{R}^{k\times k}$ diagonal positive definite and observe that H can be factorized as

$$H = LD_HL^T \stackrel{\text{def}}{=} \begin{bmatrix} L_{11} & \\ L_{21} & I_{m-k} \end{bmatrix} \begin{bmatrix} D_1 & \\ & S \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ & I_{m-k} \end{bmatrix}, \tag{2}$$

being S the Schur complement of H_{11} in H

$$S = H_{22} - H_{21}H_{11}^{-1}H_{21}^T. \tag{3}$$

Finally, the limited memory Partial Cholesky preconditioner P is obtained by approximating S with its diagonal and setting

$$P = LD_PL^T \stackrel{\text{def}}{=} \begin{bmatrix} L_{11} & \\ L_{21} & I_{m-k} \end{bmatrix} \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ & I_{m-k} \end{bmatrix}, \qquad D_2 = \operatorname{diag}(S). \tag{4}$$

The construction of the preconditioner P is summarized in Algorithm 1 where we use the equalities

$$H_{11} = L_{11}D_1L_{11}^T, \tag{5}$$

$$H_{21}^T = L_{11}D_1L_{21}^T \quad \text{i.e.,} \quad L_{21} = H_{21}L_{11}^{-T}D_1^{-1}, \tag{6}$$

derived from Equation (2).

Algorithm 1 Limited Memory Partial Cholesky Preconditioner.

Given the matrix-vector operators $u \to Hu$, $k > 0$.

1. Form the first k columns of H, i.e. H_{11}, H_{21}.
2. Compute the diagonal entries of H_{22}.
3. Compute L_{11}, D_1, L_{21} as in Equations (5) and (6). Discard H_{11} and H_{21}.
4. Set $D_2 = \operatorname{diag}(H_{22}) - \operatorname{diag}(L_{21}D_1L_{21}^T)$.
5. Let P take the form Equation (4).

This procedure is breakdown-free in exact arithmetic while Incomplete Cholesky factorizations employing a drop tolerance to reduce fill-in may fail for a general SPD matrix. The maximum storage requirement is known in advance and the upper bound on the number of nonzero entries in L is $m + k(m - k/2 - 1/2)$.

Forming the preconditioner calls for the complete diagonal of H. If H has the special form $H = A\Theta A^T$, its main diagonal can be constructed by performing m matrix-vector products $r_i = A^Te_i$, $i = 1,\ldots,m$, and then computing $(H)_{ii} = r_i^T\Theta r_i$. The products A^Te_i are cheap if A is sparse and

involve no extra effort at all if A can be accessed row-wise and then retrieving the ith row comes at no extra cost. Moreover, the k products $A\Theta A^T e_i$ in Step 1 are expected to be cheaper than the products $A\Theta A^T v$ required by a CG-like method because the unit vectors e_i are typically sparser than v.

The cost to perform the factorization Equation (5) is negligible because matrix H_{11} has small dimension k, while, using the first equation in Equation (6), the computation of L_{21} in Step 4 requires solving $m-k$ triangular linear systems of dimension k. Finally, in Step 4 computing $\mathrm{diag}(L_{21}D_1L_{21}^T)$ amounts to scaling the rows of L_{21}^T by the entries of D_1 and performing $m-k$ scalar products between vectors of dimension k.

The spectral properties of $P^{-1}H$ are analyzed in [13] and reported below for completeness.

Theorem 1. *Let k be a positive integer and P be as in Equation (4). Then, k eigenvalues of $P^{-1}H$ are equal to 1 and the remaining are equal to the eigenvalues of $D_2^{-1}S$. Moreover, any eigenvalue $\lambda(D_2^{-1}S)$ lies in the interval $\left[\frac{\lambda_{\min}(S)}{\lambda_{\max}(D_2)}, \frac{\lambda_{\max}(S)}{\lambda_{\min}(D_2)}\right] \subseteq \left[\frac{\lambda_{\min}(H)}{\lambda_{\max}(D_2)}, \frac{\lambda_{\max}(H_{22})}{\lambda_{\min}(D_2)}\right]$.*

Proof of Theorem 1. Theorem 2.1 in [13] proves that k eigenvalues of $P^{-1}H$ are equal to 1 and the remaining are equal to the eigenvalues of $D_2^{-1}S$. As for the bounds on the eigenvalues $\lambda(D_2^{-1}S)$, let $v \in \mathcal{R}^{m-k}$ be an eigenvector of $D_2^{-1}S$. Then, by $D_2^{-1}Sv = \lambda v$ we get $\lambda(D_2^{-1}S) = \dfrac{v^TSv}{v^TD_2v}$ and

$$\lambda(D_2^{-1}S) \geq \frac{\lambda_{\min}(S)}{\lambda_{\max}(D_2)} \geq \frac{\lambda_{\min}(H)}{\lambda_{\max}(D_2)}, \tag{7}$$

$$\lambda(D_2^{-1}S) \leq \frac{\lambda_{\max}(S)}{\lambda_{\min}(D_2)} \leq \frac{\lambda_{\max}(H_{22})}{\lambda_{\min}(D_2)}, \tag{8}$$

where we used the bounds $\lambda_{\min}(H) \leq \lambda(S) \leq \lambda_{\max}(H_{22})$, see [6]. □

We point out that the above preconditioner was used in [13] in conjunction with Deflated-CG [22] in order to handle also the smallest eigenvalues of $P^{-1}H$.

We conclude this section observing that the Partial Cholesky preconditioner can be formulated as an Approximate Inverse preconditioner. In fact, from (4) matrix P^{-1} can be factorized as the product of sparse matrices and takes the form

$$P^{-1} = L^{-T}D_P^{-1}L^{-1} = \begin{bmatrix} L_{11}^{-T} & -L_{11}^{-T}L_{21}^T \\ & I_{m-k} \end{bmatrix} \begin{bmatrix} D_1^{-1} & \\ & D_2^{-1} \end{bmatrix} \begin{bmatrix} L_{11}^{-1} & \\ -L_{21}L_{11}^{-1} & I_{m-k} \end{bmatrix}. \tag{9}$$

3. Limited Memory Quasi-Newton Type Preconditioners

Limited memory Quasi-Newton type preconditioners for SPD matrices were proposed in several works, see e.g., [10,17–19]. These preconditioners are generated using a small number k of linear independent vectors in $\mathbb{R}^m$.

Let us consider the formulation by Gratton et al. in [19]. Suppose that a first preconditioner M (called first-level preconditioner) is available. To improve the efficiency of the first-level preconditioner, a class of limited memory preconditioners (called second-level preconditioners) is defined on the base of the explicit knowledge of an m by k, $k \ll m$, full rank matrix Z. The aim of the second-level preconditioner is to capture directions lying in the range of HZ which have been left out by the first-level preconditioner and are slowing down the convergence of the CG solver; e.g., this is the case when the first-level preconditioner is able to cluster many eigenvalues at 1 with relatively few outliers.

Let $M \in \mathbb{R}^{m\times m}$ be symmetric and positive definite, $Z \in \mathbb{R}^{m\times k}$, $k \ll m$, be a full column-rank matrix. The symmetric second-level preconditioner, say Π, takes the form

$$\Pi = (I - TH)M(I - HT) + T, \qquad T = Z(Z^THZ)^{-1}Z^T. \tag{10}$$

The spectral properties of ΠH established in ([19], [Lemma 3.3, Theorem 3.4]) are summarized below.

Theorem 2. *Let H and M be symmetric and positive definite matrices of order m, $Z \in \mathbb{R}^{m\times k}$, be a full column-rank matrix and Π be given by Equation (10). Then the matrix Π is positive definite.*

Let the positive eigenvalues $\lambda_1(MH), \dots \lambda_m(MH)$ of MH be sorted in nondecreasing order. Then the set of eigenvalues $\lambda_1(\Pi H), \dots \lambda_m(\Pi H)$ of ΠH can be split in two subsets:

$$\lambda_i(MH) \leq \lambda_i(\Pi H) \leq \lambda_{i+k}(MH) \quad \text{for } i = 1, \dots, m-k, \tag{11}$$

and

$$\lambda_i(\Pi H) = 1 \quad \text{for } i = m-k+1, \dots, m.$$

Equation (10) provides a general formulation for designing second-level preconditioners and was inspired by the BFGS inverse Hessian approximation in Quasi-Newton algorithms [9]. In fact, if the BFGS method with exact linesearch is applied to a quadratic function, then the inverse Hessian approximation generated has the form of Π in Equation (10); we refer to ([19], [§2]) for details on this interpretation. In the general setting, any set of linearly independent vectors can provide candidates for the columns of Z and gives rise to a preconditioner of form Equation (10); k eigenvalues of ΠH equal to 1 are obtained while the remaining eigenvalues satisfy the relevant interlacing property Equation (11). On the other hand, specific choices for Z guided by information on the problem at hand are preferable.

The preconditioner Π has been specialized to the case of: *spectral-LMP* where the columns of Z consist of eigenvectors of MH, *Ritz-LMP* where the columns of Z consist of Ritz vectors generated by the iterative linear solver, *Quasi-Newton-LMP* where the columns of Z consist of descent directions from optimization methods applied to continuous optimization problems, see e.g., [10,16–19]. All these vectors are often available when systems with multiple right-hand sides of slowly varying sequence of systems are considered.

In this work, we propose and analyze preconditioners of the form Equation (10) where the first-level preconditioner is the diagonal matrix D_P^{-1} given in Equation (4) and Z is chosen as a suitable submatrix of the identity matrix I_m, i.e., HZ consists of k properly chosen columns of H. Due to the fact that Z consists of coordinate vectors in $\mathbb{R}^m$ we denote the resulting limited memory preconditioner as Coordinate-LMP. We start analyzing the case where

$$M = D_P^{-1}, \text{ and } Z = I_m(:, 1:k). \tag{12}$$

Forming the preconditioner Π with Equation (12) requires the steps listed in Algorithm 2.

Algorithm 2 Coordinate Limited Memory Preconditioner.

Given the matrix-vector operators $u \to Hu$, $k > 0$.

1. Form the first k columns of H, i.e. H_{11}, H_{21}.
2. Compute the diagonal entries of H_{22}.
3. Compute L_{11}, D_1 as in Equation (5).
4. Set $D_2 = \text{diag}(H_{22}) - \text{diag}(H_{21}H_{11}^{-1}H_{21}^T)$, D_P as in Equation (4).
5. Set M and Z as in Equation (12).
6. Let Π take the form Equation (10).

In this specific variant, Π coincides with the inverse of the Partial Cholesky preconditioner P. We show this fact in the following theorem.

Theorem 3. *Let k be a positive integer and P be as in Equation (4). If matrix Π has the form Equation (10) with M and Z as in Equation (12), then $\Pi = P^{-1}$. Moreover,*

$$P^{-1}H = \Pi H = \begin{bmatrix} I & H_{11}^{-1}H_{21}^T(I - D_2^{-1}S) \\ 0 & D_2^{-1}S \end{bmatrix}. \tag{13}$$

Proof of Theorem 3. By Equation (9) it follows

$$P^{-1} = \begin{bmatrix} L_{11}^{-T}D_1^{-1}L_{11}^{-1} + L_{11}^{-T}L_{21}^T D_2^{-1}L_{21}L_{11}^{-1} & -L_{11}^{-T}L_{21}^T D_2^{-1} \\ -D_2^{-1}L_{21}L_{11}^{-1} & D_2^{-1} \end{bmatrix}.$$

Using Equations (5) and (6) we obtain $L_{21}L_{11}^{-1} = H_{21}L_{11}^{-T}D_1^{-1}L_{11}^{-1} = H_{21}H_{11}^{-1}$ and we conclude

$$P^{-1} = \begin{bmatrix} H_{11}^{-1} + H_{11}^{-1}H_{21}^T D_2^{-1}H_{21}H_{11}^{-1} & -H_{11}^{-1}H_{21}^T D_2^{-1} \\ -D_2^{-1}H_{21}H_{11}^{-1} & D_2^{-1} \end{bmatrix}.$$

Now consider Π and first observe that the matrices appearing in Equation (10) have the form:

$$HZ = \begin{bmatrix} H_{11} \\ H_{21} \end{bmatrix}, \qquad Z^T HZ = H_{11}, \qquad T = \begin{bmatrix} H_{11}^{-1} & \\ & 0 \end{bmatrix}, \tag{14}$$

$$(I - TH) = \begin{bmatrix} 0 & -H_{11}^{-1}H_{21}^T \\ 0 & I_{m-k} \end{bmatrix}, \qquad (I - HT) = \begin{bmatrix} 0 & 0 \\ -H_{21}H_{11}^{-1} & I_{m-k} \end{bmatrix}. \tag{15}$$

Then,

$$\begin{aligned} \Pi &= \begin{bmatrix} 0 & -H_{11}^{-1}H_{21}^T \\ & I_{m-k} \end{bmatrix} \begin{bmatrix} D_1^{-1} & \\ & D_2^{-1} \end{bmatrix} \begin{bmatrix} 0 & \\ -H_{21}H_{11}^{-1} & I_{m-k} \end{bmatrix} + \begin{bmatrix} H_{11}^{-1} & \\ & 0 \end{bmatrix} \\ &= \begin{bmatrix} H_{11}^{-1} + H_{11}^{-1}H_{21}^T D_2^{-1}H_{21}H_{11}^{-1} & -H_{11}^{-1}H_{21}^T D_2^{-1} \\ -D_2^{-1}H_{21}H_{11}^{-1} & D_2^{-1} \end{bmatrix}, \end{aligned}$$

i.e., $P^{-1} = \Pi$. Finally, it is trivial to verify that $P^{-1}H$ takes the upper block triangular form Equation (13) which also provides the spectrum of $P^{-1}H$ stated in Theorem 1. □

3.1. Application of the Preconditioners

In the previous section, we have shown that P^{-1} and Π can reduce to the same preconditioner. Clearly, their application as a preconditioner calls for matrix-vector products and this computational cost may depend on the formulation used i.e., either Equation (9) or Equations (10) and (12). Let us analyze the cost for performing matrix-vector products of both P^{-1} and Π times a vector. As stated in Section 2, the symmetric row and column permutations required to move the k largest diagonal elements of H to the $(1,1)$ block are ignored to make the presentation simpler.

If the triangular factors in Equation (9) have been formed, the application of the Partial Cholesky preconditioner P^{-1} to a vector amounts to: two products of L_{11}^{-1} by a vector $\mathbb{R}^k$, one matrix-vector product with D_P^{-1}, $m-k$ scalar products in $\mathbb{R}^k$, k scalar products in $\mathbb{R}^{m-k}$. It is worthy pointing out that the partial Cholesky factorization may be dense. In fact, for sparse Cholesky factorizations, permutation matrices are normally chosen to enhance the sparsity of the triangular factor, see e.g., [4], while here we choose the k columns in advance from the largest diagonals of H.

The application of the Coordinate limited memory preconditioner Π to a vector also calls for matrix-vector and scalar products. The computation of a product, say Πv, can be implemented efficiently using Equations (10) and (14) and performing the following steps

$$\begin{aligned} a &= H_{11}^{-1}(Z^T v), \\ b &= D_P^{-1}\left(v - \begin{bmatrix} H_{11} \\ H_{21} \end{bmatrix} a\right), \\ \Pi v &= b - Z\left(H_{11}^{-1}\begin{bmatrix} H_{11} & H_{21}^T \end{bmatrix} b\right) + Za. \end{aligned}$$

These steps call for two products of H_{11}^{-1} by a vector in $\mathbb{R}^k$, one matrix-vector product with D_P^{-1}, m scalar products in $\mathbb{R}^k$, k scalar products in $\mathbb{R}^m$. The cost for the product of Z by a vector is negligible due to the form of Z.

The computational cost for applying both P^{-1} and Π to a vector is expected to be comparable if the scalar products performed have similar computational effort; this is the case when the density of the first k columns of L^{-1} is similar to the density of the first k columns of H. On the other hand, if the density of the first k columns of L^{-1} is considerably larger than the density of the first k columns of H, the application of P^{-1} is less convenient than the application of Π. This issue is shown in Table 1 where we report on the numerical solution of four linear systems with matrices of the form $H = AA^T$ and matrix A from the LPnetlib group in the University of Florida Sparse Matrix Collection [23]. Preconditioned Conjugate Gradient [4] is applied with both the Partial Cholesky preconditioner and the Coordinate-LMP , setting $k = 50$. We display the dimension m, n of A, the number of PCG iterations (Itns), the execution time in seconds (Time), the density of first k columns of L^{-1} ($\text{dens}_{\text{L,k}}$), the density of the first k columns of H ($\text{dens}_{\text{H,k}}$); the density is computed as the ratio between the number of nonzero entries and the overall number of entries of the mentioned submatrices.

Table 1. Solution of systems with $H = AA^T$, $A \in \mathbb{R}^{m\times n}$, using Partial Cholesky preconditioner and Coordinate-LMP with $k = 50$. Number of PCG iterations (Itns), execution time in seconds (Time), density of first k columns of L^{-1} ($\text{dens}_{\text{L,k}}$), density of the first k columns of H ($\text{dens}_{\text{H,k}}$).

			P^{-1}			Π		
Test name	m	n	**Itns**	**Time**	$\text{dens}_{\text{L,k}}$	**Itns**	**Time**	$\text{dens}_{\text{H,k}}$
lp_dfl001	6071	12,230	736	3.87	6.0×10^{-1}	736	1.65	2.8×10^{-2}
lpi_ceria3d	3576	4400	79	0.42	8.4×10^{-1}	80	0.27	3.9×10^{-1}
lp_ken_13	28,632	42,659	186	1.82	1.1×10^{-2}	186	1.70	1.1×10^{-2}
lp_osa_60	10,280	243,246	35	2.92	9.5×10^{-1}	39	3.09	8.0×10^{-1}

We observe that $\text{dens}_{\text{L,k}}$ is larger than $\text{dens}_{\text{H,k}}$ in the first two tests and runs with P^{-1} are slower than with Π, while the two densities are similar in the last two runs as well as the timings obtained using P^{-1} and Π.

4. Enlarging the Subspace in the Coordinate-LMP Preconditioner

The Partial Cholesky preconditioner P and the Coordinate-LMP Π with first level preconditioner and subspace as in Equation (12) aim at clustering the largest eigenvalues of H. In this section we investigate how to enlarge the subspace Z by means of information available from Algorithm 2 and the potential impact on the resulting preconditioner.

We consider the Coordinate-LMP Equation (10), suppose to use again $M = D_P^{-1}$ as first level preconditioner, and to select a larger number of columns of I_m for the subspace defined by Z. We let

$$M = D_P^{-1}, \quad Z = I_m(:, 1:q), \quad q = k + \ell, \tag{16}$$

with ℓ being a positive integer, i.e., besides the first k columns of I_m used in Equation (12) we employ ℓ more columns (for simplicity suppose the first ℓ subsequent columns). The effect of this choice can be analyzed by considering the block partition of H where the leading block $\widetilde{H}_{11}$ has dimension q by q, i.e.,

$$H = \begin{bmatrix} \widetilde{H}_{11} & \widetilde{H}_{21}^T \\ \widetilde{H}_{21} & \widetilde{H}_{22} \end{bmatrix}, \tag{17}$$

with $\widetilde{H}_{11} \in \mathcal{R}^{q\times q}$, $\widetilde{H}_{22} \in \mathcal{R}^{(m-q)\times(m-q)}$. Analogously, let us consider the block partition of D_P in Equation (4) where the leading block $\widetilde{D}_{P,1}$ has dimension q by q, i.e.,

$$D_P = \begin{bmatrix} \widetilde{D}_{P,1} & \\ & \widetilde{D}_{P,2} \end{bmatrix} \stackrel{\text{def}}{=} \begin{bmatrix} D_P(1:q,1:q) & \\ & \widetilde{D}_P(q+1:m,q+1:m) \end{bmatrix}, \tag{18}$$

with $\widetilde{D}_{P,1} \in \mathcal{R}^{q\times q}$, $\widetilde{D}_{P,2} \in \mathcal{R}^{m-q\times m-q}$.

The spectral properties of the resulting Coordinate-LMP preconditioner are given in the following theorem.

Theorem 4. *Let q be a positive integer, H and D_P be symmetric positive definite matrices partitioned as in Equations (17) and (18). If matrix Π has the form Equation (10) with M and Z as in Equation (16), then ΠH has q eigenvalues equal to 1 and the remaining are equal to the eigenvalues of $\widetilde{D}_{P,2}^{-1}\widetilde{S}$ where*

$$\widetilde{S} = \widetilde{H}_{22} - \widetilde{H}_{21}\widetilde{H}_{11}^{-1}\widetilde{H}_{21}^T. \tag{19}$$

Moreover any eigenvalue $\lambda(\widetilde{D}_{P,2}^{-1}\widetilde{S})$ lies in the interval $\left[\frac{\lambda_{\min}(\widetilde{S})}{\lambda_{\max}(\widetilde{D}_{P,2})}, \frac{\lambda_{\max}(\widetilde{S})}{\lambda_{\min}(\widetilde{D}_{P,2})}\right] \subseteq \left[\frac{\lambda_{\min}(H)}{\lambda_{\max}(\widetilde{D}_{P,2})}, \frac{\lambda_{\max}(\widetilde{H}_{22})}{\lambda_{\min}(\widetilde{D}_{P,2})}\right]$.

Proof of Theorem 4. Similarly to the proof of Theorem 3 we have

$$HZ = \begin{bmatrix} \widetilde{H}_{11} \\ \widetilde{H}_{21} \end{bmatrix}, \quad Z^THZ = \widetilde{H}_{11}, \quad T = \begin{bmatrix} \widetilde{H}_{11}^{-1} & \\ & 0 \end{bmatrix}, \tag{20}$$

$$(I - TH) = \begin{bmatrix} 0 & -\widetilde{H}_{11}^{-1}\widetilde{H}_{21}^T \\ 0 & I_{m-q} \end{bmatrix}, \quad (I - HT) = \begin{bmatrix} 0 & 0 \\ -\widetilde{H}_{21}\widetilde{H}_{11}^{-1} & I_{m-q} \end{bmatrix}. \tag{21}$$

Then, by Equation (10) we get

$$\begin{aligned} \Pi &= \begin{bmatrix} 0 & -\widetilde{H}_{11}^{-1}\widetilde{H}_{21}^T \\ & I_{m-q} \end{bmatrix} \begin{bmatrix} \widetilde{D}_{P,1}^{-1} & \\ & \widetilde{D}_{P,2}^{-1} \end{bmatrix} \begin{bmatrix} 0 & \\ -\widetilde{H}_{21}\widetilde{H}_{11}^{-1} & I_{m-q} \end{bmatrix} + \begin{bmatrix} \widetilde{H}_{11}^{-1} & \\ & 0 \end{bmatrix} \\ &= \begin{bmatrix} \widetilde{H}_{11}^{-1} + \widetilde{H}_{11}^{-1}\widetilde{H}_{21}^T\widetilde{D}_{P,2}^{-1}\widetilde{H}_{21}\widetilde{H}_{11}^{-1} & -\widetilde{H}_{11}^{-1}\widetilde{H}_{21}^T\widetilde{D}_{P,2}^{-1} \\ -\widetilde{D}_{P,2}^{-1}\widetilde{H}_{21}\widetilde{H}_{11}^{-1} & \widetilde{D}_{P,2}^{-1} \end{bmatrix}, \end{aligned}$$

and consequently

$$\Pi H = \begin{bmatrix} I_q & (\widetilde{H}_{11})^{-1}\widetilde{H}_{21}^T(I - \widetilde{D}_{P,2}^{-1}\widetilde{S}) \\ 0 & \widetilde{D}_{P,2}^{-1}\widetilde{S} \end{bmatrix}.$$

Bounds on the eigenvalues $\lambda(\widetilde{D}_{P,2}^{-1}\widetilde{S})$ can be derived by fixing an eigenvector $v \in \mathcal{R}^{m-k}$ of $\widetilde{D}_{P,2}^{-1}\widetilde{S}$. Then, by $\widetilde{D}_{P,2}^{-1}\widetilde{S}v = \lambda v$ we get $\lambda(\widetilde{D}_{P,2}^{-1}\widetilde{S}) = \dfrac{v^T\widetilde{S}v}{v^T\widetilde{D}_{P,2}v}$. Thus, similarly to the proof of Theorem 1 it follows

$$\lambda(\widetilde{D}_{P,2}^{-1}\widetilde{S}) \geq \frac{\lambda_{\min}(\widetilde{S})}{\lambda_{\max}(\widetilde{D}_{P,2})} \geq \frac{\lambda_{\min}(H)}{\lambda_{\max}(\widetilde{D}_{P,2})}, \tag{22}$$

$$\lambda(\widetilde{D}_{P,2}^{-1}\widetilde{S}) \leq \frac{\lambda_{\max}(\widetilde{S})}{\lambda_{\min}(\widetilde{D}_{P,2})} \leq \frac{\lambda_{\max}(\widetilde{H}_{22})}{\lambda_{\min}(\widetilde{D}_{P,2})}, \tag{23}$$

since $\lambda_{\min}(\widetilde{S}) \geq \lambda_{\min}(H)$ and $\lambda_{\max}(\widetilde{S}) \leq \lambda_{\max}(\widetilde{H}_{22})$ [6]. □

Remark 1. *Let* $I_1 = \left[\frac{\lambda_{\min}(H)}{\lambda_{\max}(D_2)}, \frac{\lambda_{\max}(H_{22})}{\lambda_{\min}(D_2)}\right]$ *be the interval in the statement of Theorem 1 and* $I_2 = \left[\frac{\lambda_{\min}(H)}{\lambda_{\max}(\widetilde{D}_{P,2})}, \frac{\lambda_{\max}(\widetilde{H}_{22})}{\lambda_{\min}(\widetilde{D}_{P,2})}\right]$ *be the interval in the statement of Theorem 4. It holds* $\lambda_{\max}(\widetilde{H}_{22}) \leq \lambda_{\max}(H_{22})$ *by the Cauchy Interlace Theorem [24] [p. 396]. Moreover, for any choice of* $\widetilde{D}_{P,2}$ *trivially it holds*

$$\lambda_{\min}(\widetilde{D}_{P,2}) \geq \lambda_{\min}(D_2), \qquad \lambda_{\max}(\widetilde{D}_{P,2}) \leq \lambda_{\max}(D_2).$$

Then, $I_2 \subseteq I_1$.

A comparison between bounds Equations (7) and (8) and Equations (22) and (23) suggests that the choice of the extremal diagonal elements of S can be beneficial for improving, at a low computational cost, the clustering of the eigenvalues. In fact, choosing the extremal diagonal elements of S promotes a reduction of the width of the interval containing the eigenvalues of ΠH and this issue can favorably affect the performance of the iterative solver and the condition number of ΠH.

Accordingly to Remark 1, let I_1 and I_2 be the intervals containing the eigenvalues of ΠH with Π generated by Equation (12) and by Equation (16) respectively. If the ℓ largest diagonal entries of $\mathrm{diag}(S)$ are contained in matrix $\widetilde{D}_{P,1}$ in Equation (18) and are separated from the remaining, then we obtain an increase of the lower bound of I_2 with respect to lower bound of I_1; clearly, the better the ℓ largest diagonal entries of S are separated from the remaining elements of $\mathrm{diag}(S)$ the larger such increase is. Handling small eigenvalues of ΠH seems to be convenient when enlarging the subspace Z for Π as the Partial Cholesky factorization is intended to take care of the largest eigenvalues of H. Alternatively, if the ℓ smallest diagonal entries of S are contained in matrix $\widetilde{D}_{P,1}$ in Equation (18) and are separated from the remaining, the upper bound of I_2 is expected to be smaller than the upper bound of I_1.

As mentioned in Section 2, in [13] the partial Cholesky preconditioner was used in conjunction with Deflated-CG [22] in order to handle the small eigenvalues of $P^{-1}H$. A rough approximation of the five smallest eigenvalues of $P^{-1}H$ was injected into the Krylov subspace and yielded an improvement in some tests where Deflated-CG performed consistently fewer iterations than the usual CG. Although the deflation strategy brought undeniable benefits in terms of reducing the number of CG iterations, it involved an extra storage requirement and an extra cost which ultimately increased the overall solution time. In fact, the application of such a strategy was convenient when the eigenvalue information was used for a sequence of related linear systems, such as slowly varying systems or systems with multiple right-hand-side vectors. The strategy, presented in this section and based on selecting a prefixed number of the largest diagonal entries of S, can be viewed as a cheap alternative procedure for handling the smallest eigenvalues of $P^{-1}H$.

5. Numerical Results

In this section we present a preliminary numerical validation of the performance of the Coordinate-LMP discussed in Sections 3 and 4. All numerical experiments reported were performed

on a Dell Latitude E4200 with a Intel(R) Cote(TM)2 Duo CPU U9600, @1.60 GHz, RAM 3.00 GB, using MATLAB and machine precision 2.2×10^{-16}.

We report results on a set of 18 linear systems where the coefficient matrix has the form $H = A\Theta A^T$ and the right-hand side is chosen as a normally distributed vector. In Table 2 we list the name of matrices $A \in \mathbb{R}^{m\times n}$ used along with their dimensions and the density of both A and H computed as the ratio between the number of nonzero entries and the number of rows.

Table 2. Test problems with $H = A\Theta A^T$: source and name of $A \in \mathbb{R}^{m\times n}$, dimension of A, density of A (dens(A)) and density of H (dens(H)).

Group/Test Name	m	n	dens(A)	dens(H)
LPnetlib/lp_bnl2	2424	4486	6.5	12.6
LPnetlib/lp_d2q06c	2171	5831	15.2	25.9
LPnetlib/lp_dfl001#	6071	12,230	5.9	13.5
LPnetlib/lp_degen3#	1503	2604	16.9	67.7
LPnetlib/lp_ganges	1309	1706	5.3	12.7
LPnetlib/lp_ken_13	28,632	42,659	3.4	5.7
LPnetlib/lp_ken_18	105,127	154,699	3.4	5.8
LPnetlib/lp_osa_30	4350	104,374	139.0	100.4
LPnetlib/lp_osa_60	10,280	243,246	137.0	98.9
LPnetlib/lp_pds_10#	16,558	49,932	6.5	9.0
LPnetlib/lp_pilot	1441	4680	30.8	86.3
LPnetlib/lp_pilot87	2030	6680	36.9	117.5
LPnetlib/ lp_sierra#	1227	2735	6.5	4.7
Meszaros/cq9	9278	21,534	10.4	23.9
Meszaros/nl	7039	15,325	6.7	14.9
M5_3000_maxcut	3000	9,000,000	1	3000
M2_K4_5000_maxcut	5000	25,000,000	1	5000
M3_K4_5000_maxcut	5000	25,000,000	1	5000

The first 15 matrices A are taken from the groups LPnetlib and Meszaros in the University of Florida Sparse Matrix Collection [23] and are constraint matrices of linear programming problems; in the associated linear systems we set $\Theta = I_n$. The symbol "#" indicates when matrix A was regularized by a shift 10^{-2} in order to get a numerically nonsingular matrix H. We observe that both A and H are sparse and H can be preconditioned by either Incomplete Cholesky factorizations or by our preconditioner. The last three systems were generated by the dual logarithmic barrier method [3] applied to semidefinite programming relaxations of maximum cut problems. In these problems each row of A is the unrolled representation of a rank-one $m \times m$ matrix and has one nonzero entry while Θ is a full matrix of dimension $m^2 \times m^2$ defined as the Kronecker product of matrices of dimension $m \times m$; consequently H is full. Iterative methods are an option for solving these systems when H cannot be allocated due to memory limitations [3]. Incomplete Cholesky factorizations are not applicable while our preconditioner is viable.

The linear systems have been solved by Preconditioned Conjugate Gradient (PCG) method starting from the null initial guess and using the stopping criterion:

$$\|Hx - b\| \leq 10^{-6}\|b\|. \tag{24}$$

A failure is declared after 1000 iterations. The preconditioner Π was applied as described in Section 3.1.

The preconditioners used in our tests are: the Incomplete Cholesky factorization with zero-fill (IC(0)) computed by the built-in MATLAB function ichol, the Coordinate-LMP Equation (12) with $k = 50$, and the Coordinate-LMP Equation (16) with $(k, \ell) = (50, 25)$. Concerning the preconditioner with enlarged subspace Equation (16), we consider the two strategies for enlarging Z discussed at the end of Section 4. The first strategy consists in selecting the columns of I_m associated to the ℓ

largest diagonal entries of $D_2 = \text{diag}(S)$ and in the following is denoted as D2_LARGE. The second strategy consists in selecting the columns of I_m associated to the ℓ smallest diagonal entries of D_2 and is denoted as D2_SMALL. In the following tables, the symbol "*" indicates a failure of CG solver while the symbol "†" indicates a failure in computing the Incomplete Cholesky factorization due to encountering a nonpositive pivot. The timing in seconds "Time" includes the construction of the preconditioner and the total execution time for PCG.

Our focus is on the reliability of the preconditioners tested and the computational gain provided. Regarding the latter issue, clearly it depends on both the number of PCG iterations and the cost of PCG per iteration.

Table 3 displays the results obtained solving the linear systems with: unpreconditioned CG, CG coupled with IC(0), CG coupled with the Coordinate-LMP Equation (12), CG coupled with the Coordinate-LMP Equation (16) and implemented using the D2_LARGE strategy. We report the number of PCG iterations (Itns) and the timing (Time). We observe that IC(0) factorization cannot be applied to the linear systems deriving from maximum cut problems since the resulting matrices H are full, see Table 2.

We start observing that CG preconditioned by the Coordinate-LMP preconditioners Equation (16) and the D2_LARGE strategy solved all the systems, whereas a breakdown of IC(0) occurred five times out of fifteen as a nonpositive pivot was encountered. In eight systems out of ten, PCG with IC(0) required several iterations considerably smaller than the number of iterations performed with the limited memory preconditioners; correspondingly the execution time was favorable to IC(0) preconditioner. On the other hand, IC(0) preconditioner was less effective than the limited memory preconditioner on problems lp_osa_30 and lp_osa_60; this occurrence is motivated by several linear iterations comparable to that of limited preconditioners and the density of the Cholesky factor, cf. Table 2. Finally, we point out that breakdowns of IC(0) can be recovered using Incomplete Cholesky factorization with very small threshold dropping and, consequently, high fill-in in the Incomplete Cholesky factor and computational overhead.

Comparing the limited memory preconditioners in terms of CG iterations, we observe that enlarging the subspace provides a reduction in such a number. The gain in CG iterations using the enlarged subspace is very limited for Problems lp_ganges, lp_dfl001 and lp_pds_10, while it varies between 3% and 52% for the remaining problems. Savings in time depend on both the reduction in the number of CG iterations performed and the cost of matrix-vector products. Namely, when the application of H is cheap, savings in PCG iterations between 11% and 31% do not yield a significant gain in time, see lp_degen3, lp_ken_13, lp_pilot; on the other hand when matrix-vector products are expensive, saving in time can occur even in the presence of a mild reduction in the number of CG iterations, see lp_pds_10, M2_K4_5000_maxcut, M3_K4_5000_maxcut. Interestingly the cost for forming and applying the preconditioners does not offset the convergence gain in PCG; this feature is evident from the value Time in runs where the reduction in PCG iterations is small, see lp_ganges and M5_3000_maxcut. In fact, we can expect that for matrices having the same eigenvalue distribution as our test matrices, and a substantial number of nonzero elements, significant reductions in computing time can be achieved with the Quasi-Newton preconditioner and enlarged subspace.

The effect of enlarging the subspace in the Coordinate-LMP preconditioner is further analyzed in Table 4 where we report the minimum $\lambda_{\min}$ and maximum $\lambda_{\max}$ eigenvalues of the original matrix H and of the preconditioned matrices ΠH in four problems. We observe that the maximum eigenvalue of the preconditioned matrix is consistently smaller than the eigenvalue of H and this shows the effectiveness of handling the largest eigenvalues by using the trace of H. On the other hand, the smallest eigenvalue of the matrix preconditioned by the Partial Cholesky factorization (Π with $k = 50$) is moved towards the origin. As shown in the table, an increase in the value of the smallest eigenvalue can be obtained with the D2_LARGE implementation for Π while the effect on the largest eigenvalue is marginal, as expected.

Table 3. Solution of systems with $H = A\Theta A^T$: unpreconditioned CG, CG coupled with IC(0), CG coupled with Coordinate-LMP Equation (12) $k = 50$, CG coupled with Coordinate-LMP Equation (16) $(k, \ell) = (50, 25)$ and D2_LARGE implementation. Number of PCG iterations (Itns), execution time in seconds (Time) for building the preconditioner and for PCG.

	CG		**PCG with IC(0)**		**PCG with Π** $k = 50$		**PCG with Π** $(k, \ell) = (50, 25)$ **D2_LARGE Implementation**	
Test Name	**Itns**	**Time**	**Itns**	**Time**	**Itns**	**Time**	**Itns**	**Time**
lp_bnl2	*		51	0.1	353	0.4	295	0.3
lp_d2q06c	*		†		*		844	1.2
lp_dfl001	*		320	0.7	736	1.7	720	1.7
lp_degen3	*		284	0.4	599	0.7	530	0.7
lp_ganges	215	0.1	35	0.1	126	0.1	124	0.1
lp_ken_13	510	3.3	87	0.7	186	1.6	165	1.6
lp_ken_18	*		167	10.5	499	20.7	485	20.2
lp_osa_30	126	3.4	47	2.4	38	1.5	18	0.9
lp_osa_60	127	7.7	26	7.4	39	3.6	26	2.8
lp_pds_10	*		431	5.2	897	8.1	892	7.6
lp_pilot	*		†		369	0.6	252	0.4
lp_pilot87	*		†		559	1.7	505	1.6
lp_sierra	*		241	0.1	*		590	0.3
cq9	*		†		554	2.9	472	2.5
nl	*		†		944	2.6	621	1.7
M5_3000_maxcut	60	2.5			55	2.5	51	2.4
M2_K4_5000_maxcut	497	52.2			427	46.6	413	44.9
M3_K4_5000_maxcut	641	65.8			585	62.5	542	58.5

Table 4. Minimum eigenvalue $\lambda_{\min}$ and maximum eigenvalue $\lambda_{\max}$ of: matrix H, matrix ΠH with Coordinate-LMP Equation (12) $k = 50$, matrix ΠH with Coordinate-LMP Equation (16) $(k, \ell) = (50, 25)$.

	H		ΠH **with** $k = 50$		ΠH **with** $(k, \ell) = (50, 25)$ **D2_LARGE Implementation**	
Test Name	$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$	$\lambda_{\min}$	$\lambda_{\max}$
lp_d2q06c	6.3×10^{-4}	1.2×10^{6}	3.3×10^{-5}	6.4×10^{0}	4.8×10^{-5}	5.7×10^{0}
lp_osa_30	1.0×10^{0}	1.8×10^{6}	2.4×10^{-5}	2.8×10^{0}	5.9×10^{-4}	2.2×10^{0}
lp_pilot	1.0×10^{-2}	1.0×10^{6}	2.5×10^{-4}	1.2×10^{1}	1.4×10^{-3}	1.2×10^{1}
nl	7.0×10^{-3}	8.2×10^{4}	1.6×10^{-4}	7.3×10^{0}	5.7×10^{-4}	6.7×10^{0}

We conclude our presentation reporting the performance of the D2_SMALL implementation in Table 5; the number of PCG iterations is displayed and part of the results in Table 3 are repeated for clarity. We recall that the number of unit eigenvalues is $q = k + \ell$ for both the D2_LARGE and D2_SMALL implementations, but the former strategy is more effective than the latter. In fact, the behaviour of the Partial Cholesky factorization and of the D2_SMALL implementation of the Coordinate-LMP preconditioner are similar in terms of PCG iterations, apart for problems lp_ganges, lp_sierra, cq9 where the latter approach is convenient. This confirms that the largest eigenvalues are handled by the Partial Cholesky factorization and a further reduction of the upper bound on the eigenvalues is not useful.

Table 5. Solution of systems with $H = A\Theta A^T$, $A \in \mathbb{R}^{m\times n}$: CG coupled with Coordinate-LMP Equation (12) $k = 50$, CG coupled with Coordinate-LMP Equation (16) $(k, \ell) = (50, 25)$ and D2_LARGE implementation, CG coupled with Coordinate-LMP Equation (16) $(k, \ell) = (50, 25)$ and D2_SMALL implementation. Number of PCG iterations (Itns).

	PCG with Π $k = 50$	**PCG with Π $(k, \ell) = (50, 25)$ D2_LARGE Implementation**	**PCG with Π $(k, \ell) = (50, 25)$ D2_SMALL Implementation**
Test Name	**Itns**	**Itns**	**Itns**
lp_bnl2	353	295	353
lp_d2q06c	*	844	*
lp_dfl001	736	720	733
lp_degen3	599	530	595
lp_ganges	126	124	78
lp_ken_13	186	165	186
lp_ken_18	499	485	520
lp_osa_30	38	18	37
lp_osa_60	39	26	39
lp_pds_10	897	892	900
lp_pilot	369	252	361
lp_pilot87	559	505	565
lp_sierra	*	590	706
cq9	554	472	528
nl	944	621	946
M5_3000_maxcut	55	51	56
M2_K4_5000_maxcut	427	413	428
M3_K4_5000_maxcut	585	542	596

Summarizing, the results presented in this section seem to indicate that: enlarging the subspace with columns associated to the largest diagonal entries of the Schur complement S in Equation (3) reduces the number of PCG iterations; the cost for forming and applying the preconditioner with enlarged subspace does not offset the gain from reducing PCG iterations; saving in times are obtained accordingly to the cost of matrix-vector products in PCG. Moreover, the Quasi-Newton preconditioner proposed is suitable for application to dense matrices H of the form $H = A\Theta A^T$, A sparse, where computing the Incomplete Cholesky factor is too expensive in terms of computational cost and/or storage requirement.

Funding: This work was partially supported by INdAM-GNCS under Progetti di Ricerca 2018. Università degli Studi di Firenze covered the cost to publish in open access.

Acknowledgments: The author wish to thank the referees for their helpful comments and suggestions.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Bellavia, S.; De Simone, V.; di Serafino, D.; Morini, B. Efficient preconditioner updates for shifted linear systems. *SIAM J. Sci. Comput.* **2011**, *33*, 1785–1809. [CrossRef]
2. Bellavia, S.; De Simone, V.; di Serafino, D.; Morini, B. A preconditioning framework for sequences of diagonally modified linear systems arising in optimization. *SIAM J. Numer. Anal.* **2012**, *50*, 3280–3302. [CrossRef]
3. Bellavia, S.; Gondzio, J.; Porcelli, M. An inexact dual logarithmic barrier method for solving sparse semidefinite programs. *Math. Program.* **2018**, doi:10.1007/s10107-018-1281-5. [CrossRef]
4. Björck, A. *Numerical Methods for Least Squares Problems*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1996; ISBN 978-0-89871-360-2.
5. Málek, J.; Strakoš, Z. *Preconditioning and the Conjugate Gradient Method in the Context of Solving PDEs, SIAM Spotlights*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2014; ISBN 978-1-611973-83-9.

6. Zhang, F. *The Schur Complement and Its Applications*; Series: Numerical Methods and Algorithms; Claude Brezinski; Springer: New York, NY, USA, 2005; ISBN 978-0-387-24273-6.
7. Benzi, M. Preconditioning techniques for large linear systems: A survey. *J. Comput. Phys.* **2002**, *2*, 418–477. [CrossRef]
8. Saad, Y. Iterative Method for Sparse Linear System. *Other Titles in Applied Mathematics*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2003; ISBN 978-0-89871-534-7.
9. Nocedal, J.; Wright, S.J. *Numerical Optimization*; Springer Series in Operations Research; Springer: Berlin, Germany, 1999; ISBN 978-0-387-40065-5.
10. Morales, J.L.; Nocedal, J. Automatic preconditioning by limited memory Quasi-Newton updating. *SIAM J. Optim.* **2000**, *10*, 1079–1096. [CrossRef]
11. Bellavia, S.; De Simone, V.; di Serafino, D.; Morini, B. Updating constraint preconditioners for KKT systems in quadratic programming via low-rank corrections. *SIAM J. Optim.* **2015**, *25*, 1787–1808. [CrossRef]
12. Morini, B.; Simoncini, V.; Tani, M. A comparison of reduced and unreduced KKT systems arising from interior point methods. *Comput. Optim. Appl.* **2017**, *68*, 1–27. [CrossRef]
13. Bellavia, S.; Gondzio, J.; Morini, B. A matrix-free preconditioner for sparse symmetric positive definite systems and least-squares problems. *SIAM J. Sci. Comput.* **2013**, *35*, A192–A211. [CrossRef]
14. Gondzio, J. Matrix-free interior point Method. *Comput. Optim. Appl.* **2012**, *51*, 457–480, doi:10.1007/s10589-010-9361-3. [CrossRef]
15. Fountoulakis, K.; Gondzio, J. A second-order method for strongly convex L1-regularization problems. *Math. Prog. A* **2016**, *156*, 189–219, doi:0.1007/s10107-015-0875-4. [CrossRef]
16. Bergamaschi, L.; De Simone, V.; di Serafino, A.; Martínez, D. BFGS-like updates of constraint preconditioners for sequences of KKT linear systems in quadratic programming. *Numer. Linear Algebra Appl.* **2018**, doi:10.1002/nla.2144. [CrossRef]
17. Caliciotti, A.; Fasano, G.; Roma, M. Preconditioned Nonlinear Conjugate Gradient methods based on a modified secant equation. *Appl. Math. Comput.* **2018**, *318*, 196–214. [CrossRef]
18. Fasano, G.; Roma, M. A novel class of approximate inverse preconditioners for large positive definite linear systems in optimization. *Comput. Optim. Appl.* **2016**, *65*, 399–429. [CrossRef]
19. Gratton, S.; Sartenaer, A.; Tshimanga, J. On a class of limited memory preconditioners for large scale linear systems with multiple right-hand sides. *SIAM J. Optim.* **2011**, *21*, 912–935. [CrossRef]
20. Greenbaum, A. *Iterative Methods for Solving Linear Systems*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1996; ISBN 978-0-89871-396-X.
21. Liesen, J.; Strakoš, Z. *Krylov Subspace Methods, Principles and Analysis*; Oxford University Press: Oxford, UK, 2012; ISBN 978-0-19-965541-0.
22. Saad, Y.; Yeung, M.; Erhel, J.; Guyomarc'h, F. A deflated version of the conjugate gradient method. *SIAM J. Sci. Comput.* **2000**, *21*, 1909–1926. [CrossRef]
23. Davis, T.; Hu, Y. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw.* **2011**, *38*, 1–25. [CrossRef]
24. Golub, G.H.; van Loan, C.F. *Matrix Computations*, 3rd ed.; The John Hopkins University Press: Baltimore, MD, USA, 1996; ISBN 0-8018-5414-8.