A preconditioning framework for sequences of diagonally modified linear systems arising in optimization

Stefania Bellavia[1], Valentina De Simone[2], Daniela di Serafino[2,3], Benedetta Morini[1]

Dipartimento di Energetica “Sergio Stecco”,
Università di Firenze, Pubblicazione n. 2/2011.

[1] Dipartimento di Energetica “S. Stecco”,
Università di Firenze,
Viale G.B. Morgagni 40/44, 50134 Firenze, Italy,
Email: stefania.bellavia@unifi.it, benedetta.morini@unifi.it.

[2] Dipartimento di Matematica,
Seconda Università degli Studi di Napoli,
Viale A. Lincoln 5, 81100 Caserta, Italy,
e-mail: valentina.desimone@unina2.it, daniela.diserafino@unina2.it.

[3] Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR), CNR,
Via P. Castellino 111, 80131 Napoli, Italy

A PRECONDITIONING FRAMEWORK FOR SEQUENCES OF DIAGONALLY MODIFIED LINEAR SYSTEMS ARISING IN OPTIMIZATION[*]

STEFANIA BELLAVIA[†], VALENTINA DE SIMONE[‡], DANIELA DI SERAFINO[§], AND BENEDETTA MORINI[†]

Abstract. We propose a framework for building preconditioners for sequences of linear systems of the form $(A+\Delta_k)x_k = b_k$, where A is symmetric positive semidefinite and Δ_k is diagonal positive semidefinite. Such sequences arise in several optimization methods, e.g., in affine-scaling methods for bound-constrained convex quadratic programming and bound-constrained linear least squares, as well as in trust-region and overestimation methods for convex unconstrained optimization problems and nonlinear least squares. For all the matrices of a sequence, the preconditioners are obtained by updating any preconditioner for A available in the LDL^T form. The preconditioners in the framework satisfy the natural requirement of being effective on slowly varying sequences; furthermore, under an additional property they are also able to cluster eigenvalues of the preconditioned matrix when some entries of Δ_k are sufficiently large. We present two low-cost preconditioners sharing the above-mentioned properties and evaluate them on sequences of linear systems generated by the reflective Newton method applied to bound-constrained convex quadratic programming problems, and on sequences arising in solving nonlinear least-squares problems with the Regularized Euclidean Residual method. The results of the numerical experiments show the effectiveness of these preconditioners.

Key words. Sequences of linear systems, preconditioning, incomplete LDL^T factorization, large-scale convex optimization.

AMS subject classifications. 65F08, 65F50, 90C20, 90C25.

1. Introduction.

We are interested in the efficient solution of sequences of diagonally modified linear systems by preconditioned Krylov methods. The sequences considered here have the form

$$(A + \Delta_k)x_k = b_k, \quad k = 0, 1, \ldots, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite and sparse, and $\Delta_k \in \mathbb{R}^{n\times n}$ is diagonal positive semidefinite. We assume that the systems are compatible and address the problem of preconditioning each of them by updating a preconditioner available for A, called seed preconditioner. The seed preconditioner is assumed to be a symmetric positive definite (SPD) matrix, although A may be semidefinite, and to be factorized in the LDL^T form. The goal is to perform low-cost updates of its factors to build an effective preconditioner for each matrix of the sequence.

There are several possible applications of these preconditioner updating techniques. We focus on optimization, where notable classes of algorithms require the solution of sequences of the form (1.1) to compute the step between two subsequent iterates. When it is too expensive or even not feasible to factorize the matrices, the systems are solved by Conjugate Gradient (CG) type methods and the availability

[*]Work partially supported by INdAM-GNCS under grant *Progetti 2011 (Advanced Numerical Methods for Large-Scale Constrained Nonlinear Optimization Problems)*

[†]Dipartimento di Energetica "S. Stecco", Università degli Studi di Firenze, viale G. B. Morgagni 40, 50134 Firenze, Italy, (stefania.bellavia@unifi.it, benedetta.morini@unifi.it).

[‡]Dipartimento di Matematica, Seconda Università degli Studi di Napoli, viale A. Lincoln 5, 81100 Caserta, Italy (valentina.desimone@unina2.it).

[§]Dipartimento di Matematica, Seconda Università degli Studi di Napoli, viale A. Lincoln 5, 81100 Caserta, Italy, and Istituto di Calcolo e Reti ad Alte Prestazioni, CNR, via P. Castellino 111, 80131 Napoli, Italy (daniela.diserafino@unina2.it).

of efficient preconditioners is a key issue for the overall efficiency of the optimization methods. Next, we briefly outline optimization methods that call for the solution of the sequences under consideration. Some more details are given in section 4.

Sequences of the form (1.1) arise, e.g., in affine-scaling interior-point methods for bound-constrained convex quadratic programming [4, 10]. In this case, the application of Newton-like methods to first-order optimality conditions gives rise to a system of the form shown in (1.1) at each iteration. Here A is the Hessian of the objective function, scaled on both sides by a suitable diagonal matrix, and Δ_k depends on the scaling matrix and possibly on regularization terms that mitigate the ill-conditioning of the systems [3]. The matrix Δ_k can be either positive definite or semidefinite.

Further examples are provided by trust-region [9, 22] and overestimation methods [1, 8, 18] for convex optimization problems. In these settings, at each iteration the step minimizes a regularized local model of the objective function. In trust-region methods, a sequence of systems of the form (1.1) has to be solved if the minimizer lies on the trust-region boundary and an approximation beyond the Steihaug-Toint point is required [9]. The sequence may stem from the application of a root-finding method to the so-called secular equation or by an interior-point sequential subspace minimization that solves the inequality constrained trust-region problem over a sequence of evolving low-dimensional subspaces [13, 14]. In these cases $\Delta_k = \alpha_k G$, where α_k is a nonnegative scalar and G is a diagonal positive definite matrix specifying an elliptical shape for the trust region. In overestimation methods [1, 8, 18], sequences of systems of the form (1.1) derive from the solution of the secular equation and $\Delta_k = \alpha_k I$ for a positive α_k. We note that an efficient preconditioning strategy for the matrices $A + \Delta_k$ provides a viable alternative to procedures that find an approximate step over a sequence of expanding subspaces associated with the iterative linear system solver and then recover the solution in $\mathbb{R}^n$, as proposed in [7, 16].

Preconditioner updating techniques are discussed in [2, 5, 6, 21] for sequences of SPD systems differing by a diagonal matrix; in particular, $\Delta_k = \alpha_k I$, with $\alpha_k > 0$, is considered in [2, 5, 21]. The procedures in [5, 6] are based on the use of an approximate inverse seed preconditioner, i.e., an incomplete LDL^T factorization of A^{-1}, while those in [2, 21] perform updates of an incomplete LDL^T factorization of A. All of them are effective alternatives to reusing the seed preconditioner over subsequent systems ("frozen" preconditioner) or to recomputing the preconditioner for each $A + \Delta_k$. A remarkable aspect of the preconditioner updating techniques proposed in the literature is that they work well in spite of their low cost. The frozen preconditioner may yield convergence slowdown, while the recomputed one may be too expensive and pointlessly accurate; in practice, updated preconditioners are between the frozen and the recomputed ones in terms of solver iterations, but are often the fastest ones in terms of computational time.

In this paper we propose a framework for building preconditioners for the sequences under consideration and provide an analysis of their quality in terms of the size of the entries of Δ_k. Since a natural requirement is that the preconditioners work well on slowly varying sequences, we first show their effectiveness when the matrices $A + \Delta_k$ undergo small changes. Then we discuss their ability to cluster eigenvalues of the preconditioned matrix when some entries of Δ_k are sufficiently large. Finally, we present two specific preconditioners in the proposed framework, which can be constructed at a low cost. The efficiency of these preconditioners is tested on two sets of problems. One set consists of sequences of systems arising in the application of the reflective Newton method [10], as implemented in the `quadprog` function of the Matlab

Optimization Toolbox; in this case, the preconditioners are embedded into `quadprog` and the tests are carried out on bound-constrained convex quadratic programming problems from the CUTEr collection [17]. The other set is made of sequences generated in the application of the Regularized Euclidean Residual overestimation method [1] to nonlinear least-squares problems.

The paper is organized as follows. In section 2 we describe our framework and provide a theoretical analysis of the relevant preconditioners. In section 3 we present two preconditioners belonging to the framework, which can be built at a low computational cost. Section 4 is devoted to the numerical experiments and section 5 provides some concluding remarks.

1.1. Notations.

In the following, $\|\cdot\|$ denotes the vector or matrix 2-norm. For any matrix M, either $m_{i,j}$ or $(M)_{i,j}$ denotes the (i,j)th entry of M; furthermore, for vectors and matrices having a subscript, such as v_k and M_k, we use v_i^k and $m_{i,j}^k$ to represent the ith entry of v_k and the (i,j)th entry of M_k, respectively. If $v = (v_1, \ldots, v_n)^T$, then $diag(v)$, as well as $diag(v_1, \ldots, v_n)$, is the $n \times n$ diagonal matrix having the entries of v on the main diagonal. If M is a $n \times n$ matrix, $diag(M)$ is the $n \times n$ diagonal matrix with the same diagonal entries as M, while $\mathit{off}(M) = M - diag(M)$, i.e., $\mathit{off}(M)$ is the matrix consisting of the off-diagonal part of M. For any matrix M, $nnz(M)$ is the number of nonzero entries of M. If M is symmetric, $\lambda_i(M)$ denotes the ith eigenvalue of M (with the eigenvalues sorted in any order), while $\lambda_{min}(M)$ and $\lambda_{max}(M)$ denote its minimum and maximum eigenvalues. Finally, the sequence of matrices to be preconditioned is indicated by $\{A + \Delta_k\}$ and the corresponding sequence of preconditioners by $\{P_k\}$.

2. A framework for updating preconditioners.

In this section we define a framework for the construction and analysis of a preconditioner P_k for each matrix $A + \Delta_k$, where $\Delta_k = diag(\delta_1^k, \ldots, \delta_n^k)$ and $\delta_i^k \geq 0$, $i = 1, \ldots, n$. The preconditioner P_k is obtained by updating an SPD preconditioner available for A in the factorized LDL^T form, where L is unit lower triangular and $D = diag(d_1, \ldots, d_n)$ with $d_i > 0$, $i = 1, \ldots, n$.

Definition 2.1. *A sequence of preconditioners $\{P_k\}$ for $\{A+\Delta_k\}$ belongs to the class UFP(A, Δ_k) (Updated Factorization Preconditioners for $\{A + \Delta_k\}$) if*

$$P_k = L_k D_k L_k^T, \tag{2.1}$$

with L_k lower triangular and $D_k = diag(d_1^k, \ldots, d_n^k)$ such that

(A.1) $d_i^k \geq d_i + \delta_i^k$, $i = 1, \ldots, n$;

(A.2) *there exists $\sigma > 0$ independent of k such that $\|D_k - D\| \leq \sigma \|\Delta_k\|$;*

(A.3) $diag(L_k) = diag(L) = (1, 1, ..., 1)^T$ *and* $\mathit{off}(L_k) = \mathit{off}(L) S_k$, *where* $S_k = DD_k^{-1}$.

The previous definition is motivated by the desire of having preconditioners that can be applied at the same cost as the seed preconditioner, and mimic in some sense the behaviour of the corresponding matrices $A + \Delta_k$. The first objective is pursued by keeping fixed the sparsity pattern of the preconditioner and by computing its triangular factors with the simple diagonal scaling in (A.3). The second objective is pursued by imposing that the distance of D_k from D behaves like $\|\Delta_k\|$ (see (A.2)), the entries of D_k increase at least at the same rate as the entries of $D + \Delta_k$ (see (A.1)), and the off-diagonal part of L_k decreases when the entries of Δ_k increase (see (A.3)), i.e., when the diagonal of $A + \Delta_k$ dominates over the remaining entries.

By definition, P_k is SPD and $S_k = diag\left(s_1^k, \dots, s_n^k\right)$ is such that $s_i^k \in (0,1]$, $i = 1, \dots, n$, i.e., $\|S_k\| \leq 1$. It is worth noting that, by using this property of S_k, we can show that the conditioning of the matrices L_k is at least as good as the conditioning of L. Specifically, for any unit lower triangular matrix T and its inverse, we have

$$\|T\| \leq \sqrt{n + \frac{n(n-1)}{2} M_T^2}, \quad \|T^{-1}\| \leq \frac{\sqrt{(M_T+1)^{2n} + 2n(M_T+2) - 1}}{M_T + 2}, \tag{2.2}$$

where $M_T = \max_{j<i}\{|t_{i,j}|\}$ [20]. These bounds provide an upper bound on the condition number of T, which increases if M_T increases. Since $s_i^k \in (0,1]$, it is $\max_{j<i}\{|l_{i,j}^k|\} \leq \max_{j<i}\{|l_{i,j}|\}$; hence the bound on the condition number of L_k is lower than or equal to the bound on the condition number of L. Finally, we observe that the computational cost of building P_k is at least equal to the cost of scaling L to build L_k, which is $O(nnz(L))$; then, the choice of D_k has a crucial role in determining the overall cost of the preconditioner. As shown in sections 3 and 4, the choice of D_k is important to achieve a good tradeoff between the whole computational cost and the effectiveness of preconditioner.

2.1. Analysis of the preconditioners. To analyze the properties of the preconditioners in the class $UFP(A, \Delta_k)$, we first assume that A is SPD and the seed preconditioner is $LDL^T = A$. Then, in section 2.2, we relax this assumption and discuss the extension of the properties to the case where the seed preconditioner is an incomplete factorization of A and to the case where A is positive semidefinite.

We start providing an expression of the diagonal and off-diagonal parts of $A + \Delta_k - P_k$, as well as upper bounds on their norms, that will be useful next.

LEMMA 2.2. *Let $\{P_k\} \in UFP(A, \Delta_k)$. Then,*

$$diag(A + \Delta_k - P_k) = D + \Delta_k - D_k + diag(off(L) S_k (D_k - D) off(L)^T), \tag{2.3}$$
$$off(A + \Delta_k - P_k) = off(off(L) S_k (D_k - D) off(L)^T). \tag{2.4}$$

Proof. The diagonal entries of $A + \Delta_k - P_k$ have the following expression:

$$\begin{aligned}(A + \Delta_k - P_k)_{i,i} &= a_{i,i} + \delta_i^k - (L_k D_k L_k^T)_{i,i} \\ &= a_{i,i} + \delta_i^k - d_i^k - \sum_{r=1}^{i-1} (l_{i,r}^k)^2 d_r^k.\end{aligned} \tag{2.5}$$

Hence, by using $a_{i,i} = d_i + \sum_{r=1}^{i-1} (l_{i,r})^2 d_r$ and (A.3) we get

$$(A + \Delta_k - P)_{i,i} = d_i + \delta_i^k - d_i^k + \sum_{r=1}^{i-1} (l_{i,r})^2 \left(d_r - (s_r^k)^2 d_r^k\right). \tag{2.6}$$

Furthermore, by the definition of S in (A.3),

$$D - S_k^2 D_k = S_k(D_k - D),$$

and equality (2.3) easily follows.

Concerning the off-diagonal entries of $A + \Delta_k - P_k$, without loss of generality we consider $i > j$. From (A.3) it follows that

$$(A + \Delta_k - P_k)_{i,j} = (LDL^T)_{i,j} - (L_k D_k L_k^T)_{i,j}$$

$$= \sum_{r=1}^{j} \left(l_{i,r} l_{j,r} d_r - l_{i,r}^k l_{j,r}^k d_r^k \right)$$
$$= \sum_{r=1}^{j-1} l_{i,r} l_{j,r} (d_r - (s_r^k)^2 d_r^k) + l_{i,j}(d_j - s_j^k d_j^k),$$

which yields (2.4) since $d_j - s_j^k d_j^k = 0$. □

LEMMA 2.3. *Let D, D_k and S_k be the matrices in Definition 2.1. Then*

$$\begin{aligned} \|diag(off(L) S_k (D_k - D) off(L)^T)\| &\le \|off(L)\|^2 \|S_k(D_k - D)\| \\ &\le \|off(L)\|^2 \|D\|, \end{aligned} \tag{2.7}$$

$$\begin{aligned} \|off(off(L) S_k (D_k - D) off(L)^T)\| &\le \|off(L)\|^2 \|S_k(D_k - D)\| \\ &\le \|off(L)\|^2 \|D\|. \end{aligned} \tag{2.8}$$

Proof. The first inequality in (2.7) follows from the inequality $\max_{i,j} |b_{i,j}| \le \|B\|$, which holds for any matrix B. By observing that

$$\|S_k(D_k - D)\| = \max_i \frac{d_i^k - d_i}{d_i^k} d_i \quad \text{and} \quad 0 \le \frac{d_i^k - d_i}{d_i^k} < 1,$$

we get

$$\|S_k(D_k - D)\| \le \|D\|, \tag{2.9}$$

and the second inequality in (2.7) holds. Furthermore, since $S_k(D_k - D)$ is positive semidefinite by construction, $off(L) S_k (D_k - D) off(L)^T$ is positive semidefinite; hence, by [2, Lemma 4.2], it is

$$\|off(off(L) S_k (D_k - D) off(L)^T)\| \le \|off(L) S_k (D_k - D) off(L)^T\|,$$

from which the first inequality in (2.8) trivially follows. The second inequality in (2.8) is obtained by using (2.9). □

By exploiting the previous lemmas, we show that P_k provides a good approximation of $A + \Delta_k$ when $\|\Delta_k\|$ is sufficiently small, and hence the sequence $\{P_k\}$ can be effective on slowly varying sequences.

THEOREM 2.1. *Let $\{P_k\} \in UFP(A, \Delta_k)$. Then, there exists $\zeta > 0$ independent of k such that*

$$\|A + \Delta_k - P_k\| \le \zeta \|\Delta_k\|. \tag{2.10}$$

Proof. Using Lemmas 2.2 and 2.3 we have

$$\|A + \Delta_k - P_k\| \le \|D + \Delta_k - D_k\| + 2\|off(L)\|^2 \|S_k(D_k - D)\|.$$

Then, from (A.2) and $\|S_k\| \le 1$, we get

$$\|A + \Delta_k - P\| \le (1 + \sigma)\|\Delta_k\| + 2\sigma \|off(L)\|^2 \|\Delta_k\|.$$

Thus (2.10) holds by letting $\zeta = 1 + \sigma + 2\sigma\|off(L)\|^2$. □

Clearly, $P_k^{-1}(A + \Delta_k)$ has real positive eigenvalues, since it is similar to an SPD matrix. The next theorem shows that these eigenvalues are clustered around 1 as $\|\Delta_k\|$ gets smaller.

THEOREM 2.2. *Let $\{P_k\} \in UFP(A, \Delta_k)$. For all $\varepsilon > 0$ there exists $\eta > 0$ independent of k such that, if $\|\Delta_k\| < \eta$, then*

$$|\lambda_i \left(P_k^{-1}(A + \Delta_k)\right) - 1| < \varepsilon, \quad i = 1, \dots, n.$$

Furthermore, if $A + \Delta_k - P_k$ has rank $n - l$, then l eigenvalues of $P_k^{-1}(A + \Delta_k)$ are equal to 1.

Proof. We note that

$$P_k^{-1}(A+\Delta_k) = (A+\Delta_k+(P_k-A-\Delta_k))^{-1}(A+\Delta_k) = (I+(A+\Delta_k)^{-1}(P_k-A-\Delta_k))^{-1}.$$

Then, if λ is an eigenvalue of $P_k^{-1}(A + \Delta_k)$, we have

$$v = \lambda(I + (A + \Delta_k)^{-1}(P_k - A - \Delta_k))v, \tag{2.11}$$

where v is an eigenvector corresponding to λ. Without loss of generality, we assume $\|v\| = 1$. From (2.11) it follows that $\lambda = 1$ if and only if $(A+\Delta_k)^{-1}(P_k-A-\Delta_k)v = 0$, i.e., v belongs to the null space of $P_k - A - \Delta_k$. So, if $rank(P_k - A - \Delta_k) = n - l$, then $P_k^{-1}(A + \Delta_k)$ has at least l unit eigenvalues.

Now suppose that $(A + \Delta_k)^{-1}(P_k - A - \Delta_k)v \neq 0$ and multiply (2.11) by v^T on the left, obtaining

$$1 = \lambda(1 + v^T(A + \Delta_k)^{-1}(P_k - A - \Delta_k)v), \tag{2.12}$$

and hence

$$|\lambda - 1| = \left| \frac{v^T(A + \Delta_k)^{-1}(P_k - A - \Delta_k)v}{1 + v^T(A + \Delta_k)^{-1}(P_k - A - \Delta_k)v} \right|, \tag{2.13}$$

where $1 + v^T(A + \Delta_k)^{-1}(P_k - A - \Delta_k)v \neq 0$ by (2.12). By [15, Theorem 8.1.5] we have

$$\lambda_{min}(A + \Delta_k) \geq \lambda_{min}(A) + \lambda_{min}(\Delta_k) \geq \lambda_{min}(A),$$

which implies

$$\|(A + \Delta_k)^{-1}\| \leq \|A^{-1}\|.$$

By Theorem 2.1, if $\zeta\|A^{-1}\| \, \|\Delta_k\| < 1$, then

$$\begin{aligned} |\lambda - 1| &\leq \frac{\|(A + \Delta_k)^{-1}\| \, \|P_k - A - \Delta_k\|}{1 - \|(A + \Delta_k)^{-1}\| \, \|P_k - A - \Delta_k\|}, \\ &\leq \frac{\zeta\|A^{-1}\| \, \|\Delta_k\|}{1 - \zeta\|A^{-1}\| \, \|\Delta_k\|}. \end{aligned} \tag{2.14}$$

The thesis follows by taking

$$\eta \leq \min\left\{ \frac{1}{\zeta\|A^{-1}\|}, \frac{\varepsilon}{\zeta(1 + \varepsilon)\|A^{-1}\|} \right\}.$$

□

Now we are interested in understanding the behaviour of P_k when $\|\Delta_k\|$ is large. By requiring that a sequence $\{P_k\}$ belonging to $UFP(A, \Delta_k)$ satisfies the following additional property:

(A.4) *there exists $\tau > 0$ independent of k such that $\|D + \Delta_k - D_k\| < \tau$,*

we can show that the effectiveness of P_k increases as $\|\Delta_k\|$ increases, as stated by the following theorem.

THEOREM 2.3. *Let $\{P_k\} \in UFP(A, \Delta_k)$ satisfy (A.4). For all $\varepsilon > 0$ there exists $\vartheta > 0$ independent of k such that, if $\|\Delta_k\| > \vartheta$, then*

$$\frac{\|A + \Delta_k - P_k\|}{\|A + \Delta_k\|} < \varepsilon. \tag{2.15}$$

Proof. We first show that $\|A + \Delta_k - P\|$ is bounded above independently of k. From Lemmas 2.2 and 2.3 it follows that

$$\|A + \Delta_k - P_k\| \leq \|D + \Delta_k - D_k\| + 2\|\mathit{off}(L)\|^2\|D\|;$$

thus, by using (A.4), we get

$$\|A + \Delta_k - P_k\| \leq \tau + 2\|\mathit{off}(L)\|^2\|D\|.$$

Furthermore, if $\|\Delta_k\| > \|A\|$ then $\|A + \Delta_k\| \geq \|\Delta_k\| - \|A\| > 0$, and inequality (2.15) follows by taking

$$\vartheta \geq \max\left\{\|A\|, \frac{\varepsilon\|A\| + \tau + 2\|\mathit{off}(L)\|^2\|D\|}{\varepsilon}\right\}.$$

□

REMARK 2.1. *We observe that properties (A.1)–(A.3) do not ensure that $\|A + \Delta_k - P_k\|$ is bounded independently of k if $\|\Delta_k\|$ increases, so the previous theorem does not hold without assumption (A.4). For example, by considering $\Delta_k = D + 2\Delta_k$ and defining L_k as in (A.3), we have a sequence $\{P_k\} \in UFP(A, \Delta_k)$, but*

$$\|D + \Delta_k - D_k\| = \|\Delta_k\|,$$

which cannot be bounded independently of Δ_k.

The following theorem shows that the preconditioners satisfying (A.1)–(A.4) are effective in clustering the eigenvalues when the entries of Δ_k are large.

THEOREM 2.4. *Let $\{P_k\} \in UFP(A, \Delta_k)$ be such that (A.4) holds. There exists $\xi > 0$ independent of k such that, if q entries of Δ_k satisfy $\delta_i^k > \xi$, then q eigenvalues, $\lambda_{j_1}, \dots, \lambda_{j_q}$, of $P_k^{-1}(A + \Delta_k)$ satisfy*

$$|\lambda_j\left(P_k^{-1}(A + \Delta_k)\right) - 1| < \varepsilon.$$

Proof. Without loss of generality, we assume that $\delta_1^k, \dots, \delta_q^k$ are the q largest entries of Δ_k. Then we partition A, Δ_k, L_k and D_k into blocks as follows:

$$A = \begin{pmatrix} A_{1,1} & A_{2,1}^T \\ A_{2,1} & A_{2,2} \end{pmatrix}, \quad \Delta_k = \begin{pmatrix} \Delta_1^k & 0 \\ 0 & \Delta_2^k \end{pmatrix}, \tag{2.16}$$

$$L_k = \begin{pmatrix} L_{1,1}^k & 0 \\ L_{2,1}^k & L_{2,2}^k \end{pmatrix}, \quad D_k = \begin{pmatrix} D_1^k & 0 \\ 0 & D_2^k \end{pmatrix}, \tag{2.17}$$

where $A_{1,1}, D_1^k, L_{1,1}^k, \Delta_1^k \in \mathbb{R}^{q\times q}$, $A_{2,2}, D_2^k, L_{2,2}^k, \Delta_2^k \in \mathbb{R}^{(n-q)\times(n-q)}$ and $A_{2,1}, L_{2,1}^k \in \mathbb{R}^{(n-q)\times q}$. Now we consider $(D_k)^{-\frac{1}{2}}(L_k)^{-1}(A+\Delta_k)(L_k)^{-T}(D_k)^{-\frac{1}{2}}$, which is similar to $P_k^{-1}(A+\Delta_k)$. It is easy to see that

$$(D_k)^{-\frac{1}{2}}(L_k)^{-1}(A+\Delta_k)(L_k)^{-T}(D_k)^{-\frac{1}{2}} = \begin{pmatrix} B_{1,1} & B_{2,1}^T \\ B_{2,1} & B_{2,2} \end{pmatrix},$$

where

$$B_{1,1} = \left(D_1^k\right)^{-\frac{1}{2}} \left(L_{1,1}^k\right)^{-1} \left(A_{1,1}+\Delta_1^k\right) \left(L_{1,1}^k\right)^{-T} \left(D_1^k\right)^{-\frac{1}{2}}, \tag{2.18}$$

and

$$\begin{aligned} B_{2,1} = -\left(D_2^k\right)^{-\frac{1}{2}} \left(L_{2,2}^k\right)^{-1} \Big(L_{2,1}^k \left(L_{1,1}^k\right)^{-1} \left(A_{1,1}+\Delta_1^k\right) + A_{2,1}\Big) \\ \cdot \left(L_{1,1}^k\right)^{-T} \left(D_1^k\right)^{-\frac{1}{2}}, \end{aligned} \tag{2.19}$$

(we neglect, for simplicity, the superscript k in $B_{i,j}$ and do not provide any expression for $B_{2,2}$ because it is not needed next).

In the following we show that

$$\begin{pmatrix} B_{1,1} & B_{2,1}^T \\ B_{2,1} & B_{2,2} \end{pmatrix} = X + Y, \tag{2.20}$$

where

$$X = \begin{pmatrix} I_q & 0 \\ 0 & B_{2,2} \end{pmatrix}, \quad Y = \begin{pmatrix} N & B_{2,1}^T \\ B_{2,1} & 0 \end{pmatrix},$$

and $\|Y\|$ vanishes as d_i^k, $i = 1,\dots,q$, increases.

We first consider the block $B_{1,1}$. From (A.3) it follows that

$$\left(L_{1,1}^k\right)^{-1} = \Big(I_q + \mathit{off}(L_{1,1})D_1\left(D_1^k\right)^{-1}\Big)^{-1},$$

where $L_{1,1}$ and D_1 are the (1,1) blocks of the matrices L and D partitioned as in (2.16)–(2.17). Then, letting

$$\begin{aligned} Z &= \left(D_1^k\right)^{-\frac{1}{2}} \Big(I_q + \mathit{off}(L_{1,1})D_1\left(D_1^k\right)^{-1}\Big)^{-1} \left(D_1^k\right)^{\frac{1}{2}} - I_q \\ &= \Big(I_q + \left(D_1^k\right)^{-\frac{1}{2}} \mathit{off}(L_{1,1})D_1\left(D_1^k\right)^{-\frac{1}{2}}\Big)^{-1} - I_q, \end{aligned}$$

and

$$W = \left(D_1^k\right)^{-\frac{1}{2}} \left(A_{1,1}+\Delta_1^k\right) \left(D_1^k\right)^{-\frac{1}{2}},$$

$B_{1,1}$ can be written as

$$B_{1,1} = (I_q + Z)W(I_q+Z)^T.$$

Let us analyze Z and W. We observe that for sufficiently large values of d_i^k, $i = 1,\dots,q$, it is

$$\left\|\left(D_1^k\right)^{-\frac{1}{2}} \mathit{off}(L_{1,1})D_1\left(D_1^k\right)^{-\frac{1}{2}}\right\| \le \left\|\left(D_1^k\right)^{-1}\right\| \|\mathit{off}(L_{1,1})D_1\| < 1.$$

Then, by using the Banach Lemma [15, Lemma 2.3.3] and noting that

$$Z = -\left(I_q + \left(D_1^k\right)^{-\frac{1}{2}} \mathit{off}(L_{1,1})D_1 \left(D_1^k\right)^{-\frac{1}{2}}\right)^{-1} \left(D_1^k\right)^{-\frac{1}{2}} \mathit{off}(L_{1,1})D_1 \left(D_1^k\right)^{-\frac{1}{2}},$$

we get

$$\|Z\| \le \frac{\left\|\left(D_1^k\right)^{-\frac{1}{2}} \mathit{off}(L_{1,1})D_1 \left(D_1^k\right)^{-\frac{1}{2}}\right\|}{1-\left\|\left(D_1^k\right)^{-\frac{1}{2}} \mathit{off}(L_{1,1})D_1 \left(D_1^k\right)^{-\frac{1}{2}}\right\|}, \tag{2.21}$$

hence $\|Z\|$ vanishes when the values of d_i^k, $i = 1, \dots, q$, increase. Furthermore, the matrix W can be written as

$$W = I_q + \mathit{diag}(W - I_q) + \mathit{off}(W),$$

where

$$\begin{aligned} w_{i,i} &= \frac{a_{i,i} + \delta_i^k}{d_i^k}, \quad i = 1, \dots, q, \\ w_{i,j} &= \frac{a_{i,j}}{\left(d_i^k d_j^k\right)^{\frac{1}{2}}}, \quad i, j = 1, \dots, q, \ i \ne j. \end{aligned}$$

Since

$$\left|\frac{a_{i,i} + \delta_i^k}{d_i^k} - 1\right| \le \frac{\left|\sum_{r=1}^{i-1}(l_{i,r})^2 d_r\right| + \left|d_i + \delta_i^k - d_i^k\right|}{\left|d_i^k\right|},$$

and, by hypothesis, $\left|d_i + \delta_i^k - d_i^k\right| < \tau$, we have that $\|\mathit{diag}(W - I_q)\|$ and $\|\mathit{off}(W)\|$ vanish as d_i^k, $i = 1, \dots, q$, increases.

Thus, $B_{1,1}$ in (2.18) has the following form:

$$B_{1,1} = (I_q + Z)\left(I_q + \mathit{diag}(W - I_q) + \mathit{off}(W)\right)(I_q + Z)^T = I_q + N,$$

where N is a matrix such that $\|N\|$ reduces towards zero as d_i^k, $i = 1, \dots, q$, increases. Clearly, N is symmetric because $B_{1,1}$ is symmetric.

Finally, we consider $B_{2,1}$ in (2.19), which can be written as

$$B_{2,1} = -\left(D_2^k\right)^{-\frac{1}{2}} \left(L_{2,2}^k\right)^{-1} L_{2,1}^k \left(D_1^k\right)^{\frac{1}{2}} B_{1,1} - \left(D_2^k\right)^{-\frac{1}{2}} \left(L_{2,2}^k\right)^{-1} A_{2,1} \left(L_{1,1}^k\right)^{-T} \left(D_1^k\right)^{-\frac{1}{2}}.$$

Property (A.3) implies

$$\max_{j<i}\{|(L_{1,1}^k)_{i,j}|\} \le \max_{j<i}\{|(L_{1,1})_{i,j}|\}, \qquad \max_{j<i}\{|(L_{2,2}^k)_{i,j}|\} \le \max_{j<i}\{|(L_{2,2})_{i,j}|\},$$

and, by (2.2), we have that $\|(L_{1,1}^k)^{-1}\|$ and $\|(L_{2,2}^k)^{-1}\|$ are bounded. Furthermore, $\|(D_2^k)^{-\frac{1}{2}}\| \le \|D_2^{-\frac{1}{2}}\|$ and, by (A.3),

$$L_{2,1}^k \left(D_1^k\right)^{\frac{1}{2}} = L_{2,1} D_1 \left(D_1^k\right)^{-\frac{1}{2}}.$$

Then, we can conclude that $\|B_{2,1}\|$ vanishes as the values of d_i^k, $i = 1, \dots, q$, increase.

By applying [15, Theorem 8.1.5] to the matrix in (2.20), it follows that

$$\lambda_i(X) + \lambda_{min}(Y) \leq \lambda_i \left(D_k^{-\frac{1}{2}} L_k^{-1}(A+\Delta_k) L_k^{-T} D_k^{-\frac{1}{2}} \right) \leq \lambda_i(X) + \lambda_{max}(Y).$$

Taking into account that X has q eigenvalues equal to 1, $\lambda_{min}(Y)$ and $\lambda_{max}(Y)$ vanish as the values of d_i^k, $i = 1, \dots, q$, increase, and, by (A.1), $d_i^k \geq \delta_i^k$, we can conclude that for all $\varepsilon > 0$ there exists $\xi > 0$ such that, if $\delta_i^k > \xi$ for $i = 1, \dots, q$, then q eigenvalues $\lambda_{j_1}, \dots, \lambda_{j_q}$, of $D_k^{-\frac{1}{2}} L_k^{-1}(A+\Delta_k) L_k^{-T} D_k^{-\frac{1}{2}}$ satisfy

$$|\lambda_j - 1| < \varepsilon.$$

□

2.2. Incomplete factorizations and positive semidefinite matrices. Consider now the case where A is SPD and the seed preconditioner is an incomplete factorization of A. Thus, instead of $A = LDL^T$ we have

$$A = LDL^T + C,$$

where C is a nonzero matrix. It is easy to verify that (2.10) can be extended as

$$\|A + \Delta_k - P_k\| \leq \zeta \|\Delta_k\| + \|C\|,$$

and inequality (2.14) becomes

$$|\lambda - 1| \leq \frac{\|A^{-1}\|(\zeta\|\Delta_k\| + \|C\|)}{1 - \|A^{-1}\|(\zeta\|\Delta_k\| + \|C\|)},$$

provided that $\|A^{-1}\|(\zeta\|\Delta_k\| + \|C\|) < 1$. Therefore, when $\|\Delta_k\|$ is small the quality of the preconditioner P_k depends on the size of $\|C\|$ and the ability of clustering around 1 the eigenvalues is lost if $\|C\|$ is too large. On the other hand, when $\|\Delta_k\|$ is small we cannot expect that the preconditioner is effective on $A + \Delta_k$ if it is not on A.

The situation is different when $\|\Delta_k\|$ is large. Indeed, it is easy to verify that Theorems 2.3 and 2.4 still hold independently of the size of $\|C\|$. This is in agreement with the fact that, as all the diagonal entries of Δ_k increase, the matrix $A+\Delta_k$ becomes more and more diagonally dominant and P_k tends to a diagonal preconditioner with diagonal entries increasing as fast as the diagonal entries of Δ_k.

Finally, we discuss the case where A is positive semidefinite. One possible approach is to compute, as a seed preconditioner, an incomplete LDL^T factorization of the SPD matrix $A + \beta I$ for some small positive β. We assume that A has rank r and the eigenvalues of A are ordered as follows

$$0 = \lambda_1(A) = \lambda_2(A) = \cdots = \lambda_{n-r}(A) < \lambda_{n-r+1}(A) \leq \cdots \leq \lambda_n(A).$$

Note that, if the null space of A has a nontrivial intersection with the null space of Δ_k, then $A + \Delta_k$ is singular and $P_k^{-1}(A + \Delta_k)$ has $n - p$ null eigenvalues where p is the rank of $A + \Delta_k$ and $p \geq r$. In this case, as long as the system involving the matrix $A+\Delta_k$ is consistent and we solve it by using the CG method with zero starting guess, the null space of $A+\Delta_k$ never enters the iteration, i.e., the corresponding zero eigenvalues do not affect the convergence and the rate of convergence depends on the ratio between the largest and the smallest positive eigenvalue [19, 25].

Suppose first that $A+\beta I = LDL^T$. A straightforward extension of (2.10) holds:

$$\|A+\Delta_k - P_k\| \le \zeta\|\Delta_k\| + \beta. \tag{2.22}$$

Now let us analyse the eigenvalues of $P_k^{-1}(A+\Delta_k)$ when $\|\Delta_k\|$ is small. To this aim, we prove the following theorem.

THEOREM 2.5. *Let $A+\beta I = LDL^T$ and $\{P_k\} \in UFP(A,\Delta_k)$. For all $\varepsilon \in (0,\beta)$ there exists $\eta > 0$ independent of k such that, if $\|\Delta_k\| < \eta$, then*

$$\frac{-\beta}{\lambda_i(A)+\beta-\varepsilon} - \varepsilon < \lambda_i\left(P_k^{-1}(A+\Delta_k)\right) - 1 < \frac{-\beta}{\lambda_i(A)+\beta+\varepsilon} + \varepsilon, \tag{2.23}$$

for $i = 1,\dots,n$.

Proof. We observe that

$$P_k^{-1}(A+\Delta_k) = P_k^{-1}(A+\Delta_k+\beta I) - \beta P_k^{-1}.$$

Furthermore, since $P_k^{-1}(A+\Delta_k)$ and $P_k^{-1}(A+\Delta_k+\beta I)$ are similar to the symmetric matrices $P_k^{-1/2}(A+\Delta_k)P_k^{-1/2}$ and $P_k^{-1/2}(A+\Delta_k+\beta I)P_k^{-1/2}$, respectively, by using [15, Theorem 8.1.5] we have

$$\lambda_i\left(-\beta P_k^{-1}\right) + \lambda_{min}\left(P_k^{-1}(A+\Delta_k+\beta I)\right) \le \lambda_i\left(P_k^{-1}(A+\Delta_k)\right), \tag{2.24}$$

$$\lambda_i\left(P_k^{-1}(A+\Delta_k)\right) \le \lambda_i\left(-\beta P_k^{-1}\right) + \lambda_{max}\left(P_k^{-1}(A+\Delta_k+\beta I)\right), \tag{2.25}$$

$i = 1,\dots,n$. By Theorem 2.2 we have that, for all $\varepsilon > 0$, if $\|\Delta_k\|$ is sufficiently small then

$$\begin{aligned} |\lambda_{max}\left(P_k^{-1}(A+\Delta_k+\beta I)\right) - 1| &< \varepsilon, \\ |\lambda_{min}\left(P_k^{-1}(A+\Delta_k+\beta I)\right) - 1| &< \varepsilon. \end{aligned} \tag{2.26}$$

To provide a bound on $\lambda_i(-\beta P_k^{-1})$, we write P_k as

$$P_k = A + \beta I + (P_k - A - \beta I).$$

By following the reasoning in the proof of Theorem 2.1, we find that there exists a constant ζ independent of k such that

$$\|P_k - A - \beta I\| \le \zeta\|\Delta_k\|,$$

and hence

$$|\lambda_i(P_k - A - \beta I)| \le \zeta\|\Delta_k\|, \quad i = 1,\dots,n.$$

Let $\varepsilon < \beta$; if Δ_k is such that $\zeta\|\Delta_k\| < \varepsilon$, by using the previous inequality and [15, Theorem 8.1.5] we get

$$\lambda_i(A) + \beta - \varepsilon \le \lambda_i(P_k) \le \lambda_i(A) + \beta + \varepsilon, \quad i = 1,\dots,n; \tag{2.27}$$

then, from (2.24)–(2.27), we can conclude that there exists $\eta > 0$ such that if $\|\Delta_k\| < \eta$ then (2.23) holds. ▯

The previous theorem shows that, for sufficiently small values of ε and $\|\Delta_k\|$, the eigenvalues of $P_k^{-1}(A+\Delta_k)$ are clustered around 1 if the corresponding eigenvalues of A are nonzero and β is small. When $\lambda_i(A) = 0$, we have

$$0 \le \lambda_i\left(P_k^{-1}(A+\Delta_k)\right) < c + \varepsilon, \quad c \simeq 0,$$

if $\beta \gg \varepsilon$, and

$$0 \leq \lambda_i \left(P_k^{-1}(A+\Delta_k)\right) < c+\varepsilon, \quad c \simeq \frac{1}{2},$$

if $\beta \simeq \varepsilon$.

Concerning Theorems 2.3 and 2.4, we can easily prove that they also hold when $A+\beta I = LDL^T$. This is because, as the diagonal entries of Δ_k increase, the effect of the perturbation βI on the preconditioner becomes negligible.

Finally, we note that the results obtained with $A+\beta I = LDL^T$ can be extended to incomplete factorizations of $A+\beta I$. In this case, Theorems 2.3 and 2.4 still hold, while the bounds (2.22) and (2.23) also include the error in the incomplete factorization, $\|A+\beta I - LDL^T\|$, similarly to the case where A is SPD.

3. Two practical preconditioners. In this section we present two specific sequences $\{P_k\}$ belonging to the class $UFP(A,\Delta_k)$, which have low-cost implementations.

The first one is obtained as a simple extension of the preconditioning strategy studied by the authors in [2] for sequences where $\Delta_k = \alpha_k I$, $\alpha_k > 0$. In [2], given an incomplete LDL^T factorization of A, a preconditioner for $A+\alpha_k I$ is defined as

$$P_k^1 = (L+E_k+F_k)D(L+E_k+F_k)^T, \tag{3.1}$$

where E_k is a diagonal matrix and F_k is a strictly lower triangular matrix. The diagonal entries of $E_k = diag(e_1^k,\dots,e_n^k)$ have the following form

$$e_i^k = \sqrt{\frac{d_i+\alpha_k}{d_i}} - 1,$$

for $i=1,\dots,n$, while the nonzero entries of $F_k = (f_{i,j}^k)$ are given by

$$f_{i,j}^k = \gamma_j^k\, l_{i,j}, \qquad \gamma_j^k = \sqrt{\frac{d_j}{d_j+\alpha_k}} - 1,$$

for $i=2,\dots,n$, and $j=1,\dots,i-1$. This preconditioner can be extended to the matrix $A+\Delta_k$ by generalizing the definitions of e_j^k and $f_{i,j}^k$ as follows:

$$e_i^k = \sqrt{\frac{d_i+\delta_i^k}{d_i}} - 1, \tag{3.2}$$

$$f_{i,j}^k = \gamma_j^k\, l_{i,j}, \qquad \gamma_j^k = \sqrt{\frac{d_j}{d_j+\delta_j^k}} - 1. \tag{3.3}$$

It is easy to verify that the sequence $\{P_k^1\}$ defined by (3.1), (3.2) and (3.3) belongs to $UFP(A,\Delta_k)$. By using (3.2) and (3.3), we can write $L+E_k+F_k$ as

$$\begin{aligned} L+E_k+F_k &= L + D^{-\frac{1}{2}}(D+\Delta_k)^{\frac{1}{2}} - I + \mathit{off}(L)D^{\frac{1}{2}}(D+\Delta_k)^{-\frac{1}{2}} - \mathit{off}(L) \\ &= \left(I+\mathit{off}(L)D(D+\Delta_k)^{-1}\right) D^{-\frac{1}{2}}(D+\Delta_k)^{\frac{1}{2}}, \end{aligned}$$

and this gives

$$P_k^1 = \left(I+\mathit{off}(L)D(D+\Delta_k)^{-1}\right)(D+\Delta_k)\ \left(I+\mathit{off}(L)D(D+\Delta_k)^{-1}\right)^T.$$

Hence, P_k^1 has the form (2.1), with D_k and L_k such that

$$d_i^k = d_i + \delta_i^k, \qquad i = 1, \dots, n, \tag{3.4}$$

and

$$\mathit{diag}(L_k) = (1, 1, ..., 1)^T, \quad \mathit{off}(L_k) = \mathit{off}(L) S_k, \quad S_k = D D_k^{-1},$$

(for simplicity of notations we neglect the superscript 1 from the factors L_k and D_k). Trivially, conditions (A.1)–(A.3) are satisfied. Furthermore, from (3.4) it follows that $\|D + \Delta_k - D_k\| = 0$, hence P_k^1 fullfills property (A.4) too. We note that the computational cost of building D_k is $O(n)$ and hence the overall cost for the preconditioner P_k^1 is generally low when compared with the cost of recomputing an incomplete LDL^T factorization of $A+\Delta_k$. Furthermore, all the entries of L_k can be computed in parallel.

The second preconditioner P_k^2 is a new preconditioner. It has the form $P_k^2 = L_k D_k L_k^T$ (as for P_k^1, we neglect the superscript 2 in D_k and L_k), where the entries of D_k are defined as

$$d_i^k = d_i + \delta_i^k + \sum_{j=1}^{i-1} (l_{i,j})^2 (d_j - (s_j^k)^2 d_j^k), \tag{3.5}$$

and L_k is defined as in (A.3). This choice of D_k has been suggested by equality (2.6), which holds if $A = LDL^T$; in this case, by defining d_i^k as in (3.5) we get

$$\mathit{diag}(P_k^2) = \mathit{diag}(A + \Delta_k), \tag{3.6}$$

i.e., we annihilate the diagonal of the error matrix $A + \Delta_k - P_k^2$. Obviously, (3.6) is no longer true when $A \neq LDL^T$, but we can assume that the discrepancy between $\mathit{diag}(P_k^2)$ and $\mathit{diag}(A+\Delta_k)$ is small if the seed preconditioner is a good approximation of A.

Now we show that the sequence $\{P_k^2\}$ belongs to $\mathit{UFP}(A, \Delta_k)$, by proving that conditions (A.1) and (A.2) are satisfied. We proceed by induction. We first show that

$$d_i^k \geq d_i + \delta_i^k, \tag{3.7}$$
$$d_i - (s_i^k)^2 d_i^k \geq 0, \tag{3.8}$$

for $i = 1, \dots, n$. Trivially, for $i = 1$

$$d_1^k = d_1 + \delta_1^k;$$

then

$$d_1 - (s_1^k)^2 d_1^k = d_1 \left(1 - \frac{d_1}{d_1^k}\right) \geq 0.$$

For $i > 1$, suppose that (3.7) and (3.8) hold for all $j < i$. Then, inequalities (3.7) and (3.8) follow from (3.5). Therefore $\{P_k^2\}$ satisfies (A.1). Concerning (A.2), we first show that, for $i = 1, \dots, n$, there exists $\sigma_i > 0$ such that

$$d_i^k - d_i \leq \sigma_i \|\Delta_k\|. \tag{3.9}$$

For $i = 1$ inequality (3.9) trivially holds with $\sigma_i = 1$. For $i > 1$, by assuming that (3.9) holds for all $j < i$ and using (3.5), we have

$$\begin{aligned} d_i^k - d_i &= \delta_i^k + \sum_{j=1}^{i-1} (l_{i,j})^2 s_j^k (d_j^k - d_j) \leq \delta_i^k + \sum_{j=1}^{i-1} (l_{i,j})^2 (d_j^k - d_j) \\ &\leq \delta_i^k + \sum_{j=1}^{i-1} \sigma_j (l_{i,j})^2 \|\Delta_k\| \leq \left(1 + \sum_{j=1}^{i-1} \sigma_j (l_{i,j})^2 \right) \|\Delta_k\|, \end{aligned}$$

and inequality (3.9) holds with $\sigma_i = 1 + \sum_{j=1}^{i-1} l_{i,j}^2 \sigma_j$. Then (3.9) holds for $i = 1, \ldots, n$ and (A.2) is satisfied with $\sigma = \max_i \sigma_i$.

By using (3.5) and noting that $d_j - (s_j^k)^2 d_j^k = d_j(1 - s_j^k) \leq d_j$, we have

$$\left| d_i^k - d_i - \delta_i^k \right| \leq \sum_{j=1}^{i-1} (l_{i,j})^2 d_j,$$

and hence

$$\|D_k - D - \Delta_k\| \leq \|diag(off(L) D off(L)^T)\|.$$

Then P_k^2 also satisfies property (A.4).

Finally, from (3.5) we see that the computational cost of building the factor D_k of P_k^2 is $O(nnz(L))$, as the cost of building L_k. Therefore, P_k^2 is more expensive than P_k^1 and its overhead decreases with the density of L. However, as shown in section 4, despite its higher computational cost, P_k^2 is valuable as it provides a better approximation of the diagonal of the matrix $A + \Delta_k$.

REMARK 3.1. *If $A = LDL^T$, the definition of d_i^k in (3.5) is equivalent to*

$$d_i^k = a_{i,i} + \delta_i^k - \sum_{j=1}^{i-1} (l_{i,j}^k)^2 d_j^k, \tag{3.10}$$

and the latter can be used to compute D_k. This is no longer true when $A \neq LDL^T$ and the use of (3.10) could yield a breakdown; in particular, d_i^k might become negative or smaller than d_i.

4. Numerical experiments.

We analyzed the behaviour of the preconditioners P_k^1 and P_k^2 on sequences of linear systems arising in optimization. We considered two sets of sequences: the first one is made of sequences arising in the reflective Newton method for bound-constrained quadratic programming problems [10], implemented in the Matlab function `quadprog`; the second one consists of sequences of shifted linear systems arising in the Regularized Euclidean Residual (RER) overestimation method for nonlinear least squares [1].

The updating strategies were compared with other preconditioning techniques. For the first test set we considered the diagonal and tridiagonal preconditioners provided by `quadprog`, computed from scratch for each system of the sequence. For the second test set the comparison was carried out with an incomplete factorization preconditioner recomputed for each matrix of the sequence and with the frozen seed preconditioner. More details are given in sections 4.1 and 4.2. The comparisons were performed using the performance profiles proposed by Dolan and Moré [11] and briefly described next. Let $\mathcal{S}_{\mathcal{T},\mathcal{A}} \geq 0$ be a statistic corresponding to the successful solution of a test problem $\mathcal{T}$ by an algorithm $\mathcal{A}$, and suppose that the smaller this value is, the better the algorithm is considered; furthermore, let $\mathcal{S}_{\mathcal{T}}$ be the smallest value attained on the test $\mathcal{T}$ by one of the algorithms under analysis, and χ_M be a value such that $\chi_M > \mathcal{S}_{\mathcal{T},\mathcal{A}}/\mathcal{S}_{\mathcal{T}}$ for all $\mathcal{T}$ and $\mathcal{A}$. The performance profile of the algorithm $\mathcal{A}$ is defined as

$$\pi(\chi) = \frac{\text{number of tests such that } \mathcal{S}_{\mathcal{T},\mathcal{A}}/\mathcal{S}_{\mathcal{T}} \leq \chi}{\text{number of tests}}, \quad \chi \geq 1,$$

where the ratio $\mathcal{S}_{\mathcal{T},\mathcal{A}}/\mathcal{S}_{\mathcal{T}}$ is set equal to χ_M if the algorithm $\mathcal{A}$ fails in solving the test $\mathcal{T}$. In other words, $\pi(\chi)$ is the fraction of problems for which $\mathcal{S}_{\mathcal{T},\mathcal{A}}$ is within a factor χ of the smallest value $S_{\mathcal{T}}$. At $\chi = 1$ the performance profile gives the percentage of problems for which the algorithm $\mathcal{A}$ is the best, while the percentage of problems that are successfully solved by the algorithm $\mathcal{A}$ is $\lim_{\chi \to \chi_M^-} \pi(\chi)$. Finally, we note that when the values of $\mathcal{S}_{\mathcal{T},\mathcal{A}}/\mathcal{S}_{\mathcal{T}}$ have different magnitudes, it may be convenient to use a logarithmic scale performance profile, i.e.,

$$\pi_{log}(\chi) = \frac{\text{number of tests such that } \log\left(\mathcal{S}_{\mathcal{T},\mathcal{A}}/\mathcal{S}_{\mathcal{T}}\right) \leq \chi}{\text{number of tests}}, \quad \chi \geq 0;$$

here we use a base-2 logarithm.

The experiments were run using Matlab R2010b on an Intel Core 2 Duo U9600 processor, with clock frequency of 1.6 GHz, 3 GB of RAM and 3 GB of cache memory. The seed preconditioners were computed using the `cholinc` Matlab function, which implements the zero fill-in factorization as well as a factorization based on a drop tolerance. In our experiments `cholinc` was run using a drop tolerance as specified in the next subsections. The preconditioners P_k^1 and P_k^2 were implemented as Fortran 90 mex-files with Matlab interface. We note that we initially coded both preconditioners in Matlab, but we found that accessing the columns of L_k (or the rows of L_k^T) one at a time, as required for the construction of P_k^2, produces a significant time overhead, independently of the sparsity of L_k. Conversely, P_k^1 was efficiently implemented in Matlab, since the entries of its triangular factor can be updated all at once. We also observed that the times for building P_k^1 were comparable for the Matlab and mex-Fortran 90 versions. Therefore, we decided to use mex-Fortran 90 implementations for both preconditioners. In the experiments, the elapsed times were measured, in seconds, using the `tic` and `toc` Matlab commands.

4.1. Preconditioner updates in quadprog. The function `quadprog` available in the Matlab Optimization Toolbox implements the reflective Newton method for bound-constrained quadratic programming problems:

$$\min_y \left\{ q(y) = \frac{1}{2} y^T Q y + c^T y : \ \ l \leq y \leq u \right\}, \tag{4.1}$$

where $Q \in \mathbb{R}^{n\times n}$ is symmetric, $c \in \mathbb{R}^n$, $l \in \{\mathbb{R} \cup \{-\infty\}\}^n$, $u \in \{\mathbb{R} \cup \{\infty\}\}^n$, and $l < u$.

In a preprocessing phase, `quadprog` shifts and scales the vectors l and u so that the finite values of l map to zero and the finite values of u map to 1. Hence, the quadratic programming problem becomes

$$\min_x \left\{ \bar{q}(x) = \frac{1}{2} x^T G Q G x + c^T G x : \ \ \bar{l} \leq x \leq \bar{u} \right\}, \tag{4.2}$$

where G is a diagonal matrix with the ith diagonal entry equal to $(u_i - l_i)$ if $-\infty < l_i < u_i < \infty$, and to 1 otherwise.

The procedure implemented in `quadprog` is an affine-scaling method that generates a strictly feasible sequence $\{x_k\}$. At each iteration, the solution of the linear system

$$(M_k GQGM_k + D_k^g)s = -M_k g(x_k), \tag{4.3}$$

is required, where $g(x_k) = \nabla\bar{q}(x_k) = GQGx_k + Gc$, M_k is a positive definite diagonal matrix, and D_k^g is a positive semidefinite diagonal matrix. The diagonal entries of M_k are given by

$$m_{i,i}^k = \begin{cases} (\bar{u}_i - x_i^k)^{1/2} & \text{if} \quad (g(x_k))_i < 0 \quad \text{and} \quad \bar{u}_i < \infty, \\ (x_i^k - \bar{l}_i)^{1/2} & \text{if} \quad (g(x_k))_i > 0 \quad \text{and} \quad \bar{l}_i > -\infty, \\ 1 & \quad\text{otherwise,} \end{cases}$$

and the diagonal entries of D_k^g by

$$(D_k^g)_{i,i} = \begin{cases} |(g(x_k))_i| & \text{if } m_{i,i}^k = (\bar{u}_i - x_i^k)^{1/2} \text{ or } m_{i,i}^k = (x_i^k - \bar{l}_i)^{1/2}, \\ 0 & \text{otherwise.} \end{cases}$$

If the quadratic problems are convex, then Q is positive semidefinite and $M_k GQGM_k$ is positive semidefinite too. Note that, if the box $[l, u]$ is bounded, then $\|M_k\| \leq 1$. Henceforth, $M_k GQGM_k + D_k^g$ is denoted by H_k.

The function `quadprog` solves the linear systems (4.3) by using the preconditioned CG procedure `pcgr` and supplies two kind of preconditioners computed from scratch for each system. The diagonal preconditioner

$$P_k^{diag} = diag\left(\|H_k(:,1)\|_2, \ldots, \|H_k(:,n)\|_2\right),$$

where $H_k(:,j)$ denotes the jth column of H_k, is used by default. Alternatively, the user can fix a positive integer l and choose a banded preconditioner which is formed extracting from H_k the main diagonal and l lower and upper diagonals. If the Cholesky factorization of the banded matrix fails, a shift is applied and a new Cholesky factorization is attempted. In our experiments we chose $l = 1$, corresponding to a tridiagonal preconditioner, denoted by P_k^{trid} in the following.

To embed our updating techniques in `quadprog` we proceed as follows. If Q is SPD, we compute an incomplete Cholesky factorization of GQG using the `cholinc` function with drop tolerance 10^{-2}. Otherwise, we compute an incomplete Cholesky factorization of the shifted SPD matrix $GQG + \beta I$, using the same dropping tolerance (in our experiments $\beta = 10^{-1}$). Trivially, for any k, these factorizations provide an incomplete LDL^T factorization of $M_k GQGM_k$, which is used as the seed preconditioner for the kth matrix of the sequence. We remark that, as in the original `quadprog` implementation, we precondition the linear systems (4.3); equivalent systems with matrices $GQG + M_k^{-1}D_k^g$ are not preferable since $GQG + M_k^{-1}D_k^g$ may not be bounded in a neighbourhood of a nondegenerate point satisfying the second-order sufficiency conditions (see [4, 10]).

We run `quadprog` using the default tolerances for the stopping criteria of the nonlinear procedure, and the default bound `cg_itmax`$=n/2$ on the number of CG iterations. The solver `pcgr` was stopped when the 2-norm of the relative residual was less than a tolerance `cg_tol`. In order to test the performance of the preconditioners, we used `cg_tol`$=10^{-1}, 10^{-3}, 10^{-5}$; the default value in `quadprog` is `cg_tol` $= 10^{-1}$.

Problem	n	*dens*(Q)	*dens*(L)
BIGGSB1	5000	6.00e-004	1.20e-003
CHENHARK	5000	1.00e-003	2.00e-003
CVXBQP1	10000	7.00e-004	1.40e-003
JNLBRNG1	9604	5.16e-004	1.03e-003
JNLBRNG2	9604	5.16e-004	1.03e-003
JNLBRNGA	9604	5.16e-004	1.03e-003
JNLBRNGB	9604	5.16e-004	1.03e-003
NOBNDTOR	5184	9.54e-004	1.91e-003
OBSTCLAE	9604	5.16e-004	1.03e-003
OBSTCLAL*	9604	5.16e-004	1.03e-003
OBSTCLBL*	9604	5.16e-004	1.03e-003
OBSTCLBM	9604	5.16e-004	1.03e-003
OBSTCLBU*	9604	5.16e-004	1.03e-003
PENTDI	5000	1.00e-003	2.00e-003
QUDLIN	5000	2.00e-004	4.00e-004
TORSION1*	5184	9.54e-004	1.91e-003
TORSION2	5184	9.54e-004	1.91e-003
TORSION3*	5184	9.54e-004	1.91e-003
TORSION4	5184	9.54e-004	1.91e-003
TORSION5*	5184	9.54e-004	1.91e-003
TORSION6	5184	9.54e-004	1.91e-003
TORSIONA*	5184	9.54e-004	1.91e-003
TORSIONB	5184	9.54e-004	1.91e-003
TORSIONC*	5184	9.54e-004	1.91e-003
TORSIOND	5184	9.54e-004	1.91e-003
TORSIONE*	5184	9.54e-004	1.91e-003
TORSIONF	5184	9.54e-004	1.91e-003

TABLE 4.1
Bound-constrained convex quadratic programming problems from CUTEr.

We compared the four preconditioners on a test set consisting of all the 28 bound-constrained convex quadratic programming problems of dimension $n > 500$ available in the CUTEr collection. The test problems are listed in Table 4.1 along with their dimension, the density of the matrix Q and the density of the factor L in the seed preconditioner, defined as $dens(Q) = nnz(Q)/n^2$ and $dens(L) = nnz(L)/(n(n+1)/2)$. These problems are equipped with an initial guess that in some cases is equal to the lower or the upper bound. Since `quadprog` requires a strictly feasible initial guess, we set it as $x_0 = l + (u - l)/4$ when the provided one is equal to the lower bound, and as $x_0 = l + 3(u - l)/4$ when the provided one is equal to the upper bound. Problems with modified initial guess are marked appending an asterisk to the name.

First, we briefly analyze the quality of the solutions of the optimization problems computed by `quadprog` with the four preconditioning strategies. Then, we focus on evaluating the performance of the preconditioners in the solution of the sequences of linear systems.

At the convergence of the affine-scaling method, `quadprog` provides a measure of first-order optimality, namely the value of $\|M_k g(x_k)\|_\infty$. We observe that the lowest values of this measure are generally obtained when using P_k^1 and P_k^2. This is shown in Figure 4.1, where we plot the performance profiles of `quadprog` with the four preconditioners, in logarithmic scale, using as performance statistic the first-order optimality measure. The performance profiles correspond to `cg_tol`$=10^{-3}$, but they are also representative of the runs performed with the other values of `cg_tol`. The only exception is for the problem CHENHARK with `cg_tol` $= 10^{-1}$ and the

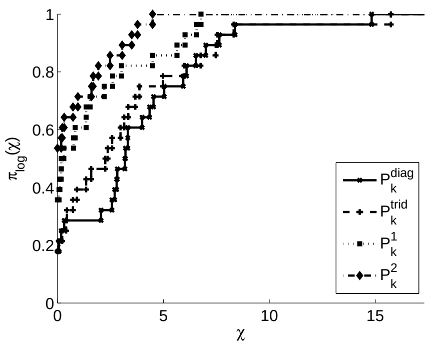

FIG. 4.1. *Performance profiles of P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2, for* `cg_tol`$=10^{-3}$*: first-order optimality measure provided by* `quadprog`*.*

preconditioner P_k^1. In this case, `quadprog` stops after 200 nonlinear iterations (the maximum number allowed) and the first-order optimality measure is greater than the optimality measure obtained by using the other three preconditioners. We consider this run as a failure of P_k^1.

We found that the number of nonlinear iterations is fairly insensitive to the preconditioner used. Hence, the preconditioning strategies were applied to sequences of linear systems of comparable lengths. Figures 4.2-4.4 display the performance profiles of the four preconditioners for the three values of `cg_tol`, using as performance statistic both the number of CG iterations and execution time. The execution time is the time required to solve the whole sequence of linear systems arising from the application of `quadprog`, including the time devoted to either the construction or the update of the preconditioner. The number of CG iterations is the sum of the CG iterations required for each system. The performance profiles are plotted in the interval $[0, 10]$ to better highlight the results of the comparison. We see that P_k^1 and P_k^2 outperform P_k^{diag} and P_k^{trid} in terms of both CG iterations and time. Furthermore, P_k^1 and P_k^2 appear much more efficient than the others when the accuracy requirement in the solution of the linear systems increases. We also note that P_k^2 generally yields a smaller number of CG iterations than P_k^1; hence a more accurate approximation of the diagonal of the matrix H_k seems to improve the quality of the preconditioner. Finally, the highest cost of P_k^2 with respect to P_k^1 is offset by the greatest effectiveness of P_k^2, so that P_k^1 and P_k^2 are comparable in terms of execution time.

Detailed results of all the runs are reported in Appendix A along with further comments.

4.2. Preconditioner updates in RER. The Regularized Euclidean Residual method is an overestimation method for nonlinear least-squares problems:

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|F(x)\|^2,$$

where F is countinuously differentiable. The step p between two successive iterates x_k and x_{k+1} is chosen to approximately minimize the regularized Gauss-Newton model

$$m(p) = \|F(x_k) + F'(x_k)p\| + \mu_k||p||^2, \tag{4.4}$$

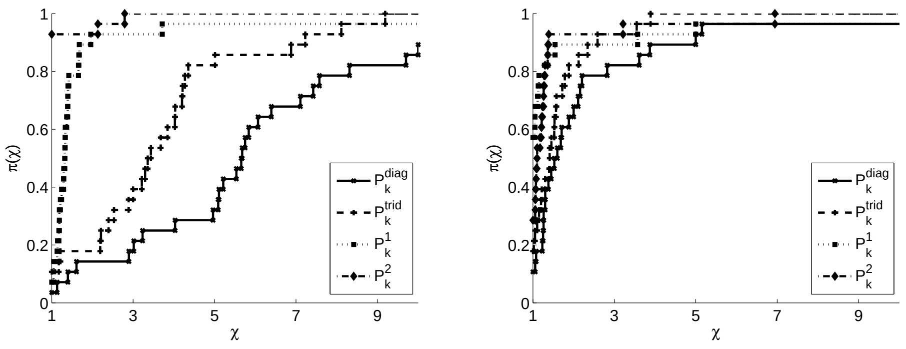

FIG. 4.2. *Performance profiles of P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2 within* `quadprog`*, for* `cg_tol`$=10^{-1}$*: number of CG iterations (left) and execution time (right).*

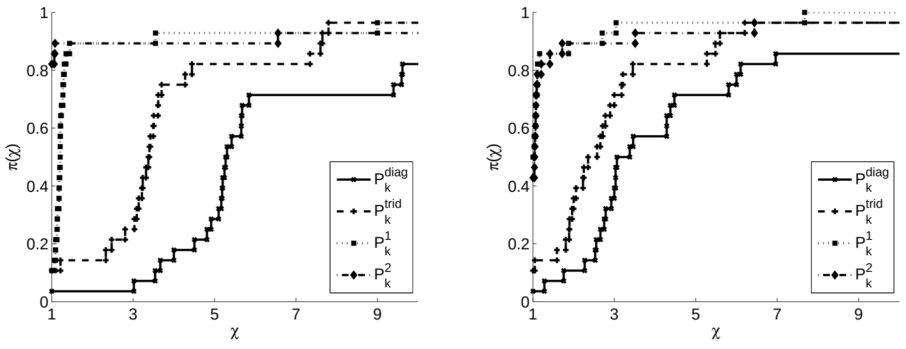

FIG. 4.3. *Performance profiles of P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2 within* `quadprog`*, for* `cg_tol`$=10^{-3}$*: number of CG iterations (left) and execution time (right).*

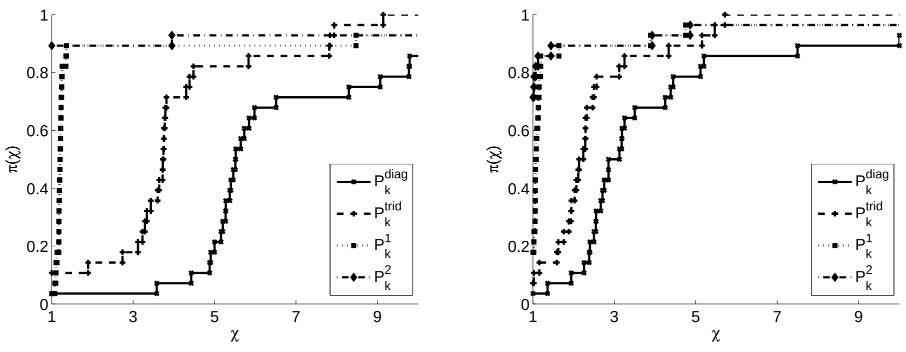

FIG. 4.4. *Performance profiles of P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2 within* `quadprog`*, for* `cg_tol`$=10^{-5}$*: number of CG iterations (left) and execution time (right).*

where F' is the Jacobian of F and μ_k is a positive parameter. Define the vector $p(\nu)$ so that

$$(F'(x_k)^T F'(x_k) + \nu I)p(\nu) = -F'(x_k)^T F(x_k); \tag{4.5}$$

then, the minimizer of (4.4) is $p(\nu^*)$, where ν^* is the positive root of the secular equation

$$\phi(\nu) = 2\mu_k \|F(x_k) + F'(x_k)p(\nu)\|. \tag{4.6}$$

The Newton or secant method applied to (4.6) produces a sequence of scalars $\{\nu_r\}$ and require the evalutation of ϕ at each iterate. This amounts to solve a sequence of shifted linear systems of the form (4.5) with $\nu = \nu_r$.

We tested the preconditioners P_k^1 and P_k^2 on sequences of shifted systems arising from the computation of some steps in the RER algorithm. In particular, we considered the eight sequences used in [2], generated in the solution of nonlinear systems of equations of dimension 10^4. The nonlinear systems were obtained from the discretization of three PDE problems: the Bratu problem [24], a PDE problem modelling a flow in a porous medium [24], and the PDE problem given in [12, §3.2.1]. For details on the eight sequences of linear systems we refer to [2, section 6.1].

The systems of each sequence were solved by using the CG method implemented in the Matlab function `pcg`, with zero initial guess. The CG method was terminated when the ratio between the 2-norm of the current and the initial residual was less than 10^{-6}, or when a maximum of 1000 iterations was reached. In all the sequences the matrix $F'(x_k)^T F'(x_k)$ is positive definite; then, the seed preconditioner was obtained by performing an incomplete LDL^T factorization of such matrix with the `cholinc` function. The drop tolerance was chosen as explained in [2, Section 6.1], so that CG coupled with the seed preconditioner succeeds in solving $F'(x_k)^T F'(x_k)p = -F'(x_k)^T F(x_k)$.

For comparison purposes, we also considered the following preconditioning strategies: recomputing the incomplete LDL^T factorization for each matrix of the sequence and "freezing" the incomplete LDL^T factorization for all the values of ν_r. The corresponding preconditioners are denoted by P_k^{rec} and P^{frz}, respectively.

Figure 4.5 displays the performance profiles of the four preconditioning strategies, in terms of the number of CG iterations and of the execution time needed to solve the whole sequences. We note that only P_k^{rec} and P_k^1 are able to solve all the systems of each sequence. The preconditioner P_k^2 fails in solving one system from one sequence, while the preconditioner P^{frz} fails in solving all the systems of one sequence but the first, and all the systems of another sequence but the first two. As expected, the behaviour of the updating strategies in terms of CG iterations is between the behaviours of P_k^{rec} and P^{frz}. The situation is different if we consider the execution time, since P_k^{rec} is outperformed by both the updating strategies because of the high cost of computing it. Finally, we note that P_k^2 requires a smaller execution time than P_k^1 for most of the problems; this means that the savings produced by P_k^2 in terms of CG iterations are significant enough to compensate for its higher computational cost.

5. Conclusions. We have proposed a framework for preconditioning sequences of positive semidefinite linear systems that differ by a diagonal matrix Δ_k. For any given sequence, the preconditioners are obtained by updating a seed preconditioner, available in factorized form. The preconditioners in the framework are effective on slowly varying sequences; furthermore, if they satisfy an additional requirement, they

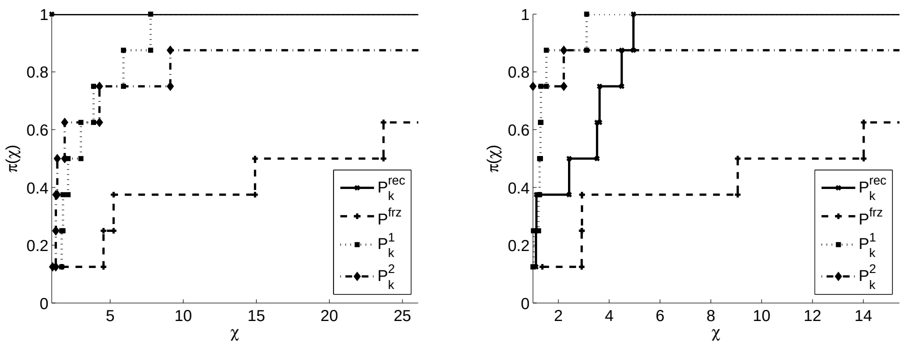

FIG. 4.5. *Performance profiles of P_k^{rec}, P^{frz}, P_k^1 and P_k^2 on the sequences generated by RER: number of CG iterations (left) and execution time (right).*

yield good spectral properties in the preconditioned matrices when the entries of Δ_k become large. Two practical low-cost preconditioners belonging to the framework have been presented and tested in the solution of sequences of linear systems arising from numerical optimization methods, showing the effectiveness of the proposed approach.

Appendix A. Complete numerical results with `quadprog`.

In Tables A.1-A.6 we provide detailed results of the numerical experiments carried out with `quadprog`, varying the stopping tolerance `cg_tol` of the `pcgr` solver. For the sake of readability, the index k is dropped from the names P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2 of the preconditioners. The headers of columns 3–6 have the following meaning:

- *Itn*: number of nonlinear iterations performed by `quadprog`.
- *Itl*: sum of the numbers of CG iterations performed at each nonlinear step.
- *Tp*: execution time, in seconds, to build the preconditioner. For the updating procedures it is the time required by the incomplete Cholesky factorization of GQG (or $GQG+\beta I$) and by the subsequent updates; for P_k^{diag} it is the overall time needed to build the diagonal preconditioner at each nonlinear iteration; finally, for P_k^{trid} it is the overall time for extracting the tridiagonal band of H_k and performing its Cholesky factorization at each nonlinear iteration.
- *Tt*: execution time, in seconds, required to solve the whole sequence of linear systems. It is the sum of the times required to build/update the preconditioners and to solve the linear systems by CG.

The tables show that our updating strategies are generally most effective than the other ones, as already observed in section 4.1. A few runs deserve some comments. First, the sequence of linear systems arising when `quadprog` is applied to CHENNARK is hard to be solved for all the strategies. Second, the matrix Q in BIGGSB1 and QUDLIN is tridiagonal and therefore CG with P_k^{trid} works as a direct method (small differences in the execution times as `cg_tol` varies are due to the resolution of the `tic` and `toc` timer); this fact justifies the number of linear iterations performed. Interestingly, with BIGGSB1 using CG with either P_k^{trid} or P_k^2 requires the same number of iterations and effort in terms of computational time. Finally, we note that our updating strategies do not seem to be able to build efficient preconditioners for the sequence arising in the solution of CVXBQP1.

Problem	*Prec*	*Itn*	*Itl*	*Tp*	*Tt*
BIGGSB1	P^{diag}	23	17573	5.2e-2	2.5e1
	P^{trid}	7	7	2.5e-2	4.5e-2
	P^1	9	26	1.7e-2	6.8e-2
	P^2	7	7	2.3e-2	4.4e-2
BQPGAUSS	P^{diag}	15	1291	2.1e-2	7.3e-1
	P^{trid}	14	938	2.7e-2	5.6e-1
	P^1	18	569	5.7e-3	2.0e0
	P^2	16	428	7.5e-1	1.8e0
CHENHARK	P^{diag}	17	12579	5.0e-2	2.0e1
	P^{trid}	16	10951	1.1e-1	1.6e1
	P^1	200	244	3.0e-1	9.2e-1
	P^2	16	9008	5.7e-2	1.7e1
CVXBQP1	P^{diag}	24	24	2.4e-1	3.8e-1
	P^{trid}	24	24	2.7e-1	3.6e-1
	P^1	25	47	1.2e0	1.8e0
	P^2	32	67	1.7e0	2.5e0
JNLBRNG1	P^{diag}	20	456	1.1e-1	1.3e0
	P^{trid}	24	432	1.9e-1	1.4e0
	P^1	18	57	1.5e-1	3.6e-1
	P^2	17	47	2.1e-1	3.9e-1
JNLBRNG2	P^{diag}	23	291	1.2e-1	8.2e-1
	P^{trid}	21	253	1.6e-1	8.9e-1
	P^1	24	58	1.8e-1	4.0e-1
	P^2	20	35	2.3e-1	3.8e-1
JNLBRNGA	P^{diag}	20	440	1.1e-1	1.2e0
	P^{trid}	20	357	1.5e-1	1.1e0
	P^1	16	46	1.5e-1	3.1e-1
	P^2	20	44	2.4e-1	4.3e-1
JNLBRNGB	P^{diag}	25	245	1.4e-1	7.5e-1
	P^{trid}	24	227	1.8e-1	8.8e-1
	P^1	20	44	1.6e-1	3.4e-1
	P^2	22	33	2.6e-1	4.1e-1
NOBNDTOR	P^{diag}	16	303	4.3e-2	4.8e-1
	P^{trid}	14	161	5.8e-2	3.2e-1
	P^1	12	47	7.4e-2	1.7e-1
	P^2	12	40	1.0e-1	2.0e-1
OBSTCLAE	P^{diag}	23	827	1.5e-1	2.2e0
	P^{trid}	17	189	1.3e-1	6.9e-1
	P^1	19	63	2.1e-1	4.9e-1
	P^2	15	45	2.4e-1	4.4e-1
OBSTCLAL	P^{diag}	27	1351	1.5e-1	3.4e0
	P^{trid}	21	451	1.6e-1	1.4e0
	P^1	24	102	2.4e-1	6.6e-1
	P^2	23	90	3.2e-1	7.1e-1
OBSTCLBL	P^{diag}	26	139	1.4e-1	5.5e-1
	P^{trid}	26	103	2.1e-1	9.1e-1
	P^1	19	45	2.4e-1	5.1e-1
	P^2	25	43	3.8e-1	6.2e-1
OBSTCLBM	P^{diag}	22	110	1.2e-1	4.7e-1
	P^{trid}	24	84	1.9e-1	4.7e-1
	P^1	25	53	2.7e-1	5.4e-1
	P^2	26	38	3.9e-1	6.1e-1
OBSTCLBU	P^{diag}	15	143	8.1e-2	4.8e-1
	P^{trid}	15	84	1.2e-1	3.9e-1
	P^1	21	47	2.5e-1	5.0e-1
	P^2	18	28	3.3e-1	5.0e-1

TABLE A.1

Comparison of P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2 in the solution of CUTEr problems by `quadprog`*, with* `cg_tol`$=10^{-1}$.

Problem	*Prec*	*Itn*	*Itl*	*Tp*	*Tt*
PENTDI	P^{diag}	19	37	5.0e-2	1.1e-1
	P^{trid}	23	27	9.7e-2	1.6e-1
	P^1	23	23	4.1e-2	1.1e-1
	P^2	23	23	9.4e-2	1.5e-1
QUDLIN	P^{diag}	16	17	2.7e-2	6.0e-2
	P^{trid}	15	15	4.1e-2	7.2e-2
	P^1	13	25	3.6e-2	8.2e-2
	P^2	13	32	2.8e-2	8.2e-2
TORSION1	P^{diag}	14	199	3.9e-2	3.4e-1
	P^{trid}	12	118	5.3e-2	2.4e-1
	P^1	12	38	7.7e-2	1.7e-1
	P^2	10	28	9.7e-2	1.7e-1
TORSION2	P^{diag}	12	182	3.6e-2	2.9e-1
	P^{trid}	12	121	5.1e-2	2.6e-1
	P^1	12	41	8.1e-2	1.7e-1
	P^2	11	30	9.9e-2	1.8e-1
TORSION3	P^{diag}	15	173	4.1e-2	2.9e-1
	P^{trid}	16	112	7.0e-2	3.2e-1
	P^1	16	45	9.2e-2	2.2e-1
	P^2	16	34	1.2e-1	2.1e-1
TORSION4	P^{diag}	17	181	5.3e-2	3.2e-1
	P^{trid}	16	110	6.7e-2	2.4e-1
	P^1	14	38	8.5e-2	1.7e-1
	P^2	16	32	1.2e-1	2.1e-1
TORSION5	P^{diag}	17	137	4.6e-2	2.4e-1
	P^{trid}	16	86	6.7e-2	2.1e-1
	P^1	16	34	1.0e-1	1.9e-1
	P^2	18	34	1.3e-1	2.4e-1
TORSION6	P^{diag}	17	134	4.7e-2	2.9e-1
	P^{trid}	15	78	6.3e-2	2.0e-1
	P^1	17	35	9.4e-2	1.9e-1
	P^2	17	27	1.3e-1	2.1e-1
TORSIONA	P^{diag}	15	176	4.2e-2	2.9e-1
	P^{trid}	15	135	6.6e-2	2.8e-1
	P^1	14	44	8.4e-2	2.6e-1
	P^2	13	31	1.1e-1	2.3e-1
TORSIONB	P^{diag}	17	211	4.7e-2	3.8e-1
	P^{trid}	15	141	7.6e-2	3.1e-1
	P^1	13	41	8.3e-2	1.8e-1
	P^2	13	33	1.1e-1	1.9e-1
TORSIONC	P^{diag}	17	181	5.1e-2	3.2e-1
	P^{trid}	16	114	6.8e-2	2.6e-1
	P^1	15	40	1.0e-1	2.1e-1
	P^2	15	31	1.2e-1	2.0e-1
TORSIOND	P^{diag}	16	184	4.7e-2	3.2e-1
	P^{trid}	18	123	7.9e-2	3.0e-1
	P^1	14	38	8.8e-2	1.9e-1
	P^2	15	32	1.3e-1	2.4e-1
TORSIONE	P^{diag}	17	146	4.7e-2	2.6e-1
	P^{trid}	17	90	7.3e-2	2.3e-1
	P^1	16	32	1.0e-1	2.0e-1
	P^2	18	28	1.4e-1	2.2e-1
TORSIONF	P^{diag}	17	155	4.6e-2	2.9e-1
	P^{trid}	18	94	7.6e-2	2.8e-1
	P^1	18	37	9.7e-2	2.0e-1
	P^2	15	28	1.3e-1	2.0e-1

Table A.2
Comparison of P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2 in the solution of CUTEr problems by `quadprog`*, with* `cg_tol` $=10^{-1}$ *(continued).*

Problem	*Prec*	*Itn*	*Itl*	*Tp*	*Tt*
BIGGSB1	P^{diag}	11	19777	2.4e-2	2.7e1
	P^{trid}	7	7	2.8e-2	4.6e-2
	P^1	7	81	1.4e-2	1.4e-1
	P^2	7	7	3.3e-2	4.9e-2
BQPGAUSS	P^{diag}	16	7470	2.3e-2	4.1e0
	P^{trid}	17	5767	3.1e-2	3.2e0
	P^1	16	2700	6.5e-1	6.0e0
	P^2	17	2476	6.9e-1	5.5e0
CHENHARK	P^{diag}	61	140841	1.6e-1	2.0e2
	P^{trid}	66	150503	5.5e-1	2.8e2
	P^1	91	3834	1.4e-1	5.9e0
	P^2	57	25140	1.8e-1	3.8e1
CVXBQP1	P^{diag}	24	62	2.4e-1	4.3e-1
	P^{trid}	24	62	2.7e-1	4.5e-1
	P^1	24	220	1.2e0	3.3e0
	P^2	26	4101	1.5e0	3.8e1
JNLBRNG1	P^{diag}	19	1631	1.3e-1	5.5e0
	P^{trid}	20	1292	2.2e-1	4.9e0
	P^1	18	194	1.6e-1	7.9e-1
	P^2	21	170	2.4e-1	8.5e-1
JNLBRNG2	P^{diag}	21	1071	1.4e-1	3.6e0
	P^{trid}	20	837	2.1e-1	3.4e0
	P^1	23	154	1.7e-1	6.9e-1
	P^2	21	114	2.4e-1	6.2e-1
JNLBRNGA	P^{diag}	16	1048	1.1e-1	3.6e0
	P^{trid}	17	850	1.8e-1	3.3e0
	P^1	20	139	1.6e-1	6.3e-1
	P^2	19	109	2.2e-1	5.9e-1
JNLBRNGB	P^{diag}	20	951	1.5e-1	3.3e0
	P^{trid}	21	696	2.3e-1	2.9e0
	P^1	21	131	1.7e-1	6.4e-1
	P^2	20	91	2.3e-1	5.5e-1
NOBNDTOR	P^{diag}	12	786	4.4e-2	1.7e0
	P^{trid}	11	502	6.9e-2	1.1e0
	P^1	13	157	7.7e-2	3.8e-1
	P^2	13	139	1.0e-1	4.0e-1
OBSTCLAE	P^{diag}	17	3268	1.2e-1	1.1e1
	P^{trid}	17	633	1.9e-1	2.6e0
	P^1	17	162	2.0e-1	8.1e-1
	P^2	18	148	2.9e-1	8.6e-1
OBSTCLAL	P^{diag}	22	4083	1.6e-1	1.3e1
	P^{trid}	21	951	2.1e-1	3.8e0
	P^1	21	229	2.3e-1	1.1e0
	P^2	23	214	3.3e-1	1.2e0
OBSTCLBL	P^{diag}	22	572	1.5e-1	2.0e0
	P^{trid}	18	356	1.9e-1	1.4e0
	P^1	21	150	2.5e-1	8.8e-1
	P^2	21	127	3.5e-1	9.2e-1
OBSTCLBM	P^{diag}	22	576	1.5e-1	2.1e0
	P^{trid}	18	356	1.9e-1	1.5e0
	P^1	18	144	2.0e-1	8.3e-1
	P^2	23	153	3.6e-1	1.0e0
OBSTCLBU	P^{diag}	22	572	1.5e-1	2.1e0
	P^{trid}	18	356	2.0e-1	1.5e0
	P^1	20	144	2.4e-1	8.8e-1
	P^2	18	110	3.1e-1	7.9e-1

TABLE A.3

Comparison of P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2 in the solution of CUTEr problems by `quadprog`, *with* `cg_tol` $=10^{-3}$.

Problem	*Prec*	*Itn*	*Itl*	*Tp*	*Tt*
PENTDI	P^{diag}	23	131	8.2e-2	3.6e-1
	P^{trid}	23	45	1.2e-1	2.4e-1
	P^1	22	37	3.9e-2	1.2e-1
	P^2	24	40	8.1e-2	1.7e-1
QUDLIN	P^{diag}	15	55	2.5e-2	1.3e-1
	P^{trid}	15	15	4.0e-2	7.4e-2
	P^1	15	135	2.2e-2	2.0e-1
	P^2	15	170	4.4e-2	2.6e-1
TORSION1	P^{diag}	12	506	4.4e-2	9.8e-1
	P^{trid}	12	355	7.3e-2	7.7e-1
	P^1	10	115	7.5e-2	3.2e-1
	P^2	10	96	9.4e-2	2.9e-1
TORSION2	P^{diag}	12	526	5.0e-2	1.0e0
	P^{trid}	12	355	6.6e-2	8.5e-1
	P^1	12	126	8.0e-2	3.3e-1
	P^2	12	107	1.0e-1	3.3e-1
TORSION3	P^{diag}	13	446	5.5e-2	8.8e-1
	P^{trid}	15	311	8.3e-2	7.2e-1
	P^1	15	104	1.0e-1	3.2e-1
	P^2	15	86	1.3e-1	3.2e-1
TORSION4	P^{diag}	13	454	4.8e-2	9.4e-1
	P^{trid}	15	311	9.6e-2	7.3e-1
	P^1	16	115	9.1e-2	3.3e-1
	P^2	15	89	1.3e-1	3.1e-1
TORSION5	P^{diag}	16	346	5.9e-2	7.9e-1
	P^1	15	81	9.0e-2	2.7e-1
	P^{trid}	15	207	8.3e-2	5.3e-1
	P^2	15	67	1.3e-1	2.9e-1
TORSION6	P^{diag}	16	346	6.1e-2	6.9e-1
	P^{trid}	15	207	8.5e-2	5.1e-1
	P^1	15	80	1.0e-1	2.7e-1
	P^2	15	66	1.3e-1	2.8e-1
TORSIONA	P^{diag}	14	462	5.5e-2	9.2e-1
	P^{trid}	14	291	9.4e-2	6.8e-1
	P^1	15	117	8.8e-2	3.4e-1
	P^2	14	96	1.2e-1	3.3e-1
TORSIONB	P^{diag}	14	451	5.1e-2	9.8e-1
	P^{trid}	14	291	7.9e-2	9.6e-1
	P^1	14	105	8.6e-2	3.3e-1
	P^2	14	85	1.1e-1	3.2e-1
TORSIONC	P^{diag}	15	454	7.2e-2	1.4e0
	P^{trid}	15	270	8.9e-2	8.9e-1
	P^1	16	102	1.0e-1	3.4e-1
	P^2	16	80	1.4e-1	3.2e-1
TORSIOND	P^{diag}	15	450	7.0e-2	1.2e0
	P^{trid}	15	270	9.1e-2	8.9e-1
	P^1	14	93	8.9e-2	2.8e-1
	P^2	14	77	1.3e-1	3.1e-1
TORSIONE	P^{diag}	16	401	6.2e-2	1.2e0
	P^{trid}	16	238	1.2e-1	7.6e-1
	P^1	16	84	9.2e-2	3.0e-1
	P^2	15	74	1.2e-1	2.8e-1
TORSIONF	P^{diag}	16	396	5.9e-2	9.0e-1
	P^{trid}	16	238	9.7e-2	5.8e-1
	P^1	14	83	8.2e-2	2.6e-1
	P^2	16	70	1.2e-1	2.7e-1

TABLE A.4
Comparison of P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2 in the solution of CUTEr problems by `quadprog`*, with* `cg_tol` $=10^{-3}$ *(continued).*

Problem	*Prec*	*Itn*	*Itl*	*Tp*	*Tt*
BIGGSB1	P^{diag}	6	20000	3.7e-2	2.6e1
	P^{trid}	7	7	2.4e-2	4.1e-2
	P^1	7	125	1.5e-2	1.9e-1
	P^2	7	7	2.4e-2	4.0e-2
BQPGAUSS	P^{diag}	18	14978	2.6e-2	8.3e0
	P^{trid}	18	11466	3.3e-2	6.1e0
	P^1	18	4917	6.6e-1	1.0e1
	P^2	18	4185	7.2e-1	8.8e0
CHENHARK	P^{diag}	63	157437	1.6e-1	2.1e2
	P^{trid}	45	110704	3.2e-1	1.6e2
	P^1	137	18973	2.1e-1	2.8e1
	P^2	30	74984	1.1e-1	1.1e2
CVXBQP1	P^{diag}	24	102	2.3e-1	5.7e-1
	P^{trid}	24	101	2.7e-1	6.6e-1
	P^1	24	856	1.6e0	1.2e1
	P^2	24	55176	1.6e0	5.7e2
JNLBRNG1	P^{diag}	20	3275	1.1e-1	7.8e0
	P^{trid}	20	2688	1.5e-1	8.2e0
	P^1	20	347	2.1e-1	1.7e0
	P^2	20	294	3.0e-1	1.5e0
JNLBRNG2	P^{diag}	23	2360	1.3e-1	5.7e0
	P^{trid}	18	1888	1.4e-1	6.7e0
	P^1	23	303	2.2e-1	1.5e0
	P^2	23	241	3.2e-1	1.3e0
JNLBRNGA	P^{diag}	16	2131	8.4e-2	5.1e0
	P^{trid}	16	1865	1.2e-1	5.2e0
	P^1	16	260	1.9e-1	1.3e0
	P^2	19	235	3.0e-1	1.2e0
JNLBRNGB	P^{diag}	20	1769	1.0e-1	4.2e0
	P^{trid}	20	1415	1.5e-1	3.9e0
	P^1	22	244	2.2e-1	1.3e0
	P^2	24	181	3.4e-1	1.2e0
NOBNDTOR	P^{diag}	13	1543	3.4e-2	2.0e0
	P^{trid}	13	1039	5.5e-2	1.6e0
	P^1	13	268	1.0e-1	7.9e-1
	P^2	13	237	1.4e-1	7.5e-1
OBSTCLAE	P^{diag}	17	6011	9.8e-2	1.5e1
	P^{trid}	17	1123	1.4e-1	3.1e0
	P^1	16	281	2.6e-1	1.5e0
	P^2	17	261	3.4e-1	1.6e0
OBSTCLAL	P^{diag}	22	8244	1.2e-1	2.4e1
	P^{trid}	22	1730	1.8e-1	5.3e0
	P^1	22	425	2.4e-1	1.9e0
	P^2	22	386	3.3e-1	1.7e0
OBSTCLBL	P^{diag}	18	1215	1.0e-1	4.5e0
	P^{trid}	18	855	1.5e-1	3.5e0
	P^1	21	303	3.2e-1	1.9e0
	P^2	21	249	4.4e-1	1.8e0
OBSTCLBM	P^{diag}	18	1212	1.1e-1	3.3e0
	P^{trid}	18	855	1.4e-1	2.7e0
	P^1	18	282	2.9e-1	1.8e0
	P^2	18	235	3.9e-1	1.7e0
OBSTCLBU	P^{diag}	22	1402	1.2e-1	3.6e0
	P^{trid}	22	999	1.8e-1	3.7e0
	P^1	22	324	2.5e-1	1.6e0
	P^2	22	266	3.5e-1	1.5e0

TABLE A.5

Comparison of P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2 in the solution of CUTEr problems by `quadprog`*, with* `cg_tol` $=10^{-5}$.

Problem	*Prec*	*Itn*	*Itl*	*Tp*	*Tt*
PENTDI	P^{diag}	24	208	6.5e-2	5.1e-1
	P^{trid}	23	89	1.0e-1	2.6e-1
	P^1	23	64	4.1e-2	1.6e-1
	P^2	23	47	7.8e-2	1.8e-1
QUDLIN	P^{diag}	15	75	3.4e-2	3.2e-1
	P^{trid}	15	15	4.2e-2	7.2e-2
	P^1	15	188	2.6e-2	2.8e-1
	P^2	15	245	3.2e-2	3.5e-1
TORSION1	P^{diag}	10	864	2.9e-2	2.2e0
	P^{trid}	10	601	5.5e-2	1.0e0
	P^1	10	188	7.7e-2	4.4e-1
	P^2	10	160	9.4e-2	4.3e-1
TORSION2	P^{diag}	11	901	3.0e-2	1.2e0
	P^{trid}	11	630	5.1e-2	9.6e-1
	P^1	11	194	8.2e-2	5.0e-1
	P^2	11	165	1.0e-1	4.2e-1
TORSION3	P^{diag}	15	848	4.2e-2	1.2e0
	P^{trid}	15	533	6.5e-2	9.1e-1
	P^1	15	179	8.9e-2	4.7e-1
	P^2	15	148	1.2e-1	4.8e-1
TORSION4	P^{diag}	15	871	4.3e-2	1.2e0
	P^{trid}	15	558	6.7e-2	9.4e-1
	P^1	15	182	9.2e-2	4.6e-1
	P^2	15	149	1.2e-1	4.2e-1
TORSION5	P^{diag}	15	541	4.1e-2	8.4e-1
	P^{trid}	15	342	6.6e-2	5.8e-1
	P^1	15	128	8.8e-2	3.5e-1
	P^2	15	104	1.2e-1	3.3e-1
TORSION6	P^{diag}	15	575	4.2e-2	8.1e-1
	P^{trid}	15	357	6.4e-2	6.4e-1
	P^1	15	133	8.9e-2	3.6e-1
	P^2	15	107	1.2e-1	3.4e-1
TORSIONA	P^{diag}	13	992	3.5e-2	1.3e0
	P^{trid}	13	680	5.6e-2	1.1e0
	P^1	13	212	8.4e-2	4.9e-1
	P^2	13	180	1.1e-1	4.8e-1
TORSIONB	P^{diag}	14	1032	4.1e-2	1.5e0
	P^{trid}	14	709	5.8e-2	1.1e0
	P^1	14	225	8.5e-2	5.7e-1
	P^2	14	187	1.1e-1	4.8e-1
TORSIONC	P^{diag}	17	927	4.7e-2	1.4e0
	P^{trid}	17	578	7.2e-2	9.3e-1
	P^1	17	191	1.0e-1	5.0e-1
	P^2	17	155	1.3e-1	4.4e-1
TORSIOND	P^{diag}	15	779	4.4e-2	1.1e0
	P^{trid}	15	514	6.2e-2	8.1e-1
	P^1	15	164	9.7e-2	4.6e-1
	P^2	15	138	1.2e-1	4.0e-1
TORSIONE	P^{diag}	17	618	4.6e-2	8.8e-1
	P^{trid}	18	407	1.0e-1	1.0e0
	P^1	17	150	9.5e-2	3.9e-1
	P^2	18	126	1.4e-1	4.0e-1
TORSIONF	P^{diag}	17	670	6.5e-2	1.3e0
	P^{trid}	16	396	9.0e-2	1.0e0
	P^1	18	158	9.8e-2	4.2e-1
	P^2	18	127	1.4e-1	4.0e-1

TABLE A.6

Comparison of P_k^{diag}, P_k^{trid}, P_k^1 and P_k^2 in the solution of CUTEr problems by `quadprog`*, with* `cg_tol`$=10^{-5}$ *(continued).*

REFERENCES

[1] S. Bellavia, C. Cartis, N. I. M. Gould, B. Morini, and Ph. L. Toint, *Convergence of a Regularized Euclidean Residual algorithm for nonlinear least-squares*, SIAM J. Num. Anal., 48 (2010), pp. 1–29.

[2] S. Bellavia, V. De Simone, D. di Serafino, and B. Morini, *Efficient preconditioner updates for shifted linear systems*, SIAM J. Sci. Comput., 33 (2011), pp. 1785–1809.

[3] S. Bellavia, J. Gondzio, and B. Morini, *Regularization and preconditioning of KKT systems arising in nonnegative least-squares problems*, Numer. Linear Algebra Appl., 16, pp. 39-61, 2009.

[4] S. Bellavia, M. Macconi, and B. Morini, *An interior Newton-like method for nonnegative least-squares problems with degenerate solution*, Numer. Linear Algebra Appl., 13 (2006), pp. 825-846.

[5] M. Benzi and D. Bertaccini, *Approximate inverse preconditioning for shifted linear systems*, BIT, 43 (2003), pp. 231–244.

[6] D. Bertaccini, *Efficient preconditioning for sequences of parametric complex symmetric linear systems*, ETNA, 18 (2004), pp. 49–64.

[7] C. Cartis, N. I. M. Gould, and Ph. L. Toint, *Trust-region and other regularisations of linear least-squares problems*, BIT, 49 (2009), pp. 21–53.

[8] C. Cartis, N.I.M. Gould, and Ph. L. Toint, *Adaptive cubic overestimation methods for unconstrained optimization. Part I: motivation, convergence and numerical results*, Math. Program. A, 127 (2011), pp. 245–295.

[9] R. Conn, N. Gould, and Ph. L. Toint, *Trust-Region Methods*, SIAM, Philadelphia, USA, 2000.

[10] T. F. Coleman and Y. Li, *A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables*, SIAM J. Optim., 6 (1996), pp. 1040–1058.

[11] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Prog., 91 (2002), pp. 201–213.

[12] S. C. Eisenstat and H. F. Walker, *Choosing the forcing term in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.

[13] J. B. Erway and P. E. Gill, *A subspace minimization method for the trust-region step*, SIAM J. Optim. 20 (2009), pp. 1439–1461.

[14] J. B. Erway, P. E. Gill, and J. D. Griffin, *Iterative methods for finding a trust-region step* SIAM J. Optim. 20 (2009), pp. 1110-1131.

[15] G. H. Golub and C. F. Van Loan, *Matrix computations*, third edition, The Johns Hopkins University Press, 1996.

[16] N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint, *Solving the trust-region subproblem using the Lanczos method*, SIAM J. Optim. 9 (1999), pp. 504-525.

[17] N. I. M. Gould, D. Orban, and Ph. L. Toint, *CUTEr and SifDec: A constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394.

[18] N. I. M. Gould, M. Porcelli, and Ph. L. Toint, *Updating the regularization parameter in the adaptive cubic regularization algorithm*, Comput. Optim. Appl. (2011), DOI: 10.1007/s10589-011-9446-7.

[19] E. F. Kaasschieter, *Preconditioned conjugate gradients for solving singular systems*, J. Comput. Appl. Math., 24 (1988) pp. 265–275.

[20] F. Lemeire, *Bounds for condition numbers of triangular and trapezoid matrices*, BIT, 15 (1975), pp. 58–64.

[21] G. Meurant, *On the incomplete Cholesky decomposition of a class of perturbed matrices*, SIAM J. Sci. Comput., 23 (2001), pp. 419–429.

[22] J. J. Moré and D. C. Sorensen, *Computing a trust-region step*, SIAM J. Sci. Stat. Comput., 4 (1983), pp. 553–572.

[23] Y. Nesterov, *Modified Gauss-Newton scheme with worst-case guarantees for global performance*, Optim. Methods Softw., 22 (2007), pp. 469–483.

[24] M. Pernice and H. F. Walker, *NITSOL: a new iterative solver for nonlinear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 302–318.

[25] A. van der Sluis and H. A. van der Vorst, *The Rate of Convergence of Conjugate Gradients*, Numer. Math., 48 (1986), pp. 543–560.