An incomplete Cholesky preconditioner based on orthogonal approximations

Artem Napov

Service de Métrologie Nucléaire
Université Libre de Bruxelles (C.P. 165-84),
50 Av. F.D. Roosevelt, B-1050 Brussels, Belgium.

Report GANMN 21-02

December 2021
Revised June 2022;
October 2022 (minor revision)

Abstract

We consider an incomplete Cholesky factorization preconditioner for the iterative solution of large sparse symmetric positive definite (SPD) systems of linear equations. The preconditioner exploits the numerical rank deficiency of some off-diagonal blocks of the Cholesky factor. As a distinctive feature, the approximations performed during the factorization procedure are orthogonal, and therefore the preconditioner falls within the framework introduced in [A. Napov, *SIAM J. Matrix Anal. Appl.*, 34(2013), pp.1148–1173]. This implies that the incomplete factorization procedure is breakdown-free, and that the resulting preconditioner is SPD. The aforementioned reference also gives some upper bounds on the spectral condition number of the preconditioned system based on the accuracy of individual approximations. The most accurate among these bounds is extended here to the considered preconditioner. On the practical side, we present and study an implementation of the preconditioner. It exploits the block sparsity structure as induced by nested dissection block partitioning, and identifies blocks with low numerical rank based on the sparsity pattern of the system matrix. The performance is assessed based on model PDE discretizations and, further, based on linear systems whose matrices correspond to large enough SPD matrices from the SuiteSparse matrix collection. The reported results are compared with those of some other solvers, including the SPD version of ILUPACK solver.

Key words. preconditioner, sparse incomplete Cholesky, breakdown-free incomplete Cholesky, low-rank approximations, orthogonal approximations, conditioning analysis

AMS subject classification. 65F08, 65F35, 65F50

1 Introduction

We study an incomplete Cholesky factorization preconditioner for the iterative solution of large sparse symmetric positive definite (SPD) $N \times N$ systems of linear equations

$$A\mathbf{u} = \mathbf{b}\,. \tag{1.1}$$

The preconditioner exploits the numerical rank deficiency of given off-diagonal blocks of the Cholesky factor, and therefore belongs to the family of data-sparse incomplete factorizations. As a distinctive feature, the approximations performed during the factorization procedure are orthogonal [25].

Cholesky factorization preconditioners based on orthogonal approximations have several attractive properties. First, the corresponding factorization procedure is breakdown-free, meaning that it produces an approximate factor for any value of the approximation accuracy threshold. This is of importance since for some problems the preconditioners yielding the fastest overall solution time are obtained with relatively high threshold values; these preconditioners then also require comparably less memory. Second, the resulting preconditioner is SPD, which makes it suitable for the conjugate gradient iteration [3, 36, 35]. Third, the spectral condition number of the preconditioned system, and hence the number of conjugate gradient iterations needed for convergence, can be bounded above as a function of accuracy of individual approximations. This result is a sign of robustness of the considered preconditioner and provides some guidance for its algorithmic design.

Regarding the upper bounds on the condition number, we show in this work that the most accurate among the bounds presented in [25], which is also the one known to be tight, can be extended to the considered preconditioner. More specifically, the bounds presented in [25] depend on the set of rows affected by the individual approximations at every step, and the most accurate bound is obtained for a relatively constraining one-level approximation pattern. Here we rely on an additional assumption, which is naturally satisfied in our setting, to extend the one-level bound to the preconditioner under consideration.

Considering the practical aspects, the preconditioner is based on a competitive direct solver: a variant of sparse block Cholesky factorization combined with the nested dissection block sparsity structure [14, 23]. Such a combination yields the Cholesky factor with dense and typically large off-diagonal blocks, making the use of low-rank approximations suitable. The considered orthogonal low-rank approximation scheme is of hierarchical nature, which further allows to exploit more efficiently the rank deficiency of some off-diagonal blocks of the factor. The rank deficient blocks are in turn identified within nested dissection blocks by using an algebraic procedure that relies on the sparsity structure of the system matrix [27].

The performance of the preconditioner is assessed through numerical experiments. The considered set of test problems is composed of some discretizations of model PDE problems, and of systems whose matrices correspond to large enough SPD matrices from the SuiteSparse matrix collection [11]. The experiments are performed with a sequential and an OpenMP version of the preconditioner [26], and the results are compared, among others, with the SPD version of ILUPACK solver [6, 7]. ILUPACK solver is considered here

since it is a state-of-the-art incomplete factorization preconditioner based on the dropping of individual entries – a reference approximation scheme for incomplete factorization preconditioning.

The results of the experiments with the PDE-based problems indicate that the preconditioner is competitive, except for the 3D Poisson and similar problems, for which ILUPACK solver is faster. This may sound disappointing at first, but incomplete factorization preconditioners in general do not represent a reference approach for Poisson and similar problems and are superseded, for instance, by algebraic multigrid methods [34, 37, 29]. The experiments with SuiteSparse test problems further demonstrate that the preconditioner is robust, spending a comparable time per nonzero entry of the system matrix for most SuiteSparse problems. Its overall solution time is competitive compared to the other solvers considered in this comparison.

The preconditioner is of general purpose, and its construction only requires the system matrix and an approximation tolerance. In this regard, it contributes to a larger effort of developing general purpose solvers exploiting numerical rank deficiency in sparse factorization methods. Many such solvers have a design similar to what is considered here: they rely on a nested dissection block sparsity structure and exploit a low-rank representation for some large dense blocks of the factor. This approach is adopted, for instance, in [17, 18] in combination with the $\mathcal{H}$-matrix representation, in [38, 40, 27, 10] together with the (sometimes implicit) hierarchical semi-separable (HSS) representation, whereas in [39, 33, 15, 41] the use of multifrontal sparse factorization [13, 24] further enables data-sparse representation of update matrices, and in [1, 31] block low-rank representation is adopted. Another family of preconditioners, see [12, 42] and the references therein, uses low-rank updates for given approximations of the inverse of the system matrix or its Schur complements. Eventually, the method in [32] relies on the low-rank representation expressed as an extended sparse matrix.

Now, like the preconditioner considered here, some preconditioners in [38, 10] are also based on a sparse Cholesky factorization, also rely on orthogonal approximations as defined in [25], and are also breakdown-free. It is therefore natural to wonder if the conditioning analysis presented here applies to these preconditioners as well. Regarding first the approach in [38], it does not fit into the framework introduced in [25] as it performs additional approximations on the Cholesky factor; however, a variant of the method without additional approximations is covered by the presented analysis, and therefore the one-level bound applies to it. Regarding the variant of the preconditioner in [10] which is based on orthogonal approximations, the analysis applies to it without additional adaptations.

A peculiar feature of orthogonal approximations is that they are commonly implemented by truncating an orthogonal factorization, such as rank-revealing QR or SVD, of a *whole* off-diagonal block row of the Cholesky factor. This limitation prevents from using orthogonal approximations to build $\mathcal{H}$-matrix and $\mathcal{H}^2$-matrix representations [4, 20, 8] other than those based on a weak admissibility condition [19]. On the other hand, it seems unclear how orthogonal approximations [25] can be obtained by approximating only a portion of the block row of the Cholesky factor.

The reminder of the paper is organized as follows. In Section 2 we provide a high-level

description of the preconditioner. In Section 3 we show that the considered preconditioner fits into the framework introduced in [25] and extend the one-level bound from this reference to the preconditioner under consideration. Section 4 is dedicated to the numerical experiments. Conclusions are drawn in Section 5.

Notation

For any integers i and $j \geq i$, $i:j = \{i, i+1, \ldots, j\}$ represents the ordered set of integers ranging from i to j. I_ℓ stands for a $\ell \times \ell$ identity matrix and $O_{\ell \times m}$ for a $\ell \times m$ zero matrix (i.e., a matrix with all entries being zero). For any vector $\mathbf{v}$, $\|\mathbf{v}\|$ is its Euclidean norm. For any matrix C, the induced Euclidean matrix norm is

$$\|C\| \;=\; \max_{\mathbf{v}\neq\mathbf{0}} \frac{\|C\mathbf{v}\|}{\|\mathbf{v}\|}\,.$$

For any SPD matrix D, $\lambda_{\max}(D)$ and $\lambda_{\min}(D)$ are, respectively, its largest and its smallest eigenvalue, both eigenvalues being real and positive; the spectral condition number $\kappa(D) = \lambda_{\max}(D)/\lambda_{\min}(D)$ is then well defined. For any $n \times n$ block matrix $E = (E_{i,j})$ and any block indices i, j and k such that $1 \leq i \leq n$, $1 \leq j \leq k \leq n$,

$$E_{i,j:k} \;=\; \begin{pmatrix} E_{i,j} & \cdots & E_{i,k} \end{pmatrix},$$

and, for any m such that $i \leq m \leq n$,

$$E_{i:m,j:k} \;=\; \begin{pmatrix} E_{i,j:k}^T & \cdots & E_{m,j:k}^T \end{pmatrix}^T.$$

2 Preconditioner description

Here we describe the considered incomplete Cholesky preconditioner. We start by introducing in Section 2.1 the orthogonal low-rank approximations and their use in the Cholesky factorization. The setting is deliberately general, as the material of this subsection forms the basis for the analysis in Section 3. The considerations of sparsity are further treated in Section 2.2 in the context of nested dissection block partitioning. Next, in Section 2.3 we further specify the approximation scheme used here, which is of hierarchical type, and briefly summarize the associated algebraic procedure for the identification of rank deficient blocks; more details on this procedure can be found in [27]. Eventually, some complexity estimates for the considered preconditioner in a model setting are provided in Section 2.4.

2.1 Orthogonal low-rank approximations

In what follows we first describe the exact Cholesky factorization method on which the considered preconditioner is based, then present the orthogonal low-rank approximation, and eventually explain how both of these are combined to yield the considered preconditioner.

The considered exact Cholesky factorization method takes as input an SPD matrix A partitioned into a $n \times n$ block form

$$A \;=\; \begin{pmatrix} A_{1,1} & \dots & A_{1,n} \\ \vdots & \ddots & \vdots \\ A_{1,n}^T & \dots & A_{n,n} \end{pmatrix}, \tag{2.1}$$

where each block $A_{i,j}$ is a $m_i \times m_j$ matrix, $i, j = 1, \dots, n$. On the output, it returns an upper triangular factor

$$R \;=\; \begin{pmatrix} R_{1,1} & \dots & R_{1,n} \\ & \ddots & \vdots \\ & & R_{n,n} \end{pmatrix} \tag{2.2}$$

with the same block partitioning as A and such that

$$A \;=\; R^T R\,.$$

Note that the diagonal blocks $R_{i,i}$ in (2.2) are $m_i \times m_i$ upper triangular and, for $i > j$, there holds $R_{i,j} = O_{m_i \times m_j}$, $i, j = 1, \dots, n$.

We consider more specifically a block version of the method, which computes one block row at a time[1]. Assuming that the first $i - 1 < n$ block rows of R have already been computed, one has

$$A \;=\; \begin{pmatrix} R_{1:i-1,1:i-1}^T & \\ R_{1:i-1,i:n}^T & S_A \end{pmatrix} \begin{pmatrix} R_{1:i-1,1:i-1} & R_{1:i-1,i:n} \\ & I_{\underline{m}_i} \end{pmatrix} \tag{2.3}$$

with

$$S_A \;=\; A_{i:n,i:n} - R_{1:i-1,i:n}^T R_{1:i-1,i:n} \tag{2.4}$$

and $\underline{m}_i = \sum_{j=i}^n m_j$. The next block row $R_{i,i:n}$ of the factor is then computed by noting that, for $1 < i < n$, it satisfies

$$R_{i,i}^T \begin{pmatrix} R_{i,i} & R_{i,i+1:n} \end{pmatrix} \;=\; A_{i,i:n} - R_{1:i-1,i}^T R_{1:i-1,i:n}\,, \tag{2.5}$$

the expressions for $i = 1$ and $i = n$ being similar. This is typically done in three stages:

- *update* : compute[2] $C_{i,i:n} = A_{i,i:n} - R_{1:i-1,i}^T R_{1:i-1,i:n}$, with $C_{1,1:n} = A_{1,1:n}$;
- *factorize* : compute upper triangular $R_{i,i}$ such that $R_{i,i}^T R_{i,i} = C_{i,i}$,
- *solve* : if $i < n$, solve $R_{i,i}^T R_{i,i+1:n} \;=\; C_{i,i+1:n}$ for $R_{i,i+1:n}$.

[1] This version is equivalent to the block left-looking Cholesky factorization method for the construction of R^T.

[2] Block matrix C is used here for clarity; the computation can be performed in place, without the need of extra storage.

Note that computing one block row at a time is suitable in view of an incomplete factorization method, since this row can subsequently be approximated, and the resulting approximation further used when computing the following rows.

The key ingredient of the considered incomplete Cholesky factorization preconditioner are orthogonal low-rank approximations. A low-rank approximation of a given $m \times p$ matrix T is a couple $(Q, \widetilde{T})$ of matrices such that Q is $m \times r$, $\widetilde{T}$ is $r \times p$, and $r \leq \min(m, p)$. Here, r is the approximation rank and $T - Q\widetilde{T}$ the approximation error. The product $Q\widetilde{T}$ of the matrices is sometimes also referred to as a low-rank approximation of T.

The approximation $(Q, \widetilde{T})$ of T is said to be *orthogonal* if $Q^T Q = I_r$ and

$$Q^T \left(T - Q\widetilde{T} \right) \;=\; O_{r \times p} \,. \tag{2.6}$$

The equality (2.6) implies in particular that the low-rank approximation $Q\widetilde{T}$ is orthogonal to the approximation error $T - Q\widetilde{T}$. In what follows, we only consider low-rank approximations which are orthogonal.

An orthogonal low-rank approximation can be obtained by truncating an orthogonal factorization, such as rank-revealing QR or SVD. For instance, if a rank-revealing QR factorization of a $m \times p$ matrix T is given by

$$T \;=\; \overline{Q}\,\overline{R} \;=\; \begin{pmatrix} \overline{Q}_1 & \overline{Q}_2 \end{pmatrix} \begin{pmatrix} \overline{R}_1 \\ \overline{R}_2 \end{pmatrix} \;=\; \overline{Q}_1 \overline{R}_1 + \overline{Q}_2 \overline{R}_2 \,,$$

where $\overline{Q}_1$ are the first r columns of $\overline{Q}$ and $\overline{R}_1$ are the first r rows of $\overline{R}$, then $(\overline{Q}_1, \overline{R}_1)$ is an orthogonal low-rank approximation of T. The orthogonality of such an approximation stems from the orthogonality of $\overline{Q}$; more specifically, the relation (2.6) follows from $\overline{Q}_1^T \overline{Q}_2 = O_{r \times (m-r)}$, whereas there also holds $\overline{Q}_1^T \overline{Q}_1 = I_r$. Regarding the approximation rank r, it is typically chosen so that the Euclidean norm of the approximation error satisfies, sometimes up to a factor,

$$\| \, \overline{Q}_2 \overline{R}_2 \, \| \;=\; \| \, \overline{R}_2 \, \| \;\leq\; \varepsilon,$$

where ε is a given approximation tolerance.

Regarding now the construction of the considered incomplete Cholesky preconditioner, it proceeds by interleaving factorization and approximation steps, the first step being a factorization one. During the factorization step, a new block row is added to the factor, whereas during the approximation step, some rows of the off-diagonal block of the already computed part of the factor undergo a low-rank approximation. After $s = (i-1) + (k-1)$ steps, of which $i-1$ and $k-1$ being, respectively, the number of factorization and approximation steps, the factor R has the following form

$$\begin{pmatrix} R_{1:i-1,1:i-1} & Q^{(s)} T^{(s)}_{i:n} \end{pmatrix} \tag{2.7}$$

where $R_{1:i-1,1:i-1}$ is upper triangular, $Q^{(s)}$ is such that ${Q^{(s)}}^T Q^{(s)} = I_{r^{(s)}}$, where $r^{(s)}$ is the number of columns of $Q^{(s)}$, and $T^{(s)}_{i:n}$ has the same number of columns (and, therefore, the

same block column structure) as the block columns $i:n$ of A. In what follows, we show by induction that the above also holds for $s+1$, the base case corresponding to $s=1$, $i-1=1$, $k-1=0$, $T_{2:n}^{(1)}=R_{1,2:n}$ and $Q^{(1)}=I_{m_1}$. We also note that in practice, matrices $R_{1:i-1,1:i-1}$ (except for $i-1=1$) and $Q^{(s)}$ do not have to be formed explicitly; at this stage, they are no longer needed for the construction of the factor, and only their product (or the product of their transpose) with a vector is required for the preconditioner application.

If step $s+1$ is a factorization step, the next block row $R_{i,i:n}$, $i \leq n$, is added to the factor. For $1 < i < n$ the block row is computed based on

$$R_{i,i}^T \begin{pmatrix} R_{i,i} & R_{i,i+1:n} \end{pmatrix} = A_{i,i:n} - T_i^{(s)\,T} T_{i:n}^{(s)}, \tag{2.8}$$

the expressions for $i=1$ and $i=n$ being similar; this is done in the same way as for the exact Cholesky factorization based on (2.5), using update, factorization and solution stages. Taking into account that $Q^{(s)\,T} Q^{(s)} = I_{r^{(s)}}$, the above displayed equality is nothing but (2.5) in which the off-diagonal block $R_{1:i-1,i:n}$ of the exact factor is replaced by that $Q^{(s)}T_{i:n}^{(s)}$ of the approximate one. At the end of step $s+1$, the available rows of the factor are, if $i < n$,

$$\begin{pmatrix} R_{1:i-1,1:i-1} & Q^{(s)}T_{i:n}^{(s)} \\ & R_{i,i:n} \end{pmatrix} = \begin{pmatrix} R_{1:i,1:i} & Q^{(s+1)}T_{i+1:n}^{(s+1)} \end{pmatrix}, \tag{2.9}$$

with hence

$$\begin{aligned} R_{1:i,1:i} &= \begin{pmatrix} R_{1:i-1,1:i-1} & Q^{(s)}T_i^{(s)} \\ & R_{i,i} \end{pmatrix}, \quad Q^{(s+1)} = \begin{pmatrix} Q^{(s)} & \\ & I_{m_i} \end{pmatrix}, \\ T_{i+1:n}^{(s+1)} &= \begin{pmatrix} T_{i+1:n}^{(s)} \\ R_{i,i+1:n} \end{pmatrix}; \end{aligned}$$

as a result, $R_{1:i,1:i}$ is upper triangular and there holds $Q^{(s+1)\,T} Q^{(s+1)} = I_{r^{(s+1)}}$. If $i=n$, there rightmost block in the right hand side of (2.9) vanishes, and the factorization is completed.

Otherwise, step $s+1$ is an approximation step, meaning that the block $T_{i:n}^{(s)}$ undergoes an orthogonal low-rank approximation, which we denote with respect to $T_{i:n}^{(s)}$ as $(Q, T_{i:n}^{(s+1)})$. The first i block rows of the factor then become

$$\begin{pmatrix} R_{1:i-1,1:i-1} & Q^{(s)}(QT_{i:n}^{(s+1)}) \end{pmatrix} = \begin{pmatrix} R_{1:i-1,1:i-1} & Q^{(s+1)}T_{i:n}^{(s+1)} \end{pmatrix}$$

with $Q^{(s+1)} = Q^{(s)}Q$; the relation $Q^{(s+1)\,T} Q^{(s+1)} = I_{r^{(s+1)}}$ follows directly from $Q^TQ = I_r$, where r is the number of columns of Q. The Euclidean norm of the error then further satisfy

$$\|Q^{(s)}T_{i:n}^{(s)} - Q^{(s)}(QT_{i:n}^{(s+1)})\| = \|T_{i:n}^{(s)} - QT_{i:n}^{(s+1)}\|, \tag{2.10}$$

which means that although the orthogonal low-rank approximation is based only on $T_{i:n}^{(s)}$, the resulting approximation error is also that of the off-diagonal block $Q^{(s)}T_{i:n}^{(s)}$ of the factor.

(a) (b) (c)

Figure 1: An example of two possible recursive subdivisions of nested dissection: connectivity graph and the separators (a) as well as the associated block sparsity structure of the system matrix (b) and Cholesky factor (c); big •, midsize • and small • markers represent first, second and third level separators, respectively.

2.2 Sparsity

An important element of the considered preconditioner is the block partitioning (2.1) of the system matrix. When speaking of block partitioning, we mean both the actual partitioning and the symmetric permutation applied to the system matrix in order to bring together (i.e., make contiguous) indices of the same block. Here we use a *coarse* block partitioning based on nested dissection, and further *refine* it for large enough coarse blocks. The nested dissection partitioning, described in this subsection, aims at giving the system matrix a specific block sparsity structure which, on one hand, can be exploited in much the same way as by exact Cholesky factorization methods and, on the other hand, is not altered by orthogonal low-rank approximations. The refinement for large enough blocks is performed in a way that enforces the numerical rank deficiency of the off-diagonal part of the resulting subblocks; this is further exploited in the considered low-rank approximation scheme, as described in Section 2.3.

Nested dissection block partitioning is based on a recursive subdivision of the connectivity graph of a sparse matrix. For a symmetric matrix $A = (a_{ij})$, the connectivity graph is an undirected graph whose vertices are row/column indices of the matrix and such that two vertices i and j are connected by an edge if and only if a_{ij} is nonzero (or treated as such). The set of vertices of the connectivity graph is recursively subdivided into three subsets, two of which are disconnected (i.e., have no edge connecting them); the subdivision is then applied again to each disconnected subset, if this subset is not small enough. The no longer subdivided vertex subsets are called separators, and the set of all separators induce the resulting coarse block partitioning of the matrix. An example illustrating a possible outcome of two recursive nested dissection subdivisions is given in Figure 1(a)-(b), where Figure 1(a) represents the considered connectivity graph and the possible separators, whereas Figure 1(b) gives the induced block sparsity structure (with zero blocks in white). Individual subdivisions typically aim at maximizing the sum of the

sizes of the two disconnected subsets while keeping the two sizes roughly equal. Nested dissection partitioning can be computed with the help of software packages like Metis [21], mt-Metis [22] or Scotch [30].

Using nested dissection block partitioning for exact and incomplete Cholesky factorizations has several attractive properties. First, the Cholesky factor inherits the same block sparsity structure as the upper triangular part of the original matrix, although the blocks are mostly dense in the case of the factor; see Figures 1(b) and (c) for an illustration. This allows to limit the number of nonzero entries in the factors for both exact and incomplete factorizations. Moreover, the block partitioning possess an inherent parallelism, as the computation of block rows corresponding to the two disconnected subsets of every nested dissection subdivision can be performed independently of each other. Further, for an exact Cholesky factorization, the nested dissection block sparsity allows for a use of block (BLAS3) matrix operations. As a consequence, variants of sparse exact block Cholesky factorization method based on nested dissection block subdivision represent good sparse direct solvers in their own right. As of incomplete factorization based on low-rank approximations, an extra advantage is the presence of relatively large dense off-diagonal blocks, which are suitable for low-rank approximations.

On the practical side, the factors of both exact and incomplete Cholesky factorizations are represented here as a collection of coarse block rows. Further, each row is stored as a single (implicit or explicit) dense matrix, and the zero blocks are therefore ignored. A representation of a single coarse block row is given in Figure 2(a); a similar row with zero blocks is highlighted, for instance, in Figure 1(c).

Note that alternative coarse block partitioning strategies may be worth of a separate investigation. A possible motivation for such strategies is the observation from [5, Section 3.3] that the use of nested dissection ordering in combination with incomplete factorization preconditioners based on dropping of individual entries can slow down convergence. Of course, since the considered preconditioner does not rely on such a dropping, this motivation is indirect. In the present work we do not discuss alternative coarse block partitioning strategies any further.

2.3 Hierarchical approximations

For the considered incomplete Cholesky factorization, large enough coarse block rows of the factor are approximated using a hierarchical low-rank approximation scheme. More specifically, the relevant coarse block row is recursively subdivided into two block rows, and the subdivision stops once the number of rows in the block comes close to a target value η. Each subdivision of a block row into two sub-blocks implies that a low-rank approximation is to be computed for each sub-block separately before it is computed for the whole block. Note that for the present description the actual implementation of the orthogonal low-rank approximation scheme is not important; the details of the scheme consider for the numerical experiments are given in Section 4.1.

Approximation step for a given block row occurs as soon as all the rows of the block are computed. More precisely, if a block row is no longer subdivided into sub-blocks,

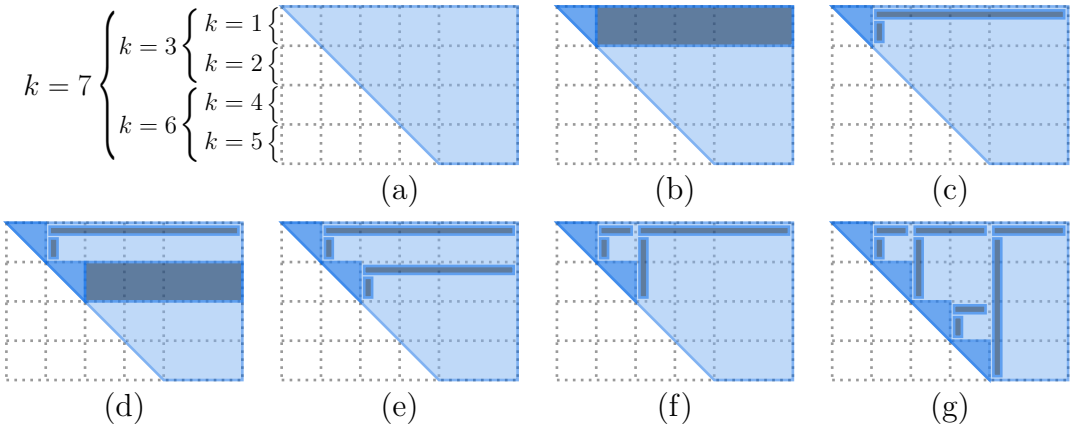

Figure 2: Two steps of recursive subdivision of a coarse block row (a), as well as the representation of some steps of the considered hierarchical approximation scheme: first factorization step (b), first approximation step (c), second factorization step (d), second approximation step (e), third approximation step (f), and the final block row after 4 factorization and 7 approximation steps (g). The indices k of the approximation steps are alos depicted on the representation (a) of the coarse block row, each index being to the left of the curly bracket delimiting the subblock row modified during the approximation step.

the corresponding factorization step is followed immediately by an approximation step restricted to this block row; see Figures 2(b)-(c) and 2(d)-(e) for an illustration. If a block row is subdivided in two sub-blocks, then the approximation step for the block row is performed once the approximation steps for the two sub-blocks are complete; see, for example, Figures 2(f) and 2(g). The advantage of this early compression strategy is that the update stage for every new block row of the factor, as based on (2.8), benefits from the matrix $T_{i:n}^{(s)}$ which incorporates all relevant already computed low-rank approximations.

Eventually we note that, although only a subset of rows of the block $T_{i:n}^{(s)}$ undergoes an orthogonal approximation at every approximation step while the other rows remain unchanged, the resulting approximation for the whole block is also orthogonal, and the Euclidean norm of the approximation error is the same in both cases.

Now, each block row is chosen with the aim to enforce the numerical rank deficiency of its off-diagonal part. More specifically, we use the heuristic approach for the partitioning of the coarse block rows from [27], which is based solely on the sparsity pattern of the system matrix A. The main idea is to put in each block the rows whose vertices in the connectivity graph $\mathcal{G}$ of A are close to each other in the sense specified below. To this end, for the set $\mathcal{S}$ of indices of each large enough coarse block row, we build an enhanced connectivity graph $\mathcal{G}_{\mathcal{S}}$ whose set of vertices is $\mathcal{S}$, and whose set of edges corresponds to couples $(i, j) \in \mathcal{S} \times \mathcal{S}$ such that either (i, j) is an edge of $\mathcal{G}$, or there exists a vertex $p \notin \mathcal{S}$ such that (i, p) and (p, j) are edges of $\mathcal{G}$; the vertices connected by an edge of $\mathcal{G}_{\mathcal{S}}$ are considered close to each other. The rationale behind the use of an enhanced connectivity

graph $\mathcal{G}_{\mathcal{S}}$ is that it is typically less fragmented (i.e., has fewer connected components) than the sub-graph of $\mathcal{G}$ induced by $\mathcal{S}$, allowing for a better assessment of the closeness of the vertices, whereas it is typically as sparse as this latter.

The recursive subdivision of a coarse block row is therefore preformed by recursively subdividing the corresponding set $\mathcal{S}$ of vertices into two subsets so as to reduce the number of connections between the subsets in the enhanced connectivity graph $\mathcal{G}_{\mathcal{S}}$; the subdivision is performed using the routine `PartGraphRecursive` from Metis [21] and stops when the vertex set size comes close to the target value η.

2.4 Complexity

Here we provide some complexity estimates for the considered preconditioner in a model setting, which is typical for the discretizations of scalar elliptic two- and three-dimensional (2D and 3D) PDEs defined on, respectively, square (2D) and cube (3D) domains, and discretized on a Cartesian grid. More specifically, regarding the nested dissection coarse block partitioning, we assume a geometric nested dissection ordering inspired by [14] in 2D (which for a $N \times N$ system matrix yields 1 separator with $N^{1/2}$ vertices after the first recursive subdivision, 2 separators with $N^{1/2}/2$ vertices after the second subdivision, 4 separators with $N^{1/2}/2$ vertices after the third subdivision, etc.) and a natural extension of this 2D ordering in 3D (which yields 1 separator with $N^{2/3}$ vertices after the first recursive subdivision, 2 separators with $N^{2/3}/2$ vertices after the second subdivision, 4 separators with $N^{2/3}/4$ vertices after the third subdivision, 8 separators with $N^{2/3}/4$ vertices after the fourth subdivision, etc.). Moreover, we assume that each coarse block is further recursively subdivided into sub-blocks of even size till the number of its indices comes close to the target value η, and that the off-diagonal rank of each block of m rows considered in the recursive subdivision (and not only the resulting sub-blocks) has a rank at most r_0 in 2D, and at most $r_0\sqrt{m}$ in 3D. Further, we assume that the number of floating point operations (flops) needed to compute the low-rank approximation $(Q, \widetilde{T})$ of a $m \times p$ matrix T of rank r is $c_0 mpr$ (with $c_0 = 4$ for the the rank revealing QR factorization with column pivoting considered for the numerical experiments in Section 4), that the product of Q and Q^T with a vector costs at most $c_1 mr$ flops (with $c_1 = 2$ if Q is stored explicitly as a matrix, and $c_1 = 4$ if only the vectors needed for the Householder transformations are stored).

Under these assumptions, and further assuming that the implementation of the preconditioner is similar to what is done in [26], the upper bounds on the number of flops needed to compute the considered incomplete Cholesky factorization preconditioner as well as the upper bound on the number of flops needed for a single application of this preconditioner are given in Table 1, along with the number of flops for the corresponding exact Cholesky factorization (for which the application actually amounts to the solution of the linear system). Note that the upper bounds on the number of flops needed for the application also correspond (up to a factor) to the upper bounds on the amount of memory needed for the corresponding Cholesky factor.

	2D		3D	
	compute	apply/store	compute	apply/store
incomplete Cholesky	$\mathcal{O}(N \log^2 N)$	$\mathcal{O}(N)$	$\mathcal{O}(N^{5/3})$	$\mathcal{O}(N \log N)$
exact Cholesky	$\mathcal{O}(N^{3/2})$	$\mathcal{O}(N \log(N))$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^{4/3})$

Table 1: Upper bounds on the number of floating point operations for computing and applying the considered incomplete Cholesky factorization preconditioner and the corresponding exact Cholesky factorization; N is the dimension of the system matrix.

3 Analysis

In this section we are concerned with an upper bound on the spectral condition number of the preconditioned system that accounts for the accuracy of the individual approximations. In particular, in Section 3.1 we specify the setting and show that the considered preconditioner fits into the framework introduced in [25]. The upper bound is proved in Section 3.2. Since it coincides with the one-level bound from [25], some related results which are proved in this latter reference, including the tightness of the bound, are further briefly summarized in Section 3.3.

3.1 Preliminaries

The analysis makes use of auxiliary matrices B_k, $k = 0, \ldots, \ell$, where ℓ is the number of approximation steps. Here, B_k is the preconditioner obtained with the incomplete Cholesky factorization, as described in Section 2.1, in which only the first k approximation steps are performed. In particular, $B_0 = A$ corresponds to the exact factorization of the system matrix, whereas $B_\ell = R^T R$ corresponds to the actual preconditioner.

Now, if $i - 1 > 0$ factorization steps are performed before the kth approximation step, it follows from (2.7) that B_{k-1} can be expressed in the following factored form

$$B_{k-1} \;=\; \begin{pmatrix} {\mathcal{R}_{11}^{(k)}}^T & \\ {\mathcal{R}_{12}^{(k)}}^T & S_B^{(k)} \end{pmatrix} \begin{pmatrix} \mathcal{R}_{11}^{(k)} & \mathcal{R}_{12}^{(k)} \\ & I_{\underline{m}_i} \end{pmatrix} . \tag{3.1}$$

were $\mathcal{R}_{11}^{(k)} = R_{1:i-1,1:i-1}$, $\mathcal{R}_{12}^{(k)} = Q^{(s)} T_{i:n}^{(s)}$, ${Q^{(s)}}^T Q^{(s)} = I_{r^{(s)}}$, $s = (i-1) + (k-1)$, and

$$S_B^{(k)} \;=\; A_{i:n,i:n} - {\mathcal{R}_{12}^{(k)}}^T \mathcal{R}_{12}^{(k)} \; . \tag{3.2}$$

This latter expression stems from the fact that approximation steps from k till ℓ are dropped, and hence the factorization after step s is exact.

Next, B_k is obtained from B_{k-1} by performing an orthogonal low-rank approximation $(Q,\, T_{i:n}^{(s+1)})$ of the matrix $T_{i:n}^{(s)}$, and therefore

$$B_k \;=\; \begin{pmatrix} {\mathcal{R}_{11}^{(k)}}^T & \\ {\widetilde{\mathcal{R}}_{12}^{(k)}}^T & \widetilde{S}_B^{(k)} \end{pmatrix} \begin{pmatrix} \mathcal{R}_{11}^{(k)} & \widetilde{\mathcal{R}}_{12}^{(k)} \\ & I_{\underline{m}_i} \end{pmatrix} , \tag{3.3}$$

where $\widetilde{\mathcal{R}}_{12}^{(k)} = Q^{(s)} Q T_{i:n}^{(s+1)}$, $Q^T Q = I_r$, r is the number of columns of Q, and

$$\widetilde{S}_B^{(k)} \;=\; A_{i:n,i:n} - \widetilde{\mathcal{R}}_{12}^{(k)\,T}\, \widetilde{\mathcal{R}}_{12}^{(k)} \;. \tag{3.4}$$

To show that the considered preconditioner fits into the framework introduced in [25], one needs to check that, for every $k = 1, \ldots, \ell$,

$$\widetilde{\mathcal{R}}_{12}^{(k)\,T} (\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)}) \;=\; T_{i:n}^{(s+1)\,T}\, Q^T \left(T_{i:n}^{(s)} - Q T_{i:n}^{(s+1)} \right) \;=\; O_{\underline{m}_i \times \underline{m}_i} \,. \tag{3.5}$$

Here, the above relation follows directly from the orthogonality (2.6) of the approximation $(Q,\, T_{i:n}^{(s+1)})$ of the matrix $T_{i:n}^{(s)}$. As a consequence, the following result holds.

Lemma 3.1 ([25])**.** *Let A be SPD and partitioned as in (2.1). Let B_k be the incomplete Cholesky factorization preconditioner in which only the first k approximation steps are performed, $k = 0, \ldots, \ell$, as described in Section 2.1. Then*

(a) B_k is SPD, $k = 1, ..., \ell$.

(b) The construction procedure described in Section 2.1 does not break down.

Note that, by construction, the remaining approximation steps $k+1, \ldots, \ell$, may affect the block $\widetilde{\mathcal{R}}_{12}^{(k)} = Q^{(s)} Q T_{i:n}^{(s+1)}$ only by approximating the matrix $T_{i:n}^{(s+1)}$. As a result, the actual preconditioner B_ℓ can also be written as

$$B_\ell \;=\; \begin{pmatrix} \mathcal{R}_{11}^{(k)\,T} & \\ \overline{\mathcal{R}}_{12}^{(k)\,T} & \mathcal{R}_{22}^{(k)\,T} \end{pmatrix} \begin{pmatrix} \mathcal{R}_{11}^{(k)} & \overline{\mathcal{R}}_{12}^{(k)} \\ & \mathcal{R}_{22}^{(k)} \end{pmatrix} \;=:\; R^T R \,, \tag{3.6}$$

with $\overline{\mathcal{R}}_{12}^{(k)} = Q^{(s-1)} Q \overline{T}_{i:n}$ for some $\overline{T}_{i:n}$, and with some $\mathcal{R}_{22}^{(k)}$. The form of the $\overline{\mathcal{R}}_{12}^{(k)}$ block further implies, in the same way as for (3.5),

$$\overline{\mathcal{R}}_{12}^{(k)\,T} (\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)}) \;=\; \overline{T}_{i:n}{}^{T}\, Q^T \left(T_{i:n}^{(s)} - Q T_{i:n}^{(s+1)} \right) \;=\; O_{\underline{m}_i \times \underline{m}_i} \,. \tag{3.7}$$

Note that this is an additional property of the considered preconditioner, and does not follow from (3.5).

3.2 Main result

The main result of this section is given in Theorem 3.3. It provides upper and lower bounds on the largest and the smallest eigenvalues of $R^{-T} B_{k-1} R^{-1}$, denoted, respectively, as $\lambda_{\max}^{(k-1)}$ and $\lambda_{\min}^{(k-1)}$. The bounds are expressed as given functions of the extreme eigenvalues $\lambda_{\max}^{(k)}$ and $\lambda_{\min}^{(k)}$ of $R^{-T} B_k R^{-1}$, and of the approximation accuracy

$$\gamma_k \;=\; \left\| \left(\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)} \right) \widetilde{S}_B^{(k)\,-1/2} \right\| \tag{3.8}$$

of the kth approximation step; the matrices $\mathcal{R}_{12}^{(k)}$, $\widetilde{\mathcal{R}}_{12}^{(k)}$ and $\widetilde{S}_B^{(k)}$ are those defined in Section 3.1. Since $\lambda_{\max}^{(\ell)} = \lambda_{\min}^{(\ell)} = 1$, applying Theorem 3.3 for $k = \ell, \ldots, 1$ yields two-sided bounds for $\lambda_{\max}^{(0)} = \lambda_{\max}(R^{-T}AR^{-1})$ and $\lambda_{\min}^{(0)} = \lambda_{\min}(R^{-T}AR^{-1})$ as a function of γ_k $k = 1, \ldots, \ell$, and therefore also provides an upper bound on the condition number

$$\kappa(R^{-T}AR^{-1}) = \frac{\lambda_{\max}(R^{-T}AR^{-1})}{\lambda_{\min}(R^{-T}AR^{-1})} \tag{3.9}$$

of the preconditioned system, which is known to determine an upper bounds on the rate of convergence of the preconditioned conjugate gradient algorithm [3, 36, 35].

The proof of Theorem 3.3 relies on the following lemma.

Lemma 3.2. *Let B_{k-1}, B_k and $B_\ell = R^T R$ be given by* (3.1), (3.3) *and* (3.6), *respectively, for some $\mathcal{R}_{11}^{(k)}$, $\mathcal{R}_{12}^{(k)}$, $S_B^{(k)}$, $\widetilde{\mathcal{R}}_{12}^{(k)}$, $\widetilde{S}_B^{(k)}$, $\overline{\mathcal{R}}_{12}^{(k)}$ and $\mathcal{R}_{22}^{(k)}$ satisfying* (3.2), (3.4) *and* (3.7), *and set*

$$\Delta R^{(k)} \;=\; \begin{pmatrix} O_{\overline{m}_i \times \overline{m}_i} & \mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)} \\ O_{\underline{m}_i \times \overline{m}_i} & O_{\underline{m}_i \times \underline{m}_i} \end{pmatrix}, \tag{3.10}$$

where $\underline{m}_i = \sum_{j=i}^{n} m_j$ and $\overline{m}_i = \sum_{j=1}^{i-1} m_j$. Then

$$B_{k-1} \;=\; B_k + R^T \Delta R^{(k)} + \Delta R^{(k)\,T} R\,.$$

Proof. Note that (3.7) implies

$$\begin{aligned} R^T \Delta R^{(k)} \;&=\; \begin{pmatrix} \mathcal{R}_{11}^{(k)\,T} & \\ \overline{\mathcal{R}}_{12}^{(k)\,T} & * \end{pmatrix} \begin{pmatrix} O_{\overline{m}_i \times \overline{m}_i} & \mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)} \\ O_{\underline{m}_i \times \overline{m}_i} & O_{\underline{m}_i \times \underline{m}_i} \end{pmatrix} \\ &=\; \begin{pmatrix} O_{\overline{m}_i \times \overline{m}_i} & \mathcal{R}_{11}^{(k)\,T} (\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)}) \\ O_{\underline{m}_i \times \overline{m}_i} & O_{\underline{m}_i \times \underline{m}_i} \end{pmatrix}. \end{aligned}$$

The result then follows from

$$B_{k-1} \;=\; B_k + \begin{pmatrix} O_{\overline{m}_i \times \overline{m}_i} & \mathcal{R}_{11}^{(k)\,T} (\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)}) \\ (\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)})^T \mathcal{R}_{11}^{(k)} & O_{\underline{m}_i \times \underline{m}_i} \end{pmatrix}.$$

■ The main result is therefore as follows. Note that the resulting bounds are the same as the one-level bounds in [25]. The key properties are (3.5) and (3.7), and this second property is what enables the extension of the one-level bound to the considered preconditioner (since, as shown in [25], the property (3.5) alone is not sufficient for such an extension).

Theorem 3.3 (main theorem)**.** *Let the assumptions of Lemma 3.2 hold for some B_{k-1}, B_k and $B_\ell = R^T R$ being SPD, $k = 1, \ldots, \ell$, and, moreover, assume that* (3.5) *holds. Let $\lambda_{\max}^{(k)}$ and $\lambda_{\min}^{(k)}$ be the largest and the smallest eigenvalue of $R^{-T}B_k R^{-1}$, $k = 0, \ldots, \ell$,*

and define γ_k , $k = 1, \ldots, \ell$, *as in* (3.8) . *Then* $\gamma_k < 1$ *and, moreover, if* $\lambda_{\max}^{(k)} \geq 1$ *and* $\lambda_{\min}^{(k)} \leq 1$, *there holds*

$$\lambda_{\max}^{(k)} \;\leq\; \lambda_{\max}^{(k-1)} \;\leq\; \lambda_{\max}^{(k)} + g(\,\lambda_{\max}^{(k)}, \gamma_k\,)\,, \tag{3.11}$$

$$\lambda_{\min}^{(k)} \;\geq\; \lambda_{\min}^{(k-1)} \;\geq\; \lambda_{\min}^{(k)} - g(\lambda_{\min}^{(k)}, \gamma_k\,)\,, \tag{3.12}$$

where

$$g(\lambda, \gamma) \;=\; \max_{\beta>0} \frac{2\gamma\beta - |\lambda - 1|\beta^2}{\beta^2 + \lambda^{-1}}\,. \tag{3.13}$$

Proof. First, for any vector $\mathbf{v}$ of compatible dimensions Lemma 3.2 implies

$$\mathbf{v}^T R^{-T} B_{k-1} R^{-1}\mathbf{v} \;=\; \mathbf{v}^T R^{-T} B_k R^{-1}\mathbf{v} + 2\;\mathbf{v}^T \Delta R^{(k)} R^{-1}\mathbf{v}\,. \tag{3.14}$$

Next, we focus on the matrices of the right-hand side. More specifically, since $B_\ell = R^T R$ is SPD, it follows from (3.6) that $\mathcal{R}_{11}^{(k)}$ and $\mathcal{R}_{22}^{(k)}$ are regular and, further, that

$$R^{-1} \;=\; \begin{pmatrix} \mathcal{R}_{11}^{(k)\;-1} & -\,\mathcal{R}_{11}^{(k)\;-1}\,\overline{\mathcal{R}}_{12}^{(k)}\;\mathcal{R}_{22}^{(k)\;-1} \\ & \mathcal{R}_{22}^{(k)\;-1} \end{pmatrix}.$$

As a consequence, (3.3) implies that

$$R^{-T} B_k R^{-1} \;=\; \begin{pmatrix} I_{\overline{m}_i} & \left(\widetilde{\mathcal{R}}_{12}^{(k)} - \overline{\mathcal{R}}_{12}^{(k)}\right)\mathcal{R}_{22}^{(k)\;-1} \\ \mathcal{R}_{22}^{(k)\;-T}\left(\widetilde{\mathcal{R}}_{12}^{(k)} - \overline{\mathcal{R}}_{12}^{(k)}\right)^T & * \end{pmatrix}, \tag{3.15}$$

whereas (3.10) yields

$$\Delta R^{(k)} R^{-1} \;=\; \begin{pmatrix} O_{\overline{m}_i \times \overline{m}_i} & (\,\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)}\,)\,\mathcal{R}_{22}^{(k)\;-1} \\ O_{\underline{m}_i \times \overline{m}_i} & O_{\underline{m}_i \times \underline{m}_i} \end{pmatrix}. \tag{3.16}$$

Our next step is a change of basis transformation to highlight the effect of orthogonal approximations in (3.14). For this, let $[\;\mathcal{V}_\parallel\;,\;\mathcal{V}_\perp\;]$ be an orthogonal matrix such that the columns of $\mathcal{V}_\parallel$ span the same subspace as the columns of $\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)}$. Further, let

$$V_\parallel \;=\; \begin{pmatrix} \mathcal{V}_\parallel \\ \end{pmatrix}, \quad V_\perp \;=\; \begin{pmatrix} \mathcal{V}_\perp & \\ & I_{\underline{m}_i} \end{pmatrix}, \quad V \;=\; \begin{pmatrix} V_\parallel & V_\perp \end{pmatrix}, \quad \mathbf{v}_\parallel \;=\; V_\parallel^T\mathbf{v}\,, \quad \mathbf{v}_\perp \;=\; V_\perp^T\mathbf{v}\,.$$

Note that V is an orthogonal matrix and, hence, that

$$\mathbf{v} \;=\; V_\parallel\mathbf{v}_\parallel + V_\perp\mathbf{v}_\perp\,. \tag{3.17}$$

On the other hand, both (3.5) and (3.7) imply

$$(\,\widetilde{\mathcal{R}}_{12}^{(k)} - \overline{\mathcal{R}}_{12}^{(k)}\,)^T(\,\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)}\,) \;=\; O_{\underline{m}_i \times \underline{m}_i}\,,$$

and, hence,

$$\mathcal{V}_{\|}^T(\widetilde{\mathcal{R}}_{12}^{(k)} - \overline{\mathcal{R}}_{12}^{(k)}) \;=\; O_{r\times \underline{m}_i}\,,$$

where r is the number of columns of $\mathcal{V}_{\|}$.

Now, (3.15) together with the above equation implies

$$V_{\|}^T R^{-T} B_k R^{-1} V_{\perp} \;=\; O_{r\times(\overline{m}_i+\underline{m}_i-r)}\,, \quad V_{\|}^T R^{-T} B_k R^{-1} V_{\|} \;=\; I_r\,. \tag{3.18}$$

Similarly, (3.16) together with $\mathcal{V}_{\perp}^T\left(\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)}\right) = O_{(\overline{m}_i-r)\times\underline{m}_i}$ yields

$$\Delta R^{(k)} R^{-1} V_{\|} \;=\; O_{(\overline{m}_i+\underline{m}_i)\times r} \quad \text{and} \quad V_{\perp}^T \Delta R^{(k)} R^{-1} \;=\; O_{(\overline{m}_i+\underline{m}_i-r)\times(\overline{m}_i+\underline{m}_i)}\,. \tag{3.19}$$

Further, injecting (3.17) into (3.14), and subsequently applying (3.18) and (3.19) entails

$$\begin{aligned}
\frac{\mathbf{v}^T R^{-T} B_{k-1} R^{-1}\mathbf{v}}{\mathbf{v}^T\mathbf{v}} \;&=\; \frac{\mathbf{v}^T R^{-T} B_k R^{-1}\mathbf{v} + 2\;\mathbf{v}^T\Delta R^{(k)} R^{-1}\mathbf{v}}{\mathbf{v}^T\mathbf{v}} \\
&=\; \frac{\mathbf{v}_{\perp}^T V_{\perp}^T R^{-T} B_k R^{-1} V_{\perp}\mathbf{v}_{\perp} + \mathbf{v}_{\|}^T\mathbf{v}_{\|} + 2\;\mathbf{v}_{\|}^T V_{\|}^T \Delta R^{(k)} R^{-1} V_{\perp}\mathbf{v}_{\perp}}{\mathbf{v}_{\|}^T\mathbf{v}_{\|} + \mathbf{v}_{\perp}^T V_{\perp}^T V_{\perp}\mathbf{v}_{\perp}} \\
&=\; \frac{\mathbf{w}^T\mathbf{w} + \mathbf{v}_{\|}^T\mathbf{v}_{\|} + 2\;\mathbf{v}_{\|}^T V_{\|}^T \Delta R^{(k)} B_k^{-1/2}\mathbf{w}}{\mathbf{v}_{\|}^T\mathbf{v}_{\|} + \mathbf{w}^T(B_k^{1/2} R^{-1} R^{-T} B_k^{1/2})^{-1}\mathbf{w}}\;,
\end{aligned} \tag{3.20}$$

where $\mathbf{w} = B_k^{1/2} R^{-1} V_{\perp}\mathbf{v}_{\perp}$. Note that, by Courant-Fischer theorem [3, Lemma 3.13], the extreme eigenvalues $\lambda_{\max}^{(k-1)}$ and $\lambda_{\min}^{(k-1)}$ of $R^{-T} B_{k-1} R^{-1}$ are obtained from any of the above quotients by taking, respectively, the maximum and the minimum of these quotients over all nonzero vectors $\mathbf{v} = V_{\|}\mathbf{v}_{\|} + V_{\perp}\mathbf{v}_{\perp}$. On one hand, setting $\mathbf{v}_{\|} = \mathbf{0}$ before maximization/minimization yields the left inequalities (3.11), (3.12). On the other hand, using (3.10) together with the fact that

$$B_k^{-1} \;=\; \begin{pmatrix} * & * \\ * & \widetilde{S}_B^{(k)\,-1} \end{pmatrix}$$

entails

$$\Delta R^{(k)} B_k^{-1}\,\Delta R^{(k)\,T} \;=\; (\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)})^T\;\widetilde{S}_B^{(k)\,-1}\,(\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)})\,,$$

and, based on (3.8), further implies

$$\widetilde{\mathbf{v}}_{\|}^T \Delta R^{(k)} B_k^{-1/2}\mathbf{w} \;\le\; \gamma_k\;\|\mathbf{v}_{\|}\|\;\|\mathbf{w}\|\,.$$

Inserting this latter inequality as well as

$$\|\mathbf{w}\|^2/\lambda_{\max}^{(k)} \;\le\; \mathbf{w}^T\left(B_k^{1/2} R^{-1} R^{-T} B_k^{1/2}\right)^{-1}\mathbf{w} \;\le\; \|\mathbf{w}\|^2/\lambda_{\min}^{(k)}$$

into (3.20) and setting $\beta = \|\mathbf{w}\|/\|\mathbf{v}_\|\|$ then leads to

$$\begin{aligned}
\lambda_{\max}^{(k-1)} &\leq \max_{\beta \geq 0} \frac{\beta^2 + 1 + 2\gamma_k \beta}{\beta^2 + 1/\lambda_{\max}^{(k)}} \,, \\
\lambda_{\min}^{(k-1)} &\geq \min_{\beta \geq 0} \frac{\beta^2 + 1 - 2\gamma_k \beta}{\beta^2 + 1/\lambda_{\min}^{(k)}} \,,
\end{aligned}$$

which, together with $\lambda_{\max}^{(k)} \geq 1$ and $\lambda_{\min}^{(k)} \leq 1$, yields the right inequalities (3.11), (3.12).

Eventually, the fact that $\gamma_k < 1$ follows from the definition (3.8) of γ_k and the observation, stemming from (3.5), that

$$\widetilde{S}_B^{(k)} - (\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)})^T (\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)}) \;=\; S_B^{(k)}$$

is SPD. ∎

3.3 Related results

Here we briefly summarize the key results related to the one-level bound. The results are taken from [25], and the proofs can be found there.

The first result relates the approximation accuracy parameter γ_k to the approximation error of the off-diagonal block in Euclidean norm. As already noted with (2.10), this approximation error is in turn directly related to the approximation error of the orthogonal low-rank approximations. Hence, using an absolute threshold parameter for the individual orthogonal low-rank approximations allows to uniformly bound γ_k.

Proposition 3.1. *Let the assumptions of Theorem 3.3 hold for $k = 1, \ldots, \ell$, and set $A = B_0$. Let S_A be the exact Schur complement of A as defined by (2.4), (2.3) with i being such that S_A and $\widetilde{S}_B^{(k)}$ have the same dimension. Then*

$$\begin{aligned}
\gamma_k &\leq \left\| (\mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)})\, S_A^{-1/2} \right\| \leq \left\| \mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)} \right\| \left\| S_A^{-1} \right\|^{1/2} \\
&\leq \left\| \mathcal{R}_{12}^{(k)} - \widetilde{\mathcal{R}}_{12}^{(k)} \right\| \left\| A^{-1} \right\|^{1/2} .
\end{aligned}$$

The next result shows that the rightmost bounds (3.11), (3.12) from Theorem 3.3 are tight, and this for any number ℓ of approximation steps and any set of values $\gamma_k < 1$, $k = 1, ..., \ell$, of approximation accuracy. This follows from the fact that a particular one-level scheme, which yields a block diagonal preconditioner, also fits into the present framework of incomplete Cholesky factorization with orthogonal low-rank approximations. More specifically, the block diagonal preconditioner is obtained from the incomplete Cholesky factorization procedure described in Section 2.1 if after every factorization step i there is one approximation step k (with, hence $k = i$) such that $\widetilde{\mathcal{R}}_{12}^{(k)} = O$.

Proposition 3.2. *For any set of positive values $\widetilde{\gamma}_k < 1$, $k = 1, ..., \ell$, there exist an SPD matrix $A = A(\widetilde{\gamma}_1, \cdots, \widetilde{\gamma}_\ell)$ partitioned as in (2.1) such that the application of the incomplete Cholesky factorization procedure described in Section 2.1 to $A(\widetilde{\gamma}_1, \cdots, \widetilde{\gamma}_{\ell-1})$ with one approximation step k corresponding to $\widetilde{\mathcal{R}}_{12}^{(k)} = O$ after each factorization step k, $k = 1, \ldots, n$, returns an upper triangular R, and there holds:*

(a)
$$\gamma_k \;=\; \widetilde{\gamma}_k\,, \qquad \textit{for } k = 1, \ldots, \ell\,,$$
where γ_k is given by (3.8), with $\widetilde{S}_B^{(k)}$ defined by (3.4) and $\mathcal{R}_{12}^{(k)}$, $\widetilde{\mathcal{R}}_{12}^{(k)}$ defined as in Section 2.1.

(b) setting $\lambda_{\max}^{(k)} = \lambda_{\max}(B_\ell^{-1}B_k)$ and $\lambda_{\min}^{(k)} = \lambda_{\min}(B_\ell^{-1}B_k)$, where B_k, $k = 0, \ldots, \ell$, is defined as in Section 2.1, there holds
$$\begin{aligned} \lambda_{\max}^{(k-1)} &= \lambda_{\max}^{(k)} + g(\,\lambda_{\max}^{(k)}, \widetilde{\gamma}_k\,)\,, \\ \lambda_{\min}^{(k-1)} &= \lambda_{\min}^{(k)} - g(\,\lambda_{\min}^{(k)}, \widetilde{\gamma}_k\,)\,, \end{aligned}$$
with g given by (3.13).

Eventually, we informally summarize the asymptotic behavior of the bounds (3.11) and (3.12); a more formal presentation with further details can be found in [25]. Considering first the case $\lambda_{\max}^{(k)} \approx 1$ and $\lambda_{\min}^{(k)} \approx 1$, which arises for k close to ℓ (remember that $\lambda_{\max}^{(\ell)} = \lambda_{\min}^{(\ell)} = 1$), we note that

$$\lambda_{\max}^{(k-1)} \;\approx\; (1+\gamma_k)\lambda_{\max}^{(k)}\,, \quad \lambda_{\min}^{(k-1)} \;\approx\; (1-\gamma_k)\lambda_{\min}^{(k)}\,.$$

On the other hand, for $\lambda_{\max}^{(k)} \gg 1$ and $0 < \lambda_{\min}^{(k)} \ll 1$, which arises later in the process of evaluation of the bounds,

$$\lambda_{\max}^{(k-1)} \;\approx\; \lambda_{\max}^{(k)} + \gamma_k^2\,, \quad \lambda_{\min}^{(k-1)} \;\approx\; (1-\gamma_k^2)\lambda_{\min}^{(k)}\,.$$

The appearance of γ_k^2 instead of γ_k in this latter case impacts the sensitivity of the bound on the value of the threshold parameter for the orthogonal low-rank approximations, since, when γ_k is divided by 10, γ_k^2 is divided by 100.

4 Numerical results

In this section we report on numerical experiments with the considered incomplete Cholesky preconditioner. More specifically, in Section 4.1 we describe the experimental setting and give further details for the preconditioner. The description of, and the results for, model PDE discretizations are given in Section 4.2; the results for the SuiteSparse problems are given in Section 4.3.

4.1 Setting

The considered preconditioner (named `ICo` below, for `I`*ncomplete* `C`*holesky with* `o`*rthogonal approximations*) corresponds to the description of Section 2 with the following details [26]. The preconditioner is implemented in C using double precision arithmetic; multithreaded implementation is based on OpenMP tasks and essentially relies on the parallelism inherited from nested dissection block sparsity. The orthogonal low-rank approximations are preformed using a truncated version of the rank revealing QR factorization. More specifically, we use a modified version of BLAS `DGEQP3` routine [2], which computes the rank revealing QR factorization with column pivoting [16, Section 5.4.1]; the modification ensures that the factorization process stops when the absolute Euclidean norm of the columns of the non-factorized part of the block is below a given (absolute) thresholds ε. Moreover, the low-rank approximation of a given matrix block is only kept if the memory needed for the low-rank representation of the matrix is smaller than the memory required for its dense representation; otherwise the matrix block is not approximated. The coarse block partitioning is performed as described in Section 2.2 based on the node-based nested dissection method implemented in Metis software (version 5.1.0) for the sequential version of `ICo`, and the same method from mt-Metis software (version 0.6.0) for the OpenMP one. The fine block partitioning is performed as described in Section 2.3 (see also [27]), with target block size set to $\eta = 32$. During the solution step, the preconditioner is combined with conjugate gradient iteration.

The performance of the preconditioner is assessed through the comparison with other solvers for SPD linear systems. These include the exact Cholesky factorization described in Section 2.1 (referred to as `CHOL`) and based on which the considered preconditioner is designed, the standalone (i.e., unpreconditioned) conjugate gradient iteration [3, 36, 35] (referred to as `CG`), and the SPD variant of ILUPACK solver [6, 7], version 2.4 (referred to as `ILUPACK`). For the model PDE problems we also consider AGMG algebraic multigrid solver [29, 28], version 3.3.5-aca (referred to as `AGMG`). Note that ILUPACK solver is used with default parameters, except for the ordering, for which Metis node-based nested dissection is used; this latter choice intends to make the comparison less ordering-dependent.

Regarding the experimental setting, the right hand side for all the systems are randomly generated based on the normal distribution $\mathcal{N}(0, 1)$ with a fixed seed. Unless stated otherwise, the experiments with threshold-based solvers, such as `ICo` and `ILUPACK`, are performed choosing the value of the threshold parameter among[3] 1, $10^{\pm 1}$, $10^{\pm 2}$, ..., for which the total time to solution is the smallest; note that threshold values larger than 1 are used for `ICo`, since it relies on an absolute threshold criterion. The iterative solvers (i.e., all solvers except `CHOL`) are used with zero vector as initial approximation, the stopping criterion being the reduction by a factor of 10^6 of the residual norm[4]. Maximal number

[3]Except for `Bump_2911` problem from SuiteSparse test set, introduced in Section 4.3, for which a more refined value of the threshold parameter allows for a substantial improvement.

[4]Since `ILUPACK` uses a different stopping criterion to achieve a given solution accuracy, the value for this latter criterion was adjusted for each problem to approximately correspond to the 10^6 reduction in the residual norm.

of iterations is $1,000$ for all the solvers, except `CG`, where $10,000$ iterations are accepted. The reported times are wall clock times; they were obtained on a workstation with an Intel Xeon E5-2620 processor (2.10GHz) and 128 GB RAM memory.

4.2 PDE problems

Although the preconditioner is of general purpose, we first assess its performance based on model PDE discretizations. For simplicity, only the sequential version is considered in this section. The considered discretizations correspond to two- and three-dimensional (2D and 3D) PDEs defined on, respectively, square (2D) and cube (3D) domains, and discretized, respectively, on $(n_x + 2) \times (n_x + 2)$ (in 2D) and $(n_x + 2) \times (n_x + 2) \times (n_x + 2)$ (in 3D) Cartesian grids. More specifically, problems `POISSON2D` and `POISSON3D` correspond to, respectively, 5-point and 7-point finite difference discretization of Poisson problem $-\Delta u = f$ with homogeneous Dirichlet boundary conditions in two and three dimensions. Further, `LINEL2D` and `LINEL3D` correspond to the displacement formulation of linear elasticity problem $-2\mu\nabla\cdot(\nabla^{(s)}\mathbf{u}) - \lambda\nabla(\nabla\cdot\mathbf{u}) = \mathbf{f}$, $\mathbf{f} \in \mathbb{R}^d$, $d = 2, 3$ with homogeneous Dirichlet boundary conditions on one of the 4 edges (in 2D) or one of the 6 faces (in 3D) of the domain, and Neumann boundary conditions elsewhere on the boundary, discretized using lowest order vector Lagrangian $(P_1)^d$ elements. Now, the problem coefficients μ and λ are constant functions, and are further expressed as $\lambda = E\nu/((1+\nu)(1-2\nu))$ and $\mu = E/(2+2\nu)$, where E is Young's modulus (here, a scaling factor, and therefore set to 1) and $0 < \nu < 1/2$ is the Poisson's ratio. Eventually, `EDGE2D` and `EDGE3D` correspond to the curl-curl problem $\nabla\times\nabla\times\mathbf{u} + 10^{-2}\,\mathbf{u} = \mathbf{f}$, $\mathbf{f} \in \mathbb{R}^d$, $d = 2, 3$, with homogeneous Dirichlet boundary conditions discretized using lowest order Whitney elements of the first kind on a square (in 2D) or a cube (in 3D) mesh; the degrees of freedom are then associated to the grid edges.

The total (i.e., setup and solve) time to solution is reported in Figure 3 for the considered solvers and model PDE discretizations of increasing size; times for each problem are represented on a separate graph. For `LINEL2D` and `LINEL3D` problems, experiments were run for $\nu = 0.4$ and $\nu = 0.49$; there results for $\nu = 0.4$ are similar to those of `POISSON2D` and `POISSON3D` problems in terms of relative solver "ranking", and are therefore not reported here[5].

Not every solver appear on each graph: `AGMG` does not converge for `EDGE2D` and `EDGE3D` problems in $1,000$ iterations, `CG` fails to converge in $10,000$ iterations for `LINEL2D` ($\nu = 0.49$) and `EDGE2D` problems, except for the smallest size, and `ILUPACK` time to solution is over an hour even for the smallest `EDGE3D` problem.

Therefore, regarding the robustness of the solvers, we note that only `ICo` and `CHOL` are able to solve large enough systems for all six problems. Moreover, `ICo` is always at least as fast as `CHOL`, and it is notably faster for large `POISSON3D` and `EDGE3D` problems.

Focusing more specifically on incomplete factorizations, we note that `ICo` is globally faster, but that `ILUPACK` is faster for the `POISSON3D` problem. This might seem disap-

[5]Note that linear elasticity problem in presence of Dirichlet boundary conditions is spectrally equivalent to the associated vector Poisson problem, as follows from the second Korn's inequality [9], and the equivalence coefficients depend on ν.

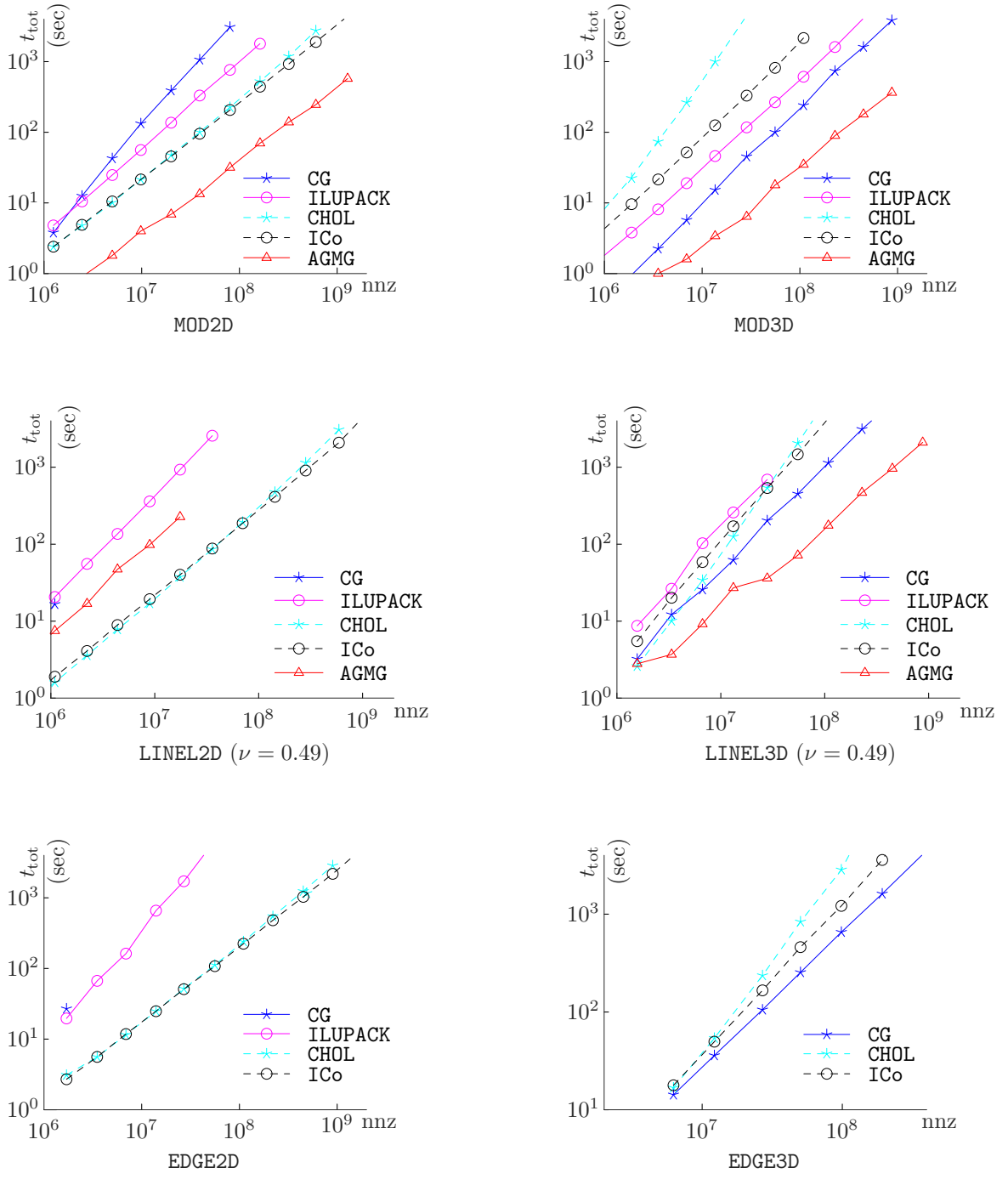

Figure 3: Total times to solution for the considered solvers and model PDE discretizations as a function of the number of nonzero entries of the system matrix. The measured times correspond to graph marks; marks are further interconnected with a line to highlight a trend.

pointing at first. However, it should be noted that pointwise dropping used in ILUPACK is particularly suitable for diagonally dominant matrices from POISSON2D and POISSON3D problems and, moreover, the multilevel nature of ILUPACK is similar to that of multigrid methods, which are considered to be reference solvers for these problems; this latter point is well illustrated by the results of AGMG. On the other hand, the design of ICo is not oriented towards diagonally dominant matrices.

Incidentally, we note that CG is fairly competitive for the considered model 3D problems.

4.3 SuiteSparse problems

We now report the results for the SuiteSparse test set. This test set is composed of all but two SPD matrices of the SuiteSparse matrix collection [11] whose number of rows/columns is larger then 10^5, as available on August 2, 2018 . The two exceptions are bmw7st_1 and thermomech_TK , which are found to be too ill-conditioned ($\kappa(A) \approx 10^{15}$) for the considered stopping criterion. If a matrix graph is found to have several connected components, the test is performed on the largest one. The final set comprises 50 test problems, which are listed in Appendix A. On the following graphs and in Appendix A, the problems are sorted in the increasing order of their number $\mathrm{nnz}(A)$ of nonzero entries of the system matrix. The time is given in seconds per million of nonzero entries of the system matrix.

In Figure 4 we report the total time to solution of ICo solver for the SuiteSparse test set, as well as the contribution of the different steps performed by the solver to this total time. The considered steps are the determination of block partition (which mainly includes the call to the corresponding Metis/mt-Metis routine), symbolic factorization, numerical factorization, and (iterative) solution. For this and the following figures, each abscissa segment corresponds to an individual problem from the test set, and each bar based on this segment is a reported value.

Regarding the total time to solution, for all problems it is below 10 seconds per million of nonzero entries. Regarding the different steps, we note that block partition and/or numerical factorization steps typically take most of the time for the small problems, whereas for large problems, the time is dominated by numerical factorization an/or solution steps.

Next, Figure 5 reports the total time to solution for the sequential version of ICo solver, as well as the OpenMP version with 2 and with 8 threads. The corresponding bars in the diagram are superposed, with the dark green (dark grey) bar of the OpenMP version using 8 threads partially covering (and always lower than) the green (grey) bar of the version using 2 threads, which itself is partially covering (and always lower than) the light green (light grey) bar of the sequential version. We note that OpenMP version with 2 threads allows for a notable speed-up with respect to the sequential version, and that OpenMP version with 8 threads is even faster. However, the scalability is not perfect. This latter observation is partially explained by the fact that only the parallelism of the nested dissection block sparsity structure is used, but also by the degradation of the quality of this block sparsity structure when using mt-Metis with multiple threads. Nevertheless, the total time to solution for the version with 2 threads is below 7 seconds per million of nonzero entries for all problems; it is below 5 seconds per million of nonzero entries for the

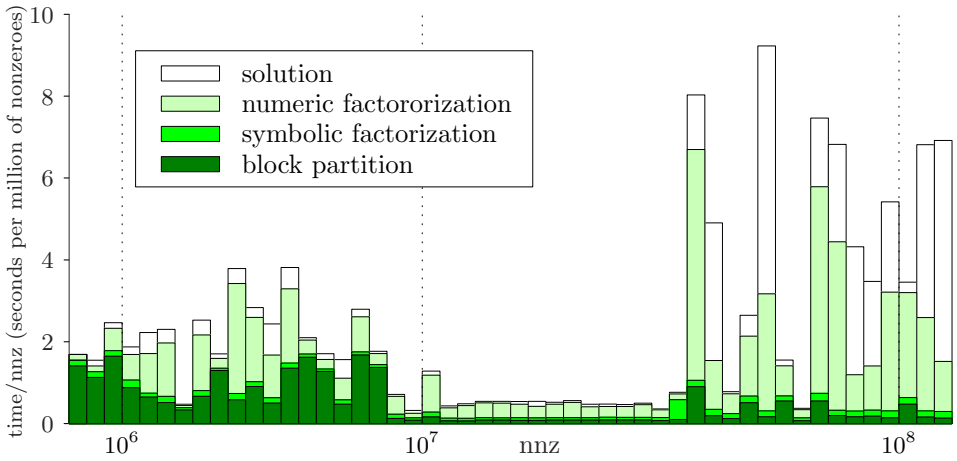

Figure 4: Total time to solution (in seconds per million of nonzero entries of A) for `ICo` solver and the problems from the SuiteSparse test set.

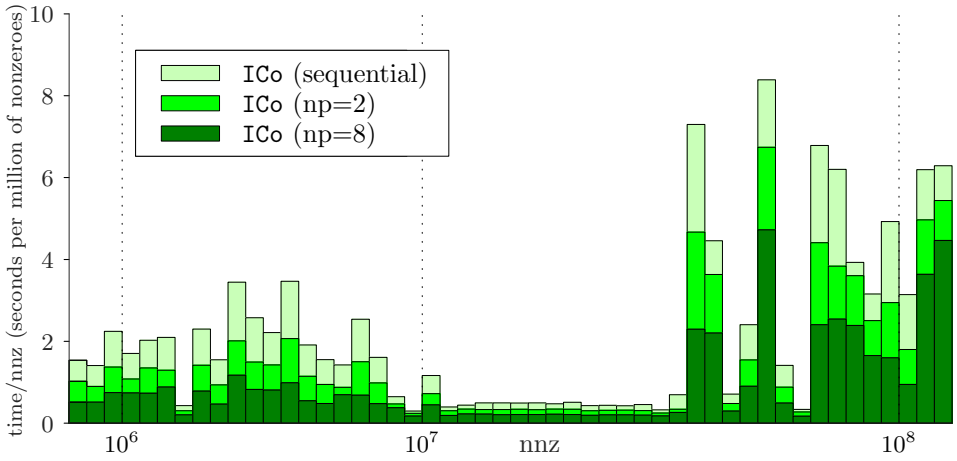

Figure 5: Total time to solution (in seconds per million of nonzero entries of A) for the sequential version of `ICo`, and for its OpenMP version with np=2 and np=8 threads.

version with 8 threads.

Regarding the comparison of `ICo` and `ILUPACK` solvers, the corresponding total times to solution are reported in Figure 6. This and the following three figures are bar diagrams which simultaneously report two parameters, which then correspond to superposed bars of different color. More precisely, the semi-transparent green (light gray) bar is always printed over the opaque blue (dark gray) bar; since the former bar is semi-transparent, the latter bar is clearly visible even when it is completely covered by the other bar; the color of the overlapping area is dark green (grey). Hence, if the blue bar is higher, as is the case for the leftmost bar in Figure 6, the blue bar remains partly uncovered. However, if the semi-transparent green bar is higher than the blue one, the result looks like the third leftmost bar of the figure.

Whereas there is obviously no clear winner, the `ICo` solver has smaller solution time for most of the problems, and the speed-up compared to `ILUPACK` reaches the factor of at

most 38 . On the other hand, ILUPACK is faster for some large problems, and the speed-up compared to ICo reaches the factor of 2.8 .

Next, Figures 7 and 8 report the comparison for the total time to solution between, respectively, ICo and CHOL, and ICo and CG. Regarding the comparison between ICo and CHOL, we note that the results are comparable for small problems, ICo being a bit slower on average; for large problems ICo is either a bit slower than CHOL, or significantly faster. A slightly better performance of CHOL for some problems comes from the fact that the refinement of large nested dissection blocks and the associated use of the hierarchical low-rank approximation scheme from Section 2.3 do not pay off compared to the use of block-matrix (BLAS3) operations with the original blocks. Regarding the comparison between ICo and CG, we note that CG is almost always either slower or non converging (these two situations are represented in the same way on this figure).

5 Conclusions

We have presented an incomplete Cholesky factorization preconditioner for the iterative solution of large sparse symmetric positive definite (SPD) systems. The preconditioner uses orthogonal low-rank approximations in an exact Cholesky factorization. On the theoretical side, we show that the preconditioner is breakdown-free for any choice of the approximation tolerance and, moreover, we extend the scope of the one-level bound from [25] to the considered preconditioner. The extension is achieved with the help of the additional assumption (3.7), which is naturally satisfied by the considered preconditioner. On the practical side, we consider an implementation of the preconditioner which is based on the sparse Cholesky factorization exploiting nested dissection block partitioning. The incomplete factorization is obtained by refining large blocks of the nested dissection partitioning and by further applying the hierarchical low-rank approximation scheme to these blocks. Numerical experiments are performed with a sequential and OpenMP implementation of the solver, and show that the approach is robust, and competitive with the state-of-the-art incomplete factorization as implemented in ILUPACK.

Acknowledgment

The work benefited from the support of the Fonds de Recherche Scientifique-FNRS (Belgium) under grant n° J.0084.16 .

Appendix A

The list of the matrices in the SuiteSparse test set, ordered per number of nonzero entries of the largest connected component, and using the names from the online SuiteSparse database [11], is:

thermomech_TC, thermomech_dM*, G2_circuit, 2cubes_sphere, cfd2,

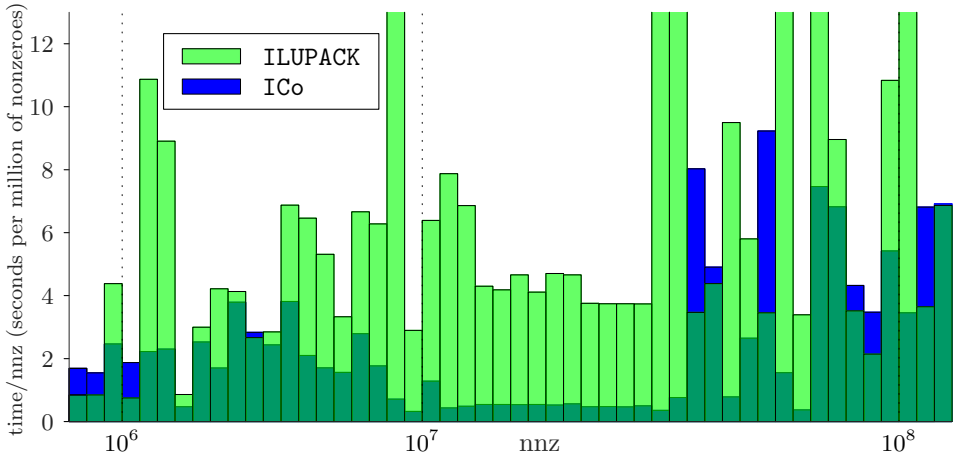

Figure 6: Total time to solution (in seconds per million of nonzero entries of A) for `ICo` and `ILUPACK` solvers.

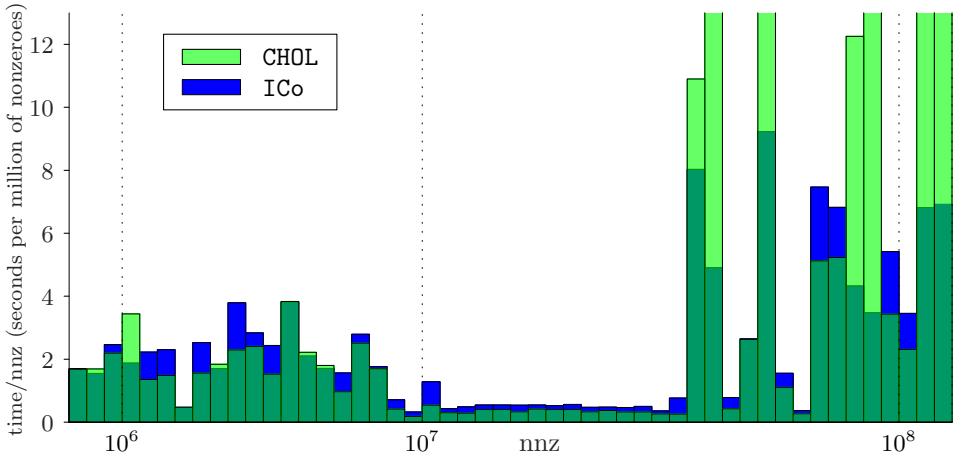

Figure 7: Total time to solution (in seconds per million of nonzero entries of A) for `ICo` and `CHOL` solvers.

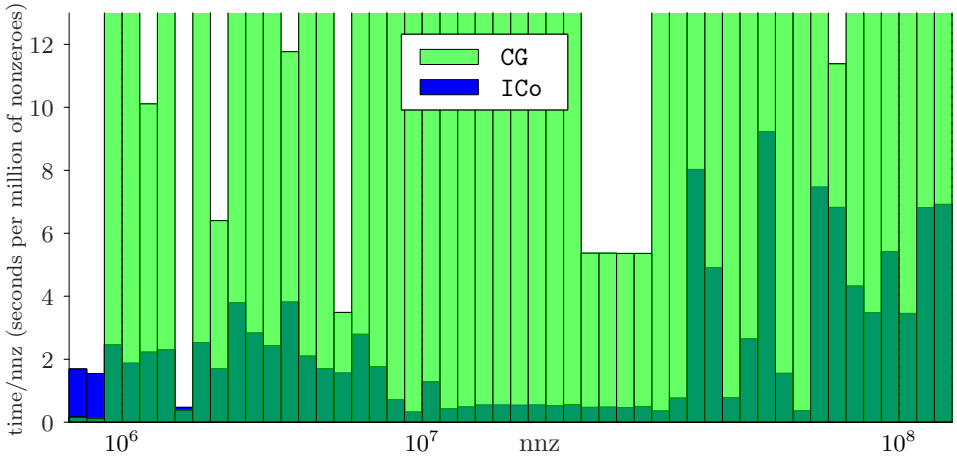

Figure 8: Total time to solution (in seconds per million of nonzero entries of A) for `ICo` and `CG` solvers.

shipsec8, Dubcova3, shipsec1, parabolic_fem, ship_003, offshore, shipsec5, apache2, ecology2, tmt_sym, boneS01, G3_circuit, thermal2, x104, hood, bmwcra_1, pwtk, BenElechi1, af_0_k101, af_1_k101, af_2_k101, af_3_k101, af_4_k101, af_5_k101, af_shell3, af_shell4, af_shell7, af_shell8, msdoor, bundle_adj, StocF-1465, Fault_639, inline_1, PFlow_742*, Emilia_923, boneS10, ldoor, bone010, Hook_1498, Geo_1438, Serena, audikw_1, Flan_1565, Bump_2911*, Queen_4147*.

The matrices marked with * had several connected components.

References

[1] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker, *Improving multifrontal methods by means of block low-rank representations*, SIAM J. Sci. Comput., 37 (2015).

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, third ed., 1999.

[3] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1994.

[4] M. Bebendorf, *Hierarchical Matrices*, vol. 63 of Lecture Notes in Computational Science and Engineering (LNCSE), Springer-Verlag, 2008.

[5] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[6] M. Bollhöfer and Y. Saad, *ILUPACK - preconditioning software package*. Available at http://ilupack.tu-bs.de/ . Release V2.4, October 2016.

[7] M. Bollhöfer and Y. Saad, *Multilevel preconditioners constructed from inverse–based ILUs*, SIAM J. Sci. Comput., 27 (2006), pp. 1627–1650.

[8] S. Börm, *Efficient Numerical Methods for Non-local Operators*, vol. 14 of EMS Tracts in Mathematics, European Mathematical Society, 2010.

[9] D. Braess, *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*, Cambridge University Press, Cambridge, UK, 1997.

[10] L. Cambier, C. Chen, E. G. Boman, S. Rajamanickam, R. S. Tuminaro, and E. Darve, *An algebraic sparsified nested dissection algorithm using low-rank approximations*, SIAM J. Sci. Comput., 41 (2020).

[11] T. A. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Software, 38 (2011), p. No. 1. Available via `https://sparse.tamu.edu/`.

[12] G. Dillon, V. Kalantzis, Y. Xi, and Y. Saad, *A hierarchical low rank schur complement preconditioner for indefinite linear systems*, SIAM J. Sci. Comput., 40 (2018), pp. A2234–A2252.

[13] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. on Math. Software, 9 (1982), pp. 302–325.

[14] A. George, *Nested dissection of a regular finite-element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.

[15] P. Ghysels, X. S. Li, F.-H. Rouet, S. Williams, and A. Napov, *An efficient multi-core implementation of a novel HSS-structured multifrontal solver using randomized sampling*, SIAM J. Sci. Comput., 38 (2016), pp. S358–S384.

[16] G. H. Golub and C. F. van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, Maryland, 1996. Third ed.

[17] L. Grasedyck, R. Kriemann, and S. L. Borne, *Parallel blackbox $\mathcal{H}$-LU preconditioning for elliptic boundary value problems*, Comput. Vis. Sci., 11 (2008), pp. 273–291.

[18] L. Grasedyck, R. Kriemann, and S. L. Borne, *Domain decomposition based $\mathcal{H}$-LU preconditioning*, Numer. Math., 112 (2009), pp. 565–600.

[19] W. Hackbusch, B. Khoromskij, and R. Kriemann, *Hierarchical matrices based on a weak admissibility criterion*, Computing, 73 (2004), pp. 207–243.

[20] W. Hackbusch, B. Khoromskij, and S. Sauter, *On $\mathcal{H}^2$-matrices*, Lectures on Applied Mathematics, Berlin, Germany, 2000, Springer-Verlag, pp. 9–29.

[21] G. Karypis and V. Kumar, *A fast and highly quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1999), pp. 359–392.

[22] D. LaSalle and G. Karypis, *Efficient nested dissection for multicore architectures*, in Euro-Par 2015: Parallel Processing, Springer Berlin Heidelberg, 2015, pp. 467–478.

[23] R. J. Lipton, D. J. Rose, and R. E. Tarjan, *Generalized nested dissection*, SIAM J. Numer. Anal., 16 (1979), pp. 346–358.

[24] J. W. H. Liu, *The Multifrontal Method for Sparse Matrix Solution: Theory and Practice*, SIAM Rev., 34 (1992), pp. 82–109.

[25] A. Napov, *Conditioning analysis of incomplete Cholesky factorizations with orthogonal dropping*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1148–1173.

[26] A. Napov, *ICo software and documentation*, 2021. See `http://metronu.ulb.ac.be/ICo`.

[27] A. Napov and X. S. Li, *An algebraic multifrontal preconditioner that exploits the low-rank property*, Numer. Linear Algebra Appl., 23 (2016), pp. 61–82.

[28] A. Napov and Y. Notay, *An algebraic multigrid method with guaranteed convergence rate*, SIAM J. Sci. Comput., 34 (2012), pp. A1079–A1109.

[29] Y. Notay, *An aggregation-based algebraic multigrid method*, Electron. Trans. Numer. Anal., 37 (2010), pp. 123–146.

[30] F. Pellegrini, *SCOTCH: Static mapping, graph partitioning, and sparse matrix block ordering package.*
Available online at `http://www.labri.fr/~pelegrin/scotch/` .

[31] G. Pichon, E. Darve, M. Faverge, P. Ramet, and J. Roman, *Sparse supernodal solver using block low-rank compression: Design, performance and analysis*, Journal of Computational Science, 27 (2018), pp. 255–270.

[32] H. Pouransari, P. Coulier, and E. Darve, *Fast hierarchical solvers for sparse matrices using extended sparsification and low-rank approximation*, SIAM J. Sci. Comput., 39 (2017), p. A797–A830.

[33] F.-H. Rouet, X. S. Li, P. Ghysels, and A. Napov, *A distributed-memory package for dense Hierarchically Semi-Separable matrix computations using randomization*, ACM Transactions on Mathematical Software, 42 (2016), p. No. 27.

[34] J. W. Ruge and K. Stüben, *Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG)*, in Multigrid Methods for Integral and Differential Equations, D. J. Paddon and H. Holstein, eds., The Institute of Mathematics and its Applications Conference Series, Clarendon Press, Oxford, 1985, pp. 169–212.

[35] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, 2003. Second ed.

[36] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear systems*, Cambridge University Press, Cambridge, 2003.

[37] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid based on smoothed aggregation for second and fourth order problems*, Computing, 56 (1996), pp. 179–196.

[38] J. Xia, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A832–A860.

[39] J. Xia, *Randomized sparse direct solvers*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 197–227.

[40] J. Xia and M. Gu, *Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2899–2920.

[41] Z. Xin, J. Xia, M. V. de Hoop, S. Cauley, and V. Balakrishnan, *A distributed-memory randomized structured multifrontal method for sparse direct solutions*, SIAM J. Sci. Comput., 39 (2017), pp. C292–C318.

[42] Q. Zheng, Y. Xi, and Y. Saad, *Multicolor low-rank preconditioner for general sparse linear systems*, Numerical Linear Algebra with Applications, 27 (2020), p. e2316.