COMMUN. MATH. SCI.
Vol. 18, No. 1, pp. 91–108

RECURSIVELY PRECONDITIONED HIERARCHICAL INTERPOLATIVE FACTORIZATION FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS*

JORDI FELIU-FABÀ†, KENNETH L. HO‡, AND LEXING YING§

Abstract. The hierarchical interpolative factorization for elliptic partial differential equations is a fast algorithm for approximate sparse matrix inversion in linear or quasilinear time. Its accuracy can degrade, however, when applied to strongly ill-conditioned problems. Here, we propose a simple modification that can significantly improve the accuracy at no additional asymptotic cost: applying a block Jacobi preconditioner before each level of skeletonization. This dramatically limits the impact of the underlying system conditioning and enables the construction of robust and highly efficient preconditioners even at quite modest compression tolerances. Numerical examples demonstrate the performance of the new approach.

Keywords. Recursive preconditioning; hierarchical interpolative factorization.

AMS subject classifications. 65F05; 65F08; 65N22.

1. Introduction

In this paper, we consider elliptic partial differential equations (PDE) of the form

$$-\nabla\cdot(a(x)\nabla u(x))+b(x)u(x)=f(x),\quad x\in\Omega\subset\mathbb{R}^d \tag{1.1}$$

in two (2D, $d=2$) and three dimensions (3D, $d=3$), with appropriate boundary conditions on $\partial\Omega$. Here, $a(x)$, $b(x)$, and $f(x)$ are given functions and $u(x)$ is the unknown field. Such equations are of great importance in science and engineering and can model a wide variety of physical phenomena. In a typical numerical solution, (1.1) is often discretized with local schemes such as finite differences or finite elements, thus leading to a linear system

$$Au=f, \tag{1.2}$$

where $A\in\mathbb{R}^{N\times N}$ is sparse and N is the number of degrees of freedom (DOFs) in the discretization. Furthermore, it is common in practice for A to be symmetric positive definite (SPD); we hereafter assume this to be the case. However, for non-SPD matrices, our approach can be used replacing Cholesky factorizations by LDL^T or LU factorizations in Section 2 and Section 3.

1.1. Background.

A significant part of research in scientific computing has been devoted to the numerical solution of (1.2). Previous methods for its solution can largely be classified into several groups as follows.

The first group consists of classical direct methods such as Gaussian elimination or other standard matrix factorizations [14], nominally with $O(N^3)$ complexity. This can be accelerated by exploiting sparsity, for instance using nested dissection (ND) [12] to

*Received: March 14, 2019; Accepted (in revised form): September 3, 2019. Communicated by Jianfeng Lu.

†Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305, USA (jfeliu@stanford.edu).

‡Center for Computational Mathematics, Flatiron Institute, New York, NY 10010, USA (klho@alumni.caltech.edu).

§Department of Mathematics and Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305, USA (lexing@stanford.edu).

$O(N^{3/2})$ in 2D and $O(N^2)$ in 3D for regular grids. These can still be quite prohibitive, especially for large-scale 3D problems. Therefore, although robust and highly accurate, such methods are generally not used beyond moderate problem sizes (at least in non-parallel environments).

The second group includes iterative methods [32] such as conjugate gradient (CG) [18] and multigrid [6,16], which can achieve $O(N)$ complexity if only a small number of iterations are required. However, when A is ill-conditioned or otherwise has scattered eigenvalues, as is common for high-contrast or non-smooth coefficients, the number of iterations needed can be very large. In such cases, we can expect fast convergence only if we use a good preconditioner, which can itself be a challenge to find. Furthermore, iterative methods can be inefficient for systems involving multiple right-hand sides, which are prevalent in many applications.

Bridging the two previous groups are the more recent rank-structured direct solvers, which are based on the low-rank compression of certain submatrices encountered during the factorization process. Many of these are essentially accelerated ND schemes, powered by techniques for structured dense linear algebra, and as such are associated with various dense classifications based on admissibility conditions and the use of nested bases, including $\mathcal{H}$-matrices [15,33,34], hierarchically semiseparable (HSS) matrices [36–38], and hierarchically off-diagonal low-rank (HODLR) matrices [3], among others [2,13,17,21,26,31,35]. Importantly, these algorithms can be much faster than classical direct methods, with some even attaining $O(N)$ or $O(N\log N)$ complexity. Moreover, they inherently offer a speed-accuracy trade-off through the compression tolerance that is naturally suited to constructing general-purpose preconditioners. Combined with standard iterative methods, such preconditioners can enable fast and robust convergence at a far lower total cost than either a full direct solve (i.e., at high accuracy) or an unpreconditioned iterative solve.

Although rank-structured solvers have proven quite successful, they nevertheless can suffer greatly from ill-conditioning. In particular, the associated low-rank compression has traditionally been performed with respect to the forward operator so that A is well-approximated but A^{-1} may not be. Indeed, for an SPD approximation $F = GG^T$ of A with $\|A-F\|/\|A\| = O(\epsilon)$, it has often been observed that $\|I - G^{-1}AG^{-T}\| = O(\epsilon\kappa(A))$, where $\kappa(\cdot)$ is the condition number. Thus, ill-conditioned problems can necessitate a much higher compression accuracy than would otherwise be required in order to achieve a given solve error. This can be a significant impediment, especially for constructing low-accuracy preconditioners.

Recently, there has been some work [1,39–41] aimed at improving the solve error and hence enabling the use of much looser compression tolerances, though only for the structured dense case so far. We highlight in particular the work of Xia et al. [39], which combines block Jacobi rescaling and off-diagonal compression for preconditioning. This strategy is further investigated in [1], where a block Jacobi rescaling combined with the singular value decomposition (SVD) is used for low-rank approximation in the HODLR format. Specifically, they showed that (simplified and rephrased from [1]):

PROPOSITION 1.1. *Let A be a block 2×2 SPD matrix*

$$A = \begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} C_1 & \\ & C_2 \end{bmatrix} \begin{bmatrix} I & B^T \\ B & I \end{bmatrix} \begin{bmatrix} C_1^T & \\ & C_2^T \end{bmatrix}, \tag{1.3}$$

where each A_{ii} has Cholesky decomposition $A_{ii} = C_i C_i^T$ and $B = C_2^{-1} A_{21} C_1^{-T}$. Furthermore, let B have truncated SVD approximation $\tilde{B}$ (i.e., projected onto the leading

singular subspace) and define

$$F = \begin{bmatrix} C_1 & \\ & C_2 \end{bmatrix} \begin{bmatrix} I & \widetilde{B}^T \\ \widetilde{B} & I \end{bmatrix} \begin{bmatrix} C_1^T & \\ & C_2^T \end{bmatrix} = GG^T. \tag{1.4}$$

If $\|B - \widetilde{B}\| \le \epsilon$, *then* $\|I - G^{-1}AG^{-T}\| \le \epsilon$. *In particular,* $\kappa(G^{-1}AG^{-T}) \le (1+\epsilon)/(1-\epsilon)$.

In other words, "preconditioning" first with block Cholesky factors before SVD compression allows very precise control over the solve error and therefore also on the convergence of preconditioned CG, which depends on the spectrum of $G^{-1}AG^{-T}$. This describes a one-level scheme; the multilevel extension is immediate via a recursive binary partitioning following the HODLR framework. Some minor loss of accuracy is incurred, but the overall control is still very tight. Indeed, effective preconditioners for Schur complements associated with 2D PDEs were demonstrated at tolerances up to $\epsilon \sim 0.1$. Very similar methods are presented in [39–41]. Other related efforts include [4, 5, 24, 29, 42].

However, HODLR/HSS methods have optimal linear or quasilinear complexity only if the approximation rank ρ of the off-diagonal blocks grows at most very slowly with N. Thus, they are really best suited for "one-dimensional" (1D) dense problems (by analogy with elliptic integral equations), where typically $\rho = O(\log N)$ [20, 27]. This is the case for sparse PDEs in 2D [7], for which elimination along, e.g., the ND ordering creates dense fill-in loaded on 1D separator edges. But it does not adequately capture the situation in 3D, where now the dense subproblems live on 2D faces, with $\rho = O(N^{1/3})$. More advanced techniques [9, 10, 22, 28] are required to reduce the cost, which are not addressed by [1, 40, 41].

1.2. Contributions. Our principal goal in this paper will be to extend the ideas of [1] beyond the HODLR framework to a suitably general accelerated ND method capable of efficiently handling both 2D and 3D problems. We focus in particular on the hierarchical interpolative factorization (HIF) [21], a fast algorithm for computing approximate generalized Cholesky decompositions in $O(N)$ or $O(N \log N)$ time. This is achieved through alternating levels of elimination and "skeletonization", which eliminates DOFs from dense matrices by exploiting low-rank structure, to sparsify and reduce the dimension of the separators. For example, in 3D, elimination first reduces the problem to 2D on separator faces and then skeletonization reduces that to 1D along edges, yielding $O(N \log N)$ cost. Additionally, skeletonizing the edges themselves can further bring this down to $O(N)$, thus completing a full dimensional reduction sweep.

However Proposition 1.1 is quite specific to the block 2×2 case, where the off-diagonal blocks can be simultaneously diagonalized and any error amplification suppressed due to the orthogonality of certain subspaces. This does not apply to HIF, which instead maintains a global view of all matrix blocks at each level of the factorization. Still, we can appeal to the same intuition and precondition before each round of skeletonization. The essential effect of this change can be understood heuristically as follows. First consider

$$A = BB^T + E, \quad \|E\| \le \epsilon \|A\|, \tag{1.5}$$

which describes a standard "unpreconditioned" approximation to relative precision ϵ. Then the solve error is

$$\|I - B^{-1}AB^{-T}\| = \|B^{-1}EB^{-T}\| \le \epsilon \|A\| \|B^{-1}\| \|B^{-T}\| \sim \epsilon \kappa(A) \tag{1.6}$$

as previously asserted. Now let

$$A = C\widetilde{A}C^T = C(\widetilde{B}\widetilde{B}^T + \tilde{E})C^T, \quad \|\tilde{E}\| \le \epsilon\|\widetilde{A}\|, \tag{1.7}$$

i.e., the approximation is done after symmetrically preconditioning with C. Then $A \approx F = GG^T$ with $G = C\widetilde{B}$, so

$$\|I - G^{-1}AG^{-T}\| = \|\widetilde{B}^{-1}\tilde{E}\widetilde{B}^{-T}\| \le \epsilon\|\widetilde{B}^{-1}\|\|\widetilde{B}^{-T}\|\|\widetilde{A}\| \sim \epsilon\kappa(\widetilde{A}) \tag{1.8}$$

and the error is now amplified only by $\kappa(\widetilde{A}) \ll \kappa(A)$. This is a much more general yet necessarily weaker result that is compatible with HIF. In particular, no detailed assumptions are made on either the block partitioning, preconditioner type, or compression method; at the same time, the accuracy is still subject to the conditioning of $\widetilde{A}$, which may nevertheless be poor. The latter can be improved in the multilevel setting, where effectively $\widetilde{A}$ itself is preconditioned at the next level and so on. Altogether, this outlines a "recursively preconditioned" HIF (PHIF) with significantly enhanced robustness to $\kappa(A)$.

We demonstrate this approach using local Cholesky factors as in [1] to precondition at each scale. As expected, we find substantial improvements in the solve accuracy (often by several orders of magnitude) and therefore in its effectiveness as a preconditioner, especially for ill-conditioned problems. Furthermore, the same asymptotic rank estimates are observed to hold as before and hence the computational complexity is preserved. With this simple modification, we thus construct the first "preconditioned" structured nested dissection (ND) solver with optimal or near-optimal performance, in terms of computational complexity, for PDEs in 2D and 3D.

2. Preliminaries

This section reviews some key linear algebraic primitives used in PHIF. We adopt the following notation hereafter: uppercase letters (A, B, F, etc.) denote matrices; calligraphic letters ($\mathcal{I}$, $\mathcal{J}$, etc.) denote sets of indices, each of which is associated with a DOF; $A_{\mathcal{I}\mathcal{J}}$ is the submatrix of A restricted to $\mathcal{I}$ and $\mathcal{J}$, respectively, for the rows and columns; and $[n] = \{1, \dots, n\}$ for $n \in \mathbb{N}$.

2.1. Block elimination. Consider an SPD matrix $A \in \mathbb{R}^{N\times N}$ with block partitioning

$$A = \begin{bmatrix} A_{\mathcal{I}\mathcal{I}} & A_{\mathcal{B}\mathcal{I}}^T & \\ A_{\mathcal{B}\mathcal{I}} & A_{\mathcal{B}\mathcal{B}} & A_{\mathcal{R}\mathcal{B}}^T \\ & A_{\mathcal{R}\mathcal{B}} & A_{\mathcal{R}\mathcal{R}} \end{bmatrix}, \tag{2.1}$$

where $[N] = \mathcal{I} \cup \mathcal{B} \cup \mathcal{R}$ up to permutation. This type of structure is characteristic of the sparse linear system (1.2), which discretizes the PDE (1.1) through a partitioning of the domain Ω as a union of multiple cells with disjoint interior. For a given cell, $\mathcal{I}$ then represents the DOFs inside that cell, $\mathcal{B}$ the DOFs on the boundary, and $\mathcal{R}$ the remaining DOFs outside the cell and so do not interact with those in $\mathcal{I}$, i.e., $A_{\mathcal{R}\mathcal{I}} = 0$.

The goal of block elimination is to zero out the $A_{\mathcal{B}\mathcal{I}}$ block in order to decouple $\mathcal{I}$ from the rest. Let $A_{\mathcal{I}\mathcal{I}}$ have Cholesky decomposition $A_{\mathcal{I}\mathcal{I}} = L_{\mathcal{I}}L_{\mathcal{I}}^T$ and define the *elimination matrix*

$$M_{\mathcal{I}} = \begin{bmatrix} L_{\mathcal{I}}^{-T} & -A_{\mathcal{I}\mathcal{I}}^{-1}A_{\mathcal{B}\mathcal{I}}^T & \\ & I & \\ & & I \end{bmatrix} = \begin{bmatrix} L_{\mathcal{I}}^{-T} & & \\ & I & \\ & & I \end{bmatrix} \begin{bmatrix} I & -L_{\mathcal{I}}^{-1}A_{\mathcal{B}\mathcal{I}}^T & \\ & I & \\ & & I \end{bmatrix} \in \mathbb{R}^{N\times N}. \tag{2.2}$$

Then

$$M_{\mathcal{I}}^T A M_{\mathcal{I}} = \begin{bmatrix} I & & \\ & \widetilde{A}_{\mathcal{B}\mathcal{B}} & A_{\mathcal{R}\mathcal{B}}^T \\ & A_{\mathcal{R}\mathcal{B}} & A_{\mathcal{R}\mathcal{R}} \end{bmatrix}, \quad \widetilde{A}_{\mathcal{B}\mathcal{B}} = A_{\mathcal{B}\mathcal{B}} - A_{\mathcal{B}\mathcal{I}} A_{\mathcal{I}\mathcal{I}}^{-1} A_{\mathcal{B}\mathcal{I}}^T. \tag{2.3}$$

Notice that $A_{\mathcal{R}\mathcal{B}}$ and $A_{\mathcal{R}\mathcal{R}}$ remain unchanged while $\mathcal{I}$ has been fully eliminated; we are now left with a smaller problem over the restricted indices $\mathcal{B}\cup\mathcal{R}$ only.

In the context of (1.2), we often want to perform block elimination on each of a collection of, say, p cells with interior and boundary indices $\{\mathcal{I}_i\}_{i=1}^p$ and $\{\mathcal{B}_i\}_{i=1}^p$, respectively, and $A_{\mathcal{I}_i,\mathcal{I}_j} = 0$ for all $i \neq j$. Because the Schur complement updates to each $A_{\mathcal{B}_i\mathcal{B}_i}$ are commutative, the cells can be eliminated independently in any order, and the resulting matrix $M = \prod_{i=1}^p M_{\mathcal{I}_i}$, where each $M_{\mathcal{I}_i}$ is given by (2.2) with $\mathcal{I} = \mathcal{I}_i$, $\mathcal{B} = \mathcal{B}_i$, and $\mathcal{R} = [N] \setminus (\mathcal{I}_i \cup \mathcal{B}_i)$, is well-defined. Observe then that $M^T A M$ consists of the identity along $\cup_{i=1}^p \mathcal{I}_i$ and reduces to a subsystem in $[N] \setminus \cup_{i=1}^p \mathcal{I}_i$ only, i.e., the DOFs in each $\mathcal{I}_i$ have been eliminated.

2.2. Skeletonization. Skeletonization is the generalization of block elimination for dense matrices with low-rank off-diagonal blocks. Let

$$A = \begin{bmatrix} A_{\mathcal{I}\mathcal{I}} & A_{\mathcal{R}\mathcal{I}}^T \\ A_{\mathcal{R}\mathcal{I}} & A_{\mathcal{R}\mathcal{R}} \end{bmatrix} \in \mathbb{R}^{N\times N} \tag{2.4}$$

be SPD with $A_{\mathcal{R}\mathcal{I}} \in \mathbb{R}^{N_{\mathcal{R}}\times N_{\mathcal{I}}}$ having numerical rank k to relative precision ϵ. Then from the interpolative decomposition (ID) [8], there exists a disjoint partitioning $\mathcal{I} = \widetilde{\mathcal{I}} \cup \hat{\mathcal{I}}$ into *redundant* and *skeleton* DOFs, respectively, and an *interpolation matrix* $T_{\mathcal{I}} \in \mathbb{R}^{k\times(N_{\mathcal{I}}-k)}$ such that

$$A_{\mathcal{R}\widetilde{\mathcal{I}}} = A_{\mathcal{R}\hat{\mathcal{I}}} T_{\mathcal{I}} + E_{\mathcal{I}}, \quad \|E_{\mathcal{I}}\| = O(\epsilon \|A_{\mathcal{R}\mathcal{I}}\|), \tag{2.5}$$

i.e., any redundant column of $A_{\mathcal{R}\mathcal{I}}$ can be well approximated by a linear combination of the skeleton columns of $A_{\mathcal{R}\mathcal{I}}$. The choice of $\widetilde{\mathcal{I}}$ and $\hat{\mathcal{I}}$ is not unique and is typically chosen so that $\|T_{\mathcal{I}}\|$ is not too large. Following this approximation, we can rewrite A (up to permutation) as

$$A = \left[\begin{array}{cc|c} A_{\widetilde{\mathcal{I}}\widetilde{\mathcal{I}}} & A_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}}^T & A_{\mathcal{R}\widetilde{\mathcal{I}}}^T \\ A_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}} & A_{\hat{\mathcal{I}}\hat{\mathcal{I}}} & A_{\mathcal{R}\hat{\mathcal{I}}}^T \\ \hline A_{\mathcal{R}\widetilde{\mathcal{I}}} & A_{\mathcal{R}\hat{\mathcal{I}}} & A_{\mathcal{R}\mathcal{R}} \end{array}\right] \approx \left[\begin{array}{cc|c} A_{\widetilde{\mathcal{I}}\widetilde{\mathcal{I}}} & A_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}}^T & T_{\mathcal{I}}^T A_{\mathcal{R}\hat{\mathcal{I}}}^T \\ A_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}} & A_{\hat{\mathcal{I}}\hat{\mathcal{I}}} & A_{\mathcal{R}\hat{\mathcal{I}}}^T \\ \hline A_{\mathcal{R}\hat{\mathcal{I}}} T_{\mathcal{I}} & A_{\mathcal{R}\hat{\mathcal{I}}} & A_{\mathcal{R}\mathcal{R}} \end{array}\right]. \tag{2.6}$$

Introducing the *zeroing matrix* $Z_{\mathcal{I}}$ as follows and applying it on both sides leads to

$$Z_{\mathcal{I}}^T A Z_{\mathcal{I}} \approx \left[\begin{array}{cc|c} \widetilde{A}_{\widetilde{\mathcal{I}}\widetilde{\mathcal{I}}} & \widetilde{A}_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}}^T & \\ \widetilde{A}_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}} & A_{\hat{\mathcal{I}}\hat{\mathcal{I}}} & A_{\mathcal{R}\hat{\mathcal{I}}}^T \\ \hline & A_{\mathcal{R}\hat{\mathcal{I}}} & A_{\mathcal{R}\mathcal{R}} \end{array}\right], \quad Z_{\mathcal{I}} = \left[\begin{array}{cc|c} I & & \\ -T_{\mathcal{I}} & I & \\ \hline & & I \end{array}\right] \in \mathbb{R}^{N\times N}, \tag{2.7}$$

where only the terms $\widetilde{A}_{\widetilde{\mathcal{I}}\widetilde{\mathcal{I}}}$ and $\widetilde{A}_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}}$ are updated with

$$\widetilde{A}_{\widetilde{\mathcal{I}}\widetilde{\mathcal{I}}} = A_{\widetilde{\mathcal{I}}\widetilde{\mathcal{I}}} - T_{\mathcal{I}}^T A_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}} - A_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}}^T T_{\mathcal{I}} + T_{\mathcal{I}}^T A_{\hat{\mathcal{I}}\hat{\mathcal{I}}} T_{\mathcal{I}}, \quad \widetilde{A}_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}} = A_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}} - A_{\hat{\mathcal{I}}\hat{\mathcal{I}}} T_{\mathcal{I}}. \tag{2.8}$$

We can now use block elimination to eliminate $\widetilde{\mathcal{I}}$, assuming that $\widetilde{A}_{\widetilde{\mathcal{I}}\widetilde{\mathcal{I}}}$ remains SPD. Letting $M_{\widetilde{\mathcal{I}}}$ be the elimination matrix so defined and introducing $K_{\mathcal{I}} \equiv Z_{\mathcal{I}} M_{\widetilde{\mathcal{I}}}$, we therefore

have

$$M_{\widetilde{\mathcal{I}}}^T Z_{\mathcal{I}}^T A Z_{\mathcal{I}} M_{\widetilde{\mathcal{I}}} \equiv K_{\mathcal{I}}^T A K_{\mathcal{I}} \approx \left[\begin{array}{c|c} I & \\ & \widetilde{A}_{\hat{\mathcal{I}}\hat{\mathcal{I}}} \quad A_{\hat{\mathcal{I}}\mathcal{R}}^T \\ \hline & A_{\mathcal{R}\hat{\mathcal{I}}} \quad A_{\mathcal{R}\mathcal{R}} \end{array}\right], \quad K_{\mathcal{I}} = Z_{\mathcal{I}} M_{\widetilde{\mathcal{I}}} \tag{2.9}$$

with $\widetilde{A}_{\hat{\mathcal{I}}\hat{\mathcal{I}}} = A_{\hat{\mathcal{I}}\hat{\mathcal{I}}} - \widetilde{A}_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}} \widetilde{A}_{\widetilde{\mathcal{I}}\widetilde{\mathcal{I}}}^{-1} \widetilde{A}_{\hat{\mathcal{I}}\widetilde{\mathcal{I}}}^T$. In other words, the redundant DOFs have been eliminated without any modification to the matrix entries involving $\mathcal{R}$.

REMARK 2.1. Recall Weyl's inequality: For any perturbation matrix P, $|\lambda_i(A+P) - \lambda_i(A)| \le \|P\|$, where $\lambda_i(\cdot)$ is the i-th ordered eigenvalue. An immediate consequence is that if $\|E_{\widetilde{\mathcal{I}}}\| < \lambda_{\min}(A)$ in the above, where $\lambda_{\min}(\cdot)$ is the smallest eigenvalue, then $\lambda_{\min}(Z_{\mathcal{I}}^T A Z_{\mathcal{I}}) > 0$, so $\widetilde{A}_{\widetilde{\mathcal{I}}\widetilde{\mathcal{I}}}$ is SPD.

As before, we often want to perform skeletonization for each of a collection of disjoint index sets $\{\mathcal{I}_i\}_{i=1}^p$. But $K_{\mathcal{I}_1} K_{\mathcal{I}_2} \neq K_{\mathcal{I}_2} K_{\mathcal{I}_1}$ in general since skeletonizing, say, $\mathcal{I}_1$ first can change the set $\mathcal{R}$ for $\mathcal{I}_2$. Commutativity can be restored by instead using the ID to first find the redundant and skeleton DOFs for all $\mathcal{I}_i$ before zeroing and elimination. We will not distinguish between these two approaches as they make little difference from a practical point of view, though we note that the former "sequential" method is typically faster in a serial setting, while the latter is amenable to parallelization [25]. In particular, we will simply write $K = \prod_{i=1}^p K_{\mathcal{I}_i}$ for the aggregate skeletonization operator, where the product is understood to be taken in some appropriate order. The skeletonized matrix $K^T A K$ then consists of the identity along $\cup_{i=1}^p \widetilde{\mathcal{I}}_i$ and reduces to a subsystem in $[N] \setminus \cup_{i=1}^p \widetilde{\mathcal{I}}_i = \cup_{i=1}^p \hat{\mathcal{I}}_i$ only.

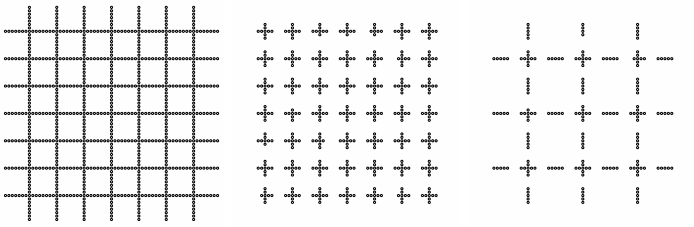
2.3. Block Jacobi preconditioning. Let A be as in (2.1) and suppose that $A_{\mathcal{I}\mathcal{I}}$ has Cholesky decomposition $A_{\mathcal{I}\mathcal{I}} = L_{\mathcal{I}} L_{\mathcal{I}}^T$. Then

$$C_{\mathcal{I}}^T A C_{\mathcal{I}} = \begin{bmatrix} I & \widetilde{A}_{\mathcal{B}\mathcal{I}}^T & \\ \widetilde{A}_{\mathcal{B}\mathcal{I}} & A_{\mathcal{B}\mathcal{B}} & A_{\mathcal{R}\mathcal{B}}^T \\ & A_{\mathcal{R}\mathcal{B}} & A_{\mathcal{R}\mathcal{R}} \end{bmatrix}, \quad C_{\mathcal{I}} = \begin{bmatrix} L_{\mathcal{I}}^{-T} & & \\ & I & \\ & & I \end{bmatrix} \in \mathbb{R}^{N \times N}, \tag{2.10}$$

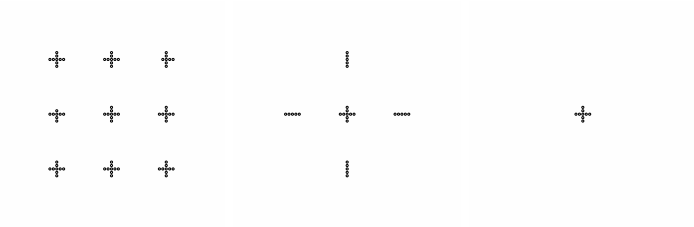
where $\widetilde{A}_{\mathcal{B}\mathcal{I}} = A_{\mathcal{B}\mathcal{I}} L_{\mathcal{I}}^{-T}$. Since $|\mathcal{B}|$ is assumed small relative to $|\mathcal{R}|$, only a limited number of matrix entries are modified. Moreover, the *preconditioning matrix* $C_{\mathcal{I}}$ in (2.10) is block diagonal, so any collection $\{\mathcal{I}_i\}_{i=1}^p$ of disjoint index sets can be block preconditioned independently via $C = \prod_{i=1}^p C_{\mathcal{I}_i}$; the preconditioned matrix $C^T A C$ has unit block diagonal.

3. Algorithm

HIF utilizes alternating levels of block elimination and skeletonization to sparsify and eliminate from the system matrix at each problem scale, following the ND tree. It has a natural geometric interpretation, with both elimination and skeletonization acting as dimensional reduction operators. In 2D, block elimination first reduces from 2D cells to 1D edges then skeletonization reduces that to "zero-dimensional" points, thus achieving $O(N)$ complexity. In 3D, the same procedure gives a reduction from 3D cells to 2D faces to 1D edges, with near-optimal $O(N \log N)$ total cost; estimated $O(N)$ scaling can be recovered by additionally skeletonizing the edges, but we will not consider that here for simplicity. Figures 3.1 and 3.2 provide such a geometric view by showing the remaining DOFs after each step for some small examples. We refer the reader to [21] for further details.

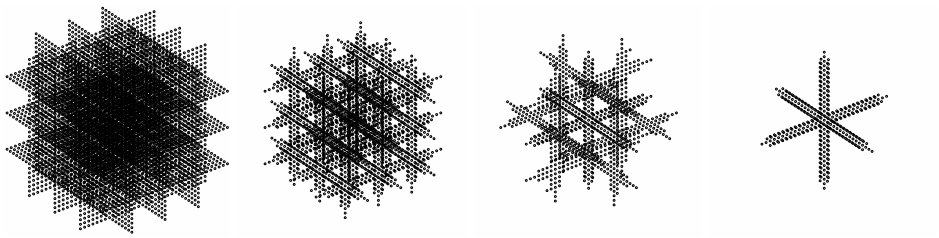


$\ell=0$ cell elimination $\ell=0$ skeletonization $\ell=1$ cell elimination



$\ell=1$ skeletonization $\ell=2$ cell elimination $\ell=2$ skeletonization

Fig. 3.1: *Active DOFs at each level ℓ of HIF in 2D.*

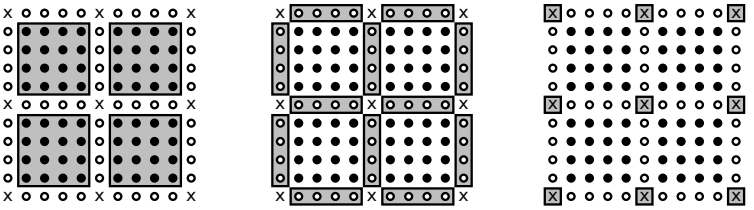


$\ell=0$ cell elimination $\ell=0$ skeletonization $\ell=1$ cell elimination $\ell=1$ skeletonization

Fig. 3.2: *Active DOFs at each level ℓ of HIF in 3D.*

Our new PHIF follows the same framework but now adds an extra preconditioning step before each level of skeletonization. As motivated in Section 1.2, this serves to control the error amplification in the matrix inverse. While the algorithm can be described in quite general terms, requiring only some sensible geometric partitioning in terms of cells, faces, and edges, as appropriate — it is robust to exactly how these are defined — we will be specific here in order to fix the ideas.

3.1. Two dimensions. Without loss of generality, consider the PDE (1.1) on $\Omega=(0,1)^2$ with Dirichlet boundary conditions, discretized using finite differences via the standard five-point stencil over a uniform grid with step size h. We assume that the grid size $n=1/h$ in each dimension satisfies $n=2^L m$ for integer L and m with $m=O(1)$. The DOFs are then the solution values $u_j=u(x_j)$ at the grid points $x_j=jh=(j_1,j_2)\cdot h$ for integers $1\le j_1,j_2\le n-1$. The resulting matrix A in (1.2) is SPD and sparse, consisting only of nearest-neighbor interactions with $\{u_{j\pm e_i}\}_{i=1}^2$ for each u_j, where e_i is the i-th



(a) interior cells (b) edges (c) corners

Fig. 3.3: *Grouping of DOFs into interior cells, edges, and corners in 2D.*

unit coordinate vector. The total number of DOFs is $N=(n-1)^2$.

Define a uniform quadtree on the domain Ω so that it partitions into $2^{(L-\ell)}\times 2^{(L-\ell)}$ square cells at each level $\ell=0,1,\ldots,L$, going from the leaves to the root. Each such cell covers $(2^\ell m+1)\times(2^\ell m+1)$ grid points (or ghost points for those touching the boundary); among these,

- $(2^\ell m-1)^2$ are *interior* points;
- $4(2^\ell m-1)$ are *edge* points, where each edge is shared between up to two cells and
- 4 are *corner* points, each shared between up to four cells.

See Figure 3.3 for a schematic. Let p_ℓ be the total number of cells on level ℓ, $q_\ell\sim 2p_\ell$ the total number of edges, and $r_\ell\sim 3p_\ell$ the total number of edges and corners altogether.

The algorithm proceeds by eliminating DOFs level by level. Let $\mathcal{S}_\ell$ be the remaining *active* DOFs at level ℓ and A_ℓ the corresponding state of the matrix; initially, $\mathcal{S}_0=[N]$ and $A_0=A$. Then we perform the following in sequence at each level $\ell=0,1,\ldots,L-1$:

(1) **Cell elimination.** Group the active DOFs $\mathcal{S}_\ell$ by interior cells and let $\{\mathcal{I}_{\ell,i}\}_{i=1}^{p_\ell}$ be the collection of all such index sets. Block elimination with respect to $\{\mathcal{I}_{\ell,i}\}_{i=1}^{p_\ell}$ as in Section 2.1 then gives

$$A_{\ell'}=M_\ell^T A_\ell M_\ell,\quad M_\ell=\prod_{i=1}^{p_\ell} M_{\mathcal{I}_{\ell,i}}, \tag{3.1}$$

where $M_{\mathcal{I}_{\ell,i}}$ is defined following (2.2). The corresponding active DOFs are $\mathcal{S}_{\ell'}=\mathcal{S}_\ell\setminus\cup_{i=1}^{p_\ell}\mathcal{I}_{\ell,i}$, which now comprise only edges and corners at this scale.

(2) **Block Jacobi preconditioning.** Now group $\mathcal{S}_{\ell'}$ by edges and corners, and let $\{\mathcal{I}_{\ell',i}\}_{i=1}^{r_\ell}$ be the collection of corresponding index sets. Preconditioning as in Section 2.3 yields

$$A_{\ell''}=C_\ell^T A_{\ell'} C_\ell,\quad C_\ell=\prod_{i=1}^{r_\ell} C_{\mathcal{I}_{\ell',i}}, \tag{3.2}$$

where $C_{\mathcal{I}_{\ell',i}}$ is as defined in (2.10). The resulting matrix $A_{\ell''}$ has unit block diagonal and the set of active DOFs $\mathcal{S}_{\ell''}=\mathcal{S}_{\ell'}$ is unchanged. This step is skipped (i.e., $C_\ell=I$) in the standard HIF.

(3) **Edge skeletonization.** Group $\mathcal{S}_{\ell''}$ by edges and let $\{\mathcal{I}_{\ell'',i}\}_{i=1}^{q_\ell}$ be the collection of

corresponding index sets. Skeletonization as in Section 2.2 then gives

$$A_{\ell+1} \approx K_\ell^T A_{\ell''} K_\ell, \qquad K_\ell = \prod_{i=1}^{q_\ell} K_{\mathcal{I}_{\ell'',i}}, \quad K_{\mathcal{I}_{\ell'',i}} = Z_{\mathcal{I}_{\ell'',i}} M_{\widetilde{\mathcal{I}}_{\ell'',i}}, \tag{3.3}$$

where $Z_{\mathcal{I}_{\ell'',i}}$ is defined following (2.7) and $M_{\widetilde{\mathcal{I}}_{\ell'',i}}$ is the associated elimination matrix (2.2) acting on the redundant indices. All DOFs $(\cup_{i=1}^{p_\ell} \mathcal{I}_{\ell,i}) \cup (\cup_{i=1}^{q_\ell} \widetilde{\mathcal{I}}_{\ell'',i})$ have now been eliminated up to level ℓ. The remaining active DOFs $\mathcal{S}_{\ell+1} = \mathcal{S}_{\ell''} \setminus \cup_{i=1}^{q_\ell} \widetilde{\mathcal{I}}_{\ell'',i}$ consist of only edge skeletons and corners, with the former typically clustering around the latter.

At the conclusion of this process, we have the final matrix

$$A_L \approx R_{L-1}^T \cdots R_0^T A R_0 \cdots R_{L-1}, \quad R_\ell = M_\ell C_\ell K_\ell \tag{3.4}$$

at the root, which is everywhere the identity except in the block indexed by the remaining active DOFs $\mathcal{S}_L$ containing, essentially, just the top-level corners. As a result, it is easily invertible, as is R_ℓ since each constituent factor is either block diagonal or triangular. Thus, we obtain the approximation F defined as follows

$$A \approx F \equiv R_0^{-T} \cdots R_{L-1}^{-T} A_L R_{L-1}^{-1} \cdots R_0^{-1}, \tag{3.5}$$

and, by applying the inverse on both sides,

$$A^{-1} \approx F^{-1} = R_0 \cdots R_{L-1} A_L^{-1} R_{L-1}^T \cdots R_0^T. \tag{3.6}$$

The factorization

$$F = GG^T, \quad G = R_0^{-T} \cdots R_{L-1}^{-T} B_L, \quad A_L = B_L B_L^T \tag{3.7}$$

is a *generalized Cholesky decomposition* (assuming that A_L is SPD), composed of a multilevel sequence of block sparse local matrices.

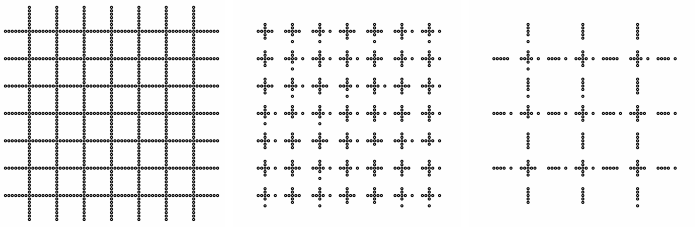
See Figure 3.4 for an example of the elimination process. Compared to HIF on the same problem (Figure 3.1), the skeletons are somewhat more numerous and disperse, but generally retain the same structure. Furthermore, we observe that the skeletonization rank ρ_ℓ at each level ℓ still scales as $\rho_\ell = O(\ell)$ (see Section 4) just as in HIF; consequently, the overall complexity is unchanged at $O(N)$ [21, Theorem 4.6].

3.2. Three dimensions. The algorithm can be extended to 3D in the natural way. Consider the analogous problem defined on $\Omega = (0,1)^3$ and discretized with the seven-point stencil. We now use an octree to hierarchically split Ω into $2^{3(L-\ell)}$ cubic cells at each level ℓ, each covering $(2^\ell m + 1)^3$ points. Among these,

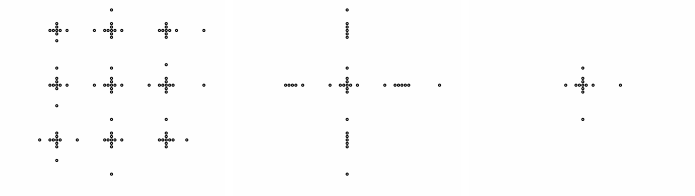
- $(2^\ell m - 1)^3$ are *interior* points;
- $6(2^\ell m - 1)$ are *face* points, where each face is shared between up to two cells;
- $12(2^\ell m - 1)$ are *edge* points, where each edge is shared between up to four cells and
- 8 are *corner* points, each shared between up to eight cells.

Denote by p_ℓ the total number of cells on level ℓ, $q_\ell \sim 3p_\ell$ the total number of faces, and $r_\ell \sim 7p_\ell$ the total number of faces, edges, and corners altogether. Then we analogously perform at each level ℓ:

(1) interior cell elimination for each of p_ℓ DOF sets;

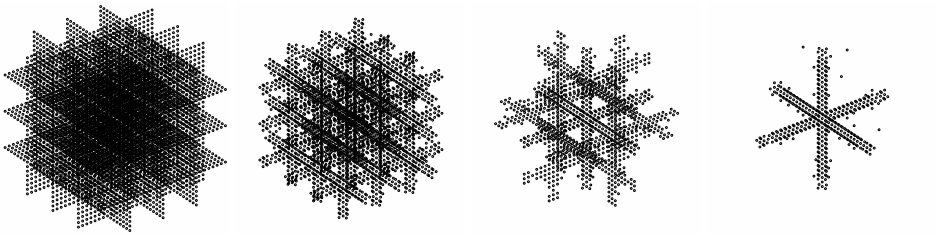


$\ell = 0$ cell elimination $\ell = 0$ skeletonization $\ell = 1$ cell elimination



$\ell = 1$ skeletonization $\ell = 2$ cell elimination $\ell = 2$ skeletonization

Fig. 3.4: *Active DOFs at each level ℓ of PHIF in 2D.*



$\ell = 0$ cell elimination $\ell = 0$ skeletonization $\ell = 1$ cell elimination $\ell = 1$ skeletonization

Fig. 3.5: *Active DOFs at each level ℓ of PHIF in 3D.*

(2) face, edge, and corner preconditioning for each of r_ℓ DOF sets and

(3) face skeletonization for each of q_ℓ DOF sets.

The remaining active DOFs tend to cluster around edges and corners. See Figure 3.5 for an example; as in 2D, the skeletons are slightly less ordered than for HIF (Figure 3.2). The skeletonization rank is $\rho_\ell = O(2^\ell)$ just as before, for a total cost of $O(N \log N)$ [21].

REMARK 3.1. In HIF, estimated $O(N)$ complexity can be achieved by additional edge skeletonization after face skeletonization to reduce the rank back down to $\rho_\ell = O(\ell)$. The same strategy presumably translates to PHIF, where we now would employ two extra steps:

- edge and corner preconditioning and
- edge skeletonization.

Note that we did not explore this approach here.

REMARK 3.2. As presently formulated, the ID-based skeletonization in Section 2.2 is actually not critical and other low-rank compression techniques may well be used, including the SVD [21]. The ID is required only for the edge skeletonization variant of Remark 3.1, which depends on the geometry being preserved.

3.3. Heuristic error analysis. In order to understand the impact of PHIF on the solve error, let us write each level of the algorithm more carefully as $A_{\ell+1} = R_\ell^T A_\ell R_\ell + E_\ell$, where $\|E_\ell\| = O(\epsilon\|A_\ell\|)$ is the approximation error from skeletonization to some relative precision ϵ. To ease the notation, we define a chain of matrix products with an ellipsis, e.g., $R_{L-1}^T \cdots R_{\ell+1}^T$, where the whole chain of products is taken to be the identity if the subscript of some matrix in the chain is greater than $L-1$. If both matrices in the endpoints are the same, the chain reduces to just one matrix, e.g., $R_{L-1}^T \cdots R_{L-1}^T = R_{L-1}^T$. Then

$$\begin{aligned} A_L &= R_{L-1}^T A_{L-1} R_{L-1} + E_{L-1} \\ &= R_{L-1}^T (R_{L-2}^T A_{L-2} R_{L-2} + E_{L-2}) R_{L-1} + E_{L-1} \\ &= R_{L-1}^T \cdots R_0^T A_0 R_0 \cdots R_{L-1} + \sum_{\ell=0}^{L-1} R_{L-1}^T \cdots R_{\ell+1}^T E_\ell R_{\ell+1} \cdots R_{L-1}, \end{aligned} \tag{3.8}$$

so

$$A = A_0 = R_0^{-T} \cdots R_{L-1}^{-T} A_L R_{L-1}^{-1} \cdots R_0^{-1} - \sum_{\ell=0}^{L-1} R_0^{-T} \cdots R_\ell^{-T} E_\ell R_\ell^{-1} \cdots R_0^{-1}, \tag{3.9}$$

with approximate factorization

$$F = GG^T, \quad G = R_0^{-T} \cdots R_{L-1}^{-T} B_L, \quad A_L = B_L B_L^T. \tag{3.10}$$

Hence the solve error is

$$\begin{aligned} \|I - G^{-1} A G^{-T}\| &= \|B_L^{-1}\| \left\| \sum_{\ell=0}^{L-1} R_{L-1}^T \cdots R_{\ell+1}^T E_\ell R_{\ell+1} \cdots R_{L-1} \right\| \|B_L^{-T}\| \\ &\lesssim \epsilon \|A_L^{-1}\| \sum_{\ell=0}^{L-1} \|A_\ell\| \|R_{\ell+1} \cdots R_{L-1}\|^2. \end{aligned} \tag{3.11}$$

To bound $\|A_\ell\|$, note that

$$A_\ell = R_\ell^{-T} \cdots R_{L-1}^{-T} A_L R_{L-1}^{-1} \cdots R_\ell^{-1} + O(\epsilon), \tag{3.12}$$

so

$$\|A_\ell\| \lesssim \|A_L\| \|R_\ell^{-1}\|^2 \|(R_{\ell+1} \cdots R_{L-1})^{-1}\|^2. \tag{3.13}$$

Combining with (3.11) yields

$$\|I - G^{-1} A G^{-T}\| \lesssim \epsilon \kappa(A_L) \sum_{\ell=0}^{L-1} \left[\|R_\ell^{-1}\| \kappa(R_{\ell+1} \cdots R_{L-1}) \right]^2. \tag{3.14}$$

The estimate (3.14) holds for both HIF and PHIF, given appropriate interpretation of R_ℓ and A_ℓ. We now contrast the two approaches by estimating and directly comparing each term.

In HIF, $R_\ell = M_\ell K_\ell$, where:

- $M_\ell = \prod_{i=1}^{p_\ell} M_{\mathcal{I}_{\ell,i}}$ with, referring to Section 2, the norm of each matrix and its inverse bounded as

$$\|M_{\mathcal{I}_{\ell,i}}^{\pm 1}\| \le \|L_{\mathcal{I}_{\ell,i}}^{\mp 1}\| \left(1 + \|L_{\mathcal{I}_{\ell,i}}^{-1}(A_\ell)_{\mathcal{B}_{\ell,i}\mathcal{I}_{\ell,i}}^T\|\right). \tag{3.15}$$

  Since M_ℓ is nearly block diagonal, with coupling between the $M_{\mathcal{I}_{\ell,i}}$ only along shared interfaces, we can expect

$$\begin{aligned}\kappa(M_\ell) &\lesssim \max_{i,j} \|L_{\mathcal{I}_{\ell,i}}\| \|L_{\mathcal{I}_{\ell,j}}^{-1}\| \left(1 + \|L_{\mathcal{I}_{\ell,i}}^{-1}(A_\ell)_{\mathcal{B}_{\ell,i}\mathcal{I}_{\ell,i}}^T\|\right)\left(1 + \|L_{\mathcal{I}_{\ell,j}}^{-1}(A_\ell)_{\mathcal{B}_{\ell,j}\mathcal{I}_{\ell,j}}^T\|\right) \\ &= \max_{i,j} \frac{\|L_{\mathcal{I}_{\ell,i}}\|}{\|L_{\mathcal{I}_{\ell,j}}\|} \kappa(L_{\mathcal{I}_{\ell,j}}) \left(1 + \|L_{\mathcal{I}_{\ell,i}}^{-1}(A_\ell)_{\mathcal{B}_{\ell,i}\mathcal{I}_{\ell,i}}^T\|\right)\left(1 + \|L_{\mathcal{I}_{\ell,j}}^{-1}(A_\ell)_{\mathcal{B}_{\ell,j}\mathcal{I}_{\ell,j}}^T\|\right).\end{aligned} \tag{3.16}$$

  In other words, $\kappa(M_\ell)$ depends on the uniformity as well as the conditioning of the $L_{\mathcal{I}_{\ell,i}}$.
- Similarly, $K_\ell = \prod_{i=1}^{q_\ell} K_{\mathcal{I}_{\ell'',i}}$ (following the above algorithm with $C_\ell = I$), where $K_{\mathcal{I}_{\ell'',i}} = Z_{\mathcal{I}_{\ell'',i}} M_{\widetilde{\mathcal{I}}_{\ell'',i}}$. But $\|Z_{\mathcal{I}_{\ell'',i}}^{\pm 1}\| \le 1 + \|T_{\mathcal{I}_{\ell'',i}}\|$, where, in practice, $\|T_{\mathcal{I}_{\ell',i}}\|$ is small due to the stability of the ID [8], so $\|Z_{\mathcal{I}_{\ell'',i}}^{\pm 1}\| = O(1)$ (or growing slowly with the size of the matrix). Hence, $\|K_{\mathcal{I}_{\ell'',i}}^{\pm 1}\| \lesssim \|M_{\widetilde{\mathcal{I}}_{\ell'',i}}^{\pm 1}\|$. Now use that K_ℓ is block diagonal since each $K_{\mathcal{I}_{\ell'',i}}$ acts on disjoint separators to obtain

$$\begin{aligned}\kappa(K_\ell) &\lesssim \max_{i,j} \|M_{\widetilde{\mathcal{I}}_{\ell'',i}}\| \|M_{\widetilde{\mathcal{I}}_{\ell'',j}}^{-1}\| \\ &\le \max_{i,j} \frac{\|L_{\widetilde{\mathcal{I}}_{\ell'',i}}\|}{\|L_{\widetilde{\mathcal{I}}_{\ell'',j}}\|} \kappa(L_{\widetilde{\mathcal{I}}_{\ell'',j}}) \left(1 + \|L_{\widetilde{\mathcal{I}}_{\ell'',i}}^{-1}(\widetilde{A_{\ell''}})_{\hat{\mathcal{I}}_{\ell'',i}\widetilde{\mathcal{I}}_{\ell'',i}}^T\|\right) \\ &\quad \left(1 + \|L_{\widetilde{\mathcal{I}}_{\ell'',j}}^{-1}(\widetilde{A_{\ell''}})_{\hat{\mathcal{I}}_{\ell'',j}\widetilde{\mathcal{I}}_{\ell'',j}}^T\|\right).\end{aligned} \tag{3.17}$$

On the other hand, in PHIF, $R_\ell = M_\ell C_\ell K_\ell$, where:

- M_0 at the initial level is the same as in HIF. At all subsequent levels, however, M_ℓ acts on an A_ℓ that has been preconditioned by $C_{\ell'}$ for all $\ell' < \ell$. This has the effect of driving $\|L_{\mathcal{I}_{\ell,i}}\|$ towards 1 (since the corresponding $(A_\ell)_{\mathcal{I}_{\ell,i}\mathcal{I}_{\ell,i}}$ has been rescaled to $O(1)$) as well as reducing $\kappa(L_{\mathcal{I}_{\ell,i}})$. Furthermore, if A is SPD, then the off-diagonal term $\|(A_\ell)_{\mathcal{B}_{\ell,i}\mathcal{I}_{\ell,i}}\|$ also cannot be too large. Thus, we can expect each quantity in (3.16) to be better behaved in PHIF than in HIF.
- $C_\ell = \prod_{i=1}^{r_\ell} C_{\ell,i}$, where clearly $\kappa(C_\ell) = \max_{i,j} \|L_{\mathcal{I}_{\ell',i}}\| \|L_{\mathcal{I}_{\ell',j}}^{-1}\|$. As with M_ℓ, each C_ℓ for $\ell > 0$ is typically not too ill-conditioned due to preconditioning at previous levels. In effect, this is similar to the $\kappa(K_\ell)$ term in HIF but can be much better for $\ell > 0$ due to prior preconditioning.
- Again, $\kappa(K_\ell)$ can be bounded as in (3.17), but now with each term well-behaved following the same remarks as for M_ℓ. In fact, we argue that $\kappa(K_\ell) = O(1)$ since the post-preconditioned submatrix on which each elimination operator acts has the form

$$\begin{bmatrix} (\widetilde{A_{\ell''}})_{\widetilde{\mathcal{I}}_{\ell'',i}\widetilde{\mathcal{I}}_{\ell'',i}} & (\widetilde{A_{\ell''}})_{\hat{\mathcal{I}}_{\ell'',i}\widetilde{\mathcal{I}}_{\ell'',i}}^T \\ (\widetilde{A_{\ell''}})_{\hat{\mathcal{I}}_{\ell'',i}\widetilde{\mathcal{I}}_{\ell'',i}} & (\widetilde{A_{\ell''}})_{\hat{\mathcal{I}}_{\ell'',i}\hat{\mathcal{I}}_{\ell'',i}} \end{bmatrix} = \begin{bmatrix} I + T_{\mathcal{I}_{\ell'',i}}^T T_{\mathcal{I}_{\ell'',i}} & -T_{\mathcal{I}_{\ell'',i}}^T \\ -T_{\mathcal{I}_{\ell'',i}} & I \end{bmatrix}, \tag{3.18}$$

where all terms are $O(1)$. In particular, $\|L_{\widetilde{\mathcal{I}}_{\ell'',i}}\| = O(1)$. For the inverse, note that $(\widetilde{A_{\ell''}})_{\widetilde{\mathcal{I}}_{\ell'',i},\widetilde{\mathcal{I}}_{\ell'',i}}$ is a positive-definite perturbation of the identity and so has minimal eigenvalue at least 1. Hence,

$$\|L^{-1}_{\widetilde{\mathcal{I}}_{\ell'',i}}\| = \lambda_{\min}^{-1/2}\left((\widetilde{A_{\ell''}})_{\widetilde{\mathcal{I}}_{\ell'',i},\widetilde{\mathcal{I}}_{\ell'',i}}\right) \leq 1. \tag{3.19}$$

Using these in (3.17) completes the claim.

Putting everything together, we have that

$$\frac{\kappa'(R_\ell)}{\kappa(R_\ell)} \sim \frac{\kappa'(M_\ell)}{\kappa(M_\ell)}\kappa'(C_\ell)\frac{\kappa'(K_\ell)}{\kappa(K_\ell)} \lesssim \frac{\kappa'(C_\ell)}{\kappa(K_\ell)} \lesssim 1, \tag{3.20}$$

where $\kappa(\cdot)$ and $\kappa'(\cdot)$ denote, respectively, quantities for HIF and PHIF. Looking at (3.14), it remains to compare $\|R_\ell^{-1}\|$ and $\kappa(A_L)$. For the former, a simplified analysis using $\|L_{\mathcal{I}_{\ell,i}}\| \sim O(1)$ in PHIF and omitting some lengthy terms of the form $(1+\|L^{-1}A^T\|)$, which are worse in HIF than PHIF, gives

$$\|R_\ell^{-1}\| \lesssim \begin{cases} \max_{i,j}\|L_{\mathcal{I}_{\ell,i}}\|\|L_{\widetilde{\mathcal{I}}_{\ell'',j}}\| & \text{(HIF)} \\ \max_i \|L_{\mathcal{I}_{\ell',i}}\| & \text{(PHIF)} \end{cases} \tag{3.21}$$

which we argue is smaller for PHIF than HIF by similar reasoning as for comparing C_ℓ in PHIF with K_ℓ in HIF above. Finally, we have $\kappa(A_L)$, which, of course, is typically much improved in PHIF since it has been preconditioned throughout.

This analysis provides some insight on the essential feature of the algorithm: whereas A_L can be very ill-conditioned in HIF, inheriting in large part from A itself, in PHIF it results from multiple levels of preconditioning through the action of the rescaling operators C_ℓ and so should be better behaved—in certain cases, much better, by several orders of magnitude, as we will see in Section 4.

It is also instructive to revisit Remark 2.1 on the required accuracy to remain SPD in light of PHIF. Applying the remark, the intermediate matrices A_ℓ are all SPD and so the algorithm succeeds if $\|E_\ell\| < \lambda_{\min}(A_\ell) = 1/\|A_\ell^{-1}\|$; for $\|E_\ell\| = O(\epsilon\|A_\ell\|)$ as above, this gives $\epsilon \lesssim 1/\max_\ell \kappa(A_\ell)$. Naturally, $\kappa(A_\ell)$ can be much smaller in PHIF than HIF, thereby significantly alleviating this issue.

4. Numerical results

We now present some benchmark examples to demonstrate the performance of PHIF on strongly ill-conditioned problems. Specifically, we consider the PDE (1.1) on $\Omega = (0,1)^d$, discretized as in Section 3 with $b(x) \equiv 0$ and $a(x)$ a quantized high-contrast random field defined as follows:

(1) Initialize by sampling each grid point $a_j = a(x_j)$ from the standard uniform distribution.

(2) Convolve with an isotropic Gaussian of width $4h$ to create some correlation structure.

(3) Quantize by setting

$$a_j = \begin{cases} 10^{-2}, & a_j \leq \mu, \\ 10^2, & a_j > \mu, \end{cases} \tag{4.1}$$

where μ is the median of $\{a_j\}$.

The resulting matrix A in (1.2) has condition number $\kappa(A) = O(\sigma h^{-2}) = O(\sigma N^{2/d})$, where $\sigma = 10^4$ is the contrast ratio.

In the following, we report

- ϵ: relative precision for ID compression;
- N: total number of DOFs in the problem;
- $|\mathcal{S}_L|$: number of active DOFs remaining at the highest level;
- e_a: estimated apply error $\|A - F\|/\|A\|$;
- e_s: estimated solve error $\|I - G^{-1}AG^{-T}\| \sim \|I - AF^{-1}\| \geq \|A^{-1} - F^{-1}\|/\|A^{-1}\|$ and
- n_i: number of CG iterations using F^{-1} as a preconditioner to solve to a relative residual of 10^{-12}.

The errors e_a and e_s are estimated using randomized power iteration [11, 23] to 10^{-2} relative precision. All numerical experiments are performed in MATLAB using codes modified from FLAM [19].

4.1. Two dimensions. Consider first the example in 2D. Numerical results are given in Table 4.1 with scaling plots shown in Figure 4.1. We immediately see the effectiveness of PHIF at improving e_s (by 10^2–10^3) while maintaining comparable $e_a = O(\epsilon)$ with HIF. This directly manifests in a smaller n_i across all cases tested. We also observe the intermediate matrices encountered throughout PHIF to be much better conditioned, in agreement with Section 3.3. Likewise, PHIF exhibits greater robustness with respect to remaining SPD; indeed, while HIF fails for $\epsilon = 10^{-4}$, PHIF is still able to produce good preconditioners at the same tolerance.

		HIF				PHIF			
ϵ	N	$\lvert\mathcal{S}_L\rvert$	e_a	e_s	n_i	$\lvert\mathcal{S}_L\rvert$	e_a	e_s	n_i
10^{-4}	1023^2	not SPD				60	4.7e−5	1.4e−1	9
	2047^2					68	6.0e−5	1.6e−1	12
	4095^2					77	7.6e−5	2.4e−1	14
	8191^2					86	7.7e−5	5.7e−1	17
10^{-6}	1023^2	59	2.9e−6	7.3e−1	16	81	4.9e−7	1.1e−3	4
	2047^2	61	2.9e−6	8.5e−1	20	91	6.9e−7	1.5e−3	4
	4095^2	63	4.2e−6	9.3e−1	32	111	8.5e−7	2.2e−3	5
	8191^2	46	5.9e−6	9.7e−1	54	125	9.8e−7	3.5e−3	5
10^{-8}	1023^2	80	3.1e−8	2.9e−3	4	105	6.5e−9	7.1e−6	4
	2047^2	82	3.2e−8	5.3e−3	5	118	8.2e−9	1.8e−5	3
	4095^2	103	3.5e−8	1.1e−2	5	133	9.7e−9	3.1e−5	3
	8191^2	112	4.4e−8	2.1e−2	6	154	1.1e−8	2.8e−5	3

Table 4.1: *Numerical results for 2D example. HIF at $\epsilon = 10^{-4}$ fails due to loss of positive definiteness.*

As with HIF, PHIF achieves linear complexity, as is evident from the figure and from verifying the mild growth of $|\mathcal{S}_L|$ with N. However, at fixed ϵ, PHIF can be quite a bit more expensive, due primarily to the extra Cholesky preconditioning steps. For example, at $N = 8191^2$ and $\epsilon = 10^{-6}$, these (1456 s) account for about 44% of the total PHIF factorization time (3317 s) or 78% of the equivalent HIF time (1869 s); in other

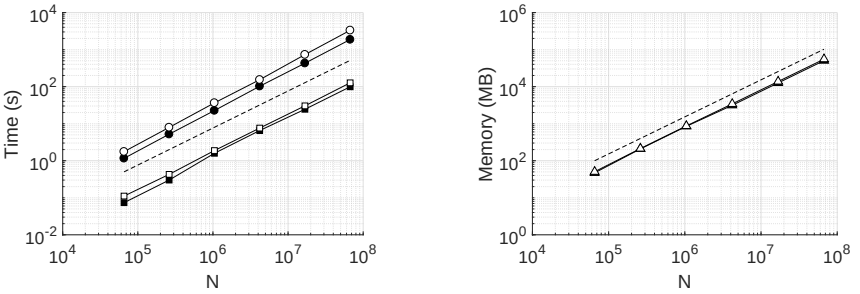


Fig. 4.1: *Scaling results for 2D example, showing factorization times (○), application times for $F^{-1}b$ (□), and memory storage (△) for HIF (black) and PHIF (white) at $\epsilon = 10^{-6}$. Reference $O(N)$ lines (dashed) are also included.*

words, just the multilevel preconditioning itself can almost double the factorization cost without even considering the potential added impact of larger skeleton sizes. Despite this overhead, PHIF is still able to decrease the total time to solution: for the same case, after factorization, HIF takes 5698 s to run $n_i = 54$ iterations for a total of 7567 s, while PHIF needs only 1199 s over $n_i = 5$ for a total of 4516 s — a reduction of close to 40%.

In a sense, the behavior of PHIF is similar to that of HIF computed at a higher precision: e_s is improved at the cost of increased ranks. But careful inspection of the results suggests that the interpretation is not quite so simple. Comparing, e.g., $N = 1023^2$ for PHIF at $\epsilon = 10^{-4}$ vs. HIF at $\epsilon = 10^{-6}$ and for PHIF at $\epsilon = 10^{-6}$ vs. HIF at $\epsilon = 10^{-8}$ reveals that PHIF can use approximately the same final rank to achieve better e_s. At the same time, e_a is slightly worse, thus indicating a certain balance between the forward and inverse errors. We will see such nuances more clearly in the next example as well.

One can also compare the performance of PHIF, when acting as a preconditioner, to that of incomplete Cholesky. As an example, with $N = 2047^2$, CG converges in 12 iterations using PHIF at $\epsilon = 10^{-4}$. Meanwhile, incomplete Cholesky with threshold dropping at $\epsilon = 10^{-5}$ has a similar memory footprint as PHIF at $\epsilon = 10^{-4}$, but CG converges in 235 steps.

4.2. Three dimensions. Now consider the analogous problem in 3D. Numerical results are shown in Table 4.2 with scaling plots in Figure 4.2. The overall conclusions are very similar as before, with PHIF yielding smaller e_s and n_i, especially at low accuracy. We highlight in particular the comparison between PHIF at $\epsilon = 10^{-2}$ and HIF at $\epsilon = 10^{-6}$, in which PHIF achieves comparable n_i despite significantly fewer skeletons $|\mathcal{S}_L|$ and worse e_s. Furthermore, note the rapid growth of e_s with increasing N for HIF; this is suppressed to a large extent in PHIF, hence producing effective preconditioners that maintain essentially constant n_i.

Similar to the 2D example, PHIF works better than threshold-based incomplete Cholesky as a preconditioner to CG. In particular, for $N = 127^3$, CG with PHIF at $\epsilon = 10^{-2}$ converges in 26 iterations. However, it takes 50 iterations to converge using incomplete Cholesky with threshold dropping and similar memory footprint.

The data certify that $|\mathcal{S}_L|$ scales as $O(N^{1/3})$, corresponding to formal $O(N \log N)$ complexity. However, we find only an empirical scaling of roughly $O(N^{1.4})$ for both HIF and PHIF. This is most likely due to non-asymptotic effects; at any rate, PHIF does not appear to incur any additional asymptotic cost.

		HIF				PHIF			
ϵ	N	$\lvert\mathcal{S}_L\rvert$	e_a	e_s	n_i	$\lvert\mathcal{S}_L\rvert$	e_a	e_s	n_i
10^{-2}	31^3	638	1.7e−2	1.0e+0	25	773	3.5e−3	8.4e−1	9
	63^3	1324	1.9e−2	1.0e+0	43	1716	5.7e−3	1.0e+0	14
	127^3	2605	1.8e−2	9.9e−1	89	3585	8.1e−3	1.0e+0	26
	255^3	4477	1.7e−2	1.0e+0	203	7026	7.5e−3	9.9e−1	43
10^{-6}	31^3	1422	9.3e−7	6.1e−3	10	1573	5.5e−7	1.2e−4	4
	63^3	3235	2.8e−6	5.4e−2	13	3775	1.7e−6	2.8e−4	3
	127^3	6809	1.4e−5	4.3e−1	16	8792	2.8e−6	5.6e−4	3
	255^3	13726	2.8e−5	7.1e−1	18	18412	2.3e−6	1.3e−3	3
10^{-10}	31^3	1967	1.2e−10	6.7e−6	2	2169	8.6e−11	1.3e−8	3
	63^3	5112	2.5e−10	2.2e−5	5	5684	1.4e−10	1.7e−8	3
	127^3	12559	7.2e−10	1.1e−4	8	14204	3.0e−10	3.0e−8	3
	255^3	25968	1.2e−9	1.3e−4	8	30946	4.0e−10	3.8e−8	3

Table 4.2: *Numerical results for 3D example.*

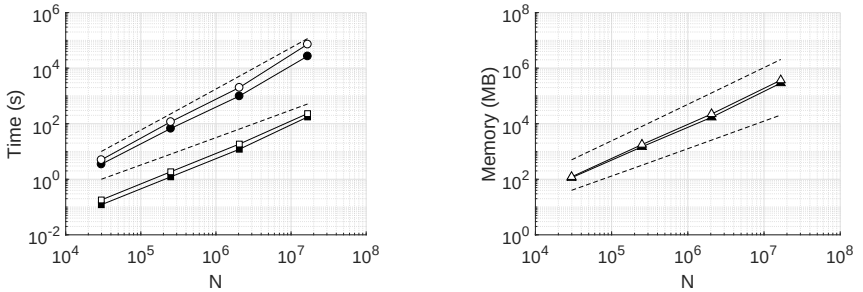


Fig. 4.2: *Scaling results for 3D example at* $\epsilon = 10^{-6}$*; notation as in Figure 4.1. Reference lines (dashed) include* $O(N)$ *and* $O(N^{3/2})$ *for timings, and* $O(N)$ *and* $O(N^{4/3})$ *for storage.*

5. Conclusions

In this paper, we have presented PHIF, a recursively preconditioned version of HIF based on simply adding a block Jacobi preconditioning step before each level of skeletonization. This leads to dramatic improvements in the solve accuracy and therefore in its effectiveness as a direct solver or preconditioner, especially for ill-conditioned problems. Importantly, it retains the near-optimal computational complexity of HIF; this makes it the first of what we anticipate will become a highly successful emerging class of fast structured ND methods with enhanced robustness to the underlying system conditioning.

It is worth emphasizing that PHIF is much more than just applying HIF to a standard "one-level" preconditioned matrix. Rather, it seeks to control the condition number at all levels, similarly to [1, 30, 39–41]. Although this control is presently not as tight as in [1, 39–41], recall that we are addressing a fundamentally different problem with higher-dimensional geometric effects. Future work may be able to bridge this gap.

The PHIF framework is quite general and can be modified in various ways. For

instance, any local preconditioner, e.g., diagonal, can be used in place of block Cholesky; this can help to save on computational costs, as discussed in Section 4. Furthermore, the algorithm can, in principle, extend to non-symmetric and indefinite matrices via local LU factorizations, though some subtleties may inevitably arise. More research is required to fully settle such matters and also to explore other important extensions including to structured dense matrices. On this latter point in particular, fast 2D and 3D integral equation solvers like [22, 28] are based on the same style of multilevel skeletonization; to what extent can the same preconditioning ideas apply? The principal roadblock would appear to be the cost of modifying the far field; a clever trick to bypass this could be of great significance.

Acknowledgements. The work of J.F. is partially supported by "la Caixa" Fellowship, LCF/BQ/AA16/11580045, sponsored by "la Caixa" Banking Foundation. The work of L.Y. is partially supported by U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program and the National Science Foundation under award DMS-1818449.

REFERENCES

[1] E. Agullo, E. Darve, L. Giraud, and Y. Harness, *Low-rank factorizations in data sparse hierarchical algorithms for preconditioning symmetric positive definite matrices*, SIAM J. Matrix Anal. Appl., 39(4):1701–1725, 2018. 1.1, 1.1, 1.2, 1.2, 5

[2] 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(3):A1451–A1474, 2015. 1.1

[3] A. Aminfar, S. Ambikasaran, and E. Darve, *A fast block low-rank dense solver with applications to finite-element matrices*, J. Comput. Phys., 304:170–188, 2016. 1.1

[4] M. Bebendorf, M. Bollhöfer, and M. Bratsch, *Hierarchical matrix approximation with blockwise constraints*, BIT Numer. Math., 53(2):311–339, 2013. 1.1

[5] M. Bebendorf, M. Bollhöfer, and M. Bratsch, *On the spectral equivalence of hierarchical matrix preconditioners for elliptic problems*, Math. Comp., 85:2839–2861, 2016. 1.1

[6] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31(138):333–390, 1977. 1.1

[7] S. Chandrasekaran, P. Dewilde, M. Gu, and N. Somasunderam, *On the numerical rank of the off-diagonal blocks of Schur complements of discretized elliptic PDEs*, SIAM J. Matrix Anal. Appl., 31(5):2261–2290, 2010. 1.1

[8] H. Cheng, Z. Gimbutas, P.G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM J. Sci. Comput., 26(4):1389–1404, 2005. 2.2, 3.3

[9] E. Corona, P.-G. Martinsson, and D. Zorin, *An $O(N)$ direct solver for integral equations on the plane*, Appl. Comput. Harmon. Anal., 38(2):284–317, 2015. 1.1

[10] P. Coulier, H. Pouransari, and E. Darve, *The inverse fast multipole method: Using a fast approximate direct solver as a preconditioner for dense linear systems*, SIAM J. Sci. Comput., 39(3):A761–A796, 2017. 1.1

[11] J.D. Dixon, *Estimating extremal eigenvalues and condition numbers of matrices*, SIAM J. Numer. Anal., 20(4):812–814, 1983. 4

[12] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10(2):345–363, 1973. 1.1

[13] A. Gillman and P.G. Martinsson, *A direct solver with $O(N)$ complexity for variable coefficient elliptic PDEs discretized via a high-order composite spectral collocation method*, SIAM J. Sci. Comput., 36(4):A2023–A2046, 2014. 1.1

[14] G.H. Golub and C.F. van Loan, *Matrix Computations*, Third Edition, Johns Hopkins, Baltimore, 1996. 1.1

[15] L. Grasedyck, R. Kriemann, and S. Le Borne, *Domain decomposition based $\mathcal{H}$-LU preconditioning*, Numer. Math., 112(4):565–600, 2009. 1.1

[16] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer Series Comput. Math., Springer, Berlin, 4, 1985. 1.1

[17] S. Hao and P.-G. Martinsson, *A direct solver for elliptic PDEs in three dimensions based on*

hierarchical merging of Poincaré-Steklov operators, J. Comput. Appl. Math., 308:419–434, 2016. 1.1

[18] M.R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Standards, 49:409–436, 1952. 1.1

[19] K.L. Ho, *FLAM: Fast linear algebra in MATLAB*, 2018. 4

[20] K.L. Ho and L. Greengard, *A fast direct solver for structured linear systems by recursive skeletonization*, SIAM J. Sci. Comput., 34(5):A2507–A2532, 2012. 1.1

[21] K.L. Ho and L. Ying, *Hierarchical interpolative factorization for elliptic operators: Differential equations*, Comm. Pure Appl. Math., 69(8):1415–1451, 2016. 1.1, 1.2, 3, 3.1, 3.2, 3.2

[22] K.L. Ho and L. Ying, *Hierarchical interpolative factorization for elliptic operators: Integral equations*, Comm. Pure Appl. Math., 69(7):1314–1353, 2016. 1.1, 5

[23] J. Kuczyński and H. Woźniakowski, *Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start*, SIAM J. Matrix Anal. Appl., 13(4):1094–1122, 1992. 4

[24] R. Li and Y. Saad, *Divide and conquer low-rank preconditioners for symmetric matrices*, SIAM J. Sci. Comput., 35(4):A2069–A2095, 2013. 1.1

[25] Y. Li and L. Ying, *Distributed-memory hierarchical interpolative factorization*, Res. Math. Sci., 4:12, 2017. 2.2

[26] P.-G. Martinsson, *A fast direct solver for a class of elliptic partial differential equations*, J. Sci. Comput., 38(3):316–330, 2009. 1.1

[27] P.G. Martinsson and V. Rokhlin, *A fast direct solver for boundary integral equations in two dimensions*, J. Comput. Phys., 205(1):1–23, 2005. 1.1

[28] V. Minden, K.L. Ho, A. Damle, and L. Ying, *A recursive skeletonization factorization based on strong admissibility*, Multiscale Model. Simul., 15(2):768–796, 2017. 1.1, 5

[29] A. Napov, *Conditioning analysis of incomplete Cholesky factorizations with orthogonal dropping*, SIAM J. Matrix Anal. Appl., 34(3):1148–1173, 2013. 1.1

[30] H. Owhadi, *Multigrid with rough coefficients and multiresolution operator decomposition from hierarchical information games*, SIAM Rev., 59(1):99–149, 2017. 5

[31] 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(3):A797–A830, 2017. 1.1

[32] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Second Edition, SIAM, Philadelphia, 2003. 1.1

[33] P.G. Schmitz and L. Ying, *A fast direct solver for elliptic problems on general meshes in 2D*, J. Comput. Phys., 231(4):1314–1338, 2012. 1.1

[34] P.G. Schmitz and L. Ying, *A fast nested dissection solver for Cartesian 3D elliptic problems using hierarchical matrices*, J. Comput. Phys., 258:227–245, 2014. 1.1

[35] D.A. Sushnikova and I.V. Oseledets, *"Compress and eliminate" solver for symmetric positive definite sparse matrices*, SIAM J. Sci. Comput., 40(3):A1742–A1762, 2018. 1.1

[36] J. Xia, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM J. Sci. Comput., 35(2):A832–A860, 2013. 1.1

[37] J. Xia, *Randomized sparse direct solvers*, SIAM J. Matrix Anal. Appl., 34(1):197–227, 2013. 1.1

[38] J. Xia, S. Chandrasekaran, M. Gu, and X.S. Li, *Superfast multifrontal method for large structured linear systems of equations*, SIAM J. Matrix Anal. Appl., 31(3):1382–1411, 2009. 1.1

[39] J. Xia and M. Gu, *Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices*, SIAM J. Matrix Anal. Appl., 31(5):2899–2920, 2010. 1.1, 1.1, 5

[40] J. Xia and Z. Xin, *Effective and robust preconditioning of general SPD matrices via structured incomplete factorization*, SIAM J. Matrix Anal. Appl., 38(4):1298–1322, 2017. 1.1, 1.1, 5

[41] X. Xing and E. Chow, *Preserving positive definiteness in hierarchically semiseparable matrix approximations*, SIAM J. Matrix Anal. Appl., 39(2):829–855, 2018. 1.1, 1.1, 5

[42] K. Yang, H. Pouransari, and E. Darve, *Sparse hierarchical solvers with guaranteed convergence*, Int. J. Numer. Meth. Eng., 120(8):964–986, 2019. 1.1