SIAM J. MATRIX ANAL. APPL.
Vol. 36, No. 3, pp. 1283–1314

FAST SPARSE SELECTED INVERSION*

JIANLIN XIA†, YUANZHE XI†, STEPHEN CAULEY‡, AND VENKATARAMANAN BALAKRISHNAN§

Abstract. We propose a fast structured selected inversion method for extracting the diagonal blocks of the inverse of a sparse symmetric matrix A, using the multifrontal method and rank structures. When A arises from the discretization of some PDEs and has a low-rank property (the intermediate dense matrices in the factorization have small off-diagonal numerical ranks), structured approximations of the diagonal blocks and certain off-diagonal blocks of A^{-1} (that are needed to find the diagonal blocks of A^{-1}) can be quickly computed. A structured multifrontal LDL factorization is first computed for A with a forward traversal of an assembly tree, which yields a sequence of local data-sparse factors. The factors are used in a backward traversal of the tree for the structured inversion. The intermediate operations in the inversion are performed in hierarchically semiseparable or low-rank forms. With the assumptions of data sparsity and appropriate rank conditions, the theoretical structured inversion cost is proportional to the matrix size n times a low-degree polylogarithmic function of n after structured factorizations. The memory counts are similar. In comparison, existing direct selected inversion methods cost $O(n^{3/2})$ flops in two dimensions and $O(n^2)$ flops in three dimensions for both the factorization and the inversion, with $O(n^{4/3})$ memory in three dimensions. Additional formulas for efficient structured operations are also derived. Numerical tests on two- and three-dimensional discretized PDEs and more general sparse matrices are done to demonstrate the performance.

Key words. fast selected inversion, data sparsity, structured multifrontal method, low-rank property, HSS matrix, linear complexity

AMS subject classifications. 15A23, 65F05, 65F30, 65F50

DOI. 10.1137/14095755X

1. Introduction.

Extracting selected entries (often the diagonal) of the inverse of a sparse matrix, usually called selected inversion, is critical in many scientific computing problems. Examples include preconditioning, uncertainty quantification in risk analysis [3], electronic structure calculations within the density functional theory framework [26], and simulations of nanotransistors and silicon nanowires [8]. The diagonal entries of the inverse are useful in providing significant insights into the original problem or its solution.

The goal of this paper is to present an efficient structured method for computing the diagonal of A^{-1} (as a column vector, denoted by $\mathrm{diag}(A^{-1})$), as well as the diagonal blocks of A^{-1} for a large $n \times n$ sparse symmetric matrix A. The method also produces some off-diagonal blocks of A^{-1}. For convenience, we usually just mention $\mathrm{diag}(A^{-1})$.

In recent years, a lot of effort has been made in the extraction of $\mathrm{diag}(A^{-1})$. Since A^{-1} is often fully dense, a brute-force formation of A^{-1} is generally impractical. On the other hand, it is possible to find $\mathrm{diag}(A^{-1})$ without forming the entire inverse.

*Received by the editors February 18, 2014; accepted for publication (in revised form) by S. Le Borne July 7, 2015; published electronically September 1, 2015.
http://www.siam.org/journals/simax/36-3/95755.html

†Department of Mathematics and Department of Computer Science, Purdue University, West Lafayette, IN 47907 (xiaj@math.purdue.edu, yxi@math.purdue.edu). The research of the first author was supported in part by an NSF CAREER Award DMS-1255416 and an NSF grant DMS-1115572.

‡Athinoula A. Martinos Center for Biomedical Imaging, Department of Radiology, Massachusetts General Hospital, Harvard University, Charlestown, MA 02129 (stcauley@nmr.mgh.harvard.edu).

§School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 (ragu@ecn.purdue.edu).

If A is diagonally dominant or positive definite, A^{-1} may have many small entries. Based on this property, a probing method is proposed in [33]. It exploits the pattern of a sparsified A^{-1} together with some standard graph theories, and computes $\mathrm{diag}(A^{-1})$ by solving a sequence of linear systems with a preconditioned Krylov subspace method. Later, several approaches were proposed for more general matrices. The fast inverse with nested dissection method in [23, 24] and the selected inversion method in [25] use domain decomposition and compute some hierarchical Schur complements of the interior points for each subdomain. This is followed by the extraction of the diagonal entries in a top-down pass. The method Selinv in [26, 27] uses a supernode left-looking LDL factorization of A to improve the efficiency. The work in [1] focuses on the computation of a subset of A^{-1} by accessing only part of the factors where the LU or LDL factorization of A is held in out-of-core storage. All these methods in [1, 23, 25, 26] are direct methods. For iterative methods, a Lanczos-type algorithm is first used in [35]. Later, a divide-and-conquer (DC) method and a domain decomposition (DD) method are presented in [34]. The DC method assumes that the matrix can be recursively decomposed into a 2×2 block-diagonal plus low-rank form, where the decomposed problem is solved and corrected by the Sherman–Morrison–Woodbury (SMW) formula at each recursion level. The DD method solves each local subdomain problem and then modifies the result by a global Schur complement. Both methods use iterative solvers and sparse approximation techniques to speed up the computations.

Our scheme for the selected inversion is a direct method. As in [25, 26, 27], the scheme includes two stages, a block LDL factorization stage and an inversion stage. We incorporate various sparse and structured matrix techniques, especially a structured multifrontal factorization and a structured selected inversion, to gain significant efficiency and storage benefits, as outlined below.

(1) *General sparse matrices and the multifrontal method.* Our method is applicable to symmetric discretized matrices on both two-dimensional (2D) and three-dimensional (3D) domains, as well as more general symmetric sparse matrices (as long as certain rank structures exist, as explained in the next item). Just like in modern sparse direct solvers, we apply the nested dissection ordering [14] to the mesh or adjacency graph by calling some graph partitioning tools. Thus, our inversion method does not rely on the special shape of the computational domain, and is more generally applicable than those in [25, 34].

To enhance the data locality of the later inversion, we use the multifrontal method [13] to compute a block LDL factorization of A. This method converts the overall sparse factorization into a sequence of operations on some local dense matrices called frontal matrices, following a nice tree structure called assembly tree. This makes it convenient to manage and access data in the inversion.

(2) *Fast structured factorization stage.* To speed up the factorization (and the later inversion), we further incorporate rank structured techniques. For problems such as some discretized elliptic/elasticity equations and Helmholtz equations with low or medium frequencies, the dense intermediate matrices in the sparse factorization often have certain rank structures (see, e.g., [2, 11, 16, 17, 29, 31, 41]). For these cases, the problem or the discretized matrix A is often said to have a *low-rank property*. We perform the LDL factorization of A with the structured multifrontal methods in [39, 40, 41], which further approximate the frontal matrices in the multifrontal method by hierarchically semiseparable (HSS) forms [10, 42]. Such HSS forms take much less storage than the original dense ones. The local dense operations are thus converted into a series of fast structured ones. The complexity of the LDL factorization is

$O(n)$ times a low-degree polylogarithmic function of n in two dimensions, and $O(n)$ to $O(n^{4/3})$ times a low-degree polylogarithmic function of n in three dimensions, depending on certain rank conditions. Moreover, after the factorization, the factors are in data-sparse forms and with storage roughly proportional to n, which makes the later efficient inversion feasible.

Related to the later inversion, we also show a fast Schur complement computation strategy in Theorem 3.4 using a concept similar to the reduced HSS matrix in [39]. The formula takes advantage of an HSS inverse computation that is needed in the inversion stage, and significantly saves the Schur complement computation cost in the factorization stage.

(3) *Fast structured selected inversion stage.* After the structured multifrontal factorization, the factors are represented by a sequence of HSS or low-rank forms. This yields a structured approximation $\tilde{A}$ to A. The inversion procedure is performed on these data-sparse factors instead of the original dense ones. In Theorem 3.8, we show the rank structures in the diagonal blocks and selected off-diagonal blocks of $\tilde{A}^{-1}$. They are either HSS or low-rank forms. The major operations of the inversion are then HSS inversions, multiplications of HSS and low-rank matrices, and low-rank updates, which can all be quickly performed. This thus significantly improves the efficiency and the storage over the standard direct selected inversion. In Theorem 3.5, we also derive another formula to quickly apply the inverse of the leading part of an HSS form to its off-diagonal part, as needed in the inversion.

(4) *Nearly linear theoretical complexity for selected inversion.* The complexity of the inversion algorithm is analyzed with a complexity optimization strategy and a rank relaxation idea in [39]. Unlike in [39, 41] where the factorization cost is optimized, here we minimize the selected inversion cost. A switching level in the assembly tree is used to shift from dense local inversions to HSS ones. The structured selected inversion has almost linear complexity for the discretized matrices with the low-rank property in both two and three dimensions. More specifically, if the off-diagonal numerical ranks of the intermediate frontal matrices are bounded by r, then the selected inversion cost is $O(rn)$ in two dimensions and $O(r^{3/2}n)$ in three dimensions. If the off-diagonal numerical ranks grow with n following the rank patterns in Tables 4–6, then the theoretical selected inversion cost and the storage are proportional to n times low-degree polylogarithmic functions of n. In contrast, the methods in [23, 25, 26, 27] cost $O(n^{3/2})$ in two dimensions and $O(n^2)$ in three dimensions, for both the LDL factorization and the selected inversion, and need $O(n^{4/3})$ storage in three dimensions. Due to the data sparsity, our method has the potential to be extended to the fast extraction of general off-diagonal entries of A^{-1}.

Numerical tests in terms of discretized Helmholtz equations in both two and three dimensions as well as various more general problems from a sparse matrix collection are performed. Significant performance gain is observed for the structured selected inversion over the standard direct one.

We would like to mention that an alternative way is to use $\mathcal{H}$- or $\mathcal{H}^2$-matrices [2, 6, 7, 17, 20, 22], especially for 2D and 3D discretized PDEs. That is, the discretized operators can be first approximated by $\mathcal{H}$- or $\mathcal{H}^2$-matrices and then a related inversion procedure is applied. In fact, the methods in [17, 22] involving nested dissection and the one in [20] involving weak admissibility conditions share some concepts similar to ours. In particular, if $\mathcal{H}^2$-matrix techniques are applied to certain algebraic operators, strict $O(n)$ inversion complexity is possible. Here by using the multifrontal method, we seek to take advantage of its data locality and related well-studied graph techniques for sparse matrices.

The remaining sections of the paper are organized as follows. Section 2 briefly reviews the structured multifrontal factorization with HSS techniques. Section 3 describes the structured selected inversion algorithm in detail. We discuss the complexity optimization in section 4. The numerical results are shown in section 5. Section 6 concludes the paper. In the presentation, we use the following notation:

- $A|_{s\times t}$ represents a submatrix of A specified by the row index set s and the column index set t, and $A|_s$ consists of selected rows of F specified by s;
- $\mathrm{diag}(D)$ denotes the diagonal entries of D as a column vector; on the other hand, $\mathrm{diag}(D_1, D_2)$ denotes a block diagonal matrix with the diagonal blocks D_1 and D_2;
- T represents a full postordered binary tree with its nodes denoted by $i = 1, 2, \ldots, \mathrm{root}(T)$, where $\mathrm{root}(T)$ is the root;
- c_1 and c_2 denote the left and the right children of a nonleaf node i in T, respectively;
- $\mathrm{sib}(i)$ and $\mathrm{par}(i)$ denote the sibling and the parent of a node i, respectively;
- boldface symbols such as $\mathbf{F}$ and $\mathbf{S}$ are for the dense matrices inside the multifrontal factorization, and $\mathbf{i}$ is for a node of the assembly tree.

2. Structured multifrontal LDL factorization. We first briefly review the structured multifrontal method in [39] and also a block LDL variation, which will be used in the factorization stage before the selected inversion.

2.1. HSS matrix and algorithms. The structured multifrontal method incorporates HSS structures into the multifrontal method. An $N \times N$ HSS matrix F with a corresponding HSS tree T can be defined as follows [10, 38, 42]. Let each node i of a full binary tree T be associated with a consecutive index set $t_i \subset \mathcal{I} \equiv \{1 : N\}$, which satisfies $t_i \cup t_{\mathrm{sib}(i)} = t_{\mathrm{par}(i)}$, $t_i \cap t_{\mathrm{sib}(i)} = \emptyset$, and $t_{\mathrm{root}(T)} = \mathcal{I}$. The index sets t_i associated with all the leaves i together form $\mathcal{I}$.

An HSS matrix F is given by

$$F \equiv D_{\mathrm{root}(T)},$$

where D_i is recursively defined for each nonleaf node i and its children c_1 and c_2 as

$$D_i = F|_{t_i\times t_i} = \begin{pmatrix} D_{c_1} & U_{c_1}B_{c_1}V_{c_2}^T \\ U_{c_2}B_{c_2}V_{c_1}^T & D_{c_2} \end{pmatrix} \tag{2.1}$$

with

$$U_i = \begin{pmatrix} U_{c_1} & \\ & U_{c_2} \end{pmatrix}\begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix}, \quad V_i = \begin{pmatrix} V_{c_1} & \\ & V_{c_2} \end{pmatrix}\begin{pmatrix} W_{c_1} \\ W_{c_2} \end{pmatrix}. \tag{2.2}$$

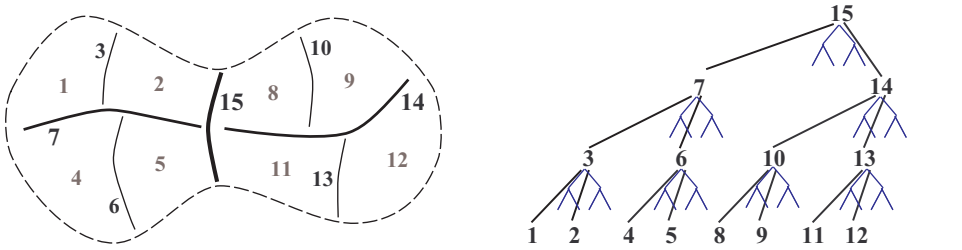
The size $|t_i|$ of D_i satisfies $|t_i| = |t_{c_1}|+|t_{c_2}|$. Notice that this structure is essentially the same as the matrix family $\mathcal{M}_{k,\tau}$ used in [20] based on a weak admissibility condition.

Here, D_i, U_i, R_i, etc., are called the (HSS) generators associated with node i. Due to (2.2), we also call U_i, V_i associated with a nonleaf node i *nested basis matrices*. The *HSS rank* of F is defined to be

$$r = \max_{i=1,2,\ldots,\mathrm{root}(T)-1} \left(\max(\mathrm{rank}(F|_{t_i\times(\mathcal{I}\setminus t_i)}),\ \mathrm{rank}(F|_{(\mathcal{I}\setminus t_i)\times t_i}))\right),$$

where $F|_{t_i\times(\mathcal{I}\setminus t_i)}$ and $F|_{(\mathcal{I}\setminus t_i)\times t_i}$ are called the ith HSS block row and column, respectively. An illustration can be found in Figure 2. If F is symmetric, then $U_i = V_i$, $R_i = W_i$, $B_{c_1} = B_{c_2}^T$.

To construct an HSS form, the HSS blocks are compressed hierarchically in a bottom-up traversal of T [6, 42]. F is partitioned following the index sets t_i. For each



(i) Mesh (ii) Outer assembly tree and inner HSS trees

FIG. 1. *Illustration of the nested dissection ordering* [14] *of a general mesh and a two-layer tree structure* [39, 41], *where the outer tree in* (ii) *is the assembly tree* $\mathcal{T}$*, and each inner tree is an HSS tree* T*. Each leaf node of* $\mathcal{T}$ *corresponds to the interior points of a subdomain and each nonleaf node corresponds to a separator or interface.*

leaf i, a QR or rank-revealing QR factorization is computed: $F|_{t_i \times (\mathcal{I} \setminus t_i)} = U_i H_i$. For a nonleaf node i, appropriate subblocks of H_{c_1} and H_{c_2} are merged and compressed to yield $\begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix}$. Similar operations are applied to the HSS block columns to obtain V_i and W_i. After such compression, B_i can be conveniently identified. The overall HSS construction from dense F costs $O(rN^2)$ flops. Later, this HSS construction can be applied to the frontal matrices in the multifrontal method. To avoid such dense HSS construction, randomized methods can be used [40].

If r is small, we can quickly perform various HSS operations. For example, ULV-type algorithms [10, 42] can be used to factorize F in $O(r^2 N)$ flops, and linear system solution with the factors costs only $O(rN)$ flops. Similarly, the multiplication and the explicit inversion of HSS matrices [10, 16] can be done in $O(r^2 N)$ flops. Even if the HSS blocks at different hierarchical levels have ranks growing with the block sizes, the factorization complexity may still be quite satisfactory, as long as the growth follows certain patterns [4, 21, 38]; see section 4.

Recently, HSS techniques were embedded into sparse matrix computations and helped the development of some fast direct solvers and efficient preconditioners [39, 40, 41]. The sparse factorization described in the next subsection is an example.

2.2. Structured multifrontal block LDL factorization. The multifrontal method [13] can be used to compute a block LDL factorization of A:

$$A = L \Lambda L^T. \tag{2.3}$$

To improve the efficiency, the fast structured sparse solver in [39] can be conveniently modified to compute an approximate multifrontal block LDL factorization. That is, structured approximations to L and Λ will be computed. This is briefly reviewed here as a background for the later structured selected inversion.

The matrix A is first reordered with nested dissection to reduce fill-in [14]. The variables or mesh points are grouped into *separators*, which are used to recursively divide the mesh or adjacency graph into smaller pieces. As in [39], we use some graph partitioning tools [15, 30] to partition the graph, so that our method is not restricted to any particular domain shape or mesh structure. For illustration, an example is shown in Figure 1(i).

The multifrontal method performs the sparse factorization via some local factorizations of a series of smaller dense *frontal matrices*. A postordered assembly tree $\mathcal{T}$ is formed to organize the elimination of the separators. Each node of $\mathcal{T}$ corresponds

to a separator in the mesh. (Later, we do not distinguish between a node of $\mathcal{T}$ and a separator in the mesh.) Label the nodes/separators of $\mathcal{T}$ as $\mathbf{i} = \mathbf{1}, \mathbf{2}, \ldots, \mathrm{root}(\mathcal{T})$. See the outer tree in Figure 1(ii). Let $\mathbf{s_i}$ be the index set of the mesh points in separator $\mathbf{i}$, and $\mathcal{N}_\mathbf{i}$ be the set of *neighbors* [39] of a node $\mathbf{i}$, defined as follows.

DEFINITION 2.1. *A neighbor* $\mathbf{j}$ *of node* $\mathbf{i}$ *is an ancestor node of* $\mathbf{i}$ *in the assembly tree* $\mathcal{T}$ *satisfying*

- *either* $A|_{\mathbf{s_j}\times\mathbf{s_i}}$ *has nonzero entries,*
- *or a nonzero entry or fill-in is introduced into* $A|_{\mathbf{s_j}\times\mathbf{s_i}}$ *due to the elimination of a descendant of* $\mathbf{i}$ *in the factorization, i.e.,* $A|_{\mathbf{s_j}\times\mathbf{s_i}} = 0$ *but* $L|_{\mathbf{s_j}\times\mathbf{s_i}}$ *has nonzero entries.*

For example, in Figure 1, $\mathcal{N}_\mathbf{2} = \{\mathbf{3}, \mathbf{7}, \mathbf{15}\}$, $\mathcal{N}_\mathbf{3} = \{\mathbf{7}, \mathbf{15}\}$. Note that $\mathcal{N}_\mathbf{3}$ includes separator $\mathbf{15}$ because the elimination of separator $\mathbf{2}$ connects separators $\mathbf{3}$ and $\mathbf{15}$.

Before reviewing the structured multifrontal method, we emphasize the following:

- The *global* sparse factorization is governed by the multifrontal framework, which follows the assembly tree $\mathcal{T}$.
- HSS techniques are applied *locally* to the intermediate frontal matrices only. That is, each (local) HSS tree T corresponds to an individual node/separator in the (global) assembly tree $\mathcal{T}$; see Figure 1(ii).
- The two types of trees $\mathcal{T}$ and T are independent of each other. In T for an HSS matrix, a parent node corresponds to an index set that includes the child index sets. In $\mathcal{T}$, a parent node and its children correspond to *different* mesh point sets. That is, each node corresponds to an *independent* separator in the mesh, regardless of the parent/child relationship.

The structured multifrontal scheme proceeds as follows. For each node $\mathbf{i}$ of $\mathcal{T}$, let

$$\mathbf{F_i^0} = \begin{pmatrix} A|_{\mathbf{s_i}\times\mathbf{s_i}} & (A|_{\mathbf{s}_{\mathcal{N}_\mathbf{i}}\times\mathbf{s_i}})^T \\ A|_{\mathbf{s}_{\mathcal{N}_\mathbf{i}}\times\mathbf{s_i}} & 0 \end{pmatrix}, \tag{2.4}$$

where $\mathbf{s}_{\mathcal{N}_\mathbf{i}}$ can be understood similarly to $\mathbf{s_i}$. If $\mathbf{i}$ is a leaf of $\mathcal{T}$, the associated frontal matrix is simply $\mathbf{F_i} \equiv \mathbf{F_i^0}$. If $\mathbf{i}$ is a nonleaf node with children $\mathbf{c}_1$ and $\mathbf{c}_2$, form a frontal matrix $\mathbf{F_i}$ by an assembly operation called *extend-add* [13]:

$$\mathbf{F_i} = \mathbf{F_i^0} \boxplus \mathbf{S}_{\mathbf{c}_1} \boxplus \mathbf{S}_{\mathbf{c}_2}, \tag{2.5}$$

where $\mathbf{S}_{\mathbf{c}_1}$ and $\mathbf{S}_{\mathbf{c}_2}$ are called *update matrices* and are obtained from early steps similar to (2.8) below, and the extend-add operator $\boxplus$ means that the matrices are permuted and extended to match the global indices in $\{\mathbf{s_i}, \mathbf{s}_{\mathcal{N}_\mathbf{i}}\}$.

The structured multifrontal LDL factorization follows the framework in [39] (partially structured) or [40, 41] (fully structured). Partition $\mathbf{F_i}$ as

$$\mathbf{F_i} \equiv \begin{pmatrix} F_{\mathbf{i},\mathbf{i}} & F_{\mathcal{N}_\mathbf{i},\mathbf{i}}^T \\ F_{\mathcal{N}_\mathbf{i},\mathbf{i}} & F_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} \end{pmatrix}, \tag{2.6}$$

so that $F_{\mathbf{i},\mathbf{i}}$ corresponds to the index set $\mathbf{s_i}$. Construct an HSS approximation to $\mathbf{F_i}$ with generators D_i, U_i, etc., so that $F_{\mathbf{i},\mathbf{i}}$ and $F_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}}$ correspond to two sibling nodes k and $\bar{k}$ of the HSS tree T, respectively; see Figure 2.

Next, compute a block LDL factorization

$$\mathbf{F_i} = \begin{pmatrix} I & \\ L_{\mathcal{N}_\mathbf{i},\mathbf{i}} & I \end{pmatrix} \begin{pmatrix} F_{\mathbf{i},\mathbf{i}} & \\ & \mathbf{S_i} \end{pmatrix} \begin{pmatrix} I & (L_{\mathcal{N}_\mathbf{i},\mathbf{i}})^T \\ & I \end{pmatrix},$$

where

$$L_{\mathcal{N}_\mathbf{i},\mathbf{i}} = F_{\mathcal{N}_\mathbf{i},\mathbf{i}} F_{\mathbf{i},\mathbf{i}}^{-1}, \tag{2.7}$$

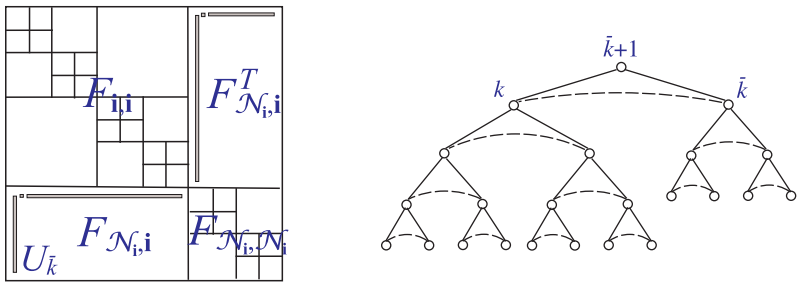


FIG. 2. *Illustration of an HSS form and the corresponding HSS tree used for a frontal matrix* $\mathbf{F_i}$.

and $\mathbf{S_i}$ is the update matrix of the form

$$\mathbf{S_i} = F_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} - F_{\mathcal{N}_\mathbf{i},\mathbf{i}} F_{\mathbf{i},\mathbf{i}}^{-1} F_{\mathcal{N}_\mathbf{i},\mathbf{i}}^T. \tag{2.8}$$

After the HSS approximation of $\mathbf{F_i}$, $F_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ (as one of its off-diagonal blocks) looks like

$$F_{\mathcal{N}_\mathbf{i},\mathbf{i}} \approx U_{\bar{k}} B_k^T U_k^T. \tag{2.9}$$

We then have

$$\mathbf{S_i} \approx F_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} - U_{\bar{k}} B_k^T (U_k^T F_{\mathbf{i},\mathbf{i}}^{-1} U_k) B_k U_{\bar{k}}^T. \tag{2.10}$$

Remark 2.1. HSS approximation errors are extensively studied in [37]. For example, if all the HSS blocks are compressed with a relative accuracy τ, then the relative HSS approximation error in the Frobenius norm is $O(\sqrt{r}\log N)\tau$. In other words, the error accumulation factor is $O(\sqrt{r}\log N)$. (This may possibly even be reduced to $O(\log N)$ [5, Corollary 6.18].) If all the frontal matrices are approximated, we expect a similar effect in the error accumulation, due to the similarity in the hierarchical structures of the assembly tree and the HSS tree. That is, the overall accumulation factor is likely a low-order power of $O(\sqrt{r}\log n)$. A more precise estimation will be conducted in the future for all the factorization steps. Numerical tests indicate that the overall approximation error is usually well controlled.

Since we are interested in the selected inversion, unlike the method in [39], an HSS inverse $F_{\mathbf{i},\mathbf{i}}^{-1}$ is computed here (see section 3.1). Theorem 3.4 below indicates that the information in the inversion of $F_{\mathbf{i},\mathbf{i}}$ can be used to quickly form (2.10).

In practice, a switching level $\mathbf{l}_s$ [39, 41] is also involved, so that standard dense factorizations are used when a node of the assembly tree $\mathcal{T}$ is below level $\mathbf{l}_s$; this is to optimize the complexity (see section 4).

Remark 2.2. The structured multifrontal LDL factorization we implement involves a shortcut in [39] which does not affect the performance of the later inversion. The shortcut is to use dense update matrices, so that dense frontal matrices are formed first and then approximated by HSS forms. This has $O(rN^2)$ complexity for local HSS compression, and makes the overall multifrontal factorization cost suboptimal, but is much simpler to use. The reasons for this shortcut are clarified in [39].

- This multifrontal algorithm is designed to take quite general sparse matrices as inputs, without specific information on the PDE or geometry. If the update matrices are in HSS forms, they may potentially need to be permuted arbitrarily in the assembly operation (2.5). Designing such a general scheme would be unnecessarily sophisticated. However, when the update matrices are kept dense, (2.5) is straightforward.

- As shown in [39], for 2D and 3D problems satisfying certain rank patterns, the difference in the complexity between this factorization (with the shortcut) and a fully structured factorization (with all HSS local operations) is insignificant. For both versions, the factorization complexity is up to $O(n\log^2 n)$ in two dimensions and up to $O(n^{4/3}\log n)$ in three dimensions.
- Most importantly, *the resulting factors are still structured*, just like in a fully structured multifrontal method. (Therefore, this shortcut does not affect the complexity of the later selected inversion, which is the focus of this work.)

3. Structured sparse selected inversion. We then describe our structured selected inversion scheme for A based on its structured LDL factors. After the structured multifrontal factorization, suppose A is approximated by $\tilde{A}$. That is, the structured factors are the exact factors of $\tilde{A}$. Since no approximation is involved in the inversion stage, we use notation such as $\mathbf{F_i}$ to denote its HSS approximation, and L to denote the structured factor. (This highly simplifies the notation that is otherwise very messy due to the multilevel approximations in the factorization stage.) Let

$$C = \tilde{A}^{-1}. \tag{3.1}$$

Thus, our selected inversion computes $\mathrm{diag}(C)$ which approximates $\mathrm{diag}(A^{-1})$. We start with the discussion of an HSS inversion algorithm, as needed in the scheme. The HSS inversion also benefits the computation of the update matrices $\mathbf{S_i}$.

3.1. HSS inversion and fast computation of the update matrices. An HSS inversion algorithm is proposed in [16], and the idea is to recursively apply the SMW formula after writing an HSS matrix as a block diagonal matrix plus a low-rank update (see (2.1)). A slightly modified form of the SMW formula is used in [16] and can be derived (based on the standard version) in a clearer way as follows. Assume all the matrices in the following derivation are of appropriate sizes, and all the relevant inverses exist, then

(3.2)
$$\begin{aligned}
(D+UBV^T)^{-1} &= D^{-1} - D^{-1}U(B^{-1}+V^TD^{-1}U)^{-1}V^TD^{-1}\\
&= D^{-1} - D^{-1}U(B^{-1}+\hat{D}^{-1})^{-1}V^TD^{-1} \qquad (\hat{D}=(V^TD^{-1}U)^{-1})\\
&= D^{-1} - D^{-1}UB(\hat{D}+B)^{-1}\hat{D}V^TD^{-1}\\
&= D^{-1} - D^{-1}U[((\hat{D}+B)-\hat{D})(\hat{D}+B)^{-1}]\hat{D}V^TD^{-1}\\
&= (D^{-1} - D^{-1}U\hat{D}V^TD^{-1}) + D^{-1}U\hat{D}(\hat{D}+B)^{-1}\hat{D}V^TD^{-1}.
\end{aligned}$$

With this formula, a simplification of the HSS inversion algorithm in [16] for a symmetric HSS matrix F is outlined below. The inputs of this algorithm are the HSS generators U_i, R_i, B_i, D_i of F, and the outputs are the HSS generators $\tilde{U}_i, \tilde{R}_i, \tilde{B}_i, \tilde{D}_i$ of F^{-1}. In the following two subsections, $\bar{D}_i, \bar{U}_i, \bar{G}_i, \hat{D}_i, \hat{G}_i$ are intermediate results in the inversion, and G_i (for a nonleaf node i) is used only for the derivation and is not actually computed.

3.1.1. Basic HSS inversion in terms of two levels. To motivate the HSS inversion procedure, first consider a two-level symmetric HSS form

$$D_3 = \begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix} + \begin{pmatrix} U_1 & \\ & U_2 \end{pmatrix} \begin{pmatrix} & B_1 \\ B_1^T & \end{pmatrix} \begin{pmatrix} U_1^T & \\ & U_2^T \end{pmatrix}. \tag{3.3}$$

According to (3.2),

$$D_3^{-1} = \operatorname{diag}(G_1, G_2) + \operatorname{diag}(\tilde{U}_1, \tilde{U}_2)\bar{D}_3^{-1} \operatorname{diag}(\tilde{U}_1^T, \tilde{U}_2^T), \tag{3.4}$$

where

$$G_j = D_j^{-1} - D_j^{-1}U_j\hat{D}_j U_j^T D_j^{-1}, \quad \tilde{U}_j = D_j^{-1}U_j\hat{D}_j, \quad j = 1, 2, \quad \text{with} \tag{3.5}$$

$$\hat{D}_j = (U_j^T D_j^{-1} U_j)^{-1}, \quad j = 1, 2, \tag{3.6}$$

$$\bar{D}_i = \begin{pmatrix} \hat{D}_{c_1} & B_{c_1} \\ B_{c_1}^T & \hat{D}_{c_2} \end{pmatrix}, \quad i = 3 \ (c_1 = 1,\ c_2 = 2). \tag{3.7}$$

3.1.2. General HSS inversion. To generalize to more levels, we consider a postordered HSS tree consisting of three levels and 7 nodes, with node 7 as the root, nodes $3, 6$ (children of 7) at the next level, and nodes $1, 2$ (children of 3) and $4, 5$ (children of 6) at the leaf level. A corresponding symmetric HSS matrix looks like

$$D_7 = \begin{pmatrix} D_3 & \\ & D_6 \end{pmatrix} + \begin{pmatrix} U_3 & \\ & U_6 \end{pmatrix} \begin{pmatrix} & B_3 \\ B_3^T & \end{pmatrix} \begin{pmatrix} U_3^T & \\ & U_6^T \end{pmatrix},$$

where D_3, D_6, U_3, U_6 are in nested forms just like (2.1)–(2.2) (with $i = 3$ or 6). Since D_3 and D_6 are two-level HSS forms, the two-level HSS inversion above yields D_3^{-1} in (3.4) and also D_6^{-1}. The $\tilde{U}$ generators associated with the leaves are obtained.

On the other hand, by treating D_7 itself as a two-level HSS form, we can get

$$D_7^{-1} = \operatorname{diag}(G_3, G_6) + \operatorname{diag}(\tilde{U}_3, \tilde{U}_6)\bar{D}_7^{-1} \operatorname{diag}(\tilde{U}_3^T, \tilde{U}_6^T), \tag{3.8}$$

where $G_3, G_6, \tilde{U}_3, \tilde{U}_6$ are defined just like in (3.5) with $j = 3$ or 6, and $\bar{D}_7$ is defined just like in (3.7) with $i = 7$. Let

$$\hat{G}_7 = \bar{D}_7^{-1} \equiv \begin{pmatrix} \hat{G}_{7;1,1} & \hat{G}_{7;1,2} \\ \hat{G}_{7;2,1} & \hat{G}_{7;2,2} \end{pmatrix},$$

where $\hat{G}_7$ is partitioned as in (3.8). Then,

$$D_7^{-1} = \begin{pmatrix} G_3 + \tilde{U}_3\hat{G}_{7;1,1}\tilde{U}_3^T & \tilde{U}_3\hat{G}_{7;1,2}\tilde{U}_6^T \\ \tilde{U}_6\hat{G}_{7;2,1}\tilde{U}_3^T & G_6 + \tilde{U}_6\hat{G}_{7;2,2}\tilde{U}_6^T \end{pmatrix}.$$

Thus, we can set

$$\tilde{B}_3 = \hat{G}_{7;1,2}, \quad \tilde{D}_3 = G_3 + \tilde{U}_3\hat{G}_{7;1,1}\tilde{U}_3^T, \quad \tilde{U}_3 = D_3^{-1}U_3\hat{D}_3, \tag{3.9}$$

where $\hat{D}_3$ is defined just like in (3.6) with $j = 3$. (Here, we focus on node 3 and its descendants. The study of node 6 and its descendants is similar.)

We then need to resolve the following issues:

1. $\bar{D}_7$ and $\tilde{U}_3$ involve $\hat{D}_3$, and the computation of $\hat{D}_3 = (U_3^T D_3^{-1} U_3)^{-1}$ needs D_3^{-1} and a nested basis matrix U_3, which are not explicitly available;
2. $\tilde{U}_3$ should appear as a nested basis matrix, so we need to find $\tilde{R}_1$ and $\tilde{R}_2$;
3. $\tilde{D}_3$ itself is a two-level HSS form, so we need to find $\tilde{D}_1, \tilde{D}_2, \tilde{B}_1$.

For these purposes, we introduce the following simple lemma.

LEMMA 3.1. *The matrices in* (3.5)–(3.7) *satisfy*

$$U_j^T\tilde{U}_j = I, \quad U_j^T G_j U_j = 0, \quad j = 1, 2,$$
$$\operatorname{diag}(U_{c_1}^T, U_{c_2}^T)D_i^{-1} \operatorname{diag}(U_{c_1}, U_{c_2}) = \bar{D}_i^{-1}, \quad i = 3.$$

Proof. From (3.5) and (3.6), we have

$$\begin{aligned} U_j^T\tilde{U}_j &= U_j^T(D_j^{-1}U_j\hat{D}_j) = \hat{D}_j^{-1}\hat{D}_j = I,\\ U_j^TG_jU_j &= U_j^T(D_j^{-1} - D_j^{-1}U_j\hat{D}_jU_j^TD_j^{-1})U_j\\ &= U_j^TD_j^{-1}U_j - U_j^TD_j^{-1}U_j\hat{D}_jU_j^TD_j^{-1}U_j = \hat{D}_j^{-1} - \hat{D}_j^{-1}\hat{D}_j\hat{D}_j^{-1} = 0. \end{aligned}$$

From these two results and (3.4), for $i = 3$ we have

$$\begin{aligned} &\operatorname{diag}(U_{c_1}^T, U_{c_2}^T)D_i^{-1}\operatorname{diag}(U_{c_1}, U_{c_2})\\ &\quad= \operatorname{diag}(U_{c_1}^T, U_{c_2}^T)[\operatorname{diag}(G_{c_1}, G_{c_2}) + \operatorname{diag}(\tilde{U}_{c_1}, \tilde{U}_{c_2})\bar{D}_i^{-1}\operatorname{diag}(\tilde{U}_{c_1}^T, \tilde{U}_{c_2}^T)]\operatorname{diag}(U_{c_1}, U_{c_2})\\ &\quad= \operatorname{diag}(U_{c_1}^TG_{c_1}\tilde{U}_{c_1}, U_{c_2}^TG_{c_2}U_{c_2}) + \operatorname{diag}(U_{c_1}^T\tilde{U}_{c_1}, U_{c_2}^T\tilde{U}_{c_2})\bar{D}_i^{-1}\operatorname{diag}(\tilde{U}_{c_1}^TU_{c_1}, \tilde{U}_{c_2}^TU_{c_2})\\ &\quad= \operatorname{diag}(0,0) + \operatorname{diag}(I,I)\bar{D}_i^{-1}\operatorname{diag}(I,I) = \bar{D}_i^{-1}. \qquad \square \end{aligned}$$

Then we resolve the three issues above Lemma 3.1. First, we derive a convenient way to find $\hat{D}_3 = (U_3^TD_3^{-1}U_3)^{-1}$. This is based on the following lemma.

LEMMA 3.2. *$\bar{D}_i$ in* (3.7) *satisfies*

$$U_i^TD_i^{-1}U_i = \bar{U}_i^T\bar{D}_i^{-1}\bar{U}_i \quad \textit{with} \quad \bar{U}_i = \begin{pmatrix} R_{c_1}\\ R_{c_2}\end{pmatrix}, \quad i = 3. \tag{3.10}$$

Thus, $\hat{D}_i = (\bar{U}_i^T\bar{D}_i^{-1}\bar{U}_i)^{-1}$ (which does not involve the nested forms D_i, U_i).

Proof. According to the nested form of U_i and Lemma 3.1,

$$\begin{aligned} U_i^TD_i^{-1}U_i &= \begin{pmatrix} R_{c_1}^T & R_{c_2}^T\end{pmatrix}\begin{pmatrix} U_1^T & \\ & U_2^T\end{pmatrix}D_i^{-1}\begin{pmatrix} U_1 & \\ & U_2\end{pmatrix}\begin{pmatrix} R_{c_1}\\ R_{c_2}\end{pmatrix}\\ &= \bar{U}_i^T\bar{D}_i^{-1}\bar{U}_i. \qquad \square \end{aligned}$$

Second, we find $\tilde{R}_1$ and $\tilde{R}_2$ in the nested form of $\tilde{U}_3$:

$$\begin{pmatrix} \tilde{U}_1 & \\ & \tilde{U}_2\end{pmatrix}\begin{pmatrix} \tilde{R}_1\\ \tilde{R}_2\end{pmatrix} = D_3^{-1}U_3\hat{D}_3 \quad (\text{from (3.9)}).$$

By Lemma 3.1, we can multiply $\operatorname{diag}(U_1^T, U_2^T)$ on the left of both sides to get

$$\begin{aligned} \begin{pmatrix} \tilde{R}_1\\ \tilde{R}_2\end{pmatrix} &= \begin{pmatrix} U_1^T & \\ & U_2^T\end{pmatrix}D_3^{-1}U_3\hat{D}_3 = \begin{pmatrix} U_1^T & \\ & U_2^T\end{pmatrix}D_3^{-1}\begin{pmatrix} U_1 & \\ & U_2\end{pmatrix}\begin{pmatrix} R_1\\ R_2\end{pmatrix}\hat{D}_3\\ &= \bar{D}_3^{-1}\bar{U}_3\hat{D}_3 \qquad (\text{from Lemma 3.1 and } \bar{U}_3 \text{ in (3.10)}). \end{aligned} \tag{3.11}$$

This gives a formula for computing $\tilde{R}_1$ and $\tilde{R}_2$.

Third, we find $\tilde{D}_1, \tilde{D}_2, \tilde{B}_1$. From (3.4), (3.9), and the nested form of $\tilde{U}_3$, we have

$$\begin{aligned} \tilde{D}_3 &= G_3 + \tilde{U}_3\hat{G}_{7;1,1}\tilde{U}_3^T = (D_3^{-1} - D_3^{-1}U_3\hat{D}_3U_3^TD_3^{-1}) + \tilde{U}_3\hat{G}_{7;1,1}\tilde{U}_3^T\\ &= D_3^{-1} - \tilde{U}_3\hat{D}_3^{-1}\tilde{U}_3^T + \tilde{U}_3\hat{G}_{7;1,1}\tilde{U}_3^T \qquad (\text{from (3.9)})\\ &= (\operatorname{diag}(G_1, G_2) + \operatorname{diag}(\tilde{U}_1, \tilde{U}_2)\bar{D}_3^{-1}\operatorname{diag}(\tilde{U}_1^T, \tilde{U}_2^T)) - \tilde{U}_3\hat{D}_3^{-1}\tilde{U}_3^T + \tilde{U}_3\hat{G}_{7;1,1}\tilde{U}_3^T\\ &= \operatorname{diag}(G_1, G_2) + \operatorname{diag}(\tilde{U}_1, \tilde{U}_2)\left[\bar{D}_3^{-1} - \begin{pmatrix} \tilde{R}_1\\ \tilde{R}_2\end{pmatrix}\hat{D}_3^{-1}\begin{pmatrix} \tilde{R}_1^T & \tilde{R}_2^T\end{pmatrix}\right.\\ &\qquad\qquad \left. + \begin{pmatrix} \tilde{R}_1\\ \tilde{R}_2\end{pmatrix}\hat{G}_{7;1,1}\begin{pmatrix} \tilde{R}_1^T & \tilde{R}_2^T\end{pmatrix}\right]\operatorname{diag}(\tilde{U}_1^T, \tilde{U}_2^T). \end{aligned}$$

Define

$$\bar{G}_3 = \bar{D}_3^{-1} - \begin{pmatrix} \tilde{R}_1 \\ \tilde{R}_2 \end{pmatrix} \hat{D}_3^{-1} \begin{pmatrix} \tilde{R}_1^T & \tilde{R}_2^T \end{pmatrix} \tag{3.12}$$

$$= \bar{D}_3^{-1} - \bar{D}_3^{-1}\bar{U}_3\hat{D}_3\hat{D}_3^{-1}\hat{D}_3\bar{U}_3^T\bar{D}_3^{-1} \quad \text{(from (3.11))}$$

$$= \bar{D}_3^{-1} - \bar{D}_3^{-1}\bar{U}_3\hat{D}_3\bar{U}_3^T\bar{D}_3^{-1},$$

$$\hat{G}_3 = \bar{G}_3 + \begin{pmatrix} \tilde{R}_1 \\ \tilde{R}_2 \end{pmatrix} \hat{G}_{7;1,1} \begin{pmatrix} \tilde{R}_1^T & \tilde{R}_2^T \end{pmatrix} \equiv \begin{pmatrix} \hat{G}_{3;1,1} & \hat{G}_{3;1,2} \\ \hat{G}_{3;2,1} & \hat{G}_{3;2,2} \end{pmatrix}, \tag{3.13}$$

where $\hat{G}_3$ is partitioned conformably. Then

$$\tilde{D}_3 = \operatorname{diag}(G_1, G_2) + \operatorname{diag}(\tilde{U}_1, \tilde{U}_2)\hat{G}_3 \operatorname{diag}(\tilde{U}_1^T, \tilde{U}_2^T)$$

$$= \begin{pmatrix} G_1 + \tilde{U}_1\hat{G}_{3;1,1}\tilde{U}_1^T & \tilde{U}_1\hat{G}_{3;1,2}\tilde{U}_2^T \\ \tilde{U}_2\hat{G}_{3;2,1}\tilde{U}_1^T & G_2 + \tilde{U}_2\hat{G}_{3;2,2}\tilde{U}_2^T \end{pmatrix}.$$

Thus, we obtain the following generators:

$$\tilde{D}_1 = G_1 + \tilde{U}_1\hat{G}_{3;1,1}\tilde{U}_1^T, \quad \tilde{D}_2 = G_2 + \tilde{U}_2\hat{G}_{3;2,2}\tilde{U}_2^T, \quad \tilde{B}_1 = \hat{G}_{3;1,2}. \tag{3.14}$$

In general HSS inversion with more levels, the generators of F^{-1} can be similarly found. (The derivation is similar to the process above. Lemma 3.3 below shows some essential ideas by induction. Another way is based on a telescoping HSS representation in [16]. We do not repeat the details since they do not affect the understanding of our later discussions on selected inversion.) The procedure can be organized into two traversals of the HSS tree T. In a bottom-up traversal, define hierarchically

$$\bar{D}_i = D_i, \qquad \bar{U}_i = U_i, \qquad \hat{D}_i = (\bar{U}_i^T\bar{D}_i^{-1}\bar{U}_i)^{-1} \quad (i\text{: leaf})$$

$$\bar{D}_i = \begin{pmatrix} \hat{D}_{c_1} & B_{c_1} \\ B_{c_1}^T & \hat{D}_{c_2} \end{pmatrix}, \quad \bar{U}_i = \begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix}, \quad \hat{D}_i = (\bar{U}_i^T\bar{D}_i^{-1}\bar{U}_i)^{-1} \quad (i\text{: nonleaf}). \tag{3.15}$$

(The derivations above and Lemma 3.3 below indicate that, $\bar{D}_i, \bar{U}_i$ can be understood as generators of an intermediate reduced HSS matrix [16, 39].) Then compute

$$\tilde{U}_i = \bar{D}_i^{-1}\bar{U}_i\hat{D}_i \quad (i\text{: leaf}) \quad \text{or} \quad \begin{pmatrix} \tilde{R}_{c_1} \\ \tilde{R}_{c_2} \end{pmatrix} = \bar{D}_i^{-1}\bar{U}_i\hat{D}_i \quad (i\text{: nonleaf}); \tag{3.16}$$

see, e.g., (3.5) and (3.11). Proceed with these steps for the nonleaf nodes i. When the node $i = \operatorname{root}(T)$ is reached, compute only $\bar{D}_i$ as in (3.15).

In a top-down traversal of T, we find the $\tilde{D}, \tilde{B}$ generators of F^{-1}. For $i = \operatorname{root}(T)$, let $\hat{G}_i \equiv \bar{D}_i^{-1}$ and partition $\hat{G}_i$ conformably following (3.15) as

$$\hat{G}_i = \begin{pmatrix} \hat{G}_{i;1,1} & \hat{G}_{i;1,2} \\ \hat{G}_{i;2,1} & \hat{G}_{i;2,2} \end{pmatrix}. \tag{3.17}$$

Then let

$$\tilde{B}_{c_1} = \hat{G}_{i;1,2}. \tag{3.18}$$

For a nonleaf node $i < \operatorname{root}(T)$, let

$$\bar{G}_i = \bar{D}_i^{-1} - \bar{D}_i^{-1}\bar{U}_i\hat{D}_i\bar{U}_i^T\bar{D}_i^{-1}, \tag{3.19}$$

$$\hat{G}_i = \bar{G}_i + \begin{pmatrix} \tilde{R}_{c_1} \\ \tilde{R}_{c_2} \end{pmatrix} \hat{G}_{\operatorname{par}(i);j,j} \begin{pmatrix} \tilde{R}_{c_1}^T & \tilde{R}_{c_2}^T \end{pmatrix},$$

where $j = 1$ or 2, depending on whether i is a left or a right child of $\mathrm{par}(i)$. ($\bar{G}_i$ can be understood as a diagonal block of the inverse of a reduced HSS matrix [16]; see, e.g., (3.5) and (3.12). $\hat{G}_i$ is the sum of $\bar{G}_i$ and a low-rank contribution from the parent level; see, e.g., (3.13).) Then partition $\hat{G}_i$ as in (3.17), and set $\tilde{B}_{c_1}$ as in (3.18).

Repeat these until all the nonleaf nodes are visited. Then for each leaf i, define $\bar{G}_i$ as in (3.19) and let

$$\tilde{D}_i = \bar{G}_i + \tilde{U}_i \hat{G}_{\mathrm{par}(i);j,j} \tilde{U}_i^T, \tag{3.20}$$

where $j = 1$ or 2, depending on whether i is a left or a right child of $\mathrm{par}(i)$; see, e.g., (3.14). Then we have the HSS generators $\tilde{D}_i, \tilde{U}_i, \tilde{R}_i, \tilde{B}_i$ of F^{-1}.

3.1.3. Fast computation of the update matrices. The HSS inversion is applied to $F_{\mathbf{i},\mathbf{i}}$ in (2.6) when we compute the diagonal blocks of C. In addition, the information computed in the inversion of $F_{\mathbf{i},\mathbf{i}}$ can also help to speed up the computation of the update matrix $\mathbf{S_i}$ in (2.10) in the structured multifrontal method, as well as some additional computations in the selected inversion.

For this purpose, we first show that the results in Lemmas 3.1 and 3.2 can be generalized.

LEMMA 3.3. *Let $G_i = D_i^{-1} - D_i^{-1} U_i \hat{D}_i U_i^T D_i^{-1}$ for a node $i \neq \mathrm{root}(T)$, then*

$$\begin{gathered} U_i^T D_i^{-1} U_i = \bar{U}_i^T \bar{D}_i^{-1} \bar{U}_i = \hat{D}_i^{-1}, \\ U_i^T \tilde{U}_i = I, \quad U_i^T G_i U_i = 0, \\ \mathrm{diag}(U_{c_1}^T, U_{c_2}^T) D_i^{-1} \,\mathrm{diag}(U_{c_1}, U_{c_2}) = \bar{D}_i^{-1} \quad \textit{(i: nonleaf)}. \end{gathered} \tag{3.21}$$

Proof. We only need to prove (3.21). The reason is, once (3.21) holds, the other formulas can be proved in the same way as in the proof of Lemma 3.1.

We show $U_i^T D_i^{-1} U_i = \bar{U}_i^T \bar{D}_i^{-1} \bar{U}_i = \hat{D}_i^{-1}$ by induction. Let T_i denote the subtree of T associated with node i and its descendants. The induction is done on the number of levels l of T_i. The result holds for T_i with two levels, as in Lemma 3.1. Assume the result holds for any subtree of T with up to $l-1$ levels. We show it also holds for T_i with l levels. Writing D_i as a block diagonal plus a low-rank form (see, e.g., (3.3)) and applying the modified SMW formula (3.2) yield

$$\begin{aligned} D_i^{-1} = {} & \mathrm{diag}(D_{c_1}^{-1}, D_{c_2}^{-1}) - \mathrm{diag}(D_{c_1}^{-1} U_{c_1}, D_{c_2}^{-1} U_{c_2}) \\ & \cdot \left[H^{-1} - H^{-1} \left(\begin{pmatrix} & B_{c_1} \\ B_{c_1}^T & \end{pmatrix} + H^{-1} \right)^{-1} H^{-1} \right] \mathrm{diag}(U_{c_1}^T D_{c_1}^{-1}, U_{c_2}^T D_{c_2}^{-1}), \end{aligned} \tag{3.22}$$

where

$$H = \mathrm{diag}\left(U_{c_1}^T D_{c_1}^{-1} U_{c_1},\ U_{c_2}^T D_{c_2}^{-1} U_{c_2} \right). \tag{3.23}$$

According to the nested form of U_i in (2.2), we have

$$\begin{aligned} U_i^T D_i^{-1} U_i = {} & \begin{pmatrix} R_{c_1}^T & R_{c_2}^T \end{pmatrix} H \begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix} - \begin{pmatrix} R_{c_1}^T & R_{c_2}^T \end{pmatrix} H H^{-1} H \begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix} \\ & + \begin{pmatrix} R_{c_1}^T & R_{c_2}^T \end{pmatrix} H H^{-1} \left(\begin{pmatrix} & B_{c_1} \\ B_{c_1}^T & \end{pmatrix} + H^{-1} \right)^{-1} H^{-1} H \begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix} \\ = {} & \begin{pmatrix} R_{c_1}^T & R_{c_2}^T \end{pmatrix} \begin{pmatrix} (U_{c_1}^T D_{c_1}^{-1} U_{c_1})^{-1} & B_{c_1} \\ B_{c_1}^T & (U_{c_2}^T D_{c_2}^{-1} U_{c_2})^{-1} \end{pmatrix}^{-1} \begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix}. \end{aligned}$$

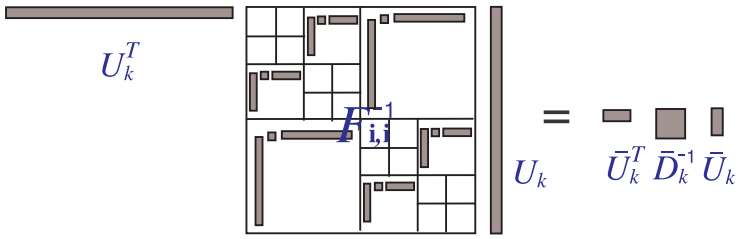


FIG. 3. *A pictorial illustration of the formula in Theorem* 3.4.

Since c_1 and c_2 are at level $l-1$, by induction,

$$U_{c_1}^T D_{c_1}^{-1} U_{c_1} = \bar{U}_{c_1}^T \bar{D}_{c_1}^{-1} \bar{U}_{c_1} = \hat{D}_{c_1}^{-1}, \quad U_{c_2}^T D_{c_2}^{-1} U_{c_2} = \bar{U}_{c_2}^T \bar{D}_{c_2}^{-1} \bar{U}_{c_2} = \hat{D}_{c_2}^{-1}. \tag{3.24}$$

Thus, $U_i^T D_i^{-1} U_i = \begin{pmatrix} R_{c_1}^T & R_{c_2}^T \end{pmatrix} \begin{pmatrix} \hat{D}_{c_1} & B_{c_1} \\ B_{c_1}^T & \hat{D}_{c_2} \end{pmatrix}^{-1} \begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix} = \bar{U}_i^T \bar{D}_i^{-1} \bar{U}_i.$ ∎

Equation (3.21) illustrates an idea of reduced matrices just like that in [39]. Thus, to compute the update matrix $\mathbf{S_i}$ in (2.10), we can avoid directly using the HSS form of $F_{\mathbf{i},\mathbf{i}}^{-1} \equiv D_k^{-1}$ in $U_k^T D_k^{-1} U_k$. The following result is a direct corollary of Lemma 3.3.

THEOREM 3.4. *Suppose D_i, U_i, etc., are the HSS generators of $F_{\mathbf{i},\mathbf{i}}$ in* (2.6). *Then*

$$U_k^T F_{\mathbf{i},\mathbf{i}}^{-1} U_k = \bar{U}_k^T \bar{D}_k^{-1} \bar{U}_k,$$

where the HSS tree of $F_{\mathbf{i},\mathbf{i}}$ has k nodes and $\bar{D}_k$ is given in (3.15) *with $i = k$ in the HSS inversion of $F_{\mathbf{i},\mathbf{i}}$. Therefore, $\mathbf{S_i}$ in* (2.10) *can be quickly computed as*

$$F_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} - U_{\bar{k}} B_k^T (\bar{U}_k^T \bar{D}_k^{-1} \bar{U}_k) B_k U_{\bar{k}}^T.$$

See Figure 3 for an illustration. This theorem indicates that $\bar{D}_k$ plays a role similar to the final reduced matrix defined in [39] (where $F_{\mathbf{i},\mathbf{i}}$ is factorized by ULV-type algorithms [10]), yet here we take advantage of HSS inversion. If $F_{\mathbf{i},\mathbf{i}}$ has size N and HSS rank r, this theorem can help to reduce the complexity of computing $U_k^T F_{\mathbf{i},\mathbf{i}}^{-1} U_k$ from $O(r^2 N)$ with HSS inversion to only $O(r^3)$ with a simple formula $\bar{U}_k^T \bar{D}_k^{-1} \bar{U}_k$.

3.2. Basic ideas of the structured selected inversion. Similarly to the existing direct selected inversion methods in [23, 25, 26, 27], the extraction of diag(C) includes two stages, a forward one (block LDL factorization) and a backward one (inversion). The basic ideas of our structured selected inversion can be illustrated with a simple example.

Consider a sparse symmetric matrix A after applying nested dissection, as well as its block LDL factorization:

$$A = \begin{pmatrix} A_{\mathbf{11}} & & A_{\mathbf{13}} \\ & A_{\mathbf{22}} & A_{\mathbf{23}} \\ A_{\mathbf{31}} & A_{\mathbf{32}} & A_{\mathbf{33}} \end{pmatrix} = \begin{pmatrix} I & & \\ & I & \\ L_{\mathbf{31}} & L_{\mathbf{32}} & I \end{pmatrix} \begin{pmatrix} A_{\mathbf{11}} & & \\ & A_{\mathbf{22}} & \\ & & \mathbf{F_3} \end{pmatrix} \begin{pmatrix} I & & L_{\mathbf{31}}^T \\ & I & L_{\mathbf{32}}^T \\ & & I \end{pmatrix},$$

where A_{33} corresponds to the (top-level) separator, and

$$L_{\mathbf{31}} = A_{\mathbf{31}} A_{\mathbf{11}}^{-1}, \quad L_{\mathbf{32}} = A_{\mathbf{32}} A_{\mathbf{22}}^{-1}, \quad \mathbf{F_3} = A_{\mathbf{33}} - L_{\mathbf{31}} A_{\mathbf{11}} L_{\mathbf{31}}^T - L_{\mathbf{32}} A_{\mathbf{22}} L_{\mathbf{32}}^T.$$

When A results from a problem with the low-rank property, $\mathbf{F_3}$ can be approximated by an HSS form, and the above factorization is performed (recursively) in a

structured way (as the structured multifrontal factorization in section 2). Then the resulting factors are structured or data sparse.

In the inversion stage, we have

(3.25)

$$A^{-1} = \begin{pmatrix} I & & -L_{\mathbf{31}}^T \\ & I & -L_{\mathbf{32}}^T \\ & & I \end{pmatrix} \begin{pmatrix} A_{\mathbf{11}}^{-1} & & \\ & A_{\mathbf{22}}^{-1} & \\ & & \mathbf{F}_{\mathbf{3}}^{-1} \end{pmatrix} \begin{pmatrix} I & & \\ & I & \\ -L_{\mathbf{31}} & -L_{\mathbf{32}} & I \end{pmatrix}$$

$$= \begin{pmatrix} \begin{pmatrix} A_{\mathbf{11}}^{-1} & \\ & A_{\mathbf{22}}^{-1} \end{pmatrix} + \begin{pmatrix} -L_{\mathbf{31}}^T \\ -L_{\mathbf{32}}^T \end{pmatrix} \mathbf{F}_{\mathbf{3}}^{-1} \begin{pmatrix} -L_{\mathbf{31}} & -L_{\mathbf{32}} \end{pmatrix} & \begin{pmatrix} -L_{\mathbf{31}}^T \mathbf{F}_{\mathbf{3}}^{-1} \\ -L_{\mathbf{32}}^T \mathbf{F}_{\mathbf{3}}^{-1} \end{pmatrix} \\ \begin{pmatrix} -\mathbf{F}_{\mathbf{3}}^{-1} L_{\mathbf{31}} & -\mathbf{F}_{\mathbf{3}}^{-1} L_{\mathbf{32}} \end{pmatrix} & \mathbf{F}_{\mathbf{3}}^{-1} \end{pmatrix}.$$

Thus,

$$\operatorname{diag}\left(A^{-1}\right) = \begin{pmatrix} \operatorname{diag}(A_{\mathbf{11}}^{-1} + L_{\mathbf{31}}^T \mathbf{F}_{\mathbf{3}}^{-1} L_{\mathbf{31}}) \\ \operatorname{diag}(A_{\mathbf{22}}^{-1} + L_{\mathbf{32}}^T \mathbf{F}_{\mathbf{3}}^{-1} L_{\mathbf{32}}) \\ \operatorname{diag}(\mathbf{F}_{\mathbf{3}}^{-1}) \end{pmatrix}, \tag{3.26}$$

where $-\mathbf{F}_{\mathbf{3}}^{-1} L_{\mathbf{31}}$, $-\mathbf{F}_{\mathbf{3}}^{-1} L_{\mathbf{32}}$, and $\mathbf{F}_{\mathbf{3}}^{-1}$ are submatrices of A^{-1} corresponding to the nonzero blocks of L. This also means that the extraction of $\operatorname{diag}\left(A^{-1}\right)$ needs the computation of the off-diagonal blocks of A^{-1} that fall into the block nonzero pattern of the L factor from the LDL factorization [32].

With the structured factorization, L and $\mathbf{F_3}$ are approximated by data-sparse forms, all the operations in (3.26) can be performed via structured (HSS or low rank) operations. For example, the HSS inversion in section 3.1 can be used to compute $\mathbf{F}_{\mathbf{3}}^{-1}$, and multiplications of HSS matrices and low-rank matrices are used for $-\mathbf{F}_{\mathbf{3}}^{-1} L_{\mathbf{31}}$ and $-\mathbf{F}_{\mathbf{3}}^{-1} L_{\mathbf{32}}$. The idea can then be recursively applied.

Remark 3.1. Since C is generally a dense matrix, a full direct inversion for (3.26) costs $O(n^3)$, which is prohibitive for large n and is impractical. In Table IV of [27], some test examples are given. For small or modest n, the selected inversion is already at least 13 times faster than the full inversion, and often hundreds of times faster. For example, for the test matrix `pwtk` of size $n = 217{,}918$ in section 5, the selected inversion is 353 times faster. In this work, we will further show that our structured selected inversion is faster than the (nonstructured) selected inversion.

3.3. Structures within C and general structured selected inversion. In the general selected inversion, the first stage is a structured multifrontal LDL factorization as in section 2.2, and the second stage is a structured inversion. In the factorization stage, we traverse the assembly tree $\mathcal{T}$ in its postorder, and the resulting factors are in data-sparse forms (HSS or low rank). We then focus on the inversion stage, where we traverse $\mathcal{T}$ in its reverse postorder. We use $C_{\mathbf{i},\mathbf{i}}$ to denote the diagonal block $C|_{\mathbf{s_i} \times \mathbf{s_i}}$ of C in (3.1) corresponding to node $\mathbf{i}$ of $\mathcal{T}$, and use $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ to denote the off-diagonal block $C|_{\mathbf{s}_{\mathcal{N}_\mathbf{i}} \times \mathbf{s_i}}$ of C with $\mathcal{N}_\mathbf{i}$ in Definition 2.1. See (3.25) for an example where, for $\mathbf{i} = 2$, we have $C_{\mathbf{i},\mathbf{i}}$ corresponding to $A_{\mathbf{22}}^{-1} + L_{\mathbf{32}}^T \mathbf{F}_{\mathbf{3}}^{-1} L_{\mathbf{32}}$ and $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ corresponding to $-\mathbf{F}_{\mathbf{3}}^{-1} L_{\mathbf{32}}$.

For the node $\mathbf{k} \equiv \operatorname{root}(\mathcal{T})$, the corresponding diagonal block of C is

$$C_{\mathbf{k},\mathbf{k}} = \mathbf{F}_{\mathbf{k}}^{-1}.$$

Apply the HSS inversion procedure in section 3.1 to the HSS form of the final frontal matrix $\mathbf{F_k}$. The diagonal blocks of $C_{\mathbf{k},\mathbf{k}}$ are simply the $\tilde{D}$ generators of the resulting HSS form of $C_{\mathbf{k},\mathbf{k}}$ as in (3.20).

For a node $\mathbf{i}$ of $\mathcal{T}$ with $\mathbf{i} < \mathbf{k}$, we show a structured computation of $C_{\mathbf{i},\mathbf{i}}$. Let $l_{\mathbf{i}}$ be the row index in A that the first row of $A|_{\mathbf{s_i}\times\mathbf{s_i}}$ in (2.4) corresponds to. The derivation of the representation of $C_{\mathbf{i},\mathbf{i}}$ follows directly from (3.25)–(3.26):

$$C_{\mathbf{i},\mathbf{i}} = F_{\mathbf{i},\mathbf{i}}^{-1} + (L|_{(l_{\mathbf{i}+1}:n)\times(l_{\mathbf{i}}:l_{\mathbf{i}+1}-1)})^T C|_{(l_{\mathbf{i}+1}:n)\times(l_{\mathbf{i}+1}:n)} L|_{(l_{\mathbf{i}+1}:n)\times(l_{\mathbf{i}}:l_{\mathbf{i}+1}-1)}. \tag{3.27}$$

Since $L|_{(l_{\mathbf{i}+1}:n)\times(l_{\mathbf{i}}:l_{\mathbf{i}+1}-1)}$ has many zero blocks (following nested dissection), only the entries of $C|_{(l_{\mathbf{i}+1}:n)\times(l_{\mathbf{i}+1}:n)}$ within the block nonzero pattern of $L|_{(l_{\mathbf{i}+1}:n)\times(l_{\mathbf{i}}:l_{\mathbf{i}+1}-1)}$ are needed to compute $C_{\mathbf{i},\mathbf{i}}$ [32]. Thus, following Definition 2.1 for $\mathcal{N}_{\mathbf{i}}$, we let

$$L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}^T = \begin{pmatrix} (L|_{\mathbf{s_{j_1}}\times\mathbf{s_i}})^T & (L|_{\mathbf{s_{j_2}}\times\mathbf{s_i}})^T & \cdots & (L|_{\mathbf{s_{j_\alpha}}\times\mathbf{s_i}})^T \end{pmatrix},$$

where we suppose

$$\mathcal{N}_{\mathbf{i}} \equiv \{\mathbf{j}_1, \mathbf{j}_2, \ldots, \mathbf{j}_\alpha\}. \tag{3.28}$$

We can similarly form a matrix $C_{\mathcal{N}_{\mathbf{i}},\mathcal{N}_{\mathbf{i}}}$ from $C|_{(l_{\mathbf{i}+1}:n)\times(l_{\mathbf{i}+1}:n)}$ so that the following formulas are used for the actual construction of $C_{\mathbf{i},\mathbf{i}}$ in (3.27):

$$C_{\mathbf{i},\mathbf{i}} = F_{\mathbf{i},\mathbf{i}}^{-1} + L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}^T C_{\mathcal{N}_{\mathbf{i}},\mathcal{N}_{\mathbf{i}}} L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}} = F_{\mathbf{i},\mathbf{i}}^{-1} - L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}^T C_{\mathcal{N}_{\mathbf{i}},\mathbf{i}} \text{ with} \tag{3.29}$$

$$C_{\mathcal{N}_{\mathbf{i}},\mathbf{i}} = -C_{\mathcal{N}_{\mathbf{i}},\mathcal{N}_{\mathbf{i}}} L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}. \tag{3.30}$$

The extraction of (the diagonal blocks of) $C_{\mathbf{i},\mathbf{i}}$ in (3.29) involves these steps:

- apply HSS inversion to the HSS form of $F_{\mathbf{i},\mathbf{i}}$ to extract (the diagonal generators of) $F_{\mathbf{i},\mathbf{i}}^{-1}$;
- compute $C_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}$ in (3.30) via the low-rank structure of $L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}$ and the rank structure of $C_{\mathcal{N}_{\mathbf{i}},\mathcal{N}_{\mathbf{i}}}$ that has already been computed;
- compute $L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}^T C_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}$ in (3.29) via the low-rank structures of $L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}$ and $C_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}$.

The first step is shown in section 3.1. We thus focus on the latter two steps. First, consider $L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}$. Recall that $F_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}$ is in a low-rank form (2.9). Thus, $L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}$ is also in a low-rank form (noticing the notation assumption above (3.1)):

$$L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}} = F_{\mathcal{N}_{\mathbf{i}},\mathbf{i}} F_{\mathbf{i},\mathbf{i}}^{-1} = U_{\bar{k}} B_k^T (U_k^T F_{\mathbf{i},\mathbf{i}}^{-1}). \tag{3.31}$$

We need to compute $U_k^T F_{\mathbf{i},\mathbf{i}}^{-1}$, where U_k is a nested basis matrix. Here, instead of using HSS solutions or HSS matrix-vector multiplications, we can compute $U_k^T F_{\mathbf{i},\mathbf{i}}^{-1}$ with a more compact form based on an idea similar to Theorem 3.4.

THEOREM 3.5. *With the notation in Lemma* 3.3 *and Theorem* 3.4, *we have*

$$U_k^T F_{\mathbf{i},\mathbf{i}}^{-1} = \bar{U}_k \bar{D}_k^{-1} \operatorname{diag}(\tilde{U}_{k_1}^T, \tilde{U}_{k_2}^T),$$

where k_1 and k_2 are the left and right children of k, respectively.

Proof. Similarly to the proof of Lemma 3.3, we use induction and just show the general induction step $U_i^T D_i^{-1} = \bar{U}_i \bar{D}_i^{-1} \operatorname{diag}(\tilde{U}_{c_1}^T, \tilde{U}_{c_2}^T)$ for a nonleaf node i. According to (3.22), (3.23), and the nested form of U_i,

$$\begin{aligned} U_i^T D_i^{-1} &= \begin{pmatrix} R_{c_1}^T & R_{c_2}^T \end{pmatrix} \operatorname{diag}(U_{c_1}^T D_{c_1}^{-1}, U_{c_2}^T D_{c_2}^{-1}) - \begin{pmatrix} R_{c_1}^T & R_{c_2}^T \end{pmatrix} \\ &\quad \cdot H \left[H^{-1} - H^{-1} \left(\begin{pmatrix} & B_{c_1} \\ B_{c_1}^T & \end{pmatrix} + H^{-1} \right)^{-1} H^{-1} \right] \operatorname{diag}(U_{c_1}^T D_{c_1}^{-1}, U_{c_2}^T D_{c_2}^{-1}) \\ &= \begin{pmatrix} R_{c_1}^T & R_{c_2}^T \end{pmatrix} \left(\begin{pmatrix} & B_{c_1} \\ B_{c_1}^T & \end{pmatrix} + H^{-1} \right)^{-1} H^{-1} \operatorname{diag}(U_{c_1}^T D_{c_1}^{-1}, U_{c_2}^T D_{c_2}^{-1}). \end{aligned}$$

According to (3.23) and (3.24), $H = \mathrm{diag}(\hat{D}_{c_1}, \hat{D}_{c_2})$. Thus,

$$U_i^T D_i^{-1} = \bar{U}_i^T \begin{pmatrix} \hat{D}_{c_1} & B_{c_1} \\ B_{c_1}^T & \hat{D}_{c_2} \end{pmatrix}^{-1} \mathrm{diag}(\hat{D}_{c_1} U_{c_1}^T D_{c_1}^{-1}, \hat{D}_{c_2} U_{c_2}^T D_{c_2}^{-1})$$

$$= \bar{U}_i^T \bar{D}_i^{-1} \mathrm{diag}(\hat{D}_{c_1} U_{c_1}^T D_{c_1}^{-1}, \hat{D}_{c_2} U_{c_2}^T D_{c_2}^{-1}).$$

D_{c_1} and D_{c_2} are submatrices of D_i and are also HSS. Let j_1 and j_2 be the children of c_1. By induction,

$$\begin{aligned} \hat{D}_{c_1} U_{c_1}^T D_{c_1}^{-1} &= \hat{D}_{c_1} (\bar{U}_{c_1}^T \bar{D}_{c_1}^{-1} \mathrm{diag}(\tilde{U}_{j_1}^T, \tilde{U}_{j_2}^T)) = (\hat{D}_{c_1} \bar{U}_{c_1}^T \bar{D}_{c_1}^{-1}) \mathrm{diag}(\tilde{U}_{j_1}^T, \tilde{U}_{j_2}^T) \\ &= \begin{pmatrix} \tilde{R}_{j_1}^T & \tilde{R}_{j_2}^T \end{pmatrix} \mathrm{diag}(\tilde{U}_{j_1}^T, \tilde{U}_{j_2}^T) = \tilde{U}_{c_1}^T, \end{aligned}$$

where (3.16) is used with i replaced by c_1. Similarly, we have $\hat{D}_{c_2} U_{c_2}^T D_{c_2}^{-1} = \tilde{U}_{c_2}^T$, and then the theorem holds. ∎

The benefit of this formula can be seen from a pictorial illustration similar to Figure 3. By this formula, $L_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ in (3.31) can be written in a compact form

$$L_{\mathcal{N}_\mathbf{i},\mathbf{i}} = U_{\bar{k}} B_k^T (\bar{U}_k \bar{D}_k^{-1} \mathrm{diag}(\tilde{U}_{k_1}^T, \tilde{U}_{k_2}^T)).$$

Note that at this point, we leave $L_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ in the above low-rank form, which can be conveniently multiplied by the structured form of $C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}}$ later.

Next, we study the structures of some blocks of C. For convenience, we write down the following simple lemma, which can be proven by definition; see, e.g., [20].

LEMMA 3.6. *If F is an invertible matrix with HSS rank bounded by r, then F^{-1} also has HSS rank bounded by r.*

Another lemma shows how the addition of matrices with the same off-diagonal nested bases preserves the structure, and is a direct extension of the results in [39, 42] and [40, Proposition 3.3].

LEMMA 3.7. *Assume F is an HSS matrix with generators D, B, U, V, etc., and corresponds to an HSS tree with root k. Let H be any square matrix with size equal to the column size of the nested basis matrix U_k. Then the following matrix is also an HSS matrix:*

$$F + U_k H V_k^T,$$

and its U, V, R, W generators are the same as those of F, and its D, B generators have the same sizes as (and can be obtained by updating) the D, B generators of F.

We can now prove an important theorem that discloses the rank structures of $C_{\mathbf{i},\mathbf{i}}$ and $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$.

THEOREM 3.8. *Assume all the frontal matrices in the multifrontal factorization of A can be approximated by HSS matrices with HSS ranks bounded by r, so that the factorization is the exact one of $\tilde{A}$. Then for C in* (3.1),

(a) *$C_{\mathbf{i},\mathbf{i}}$ is an HSS matrix with HSS rank bounded by r;*

(b) *$C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ is a low-rank form with rank bounded by r.*

Proof. We prove (a) first. For a node $\mathbf{i}$ of $\mathcal{T}$, consider $C_{\mathbf{i},\mathbf{i}}$ in (3.29). The HSS structure of $F_{\mathbf{i},\mathbf{i}}^{-1}$ follows from Lemma 3.6. According to (3.31) and Theorem 3.5,

(3.32)
$$\begin{aligned} C_{\mathbf{i},\mathbf{i}} &= F_{\mathbf{i},\mathbf{i}}^{-1} + L_{\mathcal{N}_\mathbf{i},\mathbf{i}}^T C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} L_{\mathcal{N}_\mathbf{i},\mathbf{i}} \\ &= F_{\mathbf{i},\mathbf{i}}^{-1} + \operatorname{diag}(\tilde{U}_{k_1}, \tilde{U}_{k_2}) \underbrace{[(B_k^T \bar{U}_k \bar{D}_k^{-1})^T (U_{\bar{k}}^T C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} U_{\bar{k}})(B_k^T \bar{U}_k \bar{D}_k^{-1})]}_{H} \operatorname{diag}(\tilde{U}_{k_1}^T, \tilde{U}_{k_2}^T) \\ &\equiv F_{\mathbf{i},\mathbf{i}}^{-1} + \operatorname{diag}(\tilde{U}_{k_1}, \tilde{U}_{k_2}) H \operatorname{diag}(\tilde{U}_{k_1}^T, \tilde{U}_{k_2}^T). \end{aligned}$$

This indicates that $\operatorname{diag}(\tilde{U}_{k_1}, \tilde{U}_{k_2})$ gives a column basis of $L_{\mathcal{N}_\mathbf{i},\mathbf{i}}^T C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} L_{\mathcal{N}_\mathbf{i},\mathbf{i}}$. On the other hand, $\tilde{U}_{k_1}$ and $\tilde{U}_{k_2}$ are also the generators of $F_{\mathbf{i},\mathbf{i}}^{-1}$ that give the column bases of its appropriate off-diagonal blocks. In another word, the off-diagonal blocks of $F_{\mathbf{i},\mathbf{i}}^{-1}$ and $L_{\mathcal{N}_\mathbf{i},\mathbf{i}}^T C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} L_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ have the same nested basis information. More specifically, we can partition the right-hand side of (3.32) conformably into a block 2×2 form:

$$\begin{aligned} C_{\mathbf{i},\mathbf{i}} &= \begin{pmatrix} (F_{\mathbf{i},\mathbf{i}}^{-1})_{11} & (F_{\mathbf{i},\mathbf{i}}^{-1})_{12} \\ (F_{\mathbf{i},\mathbf{i}}^{-1})_{21} & (F_{\mathbf{i},\mathbf{i}}^{-1})_{22} \end{pmatrix} + \begin{pmatrix} \tilde{U}_{k_1} & \\ & \tilde{U}_{k_2} \end{pmatrix} \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix} \begin{pmatrix} \tilde{U}_{k_1}^T & \\ & \tilde{U}_{k_2}^T \end{pmatrix} \\ &= \begin{pmatrix} (F_{\mathbf{i},\mathbf{i}}^{-1})_{11} + \tilde{U}_{k_1} H_{11} \tilde{U}_{k_1}^T & (F_{\mathbf{i},\mathbf{i}}^{-1})_{12} + \tilde{U}_{k_1} H_{12} \tilde{U}_{k_2}^T \\ (F_{\mathbf{i},\mathbf{i}}^{-1})_{21} + \tilde{U}_{k_2} H_{21} \tilde{U}_{k_1}^T & (F_{\mathbf{i},\mathbf{i}}^{-1})_{22} + \tilde{U}_{k_2} H_{22} \tilde{U}_{k_2}^T \end{pmatrix}. \end{aligned}$$

According to Lemma 3.7, the $(1,1)$ and $(2,2)$ blocks of the above matrix are HSS forms, and the HSS generators are just those of $F_{\mathbf{i},\mathbf{i}}^{-1}$, except the entries (but not the sizes) of the $\tilde{D}, \tilde{B}$ generators of $F_{\mathbf{i},\mathbf{i}}^{-1}$ are updated. In addition, the $(1,2)$ block is

$$(F_{\mathbf{i},\mathbf{i}}^{-1})_{12} + \tilde{U}_{k_1} H_{12} \tilde{U}_{k_2}^T = \tilde{U}_{k_1} (\tilde{B}_{k_1} + H_{12}) \tilde{U}_{k_2}^T,$$

and the size of $\tilde{B}_{k_1} + H_{12}$ is bounded by r.

Thus, $C_{\mathbf{i},\mathbf{i}}$ has the same $\tilde{U}, \tilde{R}$ generators as $F_{\mathbf{i},\mathbf{i}}^{-1}$, and the summation on the right-hand side of (3.32) does not increase the HSS rank, which remains bounded by r.

For (b), $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ has a low-rank form

$$C_{\mathcal{N}_\mathbf{i},\mathbf{i}} = -C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} L_{\mathcal{N}_\mathbf{i},\mathbf{i}} = -(C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} U_{\bar{k}}) B_k^T (\bar{U}_k \bar{D}_k^{-1} \operatorname{diag}(\tilde{U}_{k_1}^T, \tilde{U}_{k_2}^T)).$$

Since B_k is an HSS generator of $F_\mathbf{i}$ and has size bounded by r, the right-hand side has rank bounded by r. ∎

Therefore, the blocks of $C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}}$ in (3.29) can be conveniently represented by HSS or low-rank forms, which then participate in the computation of $C_{\mathbf{i},\mathbf{i}}$. This is illustrated in Figure 4.

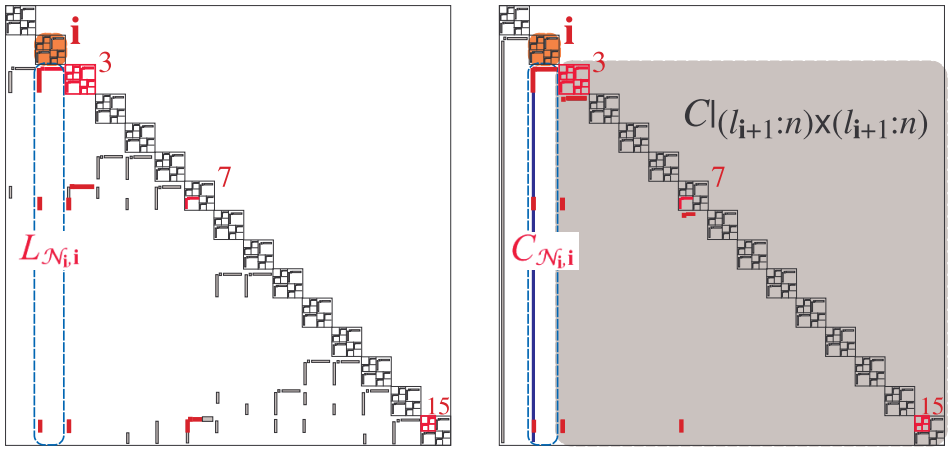
The formula (3.32) in the proof above gives our structured method for computing $C_{\mathbf{i},\mathbf{i}}$. For notational convenience, let

$$\mathcal{U}_{\mathcal{N}_\mathbf{i}} = C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} U_{\bar{k}}, \quad \mathcal{B}_\mathbf{i} = B_k, \quad \mathcal{V}_\mathbf{i}^T = \bar{U}_k \bar{D}_k^{-1} \operatorname{diag}(\tilde{U}_{k_1}^T, \tilde{U}_{k_2}^T); \tag{3.33}$$

then

$$C_{\mathcal{N}_\mathbf{i},\mathbf{i}} = -\mathcal{U}_{\mathcal{N}_\mathbf{i}} \mathcal{B}_\mathbf{i}^T \mathcal{V}_\mathbf{i}^T, \tag{3.34}$$

$$C_{\mathbf{i},\mathbf{i}} = F_{\mathbf{i},\mathbf{i}}^{-1} - L_{\mathcal{N}_\mathbf{i},\mathbf{i}}^T C_{\mathcal{N}_\mathbf{i},\mathbf{i}} = F_{\mathbf{i},\mathbf{i}}^{-1} + \mathcal{V}_\mathbf{i} \mathcal{B}_\mathbf{i} U_{\bar{k}}^T \mathcal{U}_{\mathcal{N}_\mathbf{i}} \mathcal{B}_\mathbf{i}^T \mathcal{V}_\mathbf{i}^T. \tag{3.35}$$



(i) Structures in the factors (ii) Structures in block lower-triangular C

FIG. 4. *The rank structures in the LDL factors of $\tilde{A}$ and in C, and the pieces (marked in red) that are needed to compute $C_{\mathbf{i},\mathbf{i}}$ in* (3.29).

We use the low-rank form (3.34) to form $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$, which is then used in the computation of $C_{\mathbf{i},\mathbf{i}}$ as in (3.35). The structured forms of both $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ and $C_{\mathbf{i},\mathbf{i}}$ then participate in the later inversion steps associated with the descendants of node $\mathbf{i}$.

One task is to compute the product $\mathcal{U}_{\mathcal{N}_\mathbf{i}} = C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} U_{\bar{k}}$ in (3.33). Note that $U_{\bar{k}}$ is a column basis matrix of an off-diagonal block of $\mathbf{F_i}$ in (2.9), and is hierarchically represented by lower-level generators. Thus, the cost of computing $\mathcal{U}_{\mathcal{N}_\mathbf{i}}$ is comparable to the multiplication of an HSS matrix and a nested basis matrix. For simplicity, we first briefly show how to compute $C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} U_{\bar{k}}$ with an explicit form of $U_{\bar{k}}$. For meshes with local connectivity, $\mathcal{N}_\mathbf{i}$ in (3.28) usually has only a finite/small number of nodes. Thus, this only involves a small number of HSS matrix-vector products and low-rank matrix-vector products. That is, we partition $U_{\bar{k}}$ into block rows $U_{\mathbf{j},\bar{k},}$ for $\mathbf{j} \in \mathcal{N}_\mathbf{i}$, so that the row size of $U_{\mathbf{j},\bar{k},}$ is equal to the row size of $C_{\mathbf{j},\mathbf{j}}$. Also partition $\mathcal{U}_{\mathcal{N}_\mathbf{i}}$ into block rows $\mathcal{U}_\mathbf{j}^{(\mathbf{i})} \equiv C_{\mathbf{j},\mathcal{N}_\mathbf{i}} U_{\bar{k}}$ for $\mathbf{j} \in \mathcal{N}_\mathbf{i}$. (3.34) similarly holds for each $\mathbf{j} \in \mathcal{N}_\mathbf{i}$. Thus,

$$\mathcal{U}_\mathbf{j}^{(\mathbf{i})} = C_{\mathbf{j},\mathbf{j}} U_{\mathbf{j},\bar{k},} - \sum_{\mathbf{t}\in\mathcal{N}_\mathbf{i},\mathbf{t}<\mathbf{j}} \mathcal{U}_\mathbf{t}^{(\mathbf{j})} \mathcal{B}_\mathbf{t}^T \mathcal{V}_\mathbf{t}^T U_{\mathbf{t},\bar{k}} - \mathcal{V}_\mathbf{j} \mathcal{B}_\mathbf{j} \mathcal{U}_{\mathcal{N}_\mathbf{j}}^T \hat{U}_{\mathbf{j},\bar{k}}, \tag{3.36}$$

where $\hat{U}_{\mathbf{j},\bar{k}}$ is obtained by stacking all the blocks $U_{\mathbf{t},\bar{k}}$ for $\mathbf{t} \in \mathcal{N}_\mathbf{i}$, $\mathbf{t} > \mathbf{j}$. The first term on the right-hand side involves HSS matrix-vector products, and the remaining two terms are low-rank matrix-vector products. A fast HSS matrix-vector multiplication scheme can be found in [9]. $\mathcal{U}_\mathbf{j}^{(\mathbf{i})}$ can then be quickly computed. After this, stack all the pieces $\mathcal{U}_\mathbf{j}^{(\mathbf{i})}$ to get $\mathcal{U}_{\mathcal{N}_\mathbf{i}}$.

Moreover, since $U_{\bar{k}}$ is a nested basis matrix, the cost of multiplying $C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}}$ and $U_{\bar{k}}$ may be lower than straightforward HSS matrix-vector multiplications. The is because of the existence of possible rank patterns across different levels of the HSS tree (see the next section). That is, in (2.2), U_i may have a larger column size than U_{c_1} or U_{c_2}, i.e., the HSS block corresponding to i may have a higher rank than the individual HSS blocks corresponding to c_1 and c_2. Thus, multiplying a matrix by the nested form of U_i may be cheaper than by the explicit form of U_i. With multilevel hierarchy, the difference in the efficiency can be very significant.

TABLE 1
Major structured operations for computing $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ and $C_{\mathbf{i},\mathbf{i}}$.

	Formulas	Structured operations
$C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$	$C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} U_{\bar{k}}$	Multiplication of HSS and nested basis matrices
	$-\mathcal{U}_{\mathcal{N}_\mathbf{i}} \mathcal{B}_\mathbf{i}^T \mathcal{V}_\mathbf{i}^T$	Multiplication of nested basis matrices
$C_{\mathbf{i},\mathbf{i}}$	$F_{\mathbf{i},\mathbf{i}}^{-1}$	HSS inversion
	$\mathcal{V}_\mathbf{i} \mathcal{B}_\mathbf{i} U_{\bar{k}} \mathcal{U}_{\mathcal{N}_\mathbf{i}} \mathcal{B}_\mathbf{i}^T \mathcal{V}_\mathbf{i}^T$	Multiplications of nested basis matrices
	$F_{\mathbf{i},\mathbf{i}}^{-1} + \mathcal{V}_\mathbf{i} \mathcal{B}_\mathbf{i} U_{\bar{k}} \mathcal{U}_{\mathcal{N}_\mathbf{i}} \mathcal{B}_\mathbf{i}^T \mathcal{V}_\mathbf{i}^T$	HSS generator update

Overall, computing $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ and $C_{\mathbf{i},\mathbf{i}}$ involves the structured operations in Table 1. Some of the matrix products are related to the multiplications of HSS matrices. See [28, section 3.2] for an HSS multiplication scheme. That is, the generators of the product matrix can be computed following a bottom-up traversal of the HSS tree and a top-down traversal. (The latter traversal is not needed during the multiplication of an HSS matrix and a nested basis matrix.) Interested readers can find the detailed derivation of the algorithm in [28, proof of Theorem 3.2.1].

The results of the matrix multiplications in Table 1 are also structured, i.e., can be represented by nested basis matrices. (One way to understand this is to use the idea of the fast multipole method (FMM) [19] when A^{-1} corresponds to a discretized Green's function, together with the fact that $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ is an off-diagonal block of A^{-1}.) Sometimes, after the multiplications, a recompression step [38, section 5] may be used to recover compact structured forms. That is, in a bottom-up traversal of the HSS tree, the basis matrices are compressed, and related upper-level generators are modified. Then in a top-down traversal, the B_i generators are compressed, and the related lower-level generators are modified.

As an example, for the separator $\mathbf{i} = \mathbf{2}$ in Figures 1 and 4, we have $\mathcal{N}_\mathbf{i} = \{\mathbf{3}, \mathbf{7}, \mathbf{15}\}$. Then

$$C_{\mathcal{N}_\mathbf{2},\mathcal{N}_\mathbf{2}} = \begin{pmatrix} C_{\mathbf{3},\mathbf{3}} & \mathcal{V}_\mathbf{3}\mathcal{B}_\mathbf{3} \begin{pmatrix} (\mathcal{U}_\mathbf{7}^{(\mathbf{3})})^T & (\mathcal{U}_\mathbf{15}^{(\mathbf{3})})^T \end{pmatrix} \\ \begin{pmatrix} \mathcal{U}_\mathbf{7}^{(\mathbf{3})} \\ \mathcal{U}_\mathbf{15}^{(\mathbf{3})} \end{pmatrix} \mathcal{B}_\mathbf{3}^T \mathcal{V}_\mathbf{3}^T & \begin{pmatrix} C_{\mathbf{7},\mathbf{7}} & \mathcal{V}_\mathbf{7}\mathcal{B}_\mathbf{7}\mathcal{U}_{\mathcal{N}_\mathbf{7}}^T \\ \mathcal{U}_{\mathcal{N}_\mathbf{7}}\mathcal{B}_\mathbf{7}^T\mathcal{V}_\mathbf{7}^T & C_{\mathbf{15},\mathbf{15}} \end{pmatrix} \end{pmatrix}.$$

See Figure 5 for an illustration of the multiplication of $C_{\mathcal{N}_\mathbf{2},\mathcal{N}_\mathbf{2}}$ with $U_{\bar{k}}$, where a nested form of $U_{\bar{k}}$ is shown in Figure 6.

The overall structured selected inversion scheme is summarized in Algorithm 1. As in the structured multifrontal factorization, a switching level $\mathbf{l}_s$ is involved for the optimization of the complexity in the next section.

Remark 3.2. Throughout the multifrontal method coupled with nested dissection, we only need to work on $L_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ and $C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}}$, which contain the local subblocks of L and C, respectively. This naturally takes advantage of the nonzero pattern of L and has nice *data locality*. On the other hand, the method in [25, 26] uses the indices of the nonzero entries of L and it is not immediately clear which dense pieces need to be put together. In addition, in [26], to find $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$, the blocks $C_{\mathbf{j},\mathbf{i}}$ for all the ancestors $\mathbf{j}$ of $\mathbf{i}$ are visited. Here, we only visit $\mathcal{N}_\mathbf{i}$, which is often just a small subset of the ancestors. $\mathcal{N}_\mathbf{i}$ is determined in the symbolic factorization stage after nested dissection.

Remark 3.3. In terms of the memory, in lines 5, 6, 13, and 14 of Algorithm 1, the storage for the dense or structured blocks of L and Λ may be used to (at least partly) store $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ and $C_{\mathbf{i},\mathbf{i}}$. Thus, the overall memory for the extraction of $\mathrm{diag}(C)$ is

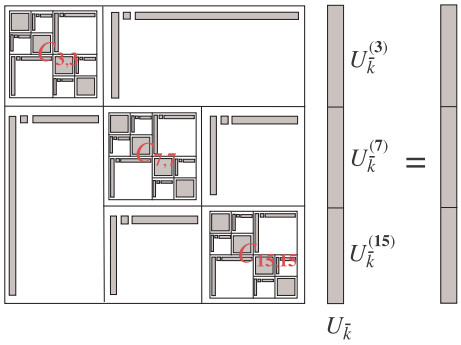


FIG. 5. *The structure of $C_{\mathcal{N}_2\mathcal{N}_2}$ related to Figure 4 (zoomed in) and the multiplication of $C_{\mathcal{N}_2\mathcal{N}_2}$ with $U_{\bar{k}}$, where $U_{\bar{k}}$ is shown in Figure 6.*

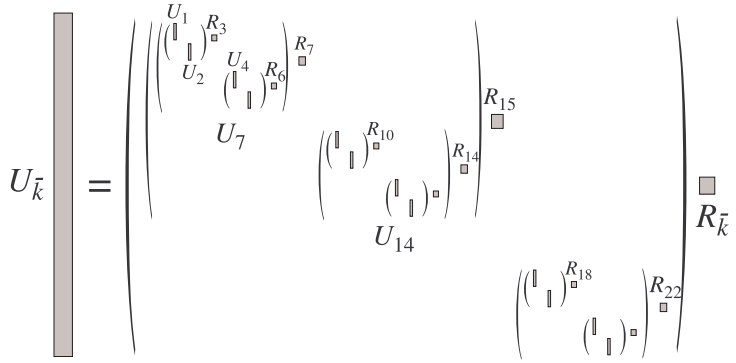


FIG. 6. *A nested basis form of $U_{\bar{k}}$ in Figure 5.*

ALGORITHM 1. STRUCTURED SELECTED INVERSION FOR EXTRACTING THE DIAGONAL BLOCKS OF A^{-1}.

procedure SINV
  Find the diagonal generators of the HSS inverse $C_{\mathbf{k},\mathbf{k}} = F_{\mathbf{k}}^{-1}$ for $\mathbf{k} \equiv \text{root}\,(\mathcal{T})$
  for node/separator $\mathbf{i}$ of $\mathcal{T}$ from $\mathbf{k}-1$ to 1 **do**
    if $\mathbf{i}$ is at level $\mathbf{l} < \mathbf{l}_s$ **then** ▷ *Dense extraction below the switching level* $\mathbf{l}_s$
      $C_{\mathcal{N}_\mathbf{i},\mathbf{i}} \leftarrow -C_{\mathcal{N}_\mathbf{i},\mathcal{N}_\mathbf{i}} L_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ ▷ *From* (3.30); *e.g.,* (3.25)
      $C_{\mathbf{i},\mathbf{i}} \leftarrow F_{\mathbf{i},\mathbf{i}}^{-1} - L_{\mathcal{N}_\mathbf{i},\mathbf{i}}^T C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ ▷ *From* (3.29); *e.g.,* (3.25)
    else ▷ *Dense matrix $L_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ below $\mathbf{l}_s$*
      Find the HSS generators $\tilde{D}_i$ and $\tilde{U}_i$ of $F_{\mathbf{i},\mathbf{i}}^{-1}$ ▷ *Structured* $F_{\mathbf{i},\mathbf{i}}^{-1}$
      Form $\mathcal{B}_\mathbf{i}$ and $\mathcal{V}_\mathbf{i}$ as in (3.33) ▷ *Structures of* $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$
      for node/separator $\mathbf{j} \in \mathcal{N}_\mathbf{i}$ **do** ▷ *Structures of* $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$
        Form $\mathcal{U}_{\mathcal{N}_\mathbf{j}}$ as in (3.33) (with $\mathbf{i}$ replaced by $\mathbf{j}$)
        ▷ *With appropriate changes of the detailed matrices*
      end for
      $C_{\mathcal{N}_\mathbf{i},\mathbf{i}} \leftarrow -\mathcal{U}_{\mathcal{N}_\mathbf{i}} \mathcal{B}_\mathbf{i}^T \mathcal{V}_\mathbf{i}^T$ ▷ *Low-rank approximation of* $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ *as in Table* 1
      $C_{\mathbf{i},\mathbf{i}} \leftarrow F_{\mathbf{i},\mathbf{i}}^{-1} + \mathcal{V}_\mathbf{i} \mathcal{B}_\mathbf{i} U_{\bar{k}} \mathcal{U}_{\mathcal{N}_\mathbf{i}} \mathcal{B}_\mathbf{i}^T \mathcal{V}_\mathbf{i}^T$ ▷ *As in Table* 1
    end if
  end for
end procedure

TABLE 2

Factorization cost ξ_{fact}, inversion cost ξ_{inv}, and storage σ_{mem} for the extraction of $\text{diag}(C)$ *for A discretized on a regular mesh, with the methods in* [23, 25, 26, 27] *or the dense multifrontal method.*

Mesh	ξ_{fact}	ξ_{inv}	σ_{mem}
2D ($m \times m$, $n = m^2$)	$O(n^{1.5})$	$O(n^{1.5})$	$O(n \log n)$
3D ($m \times m \times m$, $n = m^3$)	$O(n^2)$	$O(n^2)$	$O(n^{4/3})$

TABLE 3

Inversion cost $\tilde{\xi}_{\text{inv}}$ and storage $\tilde{\sigma}_{\text{mem}}$ of the structured method for the extraction of $\text{diag}(C)$ *for A discretized on a regular mesh, where r is the maximal HSS rank of all the frontal matrices. (In practice, the HSS ranks may depend on n. The results here based on a maximum rank bound r would then overestimate the actual costs.)*

Mesh	$\tilde{\xi}_{\text{inv}}$	$\tilde{\sigma}_{\text{mem}}$
2D ($m \times m$, $n = m^2$)	$O(rn)$	$O(rn)$
3D ($m \times m \times m$, $n = m^3$)	$O(r^{3/2}n)$	$O(r^{1/2}n)$

about the same as that for the LDL factorization. The blocks of L and Λ are accessed just like in the backward substitution for solving a linear system. Moreover, in the backward traversal of $\mathcal{T}$, some blocks $L_{\mathcal{N}_{\mathbf{j}},\mathbf{j}}$ and $C_{\mathcal{N}_{\mathbf{j}},\mathbf{j}}$ may be discarded when $\mathbf{j}$ is not within any $\mathcal{N}_{\mathbf{i}}$ for the remaining nodes $\mathbf{i}$.

4. Complexity optimization. For the purpose of later comparison, we restate the complexity and memory usage for the direct selected inversion algorithms in [23, 25, 26, 27] as well as the inversion algorithm based on the dense multifrontal method. Assume A is discretized on a 2D $m \times m$ mesh or a 3D $m \times m \times m$ mesh. Then the cost ξ_{fact} of factorizing A, the cost ξ_{inv} of extracting $\text{diag}(C)$, and the memory σ_{mem} are reported in Table 2. Here, σ_{mem} measures the storage for the LDL factors and the blocks of C corresponding to the block nonzero pattern of the factors. (Later, we also use tilde notation for the counts of the structured method. For example, $\tilde{\xi}_{\text{inv}}$ denotes the cost of extracting $\text{diag}(C)$ with the structured inversion.)

Then we turn to the analysis of our structured method when A has the low-rank property. The analysis is similar to that for the structured direct solvers in [39, 40]. We first present the traditional case when the HSS ranks of all the frontal matrices in the LDL factorization are bounded by r, and then show some similar results by letting r grow following certain patterns. In the following remarks, the cost of factorizing A and the memory may be slightly different from those in [39, 40], since we try to optimize the inversion cost.

Remark 4.1. Suppose the structured multifrontal factorization and Algorithm 1 are applied to a discretized matrix A on a 2D $m \times m$ mesh ($n = m^2$) or a 3D $m \times m \times m$ mesh ($n = m^3$), and the HSS ranks of the frontal matrices in the multifrontal method are bounded by r. Choose the switching level $\mathbf{l}_s = O(\log m) - O(\log r)$ of the assembly tree $\mathcal{T}$ so that the inversion costs before and after $\mathbf{l}_s$ are the same. Then after $\tilde{\xi}_{\text{fact}} = O(rn \log n)$ flops in two dimensions and $\tilde{\xi}_{\text{fact}} = O(rn^{4/3})$ in three dimensions for the structured factorization,

- the optimal structured inversion costs $\tilde{\xi}_{\text{inv}}$ are $O(rn)$ in two dimensions and $O(r^{3/2}n)$ in three dimensions;
- the memory requirements $\tilde{\sigma}_{\text{mem}}$ are $O(n \log r) + O(n \log \log n)$ in two dimensions and $O(r^{1/2}n)$ in three dimensions (see Table 3).

The justification of Remark 4.1 follows similar strategies to [41]. We assume the root node of $\mathcal{T}$ is at level 0, and the leaves are at level $\mathbf{l}_{\max} = O(\log m)$.

For convenience, we count the number of separators following the proof of [41, Theorem 4.2]. That is, assume each separator partitions all the directions of a domain so that it splits the domain into 2^d subdomains of the same shape, where $d = 2$ for 2D and $d = 3$ for 3D problems. (For example, a separator has a cross shape in two dinensions.) Thus, at level $\mathbf{l}$ of the assembly tree, there are $2^{d\cdot\mathbf{l}}$ separators, each of the same size $O((\frac{m}{2^{\mathbf{l}}})^{d-1})$. The size of a frontal matrix $\mathbf{F_i}$ at level $\mathbf{l}$ is then

$$N^{(\mathbf{l})} = O\left(\left(\frac{m}{2^{\mathbf{l}}}\right)^{d-1}\right). \tag{4.1}$$

(In selected inversion with the standard multifrontal method, the costs associated with node $\mathbf{i}$ are $O((N^{(\mathbf{l})})^3)$. This is precisely why the selected inversion costs in Table 2 have the same orders as the factorization costs.) In our structured inversion, the factors are data sparse, and the structured operations such as HSS matrix multiplication, addition, and inversion at step $\mathbf{i}$ cost $O(r^2 N^{(\mathbf{l})})$ (see, e.g., [38]).

Unlike the methods in [39, 41], here, we choose the switching level $\mathbf{l}_s$ to minimize $\tilde{\xi}_{\text{inv}}$, so that $\tilde{\xi}_{\text{inv}}$ is linear in n, while $\tilde{\xi}_{\text{fact}}$ and $\tilde{\sigma}_{\text{mem}}$ are roughly of the same order as those in [39, 41]. The dense and structured operations associated with $\mathbf{F_i}$ cost $\mathrm{c}_1(N^{(\mathbf{l})})^3$ and $\mathrm{c}_2 r^2 N^{(\mathbf{l})}$ flops, respectively, where c_1 and c_2 are constants, and the low-order terms are dropped. The total inversion cost is thus

$$\begin{aligned}\tilde{\xi}_{\text{inv}} &= \underbrace{\sum_{\mathbf{l}=\mathbf{l}_s+1}^{\mathbf{l}_{\max}} 2^{d\cdot\mathbf{l}}\mathrm{c}_1(N^{(\mathbf{l})})^3}_{\text{before the switching level}} + \underbrace{\sum_{\mathbf{l}=0}^{\mathbf{l}_s} 2^{d\cdot\mathbf{l}}\mathrm{c}_2 r^2 N^{(\mathbf{l})}}_{\text{after the switching level}} \\ &= \sum_{\mathbf{l}=\mathbf{l}_s+1}^{\mathbf{l}_{\max}} 2^{d\cdot\mathbf{l}}\mathrm{c}_1\left(\frac{m}{2^{\mathbf{l}}}\right)^{3(d-1)} + \sum_{\mathbf{l}=0}^{\mathbf{l}_s} 2^{d\cdot\mathbf{l}}\mathrm{c}_2 r^2\left(\frac{m}{2^{\mathbf{l}}}\right)^{d-1}.\end{aligned} \tag{4.2}$$

For the 2D case ($d = 2$),

$$\tilde{\xi}_{\text{inv}} = \mathrm{c}_1\frac{m^3}{2^{\mathbf{l}_s}} + \mathrm{c}_2 r^2 m 2^{\mathbf{l}_s} + O(m^2).$$

With the optimality condition $\mathrm{c}_1\frac{m^3}{2^{\mathbf{l}_s}} = \mathrm{c}_2 r^2 m 2^{\mathbf{l}_s}$, we have $2^{\mathbf{l}_s} = O(\frac{m}{r})$ or $\mathbf{l}_s = \mathbf{l}_{\max} - O(\log r)$, and get the optimal cost $\tilde{\xi}_{\text{inv}} = O(rm^2) = O(rn)$. The storage can be easily counted. For the 3D case ($d = 3$), the derivation follows similarly.

In practice, the assumption of bounded off-diagonal numerical ranks in Remark 4.1 is not realistic, and may usually be used for preconditioning. For direct solutions, the off-diagonal ranks of the frontal matrices usually depend on the HSS block sizes. For example, if A results from discretized 3D Poisson or Helmholtz equations, it is observed that the off-diagonal numerical ranks of the frontal matrices grow with the HSS block sizes [11, 38, 39]. A rank relaxation idea in [4, 21, 38] can be used to study the approximate patterns of such rank growth. A *rank pattern* r_l measures the maximum numerical rank of the HSS blocks at level l of the HSS tree. For convenience, we say that the HSS matrix *follows the off-diagonal rank pattern* r_l. For example, in Figure 6, upper-level HSS generators are allowed to have larger sizes. It is shown that even if r_l is not bounded, the HSS form may still be very effective. The costs of HSS construction, factorization, and solution for different rank patterns are given in [38]. Here, since the main operations in the selected inversion are HSS matrix-vector

TABLE 4
Costs $\tilde{\xi}_{\text{hssmv}}$ of HSS matrix-vector multiplication and $\tilde{\xi}_{\text{hssmm}}$ of HSS matrix-matrix multiplication with different rank patterns for the HSS blocks.

Rank pattern r_l		$r = \max_l r_l$	ξ_{hssmv}	ξ_{hssmm}
$O(1)$		$O(1)$	$O(N)$	$O(N)$
$O(\log^p N_l)$		$O(\log^p N)$		
$O(N_l^{1/p})$	$p > 3$	$O(N^{1/p})$		
	$p = 3$	$O(N^{1/3})$		$O(N \log N)$
	$p = 2$	$O(N^{1/2})$	$O(N \log N)$	$O(N^{3/2})$
$O(\alpha^{l_{\max}-l} r_0)$	$0 < \alpha < 2^{1/3}$	$< O(N^{1/3})$	$O(N)$	$O(N)$
	$\alpha = 2^{1/3}$	$O(N^{1/3})$		$O(N \log N)$
	$2^{1/3} < \alpha < 2^{1/2}$	$< O(N^{1/2})$		$O(N^{\log \alpha^3})$
	$\alpha = 2^{1/2}$	$O(N^{1/2})$	$O(N \log N)$	$O(N^{3/2})$

or matrix-matrix multiplications, we can get their costs with various rank patterns as follows, and the derivations are the same as those in [38].

Remark 4.2. Let r_l be the maximum rank of the HSS blocks at level l of the HSS tree for two conformably partitioned HSS matrices F and G of order N, and $N_l = O(\frac{N}{2^l})$ be the maximum diagonal block size at level l of the HSS partition. Assume different rank patterns r_l in Table 4 hold. Then the cost of multiplying F with a vector ranges from $O(N)$ to $O(N \log N)$, depending on the actual rank patterns r_l, and the cost of multiplying F and G to get an HSS form of FG ranges from $O(N)$ to $O(N^{3/2})$.

The rank patterns r_l in Table 4 are studied in detail in [38, 39]. They have been observed in various practical problems. Some cases can be roughly shown. For example, for Toeplitz problems in Fourier space, the off-diagonal rank pattern approximately looks like $r_l = O(\log N_l)$ following the idea of FMM [37]. For discretized Poisson's equations in three dimensions, the rank pattern for the intermediate (exact) Schur complements approximately follows $r_l = O(N_l^{1/2})$ under certain conditions [11].

The HSS rank relaxation is further extended to a sparse rank relaxation idea in [39]. Similarly, we can count the costs of the major operations in Table 1 for computing $C_{\mathcal{N}_\mathbf{i},\mathbf{i}}$ and $C_{\mathbf{i},\mathbf{i}}$. We then have the performance results of our structured selected inversion for more general problems where the frontal matrices in the factorization have unbounded HSS ranks.

Remark 4.3. Use the notation in Remark 4.1, and assume that any frontal matrix of size N follows the off-diagonal rank patterns r_l in Tables 5 and 6. Then with the same optimization strategy for $\tilde{\xi}_{\text{inv}}$ as in Remark 4.1,

- the optimal structured inversion cost $\tilde{\xi}_{\text{inv}}$ is $O(n)$ in two dimensions and up to $O(n \log n)$ in three dimensions;
- the memory requirement $\tilde{\sigma}_{\text{mem}}$ is $O(n)$ in two dimensions and up to $O(n \log n)$ in three dimensions;

The details are given in Tables 5 and 6.

The difference between the justification here and that of Remark 4.1 is using the results in Table 4 to replace the operation count in (4.2) associated with a frontal matrix after the switching level. Without loss of generality, we consider one rank pattern $r_l = O(N_l^{1/2})$ for the 3D case, where N has a specific form of $N^{(\mathbf{l})}$ as in (4.1) for a frontal matrix $\mathbf{F_i}$ at level $\mathbf{l}$ of the assembly tree. According to Remark 4.2, the costs of the HSS operations associated with $\mathbf{F_i}$ are bounded by $c_2(N^{(\mathbf{l})})^{3/2}$, where

TABLE 5

Inversion cost $\tilde{\xi}_{\text{inv}}$ and storage $\tilde{\sigma}_{\text{mem}}$ of the structured method for the extraction of $\text{diag}(C)$ *for the matrix A discretized on a $2D$ $m \times m$ regular mesh, where $n = m^2$, $p \in \mathbb{N}$, $\alpha > 0$, r_l is the rank pattern for a frontal matrix of size N as in* [39]*, and N_l is the maximum diagonal block size at level l of the HSS partition of the frontal matrix.*

Rank pattern r_l		$r = \max_l r_l$	$\tilde{\xi}_{\text{inv}}$	$\tilde{\sigma}_{\text{mem}}$
$O(1)$		$O(1)$	$O(n)$	$O(n)$
$O((\log N_l)^p)$		$O((\log N)^p)$		
$O(N_l^{1/p})$	$p > 3$	$O(N^{1/p})$		
	$p = 3$	$O(N^{1/3})$		
	$p = 2$	$O(N^{1/2})$		
$O(\alpha^{l_{\max}-l} r_0)$	$0 < \alpha < 2^{1/3}$	$< O(N^{1/3})$		
	$\alpha = 2^{1/3}$	$O(N^{1/3})$		
	$2^{1/3} < \alpha < 2^{1/2}$	$< O(N^{1/2})$		
	$\alpha = 2^{1/2}$	$O(N^{1/2})$		

TABLE 6

Inversion cost $\tilde{\xi}_{\text{inv}}$ and storage $\tilde{\sigma}_{\text{mem}}$ of the structured method for the extraction of $\text{diag}(C)$ *for the matrix A discretized on a $3D$ $m \times m \times m$ regular mesh, where $n = m^3$, $p \in \mathbb{N}$, $\alpha > 0$, r_l is the rank pattern for a frontal matrix of size N as in* [39]*, and N_l is the maximum diagonal block size at level l of the HSS partition of the frontal matrix.*

Rank pattern r_l		$r = \max_l r_l$	$\tilde{\xi}_{\text{inv}}$	$\tilde{\sigma}_{\text{mem}}$
$O(1)$		$O(1)$	$O(n)$	$O(n)$
$O((\log N_l)^p)$		$O((\log N)^p)$		
$O(N_l^{1/p})$, $p > 3$		$O(N^{1/p})$		
$O(N_l^{1/p})$	$p = 3$	$O(N^{1/3})$		
	$p = 2$	$O(N^{1/2})$	$O(n \log n)$	$O(n \log n)$
$O(\alpha^{l_{\max}-l} r_0)$	$0 < \alpha < 2^{1/3}$	$< O(N^{1/3})$	$O(n)$	$O(n)$
	$\alpha = 2^{1/3}$	$O(N^{1/3})$		
	$2^{1/3} < \alpha < 2^{1/2}$	$< O(N^{1/2})$		
	$\alpha = 2^{1/2}$	$O(N^{1/2})$	$O(n \log n)$	$O(n \log n)$

TABLE 7

Basic idea of a flop count for Remark 4.3.

	Level $\mathbf{l}$	# of separators	Inversion cost with each separator
Dense	$\mathbf{l}_s + 1, \ldots, \mathbf{l}_{\max}$	$8^{\mathbf{l}}$ at level $\mathbf{l}$	$c_1 (N^{(\mathbf{l})})^3$
Structured	$0, 1, \ldots, \mathbf{l}_s$	$8^{\mathbf{l}}$ at level $\mathbf{l}$	$c_2 (N^{(\mathbf{l})})^{3/2}$ (see $r_l = O(N_l^{1/2})$ in Table 4)

c_2 is a constant (see $r_l = O(N_l^{1/p})$ with $p = 2$ in Table 4). The count of the total inversion cost proceeds as in Table 7 (following the discussions above (4.1)).

The inversion cost in three dimensions is thus

$$\begin{aligned}\tilde{\xi}_{\text{inv}} &= \sum_{\mathbf{l}=\mathbf{l}_s+1}^{\mathbf{l}_{\max}} 8^{\mathbf{l}} c_1 (N^{(\mathbf{l})})^3 + \sum_{\mathbf{l}=0}^{\mathbf{l}_s} 8^{\mathbf{l}} c_2 (N^{(\mathbf{l})})^{3/2} = \sum_{\mathbf{l}=\mathbf{l}_s+1}^{\mathbf{l}_{\max}} 8^{\mathbf{l}} c_1 \left(\frac{m}{2^{\mathbf{l}}}\right)^6 + \sum_{\mathbf{l}=0}^{\mathbf{l}_s} 8^{\mathbf{l}} c_2 \left(\frac{m}{2^{\mathbf{l}}}\right)^3 \\ &= c_1 m^6 / 8^{\mathbf{l}_s} + c_2 m^3 \mathbf{l}_s + O(m^3).\end{aligned}$$

With the optimality condition $c_1 m^6 / 8^{\mathbf{l}_s} = c_2 m^3 \mathbf{l}_s$, the minimal cost is $\tilde{\xi}_{\text{inv}} = 2c_2 m^3 \mathbf{l}_s + O(m^3) = O(n \log n)$. The other counts can be shown in the same way.

According to the remarks above, $\tilde{\xi}_{\text{inv}}$ is significantly lower than the standard inversion cost ξ_{inv} in Table 2, especially in three dimensions, where $\xi_{\text{inv}} = O(n^2)$. The memory in three dimensions is reduced from $O(n^{4/3})$ to at most $O(n \log n)$.

Remark 4.4. The shortcut mentioned in Remark 2.2 may be avoided by the randomized method in [40] (which is much more sophisticated and is slower for modest n in our tests in MATLAB). In this case, the factorization cost for three dimensions is $O(n)$ in Remark 4.1, and between $O(n)$ and $O(n^{4/3} \log n)$ in Remark 4.3 [40]. Nevertheless, the inversion costs have the same orders (this is our main focus); see Remark 2.2.

Remark 4.5. In Algorithm 1, the multiplications such as $C_{\mathcal{N}_{\mathbf{i}},\mathcal{N}_{\mathbf{i}}} L_{\mathcal{N}_{\mathbf{i}},\mathbf{i}}$ may involve multiple HSS and nested basis matrix multiplications, followed by possible recompression. Sometimes, it may take less computing time to treat some blocks such as $U_{\bar{k}}$ in Figure 5 as a dense skinny matrix, though this might slightly increase the theoretical flop counts.

5. Numerical experiments. In this section, we test our algorithms on some discretized matrices as well as various more general sparse matrices. They include both 2D and 3D problems. Our structured inversion is compared with the direct selected inversion based on the standard multifrontal method (which has the same complexity as those in [23, 25, 26, 27]). The algorithms are implemented in MATLAB R2013a and carried out on a Unix server with 8-core Intel Xeon-E5 CPUs. The simplification in Remark 4.5 is made in the code.

In order to provide a fair comparison between structured and nonstructured inversion so as to demonstrate the efficiency gained via structured operations, we compare our structured version with our own nonstructured one, where the nonstructured one uses the same ordering as the structured one, except with a different switching level (the top level). In this way, the difference of the two algorithms can be clearly seen from the flops. On the other hand, if we compare our structured code with another multifrontal implementation, we may not have access to the flops or a selected inversion code, and even if we do, the difference in the ordering and other implementation details may affect the comparison. It would be hard to judge whether the performance difference is due to the implementation or the use of structures.

The following notation is used throughout this section:

- `Structured`: our new structured selected inversion;
- `Standard`: the corresponding nonstructured selected inversion which uses dense matrix operations in both the multifrontal factorization and the selected inversion (here, we simply set the switching level $\mathbf{l_s}$ in the structured multifrontal method and Algorithm 1 to be the top level);
- τ: relative compression tolerance used in the HSS construction in `Structured` (the off-diagonal numerical ranks of the frontal matrices are dynamically detected in the HSS construction);
- $e = \frac{\|x-\tilde{x}\|_2}{\|x\|_2}$: the relative accuracy, where $x = \text{diag}(A^{-1})$ is computed by `Standard` and is treated as the exact result, and $\tilde{x} = \text{diag}(C)$ is computed by `Structured`;
- ξ_{fact}, ξ_{inv}, σ_{mem}, $\tilde{\xi}_{\text{fact}}$, $\tilde{\xi}_{\text{inv}}$, $\tilde{\sigma}_{\text{mem}}$, $\mathbf{l}_{\max}$, $\mathbf{l}_{\text{s}}$: defined as in the previous section.

In the examples, we also report the structured multifrontal LDL factorization costs, since the factorization and the flop counts are slightly different from those in [39]. The memory we report is for the LDL factors (noticing Remark 3.3).

Example 1. Consider the Helmholtz equation

$$[-\Delta - \omega^2 v(\mathbf{x})^{-2}]\mathbf{u} = \mathbf{f}, \tag{5.1}$$

TABLE 8

Example 1 *(2D case): Factorization flops, memory (number of nonzeros in factors), selected inversion flops, and relative error* e.

Mesh ($m \times m$)		256×256	512×512	1024×1024	2048×2048
$\mathbf{l}_{\max}$		13	15	17	19
Factorization	`Standard`	3.84e8	3.43e9	2.87e10	2.41e11
	`Structured`	3.12e8	2.09e9	1.20e10	6.51e10
Memory	`Standard`	3.07e6	1.46e7	6.77e7	3.11e8
	`Structured`	2.89e6	1.25e7	5.26e7	2.17e8
Inversion	`Standard`	6.08e8	5.39e9	4.50e10	3.78e11
	`Structured`	6.49e8	3.88e9	2.07e10	9.90e10
e	`Structured`	1.43$e-6$	4.92$e-6$	1.32$e-5$	4.39$e-5$

where ω is the angular frequency and v is the P-wave velocity field. The Helmholtz operator with a frequency 4 Hz in both two dimensions and three dimensions is considered.

For the 2D case, we use a domain size $10,240$ m in each direction and choose $v = 6000$ m/s. The number of grid points in each direction increases, so that the numerical solution tends to the exact one. Similar test models are often used in seismic modeling of the earth's media. The Helmholtz operator is discretized on an $m \times m$ mesh, and the resulting matrix A is of order $n = m^2$ and is indefinite. m ranges from 256 to 2048. We choose $\tau = 10^{-6}$, and fix the number of levels before the switching level to be $\mathbf{l}_{\max} - \mathbf{l}_s = 9$. Here, according to the complexity optimization in section 4, $\mathbf{l}_{\max} - \mathbf{l}_s$ should remain roughly constant so that the inversion costs before and after $\mathbf{l}_s$ are almost equal. The nearly optimal value $\mathbf{l}_{\max} - \mathbf{l}_s$ is thus chosen based on small problem sizes. This idea is similar to that in [41]. The costs, memory, and accuracy are reported in Table 8.

In particular, we also plot the costs in Figure 7(i)–(ii) together with reference lines for $O(n)$. The curve for $\tilde{\xi}_{\text{inv}}$ is close to the $O(n)$ reference line, which indicates the nearly linear selected inversion complexity. In fact, let $\tilde{\xi}_{\text{inv}}(n)$ be the structured inversion cost corresponding to the problem size n, then when n increases, the ratio $\tilde{\xi}_{\text{inv}}(n)/\tilde{\xi}_{\text{inv}}(n/4)$ approaches 4 for `Structured` and $\xi_{\text{inv}}(n)/\xi_{\text{inv}}(n/4)$ approaches 8 for `Standard`. To better see the prefactors in the complexity for `Structured`, we also plot $\tilde{\xi}_{\text{fac}}/n$ and $\tilde{\xi}_{\text{inv}}/n$ in Figure 7(iii)–(iv). Though there is no theoretical justification of the prefactors, $\tilde{\xi}_{\text{fac}}/n$ and $\tilde{\xi}_{\text{inv}}/n$ are observed (through curve fitting) to be close to $O(\log^3 n)$ and $O(\log^2 n)$, respectively. On the other hand, the prefactors ξ_{inv}/n for `Standard` are close to $O(n^{0.5})$, which is consistent with the complexity $\xi_{\text{inv}} = O(n^{1.5})$. We also achieve reasonable accuracies as roughly determined by the compression tolerance τ. Since the accuracy measurement e is the relative forward error in $x = \text{diag}(A^{-1})$, it might be possible for e to increase with n.

Then we consider the Helmholtz operator discretized on a 3D $m_1 \times m_2 \times m_3$ mesh, and the resulting matrix A is of order $n = m_1 m_2 m_3$. The physical parameters are similar to those of the 2D case. The mesh sizes are $100 \times 50 \times 50$, $100 \times 50 \times 100$, $100 \times 100 \times 100$, etc., and n doubles every time and ranges from 250,000 to 4,000,000. We choose $\tau = 10^{-5}$ and fix $\mathbf{l}_{\max} - \mathbf{l}_{\text{s}} = 10$. The factorization costs, memory sizes, inversion costs, and accuracy are shown in Table 9. The ratio $\tilde{\xi}_{\text{inv}}(n)/\tilde{\xi}_{\text{inv}}(n/2)$ for `Structured` decreases from 3.64 to 3.07 when n increases. We expect this ratio to eventually approach 2 when n becomes sufficiently large.

In Figure 8, we also plot the flops divided by n for both methods. The prefactor $\tilde{\xi}_{\text{inv}}/n$ for `Structured` is also observed to be a low-order power of $O(\log n)$ through

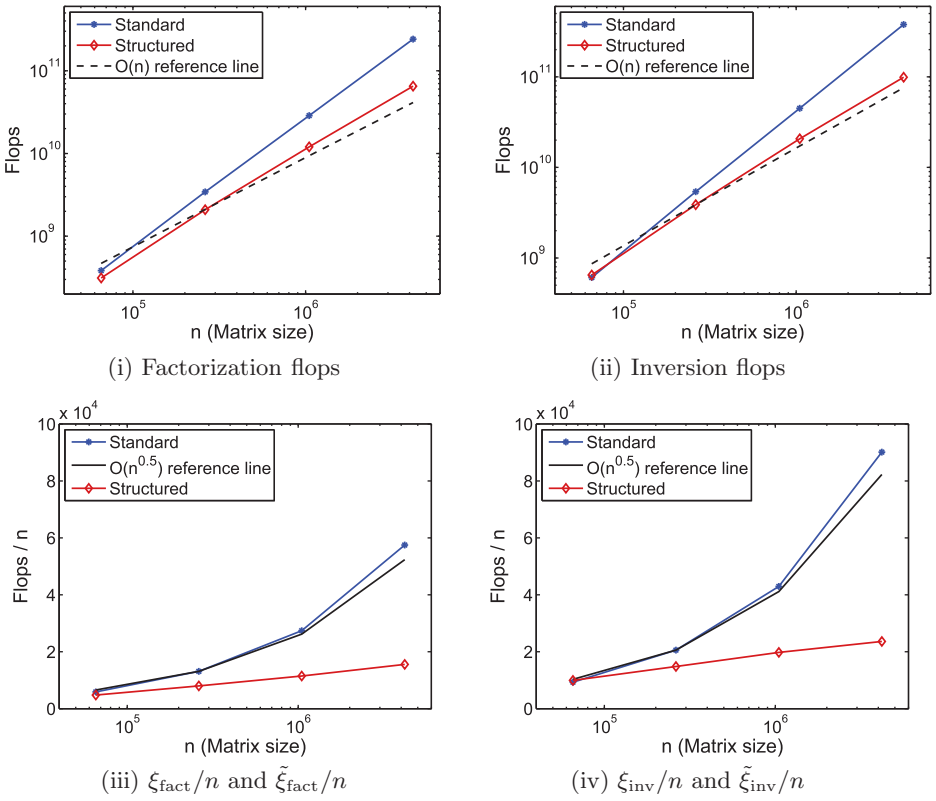


FIG. 7. *Example* 1 *(2D case): Factorization costs and selected inversion costs (divided by* n*).*

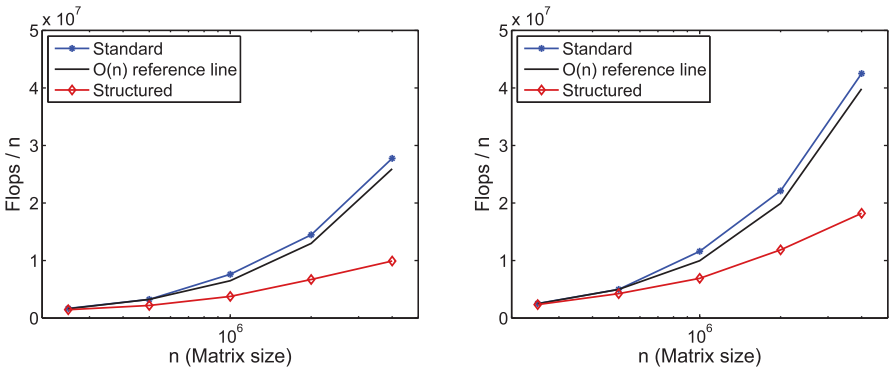
TABLE 9

Example 1 *(3D case): Factorization flops, memory (number of nonzeros in factors), selected inversion flops, and relative error* e*.*

$n\ (= m_1 m_2 m_3)$		250,000	500,000	1,000,000	2,000,000	4,000,000
$\mathbf{l}_{\max}$		13	14	15	16	17
Factorization	`Standard`	4.16e11	1.62e12	7.60e12	2.89e13	1.11e14
	`Structured`	3.62e11	1.09e12	3.75e12	1.34e13	3.96e13
Memory	`Standard`	1.67e8	4.28e8	1.18e9	2.91e9	7.64e9
	`Structured`	1.55e8	3.57e8	8.11e8	1.87e9	4.16e9
Inversion	`Standard`	6.31e11	2.49e12	1.16e13	4.42e13	1.70e14
	`Structured`	5.83e11	2.12e12	6.89e12	2.37e13	7.28e13
e	`Structured`	4.33e − 7	6.84e − 6	8.06e−6	5.85e− 6	3.39e−6

curve fitting. (Ideally, this should be $O(\log n)$.) The prefactor ξ_{inv}/n for `Standard` is close to $O(n)$, which is consistent with the complexity $\xi_{\text{inv}} = O(n^2)$.

Remark 5.1. In practice, the prefactors $\tilde{\xi}_{\text{inv}}/n$ are slightly higher than the theoretical prediction due to several reasons, such as the modest problem sizes, the complexity of the implementation, and, in particular, the discrepancy between the theoretical and the practical rank patterns. In Remark 4.3, it is assumed that the rank patterns hold for all the Schur complements. This is the case if the Schur complements in the relevant problems are *exact* (as in the exact multifrontal method).



(i) ξ_{fact}/n and $\tilde{\xi}_{\text{fact}}/n$ (ii) ξ_{inv}/n and $\tilde{\xi}_{\text{inv}}/n$

FIG. 8. *Example* 1 *(3D case): Factorization costs and selected inversion costs (divided by* n*). (Note that the theoretical inversion cost for* `Structured` *is* $O(n \log n)$ *and yet the vertical axis is the cost divided by* n*, for consistency in the comparison of the two methods.)*

However, the Schur complements in the structured factorization are approximate, as often is in the case of hierarchical structured methods. Within a frontal matrix, the HSS blocks are hierarchically compressed. Within the structured multifrontal scheme, lower-level approximate Schur complements are used in later factorizations so that the structure of upper-level Schur complements may further deviate from the theoretical rank patterns. It is not yet clear how much the deviation of the rank patterns is likely as a low-degree power of $\log n$ (see Remark 2.1). The deviation also depends on the problem and the approximation accuracy. Thus in practice, $\tilde{\xi}_{\text{inv}}$ is likely equal to the estimates in Remark 4.3 magnified by a low-degree power of $\log n$.

Since the current implementation is in MATLAB and `Structured` is much more complicated, its runtime is not very competitive. `Structured` is slower for smaller n and has comparable timing to `Standard` for larger n. For `Structured`, the ratio of the timing still approaches 2 when n doubles. For example, for the 3D cases, for $n = 1{,}000{,}000$, 2,000,000, and 4,000,000, the `Structured` inversion timings are 275 s, 772 s, and 1980 s, respectively. (In the future, it would also be interesting to compare our methods with selected inversion algorithms based on other structured methods such as $\mathcal{H}$-LU factorizations [18, 22].)

Example 2. We then apply our algorithm to more general symmetric sparse matrices, mostly from the University of Florida Sparse Matrix Collection [12]. They include several 3D problems and problems with practical background. See Table 10 for the information on the test matrices.

We choose τ to be between 10^{-5} and 2×10^{-2} so as to reach about four digits of accuracy or higher. $\mathbf{l}_{\max} - \mathbf{l}_{\text{s}}$ also varies for the different matrices. We report the factorization and inversion flops for `Standard` and `Structured` in Table 11, together with the accuracy of `Structured`. `Structured` is faster in both the factorization and the inversion. In particular, for `shallow_water1` and `shallow_water2`, which are quite small, the structured inversion is significantly faster. In fact, according to [13], `shallow_water1` is much slower to solve directly in practice since there are many denormals in the factorization. Here with our structured factorization and inversion, `shallow_water1` can be solved at less cost with even higher accuracy.

In particular, the ratios of the costs of the two methods are given in Table 12. Although all the matrix sizes are relatively small, the structured inversion cost is from

TABLE 10

Example 2: *Test matrices in two and three dimensions from the University of Florida Sparse Matrix Collection (except the case* `random`*), where nnz stands for the number of nonzeros in the matrix.*

Matrix	n	nnz	Description
`apache2`	715,176	4,817,870	3D finite difference matrix from APACHE small
`Dubcova3`	146,689	3,636,643	Matrix from a PDE solver with high fill-in in the leading submatrix
`G3_circuit`	1,585,478	7,660,826	Circuit simulation problem
`parabolic_fem`	525,825	3,674,625	Parabolic FEM problem for constant homogeneous diffusion-convection reaction
`pwtk`	217,918	11,524,432	Stiffness matrix from a pressurized wind tunnel problem
`qa8fm`	66,127	1,660,579	3D acoustic FE mass matrix
`random`	216,000	3,154,318	Ill-conditioned symmetric random matrix from [40] generated with the MATLAB function sprandsym following a 3D tetrahedral grid pattern
`shallow_water1`	81,920	327,680	Automatic differentiation problem from shallow water modeling (slower to solve due to many denormals)
`shallow_water2`	81,920	327,680	Automatic differentiation problem from shallow water modeling
`wathen120`	36,441	301,101	Positive definite matrix in the GHS_psdef group

TABLE 11

Example 2: *Factorization and inversion costs for the matrices in Table* 10.

Matrix	$\mathrm{l}_{\max}$	Factorization flops		Inversion flops		e
		`Standard`	`Structured`	`Standard`	`Structured`	
`apache2`	15	$2.82e11$	$1.35e11$	$4.42e11$	$2.19e11$	$7.94e-5$
`Dubcova3`	15	$2.58e9$	$1.26e9$	$4.05e9$	$2.03e9$	$4.67e-5$
`G3_circuit`	17	$1.10e11$	$4.06e10$	$1.74e11$	$7.10e10$	$2.07e-4$
`parabolic_fem`	15	$9.96e9$	$4.57e9$	$1.60e10$	$7.92e9$	$5.01e-5$
`pwtk`	13	$4.36e10$	$1.11e10$	$6.81e10$	$1.63e10$	$5.22e-5$
`qa8fm`	14	$3.31e10$	$1.22e10$	$5.02e10$	$1.56e10$	$7.73e-5$
`random`	15	$4.19e11$	$6.51e10$	$6.39e11$	$6.63e10$	$4.52e-4$
`shallow_water1`	15	$4.64e8$	$8.93e7$	$7.17e8$	$1.11e8$	$4.39e-10$
`shallow_water2`	15	$4.64e8$	$1.22e8$	$7.17e8$	$1.66e8$	$6.02e-9$
`wathen120`	13	$3.33e8$	$1.14e8$	$5.28e8$	$1.49e8$	$7.02e-5$

2.00 to 9.64 times lower. The largest performance improvement is achieved for the matrix `random` corresponding to a 3D grid. We also notice that the improvement in the inversion cost is generally more significant than that in the factorization cost, which is consistent with the complexity analysis.

Since the implementation is in MATLAB, and the matrix sizes are relatively small, `Structured` is not very competitive in the computational timing, and is slightly slower for some cases. For example, for the case `Dubcova3`, `Standard` and `Structured` take 32 s and 39 s, respectively. We hope a more practical code will provide much better timing in the future.

TABLE 12

Example 2: *Ratios of the costs in Table* 11 *for* `Standard` *over those for* `Structured`*, which measure how much faster* `Structured` *is than* `Standard`*.*

	$\frac{\xi_{\text{fact}}(\texttt{Standard})}{\xi_{\text{fact}}(\texttt{Sstructured})}$	$\frac{\xi_{\text{inv}}(\texttt{Standard})}{\xi_{\text{inv}}(\texttt{Structured})}$
apache2	2.09	2.02
Dubcova3	2.05	2.00
G3_circuit	2.71	2.45
parabolic_fem	2.18	2.02
pwtk	3.93	4.18
qa8fm	2.71	3.22
random	6.44	9.64
shallow_water1	5.20	6.46
shallow_water2	3.80	4.32
wathen120	2.92	3.54

6. Conclusions. We show a structured selected inversion scheme for symmetric sparse matrices. When the matrices have the low-rank property, the selected inversion cost and the memory requirement are nearly linear in n in both two and three dimensions. This is significantly more efficient than some recently proposed direct selected inversion methods. The work here serves as a first attempt in applying structured techniques to selected inversion. We would like to mention that the prefactors in the complexity of these types of structured factorizations (and inversions) are usually quite big, which is also the case for a lot of existing structured solvers. One reason is the overhead. Another reason is that the intermediate off-diagonal numerical ranks are generally considered relatively small only when n is quite large. This is problem dependent. In addition, the assumptions in Remark 4.3 are theoretical, and the practical rank behaviors may be worse by a low-degree power of $\log n$. These structured direct methods are more attractive for large-scale problems. The algorithms are quite involved and we are in the process of developing a parallel code to test larger n for real-world applications such as those in seismic imaging [36]. In our future work, we would also like to compare the method with some optimized nonstructured factorization and inversion codes (when available). Nevertheless, we already observe satisfactory gains in the efficiency with the current preliminary implementation. For example, for $n = 2048^2$ in Table 8, we observe a speedup of about four times in the inversion complexity. In Table 12, gains up to 9.6 times are observed for modest n.

The studies of the structures in C here are useful for the structured solution of higher dimensional problems. Our inversion scheme can also be extended to the extraction of general off-diagonal entries of C. This will appear in our future work.

Acknowledgment. We are grateful to the editor and the anonymous referees for the suggestions which greatly help to improve the paper.

REFERENCES

[1] P. R. AMESTOY, I. S. DUFF, J. Y. L'EXCELLENT, Y. ROBERT, F. H. ROUET, AND B. UÇAR, *On computing inverse entries of a sparse matrix in an out-of-core environment*, SIAM J. Sci. Comput., 34 (2012), pp. A1975–A1999.

[2] M. BEBENDORF AND W. HACKBUSCH, *Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-Coefficients*, Numer. Math., 95 (2003), pp. 1–28.

[3] C. BEKAS, A. CURIONI, AND I. FEDULOVA, *Low cost high performance uncertainty quantification*, in Proceedings of the 2nd Workshop on High Performance Computational Finance, ACM, New York, 2009, 8.

[4] S. Börm, *Adaptive variable-rank approximation of general dense matrices*, SIAM J. Sci. Comput., 30 (2008), pp. 148–168.
[5] S. Börm, *Efficient Numerical Methods for Non-local Operators*, EMS Tracts Math. 14, European Mathematical Society, Zurich, 2010.
[6] S. Börm and W. Hackbusch, *Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices*, Computing, 69 (2002), pp. 1–35.
[7] S. Börm, L. Grasedyck, and W. Hackbusch, *Introduction to hierarchical matrices with applications*, Eng. Anal. Bound. Elem., 27 (2003), pp. 405–422.
[8] S. Cauley, J. Jain, C. K. Koh, and V. Balakrishnan, *A scalable distributed method for quantum-scale device simulation*, J. Appl. Phys., 101 (2007), 123715.
[9] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals, *A fast solver for HSS representations via sparse matrices*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 67–81.
[10] S. Chandrasekaran, M. Gu, and T. Pals, *A fast ULV decomposition solver for hierarchically semiseparable representations*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 603–622.
[11] 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 (2010), pp. 2261–2290.
[12] T. A. Davis and Y. Hu, *University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/research/sparse/matrices/index.html.
[13] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.
[14] J. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.
[15] J. R. Gilbert and S.-H. Teng, *MESHPART, A Matlab Mesh Partitioning and Graph Separator Toolbox*, http://www.cerfacs.fr/algor/Softs/MESHPART/.
[16] A. Gillman, P. Young, and P. G. Martinsson, *A direct solver with $O(N)$ complexity for integral equations on one-dimensional domains*, Front. Math. China, 7 (2012), pp. 217–247.
[17] L. Grasedyck, R. Kriemann, and S. Le Borne, *Domain-decomposition based $\mathcal{H}$-LU preconditioners*, in Domain Decomposition Methods in Science and Engineering XVI, Lecture Notes in Computational Sci. Eng. 55, O. B. Widlund and D. E. Keyes, eds., Springer, Berlin, 2007, pp. 661–668.
[18] L. Grasedyck, R. Kriemann, and S. Le Borne, *Domain decomposition based $\mathcal{H}$-LU preconditioning*, Numer. Math., 112 (2009), pp. 565–600.
[19] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.
[20] W. Hackbusch, B. N. Khoromskij, and R. Kriemann, *Hierarchical matrices based on a weak admissibility criterion*, Computing, 73 (2004), pp. 207–243.
[21] W. Hackbusch, B. N. Khoromskij, and S. Sauter, *On $\mathcal{H}^2$-matrices*, in Lectures on Applied Mathematics, Springer, Berlin, 2000, pp. 9–29.
[22] S. Le Borne, *$\mathcal{H}$-FAINV: Hierarchically Factored Approximate Inverse Preconditioners*, private communication.
[23] S. Li, S. Ahmed, G. Klimeck, and E. Darve, *Computing entries of the inverse of a sparse matrix using the FIND algorithm*, J. Comput. Phys., 227 (2008), pp. 9408–9427.
[24] S. Li and E. Darve, *Extension and optimization of the FIND algorithm: Computing Green's and less-than Green's functions*, J. Comput. Phys., 231 (2012), pp. 1121–1139.
[25] L. Lin, J. Lu, L. Ying, R. Car, and W. E, *Fast algorithm for extracting the diagonal of the inverse matrix with application to the electronic structure analysis of metallic systems*, Commun. Math. Sci. 7 (2009), pp. 755–777.
[26] L. Lin, C. Yang, J. Lu, L. Ying, and W. E, *A fast parallel algorithm for selected inversion of structured sparse matrices with application to 2D electronic structure calculations*, SIAM J. Sci. Comput., 33 (2011), pp. 1329–1351.
[27] L. Lin, C. Yang, J. Meza, J. Lu, L. Ying, and W. E, *SelInv–An algorithm for selected inversion of a sparse symmetric matrix*, ACM Trans. Math. Software, 37 (2011), 40.
[28] W. Lyons, *Fast Algorithms with Applications to PDEs*, Ph.D. thesis, University of California Santa Barbara, Santa Barbara, CA, 2005.
[29] P. G. Martinsson, *A fast direct solver for a class of elliptic partial differential equations*, J. Sci. Comput., 3, 2009, pp. 316–330.
[30] METIS, *Family of Graph and Hypergraph Partitioning Software Applications*, http://glaros.dtc.umn.edu/gkhome/views/metisviews/metis.
[31] P. Schmitz and L. Ying, *A fast direct solver for elliptic problems on general meshes in 2D*, J. Comput. Phys., 231 (2012), pp. 1314–1338.

[32] K. Takahashi, J. Fagan, and M. Chin, *Formation of a sparse bus impedance matrix and its application to short circuit study*, Proceedings 8th Power Industry Computer Applications Conference, Minneapolis, MN, 1973, IEEE, New York, 1973.
[33] J. Tang and Y. Saad, *A Probing Method for Computing the Diagonal of a Matrix Inverse*, Numer. Linear Algegra Appl., 19 (2012), pp. 485–501.
[34] J. M. Tang and Y. Saad, *Domain-decomposition-type methods for computing the diagonal of a matrix inverse*, SIAM J. Sci. Comput., 33 (2011), pp. 2823–2847.
[35] R. B. Sidje and Y. Saad, *Rational Approximation to the Fermi-Dirac Function with Applications in Density Functional Theory*, Numer. Algorithms, 56 (2011), pp. 455–479.
[36] S. Wang, M. V. de Hoop, J. Xia, and X. S. Li, *Massively parallel structured multifrontal solver for time-harmonic elastic waves in 3D anisotropic media*, Geophys. J. Int., 191 (2012), pp. 346–366.
[37] Y. Xi, J. Xia, S. Cauley, and V. Balakrishnan, *Superfast and stable structured solvers for Toeplitz least squares via randomized sampling*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 44–72.
[38] J. Xia, *On the complexity of some hierarchical structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 388–410.
[39] J. Xia, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A832–A860.
[40] J. Xia, *Randomized sparse direct solvers*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 197–227.
[41] 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 (2010), pp. 1382–1411.
[42] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976.