A fast direct elliptic solver via interconnected hierarchical structures[☆]

Xiao Liu[a], Jianlin Xia[b,*], Maarten V. de Hoop[a]

[a] *Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005, USA.*
[b] *Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA.*

Abstract

We propose a fast direct solver for discretized elliptic problems via interconnected hierarchical rank structures with a minimum amount of low-rank compression operations. The interconnected structures within two hierarchical layers are exploited: the hierarchical partitioning of a large problem into subproblems that are local Schur complements on smaller subdomains, and the interconnected hierarchical structured approximations of the subproblems. The interconnected structures make it feasible to extensively reuse off-diagonal basis matrices produced in the rank-structured factorization of smaller local Schur complements. Such basis matrices are produced only once and then reused across multiple hierarchical levels of the sparse factorization. Unlike many existing rank-structured direct solvers where explicit low-rank compression is often the major computation, our new solver can then avoid most of the compression operations. This helps to both reduce the cost and preserves the rank structures. The interconnected structures are further extended to accelerate factorization update problems where many local coefficient updates are involved. Numerical tests on Poisson's equation and the Helmholtz equation are used to demonstrate the efficiency and speedup. In particular, certain reuse factors in our tests indicate dramatic reduction in the number of low-rank compression operations.

Key words: fast sparse direct solver, elliptic equation, interconnected hierarchical structure, basis reuse, neighbor tree, Schur complement update

1. Introduction

This paper studies the fast direct solution of elliptic PDEs of the form

$$Lu(x) = f(x), \quad x \in \Omega, \tag{1.1}$$

where L is a second-order elliptic partial differential operator in the domain of interest Ω. When the PDE is discretized with locally supported basis functions, a linear system with a sparse coefficient matrix A is obtained. Here, we are interested in designing a fast direct solver for the sparse linear system by exploring features from both the discretization and some intrinsic rank structures.

Traditional direct solvers with nested dissection ordering [11] of the matrix A has complexities $O(n^{3/2})$ and $O(n^2)$ flops for two and three dimensions, respectively, where n is the order of A. The multifrontal method [7] is one of the most popular sparse direct solvers and it performs the sparse factorization in terms of multiple smaller local dense matrices called frontal matrices. These direct solvers are generally expensive for large problems where the local dense matrices become challenging to store and factorize.

A lot of recent developments on sparse direct solvers focus on approximating the intermediate dense matrices by rank-structured representations. For elliptic problems, the intermediate dense matrices have

[☆]The research of Jianlin Xia was supported in part by an NSF grant DMS-1819166. Maarten V. de Hoop gratefully acknowledges support from the Simons Foundation under the MATH+X program, the NSF grant DMS-1559587, and the corporate members of the Geo-Mathematical Group at Rice University and Total.

[*]Corresponding author.
Email addresses: xiao.liu@rice.edu (Xiao Liu), xiaj@purdue.edu (Jianlin Xia), mdehoop@rice.edu (Maarten V. de Hoop)

off-diagonal blocks with small numerical ranks (see, e.g., [2, 5]). Low-rank approximations to these off-diagonal blocks yield rank-structured representations that effectively reduce the amount of data. Some frequently used rank-structured representations are $\mathcal{H}$-matrices [17], $\mathcal{H}^2$-matrices [16], and hierarchically semiseparable (HSS) matrices [6, 43]. Such representations often allow fast matrix-vector multiplication, factorization, inversion, and so on. Examples of rank-structured sparse direct solvers are $\mathcal{H}$-LU methods [15] and structured multifrontal methods [12, 41, 40, 45]. Specialized PDE solvers have also been proposed in [8, 18, 21, 23, 29, 33, 42] and can exploit additional properties of Cartesian coordinates and elliptic boundary value problems. Examples are Dirichlet-to-Neumann formulations in [14, 18, 29] and Robin-to-Robin formulations in [13, 27, 31].

When these structured solvers are applied to the discretized system for (1.1), they heavily rely on the construction of intermediate rank-structured representations. The rank-structured constructions are usually done through repeated use of low-rank compression techniques such as rank-revealing factorizations and randomized methods. See [12, 24, 28, 43] for some examples. These constructions typically contribute to the dominant cost of the sparse structured solvers [12, 41, 40, 35, 45]. They also make the implementation of the solvers sophisticated. Thus, some methods sacrifice certain efficiency to simplify the implementation by allowing some dense intermediate operations [1, 35, 41].

Here, we propose an interconnected hierarchical structured solver that extensively *reuses information* to significantly reduce the cost for structured construction. Unlike many existing approaches, we can preserve off-diagonal basis matrices among local dense matrices across different levels of the sparse factorization. This avoids repetitive low-rank compression operations as well as dense intermediate Schur complements. Specifically, the novelties of our solver include the following.

- *Interconnected tree structures* are designed to organize the factorization of the sparse solver. After repeated bisection of the domain, an *outer tree* structure is generated to govern the sparse factorization, and the tree nodes correspond to lists of interfaces for subdomains. Each node of the outer tree is further associated with an *inner tree* used for local structured operations. Inner trees at different outer levels *share subtrees* and are interconnected. Subtrees of the inner trees are preserved as much as possible when we traverse the outer tree, which is a major difference between our solver and existing solvers in [40, 41] that also use two layers of trees. The sharing of subtrees is natural following our discretizations and the levelwise elimination.
- *Interconnected hierarchically semiseparable (IHSS) matrices* are proposed to enable an efficient rank-structured factorization by taking advantage of both the intrinsic rank structures of the problem and the particular discretizations we follow (high-order finite element method with Robin-to-Robin formulations). Each inner tree is associated with an IHSS matrix corresponding to a local dense Schur complement. (The concept of a local Schur complement will be made precise later.) Following interconnected inner trees, IHSS matrices are also interconnected via the *reuse of basis matrices*. That is, IHSS matrices at different outer factorization levels share the same off-diagonal basis matrices. This avoids the majority of the cost needed for low-rank compression.
- Unlike most other structured sparse direct solvers that need extensive off-diagonal compression, here the off-diagonal basis matrices are only computed once at a lower sparse factorization level and then *reused for later factorization levels*. This avoids the majority of the cost needed for low-rank compression. The basis reuse in IHSS structures helps to not only save the cost but also preserve the rank structures.
- For some discretized elliptic problems, the complexity of our direct solver is roughly $O(n)$.

Thus, along the outer tree, local dense factorizations are performed in terms of IHSS structures. We will illustrate why such IHSS structures are feasible and how the off-diagonal bases are shared across different outer factorization levels. In fact, IHSS approximations at upper factorization levels can be conveniently obtained via fast updates of lower-level IHSS forms. Therefore, our solver does not need the expensive rank-structured construction used in other structured sparse direct solvers. The performance of the solver is confirmed in numerical tests on Poisson's equation and the Helmholtz equation. Our tests further show by how many times the basis matrices can be reused, which indicates significant reduction in the number of low-rank compression operations.

Our solver can also be extended to a challenging sparse factorization update problem, where there are multiple local changes to the coefficient of the PDE. We briefly show how to use our IHSS algorithms to

accelerate the factorization of some exterior problems in a recent factorization update algorithm in [27].

The remaining sections are organized as follows. In Section 2, we give a basic framework that we follow for the design of our sparse direct elliptic solver. Section 3 presents the detailed interconnected hierarchical structured factorization, its advantages, and the complexity optimization. The extension to sparse factorization update is discussed in Section 4. Numerical examples are shown in Section 5 and some conclusions are drawn in Section 6.

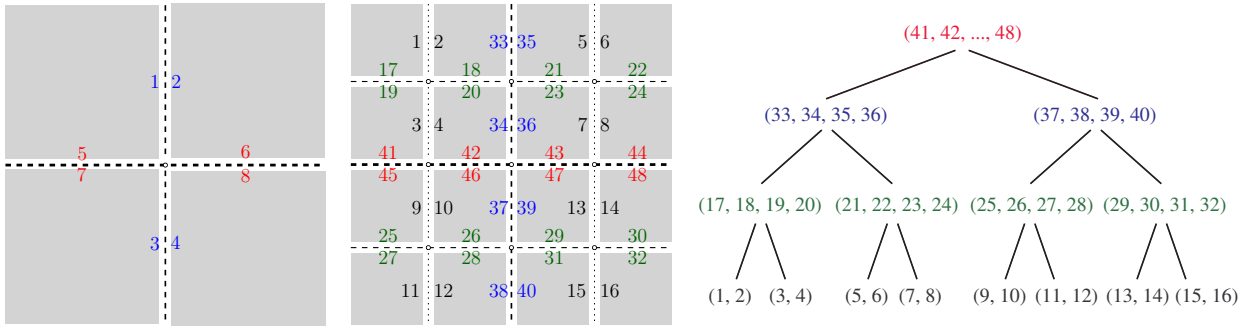
2. Basic framework of our sparse direct elliptic solver

Sparse direct solvers often organize local factorizations following certain hierarchical tree structures. A typical example is the supernodal multifrontal factorization [7, 25] using the assembly tree structure. We first show a basic framework that we follow for the direct solution of elliptic PDEs. The hierarchical factorization is organized based on certain features of the discretization so as to facilitate the design of the interconnected hierarchical structures later in Section 3.

2.1. Domain partitioning, interface organization, and neighbor tree

We start from a hierarchical domain partitioning based on repeated bisection. Figure 2.1(a) illustrates a simple 2-level domain partitioning. Initially, a horizontal cut splits the domain into two level-1 subdomains. The *interfaces* or boundaries along the horizontal cut (marked as $5, 6, 7, 8$ in red) are labeled as the level-1 interfaces. Here, the two sides of the same cut have distinct labels. Next, for each level-1 subdomain, a vertical cut splits it into two level-2 subdomains. The interfaces along the vertical cuts (marked as $1, 2$ and $3, 4$ in blue) are labeled as level-2 interfaces. This process then repeats. A 4-level partitioning is shown in Figure 2.1(b).

In the sparse factorization, all the finest level interior mesh points and the outermost boundary points of the entire domain are eliminated first. Thus, all our later discussions focus on processing the interfaces. Following the multilevel bisection, the interfaces can be organized into a binary tree $\mathbf{T}$ similarly to a separator tree from nested dissection [11]. Each node of $\mathbf{T}$ corresponds to a separator (or cut) of the mesh. Figure 2.1(c) shows the tree corresponding to Figure 2.1(b). $\mathbf{T}$ can essentially be used as an assembly tree [25] for sparse eliminations. The difference is that a separator or tree node here consists of a set of interfaces of subdomains. In addition, the leaf nodes of $\mathbf{T}$ are also interfaces instead of finest level subdomains. Later, we use a set of interfaces grouped inside a pair of parentheses to denote a separator as marked in 2.1(c). Also, we do not distinguish between a node of $\mathbf{T}$ and a separator in the mesh.



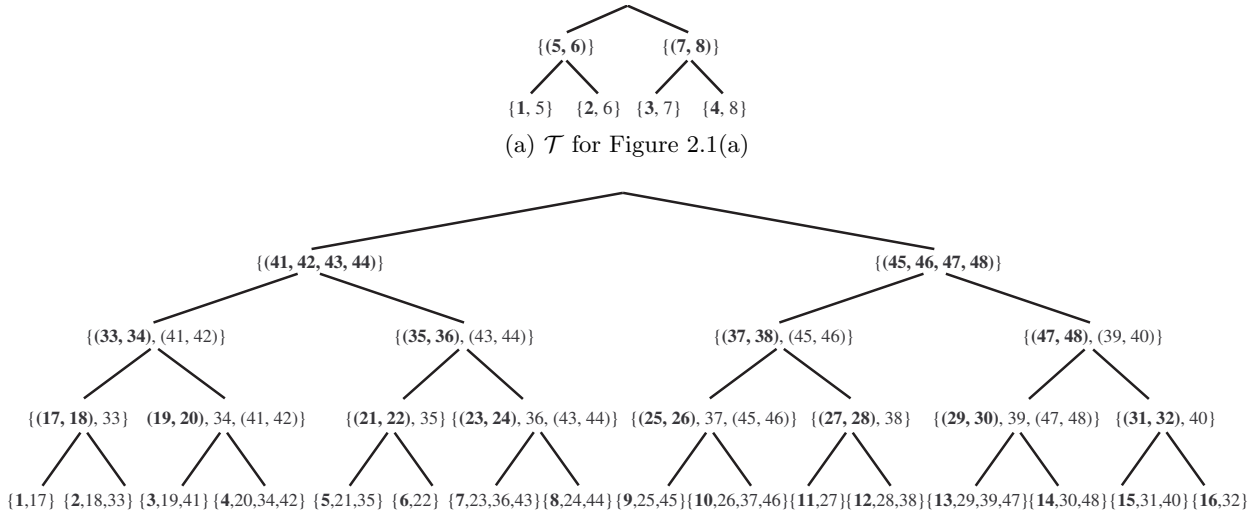
(a) 2-level partitioning (b) 4-level partitioning (c) Assembly/separator tree $\mathbf{T}$ for (b)

Figure 2.1: *Partitioning of a domain. The dashed lines are for the purpose of illustrating the partitioning and are not part of the domain. The interfaces of the subdomains introduced by the partitioning are indexed.*

The interfaces at different levels of the tree $\mathbf{T}$ correspond to intermediate Schur complements at different levels of sparse factorization. In the PDE solution, unknowns associated with lower-level interfaces are ordered before upper-level interfaces and are eliminated first. The interfaces are also labeled consecutively at each level so that a convenient levelwise presentation can be used when we discuss the actual factorization later.

The elimination of lower-level separators mutually connects some upper-level separators and creates fill-in in the Schur complements [30]. To clearly identify such connections, we use the following terms and notation.

- For a separator S at level l of $\mathbf{T}$, its *neighbors* are those interfaces which are at upper levels and are connected to S due to the lower-level eliminations. A similar concept is used in [41, 42] for deriving some rank structured solvers. Accordingly, S is said to be a *pivot separator* (with respect to its neighbors). For example, the pivot separator $(3, 4)$ (formed by interfaces $3, 4$) in Figure 2.1(b–c) has neighbors or neighbor interfaces $19, 20, 34, 41, 42$.
- A *pivot interface* within a pivot separator at level l is the set of all interfaces belonging to one same level-l subdomain in the bisection process. We mark pivot interfaces in bold fonts. For example, during the elimination of the separator $(17, 18, 19, 20)$ in Figure 2.1(b–c), $(\mathbf{17}, \mathbf{18})$ is a pivot interface and so is $(\mathbf{19}, \mathbf{20})$. Thus, each pivot separator includes two pivot interfaces associated with a pair of adjacent subdomains.
- For a pivot interface σ (within a pivot separator S at level l), the *neighbor list* associated with σ is the list consisting of σ itself and those interfaces that are neighbors of S and are also in the same level-l subdomain as σ. We use a pair of braces to identify a neighbor list. We also use a pair of parentheses to group interfaces from the same separator. For example, the pivot interface $(\mathbf{19}, \mathbf{20})$ in Figure 2.1(b–c) has neighbor list $\{(\mathbf{19}, \mathbf{20}), 34, (41, 42)\}$.
- In a neighbor list, the interfaces that are from one same separator S in $\mathbf{T}$ are said to be *common-origin interfaces*, and the separator S is the *origin* of the interfaces. For example, in Figure 2.1(b), $19, 20$ are common-origin interfaces in the neighbor list $\{(\mathbf{19}, \mathbf{20}), 34, (41, 42)\}$ since they are from the same origin which is the separator $(17, 18, 19, 20)$ of $\mathbf{T}$ in Figure 2.1(c). $41, 42$ are also common-origin interfaces in this neighbor list.
- Finally, we introduce another tree $\mathcal{T}$ called *neighbor tree*. Each node of $\mathcal{T}$ is a neighbor list associated with a pivot interface, except the root which is empty. See Figure 2.2. The neighbor tree $\mathcal{T}$ plays a key role in the design of our IHSS structures. In our later discussions, the sparse factorization will mainly be discussed in terms of the neighbor tree $\mathcal{T}$, with the aid of the assembly/separator tree $\mathbf{T}$. For two neighbor lists that are siblings at level l of $\mathcal{T}$, their pivot interfaces form one separator at level l of $\mathbf{T}$. Here, we assume $\mathbf{T}$ has levels $1, 2, \ldots, \mathbf{l}$ but $\mathcal{T}$ has levels $0, 1, \ldots, \mathbf{l}$. The reason is that $\mathcal{T}$ has an empty root at level 0.



(a) $\mathcal{T}$ for Figure 2.1(a)

(b) $\mathcal{T}$ for Figure 2.1(b)

Figure 2.2: *Illustration of neighbor trees $\mathcal{T}$ for organizing neighbor lists.*

2.2. Initial Schur complements from the finest subdomains

For elliptic PDEs like (1.1), the sparse direct factorization involves particular features in the Schur complements. We show the strategies we follow for processing the Schur complements. We first describe

how to form an *initial Schur complement* that results from the elimination of the interior mesh points in all the finest subdomains as well as all exterior boundary points of the entire domain. Thus, the initial Schur complement corresponds to all the leaves in the neighbor tree $\mathcal{T}$ and is denoted $\mathbf{S^{(1)}}$.

$\mathbf{S^{(1)}}$ is computed from factorizing the discretized PDE restricted to the disjoint finest subdomains. Various discretization methods have been used for the restricted problems in subdomains, including finite difference methods in [21, 42], spectral collocation methods in [29], and finite element methods in [11, 18]. Here, we use a high-order finite element method with *Lagrange bases* set up according to [20]. Within each subdomain, the factorization usually yields a dense Schur complement that corresponds to a pseudo-differential operator [4, Section 1.2]. We choose the recent Robin-to-Robin formulation in [13, 31]. Note that unknowns on the corners of subdomains give rise to book-keeping issues [32] and can be removed by reducing the polynomial order on the boundary [13]. We use *Legendre polynomials on the boundary*, and this reduction of order is an orthogonal projection. Thus, in our organization of the interfaces as in Figure 2.1, there are no unknowns associated with the corner mesh points of the subdomains.

We follow the derivation of [31, Equation 2.9] to get the initial Schur complement $\mathbf{S^{(1)}}$. For simplicity, we illustrate the form of $\mathbf{S^{(1)}}$ using the 2-level partitioning in Figure 2.1(a). Here, $\mathbf{l} = 2$ and the neighbor tree $\mathcal{T}$ is shown in Figure 2.2(a). $\mathbf{S}^{(2)}$ has the following form:

$$\mathbf{S}^{(2)} = \begin{pmatrix} \mathbf{S}_{11}^{(2)} & I & & & \mathbf{S}_{15}^{(2)} & & & \\ I & \mathbf{S}_{22}^{(2)} & & & & \mathbf{S}_{26}^{(2)} & & \\ & & \mathbf{S}_{33}^{(2)} & I & & & \mathbf{S}_{37}^{(2)} & \\ & & I & \mathbf{S}_{44}^{(2)} & & & & \mathbf{S}_{48}^{(2)} \\ \mathbf{S}_{51}^{(2)} & & & & \mathbf{S}_{55}^{(2)} & & I & \\ & \mathbf{S}_{62}^{(2)} & & & & \mathbf{S}_{66}^{(2)} & & I \\ & & \mathbf{S}_{73}^{(2)} & & I & & \mathbf{S}_{77}^{(2)} & \\ & & & \mathbf{S}_{84}^{(2)} & & I & & \mathbf{S}_{88}^{(2)} \end{pmatrix}, \tag{2.1}$$

where the subscripts $1, 2, \ldots, 8$ correspond to the interfaces in Figure 2.1(a) and the following submatrices correspond to the four leaves of $\mathcal{T}$:

$$\begin{pmatrix} \mathbf{S}_{11}^{(2)} & \mathbf{S}_{15}^{(2)} \\ \mathbf{S}_{51}^{(2)} & \mathbf{S}_{55}^{(2)} \end{pmatrix}, \quad \begin{pmatrix} \mathbf{S}_{22}^{(2)} & \mathbf{S}_{26}^{(2)} \\ \mathbf{S}_{62}^{(2)} & \mathbf{S}_{66}^{(2)} \end{pmatrix}, \quad \begin{pmatrix} \mathbf{S}_{33}^{(2)} & \mathbf{S}_{37}^{(2)} \\ \mathbf{S}_{73}^{(2)} & \mathbf{S}_{77}^{(2)} \end{pmatrix}, \quad \begin{pmatrix} \mathbf{S}_{44}^{(2)} & \mathbf{S}_{48}^{(2)} \\ \mathbf{S}_{84}^{(2)} & \mathbf{S}_{88}^{(2)} \end{pmatrix}, \tag{2.2}$$

These submatrices are the results of the elimination of the interior mesh points of the four subdomains and the boundary of the entire domain. The identity blocks in (2.1) introduce coupling to shared interfaces. This is due to the transmission condition on shared boundaries. See [13, 31] for the derivation.

2.3. Sparse hierarchical block LDU factorization via Schur complement update

Once $\mathbf{S^{(1)}}$ is obtained, we can perform sparse hierarchical block LDU factorization and solution as in standard sparse direct solvers such as the multifrontal method. The factorization at a level $l = \mathbf{l}, \mathbf{l-1}, \ldots, 1$ can be written collectively as

$$\mathbf{S}^{(l)} = \begin{pmatrix} I & \\ \mathbf{L}^{(l)} & I \end{pmatrix} \begin{pmatrix} \mathbf{D}^{(l)} & \\ & \mathbf{S}^{(l-1)} \end{pmatrix} \begin{pmatrix} I & \mathbf{U}^{(l)} \\ & I \end{pmatrix}, \tag{2.3}$$

where $\mathbf{S}^{(l)}$ is the input corresponding to all the level-l nodes of $\mathcal{T}$, $\mathbf{L}^{(l)}$ and $\mathbf{U}^{(l)}$ are obtained from eliminating the pivot block $\mathbf{D}^{(l)}$ of $\mathbf{S}^{(l)}$, and $\mathbf{S}^{(l-1)}$ is the Schur complement to be factorized at level $l-1$. Here, the pivot block $\mathbf{D}^{(l)}$ is the submatrix of $\mathbf{S}^{(l)}$ corresponding to all pivot interfaces in the level-l neighbor lists of $\mathcal{T}$. With each level of factorization, the matrix size reduces and the tree $\mathcal{T}$ shrinks by one level. The process continues until level 1 of $\mathcal{T}$ is reached and the Schur complement $\mathbf{S}^{(1)}$ is factorized. For convenience, we refer to the computation of $\mathbf{S}^{(l-1)}$ from $\mathbf{S}^{(l)}$ via the sparse LDU factorization (2.3) as the *Schur complement update problem*, which is the main task for the sparse factorization.

After the sparse block LDU factorization, the solution can be computed by solving a sequence of intermediate linear systems. To solve an intermediate problem $\mathbf{S}^{(l)}x^{(l)} = b^{(l)}$ at level l, where $\mathbf{S}^{(l)}$ has the block

factorization (2.3), the following three solution steps are needed:

$$\begin{pmatrix} I & \\ \mathbf{L}^{(l)} & I \end{pmatrix} \begin{pmatrix} y \\ b^{(l-1)} \end{pmatrix} = b^{(l)}, \tag{2.4}$$

$$\begin{pmatrix} \mathbf{D}^{(l)} & \\ & \mathbf{S}^{(l-1)} \end{pmatrix} \begin{pmatrix} z \\ x^{(l-1)} \end{pmatrix} = \begin{pmatrix} y \\ b^{(l-1)} \end{pmatrix}, \tag{2.5}$$

$$\begin{pmatrix} I & \mathbf{U}^{(l)} \\ & I \end{pmatrix} x^{(l)} = \begin{pmatrix} z \\ x^{(l-1)} \end{pmatrix}. \tag{2.6}$$

This is a recursive process since the second step needs to solve $\mathbf{S}^{(l-1)}x^{(l-1)} = b^{(l-1)}$ at level $l-1$.

As an example, consider the factorization of the matrix (2.1). After performing the elimination on $\mathbf{S}^{(2)}$ as in (2.3) with the leaf level of $\mathcal{T}$ (Figure 2.2(a)) removed, the level-1 Schur complement is

$$\mathbf{S}^{(1)} = \begin{pmatrix} \begin{pmatrix} \mathbf{S}^{(1)}_{55} & \mathbf{S}^{(1)}_{56} \\ \mathbf{S}^{(1)}_{65} & \mathbf{S}^{(1)}_{66} \end{pmatrix} & I \\ I & \begin{pmatrix} \mathbf{S}^{(1)}_{77} & \mathbf{S}^{(1)}_{78} \\ \mathbf{S}^{(1)}_{87} & \mathbf{S}^{(1)}_{88} \end{pmatrix} \end{pmatrix}, \quad \text{with}$$

$$\begin{pmatrix} \mathbf{S}^{(1)}_{55} & \mathbf{S}^{(1)}_{56} \\ \mathbf{S}^{(1)}_{65} & \mathbf{S}^{(1)}_{66} \end{pmatrix} = \begin{pmatrix} \mathbf{S}^{(2)}_{55} & \\ & \mathbf{S}^{(2)}_{66} \end{pmatrix} - \begin{pmatrix} \mathbf{S}^{(2)}_{51} & \\ & \mathbf{S}^{(2)}_{62} \end{pmatrix} \begin{pmatrix} \mathbf{S}^{(2)}_{11} & I \\ I & \mathbf{S}^{(2)}_{22} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{S}^{(2)}_{15} & \\ & \mathbf{S}^{(2)}_{26} \end{pmatrix}, \tag{2.7}$$

$$\begin{pmatrix} \mathbf{S}^{(1)}_{77} & \mathbf{S}^{(1)}_{78} \\ \mathbf{S}^{(1)}_{87} & \mathbf{S}^{(1)}_{88} \end{pmatrix} = \begin{pmatrix} \mathbf{S}^{(2)}_{77} & \\ & \mathbf{S}^{(2)}_{88} \end{pmatrix} - \begin{pmatrix} \mathbf{S}^{(2)}_{73} & \\ & \mathbf{S}^{(2)}_{84} \end{pmatrix} \begin{pmatrix} \mathbf{S}^{(2)}_{33} & I \\ I & \mathbf{S}^{(2)}_{44} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{S}^{(2)}_{37} & \\ & \mathbf{S}^{(2)}_{48} \end{pmatrix}. \tag{2.8}$$

The representations (2.7)–(2.8) have a clear geometric interpretation. Take (2.7) as an example. $\mathbf{S}^{(2)}_{55}$ and $\mathbf{S}^{(2)}_{66}$ arise from two disjoint interfaces 5 and 6, but $\begin{pmatrix} \mathbf{S}^{(1)}_{55} & \mathbf{S}^{(1)}_{56} \\ \mathbf{S}^{(1)}_{65} & \mathbf{S}^{(1)}_{66} \end{pmatrix}$ joins the two interfaces into one pivot interface $(\mathbf{5}, \mathbf{6})$ of the separator $(5, 6, 7, 8)$, and the additional information is due to the update term contributed by the shared separator $(1, 2)$ via the coupling term $\begin{pmatrix} \mathbf{S}^{(2)}_{11} & I \\ I & \mathbf{S}^{(2)}_{22} \end{pmatrix}^{-1}$. In another word, the elimination of the separator $(1, 2)$ mutually connects the points in $(\mathbf{5}, \mathbf{6})$.

In general, in our direct elliptic solver, the Schur complement update problem is processed along the levelwise bottom-up traversal of the neighbor tree $\mathcal{T}$. Each $\mathbf{S}^{(l)}$ is processed in terms of the factorization of a sequence of local submatrices corresponding to the neighbor lists at level l of $\mathcal{T}$. For convenience, we call each such submatrix a level-l *local Schur complement* (associated with a corresponding neighbor list). For example, for $\mathbf{S}^{(2)}$ in (2.1) with the neighbor tree $\mathcal{T}$ in Figure 2.2(a), the four submatrices in (2.2) are level-2 local Schur complements corresponding to the four leaf nodes of $\mathcal{T}$. The two submatrices of $\mathbf{S}^{(1)}$ in (2.7) and (2.8) are level-1 local Schur complements corresponding to the two level-1 nodes of $\mathcal{T}$. In the next section, we show how to accelerate the Schur complement update by applying structured methods to local Schur complements.

3. Interconnected hierarchical structured factorization

We then consider the incorporation of rank structured methods into the basic factorization framework in the previous section. Directly performing the Schur complement update (2.3) is expensive for large sizes. The cost can be reduced by converting local Schur complements into data-sparse forms, and the feasibility follows from [2, 5]. In this section, we design IHSS structures and construct IHSS approximations to the local Schur complements. The IHSS approximations at different levels of the sparse factorization are closely related via shared structures. We introduce IHSS approximations and factorizations based on interconnected tree structures.

3.1. Interconnected tree structures

Our sparse hierarchical factorization (2.3) follows the levelwise traversal of the neighbor tree $\mathcal{T}$. For each node (neighbor list) of $\mathcal{T}$, we construct an IHSS approximation to the corresponding local Schur complement. (The details are in Section 3.2.) Each IHSS approximation also corresponds to a tree called *IHSS tree*. Thus, we have two layers of trees: the neighbor tree $\mathcal{T}$ (as the *outer tree*) for organizing the sparse factorization, and an inner IHSS tree for the structured approximation of a local Schur complement. This two-layer tree structure is similar to structured multifrontal methods in [1, 40, 41, 42].

However, there are some major differences between our new solver and those existing solvers. One difference is that our outer tree is a neighbor tree instead of the assembly tree. More importantly, the methods in [1, 40, 41, 42] study the two layers of trees separately, which may result in repetitive constructions of inner structures. In our new solver, the inner trees at different levels of the outer tree are interconnected such that subtrees of the inner trees are preserved and reused as much as possible when we traverse the outer tree. That is, our solver uses *interconnected tree structures*, which facilitate optimized data assembly and factorization. This will become clear in our remaining discussions.

A key mechanism for the interconnected tree structures is the use of neighbor lists.

1. A parent node (neighbor list) of $\mathcal{T}$ includes interfaces resulting from child level eliminations. The pivot interfaces in two child neighbor lists are eliminated, and the remaining interfaces from the child neighbor lists are collected to form the parent neighbor list. For example, in Figure 2.2(a), the elimination of the pivot interface $\mathbf{1}$ from $\{\mathbf{1}, 5\}$ and $\mathbf{2}$ from $\{\mathbf{2}, 6\}$ yields the parent node $\{(\mathbf{5}, \mathbf{6})\}$.
2. The interfaces within each node of $\mathcal{T}$ are used for the partitioning of the associated local Schur complement, and such a partitioning is used in the construction of an IHSS approximation to the local Schur complement. For example, the four local Schur complements in (2.2) are partitioned precisely based on the interfaces inside the four leaf nodes in Figure 2.2(a).

These strategies ensure that structures passed from a lower level to the parent level are fully preserved in the IHSS approximation. This is elaborated in the next two subsections.

3.2. IHSS representation

Suppose a node (neighbor list) at level l of $\mathcal{T}$ is $\alpha = \{c_1, c_2, \ldots, c_s\}$, where we assume all common-origin interfaces are already grouped into each c_j. Thus, the interfaces come from s different origins. Then the corresponding local Schur complement $S_\alpha^{(l)}$ within $\mathbf{S}^{(l)}$ can be represented as

$$S_\alpha^{(l)} = \begin{pmatrix} \mathbf{S}_{c_1,c_1}^{(l)} & \mathbf{S}_{c_1,c_2}^{(l)} & \cdots & \mathbf{S}_{c_1,c_s}^{(l)} \\ \mathbf{S}_{c_2,c_1}^{(l)} & \mathbf{S}_{c_2,c_2}^{(l)} & \cdots & \mathbf{S}_{c_2,c_s}^{(l)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_{c_s,c_1}^{(l)} & \mathbf{S}_{c_s,c_2}^{(l)} & \cdots & \mathbf{S}_{c_s,c_s}^{(l)} \end{pmatrix}. \tag{3.1}$$

We introduce an IHSS approximation to $S_\alpha^{(l)}$ via the following three key features.

1. *Low-rank off-diagonal forms.*
   The off-diagonal blocks of $S_\alpha^{(l)}$ can be approximated by low-rank forms. Following the partitioning in (3.1), the IHSS approximation to $S_\alpha^{(l)}$ looks like

$$S_\alpha^{(l)} \approx \begin{pmatrix} D_{c_1} & U_{c_1} B_{c_1,c_2} V_{c_2}^T & \cdots & U_{c_1} B_{c_1,c_s} V_{c_s}^T \\ U_{c_2} B_{c_2,c_1} V_{c_1}^T & D_{c_2} & \cdots & U_{c_2} B_{c_2,c_s} V_{c_s}^T \\ \vdots & \vdots & \ddots & \vdots \\ U_{c_s} B_{c_s,c_1} V_{c_1}^T & U_{c_s} B_{c_s,c_2} V_{c_2}^T & \cdots & D_{c_s} \end{pmatrix}, \tag{3.2}$$

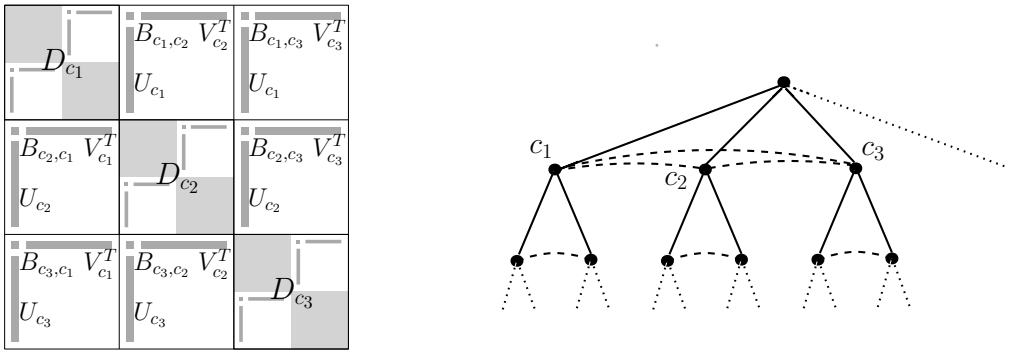
   where

   - each D_{c_j} is a diagonal block representing the self-interaction within the interface c_j;
   - each U_{c_j} (V_{c_j}) is the column (row) basis matrix representing the contribution of the interface c_j to the interaction between c_j and the other $s-1$ interfaces in the neighbor list;

- each B_{c_j,c_k} describes the mutual interaction between a pair of interfaces c_j, c_k (ignoring the U_{c_j}, V_{c_k} basis matrices).

2. *Hierarchical structures.*
The structured approximation further appears in a hierarchical or nested form. Each c_i in α may be formed by grouping smaller common-origin interfaces pairwise and hierarchically. For example, c_i may be formed by two common-origin interfaces z_1 and z_2, each of which may in turn be formed by two smaller common-origin interfaces. Thus, each c_i corresponds to a binary subtree T_{c_i} and the children of c_i are z_1 and z_2, which may also have children. The corresponding structured representation for D_{c_i} has a hierarchical form and is essentially a standard HSS form as in [6, 43]. The nested structures look like
$$D_{c_i} = \begin{pmatrix} D_{z_1} & U_{z_1} B_{z_1,z_2} V_{z_2}^T \\ U_{z_2} B_{z_2,z_1} V_{z_1}^T & D_{z_2} \end{pmatrix}, \quad U_{c_i} = \begin{pmatrix} U_{z_1} R_{z_1} \\ U_{z_2} R_{z_2} \end{pmatrix}, \quad V_{c_i} = \begin{pmatrix} V_{z_1} W_{z_1} \\ V_{z_2} W_{z_2} \end{pmatrix}, \tag{3.3}$$
where the R, W matrices are used to translate the basis matrices from child nodes to their parent. Thus, off-diagonal blocks at different hierarchical levels of T_{c_i} are low rank. T_{c_i} is also said to be an *HSS subtree*. All the HSS subtrees $T_{c_i}, i = 1, 2, \ldots, s$ are organized together under a root node which is the neighbor list α. This leads to a hierarchical tree T called *IHSS tree*. Figure 3.1 shows an example.

3. *Interconnected structures via shared off-diagonal bases.*
The off-diagonal basis matrices U, V are preserved across different levels l of the outer tree $\mathcal{T}$. In the factorization (2.3), some of the off-diagonal basis matrices U, V of $S_\alpha^{(l)}$ are passed to level $l-1$ and are directly used to construct the IHSS approximations to the local Schur complements at level $l-1$. This will be explained in detail in Section 3.3.



(a) An IHSS matrix

(c) The corresponding IHSS tree

Figure 3.1: *Illustration of an IHSS matrix and the corresponding IHSS tree defined based on the interfaces c_1, c_2, c_3. The shaded boxes visualize the shapes of the D, B, U, V generators. The interfaces may consist of smaller lower level interfaces.*

The matrices D, B, U, V, R, W are called *generators* which are associated with the nodes of the IHSS tree T, as indicated by the subscripts of the generators. Due to the nested relation (3.3), only D, U, V generators associated with leaf nodes need to be stored. The U, V generators have small column sizes so that the representation is data sparse. If the IHSS tree is a binary tree, then the IHSS form is simply an HSS form and the generators are standard HSS generators [43].

Remark 3.1. To simplify notation, we sometimes use $B_{c_i,*}$ to denote the B generator corresponding to an interface c_i and another unspecified interface. If an interface z_1 has only one sibling z_2 in the tree T, then we may use $\mathrm{sib}(z_1)$ to represent z_2 and write B_{z_1,z_2} as $B_{z_1,\mathrm{sib}(z_1)}$.

3.3. Schur complement update via IHSS factorizations

We then use IHSS factorizations to perform the Schur complement update in (2.3). IHSS approximations like in (3.2)–(3.3) can be applied individually to each local Schur complement within $\mathbf{S}^{(l)}$ based on the associated neighbor list. IHSS approximations fully respect the interconnected tree structures by exploiting

the connections of the neighbor lists at different levels of $\mathcal{T}$. That is, the U, V basis matrices of the IHSS forms used in $\mathbf{S}^{(l)}$ are preserved and reused in the computation of $\mathbf{S}^{(l-1)}$. The details are as follows.

Starting from a certain switching level $\mathbf{l}_s$ of $\mathcal{T}$ (the reason why the switching level is used will be explained in Section 3.4), an IHSS approximation is computed directly for each local Schur complement within $\mathbf{S}^{(\mathbf{l})}$. This may be based on direct off-diagonal block compression as in [43] or randomized construction like in [28, 44]. With the number of bisection levels $\mathbf{l}$ large enough, these local Schur complements have small sizes and the costs for the direct IHSS construction are low. Also, such direct IHSS construction is done only once at level $\mathbf{l}_s$ and the off-diagonal basis matrices will be reused at upper levels.

Then for $l = \mathbf{l}_s, \mathbf{l}_s - 1, \ldots, 1$, we perform the factorization of $\mathbf{S}^{(l)}$ in (2.3). We compute a structured approximation to $\mathbf{S}^{(l-1)}$ by constructing an IHSS approximation to each local Schur complement within $\mathbf{S}^{(l-1)}$. The interconnected tree structures are used to quickly get the IHSS approximations and to avoid expensive direct construction. Specifically, suppose two sibling nodes at level l of $\mathcal{T}$ are associated with the following two neighbor lists, respectively:

$$\alpha = \{\boldsymbol{\sigma}, c_1, c_2, \ldots, c_s\}, \quad \beta = \{\boldsymbol{\delta}, d_1, d_2, \ldots, d_t\}, \tag{3.4}$$

where $\boldsymbol{\sigma}$ and $\boldsymbol{\delta}$ are pivot interfaces to be eliminated, and we assume all common-origin interfaces have already been grouped into each $\boldsymbol{\sigma}$, c_i, $\boldsymbol{\delta}$, or d_j. (The grouping will be discussed at the end of Section 3.3.2 as part of the process for forming hierarchical representations.) Examples of two such interface lists in Figures 2.1(b) and 2.2(b) are as follows:

$$\underset{\boldsymbol{\sigma}}{\{(\mathbf{21},\mathbf{22})},\ \underset{c_1}{35\}}, \quad \underset{\boldsymbol{\delta}}{\{(\mathbf{23},\mathbf{24})},\ \underset{d_1}{36},\ \underset{d_2}{(43,44)\}}.$$

Suppose the two neighbor lists α and β correspond to two local Schur complements $S_\alpha^{(l)}$ and $S_\beta^{(l)}$, respectively. Also, let the parent node of α and β in (3.4) be γ, which corresponds to a neighbor list formed by $c_1, \ldots, c_s, d_1, \ldots, d_t$. In the factorization (2.3), we eliminate the pivot blocks of $S_\alpha^{(l)}$ and $S_\beta^{(l)}$ and form the local Schur complement $S_\gamma^{(l-1)}$ corresponding to γ at level $l-1$.

We first rewrite the local Schur complement updates like (2.7) or (2.8) in a general form (see [31, equations (2.9)–(2.14)] for the detailed derivation). That is, for $i, j = 1, 2, \ldots, \min\{s, t\}$,

$$\begin{pmatrix} \mathbf{S}_{c_i,c_j}^{(l-1)} & \mathbf{S}_{c_i,d_j}^{(l-1)} \\ \mathbf{S}_{d_i,c_j}^{(l-1)} & \mathbf{S}_{d_i,d_j}^{(l-1)} \end{pmatrix} = \begin{pmatrix} \mathbf{S}_{c_i,c_j}^{(l)} & \\ & \mathbf{S}_{d_i,d_j}^{(l)} \end{pmatrix} - \begin{pmatrix} \mathbf{S}_{c_i,\boldsymbol{\sigma}}^{(l)} & \\ & \mathbf{S}_{d_j,\boldsymbol{\delta}}^{(l)} \end{pmatrix} \begin{pmatrix} \mathbf{S}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}^{(l)} & I \\ I & \mathbf{S}_{\boldsymbol{\delta},\boldsymbol{\delta}}^{(l)} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{S}_{\boldsymbol{\sigma},c_i}^{(l)} & \\ & \mathbf{S}_{\boldsymbol{\delta},d_j}^{(l)} \end{pmatrix}. \tag{3.5}$$

Let $\begin{pmatrix} G_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & G_{\boldsymbol{\sigma},\boldsymbol{\delta}} \\ G_{\boldsymbol{\delta},\boldsymbol{\sigma}} & G_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix} \equiv \begin{pmatrix} \mathbf{S}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}^{(l)} & I \\ I & \mathbf{S}_{\boldsymbol{\delta},\boldsymbol{\delta}}^{(l)} \end{pmatrix}^{-1}$. Then (3.5) can be written into the following individual equations:

$$\mathbf{S}_{c_i,c_j}^{(l-1)} = \mathbf{S}_{c_i,c_j}^{(l)} - \mathbf{S}_{c_i,\boldsymbol{\sigma}}^{(l)} G_{\boldsymbol{\sigma},\boldsymbol{\sigma}} \mathbf{S}_{\boldsymbol{\sigma},c_j}^{(l)}, \quad i, j \le s, \tag{3.6}$$

$$\mathbf{S}_{d_i,d_j}^{(l-1)} = \mathbf{S}_{d_i,d_j}^{(l)} - \mathbf{S}_{d_i,\boldsymbol{\delta}}^{(l)} G_{\boldsymbol{\delta},\boldsymbol{\delta}} \mathbf{S}_{\boldsymbol{\delta},d_j}^{(l)}, \quad i, j \le t, \tag{3.7}$$

$$\mathbf{S}_{c_i,d_j}^{(l-1)} = -\mathbf{S}_{c_i,\boldsymbol{\sigma}}^{(l)} G_{\boldsymbol{\sigma},\boldsymbol{\delta}} \mathbf{S}_{\boldsymbol{\delta},d_j}^{(l)}, \quad i \le s, \quad j \le t, \tag{3.8}$$

$$\mathbf{S}_{d_j,c_i}^{(l-1)} = -\mathbf{S}_{d_j,\boldsymbol{\delta}}^{(l)} G_{\boldsymbol{\delta},\boldsymbol{\sigma}} \mathbf{S}_{\boldsymbol{\sigma},c_i}^{(l)}, \quad i \le s, \quad j \le t. \tag{3.9}$$

Note that (3.5) requires $i, j \le \min\{s, t\}$, but (3.6)–(3.9) apply to all $1 \le i \le s, 1 \le j \le t$. For convenience, we will use (3.5) for some intuitive derivations and the results can be easily adapted to (3.6)–(3.9).

The blocks in (3.6)–(3.9) together form the local Schur complement $S_\gamma^{(l-1)}$. Each block represents the computation of the interaction between a pair of nodes at level $l-1$ of $\mathcal{T}$ based on the interactions at level l.

- (3.6) shows how $\mathbf{S}_{c_i,c_j}^{(l)}$ is updated to produce $\mathbf{S}_{c_i,c_j}^{(l-1)}$ due to the elimination of $\boldsymbol{\sigma}$. (3.7) can be similarly understood.
- (3.8) shows how new fill-in $\mathbf{S}_{c_i,d_j}^{(l-1)}$ is created due to the elimination of $\boldsymbol{\sigma}, \boldsymbol{\delta}$. (3.9) can be similarly understood.

We then derive the structured version of (3.5) and (3.6)–(3.9). This involves two steps: structured factorization of the pivot blocks and structured formation of $S_\gamma^{(l-1)}$.

3.3.1. Structured factorization of the pivot matrix

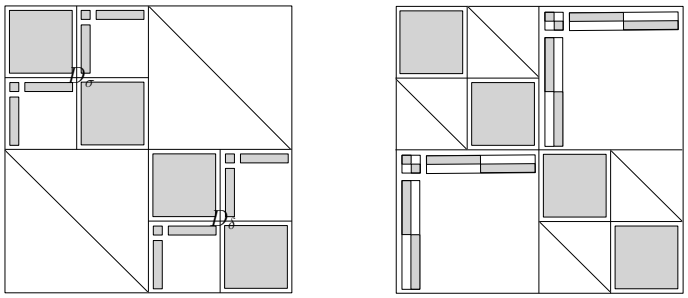
In this step, we compute a structured LDU factorization of the pivot matrix $\begin{pmatrix} \mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & I \\ I & \mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix}$ in (3.5). In the IHSS approximations to $S^{(l)}_\alpha$ and $S^{(l)}_\beta$, the blocks $\mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}$ and $\mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}}$ are approximated by standard HSS forms. We suppose the HSS approximations to $\mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}$ and $\mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}}$ have generators

$$\{D_{\sigma_j}, B_{\sigma_j,\mathrm{sib}(\sigma_j)}, U_{\sigma_j}, V_{\sigma_j}, R_{\sigma_j}, W_{\sigma_j}\} \quad \text{and} \quad \{D_{\delta_j}, B_{\delta_j,\mathrm{sib}(\delta_j)}, U_{\delta_j}, V_{\delta_j}, R_{\delta_j}, W_{\delta_j}\}, \tag{3.10}$$

respectively, where $\sigma_j, j = 1, 2, \ldots$ and $\delta_j, j = 1, 2, \ldots$ are the smaller interfaces that form $\boldsymbol{\sigma}$ and $\boldsymbol{\delta}$, respectively, and are from the leaf level of $\mathcal{T}$. (We omitted the superscript l in the D, B generators in this subsection. Also see Remark 3.1 about the subscripts of the B generators.) The pivot matrix $\begin{pmatrix} \mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & I \\ I & \mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix}$ can be permuted into an HSS form, and the permutation is defined by rearranging $\{\sigma_1, \sigma_2, \ldots, \delta_1, \delta_2, \ldots\}$ into $\{\sigma_1, \delta_1, \sigma_2, \delta_2, \ldots\}$. Suppose Π is the permutation matrix and the permuted matrix is

$$H = \Pi \begin{pmatrix} \mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & I \\ I & \mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix} \Pi^T.$$

See Figure 3.2 for an example.



(a) $\begin{pmatrix} \mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & I \\ I & \mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix}$ (b) Matrix H after permutation

Figure 3.2: *Illustration of a pivot matrix and its permuted form H.*

By combining the generators of $\mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}$ and $\mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}}$ in (3.10), we get the generators of the HSS approximation to H as follows:

$$\begin{pmatrix} D_{\sigma_j} & I \\ I & D_{\delta_j} \end{pmatrix}, \begin{pmatrix} B_{\sigma_j,\mathrm{sib}(\sigma_j)} & 0 \\ 0 & B_{\delta_j,\mathrm{sib}(\delta_j)} \end{pmatrix}, \begin{pmatrix} U_{\sigma_j} & 0 \\ 0 & U_{\delta_j} \end{pmatrix}, \begin{pmatrix} V_{\sigma_j} & 0 \\ 0 & V_{\delta_j} \end{pmatrix}, \begin{pmatrix} R_{\sigma_j} & 0 \\ 0 & R_{\delta_j} \end{pmatrix}, \begin{pmatrix} W_{\sigma_j} & 0 \\ 0 & W_{\delta_j} \end{pmatrix}. \tag{3.11}$$

(Later, we will abuse notation and still use H to mean its HSS approximation.) Note that only the D generators are coupled. We can design a (nonsymmetric) HSS LDU factorization by modifying a (symmetric) HSS LDL factorization in [38]. However, since the U, V, R, W generators in (3.11) have block diagonal or decoupled forms, we can design the HSS LDU factorization to preserve these decoupled forms. We also use this factorization to produce results useful for the fast computation of (3.6)–(3.9). We traverse the HSS tree of H in a bottom-up order. (It is essentially to simultaneously traverse the HSS trees of $\mathbf{S}^{(l)}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}$ and $\mathbf{S}^{(l)}_{\boldsymbol{\delta},\boldsymbol{\delta}}$.)

If σ_j and δ_j are leaf nodes, we introduce zeros into the corresponding basis matrices using QL factorizations

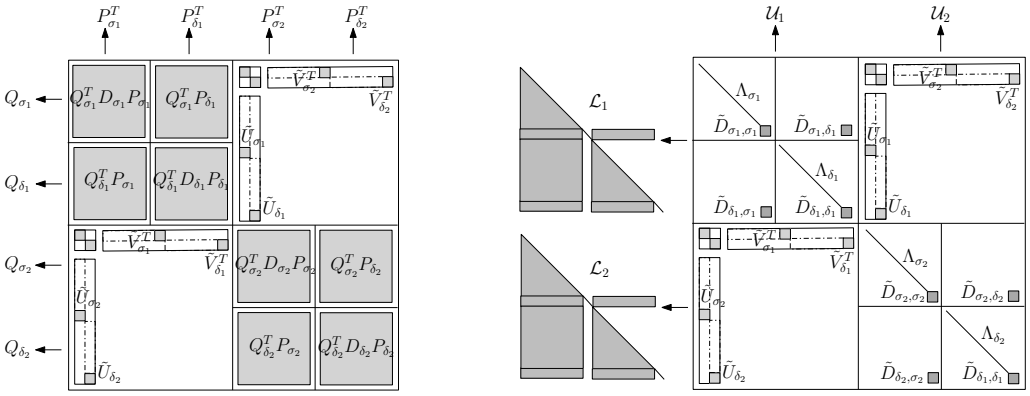
$$U_{\sigma_j} = Q_{\sigma_j} \begin{pmatrix} 0 \\ \tilde{U}_{\sigma_j} \end{pmatrix}, \quad V_{\sigma_j} = P_{\sigma_j} \begin{pmatrix} 0 \\ \tilde{V}_{\sigma_j} \end{pmatrix}, \quad U_{\delta_j} = Q_{\delta_j} \begin{pmatrix} 0 \\ \tilde{U}_{\delta_j} \end{pmatrix}, \quad V_{\delta_j} = P_{\delta_j} \begin{pmatrix} 0 \\ \tilde{V}_{\delta_j} \end{pmatrix}. \tag{3.12}$$

Then apply $Q^T_{\sigma_j}$ on the left to the block row of H corresponding to σ_j and apply P_{σ_j} on the right to the block column of H corresponding to σ_j. Similarly, apply $Q^T_{\delta_j}$ and P_{δ_j}. These introduce zero blocks into the

corresponding off-diagonal blocks as in standard HSS ULV factorizations [6]. The corresponding diagonal blocks are transformed to

$$\begin{pmatrix} \bar{D}_{\sigma_j,\sigma_j} & \bar{D}_{\sigma_j,\delta_j} \\ \bar{D}_{\delta_j,\sigma_j} & \bar{D}_{\delta_j,\delta_j} \end{pmatrix} \equiv \begin{pmatrix} Q_{\sigma_j}^T D_{\sigma_j} P_{\sigma_j} & Q_{\sigma_j}^T P_{\delta_j} \\ Q_{\delta_j}^T P_{\sigma_j} & Q_{\delta_j}^T D_{\delta_j} P_{\delta_j} \end{pmatrix}. \tag{3.13}$$

Accordingly, suppose H is transformed into $\bar{H}$. See Figure 3.3(a) for an illustration.



(a) Introducing zeros into off-diagonal blocks (b) Partial diagonal elimination

Figure 3.3: *Illustration of some major steps of the HSS LDU factorization.*

Then partition the blocks of (3.13) conformably following (3.12) and compute the following partial LDU factorization:

$$\begin{pmatrix} \bar{D}_{\sigma_j,\sigma_j} & \bar{D}_{\sigma_j,\delta_j} \\ \bar{D}_{\delta_j,\sigma_j} & \bar{D}_{\delta_j,\delta_j} \end{pmatrix} = \mathcal{L}_j \mathcal{D}_j \mathcal{U}_j, \quad \text{with} \quad \mathcal{D}_j = \begin{pmatrix} \Lambda_{\sigma_j} & & \\ & \tilde{D}_{\sigma_j,\sigma_j} & \tilde{D}_{\sigma_j,\delta_j} \\ & \tilde{D}_{\delta_j,\sigma_j} & \tilde{D}_{\delta_j,\delta_j} \end{pmatrix},$$

$$\mathcal{L}_j = \begin{pmatrix} \mathcal{L}_{\sigma_j,\sigma_j} & & & \\ \tilde{\mathcal{L}}_{\sigma_j,\sigma_j} & I & \tilde{\mathcal{L}}_{\sigma_j,\delta_j} & \\ \mathcal{L}_{\delta_j,\sigma_j} & & \mathcal{L}_{\delta_j,\delta_j} & \\ \tilde{\mathcal{L}}_{\delta_j,\sigma_j} & & \tilde{\mathcal{L}}_{\delta_j,\delta_j} & I \end{pmatrix}, \quad \mathcal{U}_j = \begin{pmatrix} \mathcal{U}_{\sigma_j,\sigma_j} & \tilde{\mathcal{U}}_{\sigma_j,\sigma_j} & \mathcal{U}_{\sigma_j,\delta_j} & \tilde{\mathcal{U}}_{\sigma_j,\delta_j} \\ & I & & \\ & \tilde{\mathcal{U}}_{\delta_j,\sigma_j} & \mathcal{U}_{\delta_j,\delta_j} & \tilde{\mathcal{U}}_{\delta_j,\delta_j} \\ & & & I \end{pmatrix},$$

where Λ_{σ_j} and Λ_{δ_j} are diagonal matrices and match the zero positions in (3.12). The purpose for such a factorization is partial elimination. That is, we apply $\mathcal{L}_j^{-1}$ on the left to the block row of $\bar{H}$ corresponding to (3.13) and apply $\mathcal{U}_j^{-1}$ on the right to the block column of $\bar{H}$ corresponding to (3.13), and suppose the resulting matrix is $\tilde{H}$. Then Λ_{σ_j} and Λ_{δ_j} can be eliminated from $\tilde{H}$. In addition, this partial elimination does not impact the modified basis matrices $\tilde{U}_{\sigma_j}, \tilde{V}_{\sigma_j}, \tilde{U}_{\delta_j}, \tilde{V}_{\delta_j}$ previously computed in (3.12). See Figure 3.3(b).

If σ_i and δ_i are non-leaf nodes, suppose they have children μ_1, μ_2 and ν_1, ν_2, respectively. Partial eliminations have been performed when these child nodes are visited. The remaining blocks that are not eliminated at the child level can be merged to form some new HSS generators. Let

$$\hat{D}_{\sigma_j} = \begin{pmatrix} \tilde{D}_{\mu_1,\mu_1} & \tilde{U}_{\mu_1} B_{\mu_1,\mu_2} \tilde{V}_{\mu_2}^T \\ \tilde{U}_{\mu_2} B_{\mu_2,\mu_1} \tilde{V}_{\mu_1}^T & \tilde{D}_{\mu_2,\mu_2} \end{pmatrix}, \quad \hat{U}_{\sigma_j} = \begin{pmatrix} \tilde{U}_{\mu_1} R_{\mu_1} \\ \tilde{U}_{\mu_2} R_{\mu_2} \end{pmatrix}, \quad \hat{V}_{\sigma_j} = \begin{pmatrix} \tilde{V}_{\mu_1} W_{\mu_1} \\ \tilde{V}_{\mu_2} W_{\mu_2} \end{pmatrix},$$

$$\hat{D}_{\delta_j} = \begin{pmatrix} \tilde{D}_{\nu_1,\nu_1} & \tilde{U}_{\nu_1} B_{\nu_1,\nu_2} \tilde{V}_{\nu_2}^T \\ \tilde{U}_{\nu_2} B_{\nu_2,\nu_1} \tilde{V}_{\nu_1}^T & \tilde{D}_{\nu_2,\nu_2} \end{pmatrix}, \quad \hat{U}_{\delta_j} = \begin{pmatrix} \tilde{U}_{\nu_1} R_{\nu_1} \\ \tilde{U}_{\nu_2} R_{\nu_2} \end{pmatrix}, \quad \hat{V}_{\delta_j} = \begin{pmatrix} \tilde{V}_{\nu_1} W_{\nu_1} \\ \tilde{V}_{\nu_2} W_{\nu_2} \end{pmatrix},$$

Associate σ_j with HSS generators $\hat{D}_{\sigma_j}, B_{\sigma_j,\mathrm{sib}(\sigma_j)}, \hat{U}_{\sigma_j}, \hat{V}_{\sigma_j}, R_{\sigma_j}, W_{\sigma_j}$ and associate δ_j with HSS generators $\hat{D}_{\delta_j}, B_{\delta_j,\mathrm{sib}(\delta_j)}, \hat{U}_{\delta_j}, \hat{V}_{\delta_j}, R_{\delta_j}, W_{\delta_j}$. Then we can remove the children of σ_j and δ_j from their associated HSS

trees respectively. Accordingly, their associated HSS forms can be merged and permuted into a larger HSS form with generators similar to (3.11) but with small differences:

$$\begin{pmatrix} \hat{D}_{\sigma_j} & \operatorname{diag}(\tilde{D}_{\mu_1,\nu_1}, \tilde{D}_{\mu_2,\nu_2}) \\ \operatorname{diag}(\tilde{D}_{\nu_1,\mu_1}, \tilde{D}_{\nu_2,\mu_2}) & \hat{D}_{\delta_j} \end{pmatrix}, \begin{pmatrix} B_{\sigma_j,\mathrm{sib}(\sigma_j)} & 0 \\ 0 & B_{\delta_j,\mathrm{sib}(\delta_j)} \end{pmatrix},$$
$$\begin{pmatrix} \hat{U}_{\sigma_j} & 0 \\ 0 & \hat{U}_{\delta_j} \end{pmatrix}, \begin{pmatrix} \hat{V}_{\sigma_j} & 0 \\ 0 & \hat{V}_{\delta_j} \end{pmatrix}, \begin{pmatrix} R_{\sigma_j} & 0 \\ 0 & R_{\delta_j} \end{pmatrix}, \begin{pmatrix} W_{\sigma_j} & 0 \\ 0 & W_{\delta_j} \end{pmatrix}.$$

where diag() is used to denote a block diagonal matrix. Then the elimination process above can be similarly applied to σ_j and δ_j.

This process is repeated in a bottom-up sweep. If σ_j and δ_j are root nodes, after the factorizations, we also store the following matrix:

$$\tilde{G} = \begin{pmatrix} \tilde{V}_{\sigma_j}^T & \\ & \tilde{V}_{\delta_j}^T \end{pmatrix} \begin{pmatrix} \tilde{D}_{\sigma_j,\sigma_j} & \tilde{D}_{\sigma_j,\delta_j} \\ \tilde{D}_{\delta_j,\sigma_j} & \tilde{D}_{\delta_j,\delta_j} \end{pmatrix}^{-1} \begin{pmatrix} \tilde{U}_{\sigma_j} & \\ & \tilde{U}_{\delta_j} \end{pmatrix}. \tag{3.14}$$

$\tilde{G}$ is a small matrix and will be used a little later in (3.18) to facilitate the fast computation of (3.6)–(3.9).

3.3.2. Structured formation of $S_\gamma^{(l-1)}$

With the pivot separators eliminated from $S_\alpha^{(l)}$ and $S_\beta^{(l)}$, we then show how to find an IHSS approximation to the local Schur complement $S_\gamma^{(l-1)}$ based on (3.5) or (3.6)–(3.9). Suppose the IHSS generators of $S_\alpha^{(l)}$ and $S_\beta^{(l)}$ associated with c_j and d_j in (3.4) are

$$\{D_{c_j}^{(l)}, B_{c_j,*}^{(l)}, U_{c_j}, V_{c_j}, R_{c_j}, W_{c_j}\} \quad \text{and} \quad \{D_{d_j}^{(l)}, B_{d_j,*}^{(l)}, U_{d_j}, V_{d_j}, R_{d_j}, W_{d_j}\}, \tag{3.15}$$

respectively, where the superscript l in the D, B generators is used to indicate the level dependence. The U, V, R, W generators are related to the off-diagonal bases and will be reused across different levels l in our interconnected structured method.

With the generators in (3.15), the second term on the right-hand side of (3.5) can be approximated by a structured form

$$\begin{aligned} &\begin{pmatrix} U_{c_i} B_{c_i,\boldsymbol{\sigma}}^{(l)} V_{\boldsymbol{\sigma}}^T & \\ & U_{d_j} B_{d_j,\boldsymbol{\delta}}^{(l)} V_{\boldsymbol{\delta}}^T \end{pmatrix} \begin{pmatrix} \mathbf{S}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}^{(l)} & I \\ I & \mathbf{S}_{\boldsymbol{\delta},\boldsymbol{\delta}}^{(l)} \end{pmatrix}^{-1} \begin{pmatrix} U_{\boldsymbol{\sigma}} B_{\boldsymbol{\sigma},c_i}^{(l)} V_{c_i}^T & \\ & U_{\boldsymbol{\delta}} B_{\boldsymbol{\delta},d_j}^{(l)} V_{d_j}^T \end{pmatrix} \\ &= \begin{pmatrix} U_{c_i} & \\ & U_{d_j} \end{pmatrix} \begin{pmatrix} B_{c_i,\boldsymbol{\sigma}}^{(l)} & \\ & B_{d_j,\boldsymbol{\delta}}^{(l)} \end{pmatrix} \begin{pmatrix} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\delta}} \\ \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix} \begin{pmatrix} B_{\boldsymbol{\sigma},c_i}^{(l)} & \\ & B_{\boldsymbol{\delta},d_j}^{(l)} \end{pmatrix} \begin{pmatrix} V_{c_i}^T & \\ & V_{d_j}^T \end{pmatrix}, \end{aligned} \tag{3.16}$$

where

$$\begin{pmatrix} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\delta}} \\ \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix} = \begin{pmatrix} V_{\boldsymbol{\sigma}}^T & \\ & V_{\boldsymbol{\delta}}^T \end{pmatrix} \begin{pmatrix} \mathbf{S}_{\boldsymbol{\sigma},\boldsymbol{\sigma}}^{(l)} & I \\ I & \mathbf{S}_{\boldsymbol{\delta},\boldsymbol{\delta}}^{(l)} \end{pmatrix}^{-1} \begin{pmatrix} U_{\boldsymbol{\sigma}} & \\ & U_{\boldsymbol{\delta}} \end{pmatrix}. \tag{3.17}$$

Note that there is no need to use extra computations to directly form (3.17). Instead, following the HSS LDU factorization in Section 3.3.1, an idea of reduced matrices in [41, Theorem 3.1] and [40, Theorem 3.2] can be used to show that (3.17) is actually given by $\tilde{G}$ in (3.14):

$$\begin{pmatrix} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\delta}} \\ \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\sigma}} & \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} \end{pmatrix} = \tilde{G}. \tag{3.18}$$

This avoids the use of HSS solutions and multiplications to form (3.17).

We then discuss how to quickly obtain the structured form of the local Schur complement $S_\gamma^{(l-1)}$ in $\mathbf{S}^{(l-1)}$. There are three cases to consider.

I. For the diagonal blocks $\mathbf{S}^{(l-1)}_{c_i,c_i}$ and $\mathbf{S}^{(l-1)}_{d_j,d_j}$ corresponding to interactions within the interfaces c_i and d_j, respectively, by substituting (3.15) into (3.6) and (3.7), we get

$$\begin{aligned}\mathbf{S}^{(l-1)}_{c_i,c_i} &\approx D^{(l)}_{c_i} - U_{c_i} B^{(l)}_{c_i,\boldsymbol{\sigma}} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} B^{(l)}_{\boldsymbol{\sigma},c_i} V^T_{c_i} \equiv D^{(l-1)}_{c_i}, \quad i \le s,\\ \mathbf{S}^{(l-1)}_{d_j,d_j} &\approx D^{(l)}_{d_j} - U_{d_j} B^{(l)}_{d_j,\boldsymbol{\delta}} \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} B^{(l)}_{\boldsymbol{\delta},d_j} V^T_{d_j} \equiv D^{(l-1)}_{d_j}, \quad j \le t.\end{aligned} \tag{3.19}$$

The two equations describe how to compute a generator $D^{(l-1)}$ from $D^{(l)}$ in the form of a low-rank update. Note that c_i is allowed to include lower level interfaces so that $D^{(l)}_{c_i}$ in (3.19) itself is in an HSS form. (It is similar for d_i and $D^{(l)}_{d_i}$.) Figure 3.4 illustrates this. In this case, the off-diagonal blocks of $D^{(l)}_{c_i}$ have U and V basis matrices related to U_{c_i} and V_{c_i}, respectively. This is due to the nested bases in the HSS structure [43]. A result in [40, Proposition 3.3] can then be used to quickly update the lower level D, B generators within $D^{(l)}_{c_i}$ to get those of $D^{(l-1)}_{c_i}$. The basis generators U, V (and R, W) remain the same.

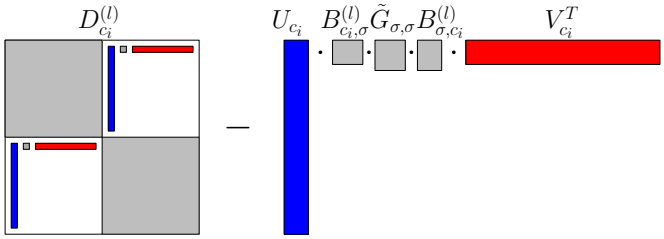


Figure 3.4: *Illustration of using (3.19) to update D generators of local Schur complements.*

II. For the off-diagonal block $\mathbf{S}^{(l-1)}_{c_i,c_j}$ corresponding to interactions between c_i and c_j (and also $\mathbf{S}^{(l-1)}_{d_i,d_j}$ corresponding to interactions between d_i and d_j), where $i \neq j$, we get also from (3.6) and (3.7) that

$$\begin{aligned}\mathbf{S}^{(l-1)}_{c_i,c_j} &\approx U_{c_i}(B^{(l)}_{c_i,c_j} - B^{(l)}_{c_i,\boldsymbol{\sigma}} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} B^{(l)}_{\boldsymbol{\sigma},c_j}) V^T_{c_j} \equiv U_{c_i} B^{(l-1)}_{c_i,c_j} V^T_{c_j}, \quad i,j \le s, \quad i \neq j,\\ \mathbf{S}^{(l-1)}_{d_i,d_j} &\approx U_{d_i}(B^{(l)}_{d_i,d_j} - B^{(l)}_{d_i,\boldsymbol{\delta}} \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} B^{(l)}_{\boldsymbol{\delta},d_j}) V^T_{d_j} \equiv U_{d_i} B^{(l-1)}_{d_i,d_j} V^T_{d_j}, \quad i,j \le t, \quad i \neq j.\end{aligned}$$

Thus, we obtain low-rank approximations to $\mathbf{S}^{(l-1)}_{c_i,c_j}$ and $\mathbf{S}^{(l-1)}_{d_i,d_j}$ by simply reusing existing U, V basis matrices. We just need to update the B generators as follows:

$$\begin{aligned}B^{(l-1)}_{c_i,c_j} &= B^{(l)}_{c_i,c_j} - B^{(l)}_{c_i,\boldsymbol{\sigma}} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\sigma}} B^{(l)}_{\boldsymbol{\sigma},c_j}, \quad i,j \le s, \quad i \neq j,\\ B^{(l-1)}_{d_i,d_j} &= B^{(l)}_{d_i,d_j} - B^{(l)}_{d_i,\boldsymbol{\delta}} \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\delta}} B^{(l)}_{\boldsymbol{\delta},d_j}, \quad i,j \le t, \quad i \neq j.\end{aligned} \tag{3.20}$$

Again, the U, V (and R, W) generators remain the same. See Figure 3.5 for an illustration.

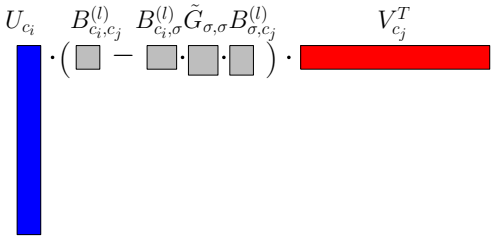


Figure 3.5: *Illustration of using (3.20) to update B generators of local Schur complements.*

III. For the off-diagonal blocks $\mathbf{S}^{(l-1)}_{c_i,d_j}$ and $\mathbf{S}^{(l-1)}_{d_j,c_i}$ corresponding to new fill-in, we get from (3.8) and (3.9) that

$$\begin{aligned}\mathbf{S}^{(l-1)}_{c_i,d_j} &\approx -U_{c_i} B^{(l)}_{c_i,\boldsymbol{\sigma}} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\delta}} B^{(l)}_{\boldsymbol{\delta},d_j} V^T_{d_j} \equiv U_{c_i} B^{(l-1)}_{c_i,d_j} V^T_{d_j}, \quad i \le s, \quad j \le t,\\ \mathbf{S}^{(l-1)}_{d_j,c_i} &\approx -U_{d_j} B^{(l)}_{d_j,\boldsymbol{\delta}} \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\sigma}} B^{(l)}_{\boldsymbol{\sigma},c_i} V^T_{c_i} \equiv U_{d_j} B^{(l-1)}_{d_j,c_i} V^T_{c_i}, \quad i \le s, \quad j \le t.\end{aligned}$$

Thus, we also obtain low-rank approximations to $\mathbf{S}^{(l-1)}_{c_i,d_j}$ and $\mathbf{S}^{(l-1)}_{d_j,c_i}$ by simply reusing existing U, V basis matrices. We just need to create new B generators as follows:

$$\begin{aligned} B^{(l-1)}_{c_i,d_j} &= -B^{(l)}_{c_i,\boldsymbol{\sigma}} \tilde{G}_{\boldsymbol{\sigma},\boldsymbol{\delta}} B^{(l)}_{\boldsymbol{\delta},d_j}, \quad i \le s, \quad j \le t, \\ B^{(l-1)}_{d_j,c_i} &= -B^{(l)}_{d_j,\boldsymbol{\delta}} \tilde{G}_{\boldsymbol{\delta},\boldsymbol{\sigma}} B^{(l)}_{\boldsymbol{\sigma},c_i}, \quad i \le s, \quad j \le t. \end{aligned} \tag{3.21}$$

See Figure 3.6.

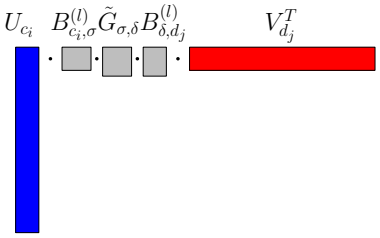


Figure 3.6: *Illustration of using (3.21) to create B generators of new fill-in in local Schur complements.*

From the three cases (3.19)–(3.21) we can observe that existing U, V basis matrices for relevant off-diagonal blocks of $S^{(l)}_\alpha$ and $S^{(l)}_\beta$ are used directly for the off-diagonal blocks of $S^{(l-1)}_\gamma$ and only the D, B generators need to be updated. That is, *the off-diagonal basis information is reused for local Schur complements across different levels l without extra compression.* This confirms the third feature in Section 3.2 when IHSS representations are introduced.

We then finalize the IHSS representation for $S^{(l-1)}_\gamma$. The neighbor list associated with γ can be obtained by collecting the interfaces $c_1, \ldots, c_s$ and $d_1, \ldots, d_t$ in (3.4). Accordingly, the corresponding HSS subtrees together with the associated U, V basis matrices are preserved. The collection of the HSS subtrees is used to form the IHSS tree for $S^{(l-1)}_\gamma$. There is only one thing left. That is, we need to merge common-origin interfaces c_j and d_k that are in α and β separately. The corresponding HSS subtrees also need to be merged into a larger HSS subtree by introducing one additional hierarchical level. Without loss of generality, suppose c_1 and d_1 are common-origin interfaces (belonging to a single separator at level $l-1$ of the separator tree $\mathbf{T}$) and is to be combined into an interface $e \equiv (c_1, d_1)$. According to [43, Algorithm 1], this just needs to introduce the generators $\begin{pmatrix} R_{c_1} \\ R_{d_1} \end{pmatrix}$ and $\begin{pmatrix} W_{c_1} \\ W_{d_1} \end{pmatrix}$ into the nested basis structure. For example, $\begin{pmatrix} R_{c_1} \\ R_{d_1} \end{pmatrix}$ is an approximate column basis matrix of

$$\begin{pmatrix} B^{(l-1)}_{c_1,c_2} & \ldots & B^{(l-1)}_{c_1,c_s} & B^{(l-1)}_{c_1,d_2} & \ldots & B^{(l-1)}_{c_1,d_t} \\ B^{(l-1)}_{d_1,c_2} & \ldots & B^{(l-1)}_{d_1,c_s} & B^{(l-1)}_{d_1,d_2} & \ldots & B^{(l-1)}_{d_1,d_t} \end{pmatrix}. \tag{3.22}$$

Since s and t are usually very small, the size of the matrix in (3.22) is small. A rank-revealing QR factorization can be applied to (3.22):

$$\begin{pmatrix} B^{(l-1)}_{c_1,c_2} & \ldots & B^{(l-1)}_{c_1,c_s} & B^{(l-1)}_{c_1,d_2} & \ldots & B^{(l-1)}_{c_1,d_t} \\ B^{(l-1)}_{d_1,c_2} & \ldots & B^{(l-1)}_{d_1,c_s} & B^{(l-1)}_{d_1,d_2} & \ldots & B^{(l-1)}_{d_1,d_t} \end{pmatrix} \approx \begin{pmatrix} R_{c_1} \\ R_{d_1} \end{pmatrix} \begin{pmatrix} B^{(l-1)}_{e,c_2} & \ldots & B^{(l-1)}_{e,c_s} & B^{(l-1)}_{e,d_2} & \ldots & B^{(l-1)}_{e,d_t} \end{pmatrix}. \tag{3.23}$$

The new B generators on the right-hand side are introduced due to the creation of e. After all such merging steps, we obtain an IHSS approximation to $S^{(l-1)}_\gamma$.

Applying the above local Schur complement updates to all the interface lists at level l of $\mathcal{T}$ produces the IHSS approximations to local Schur complements at level $l-1$. That is, we get a structured approximation to $\mathbf{S}^{(l-1)}$. This completes the Schur complement update problem (2.3) at level l. The process can then be repeated, until level 1 of $\mathcal{T}$ is reached. At this point, we only need to compute an HSS LDU factorization as in Section 3.3.1.

This factorization process produces structured factors associated with each node of $\mathcal{T}$. The factors can be used to quickly solve linear systems in (2.4)–(2.6). The structured solution algorithms can be designed conveniently and the details are omitted.

3.4. Advantages and complexity

Comparing with the Dirichlet-to-Neumann formulation [14, 29] that requires the addition and recompression of HSS matrices, and the randomized structured multifrontal method [40] that requires randomized HSS construction [28] and submatrix extraction, our solver performs the Schur complement update via the fast structured updates (3.19)–(3.21). Relevant off-diagonal basis matrices U, V are computed only once (for small blocks) and then reused across different outer factorization levels without the need for repeated direct low-rank compression. This feature helps to both save the cost and guarantee that the rank structure is fully preserved in the sparse factorization.

Our IHSS structure also avoids a tricky issue of reordering different separators in the formation of local Schur complements as needed in [42].

In addition, without the heavy need of expensive low-rank compression, we can potentially take better advantages of high-level BLAS kernels in the implementation and avoid many vector operations. For example, it is easy to see that (3.19)–(3.21) just involve BLAS-3 operations.

The complexity of our algorithm can be studied as follows. With $\mathbf{l}$ levels in $\mathcal{T}$, we choose a switching level $\mathbf{l}_s (\leq \mathbf{l})$ as in [41, 42] so as to optimize the complexity. That is, structured factorizations are applied just to the upper $\mathbf{l}_s$ levels and dense operations are applied to the bottom $\mathbf{l} - \mathbf{l}_s$ levels. One reason is that the block sizes at bottom levels may be small enough for dense operations to outperform rank-structured ones. Another reason is the complexity optimization. To estimate the total complexity, suppose n is the size of the discretized elliptic problem in two dimensions, N_l is the largest size of the local Schur complement at level l, and r is the maximum size of all the B generators in the IHSS approximations in the sparse factorization.

At a level l below the switching level, the factorization cost associated with each local Schur complement is $O(N_l^3)$. At the switching level $l = \mathbf{l}_s$, a (one-time) compression cost is used to approximate each local Schur complement as an IHSS form. The cost is $O(rN_l^2)$ like in the HSS construction in [43]. Then for $l \leq \mathbf{l}_s$, structured factorizations are performed as in Sections 3.3.1 and 3.3.2. Note that the U, V generators are reused for upper levels so the only extra compression is for a small matrix as in (3.23). The cost associated with each local Schur complement is $O(r^2 N_l)$. Without loss of generality, we assume $N_l = O(\sqrt{n/2^l})$, and the total factorization complexity is

$$\underbrace{\sum_{l=\mathbf{l}_s+1}^{\mathbf{l}} 2^l O(N_l^3)}_{\text{dense factorizations}} + \underbrace{2^{\mathbf{l}_s} O(rN_{\mathbf{l}_s}^2)}_{\substack{\text{IHSS constructions} \\ \text{at level } \mathbf{l}_s}} + \underbrace{\sum_{l=1}^{\mathbf{l}_s} 2^l O(r^2 N_l)}_{\text{IHSS factorizations}}$$
$$= O(r^2 2^{\mathbf{l}_s/2} n^{1/2}) + O(rn) + O(2^{-\mathbf{l}_s/2} n^{3/2}).$$

The optimal complexity is $O(rn)$ if we choose $2^{\mathbf{l}_s} = O(r^{-2}n)$. This complexity optimization is similar to [41, 42].

If r is bounded, then the method has a linear factorization complexity. In practice, r is allowed to slightly increase with n as $r = O(\log^p n_l)$. Then the total complexity remains $O(n)$ based on a rank relaxation study in [41]. For 2D elliptic equations and Helmholtz equations with small or complex-valued wavenumbers, we expect r to be small [2, 5, 8].

4. Extension to structured factorization update under multiple coefficient changes

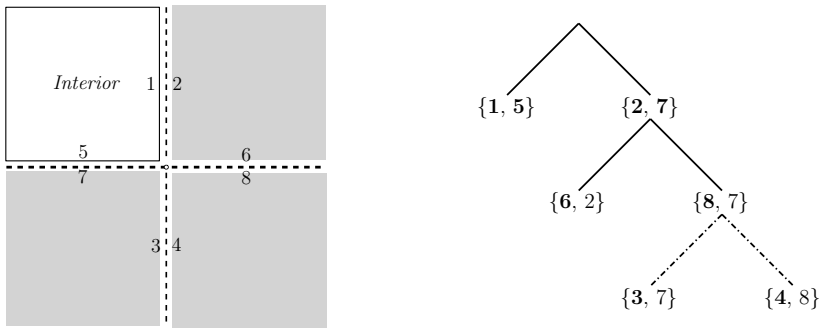
Our algorithm can be extended to a more challenging situation where the factorization needs to be updated frequently due to various local changes to the coefficient of the PDE. The magnitudes of the local changes may also be large. For such a situation, conventional sparse hierarchical factorization becomes too expensive since any local update will be propagated upward along the assembly tree. Recently in [27], a fast factorization update method is proposed that uses a set of interior and exterior factorizations. Whenever a subdomain is updated, only a small interior factorization in that subdomain needs to be done. The updated interior factor is combined with a set of existing exterior factors to produce the updated sparse factorization.

The factorization update method in [27] uses a formulation consistent with Section 2. Thus, we can use our interconnected hierarchical structured methods to accelerate the factorization update. We sketch the main ideas here without going into the tedious technical details and show some numerical tests later.

- The method in [27] starts with the hierarchical direct factorization described in Section 2 here. To distinguish from additional steps needed for the factorization update, the subdomains generated by the hierarchical domain partitioning as in Section 2.1 are called *interior subdomains*, and the factorization or elimination of unknowns associated with each interior subdomain is called an *interior factorization*. As discussed in Section 3, the IHSS factorization can be used to replace the standard interior factorization.
- The complement of an interior subdomain is called an *exterior subdomain*. In [27], an *exterior factorization* method is developed to take advantage of shared factors during the elimination of unknowns in exterior subdomains. It is shown in [27] that every exterior subdomain is a union of certain interior subdomains. Therefore, an exterior factorization can always reuse the results of certain interior factorizations to save the cost. A neighbor tree can also be constructed for the exterior factorization.
- For the factorization update due to coefficient updates in an interior subdomain, we can compute the new factorization by combining the new factors in that interior subdomain and existing factors in the corresponding exterior subdomain. This avoid the propagation of local updates to the entire sparse factorization. See [27, Section 3] for more details.
- IHSS approximations can be used to accelerate the exterior factorizations as well. When an existing factor is involved, the existing tree structures and IHSS approximations are reused. The existing formulas (3.19)–(3.21) still hold. On the boundary of a subdomain, the intermediate Schur complements from the interior and exterior factorization have the same size and the same block partitioning.

As an example, consider the matrix in (2.1) with the domain partitioning shown in Figure 2.1(a). Suppose there are coefficient updates in the upper-left subdomain which is the interior subdomain (Figure 4.1(a)). The exterior subdomain is the union of the three remaining subdomains. We can transform the neighbor tree in Figure 2.2(a) to get a new neighbor tree $\widetilde{\mathcal{T}}$ to organize the factorizations (Figure 4.1(b)). The exterior factorization can be computed by eliminating the separators $(3,4)$ and $(6,8)$ following $\widetilde{\mathcal{T}}$. The elimination of $(3,4)$ has been performed in (2.8) during the interior factorization and is reused. Thus, the corresponding subtree of $\mathcal{T}$ is reused in $\widetilde{\mathcal{T}}$. See the dash-dotted subtree in Figure 4.1(b). The elimination of $(6,8)$ can be performed similarly afterwards by

$$\begin{pmatrix} \mathbf{S}_{22}^{(1)} & \mathbf{S}_{27}^{(1)} \\ \mathbf{S}_{72}^{(1)} & \mathbf{S}_{77}^{(1)} \end{pmatrix} = \begin{pmatrix} \mathbf{S}_{22}^{(2)} & \\ & \mathbf{S}_{77}^{(1)} \end{pmatrix} - \begin{pmatrix} \mathbf{S}_{26}^{(2)} & \\ & \mathbf{S}_{78}^{(1)} \end{pmatrix} \begin{pmatrix} \mathbf{S}_{66}^{(2)} & I \\ I & \mathbf{S}_{88}^{(1)} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{S}_{62}^{(2)} & \\ & \mathbf{S}_{87}^{(1)} \end{pmatrix}.$$



(a) Interior and exterior subdomains (b) New neighbor tree $\widetilde{\mathcal{T}}$

Figure 4.1: *A pair of interior and exterior subdomains and the corresponding neighbor tree $\widetilde{\mathcal{T}}$ for organizing the factorizations. The exterior subdomain is formed by the shaded areas. The neighbor tree is transformed from the tree $\mathcal{T}$ in Figure 2.2(a). The dash-dotted subtree is reused from $\mathcal{T}$.*

After changing the coefficients in the interior subdomain, we only need to update the interior factorization in that subdomain, which corresponds to the left child of the root in Figure 4.1(b). No updated is needed for the exterior factorization (corresponding to the entire right subtree of the root in Figure 4.1(b)). The updated interior factor is then combined with the existing exterior factors to get the new overall factor. This

avoids the global propagation of local updates. The reader is referred to [27] for more details. We would just like to point out that our IHSS strategies can be used to accelerate all these elimination steps just like in Section 3. The complexity improvement is similar to that in Section 3. In the next section, a test example will be included to show the performance improvement.

5. Numerical examples

We apply our interconnected hierarchical structured methods to solve two-dimensional elliptic PDEs. The PDEs are discretized with a continuous Galerkin method, and the coefficients may contain large jump discontinuities between elements. The following two test problems are considered.

- **Problem 1.** The first problem is Poisson's equation

$$-\Delta u(x) = f(x), \quad x \in \Omega, \tag{5.1}$$

  where each local Schur complement like in (2.2) is a discretized Robin-to-Robin map that describes the linear relation from $\partial_n u + au$ to $\partial_n u - au$ on the boundary. Here, $\partial_n u$ denotes the Neumann boundary value and a is a positive constant to ensure well-posedness. For the tests, we simply choose $a = 1/h$, where h is the mesh spacing.

- **Problem 2.** The Helmholtz's equation is a more challenging test problem:

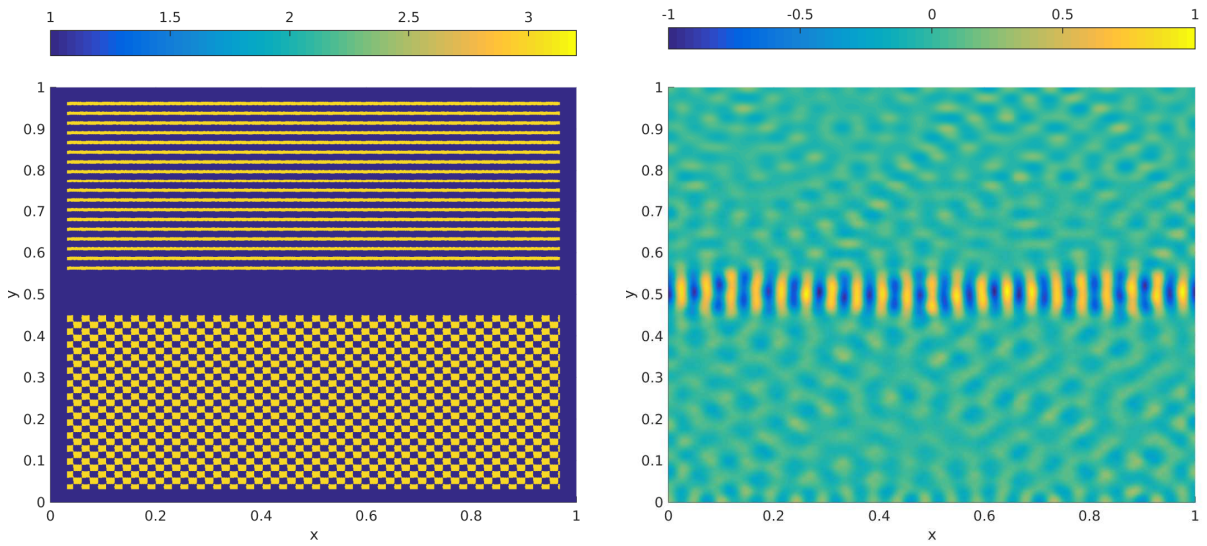
$$-\Delta u(x) - k^2(x)u(x) = f(x), \quad x \in \Omega, \tag{5.2}$$

  where $k(x)$ is the wavenumber. We follow the impedance-to-impedance formulation [13, 31], and each local Schur complement (2.2) maps from $\partial_n u - \mathrm{i}\eta u$ to $\partial_n u + \mathrm{i}\eta u$, where η is chosen as the average value of $k(x)$.

For both problems, the computational domain is $\Omega = [0,1]^2$ with a uniform triangulation. The number of unknowns is $n = (p/h+1)^2$, where p is the polynomial order of the nodal Lagrange basis functions and is chosen to be 4. The right-hand side function is chosen as $f(x) = \exp\left(-\|x-(0.5,0.5)\|^2/h^2\right)$ for both (5.1) and (5.2). For the Helmholtz's equation, the wavenumber $k(x)$ is visualized in Figure 5.1 and has a 300% jump, and Neumann boundary condition is imposed on the physical boundary $\partial\Omega$ which leads to a challenging Hermitian indefinite linear system. As the problem size increases, we fix the sampling rate: comparing with the coefficient function shown in Figure 5.1 that is constant in each rectangular piece, there are 4 points for the smallest side length among those rectangles; comparing with the shortest wavelength, there are 24 points per wavelength. In out IHSS approximations, we set a 2-norm relative tolerance $\tau = 10^{-6}$ in the compression at the switching level $\mathbf{l}_s$ and the compression step (3.23). The test workstation has a 12-core 3.5GHz Intel® Xeon® CPU with 64GB RAM. We run the algorithms in MATLAB R2015a with a single computational thread.

We compare our new sparse factorization based on IHSS methods (denoted NEW) with the standard sparse factorization without structured approximation (denoted STD). We test the factorization performance for the two problems (5.1)–(5.2) and report the following results.

- Factorization and solution time.
- Factorization and solution costs in terms of flops.
- Storage of factors in terms of the number of nonzeros.
- Solution accuracy as the relative error of the solution from NEW compared against the solution from STD.
- IHSS rank which is the size r mentioned in Section 3.4.
- *Reuse factor* defined as the ratio between the number of low-rank compression operations (rank-revealing QR factorizations) in the standard HSS construction and that in the IHSS construction for the largest three separators (in the top two levels of the separator tree). This factor measures by how many times IHSS methods reduce the number of low-rank compression operations during the elimination of those separators.



(a) Wavenumber

(b) Solution

Figure 5.1: *Visualization of the wavenumber and a solution with Neumann boundary condition for Problem 2. The wavenumber is normalized by its minimum value. The amount of jump in the wavenumber is 300% across its discontinuities. The solution corresponds to the smallest matrix size* 513^2 *listed in Table 5.1.*

Table 5.1 gives the test results of the solvers as the matrix size increases. Note that the cost of STD is the same for both Problems 1 and 2 since the discretization and nonzero patterns are the same. The corresponding scaling plots are shown in Figure 5.2. We have the following observations.

- For both problems, the factorization flops of NEW and STD scale roughly like $O(n)$ and $O(n^{1.5})$, respectively. This is consistent with the theoretical estimates.
- For the largest problem size $n = 4097^2$, the factorization flop count of NEW is 4.3 times lower than STD. The storage requirement of NEW is about 40% lower. We expect a bigger difference in the run time with more efficient future C/Fortran implementations.
- Significant basis reuse can be observed in all the tests. For the largest problem size $n = 4097^2$, the reuse factor is already 8.9, which indicates a dramatic reduction in the number of low-rank compression operations.
- The factorization costs of NEW are similar for both test problems, although the IHSS rank r increases faster with n for the Helmholtz problem. This confirms the rank relaxation idea mentioned at the end of Section 3.4.
- The interconnected structured solution is very accurate. The relative solution errors are below or around the relative compression tolerance for all the cases.

For the Helmholtz equation, we also demonstrate the performance of IHSS methods for coefficient update problems as mentioned in Section 4. In order to perform quick factorization update after local coefficient updates, we precompute the interior and exterior factorizations as mentioned in Section 4. IHSS operations are used to accelerate both types of factorizations. The results with the same discretization and wavenumber function as in the previous test are shown in Table 5.2. Here, we are not including the costs of STD since they are given in [27].

To show the efficiency of factorization updates, we introduce a local update to the coefficient by reducing the wavenumber by 1/2 in an interior subdomain containing 160^2 unknowns. We have the following observations.

- As compared with the method in [27] (which is just based on STD), the IHSS compression greatly reduces the factorization costs. For the largest problem size $n = 2561^2$, NEW needs 2.7 times fewer flops than STD for the interior factorization and 10.2 times fewer flops for the exterior factorization.

Table 5.1: *Test results with* NEW *and* STD *for Problems 1 and 2.*

Matrix size			513^2	1025^2	2049^2	4097^2
Number of nonzeros			$6,258,944$	$25,035,776$	$100,143,104$	$400,572,416$
$(\mathbf{l}, \mathbf{l}_s)$			$(8,5)$	$(10,7)$	$(12,9)$	$(14,11)$
Factorization flops	STD	Problems 1,2	$7.91E9$	$4.49E10$	$2.84E11$	$1.97E12$
	NEW	Problem 1	$6.59E9$	$2.76E10$	$1.13E11$	$4.59E11$
		Problem 2	$6.61E9$	$2.77E10$	$1.14E11$	$4.63E11$
Factorization time	STD	Problems 1,2	3.96s	17.60s	80.71s	400.94s
	NEW	Problem 1	4.52s	18.61s	77.22s	316.17s
		Problem 2	4.53s	19.28s	79.01s	319.58s
Factor storage	STD	Problems 1,2	$1.90E7$	$8.63E7$	$3.88E8$	$1.72E9$
	NEW	Problem 1	$1.61E7$	$6.64E7$	$2.70E8$	$1.10E9$
		Problem 2	$1.62E7$	$6.67E7$	$2.71E8$	$1.10E9$
IHSS rank	NEW	Problem 1	23	28	32	36
		Problem 2	25	32	53	93
Reuse factor	NEW	Problems 1,2	$\frac{292}{84} \approx 3.5$	$\frac{932}{180} \approx 5.2$	$\frac{2596}{372} \approx 7.0$	$\frac{6692}{756} \approx 8.9$
Solution flops	STD	Problems 1,2	$6.60E7$	$2.90E8$	$1.27E9$	$5.50E9$
	NEW	Problem 1	$6.30E7$	$2.61E8$	$1.06E9$	$4.30E9$
		Problem 2	$6.32E7$	$2.62E8$	$1.07E9$	$4.33E9$
Solution accuracy	NEW	Problem 1	$8.00E-9$	$1.08E-8$	$1.88E-8$	$3.11E-8$
		Problem 2	$3.79E-8$	$3.34E-7$	$4.83E-7$	$1.45E-6$

- The sizes of interior and exterior factors are reduced simultaneously by NEW. The reduction is up to 60% for the interior factors and up to 48% for the exterior ones.
- The basis reuse is even more significant than just in the interior factorization. The exterior factorization can reuse the result of the interior factorization. For the largest problem size, the reuse factor is already 16.
- Since the factorization update just needs the refactorization in the local subdomain where the coefficient changes, the factorization update cost is nearly independent of the matrix size n. The update cost with IHSS methods is also lower than the original update method in [27] based on STD and the reduction of flops is up to 30%.
- The solution accuracy after the factorization update is also well controlled by the compression accuracy.

6. Conclusions

We have designed a type of interconnected hierarchical rank structures called IHSS structures and developed fast IHSS methods for solving elliptic PDEs. Our structured sparse factorization is organized in terms of an outer neighbor tree with inner IHSS trees. Unlike the majority of existing rank-structured direct methods, our IHSS strategies can reuse off-diagonal basis matrices across the outer sparse factorization levels. It thus not only eliminates large dense intermediate matrix operations, but also avoids excessive low-rank compression operations during the structured factorization. The extensive basis reuse in IHSS structures helps to both save the cost and preserve the rank structures. The complexity of the solver is nearly linear in n for two dimensional elliptic problems and Helmholtz equations with low wavenumbers. We have also extended the solver to sparse factorization update problems. The large reuse factors in the numerical tests indicate significant reduction in the need of low-rank compression operations. We expect similar ideas can

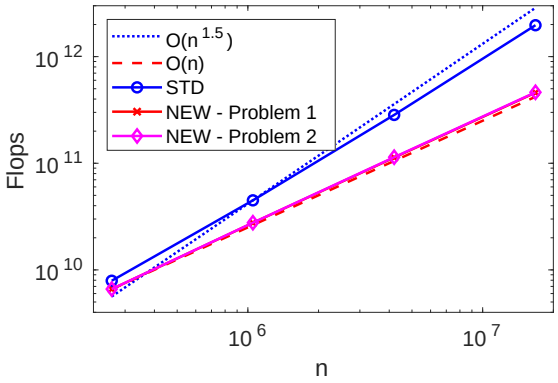


Figure 5.2: Costs of STD and NEW as in Table 5.1. The plots of NEW for Problems 1 and 2 almost overlap.

Table 5.2: *Test results using IHSS methods for interior and exterior factorizations in a factorization update for Problem 2.*

Matrix size		321^2	641^2	1281^2	2561^2
Number of nonzeros		$2,437,184$	$9,748,736$	$38,994,944$	$155,979,776$
$(\mathbf{l}, \mathbf{l}_s)$		$(6,4)$	$(8,6)$	$(10,8)$	$(12,10)$
Factorization flops	Interior	$2.90E9$	$1.23E10$	$5.10E10$	$2.08E11$
	Exterior	$1.05E9$	$6.10E9$	$2.98E10$	$1.33E11$
Factorization time	Interior	1.87s	8.09s	34.30s	141.52s
	Exterior	0.59s	3.21s	15.69s	70.78s
Factor storage	Interior	$6.29E6$	$2.62E7$	$1.08E8$	$4.36E8$
	Exterior	$3.43E6$	$1.89E7$	$8.99E7$	$3.96E8$
Reuse factor		$\frac{296}{52} \approx 5.7$	$\frac{1032}{116} \approx 8.9$	$\frac{3016}{244} \approx 12.4$	$\frac{8008}{500} \approx 16.0$
Factorization update flops		$7.33E8$	$8.37E8$	$8.37E8$	$8.37E8$
Solution update flops	Interior	$6.60E6$	$6.80E6$	$7.01E6$	$7.01E6$
	Exterior	$9.15E6$	$4.86E7$	$2.13E8$	$8.85E8$
Solution accuracy		$3.15E-8$	$2.53E-8$	$5.58E-7$	$5.92E-6$

be generalized to three dimensional problems, although the implementation and the management of the discretizations and tree structures would be much more sophisticated. This will be studied and tested in future work.

Acknowledgements

The research of Jianlin Xia was supported in part by an NSF grant DMS-1819166. Maarten V. de Hoop gratefully acknowledges support from the Simons Foundation under the MATH+X program, the NSF grant DMS-1559587, and the corporate members of the Geo-Mathematical Group at Rice University and Total.

References

[1] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker, Improving multifrontal methods by means of block low-rank representations, SIAM J. Sci. Comput. 37 (2015) A1451–A1474.

[2] M. Bebendorf and W. Hackbusch, Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operator with L^∞-coefficients, Numer. Math. 95 (2003) 1–28.

[3] J. Benamou and B. Desprès, A domain decomposition method for the Helmholtz equation and related optimal control problems, J. Comput. Phys. 136 (1997) 68–82.

[4] T. F. Chan and T. P. Mathew, Domain decomposition algorithms, Acta Numer. 3 (1994) 61–143.

[5] 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) 2261–2290.

[6] S. Chandrasekaran, M. Gu, and T. Pals, A fast ULV decomposition solver for hierarchically semiseparable representations, SIAM J. Matrix Anal. Appl. 28 (2006) 603–622.

[7] I. S. Duff and J. K. Reid, The multifrontal solution of indefinite sparse symmetric linear equations, ACM Trans. Math. Software 9 (1983) 302–325.

[8] B. Engquist and L. Ying, Sweeping preconditioner for the Helmholtz equation: hierarchical matrix representation, Comm. Pure Appl. Math 64 (2011) 697–735.

[9] Y. A. Erlangga, C. Vuik, and C. W. Oosterlee, On a class of preconditioners for the Helmholtz equation, Appl. Numer. Math. 50 (2005) 409–425.

[10] P. Gatto and J. S. Hesthaven, Efficient Preconditioning of hp-FEM matrices by hierarchical low-rank approximations, J. Sci. Comput. 72 (2017) 49–80.

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

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

[13] A. Gillman, A. H. Barnett, and P. G. Martinsson, A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media, BIT Numer. Math. 55 (2015) 141–170.

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

[15] L. Grasedyck, R. Kriemann, and S. Le Borne, Domain-decomposition based $\mathcal{H}$-LU *preconditioners*, in Domain Decomposition Methods in Science and Engineering XVI, O.B. Widlund and D.E. Keyes (eds.), Springer LNCSE, 55 (2006) 661–668.

[16] W. Hackbusch and S. Börm, Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices, Computing 69 (2002) 1–35.

[17] W. Hackbusch, L. Grasedyck, and S. Börm, An introduction to hierarchical matrices, Math. Bohem. 127 (2002) 229–241.

[18] W. Hackbusch, B. N. Khoromskij, and R. Kriemann, Direct Schur complement method by domain decomposition based on $\mathcal{H}$-matrix approximation, Comput. Vis. Sci. 8 (2005) 179–188.

[19] N. Halko, P. G. Martinsson, and J. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM Review 53 (2011) 217–288.

[20] J. S. Hesthaven and T. Warburton, Nodal discontinuous Galerkin methods: algorithms, analysis, and applications, Springer Science & Business Media 2007.

[21] K. L. Ho and L. Ying, Hierarchical interpolative factorization for elliptic operators: differential equations, Commun. Pure Appl. Math. 69 (2016) 1415–1451.

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

[23] Y. Li and L. Ying, Distributed-memory hierarchical interpolative factorization, Res. Math. Sci. 4 (2017) 1–23.

[24] L. Lin, J. Lu, and L. Ying, Fast construction of hierarchical matrix representation from matrix-vector multiplication, J. Comput. Phys. 230 (2011) 4071–4087.

[25] J. W. Liu, The multifrontal method for sparse matrix solution: Theory and practice, SIAM Review 34 (1992) 82–109.

[26] X. Liu, J. Xia, and M. V. de Hoop, Parallel randomized and matrix-free direct solvers for large structured dense linear systems, SIAM J. Sci. Comput. 38 (2016) S508–S538.

[27] X. Liu, J. Xia, and M. V. de Hoop, Fast factorization update for general elliptic equations under multiple coefficient updates, SIAM J. Sci. Comput., under revision, Purdue CCAM Report CCAM-2018-3, http://ccam.math.purdue.edu/php-scripts/download_preprint.php/locupd.pdf?id=77.

[28] P. G. Martinsson, A fast randomized algorithm for computing hierarchically semiseparable representation of a matrix, SIAM. J. Matrix Anal. Appl. 32 (2011) 1251–1274.

[29] P. G. Martinsson, A direct solver for variable coefficient elliptic PDEs discretized via a composite spectral collocation method, J. Comput. Phys. 242 (2013) 460–479.

[30] S. Parter, The use of linear graphs in Gauss elimination, SIAM Rev., 3 (1961), pp. 119–130.

[31] M. Pedneault, T. Catalin, and Y. Boubendir, Schur complement domain decomposition methods for the solution of multiple scattering problems, IMA J. Appl. Math. 82 (2017) 1104–1134.

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

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

[34] A. Toselli and W. B. Olof, Domain decomposition methods: algorithms and theory, Berlin: Springer 34 (2005).

[35] S. Wang, M. V. de Hoop, and J. Xia, On 3D modeling of seismic wave propagation via a structured parallel multifrontal direct Helmholtz solver, Geophys. Prospect. 59 (2011) 857–873.

[36] Y. Xi and J. Xia, On the stability of some hierarchical rank structured matrix algorithms, SIAM J. Matrix Anal. Appl. 37 (2016) 1279–1303.

[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) 44–72.

[38] Y. Xi, J. Xia, and R. Chan, A fast randomized eigensolver with structured LDL factorization update, SIAM J. Matrix Anal. Appl. 35 (2014) 974–996.

[39] J. Xia, On the complexity of some hierarchical structured matrix algorithms, SIAM J. Matrix Anal. Appl. 33 (2012) 388–410.

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

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

[42] 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 (2009) 1382–1411.

[43] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, Fast algorithms for hierarchically semiseparable matrices, Numer. Linear Algebra Appl. 17 (2010) 953–976.

[44] J. Xia, Y. Xi, and M. Gu, A superfast structured solver for Toeplitz linear systems via randomized sampling, SIAM J. Matrix Anal. Appl. 33 (2012) 837–858.

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