SIAM J. Matrix Anal. Appl.
Vol. 39, No. 2, pp. 829–855

PRESERVING POSITIVE DEFINITENESS IN HIERARCHICALLY SEMISEPARABLE MATRIX APPROXIMATIONS[*]

XIN XING[†] AND EDMOND CHOW[†]

Abstract. Given a symmetric positive definite (s.p.d.) matrix, two methods are proposed for directly constructing hierarchically semiseparable (HSS) matrix approximations that are guaranteed to be positive definite. The methods are based on a new, recursive description of the HSS approximation process where projection is used to compress the off-diagonal blocks. The recursive description also leads to a new error analysis of HSS approximations. By constructing an s.p.d. HSS approximation directly, rather than in a factored form, the approximation errors can be better understood. As could be expected, larger approximation errors are introduced in the new s.p.d. methods compared to those in existing HSS approximation methods where positive definiteness is not guaranteed. However, numerical tests using the approximations as preconditioners show that methods that preserve positive definiteness may be better than other methods even when those methods happen to generate a positive definite preconditioner. Like existing HSS construction algorithms, the new methods have quadratic computational complexity and can be implemented in parallel.

Key words. hierarchically semiseparable, HSS, hierarchical matrix representation, symmetric positive definite

AMS subject classifications. 65F08, 65F30

DOI. 10.1137/17M1137073

1. Introduction.

Using the low-rank nature of off-diagonal blocks, hierarchical matrix representations enable scalable computations for certain types of dense matrices arising in applications. Among established hierarchical matrix representations, $\mathcal{H}$ [15, 18] and $\mathcal{H}^2$ [17, 16, 8] are the most flexible, allowing high-rank off-diagonal blocks and adaptive matrix partitionings, and can be highly accurate. Conversely, hierarchically semiseparable (HSS) [10] and hierarchically off-diagonal low-rank (HODLR) [1] representations only allow low-rank off-diagonal blocks and rigid matrix partitionings, but these features lead to particularly fast decomposition and solve algorithms [2, 3, 28, 25]. As many matrices do not exactly have HSS or HODLR forms, these representations are often used as approximations to a matrix. To take advantage of hierarchical structures for solving linear systems, it is natural to use highly accurate $\mathcal{H}$ or $\mathcal{H}^2$ approximations to accelerate matrix-vector multiplications or to use less accurate HSS or HODLR approximations as preconditioners in Krylov subspace methods.

Given a symmetric positive definite (s.p.d.) matrix A, it is desirable to compute an approximate hierarchical representation that is also positive definite. However, positive definiteness is not guaranteed as the hierarchical representations focus on compressing off-diagonal blocks into low-rank representations. Preserving positive definiteness is essential for hierarchical approximations to be used efficiently in various algorithms and applications, e.g., using the hierarchical representation to accelerate matrix-vector products in the conjugate gradient algorithm. The goal of this paper

[*]Received by the editors July 3, 2017; accepted for publication (in revised form) by S. LeBorne February 6, 2018; published electronically May 15, 2018.
http://www.siam.org/journals/simax/39-2/M113707.html
Funding: This work was supported by NSF grant ACI-1306573.
[†]School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA 30332 (xxing33@gatech.edu, echow@cc.gatech.edu).

is to propose two methods for constructing HSS approximations to an s.p.d. matrix that preserve positive definiteness.

Instead of directly producing an s.p.d. hierarchical representation of A, there currently exist methods that construct approximate Cholesky factors of A in HSS form [29, 20]. These algorithms are sequential in the sense that they all involve sequential updates to Schur complements. Our algorithms are different in that they construct s.p.d. HSS matrices directly and are much more parallel.

Another approach to avoid the loss of positive definiteness is diagonal compensation. Bebendorf and Hackbusch [4] build an s.p.d. $\mathcal{H}$ approximation by adding corrections to the diagonal blocks based on the approximation of each off-diagonal block. Xia [27] adds diagonal shifts to the intermediate blocks in the symmetric ULV decomposition [28] of HSS approximations when breakdowns occur. In comparison, our algorithms construct an s.p.d. HSS approximation only through the compression of off-diagonal blocks.

Of our two methods, the second (called Method 2) is related to the recently developed "structured incomplete factorization" (SIF) from Xia's group [31], which also targets s.p.d. matrices. Both Method 2 and SIF use scalings by diagonal blocks, which for us is one way to preserve positive definiteness. SIF produces a ULV-type factorization [10], which can be constructed with much more parallelism than Cholesky factorizations. Although SIF is likely but not guaranteed to be positive definite when more than one level is used, our paper provides a framework for establishing SIF variants that are positive definite by construction.

For constructing an HSS approximation using projection to compress the off-diagonal blocks into low-rank form (e.g., using a truncated form of QR factorization) [10, 28, 25], we give a new recursive description of the process that says the HSS approximation A_{hss} can be constructed recursively from the leaf level to the root level of the partition tree as

$$A = A^{(0)} \Rightarrow A^{(1)} \Rightarrow A^{(2)} \Rightarrow \cdots \Rightarrow A^{(L)} = A_{\text{hss}},$$

where $A^{(k)}$ is an approximant constructed from $A^{(k-1)}$.

Based on this new description, two positive-definite-preserving methods are designed by making sure that every stage of the construction, $A^{(k-1)} \Rightarrow A^{(k)}$, maintains positive definiteness. Both methods are efficient in that they can be implemented in parallel at each level and their computational complexities are of the same order as existing HSS construction methods [28, 25]. Furthermore, the two methods are also flexible in that they can be combined with existing HSS approximation methods and techniques to help reduce computational complexity.

This paper also gives a new way of estimating the HSS approximation error, based on the recursive description above. In fact, it can be proved that the errors at each stage are orthogonal, giving

$$\|A - A_{\text{hss}}\|_F^2 = \|A^{(0)} - A^{(1)}\|_F^2 + \|A^{(1)} - A^{(2)}\|_F^2 + \cdots + \|A^{(L-1)} - A^{(L)}\|_F^2,$$

which is also closely related to the error analysis of $\mathcal{H}^2$ approximations in [16]. By working with $\|A^{(k)} - A^{(k-1)}\|_F^2$, both lower and upper bounds of $\|A - A_{\text{hss}}\|_F^2$ are obtained. It turns out that $\|A^{(k)} - A^{(k-1)}\|_F^2$ at each stage can be bounded above and below within a factor of 2. These bounds are directly related to the corresponding low-rank approximation errors in the HSS construction process. Thus this error analysis justifies existing methods (e.g., [28, 25]) in that they minimize the square of the approximation error at each stage within a factor of 2.

By directly constructing an s.p.d. HSS approximation rather than an approximation in factored form, the errors incurred by the approximation process are better understood, especially by using the orthogonality of the errors at each stage, as shown above. Once an s.p.d. HSS approximation is constructed, an *exact* symmetric ULV decomposition [28] can be computed, if needed, to be applied in various applications.

2. Symmetric HSS definition and notation.

Any matrix $A \in \mathbb{R}^{n\times n}$ can be associated with an index set $I = \{1, 2, \ldots, n\}$. A block of A can be represented as $A_{I_i\times I_j}$, where I_i and I_j are two subsets of I. Hierarchical matrix structures partition matrices into blocks which may be nested inside other blocks. Thus we need a notation for a hierarchy of index sets. This is naturally provided by a tree structure called a "partition tree." HSS representations use binary partitions and thus binary trees. For simplicity, we assume the binary trees are perfect (all levels are completely filled).

A perfect binary tree $\mathcal{T}$ with m leaves is a set of nodes, $\mathcal{T} = \{1, 2, \ldots, 2m-1\}$. To use this tree to describe the partitioning of an HSS matrix, each node i is associated with an index set I_i. If i is a nonleaf node, then $I_i = I_{c_1} \bigcup I_{c_2}$, where c_1 and c_2 are children of node i, and $I_{c_1} \bigcap I_{c_2} = \varnothing$. The index set associated with the root node is I, containing all the indices. Figure 1 shows an example of a 2-level binary partition tree using a postorder numbering of the nodes [28] and the associated hierarchical partitioning of a matrix.

The *level* structure of the binary tree is important in this paper. Nodes at the leaf level are in level 1; their parents are in level 2, etc. The last level is the level just below the root, called level L. This numbering of the levels is the reverse of what is generally used in the HSS literature but is more natural in this paper, as the construction of HSS representations is conceptually from the leaves toward the root.

We now define a *symmetric* HSS matrix, following the definition of general HSS matrices from [26].

DEFINITION 2.1. *Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$, a binary partition tree $\mathcal{T}$, and hierarchical index sets $\{I_i\}_{i\in\mathcal{T}}$, the matrix A is a* symmetric HSS matrix *if there are generators D_i, U_i, R_i associated with each nonroot node $i \in \mathcal{T}$ and generators B_{ij} associated with each pair of siblings $i, j \in \mathcal{T}$ that can be defined recursively from the leaf level to the root level as follows. For each leaf node i, we define the generator $D_i = A_{I_i\times I_i}$. For each nonleaf node i with children c_1 and c_2, the generator $D_i = A_{I_i\times I_i}$ has the structure*

$$D_i = \begin{pmatrix} D_{c_1} & U_{c_1}B_{c_1c_2}U_{c_2}^T \\ U_{c_2}B_{c_1c_2}^TU_{c_1}^T & D_{c_2} \end{pmatrix}$$

with the nested basis property

$$U_i = \begin{pmatrix} U_{c_1} & \\ & U_{c_2} \end{pmatrix} \begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix}. \tag{1}$$

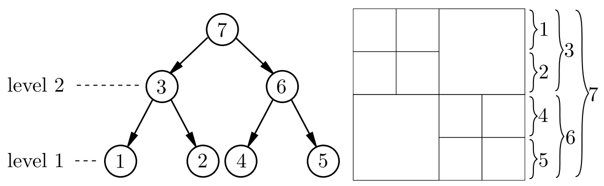


FIG. 1. *Example partition tree and associated partitioning of a matrix. Following most HSS literature, this is called a 2-level tree. In this paper, the levels are numbered upward from the leaf level.*

As an example, the HSS matrix structure corresponding to the tree in Figure 1 is

$$\left(\begin{array}{c|c} \begin{smallmatrix} D_1 & U_1 B_{12} U_2^T \\ U_2 B_{12}^T U_1^T & D_2 \end{smallmatrix} & U_3 B_{36} U_6^T \\ \hline U_6 B_{36}^T U_3^T & \begin{smallmatrix} D_4 & U_4 B_{45} U_5^T \\ U_5 B_{45}^T U_4^T & D_5 \end{smallmatrix} \end{array}\right).$$

If we assume a fixed off-diagonal block rank of r, then at each level, generator U_i has dimensions $|I_i| \times r$, generator B_{ij} has dimensions $r \times r$, and in the expressions

$$U_3 = \begin{pmatrix} U_1 & \\ & U_2 \end{pmatrix} \begin{pmatrix} R_1 \\ R_2 \end{pmatrix} \qquad \text{and} \qquad U_6 = \begin{pmatrix} U_4 & \\ & U_5 \end{pmatrix} \begin{pmatrix} R_4 \\ R_5 \end{pmatrix}$$

the generators R_1, R_2, R_4, and R_5 have dimensions $r \times r$. For simplicity, we construct HSS approximations using a fixed rank r for off-diagonal blocks and using perfect binary partition trees, but these simplifications are easily lifted.

We further use the following notation:

- The dimension of matrix A is denoted by n, and the number of elements in I_i is denoted by n_i.
- For all $i, j \in \mathcal{T}$, denote $A_{I_i \times I_j}$ as A_{ij} when there is no ambiguity.
- Denote $\text{lvl}(k)$ as the set of tree nodes at the kth level, e.g., $\text{lvl}(1) = \{1, 2, 4, 5\}$ for the tree in Figure 1.
- For all $i \in \text{lvl}(k)$, i^c is the complement set $\text{lvl}(k) \setminus \{i\}$.
- The block $A_{I_i \times (I \setminus I_i)}$ (which is not a contiguous block of A in general) is called the *HSS block row* of index set I_i, which we abbreviate as A_{ii^c}. It has dimensions $n_i \times (n - n_i)$. (An *HSS block column* can be defined similarly, but due to symmetry of A, this concept will not be needed in this paper.) The HSS representation exploits the low-rank nature of these blocks and existing HSS construction techniques choose U_i to *compress* these blocks into low-rank form. As an example, A_{22^c} for the above example is (A_{21}, A_{24}, A_{25}).
- For any nonleaf node $i \in \text{lvl}(k)$, denote its left and right children as $l_i, r_i \in \text{lvl}(k-1)$.
- Denote $\hat{R}_i = \left(R_{l_i}^T, R_{r_i}^T\right)^T$ for every nonleaf node $i \in \mathcal{T}$ so that the nested basis property can be written as $U_i = \begin{pmatrix} U_{l_i} & \\ & U_{r_i} \end{pmatrix} \hat{R}_i$.

3. Recursive description of HSS construction.

Constructing the HSS representation of a matrix A means constructing the generators D_i, U_i, R_i, and $B_{l_i r_i}$, some of which are stored implicitly or are not needed at each level. If an approximate representation is desired, then approximations are made in the choice of the U_i generators when compressing the HSS block rows and columns.

Construction of the generators for two sibling nodes generally precedes the construction of the generators for their parent (however, see [21]). Although this affects the choice of approximations (when an approximate HSS representation is desired), this is a practical ordering of the computation since the construction cost for the parent can be reduced by using the nested basis property and the compressed representations at the children. Thus, construction may proceed, for example, by visiting the tree nodes in postorder fashion [28, 26], which promotes data locality, or in bottom-up level-by-level fashion [22, 25, 23, 14], which promotes parallelism.

Different HSS methods use different methods of compressing off-diagonal blocks. When a matrix exactly fits the HSS form (for a given rank r), then the methods

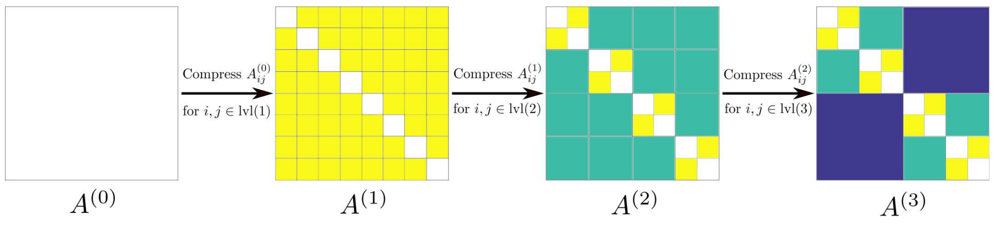

Compress $A_{ij}^{(0)}$ for $i,j \in$ lvl(1)
Compress $A_{ij}^{(1)}$ for $i,j \in$ lvl(2)
Compress $A_{ij}^{(2)}$ for $i,j \in$ lvl(3)
$A^{(0)}$
$A^{(1)}$
$A^{(2)}$
$A^{(3)}$

FIG. 2. *Recursive construction of a 3-level HSS approximation. Yellow, green, and blue denote blocks that have been compressed at each level. In general, we say that $A^{(k-1)}$ is compressed at the kth level to obtain $A^{(k)}$ for $k = 1, \ldots, L$. To illustrate our notation, if $k = 2$ and $i \neq j \in lvl(k)$, then $A_{ij}^{(k)}$ is a green block in $A^{(2)}$, while $A_{ij}^{(k-1)}$ is the corresponding four yellow blocks in $A^{(1)}$ to be compressed to form the green block.*

mathematically produce the same result. Results may be different when an approximation in HSS form is sought. We focus on compressing off-diagonal blocks A_{ij} into the form $U_i U_i^T A_{ij} U_j U_j^T$, where the columns of U_i are orthonormal and $U_i U_i^T$ is a projector usually designed to minimize the projection error $\|A_{ii^c} - U_i U_i^T A_{ii^c}\|$ of the HSS block row A_{ii^c} (similarly for U_j and $U_j U_j^T$). We refer to this as the "projection method" for compressing off-diagonal blocks and it is equivalent to existing methods, e.g., [16, 10, 28, 25]. SVD, rank revealing QR factorization, or QR factorization with pivoting may be used to construct U_i. Alternative compression methods include interpolative decomposition [22, 30, 23, 14, 9] and adaptive cross approximation [5, 6].

We now introduce a recursive description of HSS construction using the projection method that will simplify our derivation of s.p.d. HSS approximations. The process uses the projection method just mentioned and is described using the bottom-up level-by-level construction order, but our methods can also be implemented using other construction orderings.

Figure 2 gives an overview of the recursive description. Denote the original matrix A as $A^{(0)}$. The blocks of $A^{(0)}$ partitioned at the first level are denoted as $A_{ij}^{(0)}$, where i and j refer to index sets I_i and I_j, where $i, j \in \text{lvl}(1)$. These nondiagonal blocks are compressed into the form $U_i U_i^T A_{ij}^{(0)} U_j U_j^T$ and the overall compressed matrix (with its diagonal blocks untouched) is called $A^{(1)}$, as shown in Figure 2. The process is then repeated using the index sets at levels 2 and 3, etc., corresponding to partitioning the matrix by coarser and coarser blocks, until $A^{(L)}$ is obtained as the HSS approximation.

Formally, for levels k from 1 to L,

$$\begin{aligned} A^{(k)} &= \text{diag}(\{A_{ii}^{(k-1)}\}_{i\in\text{lvl}(k)}) \\ &\quad + \text{diag}(\{U_i U_i^T\}_{i\in\text{lvl}(k)})(A^{(k-1)} - \text{diag}(\{A_{ii}^{(k-1)}\}_{i\in\text{lvl}(k)}))\text{diag}(\{U_i U_i^T\}_{i\in\text{lvl}(k)}), \end{aligned} \tag{2}$$

where the notation $\text{diag}(\{M_i\}_{i\in\text{lvl}(k)})$ denotes the block diagonal matrix composed of all the blocks $\{M_i\}_{i\in\text{lvl}(k)}$ in order. For example, for the 2-level HSS matrix in Figure 1,

$$\text{diag}(\{A_{ii}\}_{i\in\text{lvl}(1)}) = \begin{pmatrix} A_{11} & & & \\ & A_{22} & & \\ & & A_{44} & \\ & & & A_{55} \end{pmatrix}.$$

This notation will be abusively simplified as $\text{diag}(M_i)$ with $i \in \text{lvl}(k)$ implied for a given level k.

We note that by the nested basis property, it can be proved that

$$U_i U_i^T A_{ij}^{(k-1)} U_j U_j^T = U_i U_i^T A_{ij}^{(0)} U_j U_j^T \quad \forall i \neq j \in \text{lvl}(k),$$

which says that it did not matter that, for example, blocks at level 2 were compressed using compressed blocks at level 1; the result of the compression at level 2 is the same as if we compressed the blocks of the original matrix directly. Practically, this means that for each pair of siblings i and j belonging to $\mathrm{lvl}(k)$, the generator B_{ij} satisfies

$$B_{ij} = U_i^T A_{ij}^{(k-1)} U_j = U_i^T A_{ij}^{(0)} U_j.$$

4. New HSS approximations that preserve positive definiteness.

Given an s.p.d. matrix A, our goal is to find an s.p.d. HSS approximation. The recursive description of HSS approximation presented in the last section allows us to simplify this task. The recursive description can be written abstractly as

$$A = A^{(0)} \underset{i\in \mathrm{lvl}(1)}{\overset{U_i}{\Longrightarrow}} A^{(1)} \underset{i\in \mathrm{lvl}(2)}{\overset{U_i}{\Longrightarrow}} A^{(2)} \cdots \underset{i\in \mathrm{lvl}(L)}{\overset{U_i}{\Longrightarrow}} A^{(L)} = A_{\mathrm{hss}}.$$

To construct an s.p.d. HSS approximation A_{hss}, it suffices to make sure that each stage, $A^{(k-1)} \Rightarrow A^{(k)}$, computed by update formula (2) maintains positive definiteness. Following this approach, two methods are now presented.

4.1. Method 1.

First, rewrite the update formula (2) for level k as

$$A^{(k)} = \mathrm{diag}(U_i U_i^T) A^{(k-1)} \mathrm{diag}(U_i U_i^T) + \mathrm{diag}(A_{ii}^{(k-1)} - U_i U_i^T A_{ii}^{(k-1)} U_i U_i^T), \tag{3}$$

where we remind the reader that $i \in \mathrm{lvl}(k)$ within each $\mathrm{diag}(\cdot)$ is implied. The first term on the right-hand side is positive semidefinite as we assume $A^{(k-1)}$ to be positive definite. Meanwhile, the second term is a block diagonal matrix and thus it suffices to make sure that each block in that matrix is positive definite.

As shown by Proposition 4.1 below, the only possible choice of U_i that makes $A_{ii}^{(k-1)} - U_i U_i^T A_{ii}^{(k-1)} U_i U_i^T$ positive semidefinite is the one where $\mathrm{col}(U_i)$ is an invariant subspace of $A_{ii}^{(k-1)}$. In this paper, we focus on the simplest invariant subspaces, those spanned by any set of eigenvectors. The columns of U_i are chosen to be orthonormal eigenvectors of $A_{ii}^{(k-1)}$. In addition, Proposition 4.2 shows that this choice of U_i can guarantee $A^{(k)}$ to be positive definite even though $A_{ii}^{(k-1)} - U_i U_i^T A_{ii}^{(k-1)} U_i U_i^T$ can only be guaranteed to be positive semidefinite. It is worth noting that a more general method might exist by exploiting different invariant subspaces of $A_{ii}^{(k-1)}$.

PROPOSITION 4.1. *Given an s.p.d. matrix $A \in \mathbb{R}^{n\times n}$ and a tall and skinny matrix $U \in \mathbb{R}^{n\times r}$ with orthonormal columns, $A - UU^TAUU^T$ is positive semidefinite if and only if $\mathrm{col}(U)$ is an invariant subspace of A.*

Proof. Define $\tilde{U} \in \mathbb{R}^{n\times(n-r)}$ with columns forming an orthonormal basis of $\mathrm{col}(U)^\perp$. Then,

$$\begin{aligned} A - UU^TAUU^T &= (UU^T + \tilde{U}\tilde{U}^T)A(UU^T + \tilde{U}\tilde{U}^T) - UU^TAUU^T \\ &= \tilde{U}\tilde{U}^TA\tilde{U}\tilde{U}^T + \tilde{U}\tilde{U}^TAUU^T + UU^TA\tilde{U}\tilde{U}^T. \end{aligned} \tag{4}$$

For any vector $z = Ux + \tilde{U}y$ with $x \in \mathbb{R}^r$, $y \in \mathbb{R}^{n-r}$, the quadratic form is written as $z^T(A - UU^TAUU^T)z = y^T\tilde{U}^TA\tilde{U}y + 2x^TU^TA\tilde{U}y$. As long as $U^TA\tilde{U}y$ is not zero, there exists $x \in \mathbb{R}^r$ such that the quadratic form is negative. Thus, $A - UU^TAUU^T$ is positive semidefinite if and only if $U^TA\tilde{U} = 0$. According to the definition of $\tilde{U}$, $(AU)^T\tilde{U} = 0$ holds true if and only if $\mathrm{col}(AU) \subset \mathrm{col}(U)$ which is equivalent to $\mathrm{col}(U)$ being an invariant subspace of A. ∎

PROPOSITION 4.2. *The choice of U_i being composed of orthonormal eigenvectors of $A_{ii}^{(k-1)}$ for any $i \in lvl(k)$ guarantees that $A^{(k)}$ constructed by formula* (3) *is always positive definite.*

Proof. Based on (4), if the columns of U_i are orthonormal eigenvectors of $A_{ii}^{(k-1)}$, then $A_{ii}^{(k-1)} - U_i U_i^T A_{ii}^{(k-1)} U_i U_i^T = \tilde{U}_i \tilde{U}_i^T A_{ii}^{(k-1)} \tilde{U}_i \tilde{U}_i^T$, where the columns of $\tilde{U}_i$ form an orthonormal basis of $\mathrm{col}(U_i)^{\perp}$ as before. Formula (3) for level k becomes

$$A^{(k)} = \mathrm{diag}(U_i U_i^T) A^{(k-1)} \mathrm{diag}(U_i U_i^T) + \mathrm{diag}(\tilde{U}_i \tilde{U}_i^T) \mathrm{diag}(A_{ii}^{(k-1)}) \mathrm{diag}(\tilde{U}_i \tilde{U}_i^T).$$

Note that both $A^{(k-1)}$ and $\mathrm{diag}(A_{ii}^{(k-1)})$ are positive definite. In addition, $\mathrm{diag}(U_i U_i^T)$ and $\mathrm{diag}(\tilde{U}_i \tilde{U}_i^T)$ are exactly the projection matrices that correspond to a pair of complementary subspaces in $\mathbb{R}^n$. As a result, for any nonzero $x \in \mathbb{R}^n$, at least one of its projections $z_1 = \mathrm{diag}(U_i U_i^T)x$ and $z_2 = \mathrm{diag}(\tilde{U}_i \tilde{U}_i^T)x$ is nonzero and hence the quadratic form $x^T A^{(k)} x$ written as $x^T A^{(k)} x = z_1^T A^{(k-1)} z_1 + z_2^T \mathrm{diag}(A_{ii}^{(k-1)}) z_2$ is always positive. Thus, $A^{(k)}$ is positive definite. ∎

In summary, Method 1 is to choose each U_i to be composed of r orthonormal eigenvectors of $A_{ii}^{(k-1)}$. How to do this while minimizing the projection error and to also satisfy the nested basis property at nonleaf levels will be discussed in section 5. Pseudocode for Method 1 thus far using the bottom-up level-by-level construction order is shown in Algorithm 4.1.

Algorithm 4.1. Method 1 (abstract version).

```
Input: Original s.p.d. matrix $A$
Output: S.p.d. HSS approximation $A^{(L)}$
  set $A^{(0)} = A$
  for $k = 1, 2, \ldots, L$ do
    compute $U_i$ $\forall i \in \mathrm{lvl}(k)$ satisfying
      • columns of $U_i$ are orthonormal eigenvectors of $A_{ii}^{(k-1)}$
      • $U_i$ should minimize the projection error $\|A_{ii^c}^{(k-1)} - U_i U_i^T A_{ii^c}^{(k-1)}\|_F$
      • if $k > 1$, the nested basis property must also be satisfied
    compress $A^{(k-1)}$ by formula (2) to obtain $A^{(k)}$
  end for
```

4.2. Method 2. From (3) and (4), the update formula (2) for level k can be written as

$$\begin{aligned} A^{(k)} &= \mathrm{diag}(U_i U_i^T) A^{(k-1)} \mathrm{diag}(U_i U_i^T) + \mathrm{diag}(\tilde{U}_i \tilde{U}_i^T A_{ii}^{(k-1)} \tilde{U}_i \tilde{U}_i^T) \\ &\quad + \mathrm{diag}(U_i U_i^T A_{ii}^{(k-1)} \tilde{U}_i \tilde{U}_i^T + \tilde{U}_i \tilde{U}_i^T A_{ii}^{(k-1)} U_i U_i^T), \quad i \in \mathrm{lvl}(k), \end{aligned} \tag{5}$$

where the columns of $\tilde{U}_i$ form an orthonormal basis of $\mathrm{col}(U_i)^{\perp}$ as before. The sum of the first and second terms can be proved to be positive definite by an argument similar to that in Proposition 4.2. Thus, the block diagonal matrix $\mathrm{diag}(U_i U_i^T A_{ii}^{(k-1)} \tilde{U}_i \tilde{U}_i^T + \tilde{U}_i \tilde{U}_i^T A_{ii}^{(k-1)} U_i U_i^T)$ is the term that might make $A^{(k)}$ indefinite.

In fact, the constraint that the columns of U_i are orthonormal eigenvectors of $A_{ii}^{(k-1)}$ in Method 1 makes $U_i U_i^T A_{ii}^{(k-1)} \tilde{U}_i \tilde{U}_i^T + \tilde{U}_i \tilde{U}_i^T A_{ii}^{(k-1)} U_i U_i^T$ exactly zero. From this point of view, the key is to try to get rid of this term. Thus, Method 2 can be designed as follows.

Notice that if $A_{ii}^{(k-1)}$ is an identity matrix, then $U_iU_i^TA_{ii}^{(k-1)}\tilde{U}_i\tilde{U}_i^T$ will be zero and $A^{(k)}$ will be positive definite. Identity diagonal blocks remind us of scaling by diagonal blocks. Ignoring for now the nested basis property and focusing on one update, $A^{(k-1)} \Rightarrow A^{(k)}$, the idea is to symmetrically scale $A^{(k-1)}$ by its diagonal blocks and then to compress the scaled off-diagonal blocks using formula (2).

Formally, first calculate the symmetric factorization of $A_{ii}^{(k-1)} = S_iS_i^T$ for each node $i \in \mathrm{lvl}(k)$, and scale the matrix $A^{(k-1)}$ as $C^{(k-1)} = \mathrm{diag}(S_i^{-1})A^{(k-1)}\mathrm{diag}(S_i^{-T})$. Then, compress the off-diagonal blocks of $C^{(k-1)}$ using formula (2) to obtain

$$C^{(k)} = \mathrm{diag}(C_{ii}^{(k-1)}) + \mathrm{diag}(V_iV_i^T)(C^{(k-1)} - \mathrm{diag}(C_{ii}^{(k-1)}))\mathrm{diag}(V_iV_i^T) \tag{6}$$

with $i \in \mathrm{lvl}(k)$, where $V_i \in \mathbb{R}^{n_i\times r}$ has orthonormal columns. Intuitively, V_i should be chosen to minimize the projection error $\|C_{ii^c}^{(k-1)} - V_iV_i^TC_{ii^c}^{(k-1)}\|_F$. Finally, define $A^{(k)} = \mathrm{diag}(S_i)C^{(k)}\mathrm{diag}(S_i^T)$, which can be generated recursively as

$$A^{(k)} = \mathrm{diag}(A_{ii}^{(k-1)}) + \mathrm{diag}(S_iV_iV_i^TS_i^{-1})(A^{(k-1)} - \mathrm{diag}(A_{ii}^{(k-1)}))\mathrm{diag}(S_i^{-T}V_iV_i^TS_i^T), \tag{7}$$

where each off-diagonal block is compressed as $A_{ij}^{(k)} = S_iV_iV_i^TS_i^{-1}A_{ij}^{(k-1)}S_j^{-T}V_jV_j^TS_j^T$. Denote $U_i = S_iV_i$ and $W_i^T = V_i^TS_i^{-1}$. Unlike before, this newly defined U_i does not have orthonormal columns and $U_iW_i^T$ is not a projection matrix. However, the matrix $S_iV_iV_i^TS_i^{-1}$ can be proved to be the projection onto the subspace $\mathrm{col}(S_iV_i)$ with inner product defined as $(x, y) = x^TS_i^{-T}S_i^{-1}y$.

The update formula (7) gives a positive definite matrix and the only requirement so far has been that the columns of V_i are orthonormal. However, we still must guarantee that the nested basis property is satisfied, which will place an additional condition on V_i.

Based on the definition of the nested basis property in (1), there should exist $\hat{R}_i = \left(R_{l_i}^T, R_{r_i}^T\right)^T$ such that

$$V_i = S_i^{-1}\begin{pmatrix} U_{l_i} & \\ & U_{r_i}\end{pmatrix}\hat{R}_i,$$

which is equivalent to $\mathrm{col}(V_i) \subset \mathrm{col}\left(S_i^{-1}\begin{pmatrix} U_{l_i} & \\ & U_{r_i}\end{pmatrix}\right)$. By the definition of $C^{(k-1)}$ and the update of $A^{(k-2)}$ to $A^{(k-1)}$ through (7), it can be noted that

$$\mathrm{col}(C_{ii^c}^{(k-1)}) \subset \mathrm{col}(S_i^{-1}A_{ii^c}^{(k-1)}) \subset \mathrm{col}\left(S_i^{-1}\begin{pmatrix} U_{l_i}W_{l_i}^TA_{l_ii^c}^{(k-2)} \\ U_{r_i}W_{r_i}^TA_{r_ii^c}^{(k-2)}\end{pmatrix}\right) \subset \mathrm{col}\left(S_i^{-1}\begin{pmatrix} U_{l_i} & \\ & U_{r_i}\end{pmatrix}\right).$$

Previously, V_i was chosen to compress $C_{ii^c}^{(k-1)}$ as $V_iV_i^TC_{ii^c}^{(k-1)}$. Thus the sufficient condition $\mathrm{col}(V_i) \subset \mathrm{col}(C_{ii^c}^{(k-1)})$ to satisfy the nested basis property can be enforced naturally.

Pseudocode for Method 2 thus far using the bottom-up level-by-level construction order is shown in Algorithm 4.2.

5. Implementation of Method 1.

The previous section gave the description of two s.p.d. HSS construction methods without implementation details. Here, an efficient implementation of Method 1 is presented. The implementation is related to building an HSS representation using projection, which we explain first. This method, which does not try to preserve positive definiteness, will be called the *standard HSS* method in this paper and it is exactly the symmetric version of [25].

Algorithm 4.2. Method 2 (abstract version).

```
Input: Original s.p.d. matrix $A$
Output: S.p.d. HSS approximation $A^{(L)}$
  set $A^{(0)} = A$
  for $k = 1, 2, \ldots, L$ do
    construct a symmetric factorization $A_{ii}^{(k-1)} = S_i S_i^T \ \forall i \in \mathrm{lvl}(k)$
    calculate the scaled off-diagonal block $C_{ij}^{(k-1)} = S_i^{-1} A_{ij}^{(k-1)} S_j^{-T} \forall i \neq j \in \mathrm{lvl}(k)$
    compute $V_i \ \forall i \in \mathrm{lvl}(k)$ satisfying
      • columns of $V_i$ are orthonormal and $\mathrm{col}(V_i) \subset \mathrm{col}(C_{ii^c}^{(k-1)})$
      • $V_i$ should minimize the projection error $\|C_{ii^c}^{(k-1)} - V_i V_i^T C_{ii^c}^{(k-1)}\|_F$
    compress $A^{(k-1)}$ by formula (7) to obtain $A^{(k)}$
  end for
```

5.1. Standard HSS construction using projection. For level k from 1 to L, define the $r \times r$ matrix $M_{ij}^{(k)} = U_i^T A_{ij}^{(k-1)} U_j \ \forall i \neq j \in \mathrm{lvl}(k)$ which satisfies $A_{ij}^{(k)} = U_i M_{ij}^{(k)} U_j^T$ by the update formula (2). Here, i, j for $M_{ij}^{(k)}$ are used as partition tree node indices like those in U_i and B_{ij} and do not refer to $I_i \times I_j$ like in $A_{ij}^{(k)}$. Notice that for each pair of siblings $i, j \in \mathrm{lvl}(k)$, generator $B_{ij} = U_i^T A_{ij}^{(k-1)} U_j = M_{ij}^{(k)}$.

We also define $M_{ij}^{(k-1)}$ where i and j (with $i \neq j$) belong to $\mathrm{lvl}(k)$ rather than $\mathrm{lvl}(k-1)$ as the $2r \times 2r$ matrix that satisfies

$$A_{ij}^{(k-1)} = \begin{pmatrix} A_{l_i l_j}^{(k-1)} & A_{l_i r_j}^{(k-1)} \\ A_{r_i l_j}^{(k-1)} & A_{r_i r_j}^{(k-1)} \end{pmatrix} = \begin{pmatrix} U_{l_i} & \\ & U_{r_i} \end{pmatrix} \begin{pmatrix} M_{l_i l_j}^{(k-1)} & M_{l_i r_j}^{(k-1)} \\ M_{r_i l_j}^{(k-1)} & M_{r_i r_j}^{(k-1)} \end{pmatrix} \begin{pmatrix} U_{l_j}^T & \\ & U_{r_j}^T \end{pmatrix}.$$

At the leaf level, the generators U_i for $i \in \mathrm{lvl}(1)$ are computed to minimize the projection error when compressing $A_{ii^c}^{(0)}$. At level $k > 1$, U_i for $i \in \mathrm{lvl}(k)$ are defined implicitly by R_{l_i} and R_{r_i} which are computed to minimize the projection error when compressing $A_{ii^c}^{(k-1)}$.

By exploiting the nested basis property and the above relation between $A_{ij}^{(k-1)}$ and $M_{ij}^{(k-1)}$ for $i, j \in \mathrm{lvl}(k)$, the projection error is

$$\left\| A_{ii^c}^{(k-1)} - U_i U_i^T A_{ii^c}^{(k-1)} \right\|_F = \left\| M_{ii^c}^{(k-1)} - \hat{R}_i \hat{R}_i^T M_{ii^c}^{(k-1)} \right\|_F, \tag{8}$$

where $M_{ii^c}^{(k-1)}$ is naturally defined as $\left(M_{ij_1}^{(k-1)}, M_{ij_2}^{(k-1)} \ \ldots\right)$, with $\{j_1, j_2, \ldots\} = i^c$. In addition, by the nested basis property, the columns of U_i being orthonormal is equivalent to the columns of $\hat{R}_i$ being orthonormal. Thus the thin matrix $M_{ii^c}^{(k-1)}$ of size $2r \times 2r(|\mathrm{lvl}(k)| - 1)$ is the target matrix to compress to obtain $\hat{R}_i$. After that, $\{M_{ij}^{(k)}\}_{i,j \in \mathrm{lvl}(k)}$ can be calculated from $\{M_{pq}^{(k-1)}\}_{p,q \in \mathrm{lvl}(k-1)}$ as

$$M_{ij}^{(k)} = \hat{R}_i^T M_{ij}^{(k-1)} \hat{R}_j.$$

To summarize, the standard HSS approximation is given in Algorithm 5.1 and it has $O(rn^2)$ computational complexity with fixed rank r.

5.2. Method 1: Constrained optimization problem. In Method 1, at level k, we seek U_i that minimizes the projection error $\|A_{ii^c}^{(k-1)} - U_i U_i^T A_{ii^c}^{(k-1)}\|_F$, for $i \in$

Algorithm 5.1. Standard bottom-up level-by-level HSS construction.

```
Input: HSS rank $r$, original matrix $A$
Output: HSS approximation with generators $\{D_i\}, \{B_{ij}\}, \{U_i\}, \{R_i\}$
  At the leaf level
    set $D_i = A_{ii}$ $\forall i \in \mathrm{lvl}(1)$
    compute $U_i \in \mathbb{R}^{n_i \times r}$ $\forall i \in \mathrm{lvl}(1)$ satisfying
      • columns of $U_i$ are orthonormal
      • $U_i$ should minimize $\|A_{ii^c} - U_i U_i^T A_{ii^c}\|_F$
    compute $M_{ij}^{(1)} = U_i^T A_{ij} U_j$ $\forall i \neq j \in \mathrm{lvl}(1)$
    set $B_{ij} = M_{ij}^{(1)}$ for every pair of siblings $i, j \in \mathrm{lvl}(1)$
  for $k = 2, 3, \ldots, L$ do
    compute $\hat{R}_i = (R_{l_i}^T, R_{r_i}^T)^T \in \mathbb{R}^{2r \times r}$ $\forall i \in \mathrm{lvl}(k)$ satisfying
      • columns of $\hat{R}_i$ are orthonormal
      • $\hat{R}_i$ should minimize $\|M_{ii^c}^{(k-1)} - \hat{R}_i \hat{R}_i^T M_{ii^c}^{(k-1)}\|_F$
    compute $M_{ij}^{(k)} = \hat{R}_i^T M_{ij}^{(k-1)} \hat{R}_j$ $\forall i \neq j \in \mathrm{lvl}(k)$
    set $B_{ij} = M_{ij}^{(k)}$ for every pair of siblings $i, j \in \mathrm{lvl}(k)$
  end for
```

lvl(k), while U_i is also constrained to be exactly r orthonormal eigenvectors of the diagonal block $A_{ii}^{(k-1)}$. Ignoring for now the nested basis property that must also be satisfied at nonleaf levels, this leads to the following constrained optimization problem.

Problem 5.1. *Given an orthogonal matrix $V \in \mathbb{R}^{m \times m}$ and a target matrix $B \in \mathbb{R}^{m \times l}$, find r columns of V, which we call $U \in \mathbb{R}^{m \times r}$, such that U minimizes the projection error $\|B - UU^T B\|_F$.*

The solution is as follows. Any r columns of V can be represented as $U = VP\,(I_r, 0)^T$, where P is a permutation matrix. Substituting this U into the projection error expression, the problem becomes finding the permutation P that minimizes

$$\|B - UU^T B\|_F = \left\|B - VP\left(\begin{smallmatrix} I_r & 0 \\ 0 & 0 \end{smallmatrix}\right) P^T V^T B\right\|_F = \left\|P^T G - \left(\begin{smallmatrix} I_r & 0 \\ 0 & 0 \end{smallmatrix}\right) P^T G\right\|_F,$$

where $G = V^T B \in \mathbb{R}^{m \times l}$. Thus we only need to choose a permutation of the rows of G such that the first r rows of $P^T G$ have the largest norms.

As $\|P^T G\|_F = \|B\|_F$, we have $\|B - UU^T B\|_F \leqslant (1 - \frac{r}{n})^{\frac{1}{2}} \|B\|_F$, where equality holds only when all the rows of G have the same norm.

5.3. Method 1: Full implementation. At level 1, the diagonal blocks $A_{ii}^{(0)} \in \mathbb{R}^{n_i \times n_i}$ for $i \in \mathrm{lvl}(1)$, are typically small, and their eigendecompositions are readily computed. The U_i that are sought can be computed by solving Problem 5.1, where V is the matrix of eigenvectors and the target matrix B is the HSS block row $A_{ii^c}^{(0)}$.

At nonleaf levels k, we need U_i to satisfy the nested basis property in addition to minimizing the projection error and the columns of U_i being constrained to be selected orthonormal eigenvectors of the diagonal block $A_{ii}^{(k-1)}$.

For $i \in \mathrm{lvl}(k)$, the diagonal block $A_{ii}^{(k-1)}$ can be written as

$$A_{ii}^{(k-1)} = \begin{pmatrix} A_{l_i l_i}^{(k-2)} & U_{l_i} U_{l_i}^T A_{l_i r_i}^{(k-2)} U_{r_i} U_{r_i}^T \\ U_{r_i} U_{r_i}^T A_{r_i l_i}^{(k-2)} U_{l_i} U_{l_i}^T & A_{r_i r_i}^{(k-2)} \end{pmatrix}.$$

The columns of $U_i \in \mathbb{R}^{n_i \times r}$ being eigenvectors of $A_{ii}^{(k-1)}$ is equivalent to there being an $r \times r$ diagonal matrix Σ_i such that

$$\begin{aligned} A_{ii}^{(k-1)} U_i &= U_i \Sigma_i, \\ A_{ii}^{(k-1)} \begin{pmatrix} U_{l_i} & \\ & U_{r_i} \end{pmatrix} \hat{R}_i &= \begin{pmatrix} U_{l_i} & \\ & U_{r_i} \end{pmatrix} \hat{R}_i \Sigma_i, \\ \begin{pmatrix} A_{l_i l_i}^{(k-2)} U_{l_i} & U_{l_i} U_{l_i}^T A_{l_i r_i}^{(k-2)} U_{r_i} \\ U_{r_i} U_{r_i}^T A_{r_i l_i}^{(k-2)} U_{l_i} & A_{r_i r_i}^{(k-2)} U_{r_i} \end{pmatrix} \hat{R}_i &= \begin{pmatrix} U_{l_i} & \\ & U_{r_i} \end{pmatrix} \hat{R}_i \Sigma_i. \end{aligned}$$

By induction, $A_{l_i l_i}^{(k-2)} U_{l_i} = U_{l_i} \Sigma_{l_i}$, and $A_{r_i r_i}^{(k-2)} U_{r_i} = U_{r_i} \Sigma_{r_i}$. Thus the calculation continues as

$$\begin{pmatrix} U_{l_i} \Sigma_{l_i} & U_{l_i} U_{l_i}^T A_{l_i r_i}^{(k-2)} U_{r_i} \\ U_{r_i} U_{r_i}^T A_{r_i l_i}^{(k-2)} U_{l_i} & U_{r_i} \Sigma_{r_i} \end{pmatrix} \hat{R}_i = \begin{pmatrix} U_{l_i} & \\ & U_{r_i} \end{pmatrix} \hat{R}_i \Sigma_i,$$

$$\begin{pmatrix} \Sigma_{l_i} & U_{l_i}^T A_{l_i r_i}^{(k-2)} U_{r_i} \\ U_{r_i}^T A_{r_i l_i}^{(k-2)} U_{l_i} & \Sigma_{r_i} \end{pmatrix} \hat{R}_i = \hat{R}_i \Sigma_i. \tag{9}$$

The term $U_{l_i}^T A_{l_i r_i}^{(k-2)} U_{r_i}$ is exactly $B_{l_i r_i}$ or $M_{l_i r_i}^{(k-1)}$ and has been calculated at the previous level. Denote the leading matrix in (9) as $E_i = \begin{pmatrix} \Sigma_{l_i} & B_{l_i r_i} \\ B_{l_i r_i}^T & \Sigma_{r_i} \end{pmatrix}$.

Every step above is invertible. Thus, the columns of U_i being orthonormal eigenvectors of $A_{ii}^{(k-1)}$ is equivalent to the columns of $\hat{R}_i$ being orthonormal eigenvectors of the small $2r \times 2r$ symmetric matrix E_i.

Now, consider computing U_i to minimize the projection error of $A_{ii^c}^{(k-1)}$. From (8), minimizing $\|A_{ii^c}^{(k-1)} - U_i U_i^T A_{ii^c}^{(k-1)}\|_F$ under the nested basis constraint is equivalent to minimizing $\|M_{ii^c}^{(k-1)} - \hat{R}_i \hat{R}_i^T M_{ii^c}^{(k-1)}\|_F$. By the analysis above, with eigendecomposition $E_i = V_i \Lambda_i V_i^T$, the optimal $\hat{R}_i$ can be obtained by solving Problem 5.1 with orthogonal matrix V_i, target matrix $M_{ii^c}^{(k-1)}$, and desired rank r.

The efficient implementation of Method 1 can now be given in Algorithm 5.2. With fixed rank r, the complexity of the algorithm is $O(rn^2)$.

6. Implementation of Method 2.

6. Implementation of Method 2. The main difference between Method 2 and the standard HSS method is the symmetric scaling by diagonal blocks at each level. At first glance, the decomposition $A_{ii}^{(k-1)} = S_i S_i^T$ and the application of S_i^{-1} do not appear to be practical due to the large size of these blocks at higher levels. However, as shown below, the storage cost and computational complexity to obtain S_i and to apply S_i^{-1} at each level can be reduced and are only related to the off-diagonal block rank r. The complexity of Method 2 is still of the same order as that of the standard HSS method.

Similar to the standard HSS construction procedure, for level k from 1 to L, define $M_{ij}^{(k)} = V_i^T S_i^{-1} A_{ij}^{(k-1)} S_j^{-T} V_j \in \mathbb{R}^{r \times r}$ for $i \neq j \in \mathrm{lvl}(k)$ which satisfies $A_{ij}^{(k)} = U_i M_{ij}^{(k)} U_j^T$. If i and j are siblings, also define $B_{ij} = M_{ij}^{(k)}$ by the update formula (7).

At the leaf level, all calculations can be performed directly because the matrices are small. For compressions at nonleaf level k, i.e., to obtain $A^{(k)}$ from $A^{(k-1)}$, the following quantities need to be calculated:

- For $i \in \mathrm{lvl}(k)$, the symmetric decomposition $A_{ii}^{(k-1)} = S_i S_i^T$.
- For $i \neq j \in \mathrm{lvl}(k)$, the scaled off-diagonal block $C_{ij}^{(k-1)} = S_i^{-1} A_{ij}^{(k-1)} S_j^{-T}$.

Algorithm 5.2. Method 1 with bottom-up level-by-level construction.

```
Input: HSS rank $r$, original s.p.d. matrix $A$
Output: S.p.d. HSS approximation with generators $\{D_i\}, \{B_{ij}\}, \{U_i\}, \{R_i\}$
  At the leaf level
    set $D_i = A_{ii}$ $\forall i \in \mathrm{lvl}(1)$
    compute eigendecomposition $A_{ii} = V_i \Lambda_i V_i^T$ $\forall i \in \mathrm{lvl}(1)$
    compute $U_i$ by solving Problem 5.1 with $V_i$, $A_{ii^c}$, and $r$ $\forall i \in \mathrm{lvl}(1)$
    store diagonal matrix $\Sigma_i = U_i^T A_{ii} U_i$ $\forall i \in \mathrm{lvl}(1)$
    compute $M_{ij}^{(1)} = U_i^T A_{ij} U_j$ $\forall i \neq j \in \mathrm{lvl}(1)$
    set $B_{ij} = M_{ij}^{(1)}$ for each pair of siblings $i, j \in \mathrm{lvl}(1)$
  for $k = 2, 3, \ldots, L$ do
    compute eigendecomposition $E_i = \left(\begin{smallmatrix} \Sigma_{l_i} & B_{l_i r_i} \\ B_{l_i r_i}^T & \Sigma_{r_i} \end{smallmatrix}\right) = V_i \Lambda_i V_i^T$ $\forall i \in \mathrm{lvl}(k)$
    compute $\hat{R}_i$ by solving Problem 5.1 with $V_i$, $M_{ii^c}^{(k-1)}$, and $r$ $\forall i \in \mathrm{lvl}(k)$
    store diagonal matrix $\Sigma_i = \hat{R}_i^T E_i \hat{R}_i$ $\forall i \in \mathrm{lvl}(k)$
    compute $M_{ij}^{(k)} = \hat{R}_i^T M_{ij}^{(k-1)} \hat{R}_j$ $\forall i \neq j \in \mathrm{lvl}(k)$
    set $B_{ij} = M_{ij}^{(k)}$ for each pair of siblings $i, j \in \mathrm{lvl}(k)$
  end for
```

- For $i \neq j \in \mathrm{lvl}(k)$, $M_{ij}^{(k)} = V_i^T S_i^{-1} A_{ij}^{(k-1)} S_j^{-T} V_j = V_i^T C_{ij}^{(k-1)} V_j$.
- For $i \in \mathrm{lvl}(k)$,

$$\hat{R}_i = \begin{pmatrix} V_{l_i}^T & \\ & V_{r_i}^T \end{pmatrix} \begin{pmatrix} S_{l_i}^{-1} & \\ & S_{r_i}^{-1} \end{pmatrix} S_i V_i \tag{10}$$

to satisfy the nested basis property for $U_i = S_i V_i$.

We now show how each of these four quantities can be computed efficiently.

Symmetric decomposition $A_{ii}^{(k-1)} = S_i S_i^T$. At the leaf level, the Cholesky decomposition $A_{ii}^{(0)} = S_i S_i^T$ can be directly calculated. At nonleaf levels k, using compression at the previous level, $A_{ii}^{(k-1)}$ for $i \in \mathrm{lvl}(k)$ can be written as

$$A_{ii}^{(k-1)} = \begin{pmatrix} A_{l_i l_i}^{(k-2)} & U_{l_i} B_{l_i r_i} U_{r_i}^T \\ U_{r_i} B_{l_i r_i}^T U_{l_i}^T & A_{r_i r_i}^{(k-2)} \end{pmatrix}$$

with $U_{l_i} = S_{l_i} V_{l_i}$, $U_{r_i} = S_{r_i} V_{r_i}$, and $B_{l_i r_i} = V_{l_i}^T S_{l_i}^{-1} A_{l_i r_i}^{(k-2)} S_{r_i}^{-T} V_{r_i}$. Knowing that $A_{l_i l_i}^{(k-2)} = S_{l_i} S_{l_i}^T$ and $A_{r_i r_i}^{(k-2)} = S_{r_i} S_{r_i}^T$, diagonal block $A_{ii}^{(k-1)}$ can be decomposed as

$$A_{ii}^{(k-1)} = \begin{pmatrix} S_{l_i} & \\ & S_{r_i} \end{pmatrix} \begin{pmatrix} I & V_{l_i} B_{l_i r_i} V_{r_i}^T \\ V_{r_i} B_{l_i r_i}^T V_{l_i}^T & I \end{pmatrix} \begin{pmatrix} S_{l_i}^T & \\ & S_{r_i}^T \end{pmatrix}. \tag{11}$$

Hence, only a symmetric decomposition of the middle matrix is needed. For this, we will use the following proposition.

Proposition 6.1. *Consider a matrix* $H = \left(\begin{smallmatrix} I_{n_1} & M \\ M^T & I_{n_2} \end{smallmatrix}\right) \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)}$*, where subblock* $M \in \mathbb{R}^{n_1 \times n_2}$ *is rank* r *with SVD* $M = U\Sigma V^T$, $U \in \mathbb{R}^{n_1 \times r}$, $\Sigma \in \mathbb{R}^{r \times r}$,

$V \in \mathbb{R}^{n_2 \times r}$. The eigendecomposition of this matrix is

$$\begin{pmatrix} \frac{U}{\sqrt{2}} & \frac{U}{\sqrt{2}} & \tilde{U} & 0 \\ \frac{V}{\sqrt{2}} & \frac{-V}{\sqrt{2}} & 0 & \tilde{V} \end{pmatrix} \begin{pmatrix} I_r + \Sigma & & & \\ & I_r - \Sigma & & \\ & & I_{n_1-r} & \\ & & & I_{n_2-r} \end{pmatrix} \begin{pmatrix} \frac{U}{\sqrt{2}} & \frac{U}{\sqrt{2}} & \tilde{U} & 0 \\ \frac{V}{\sqrt{2}} & \frac{-V}{\sqrt{2}} & 0 & \tilde{V} \end{pmatrix}^T,$$

where the columns of $\tilde{U}$ and $\tilde{V}$ form orthonormal bases of $\mathrm{col}(U)^\perp$ *and* $\mathrm{col}(V)^\perp$*, respectively. Furthermore, based on this eigendecomposition, if H is s.p.d., singular values of M must be less than one and hence the eigenvalues of H are within* $(0, 2)$.

The proposition can be verified by direct calculation. Now, $B_{l_i r_i}$ is a small $r \times r$ matrix and its full SVD can be readily calculated as $B_{l_i r_i} = Q_{l_i} \Sigma_{l_i r_i} Q_{r_i}^T$. As $A_{ii}^{(k-1)}$ is s.p.d., the matrix in the middle of (11) is also s.p.d. and can be decomposed as $\bar{S}_i \bar{S}_i^T$, where

$$\bar{S}_i = \begin{pmatrix} \frac{V_{l_i} Q_{l_i}}{\sqrt{2}} & \frac{V_{l_i} Q_{l_i}}{\sqrt{2}} & \tilde{V}_{l_i} & \\ \frac{V_{r_i} Q_{r_i}}{\sqrt{2}} & \frac{-V_{r_i} Q_{r_i}}{\sqrt{2}} & & \tilde{V}_{r_i} \end{pmatrix} \begin{pmatrix} (I_r + \Sigma_{l_i r_i})^{\frac{1}{2}} & & \\ & (I_r - \Sigma_{l_i r_i})^{\frac{1}{2}} & \\ & & I_{n_1+n_2-2r} \end{pmatrix} \tag{12}$$

based on Proposition 6.1 and the columns of $\tilde{V}_{l_i}$ and $\tilde{V}_{r_i}$ form orthonormal bases of $\mathrm{col}(V_{l_i})^\perp$ and $\mathrm{col}(V_{r_i})^\perp$, respectively. Thus, a recursive definition of S_i by (11) is obtained as

$$S_i = \begin{pmatrix} S_{l_i} & \\ & S_{r_i} \end{pmatrix} \bar{S}_i, \tag{13}$$

where $\bar{S}_i$ only requires V_{l_i}, V_{r_i}, and $B_{l_i r_i}$ at the previous level.

Scaled off-diagonal block $C_{ij}^{(k-1)} = S_i^{-1} A_{ij}^{(k-1)} S_j^{-T}$. First consider the one-sided multiplication $S_i^{-1} A_{ij}^{(k-1)}$ for $i \neq j \in \mathrm{lvl}(k)$. Using compression at the previous level, $A_{ij}^{(k-1)}$ can be written as

$$\begin{aligned} A_{ij}^{(k-1)} &= \begin{pmatrix} S_{l_i} V_{l_i} V_{l_i}^T S_{l_i}^{-1} & \\ & S_{r_i} V_{r_i} V_{r_i}^T S_{r_i}^{-1} \end{pmatrix} A_{ij}^{(k-2)} \begin{pmatrix} S_{l_j} V_{l_j} V_{l_j}^T S_{l_j}^{-1} & \\ & S_{r_j} V_{r_j} V_{r_j}^T S_{r_j}^{-1} \end{pmatrix}^T \\ &= \begin{pmatrix} S_{l_i} V_{l_i} V_{l_i}^T S_{l_i}^{-1} & \\ & S_{r_i} V_{r_i} V_{r_i}^T S_{r_i}^{-1} \end{pmatrix} \tilde{A}_{ij}^{(k-2)}, \end{aligned}$$

where $\tilde{A}_{ij}^{(k-2)}$ denotes the multiplication of the last two matrices in the first line above. Combining this expression with (12) and (13), we obtain

$$\begin{aligned} &S_i^{-1} A_{ij}^{(k-1)} \\ &= \bar{S}_i^{-1} \begin{pmatrix} V_{l_i} V_{l_i}^T S_{l_i}^{-1} & \\ & V_{r_i} V_{r_i}^T S_{r_i}^{-1} \end{pmatrix} \tilde{A}_{ij}^{(k-2)} \\ &= \begin{pmatrix} (I_r + \Sigma_{l_i r_i})^{-\frac{1}{2}} & & \\ & (I_r - \Sigma_{l_i r_i})^{-\frac{1}{2}} & \\ & & I \end{pmatrix} \begin{pmatrix} \frac{Q_{l_i}^T V_{l_i}^T}{\sqrt{2}} & \frac{Q_{r_i}^T V_{r_i}^T}{\sqrt{2}} \\ \frac{Q_{l_i}^T V_{l_i}^T}{\sqrt{2}} & \frac{-Q_{r_i}^T V_{r_i}^T}{\sqrt{2}} \\ \tilde{V}_{l_i}^T & 0 \\ 0 & \tilde{V}_{r_i}^T \end{pmatrix} \begin{pmatrix} V_{l_i} V_{l_i}^T S_{l_i}^{-1} & \\ & V_{r_i} V_{r_i}^T S_{r_i}^{-1} \end{pmatrix} \tilde{A}_{ij}^{(k-2)} \end{aligned}$$

$$= \begin{pmatrix} (I_r+\Sigma_{l_ir_i})^{-\frac{1}{2}} & & \\ & (I_r-\Sigma_{l_ir_i})^{-\frac{1}{2}} & \\ & & I \end{pmatrix} \begin{pmatrix} \frac{Q_{l_i}^T}{\sqrt{2}} & \frac{Q_{r_i}^T}{\sqrt{2}} \\ \frac{Q_{l_i}^T}{\sqrt{2}} & \frac{-Q_{r_i}^T}{\sqrt{2}} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_{l_i}^T S_{l_i}^{-1} & \\ & V_{r_i}^T S_{r_i}^{-1} \end{pmatrix} \tilde{A}_{ij}^{(k-2)}$$

$$= \begin{pmatrix} I_{2r} \\ 0 \end{pmatrix} \begin{pmatrix} (I_r+\Sigma_{l_ir_i})^{-\frac{1}{2}} & \\ & (I_r-\Sigma_{l_ir_i})^{-\frac{1}{2}} \end{pmatrix} \begin{pmatrix} \frac{Q_{l_i}^T}{\sqrt{2}} & \frac{Q_{r_i}^T}{\sqrt{2}} \\ \frac{Q_{l_i}^T}{\sqrt{2}} & \frac{-Q_{r_i}^T}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} V_{l_i}^T S_{l_i}^{-1} & \\ & V_{r_i}^T S_{r_i}^{-1} \end{pmatrix} \tilde{A}_{ij}^{(k-2)}.$$

Denote

$$T_i = \begin{pmatrix} (I_r+\Sigma_{l_ir_i})^{-\frac{1}{2}} & \\ & (I_r-\Sigma_{l_ir_i})^{-\frac{1}{2}} \end{pmatrix} \begin{pmatrix} \frac{Q_{l_i}^T}{\sqrt{2}} & \frac{Q_{r_i}^T}{\sqrt{2}} \\ \frac{Q_{l_i}^T}{\sqrt{2}} & \frac{-Q_{r_i}^T}{\sqrt{2}} \end{pmatrix},$$

which is a $2r \times 2r$ matrix. Multiplying the equation above by S_j^{-T} on the right and unraveling $\tilde{A}_{ij}^{(k-2)}$, we obtain

$$S_i^{-1} A_{ij}^{(k-1)} S_j^{-T} = \begin{pmatrix} I_{2r} \\ 0 \end{pmatrix} T_i \begin{pmatrix} V_{l_i}^T S_{l_i}^{-1} & \\ & V_{r_i}^T S_{r_i}^{-1} \end{pmatrix} A_{ij}^{(k-2)} \begin{pmatrix} S_{l_j}^{-T} V_{l_j} & \\ & S_{r_j}^{-T} V_{r_j} \end{pmatrix} T_j^T \begin{pmatrix} I_{2r} & 0 \end{pmatrix}$$

$$= \begin{pmatrix} I_{2r} \\ 0 \end{pmatrix} T_i M_{ij}^{(k-1)} T_j^T \begin{pmatrix} I_{2r} & 0 \end{pmatrix}, \tag{14}$$

where

$$M_{ij}^{(k-1)} = \begin{pmatrix} M_{l_il_j}^{(k-1)} & M_{l_ir_j}^{(k-1)} \\ M_{r_il_j}^{(k-1)} & M_{r_ir_j}^{(k-1)} \end{pmatrix}.$$

From this equation, each scaled off-diagonal block $C_{ij}^{(k-1)}$ only has the top-left $2r \times 2r$ subblock being nonzero. This structure of $C_{ij}^{(k-1)}$ arises from the ordering of the eigenvalues in the eigendecomposition in Proposition 6.1. Evidently, the calculation of $C_{ij}^{(k-1)}$ only requires multiplications of matrices with dimensions $2r \times 2r$.

Let us also briefly consider the cost of choosing V_i to minimize the projection error $\|C_{ii^c}^{(k-1)} - V_i V_i^T C_{ii^c}^{(k-1)}\|_F$. With the constraint $\mathrm{col}(V_i) \subset \mathrm{col}(C_{ii^c}^{(k-1)})$, only the first $2r$ rows of V_i are nonzero. Thus both the storage and computational complexity of choosing V_i are similar to that of choosing $\hat{R}_i$ in the standard HSS construction procedure.

Calculation of $M_{ij}^{(k)}$. After finding V_i to compress the scaled off-diagonal blocks $C_{ij}^{(k-1)}$, matrix $M_{ij}^{(k)}$ at the kth level can be efficiently obtained by

$$M_{ij}^{(k)} = V_i^T \begin{pmatrix} I_{2r} \\ 0 \end{pmatrix} T_i M_{ij}^{(k-1)} T_j^T \begin{pmatrix} I_{2r} & 0 \end{pmatrix} V_j. \tag{15}$$

Calculation of $\hat{R}_i$. By the definition of S_i in (13) and (12), the calculation of $\hat{R}_i$ through (10) can be simplified as

$$\hat{R}_i = \begin{pmatrix} V_{l_i}^T & \\ & V_{r_i}^T \end{pmatrix} \bar{S}_i V_i$$

$$= \begin{pmatrix} V_{l_i}^T & \\ & V_{r_i}^T \end{pmatrix} \begin{pmatrix} \frac{V_{l_i}Q_{l_i}}{\sqrt{2}} & \frac{V_{l_i}Q_{l_i}}{\sqrt{2}} & \tilde{V}_{l_i} & \\ \frac{V_{r_i}Q_{r_i}}{\sqrt{2}} & \frac{-V_{r_i}Q_{r_i}}{\sqrt{2}} & & \tilde{V}_{r_i} \end{pmatrix} \begin{pmatrix} (I_r+\Sigma_{l_ir_i})^{\frac{1}{2}} & & \\ & (I_r-\Sigma_{l_ir_i})^{\frac{1}{2}} & \\ & & I \end{pmatrix} V_i$$

$$= \begin{pmatrix} \frac{Q_{l_i}}{\sqrt{2}} & \frac{Q_{l_i}}{\sqrt{2}} & 0 & 0 \\ \frac{Q_{r_i}}{\sqrt{2}} & \frac{-Q_{r_i}}{\sqrt{2}} & 0 & 0 \end{pmatrix} \begin{pmatrix} (I_r + \Sigma_{l_i r_i})^{\frac{1}{2}} & & \\ & (I_r - \Sigma_{l_i r_i})^{\frac{1}{2}} & \\ & & I \end{pmatrix} V_i. \tag{16}$$

Due to the zeros in the leading matrix above, the calculation of $\hat{R}_i$ only requires the multiplication of a $2r \times 2r$ matrix by a $2r \times r$ matrix.

The efficient implementation of Method 2 is now given in Algorithm 6.1. With fixed rank r, the complexity of the algorithm is also $O(rn^2)$. We remark that for the general case where HSS off-diagonal blocks have different ranks, the above procedures can be easily adapted.

Algorithm 6.1. Method 2 with bottom-up level-by-level construction.

```
Input: HSS rank $r$, original s.p.d. matrix $A$
Output: S.p.d. HSS approximation with generators $\{D_i\}, \{B_{ij}\}, \{U_i\}, \{R_i\}$
  At the leaf level
    set $D_i = A_{ii}\ \forall i \in \text{lvl}(1)$
    compute the Cholesky decomposition $A_{ii} = S_i S_i^T\ \forall i \in \text{lvl}(1)$
    compute the scaled off-diagonal block $C_{ij}^{(0)} = S_i^{-1} A_{ij} S_j^{-T}\ \forall i \neq j \in \text{lvl}(1)$
    compute $V_i \in \mathbb{R}^{n_i \times r}\ \forall i \in \text{lvl}(1)$ satisfying
      • columns of $V_i$ are orthonormal and $\text{col}(V_i) \subset \text{col}(C_{ii^c}^{(0)})$
      • $V_i$ should minimize $\|C_{ii^c}^{(0)} - V_i V_i^T C_{ii^c}^{(0)}\|_F$
    set $U_i = S_i V_i\ \forall i \in \text{lvl}(1)$
    compute $M_{ij}^{(1)} = V_i^T C_{ij}^{(0)} V_j\ \forall i \neq j \in \text{lvl}(1)$
    set $B_{ij} = M_{ij}^{(1)}$ for every pair of siblings $i, j \in \text{lvl}(1)$
  for $k = 2, 3, \ldots, L$ do
    compute SVD $B_{l_i r_i} = Q_{l_i} \Sigma_{l_i r_i} Q_{r_i}^T\ \forall i \in \text{lvl}(k)$
    construct $T_i$ in (14) by $\Sigma_{l_i r_i}, Q_{l_i}$ and $Q_{r_i}\ \forall i \in \text{lvl}(k)$
    compute the top-left $2r \times 2r$ nonzero subblock of $C_{ij}^{(k-1)}$ as $\overline{C}_{ij}^{(k-1)} = T_i M_{ij}^{(k-1)} T_j^T$
    by (14) $\forall i \neq j \in \text{lvl}(k)$
    compute the first $2r$ rows $\overline{V}_i$ of $V_i\ \forall i \in \text{lvl}(k)$ satisfying
      • columns of $\overline{V}_i$ are orthonormal and $\text{col}(\overline{V}_i) \subset \text{col}(\overline{C}_{ii^c}^{(k-1)})$
      • $\overline{V}_i$ should minimize $\|\overline{C}_{ii^c}^{(k-1)} - \overline{V}_i \overline{V}_i^T \overline{C}_{ii^c}^{(k-1)}\|_F$
    compute $\hat{R}_i$ by (16) using $\Sigma_{l_i r_i}, Q_{l_i}, Q_{r_i}$, and $\overline{V}_i\ \forall i \in \text{lvl}(k)$
    compute $M_{ij}^{(k)} = \overline{V}_i^T \overline{C}_{ij}^{(k-1)} \overline{V}_j\ \forall i \neq j \in \text{lvl}(k)$
    set $B_{ij} = M_{ij}^{(k)}$ for every pair of siblings $i, j \in \text{lvl}(k)$
  end for
```

7. HSS approximation error analysis.

7. HSS approximation error analysis. For all the construction methods discussed above, the HSS approximation is constructed level-by-level as $A = A^{(0)} \Rightarrow A^{(1)} \Rightarrow \cdots \Rightarrow A^{(L)} = A_{\text{hss}}$ using update formula (2) or (7). The approximation error $\|A^{(0)} - A^{(L)}\|_F$ can be bounded as

$$\|A^{(0)} - A^{(L)}\|_F \leqslant \sum_{k=1}^{L} \|A^{(k-1)} - A^{(k)}\|_F, \tag{17}$$

which can be found by computing $\|A^{(k-1)} - A^{(k)}\|_F$ at each level.

7.1. Error estimation for the standard HSS method. For update formula (2), we now provide an exact expression for $\|A^{(0)} - A^{(L)}\|_F^2$, demonstrating that the errors at each level are orthogonal to each other in a specific sense.

PROPOSITION 7.1. *For the recursive construction formula* (2) *with the columns of* U_i *being orthonormal,*

$$\|A^{(0)} - A^{(L)}\|_F^2 = \sum_{k=1}^{L} \|A^{(k-1)} - A^{(k)}\|_F^2. \tag{18}$$

Proof. With a matrix partitioning by index subsets at the kth level, $0 < k < L$, the approximation error can be written as

$$\|A^{(0)} - A^{(L)}\|_F^2 = \sum_{i,j \in \mathrm{lvl}(k)} \|A_{ij}^{(0)} - A_{ij}^{(k)} + A_{ij}^{(k)} - A_{ij}^{(L)}\|_F^2.$$

We will first show that each term in the summation above can be split as

$$\|A_{ij}^{(0)} - A_{ij}^{(k)} + A_{ij}^{(k)} - A_{ij}^{(L)}\|_F^2 = \|A_{ij}^{(0)} - A_{ij}^{(k)}\|_F^2 + \|A_{ij}^{(k)} - A_{ij}^{(L)}\|_F^2, \tag{19}$$

which then implies that

$$\|A^{(0)} - A^{(L)}\|_F^2 = \|A^{(0)} - A^{(k)}\|_F^2 + \|A^{(k)} - A^{(L)}\|_F^2, \quad 0 < k < L. \tag{20}$$

Consider any $i, j \in \mathrm{lvl}(k)$. If $i = j$ or if i and j are siblings, then $A_{ij}^{(k)}$ will not be modified by the update formula at any of the subsequent levels. Thus, $A_{ij}^{(k)} = A_{ij}^{(L)}$ and (19) holds true in this case. If i and j are not siblings, block $A_{ij}^{(L)}$ is obtained by the compression of some larger block with indices $I_p \times I_q$ where p, q are siblings and $I_i \subset I_p$, $I_j \subset I_q$. By the equation $A_{pq}^{(L)} = U_p U_p^T A_{pq}^{(0)} U_q U_q^T$ and the nested basis property for U_p, it can be proved that $\mathrm{col}(A_{ij}^{(L)}) \subset \mathrm{col}(U_i)$. Symmetrically, the row space of $A_{ij}^{(L)}$ satisfies $\mathrm{col}((A_{ij}^{(L)})^T) \subset \mathrm{col}(U_j)$.

From the construction process, $A_{ij}^{(k)} = U_i U_i^T A_{ij}^{(k-1)} U_j U_j^T = U_i U_i^T A_{ij}^{(0)} U_j U_j^T$ and thus $A_{ij}^{(0)} - A_{ij}^{(k)}$ can be written as

$$A_{ij}^{(0)} - A_{ij}^{(k)} = A_{ij}^{(0)} - U_i U_i^T A_{ij}^{(0)} + U_i U_i^T (A_{ij}^{(0)} - A_{ij}^{(0)} U_j U_j^T).$$

Notice that the columns of $A_{ij}^{(0)} - U_i U_i^T A_{ij}^{(0)}$ are orthogonal to $\mathrm{col}(U_i)$ and the rows of $A_{ij}^{(0)} - A_{ij}^{(0)} U_j U_j^T$ are orthogonal to $\mathrm{col}(U_j)$. Meanwhile, the columns and rows of $A_{ij}^{(k)} - A_{ij}^{(L)}$ are within $\mathrm{col}(U_i)$ and $\mathrm{col}(U_j)$, respectively. By the Pythagorean theorem and properties mentioned above, the splitting in (19) can be proved as

$$\begin{aligned}
&\|A_{ij}^{(0)} - A_{ij}^{(k)} + A_{ij}^{(k)} - A_{ij}^{(L)}\|_F^2 \\
&= \|A_{ij}^{(0)} - U_i U_i^T A_{ij}^{(0)}\|_F^2 + \|U_i U_i^T (A_{ij}^{(0)} - A_{ij}^{(0)} U_j U_j^T) + A_{ij}^{(k)} - A_{ij}^{(L)}\|_F^2 \\
&= \|A_{ij}^{(0)} - U_i U_i^T A_{ij}^{(0)}\|_F^2 + \|U_i U_i^T (A_{ij}^{(0)} - A_{ij}^{(0)} U_j U_j^T)\|_F^2 + \|A_{ij}^{(k)} - A_{ij}^{(L)}\|_F^2 \\
&= \|A_{ij}^{(0)} - A_{ij}^{(k)}\|_F^2 + \|A_{ij}^{(k)} - A_{ij}^{(L)}\|_F^2,
\end{aligned}$$

from which (20) follows.

Now, it can be observed that in the level-by-level construction process, $A^{(L)}$ can also be regarded as an HSS approximation to $A^{(l)}$ for any $0 < l < L$. Thus, (20) also holds true when replacing index 0 by l, allowing us to recursively apply (20) for l from 0 to $(L-2)$ with $k = l + 1$ to prove the proposition. ∎

It is worth mentioning that this proposition is closely related to the error analysis of $\mathcal{H}^2$ approximations in [7, 16].

For the HSS approximation with update formula (2), an upper bound for the square of the approximation error for one stage can be obtained as

$$\begin{aligned}
\|A^{(k-1)} - A^{(k)}\|_F^2 &= \sum_{i \neq j \in \mathrm{lvl}(k)} \|A_{ij}^{(k-1)} - U_i U_i^T A_{ij}^{(k-1)} U_j U_j^T\|_F^2 \\
&= \sum_{i \neq j \in \mathrm{lvl}(k)} \|A_{ij}^{(k-1)} - U_i U_i^T A_{ij}^{(k-1)}\|_F^2 + \|U_i U_i^T (A_{ij}^{(k-1)} - A_{ij}^{(k-1)} U_j U_j^T)\|_F^2 \\
&\leqslant \sum_{i \neq j \in \mathrm{lvl}(k)} \|A_{ij}^{(k-1)} - U_i U_i^T A_{ij}^{(k-1)}\|_F^2 + \|A_{ij}^{(k-1)} - A_{ij}^{(k-1)} U_j U_j^T\|_F^2 \\
&= 2 \sum_{i \in \mathrm{lvl}(k)} \|A_{ii^c}^{(k-1)} - U_i U_i^T A_{ii^c}^{(k-1)}\|_F^2. \qquad (21)
\end{aligned}$$

Meanwhile, from the second line above, $\|A^{(k-1)} - A^{(k)}\|_F^2$ can be bounded from below as

$$\|A^{(k-1)} - A^{(k)}\|_F^2 \geqslant \sum_{i \in \mathrm{lvl}(k)} \|A_{ii^c}^{(k-1)} - U_i U_i^T A_{ii^c}^{(k-1)}\|_F^2.$$

With these lower and upper bounds, the strategy of choosing each U_i to minimize the projection error of HSS block rows $A_{ii^c}^{(k-1)}$ is justified in that the strategy minimizes the square of the approximation error at each stage within a factor of 2.

Instead of a fixed rank r for U_i, the compression may be performed with a threshold ε such that $\|A_{ii^c}^{(k-1)} - U_i U_i^T A_{ii^c}^{(k-1)}\|_F^2 \leqslant \varepsilon^2$. In terms of ε, the upper bound on the square of the approximation error for one stage is

$$\|A^{(k-1)} - A^{(k)}\|_F^2 \leqslant 2|\mathrm{lvl}(k)|\varepsilon^2 = 2^{L-k+2}\varepsilon^2,$$

giving a bound on the error for the entire approximation as

$$\|A - A^{(L)}\|_F \leqslant \left(\sum_{k=1}^{L} 2^{(L-k+2)}\varepsilon^2\right)^{\frac{1}{2}} = \varepsilon\left(2^{L+2} - 4\right)^{\frac{1}{2}} \approx 2\varepsilon(n/n_0)^{\frac{1}{2}},$$

where n_0 is the average size of the index subsets $\{I_i\}_{i \in \mathrm{lvl}(1)}$ at the leaf level. Meanwhile, we have the approximate lower bound $\|A - A^{(L)}\|_F \geqslant \sqrt{2}\varepsilon(n/n_0)^{\frac{1}{2}}$ if the compression satisfies $\|A_{ii^c}^{(k-1)} - U_i U_i^T A_{ii^c}^{(k-1)}\|_F^2 \approx \varepsilon^2$.

7.2. Error estimation for Method 1. In the standard HSS method, U_i is chosen at each level to minimize the projection error $\|A_{ii^c}^{(k-1)} - UU^T A_{ii^c}^{(k-1)}\|_F$ such that the columns of U_i are orthonormal and, for nonleaf levels, U_i satisfies the nested basis property. In Method 1, we have the additional requirement that the columns of U_i are eigenvectors of $A_{ii}^{(k-1)}$. It is clear that the achievable minimum of Method 1 will not be better than that of the standard HSS method. From the error analysis of the constrained optimization problem in subsection 5.2, the minimum projection error can be as large as $(1 - \frac{r}{n_i})^{\frac{1}{2}}\|A_{ii^c}^{(k-1)}\|_F$. This worst-case projection error can be much worse than the bounds on the projection error for the standard HSS method.

However, for the Gram matrix of many s.p.d. smooth kernel functions, Method 1 may sometimes give good approximations as measured by $\|A_{ii^c}^{(k-1)} - UU^T A_{ii^c}^{(k-1)}\|_F$. A plausible explanation of when this might happen is as follows.

Based on the Mercer's theorem, any s.p.d. smooth kernel $K(x,y)$ on a compact domain Ω has an eigenfunction expansion as

$$K(x,y) = \sum_{k=1}^{\infty} \lambda_k \phi_k(x)\phi_k(y) \quad \forall x,y \in \Omega,$$

where $\{\phi_k(x)\}$ are orthonormal in $L_2(\Omega)$ and $\{\lambda_k\}$ decreases monotonically to zero. Denote the sum of the first r terms as $K^{(r)}(x,y)$ and the remainder term as $R^{(r)}(x,y)$. Their L_2-norms in $\Omega \times \Omega$ are $\|K^{(r)}\|_{L_2}^2 = \sum_{k=1}^{r} \lambda_k^2$ and $\|R^{(r)}\|_{L_2}^2 = \sum_{k=r+1}^{\infty} \lambda_k^2$.

Consider two point sets $I = \{x_j\}_{j=1}^p$ and $J = \{y_j\}_{j=1}^q$ in domains Ω_1 and Ω_2, respectively. Define $\Omega = \Omega_1 \cup \Omega_2$ for the above eigenfunction decomposition. Choose r such that $\|R^{(r)}\|_{L_2}/\|K^{(r)}\|_{L_2}$ is relatively small, say, 10^{-2}. We can write the diagonal block A_{II} and off-diagonal block A_{IJ} as

$$\begin{aligned}
A_{II} &= (K(x_j,x_l))_{x_j,x_l \in I} = K_{II}^{(r)} + R_{II}^{(r)} \\
&= \begin{pmatrix} \lambda_1\phi_1(x_1) & \dots & \lambda_r\phi_r(x_1) \\ \lambda_1\phi_1(x_2) & \dots & \lambda_r\phi_r(x_2) \\ \vdots & & \vdots \\ \lambda_1\phi_1(x_p) & \dots & \lambda_r\phi_r(x_p) \end{pmatrix} \begin{pmatrix} \phi_1(x_1) & \dots & \phi_r(x_1) \\ \phi_1(x_2) & \dots & \phi_r(x_2) \\ \vdots & & \vdots \\ \phi_1(x_p) & \dots & \phi_r(x_p) \end{pmatrix}^T + (R^{(r)}(x_j,x_l))_{x_j,x_l \in I}, \\
A_{IJ} &= (K(x_j,y_l))_{x_j \in I, y_l \in J} = K_{IJ}^{(r)} + R_{IJ}^{(r)} \\
&= \begin{pmatrix} \lambda_1\phi_1(x_1) & \dots & \lambda_r\phi_r(x_1) \\ \lambda_1\phi_1(x_2) & \dots & \lambda_r\phi_r(x_2) \\ \vdots & & \vdots \\ \lambda_1\phi_1(x_p) & \dots & \lambda_r\phi_r(x_p) \end{pmatrix} \begin{pmatrix} \phi_1(y_1) & \dots & \phi_r(y_1) \\ \phi_1(y_2) & \dots & \phi_r(y_2) \\ \vdots & & \vdots \\ \phi_1(y_q) & \dots & \phi_r(y_q) \end{pmatrix}^T + (R^{(r)}(x_j,y_l))_{x_j \in I, y_l \in J}.
\end{aligned}$$

From the viewpoint of numerical integration, $\|R_{II}^{(r)}\|_F^2$ can be roughly estimated as

$$\|R_{II}^{(r)}\|_F^2 = \frac{|I|^2}{|\Omega_1 \times \Omega_1|} \sum \frac{|\Omega_1 \times \Omega_1|}{|I|^2} |R^{(r)}(x_j,x_l)|^2 \approx \frac{|I|^2}{|\Omega_1 \times \Omega_1|} \|R^{(r)}|_{\Omega_1 \times \Omega_1}\|_{L_2}^2.$$

Thus, similar to the other matrices above, it is likely to hold that

$$\begin{aligned}
&\|R_{II}^{(r)}\|_F \sim O(|I| \|R^{(r)}|_{\Omega_1 \times \Omega_1}\|_{L_2}), && \|K_{II}^{(r)}\|_F \sim O(|I| \|K^{(r)}|_{\Omega_1 \times \Omega_1}\|_{L_2}), \\
&\|R_{IJ}^{(r)}\|_F \sim O(\sqrt{|I||J|} \|R^{(r)}|_{\Omega_1 \times \Omega_2}\|_{L_2}), && \|K_{IJ}^{(r)}\|_F \sim O(\sqrt{|I||J|} \|K^{(r)}|_{\Omega_1 \times \Omega_2}\|_{L_2}).
\end{aligned}$$

Call Φ the matrix that is common in the above expressions for A_{II} and A_{IJ}. If we assume that both $\|K^{(r)}|_{\Omega_1 \times \Omega_1}\|_{L_2}$ and $\|K^{(r)}|_{\Omega_1 \times \Omega_2}\|_{L_2}$ are of the same scale as $\|K^{(r)}\|_{L_2}$, $\|R_{II}^{(r)}\|_F$ will be relatively small compared to $\|K_{II}^{(r)}\|_F$. Thus, based on $A_{II} = K_{II}^{(r)} + R_{II}^{(r)}$ and $K_{II}^{(r)}$ being rank r, it is likely that the rank-r principal eigenvector space of A_{II} is close to $\mathrm{col}(K_{II}^{(r)}) = \mathrm{col}(\Phi)$. Similarly, the rank-$r$ principal left-singular vector space of A_{IJ} is also likely close to $\mathrm{col}(K_{IJ}^{(r)}) = \mathrm{col}(\Phi)$. As a result, it is possible that the rank-r principal eigenvector space of A_{II} and the rank-r principal left-singular vector space of A_{IJ} are close. If the spaces are close, then choosing U as some principal orthonormal eigenvectors of A_{II} may give a small projection error $\|A_{IJ} - UU^T A_{IJ}\|_F$.

The above assumption about $\|K^{(r)}|_{\Omega_1 \times \Omega_1}\|_{L_2}$ and $\|K^{(r)}|_{\Omega_1 \times \Omega_2}\|_{L_2}$ may not hold in general. A common example is for $K(x,y) = e^{-\|x-y\|^2}$ with two domains, Ω_1 and

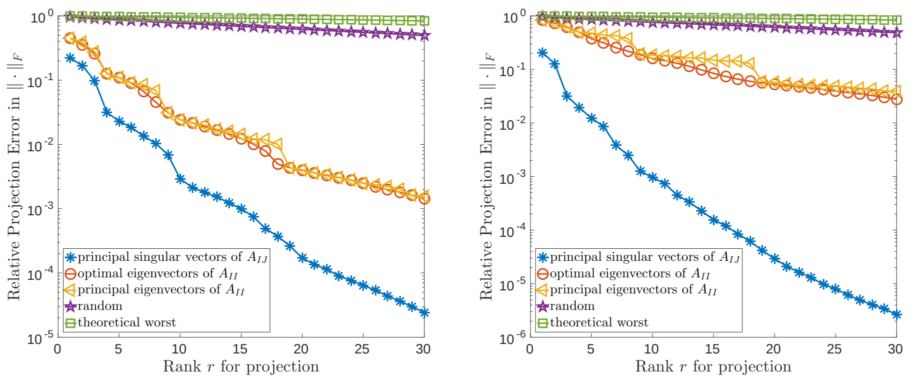
Ω_2, that are far apart. As a good approximation of $K(x, y)$, $K^{(r)}(x, y)$ on $\Omega_1 \times \Omega_2$ should have much smaller L_2 norm than on $\Omega_1 \times \Omega_1$. However, it is worth noting that, in this case, a highly accurate approximation to A_{IJ} may not be necessary for preconditioning. Numerical tests are now presented to illustrate the above argument.

Consider two clusters I and J where each contains 100 uniformly randomly distributed points within a unit cube. The centers of the two cubes lie at $(0, 0, 0)$ and $(L, 0, 0)$. For $K(x, y) = e^{-\|x-y\|^2}$, the diagonal block A_{II} and off-diagonal block A_{IJ} are defined as above. The relative projection error $\|A_{IJ} - UU^T A_{IJ}\|_F / \|A_{IJ}\|_F$ versus rank r is shown in Figure 3 with two different cluster locations $(L, 0, 0)$ for cluster J. The columns of U are chosen as follows:

- r left-singular vectors of A_{IJ} associated with the largest singular values,
- optimal r orthonormal eigenvectors of A_{II} chosen by solving Problem 5.1,
- r orthonormal eigenvectors of A_{II} associated with the largest eigenvalues,
- optimal r columns chosen by solving Problem 5.1 using a random orthogonal matrix generated by the QR decomposition of a Gaussian random matrix. The mean value from 10 tests is plotted in the figure.

In addition, the curve $(1 - \frac{r}{n})^{\frac{1}{2}}$ for the worst-case error is also plotted for comparison.

Figure 3 shows that using eigenvectors of A_{II} gives good approximation errors especially when compared to using columns of a random orthogonal matrix. In addition, the closeness between using optimal and principal eigenvectors of A_{II}, as well as the difference between results for nearby and distant clusters, both support the argument above.



(a) $(L, 0, 0) = (1, 0, 0)$, $\frac{\|A_{IJ}\|_F}{\|A_{II}\|_F} = 0.53$

(b) $(L, 0, 0) = (3, 0, 0)$, $\frac{\|A_{IJ}\|_F}{\|A_{II}\|_F} = 2.2\text{e-}3$

FIG. 3. *Relative projection error with* $K(x, y) = e^{-\|x-y\|^2}$ *for the cases of* (a) *nearby clusters and* (b) *distant clusters.*

7.3. Error estimation for Method 2.

7.3. Error estimation for Method 2. Due to the scaling of the off-diagonal blocks, Proposition 7.1 does not work for Method 2. Here, we directly estimate the HSS approximation error of Method 2 using the inequality (17). The square of the error at each stage k can be bounded as

$$\|A^{(k-1)} - A^{(k)}\|_F^2 = \sum_{i \neq j \in \mathrm{lvl}(k)} \|A_{ij}^{(k-1)} - S_i V_i V_i^T S_i^{-1} A_{ij}^{(k-1)} S_j^{-T} V_j V_j^T S_j^T\|_F^2$$

$$
\begin{aligned}
&= \sum_{i \neq j \in \mathrm{lvl}(k)} \|S_i(C_{ij}^{(k-1)} - V_i V_i^T C_{ij}^{(k-1)} V_j V_j^T) S_j^T\|_F^2 \\
&\leqslant \sum_{i \neq j \in \mathrm{lvl}(k)} \|S_i\|_2^2 \, \|S_j\|_2^2 \, \|C_{ij}^{(k-1)} - V_i V_i^T C_{ij}^{(k-1)} V_j V_j\|_F^2 \\
&\leqslant \max_{i \in \mathrm{lvl}(k)} \|S_i\|_2^4 \sum_{i \neq j \in \mathrm{lvl}(k)} \|C_{ij}^{(k-1)} - V_i V_i^T C_{ij}^{(k-1)} V_j V_j\|_F^2 \\
&\leqslant 2 \max_{i \in \mathrm{lvl}(k)} \|A_{ii}^{(k-1)}\|_2^2 \sum_{i \in \mathrm{lvl}(k)} \|C_{ii^c}^{(k-1)} - V_i V_i^T C_{ii^c}^{(k-1)}\|_F^2.
\end{aligned}
$$

The last inequality above is obtained by the same method used in (21).

Assume V_i is chosen to satisfy $\|C_{ii^c}^{(k-1)} - V_i V_i^T C_{ii^c}^{(k-1)}\|_F^2 \leqslant \varepsilon^2$ with error threshold ε. The remaining part is to estimate $\max_{i \in \mathrm{lvl}(k)} \|A_{ii}^{(k-1)}\|_2$ at each level. For any node $i \in \mathrm{lvl}(k)$, (11) gives the diagonal block as

$$
A_{ii}^{(k-1)} = \begin{pmatrix} S_{l_i} & \\ & S_{r_i} \end{pmatrix} \begin{pmatrix} I & V_{l_i} B_{l_i r_i} V_{r_i}^T \\ V_{r_i} B_{l_i r_i}^T V_{l_i}^T & I \end{pmatrix} \begin{pmatrix} S_{l_i}^T & \\ & S_{r_i}^T \end{pmatrix}.
$$

As $A_{ii}^{(k-1)}$ is positive definite, the matrix in the middle is also s.p.d. and its largest eigenvalue should be less than 2 based on Proposition 6.1. Thus, $\|A_{ii}^{(k-1)}\|_2$ can be bounded as

$$
\begin{aligned}
\|A_{ii}^{(k-1)}\|_2 &\leqslant \left\| \begin{pmatrix} S_{l_i} & \\ & S_{r_i} \end{pmatrix} \right\|_2^2 \left\| \begin{pmatrix} I & V_{l_i} B_{l_i r_i} V_{r_i}^T \\ V_{r_i} B_{l_i r_i}^T V_{l_i}^T & I \end{pmatrix} \right\|_2 \\
&\leqslant 2 \max(\|A_{l_i l_i}^{(k-2)}\|_2, \|A_{r_i r_i}^{(k-2)}\|_2) \\
&\leqslant 2 \max_{j \in \mathrm{lvl}(k-1)} \|A_{jj}^{(k-2)}\|_2 \leqslant \ldots \leqslant 2^{k-1} \max_{j \in \mathrm{lvl}(1)} \|A_{jj}^{(0)}\|_2 \leqslant 2^{k-1} \|A\|_2.
\end{aligned}
$$

The square of the approximation error can then be bounded as

$$
\|A^{(k-1)} - A^{(k)}\|_F^2 \leqslant 2^{2k-1} \|A\|_2^2 |\mathrm{lvl}(k)| \varepsilon^2 = 2^{L+k} \|A\|_2^2 \varepsilon^2.
$$

Finally, the HSS approximation error of Method 2 satisfies

$$
\|A - A^{(L)}\|_F \leqslant 2^{L/2} \|A\|_2 \varepsilon \sum_{k=1}^{L} 2^{k/2} = \|A\|_2 \varepsilon \frac{\sqrt{2}}{\sqrt{2}-1} (2^L - 2^{L/2}) \approx \frac{\sqrt{2}}{\sqrt{2}-1} \frac{n}{n_0} \|A\|_2 \varepsilon,
$$

where n_0 is the average size of the index subsets $\{I_i\}_{i \in \mathrm{lvl}(1)}$ at the leaf level.

Note that the compression in Method 2 is applied to the scaled off-diagonal block $C_{ii^c}^{(k-1)}$, i.e., finding $V_i \in \mathbb{R}^{n_i \times r}$ that minimizes $\|C_{ii^c}^{(k-1)} - V_i V_i^T C_{ii^c}^{(k-1)}\|_F^2$. Assuming that the original HSS block row $A_{ii^c}^{(k-1)}$ has fast-decaying singular values, the scaled block row $C_{ii^c}^{(k-1)}$ may not necessarily have this property. Hence, the rank of the approximation for a given ε may be large. Heuristically, choosing V_i to minimize the approximation error $\|A^{(k)} - A^{(k-1)}\|_F^2$ at each stage might be a better method, but an efficient way to do this is currently unclear.

8. Numerical results. As example applications, we are concerned with the preconditioning of s.p.d. matrices by HSS approximations in two general problems: solving linear systems $Ax = b$ using the preconditioned conjugate gradient (PCG) method and sampling correlated random vectors $y \sim \mathcal{N}(0, A)$ using a preconditioned Lanczos process [11]. To apply the preconditioners, we use the symmetric ULV factorization [28]. The following settings are shared for all experiments:

- *Hierarchical partitioning of points.* A full binary partition tree is constructed by recursively partitioning a set of points using the principal components analysis algorithm such that leaf nodes have no more than 100 points.
- *Rank of HSS off-diagonal blocks.* There are two settings: (1) The rank of U_i is fixed as a constant, r. (2) The rank of U_i is adaptively chosen using a constant relative error threshold τ in the associated compression of $A_{ii^c}^{(k-1)}$ or $C_{ii^c}^{(k-1)}$.
- *Algorithm for the projection method.* The randomized algorithms in [19] are used to estimate the principal column space of the compression target matrix with fixed rank or relative error threshold in all the HSS approximations.
- *Stopping criteria.* A threshold $\varepsilon = 10^{-8}$ is applied for all the experiments. PCG stops at iteration i when the relative reduction of the residual satisfies $\|r_i\|/\|r_0\| \leqslant \varepsilon$ and the Lanczos method stops when the relative difference between two consecutive iterates satisfies $\|y_{i+1} - y_i\|/\|y_i\| \leqslant \varepsilon$.
- *Methods.* Four methods are tested: block Jacobi, standard HSS with the projection method, Method 1, and Method 2, denoted as BJ, HSS, SPDHSS1, and SPDHSS2, respectively. The block Jacobi preconditioner is composed of the diagonal blocks associated with the leaf nodes of the partition tree. All methods are implemented in MATLAB.

8.1. Inverse multiquadric kernel. The inverse multiquadric kernel is a noncompact radial basis function whose Gram matrix is dense. The kernel is

$$K(x, y) = \frac{1}{\sqrt{1 + c\|x - y\|^2}}, \quad x, y \in \mathbb{R}^d,$$

where c is a parameter that controls the flatness of the kernel function.

To maintain constant point density, N points are randomly and uniformly distributed in a cube with edge length $\sqrt[3]{N}$ in three dimensions. We use parameters $c = 0.5$ with fixed rank $r = 50$ or relative error threshold τ= 1e-2. A test with $\tau =$ 8e-2 only for SPDHSS2 is included. Results are shown for different values of N in Table 1. No preconditioning results can be shown for standard HSS as these approximations are not positive definite in any of our settings. Also, BiCGStab with the standard HSS approximation as a preconditioner usually does not converge within $2N$ steps for most of the test settings.

With fixed rank r, the approximation errors for both SPDHSS1 and SPDHSS2 are always larger than for standard HSS, as expected. The iteration counts for SPDHSS1 and SPDHSS2 both increase with N, as the compression of larger blocks with fixed rank gives less accurate HSS approximations. In addition, SPDHSS1 requires less construction time than standard HSS, which could also have been expected.

With τ=1e-2, the iteration count using SPDHSS2 is scalable for both solving and sampling. This is at the price of much larger ranks for off-diagonal blocks as reflected by the storage cost. The results suggest that, in this example, the scaled HSS block rows $C_{ii^c}^{(k-1)}$ in Method 2 have slower-decaying singular values than HSS block rows $A_{ii^c}^{(k-1)}$ in standard HSS. Meanwhile, SPDHSS2 with τ=8e-2 obtains a better balance between construction cost and preconditioner effectiveness.

Table 1

Numerical results for the inverse multiquadric kernel with $c = 0.5$. The iteration count and consumed time for solving and sampling, relative approximation error $\|A_{apprx} - A\|_F/\|A\|_F$, construction time of preconditioners (including HSS approximation and symmetric ULV decomposition), and storage cost of the ULV factors of the HSS approximations are presented.

	N	4000	8000	12000	16000	20000
Solving iter/time (sec.)	Unprecond.	5970/20.8	11542/144.8	15708/372.2	15973/741.3	22240/1533.6
	BJ	757/3.1	1299/17.9	1248/34.8	1400/66.8	1540/111.9
	r=50 SPDHSS1	253/3.9	375/15.0	475/28.0	463/38.6	483/51.9
	SPDHSS2	195/2.9	305/10.1	373/20.9	348/29.7	430/46.8
	τ=1e-2 SPDHSS1	184/2.5	294/10.0	329/19.2	308/29.4	348/43.0
	SPDHSS2	11/0.3	11/0.8	15/2.0	11/2.5	13/3.8
	τ=8e-2 SPDHSS2	48/1.0	58/3.4	99/8.7	64/9.3	78/15.7
Sampling iter/time (sec.)	Unprecond.	567/12.1	670/36.2	962/107.1	912/129.9	989/185.8
	BJ	269/2.7	374/12.7	283/15.9	308/23.1	352/41.0
	r=50 SPDHSS1	136/3.3	163/8.2	181/11.9	164/16.9	183/26.0
	SPDHSS2	113/2.3	126/5.9	156/11.3	155/15.6	145/20.0
	τ=1e-2 SPDHSS1	104/1.7	146/5.8	138/9.5	129/14.0	126/17.2
	SPDHSS2	9/0.3	9/0.8	11/1.7	8/2.1	10/3.3
	τ=8e-2 SPDHSS2	33/0.8	37/2.4	53/5.1	35/5.5	41/8.8
Relative error	r=50 SPDHSS1	6.3e-2	7.7e-2	8.7e-2	9.1e-2	9.6e-2
	SPDHSS2	6.9e-2	8.3e-2	1.1e-1	1.1e-1	1.1e-1
	HSS	1.1e-2	1.6e-2	2.1e-2	2.4e-2	2.7e-2
	τ=1e-2 SPDHSS1	2.7e-2	3.0e-2	3.1e-2	3.2-e2	3.2e-2
	SPDHSS2	1.3e-3	1.2e-3	1.2e-3	1.2e-3	1.2e-3
	HSS	2.4e-2	2.6e-2	2.8e-2	3.0e-2	3.0e-2
	τ=8e-2 SPDHSS2	1.6e-2	1.4e-2	1.6e-2	1.5-e2	1.8e-2
Construct. time (sec.) apprx/ulv	r=50 SPDHSS1	0.7/0.1	3.5/0.2	8.0/0.3	17.4/0.5	22.1/0.5
	SPDHSS2	2.3/0.1	10.4/0.3	16.4/0.4	34.8/0.5	51.4/0.4
	HSS	1.7/-	7.4/-	13.4/-	20.4/-	34.6/-
	τ=1e-2 SPDHSS1	0.5/0.1	2.1/0.3	4.6/0.5	8.8/0.6	13.1/0.7
	SPDHSS2	11.9/0.4	50.8/1.2	130.9/2.5	257.4/4.3	421.1/6.4
	HSS	3.0/-	11.1/-	23.3/-	48.5/-	61.5/-
	τ=8e-2 SPDHSS2	5.1/0.2	24.2/0.7	46.9/1.1	98.9/1.8	157.3/2.7
Storage (MB)	Dense matrix	122	488	1098	1953	3051
	r = 50*	14	28	41	58	68
	τ=1e-2 SPDHSS1	20	45	75	101	134
	SPDHSS2	87	248	485	744	1062
	τ=8e-2 SPDHSS2	44	119	214	334	479

*With a fixed rank, ULV factors of all the HSS approximations have the same storage cost.

To compare with standard HSS, we increase the rank r for the cases with $N =$ 8000, 12000, 16000 such that standard HSS approximations are also positive definite. Results are shown in Table 2. An interesting phenomenon, which also appears with larger r and with other kernels, is that although standard HSS has a smaller approximation error, it has worse preconditioning performance compared to SPDHSS2.

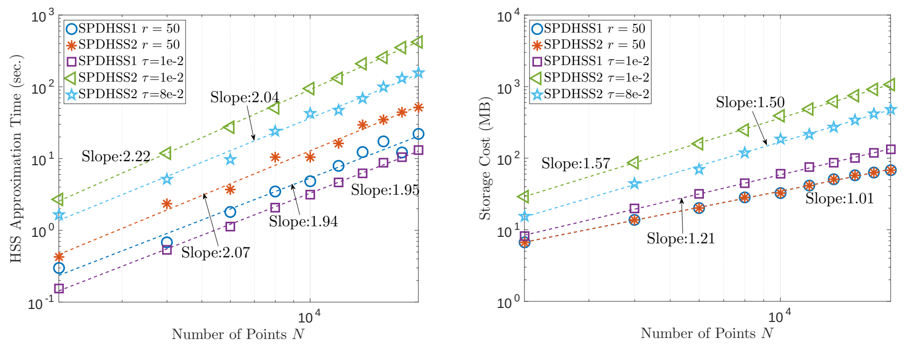
The HSS approximation time and storage cost versus N are shown in Figure 4. With fixed rank, SPDHSS1 and SPDHSS2 both have quadratic computational complexities. With fixed τ, these superlinear storage results indicate that ranks of the off-diagonal blocks in both methods are related to N.

8.2. Rotne-Prager-Yamakawa kernel. The Rotne–Prager–Yamakawa (RPY) kernel $D(x, y) : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ is a positive definite tensor function that describes the hydrodynamic interactions between particles in a viscous fluid. We previously used this kernel in coarse-grained macromolecular simulations [12, 13], and the need

TABLE 2

Comparison among different preconditioning methods for the inverse multiquadric kernel for three problem sizes when the standard HSS approximation is positive definite. The relative error (relerr), iteration count for solving (iterSO), and sampling (iterSA) are shown. The relative error does not appear to be always correlated with iteration count.

	$r = 518,\ N = 8000$			$r = 665,\ N = 12000$			$r = 730,\ N = 16000$		
	relerr	iterSO	iterSA	relerr	iterSO	iterSA	relerr	iterSO	iterSA
HSS	3.0e-5	12	10	5.3e-5	19	13	7.4e-5	22	16
SPDHSS1	1.2e-2	147	81	1.2e-2	174	85	1.3e-2	143	66
SPDHSS2	8.8e-4	7	6	2.1e-3	7	6	4.2e-3	10	8



(a) HSS approximation time (b) ULV factor storage

FIG. 4. *HSS approximation time and ULV factor storage cost versus N for the inverse multiquadric kernel with $c = 0.5$. Linear fittings are drawn with dashed lines.*

to construct positive definite preconditioners for sampling in this application was the original motivation for this work. The RPY kernel is defined as

$$D(x,y) = \begin{cases} \frac{k_BT}{6\pi\eta a} I_3 & \text{if } x = y, \\ \frac{k_BT}{8\pi\eta|r|}\left[\left(I_3 + \frac{rr^T}{|r|^2}\right) + \frac{2a^2}{|r|^2}\left(\frac{1}{3}I_3 - \frac{rr^T}{|r|^2}\right)\right] & \text{if } \|x-y\| \geqslant 2a, \\ \frac{k_BT}{6\pi\eta a}\left[\left(1 - \frac{9}{32}\frac{|r|}{a}\right) I_3 + \frac{3}{32}\frac{|r|}{a}\frac{rr^T}{|r|^2}\right] & \text{if } \|x-y\| < 2a, \end{cases}$$

with $r = x - y$, where k_B, T, η are fixed physical quantities and a is the radius of the particles. In this test, $a = 1$ and constant $\frac{k_BT}{6\pi\eta a} = 1$. We place N nonoverlapping particles randomly inside a cube with a width chosen such that the volume fraction is 0.3. Note that the RPY kernel matrix with N particles is of size $3N \times 3N$.

Results are presented in Table 3. The standard HSS approximations with $r = 50$ or τ=1e-2 are not positive definite for any tested N. This is a challenging problem, as none of the methods give scalable preconditioning performance. Although the construction cost is high in some cases, it can be amortized over the many sample vectors that must be computed for the same matrix in real applications.

8.3. Boundary integral equation. Consider the three-dimensional Laplace equation in a bounded Lipschitz domain Ω with Dirichlet condition $u = u_D$ on $\partial\Omega$. The indirect boundary integral equation [7] for this problem leads to a linear system $Vx = b$ with

Table 3
Numerical results for RPY kernel with particle volume fraction 0.3.

	N	4000	6000	8000	10000	12000
Sampling iter/time (sec.)	Unprecond.	105/4.4	116/10.2	123/15.3	131/24.9	136/36.4
	BJ	113/5.2	127/12.0	128/16.9	136/27.4	143/40.0
	r=50 SPDHSS1	68/4.3	75/9.6	81/13.3	89/21.4	98/33.5
	SPDHSS2	65/4.0	80/10.6	86/14.3	95/22.8	103/35.3
	τ=2e-2 SPDHSS1	24/3.2	26/5.7	28/8.8	29/12.3	30/16.4
	τ=8e-2 SPDHSS2	19/2.4	22/4.7	24/7.7	24/11.0	26/17.0
Relative error	r=50 SPDHSS1	2.2e-1	2.3e-1	2.4e-1	2.4e-1	2.5e-1
	SPDHSS2	1.9e-1	1.9e-1	2.0e-1	2.1e-1	2.2e-1
	τ=2e-2 SPDHSS1	5.2e-2	5.4e-2	5.8e-2	5.8e-2	6.0e-2
	τ=8e-2 SPDHSS2	2.7e-2	2.9e-2	2.8e-2	2.6e-2	2.9e-2
Construct. time (sec.) apprx/ulv	r=50 SPDHSS1	3.1/0.19	6.6/0.36	12.0/0.39	16.9/0.50	24.6/0.77
	SPDHSS2	6.7/0.19	16.0/0.37	25.8/0.39	37.3/0.51	54.5/0.65
	τ=2e-2 SPDHSS1	9.9/2.11	20.4/3.44	35.8/4.59	50.4/6.14	72.7/7.52
	τ=8e-2 SPDHSS2	87.8/1.89	178.5/3.22	349.6/5.23	581.3/7.71	866.0/12.96
Storage (MB)	Dense matrix	1099	2472	4395	6867	9888
	$r = 50$	45	76	92	129	155
	τ=2e-2 SPDHSS1	461	731	955	1251	1509
	τ=8e-2 SPDHSS2	380	645	1008	1455	1837

$$V_{ij} = \int_{\tau_i}\int_{\tau_j} \frac{1}{4\pi\|x-y\|_2}\mathrm{d}x\mathrm{d}y, \quad b_i = \int_{\tau_i} u_D(x)\mathrm{d}x, \tag{22}$$

where $\{\tau_i\}$ is a partitioning of $\partial\Omega$ and matrix V is known to be always s.p.d.

In this test, $\partial\Omega$ is the unit sphere and we solve the linear system above with uniform triangulations with different numbers of triangles, N. Entries of b are randomly selected from $[-1, 1]$. Results are shown in Table 4.

In contrast to the previous problems, the standard HSS approximations using our chosen rank r and thresholds τ are found to be positive definite for these integral equation problems. Here, standard HSS should be the preconditioner of choice but there is no guarantee that the standard HSS approximations for these problems are always positive definite. Thus, it is still of interest to see how SPDHSS1 and SPDHSS2 perform.

Similar to what was observed in Table 2, the preconditioning performance of SPDHSS2 is close to that of standard HSS, even though standard HSS gives more accurate approximations. A disadvantage of SPDHSS1 here is also evident: for the small relative error threshold τ=1e-2, SPDHSS1 needs much larger ranks than standard HSS to compress the off-diagonal blocks as reflected by the enormous storage cost. This is corroborated by Figure 3 shown earlier, where the decay of relative errors in Method 1 is much slower than that of the truncated SVD. This disadvantage suggests that SPDHSS1 should only be used for low-accuracy preconditioning operations by controlling its off-diagonal block rank r.

To summarize, based on the three tests above, SPDHSS2 is more effective than SPDHSS1 for preconditioning but requires greater construction time and storage. SPDHSS1 is faster to construct but cannot provide highly accurate approximations with low-rank compression. A careful rank selection for HSS off-diagonal blocks in both methods is needed to balance the trade-off between preconditioner quality and construction cost. Like the standard HSS construction, both methods have $O(rn^2)$ computational complexity. We remark that our implementations are sequential and

Table 4
Numerical results for the boundary integral equation over the unit sphere with different N. The meshes and matrices are constructed by the package BEM++ [24].

	N	3206	6500	9008	12600	34184
Solving	Unprecond.	80/0.32	92/0.96	104/1.92	106/3.56	140/34.32
iter/time	BJ	35/0.16	37/0.40	39/0.79	44/1.58	51/11.14
(sec.)	r=50 SPDHSS1	20/0.23	21/0.67	22/0.91	23/1.52	25/7.36
	SPDHSS2	11/0.14	13/0.43	13/0.54	15/1.06	18/5.34
	HSS	11/0.14	13/0.35	14/0.58	15/1.14	17/5.55
	τ=1e-2 SPDHSS1	9/0.36	10/1.28	10/1.74	10/3.00	11/17.26
	SPDHSS2	10/0.13	11/0.35	5/0.33	11/1.00	10/3.82
	HSS	9/0.10	10/0.27	10/0.44	10/0.71	11/3.77
	τ=5e-2 SPDHSS1	16/0.19	18/0.53	18/0.84	20/1.50	22/8.9
Relative	r=50 SPDHSS1	1.3e-1	1.4e-1	1.4e-1	1.4e-1	1.6e-1
error	SPDHSS2	6.0e-2	9.0e-2	9.5e-2	1.0e-1	1.3e-1
	HSS	2.0e-2	2.8e-2	3.3e-2	3.7e-2	4.7e-2
	τ=1e-2 SPDHSS1	1.7e-2	2.0e-2	2.0e-2	2.1e-2	2.4e-2
	SPDHSS2	1.7e-2	2.0e-2	4.0e-3	1.7e-2	7.0e-3
	HSS	1.4e-2	1.7e-2	1.5e-2	1.8e-2	2.1e-2
	τ=5e-2 SPDHSS1	8.7e-2	9.8e-2	1.0e-1	1.1e-1	1.2e-1
Construct.	r=50 SPDHSS1	0.4/0.07	1.8/0.23	2.8/0.28	6.9/0.45	52.2/1.17
time (sec.)	SPDHSS2	1.1/0.08	4.6/0.17	7.6/0.28	23.9/0.32	125.5/0.82
apprx/ulv	HSS	0.8/0.08	3.4/0.17	5.6/0.23	16.6/0.32	91.2/0.82
	τ=1e-2 SPDHSS1	1.7/0.82	8.1/3.19	15.8/6.14	33.0/11.41	410.2/127.95
	SPDHSS2	1.4/0.10	5.4/0.23	12.8/0.37	22.1/0.62	164.1/1.68
	HSS	0.7/7e-2	2.7/0.16	6.1/0.23	10.6/0.32	67.5/0.91
	τ=5e-2 SPDHSS1	0.4/0.10	1.4/0.20	2.6/0.27	5.16/0.41	37.3/1.10
Storage	Dense matrix	78	322	619	1211	8915
(MB)	$r = 50$	11	23	30	46	128
	τ=1e-2 SPDHSS1	157	552	941	1588	8237
	SPDHSS2	15	35	58	81	275
	HSS	10	21	32	46	143
	τ=5e-2 SPDHSS1	16	32	43	59	151

runtimes can be improved by taking advantage of parallelism. In addition, if the original matrix can be represented in $\mathcal{H}$ or $\mathcal{H}^2$ matrix form that is accurate enough to be s.p.d., it is then possible to reduce the construction cost of both methods by taking advantage of the low-rank structures in these forms.

9. Conclusion.

In this paper, we designed two positive-definite-preserving HSS approximation algorithms based on a recursive description of constructing HSS representations. Method 1 is different from all existing methods in that the U_i used to compress the off-diagonal blocks are based on the diagonal blocks and do not require factoring HSS block rows. That this cheaper alternative can provide good approximations in some cases, as well as its generalization to different choices of invariant subspaces for U_i, is worthy of further study. Method 2 chooses V_i to compress off-diagonal blocks after scaling. It is also worthwhile to explore how to choose V_i to directly minimize the approximation error $\|A^{(k)} - A^{(k-1)}\|$ (before scaling) at each stage, while also preserving positive definiteness.

This paper also provided a method of understanding the errors incurred at each stage of an HSS approximation when projection is used to compress off-diagonal blocks. This use of projection led to the elegant result that the errors at each stage are orthogonal to each other. Experimentally, we observed that smaller approximation error is not always correlated with better preconditioned convergence rate. Better control of the preconditioned convergence behavior via the approximations chosen in hierarchical representations is a long-term goal of this research.

REFERENCES

[1] S. Ambikasaran and E. Darve, *An $\mathcal{O}(n \log n)$ fast direct solver for partial hierarchically semi-separable matrices*, J. Sci. Comput., 57 (2013), pp. 477–501.

[2] S. Ambikasaran, M. O'Neil, and K. R. Singh, *Fast Symmetric Factorization of Hierarchical Matrices with Applications*, arXiv:1405.0223[physics, stat], 2014.

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

[4] M. Bebendorf and W. Hackbusch, *Stabilized rounded addition of hierarchical matrices*, Numer. Linear Algebra Appl., 14 (2007), pp. 407–423.

[5] M. Bebendorf and S. Rjasanow, *Adaptive low-rank approximation of collocation matrices*, Computing, 70 (2003), pp. 1–24.

[6] M. Bebendorf and R. Venn, *Constructing nested bases approximations from the entries of non-local operators*, Numer. Math., 121 (2012), pp. 609–635.

[7] S. Börm, *Efficient Numerical Methods for Non-Local Operators: $\mathcal{H}^2$-Matrix Compression, Algorithms and Analysis*, European Mathematical Society, Zürich, 2013.

[8] S. Börm, L. Grasedyck, and W. Hackbusch, *Introduction to hierarchical matrices with applications*, Engrg. Anal. Bound. Elem., 27 (2003), pp. 405–422.

[9] D. Cai, E. Chow, Y. Saad, and Y. Xi, *SMASH : Structured Matrix Approximation by Separation and Hierarchy*, submitted.

[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] E. Chow and Y. Saad, *Preconditioned Krylov subspace methods for sampling multivariate Gaussian distributions*, SIAM J. Sci. Comput., 36 (2014), pp. A588–A608.

[12] E. Chow and J. Skolnick, *Effects of confinement on models of intracellular macromolecular dynamics*, Proc. Nat. Acad. Sci., 112 (2015), pp. 14846–14851.

[13] E. Chow and J. Skolnick, *DNA internal motion likely accelerates protein target search in a packed nucleoid*, Biophys. J., 112 (2017), pp. 2261–2270.

[14] P. Ghysels, X. Li, F. 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), pp. S358–S384.

[15] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part* I*: Introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108.

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

[17] W. Hackbusch, B. Khoromskij, and S. A. Sauter, *On $\mathcal{H}^2$-matrices*, in Lectures on Applied Mathematics, H.-J. Bungartz, R. Hoppe, and C. Zenger, eds., Springer-Verlag, Berlin, 2000, pp. 9–29.

[18] W. Hackbusch and B. N. Khoromskij, *A sparse $\mathcal{H}$-matrix arithmetic. Part* II*: Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47.

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

[20] S. Li, M. Gu, C. Wu, and J. Xia, *New efficient and robust HSS Cholesky factorization of SPD matrices*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 886–904.

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

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

[23] F.-H. Rouet, X. S. Li, P. Ghysels, and A. Napov, *A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization*, ACM Trans. Math. Software, 42 (2016), pp. 27:1–27:35.

[24] W. Śmigaj, T. Betcke, S. Arridge, J. Phillips, and M. Schweiger, *Solving boundary integral problems with BEM++*, ACM Trans. Math. Software, 41 (2015), pp. 6:1–6:40.

[25] S. Wang, X. Li, J. Xia, Y. Situ, and M. de Hoop, *Efficient scalable algorithms for solving dense linear systems with hierarchically semiseparable structures*, SIAM J. Sci. Comput., 35 (2013), pp. C519–C544.

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

[27] J. Xia, *A robust inner–outer hierarchically semi-separable preconditioner*, Numer. Linear Algebra Appl., 19 (2012), pp. 992–1016.

[28] 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.

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

[30] 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), pp. 837–858.

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