ON THE STABILITY OF SOME HIERARCHICAL RANK STRUCTURED MATRIX ALGORITHMS

YUANZHE XI* AND JIANLIN XIA†

Abstract. In this paper, we investigate the numerical error propagation and provide systematic backward stability analysis for some hierarchical rank structured matrix algorithms. We prove the backward stability of various important hierarchically semiseparable (HSS) methods, such as HSS matrix-vector multiplications, HSS ULV linear system solutions, HSS linear least squares solutions, HSS inversions, and some variations. Concrete backward error bounds are given, including a structured backward error for the solution in terms of the structured factors. The error propagation factors only involve low-degree powers of the maximum off-diagonal numerical rank and the logarithm of the matrix size. Thus, as compared with the corresponding standard dense matrix algorithms, the HSS algorithms are not only faster, but also have much better stability. We also show that factorization-based HSS solutions are usually preferred, while inversion-based ones may suffer from numerical instability. The analysis builds a comprehensive framework for understanding the backward stability of hierarchical rank structured methods. The error propagation patterns also provide insights into the improvement of other types of structured solvers and the design of new stable hierarchical structured algorithms. Some numerical examples are included to support the studies.

Key words. hierarchical rank structure, backward stability, structured backward stability, error propagation, HSS algorithms, ULV factorization

AMS subject classifications. 65F05, 65F30

1. Introduction.

In the past decade, tremendous progress has been made on the development of rank structured matrix techniques. Many innovative structured algorithms have been proposed to solve linear systems and eigenvalue problems efficiently [3, 11, 21, 25, 26, 27, 29, 32]. In general, these algorithms have much lower complexity than classical matrix algorithms. One major class among these rank structures is the hierarchical rank structured matrix, such as $\mathcal{H}$-matrices [4, 15, 16, 18], $\mathcal{H}^2$-matrices [5, 17], and hierarchically semiseparable (HSS) matrices [6, 30]. In these matrix representations, a given matrix is hierarchically partitioned into appropriate blocks at multiple levels, and certain off-diagonal blocks are approximated by low-rank forms. In particular, HSS matrices have been used to develop efficient algorithms for both dense and sparse matrix problems [25, 29, 32, 33].

While the algorithms of rank structured matrices have been extensively studied, relatively little work has been done on their numerical stability. This is partially due to the complex nature of the algorithms, which usually involve a sequence of local operations. For instance, stability properties for quasiseparable matrices have not been analyzed until recently [2, 9]. In [9], two fast QR-based quasiseparable matrix algorithms are compared and the conclusion that one is stable and the other is not is made. In [2], another quasiseparable matrix method based on a nested product decomposition is proven to be backward stable. However, the analysis in [2, 9] does not provide actual error bounds in terms of the matrix size or the rank structure, and thus fails to disclose how the errors propagate in these algorithms. For hierarchical structured matrices, the stability analysis is even harder because the off-diagonal

*Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 55455, USA (yxi@cs.umn.edu)

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

blocks at different hierarchical levels are represented by nested products of many local dense matrices. The first backward stability result for hierarchical structured matrices appears in [25], where only the factorization process of HSS matrices is investigated. It still leaves many other algorithms not studied, such as the multiplication of the structured matrix with vectors, the solution of the structured linear systems, and the least squares solution.

1.1. Contributions. In this paper we present systematic stability analysis for various important HSS matrix algorithms, such as HSS matrix-vector multiplications, HSS factorization-based linear system solutions, HSS linear least squares solutions, HSS inversions, and some variations. Following [25], we express an HSS matrix A in a telescoping representation [23] and describe the HSS algorithms in an explicit way so as to ease the estimation of the rounding errors across different hierarchical levels. We first review the approximation error bound for the HSS construction and then assume the HSS form A is available and focus on the backward stability of various other HSS methods.

We start with the HSS matrix-vector multiplication since it serves as a basic component in several other HSS algorithms and is also often used in iterative solvers. We show that the error bound is $O(\mathbf{u}r^p \log^2 n)$, where p is a small constant and r is the HSS rank (See Definition 2.1). We continue to study HSS ULV-type solution algorithms [6, 25, 32] that are often used in structured dense and sparse matrix computations. The analysis of their variations, which take advantages of additional matrix properties (e.g., symmetric positive definiteness), is also provided. In order to exploit the structures in the factorization, we perform structured backward stability analysis in terms of the ULV factors. This kind of analysis helps to distinguish between different error propagation patterns resulting from different parts of the factors. We then generalize this to an HSS URV linear least squares solution method proposed in [25].

Besides HSS ULV/URV factorization-based algorithms, there are also some HSS inversion-based solution schemes [10, 11]. These schemes compute the HSS representation of A^{-1} and solve linear systems by HSS matrix-vector multiplications. They are often used in the fast solutions of certain types of integral equations and can take into consideration the physical properties of the underling problems. However, we show that the inversion algorithm in [11] is potentially unstable. Thus, factorization-based HSS solutions are preferred in practice.

Although the derivation of the stability results is tedious, it is convenient to understand them via the hierarchical tree structure. We can observe that all the error bounds are $\mathbf{u}$ magnified by factors that involve low-degree powers of r and $\log n$. That is, the hierarchical rank structure has significant benefits in not only the efficiency (as known before), but also the stability. For problems with small off-diagonal (numerical) ranks, the numerical errors in the hierarchical algorithms only propagate levelwise along the hierarchical tree structures by $O(\log^p n)$ for a small p. Thus, we can conclude that HSS methods in general are both faster and more stable than the corresponding standard dense matrix methods.

The analysis presented in this paper builds a comprehensive framework for the numerical stability of HSS methods. Meanwhile, the analysis can also greatly benefit the study of the numerical stability of other hierarchical rank structured algorithms, including both dense and sparse ones, since the hierarchical error propagations are similar. Our results also give new insights into the improvement of existing structured solvers (such as those based on sequentially semiseparable or quasiseparable structures [7, 8, 24]), as well as the design of new reliable structured algorithms.

1.2. Outline and notation. The structure of the paper is organized as follows. In Section 2, we briefly discuss some preliminaries and the numerical issues related to HSS approximations. Section 3 presents the stability results of HSS matrix-vector multiplications. Section 4 studies the structured stability of HSS ULV-type algorithms for linear system solutions. The stability of an HSS URV algorithm for linear least squares solutions is analyzed in Section 5. An HSS inversion algorithm is investigated in Section 6. We provide some numerical examples in Section 7 and draw some concluding remarks in Section 8.

The following notation is used throughout the paper:

- $\mathrm{fl}(\cdot)$ denotes the computed result in a floating point operation;
- the relative perturbations to basic arithmetic operations are all represented by $O(\mathbf{u})$;
- $A|_{\mathbf{I}\times\mathbf{J}}$ represents a submatrix of A with a row index set $\mathbf{I}$ and a column index set $\mathbf{J}$;
- $\mathrm{diag}(\cdots)$ represents a block diagonal matrix;
- $\mathcal{T}$ represents a full binary tree with nodes $i = 1, 2, \ldots, \mathrm{root}(\mathcal{T})$, where $\mathrm{root}(\mathcal{T})$ is the root of $\mathcal{T}$;
- $\mathrm{sib}(i)$ and $\mathrm{par}(i)$ are the sibling and the parent of a node i in $\mathcal{T}$, respectively;
- the inequality in the following form between two matrices A and B of the same shape is interpreted to hold componentwise:
$$|A| \le |B|;$$
- $\mathbf{u}$ is the unit roundoff or machine epsilon in IEEE double precision arithmetic;
- let $\gamma_r = \frac{r\mathbf{u}}{1-r\mathbf{u}}$ for an integer r and $\tilde{\gamma}_r = \frac{cr\mathbf{u}}{1-cr\mathbf{u}}$ for a small constant c, as frequently used in backward error analysis.

2. Preliminaries on HSS structures. We first give some preliminaries on HSS structures, and then discuss the construction accuracy in HSS construction. An HSS matrix is defined in a recursive way as in [6, 30, 32]. Here, we follow a slightly more general form for rectangular HSS matrices [25] as below.

DEFINITION 2.1. *An $m \times n$ matrix A is an HSS matrix if it satisfies the following conditions.*

- *There is a postordered full binary tree $\mathcal{T}$ associated with A, and $\mathrm{root}(\mathcal{T})$ is at level 0 and its leaves are at level L.*
- *There are a row index set $\mathbf{I}_i$ and a column index set $\mathbf{J}_i$ associated with each node i of $\mathcal{T}$ and defined recursively as*
$$\mathbf{I}_i = \mathbf{I}_{c_1} \cup \mathbf{I}_{c_2},\ \mathbf{I}_{c_1} \cap \mathbf{I}_{c_2} = \emptyset, \quad \mathbf{J}_i = \mathbf{J}_{c_1} \cup \mathbf{J}_{c_2},\ \mathbf{J}_{c_1} \cap \mathbf{J}_{c_2} = \emptyset,$$
where i is the parent of c_1 and c_2, and $\mathbf{I}_{\mathrm{root}(\mathcal{T})} \equiv \{1, 2, \ldots, m\}$, $\mathbf{J}_{\mathrm{root}(\mathcal{T})} \equiv \{1, 2, \ldots, n\}$.
- *There are matrices or generators D_i, U_i, V_i, etc. associated with each node i of $\mathcal{T}$ and defined recursively as*

$$D_i \equiv A|_{\mathbf{I}_i\times\mathbf{J}_i} = \begin{pmatrix} D_{c_1} & U_{c_1}B_{c_1}V_{c_2}^T \\ U_{c_2}B_{c_2}V_{c_1}^T & D_{c_2} \end{pmatrix}, \tag{2.1}$$

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

See Figure 2.1 for an example. When A is a square matrix, we can set $\mathbf{I}_i \equiv \mathbf{J}_i$.

Let

$$A_i^- = A|_{\mathbf{I}_i \times (\mathbf{J}_{\mathrm{root}(\mathcal{T})} \backslash \mathbf{J}_i)}, \quad A_i^| = A|_{(\mathbf{I}_{\mathrm{root}(\mathcal{T})} \backslash \mathbf{I}_i) \times \mathbf{J}_i}, \tag{2.3}$$

which are called the *HSS block rows* and *columns* associated with node i, respectively. The maximum numerical rank r of all the HSS block rows and columns is the *HSS rank* of A. We assume r is small or bounded.

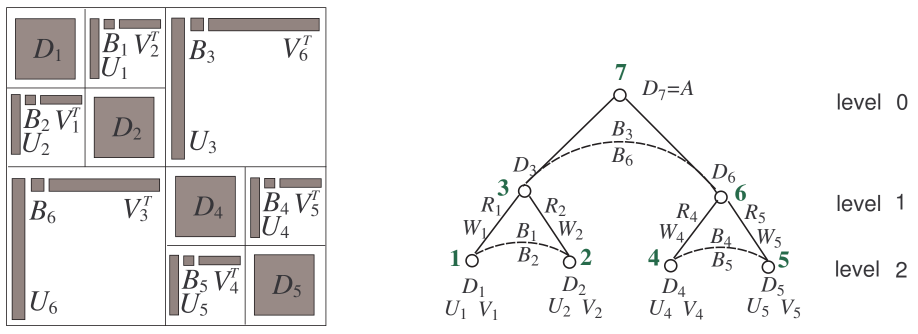


FIG. 2.1. *A* 3*-level HSS matrix and its corresponding HSS tree.*

To facilitate the stability analysis, we expand the recursive expression in (2.1) and write an HSS matrix A explicitly in the following telescoping form [23]:

$$\begin{aligned} A &= U^{(L)}(U^{(L-1)}(\cdots(U^{(1)}B^{(1)}(V^{(1)})^T + B^{(2)})\cdots) + B^{(L-1)})(V^{(L-1)})^T \\ &\quad + B^{(L)})(V^{(L)})^T + D^{(L)}, \end{aligned} \tag{2.4}$$

where $D^{(l)}$, $U^{(l)}$, $V^{(l)}$ and $B^{(l)}$ are block diagonal matrices defined in terms of the HSS generators as follows [25]:

$$\begin{aligned} &D^{(L)} = \mathrm{diag}(D_i,\ i\text{: all leaves}), \\ &U^{(l)} = \begin{cases} \mathrm{diag}(U_i,\ i\text{: all leaves}), & \text{if } l = L, \\ \mathrm{diag}\left(\begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix},\ c_1, c_2\text{: children of all } i \text{ at level } l\right), & \text{otherwise}, \end{cases} \\ &V^{(l)} = \begin{cases} \mathrm{diag}(V_i,\ i\text{: all leaves}), & \text{if } l = L, \\ \mathrm{diag}\left(\begin{pmatrix} W_{c_1} \\ W_{c_2} \end{pmatrix},\ c_1, c_2\text{: children of all } i \text{ at level } l\right), & \text{otherwise}, \end{cases} \\ &B^{(l)} = \mathrm{diag}\left(\begin{pmatrix} 0 & B_i \\ B_{\mathrm{sib}(i)} & 0 \end{pmatrix},\ i\text{: left nodes at level } l\right). \end{aligned} \tag{2.5}$$

See Figure 2.2 for an illustration. Also for simplicity, if a level l of the HSS tree has $k(l)$ nodes, we denote them by

$$l_1, l_2, \ldots, l_{k(l)}. \tag{2.6}$$

REMARK 2.1. Without loss of generality, we assume $m \geq n$, each diagonal block in both $U^{(l)}$ and $V^{(l)}$ has column size r, and each diagonal block in $D^{(L)}$ has column size $2r$ throughout the paper. This also means that the matrix A has HSS rank r.

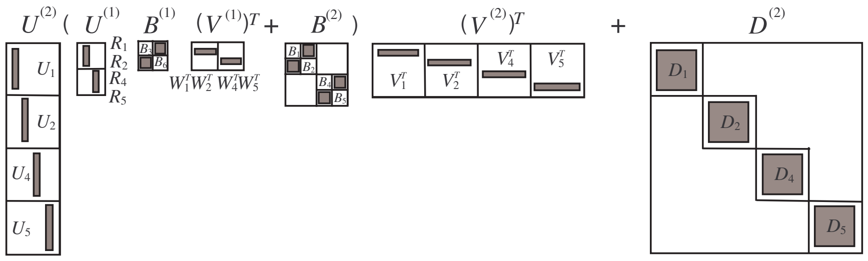


FIG. 2.2. *A telescoping representation [23] as in (2.4) for the HSS matrix in Figure 2.1.*

For a matrix whose off-diagonal blocks have small numerical ranks, it is beneficial to first compress those off-diagonal blocks and approximate this matrix by an HSS matrix representation so as to speed up further matrix operations. The following theorem measures the HSS approximation error in the Frobenius norm.

THEOREM 2.2. *(Corollary 4.3 [25]) Let A be an HSS approximation to $\mathcal{C} \in \mathbb{R}^{m\times n}$ from the standard HSS construction algorithm [30]. Under the assumption in Remark 2.1, the relative tolerance τ used in the truncation of the singular values σ_i of intermediate off-diagonal blocks satisfies $\sigma_{r+1} < \tau\sigma_1$ for a fixed r. Suppose the HSS tree has $L = O(\log(\min\{m, n\}))$ levels and the operations in the HSS construction procedure are performed in exact arithmetic, we then have*

$$\mathcal{C} = A + E, \tag{2.7}$$

where

$$||E||_F \leq 2\tau L\sqrt{2r}||\mathcal{C}||_F = O(\tau\sqrt{r}\log(\min\{m, n\}))||\mathcal{C}||_F.$$

Theorem 2.2 shows how the HSS approximation error can be controlled. It only depends on n polylogarithmically so that we can expect the accuracy to be well under control even for very large sizes. The tolerance τ corresponds to the trade-off between the approximation accuracy and the computational cost. It is known that the HSS approximation algorithm has complexity $O(rmn)$ [28], where the maximal off-diagonal numerical rank r generally increases as τ decreases. We want to emphasize that in some applications like eigenvalue computations [26] or preconditioning [31], a larger τ is sufficient to yield satisfactory results.

In the following sections, we assume that we have constructed an HSS matrix A and will focus on the numerical stability of various HSS matrix methods.

REMARK 2.2. The HSS generators U, V obtained from standard HSS construction algorithms [30] have orthonormal columns, which serves as a key feature to ensure the numerical stability of many HSS matrix algorithms. This will be further illustrated by the stability proofs in the remaining sections. On the other hand, randomized HSS construction algorithms [23, 25, 32] usually require less operations to construct an HSS matrix but may produce HSS generators U, V with non-orthonormal columns. However, the norms of these generators are bounded. Therefore, their numerical stability is similar to the orthogonal case except that the prefactors in their corresponding error bounds are larger. In this paper, we focus on the case where U, V have orthonormal columns.

In remaining sections, the stability analysis of various HSS algorithms is conducted for a computed HSS matrix rather than an exact one. Moreover, in order to distinguish between exact and computed quantities, wide-hatted notation is used to represent the corresponding computed results in floating point operations.

3. HSS matrix-vector multiplication. We first analyze the HSS matrix-vector multiplication algorithm in [6]. This algorithm can be considered as an algebraic form of the fast multipole method (FMM) [13] for 1D problems [21]. From (2.4), it is straightforward to derive this algorithm. For convenience, we describe the operations levelwise and introduce notation in Algorithm 1 for the multiplication of an HSS matrix A and a vector x. To facilitate the analysis, superscripts are used to distinguish intermediate vectors in the levelwise traversal. In practice, the storage should be reused.

Algorithm 1 HSS matrix-vector multiplication for $d = Ax$ (revised from [6])

```
1: procedure hssmv
2:     $x^{(L+1)} \leftarrow x$
3:     for level $l = L, L-1, \ldots, 1$ do
4:         $x^{(l)} \leftarrow \left(V^{(l)}\right)^T x^{(l+1)}$
5:         $t^{(l)} \leftarrow B^{(l)} x^{(l)}$
6:     end for
7:     $d^{(0)} \leftarrow t^{(1)}$
8:     for level $l = 1, 2, \ldots, L-1$ do
9:         $d^{(l)} \leftarrow U^{(l)} d^{(l-1)} + t^{(l+1)}$
10:    end for
11:    $d \leftarrow U^{(L)} d^{(L-1)} + D^{(L)} x$
12: end procedure
```

We can see that Algorithm 1 consists of frequent use of matrix-vector multiplications associated with HSS generators. Thus, we first review the rounding error for the standard matrix-vector multiplication in the following lemma.

Lemma 3.1. *[20] The numerical matrix-vector product for $N \in \mathbb{R}^{p\times r}$ and $y \in \mathbb{R}^r$ satisfies*

$$\mathrm{fl}(Ny) = (N + \Delta N)y, \quad |\Delta N| \le \gamma_r |N|, \quad ||\Delta N||_F \le \gamma_r ||N||_F.$$

For N with numerically orthonormal rows or columns, the following extension is obvious.

Lemma 3.2. *If the matrix N in Lemma 3.1 has numerically orthonormal rows or columns, the Frobenius norm bound then has a specific form*

$$||\Delta N||_F \le \sqrt{r}\gamma_r + o(\mathbf{u}).$$

Proof. Since the orthogonality of N only holds numerically, we have

$$||N||_F \le \sqrt{r} + O(\mathbf{u}).$$

Therefore, the following estimation holds for ΔN

$$||\Delta N||_F \le \gamma_r ||N||_F \le \gamma_r(\sqrt{r} + O(\mathbf{u})) = \sqrt{r}\gamma_r + o(\mathbf{u}).$$

□

The next lemma shows the relation between the norm of $\widehat{B}^{(l)}$ in (2.5) and $||\widehat{A}||_2$.

LEMMA 3.3. *Suppose the HSS tree associated with the computed HSS matrix $\widehat{A}$ has L levels and $\widehat{B}^{(l)}$ is defined as in (2.5). Then for $1 \le l \le L$,*

$$||\widehat{B}^{(l)}||_2 \le [1 + O(\mathbf{u})]||\widehat{A}||_2.$$

Proof. Following (2.5), we have

$$||\widehat{B}^{(l)}||_2 = \max_i \left\| \begin{pmatrix} 0 & \widehat{B}_i \\ \widehat{B}_{\mathrm{sib}(i)} & 0 \end{pmatrix} \right\|_2 = \max_i \{||\widehat{B}_i||_2, ||\widehat{B}_{\mathrm{sib}(i)}||_2\}.$$

Based on the interlacing property and the fact that $\widehat{U}_i \widehat{B}_i \widehat{V}^T_{\mathrm{sib}(i)}$ forms the submatrix $\widehat{A}|_{\mathbf{I}_i \times \mathbf{J}_{\mathrm{sib}(i)}}$ of $\widehat{A}$, we have

$$||\widehat{U}_i \widehat{B}_i \widehat{V}^T_{\mathrm{sib}(i)}||_2 \le ||\widehat{A}||_2.$$

Since $\widehat{U}_i$ and $\widehat{V}_{\mathrm{sib}(i)}$ have numerically orthonormal columns, we further have

$$||\widehat{B}_i||_2 \le [1 + O(\mathbf{u})]||\widehat{U}_i \widehat{B}_i \widehat{V}^T_{\mathrm{sib}(i)}||_2 \le [1 + O(\mathbf{u})]||\widehat{A}||_2.$$

Similarly, $||\widehat{B}_{\mathrm{sib}(i)}||_2 \le [1 + O(\mathbf{u})]||\widehat{A}||_2$. The result then follows. □

We then proceed to show that the HSS matrix-vector multiplication algorithm is backward stable by recursively applying Lemmas 3.1–3.3.

THEOREM 3.4. *The HSS matrix-vector multiplication Algorithm 1 is backward stable. That is, it produces a numerical result*

$$\mathrm{fl}(\widehat{A}x) = (\widehat{A} + \Delta A)x + o(\mathbf{u}),$$

where

$$||\Delta A||_2 = O((\sqrt{r}\log^2 n)\gamma_r)||\widehat{A}||_2.$$

Proof. Algorithm 1 involves one bottom-up and one top-down traversal of the HSS tree. In the bottom-up traversal (steps 3–6 of Algorithm 1), we multiply $(\widehat{V}^{(l)})^T$ and $\widehat{B}^{(l)}$ with some vectors. Due to the block diagonal structure of $\widehat{V}^{(l)}$, the following matrix-vector multiplication holds in the floating point operation:

$$\mathrm{fl}((\widehat{V}^{(l)})^T v) = (\widehat{V}^{(l)} + \Delta V^{(l)})^T v,$$

where $\Delta V^{(l)} = \mathrm{diag}\left(\Delta V_{l_1}, \ldots, \Delta V_{l_{k(l)}}\right)$ with the notation in (2.6), and

$$||\Delta V^{(l)}||_2 = \max_{l_j} ||\Delta V_{l_j}||_2 \le \max_{l_j} ||\Delta V_{l_j}||_F \le \sqrt{r}\gamma_r + o(\mathbf{u}).$$

The last inequality follows from Lemma 3.2 since $\widehat{V}_{l_j}$ have numerically orthonormal columns.

Thus, we obtain

$$\mathrm{fl}(x^{(l)}) = (\widehat{V}^{(l)} + \Delta V^{(l)})^T \, \mathrm{fl}(x^{(l+1)}), \quad ||\Delta V^{(l)}||_2 \le \sqrt{r}\gamma_r + o(\mathbf{u}),$$

for $l = L, L-1, \ldots, 1$. Similarly, we have

$$\mathrm{fl}(t^{(l)}) = (\widehat{B}^{(l)} + \Delta B^{(l)})\,\mathrm{fl}(x^{(l)}), \quad |\Delta B^{(l)}| \le \gamma_r |\widehat{B}^{(l)}|,$$

where $\Delta B^{(l)} = \mathrm{diag}\left(\begin{pmatrix} 0 & \Delta B_{l_j} \\ \Delta B_{l_{j+1}} & 0 \end{pmatrix}\right)$ for $l = L, L-1, \ldots, 1$.

The estimation of $||\Delta B^{(l)}||_2$ can also be simplified by its block diagonal structure:

$$\begin{aligned} ||\Delta B^{(l)}||_2 &= \max_{l_j} ||\Delta B_{l_j}||_2 \le \max_{l_j} ||\Delta B_{l_j}||_F \le \gamma_r \max_{l_j} ||\widehat{B}_{l_j}||_F \\ &\le \sqrt{r}\gamma_r \max_{l_j} ||\widehat{B}_{l_j}||_2 = \sqrt{r}\gamma_r ||\widehat{B}^{(l)}||_2 \le \sqrt{r}\gamma_r ||\widehat{A}||_2 + o(\mathbf{u}). \end{aligned}$$

The last inequality holds due to Lemma 3.3.

Therefore, we know that for a fixed l,

$$\mathrm{fl}(x^{(l)}) = \prod_{j=l}^{L} (\widehat{V}^{(j)} + \Delta V^{(j)})^T x = \Big(\prod_{j=l}^{L} (\widehat{V}^{(j)})^T + \Delta \widehat{V}_l\Big) x + o(\mathbf{u}), \tag{3.1}$$

$$\mathrm{fl}(t^{(l)}) = \Big(\widehat{B}^{(l)} \prod_{j=l}^{L} (\widehat{V}^{(j)})^T + \Delta \widehat{B}_l\Big) x + o(\mathbf{u}), \tag{3.2}$$

where

$$\Delta \widehat{V}_l = \sum_{j=l}^{L} (\widehat{V}^{(l)})^T \cdots (\widehat{V}^{(j-1)})^T (\Delta V^{(j)})^T (\widehat{V}^{(j+1)})^T \cdots (\widehat{V}^{(L)})^T,$$

$$\Delta \widehat{B}_l = \widehat{B}^{(l)} \Delta \widehat{V}_l + \Delta B^{(l)} \prod_{j=l}^{L} (\widehat{V}^{(j)})^T.$$

We also have the following bounds:

$$||\Delta \widehat{V}_l||_2 \le (L-l+1)\sqrt{r}\gamma_r + o(\mathbf{u}), \qquad ||\Delta \widehat{B}_l||_2 \le [(L-l+2)\sqrt{r}\gamma_r + o(\mathbf{u})]||\widehat{A}||_2.$$

In the top-down traversal of the HSS tree (steps 8–10 in Algorithm 1), we know that for $l = 1, 2, \ldots, L-1$,

$$\mathrm{fl}(d^{(l)}) = (I + \mathbf{u} Z_l)[(\widehat{U}^{(l)} + \Delta U^{(l)})\,\mathrm{fl}(d^{(l-1)}) + \mathrm{fl}(t^{(l+1)})],$$

where $|\Delta U^{(l)}| \le \gamma_r |\widehat{U}^{(l)}|$ and $|Z_l| \le I$.

Replacing $\mathrm{fl}(t^{(l+1)})$ on the right-hand side of the above equation with (3.2) yields

$$\begin{aligned} \mathrm{fl}(d^{(l)}) &= (\widehat{U}^{(l)} + \Delta U^{(l)})\,\mathrm{fl}(d^{(l-1)}) + \mathbf{u} Z_l \cdot \widehat{U}^{(l)} \cdot \mathrm{fl}(d^{(l-1)}) + \mathbf{u} Z_l\,\mathrm{fl}(t^{(l+1)}) \\ &\quad + \mathrm{fl}(t^{(l+1)}) + o(\mathbf{u}) \\ &= \Big(\widehat{B}^{(l+1)} \cdot \prod_{j=l+1}^{L} (\widehat{V}^{(j)})^T + \Delta B_{l+1} + \mathbf{u} Z_l \cdot \widehat{B}^{(l+1)} \cdot \prod_{j=l+1}^{L} (\widehat{V}^{(j)})^T\Big) x \\ &\quad + (\widehat{U}^{(l)} + \Delta U^{(l)} + \mathbf{u} Z_l \cdot \widehat{U}^{(l)})\,\mathrm{fl}(d^{(l-1)}) + o(\mathbf{u}). \end{aligned}$$

Expand the above recursion to get

$$\mathrm{fl}(d^{(l)}) = \left(\sum_{i=1}^{l} \Big(\prod_{k=l}^{i} \widehat{U}^{(k)} \cdot \widehat{B}^{(i)} \cdot \prod_{j=i}^{L} (\widehat{V}^{(j)})^T\Big) + \widehat{B}^{(l+1)} \cdot \prod_{j=l+1}^{L} (\widehat{V}^{(j)})^T + F^{(l)}\right) x + o(\mathbf{u}),$$

where

$$
\begin{aligned}
F^{(l)} = {}& \widehat{U}^{(l)} F^{(l-1)} + \mathbf{u} Z_l \cdot \widehat{B}^{(l+1)} \cdot \prod_{i=l+1}^{L} \big(\widehat{V}^{(i)}\big)^T + \Delta \widehat{B}_{l+1} \\
& + \Delta U^{(l)} \cdot \sum_{j=1}^{l-1} \Big(\prod_{k=l-1}^{j} \widehat{U}^{(k)} \cdot \widehat{B}^{(j)} \cdot \prod_{i=j}^{L} (\widehat{V}^{(i)})^T \Big) + \Delta U^{(l)} \cdot \widehat{B}^{(l)} \cdot \prod_{j=l}^{L} (\widehat{V}^{(j)})^T \\
& + \mathbf{u} Z_l \cdot \sum_{j=1}^{l} \Big(\prod_{k=l}^{j} \widehat{U}^{(k)} \cdot \widehat{B}^{(j)} \cdot \prod_{i=j}^{L} (\widehat{V}^{(i)})^T \Big), \\
& 2 \le l \le L-1, \\
F^{(1)} = {}& \widehat{U}^{(1)} \cdot \Delta \widehat{B}_1 + \Delta U^{(1)} \cdot \widehat{B}^{(1)} \cdot \prod_{i=1}^{L} \big(\widehat{V}^{(i)}\big)^T + \mathbf{u} Z_1 \cdot \widehat{U}^{(1)} \cdot \widehat{B}^{(1)} \cdot \prod_{i=1}^{L} (\widehat{V}^{(i)})^T \\
& + \mathbf{u} Z_1 \cdot \widehat{B}^{(2)} \cdot \prod_{i=2}^{L} (\widehat{V}^{(i)})^T + \Delta \widehat{B}_2.
\end{aligned}
$$

From Lemma 3.3, we also obtain the estimates

$$
\begin{aligned}
\|F^{(l)}\|_2 &\le \|F^{(l-1)}\|_2 + \big[(L+1)\sqrt{r}\gamma_r + \mathbf{u}\,(l+1)\big] \, \|\widehat{A}\|_2 \\
&\le [(l-1)(L+1)\sqrt{r}\gamma_r + \frac{\mathbf{u}}{2}(l-1)(l+4) + 2(L+1)\sqrt{r}\gamma_r + O(\mathbf{u})] \|\widehat{A}\|_2, \\
& \quad 2 \le l \le L-1, \\
\|F^{(1)}\|_2 &\le \big[2(L+1)\sqrt{r}\gamma_r + O(\mathbf{u})\big] \, \|\widehat{A}\|_2.
\end{aligned}
$$

In step 11, we get

$$
\begin{aligned}
\mathrm{fl}(d) &= (I + \mathbf{u} Z_L) \, [(\widehat{U}^{(L)} + \Delta U^{(L)}) \, \mathrm{fl}(d^{(L-1)}) + \mathrm{fl}(\widehat{D}^{(L)} x)] \\
&= (I + \mathbf{u} Z_L) \, [(\widehat{U}^{(L)} + \Delta U^{(L)}) \, \mathrm{fl}(d^{(L-1)}) + (\widehat{D}^{(L)} + \Delta D^{(L)}) x],
\end{aligned}
$$

where $|\Delta U^{(L)}| \le \gamma_r |\widehat{U}^{(L)}|$, $|\Delta D^{(L)}| \le \gamma_{2r} |\widehat{D}^{(L)}|$ and $|Z_L| \le I$. Then

$$
\begin{aligned}
\mathrm{fl}(\widehat{A}x) &= \mathrm{fl}(d) \\
&= (\widehat{U}^{(L)} + \Delta U^{(L)}) \, \mathrm{fl}(d^{(L-1)}) + (\widehat{D}^{(L)} + \Delta D^{(L)}) x \\
&\quad + \mathbf{u} Z_L \cdot \widehat{U}^{(L)} \cdot \mathrm{fl}(d^{(L-1)}) + \mathbf{u} Z_L \cdot \widehat{D}^{(L)} \cdot x + o(\mathbf{u}) \\
&= \left(\sum_{j=1}^{L} \Big(\prod_{k=L}^{j} \widehat{U}^{(k)} \cdot \widehat{B}^{(j)} \cdot \prod_{l=j}^{L} (\widehat{V}^{(l)})^T \Big) + \widehat{D}^{(L)} + \Delta A \right) x + o(\mathbf{u}) \\
&= (\widehat{A} + \Delta A) x + o(\mathbf{u}),
\end{aligned}
$$

where

$$
\begin{aligned}
\Delta A = {}& \Delta U^{(L)} \cdot \sum_{j=1}^{L-1} \Big(\prod_{k=L-1}^{j} \widehat{U}^{(k)} \cdot \widehat{B}^{(j)} \cdot \prod_{l=j}^{L} (\widehat{V}^{(l)})^T + \widehat{B}^{(L)} \cdot (\widehat{V}^{(L)})^T \Big) \\
& + \mathbf{u} Z_L \Big(\sum_{j=1}^{L-1} \Big(\prod_{k=L-1}^{j} \widehat{U}^{(k)} \cdot \widehat{B}^{(j)} \cdot \prod_{l=j}^{L} (\widehat{V}^{(l)})^T \Big) + \widehat{B}^{(L)} \cdot (\widehat{V}^{(L)})^T \Big) \\
& + \mathbf{u} Z_L \cdot \widehat{D}^{(L)} + \widehat{U}^{(L)} \cdot F^{(L-1)} + \Delta D^{(L)},
\end{aligned}
$$

and

$$||\Delta A||_2 = O(\sqrt{r}L^2\gamma_r)||\widehat{A}||_2 = O((\sqrt{r}\log^2 n)\gamma_r)||\widehat{A}||_2.$$

This completes the proof. □

Theorem 3.4 can be interpreted in an intuitive way as follows. The HSS matrix-vector multiplication Algorithm 1 consists of a bottom-up sweep and a top-down sweep of the HSS tree $\mathcal{T}$. The operations at the leaf level L introduce a rounding error bounded by $O(\sqrt{r}\gamma_r)||\widehat{A}||_2$. During the traversal of $\mathcal{T}$ from the leaf level to the root and then back to the leaf level, this error propagates up and down along the tree and is magnified by a factor of $O(L)$. Therefore, the contribution to the total error from the leaf level operations is bounded by $O(\sqrt{r}L\gamma_r)||\widehat{A}||_2$. The rounding errors incurred at other levels follow similar amplification patterns. When we sum them up, we obtain the error bound in Theorem 3.4.

It is interesting to compare the amplification factors in the error bounds in Lemma 3.1 for standard dense matrix-vector multiplications and Theorem 3.4 for HSS matrix-vector multiplications. The factor in front of $\mathbf{u}$ in Lemma 3.1 shows that the error propagation in the standard dense matrix-vector multiplication is proportional to the matrix size n. On the other hand, the HSS matrix-vector multiplication only amplifies $\mathbf{u}$ by $O(r\sqrt{r}\log^2 n)$. This is proportional to $\log^2 n$ for bounded r. Thus, the HSS multiplication exhibits much better numerical stability. This is due to the fact that the HSS method decouples the matrix into hierarchical low-rank forms following a tree structure. Any HSS operation corresponds to an HSS tree traversal process, and the error magnification is proportional to the length of the path traversed. Since the longest path in an HSS tree is $O(\log n)$, HSS algorithms only involve polylogarithmic error amplification.

4. HSS ULV-type algorithms for linear system solutions.

We then study the numerical stability of ULV-type algorithms for directly solving an HSS linear system

$$Ax = b, \tag{4.1}$$

where A is an $n \times n$ HSS matrix. (The rectangular least-squares case will be studied in the next section.) Both the classical HSS ULV algorithm and its variations are analyzed in this section.

4.1. General nonsymmetric case.

In general, HSS ULV-type algorithms can be separated into two disjoint stages: (1) HSS ULV factorizations and (2) HSS ULV solutions. In the ULV factorization [6], a diagonal block associated with each leaf is partially eliminated. The remaining blocks are merged into a reduced HSS form [29], which is factorized recursively. The stability for this factorization has been investigated in [25] for nonsymmetric HSS matrices. But the analysis of the ULV solution is not conducted in [25]. For the sake of completeness, we first briefly review the related results in [25] and then proceed to prove the numerical stability of the solution stage.

The following notation similar to those in [25] is used in the analysis:

- $A^{(l)}$ represents the extended reduced matrix after partially factorizing $A^{(l+1)}$ (with those eliminated blocks replaced by zeros), with $A^{(L)} \equiv A$;
- $\Psi^{(l)}$ represents the permutation matrix needed at level l to form the reduced matrix $A^{(l)}$;

- $\mathbf{G}^{(l)}$ represents the blocks eliminated at level l with appropriate zero blocks and permutations ($\Psi^{(l)}$);
- wide-tilded notation represents unitary matrices computed from Householder transformations.

See Figure 4.1 below for an illustration of the matrices and operations, and more details are given below. Following the notation in (2.6), for $l = L, L-1, \ldots, 1$, define $n \times n$ matrices

$$\begin{aligned} \mathbf{Q}^{(l)} &= \operatorname{diag}(Q_{l_1}, Q_{l_2}, \ldots, Q_{l_{k(l)}}, I)\Psi^{(l)}, \\ \mathbf{P}^{(l)} &= \operatorname{diag}(P_{l_1}, P_{l_2}, \ldots, P_{l_{k(l)}}, I)\Psi^{(l)}, \end{aligned} \tag{4.2}$$

where Q_{l_k} is the orthogonal matrix computed from the QL factorization of the HSS generator U_{l_k}, P_{l_k} is an orthogonal matrix used for the partial elimination of D_{l_k}, and the identity matrices in $\operatorname{diag}(\cdots)$ correspond to the reduced matrices. More specifically, the ULV factorization can be represented by

$$\begin{aligned} A^{(L)} &\equiv A, \\ A^{(l-1)} + \mathbf{G}^{(l)} &= (\mathbf{Q}^{(l)})^T A^{(l)} \mathbf{P}^{(l)}, \quad l = L, L-1, \ldots, 1. \end{aligned}$$

Expand the above recursion and obtain

$$\begin{aligned} A = \mathbf{Q}^{(L)}(\mathbf{Q}^{(L-1)}(\cdots(\mathbf{Q}^{(1)}(\mathbf{G}^{(0)}(\mathbf{P}^{(0)})^T + \mathbf{G}^{(1)})(\mathbf{P}^{(1)})^T) \\ + \cdots + \mathbf{G}^{(L-1)})(\mathbf{P}^{(L-1)})^T + \mathbf{G}^{(L)})(\mathbf{P}^{(L)})^T, \end{aligned} \tag{4.3}$$

where we further suppose an QL factorization is computed for the final reduced matrix:

$$A^{(0)} = \mathbf{G}^{(0)}(\mathbf{P}^{(0)})^T. \tag{4.4}$$

Thus, we can also include $l = 0$ for $\mathbf{P}^{(l)}$ in (4.2). The ULV factorization is illustrated in Figure 4.1 for an HSS matrix with $L = 2$.

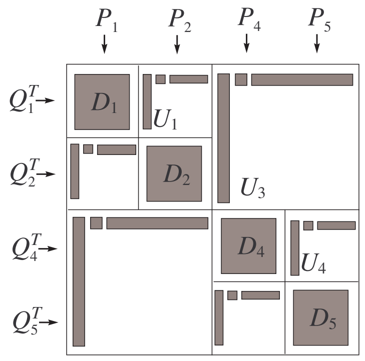
$\mathbf{G}^{(l)}$ represents the contribution of level-l eliminations to the overall ULV factors. Here, we organize the nonzero blocks within $\mathbf{G}^{(l)}$ in a way different from those in [6, 30]. We partition the blocks in $\mathbf{G}^{(l)}$ into two distinct parts: a block diagonal matrix with lower triangular matrices along its diagonal which we denote as $T^{(l)}$, and the factors right below $T^{(l)}$ which can actually be organized into an HSS form and are denoted as $H^{(l)}$. That is, we can write

$$\mathbf{G}^{(l)} = \begin{pmatrix} T^{(l)} & \\ H^{(l)} & 0 \end{pmatrix}. \tag{4.5}$$

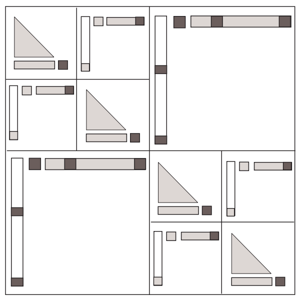
When $l = 0$, we suppose $\mathbf{G}^{(l)} \equiv T_0$ as produced in (4.4). See Figure 4.1(vi) for such a partition inside $\mathbf{G}^{(l)}$. After permutations, the $\mathbf{G}^{(l)}$ factors at all levels l can be assembled together into a structured lower triangular factor $\mathcal{L}$ in the ULV factorization. For example, for $L = 2$, we have

$$\mathcal{L} = \begin{pmatrix} T^{(2)} & \\ H^{(2)} & \begin{pmatrix} T^{(1)} & \\ H^{(1)} & T^{(0)} \end{pmatrix} \end{pmatrix},$$

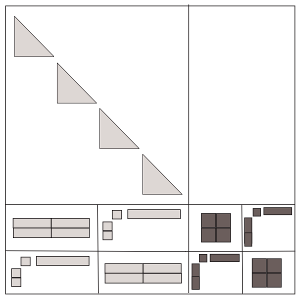
where $T^{(0)} \equiv \mathbf{G}^{(0)}$. Following the assumptions made in Remark 2.1, $T^{(l)}$ and $H^{(l)}$ are square matrices, and their row sizes are equal to $n/2$ when $l = L$ and are reduced by



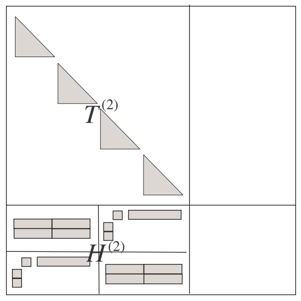
(i) Apply Q_i and P_i to $A^{(2)}$



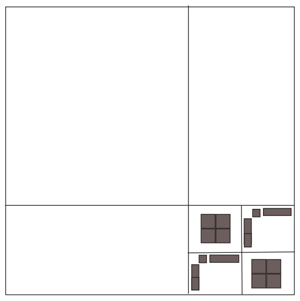
(ii) Nonzero pattern of $(\mathbf{Q}^{(2)})^T A^{(2)} \mathbf{P}^{(2)}$



(iii) Nonzero pattern of $(\mathbf{Q}^{(2)})^T A^{(2)} \mathbf{P}^{(2)}$ after permutations



(iv) Nonzero pattern of $\mathbf{G}^{(2)}$



(v) Nonzero pattern of $A^{(1)}$

FIG. 4.1. *Illustration of the ULV factorization of an HSS form with $L = 2$.*

half as l becomes $l-1$. With more levels, the structure of $\mathcal{L}$ can be seen from Figure 4.2.

Since the elimination of $T^{(l)}$ and $H^{(l)}$ involves different operations, we analyze their associated error propagation separately. Corollary 4.12 of [25] is adapted from Theorem 4.11 regarding the backward stability of HSS URV factorizations in [25] to study the backward stability of HSS ULV factorizations and is summarized as follows.

PROPOSITION 4.1. *[25, Corollary 4.12] The HSS ULV factorization in [6] yields*

$$\widehat{A} + \mathcal{F} = \widetilde{\mathbf{Q}}^{(L)}(\widetilde{\mathbf{Q}}^{(L-1)}(\cdots(\widetilde{\mathbf{Q}}^{(1)}(\widehat{\mathbf{G}}^{(0)}(\widetilde{\mathbf{P}}^{(0)})^T + \widehat{\mathbf{G}}^{(1)})(\widetilde{\mathbf{P}}^{(1)})^T)$$
$$+ \cdots + \widehat{\mathbf{G}}^{(L-1)})(\widetilde{\mathbf{P}}^{(L-1)})^T + \widehat{\mathbf{G}}^{(L)})(\widetilde{\mathbf{P}}^{(L)})^T,$$

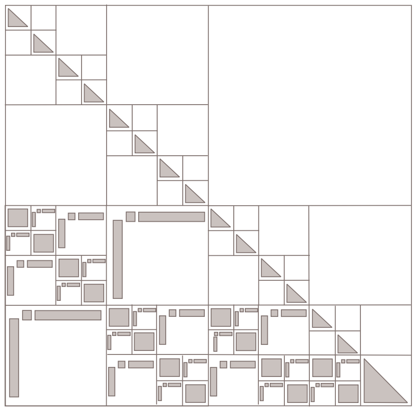


FIG. 4.2. *Illustration of the $\mathcal{L}$ factor in the ULV factorization of an HSS form with $L = 3$.*

where the backward error $\mathcal{F}$ satisfies

$$||\mathcal{F}||_F = O((r\log n)\tilde{\gamma}_r)||\widehat{A}||_F.$$

This stability result can be interpreted in a similar way as the HSS matrix-vector multiplication algorithm. Rounding errors introduced at each level of the HSS tree are bounded by $O(r\tilde{\gamma}_r)||\widehat{A}||_F$. Because the computed factors $\widetilde{\mathbf{Q}}^{(l)}$ and $\widetilde{\mathbf{P}}^{(l)}$ are orthogonal matrices, these rounding errors are not amplified during the HSS tree traversal. Thus, the overall rounding error is bounded by the sum of the rounding errors at each level, which is $O(rL\tilde{\gamma}_r)||\widehat{A}||_F$.

In the ULV solution stage, a sequence of orthogonal transformations and structured substitutions are performed. In order to utilize the stack data structure to save the storage, HSS ULV solution schemes in [6, 30, 32] follow a postordering traversal of the HSS tree $\mathcal{T}$. However, rounding errors associated with the nodes at the same level of $\mathcal{T}$ have the same structure. This makes it easier to analyze the overall backward error if we group the operations at each level together and rephrase the HSS ULV solution scheme levelwise based on (4.3). See Algorithm 2.

Notice that besides the orthogonal transformations (matrix-vector multiplications associated with $\mathbf{Q}^{(l)}$ and $\mathbf{P}^{(l)}$), the major operations in Algorithm 2 are the lower triangular solution step 6 and the right-hand side update step 7. This shows that the HSS ULV solution scheme essentially amounts to performing a block forward substitution levelwise/hierarchically with the structured form $\mathcal{L}$ (instead sequentially as in the standard lower triangular solution). See Figure 4.2 for illustration. Therefore, we first review the backward error analysis for the block forward substitution algorithm. The following lemma is a direct extension of [20, Theorem 8.5] and Lemma 3.1.

LEMMA 4.2. *Suppose a nonsingular lower triangular matrix $L \in \mathbb{R}^{n\times n}$ is partitioned into the following block 2×2 form:*

$$L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix}, \tag{4.6}$$

where the (1,1) block has size $r\times r$. When the linear system $Lx = t$ is solved by

Algorithm 2 HSS ULV solution scheme for solving $Ax = b$ (revised from [6, 30])

```
1: procedure hssulvsol
2:     $b^{(L+1)} \leftarrow b$
3:     for level $l = L, L-1, \ldots, 1$ do
4:         $b^{(l)} \leftarrow (\mathbf{Q}^{(l)})^T b^{(l+1)}$
5:         $\begin{pmatrix} b_1^{(l)} \\ b_2^{(l)} \end{pmatrix} \leftarrow b^{(l)}$        ▷ Conformable partition of $b^{(l)}$ following (4.5)
6:         Solve $T^{(l)} y_1^{(l)} = b_1^{(l)}$
7:         $b^{(l-1)} \leftarrow b_2^{(l)} - H^{(l)} y_1^{(l)}$
8:     end for
9:     Solve $T^{(0)} y_1^{(0)} = b^{(0)}$
10:    $x^T \leftarrow \begin{pmatrix} (y_1^{(L)})^T & \cdots & (y_1^{(0)})^T \end{pmatrix}$
11:    for level $l = 0, 1, \ldots, L$ do
12:        $x \leftarrow \mathbf{P}^{(l)} x$
13:    end for
14: end procedure
```

forward substitution, the computed solution $\widehat{x} = (\widehat{x}_1^T, \widehat{x}_2^T)^T$ *satisfies*

$$\begin{pmatrix} L_{11} + \Delta L_{11} & \\ L_{21} + \Delta L_{21} & L_{22} + \Delta L_{22} \end{pmatrix} \begin{pmatrix} \widehat{x}_1 \\ \widehat{x}_2 \end{pmatrix} = t,$$

where

$$||\Delta L_{11}||_F \le \gamma_r ||L_{11}||_F, \quad ||\Delta L_{21}||_F \le \gamma_r ||L_{21}||_F, \text{ and } ||\Delta L_{22}||_F \le \gamma_{n-r} ||L_{22}||_F.$$

The backward error ΔL_{11} results from triangular solutions and ΔL_{21} results from a matrix-vector multiplication. By relating (4.6) to the $\mathcal{L}$ factor (Figure 4.2) from the ULV factorization, we can treat the ULV solution process with $\mathcal{L}$ as a repeated block 2×2 lower triangular solution, except in a levelwise/hierarchical way. L_{11} and L_{21} in (4.6) correspond to $T^{(l)}$ and $H^{(l)}$ in (4.5), respectively, and L_{22} corresponds to the nonzero blocks in the extended reduced HSS matrix $A^{(l-1)}$. At each level l, only part of the solution corresponding to $T^{(l)}$ is computed and the remaining part needs to be computed at upper levels. We then derive the overall backward error for the HSS ULV solution scheme as follows. We will repeatedly use the error bounds similar to those for $||\Delta L_{11}||_F$ and $||\Delta L_{21}||_F$ in Lemma 4.2.

Theorem 4.3. *Suppose the ULV factors of a computed nonsingular HSS matrix* $\widehat{A} \in \mathbb{R}^{n \times n}$ *are given in (4.3). Then the ULV solution Algorithm 2 applied to the linear system (4.1) produces a computed solution* $\widehat{x}$ *satisfying*

$$\begin{aligned} (4.7) \qquad & (\widetilde{\mathbf{Q}}^{(L)} + \Delta\widetilde{Q}^{(L)})((\widetilde{\mathbf{Q}}^{(L-1)} + \Delta\widetilde{Q}^{(L-1)})(\cdots(\widetilde{\mathbf{Q}}^{(1)} + \Delta\widetilde{Q}^{(1)}) \\ & \cdot ((\widehat{\mathbf{G}}^{(0)} + \Delta G^{(0)})((\widetilde{\mathbf{P}}^{(0)})^T + \Delta\widetilde{P}^{(0)}) \\ & + \widehat{\mathbf{G}}^{(1)} + \Delta G^{(1)})((\widetilde{\mathbf{P}}^{(1)})^T + \Delta\widetilde{P}^{(1)}) + \cdots + \widehat{\mathbf{G}}^{(L-1)} + \Delta G^{(L-1)}) \\ & \cdot ((\widetilde{\mathbf{P}}^{(L-1)})^T + \Delta\widetilde{P}^{(L-1)}) + \widehat{\mathbf{G}}^{(L)} + \Delta G^{(L)})((\widetilde{\mathbf{P}}^{(L)})^T + \Delta\widetilde{P}^{(L)})\widehat{x} = b, \end{aligned}$$

where

$$||\Delta \widetilde{Q}^{(l)}||_2 \leq r\tilde{\gamma}_{2r} + o(\mathbf{u}), \quad ||\Delta \widetilde{P}^{(l)}||_2 \leq r\tilde{\gamma}_{2r} + o(\mathbf{u}), \tag{4.8}$$

$$\Delta G^{(l)} = \begin{pmatrix} \Delta T^{(l)} & \\ \Delta H^{(l)} & 0 \end{pmatrix} \textit{ with} \tag{4.9}$$

$$||\Delta T^{(l)}||_2 \leq \sqrt{r}\gamma_r ||\widehat{T}^{(l)}||_2, \quad ||\Delta H^{(l)}||_2 = O(l^2\sqrt{r}\gamma_r)||\widehat{H}^{(l)}||_2.$$

Proof. We follow the solution process. At level L, we solve for $y_1^{(L)}$ at step 6 in Algorithm 2, which is part of an overall solution process as follows:

$$(\mathbf{G}^{(L)} + A^{(L-1)})y^{(L)} = (\mathbf{Q}^{(L)})^T b, \tag{4.10}$$

where $y^{(L)} = \begin{pmatrix} y_1^{(L)} \\ y_2^{(L)} \end{pmatrix}$ and the portions $y_1^{(L)}$ and $y_2^{(L)}$ correspond to the nonzero blocks in $\mathbf{G}^{(L)}$ and $A^{(L-1)}$, respectively. The portion $y_1^{(L)}$ is computed at the current level L. The portion $y_2^{(L)}$ is to be computed at upper levels. Based on Lemma 4.2, we know that the computed solution $\widehat{y}^{(L)}$ at level L satisfies

$$(\widehat{\mathbf{G}}^{(L)} + \Delta G^{(L)} + \widehat{A}^{(L-1)} + \Delta A^{(L-1)}) \begin{pmatrix} \widehat{y}_1^{(L)} \\ \widehat{y}_2^{(L)} \end{pmatrix} = ((\widetilde{\mathbf{Q}}^{(L)})^T + \Delta Q^{(L)})b,$$

where $\Delta G^{(L)}$ is defined in (4.9) with l set to be L. Here, $\Delta Q^{(l)}$, $\Delta T^{(l)}$ and $\Delta H^{(l)}$ for $l = L$ represent the perturbation matrices associated with $\widetilde{\mathbf{Q}}^{(l)}$, $\widehat{T}^{(l)}$ and $\widehat{H}^{(l)}$, respectively, and

$$\Delta Q^{(l)} = \operatorname{diag}\left(\Delta Q_{l_1}, \ldots, \Delta Q_{l_{k(l)}}\right), \; \Delta T^{(l)} = \operatorname{diag}\left(\Delta T_{l_1}, \ldots, \Delta T_{l_{k(l)}}\right), \tag{4.11}$$

where the notation in (2.6) is used. $\widehat{A}^{(L-1)}$ is defined by the following recursive formula with l set to L:

$$\widehat{A}^{(l-1)} + \widehat{\mathbf{G}}^{(l)} = (\widetilde{\mathbf{Q}}^{(l)})^T \widehat{A}^{(l)} \widetilde{\mathbf{P}}^{(l)}, \quad \text{and} \quad \widehat{A}^{(L)} \equiv \widehat{A},$$

and $\Delta A^{(L-1)}$ denotes the rounding errors to be introduced by upper level operations when we compute $\widehat{y}_2^{(L)}$. Note that we do not need to find the explicit form of $\Delta A^{(L-1)}$ but the perturbations associated with the factors at upper levels.

We then derive bounds for $||\Delta Q^{(L)}||_2, ||\Delta T^{(L)}||_2$ and $||\Delta H^{(L)}||_2$. Since each diagonal block $\widetilde{Q}_{L_j}$ in $\widetilde{\mathbf{Q}}^{(L)}$ has size $2r \times 2r$ and is formed as the product of r Householder matrices [25, 30], according to [20, Lemma 19.3], we have

$$||\Delta Q^{(L)}||_2 = \max_j ||\Delta Q_{L_j}||_2 \leq \max_j ||\Delta Q_{L_j}||_F \leq r\tilde{\gamma}_{2r}. \tag{4.12}$$

The bound for $||\Delta T^{(L)}||_2$ can be estimated based on its block diagonal structure (with each diagonal block size $r \times r$) and Lemma 4.2. Let $\widehat{T}^{(L)} = \operatorname{diag}(\widehat{T}_{L_1}, \ldots, \widehat{T}_{L_k})$. Then

$$||\Delta T^{(L)}||_2 \leq \max_j ||\Delta T_{L_j}||_F \leq \gamma_r \max_j ||\widehat{T}_{L_j}||_F \leq \sqrt{r}\gamma_r ||\widehat{T}^{(L)}||_2. \tag{4.13}$$

The estimation of $||\Delta H^{(L)}||_2$ is a direct generalization of Theorem 3.4:

$$||\Delta H^{(L)}||_2 = O(L^2\sqrt{r}\gamma_r)||\widehat{A}||_2. \tag{4.14}$$

Next, we move one level up to level $L-1$. A portion of $\widehat{y}_2^{(L)}$ corresponding to $\widehat{T}^{(L-1)}$ can be computed. This is part of an overall solution process as follows:

$$\mathbf{Q}^{(L-1)}(\mathbf{G}^{(L-1)} + A^{(L-2)})y^{(L-1)} = (\mathbf{Q}^{(L)})^T b - \mathbf{G}^{(L)} y^{(L)},$$

where $y^{(L-1)} = \begin{pmatrix} 0 \\ y_1^{(L-1)} \\ y_2^{(L-1)} \end{pmatrix}$ and the portion represented by $y_2^{(L-1)}$ is computed at upper levels. Notice that the term $\mathbf{G}^{(L)}y^{(L)}$ is fully available since it is equal to $\begin{pmatrix} T^{(L)} \\ H^{(L)} \end{pmatrix} y_1^{(L)}$. The operations at level $L-1$ introduce the perturbations $\Delta Q^{(L-1)}$ and $\Delta G^{(L-1)}$, and the computed solution $\widehat{y}_1^{(L-1)}$ satisfies

$$\begin{aligned}
&(\widehat{\mathbf{G}}^{(L-1)} + \Delta G^{(L-1)} + \widehat{A}^{(L-2)} + \Delta A^{(L-2)}) \begin{pmatrix} 0 \\ \widehat{y}_1^{(L-1)} \\ \widehat{y}_2^{(L-1)} \end{pmatrix} \\
&= ((\widetilde{\mathbf{Q}}^{(L-1)})^T + \Delta Q^{(L-1)}) \left[((\widetilde{\mathbf{Q}}^{(L)})^T + \Delta Q^{(L)})b - (\widehat{\mathbf{G}}^{(L)} + \Delta G^{(L)}) \begin{pmatrix} \widehat{y}_1^{(L)} \\ 0 \end{pmatrix} \right],
\end{aligned}$$

where $\Delta G^{(L-1)}$ and $\Delta Q^{(L-1)}$ are defined as in (4.9) and (4.11) with $l = L-1$, respectively, and $\Delta A^{(L-2)}$ denotes the rounding errors to be introduced by operations from level $L-2$ up to level 0. The bounds for the errors $\Delta G^{(L-1)}$ and $\Delta Q^{(L-1)}$ at the current step are obtained similarly to those in (4.12)–(4.14).

This process is then repeated for the upper levels, and we obtain bounds for the errors introduced at each level. When level 0 is reached, the computed solution $\widehat{y}_1^{(0)}$ satisfies

(4.15)

$$\begin{aligned}
(\widehat{\mathbf{G}}^{(0)} + \Delta G^{(0)}) \begin{pmatrix} 0 \\ \widehat{y}_1^{(0)} \end{pmatrix} &= \prod_{l=1}^{L} ((\widetilde{\mathbf{Q}}^{(l)})^T + \Delta Q^{(l)}) b \\
&\quad - \sum_{l=1}^{L} \left[\Big(\prod_{j=1}^{l-1} ((\widetilde{\mathbf{Q}}^{(j)})^T + \Delta Q^{(j)}) \Big) (\widehat{\mathbf{G}}^{(l)} + \Delta G^{(l)}) \begin{pmatrix} 0 \\ \widehat{y}_1^{(l)} \\ 0 \end{pmatrix} \right],
\end{aligned}$$

where, for convenience, $\begin{pmatrix} 0 \\ \widehat{y}_1^{(l)} \\ 0 \end{pmatrix}$ with $l = L$ should be understood as $\begin{pmatrix} \widehat{y}_1^{(L)} \\ 0 \end{pmatrix}$. Notice $\widehat{G}^{(0)} = T^{(0)}$. (4.15) is then solved simply by forward substitution in step 9 of Algorithm 2, and Lemma 4.2 can be applied to obtain an error estimate similar to (4.13) for $\Delta G^{(0)}$.

Multiply $\Big(\prod_{l=1}^{L}((\widetilde{\mathbf{Q}}^{(l)})^T + \Delta Q^{(l)})\Big)^{-1}$ to (4.15) on the left to obtain

$$\begin{aligned}
(4.16) \qquad &\Big(\prod_{l=1}^{L} ((\widetilde{\mathbf{Q}}^{(l)})^T + \Delta Q^{(l)}) \Big)^{-1} (\widehat{\mathbf{G}}^{(0)} + \Delta G^{(0)}) \begin{pmatrix} 0 \\ \widehat{y}_1^{(0)} \end{pmatrix} \\
&+ \sum_{l=1}^{L} \left[\Big(\prod_{j=l}^{L} ((\widetilde{\mathbf{Q}}^{(j)})^T + \Delta Q^{(j)}) \Big)^{-1} (\widehat{\mathbf{G}}^{(l)} + \Delta G^{(l)}) \begin{pmatrix} 0 \\ \widehat{y}_1^{(l)} \\ 0 \end{pmatrix} \right] = b.
\end{aligned}$$

Let $\widehat{z} = \begin{pmatrix} (\widehat{y}_1^{(L)})^T & \cdots & (\widehat{y}_1^{(0)})^T \end{pmatrix}^T$. From steps 10 and 12 of Algorithm 2, the computed solution $\widehat{x}$ to the original HSS system is obtained by

$$\widehat{x} = \mathrm{fl}\Big(\prod_{l=L}^{0} \widetilde{\mathbf{P}}^{(l)} \widehat{z}\Big) = \prod_{l=L}^{0} ((\widetilde{\mathbf{P}}^{(l)}) + \Delta P^{(l)}) \widehat{z},$$

where the estimation of $||\Delta P^{(l)}||_2$ is similar to that of $||\Delta Q^{(l)}||_2$ in (4.12) since they have the same structure. This immediately leads to

$$\widehat{z} = \prod_{l=0}^{L} ((\widetilde{\mathbf{P}}^{(l)}) + \Delta P^{(l)})^{-1} \widehat{x} = \prod_{l=0}^{L} ((\widetilde{\mathbf{P}}^{(l)})^T + \Delta \widetilde{P}^{(l)}) \widehat{x},$$

where $||\Delta \widetilde{P}^{(l)}||_2$ satisfies the bound in (4.8). The estimation above is derived from the Neumann series expansion:

$$(\widetilde{\mathbf{P}}^{(l)} + \Delta P^{(l)})^{-1} = (\widetilde{\mathbf{P}}^{(l)})^T \underbrace{-(\widetilde{\mathbf{P}}^{(l)})^T \Delta P^{(l)} (\widetilde{\mathbf{P}}^{(l)})^T + \cdots}_{\Delta \widetilde{P}^{(l)}} \tag{4.17}$$

Due to the zero structures in $\widehat{\mathbf{G}}^{(l)}$ and $\Delta G^{(l)}$, we have

$$(\widehat{\mathbf{G}}^{(l)} + \Delta G^{(l)}) \begin{pmatrix} 0 \\ \widehat{y}_1^{(l)} \\ 0 \end{pmatrix} = (\widehat{\mathbf{G}}^{(l)} + \Delta G^{(l)}) \prod_{j=l}^{L} ((\widetilde{\mathbf{P}}^{(j)})^T + \Delta \widetilde{P}^{(j)}) \widehat{x}.$$

Thus, (4.16) can be rewritten in a nested form as below:

$$\begin{aligned} &((\widetilde{\mathbf{Q}}^{(L)})^T + \Delta Q^{(L)})^{-1} ((\widetilde{\mathbf{Q}}^{(L-1)})^T + \Delta Q^{(L-1)})^{-1} (\cdots ((\widetilde{\mathbf{Q}}^{(1)})^T + \Delta Q^{(1)})^{-1} \\ &\quad \cdot ((\widehat{\mathbf{G}}^{(0)} + \Delta G^{(0)})((\widetilde{\mathbf{P}}^{(0)})^T + \Delta \widetilde{P}^{(0)}) \\ &\quad + \widehat{\mathbf{G}}^{(1)} + \Delta G^{(1)})((\widetilde{\mathbf{P}}^{(1)})^T + \Delta \widetilde{P}^{(1)}) + \cdots + \widehat{\mathbf{G}}^{(L-1)} + \Delta G^{(L-1)}) \\ &\quad \cdot ((\widetilde{\mathbf{P}}^{(L-1)})^T + \Delta \widetilde{P}^{(L-1)}) + \widehat{\mathbf{G}}^{(L)} + \Delta G^{(L)})((\widetilde{\mathbf{P}}^{(L)})^T + \Delta \widetilde{P}^{(L)}) \widehat{x} = b. \end{aligned} \tag{4.18}$$

Expanding $((\widetilde{\mathbf{Q}}^{(l)})^T + \Delta Q^{(l)})^{-1}$ into a Neumann series similarly to (4.17) yields

$$((\widetilde{\mathbf{Q}}^{(l)})^T + \Delta Q^{(l)})^{-1} = \widetilde{\mathbf{Q}}^{(l)} + \Delta \widetilde{Q}^{(l)},$$

where $||\Delta \widetilde{Q}^{(l)}||_2$ satisfies the bound in (4.8). Then, (4.18) can be further simplified into (4.7), where $||\Delta \widetilde{Q}^{(l)}||_2$ and $||\Delta \widetilde{P}^{(l)}||_2$ satisfy the bounds in (4.8). This completes the proof. ∎

REMARK 4.1. Analogously to [1], Theorem 4.3 shows that the ULV solution scheme in Algorithm 2 is *structured numerically backward stable* in the sense that the computed $\widehat{x}$ is the exact solution to a nearby linear system with small perturbation to each computed factor. This is different from standard backward stability results for dense GEPP/QR methods [20]. This is more beneficial for structured matrix analysis since it reveals the error propagation associated with different parts of the factors.

If we premultiply (4.15) with $\prod_{l=1}^{L} \widetilde{\mathbf{Q}}^{(l)}$ in the proof of Theorem 4.3, we can obtain the following corollary, where the perturbation $\Delta \widetilde{Q}^{(l)}$ in Theorem 4.3 is removed and a new perturbation Δb is added.

COROLLARY 4.4. *With the same conditions and notation as in Theorem 4.3, the computed solution $\widehat{x}$ is the exact solution to*

$$(\widetilde{\mathbf{Q}}^{(L)}(\widetilde{\mathbf{Q}}^{(L-1)}(\cdots(\widetilde{\mathbf{Q}}^{(1)}((\widehat{\mathbf{G}}^{(0)}+\Delta G^{(0)})((\widetilde{\mathbf{P}}^{(0)})^T+\Delta\widetilde{P}^{(0)}) \\ +\widehat{\mathbf{G}}^{(1)}+\Delta G^{(1)})((\widetilde{\mathbf{P}}^{(1)})^T+\Delta\widetilde{P}^{(1)})\cdots)+\widehat{\mathbf{G}}^{(L-1)}+\Delta G^{(L-1)}) \\ ((\widetilde{\mathbf{P}}^{(L-1)})^T+\Delta\widetilde{P}^{(L-1)})+\widehat{\mathbf{G}}^{(L)}+\Delta G^{(L)})((\widetilde{\mathbf{P}}^{(L)})^T+\Delta\widetilde{P}^{(L)})\widehat{x}=b+\Delta b,$$

where

$$||\Delta b||_2\leq Lr\tilde{\gamma}_{2r}||b||_2,$$

and $||\Delta\widetilde{P}^{(l)}||_2, ||\Delta T^{(l)}||_2, ||\Delta H^{(l)}||_2$ *satisfy the same bounds as in Theorem 4.3.*

4.2. Symmetric cases. When A is symmetric, ULV algorithms as in [30, 31] can take advantage of the symmetry. In particular, if A is further symmetric and positive definite (SPD), the generalized HSS Cholesky factorization in [30] replaces the partial QR factorizations of the D generators by Cholesky factorizations. Following the notation in (4.2), we define $\mathbf{Q}^{(l)}$ as in (4.2) and

$$\mathbf{L}^{(l)}=\mathrm{diag}(S_{l_1},S_{l_2},\ldots,S_{l_{k(l)}},I)\Psi^{(l)},\ l=L,L-1,\ldots,1,$$

where S_{l_j} is the lower triangular matrix computed from the Cholesky factorization of HSS generators D_{l_j}. We are then able to express the generalized HSS Cholesky factorization as

$$A=\mathbf{L}^{(L)}\mathbf{Q}^{(L)}\cdots\mathbf{L}^{(1)}\mathbf{Q}^{(1)}\mathbf{L}^{(0)}(\mathbf{L}^{(0)})^T(\mathbf{Q}^{(1)})^T(\mathbf{L}^{(1)})^T\cdots(\mathbf{Q}^{(L)})^T(\mathbf{L}^{(L)})^T. \tag{4.19}$$

Notice that this form is slightly different from (4.3). The stability analysis is similar to that for the HSS ULV factorization in [25]. We skip the details and only concentrate on the stability of the solution stage.

After factorizing A into the form of (4.19), it becomes obvious that the linear system $Ax=b$ can be solved by orthogonal matrix-vector multiplications and back/forward substitutions. In the following theorem we show that the generalized HSS Cholesky solution is structured backward stable.

THEOREM 4.5. *Suppose the generalized HSS Cholesky factors of a computed SPD HSS matrix $\widehat{A}\in\mathbb{R}^{n\times n}$ are given in (4.19). Then HSS Cholesky solution algorithm produces a computed solution $\widehat{x}$ satisfying*

$$(\widehat{\mathbf{L}}^{(L)}+\Delta S^{(L)})(\widetilde{\mathbf{Q}}^{(L)}+\Delta Q^{(L)})\cdots(\widehat{\mathbf{L}}^{(0)}+\Delta S^{(0)})(\widehat{\mathbf{L}}^{(0)}+\Delta S^{(0)})^T \\ \cdots(\widetilde{\mathbf{Q}}^{(L)}+\Delta Q^{(L)})^T(\widehat{\mathbf{L}}^{(L)}+\Delta S^{(L)})^T\widehat{x}=b,$$

where

$$||\Delta S^{(l)}||_2\leq\sqrt{r}\gamma_r||\widehat{\mathbf{L}}^{(l)}||_2,\quad ||\Delta Q^{(l)}||_2\leq r\tilde{\gamma}_{2r}+o(\mathbf{u}).$$

Proof. The proof just relies on recursively applying estimates similar to (4.12) and (4.13) in the proof of Theorem 4.3. ◻

The HSS LDL factorization [26] for general symmetric HSS matrices can be analyzed similarly. The details are skipped here.

5. HSS linear least squares solutions. For a rectangular HSS matrix $A \in \mathbb{R}^{m\times n}$ $(m > n)$, an efficient HSS URV least squares solution algorithm is proposed in [25]. Similar to HSS ULV algorithms, the HSS URV factorization also consists of two stages: an HSS URV factorization and an HSS URV solution. The backward stability for the factorization stage has been addressed in [25]. In order to perform the stability analysis for the solution stage, we follow the proof in Section 4 and write the HSS URV factorization explicitly in the form of

$$\begin{aligned}A =&\mathbf{Q}^{(L)}(\mathbf{Q}^{(L-1)}(\cdots(\mathbf{Q}^{(1)}(\mathbf{Q}^{(0)}\mathbf{G}^{(0)}+\mathbf{G}^{(1)})(\mathbf{P}^{(1)})^T)+\cdots+\mathbf{G}^{(L-1)})\\ &(\mathbf{P}^{(L-1)})^T+\mathbf{G}^{(L)})(\mathbf{P}^{(L)})^T,\end{aligned} \tag{5.1}$$

where $\mathbf{Q}^{(l)} \in \mathbb{R}^{m\times m}$ and $\mathbf{P}^{(l)} \in \mathbb{R}^{n\times n}$ have forms similar to (4.2), and $\mathbf{G}^{(l)}$ are defined in the same way as at the beginning of Section 4 except that each diagonal block in $T^{(l)}$ is an upper triangular matrix and the final reduced matrix $A^{(0)}$ is now factored by QR factorization into the form of $A^{(0)} = \mathbf{Q}^{(0)}\mathbf{G}^{(0)}$.

The HSS URV solution scheme can be interpreted similarly to HSS ULV solutions. A rectangular HSS matrix is transformed by two-sided orthogonal transformations ($\mathbf{Q}^{(l)}$ and $\mathbf{P}^{(l)}$) into a structured block upper triangular matrix. This makes it convenient to perform the least squares solution.

REMARK 5.1. A size reduction strategy is proposed in [25] to improve the performance of HSS URV algorithms when $m \gg n$. This strategy reduces the row size m of rectangular HSS matrices to be as close to n as possible before URV factorization by introducing zeros into the D and U generators. If this process is applied, (5.1) should be modified by premultiplying A with a block diagonal matrix with orthogonal diagonal blocks. This does not affect the stability analysis.

The stability analysis for the HSS ULV solution can be modified to obtain a similar result for the HSS URV solution. The backward stability result is provided in the next theorem.

THEOREM 5.1. *Suppose the HSS URV factors of a computed rectangular HSS matrix $\widehat{A} \in \mathbb{R}^{m\times n}$ $(m \geq n)$ are given in (5.1). Then HSS URV least squares solution algorithm produces a computed solution $\widehat{x}$ that is the exact solution to*

$$\begin{aligned}\min_{\widehat{x}}||(b+\Delta b)-&\widetilde{\mathbf{Q}}^{(L)}(\widetilde{\mathbf{Q}}^{(L-1)}(\cdots(\widetilde{\mathbf{Q}}^{(1)}(\widetilde{\mathbf{Q}}^{(0)}(\widetilde{\mathbf{G}}^{(0)}+\Delta G^{(0)})+\widehat{\mathbf{G}}^{(1)}+\Delta G^{(1)})\\ &((\widetilde{\mathbf{P}}^{(1)})^T+\Delta\widetilde{P}^{(2)})+\cdots+\widehat{\mathbf{G}}^{(L-1)}+\Delta G^{(L-1)})((\widetilde{\mathbf{P}}^{(L-1)})^T\\ &\quad+\Delta\widetilde{P}^{(L-1)})+\widehat{\mathbf{G}}^{(L)}+\Delta\widehat{G}^{(L)})((\widetilde{\mathbf{P}}^{(L)})^T+\Delta\widetilde{P}^{(L)})\widehat{x}||_2,\end{aligned}$$

where

$$\begin{aligned}||\Delta b||_2 \leq Lr\tilde{\gamma}_{2r}||b||_2,&\quad ||\Delta\widetilde{P}^{(l)}||_2\leq r\tilde{\gamma}_{2r}+o(\mathbf{u})\\ \Delta G^{(l)}=&\begin{pmatrix} \Delta T^{(l)} & \\ \Delta H^{(l)} & 0\end{pmatrix} \textit{ with}\\ ||\Delta T^{(l)}||_2 \leq \sqrt{r}\gamma_r||\widehat{T}^{(l)}||_2,&\quad ||\Delta H^{(l)}||_2= O(l^2\sqrt{r}\gamma_r)||\widehat{H}^{(l)}||_2.\end{aligned}$$

6. HSS inversion algorithm for linear system solutions. Another type of direct HSS solvers is based on HSS inversion. The HSS inversion algorithm proposed in [11] recursively applies the Sherman-Morrison-Woodbury (SMW) formula to obtain the HSS generators of A^{-1} and then solves the linear system $Ax = b$ by HSS matrix-vector multiplications. This method is often used in the fast solutions of certain types of integral equations [11], since the discretized matrices in those problems are often well-conditioned. In addition, the inversion can take into consideration some physical properties of the underling problems.

However, this HSS inversion may suffer from potential numerical instability due to two main reasons. First, the U, V generators for A^{-1} may not have orthogonal columns or bounded norms. Thus, the numerical stability for $A^{-1}b$ is expected to be much worse than the case proved in Theorem 3.4. This issue can be fixed by the HSS recompression algorithm proposed in [28], which orthogonalizes the generators U, V for A^{-1} and results in a new set of orthogonal HSS generators. But this recompression algorithm is much more expensive compared with other HSS algorithms. The other reason is the stability issue of the SMW formula for general matrices. For example, intuitions suggest that instability likely happens when the diagonal blocks of A are not well conditioned, even if A itself is well-conditioned.

As a simple example, consider the solution of $Ax = b$ with

$$A = \begin{pmatrix} D_1 & U_1 B_1 V_2^T \\ U_2 B_2 V_1^T & D_2 \end{pmatrix}, \quad b \equiv (b_1, b_2, b_3, b_4)^T, \tag{6.1}$$

where

$$\begin{aligned} & D_1 = D_2 = \mathrm{diag}(\epsilon, 1), \ 0 < \epsilon < \mathbf{u}, \ B_1 = I, \\ & U_1 = U_2 = V_1 = V_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \\ & b_i \text{ has magnitude of } O(1). \end{aligned} \tag{6.2}$$

The matrix A is well conditioned, as shown below.

Lemma 6.1. *The matrix A defined in (6.1)–(6.2) is invertible and well conditioned.*

Proof. A is invertible since its determinant is $-2\epsilon - 1$. To find the condition number of A, we find the largest and the smallest eigenvalues of $A^T A$ as follows:

$$\frac{7}{2} + \epsilon + \frac{1}{2}\epsilon^2 \pm \frac{1}{2}(3 + \epsilon)\sqrt{\epsilon^2 - 2\epsilon + 5}.$$

Thus, the condition number of A is

$$\begin{aligned} \left(\frac{\frac{7}{2} + \epsilon + \frac{1}{2}\epsilon^2 + \frac{1}{2}(3+\epsilon)\sqrt{\epsilon^2 - 2\epsilon + 5}}{\frac{7}{2} + \epsilon + \frac{1}{2}\epsilon^2 - \frac{1}{2}(3+\epsilon)\sqrt{\epsilon^2 - 2\epsilon + 5}} \right)^{\frac{1}{2}} &= \frac{1 + 2\epsilon}{\frac{7}{2} + \epsilon + \frac{1}{2}\epsilon^2 - \frac{1}{2}(3+\epsilon)\sqrt{\epsilon^2 - 2\epsilon + 5}} \\ &\le \frac{1}{\frac{7}{2} - \frac{3}{2}\sqrt{5}} < 7. \end{aligned}$$

$\square$

We first consider the numerical solution with HSS inversion. The telescoping representation of A looks like:

$$A = D^{(1)} + U^{(1)} B^{(1)} (U^{(1)})^T,$$

where

$$\begin{aligned} D^{(1)} &= \operatorname{diag}\left(\begin{pmatrix} \epsilon & \\ & 1 \end{pmatrix}, \begin{pmatrix} \epsilon & \\ & 1 \end{pmatrix} \right), \\ U^{(1)} &= \operatorname{diag}\left(\begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right), \quad B^{(1)} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \end{aligned}$$

In HSS inversion, the SMW formula yields

$$A^{-1} = (D^{(1)})^{-1} - (D^{(1)})^{-1}U^{(1)}(\underbrace{(B^{(1)})^{-1} + (U^{(1)})^T(D^{(1)})^{-1}U^{(1)}}_{T})^{-1}(U^{(1)})^T(D^{(1)})^{-1}.$$

We then solve $Ax = b$ by multiplying A^{-1} and b through the following procedure.

1. Compute $\widehat{z} = \mathrm{fl}\left((D^{(1)})^{-1}b\right) = \begin{pmatrix} \frac{b_1}{\epsilon}(1+O(\mathbf{u})) & b_2 & \frac{b_3}{\epsilon}(1+O(\mathbf{u})) & b_4 \end{pmatrix}^T$.
2. Compute $\widehat{T} = \mathrm{fl}(T)$. In exact arithmetic,

$$\begin{aligned} T &= (B^{(1)})^{-1} + (U^{(1)})^T(D^{(1)})^{-1}U^{(1)} \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \begin{pmatrix} 1+\frac{1}{\epsilon} & \\ & 1+\frac{1}{\epsilon} \end{pmatrix} = \begin{pmatrix} 1+\frac{1}{\epsilon} & 1 \\ 1 & 1+\frac{1}{\epsilon} \end{pmatrix}. \end{aligned}$$

   Due to the floating point rounding error, $\mathrm{fl}(1+\frac{1}{\epsilon}) = \frac{1}{\epsilon}$. Thus, we get

$$\widehat{T} \equiv \mathrm{fl}(T) = \begin{pmatrix} \frac{1}{\epsilon}(1+O(\mathbf{u})) & 1 \\ 1 & \frac{1}{\epsilon}(1+O(\mathbf{u})) \end{pmatrix}.$$

3. Finally, compute $\widehat{x} = \mathrm{fl}\left(A^{-1}b\right) = \mathrm{fl}(\widehat{z} - (D^{(1)})^{-1}U^{(1)}\widehat{T}^{-1}(U^{(1)})^T\widehat{z})$. Denote

$$\widehat{t} = \mathrm{fl}((U^{(1)})^T\widehat{z}) = \begin{pmatrix} \frac{b_1}{\epsilon}(1+O(\mathbf{u})) \\ \frac{b_3}{\epsilon}(1+O(\mathbf{u})) \end{pmatrix}, \quad \widehat{v} = \mathrm{fl}(\widehat{T}^{-1}\widehat{t}) = \begin{pmatrix} b_1(1+O(\mathbf{u})) \\ b_3(1+O(\mathbf{u})) \end{pmatrix},$$

   where some terms are truncated since $\epsilon^2 < \mathbf{u}$. Then the computed solution $\widehat{x}$ equals

$$\widehat{x} = \mathrm{fl}(A^{-1}b) = \mathrm{fl}(\widehat{z} - (D^{(1)})^{-1}U^{(1)}\widehat{v}) = \begin{pmatrix} \frac{b_1 O(\mathbf{u})}{\epsilon} \\ (b_1+b_2)(1+O(\mathbf{u})) \\ \frac{b_3 O(\mathbf{u})}{\epsilon} \\ (b_3+b_4)(1+O(\mathbf{u})) \end{pmatrix}.$$

   We check the numerical stability by computing

$$A\widehat{x} = \begin{pmatrix} -(b_3+b_4) + \frac{b_3 O(\mathbf{u})}{\epsilon} + O(\mathbf{u}) \\ (b_1+b_2+b_3+b_4) - \frac{b_3 O(\mathbf{u})}{\epsilon} + O(\mathbf{u}) \\ -(b_1+b_2) + \frac{b_1 O(\mathbf{u})}{\epsilon} + O(\mathbf{u}) \\ (b_1+b_2+b_3+b_4) - \frac{b_1 O(\mathbf{u})}{\epsilon} + O(\mathbf{u}) \end{pmatrix},$$

   which is far away from b. Thus, HSS inversion produces a poor solution even if A is well conditioned.

On the other hand, if we apply HSS ULV-type algorithms to solve this problem, we can get computed solutions with residual norms $O(\mathbf{u})$. One ULV factorization works as follows.

1. Factorize $D^{(1)} = \mathbf{L}^{(1)}(\mathbf{L}^{(1)})^T$, where $\mathbf{L}^{(1)} = \operatorname{diag}\left(\begin{pmatrix} \sqrt{\epsilon} & \\ & 1 \end{pmatrix}, \begin{pmatrix} \sqrt{\epsilon} & \\ & 1 \end{pmatrix}\right)$. Since $D^{(1)}$ is a diagonal matrix, $\mathbf{L}^{(1)}$ can be computed with high relative accuracy.
2. Update off-diagonal by computing $\overline{U}_1 = \begin{pmatrix} \frac{1}{\sqrt{\epsilon}} & \\ & 1 \end{pmatrix} U_1 = \begin{pmatrix} \frac{1}{\sqrt{\epsilon}} \\ -1 \end{pmatrix}$, and find a Givens rotation matrix Q_1 from $\overline{U}_1$. Here,
$$\mathrm{fl}(Q_1) = \begin{pmatrix} \sqrt{\epsilon}(1+O(\mathbf{u})) & 1+O(\mathbf{u}) \\ 1+O(\mathbf{u}) & -\sqrt{\epsilon}(1+O(\mathbf{u})) \end{pmatrix}.$$
Apply Q_1^T to $\overline{U}_1$ to get $\widehat{U}_1 = \mathrm{fl}(Q_1^T\overline{U}_1) = \begin{pmatrix} 0 \\ \frac{1}{\sqrt{\epsilon}}(1+O(\mathbf{u})) \end{pmatrix}$.
3. Denote $\mathbf{Q}^{(1)} = \operatorname{diag}(Q_1, Q_1)\Psi^{(1)}$, and obtain
$$\begin{aligned}\widehat{A}^{(0)} &= \mathrm{fl}((\mathbf{Q}^{(1)})^T(\mathbf{L}^{(1)})^{-1}A(\mathbf{L}^{(1)})^{-T}\mathbf{Q}^{(1)}) \\ &= \operatorname{diag}\left(I, \begin{pmatrix} 1 & \frac{1}{\epsilon}(1+O(\mathbf{u})) \\ \frac{1}{\epsilon}(1+O(\mathbf{u})) & 1 \end{pmatrix}\right),\end{aligned}$$
where the ones on the diagonal are analytically obtained from the identity matrices due to the application of $(\mathbf{L}^{(1)})^{-1}$ and $(\mathbf{L}^{(1)})^{-T}$ to the diagonal blocks.
4. Update the right hand side as
$$\mathrm{fl}(\overline{b}) = \mathrm{fl}((\mathbf{Q}^{(1)})^T(\mathbf{L}^{(1)})^{-1}b) = \begin{pmatrix} (b_1+b_2)(1+O(\mathbf{u})) \\ (b_3+b_4)(1+O(\mathbf{u})) \\ \frac{b_1}{\sqrt{\epsilon}}(1+O(\mathbf{u})) \\ \frac{b_3}{\sqrt{\epsilon}}(1+O(\mathbf{u})) \end{pmatrix}.$$
5. Compute the numerical solution $\widehat{y}$ to $A^{(0)}y = \overline{b}$ and get
$$\widehat{y} = \begin{pmatrix} (b_1+b_2)(1+O(\mathbf{u})) \\ (b_3+b_4)(1+O(\mathbf{u})) \\ \sqrt{\epsilon}b_3(1+O(\mathbf{u})) \\ \sqrt{\epsilon}b_1(1+O(\mathbf{u})) \end{pmatrix}.$$
6. Solve $(\mathbf{Q}^{(1)})^T(\mathbf{L}^{(1)})^T x = \widehat{y}$ to get the numerical solution $\widehat{x}$ of $Ax = b$:
$$\widehat{x} = \begin{pmatrix} b_1+b_2+b_3+O(\mathbf{u}) \\ b_1+b_2+O(\mathbf{u}) \\ b_1+b_3+b_4+O(\mathbf{u}) \\ b_3+b_4+O(\mathbf{u}) \end{pmatrix}.$$

To check the computational accuracy, we can see that the residual is $||A\widehat{x} - b||_2 = O(\mathbf{u})$. This demonstrates the superior accuracy of the ULV solution method. Comparison between the HSS inversion and HSS ULV-type algorithms on more general matrices are provided in the next section.

7. Numerical experiments.

In this section, we present numerical experiments for testing the backward stability when solving $Cx = b$ with different HSS algorithms for C with small off-diagonal numerical ranks. In particular, the accuracies of various HSS ULV factorization algorithms are compared with that of the HSS inversion. For convenience, we use the following notation in the experiments:

- $\kappa_2(C)$: 2-norm condition number of matrix C;
- $\mathbf{r} = \frac{||C\widehat{x}-b||_2}{||C||_2||\widehat{x}||_2}$, where $\widehat{x}$ is the computed solution to $Cx = b$;
- $\mathbf{r}_1$: $\mathbf{r}$ when $\widehat{x}$ is computed with the HSS ULV algorithm in [6];
- $\mathbf{r}_2$: $\mathbf{r}$ when $\widehat{x}$ is computed with the generalized HSS Cholesky algorithm in [30];
- $\mathbf{r}_3$: $\mathbf{r}$ when $\widehat{x}$ is computed with the HSS LDL algorithm in [26];
- $\mathbf{r}_4$: $\mathbf{r}$ when $\widehat{x}$ is computed with the HSS inversion algorithm in [11].

Here, we use above relative residual $\mathbf{r}$ as in [2, 7]. This choice is justified in [20] to be around $1e{-}16$ in double precision for classical numerically backward stable solvers, which is independent of the condition number of test matrices.

The numerical experiments are performed in double precision and a relative tolerance $\tau = 10^{-15}$ is used in the HSS construction algorithm to make the approximation error roughly in the same order as $\mathbf{u}$. An exact solution $x = (1, 1 \ldots, 1)^T$ is chosen and the right hand side $b = \mathrm{fl}(Cx)$ is computed with unstructured matrix-vector multiplication. We fix the leaf level HSS block row size to be 80.

EXAMPLE 1. Consider a test matrix C defined by

$$C = \lambda^n I + H_n + H_n|_{\mathbf{I}\times\mathbf{I}},$$

where $\lambda = 0.994$, $H_n = (\frac{1}{i+j-1})_{n\times n}$ is the Hilbert matrix, and $\mathbf{I} = \{n, n-1, \ldots, 1\}$.

This test matrix is frequently used in the backward stability analysis for quasiseparable matrices [2, 7]. It is SPD and has small off-diagonal numerical ranks. We can test the HSS solutions algorithms and provide a comprehensive comparison. The numerical results are reported in Table 7.1.

TABLE 7.1
Relative residuals from various HSS algorithms for the matrix C in Example 1.

n	$\kappa_2(C)$	ULV factorization-based			Inversion-based
		$\mathbf{r}_1$	$\mathbf{r}_2$	$\mathbf{r}_3$	$\mathbf{r}_4$
1000	$1.31e3$	$1.12e-15$	$1.14e-15$	$1.11-15$	$6.27e-14$
1500	$2.66e4$	$1.78e-15$	$2.01e-15$	$1.66-15$	$6.75e-13$
2000	$5.40e5$	$1.78e-15$	$1.68e-15$	$1.81-15$	$1.48e-11$
2500	$1.10e7$	$2.18e-15$	$2.26e-15$	$2.18-15$	$5.15e-10$
3000	$2.23e8$	$1.98e-15$	$2.00e-15$	$1.98-15$	$2.76e-8$
3500	$4.52e9$	$2.03e-15$	$2.10e-15$	$1.99-15$	$6.57e-9$
4000	$9.16e10$	$2.64e-15$	$2.84e-15$	$2.72-15$	$2.32e-8$
4500	$1.86e12$	$3.12e-15$	$3.34e-15$	$3.29-15$	$1.23e-7$

We can see from Table 7.1 that, as we increase the size n of the matrix C, the condition number $\kappa_2(C)$ increases accordingly. In the extreme case when $n = 4500$, $\kappa_2(C) = 1.86e12$ and C is very ill conditioned. With regard to the computed residuals, $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$ from HSS ULV-type algorithms retain orders of $1e{-}15$ while $\mathbf{r}_4$ from the HSS inversion algorithm increases from $6.24e{-}14$ to only $1.23e{-}7$, losing 7 digits of accuracy. The results in Table 7.1 are consistent with our structured backward stability conclusion in Section 4 for HSS ULV-type algorithms and the instability in Section 6 for the HSS inversion algorithm.

EXAMPLE 2. Consider a severely ill-conditioned nonsymmetric Cauchy matrix C defined by

$$C = \left(\frac{1}{u_i + v_i} \right)_{n \times n},$$

where the vectors u and v are generated by the MATLAB function `randn`.

The condition numbers of these Cauchy matrices are in the orders of 10^{22}, which causes severe numerical instability for many direct solvers. Here, we only test the HSS ULV algorithm in [6] and report the numerical results in Table 7.2 since other ULV algorithms do not work for the nonsymmetric matrix. The relative residual $\mathbf{r}_1$ still maintains an order of $1e-17$ for all the tests.

TABLE 7.2
Relative residuals computed by the HSS ULV solution for the matrix C in Example 2.

n	1000	1500	2000	2500	3000	3500	4000	4500
$\kappa_2(C)$	$1.06e22$	$4.65e24$	$9.67e23$	$1.93e23$	$2.85e23$	$4.17e22$	$1.61e24$	$2.43e23$
$\mathbf{r}_1$	$3.75e-17$	$3.49e-17$	$3.87e-18$	$1.33-17$	$2.11e-18$	$1.45-17$	$4.77-18$	$5.99e-18$

8. Conclusion. This paper provides systematic stability analysis for a series of important HSS algorithms. We give rigorous justifications of the numerical error propagation during the algorithms. The hierarchical rank structure plays a key role in the stability. This provides insights into how we can improve the stability of some existing algorithms based on quasiseparable or sequentially semiseparable algorithms. We also show that ULV factorization-based HSS solutions are generally more accurate that an HSS inversion based one. The analysis is justified with some numerical examples and reminds readers that the stability of HSS algorithms should not be taken as granted. The derivation here can greatly benefit the study of the numerical stability of other structured matrices and even structured sparse direct solvers.

Acknowledgments. The authors would like to thank Ming Gu and Huaian Diao for some discussions. The authors also thank the anonymous referees for valuable comments and suggestions which led to a significant improvement of the presentation.

REFERENCES

[1] Z. BAI, C.-R. LEE, R.-C. LI, AND S. XU, *Stable solutions of linear systems involving long chain of matrix multiplications*, Linear Algebra Appl., 435 (2010), pp. 659–673.
[2] T. BELLA, V. OLSHEVSKY, AND M. STEWART, *Nested product decomposition of quasiseparable matrices*, SIAM. J. Matrix Anal. Appl., 34 (2013), pp. 1520–1555.
[3] D.A.BINI, P. BOITO, Y. EIDELMAN, L. GEMIGNANI AND I. GOHBERG, *A fast implicit QR eigenvalue algorithm for companion matrices*, Linear Algebra Appl. 432 (2010), pp. 2006–2031.
[4] S. BÖRM, L. GRASEDYCK, AND W. HACKBUSCH, *Introduction to hierarchical matrices with applications*, Eng. Anal. Bound. Elem, 27 (2003), pp. 405–422.
[5] S. BÖRM, *Construction of data-sparse $\mathcal{H}^2$-matrices by hierarchical compression*, SIAM J. Sci. Comput., 31 (2009), pp. 1820–1839.
[6] S. CHANDRASEKARAN, P. DEWILDE, M. GU, AND T. PALS, *A fast ULV decomposition solver for hierarchically semiseparable representations*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 603–622.
[7] S. CHANDRASEKARAN, P. DEWILDE, M. GU, T. PALS, X. SUN, A.-J. VAN DER VEEN, AND D. WHITE, *Some fast algorithms for sequentially semiseparable representations*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 341–364.

[8] Y. Eidelman and I. Gohberg, *On a new class of structured matrices*, Integral Equations Operator Theory, 34 (1999) pp. 293–324.

[9] F. M. Dopico, V. Olshevsky, and P. Zhlobich, *Stability of QR-based fast system solvers for a subclass of quasiseparable rank one matrices*, Math. Comp. 82 (2013), pp. 2007–2034.

[10] 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), pp. A2023–A2046.

[11] A. Gillman, P. Young, and P. G. Martinsson, *A direct solver with $O(N)$ complexity for integral equations on one-dimensional domains*, Front. Math. China, 7 (2012), pp. 217–247.

[12] G. H. Golub and C. V. Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[13] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[14] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong-rank revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869.

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

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

[18] W. Hackbusch, *Hierarchische Matrizen: Algorithmen und Analysis*, Springer, Berlin, 2009.

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

[20] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, second ed., 2002.

[21] K. L. Ho and L. Greengard, *A fast direct solver for structured linear systems by recursive skeletonization*, SIAM J. Sci. Comput., 34 (2012), pp. 2507–2532.

[22] W. Lyons, *Fast Algorithms with Applications to PDEs*, PhD thesis, University of California Santa Barbara, USA, 2005.

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

[24] R. Vandebril, M. Van Barel, G. Golub, and N. Mastronardi, *A bibliography on semiseparable matrices*, CALCOLO, 42 (2005), pp. 249–270.

[25] Y. Xi, J. Xia, S. Cauley, and V. Balakrishnan, *Superfast and stable structured solvers for Toeplitz least squares via randomized sampling*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 44–72.

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

[27] Y. Xi, R. Li, and Y. Saad, *An algebraic multilevel preconditioner with low-rank corrections for sparse symmetric matrices*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 235–259.

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

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

[30] 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.

[31] 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.

[32] 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.

[33] J. Xia, Y. Xi, S. Cauley, and V. Balakrishnan, *Fast sparse selected inversion*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 1283–1314.