SIAM J. Matrix Anal. Appl.
Vol. 35, No. 1, pp. 44–72

SUPERFAST AND STABLE STRUCTURED SOLVERS FOR TOEPLITZ LEAST SQUARES VIA RANDOMIZED SAMPLING*

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

Abstract. We present some superfast ($O((m+n)\log^2(m+n))$ complexity) and stable structured direct solvers for $m \times n$ Toeplitz least squares problems. Based on the displacement equation, a Toeplitz matrix T is first transformed into a Cauchy-like matrix $\mathcal{C}$, which can be shown to have small off-diagonal numerical ranks when the diagonal blocks are rectangular. We generalize standard hierarchically semiseparable (HSS) matrix representations to rectangular ones, and construct a rectangular HSS approximation to $\mathcal{C}$ in nearly linear complexity with randomized sampling and fast multiplications of $\mathcal{C}$ with vectors. A new URV HSS factorization and a URV HSS solution are designed for the least squares solution. We also present two structured normal equation methods. Systematic error and stability analysis for our HSS methods is given, which is also useful for studying other HSS and rank structured methods. We derive the growth factors and the backward error bounds in the HSS factorizations, and show that the stability results are generally much better than those in dense LU factorizations with partial pivoting. Such analysis has not been done before for HSS matrices. The solvers are tested on various classical Toeplitz examples ranging from well-conditioned to highly ill-conditioned ones. Comparisons with some recent fast and superfast solvers are given. Our new methods are generally much faster, and give better (or at least comparable) accuracies, especially for ill-conditioned problems.

Key words. Toeplitz least squares, superfast and stable solvers, randomized sampling, rectangular HSS matrix, URV factorization, HSS error and stability analysis

AMS subject classifications. 65F05, 65F30, 15A06

DOI. 10.1137/120895755

1. Introduction.

Toeplitz least squares problems arise frequently in practical applications such as signal and image processing [26]. Consider a Toeplitz least squares problem in the following form:

$$\min_x \|Tx-b\|_2, \quad T = \begin{pmatrix} t_0 & t_{-1} & \cdots & t_{-(n-1)} \\ t_1 & t_0 & \ddots & t_{-(n-2)} \\ \vdots & t_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ t_{m-1} & t_{m-2} & \ddots & t_{m-n} \end{pmatrix}, \tag{1.1}$$

where $T \in \mathbb{C}^{m\times n}$ is a Toeplitz matrix with $m \geq n$, and $b \in \mathbb{C}^m$. That is, if the (j,k) element of T is denoted by $T_{j,k}$, then we have $T_{j,k} = T_{j+1,k+1}$ for $j = 1, 2, \ldots, m-1$ and $k = 1, 2, \ldots, n-1$.

*Received by the editors October 19, 2012; accepted for publication (in revised form) by N. Mastronardi December 4, 2013; published electronically January 21, 2014.
http://www.siam.org/journals/simax/35-1/89575.html

†Department of Mathematics, Purdue University, West Lafayette, IN 47907 (yxi@math.purdue.edu, xiaj@math.purdue.edu). The research of the second author was supported in part by an NSF CAREER Award DMS-1255416 and NSF grants DMS-1115572 and CHE-0957024.

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

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

1.1. Displacement structures. Our methods are based on displacement equations. The idea of displacement structures is first proposed in [20], and some further investigations and generalizations are made in [4, 10, 17, 18, 27]. The Toeplitz matrix T satisfies the following generalized Sylvester-type displacement equation [13]:

$$Z_m^{(1)}T - TZ_n^{(\delta)} = GH^*, \tag{1.2}$$

where $Z_n^{(\delta)} = \left(\begin{smallmatrix} 0 & \delta \\ I_{n-1} & 0 \end{smallmatrix}\right)$ with I_{n-1} the identity matrix of size $n-1$, $Z_m^{(1)}$ is defined following $Z_n^{(\delta)}$, and $G \in \mathbb{C}^{m\times 2}$, $H \in \mathbb{C}^{n\times 2}$. The choice of δ is mentioned after (1.7) below. The matrices G and H can be explicitly written down based on the Toeplitz vector $t_{-(n-1):(m-1)}$ since $Z_m^{(1)}$ and $Z_n^{(\delta)}$ are just shifting operators to make the left-hand side of (1.2) a rank-2 matrix. Clearly, $Z_n^{(\delta)}$ can be diagonalized by a normalized inverse discrete Fourier transform matrix

$$\mathcal{F}_n = \frac{1}{\sqrt{n}}(\omega_n^{(j-1)(k-1)})_{1\le j,k\le n}, \quad \omega_n = e^{\frac{2\pi \mathbf{i}}{n}}, \ \mathbf{i} = \sqrt{-1}. \tag{1.3}$$

That is, T can be transformed into a *Cauchy-like matrix* $\mathcal{C}$:

$$\mathcal{C} = \mathcal{F}_m T \mathcal{D}^{-1} \mathcal{F}_n^*, \tag{1.4}$$

where

$$\mathcal{D} = \operatorname{diag}\left(1, \delta^{\frac{1}{n}}, \delta^{\frac{2}{n}}, \ldots, \delta^{\frac{n-1}{n}}\right). \tag{1.5}$$

(The notation diag() means a diagonal matrix and is defined at the end of section 1.3.) (1.2) can then be transformed into another displacement equation [13]:

$$\Phi\mathcal{C} - \mathcal{C}\Lambda = \hat{G}\hat{H}^*, \tag{1.6}$$

where

$$\hat{G} = \mathcal{F}_m G \equiv (\hat{g}_1, \ldots, \hat{g}_m)^*, \qquad \hat{H} = \mathcal{F}_n \mathcal{D} H \equiv (\hat{h}_1, \ldots, \hat{h}_n)^*,$$
$$\Phi = \operatorname{diag}(1, \omega_m, \omega_m^2, \ldots, \omega_m^{m-1}), \qquad \Lambda = \delta^{\frac{1}{n}} \operatorname{diag}(1, \omega_n, \omega_n^2, \ldots, \omega_n^{n-1}).$$

An entry of $\mathcal{C}$ looks like

$$\mathcal{C}_{j,k} = \frac{\hat{g}_j^* h_k}{\omega_m^j - \delta^{\frac{1}{n}}\omega_n^k}. \tag{1.7}$$

(1.4) is used to perform the fast multiplication of $\mathcal{C}$ with vectors, and (1.7) is used to form selected (about $O(m+n)$) entries of $\mathcal{C}$, as needed later. Traditionally, choose δ to have unit modulus. For the stability purpose, we seek to maximize the minimal value of the denominator in (1.7), especially when $j = k$ [13, 31]. In [31], it is proven that the optimal value of δ is $e^{\mathbf{i}\pi\frac{\gcd(m,n)}{m}}$.

Thus, the least squares problem (1.1) can be converted into a Cauchy-like one:

$$\min_{\tilde{x}} ||\mathcal{C}\tilde{x} - \tilde{b}||_2, \tag{1.8}$$

where $\tilde{x} = \mathcal{F}_n \mathcal{D} x$ and $\tilde{b} = \mathcal{F}_m b$.

Algorithms that solve (1.1) with $O(mn)$ flops are called *fast* algorithms and algorithms with roughly $O(m+n)$ flops are called *superfast* ones. For the Toeplitz linear

system case with $m = n$, some superfast direct solvers based on low-rank structures are developed in [8, 25, 38]. That is, the off-diagonal blocks of $\mathcal{C}$ are shown to have small numerical ranks bounded by $O(\log n)$ [8, 25, 29, 38]. This property enables the approximation of $\mathcal{C}$ with data-sparse structured matrices such as sequentially semiseparable matrices [7] as in [8] and *hierarchically semiseparable* (HSS) matrices [6, 39] as in [38]. In [38], randomized sampling is used to quickly construct an HSS approximation to $\mathcal{C}$ with fast Toeplitz matrix-vector multiplications.

1.2. Main results. Here, we work on the least squares case where $\mathcal{C}$ is a rectangular matrix. We partition $\mathcal{C}$ into diagonal and off-diagonal blocks. Unlike in classical structured matrix methods, the diagonal blocks we use are rectangular and do not necessarily contain the diagonal entries $\mathcal{C}_{j,j}$. Then similarly to the square matrix case, we can show that the rectangular $\mathcal{C}$ has off-diagonal numerical ranks bounded by $O(\log(m+n))$ (independent of δ since (1.2) and (1.7) hold independent of δ). We thus define a rectangular HSS representation and generalize the algorithms in [24, 38] to construct a rectangular HSS approximation to $\mathcal{C}$ or a square HSS approximation to $\mathcal{C}^*\mathcal{C}$. This enables quick solutions of (1.2). We take advantage of the fast multiplication of $\mathcal{C}$ (or $\mathcal{C}^*\mathcal{C}$) with vectors in the hierarchical compression of the off-diagonal blocks of $\mathcal{C}$ (or $\mathcal{C}^*\mathcal{C}$) based on randomized sampling like in [24, 38]. Similarly to the case in [38], additional structures are built into the HSS approximations.

Three superfast schemes are then developed. One computes directly a rectangular HSS approximation to $\mathcal{C}$ with randomized sampling. Then a URV factorization algorithm and a row size reduction strategy are designed for the HSS form. Existing HSS factorization algorithms generally introduce zeros into appropriate off-diagonal block rows, and may not be efficient for rectangular matrices. Our URV algorithm introduces zeros into the off-diagonal block columns instead, and factorizes the HSS form into the product of a sequence of small factors. (The name URV is from these factors, as explained at the beginning of section 2.3.)

Another two schemes solve the normal equation, where an HSS approximation to $\mathcal{C}^*\mathcal{C}$ is computed by either randomized sampling or via HSS multiplications involving the HSS approximation to $\mathcal{C}$. The HSS multiplication scheme is followed by a recompression step to enhance the compactness of the HSS approximation to $\mathcal{C}^*\mathcal{C}$.

We provide detailed error and stability analysis for our randomized HSS methods. We also show the analysis for standard HSS methods as corollaries. Such analysis is not done in [38] or in existing HSS work, and is very useful for studying HSS and other rank structured methods. The stability analysis shares some common features with that in [1]. However, some factors here are not unitary, and hierarchical structures are used here instead of sequential ones. We also show a growth factor ρ for the norms of the factors in the URV HSS factorization. ρ is much smaller than the worst case element growth factor for LU factorizations with column pivoting. The backward stability of the URV factorization is proven in detail.

The HSS constructions cost $O((m+n)\log^2(m+n))$ flops, and the HSS factorizations and solutions cost $O(m+n)$ flops. Thus, our methods are stable and superfast.

In comparison, the first fast algorithm for (1.1) is proposed in [32], followed by some other developments in [3, 9, 30]. However, none of these algorithms is proven to be stable. The fast methods in [13, 31] are also based on displacement equations, and the Cauchy-like systems are solved by stable generalized Schur algorithms. In [34], an augmented matrix approach is used to convert (1.1) into a tangential interpolation problem, and then the solution is given by a divide and conquer method with a stabilized technique in $O((m+n)\log^2(m+n))$ complexity. Generalized inverses for

Toeplitz matrices are computed in parallel in [28]. An algorithm based on Newton's iterations is given in [2]. See [5, 26] for more reviews.

Here, we compare our three superfast methods with the two recently developed methods (the fast one in [31] and the superfast one in [34]) and the QR factorization through various numerical tests on classical Toeplitz examples, and demonstrate the stability and efficiency of our methods. The new methods are generally faster, and give better or comparable accuracies, especially for ill-conditioned problems. They still give satisfactory results for some cases when the methods in [31, 34] fail. In general, when the matrix size grows, our methods becomes significantly faster.

1.3. Outline. The remaining sections are organized as follows. Section 2 shows the construction of a rectangular HSS approximation to $\mathcal{C}$ via randomized sampling, followed by the new URV HSS factorization and least squares solution. In section 3, the two normal equation schemes are presented. The detailed stability and error analysis for the HSS methods are given in section 4. The numerical results are shown in section 5, and we draw some conclusions in section 6. The following notation is used in the presentation for an $m \times n$ matrix A.

- $A_{j,k}$ denotes an entry of A with the row index j and the column index k.
- Let $\mathbf{I}$ be a subset of $\{1:m\} \equiv \{1,2,\ldots,m\}$ and $\mathbf{J}$ be a subset of $\{1:n\} \equiv \{1,2,\ldots,n\}$. We use $A|_{\mathbf{I}}$ to denote a submatrix of A with the row index set $\mathbf{I}$, and use $A|_{\mathbf{I}\times\mathbf{J}}$ to denote a submatrix of A with the row index set $\mathbf{I}$ and the column index set $\mathbf{J}$.
- $|\mathbf{I}|$ denotes the number of elements in the set $\mathbf{I}$.
- $\mathrm{diag}(D_1,\ldots,D_k)$ or $\mathrm{diag}(D_j)_{j=1:k}$ denotes a diagonal matrix with the diagonal blocks $D_1,\ldots,D_k$.

2. Structured rectangular least squares solution for $\mathcal{C}$. We first show some HSS algorithms for the rectangular matrix $\mathcal{C}$.

2.1. Rectangular HSS representations. Here, we extend the square HSS representations in [6, 39] to rectangular ones. The diagonal blocks are allowed to be rectangular and may not contain the diagonal entries. An $m \times n$ matrix A is in a *(rectangular) HSS form* with the associated *HSS tree* $\mathcal{T}$ if the following conditions hold.

- $\mathcal{T}$ is a postordered full binary tree with nodes $i = 1,2,\ldots,\mathrm{root}(\mathcal{T})$, where $\mathrm{root}(\mathcal{T})$ is the root. That is, any nonleaf node i of $\mathcal{T}$ has a left child c_1 and a right child c_2 satisfying $c_1 < c_2 < i$.
- There are two index sets $\mathbf{I}_i$ and $\mathbf{J}_i$ associated with each node i of $\mathcal{T}$, which satisfy the following recursions for a nonleaf node i with children c_1 and c_2:

$$\begin{aligned} &\mathbf{I}_{\mathrm{root}(\mathcal{T})} = \{1,2,\ldots,m\}, && \mathbf{J}_{\mathrm{root}(\mathcal{T})} = \{1,2,\ldots,n\}, \\ &\mathbf{I}_i = \mathbf{I}_{c_1} \cup \mathbf{I}_{c_2},\ \mathbf{I}_{c_1} \cap \mathbf{I}_{c_2} = \emptyset, && \mathbf{J}_i = \mathbf{J}_{c_1} \cup \mathbf{J}_{c_2},\ \mathbf{J}_{c_1} \cap \mathbf{J}_{c_2} = \emptyset. \end{aligned}$$

- There are matrices D_i, U_i, V_i, R_i, W_i, B_i (called *HSS generators*) associated with each node i of $\mathcal{T}$, which satisfy the following recursions for a nonleaf node i with children c_1 and c_2:

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

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

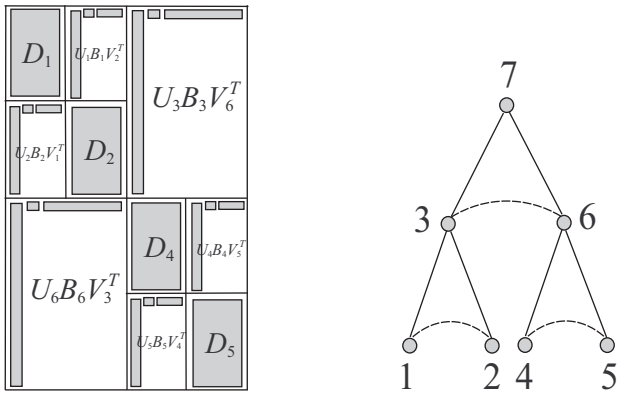


FIG. 2.1. *A rectangular HSS matrix and its HSS tree, where* $U_i = \left(\begin{smallmatrix} U_{c_1} R_{c_1} \\ U_{c_2} R_{c_2} \end{smallmatrix}\right)$, $V_i = \left(\begin{smallmatrix} V_{c_1} W_{c_1} \\ V_{c_2} W_{c_2} \end{smallmatrix}\right)$ *for* $i = 3$ *and* 6.

(The node $i \equiv \mathrm{root}(\mathcal{T})$ is associated with only the generator $D_i \equiv A$ and U_i, V_i, R_i, W_i, B_i are not needed.)

See Figure 2.1 for an example. It is clear that U_i and V_i^* (called *basis matrices*) are appropriate bases for the column or row spaces of the following blocks:

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

A_i^- and $A_i^|$ are called *HSS block rows and columns*, respectively. Each node corresponds to an HSS block row and column. Thus, associated with the nodes at the same level of $\mathcal{T}$, the numbers of HSS block rows and HSS block columns are equal. The maximum (numerical) rank of all the HSS blocks of A is called the *HSS rank* of A.

The HSS tree $\mathcal{T}$ is formed in the following way. We first decide the number (α) of leaves or the number of bottom level blocks. This is usually based on the criterion that the bottom level diagonal blocks have sizes close to the HSS rank [39]. Then we form a binary tree $\mathcal{T}$ with $2\alpha - 1$ nodes. For scalability purposes, we usually make $\mathcal{T}$ as balanced as possible.

For convenience, the following notation is used for a node i of $\mathcal{T}$:

- $\mathrm{par}(i)$ and $\mathrm{sib}(i)$ denote the parent and the sibling of i in $\mathcal{T}$, respectively;
- if i is a nonleaf node, c_1 and c_2 denote its left and right children, respectively.

2.2. Randomized HSS construction and URV least squares solution for $\mathcal{C}$. Similarly to the method in [38], a rectangular HSS approximation to $\mathcal{C}$ can be quickly constructed. Here, the main steps in [38] are briefly reviewed, with the differences emphasized.

2.2.1. Randomized sampling. The essential idea of randomized sampling [16, 21] for *compressing* an $M \times N$ block Θ (finding a low-rank approximation to it) is to work on a skinny matrix $Z = \Theta X$ instead of the original Θ, where X is a Gaussian random matrix. X is chosen to be $N \times (r + \gamma)$, where r is the rank of Θ (or *numerical rank* when a tolerance is used for the truncation of the singular values of Θ), and γ is a small integer (greater than 2). This is briefly explained as follows, following [38].

Compute a strong rank-revealing factorization [14] of Z, which looks like $Z \approx \Pi Q S$, where Π is a permutation matrix and Q is $M \times r$. Partition Q as $\left(\begin{smallmatrix} Q_1 \\ Q_2 \end{smallmatrix}\right)$, where

TABLE 2.1
Probabilities of failure (Pr) for (2.5) with different γ [16, 21, 24].

γ	3	4	5	6	7	8	9	10
Pr	$2.22e{-1}$	$2.34e{-2}$	$1.92e{-3}$	$1.29e{-4}$	$7.29e{-6}$	$3.58e{-7}$	$1.55e{-8}$	$6.00e{-10}$

Q_1 is $r \times r$, and we have

$$Z \approx \Pi \begin{pmatrix} I \\ E \end{pmatrix} Q_1 S \equiv UZ|_{\hat{\mathbf{I}}} \text{ with } U = \Pi \begin{pmatrix} I \\ E \end{pmatrix}, \ E = Q_2 Q_1^{-1}, \tag{2.4}$$

where $\hat{\mathbf{I}}$ is a row index set and E is obtained as in [14]. Then it is shown in [16, 21, 24] that Θ can be approximated by

$$\Theta \approx U\Theta|_{\hat{\mathbf{I}}},$$

and the following approximation error bound holds (for a slightly varied form) with a probability of at least $1 - 6\gamma^{-\gamma}$:

$$\left\|\Theta - U\Theta|_{\hat{\mathbf{I}}}\right\|_2 \le (1 + 11\sqrt{r \cdot \min(M, N)})\sigma_{r+1}, \tag{2.5}$$

where σ_{r+1} is the $(r+1)$-st largest singular value of Θ. As mentioned in [16], a small integer $\gamma > 2$ can already give a high probability of success. For example, the probability of failure with respect to different γ is reported in Table 2.1. The above randomized scheme is used when we have a good estimate of r (or an empirical value) in advance for a given accuracy. Otherwise, we can use the adaptive randomized method in [16] by specifying the accuracy instead.

Since $\Theta|_{\hat{\mathbf{I}}}$ is a subblock of Θ, such a factorization is also called a structure-preserving rank-revealing (SPRR) factorization in [38]. We denote it by

$$[U, \Theta|_{\hat{\mathbf{I}}}] = \text{SPRR}(\Theta). \tag{2.6}$$

This feature of structure preservation is very important for our error and stability analysis later in section 4.

2.2.2. Randomized HSS construction for $\mathcal{C}$. The benefits of randomized sampling for HSS constructions can be found in [22, 24, 38]. The rectangular HSS construction for $\mathcal{C}$ is a direct generalization of the one in [38]. This needs the multiplication of $\mathcal{C}$ with random vectors, which is done with the aid of fast Toeplitz matrix-vector multiplications. One useful way is to split the lower left and the upper right triangular pieces (which can be extended to circulant matrices), and the remaining part can be handled with a pruned FFT (http://www.fftw.org/pruned.html). This multiplication cost is usually less significant as compared with the other costs in the HSS construction.

To facilitate the presentation, for a node i of a given HSS tree $\mathcal{T}$, let the row dimension of $\mathcal{C}_i^-$ be $m_i \equiv |\mathbf{I}_i|$ and its starting row index in $\mathcal{C}$ be l_i, and let the column dimension of $\mathcal{C}_i^|$ be $n_i \equiv |\mathbf{J}_i|$ and its starting column index in $\mathcal{C}$ be s_i. That is,

$$l_i = \begin{cases} \sum_{j:\ \text{leaf},\ j<i} m_j + 1 & \text{if } i \text{ is a leaf}, \\ l_{c_1} & \text{otherwise}, \end{cases} \quad s_i = \begin{cases} \sum_{j:\ \text{leaf},\ j<i} n_j + 1 & \text{if } i \text{ is a leaf}, \\ s_{c_1} & \text{otherwise}. \end{cases} \tag{2.7}$$

See Algorithm 1 for the randomized HSS construction with the Toeplitz vector $t_{-(n-1):(m-1)}$ and an estimate of the HSS rank r of $\mathcal{C}$ as the inputs. The main step of

the randomized HSS construction is to recursively apply SPRR factorizations to the HSS blocks of $\mathcal{C}$. Computed basis matrices are ignored in upper level compression. The detailed derivations are similar to those in [38] and are skipped. The algorithm uses two Gaussian random matrices X and Y of sizes $n \times \tilde{r}$ and $m \times \tilde{r}$, respectively, where $\tilde{r} = r + \gamma$ with r the HSS rank of $\mathcal{C}$ and γ a small integer in the previous subsection. X and Y are partitioned into X_i and Y_i block rows following the sizes m_i and n_i, respectively, for the leaves i of $\mathcal{T}$. The cost of the algorithm is discussed in section 2.5.

ALGORITHM 1. RANDOMIZED RECTANGULAR HSS CONSTRUCTION FOR $\mathcal{C}$.

Input: Toeplitz vector $t_{-(n-1):(m-1)}$ and $\tilde{r} = r + \gamma$

Output: HSS generators D_i, U_i, V_i, W_i, B_i for the HSS approximation to $\mathcal{C}$

1: **procedure** `RectHSS`
2: $Z \leftarrow \mathcal{C}X, \quad S \leftarrow \mathcal{C}^*Y$ ($X \in \mathbb{R}^{n\times\tilde{r}}$, $Y \in \mathbb{R}^{m\times\tilde{r}}$: random) ▷ Via (1.4)
3: **for** node i from 1 to root $(\mathcal{T})$ **do**
4: **if** i is a leaf of $\mathcal{T}$ **then** ▷ D, U, V generators
5: $D_i \leftarrow \mathcal{C}|_{\mathbf{I}_i \times \mathbf{J}_i}$ ▷ Via (1.7)
6: $\Phi_i \leftarrow Z_i - D_i X_i, \quad [U_i, \Phi_i|_{\hat{\mathbf{I}}_i}] \leftarrow \mathrm{SPRR}(\Phi_i)$ ▷ $\hat{\mathbf{I}}$: obtained as in (2.6)
7: $\Theta_i \leftarrow S_i - D_i^* Y_i, \quad [V_i, \Theta_i|_{\hat{\mathbf{J}}_i}] \leftarrow \mathrm{SPRR}(\Theta_i)$ ▷ $\hat{\mathbf{J}}$: obtained as in (2.6)
8: $\hat{Z}_i \leftarrow V_i^* X_i, \quad \hat{S}_i \leftarrow U_i^* Y_i$
9: **else** ▷ R, W generators; c_1, c_2: children of i
10: $B_{c_1} = \mathcal{C}|_{\tilde{\mathbf{I}}_{c_1} \times \tilde{\mathbf{J}}_{c_2}}, \ B_{c_2} = \mathcal{C}|_{\tilde{\mathbf{I}}_{c_2} \times \tilde{\mathbf{J}}_{c_1}}$ ▷ Via (1.7); $\tilde{\mathbf{I}}, \tilde{\mathbf{J}}$: from line 18
11: **if** $i = \mathrm{root}\,(\mathcal{T})$ **then**
12: **return**
13: **end if**
14: $\Phi_i \leftarrow \begin{pmatrix} \Phi_{c_1}|_{\mathbf{I}_{c_1}} - B_{c_1}\hat{Z}_{c_2} \\ \Phi_{c_2}|_{\mathbf{I}_{c_2}} - B_{c_2}\hat{Z}_{c_1} \end{pmatrix}, \quad \left[\begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix}, \Phi_i|_{\hat{\mathbf{I}}_i} \right] \leftarrow \mathrm{SPRR}(\Phi_i)$
15: $\Theta_i \leftarrow \begin{pmatrix} \Theta_{c_1}|_{\mathbf{I}_{c_1}} - B_{c_2}^*\hat{S}_{c_1} \\ \Theta_{c_2}|_{\mathbf{I}_{c_2}} - B_{c_1}^*\hat{S}_{c_2} \end{pmatrix}, \quad \left[\begin{pmatrix} W_{c_1} \\ W_{c_2} \end{pmatrix}, \Theta_i|_{\hat{\mathbf{J}}_i} \right] \leftarrow \mathrm{SPRR}(\Theta_i)$
16: $\hat{Z}_i \leftarrow \begin{pmatrix} W_{c_1}^* & W_{c_2}^* \end{pmatrix} \begin{pmatrix} \hat{Z}_{c_1} \\ \hat{Z}_{c_2} \end{pmatrix}, \quad \hat{S}_i \leftarrow \begin{pmatrix} R_{c_1}^* & R_{c_2}^* \end{pmatrix} \begin{pmatrix} \hat{S}_{c_1} \\ \hat{S}_{c_2} \end{pmatrix}$
17: **end if**
18: $\tilde{\mathbf{I}}_i \leftarrow l_i + \hat{\mathbf{I}}_i - 1, \ \tilde{\mathbf{J}}_i \leftarrow s_i + \hat{\mathbf{J}}_i - 1$ ▷ (More details can be found in [38])
▷ Shifting $\hat{\mathbf{I}}_i, \hat{\mathbf{J}}_i$ to global index sets with entrywise additions
19: **end for**
20: **end procedure**

We point out that the generators U, V, R, W have special structures as in [38]. That is, U_i, V_i for a leaf i have the following forms:

$$U_i = \Pi_i \begin{pmatrix} I \\ E_i \end{pmatrix}, \ V_i = \Upsilon_i \begin{pmatrix} I \\ F_i \end{pmatrix}, \tag{2.8}$$

where $\Pi_i, E_i, \Upsilon_i, F_i$ can be understood following (2.4). If i is a nonleaf node, then

$$\begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix} = \Pi_i \begin{pmatrix} I \\ E_i \end{pmatrix}, \ \begin{pmatrix} W_{c_1} \\ W_{c_2} \end{pmatrix} = \Upsilon_i \begin{pmatrix} I \\ F_i \end{pmatrix}. \tag{2.9}$$

Remark 2.1. Once we choose a proper γ (such as 10) in Algorithm 1, the overall probability of the HSS construction to a desired accuracy is still very satisfactory in

practice. A conservative probability estimate works as follows. If $\gamma = 10$ and the HSS blocks are compressed to a given accuracy with a probability of failure 6×10^{-10}, then a pessimistic probability estimate for constructing an HSS approximation to the given accuracy (with the error amplification as estimated in Theorem 4.2) is at least $(1 - 6 \times 10^{-10})^{\frac{2n}{n_1}-1}$, where n_1 is the leaf level diagonal block column size and $\frac{2n}{n_1} - 1$ is the total number of nodes in the HSS tree. When r is, say, 125 as in our tests, this conservative probability is still larger than $\frac{1}{2}$ for n as large as 7.2×10^{10}. In fact, the numerical tests for the randomized HSS constructions in [22, 24, 38] show that high probabilities are achieved in practice. In later sections, to be precise and avoid confusion, we assume that the off-diagonal compression with the given accuracy has probability 1 when we state the theorems on the approximation errors.

2.3. Rectangular URV HSS factorization with size reduction. In this section, we discuss the factorization of the HSS approximation to $\mathcal{C}$. We use A to represent the HSS approximation to $\mathcal{C}$. The traditional ULV-type HSS factorizations [6, 39] are designed for square HSS matrices, and are generally not efficient for rectangular HSS matrices when $m \gg n$ (see Remark 2.2 below). (ULV represents the factorization where U and V are given by the products of sequences of orthogonal matrices and L is given by the product of a sequence of lower triangular matrices [6, 39].) Here, we present a URV-type factorization together with a size reduction strategy. (The meaning of URV can be similarly understood, except that R represents the product of a sequence of upper triangular matrices, and V may not be unitary, but still has a bounded norm and an explicit representation for its inverse.) The size reduction strategy helps reduces the row size of A. The URV scheme simulates QR least squares solutions at multiple levels, and hierarchically reduces the HSS form into smaller ones.

For simplicity, assume the sizes $m_i \times n_i$ of the D_i generators associated with all leaves i of the HSS tree $\mathcal{T}$ are the same. That is, $m_i \equiv m_1$ and $n_i \equiv n_1$ for all the leaves i. Similarly, assume the U_i, V_i generators have column sizes r, and $m_1 \geq r$.

2.3.1. Size reduction strategy. If $m \gg n$ or $m_1 \gg n_1$, we first use a size reduction strategy to reduce the row size of A to be as close to n as possible. This can be done by modifying some HSS generators. The idea is to introduce zero rows into the D and U generators and thus A.

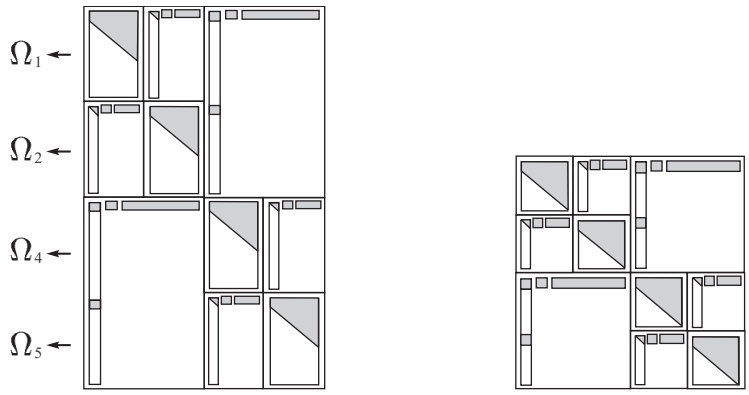
For a leaf i of $\mathcal{T}$, compute a block QR factorization for the matrix $\begin{pmatrix} U_i & D_i \end{pmatrix}$:

$$\begin{pmatrix} U_i & D_i \end{pmatrix} = \Omega_i \begin{array}{c} \begin{matrix} r & n_1 \end{matrix} \\ \begin{pmatrix} \tilde{U}_i & \tilde{D}_{i,1} \\ & \tilde{D}_{i,2} \\ & 0 \end{pmatrix} \begin{matrix} r \\ n_1 \\ m_1 - r - n_1 \end{matrix} \end{array}. \tag{2.10}$$

Then multiply Ω_i^* with D_i and U_i on the left. See Figure 2.2. The zero rows on the right-hand side of (2.10) correspond to the zero rows introduced into A. In our least squares solution for (1.8), we can ignore such zero rows. That is, we can replace the generators D_i and U_i for the leaf i by $\tilde{D}_i \equiv \begin{pmatrix} \tilde{D}_{i,1} \\ \tilde{D}_{i,2} \end{pmatrix}$ and $\begin{pmatrix} \tilde{U}_i \\ 0 \end{pmatrix}$, respectively.

Assume this reduction is applied to all leaves i, and the resulting new HSS form (with all the zero rows ignored) is $\tilde{A}$. That is, we can write

$$A = \mathrm{diag}(\Omega_i)_{i:\ \mathrm{leaf}} \mathcal{P} \begin{pmatrix} \tilde{A} \\ 0 \end{pmatrix}, \tag{2.11}$$



(i) Introducing zeros with one level of size reduction (ii) After size reduction

FIG. 2.2. *Size reduction for the rectangular HSS approximation to A as in Figure* 2.1.

where $\mathcal{P}$ is an appropriate permutation matrix. If necessary, such a reduction can also be applied before the elimination of the nonleaf nodes in the factorization process in the next subsection. The reason is as follows. After the elimination of the lower level nodes in the factorization, an intermediate reduced HSS matrix is obtained, as explained after (2.18). The ratio of the row dimension to the column dimension of such reduced matrices increases along the URV factorization. We can thus apply the row size reduction to control this growth and to save the computational cost.

2.3.2. URV factorization. Then we present our URV factorization scheme for $\tilde{A}$, which introduces zeros into the HSS block columns of $\tilde{A}$ (instead of HSS block rows as in ULV factorizations). We traverse the tree $\mathcal{T}$ bottom-up. Noticing (2.8), for convenience, let

$$P_i = \Upsilon_i \begin{pmatrix} -F_i^* & I \\ I & 0 \end{pmatrix}. \tag{2.12}$$

If a node i is a leaf of $\mathcal{T}$, according to (2.8) and similar to [38], we have

$$V_i^* P_i = \begin{pmatrix} 0 & I \end{pmatrix}. \tag{2.13}$$

Then multiply P_i with $\tilde{A}_i^{|}$ and $\tilde{D}_i$ on the right. Equation (2.13) indicates that some zero columns are introduced into $\tilde{A}_i^{|}$. See Figures 2.3(i)–(ii). Let

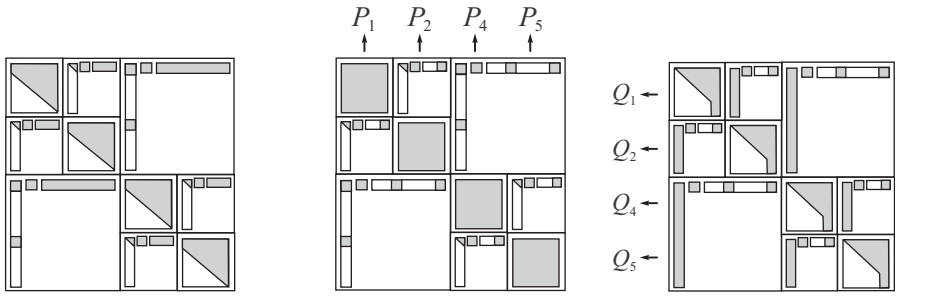
$$\hat{D}_i = \tilde{D}_i P_i. \tag{2.14}$$

Note that the special structure of P_i helps in saving the multiplication cost. Partition $\hat{D}_i$ as

$$\hat{D}_i = \begin{pmatrix} \hat{D}_{i;1,1} & \hat{D}_{i;1,2} \\ \hat{D}_{i;2,1} & \hat{D}_{i;2,2} \end{pmatrix}, \tag{2.15}$$

so that $\hat{D}_{i;1,1}$ is $r \times r$, and compute a QR factorization

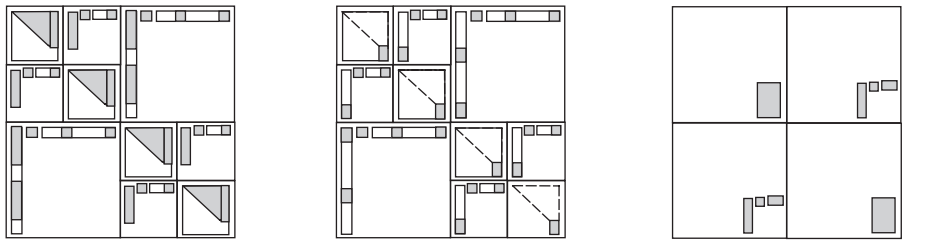
$$\begin{pmatrix} \hat{D}_{i;1,1} \\ \hat{D}_{i;2,1} \end{pmatrix} = Q_i \begin{pmatrix} \bar{D}_{i;1,1} \\ 0 \end{pmatrix}. \tag{2.16}$$



(i) HSS form after size reduction

(ii) Introducing zeros into off-diagonal columns

(iii) QR factorization of diagonal blocks



(iv) Blocks remaining at current level

(v) Blocks for upper level factorization

(vi) Upper level HSS form after merging blocks

FIG. 2.3. *Illustration of the URV HSS factorization scheme.*

Then multiply Q_i^* with $\hat{D}_i$ and $\left(\begin{smallmatrix}\bar{U}_i\\0\end{smallmatrix}\right)$ on the left:

$$\bar{D}_i = Q_i^* \hat{D}_i = \begin{pmatrix} \bar{D}_{i;1,1} & \bar{D}_{i;1,2} \\ 0 & \bar{D}_{i;2,2} \end{pmatrix},\ \bar{U}_i = Q_i^* \begin{pmatrix} \tilde{U}_i \\ 0 \end{pmatrix} \equiv \begin{pmatrix} \bar{U}_{i;1} \\ \bar{U}_{i;2} \end{pmatrix}, \tag{2.17}$$

where $\bar{D}_i$ and $\bar{U}_i$ are partitioned conformably. See Figure 2.3(iii).

If i is a nonleaf node, we merge appropriate blocks and let

$$\tilde{D}_i = \begin{pmatrix} \bar{D}_{c_1;2,2} & \bar{U}_{c_1;2} B_{c_1} \\ \bar{U}_{c_2;2} B_{c_2} & \bar{D}_{c_2;2,2} \end{pmatrix}, \tag{2.18}$$

$$\tilde{V}_i = \begin{pmatrix} W_{c_1} \\ W_{c_2} \end{pmatrix} \equiv \Upsilon_i \begin{pmatrix} I \\ F_i \end{pmatrix},\ \tilde{U}_i = \begin{pmatrix} \bar{U}_{c_1;2} R_{c_1} \\ \bar{U}_{c_2;2} R_{c_2} \end{pmatrix} \equiv \begin{pmatrix} \bar{U}_{c_1;2} & \\ & \bar{U}_{c_2;2} \end{pmatrix} \Pi_i \begin{pmatrix} I \\ E_i \end{pmatrix}.$$

Then we can remove the children c_1 and c_2 of i from $\mathcal{T}$. By recursion, i becomes a leaf with the associated generators $\tilde{D}_i, \tilde{U}_i, \tilde{V}_i, R_i, W_i$. The generators $\tilde{D}_i, \tilde{U}_i, \tilde{V}_i, R_i, W_i, B_i$ define a new HSS form called a *reduced (HSS) matrix* [35]. For convenience, we also define an *extended reduced matrix* which is the original HSS matrix with the blocks $\left(\begin{smallmatrix}\bar{D}_{i;1,1} & \bar{D}_{i;1,2}\end{smallmatrix}\right)$ and $\bar{U}_{i;1} R_i B_i$ (Figure 2.3(iv)) replaced by zeros (Figure 2.3(v)). That is, the extended one includes some extra zero blocks.

At this point, the HSS matrix is reduced to a special form, where the block rows are either in the extended reduced matrix, or are given by $\left(\begin{smallmatrix}\bar{D}_{i;1,1} & \bar{D}_{i;1,2}\end{smallmatrix}\right)$ and $\bar{U}_{i;1} R_i B_i$. The latter blocks remain at the current level and will be used for the solutions (in the solution stage, these blocks will be visited in a top-down order).

The reduced HSS matrix continues to be factorized, and the above process repeats on i. See Figures 2.3(v)–(vi). When $i = \text{root}(\mathcal{T})$ is reached, we have the final reduced

matrix $\tilde{D}_i$, and then replace the QR factorization (2.16) by

$$\tilde{D}_i = Q_i \begin{pmatrix} \bar{D}_{i;1,1} \\ 0 \end{pmatrix}. \tag{2.19}$$

The algorithm is summarized in Algorithm 2, which computes a sequence of factors.

ALGORITHM 2. URV FACTORIZATION FOR A RECTANGULAR HSS MATRIX.

Input: HSS generators D_i, B_i and U_i, V_i, R_i, W_i (as in (2.8) and (2.9))

Output: Factors $\hat{D}_{i;1,1}, \hat{D}_{i;1,2}, \hat{U}_{i;1}, Q_i, P_i$ (for later URV solutions)

procedure URV
  for node i from 1 to root($\mathcal{T}$) $-$ 1 **do**
    if i is a leaf of $\mathcal{T}$ **then**
      if $m_i \gg n_i$ **then** ▷ E.g., $m_i \geq 2n_i$ ($m_i \times n_i$: size of D_i)
        Compute a QR factorization (2.10) ▷ Row size reduction
        $\tilde{D}_i \leftarrow \begin{pmatrix} \tilde{D}_{i,1} \\ \tilde{D}_{i,2} \end{pmatrix}$
      end if
    else ▷ Merge child information
      $\tilde{D}_i \leftarrow \begin{pmatrix} \bar{D}_{c_1;2,2} & \bar{U}_{c_1;2} B_{c_1} \\ \bar{U}_{c_2;2} B_{c_2} & \bar{D}_{c_2;2,2} \end{pmatrix}$
      $\tilde{V}_i \leftarrow \Upsilon_i \begin{pmatrix} I \\ F_i \end{pmatrix}, \quad \tilde{U}_i \leftarrow \begin{pmatrix} \hat{U}_{c_1;2} & \\ & \hat{U}_{c_2;2} \end{pmatrix} \Pi_i \begin{pmatrix} I \\ E_i \end{pmatrix}$
    end if
    $\hat{D}_i \leftarrow \tilde{D}_i \Upsilon_i \begin{pmatrix} -F_i^* & I \\ I & 0 \end{pmatrix} \equiv \begin{pmatrix} \hat{D}_{i;1,1} & \hat{D}_{i;1,2} \\ \hat{D}_{i;2,1} & \hat{D}_{i;2,2} \end{pmatrix}$
    Compute a QR factorization (2.16)
    $\begin{pmatrix} \bar{D}_{i;1,2} \\ \bar{D}_{i;2,2} \end{pmatrix} \leftarrow Q_i^* \begin{pmatrix} \hat{D}_{i;1,2} \\ \hat{D}_{i;2,2} \end{pmatrix}, \quad \bar{U}_i \leftarrow Q_i^* \begin{pmatrix} \tilde{U}_i \\ 0 \end{pmatrix} \equiv \begin{pmatrix} \hat{U}_{i;1} \\ \hat{U}_{i;2} \end{pmatrix}$
  end for
  Compute a QR factorization (2.19)
end procedure

Remark 2.2. It is possible to use the ULV factorization scheme in [6, 39] to factorize the HSS form, without using the size reduction strategy. However, it can be shown that the URV factorization with the size reduction is faster than the existing ULV factorization by a complexity of $O(2rm_1m)$ when m is much larger than n.

2.4. URV least squares solution. According to (2.11), the least squares problem (1.8) becomes

$$\min_{\tilde{x}} \left\| \begin{pmatrix} \tilde{A} \\ 0 \end{pmatrix} \tilde{x} - \mathcal{P}^* \operatorname{diag}(\Omega_i^*)_{i:\ \text{leaf}} \tilde{b} \right\|_2 = \min_{\mathbf{x}} ||\tilde{A}\mathbf{x} - \mathbf{b}||_2 + \tilde{c}_1, \tag{2.20}$$

where we write $\tilde{x}$ as $\mathbf{x}$ for notational convenience, $\mathbf{b}$ results from appropriate transformations of (selected entries of) $\tilde{b}$ in (1.8), and $\tilde{c}_1$ is a constant. ((2.20) approximately solves (1.8) since the HSS form approximates $\mathcal{C}$.) The solution of (2.20) is that of

$$\min_{\mathbf{x}} ||\tilde{A}\mathbf{x} - \mathbf{b}||_2. \tag{2.21}$$

This involves a top-down traversal of $\mathcal{T}$. For convenience, partition $\mathbf{b}$ into $\mathbf{b}_i$ pieces corresponding to $\bar{D}_i$, and further partition $\mathbf{b}_i$ as $\mathbf{b}_i = \left(\begin{smallmatrix} \mathbf{b}_{i,1} \\ \mathbf{b}_{i,2} \end{smallmatrix}\right)$ following (2.17).

For the node $i = \text{root}(\mathcal{T})$, solve a triangular system

$$\bar{D}_{i;1,1}\mathbf{x}_i = Q_i^*\mathbf{b}_{i,1}.$$

For the children c_1 and c_2 of i, partition $\mathbf{x}_i$ following (2.18) as

$$\mathbf{x}_i = \begin{pmatrix} \hat{\mathbf{x}}_{c_1} \\ \hat{\mathbf{x}}_{c_2} \end{pmatrix}. \tag{2.22}$$

Also let $\mathbf{z}_{c_1} = B_{c_1}\hat{\mathbf{x}}_{c_2}$, $\mathbf{z}_{c_2} = B_{c_2}\hat{\mathbf{x}}_{c_1}$.

For the nodes $i = \text{root}(\mathcal{T}) - 1, \text{root}(\mathcal{T}) - 2, \ldots, 1$, we solve the triangular systems

$$\bar{D}_{i;1,1}\bar{\mathbf{x}}_i = \mathbf{b}_{i,1} - \bar{D}_{i;1,2}\hat{\mathbf{x}}_i - \hat{U}_{i;1}\mathbf{z}_j, \tag{2.23}$$

where $j = \text{sib}\,(i)$ and the results in (2.17) are used. Also let

$$\mathbf{x}_i = P_i \begin{pmatrix} \bar{\mathbf{x}}_i \\ \hat{\mathbf{x}}_i \end{pmatrix} = \Upsilon_i \begin{pmatrix} -F_i^* & I \\ I & 0 \end{pmatrix} \begin{pmatrix} \bar{\mathbf{x}}_i \\ \hat{\mathbf{x}}_i \end{pmatrix}.$$

If i is also a nonleaf node, partition $\mathbf{x}_i$ as in (2.22), and compute

$$\mathbf{z}_{c_1} = B_{c_1}\hat{\mathbf{x}}_{c_2} + R_{c_1}\mathbf{z}_i,\ \mathbf{z}_{c_2} = B_{c_2}\hat{\mathbf{x}}_{c_1} + R_{c_2}\mathbf{z}_i.$$

After the traversal, merge $\mathbf{x}_i$ associated with all the leaves i following the indices of $\bar{D}_{i;1,1}$ in (2.17) and form the solution $\mathbf{x}$. The summary of the algorithm is skipped.

2.5. Complexity. The complexity analysis for rectangular HSS methods is a direct extension of that for square HSS methods in [35, 38, 39]. We demonstrate the flop counts for the rectangular $(m \times n)$ HSS construction, and those for the factorization and solution can be similarly obtained. For simplicity, assume the size $(m_i \times n_i)$ of D_i for each leaf i satisfies $m_i \equiv m_1 = O(r)$, $n_i \equiv n_1 = O(r)$ and all the HSS blocks have numerical ranks $r = O(\log(m+n))$. Also use $\tilde{r}$ to denote the sampling size $r + \gamma$.

In the HSS construction algorithm, computing the products $\mathcal{C}X$ and $\mathcal{C}^*Y$ costs about $40\tilde{r}(m+n)\log(m+n)$ flops. Other costs associated with each node of the HSS tree are summarized in Table 2.2. A direct summation of all the costs gives the total HSS construction complexity $O((m+n)\log^2(m+n))$. Similarly, the HSS URV factorization and solution costs are $O((m+n)\log^2(m+n))$ and $O((m+n)\log(m+n))$, respectively. As in [35, 38], when the actual rank patterns at the individual hierarchical levels are considered, we can further give a tighter complexity bound $O(m+n)$ for the URV factorization and solution.

TABLE 2.2
Operations and costs in randomized rectangular HSS construction in Algorithm 1.

Node	Operation	Flops (leading terms)	# of nodes
Leaf	Computing Φ_i and Θ_i	$4m_1n_1\tilde{r}$	$\times(m/m_1)$
node i	Computing $\hat{Z}_i$	$2r(n_1-r)\tilde{r}$	(or n/n_1)
	Computing $\hat{S}_i$	$2r(m_1-r)\tilde{r}$	
	SPRR factorization	$4(m_1+n_1)\tilde{r}r - r^2(m_1+n_1+4\tilde{r}) + \frac{2}{3}r^3$	
Non-leaf	Computing Φ_i and Θ_i	$8r^2\tilde{r}$	$\times(m/m_1-1)$
node i	Computing $\hat{Z}_i$ and $\hat{S}_i$	$4r^2\tilde{r}$	(or n/n_1-1)
	SPRR factorization	$16\tilde{r}r^2 - 4r^2(r+\tilde{r}) + \frac{2}{3}r^3$	

3. Structured normal equation methods. Classical least squares solution schemes also include normal equation methods. We can similarly propose structured methods for the normal equation corresponding to (1.8):

$$\mathcal{C}^*\mathcal{C}\tilde{x} = \mathcal{C}^*\tilde{b}. \tag{3.1}$$

A simple multiplication can show that the HSS rank of $\mathcal{C}^*\mathcal{C}$ is at most twice that of $\mathcal{C}$, and is thus still $O(\log(m+n))$.

There are some benefits in using the normal equation. One is that we can apply iterative refinement together with a modest-accuracy HSS solution, especially since the solution costs about $O(n)$ and is much faster than the HSS construction and factorization. For example, in Figures 5.1 and 5.3 below, iterative refinement helps the methods achieve nearly machine precision. Another advantage is that, when $m \gg n$, $\mathcal{C}^*\mathcal{C}$ is only $n \times n$ and is a much smaller problem to solve. For example, in Figures 5.3(i) and 5.5(i), when $\frac{m}{n}$ is larger than 7 and grows, the normal equation methods gets faster and faster than the URV method in the timing, even with iterative refinement. In addition, normal equations are also useful in regularization methods. Based on different strategies for the HSS construction for $\mathcal{C}^*\mathcal{C}$, we have two methods.

3.1. HSS multiplication and recompression for $\mathcal{C}^*\mathcal{C}$. The first method is to use the HSS approximation to $\mathcal{C}$ to construct one for $\mathcal{C}^*\mathcal{C}$ via HSS multiplications. Assume the HSS generators for $\mathcal{C}^*\mathcal{C}$ are $\tilde{D}_i$, $\tilde{U}_i$, etc. Since $\mathcal{C}^*\mathcal{C}$ is Hermitian, we have $\tilde{U}_i = \tilde{V}_i$, $\tilde{R}_i = \tilde{W}_i$, $\tilde{B}_i = \tilde{B}_j^*$, where $j = \mathrm{sib}(i)$ [39]. The HSS multiplication algorithm in [6, 23] can be used with simplification due to the symmetry.

After the HSS multiplication, the sizes of the generators increase additively, although the actual HSS rank of $\mathcal{C}^*\mathcal{C}$ may be smaller. Then we can modify a recompression scheme in [35] for general matrices and derive a simplified version for the Hermitian matrix $\mathcal{C}^*\mathcal{C}$. This helps make the HSS representations more compact.

Both algorithms involve bottom-up and top-down traversals of the HSS tree. To save space, we skip the details which can be extracted from [6, 23, 35].

3.2. Direct HSS construction for $\mathcal{C}^*\mathcal{C}$. Our second normal equation method avoids using the recompression step. That is, we modify the HSS construction method in section 2.2 so that it can be applied to the Hermitian matrix $\mathcal{C}^*\mathcal{C}$. Clearly, the multiplication of $\mathcal{C}^*\mathcal{C}$ with a random vector can be quickly done. Then the HSS construction is similar to the one in section 2.2, except that the entries of $\mathcal{C}^*\mathcal{C}$ are not explicitly available. A straightforward way to obtain these entries is HSS multiplication with the HSS approximation to $\mathcal{C}$. This needs to traverse $\mathcal{T}$, and some multiplications can be reused for those entries in the same rows or columns, as discussed in [36].

It is also possible to modify the HSS multiplication scheme in the previous subsection into a selected HSS multiplication method.

3.3. ULV factorization and solution for the normal equations. Then we solve the Hermitian linear system (3.1) with the HSS approximation to $\mathcal{C}^*\mathcal{C}$. The ULV factorization and solution methods in [38, 39] can work with minor modifications.

Moreover, the HSS approximation obtained with the method in section 3.2 has additional structures just like in (2.8)–(2.9). Thus, the technique as in (2.13) can be used. Due to the symmetry, we can modify the method in [38] to further save costs.

For both methods, we can also use iterative refinement to improve the solutions, as mentioned at the beginning of this section. Thus, we can use modest accuracies when computing the HSS approximations to $\mathcal{C}^*\mathcal{C}$.

3.4. Complexity. The complexities of the square HSS algorithms used in the normal equation methods have been systematically studied in [35, 38]. For example, the rectangular HSS matrix multiplication cost can be obtained via the summation of the cost of $O(r^3)$ for each of the $O((m+n)/r)$ nodes. Similarly, we can obtain the total cost of $O((m+n)\log^2(m+n))$ for the HSS construction for $\mathcal{C}^*\mathcal{C}$. The ULV factorization and solution with the HSS approximation to $\mathcal{C}^*\mathcal{C}$ cost $O((m+n)\log^2(m+n))$ and $O((m+n)\log(m+n))$, respectively, also with the feasibility of further reduction to $O(m+n)$ via the consideration of the hierarchical rank patterns [38].

4. HSS error and stability analysis. Here, we discuss the HSS approximation error and the stability of the URV factorization. For convenience, we can assume $m = n$ in analyzing the approximation error, since the rectangular HSS construction is the same as the square one, except that the diagonal blocks are rectangular. Similarly, in the URV factorization, the same operations are performed regardless of the shape of the HSS matrix, except that the final reduced matrix $\tilde{D}_i$ in (2.19) is rectangular when m and n are different. (2.19) is small and is the same as in the regular QR least squares method. The standard least squares stability analysis applies to (2.19) and indicates it is stable. Thus, it is critical to analyze the stability of the hierarchical row and column reductions in the URV factorization process that yield $\tilde{D}_i$. Such reductions are the same whether m and n are equal or not. Therefore, we can also assume $m = n$ for convenience. (The optional row reduction in section 2.3.1 is also stable since it uses orthogonal operations to introduce zero rows into the HSS matrix.) Some comments on the case of $m \neq n$ will be given in Remark 4.2.

The stability results for the URV factorization with $m = n$ can be directly extended to the ULV factorization of the HSS normal matrix in section 3, since the latter is closely related to the application of the former to the Hermitian of the matrix. We also give the stability results for the standard HSS ULV factorization in [6, 39] when the off-diagonal bases U_i and V_i have unitary columns.

For convenience, let $L \equiv O(\log \frac{n}{r})$ be the number of levels in $\mathcal{T}$ with the leaves at level L. In this case, all the HSS generators are of orders $O(r)$ (as often used [35], we assume that D_i has order $2r$ and U_i, V_i have sizes $2r \times r$ for a leaf i, and R_i, W_i, B_i have orders r for all nodes i). Also assume that $\mathcal{T}$ is traversed levelwise in the algorithms, and in the HSS construction, all the HSS block rows at level l are compressed first, followed by the compression of all the HSS block columns at level l.

4.1. Approximation error for HSS construction. We first consider the approximation error introduced by the HSS construction in section 2.2.2. For convenience, assume we know the relative tolerance τ for truncating the singular values of the HSS blocks (corresponding to the numerical rank r). That is, for (2.5), we have

$$\sigma_{r+1} \leq \tau\sigma_1. \tag{4.1}$$

According to (2.1), we can see that the HSS matrix A has the following form:

$$A = \begin{pmatrix} D_{c_1} & \\ & D_{c_2} \end{pmatrix} + \begin{pmatrix} U_{c_1} & \\ & U_{c_2} \end{pmatrix} \begin{pmatrix} & B_{c_1} \\ B_{c_2} & \end{pmatrix} \begin{pmatrix} V_{c_1}^* & \\ & V_{c_2}^* \end{pmatrix},$$

where c_1 and c_2 are the children of $i = \text{root}(\mathcal{T})$. A recursive expansion of the generators with (2.1)–(2.2) yields a telescoping representation [24]:

$$A = D^{(L)} + U^{(L)}(U^{(L-1)}(\cdots(U^{(2)}B^{(2)}(V^{(2)})^* + B^{(3)})\cdots)(V^{(L-1)})^* + B^{(L)})(V^{(L)})^*, \tag{4.2}$$

where $D^{(l)}$, $U^{(l)}$, $V^{(l)}$, and $B^{(l)}$ are block diagonal matrices defined as follows:

$$D^{(L)} = \operatorname{diag}(D_i)_{i:\text{ leaf}},$$

$$U^{(l)} = \begin{cases} \operatorname{diag}(U_i)_{i:\text{ leaf}} & \text{if } l = L, \\ \operatorname{diag}\left(\begin{pmatrix} R_{c_1} \\ R_{c_2} \end{pmatrix}\right)_{c_1,c_2:\text{ children of } i} & \text{otherwise,} \end{cases}$$

$$V^{(L)} = \begin{cases} \operatorname{diag}(V_i)_{i:\text{ leaf}} & \text{if } l = L, \\ \operatorname{diag}\left(\begin{pmatrix} W_{c_1} \\ W_{c_2} \end{pmatrix}\right)_{c_1,c_2:\text{ children of } i} & \text{otherwise,} \end{cases}$$

$$B^{(l)} = \operatorname{diag}\left(\begin{pmatrix} 0 & B_i \\ B_{\mathrm{sib}(i)} & 0 \end{pmatrix}\right)_{i:\text{ left nodes at level } l.}$$

We point out that $||U^{(l)}||_2$ and $||V^{(l)}||_2$ are bounded. In fact, the strong rank-revealing factorization [14] bounds all the entries of E in (2.4) by a small constant c_0 (larger than 1). Since each diagonal block of $U^{(l)}$ or $V^{(l)}$ has size $O(r\times r)$, its Frobenius norm is bounded by $O(r)$, and so is its 2-norm. Thus, we have the following lemma.

LEMMA 4.1. *The generators in the randomized HSS construction satisfy*

$$1 \le ||U^{(l)}||_2 = O(r), \quad 1 \le ||V^{(l)}||_2 = O(r), \quad 2 \le l \le L.$$

Specifically, if E_i and F_i in (2.8) *and* (2.9) *are $r \times r$, and the magnitudes of their entries are bounded by a constant c_0, then $||U^{(l)}||_2 \le 1 + c_0 r$, $||V^{(l)}||_2 \le 1 + c_0 r$.*

The following theorem provides the approximation error for the HSS construction Algorithm 1. (As mentioned in Remark 2.1, in our theorems, we assume that the off-diagonal blocks are compressed to the accuracy τ with probability 1.)

THEOREM 4.2. *Let A in* (4.2) *be the HSS approximation to $\mathcal{C}$ after the randomized HSS construction, and τ be the relative tolerance in truncating the singular values of the HSS blocks. Then*

$$\mathcal{C} = A + \mathcal{E}, \tag{4.3}$$

where

$$\|\mathcal{E}\|_F = O(\tau \cdot \max(r^{2L-3/2}, n))\, \|\mathcal{C}\|_F = O(\tau \cdot \max(r^{O(\log \frac{n}{r})}, n))\, \|\mathcal{C}\|_F\,. \tag{4.4}$$

Proof. The proof follows the construction process. At the leaf level L, all D_i's are the diagonal blocks of $\mathcal{C}$, so $D^{(L)}$ is exact.

When we compute U_i in (2.8), an approximation error as in (2.5) is introduced. Since $\mathcal{C}_i^-$ has its row size $2r$, we have

$$\begin{aligned} ||\mathcal{C}_i^- - U_i \mathcal{C}_i^-|_{\hat{\mathbf{I}}_i}||_F &\le \sqrt{2r}||\mathcal{C}_i^- - U_i \mathcal{C}_i^-|_{\hat{\mathbf{I}}_i}||_2 \\ &\le \sqrt{2r}(1 + 11\sqrt{2r \cdot \min(2r, n-2r)})\sigma_{r+1} \\ &\le \sqrt{2r}(1+22r)\tau||\mathcal{C}_i^-||_2 = O(\tau r\sqrt{r})||\mathcal{C}_i^-||_F. \end{aligned} \tag{4.5}$$

When U_i for all the leaves i are computed, we get

$$\mathcal{C} - D^{(L)} = U^{(L)} \mathcal{C}_1^{(L)} + \mathcal{E}_1^{(L)}, \tag{4.6}$$

where $\mathcal{C}_1^{(L)}$ is the submatrix of $\mathcal{C}$ with row index set $\cup_{i:\ \text{leaf}} \hat{I}_i$ and with the diagonal blocks set to be zeros, and $\mathcal{E}_1^{(L)}$ satisfies

$$||\mathcal{E}_1^{(L)}||_F = O(\tau r\sqrt{r})\,\|\mathcal{C}\|_F\,. \tag{4.7}$$

Similarly, when we compute $V^{(L)}$, we introduce an error $\mathcal{E}_2^{(L)}$ satisfying the same relationship as in (4.7). For convenience, we write $\mathcal{E}^{(L)} \equiv \mathcal{E}_1^{(L)} + \mathcal{E}_2^{(L)}$. Then

$$\mathcal{C} - D^{(L)} = U^{(L)}\mathcal{C}_2^{(L)}(V^{(L)})^* + \mathcal{E}^{(L)},$$

where $\mathcal{C}_2^{(L)}$ is a submatrix of matrix $\mathcal{C}$ with appropriate zero blocks, and $\mathcal{E}^{(L)}$ also satisfies the same relationship as in (4.7). $\mathcal{C}_2^{(L)}$ is used to extract the B_i generator for all leaves i and also for the upper level compression.

The derivation can be generalized to any level l for the computation of $U^{(l)}$ and $V^{(l)}$, and we obtain a matrix $\mathcal{E}^{(l)}$ satisfying the same bound in (4.7). The reason is that *the blocks compressed are submatrices of* $\mathcal{C}$ due to the SPRR factorizations. The overall HSS construction then yields

$$\mathcal{C} - D^{(L)} = U^{(L)}(U^{(L-1)}(\cdots(U^{(2)}B^{(2)}(V^{(2)})^* + \mathcal{E}^{(2)})\cdots) + B^{(L)})(V^{(L)})^* + \mathcal{E}^{(L)}.$$

According to (4.2), this equation can be reorganized as (4.3), where

$$\mathcal{E} = (U^{(L)}\cdots U^{(3)})\mathcal{E}^{(2)}(V^{(L)}\cdots V^{(3)})^* + \cdots + U^{(L)}\mathcal{E}^{(L-1)}(V^{(L)})^* + \mathcal{E}^{(L)}. \tag{4.8}$$

Notice that the matrices $(U^{(L)}\cdots U^{(l+1)})$ and $(V^{(L)}\cdots V^{(l+1)})$ have sizes $n \times 2^l r$ for $l = 2, 3, \ldots, L-1$. Thus, Lemma 4.1 yields

$$||U^{(L)}\cdots U^{(l+1)}||_F \le \sqrt{2^l r}||U^{(L)}\cdots U^{(l+1)}||_2 = O(2^{l/2}r^{L-l+1/2}).$$

The same bound holds for $||V^{(L)}\cdots V^{(l+1)}||_F$.

Therefore, based on these bounds and Lemma 4.1, we have

$$\begin{aligned}
\|\mathcal{E}\|_F &\le \sum_{l=2}^{L-1} ||(U^{(L)}\cdots U^{(l+1)})\mathcal{E}^{(l)}(V^{(L)}\cdots V^{(l+1)})^*||_F + ||\mathcal{E}^{(L)}||_F \\
&= \sum_{l=2}^{L-1} O(2^l r^{2L-2l+1})||\mathcal{E}^{(l)}||_F + ||\mathcal{E}^{(L)}||_F \\
&= \sum_{l=2}^{L} O\left(r^{2L+1}\left(\frac{2}{r^2}\right)^l\right) O(\tau r\sqrt{r})\,\|\mathcal{C}\|_F \\
&= O(\tau\cdot\max(r^{2L-3/2}, n))\,\|\mathcal{C}\|_F = O(\tau\cdot\max(r^{2L-3/2}, n))\,\|\mathcal{C}\|_F\,. \qquad \square
\end{aligned} \tag{4.9}$$

The approximate error from a standard HSS construction method as in [39] can be similarly derived as follows.

COROLLARY 4.3. *If a truncated SVD with* (4.1) *is used for the compression, then* (4.4) *becomes*

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

Proof. The proof is similar to that in Theorem 4.2. Here we only emphasize the differences between these two proofs.

With a truncated SVD $\mathcal{C}_i^- \approx U_i(\Sigma_i \hat{V}_i^*) \equiv U_i K_i$, (4.5) is replaced by

$$||\mathcal{C}_i^- - U_i K_i||_F \le \sqrt{r}||\mathcal{C}_i^- - U_i K_i||_2 \le \tau\sqrt{r}||\mathcal{C}_i^-||_2 \le \tau\sqrt{r}||\mathcal{C}_i^-||_F.$$

Then (4.6) is replaced by

$$\mathcal{C} - D^{(L)} = U^{(L)} K^{(L)} + \mathcal{E}_1^{(L)},$$

where $K^{(L)}$ is formed by stacking the K_i matrices for all leaves i, with appropriate zero blocks inserted, and

$$||\mathcal{E}_1^{(L)}||_F = O(\tau\sqrt{r})\,\|\mathcal{C}\|_F\,. \tag{4.10}$$

Then consider the compression of $\mathcal{C}_i^|$, which yields an error $\mathcal{E}_2^{(L)}$ satisfying the same relationship as in (4.10). We can get

$$\mathcal{C} - D^{(L)} = U^{(L)} \mathcal{C}_2^{(L)} (V^{(L)})^* + \mathcal{E}^{(L)},$$

where $\mathcal{E}^{(L)} \equiv \mathcal{E}_1^{(L)} + \mathcal{E}_2^{(L)}$ satisfies the same relationship as in (4.10). Similarly, we can get $\mathcal{E}^{(l)}$ as in (4.8) satisfying the same relationship as in (4.10), due to the column orthonormality of $U^{(l)}$ and $V^{(l)}$. Thus, (4.9) becomes

$$||\mathcal{E}||_F \le \sum_{l=2}^{L} ||\mathcal{E}^{(l)}||_F = O(\tau L\sqrt{r})\,\|\mathcal{C}\|_F = O(\tau\sqrt{r}\log n)\,\|\mathcal{C}\|_F\,. \qquad \square$$

4.2. Stability for HSS URV factorization. Then we study the stability of the HSS URV factorization. (In this paper, to save space, we focus on the rounding errors in the HSS factorization and skip those from the HSS construction, which we will perform in future work. In practice, when modest accuracies such as six to ten digits are used in HSS constructions, the approximation errors often dominate the rounding errors.) According to (4.2) and similarly to [37], we can describe the URV factorization in the following nested representation. Let

- $A^{(l)}$ be the extended reduced matrix resulting from $A^{(l+1)}$ after the eliminations of the nodes at level $l+1$, with $A^{(L)} \equiv A$,
- $\Psi^{(l)}$ be a permutation matrix during the factorizations at level $l+1$ which performs all the merging steps on $A^{(l)}$ to form the reduced matrix, and
- $\mathbf{G}^{(l)}$ be the blocks eliminated from $A^{(l+1)}$ together with appropriate zero blocks and permutations ($\Psi^{(l)}$) (Figure 2.3(iv)).

Assume the nodes at a level l of $\mathcal{T}$ are $i_1, i_2, \ldots$. Define

$$\begin{aligned} \mathbf{Q}^{(l)} &= \mathrm{diag}(I, Q_{i_1}, Q_{i_2}, \ldots)\Psi^{(l)}, && l = L, L-1, \ldots, 2, \\ \mathbf{P}^{(l)} &= \mathrm{diag}(I, P_{i_1}, P_{i_2}, \ldots)\Psi^{(l)}, && l = L, L-1, \ldots, 2, \end{aligned}$$

where the identity matrices in diag() correspond to $\mathbf{G}^{(l+1)}$ and do not exist if $l = L$. Then the URV factorization process can be recursively represented by

$$A^{(l)} + \mathbf{G}^{(l)} = (\mathbf{Q}^{(l+1)})^* A^{(l+1)} \mathbf{P}^{(l+1)}, \quad l = L-1, L-2, \ldots, 1, \text{ or} \tag{4.11}$$

$$\begin{aligned} A = \ & \mathbf{Q}^{(L)}(\mathbf{Q}^{(L-1)}(\cdots \mathbf{Q}^{(2)}(A^{(1)} + \mathbf{G}^{(1)})(\mathbf{P}^{(2)})^{-1} \\ & + \cdots + \mathbf{G}^{(L-1)})(\mathbf{P}^{(L-1)})^{-1} + \mathbf{G}^{(L)})(\mathbf{P}^{(L)})^{-1}. \end{aligned} \tag{4.12}$$

First, we can show a result similar to the growth factor in LU factorizations [11]. It is easy to verify that, if the standard ULV factorization in [6] is computed, then $\mathbf{Q}^{(l)}$ and $\mathbf{P}^{(l)}$ are unitary and $||A^{(l)}||_2 \le ||A||_2$. Here in the URV factorizations, $\mathbf{Q}^{(l)}$ is unitary. $\mathbf{P}^{(l)}$ is not, but has a bounded norm, as shown below.

LEMMA 4.4. *For a node $i = 1, 2, \ldots, \mathrm{root}(\mathcal{T}) - 1$ of $\mathcal{T}$,*

$$||P_i||_F = O(r), \quad ||P_i^{-1}||_F = O(r).$$

Proof. $||P_i||_F = O(r)$ follows from the strong rank-revealing QR factorization in [14], where the magnitudes of the entries of F_i are bounded by a small constant. $||P_i^{-1}||_F = O(r)$ is obvious with

$$P_i^{-1} = \begin{pmatrix} 0 & I \\ I & F_i^* \end{pmatrix} \Upsilon_i^*. \qquad \square \tag{4.13}$$

Moreover, we have the following result. A similar one is first mentioned in [15] without proof.

THEOREM 4.5. *The URV factorization (in exact arithmetic) produces factors that satisfy*

$$\mathbf{Q}^{(l)} \text{ is unitary}, \quad ||\mathbf{P}^{(l)}||_2 \le \beta, \quad ||A^{(l)}||_2 \le \rho ||A||_2, \quad l = L, L-1, \ldots, 2,$$

where

$$\beta \equiv \max_{i=1:\mathrm{root}(\mathcal{T})-1} ||P_i||_2 = O(r), \quad \rho = O(r^{L-l}) = O(r^{O(\log \frac{n}{r})}). \tag{4.14}$$

Proof. Clearly, for $l = L, L-1, \ldots, 2$,

$$||\mathbf{P}^{(l)}||_2 \le \beta \equiv \max_{i=1:\mathrm{root}(\mathcal{T})-1} ||P_i||_2 \le \max_{i=1:\mathrm{root}(\mathcal{T})-1} ||P_i||_F = O(r).$$

Since the nonzero entries of $A^{(l)}$ form a submatrix of $(\mathbf{Q}^{(l+1)})^* A^{(l+1)} \mathbf{P}^{(l+1)}$ as in (4.11), we have

$$||A^{(l)}||_2 \le ||(\mathbf{Q}^{(l+1)})^* A^{(l+1)} \mathbf{P}^{(l+1)}||_2 \le \beta ||A^{(l+1)}||_2 = O(r) ||A^{(l+1)}||_2. \qquad \square$$

Thus, ρ is the 2-norm *growth factor*, which performs a role similar to the element growth factor ρ_0 in LU factorizations with partial pivoting. However, ρ here is much smaller than the classical worst case bound 2^n for ρ_0 [11].

Next, we perform the stability analysis. We use wide-hatted notation to mean the computed results. For example, $\widehat{\mathbf{P}}^{(l)} \equiv \mathrm{fl}(\mathbf{P}^{(l)})$ denotes the actually computed matrix for $\mathbf{P}^{(l)}$. Also let $\mathbf{u}$ be the unit roundoff or machine epsilon in IEEE double precision arithmetic. We study the numerical stability step by step.

The following lemma is a direct extension of the results from section 19.3 of [19], as similarly given in [1].

LEMMA 4.6. *Consider a numerical transformation Q^*D, where Q and D are $2r \times 2r$ matrices and Q is a product of r Householder matrices. Let*

$$\tilde{\gamma}_r = \frac{2\mathbf{c}r\mathbf{u}}{1 - 2\mathbf{c}r\mathbf{u}} < \frac{1}{r},$$

where $\mathbf{c}$ is a small positive constant. Then there exists a unitary matrix $\widetilde{Q}$, so that

$$\mathrm{fl}(Q^*D) = \widetilde{Q}^* D + \mathcal{Z}, \ ||\mathcal{Z}||_F \le 2r\tilde{\gamma}_r ||D||_F.$$

For general matrix multiplications, another result is given in section 3.5 of [19].

LEMMA 4.7. *Consider the numerical multiplication of two $2r \times 2r$ matrices D and P. We have*

$$\text{fl}(DP) = DP + \mathcal{H}, \quad ||\mathcal{H}||_F \le \gamma_r ||D||_F ||P||_F \quad \textit{with}$$
$$\gamma_r = \frac{2r\mathbf{u}}{1 - 2r\mathbf{u}}.$$

The combined rounding error in the multiplication steps (2.14) and (2.17) is given as follows.

LEMMA 4.8. *The numerical multiplications in $Q_i^* \tilde{D}_i P_i$ in the URV factorization satisfy*

$$\text{fl}(Q_i^* \tilde{D}_i P_i) = \widetilde{Q}_i^* \tilde{D}_i P_i + \mathcal{G}_i, \quad ||\mathcal{G}_i||_F = O(r\hat{\gamma}_r)||\tilde{D}_i||_F, \tag{4.15}$$

where $\widetilde{Q}_i$ is a unitary matrix, and

$$\hat{\gamma}_r = \gamma_r + r\tilde{\gamma}_r.$$

Proof. According to Lemmas 4.6 and 4.7,

$$\text{fl}((Q_i^* \tilde{D}_i)P_i) = (\widetilde{Q}_i^* \tilde{D}_i + \mathcal{Z}_i)P_i + \mathcal{H}_i = \widetilde{Q}_i^* \tilde{D}_i P_i + \mathcal{Z}_i P_i + \mathcal{H}_i,$$

where $\widetilde{Q}_i$ is a unitary matrix, and

$$||\mathcal{Z}_i||_F \le 2r\tilde{\gamma}_r ||\tilde{D}_i||_F, \quad ||\mathcal{H}_i||_F \le \gamma_r ||\widetilde{Q}_i^* \tilde{D}_i + \mathcal{Z}_i||_F ||P_i||_F.$$

Then

$$\begin{aligned} ||\mathcal{G}_i||_F \equiv ||\mathcal{Z}_i P_i + \mathcal{H}_i||_F &\le 2r\tilde{\gamma}_r ||\tilde{D}_i||_F ||P_i||_F + \gamma_r(||\tilde{D}_i||_F + ||\mathcal{Z}_i||_F)||P_i||_F \\ &= O(r^2\tilde{\gamma}_r)||\tilde{D}_i||_F + O(r\gamma_r)||\tilde{D}_i||_F + O(\mathbf{u}^2) = O(r\hat{\gamma}_r)||\tilde{D}_i||_F, \end{aligned}$$

where Lemma 4.4 is used. ∎

THEOREM 4.9. *For $l = 1, 2, \ldots, L-1$,*

$$\text{fl}((\widehat{\mathbf{Q}}^{(l+1)})^* \widehat{A}^{(l+1)} \widehat{\mathbf{P}}^{(l+1)}) = (\widetilde{\mathbf{Q}}^{(l+1)})^* \widehat{A}^{(l+1)} \widehat{\mathbf{P}}^{(l+1)} + \mathcal{G}^{(l)},$$

where $\widetilde{\mathbf{Q}}^{(l+1)}$ is a unitary matrix, and

$$||\widehat{\mathbf{P}}^{(l+1)}||_F \le \beta\sqrt{n} = O(r\sqrt{n}), \tag{4.16}$$

$$||\mathcal{G}^{(l)}||_F = O(r\hat{\gamma}_r)||\widehat{A}^{(l+1)}||_F, \tag{4.17}$$

$$||\widehat{A}^{(l+1)}||_F = (\beta\sqrt{n})^{L-l-1}||A||_F = O((r\sqrt{n})^{L-l-1})||A||_F. \tag{4.18}$$

Thus,

$$||\mathcal{G}^{(l)}||_F = O((r\sqrt{n})^{L-l-1} r\hat{\gamma}_r)||A||_F. \tag{4.19}$$

Proof. (4.16) holds due to the way $\widehat{\mathbf{P}}^{(l+1)}$ is computed with strong rank-revealing QR factorizations, as mentioned in Lemma 4.4. That is,

$$||\widehat{\mathbf{P}}^{(l+1)}||_F \le \sqrt{n}||\widehat{\mathbf{P}}^{(l+1)}||_2 \le \beta\sqrt{n}.$$

(4.17) follows from Lemma 4.8. We then prove (4.18) by induction for $l = L-1, L-2, \ldots, 1$. For $l = L-1$,

$$||\widehat{A}^{(l+1)}||_F \equiv ||A||_F.$$

Assume the result holds for $l < L-1$. Since the nonzero entries of $\widehat{A}^{(l)}$ form a submatrix of $\mathrm{fl}((\widetilde{\mathbf{Q}}^{(l+1)})^* \widehat{A}^{(l+1)} \widehat{\mathbf{P}}^{(l+1)})$, we have

$$\begin{aligned}||\widehat{A}^{(l)}||_F &\le ||(\widetilde{\mathbf{Q}}^{(l+1)})^* \widehat{A}^{(l+1)} \widehat{\mathbf{P}}^{(l+1)} + \mathcal{G}^{(l)}||_F \le ||\widehat{A}^{(l+1)} \widehat{\mathbf{P}}^{(l+1)}||_F + ||\mathcal{G}^{(l)}||_F \\ &\le (\beta\sqrt{n})^{L-l-1} ||A||_F \cdot \beta\sqrt{n} + r\hat{\gamma}_r ||A||_F = O((\beta\sqrt{n})^{L-l})||A||_F. \qquad \square\end{aligned}$$

We need an additional lemma for the stability analysis.

LEMMA 4.10.

$$||\mathbf{P}^{(L)}\mathbf{P}^{(L-1)} \cdots \mathbf{P}^{(l)}||_F \le O(\sqrt{2^L r}) = O(\sqrt{nr}), \tag{4.20}$$

$$||(\mathbf{P}^{(L)}\mathbf{P}^{(L-1)} \cdots \mathbf{P}^{(l)})^{-1}||_F \le O(\sqrt{2^L r}) = O(\sqrt{nr}). \tag{4.21}$$

Proof. Noticing (2.12) for P_i and (4.13) for P_i^{-1}, we only need to prove (4.20).

Consider the multiplication involving P_i and $\mathrm{diag}(P_{c_1}, P_{c_2})$ for a node i at level l and its children c_1, c_2 at level $l+1$, respectively. According to the HSS construction, the computation of V_{c_1}, V_{c_2} only uses the columns that correspond to the identity matrix in the representation of V_i in (2.8). Thus, P_i is only multiplied by the identity matrices in the presentations of P_{c_1}, P_{c_2} as in (2.12). More specifically, without loss of generality, assume $\Upsilon_i, \Upsilon_{c_1}, \Upsilon_{c_1}$ are identity matrices, and then the multiplication looks like

$$\begin{aligned}&\begin{pmatrix} \begin{pmatrix} -F_{c_1}^* & I \\ I & 0 \end{pmatrix} & \\ & \begin{pmatrix} -F_{c_2}^* & I \\ I & 0 \end{pmatrix} \end{pmatrix} \begin{pmatrix} I & & \\ & -F_i^* & I \\ & I & 0 \end{pmatrix} \\ &= \begin{pmatrix} \underbrace{\begin{pmatrix} & I & & \\ I & & & \\ & & & I \\ & & I & \end{pmatrix}}_{\Pi_1} + \mathrm{diag}(-F_{c_1}^*, 0, -F_{c_2}^*, 0) \end{pmatrix} \begin{pmatrix} \underbrace{\begin{pmatrix} I & & & \\ & & & I \\ & & I & \\ & I & & \end{pmatrix}}_{\Pi_2} + \mathrm{diag}(0, -F_i^*, 0, 0) \end{pmatrix} \\ &= \Pi_1\Pi_2 + \Pi_1\,\mathrm{diag}(0, -F_i^*, 0, 0) + \mathrm{diag}(-F_{c_1}^*, 0, -F_{c_2}^*, 0)\Pi_2 \\ &= \begin{pmatrix} -F_{c_1}^* & -F_i^* & & I \\ I & & & \\ & & I & -F_{c_2}^* \\ & & & I \end{pmatrix}.\end{aligned}$$

Clearly, the blocks $-F_i^*$, $-F_{c_1}^*$, and $-F_{c_2}^*$ appear individually in the product, since they always appear in different block columns of the result.

Therefore, the nonzero blocks of $\mathbf{P}^{(L)}\mathbf{P}^{(L-1)} \cdots \mathbf{P}^{(l)}$ include the blocks of a permutation matrix and $-F_i^*$ for the nodes i at levels l to L of $\mathcal{T}$. Noticing (2.12), we have

$$||\mathbf{P}^{(L)}\mathbf{P}^{(L-1)} \cdots \mathbf{P}^{(l)}||_F \le \sqrt{\sum_{i=1}^{\mathrm{root}(\mathcal{T})-1} (2||I_r||_F^2 + ||F_i||_F^2)} \le \sqrt{\sum_{i=1}^{\mathrm{root}(\mathcal{T})-1} ||P_i||_F^2} = O(\sqrt{2^L r}),$$

where I_r is the identity matrix of size r. $\square$

Remark 4.1. With similar ideas, it is possible to improve the growth factor in Theorem 4.5 with Frobenius norms.

We are then ready to present our main stability result.

THEOREM 4.11. *The URV factorization is backward stable. That is, it produces a numerical factorization*

$$\begin{aligned} A+\mathcal{G} = \widetilde{\mathbf{Q}}^{(L-1)}(\widetilde{\mathbf{Q}}^{(L-2)}(\cdots(\widetilde{\mathbf{Q}}^{(2)}(\widehat{A}^{(2)}+\widehat{\mathbf{G}}^{(2)})(\widehat{\mathbf{P}}^{(2)})^{-1}) \\ +\cdots+\widehat{\mathbf{G}}^{(L-2)})(\widehat{\mathbf{P}}^{(L-2)})^{-1}+\widehat{\mathbf{G}}^{(L-1)})(\widehat{\mathbf{P}}^{(L-1)})^{-1}, \end{aligned}$$

where $\widetilde{\mathbf{Q}}^{(l)}$ *is a unitary matrix,* $\widehat{\mathbf{P}}^{(l)}$ *satisfies* (4.16), *and*

$$||\mathcal{G}||_F = O((r\sqrt{n})^{L-2}\sqrt{r}\hat{\gamma}_r)||A||_F = O((r\sqrt{n})^{O(\log\frac{n}{r})}\sqrt{r}\hat{\gamma}_r)||A||_F.$$

Proof. Theorem 4.9 means, for $l = L-1, L-2, \ldots, 1$,

$$\widehat{A}^{(l)}+\widehat{\mathbf{G}}^{(l)} = (\widetilde{\mathbf{Q}}^{(l+1)})^* A^{(l+1)}\widehat{\mathbf{P}}^{(l+1)}+\mathcal{G}^{(l)},$$

where we assume that the formation of $\widehat{\mathbf{G}}^{(l)}$ does not introduce any error (or this error can be absorbed by $\mathcal{G}^{(l)}$). Then,

$$\widehat{A}^{(l+1)} = (\widetilde{\mathbf{Q}}^{(l+1)})^*(\widehat{A}^{(l)}+\widehat{\mathbf{G}}^{(l)}-\mathcal{G}^{(l)})(\widehat{\mathbf{P}}^{(l+1)})^{-1}.$$

Thus, according to (4.12) and similar to the procedure in [1], we have

$$\begin{aligned} A \equiv \widehat{A}^{(L)} &= \widetilde{\mathbf{Q}}^{(L)}(\widetilde{\mathbf{Q}}^{(L-1)}(\cdots(\widetilde{\mathbf{Q}}^{(2)}(\widehat{A}^{(1)}+\widehat{\mathbf{G}}^{(1)}-\mathcal{G}^{(1)})(\widehat{\mathbf{P}}^{(2)})^{-1}) \\ &\quad +\cdots)(\widehat{\mathbf{P}}^{(L-1)})^{-1}+\widehat{\mathbf{G}}^{(L-1)}-\mathcal{G}^{(L-1)})(\widehat{\mathbf{P}}^{(L)})^{-1} \\ &= \widetilde{\mathbf{Q}}^{(L)}(\widetilde{\mathbf{Q}}^{(L-1)}(\cdots(\widetilde{\mathbf{Q}}^{(2)}(\widehat{A}^{(1)}+\widehat{\mathbf{G}}^{(1)})(\widehat{\mathbf{P}}^{(2)})^{-1}) \\ &\quad +\cdots+\widehat{\mathbf{G}}^{(L-2)})(\widehat{\mathbf{P}}^{(L-1)})^{-1}+\widehat{\mathbf{G}}^{(L-1)})(\widehat{\mathbf{P}}^{(L)})^{-1}-\mathcal{G}, \end{aligned}$$

where

$$\begin{aligned} \mathcal{G} = &(\widetilde{\mathbf{Q}}^{(L)}\widetilde{\mathbf{Q}}^{(L-1)}\cdots\widetilde{\mathbf{Q}}^{(2)})\mathcal{G}^{(1)}(\widehat{\mathbf{P}}^{(L)}\cdots\widehat{\mathbf{P}}^{(2)})^{-1} \\ &+\cdots+(\widetilde{\mathbf{Q}}^{(L)}\widetilde{\mathbf{Q}}^{(L-1)})\mathcal{G}^{(L-2)}(\widehat{\mathbf{P}}^{(L)}\widehat{\mathbf{P}}^{(L-1)})^{-1}+\widetilde{\mathbf{Q}}^{(L)}\mathcal{G}^{(L-1)}(\widehat{\mathbf{P}}^{(L)})^{-1}. \end{aligned}$$

Due to the construction of $\widehat{\mathbf{P}}^{(l)}$, Lemma 4.10 means

$$||(\widehat{\mathbf{P}}^{(L)}\widehat{\mathbf{P}}^{(L-1)}\cdots\widehat{\mathbf{P}}^{(l)})^{-1}||_F = O(\sqrt{2^L r}) = O(\sqrt{nr}).$$

Then with (4.19), we have

(4.22)

$$\begin{aligned} ||\mathcal{G}||_F &\le \sum_{l=2}^{L}||(\widetilde{\mathbf{Q}}^{(L)}\cdots\widetilde{\mathbf{Q}}^{(l)})\mathcal{G}^{(l-1)}(\widehat{\mathbf{P}}^{(L)}\cdots\widehat{\mathbf{P}}^{(l)})^{-1}||_F \\ &\le \sum_{l=2}^{L}||(\widehat{\mathbf{P}}^{(L)}\cdots\widehat{\mathbf{P}}^{(l)})^{-1}||_F||\mathcal{G}^{(l-1)}||_F = O(\sqrt{nr})\sum_{l=2}^{L}||\mathcal{G}^{(l-1)}||_F \\ &= O(\sqrt{nr})\sum_{l=2}^{L}O((r\sqrt{n})^{L-l-1}r\hat{\gamma}_r)||A||_F = O((r\sqrt{n})^{L-2}\sqrt{r}\hat{\gamma}_r)||A||_F. \end{aligned}$$

∎

Clearly, the error grows following a factor $\hat{\rho} = O((r\sqrt{n})^{O(\log\frac{n}{r})})$, which is in a higher order than ρ in (4.14), but is still in a much smaller order than the worst case element growth factor 2^n in Gaussian elimination with partial pivoting.

Remark 4.2. For the case where the row size m of A is different from the column size n, the above derivation of the backward stability of the URV factorization remains nearly identical, with certain n replaced by m. For example, if $m \geq n$, then n on the right-hand sides of (4.18), (4.19), etc. is replaced by m, which can be seen from the proof of (4.18). Similarly, the proof in (4.22) still holds due to the orthogonality of the $m \times m$ matrices $\widetilde{\mathbf{Q}}^{(l)}$, except that n in the final result is replaced by m.

A similar method can be used to prove the stability of the standard HSS ULV factorization in [6, 39].

COROLLARY 4.12. *The standard HSS ULV factorization is backward stable. That is, it produces a numerical factorization*

$$\begin{aligned} A + \mathcal{G} = \widetilde{\mathbf{Q}}^{(L-1)}(\widetilde{\mathbf{Q}}^{(L-2)}(\cdots(\widetilde{\mathbf{Q}}^{(2)}(\widehat{A}^{(2)} + \widehat{\mathbf{G}}^{(2)})(\widetilde{\mathbf{P}}^{(2)})^*) \\ + \cdots + \widehat{\mathbf{G}}^{(L-2)})(\widetilde{\mathbf{P}}^{(L-2)})^* + \widehat{\mathbf{G}}^{(L-1)})(\widetilde{\mathbf{P}}^{(L-1)})^*, \end{aligned}$$

where $\widetilde{\mathbf{Q}}^{(l)}$ *and* $\widetilde{\mathbf{P}}^{(l)}$ *are unitary matrices, and*

$$||\mathcal{G}||_F \leq O(rL\tilde{\gamma}_r)\,\|A\|_F = O(r(\log n)\tilde{\gamma}_r)\,\|A\|_F\,.$$

Proof. The proof is similar to that in Theorem 4.11, except that $\widetilde{\mathbf{P}}^{(l)}$ is now unitary. Based on Lemma 4.6, (4.15) becomes

$$\mathrm{fl}(Q_i^* \tilde{D}_i P_i) = \widetilde{Q}_i^* \tilde{D}_i \widetilde{P}_i + \mathcal{G}_i, \quad ||\mathcal{G}_i||_F = O(r\tilde{\gamma}_r)||\tilde{D}_i||_F,$$

where $\widetilde{Q}_i$ and $\widetilde{P}_i$ are unitary. This means that the results in Theorem 4.9 are replaced by

$$\begin{aligned} \mathrm{fl}((\widehat{\mathbf{Q}}^{(l+1)})^* \widehat{A}^{(l+1)} \widehat{\mathbf{P}}^{(l+1)}) = (\widetilde{\mathbf{Q}}^{(l+1)})^* \widehat{A}^{(l+1)} \widetilde{\mathbf{P}}^{(l+1)} + \mathcal{G}^{(l)}, \\ ||\mathcal{G}^{(l)}||_F = O(r\tilde{\gamma}_r)||\widehat{A}^{(l+1)}||_F = O(r\tilde{\gamma}_r)||A||_F. \end{aligned}$$

Then (4.22) becomes

$$\begin{aligned} ||\mathcal{G}||_F &\leq \sum_{l=2}^{L} ||(\widetilde{\mathbf{Q}}^{(L)} \cdots \widetilde{\mathbf{Q}}^{(l)})\mathcal{G}^{(l-1)}(\widetilde{\mathbf{P}}^{(L)} \cdots \widetilde{\mathbf{P}}^{(l)})^*||_F \\ &\leq \sum_{l=2}^{L} ||\mathcal{G}^{(l-1)}||_F = O(rL\tilde{\gamma}_r)||A||_F. \end{aligned}$$

∎

We can similarly show that the URV solution is stable. The details are skipped. Interested readers are referred to the stability analysis of QR least squares solutions and of triangular solutions in [11], as well as some results in [1].

Remark 4.3. According to our analysis, the standard HSS construction and factorization methods in [6, 39] (where the basis matrices U_i, V_i have orthonormal columns) have better error and stability bounds, respectively, than the randomized HSS methods here (where the basis matrices have special structures as in (2.8)–(2.9)). However, in practice, there is no significant difference in the accuracy and stability. Furthermore, the structures in (2.8)–(2.9) enable fast computations as in (2.13). We also avoid the extra work to orthonormalize the columns of the basis matrices.

5. Numerical experiments. We test the following methods, including our three new methods and three older ones for comparison:

- `URV`: our new superfast method based on the URV factorization of the HSS approximations to $\mathcal{C}$ as in section 2;
- `NE1`: our new superfast method based on the normal equation method in section 3.1;
- `NE2`: our new superfast method based on the normal equation method in section 3.2;
- `QR`: the standard least squares QR algorithm based on the economic QR factorization (provided in MATLAB) followed by a triangular solution with the R and Q factors;
- `TLLS`: the fast algorithm in [31] based on pseudoinverses;
- `Super`: the superfast algorithm in [34] based on augmented matrices.

All the methods are implemented in MATLAB, including the codes for `TLLS` and `Super` from their original authors. Our code is available at http://www.math.purdue.edu/~xiaj/work/toepls_code.zip. The numerical results are computed in double precision. The costs and accuracies are compared. The code includes lines to count the flops (number of floating point operations) for all the steps and subroutines. As usual, one addition, multiplication, or division is counted as one flop. For commonly used basic or internal routines, the standard theoretical counts are used, with the low-order terms dropped. For example, for the QR factorization of a tall and skinny $m \times r$ matrix, we use the count $2r^2(m - \frac{r}{3})$.

The following notation is used in the tests:

- m_1 (n_1): leaf level HSS block row (column) size;
- $\tilde{r}$: sampling size for `URV`;
- $\hat{r}$: sampling size for `NE1` and `NE2`;
- $\mathbf{r} = \frac{\|T^T(T\tilde{x}-b)\|_2}{\|T^Tb\|_2}$ and $\tilde{\mathbf{r}} = \frac{\|T^T(T\tilde{x}-b)\|_2}{\||T^T|(|T||\tilde{x}|+|b|)\|_2}$: measurement of the orthogonality of $T\tilde{x} - b$ (with respect to T^T), or relative residuals of the normal equation, where $\tilde{x}$ is the numerical solution.

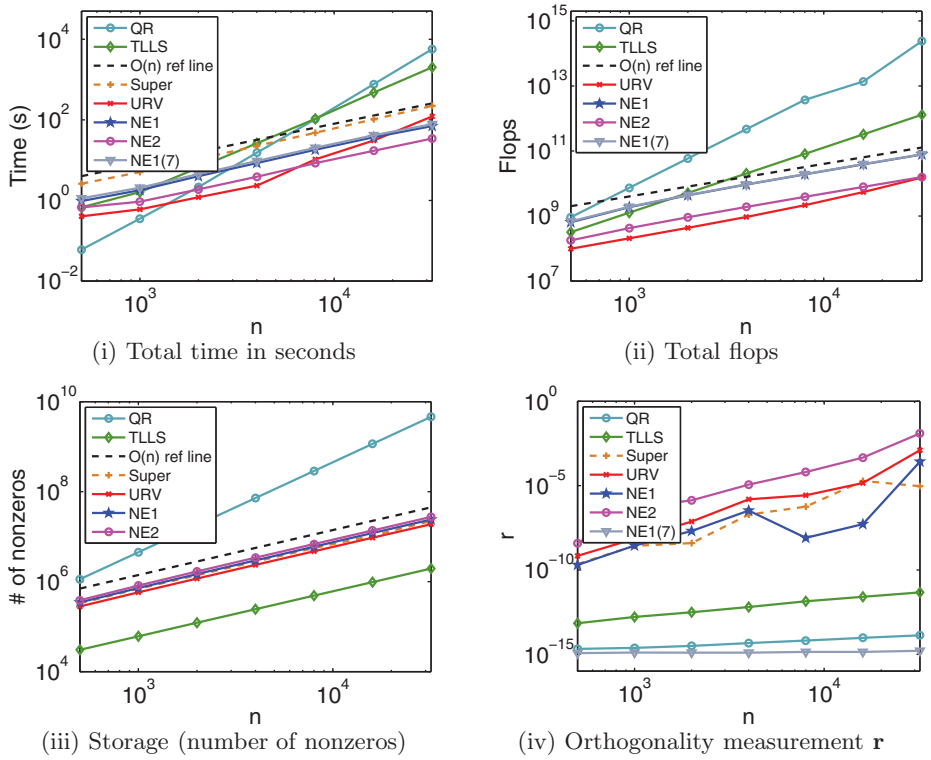
We test some Toeplitz systems $Tx = b$ which are inconsistent, with b randomly generated from the uniform distribution on $(0, 1)$ as in [13].

Example 1. A random Toeplitz matrix T defined by the vector which is generated with the MATLAB function randn:

$$t_{-(n-1):(m-1)} = \text{randn}(m + n - 1, 1).$$

T is generally well conditioned.

With this example, we show that our methods are superfast, more generally applicable, and faster than the other three methods, while giving comparable accuracies. First, we fix $m = 2n$. For n ranging from 500 to 32,000, we report the costs and accuracies of the methods. Here (and also in the other examples), we choose $m_1 = 250$, $n_1 = 125$, $\tilde{r} = 50$, and $\hat{r} = 70$. The total computation time, flops, storage (number of entries in all the HSS generators and factors), and accuracies are shown in Figure 5.1. (For our methods, the total cost includes the individual costs in all the steps.) A dotted reference line for $\hat{c}n$ with an appropriate constant $\hat{c}$ is also plotted and marked as "$O(n)$ reference line." Clearly, the four superfast methods (`URV`, `NE1`, `NE2`, and `Super`) have roughly $O(n)$ costs. Our three methods are all faster than `Super`. When n is sufficiently large, they are much faster than both `QR` and the fast method `TLLS`. `URV` and `NE2` have the fewer flops in Figure 5.1(ii). (When m is closer to n, `URV` becomes the fastest.)



(i) Total time in seconds (ii) Total flops

(iii) Storage (number of nonzeros) (iv) Orthogonality measurement $\mathbf{r}$

FIG. 5.1. *Example* 1, *random matrix: Numerical results with* $m = 2n$ *for* n *ranging from* 500 *to* 32,000, *where the flops for* `Super` *is not available, and* `NE1`(7) *denotes* `NE1` *followed by* 7 *steps of iterative refinement.*

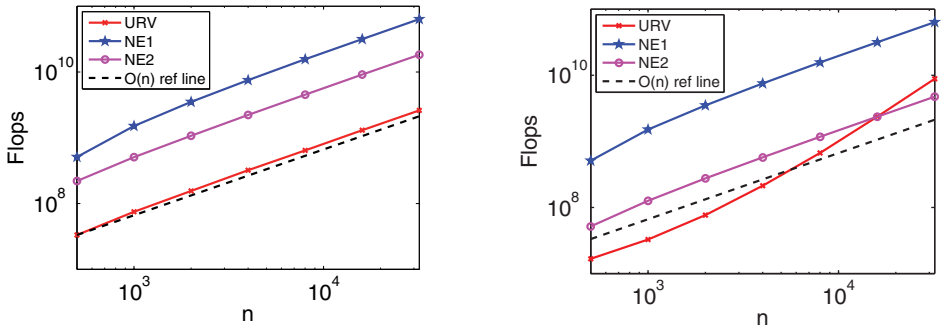
The storage of the four superfast methods are comparable (nearly $O(n)$) and are much better than that of `QR`. `TLLS` has a storage scheme mainly based on the displacement equation and is the most memory efficient.

The accuracies of the four superfast methods are also comparable. The measurement of the orthogonality $\mathbf{r}$ is shown in Figure 5.1(iv). `QR` is the most accurate. However, with a few steps of iterative refinements, `NE1` and `NE2` can reach similar accuracies. The cost of the iterative refinement is very low since the number of steps is small and each HSS solution step costs about $O(n)$ and is much faster than the HSS construction and factorization.

We would like to mention that the accuracy of the methods may decrease when n increases. This is partly because we chose a fixed sampling size, and is also consistent with the approximation error in Theorem 4.2 and the backward error in Theorem 4.11, which grow with n. In fact, in [13] for Toeplitz least squares, the author explicitly includes the square root of the matrix size in the denominator of the error report. Thus, for larger n, a larger sampling size and/or more iterative refinement steps in `NE1` and `NE2` (much smaller than n) may be needed.

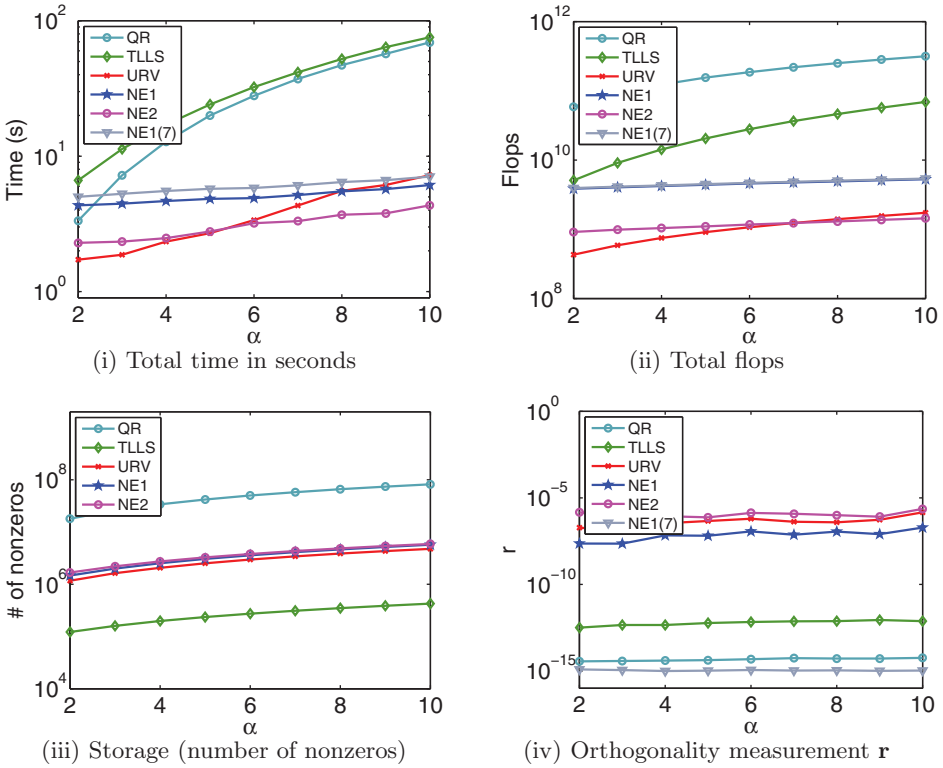
For our three methods, we also show the actual costs of the HSS construction and factorization steps in Figure 5.2, which are the major computations. Each step costs about $O(n)$ flops or slightly higher.

Remark 5.1. The slopes of the curves for `URV` in Figure 5.2(ii) are higher than that of the $O(n)$ reference line, because the ratio of the row and column dimensions of the reduced matrices gets larger and larger at higher elimination levels. The current



(i) Construction cost (ii) Factorization cost

FIG. 5.2. *Example* 1, *random matrix: Flops of the individual steps of* URV, NE1, *and* NE2, *with* $m = 2n$ *for* n *ranging from* 500 *to* 32,000.



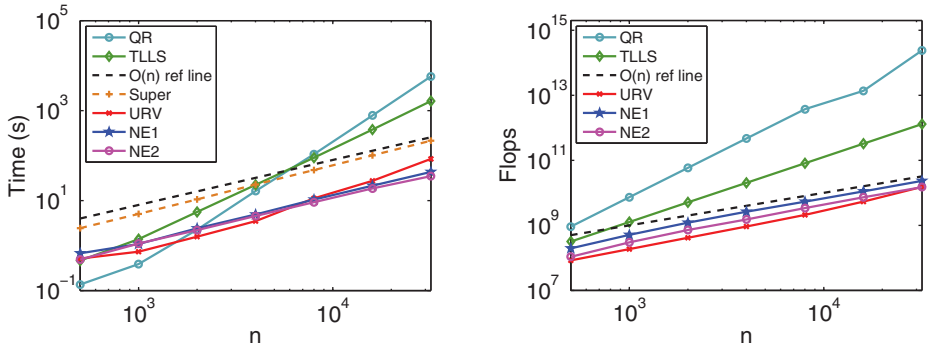
(i) Total time in seconds (ii) Total flops

(iii) Storage (number of nonzeros) (iv) Orthogonality measurement $\mathbf{r}$

FIG. 5.3. *Example* 1, *random matrix: Numerical results with* $n = 2000$ *and* $m = \alpha n$ *for* α *ranging from* 2 *to* 10, *where* NE1(7) *denotes* NE1 *followed by* 7 *steps of iterative refinement.*

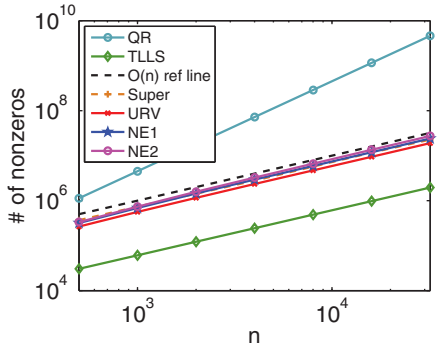
code only includes the row size reduction (section 2.3.1) for the leaf level. With additional size reductions at more levels, the costs of URV can be further reduced.

Next, we set $n = 2000$, $m = \alpha n$, and $m_1 = \alpha n_1$ with α ranging from 2 to 10. Here, Super only works for certain special α and is not included in the test. See Figure 5.3. The comparison of our methods with the others is similar. Since n is fixed, the costs of NE1 and NE2 remain almost constant. The flops of URV increase very slowly.

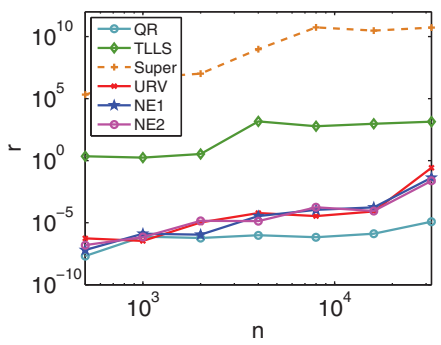


(i) Total time in seconds

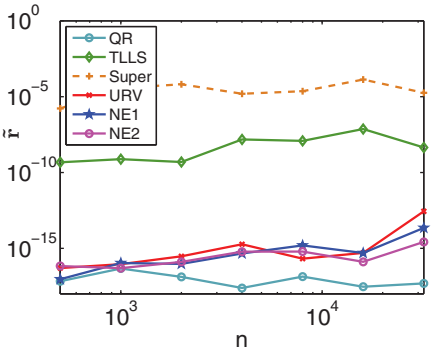
(ii) Total flops



(iii) Storage/number of entries in HSS representations and factors



(iv) Orthogonality measurement $\mathbf{r}$



(v) Orthogonality measurement $\tilde{\mathbf{r}}$

FIG. 5.4. *Example* 2, *KMS matrix: Numerical results with* $\rho = 0.99999$.

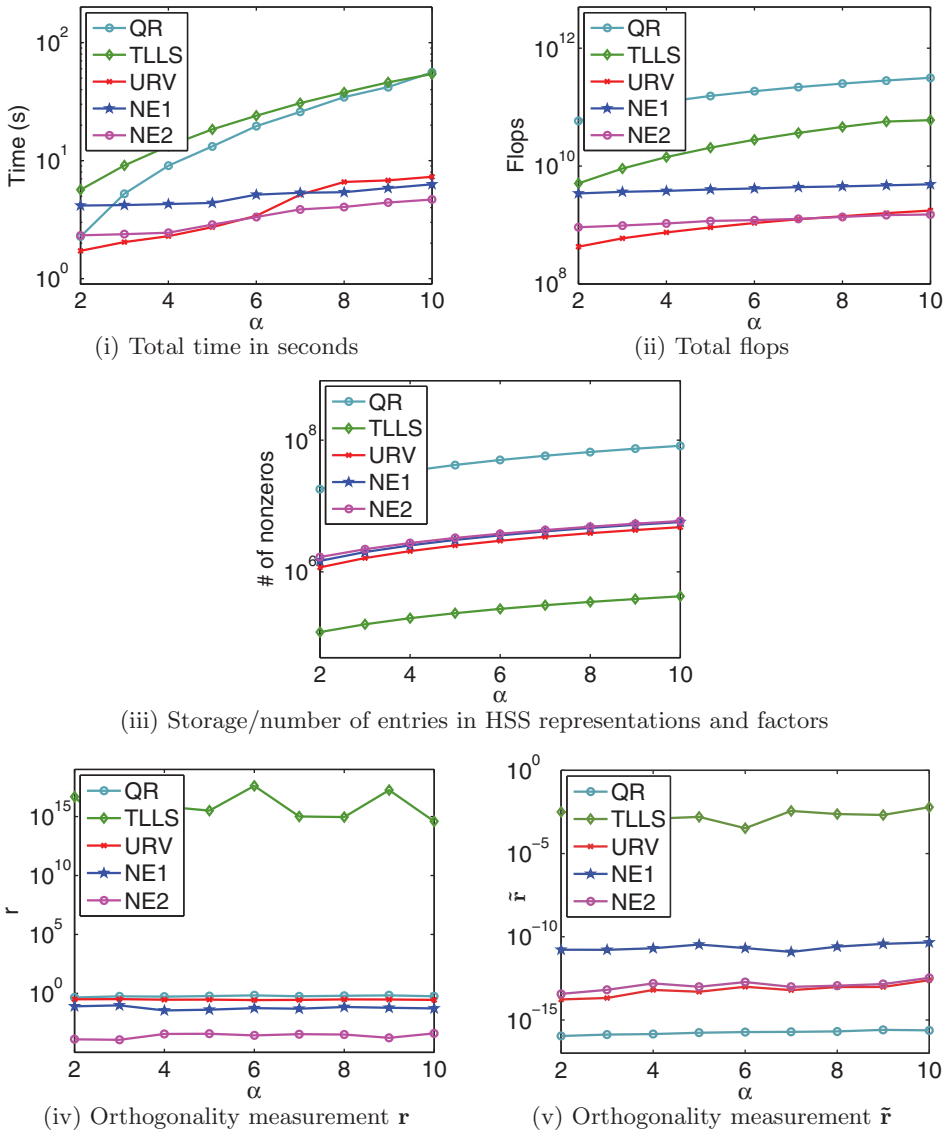
Example 2. The KMS matrix T [33] defined by

$$t_{i-j} = \rho^{|i-j|}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

We choose $\rho = 0.99999$, and T is ill conditioned.

Since the problem is ill conditioned and the condition of $T^T T$ is even worse, we use both $\mathbf{r}$ and $\tilde{\mathbf{r}}$ to measure the orthogonality of $T\tilde{x} - b$. Our superfast methods can achieve accuracies comparable to those of `QR`, and are much faster. On the other hand, `TLLS` and `Super` cost more and have much lower accuracies. See Figure 5.4.

When n is reasonably large, our methods are significantly faster. For example, when $n = 32,000$, `NE1` is about 133, 38, and 5 times faster than `QR`, `TLLS`, and `Super`, respectively. The difference in the flops is even more dramatic.



(i) Total time in seconds (ii) Total flops

(iii) Storage/number of entries in HSS representations and factors

(iv) Orthogonality measurement $\mathbf{r}$ (v) Orthogonality measurement $\tilde{\mathbf{r}}$

FIG. 5.5. *Example* 3, *prolate matrix: Numerical results with* $\omega = 0.44$, $n = 2000$, $m = \alpha n$, *and* α *ranging from* 2 *to* 10.

Example 3. The prolate Toeplitz matrix T [10, 12] defined by

$$t_0 = 2\omega, \quad t_k = \frac{\sin(2\omega k\pi)}{k\pi} \text{ for } k \neq 0, \quad \omega \in [0, 1/2].$$

T is very ill conditioned. Here, we use $\omega = 0.44$, and the 2-norm condition number of T is $O(10^{12})$.

Our new methods still give reasonable accuracies close to those of `QR`, though both `Super` (if it applies) and `TLLS` fail to provide solutions with any accuracies. We set $n = 2000$, $m = \alpha n$, and $m_1 = \alpha n_1$ with α ranging from 2 to 10. Our methods are also much faster. See Figure 5.5. The complexity of the new methods for varying n is similar to that in the previous examples, and is omitted.

6. Conclusions. In this work, we extend rank structured direct solutions with randomization to Toeplitz least squares problems, and generalize HSS representations to rectangular ones. We propose three superfast solvers, a URV one and two structured normal equation ones. The URV factorization generalizes the regular QR factorization. (Thus, it can be applied to $\mathcal{C}^*$ when $m < n$ so that zero columns are introduced into $\mathcal{C}$.) The detailed error and stability analysis for both randomized and classical HSS methods are given. The stability results are generally much better than existing results for standard LU factorizations with partial pivoting. Numerical experiments on some classical test examples show that the complexity and storage of the methods are roughly $O(m + n)$. The methods are compared with some recently proposed fast and superfast methods, and are generally much faster and more accurate, especially for ill-conditioned problems. We will further optimize the implementation in our future developments.

Acknowledgments. The authors would like to thank the anonymous referees for their valuable suggestions, and thank Marc Van Barel for providing the code for `Super`. The code for `TLLS` is downloaded from Giuseppe Rodriguez's webpage (http://bugs.unica.it/~gppe/soft). We are also grateful to Ming Gu for some early discussions and to Michael Stewart for sharing the manuscript [1].

REFERENCES

[1] T. Bella, V. Olshevsky, and M. Stewart, *Nested product decomposition of quasiseparable matrices*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1520–1555.
[2] D. A. Bini, G. Codevico, and M. Van Barel, *Solving Toeplitz least squares problems by means of Newton's iteration*, Numer. Algorithms, 33, (2003), pp. 93–103.
[3] A. W. Bojanczyk, R. P. Brent, F. R. de Hoog, and D. R. Sweet, *On the stability of the Bareiss and related Toeplitz factorization algorithms*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 40–57.
[4] R. H. Chan, J. G. Nagy, and R. J. Plemmons, *Displacement preconditioners for Toeplitz least squares iterations*, Electron. Trans. Numer. Anal., 2 (1994), pp. 44–56.
[5] R. H.-F Chan and X.-Q. Jin, *An Introduction to Iterative Toeplitz Solvers*, SIAM, Philadelphia, 2007.
[6] 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.
[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] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, and J. Zhu, *A superfast algorithm for Toeplitz systems of linear equations*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 1247–1266.
[9] J. Chun, T. Kailath, and H. Lev-Ari, *Fast parallel algorithms for QR and triangular factorization*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 899–913.
[10] I. Gohberg, T. Kailath, and V. Olshevsky, *Fast Gaussian elimination with partial pivoting for matrices with displacement structure*, Math. Comp., 64 (1995), pp. 1557–1576.
[11] G. H. Golub and C. V. Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[12] M. Gu, *Stable and efficient algorithms for structured systems of linear equations*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 279–306.
[13] M. Gu, *New fast algorithms for structured linear least squares problems*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 244–269.
[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] M. Gu and J. Xia, *A numerically stable and superfast algorithm for solving Toeplitz systems of linear equations*, SIAM Conference on Applied Linear Algebra, Monterey, 2009.
[16] N. Halko, P. G. Martinsson, and J. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[17] G. Heinig, *Inversion of generalized Cauchy matrices and other classes of structured matrices*, in Linear Algebra for Signal Processing, IMA Vol. Math. Appl. 69, Springer, New York, 1995, pp. 63–81.
[18] G. Heinig and K. Rost, *Algebraic Methods for Toeplitz-like Matrices and Operators*, Oper. Theory Adv. Appl. 13, Birkhäuser Verlag, Basel, 1984, pp. 109–127.
[19] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, PA, 2002.
[20] T. Kailath, S. Kung, and M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68 (1979), pp. 395–407.
[21] E. Liberty, F. Woolfe, P. G. Martinsson, V. Rokhlin, and M. Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proc. Natl. Acad. Sci. USA, 104 (2007), pp. 20167–20172.
[22] 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.
[23] W. Lyons, *Fast Algorithms with Applications to PDEs*, Ph.D. thesis, University of California Santa Barbara, Santa Barbara, CA, 2005.
[24] 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.
[25] P. G. Martinsson, V. Rokhlin, and M. Tygert, *A fast algorithm for the inversion of general Toeplitz matrices*, Comput. Math. Appl. 50 (2005), pp. 741–752.
[26] M. K. Ng, *Iterative Methods for Toeplitz Systems*, Numer. Math. Sci. Comput., Oxford University Press, New York, 2004.
[27] V. Y. Pan, *On computations with dense structured matrices*, Math. Comp., 55 (1990), pp. 179–190.
[28] V. Y. Pan, *Parallel least-squares solution of general and Toeplitz-like linear systems*, in Proceedings of the 2nd Annual ACM Symposium on Parallel Algorithms and Architecture, ACM, New York, 1990, pp. 244–253.
[29] V. Y. Pan, *Transformations of Matrix Structures Work Again*, preprint, arXiv:1303.0353, 2013.
[30] S. Qiao, *Hybrid algorithm for fast Toeplitz orthogonalization*, Numer. Math., 53 (1988), pp. 351–366.
[31] G. Rodriguez, *Fast solution of Toeplitz- and Cauchy-like least-squares problems*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 724–748.
[32] D. R. Sweet, *Numerical Methods for Toeplitz Matrices*, Ph.D. thesis, University of Adelaide, Adelaide, Australia, 1982.
[33] F. W. Trench, *Numerical solution of the eigenvalue problem for Hermitian Toeplitz matrices*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 135–146.
[34] M. Van Barel, G. Heinig, and P. Kravanja, *A superfast method for solving Toeplitz linear least squares problems*, Linear Algbera Appl., 366 (2003), pp. 441–457.
[35] J. Xia, *On the complexity of some hierarchical structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 388–410.
[36] J. Xia, *Randomized sparse direct solvers*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 197–227.
[37] J. Xia, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A832–A860.
[38] 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.
[39] 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.