KERNEL MATRIX APPROXIMATIONS BY SUMS OF EXPONENTIALS AND STABILITY OF FAST STRUCTURED TRANSFORMS*

CHENYANG CAO† AND JIANLIN XIA*

Abstract. Sum-of-exponentials (SoE) expansions provide an efficient strategy to perform some matrix transforms. In this paper, we show that they can also serve as a valuable way to compute structured approximations to some kernel matrices. We first illustrate that some existing fast transforms (Hilbert, Gauss, etc.) are essentially using generalized sequentially semiseparable (SSS) structures for which the stability has been in question before. We then give comprehensive analysis of the stability of transforms via general SSS structures and rigorously prove that such transforms may have numerical errors growing exponentially, while the use of SoE expansions leads to polynomial error growth. Moreover, we give a way to further reduce the error growth to poly-logarithmic via the use of hierarchical tree structures. Our analysis reveals the two key components that ensure stability of rank-structured transforms and other algorithms: algorithm architecture and norm bounds of the generators of the structure. It concretely confirms some long-standing speculations: sequential structured matrix (like SSS) algorithms are potentially unstable, even if relevant generators have bounded norms; hierarchical structured algorithms are stable as long as relevant generator norms are bounded. SoE expansion is then just an effective way to further control the norms of the generators.

Key words. kernel matrix, fast transform, sum-of-exponentials expansion, low-rank approximation, rank-structured matrix, backward stability

AMS subject classifications. 15A23, 15B05, 65D15, 65F55

1. Introduction.

Kernel matrices are frequently used in numerical computations and data analysis. Consider a kernel matrix of the form of

$$H = \left(\kappa(x_i, y_j)\right)_{x_i\in\mathbf{x}, y_j\in\mathbf{y}}, \tag{1.1}$$

where $\kappa(x, y)$ is a kernel function and x_i and y_j are points in data sets $\mathbf{x}$ and $\mathbf{y}$, respectively. The numbers of points in $\mathbf{x}$ and $\mathbf{y}$ may be different, but we suppose $|\mathbf{x}| = |\mathbf{y}| = n$ for convenience.

We are particularly interested in some kernel matrices arising from certain transforms (such as Hilbert, Gauss, and Hankel transforms). A straightforward way to perform these transforms (multiplications of H with vectors) costs $\mathcal{O}(n^2)$. There are fast transform algorithms that can reach nearly linear complexity. One such algorithm that has been very popular in the studies of integral equations is based on *sum-of-exponentials* (SoE) approximations for $\kappa(x, y)$. They are truncated expansions in terms of sums of exponential functions. For some kernels, SoE expansions with a small number of terms can reach high accuracy and the complexity to multiply H with vectors can be reduced to $\mathcal{O}(n)$ [13, 21, 32]. These algorithms are much simpler as compared with methods such as the fast multipole method (FMM) [14], which requires to consider local-multipole and multipole-multipole expansions for certain dense translation operators. SoE-based schemes instead use translation operators that have simple diagonal forms and also have nice norm bounds [2, 6, 12, 32]. SoE expansions are also useful for accelerating some other computations [3, 4, 5, 30].

This work has two main subjects: showing how SoE expansions may be used for fast structured approximations of some kernel matrices, and further providing

*Submitted for review.

Funding: The research of Jianlin Xia was supported in part by an NSF grant DMS-2111007.

†Department of Mathematics, Purdue University, West Lafayette, IN 47907 (cao302@purdue.edu, xiaj@purdue.edu).

comprehensive understanding of the stability of fast algorithms like transforms in some structured matrix forms.

1.1. Background on SoE expansions for fast transforms. To prepare for later discussions, we briefly review some background materials on how SoE expansions may be used to accelerate some transforms. Like in various previous studies [2, 6, 13, 16, 20, 32], our discussions focus on point sets $\mathbf{x}$ and $\mathbf{y}$ in (1.1) on the real line. (Extensions to higher dimensions will also be discussed later.)

For various kernel functions defined on one dimensional $\mathbf{x}$ and $\mathbf{y}$, the methods typically represent a kernel function $\kappa(x,y)$ in an appropriate integral form and then approximate the integral by a quadrature rule [2, 6, 13, 21, 26, 31]. As an example, one frequently used kernel function is the Cauchy kernel $\kappa(x,y)=\frac{1}{x-y}$ corresponding to the Hilbert transform

$$f_i=\sum_{\substack{j=1\\y_j\neq x_i}}^{n}\frac{z_j}{x_i-y_j},\quad i=1,2,\ldots,n, \tag{1.2}$$

where x_i and y_j are points in an interval $[a,b]$ and z_j's are scalars. When $s=x_i-y_j>0$, an SoE expansion may be obtained based on the Laplace transform followed by an appropriate quadrature approximation:

$$\frac{1}{s}=\int_0^\infty e^{-st}dt\approx\sum_{k=1}^{p}w_ke^{-st_k}, \tag{1.3}$$

where p is the number of quadrature nodes and t_k and w_k are the quadrature nodes and weights, respectively.

A key step in the fast Hilbert transform is to utilize the above expansion to evaluate

$$f_i^+=\sum_{\substack{j=1\\x_i>y_j}}^{\alpha}\frac{z_j}{x_i-y_j}, \tag{1.4}$$

where α is an appropriately chosen index (see Section 2.1.1). According to (1.3), we have

$$f_i^+\approx\sum_{j=1}^{\alpha}\sum_{k=1}^{p}w_kz_je^{-(x_i-y_j)t_k}=\sum_{k=1}^{p}w_k\beta_{k,\alpha}e^{-(x_i-y_\alpha)t_k}\quad\text{with} \tag{1.5}$$

$$\beta_{k,\alpha}=\sum_{j=1}^{\alpha}z_je^{-(y_\alpha-y_j)t_k}. \tag{1.6}$$

It can be shown that $\beta_{k,\alpha}$ for all $k=1,2,\ldots,p$, $\alpha=1,2,\ldots,n$ can be precomputed via fast updates with total cost $\mathcal{O}(pn)$ [13, 21, 32]. Following this precomputation, it costs $\mathcal{O}(p)$ to evaluate each f_i^+. Accordingly, the total cost for evaluating f_i^+ for all $i=1,2,\ldots,n$ is $\mathcal{O}(pn)$. Some details will be given in Section 2.1.1.

1.2. Motivations and contributions. The aforementioned evaluation process (1.5)–(2.4) is very efficient, but is not immediately intuitive to understand. The evaluation of all f_i and the update for all $\beta_{k,\alpha}$ are performed in iterative updates of some vectors and the numerical stability is unclear.

In fact, the whole transform may be assembled into a structured way to perform a fast matrix-vector multiplication. That is, the kernel matrix H in (1.1) can be approximated by a structured matrix. This leads to a matrix form of the fast transform and suggests that SoE expansions are also useful for obtaining structured approximations of kernel matrices. Indeed, later we can see that SoE expansions produce effective compression of some blocks of H and further have some attractive features.

Thus, this work aims to give an intuitive algebraic way to reveal and extract the underlying structure within fast transforms based on SoE expansions. Next, the structured matrix form makes it convenient to analyze the numerical error propagation and uncover potential stability issues in the transforms. Also, we obtain another structured matrix transform with superior stability and reduced error growth.

Specifically, the main contributions of this work include the following.

1. We provide an intuitive matrix version that facilitates the understanding of the mechanism of some fast transforms and helps to make the ideas more accessible. We show that they essentially perform matrix-vector multiplications in terms of some rank-structured approximations to relevant kernel matrices. The SoE framework is a strategy to organize data points into certain separated clusters that correspond accurate low-rank off-diagonal approximations. For the Hilbert transform above, the structured form is just a generalization of the so-called sequentially semiseparable (SSS) matrix [8, 11], represented by a sequence of smaller matrices called generators. With SoE expansions, the so-called translation generators further have diagonal forms.
2. The matrix version further makes it convenient to inspect the stability and error propagation of the transforms. It has long been suspected that SSS matrices may be susceptible to stability issues and the stability of SSS matrix-vector multiplications may be much worse than that of usual full matrix-vector multiplications [1]. However, the general stability analysis is lacking. Here, we provide a comprehensive rigorous study of the stability of (generalized) SSS matrix-vector multiplications and show that the backward error may potentially grow exponentially with respect to the matrix size (Theorem 4.9). This clearly reveals the stability risk.
3. To improve the stability, we convert the generalized SSS form into a generalized form of the hierarchically semiseparable (HSS) structure [9, 29] that has superior stability in its operations. The error propagation of generalized HSS matrix-vector multiplications is significantly lower than with generalized SSS forms (Theorem 4.13). What's more, when SoE expansions are used to obtain the generalized HSS forms, the error propagation can be further reduced.
4. The studies give comprehensive insights into the stability of structured algorithms like matrix-vector multiplications. That is, there are two key components that impact the stability: *algorithm architecture* and *norm bounds of generators*. The former controls the length of the error propagation path and the latter determines the error growth rate at each algorithm step. Hierarchical structured algorithms like HSS ones have lower error growth than sequential ones like SSS. Carefully bounded norms of generators (like from SoE expansions) can significantly lower the error growth factor. See Theorems 4.9 and 4.13 and Corollaries 4.10 and 4.14 and some numerical validations.

The paper is organized as follows. Section 2 shows how some fast transforms via SoE expansions may be formulated as matrix forms by constructing generalized SSS approximations to the kernel matrices. Section 3 further shows how SoE expansions may be used to produce generalized HSS approximations. The stability of transforms via these two types of structures is then analyzed in Section 4, with some proofs supplied in Appendix A. The stability results are verified by some numerical tests in Section 5. Finally, Section 6 concludes the paper.

The following is a collection of commonly used notation in the paper.

- Throughout the presentation, bold lower-case letters are used for vectors and sets of points.
- Without loss of generality, assume $\mathbf{x}$ and $\mathbf{y}$ in (1.1) can be partitioned as

$$\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_N, \quad \mathbf{y} = \mathbf{y}_1 \cup \cdots \cup \mathbf{y}_N, \tag{1.7}$$

  where each cluster $\mathbf{x}_k$ has m points so that $\mathbf{x}_k = (x_{(k-1)m+1}, \ldots, x_{km})^T$ and $n = Nm$, and $\mathbf{y}_k$ has a similar form.
- $\mathrm{diag}(\mathbf{v})$ denotes a diagonal matrix defined by a vector $\mathbf{v}$.
- For a vector $\mathbf{v}$, a function $f(\mathbf{v})$ represents a vector function defined entrywise.
- For a matrix A, a function $f(A)$ is a matrix function defined entrywise. For example, $\exp(A)$ represents a matrix with entries $\exp(A_{ij})$ (instead of the usual matrix exponential).
- For a vector $\mathbf{v}$ and a scalar c, $c+\mathbf{v}$ or $\mathbf{v}+c$ is the vector resulting from entrywise summation by c. For a matrix A, $c + A$ can be similarly understood.

2. Matrix version of fast transforms via SoE expansions of kernel functions. In this section, we present some types of fast transforms in terms of structured matrix-vector multiplications. Selected types of kernel functions are shown as examples. For simplicity, we focus on the real line with the points $x_i \in \mathbf{x}$ and $y_j \in \mathbf{y}$ inside an interval $[a, b]$ and suppose the points in each set are ordered from the smallest to the largest. Generalization to higher dimensions will be discussed in Section 2.3.

2.1. Matrix version of the fast Hilbert transform. The first transform we consider is the Hilbert transform defined in (1.2). It corresponds to the Cauchy kernel $\kappa(x, y) = \frac{1}{x-y}$.

2.1.1. Fast Hilbert transform via SoE expansions. We first provide some details on how to quickly perform the Hilbert transform via SoE expansions for the Cauchy kernel. The discussions in this subsection are based on [13, 21, 32].

To utilize the expansion in (1.3), pick $\delta \in (0, 1)$ so that the number of x_i, y_j points satisfying $|x_i - y_j| \le \delta(b - a)$ is small. Then rewrite (1.2) as

$$f_i = f_i^+ + f_i^- + f_i^0, \tag{2.1}$$

where f_i^+ has the form in (1.4) and consists of all x_i, y_j points satisfying $x_i - y_j > \delta(b-a)$, f_i^- consist of all x_i, y_j points satisfying $x_i - y_j < -\delta(b-a)$, and f_i^0 corresponds to the remaining points. It is then sufficient to just consider f_i^+ since f_i^- can be handled in the same way for its negative. The choice of α in (1.4) is to make

$$x_i - y_{\alpha+1} \le \delta(b - a) < x_i - y_\alpha. \tag{2.2}$$

To accurately approximate f_i^+, it can be shown that there exist quadrature nodes $t_1, \ldots, t_p$ and weights $w_1, \ldots, w_p$ with $p = \mathcal{O}(\log(\epsilon^{-1}) \log(\delta^{-1}))$ such that [2, 7, 13]

$$\left| \frac{1}{s} - \sum_{k=1}^{p} w_k e^{-st_k} \right| \le \epsilon \quad \text{for} \quad s \in [\delta(b - a), b - a]. \tag{2.3}$$

(Note that for $s = x_i - y_j$, we also have $s \le b - a$.) In practice, the quadrature nodes and weights are first found for a single interval $[1, \delta^{-1}]$ and then adapted to $[\delta(b - a), b - a]$ by scaling.

With this quadrature approximation, we obtain (1.5)–(1.6). For all $k = 1, 2, \ldots, p$, $\alpha = 2, \ldots, n$, $\beta_{k,\alpha}$ in (1.6) can be precomputed via updates:

$$\beta_{k,\alpha} = e^{-(y_\alpha - y_{\alpha-1})t_k} \sum_{j=1}^{\alpha-1} z_j e^{-(y_{\alpha-1}-y_j)t_k} + z_\alpha = \beta_{k,\alpha-1} e^{-(y_\alpha - y_{\alpha-1})t_k} + z_\alpha, \tag{2.4}$$

where $\beta_{k,1} = z_1$ for all k. The total cost of these updates is $\mathcal{O}(pn)$. Accordingly, f_i^+ for all $i = 1, 2, \ldots, n$ can be approximated using (1.5) in $\mathcal{O}(pn)$ complexity. Similarly, apply the idea to f_i^-. f_i^0 is directly evaluated, which costs $\mathcal{O}(mn)$. With small p and m, the Hilbert transform can be accurately performed in linear complexity.

2.1.2. Matrix form of the SoE expansion. The fast Hilbert transform in [13] as in the previous subsection may be made more intuitive through a matrix form of the SoE expansion. For the quadrature weights w_k and nodes t_k in (2.3), let

$$\mathbf{w} = \begin{pmatrix} w_1 & \cdots & w_p \end{pmatrix}^T, \quad \mathbf{t} = \begin{pmatrix} t_1 & \cdots & t_p \end{pmatrix}^T. \tag{2.5}$$

Picking α as in (2.2), we can rewrite (1.3) as a matrix form

$$\frac{1}{x_i - y_j} \approx \mathbf{p}_i^T B \mathbf{s}_j \quad \text{with} \tag{2.6}$$

$$\mathbf{p}_i = \exp(-(x_i - y_\alpha)\mathbf{t}), \quad B = \operatorname{diag}(\mathbf{w}), \quad \mathbf{s}_j = \exp(-(y_\alpha - y_j)\mathbf{t}). \tag{2.7}$$

Then f_i^+ can be approximated as

$$f_i^+ \approx \mathbf{p}_i^T B \boldsymbol{\beta}_\alpha \quad \text{with} \quad \boldsymbol{\beta}_\alpha = \begin{pmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_\alpha \end{pmatrix} \mathbf{z}_\alpha, \quad \mathbf{z}_\alpha = \begin{pmatrix} z_1 & \cdots & z_\alpha \end{pmatrix}^T.$$

We essentially have $\boldsymbol{\beta}_\alpha = (\beta_{1,\alpha}, \beta_{2,\alpha}, \ldots, \beta_{p,\alpha})^T$ with $\beta_{k,\alpha}$ given in (1.6). Furthermore, the updates of $\beta_{k,\alpha}$ in (2.4) may be written in the following matrix form:

$$\boldsymbol{\beta}_\alpha = z_\alpha + \begin{pmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \cdots & \mathbf{s}_{\alpha-1} \end{pmatrix} \mathbf{z}_{\alpha-1} = z_\alpha + \Lambda_{\alpha-1} \boldsymbol{\beta}_{\alpha-1} \quad \text{with} \tag{2.8}$$

$$\Lambda_{\alpha-1} = \operatorname{diag}(\exp(-(y_\alpha - y_{\alpha-1})\mathbf{t})), \quad \boldsymbol{\beta}_1 = \begin{pmatrix} z_1 & \cdots & z_1 \end{pmatrix}^T. \tag{2.9}$$

(Note the notation of scalar vector summation in (2.8).) This clearly illustrates why $\boldsymbol{\beta}_\alpha$ for all $\alpha = 1, 2, \ldots, n$ can be precomputed in $\mathcal{O}(pn)$ complexity.

The process of approximating $f_i^+, i = 1, \ldots, n$ can then be summarized as follows.

1. With $\boldsymbol{\beta}_1$ in (2.9), for $j = 2, 3, \ldots, n$, compute

$$\boldsymbol{\beta}_j = z_j + \Lambda_{j-1} \boldsymbol{\beta}_{j-1}.$$

2. For $i = 1, 2, \ldots, n$, pick α by using i in (2.2) and compute

$$f_i^+ \approx \mathbf{p}_i^T B \boldsymbol{\beta}_\alpha.$$

2.1.3. Structured matrix approximation to the Cauchy kernel matrix. The approximation (2.6) suggests that the SoE expansion essentially leads to low-rank approximations to some blocks of the kernel matrix H in (1.1) with the Cauchy kernel. It further means we can essentially get a structured matrix approximation to H. In this section, we derive this structured approximation, which can reveal the actual structured matrix operations beneath the fast Hilbert transform.

For the clusters in (1.7), the size m of each cluster is made small by choosing appropriate δ in (2.3) such that

$$x_i - y_j > \delta(b-a) \quad \text{for all} \quad |i-j| \geq m. \tag{2.10}$$

For convenience, we assume that the clusters $\mathbf{x}_k$ and $\mathbf{y}_k$ fully interlace in the following sense:

$$x_k^{\max} < y_{k+1}^{\min}, \quad y_k^{\max} < x_{k+1}^{\min}, \tag{2.11}$$

where $x_k^{\max}$ and $x_k^{\min}$ respectively represent the largest and the smallest points in $\mathbf{x}_k$, and $y_k^{\max}$ and $y_k^{\min}$ are similarly defined. (If the clusters are not interlaced as this, it can actually be shown that the rank structure below becomes simpler.)

Then consider a block $H_{k,l} = \left(\frac{1}{x_i - y_j}\right)_{x_i \in \mathbf{x}_k, y_j \in \mathbf{y}_l}$ of H defined by the clusters $\mathbf{x}_k$ and $\mathbf{y}_l$, where $k > l+1$. For each $x_i \in \mathbf{x}_k, y_j \in \mathbf{y}_l$, (2.6) holds. It is obvious that we can replace y_α in (2.7) by $y_{k-2}^{\max}$ so that the approximation (2.6) remains unchanged. Accordingly, we get a low-rank approximation

$$H_{k,l} \approx P_k B S_l, \tag{2.12}$$

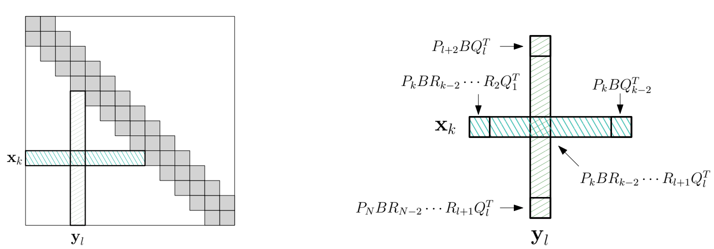
where P_k is formed by stacking the rows $\mathbf{p}_i^T$ corresponding to all $x_i \in \mathbf{x}_k$ and S_l consists of the columns $\mathbf{s}_j$ corresponding to all $y_j \in \mathbf{y}_l$:

$$P_k = \left(\exp(-(\mathbf{x}_k - y_{k-2}^{\max})\mathbf{t}^T)\right), \quad S_l = \left(\exp(-\mathbf{t}(y_{k-2}^{\max} - \mathbf{y}_l)^T)\right). \tag{2.13}$$

(Again, here the exponential functions are defined entrywise and are not matrix exponentials.) Note that (2.12) holds for all $1 \leq l \leq k-2$ and P_k only depends on k. Thus, P_k can serve as a column basis matrix for the low-rank approximation of the following off-diagonal block row:

$$\begin{pmatrix} H_{k,1} & \cdots & H_{k,l} & \cdots & H_{k,k-2} \end{pmatrix}. \tag{2.14}$$

See Figure 2.1(i) for an illustration.



(i) Off-diagonal blocks (ii) Structured off-diagonal approximations

FIG. 2.1. *How the off-diagonal blocks (pattern-filled blocks in (i)) of the kernel matrix H are approximated by low-rank forms from SoE expansions as in (ii).*

Similarly, we may obtain an approximate row basis matrix Q_l^T for $H_{k,l}$, where

$$Q_l = \left(\exp(-(y_l^{\max} - \mathbf{y}_l)\mathbf{t}^T)\right).$$

Q_l only depends on l and can serve as a row basis matrix for the low-rank approximation of the following off-diagonal block column:

$$\begin{pmatrix} H_{l+2,l}^T & \cdots & H_{k,l}^T & \cdots & H_{N,l}^T \end{pmatrix}^T. \tag{2.15}$$

Also see Figure 2.1(i). Therefore, $H_{k,l}$ has an approximation column basis matrix P_k and row basis matrix Q_l. To reflect such a structure for all $k = 3, \ldots, N, l = 1, \ldots, k-2$, we may rewrite (2.12) as

$$H_{k,l} \approx P_k B R_{k-2} R_{k-3} \cdots R_{l+1} Q_l^T, \tag{2.16}$$

where each R_s is defined to make $S_l = R_{k-2} R_{k-3} \cdots R_{l+1} Q_l^T$ and has the form

$$R_s = \mathrm{diag}(\exp(-(y_s^{\max} - y_{s-1}^{\max})\mathbf{t})). \tag{2.17}$$

At this point, the block row (2.14) and block column (2.15) both appear in nested structured forms as illustrated in Figure 2.1(ii). Such a structure is consistent with the SSS structure in [10, 11], except that the blocks with $l = k \pm 1$ are kept dense.

Similarly, when $k + 1 < l$, we may consider

$$H_{k,l} = \left(\tfrac{1}{x_i - y_j}\right)_{x_i \in \mathbf{x}_k, y_j \in \mathbf{y}_l} = -\left(\tfrac{1}{y_i - x_j}\right)^T_{y_i \in \mathbf{y}_l, x_j \in \mathbf{x}_k}.$$

Following the same strategy as above, we can get a structured approximation

$$\begin{aligned} &H_{k,l} \approx U_k W_{k+1} W_{k+2} \cdots W_{l-2} B V_l^T \quad \text{with} \\ &U_k = -\exp(-(x_k^{\max} - \mathbf{x}_k)\mathbf{t}^T), \quad V_l = \exp(-(\mathbf{y}_l - x_{l-2}^{\max})\mathbf{t}^T), \\ &W_s = \mathrm{diag}(\exp(-(x_s^{\max} - x_{s-1}^{\max})\mathbf{t})). \end{aligned} \tag{2.18}$$

To summarize, we have a structured matrix A that approximates H as follows:

$$H_{k,l} \approx A_{k,l} := \begin{cases} D_k, & \text{if } k = l, \\ E_k, & \text{if } k = l-1, \\ F_{k-1}, & \text{if } k = l+1, \\ U_k W_{k+1} \cdots W_{l-2} B V_l^T, & \text{if } k < l-1, \\ P_k B R_{k-2} \cdots R_{l+1} Q_l^T, & \text{if } k > l+1, \end{cases} \tag{2.19}$$

where D_k, E_k, and F_k are equal to the corresponding dense blocks in H and

$$\begin{cases} U_k = -\exp(-(x_k^{\max} - \mathbf{x}_k)\mathbf{t}^T), & V_l = \exp(-(\mathbf{y}_l - x_{l-2}^{\max})\mathbf{t}^T), \\ P_k = \exp(-(\mathbf{x}_k - y_{k-2}^{\max})\mathbf{t}^T), & Q_l = \exp(-(y_l^{\max} - \mathbf{y}_l)\mathbf{t}^T), \\ W_s = \mathrm{diag}(\exp(-(x_s^{\max} - x_{s-1}^{\max})\mathbf{t})), & R_s = \mathrm{diag}(\exp(-(y_s^{\max} - y_{s-1}^{\max})\mathbf{t})). \end{cases} \tag{2.20}$$

To better illustrate the block structure in (2.19), an example is shown as follows:

$$A = \begin{pmatrix} D_1 & E_1 & U_1 B V_3^T & U_1 W_2 B V_4^T & U_1 W_2 W_3 B V_5^T \\ F_1 & D_2 & E_2 & U_2 B V_4^T & U_2 W_3 B V_5^T \\ P_3 B Q_1^T & F_2 & D_3 & E_3 & U_3 B V_5^T \\ P_4 B R_2 Q_1^T & P_4 B Q_2^T & F_3 & D_4 & E_4 \\ P_5 B R_3 R_2 Q_1^T & P_5 B R_3 Q_2^T & P_5 B Q_3^T & F_4 & D_5 \end{pmatrix}. \tag{2.21}$$

The matrix A has a form as mentioned in [11] that generalizes the classical SSS form, and is said to be a *generalized SSS* matrix for convenience. That is, its blocks on and below the first block sub-diagonal form a block lower triangular part of an SSS matrix and its blocks on and above the first block super-diagonal form a block upper triangular part of an SSS matrix. Note that the matrices P, Q, U, V, R, W, etc. that define the generalized SSS form are still the generators of SSS forms so we just call them *SSS generators*. The generators P, Q, U, V are *basis generators* and R, W are *translation generators*. Note that SoE expansions further produce translation generators in diagonal forms and with norm bound 1.

The generalized SSS form in (2.19) can be quickly multiplied with a vector via the SSS matrix-vector multiplication algorithm in [10, 11]. Write $A(\approx H)$ as

$$A = A_{\mathbf{D}} + A_{\mathbf{L}} + A_{\mathbf{U}}, \tag{2.22}$$

where $A_{\mathbf{D}}$ is a block banded form corresponding to all the D, E, and F generators in (2.19), and the nonzero blocks of $A_{\mathbf{L}}$ and $A_{\mathbf{U}}$ are respectively block lower and upper triangular SSS forms. See Figure 2.2. Each part can be multiplied with a vector quickly. To facilitate our later stability analysis, we present the process to compute $\mathbf{f}^{+} = A_{\mathbf{L}}\mathbf{z}$ with a vector $\mathbf{z}$ in Algorithm 2.1. Suppose the block sizes is $m = \mathcal{O}(p)$ and the W, R generators are $p \times p$. Then the entire matrix-vector multiplication $A\mathbf{z}$ costs $\mathcal{O}(pn)$ flops.

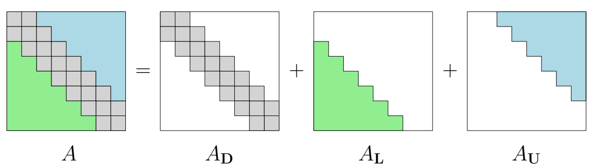


FIG. 2.2. *Splitting of a generalized SSS matrix.*

Algorithm 2.1 Fast lower-triangular SSS matrix-vector multiplication for $\mathbf{f}^{+}$ [11]

Input: SSS generators $\{P_k, Q_l, R_s, B\}$ of $A_{\mathbf{L}}$ and vector $\mathbf{z}$
Output: matrix-vector product $\mathbf{f}^{+} = A_{\mathbf{L}}\mathbf{z}$

```
Partition z = (z_1^T ··· z_N^T)^T conformably following the block partitioning of A_L
v_1 ← Q_1^T z_1
for k = 2, 3, ..., N − 2 do
    v_k ← Q_k^T z_k + R_k v_{k−1};                          ▷ Backward traversal
end for
for k = 3, 4, ..., N do
    f_k^+ ← P_k B v_{k−2};                                  ▷ Forward traversal
end for
f^+ ← (0, 0, (f_3^+)^T ..., (f_N^+)^T)^T        ▷ Attaching zeros for the zero block rows in A_L
```

Remark 2.1. SoE expansions of $\kappa(x, y)$ like in (1.3) depend on the sign of $x - y$. An SoE expansion valid for $x - y > 0$ may need to change the signs of exponentials for $x - y < 0$. Thus in the splitting (2.22), the subblocks of $A_{\mathbf{L}}$ and those of $A_{\mathbf{U}}$ typically do not share the same block row or column basis matrices.

2.2. Approximations to other kernel matrices.

The generalized SSS structure given in (2.19) also holds for kernel matrices H defined by various other $\kappa(x, y)$ that can be approximated by SoE expansions. Some generators may differ. We follow similar notation as in the previous subsection and also suppose all $x_i, y_j \in [a, b]$.

2.2.1. Gaussian kernels.

In the fast Gauss transform, it needs to quickly evaluate matrix-vector products involving the kernel matrix H in (1.1) with the Gaussian kernel $\kappa(x, y) = e^{-(x-y)^2/(4\mu)}$, where $\mu > 0$. SoE expansions for the Gaussian kernel based on the Carathéodory-Fejér method [25] have been well studied [21]. They can be used to obtain a generalized SSS approximation to H as follows.

- Rewrite $e^{-s^2/(4\mu)}$ through an inverse Laplace transform in the complex plane for $s \in \mathbb{R}$ and then discretize the complex integral to obtain an SoE form:

$$e^{-s^2/(4\mu)} = \frac{1}{2\pi \mathbf{i}} \int_{\Gamma} e^z \sqrt{\frac{\pi}{z}} e^{-\sqrt{z/\mu}|s|} dz \approx -\sum_{k=1}^{p} c_k \sqrt{\frac{\pi}{z_k}} e^{-\sqrt{z_k/\mu}|s|}, \tag{2.23}$$

  where Γ is a carefully chosen contour in the complex plane and c_k, z_k are generated from an algorithm developed in [26] that computes a nearly optimal approximation to e^z based on the Carathéodory-Fejér method.
- Substitute $s = x_i - y_j$ into (2.23) for $x_i > y_j$ to obtain the SoE approximation

$$e^{-(x_i-y_j)^2/(4\mu)} \approx \sum_{k=1}^{p} w_k e^{-(x_i-y_j)t_k},$$

  where $w_k = -c_k \sqrt{\pi/z_k}$ and $t_k = \sqrt{z_k/\mu}$.
- This approximation holds for all $x_i \in \mathbf{x}, y_j \in \mathbf{y}$ that are used to define H in (1.1). Use it to obtain a generalized SSS approximation similarly to that in Section 2.1.3. Since the Gaussian kernel function has no singularity at the origin, the dense blocks E_k, F_k are omitted from the generalized SSS form in (2.19). Accordingly, the matrix approximation is given by

$$H_{k,l} \approx A_{k,l} = \begin{cases} D_k, & \text{if } k = l, \\ U_k W_{k+1} \cdots W_{l-1} B V_l^T, & \text{if } k \le l-1, \\ P_k B R_{k-1} \cdots R_{l+1} Q_l^T, & \text{if } k \ge l+1, \end{cases} \tag{2.24}$$

  where D_k is equal to the corresponding dense block in H and

$$\begin{cases} U_k &= \exp(-(x_k^{\max} - \mathbf{x}_k)\mathbf{t}^T), & V_l &= \exp(-(\mathbf{y}_l - x_{l-1}^{\max})\mathbf{t}^T), \\ P_k &= \exp(-(\mathbf{x}_k - y_{k-1}^{\max})\mathbf{t}^T), & Q_l &= \exp(-(y_l^{\max} - \mathbf{y}_l)\mathbf{t}^T), \\ W_s &= \operatorname{diag}(\exp(-(x_s^{\max} - x_{s-1}^{\max})\mathbf{t})), & R_s &= \operatorname{diag}(\exp(-(y_s^{\max} - y_{s-1}^{\max})\mathbf{t})). \end{cases} \tag{2.25}$$

  Here, the same notation as in (2.5), (1.7), and (2.11) is used (throughout this entire paper). Fast evaluation of the Gauss transform is achieved through a procedure similar to Algorithm 2.1 with $\mathcal{O}(pn)$ complexity given the block size $m = \mathcal{O}(p)$.

2.2.2. Logarithmic kernels. For the logarithmic kernel $\log|x - y|$, one way to obtain a generalized SSS approximation to the corresponding kernel matrix H is based on the SoE expansion in [32]. In the following, we let $\lambda = \delta(b - a)$.

- Represent the function $\log(s)$ for $s > \lambda$ in an integral form:

$$\log(s) = \log(\lambda) + \int_{\lambda}^{s} \frac{1}{t} dt.$$

- Apply the SoE expansion in (1.3) to $1/t$ and integrate explicitly:

$$\log(s) \approx \log(\lambda) + \sum_{k=1}^{p} \frac{w_k}{t_k} e^{-\lambda t_k} + \sum_{k=1}^{p} -\frac{w_k}{t_k} e^{-s t_k}.$$

- Replace s by $x_i - y_j$ for $x_i - y_j > \lambda$:
$$\log(x_i - y_j) \approx c + \sum_{k=1}^{p} \hat{w}_k e^{-(x_i - y_j)t_k},$$
where $\hat{w}_k = -w_k/t_k$ and $c = \log(\lambda) + \sum_{k=1}^{p} \hat{w}_k e^{-\lambda t_k}$.
- Accordingly, the generalized SSS approximation to the logarithmic kernel matrix H is given by
$$H_{k,l} \approx A_{k,l} = \begin{cases} D_k, & \text{if } k = l, \\ E_k, & \text{if } k = l-1, \\ F_{k-1}, & \text{if } k = l+1, \\ c - U_k W_{k+1} \cdots W_{l-2} B V_l^T, & \text{if } k < l-1, \\ c + P_k B R_{k-2} \cdots R_{l+1} Q_l^T, & \text{if } k > l+1, \end{cases} \tag{2.26}$$
where D_k, E_k, and F_k are equal to the corresponding dense blocks in H, $B = -\mathrm{diag}(w_1/t_1, \ldots, w_p/t_p)$, and P_k, R_s, Q_l, U_k, W_s, V_l are the same as those in (2.20).

2.2.3. Square-root kernels. Next, consider the square-root kernel $\kappa(x, y) = 1/\sqrt{|x^2 - y^2|}$. Without loss of generality, assume the data sets $\mathbf{x}$ and $\mathbf{y}$ are in $[a, b] \subset \mathbb{R}_+$. A generalized SSS approximation to the corresponding kernel matrix may be obtained following an SoE approximation procedure in [32].

- For $x_i > y_j$, write the kernel as the Laplace transform of the modified Bessel function $I_0(\cdot)$ of the first kind of order zero:
$$\frac{1}{\sqrt{x_i^2 - y_j^2}} = \int_0^\infty I_0(y_j t) e^{-x_i t} dt = \int_0^\infty \frac{I_0(y_j t)}{e^{y_j t}} e^{-(x_i - y_j)t} dt.$$
The last equality is to use the scaled modified Bessel function $I_0(x)/e^x$ to avoid computational instability since it is a bounded function on $\mathbb{R}_+$.
- Apply the algorithm provided in [32] to get a generalized Gaussian quadrature approximation and thus the SoE expansion for $x_i - y_j > \delta(b - a)$:
$$\frac{1}{\sqrt{x_i^2 - y_j^2}} = \int_0^\infty \frac{I_0(y_j t)}{e^{y_j t}} e^{-(x_i - y_j)t} dt \approx \sum_{k=1}^{p} w_k \frac{I_0(y_j t_k)}{e^{y_j t_k}} e^{-(x_i - y_j)t_k},$$
where the quadrature weights w_k and nodes t_k are close to those in (1.3).
- Based on this expansion, we get a generalized SSS approximation to the square-root kernel matrix almost in the same form as in (2.19)–(2.20) other than slight modifications to some generators:
$$\begin{cases} U_k = \exp(-(x_k^{\max} - \mathbf{x}_k)\mathbf{t}^T) \odot \frac{I_0(\mathbf{x}_k \mathbf{t}^T)}{\exp(\mathbf{x}_k \mathbf{t}^T)}, \\ W_s = \mathrm{diag}(\exp(-(x_s^{\max} - x_{s-1}^{\max})\mathbf{t})), \\ V_l = \exp(-(\mathbf{y}_l - x_{l-2}^{\max})\mathbf{t}^T), \\ P_k = \exp(-(\mathbf{x}_k - y_{k-2}^{\max})\mathbf{t}^T), \\ R_s = \mathrm{diag}(\exp(-(y_s^{\max} - y_{s-1}^{\max})\mathbf{t})), \\ Q_l = \exp(-(y_l^{\max} - \mathbf{y}_l)\mathbf{t}^T) \odot \frac{I_0(\mathbf{y}_l \mathbf{t}^T)}{\exp(\mathbf{y}_l \mathbf{t}^T)}, \end{cases} \tag{2.27}$$
where $\odot$ denotes the Hadamard product.

2.3. SoE approximations for other kernels and higher dimensions. SoE expansions for more kernel functions have been studied in various literatures and generalized SSS approximations may be obtained for the corresponding kernel matrices. For instance, strategies similar to those in Sections 2.1 can also be applied to other functions that can be rewritten as Laplace transforms [7, 32]. Another such example is $\frac{1}{\sqrt{x^2+y^2}}$, which is useful for designing fast Hankel transforms. An SoE expansion can be obtained similarly to those for $\frac{1}{x-y}$ and $\frac{1}{\sqrt{x^2-y^2}}$ based on generalized Gaussian quadratures [31]. This idea can also be extended to find SoE expansions of the Cauchy kernel in certain specific regions on the complex plane [31].

In [2, 6], some algorithms are designed to obtain SoE expansions for some one-dimensional translation-invariant kernels $\tilde{\kappa}(s) := \kappa(x, y)$ with $s = x - y$. The algorithms are based on solutions of some structured linear system and eigenvalue problems. Examples of $\tilde{\kappa}(s)$ mentioned in [2] include the following:

- $1/s^\alpha$ with α a positive parameter;
- $J_0(\alpha s)$ with α a positive parameter and $J_0(\cdot)$ the Bessel function of the first kind of order zero;
- the Dirichlet kernel $\frac{\sin(N\pi s)}{N\sin(\pi s)}$ with $N \in \mathbb{N}$;
- kernels like $\log(\sin^2(\pi s))$ and $\cot(\pi s)$ in harmonic analysis.

In [22], a strategy based on Cauchy integration is used to construct SoE expansions for general analytical kernel functions $\tilde{\kappa}(s)$ such as s^n with odd n, $s^n \log s$ with even n, $\exp(-\alpha s^2)$, the Helmholtz kernel $e^{2\pi \mathbf{i} s}/s$, $\sqrt{s^2+\alpha^2}$, and $1/\sqrt{s^2+\alpha^2}$, where α is a certain a parameter. The main idea of this method is presented as follows.

- Apply the Cauchy integral formula to $\tilde{\kappa}(s)$:

$$\tilde{\kappa}(s) = \frac{1}{2\pi\mathbf{i}} \int_\Gamma \frac{\tilde{\kappa}(z)}{z-s} dz, \quad s \in \mathbb{R}, \tag{2.28}$$

  where Γ is a Jordan curve in the complex plane and encloses the point $(s, 0)$.
- Partition Γ into pieces Γ_j such that $\Re(e^{-\mathbf{i}\theta_j}(z-s)) > 0$ after θ_j-rotation of $z - s$ for any $z \in \Gamma_j$. Then an SoE expansion is given by

$$\tilde{\kappa}(s) = \frac{1}{2\pi\mathbf{i}} \sum_j e^{-\mathbf{i}\theta_j} \int_0^\infty \left(\int_{\Gamma_j} \tilde{\kappa}(z) e^{-tze^{-\mathbf{i}\theta_j}} dz \right) e^{tse^{-\mathbf{i}\theta_j}} dt \approx \sum_{k=1}^p w_k e^{t_k s}, \tag{2.29}$$

  where w_k and t_k are complex weights and quadrature nodes, respectively, and $p = \mathcal{O}\left(\sum_j \log(\max_{z\in\Gamma_j} |\tilde{\kappa}(z)|/\epsilon)\right)$ for a given tolerance ϵ.

Next, we comment on SoE expansions in higher dimensions. There are different ways to get multi-dimensional SoE expansions. As one example, for kernel functions like Gaussian in two dimensions, a splitting along the two directions may be made:

$$e^{-\|\mathbf{c}-\mathbf{s}\|_2^2/(4\mu)} = e^{-(c_1-s_1)^2/(4\mu)} e^{-(c_2-s_2)^2/(4\mu)}, \tag{2.30}$$

where $\mathbf{c} = (c_1, c_2)$ and $\mathbf{s} = (s_1, s_2)$. When $\mathbf{c}$ and $\mathbf{s}$ are respectively located within two clusters of data points, SoE expansions like in (2.23) hold for each dimension if the two clusters are separated in that dimension. That is, if an interval for, say, all the c_1 values does not overlap with an interval for all the s_1 values. If this does not hold for a certain dimension, other expansions (like Hermite expansions) may be used [21], which leads to a mixture of expansions for the overall kernel.

As another example, consider the kernel function $\frac{1}{\|\mathbf{c}-\mathbf{s}\|_2}$ with $\mathbf{c} = (c_1, c_2)$ and $\mathbf{s} = (s_1, s_2)$ in two dimensions. As SoE expansion may be obtained based on a Laplace transform of the Bessel functions for $c_2 > s_2$ [15]:

$$\frac{1}{\|\mathbf{c}-\mathbf{s}\|_2} = \int_0^\infty e^{-t(c_2-s_2)} J_0(t(c_1-s_1))dt = \frac{1}{\pi}\int_0^\infty e^{-t(c_2-s_2)} \int_0^\pi e^{\mathbf{i}t(c_1-s_1)\cos\theta d\theta} dt$$

$$\approx \sum_{k=1}^{p} \frac{w_k}{q_k} \sum_{l=1}^{q_k} e^{-t_k[(c_2-s_2)-\mathbf{i}(c_1-s_1)\cos(\pi l/q_k)]},$$

where w_k and t_k are respectively quadrature weights and nodes and q_k is a positive integer depending on k. The fast algorithm in [15] provides a way to generate w_k and t_k from multipole expansions and is essentially performing a rank-structured matrix-vector multiplication like in this paper.

There are also other useful techniques for generating SoE expansions for multi-dimensional kernels. The Cauchy integration method in [22] for finding SoE expansions (like in (2.28)–(2.29) above) can be further extended to analytical kernels in higher dimensions. In [19], for the Cauchy kernel in complex regions, a system of quadrature weights and nodes is obtained via a conversion from multipole expansions.

For multi-dimensional SoE expansions, similarly to the work here, it can be shown that fast algorithms like transforms are essentially performed in terms of certain rank-structured matrices.

3. Stable transforms via generalized HSS approximations from SoE expansions.

Fast transforms based on generalized SSS forms essentially compute matrix-vector products in a sequential way as in Algorithm 2.1. Later in subsection 4.2, we shall see the potential stability limitation. In this section, we give a strategy that can significantly enhance the stability by converting the generalized SSS form resulting from SoE expansions into a *generalized HSS form*.

3.1. Generalized HSS approximations from SoE expansions.

We say a matrix A a generalized HSS matrix if it can be split as in (2.22) (also see Figure 2.2) and the nonzero parts of $A_{\mathbf{L}}$ and $A_{\mathbf{U}}$ are triangular lower and upper parts of standard HSS matrices, respectively.

A brief review of the standard HSS structure in [9, 29] is as follows. An HSS matrix K has a block off-diagonal low-rank form and its blocks follow a partitioning strategy as given by a postordered binary tree $\mathcal{T}$ called HSS tree. To be specific, suppose $\mathcal{T}$ has nodes labeled as $i = 1, 2, \ldots, \sigma$ with σ the root of the HSS tree. Then for each non-leaf node i, the diagonal block $\mathcal{D}_i$ has the form

$$\mathcal{D}_i = \begin{pmatrix} \mathcal{D}_{c_1} & \mathcal{U}_{c_1}\mathcal{B}_{c_1}\mathcal{V}_{c_2}^T \\ \mathcal{U}_{c_2}\mathcal{B}_{c_2}\mathcal{V}_{c_1}^T & \mathcal{D}_{c_2} \end{pmatrix}, \tag{3.1}$$

where c_1, c_2 are the left and right children of node i and the calligraphic letters $\mathcal{D}$, $\mathcal{U}$, $\mathcal{V}$, $\mathcal{B}$ represent the so-called *HSS generators* so that $\mathcal{D}_\sigma \equiv K$. $\mathcal{U}$, $\mathcal{V}$ are basis generators for off-diagonal blocks and further satisfy the following nested relations:

$$\mathcal{U}_i = \begin{pmatrix} \mathcal{U}_{c_1}\mathcal{R}_{c_1} \\ \mathcal{U}_{c_2}\mathcal{R}_{c_2} \end{pmatrix}, \quad \mathcal{V}_i = \begin{pmatrix} \mathcal{V}_{c_1}\mathcal{W}_{c_1} \\ \mathcal{V}_{c_2}\mathcal{W}_{c_2} \end{pmatrix}, \tag{3.2}$$

where $\mathcal{R}$, $\mathcal{W}$ are known as translation generators.

In the following, we convert the generalized SSS approximations in the previous section into generalized HSS approximations. More specifically, for an generalized

SSS form like in (2.22) and Figure 2.2, we may write the nonzero blocks of, say, $A_{\mathbf{L}}$ in an HSS form. That is, in an HSS construction process, all the generators in (3.1) and (3.2) can be explicitly written based on SoE expansions.

Remark 3.1. (*Notation*) The HSS construction is essentially for the submatrix of $A_{\mathbf{L}}$ corresponding to block rows 2 to N and block columns 1 to $N-1$, denoted by $\hat{A}_{\mathbf{L}}$. Without loss of generality, assume N in (1.7) satisfies $N = 2^L + 1$ so that $\hat{A}_{\mathbf{L}}$ has 2^L block rows and 2^L block columns and can be converted into an HSS matrix corresponding to an L-level full binary HSS tree $\mathcal{T}$. The HSS form has an L-level hierarchical block structure. The leaf-level blocks correspond to the same partitioning as used in the SSS form of $\hat{A}_{\mathbf{L}}$. For convenience, we relabel the point sets associated with the block rows and columns. That is, the set $\mathbf{x}_j$ corresponding to the jth block row of $\hat{A}_{\mathbf{L}}$ is relabeled as $\hat{\mathbf{x}}_i$, where i is the node of $\mathcal{T}$ that is the jth leaf ordered from the left. Then for a non-leaf node i with children c_1 and c_2, define $\hat{\mathbf{x}}_i = \hat{\mathbf{x}}_{c_1} \cup \hat{\mathbf{x}}_{c_2}$. Similarly, define sets $\hat{\mathbf{y}}_j$. See Figure 3.1 below. We also introduce notation $\hat{x}_i^{\max}$ and $\hat{x}_i^{\min}$ like in (2.11). We further define the point that immediately precedes $\hat{x}_i^{\min}$, if any, to be the predecessor of $\hat{\mathbf{x}}$, denoted $\hat{x}_i^{\mathrm{pred}}$. If this point does not exist (when i is the leftmost node at its level of the tree), then $\hat{x}_i^{\mathrm{pred}}$ is set to be empty.

Without loss of generality, we just show the HSS generators for translation-invariant kernels $\kappa(x, y)$ with SoE expansion

$$\kappa(x,y) \approx \sum_{k=1}^{p} w_k e^{-(x-y)t_k} \quad \text{with} \quad x - y \in [\delta(b-a), b-a]. \tag{3.3}$$

For other kernels with different SoE expansions, minor modifications may be made to the HSS generators.

Following (3.3), for any clusters $\mathbf{x}_k, \mathbf{y}_l$ satisfying $x_k^{\min} - y_l^{\max} > \delta(b-a)$, we can obtain a low-rank approximation to the corresponding block in the kernel matrix as

$$\begin{aligned}(\kappa(x,y))_{x\in\mathbf{x}_k, y\in\mathbf{y}_l} &\approx \sum_{k=1}^{p} w_k e^{-t_k(\mathbf{x}_k-\mathbf{y}_l)} = \sum_{k=1}^{p} e^{-t_k(\mathbf{x}_k-y_l^{\max})} w_k e^{-t_k(y_l^{\max}-\mathbf{y}_l)} \\ &= \exp(-(\mathbf{x}_k - y_l^{\max})\mathbf{t}^T) B \exp(-\mathbf{t}(y_l^{\max} - \mathbf{y}_l)^T),\end{aligned} \tag{3.4}$$

where $B = \mathrm{diag}(\mathbf{w})$ as before. Now, for $\mathbf{x}_k, \mathbf{y}_l$ corresponding to any nonzero off-diagonal block of $\hat{A}_{\mathbf{L}}$, with the interlacing of the point sets as in (2.11), (3.4) naturally holds. Based on this, we can find the low-rank form of the corresponding block of $\hat{A}_{\mathbf{L}}$.

The following lemma shows how to obtain the HSS generators.

LEMMA 3.2. *Suppose the kernel function $\kappa(x,y)$ satisfies (3.3). Then $\hat{A}_{\mathbf{L}}$ can be written as an HSS form with generators as follows.*

- *For a leaf node i of $\mathcal{T}$,*

$$\mathcal{D}_i = 0, \quad \mathcal{U}_i = \exp(-(\hat{\mathbf{x}}_i - \hat{y}_i^{\mathrm{pred}})\mathbf{t}^T), \quad \mathcal{V}_i = \exp(-(\hat{y}_i^{\max} - \hat{\mathbf{y}}_i)\mathbf{t}^T).$$

- *For a non-leaf node i with left and right children c_1 and c_2, respectively,*

$$\mathcal{B}_{c_1} = 0, \quad \mathcal{B}_{c_2} = B(= \mathrm{diag}(\mathbf{w})).$$

If further $i \neq \sigma$, then

$$\begin{aligned}&\mathcal{R}_{c_1} = I, \quad \mathcal{R}_{c_2} = \mathrm{diag}(\exp(-(\hat{y}_{c_1}^{\max} - \hat{y}_{c_1}^{\mathrm{pred}})\mathbf{t})), \\ &\mathcal{W}_{c_1} = \mathrm{diag}(\exp(-(\hat{y}_{c_2}^{\max} - \hat{y}_{c_1}^{\max})\mathbf{t})), \quad \mathcal{W}_{c_2} = I.\end{aligned}$$

Proof. Let i and j be leaf nodes and respectively be the left and right children of their parent r. Clearly, $\mathcal{D}_i = \mathcal{D}_j = 0$ by the definition of $\hat{A}_{\mathbf{L}}$. According to (3.4), the lower off-diagonal block corresponding to $\hat{\mathbf{x}}_j$ and $\hat{\mathbf{y}}_i$ has a low-rank approximation

$$\mathcal{U}_j \mathcal{B}_j \mathcal{V}_i^T = \exp(-(\hat{\mathbf{x}}_j - \hat{y}_i^{\max})\mathbf{t}^T) B \exp(-\mathbf{t}(\hat{y}_i^{\max} - \hat{\mathbf{y}}_i)^T). \tag{3.5}$$

Since i is the left sibling of j, $\hat{y}_j^{\text{pred}} = \hat{y}_i^{\max}$. We can then let

$$\mathcal{B}_j = B, \quad \mathcal{U}_j = \exp(-(\hat{\mathbf{x}}_j - \hat{y}_j^{\text{pred}})\mathbf{t}^T), \quad \mathcal{V}_i = \exp(-(\hat{y}_i^{\max} - \hat{\mathbf{y}}_i)\mathbf{t}^T). \tag{3.6}$$

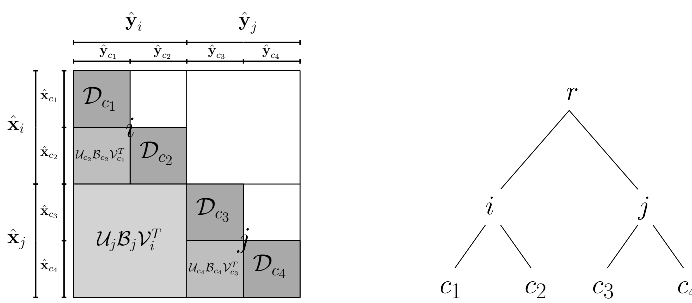
The upper off-diagonal block $\mathcal{U}_i \mathcal{B}_i \mathcal{V}_j^T = 0$ so we may set

$$\mathcal{B}_i = 0, \quad \mathcal{U}_i = \exp(-(\hat{\mathbf{x}}_i - \hat{y}_i^{\text{pred}})\mathbf{t}^T), \quad \mathcal{V}_j = \exp(-(\hat{y}_j^{\max} - \hat{\mathbf{y}}_j)\mathbf{t}^T), \tag{3.7}$$

where $\mathcal{U}_i$ and $\mathcal{V}_j$ have forms consistent with those in (3.6).

If i and j are non-leaf sibling nodes, (3.5) still holds so the same forms of $\mathcal{U}, \mathcal{B}, \mathcal{V}$ generators as above can be used.

We then derive the translation generators $\mathcal{R}, \mathcal{W}$. For convenience, suppose i has children c_1 and c_2, and j has children c_3 and c_4, as shown in Figure 3.1. The $\mathcal{U}, \mathcal{B}, \mathcal{V}$ generators associated with $c_1, \ldots, c_4$ can be similarly written out.



(i) Partition of $\mathcal{D}_r$ (ii) The HSS tree for matrix structure in (i)

Fig. 3.1. *Partitioning of $\mathcal{D}_r$ corresponding to the tree nodes.*

Noticing $\hat{y}_j^{\text{pred}} = \hat{y}_{c_3}^{\text{pred}}$ and $\hat{y}_i^{\max} = \hat{y}_{c_2}^{\max}$, we have

$$\begin{aligned} \mathcal{U}_j \mathcal{B}_j \mathcal{V}_i^T &= \exp\left(-\begin{pmatrix} \hat{\mathbf{x}}_{c_3} - \hat{y}_{c_3}^{\text{pred}} \\ \hat{\mathbf{x}}_{c_4} - \hat{y}_{c_3}^{\text{pred}} \end{pmatrix} \mathbf{t}^T\right) B \exp\left(-\mathbf{t}\begin{pmatrix} \hat{y}_{c_2}^{\max} - \hat{\mathbf{y}}_{c_1}^T & \hat{y}_{c_2}^{\max} - \hat{\mathbf{y}}_{c_2}^T \end{pmatrix}\right) \\ &= \begin{pmatrix} \mathcal{U}_{c_3} \\ \mathcal{U}_{c_4} \operatorname{diag}(\exp(-(\hat{y}_{c_3}^{\max} - \hat{y}_{c_3}^{\text{pred}})\mathbf{t})) \end{pmatrix} B \begin{pmatrix} \mathcal{V}_{c_1} \operatorname{diag}(\exp(-(\hat{y}_{c_2}^{\max} - \hat{y}_{c_1}^{\max})\mathbf{t})) \\ \mathcal{V}_{c_2} \end{pmatrix}^T \\ &= \begin{pmatrix} \mathcal{U}_{c_3}\mathcal{R}_{c_3} \\ \mathcal{U}_{c_4}\mathcal{R}_{c_4} \end{pmatrix} B \begin{pmatrix} \mathcal{V}_{c_1}\mathcal{W}_{c_1} \\ \mathcal{V}_{c_2}\mathcal{W}_{c_2} \end{pmatrix}^T. \end{aligned} \tag{3.8}$$

Accordingly, we can set

$$\begin{aligned} \mathcal{R}_{c_3} &= I, \quad \mathcal{R}_{c_4} = \operatorname{diag}(\exp(-(\hat{y}_{c_3}^{\max} - \hat{y}_{c_3}^{\text{pred}})\mathbf{t})), \\ \mathcal{W}_{c_1} &= \operatorname{diag}(\exp(-(\hat{y}_{c_2}^{\max} - \hat{y}_{c_1}^{\max})\mathbf{t})), \quad \mathcal{W}_{c_2} = I. \end{aligned}$$

Now, when $\mathcal{U}_i \mathcal{B}_i \mathcal{V}_j^T$ is considered, we can similarly obtain

$$\begin{aligned} \mathcal{R}_{c_1} &= I, \quad \mathcal{R}_{c_2} = \operatorname{diag}(\exp(-(\hat{y}_{c_1}^{\max} - \hat{y}_{c_1}^{\text{pred}})\mathbf{t})), \\ \mathcal{W}_{c_3} &= \operatorname{diag}(\exp(-(\hat{y}_{c_4}^{\max} - \hat{y}_{c_3}^{\max})\mathbf{t})), \quad \mathcal{W}_{c_4} = I. \end{aligned}$$

To summarize, we get the generators as given in the lemma. $\square$

From this lemma, we can see that the HSS form for $\hat{A}_{\mathbf{L}}$ further has highly structured generators. That is, other than the leaf-level $\mathcal{U}, \mathcal{V}$ generators, all the other generators are diagonal matrices (with some even equal to 0 or I).

By comparing the HSS generators in Lemma 3.2 with the generalized SSS generators in (2.19), we can observe their connections. The HSS generators $\mathcal{U}$, $\mathcal{V}$, and $\mathcal{D}$ corresponding to the leaf nodes are just the P, Q generators of the SSS form. The translation generators $\mathcal{R}$, $\mathcal{W}$ are basically the products of some R generators. This motivates a way to convert a general SSS form (not necessarily from SoE approximations) to an HSS form.

Remark 3.3. As mentioned in Remark 3.1, when i is the leftmost node at its level of the tree, then $\hat{x}_i^{\text{pred}}$ is set to be empty. This does not impact the HSS generators above needed for multiplying $\hat{A}_{\mathbf{L}}$ with a vector. The reason is that $\hat{A}_{\mathbf{L}}$ is block lower triangular and any nonzero block $\mathcal{U}_i \mathcal{B}_i \mathcal{V}_j^T$ in its block lower triangular part satisfies $i > j$. Accordingly, this i is never the leftmost node at its level.

3.2. Fast transforms via HSS matrix-vector multiplications. Following the splitting (2.22), it suffices to look at the multiplication of $A_{\mathbf{L}}$ with a vector $\mathbf{z}$. With the notation in Remark 3.1, this is just to multiply the block lower triangular HSS matrix $\hat{A}_{\mathbf{L}}$ with a part of $\mathbf{z}$. We may adapt the HSS matrix-vector multiplication algorithm in [9, 27] and further take advantage of the diagonal forms of many generators. To facilitate the stability analysis later, we briefly review a telescoping form of an HSS matrix and list the main steps of the HSS matrix-vector multiplication.

The telescoping form of $\hat{A}_{\mathbf{L}}$ with generators $\mathcal{D}, \mathcal{U}, \mathcal{V}, \mathcal{R}, \mathcal{W}, \mathcal{B}$ corresponding to an L-level full binary HSS tree looks like [23, 27]

$$\hat{A}_{\mathbf{L}} = \sum_{k=1}^{L} \left(\prod_{j=L}^{k} U^{(j)} \right) B^{(k)} \left(\prod_{j=k}^{L} (V^{(j)})^T \right) \quad \text{with} \tag{3.9}$$

$$B^{(l)} = \operatorname{diag}(\{\mathbb{B}_i : i \text{ at level } l-1\}),$$

$$U^{(l)} = \operatorname{diag}(\{\mathbb{U}_i : i \text{ at level } l\}) \quad \text{and} \quad V^{(l)} = \operatorname{diag}(\{\mathbb{V}_i : i \text{ at level } l\}),$$

where

$$\mathbb{B}_i = \begin{pmatrix} 0 & \mathcal{B}_{c_1} \\ \mathcal{B}_{c_2} & 0 \end{pmatrix}, \quad i\text{: non-leaf node with children } c_1, c_2,$$

$$\mathbb{U}_i = \begin{cases} \mathcal{U}_i, & i\text{: leaf,} \\ \begin{pmatrix} \mathcal{R}_{c_1} \\ \mathcal{R}_{c_2} \end{pmatrix}, & i\text{: non-leaf node,} \end{cases} \qquad \mathbb{V}_i = \begin{cases} \mathcal{V}_i, & i\text{: leaf,} \\ \begin{pmatrix} \mathcal{W}_{c_1} \\ \mathcal{W}_{c_2} \end{pmatrix}, & i\text{: non-leaf node.} \end{cases}$$

To evaluate $\mathbf{f}^+ = A_{\mathbf{L}} \mathbf{z}$, we apply the fast HSS matrix-vector multiplication algorithm in [9, 27] to find $\hat{\mathbf{f}}^+ = \hat{A}_{\mathbf{L}} \hat{\mathbf{z}}$, where $\hat{\mathbf{z}}$ is the portion in $\mathbf{z}$ corresponding to $\hat{A}_{\mathbf{L}}$ and $\mathbf{f}^+ = \begin{pmatrix} 0 \\ \hat{\mathbf{f}}^+ \end{pmatrix}$. The main steps are as follows.

1. (*Bottom-up traversal*) Let $\mathbf{z}^{(L+1)} = \hat{\mathbf{z}}$. For l from L to 1, compute

$$\mathbf{z}^{(l)} = {V^{(l)}}^T \mathbf{z}^{(l+1)}, \quad \mathbf{y}^{(l)} = B^{(l)} \mathbf{z}^{(l)}. \tag{3.10}$$

2. (*Top-down traversal*) Let $\mathbf{f}^{(0)} = \mathbf{y}^{(1)}$. For l from 1 to L, compute

$$\mathbf{f}^{(l)} = U^{(l)} \mathbf{f}^{(l-1)} + \mathbf{y}^{(l+1)}. \tag{3.11}$$

Then output $\hat{\mathbf{f}}^+ = \mathbf{f}^{(L)}$.

4. Stability analysis for generalized SSS and HSS matrix-vector multiplications.

In this section, we discuss the stability of matrix-vector multiplications with generalized SSS and HSS forms. The results are presented in a general framework so that they hold for all (generalized) SSS and HSS matrices and the fast transforms via SoE approximations in this paper may be treated as special cases.

4.1. Motivations and preliminaries.

Our motivations for the stability analysis are as follows.

- For an SSS matrix A with non-orthogonal basis generators, rigorous stability analysis for the matrix-vector multiplication has not been done before. The work in [1] illustrates the potential instability via an example, although the multiplication is structured backward stable in terms of the generators. As a remedy, reorthogonalization of basis generators is used to improve stability in [1]. Here, we would like to show the stability in terms of A without orthogonality of the basis generators. The errors may grow exponentially with respect to the matrix size n, which rigorously confirms the stability risk.
- When A is written in an HSS form, stability analysis is done in [27] for the case again when the basis generators have orthonormal columns. Here, we also relax this requirement and show the stability of HSS transforms. Another related study is the stability analysis in [24] for the more sophisticated 2D FMM. However, the stability study in [24] has a very strict assumption on the norm bounds of off-diagonal basis generators at all hierarchical levels. In the following, we use separate norm bounds for leaf-level basis generators and translation generators, which enables to reveal the importance of the norm bounds of translation generators.

In the stability analysis below, we study perturbation terms like ΔA arising from floating point operations involving A. The analysis will frequently utilize the following preliminary lemmas.

LEMMA 4.1. *[17, p. 69] Let* $A \in \mathbb{R}^{n\times p}$, $\mathbf{z} \in \mathbb{R}^p$, *and*

$$\tau_p = p\epsilon_{\mathrm{mach}}/(1-p\epsilon_{\mathrm{mach}}), \tag{4.1}$$

where ϵ_{mach} *is the machine epsilon. Then the floating point result of the numerical matrix-vector multiplication* $A\mathbf{z}$*, denoted* $\mathrm{fl}(A\mathbf{z})$*, satisfies*

$$\mathrm{fl}(A\mathbf{z}) = (A+\Delta A)\mathbf{z} \quad \textit{with} \quad |\Delta A| \le \tau_p |A|.$$

LEMMA 4.2. *[17, p. 67] For* $i, j \in \mathbb{N}_+$, τ_i *and* τ_j *defined as in (4.1) satisfy*

$$i\tau_j \le \tau_{ij}, \quad \tau_i + \tau_j + \tau_i\tau_j \le \tau_{i+j},$$
$$\tau_i\tau_j \le \tau_{\min(i,j)} \quad \textit{for} \quad \max(i,j)\epsilon_{\mathrm{mach}} \le 1/2.$$

LEMMA 4.3. *[24] Let* $P \in \mathbb{C}^{m\times r}$ *and* $Q \in \mathbb{C}^{r\times n}$ *. Then,*

$$\|PQ\|_{\max} \le \|P\|_\infty \|Q\|_{\max} \quad \textit{and} \quad \|PQ\|_{\max} \le \|P\|_{\max}\|Q\|_1.$$

The following *multi-index* notation (see, e.g., [24]) will be used for convenience.

DEFINITION 4.4. *(Notation) Let* $\boldsymbol{\xi}$ *be a multi-index* $\boldsymbol{\xi} = (\xi_k, \xi_{k+1}\dots,\xi_l)$ *with* $\xi_j \in \{0,1\}$ *for* $k \le l$ *and* $k, l \in \mathbb{N}$*. Define*

$$\Delta^{\boldsymbol{\xi}}\left(\prod_{j=k}^{l} A_j\right) = \prod_{j=k}^{l} \Delta^{\xi_j} A_j,$$

where $\Delta^0 A_j = A_j$, $\Delta^1 A_j = \Delta A_j$. Also, denote $|\boldsymbol{\xi}| = \xi_k + \cdots + \xi_l$. It is easy to verify the following identity [24]:

$$\prod_{j=k}^{l}(A_j + \Delta A_j) = \prod_{j=k}^{l} A_j + \sum_{|\boldsymbol{\xi}|=1}^{l-k+1} \Delta^{\boldsymbol{\xi}} \left(\prod_{j=k}^{l} A_j \right). \tag{4.2}$$

Throughout the stability analysis, we suppose the generalized SSS/HSS matrices meet the following assumptions.

ASSUMPTION 4.5. *For a generalized SSS or HSS matrix A, assume the following.*

1. *Since the algorithm under consideration is matrix-vector multiplication and our focus is the stability study related to off-diagonal structures, we suppose all the entries of A that are not from $A_{\mathbf{D}}$ are nonzero. (Also note that A is used to approximate kernel matrices in this work.)*
2. *The generators of the generalized SSS form defined in (2.19) satisfy*
$$\|U_k\|_{\max} \le c_U, \quad \|V_l\|_{\max} \le c_U, \quad \|P_k\|_{\max} \le c_U, \quad \|Q_l\|_{\max} \le c_U,$$
$$\|R_s\|_1 \le c_T, \quad \|W_s^T\|_1 \le c_T, \quad \|B\|_{\max} \le c_B |A|_{\min}.$$
where c_B is a constant and $|A|_{\min}$ is the minimum magnitude of those entries of A that are not from $A_{\mathbf{D}}$. For convenience, we assume that U_k, V_l, P_k, Q_l have sizes $m \times p$ and R_s, W_s, B have sizes $p \times p$.
3. *The generators of the generalized HSS form like in (3.1) and (3.2) satisfy*
$$\|\mathcal{U}_i\|_{\max} \le c_U, \quad \|\mathcal{V}_i\|_{\max} \le c_U, \quad \|\mathcal{R}_i\|_{\infty} \le c_T, \quad \|\mathcal{W}_i\|_{\infty} \le c_T,$$
$$\|\mathcal{B}_i\|_{\max} \le c_B |A|_{\min}.$$
Note that for A in (2.22), the generators for $A_{\mathbf{L}}$ and $A_{\mathbf{U}}$ may be different. Nevertheless, we suppose all the relevant generators satisfy these norm bounds. For convenience, we also assume that the HSS tree $\mathcal{T}$ is a full binary tree with $L(\approx \log N)$ levels, where N is the number of leaves in $\mathcal{T}$. Also, we assume that $\mathcal{U}_i$, $\mathcal{V}_i$ have sizes $m \times p$ and $\mathcal{R}_i$, $\mathcal{W}_i$, $\mathcal{B}_i$ have sizes $p \times p$.
4. *For m, p, and N, we assume that $p \le m$ as in typical structured matrix algorithms (so that the leaf-level block sizes are not too small to have any cost saving), and assume $n\epsilon_{\text{mach}} \approx N\tau_m \ll 1$ as in typical backward stability analysis. (Note $n = Nm$.)*

Remark 4.6. To validate such assumptions within the context of this paper, we take the Cauchy kernel matrices in Section 2.1 as an example, where the generalized HSS matrix is constructed in Lemma 3.2 with the SoE expansion in the form of (3.3) and further satisfying $w_k, t_k \ge 0$ for all k. (The assumptions can be similarly validated for the other kernel matrices.) Notice that the generators $\mathcal{U}, \mathcal{V}, \mathcal{R}, \mathcal{W}$ have entries with magnitudes bounded by 1. In this case, $c_U = c_T = 1$. For the $\mathcal{B}$ generators,

$$\begin{aligned}\|\mathcal{B}_i\|_{\max} &= \max_{k=1,\ldots,p} |w_k| \le \sum_{k=1}^{p} w_k \le \sum_{k=1}^{p} w_k e^{[(b-a)-|x-y|]t_k} \\ &\le c_B \sum_{k=1}^{p} w_k e^{-|x-y|t_k} \quad \text{with} \quad c_B := \max_{k=1,\ldots,p} e^{(b-a)t_k}.\end{aligned} \tag{4.3}$$

Since this holds for all $|x-y| \ge \delta(b-a)$, we get $\|\mathcal{B}_i\|_{\max} \le c_B |A|_{\min}$. Note (3.3) means $|A|_{\min} = \min_{|x-y|\ge\delta(b-a)} \sum_{k=1}^{p} w_k e^{-|x-y|t_k}$.

We then present the stability analysis in the next two subsections.

4.2. Stability analysis for generalized SSS matrix-vector multiplications. The fast transforms in Section 2 are done through generalized SSS matrix-vector multiplications following the splitting (2.22): $A\mathbf{z} = A_{\mathbf{D}}\mathbf{z} + A_{\mathbf{L}}\mathbf{z} + A_{\mathbf{U}}\mathbf{z}$, where $A_{\mathbf{D}}\mathbf{z}$ is computed through a direct block banded matrix-vector multiplication, $A_{\mathbf{L}}\mathbf{z}$ is computed following Algorithm 2.1, and $A_{\mathbf{U}}\mathbf{z}$ is computed similarly to $A_{\mathbf{L}}\mathbf{z}$ because of the structural symmetry. Hence, it suffices to analyze the stability of $\mathbf{f}^+ = A_{\mathbf{L}}\mathbf{z}$.

Suppose $\mathrm{fl}(\mathbf{f}^+) = (A_{\mathbf{L}} + \Delta\tilde{A}_{\mathbf{L}})\mathbf{z}$ with $\Delta\tilde{A}_{\mathbf{L}}$ the perturbation due to the numerical computation. For a block $A_{k,l}$ with $k > l+1$ like in (2.19), the perturbation $\Delta\tilde{A}_{k,l}$ (a block of $\Delta\tilde{A}_{\mathbf{L}}$) is produced from $\mathrm{fl}(\mathbf{f}_k^+) = \sum_{l=1}^{k-2}(A_{k,l} + \Delta\tilde{A}_{k,l})\mathbf{z}_l$ via the two traversals in Algorithm 2.1. Our task is to find an entrywise bound for each such $\Delta\tilde{A}_{k,l}$. Two lemmas in the following measure the perturbations in these traversals and will be used in the proof of the main Theorem 4.9. The proofs of these lemmas are included in Appendix A. The discussions below involve the following notation from [24]:

$$\prod_{s=k}^{\geq l} A_s = \begin{cases} A_k A_{k-1}\cdots A_l, & k \geq l, \\ I, & k < l. \end{cases}$$

Lemma 4.7. *Suppose A in the form of (2.22) is a generalized SSS matrix satisfying the assumptions in Assumption 4.5. Then in the evaluation of $\mathbf{f}^+ = A_{\mathbf{L}}\mathbf{z}$, $\mathrm{fl}(\mathbf{v}_k)$ produced via the backward traversal stage of Algorithm 2.1 for $1 \leq k \leq N-2$ satisfies*

$$\mathrm{fl}(\mathbf{v}_k) = \mathbf{v}_k + \Delta\mathbf{v}_k \quad \textit{with} \quad \Delta\mathbf{v}_k = \sum_{l=1}^{k}\left[\sum_{|\boldsymbol{\xi}|=1}^{k-l+1}\Delta^{\boldsymbol{\xi}}\left(\left(\prod_{s=k}^{\geq l+1}\tilde{R}_s\right)\tilde{Q}_l^T\right)\right]\mathbf{z}_l, \tag{4.4}$$

where

$$\Delta^{\xi}\tilde{Q}_l^T := \begin{cases} Q_l^T, & \xi = 0, \\ \Delta Q_l^T + \Delta Y_l Q_l^T + \Delta Y_l \Delta Q_l^T, & \xi = 1, \end{cases} \quad \textit{with} \quad |\Delta Q_l^T| \leq \tau_m |Q_l^T|, \tag{4.5}$$

$$\Delta^{\xi}\tilde{R}_s := \begin{cases} R_s, & \xi = 0, \\ \Delta R_s + \Delta Y_s R_s + \Delta Y_s \Delta R_s, & \xi = 1, \end{cases} \quad \textit{with} \quad |\Delta R_s| \leq \tau_p |R_s|, \tag{4.6}$$

$|\Delta Y_l| \leq \epsilon_{\mathrm{mach}} I$, and the notation in (4.1) is used. Besides, we have

$$\|\Delta^1\tilde{Q}_l^T\|_1 \leq p c_U \tau_{3m}, \quad \|\Delta^1\tilde{R}_s\|_1 \leq c_T \tau_{3p}.$$

Lemma 4.8. *Suppose A in the form of (2.22) is a generalized SSS matrix satisfying the assumptions in Assumption 4.5. Then in the evaluation of $\mathbf{f}^+ = A_{\mathbf{L}}\mathbf{z}$, $\mathrm{fl}(\mathbf{f}_k^+)$ produced after the forward traversal in Algorithm 2.1 for $3 \leq k \leq N$ satisfies*

$$\mathrm{fl}(\mathbf{f}_k^+) = \mathbf{f}_k^+ + \Delta\mathbf{f}_k^+ \quad \textit{with} \quad \Delta\mathbf{f}_k^+ = \sum_{l=1}^{k-2}\Delta\tilde{A}_{k,l}\mathbf{z}_l,$$

where

$$\Delta\tilde{A}_{k,l} = \sum_{|\boldsymbol{\xi}|=1}^{k-l+1}\Delta^{\boldsymbol{\xi}}\left[P_k B\left(\prod_{s=k-2}^{\geq l+1}\tilde{R}_s\right)\tilde{Q}_l^T\right], \quad |\Delta P_k| \leq \tau_p|P_k|, \quad |\Delta B| \leq \tau_p |B|,$$

and $\Delta^{\xi}\tilde{R}_s$ and $\Delta^{\xi}\tilde{Q}_l^T$ are respectively defined in (4.5) and (4.6). Besides,

$$|\Delta\tilde{A}_{k,l}| \leq \frac{4}{3}(k-l+1)\tau_{3m}p^2 c_B c_U^2 c_T^{k-l-2}|A_{k,l}| \quad \textit{for} \quad k > l+1. \tag{4.7}$$

We can now inspect the stability of generalized SSS matrix-vector multiplications.

THEOREM 4.9. *Suppose A in the form of (2.22) is a generalized SSS matrix satisfying the assumptions in Assumption 4.5. Then the matrix-vector multiplication of A with a vector $\mathbf{z}$ via Algorithm 2.1 satisfies*

$$\mathrm{fl}(A\mathbf{z}) = (A + \Delta A)\mathbf{z} \quad \textit{with}$$
$$|\Delta A| \le \max\{1, \frac{4}{3}Np^2 c_B c_U^2 \max\{1, c_T^{N-3}\}\}\tau_{3m+4}|A|.$$

Proof. We discuss the perturbations to a nonzero block $A_{k,l}$ from $A_{\mathbf{L}}$, $A_{\mathbf{U}}$, or $A_{\mathbf{D}}$ in (2.22) due to the multiplications $\mathbf{f}^0 = A_{\mathbf{D}}\mathbf{z}$, $\mathbf{f}^+ = A_{\mathbf{L}}\mathbf{z}$, and $\mathbf{f}^- = A_{\mathbf{U}}\mathbf{z}$, respectively. According to Lemma 4.1,

$$\mathrm{fl}(\mathbf{f}^0) = (A_{\mathbf{D}} + \Delta\tilde{A}_{\mathbf{D}})\mathbf{z},$$

where $\Delta\tilde{A}_{\mathbf{D}}$ has the same block structure as $A_{\mathbf{D}}$ and its blocks satisfy $|\Delta\tilde{A}_{k,l}| \le \tau_{3m}|A_{k,l}|$ since there are at most $3m$ nonzero columns in each block row of $A_{\mathbf{D}}$.

Next,

$$\mathrm{fl}(\mathbf{f}^+) = (A_{\mathbf{L}} + \Delta\tilde{A}_{\mathbf{L}})\mathbf{z},$$

where $\Delta\tilde{A}_{\mathbf{L}}$ has the same block structure as $A_{\mathbf{L}}$ and its blocks $\Delta\tilde{A}_{k,l}$ satisfy (4.7) in Lemma 4.8. Then, by the structure symmetry between $A_{\mathbf{U}}$ and $A_{\mathbf{L}}$, Lemma 4.7 and Lemma 4.8 also apply to $A_{k,l}$ from $A_{\mathbf{U}}$ or when $k < l-1$. Thus,

$$\mathrm{fl}(\mathbf{f}^-) = (A_{\mathbf{U}} + \Delta\tilde{A}_{\mathbf{U}})\mathbf{z},$$

where $\Delta\tilde{A}_{\mathbf{U}}$ has the same block structure as $A_{\mathbf{U}}$ and its blocks $\Delta\tilde{A}_{k,l}$ satisfy the same bound (4.7) in Lemma 4.8 when $k > l+1$. Thus, for any $1 \le k, l \le N$,

$$|\Delta\tilde{A}_{k,l}| \le \begin{cases} \tau_{3m}|A_{k,l}|, & |k-l| \le 1, \\ \frac{4}{3}(|k-l|+1)p^2 c_B c_U^2 c_T^{|k-l|-2}\tau_{3m}|A_{k,l}|, & \text{otherwise.} \end{cases} \tag{4.8}$$

In the final summation stage, we then have

$$\begin{aligned} \mathrm{fl}(A\mathbf{z}) &= \mathrm{fl}\left(\mathrm{fl}(\mathbf{f}^0) + \mathrm{fl}(\mathbf{f}^+) + \mathrm{fl}(\mathbf{f}^-)\right) \\ &= (I + \Delta f_2)\Big((I + \Delta f_1)(\mathbf{f}^0 + \Delta\tilde{A}_{\mathbf{D}}\mathbf{z} + \mathbf{f}^+ + \Delta\tilde{A}_{\mathbf{L}}\mathbf{z}) + \mathbf{f}^- + \Delta\tilde{A}_{\mathbf{U}}\mathbf{z}\Big), \end{aligned} \tag{4.9}$$

where Δf_1 results from the floating point addition $\mathrm{fl}(\mathbf{f}^0) + \mathrm{fl}(\mathbf{f}^+)$ and Δf_2 results from the further floating point addition of $\mathrm{fl}(\mathbf{f}^-)$ and they are diagonal matrices satisfying $|\Delta f_1| \le \epsilon_{\mathrm{mach}} I$, $|\Delta f_2| \le \epsilon_{\mathrm{mach}} I$.

Let

$$\Delta\tilde{A} = \Delta\tilde{A}_{\mathbf{D}} + \Delta\tilde{A}_{\mathbf{L}} + \Delta\tilde{A}_{\mathbf{U}}. \tag{4.10}$$

Then

$$\mathrm{fl}(A\mathbf{z}) = \mathbf{f}^0 + \mathbf{f}^+ + \mathbf{f}^- + (\Delta A)\mathbf{z} = A\mathbf{z} + (\Delta A)\mathbf{z}, \tag{4.11}$$
$$\Delta A = \Delta\tilde{A} + \Delta f_2(A + \Delta\tilde{A}) + (\Delta f_2 \Delta f_1 + \Delta f_1)(A_{\mathbf{D}} + A_{\mathbf{L}} + \Delta\tilde{A}_{\mathbf{D}} + \Delta\tilde{A}_{\mathbf{L}}). \tag{4.12}$$

Since Δf_1 and Δf_2 are diagonal and $A_{\mathbf{D}}$ and $A_{\mathbf{L}}$ have non-overlapping nonzero patterns, we then have

$$
\begin{aligned}
(4.13)\qquad |\Delta A| &\le |\Delta\tilde{A}| + |\Delta f_2|(|A| + |\Delta\tilde{A}|)\\
&\quad + (|\Delta f_2||\Delta f_1| + |\Delta f_1|)(|A_{\mathbf{D}}| + |A_{\mathbf{L}}| + |\Delta\tilde{A}_{\mathbf{D}}| + |\Delta\tilde{A}_{\mathbf{L}}|)\\
&\le |\Delta\tilde{A}| + \epsilon_{\mathrm{mach}}(|A| + |\Delta\tilde{A}|) + (\epsilon_{\mathrm{mach}} + \epsilon_{\mathrm{mach}}^2)(|A| + |\Delta\tilde{A}|)\\
&\le (2\epsilon_{\mathrm{mach}} + \epsilon_{\mathrm{mach}}^2)|A| + (1+\epsilon_{\mathrm{mach}})^2|\Delta\tilde{A}|\\
&\le (\tau_2 + (1+\tau_2)\max\{1, \frac{4}{3}Np^2 c_B c_U^2 \max\{1, c_T^{N-3}\}\}\tau_{3m})|A|\\
&\le \max\{1, \frac{4}{3}Np^2 c_B c_U^2 \max\{1, c_T^{N-3}\}\}\tau_{3m+4}|A|,
\end{aligned}
$$

where the last two steps follow from Lemma 4.2 and (4.8). ∎

Theorem 4.9 shows that generalized SSS transforms may potentially have exponential error growth with respect to N when c_T, the norm bound of the translation generators, is larger than 1. (Note N is proportional to n.) Translation generators with large norms may cause instability. On the other hand, SoE expansions provide an effective way to resolve this issue by producing nice bounds for the translation generators.

COROLLARY 4.10. *Suppose the generators of the generalized SSS matrix A are produced via SoE expansions as in (3.4) so that the generators further satisfy $c_T = c_U = 1$. Then, generalized SSS matrix-vector multiplications via Algorithm 2.1 satisfy*

$$\mathrm{fl}(A\mathbf{z}) = (A + \Delta A)\mathbf{z} \quad \textit{with} \quad |\Delta A| \le \max\{1, \frac{4}{3}Np^2 c_B\}\tau_{3m+4}|A|.$$

In this corollary, the error grows at most linearly with respect to N.

4.3. Stability analysis for generalized HSS matrix-vector multiplications. The previous subsection shows the importance of controlling the norms of translation generators. In practice, it is possible for structured representations to have translation operators with norms larger than 1. In this subsection, we consider another important factor that impacts the stability of transforms: the algorithm architecture. As mentioned in [27, 28], hierarchical structured (like HSS) algorithms can further reduce the length of the error propagation path or the number of times the error gets magnified by.

Theorem 4.13 below shows how the HSS architecture benefits the stability and can be shown based on the following two lemmas, which are proved in Appendix A.

LEMMA 4.11. *Suppose A in the form of (2.22) is a generalized HSS matrix satisfying the assumptions in Assumption 4.5. Then in the evaluation of $\hat{\mathbf{f}}^+ = \hat{A}_{\mathbf{L}}\hat{\mathbf{z}}$, $\mathrm{fl}(\mathbf{y}^{(i)})$ produced via the bottom-up traversal in (3.10) for $1 \le i \le L$ satisfies*

$$(4.14)\qquad \mathrm{fl}(\mathbf{y}^{(i)}) = \mathbf{y}^{(i)} + \Delta\mathbf{y}^{(i)} \quad \textit{with} \quad \Delta\mathbf{y}^{(i)} = \left(\sum_{|\boldsymbol{\xi}|=1}^{L-i+2} \Delta^{\boldsymbol{\xi}}\Big(B^{(i)}\prod_{j=i}^{L}(V^{(j)})^T\Big)\right)\hat{\mathbf{z}},$$

where $|\Delta B^{(i)}| \le \tau_p|B^{(i)}|$ and $|(\Delta V^{(j)})^T| \le \tau|(V^{(j)})^T|$ for $\tau = \max\{\tau_m, \tau_{2p}\}$. Besides, $\|\Delta^{\boldsymbol{\xi}}(\prod_{j=i}^{L}(V^{(j)})^T)\|_1 \le p c_U c_T^{L-i}\tau_{2m}^{|\boldsymbol{\xi}|}$.

LEMMA 4.12. *Suppose A in the form of (2.22) is a generalized HSS matrix satisfying the assumptions in Assumption 4.5. Then in the evaluation of $\hat{\mathbf{f}}^+ = \hat{A}_{\mathbf{L}}\hat{\mathbf{z}}$, $\mathrm{fl}(\hat{\mathbf{f}}^+)$ produced after the top-down traversal in (3.11) satisfies*

$$\mathrm{fl}(\hat{\mathbf{f}}^+) = (\hat{A}_{\mathbf{L}} + \Delta\hat{A}_{\mathbf{L}})\hat{\mathbf{z}} \quad \text{with} \quad \Delta\hat{A}_{\mathbf{L}} = \sum_{k=1}^{L} \Delta\hat{A}_{\mathbf{L}}^{(k)}, \tag{4.15}$$

where

$$\Delta\hat{A}_{\mathbf{L}}^{(k)} = \sum_{|\boldsymbol{\xi}|=1}^{2L-2k+3} \Delta^{\boldsymbol{\xi}} \left(\Big(\prod_{j=L}^{\geq k} \tilde{U}^{(j)} \Big) B^{(k)} \Big(\prod_{j=k}^{L} (V^{(j)})^T \Big) \right), \tag{4.16}$$

$\Delta V^{(j)}$ and $\Delta B^{(k)}$ are defined in Lemma 4.11, and

$$\Delta^{\xi}\tilde{U}^{(j)} := \begin{cases} U^{(j)}, & \text{if } \xi = 0, \\ \Delta U^{(j)} + U^{(j)}\Delta Z^{(j-1)} + \Delta U^{(j)}\Delta Z^{(j-1)}, & \text{if } \xi = 1, \end{cases} \quad \text{with}$$
$$|\Delta Z^{(j)}| \leq \epsilon_{\mathrm{mach}} I, \quad |\Delta U^{(j)}| \leq \tau_p |U^{(j)}|.$$

Besides,

$$|\Delta\hat{A}_{\mathbf{L}}| \leq \frac{4}{3}(2L+1)p^2 c_B c_U^2 \max\{1, c_T^{2L-2}\}\tau_{3m}|\hat{A}_{\mathbf{L}}|. \tag{4.17}$$

We can then show the backward stability of transforms with generalized HSS matrix-vector multiplications.

THEOREM 4.13. *Suppose A in the form of (2.22) is a generalized HSS matrix satisfying the assumptions in Assumption 4.5. Assume the matrix-vector multiplication of A with a vector $\mathbf{z}$ is performed with $A_{\mathbf{L}}\mathbf{z}$ computed via the traversals in (3.10) and (3.11), $A_{\mathbf{U}}\mathbf{z}$ computed similarly based on structure symmetry, and $A_{\mathbf{D}}\mathbf{z}$ computed directly. Then*

$$\mathrm{fl}(A\mathbf{z}) = (A + \Delta A)\mathbf{z} \quad \text{with}$$
$$|\Delta A| \leq \max\{1, \frac{4}{3}(2L+1)p^2 c_B c_U^2 \max\{1, c_T^{2L-2}\}\}\tau_{3m+4}|A|.$$

Proof. The framework of the stability analysis is similar to that in the proof of Theorem 4.9. For convenience, we follow the same definitions and notation as in the proof of Theorem 4.9 up to (4.12).

Note that $\Delta\tilde{A}_{\mathbf{L}}$ and $\Delta\tilde{A}_{\mathbf{U}}$ in (4.10) are perturbations generated from HSS matrix-vector multiplications that have the same nonzero structure as $A_{\mathbf{L}}$ and $A_{\mathbf{U}}$, respectively. According to (4.17) in Lemma 4.12, we have

$$|\Delta\tilde{A}_{\mathbf{L}}| \leq \frac{4}{3}(2L+1)p^2 c_B c_U^2 \max\{1, c_T^{2L-2}\}\tau_{3m}|A_{\mathbf{L}}|.$$

By the structure symmetry between $A_{\mathbf{U}}$ and $A_{\mathbf{L}}$, Lemmas 4.11 and 4.12 also apply to $\Delta\tilde{A}_{\mathbf{U}}$. Then,

$$|\Delta\tilde{A}_{\mathbf{U}}| \leq \frac{4}{3}(2L+1)p^2 c_B c_U^2 \max\{1, c_T^{2L-2}\}\tau_{3m}|A_{\mathbf{U}}|.$$

Since the nonzero patterns of $\Delta\tilde{A}_{\mathbf{D}}$, $\Delta\tilde{A}_{\mathbf{L}}$, and $\Delta\tilde{A}_{\mathbf{U}}$ do not overlap, we then have

$$|\Delta\tilde{A}| \leq \max\{1, \frac{4}{3}(2L+1)p^2 c_B c_U^2 \max\{1, c_T^{2L-2}\}\}\tau_{3m}|A|,$$

Accordingly, by (4.13), we have

$$\begin{aligned}|\Delta A| &\leq |\Delta\tilde{A}| + (2\epsilon_{\text{mach}} + \epsilon_{\text{mach}}^2)(|A| + |\Delta\tilde{A}|)\\ &\leq (\tau_2 + (1+\tau_2)\max\{1, \frac{4}{3}(2L+1)p^2 c_B c_U^2 \max\{1, c_T^{2L-2}\}\}\tau_{3m})|A|\\ &\leq \max\{1, \frac{4}{3}(2L+1)p^2 c_B c_U^2 \max\{1, c_T^{2L-2}\}\}\tau_{3m+4}|A|. \qquad \square\end{aligned}$$

Theorem 4.13 shows that the generalized HSS transform is backward stable. This holds even if the norm bound c_T of the translation generators is larger than 1. In that case, the backward error has polynomial (instead of exponential) growth. With further control on the norm bounds of the generators (via the SoE expansions), the error propagation of the generalized HSS transform can be even reduced to poly-logarithmic.

COROLLARY 4.14. *Suppose the generators of the generalized HSS matrix A are produced via the SoE expansions as in (3.4) so that the generators further satisfy $c_T = c_U = 1$. Then the generalized HSS matrix-vector multiplication as in Theorem 4.13 satisfies*

$$\text{fl}(A\mathbf{z}) = (A + \Delta A)\mathbf{z} \quad \textit{with} \quad |\Delta A| \leq \max\{1, \frac{4}{3}(2L+1)p^2 c_B\}\tau_{3m+4}|A|.$$

Remark 4.15. (*Key observations*) The studies in this section provide some useful insights into the stability of rank-structured algorithms like matrix-vector multiplications. Two key components play crucial roles in the stability: *algorithm architecture* and *norm bounds of translation generators*. As compared with sequential architectures, hierarchical architectures help reduce the length of the error propagation path from $\mathcal{O}(n)$ to $\mathcal{O}(\log n)$. See Theorems 4.9 and 4.13. Smaller norm bounds for translation generators yield lower error growth factors. Depending on these two components, the possible error growth patterns are as follows:

- exponential (e.g., generalized SSS with $c_T > 1$ as in Theorem 4.9);
- polynomial (e.g., generalized SSS with $c_T = 1$ as in Corollary 4.10, and HSS with $c_T > 1$ as in Theorem 4.13);
- poly-logarithmic (e.g, generalized HSS with $c_T = 1$ as in Corollary 4.14).

Thus, to perform the fast transforms in this paper, using generalized HSS structures derived from SoE expansions potentially has the best stability.

5. Numerical experiments.

In this section, we use some numerical experiments to illustrate the stability of transformations via generalized SSS and HSS matrix-vector multiplications. We also confirm the high accuracy and efficiency of the generalized HSS matrix-vector multiplication.

Four kernel functions in Section 2 are considered in the tests: Cauchy $(1/(x-y))$, Gaussian $(e^{-(x-y)^2})$, logarithmic $(\log|x-y|)$, and square-root $(1/\sqrt{|x^2-y^2|})$. For convenience, we let the data sets $\mathbf{x}$ and $\mathbf{y}$ be identical. Whenever a diagonal entry $\kappa(x_i, x_i)$ is undefined, it is set to be zero. This does not really matter for the stability tests related to off-diagonal structures.

For each kernel matrix, SoE expansions are used to obtain a generalized SSS or HSS approximation A. For SoE approximations to Cauchy, logarithmic, and square-root kernels, a parameter δ like in (2.3) is needed to determine the valid intervals of the approximations. With the finest-level block size m fixed, appropriate δ and the corresponding quadrature nodes and weights are chosen for different data sizes n to meet conditions (2.10)– (2.11). Given a tolerance $\epsilon = 10^{-15}$ of SoE approximations, some sets of quadrature nodes and weights for $\delta = \frac{1}{4^k}$, $k = 1, 2, \ldots, 10$ are precomputed. The number of quadrature points p varies from 30 to 67 among such δ. This is also the numerical rank for the off-diagonal low-rank approximations. For the Gaussian kernel case, they do not depend on δ (see Section 2.2.1) because the kernel function has no singularity at 0. Accordingly, one set of quadrature nodes and weights is sufficient for all Gaussian kernel tests. Note that this set of quadrature nodes and weights is calculated from a piece of Matlab code in a double-precision environment. This restricts a relative accuracy to up to 10^{-12} for the Gaussian kernel case [21].

The tests are performed in Matlab R2021a on a server with two Intel Xeon E5-2660V3 CPUs and 192GB of memory.

5.1. Stability of generalized SSS and HSS transforms. In exact arithmetic, the generalized SSS and HSS approximations A are equal. On the other hand, they have different stability behaviors in numerical computations.

For the matrix-vector multiplication $\mathbf{f} = A\mathbf{z}$, the backward error of an approximate product $\tilde{\mathbf{f}}$ is as follows [18, (3.6)]:

$$\varepsilon_{\mathrm{bwd}} = \min\{\varepsilon > 0 : \tilde{\mathbf{f}} = (A + \Delta A)\mathbf{z}, |\Delta A| \le \varepsilon |A|\} = \max_{i=1:n} \frac{|\tilde{\mathbf{f}}_i - \mathbf{f}_i|}{(|A||\mathbf{z}|)_i},$$

where ΔA is the perturbation of A when performing the matrix-vector multiplication. This guarantees $|\Delta A| \le \varepsilon_{\mathrm{bwd}}|A|$. According to the stability analysis in Section 4, results like (4.7) indicate that a block $A_{k,l}$ potentially has larger perturbation errors when $|k-l|$ is larger. With fixed finest-level block size m, in order to test large $|k-l|$, large matrix sizes n are needed. Since it becomes impractical to evaluate $\mathbf{f}$ via dense A when n is too large, we can just look at the $m \times m$ finest-level lower-left corner block of A with row index set $n-m+1 : n$ and column index set $1 : m$. Denote such a block by A^c. We evaluate $\mathbf{f}^c = A^c\mathbf{z}^c$ with a vector $\mathbf{z}^c$. With numerical evaluations using either generalized SSS or HSS forms, we obtain $\tilde{\mathbf{f}}^c = \mathrm{fl}(A^c\mathbf{z}^c)$ and inspect the backward error

$$\varepsilon^c_{\mathrm{bwd}} = \max_{i=1:m} \frac{|\tilde{\mathbf{f}}^c_i - \mathbf{f}^c_i|}{(|A^c||\mathbf{z}^c|)_i}. \tag{5.1}$$

In the stability test, we use $\mathbf{x}$ with n equal-spaced data points distributed on $[0, 1]$, where

$$n = 2^k \times 10^4, \quad k = 1, 2, \ldots, 10.$$

Set the size of the corner block A^c to be $m = 100$. A^c is multiplied with $\mathbf{z}^c = (1, \ldots, 1)^T$.

With generalized SSS/HSS forms, we plot $\varepsilon^c_{\mathrm{bwd}}$ in Figure 5.1. For the generalized SSS cases, $\varepsilon^c_{\mathrm{bwd}}$ increases with n for different kernels. For the generalized HSS cases, $\varepsilon^c_{\mathrm{bwd}}$ remains nearly steady for different n, which aligns with Lemma 4.12. The results are consistent with our analysis and confirm the superior stability of transforms via generalized HSS structures.

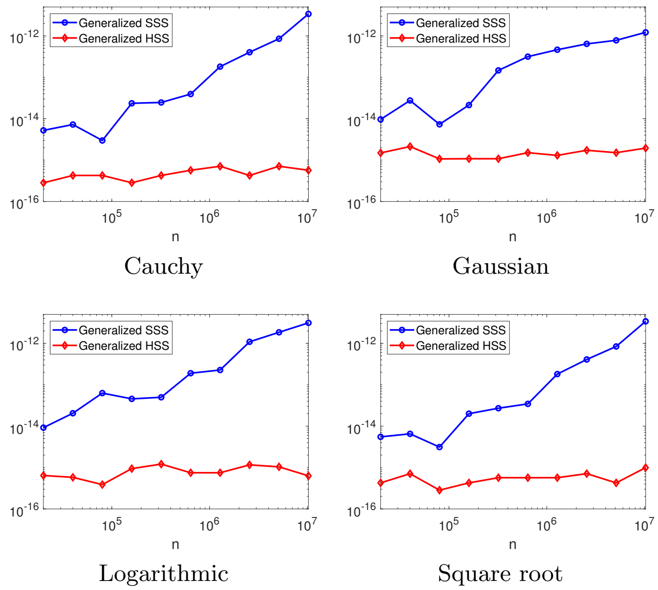


FIG. 5.1. *Backward errors* $\varepsilon^c_{\mathrm{bwd}}$ *in* (5.1)*, with* A *being generalized SSS or HSS approximations to some kernel matrices.*

5.2. Efficiency and accuracy of generalized HSS matrix-vector multiplications. We now demonstrate the efficiency and accuracy of generalized HSS matrix-vector multiplications. $\mathbf{x}$ has n random data points uniformly distributed on $[0, 1]$. We set the finest-level block size $m = 200$.

For Cauchy kernel matrices with varying n, we report the time to construct the generalized HSS approximation from SoE expansions, the time to evaluate matrix-vector products with the generators, and the storage for the generators. See Figure 5.2, which shows nearly linear complexity and storage. For the other kernels mentioned above, the results are similar.

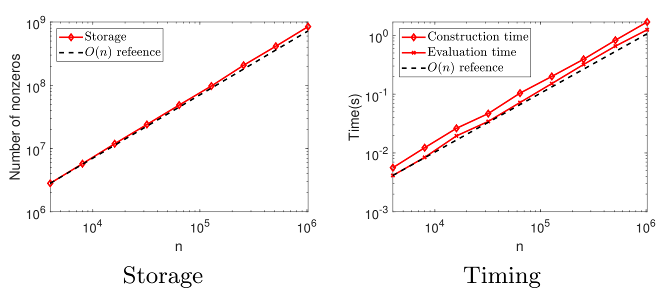


FIG. 5.2. *Storage and timing for Cauchy kernel matrices.*

For the four types of kernel matrices, Table 5.1 shows the relative errors for the generalized HSS matrix-vector multiplications. The results confirm the high numerical accuracy of the multiplications.

6. Conclusions. This work reveals how some popular fast transforms via SoE expansions are eventually performing certain structured matrix-vector multiplications. This in turn leads to a valuable strategy for approximating some kernel matrices via

TABLE 5.1
Relative errors $\frac{\|A\mathbf{z}-H\mathbf{z}\|_2}{\|H\mathbf{z}\|_2}$ for different kernels and data sizes, where H is the original kernel matrix.

n	Cauchy	Gaussian	Logarithmic	Square-root
$(\times 10^3)$	$\epsilon = 10^{-15}$	$\epsilon = 10^{-12}$	$\epsilon = 10^{-15}$	$\epsilon = 10^{-15}$
4	5.24e-17	2.73e-13	1.34e-15	1.83e-16
8	3.42e-16	3.45e-13	7.25e-16	3.06e-16
16	1.36e-15	3.93e-13	3.68e-15	6.17e-16
32	9.62e-16	4.11e-13	1.53e-14	7.27e-16
64	1.79e-15	4.28e-13	9.66e-15	6.83e-16
128	2.95e-15	4.38e-13	2.18e-15	9.91e-16
256	3.59e-15	4.42e-13	1.95e-14	1.66e-15
512	2.28e-14	4.43e-13	1.83e-14	2.08e-15
1024	5.15e-14	4.43e-13	2.17e-15	3.13e-15

SoE expansions. It also gives an intuitive way to study the backward stability of these transforms. We have shown the stability limitation of the previous transforms based on generalized SSS forms, and demonstrated how the stability may be further improved via generalized HSS forms. Following the stability studies, the work eventually provides a comprehensive picture of stability issues of structured algorithms. That is, algorithm architectures and norm bounds of translation generators determine the backward stability. Hierarchical structured algorithms are typically preferred to sequential ones. Methods like SoE expansions are nice ways to produce generators with controlled norms. In future work, it would be interesting to lay out the detailed matrix structures for algorithms based on higher dimensional SoE expansions like mentioned in Section 2.3. We expect that the essential ideas of our stability studies can be naturally extended to higher dimensions.

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] G. Beylkin and L. Monzón, *On approximation of functions by exponential sums*, Appl. Comput. Harmon. Anal., 19 (2005), pp. 17–48.

[3] G. Beylkin, V. Cheruvu, and F. Pérez, *Fast adaptive algorithms in the non-standard form for multidimensional problems*, Appl. Comput. Harmon. Anal., 24 (2008), pp. 354–377.

[4] G. Beylkin, C. Kurcz, and L. Monzón, *Fast algorithms for Helmholtz Green's functions*, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 464 (2008), pp. 3301–3326.

[5] G. Beylkin, C. Kurcz, and L. Monzón, *Fast convolution with the free space Helmholtz Green's function*, J. Comput. Phys., 228 (2009), pp. 2770–2791.

[6] G. Beylkin and L. Monzón, *Approximation by exponential sums revisited*, Appl. Comput. Harmon. Anal., 28 (2010), pp. 131–149.

[7] J. Bremer, Z. Gimbutas, and V. Rokhlin, *A nonlinear optimization procedure for generalized Gaussian quadratures*, SIAM J. Sci. Comput., 32 (2010), pp. 1761–1788.

[8] S. Chandrasekaran and M. Gu, *Fast and stable algorithms for banded plus semiseparable matrices*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 373–384.

[9] 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.

[10] 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.

[11] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals and, A. J. van der Veen, *Fast stable solver*

for sequentially semi-separable linear systems of equations, in International Conference on High-Performance Computing, Springer, Berlin, Heidelberg, 2002, pp. 545–554.

[12] A. Dutt, M. Gu, and V. Rokhlin, *Fast algorithms for polynomial interpolation, integration, and differentiation*, SIAM J. Numer. Anal., 33 (1996), pp. 1689–1711.

[13] Z. Gimbutas, N. F. Marshall, and V. Rokhlin, *A fast simple algorithm for computing the potential of charges on a line*, Appl. Comput. Harmon. Anal., 49 (2020), pp. 815–830.

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

[15] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numer., 6 (1997), pp. 229–269.

[16] P. K. Gupta, S. Niwas, and N. Chaudhary, *Fast computation of Hankel Transform using orthonormal exponential approximation of complex kernel function*, J. Earth Syst. Sci., 115 (2006), pp. 267–276.

[17] N. J. Higham, *Accuracy and stability of numerical algorithms*, 2nd ed., SIAM, 2002.

[18] N. J. Higham and T. Mary, *Sharper probabilistic backward error analysis for basic linear algebra kernels with random data*, SIAM J. Sci. Comput., 42 (2020), pp. A3427-A3446.

[19] T. Hrycak and V. Rokhlin, *An improved fast multipole algorithm for potential fields*, SIAM J. Sci. Comput., 19 (1998), pp. 1804–1826.

[20] H. Ikeno, *Spherical Bessel transform via exponential sum approximation of spherical Bessel function*, J. Comput. Phys., 355 (2018), pp. 426–435.

[21] S. Jiang and L. Greengard, *Approximating the Gaussian as a sum of exponentials and its applications to the fast gauss transform*, Commun. Comput. Phys., 31 (2022), pp. 1–26.

[22] P.-D. Létourneau, C. Cecka, and E. Darve, *Cauchy fast multipole method for general analytic kernels*, SIAM J. Sci. Comput., 36 (2014), pp. A396–A426.

[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] X. Ou, M. Michelle, and J. Xia, *A stable matrix version of the 2D fast multipole method*, SIAM J. Matrix Anal. Appl., 46 (2025), pp. 530–560.

[25] L. N. Trefethen and M. H. Gutknecht, *The Carathéodory–Fejér method for real rational approximation*, SIAM J. Numer. Anal., 20 (1983), pp.420–436.

[26] L. N. Trefethen, J. A. C. Weideman, and T. Schmelzer, *Talbot quadratures and rational approximations*, BIT, 46 (2006), pp. 653–670.

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

[28] 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.

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

[30] T. Yanai, G. I. Fann, Z. Gan, R. J. Harrison, and G. Beylkin, *Multiresolution quantum chemistry in multiwavelet bases: Hartree-Fock exchange*, J. Chem. Phys., 121 (2004), pp. 6680–6688.

[31] N. Yarvin and V. Rokhlin, *Generalized Gaussian quadratures and singular value decompositions of integral operators*, SIAM J. Sci. Comput., 20 (1998), pp. 699–718.

[32] N. Yarvin and V. Rokhlin, *An improved fast multipole algorithm for potential fields on the line*, SIAM J. Numer. Anal., 36 (1999), pp. 629–666.

A. Appendix: Proofs of the lemmas in Section 4. This appendix includes proofs for some lemmas in the stability analysis. We first give a lemma that will be used in later proofs.

Lemma A.1. *Let $n \in \mathbb{N}_+$ and ϵ be a small quantity such that $0 < n\epsilon < 1/2$. Then*

$$\sum_{k=1}^{n} \binom{n}{k} \epsilon^k \le \frac{4}{3} n\epsilon.$$

Proof. By the binomial theorem, for $0 < n\epsilon < 1/2$,

$$\sum_{k=1}^{n} \binom{n}{k} \epsilon^k \le \sum_{k=1}^{n} \frac{(n\epsilon)^k}{k!} \le (n\epsilon) \sum_{k=0}^{n-1} \left(\frac{n\epsilon}{2}\right)^k \le \quad \frac{n\epsilon}{1-(n\epsilon)/2} \le \frac{4}{3} n\epsilon.$$

Proof of Lemma 4.7. In the backward traversal stage, we have the following recursive relation of $\mathrm{fl}(\mathbf{v}_k)$ through the update formula $\mathbf{v}_k = Q_k^T\mathbf{z}_k + R_k\mathbf{v}_{k-1}$:

(A.1)
$$\mathrm{fl}(\mathbf{v}_k) = \begin{cases} (I+\Delta Y_1)(Q_1+\Delta Q_1)^T\mathbf{z}_1, & k=1, \\ (I+\Delta Y_k)\left[(Q_k+\Delta Q_k)^T\mathbf{z}_k + (R_k+\Delta R_k)\,\mathrm{fl}(\mathbf{v}_{k-1})\right], & 2\le k\le N-2, \end{cases}$$

where $|\Delta Q_k^T| \le \tau_m|Q_k^T|$, $|\Delta R_k|\le\tau_p|R_k|$ by Lemma 4.1, and $|\Delta Y_k|\le\epsilon_{\mathrm{mach}}I$.

By expanding the recursive relation (A.1) and applying identity (4.2), we obtain the following summation form of $\mathrm{fl}(\mathbf{v}_k)$, for $k=1,\ldots,N-2$:

$$\begin{aligned}
\mathrm{fl}(\mathbf{v}_k) &= \sum_{l=1}^{k}\left[\prod_{s=k}^{\ge l+1}(I+\Delta Y_s)(R_s+\Delta R_s)\right](I+\Delta Y_l)(Q_l+\Delta Q_l)^T\mathbf{z}_l \\
&= \sum_{l=1}^{k}\left[\prod_{s=k}^{\ge l+1}R_s + \sum_{|\boldsymbol{\xi}|=1}^{k-l}\Delta^{\boldsymbol{\xi}}\left(\prod_{s=k}^{\ge l+1}\tilde{R}_s\right)\right](Q_l+\Delta\tilde{Q}_l)^T\mathbf{z}_l \\
&=: \mathbf{v}_k + \Delta\mathbf{v}_k \quad \text{with} \\
\Delta\mathbf{v}_k &= \sum_{l=1}^{k}\left[\sum_{|\boldsymbol{\xi}|=1}^{k-l+1}\Delta^{\boldsymbol{\xi}}\left[\left(\prod_{s=k}^{\ge l+1}\tilde{R}_s\right)\tilde{Q}_l^T\right]\right]\mathbf{z}_l,
\end{aligned}$$

where $\Delta^{\xi}\tilde{Q}_l^T$ and $\Delta^{\xi}\tilde{R}_s$, for $\xi\in\{0,1\}$, are given by (4.5) and (4.6), respectively.

With Assumption 4.5, Q_l^T is a $p\times m$ matrix for each l. Then, $\|Q_l^T\|_1\le c_U p$. Thus, with (4.5) and (4.6),

$$\begin{aligned}
\|\Delta^1\tilde{Q}_l^T\|_1 &\le \tau_m\|Q_l^T\|_1 + \epsilon_{\mathrm{mach}}\|Q_l^T\|_1 + \epsilon_{\mathrm{mach}}\tau_p\|Q_l^T\|_1 \le pc_U\tau_{3m}, \\
\|\Delta^1\tilde{R}_s\|_1 &\le \tau_p\|R_s\|_1 + \epsilon_{\mathrm{mach}}\|R_s\|_1 + \epsilon_{\mathrm{mach}}\tau_p\|R_s\|_1 \le c_T\tau_{3p}. \qquad \square
\end{aligned}$$

Proof of Lemma 4.8. In this stage, we compute $\mathrm{fl}(\mathbf{f}_k^+)$ by multiplying P_kB with $\mathrm{fl}(\mathbf{v}_{k-2})$ defined in (4.4). Combing with the definition of $\Delta\mathbf{v}_{k-2}$ in Lemma 4.7 to get

$$\begin{aligned}
\mathrm{fl}(\mathbf{f}_k^+) &= (P_k+\Delta P_k)(B+\Delta B)\,\mathrm{fl}(\mathbf{v}_{k-2}) = P_kB\mathbf{v}_{k-2} + \sum_{|\boldsymbol{\xi}|=1}^{3}\Delta^{\boldsymbol{\xi}}(P_kB\mathbf{v}_{k-2}) \\
&= \mathbf{f}_k^+ + \Delta\mathbf{f}_k^+ \quad \text{with}
\end{aligned}$$

$$\Delta\mathbf{f}_k^+ = \sum_{l=1}^{k-2}\left[\sum_{|\boldsymbol{\xi}|=1}^{k-l+1}\Delta^{\boldsymbol{\xi}}\left[P_kB\left(\prod_{s=k-2}^{\ge l+1}\tilde{R}_s\right)\tilde{Q}_l^T\right]\right]\mathbf{z}_l, \tag{A.2}$$

where $|\Delta B|\le\tau_p|B|$, $|\Delta P_k|\le\tau_p|P_k|$ by Lemma 4.1 and $\Delta^{\xi}\tilde{Q}_l^T$ and $\Delta^{\xi}\tilde{R}_s$ are respectively defined as in (4.5) and (4.6).

Let $\Delta\tilde{A}_{k,l} := \sum_{|\boldsymbol{\xi}|=1}^{k-l+1}\Delta^{\boldsymbol{\xi}}\left[P_kB\left(\prod_{s=k-2}^{\ge l+1}\tilde{R}_s\right)\tilde{Q}_l^T\right]$ for $k>l+1$. It has the following norm relation by setting $\boldsymbol{\xi}=(\xi_k,\xi_{k-1},\ldots,\xi_l)$ and using Lemma 4.3:

$$\|\Delta\tilde{A}_{k,l}\|_{\max} \le \sum_{|\boldsymbol{\xi}|=1}^{k-l+1}\|\Delta^{\xi_k}P_k\|_\infty\|\Delta^{\xi_{k-1}}B\|_{\max}\prod_{s=k-2}^{\ge l+1}\|\Delta^{\xi_s}\tilde{R}_s\|_1\|\Delta^{\xi_l}\tilde{Q}_l^T\|_1. \tag{A.3}$$

To bound the right-hand side of (A.3), we list the norm bounds for the matrices (derived from Assumption 4.5 or given in Lemma 4.7):

$$\|\Delta^0 P_k\|_\infty \le pc_U, \quad \Delta^0 B\|_{\max} \le c_B|A|_{\min}, \quad \|\Delta^0 \tilde{R}_s\|_1 \le c_T, \quad \|\Delta^0 \tilde{Q}_l^T\|_1 \le pc_U,$$
$$\|\Delta^1 P_k\|_\infty \le pc_U\tau_p, \quad \|\Delta^1 B\|_{\max} \le c_B\tau_p|A|_{\min}, \quad \|\Delta^1 \tilde{R}_s\|_1 \le c_T\tau_{3p},$$
$$\|\Delta^1 \tilde{Q}_l^T\|_1 \le pc_U\tau_{3m}.$$

Thus, from (A.3) and Lemma A.1, we obtain

$$\begin{aligned}
|\Delta\tilde{A}_{k,l}| \le \|\Delta\tilde{A}_{k,l}\|_{\max} &\le \sum_{|\boldsymbol{\xi}|=1}^{k-l+1} (pc_U\tau_p^{\xi_k})(c_B|A|_{\min}\tau_p^{\xi_{k-1}}) \left(\prod_{s=k-2}^{\ge l+1} c_T\tau_{3p}^{\xi_s}\right)(pc_U\tau_{3m}^{\xi_l}) \\
&\le \sum_{|\boldsymbol{\xi}|=1}^{k-l+1} \binom{k-l+1}{|\boldsymbol{\xi}|} p^2 c_B c_U^2 c_T^{k-l-2}|A|_{\min}\tau_{3m}^{|\boldsymbol{\xi}|} \\
&\le \frac{4}{3}(k-l+1)\tau_{3m} p^2 c_B c_U^2 c_T^{k-l-2}|A_{k,l}|. \qquad \square
\end{aligned}$$

Proof of Lemma 4.11. Following the bottom-up traversal in (3.10), we obtain the following equation for $1 \le i \le L$ via the recursive relations by noting $\mathbf{z}^{(L+1)} = \hat{\mathbf{z}}$:

$$\mathrm{fl}(\mathbf{z}^{(i)}) = \prod_{j=i}^{L}(V^{(j)} + \Delta V^{(j)})^T\mathbf{z}^{(L+1)} = \left(\prod_{j=i}^{L}(V^{(j)})^T + \sum_{|\boldsymbol{\xi}|=1}^{L-i+1} \Delta^{\boldsymbol{\xi}} \prod_{j=i}^{L}(V^{(j)})^T\right)\mathbf{z}^{(L+1)},$$

where

$$|(\Delta V^{(j)})^T| \le \begin{cases} \tau_m|(V^{(j)})^T|, & \text{if } j = L, \\ \tau_{2p}|(V^{(j)})^T|, & \text{if } 1 \le j < L, \end{cases} \tag{A.4}$$

by Lemma 4.1. Hence, $|(\Delta V^{(j)})^T| \le \tau|(V^{(j)})^T|$, with $\tau = \max\{\tau_m, \tau_{2p}\}$, for all j.

The coefficient for the case $j = L$ in (A.4) is τ_m because $(V^{(L)})^T$ is a block diagonal matrix with the blocks $\{\mathcal{V}_i^T\}$ defined in (3.9) and each $\mathcal{V}_i^T$ is a $p \times m$ matrix by Assumption 4.5. Based on this, we also obtain

$$\|(V^{(L)})^T\|_1 \le pc_U \quad \text{and} \quad \|(\Delta V^{(L)})^T\|_1 \le pc_U\tau_m.$$

For the cases when $1 \le j < L$ in (A.4), the coefficient would be τ_{2p} since $(V^{(j)})^T$ is a block diagonal matrix with $p \times 2p$ blocks $\begin{pmatrix}\mathcal{W}_{c_1}^T & \mathcal{W}_{c_2}^T\end{pmatrix}$. Accordingly, for $1 \le j < L$,

$$\|(V^{(j)})^T\|_1 \le c_T \quad \text{and} \quad \|(\Delta V^{(j)})^T\|_1 \le c_T\tau_{2p}.$$

Thus, by Assumption 4.5, for $1 \le i \le L$,

$$\left\|\Delta^{\boldsymbol{\xi}} \prod_{j=i}^{L}(V^{(j)})^T\right\|_1 \le \tau_{2m}^{|\boldsymbol{\xi}|} \prod_{j=i}^{L} \left\|(V^{(j)})^T\right\|_1 \le pc_U c_T^{L-i}\tau_{2m}^{|\boldsymbol{\xi}|}.$$

For $\mathrm{fl}(\mathbf{y}^{(i)})$ obtained via recursive formulae in (3.10), we have

$$\begin{aligned}
\mathrm{fl}(\mathbf{y}^{(i)}) &= (B^{(i)} + \Delta B^{(i)})\left(\prod_{j=i}^{L}(V^{(j)})^T + \sum_{|\boldsymbol{\xi}|=1}^{L-i+1} \Delta^{\boldsymbol{\xi}} \prod_{j=i}^{L}(V^{(j)})^T\right)\mathbf{z}^{(L+1)} \\
&= \mathbf{y}^{(i)} + \left(\sum_{|\boldsymbol{\xi}|=1}^{L-i+2} \Delta^{\boldsymbol{\xi}}\left(B^{(i)}\prod_{j=i}^{L}(V^{(j)})^T\right)\right)\mathbf{z}^{(L+1)},
\end{aligned}$$

with $|\Delta B^{(i)}| \leq \tau_p |B^{(i)}|$ for $1 \leq i \leq L$. ∎

Proof of Lemma 4.12. Following the top-down traversal in (3.11), the following expansion of $\mathrm{fl}(\mathbf{f}^{(i)})$ holds for $1 \leq i \leq L-1$:

$$\begin{aligned}\mathrm{fl}(\mathbf{f}^{(i)}) &= (I+\Delta Z^{(i)})((U^{(i)}+\Delta U^{(i)})\,\mathrm{fl}(\mathbf{f}^{(i-1)})+\mathrm{fl}(\mathbf{y}^{(i+1)}))\\ &= \sum_{k=1}^{i+1}(I+\Delta Z^{(i)})\left(\prod_{j=i}^{\geq k}\left[(U^{(j)}+\Delta U^{(j)})(I+\Delta Z^{(j-1)})\right]\right)\mathrm{fl}(\mathbf{y}^{(k)}),\end{aligned}$$

where $|\Delta Z^{(j)}| \leq \epsilon_{\mathrm{mach}} I$, and $\Delta U^{(j)}$ is a block diagonal matrix with block size $2p \times p$ that satisfies $|\Delta U^{(j)}| \leq \tau_p |U^{(j)}|$ according to Lemma 4.1. Hence, for $1 \leq j \leq L-1$,

$$\|U^{(j)}\|_\infty \leq c_T, \quad \|\Delta U^{(j)}\|_\infty \leq c_T \tau_p. \tag{A.5}$$

By multiplying $U^{(L)}$ with $\mathrm{fl}(\mathbf{f}^{(L-1)})$ in the evaluation stage, we have

$$\begin{aligned}\mathrm{fl}(\hat{A}_{\mathbf{L}}\mathbf{z}^{(L+1)}) &= (U^{(L)}+\Delta U^{(L)})\,\mathrm{fl}(\mathbf{f}^{(L-1)})\\ &= \sum_{k=1}^{L}\left(\prod_{j=L}^{\geq k}\left[(U^{(j)}+\Delta U^{(j)})(I+\Delta Z^{(j-1)})\right]\right)\mathrm{fl}(\mathbf{y}^{(k)})\\ &= \sum_{k=1}^{L}\left(\prod_{j=L}^{\geq k}U^{(j)}+\sum_{|\boldsymbol{\xi}|=1}^{L-k+1}\Delta^{\boldsymbol{\xi}}\left(\prod_{j=L}^{\geq k}\tilde{U}^{(j)}\right)\right)\mathrm{fl}(\mathbf{y}^{(k)}),\end{aligned} \tag{A.6}$$

where

$$\Delta^0\tilde{U}^{(j)} = U^{(j)}, \quad \Delta^1\tilde{U}^{(j)} = \Delta U^{(j)} + U^{(j)}\Delta Z^{(j-1)} + \Delta U^{(j)}\Delta Z^{(j-1)},$$

and $\Delta U^{(L)}$ is a block diagonal matrix with block size $m \times p$ that satisfies $|\Delta U^{(L)}| \leq \tau_p |U^{(L)}|$. Thus,

$$\|\Delta\tilde{U}^{(j)}\|_\infty \leq (\tau_p + \epsilon_{\mathrm{mach}} + \epsilon_{\mathrm{mach}}\tau_p)\|U^{(j)}\|_\infty \leq \tau_{3p}\|U^{(j)}\|_\infty \leq c_T\tau_{3p}, \tag{A.7}$$

and by Assumption 4.5,

$$\|U^{(L)}\|_\infty \leq p c_U \quad \text{and} \quad \|\Delta U^{(L)}\|_\infty \leq p c_U \tau_p. \tag{A.8}$$

Moreover, if we combine (A.5), (A.8) together with (A.7), we get

$$\left\|\Delta^{\boldsymbol{\xi}}\prod_{j=L}^{\geq k}\tilde{U}^{(j)}\right\|_\infty \leq \tau_{3p}^{|\boldsymbol{\xi}|}\prod_{j=L}^{\geq k}\left\|\tilde{U}^{(j)}\right\|_\infty \leq p c_U c_T^{L-k}\tau_{3p}^{|\boldsymbol{\xi}|}, \quad \text{for } 1 \leq k \leq L. \tag{A.9}$$

Next, we discuss the perturbation $\Delta\hat{A}_{\mathbf{L}}$. If we plug (4.14) into (A.6), we obtain $\mathrm{fl}(\hat{A}_{\mathbf{L}}\mathbf{z}^{(L+1)}) = \hat{A}_{\mathbf{L}}\mathbf{z}^{(L+1)} + \Delta\hat{A}_{\mathbf{L}}\mathbf{z}^{(L+1)}$ with $\Delta\hat{A}_{\mathbf{L}}$ defined in (4.15). To analyze $|\Delta\hat{A}_{\mathbf{L}}|$, we observe that the nonzero patterns of $\Delta\hat{A}_{\mathbf{L}}^{(k)}$ defined in (4.16) do not overlap for distinct k. Then

$$|\Delta\hat{A}_{\mathbf{L}}| \leq \|\Delta\hat{A}_{\mathbf{L}}\|_{\max} \leq \max_{1\leq k\leq L}\|\Delta\hat{A}_{\mathbf{L}}^{(k)}\|_{\max}. \tag{A.10}$$

Hence, it suffices to find an upper bound for $\|\Delta \hat{A}_{\mathbf{L}}^{(k)}\|_{\max}$, for each k.

Let $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \boldsymbol{\xi}_3)$. Based on (4.16), we use norm bounds (A.9) and Lemma 4.11 to obtain

$$\begin{aligned}
\left\|\Delta \hat{A}_{\mathbf{L}}^{(k)}\right\|_{\max} &\leq \sum_{|\boldsymbol{\xi}|=1}^{2L-2k+3} \left\|\Delta^{\boldsymbol{\xi}_1} \prod_{j=L}^{\geq k} \tilde{U}^{(j)}\right\|_{\infty} \left\|\Delta^{\boldsymbol{\xi}_2} B^{(k)}\right\|_{\max} \left\|\Delta^{\boldsymbol{\xi}_3} \prod_{j=k}^{L} (V^{(j)})^T\right\|_1 \\
&\leq \sum_{|\boldsymbol{\xi}|=1}^{2L-2k+3} p c_U c_T^{L-k} \tau_{3p}^{|\boldsymbol{\xi}_1|} c_B |A|_{\min} \tau_p^{|\boldsymbol{\xi}_2|} p c_U c_T^{L-k} \tau_{2m}^{|\boldsymbol{\xi}_3|} \\
&\leq p^2 c_B c_U^2 c_T^{2L-2k} |A|_{\min} \sum_{|\boldsymbol{\xi}|=1}^{2L-2k+3} \binom{2L-2k+3}{|\boldsymbol{\xi}|} \tau_{3m}^{|\boldsymbol{\xi}|} \\
&\leq \frac{4}{3}(2L-2k+3) p^2 c_B c_U^2 c_T^{2L-2k} \tau_{3m} |\hat{A}_{\mathbf{L}}|,
\end{aligned}$$

where the last step is given by Lemma A.1.

By applying (A.10) and Assumption 4.5, we obtain (4.17). ∎