Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Rank-structured approximation of some Cauchy matrices with sublinear complexity[†]

Mikhail Lepilov[1] | Jianlin Xia*[1]

[1]Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA

Correspondence
*Jianlin Xia, Department of Mathematics, Purdue University, West Lafayette, IN 47907, USA. Email: xiaj@purdue.edu

Summary

In this paper, we consider the rank-structured approximation of one important type of Cauchy matrix. This approximation plays a key role in some structured matrix methods such as stable and efficient direct solvers and other algorithms for Toeplitz matrices and certain kernel matrices. Previous rank-structured approximations (specifically hierarchically semiseparable, or HSS, approximations) for such a matrix of size n cost at least $O(n)$ complexity. Here, we show how to construct an HSS approximation with sublinear (specifically, $O(\log^3 n)$) complexity. The main ideas include extensive computation reuse and an analytical far-field compression strategy. Low-rank compression at each hierarchical level is restricted to just a single off-diagonal block row, and a resulting basis matrix is then reused for other off-diagonal block rows as well as off-diagonal block columns. The relationships among the off-diagonal blocks are rigorously analyzed. The far-field compression uses an analytical proxy point method where we optimize the choice of some parameters so as to obtain accurate low-rank approximations. Both the basis reuse ideas and the resulting analytical hierarchical compression scheme can be generalized to some other kernel matrices and are useful for accelerating relevant rank-structured approximations (though not subsequent operations like matrix-vector multiplications).

KEYWORDS:
Cauchy matrix, rank-structured approximation, far-field compression, proxy point method, sublinear complexity, basis reuse

1 | INTRODUCTION

Cauchy matrices frequently arise in numerical computations such as solutions of differential and integral equations, solutions of Toeplitz, Hankel, or Vandermonde systems, kernel matrix methods, and electrostatic potential evaluations. We consider some Cauchy matrices defined by the evaluation of the following Cauchy kernel function at uniform points $x_i, y_j \in \mathbb{C}, i, j = 1, 2, \ldots, n$ on a circle or a straight line:

$$\kappa(x, y) = \frac{1}{x - y}. \tag{1}$$

Specifically, the case with the following points plays an important role in Toeplitz matrix computations:

$$\mathbf{x} \equiv \{x_i\}_{i=1:n} \quad \text{with} \quad x_i = \omega^{2i-2}, \quad \mathbf{y} \equiv \{y_j\}_{j=1:n} \quad \text{with} \quad y_j = \omega^{2j-1}, \tag{2}$$

[†]The research of Jianlin Xia was supported in part by NSF grants DMS-1819166 and DMS-2111007.

where $\omega = e^{\frac{\pi \mathbf{i}}{n}}$, $\mathbf{i} = \sqrt{-1}$, and the Matlab notation $1 : n$ means the natural numbers $1, 2, \ldots, n$. In fact, a Toeplitz matrix in Fourier space or in conjunction with displacement structures (see, e.g., References[9, 14, 16, 22]) is related to the Cauchy matrix

$$C = \kappa(\mathbf{x}, \mathbf{y}) \equiv (\kappa(x_i, y_j))_{n\times n} = \left(\frac{1}{\omega^{2i-2} - \omega^{2j-1}} \right)_{n\times n}, \tag{3}$$

where $\kappa(\mathbf{x}, \mathbf{y})$ or $(\kappa(x_i, y_j))_{n\times n}$ denotes the $n \times n$ matrix with the (i, j) entry $\kappa(x_i, y_j)$.

We are interested in computing a rank-structured approximation to C quickly. One motivation is that rank structures can be used to design efficient and stable direct solvers for general Toeplitz linear systems, as in References[6, 20, 23, 27, 33]. Such solvers rely on the efficient low-rank approximations to the off-diagonal blocks of C. In particular, hierarchically semiseparable (HSS) matrices[5, 31] have been successfully used to design so-called superfast (nearly linear-complexity) direct Toeplitz solvers with guaranteed backward stability[27, 33]. Such methods essentially use HSS approximations of C to construct HSS approximations to Cauchy-like matrices corresponding to Toeplitz matrices in Fourier space. More precisely, for each order-n Toeplitz matrix T, there exist $n \times 2$ matrices G, H and a diagonal matrix D such that[9, 14, 16, 22]

$$\hat{C} = F_n T D F_n^* = \left(\frac{G_i H_j^*}{\omega^{2i-2} - \omega^{2j-1}} \right)_{n\times n},$$

where F_n is the discrete Fourier transform matrix of order n and G_i and H_i are respectively the ith rows of G and H. $\hat{C}$ may be approximated by an HSS form so as to produce fast direct Toeplitz solvers. It has been shown in Theorem 4.1 of Reference[6] that, once an HSS approximation is computed for C so that its off-diagonal blocks are compressed into low-rank forms, low-rank approximations to the corresponding off-diagonal blocks of $\hat{C}$ can be conveniently written out. This saves the majority of the compression cost that is otherwise needed in rank-revealing factorizations or randomized methods previously used in References[6, 20, 27, 33]. (That is, the HSS approximation of C can serve as a precomputation stage for the HSS approximation of $\hat{C}$ and some details may be found in References[6, 20, 33].) The approximation to C can also be used to accelerate HSS constructions in other computations related to Toeplitz matrices such as least squares solution[27] and eigenvalue solution[21, 26, 28].

Another application of the rank-structured approximation to C is to quickly find structured approximations to some discretized matrices like those in Reference[24], where implicitly defined kernel matrices related to C appear after some transformations based on certain displacement equations[32]. The rank-structured approximation method for C can also be immediately extended to some kernel matrices related to point sets with certain geometries like those in References[17, 38]. In these applications, the rank-structured approximation to C essentially helps convert some other matrix structures into rank structures that are preserved under multiplications, additions, factorizations, diagonal scalings, etc.

Thus, the quick approximation of C by an HSS form is very useful for computations with Toeplitz matrices and some kernel matrices. The key operation in constructing this HSS form is the low-rank approximation or compression of relevant off-diagonal blocks. In previous work, the off-diagonal blocks of C are compressed by either algebraic rank-revealing factorizations as in References[3, 6, 20] or randomized methods as in References[27, 33]. The former has a total cost of about $O(n^2)$ flops or more, and the latter costs about $O(n)$ flops (with the use of Fast Fourier Transforms). In each case, the compression is done for each individual off-diagonal block row and column at each hierarchical level, resulting in a total of $O(n)$ compression steps. (This is also the case for other hierarchical construction methods like those based on analytical compression or some geometric point heuristics[3, 8].)

However, C in (3) is highly structured and only depends on n. It has previously been mentioned in References[6, 33] that certain subblocks of C are related. Here, we would like to explore more connections among the off-diagonal blocks of C in order to extensively reuse computations in all compression steps. That is, we intend to reuse compression information across all the off-diagonal block rows at each hierarchical level of the HSS approximation, and furthermore share information between off-diagonal block rows and columns. The feasibility of this is justified through our study of the Cauchy structures. Accordingly, we show that we only need to approximate a single off-diagonal block row at each level, which is sufficient to produce a basis matrix for all the other off-diagonal block rows and columns at the same level. Consequently, the strategy significantly reduces the number of compression steps, from $O(n)$ to $O(\log n)$.

Moreover, analytical methods can be combined with the algebraic strategy above to further reduce the computational cost. Analytical approaches have been effectively used in earlier work to quickly construct hierarchical matrix approximations. Examples related to HSS matrices can be found in References[2, 4]. Here, we aim to use analytical methods to obtain HSS constructions with sublinear complexity. Following the terminology in the fast multipole method (FMM)[10], we split the off-diagonal compression step into a "near-field part" and a "far-field part". To avoid expensive rank-revealing factorizations, we approximate the far-field blocks with an analytical compression strategy called the proxy point method. The method is explicitly given in[19, 36, 37] and is also closely related to the Green quadrature approach in Reference[1]. The general idea is to use some so-called proxy points to quickly obtain degenerate function approximations to pairwise interactions based on the interactions' discretized integral representations. Such proxy point methods can be used to directly provide a basis matrix without actual compression operations. Their feasibility for some kernel matrices related to this work has been justified in Reference[35]. Here, for the Cauchy case in (3) with the kernel function (1), we provide nearly optimal choices of some parameters in the proxy point approximation. These choices are nearly optimal in the sense that the resulting low-rank approximation error is as small as possible for a given numerical rank (or the numerical rank is as small as possible for a given accuracy).

We then give an analytical hierarchical compression scheme to construct an HSS approximation to C. The scheme ensures that the far-field part of an off-diagonal block row at each hierarchical level meets two efficiency requirements. First, it satisfies a separation condition so that the far-field part is numerically low rank. Second, a near-field subblock has a small row size. Accordingly, a compact numerical column basis matrix for the off-diagonal block row can be quickly found. Then the basis reuse idea and the proxy point method are integrated into this scheme to reach sublinear complexity for the HSS construction. Specifically, we can approximate C by an HSS form in $O(\log^3 n)$ flops and with $O(\log^3 n)$ storage when a prespecified accuracy is used for off-diagonal compression. This is a significant reduction from the previous roughly $O(n)$ or $O(n^2)$ counts. An algorithm is given and the sublinear complexity is confirmed both theoretically and numerically.

The study in this paper provides a convenient algebraic method to take advantage of translation invariance and geometric symmetry in rank-structured methods. It also hints at useful directions for exploring different intrinsic structures and analytic properties involved in some other kernel matrices. In addition, our hierarchical structured approximation framework based on analytical far-field compression is useful for producing rank-structured approximations to kernel matrices evaluated at more general data sets. This can be used to design either direct solvers or preconditioners for those kernel matrices.

We would like to point out that this work only aims to make the structured approximation stage (in HSS solvers for problems like Toeplitz systems) highly efficient, and subsequent algorithms like matrix-vector multiplications and linear system solutions using the constructed HSS form would, of course, still require $O(n)$ complexity and $O(n)$ storage. It is possible to make algorithms like ULV-type factorizations[5, 31] of C sublinear complexity as well, which is not considered here. The focus of this work is on removing the performance bottleneck of the HSS construction phase. This work then helps move the bottleneck elsewhere (e.g., to the solution or matrix-vector multiplication phase).

The paper is structured as follows. In Section 2, we detail the application of the proxy point method to the far-field compression for the off-diagonal blocks of C. Section 3 reveals some intrinsic structural relationships among the off-diagonal blocks of C. The analytical hierarchical compression scheme is then presented in Section 4. The algorithm, its analysis, and some tests are given in Section 5. Section 6 further discusses the generalization of these ideas. In the presentation, we use $C|_{\mathbf{s}\times\mathbf{t}}$ to denote a submatrix of C with row and column index sets $\mathbf{s}$ and $\mathbf{t}$, respectively. Also, $C|_{\mathbf{s}}$ and $C|_{\{:\}\times\mathbf{t}}$ mean the submatrices of C corresponding to the rows specified by $\mathbf{s}$ and columns specified by $\mathbf{t}$, respectively.

2 | ANALYTICAL FAR-FIELD COMPRESSION BY THE PROXY POINT METHOD

In this section, we consider the low-rank approximation of the following subblock of C:

$$K = \kappa(\mathbf{s}, \mathbf{t}) \equiv \left(\kappa(x_i, y_j)\right)_{x_i\in\mathbf{s}, y_j\in\mathbf{t}}, \tag{4}$$

where κ is given in (1), and

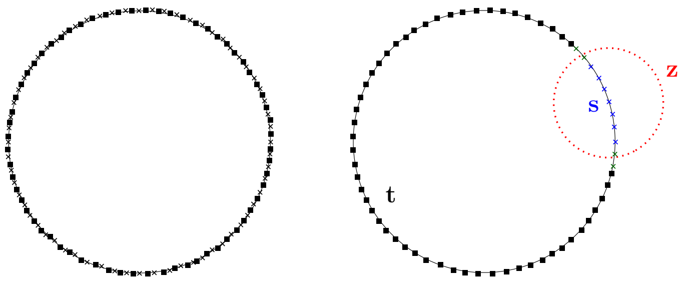
$$\mathbf{s} = \{\omega^{2(i-1)},\ 1 \le i \le k\} \subset \mathbf{x}, \quad \mathbf{t} = \{\omega^{2j-1},\ k+s < j \le n-s-1\} \subset \mathbf{y}, \tag{5}$$
$$1 \le k < \frac{n}{2}, \quad 1 \le s < \frac{n-k}{2}.$$

Here, s is the number of points on each side of $\mathbf{s}$ that separate $\mathbf{s}$ from $\mathbf{t}$. An illustration of the sets $\mathbf{x}, \mathbf{y}, \mathbf{s}, \mathbf{t}$ is given in Figure 1. The reason why such sets are considered will become clear later in this section. For convenience, $\mathbf{s}$ and $\mathbf{t}$ are sometimes referred to as the *source* and *target sets*, respectively, and K is the *interaction* between $\mathbf{s}$ and $\mathbf{t}$.

The two sets $\mathbf{s}$ and $\mathbf{t}$ are well separated in the sense that there exists a point $c \in \mathbb{C}$ such that for every $x \in \mathbf{s}$ and $y \in \mathbf{t}$,

$$\frac{\max_{x\in\mathbf{s}} |c-x|}{\min_{y\in\mathbf{t}} |c-y|} \le \delta < 1, \tag{6}$$

where δ is a constant often referred to as the separation ratio. The point c can be viewed as a center for $\mathbf{s}$. This concept of separation is a basic tool in the FMM and also hierarchical matrix methods[12]. It can be used to show that K is numerically low rank.



(a) Sets $\mathbf{x}, \mathbf{y}$ (b) Sets $\mathbf{s}, \mathbf{t}, \mathbf{z}$

Figure 1 Illustration of the sets under consideration, where each cross (×) is a point in $\mathbf{x}$, each box (■) is a point in $\mathbf{y}$, $\mathbf{s} \subset \mathbf{x}$ is the source set, $\mathbf{t} \subset \mathbf{y}$ is the target set, and $\mathbf{z}$ is a set of proxy points.

We seek to quickly write a (column) basis matrix G in a low-rank approximation to K. This is an essential component in the HSS approximation of C in Section 4. For this purpose, we use the proxy point method in Reference[35] to directly produce G as follows:

$$K \approx GH^T, \quad \text{with} \quad G = \kappa(\mathbf{s}, \mathbf{z}), \tag{7}$$

where $\mathbf{z}$ is a set of points (called *proxy points*) located on a contour (called a proxy surface) that separates $\mathbf{s}$ from $\mathbf{t}$ in $\mathbb{C}$. See Figure 1(b) for an illustration. With G in (7), H is then appropriately decided. (Note that we may also use Taylor expansions to obtain the low-rank approximation like in the standard FMM. However, there are potential stability issues. Stabilization like in Reference[4] may be applied but it needs careful implementations. Here, the proxy point method followed by quick algebraic compression can conveniently produce an accurate low-rank approximation.)

The reader is referred to Reference[35] for more details. In particular, it is shown in Reference[35] that, if $\mathbf{s}$ is located inside a circle with radius γ_1 and center c and $\mathbf{t}$ is outside a circle with radius $\gamma_2 > \gamma_1$ and the same center c, then a nearly optimal choice of proxy points can be obtained by choosing some quadrature points on a circle with radius $\gamma = \sqrt{\gamma_1 \gamma_2}$. With such a choice, the approximation error satisfies[35]

$$\frac{\|K - GH^T\|_F}{\|K\|_F} \le \frac{2}{(\frac{\gamma_2}{\gamma_1})^r - 1} = O\left(\left(\frac{\gamma_1}{\gamma_2}\right)^r\right), \tag{8}$$

where r is the number of points in $\mathbf{z}$. If we take $\gamma_1 = \max_{x \in \mathbf{s}} |c - x|$ and $\gamma_2 = \min_{y \in \mathbf{t}} |c - y|$, then (6) is just $\frac{\gamma_1}{\gamma_2} \le \delta$, so that

$$\frac{\|K - GH^T\|_F}{\|K\|_F} = O(\delta^r). \tag{9}$$

Thus, the approximation error of the proxy point method decreases as $\frac{\gamma_1}{\gamma_2}$ get smaller. We then try to minimize $\frac{\gamma_1}{\gamma_2}$ (by moving the center point c in (6)) so as to minimize the error for given r. To this end, we have the following theorem.

Theorem 1. Consider $\mathbf{s}$ and $\mathbf{t}$ as in (5). Let $g : \mathbb{C} \to \mathbb{R}$ be the function defined by

$$g(c) = \frac{\gamma_1(c)}{\gamma_2(c)}, \quad \text{with} \quad \gamma_1(c) = \max_{x \in \mathbf{s}} |c - x|, \quad \gamma_2(c) = \min_{y \in \mathbf{t}} |c - y|.$$

Then

$$\arg\min(g) = \omega^{k-1},$$

$$\min_{c \in \mathbb{C}} g(c) = \sin \frac{(k-1)\pi}{2n} \Big/ \sin \frac{(k + 2(s+1))\pi}{2n} < \cos \frac{(2s+3)\pi}{2n} < 1. \tag{10}$$

Proof. Since $g(0) = 1$ and $g(\omega^{k-1}) < 1$, we have $\arg\min(g) \neq 0$. Use $\mathrm{Arg}(c)$ to denote the principal argument of c in $[0, 2\pi)$. Let the map $\mathcal{A} : \mathbb{C} \to [0, n)$ be defined by

$$\mathcal{A}(c) = \frac{\mathrm{Arg}(c)}{2\pi} n, \tag{11}$$

which can be thought of as the argument of c scaled to the interval $[0, n)$. Let

$$\Theta_1 = \left\{ c \;\middle|\; \frac{k-1}{2} \le \mathcal{A}(c) < \frac{n}{2},\ c \neq 0 \right\},$$
$$\Theta_2 = \left\{ c \;\middle|\; 0 \le \mathcal{A}(c) < \frac{k-1}{2} \quad \text{or} \quad k-1+\frac{n}{2} \le \mathcal{A}(c) < n,\ c \neq 0 \right\},$$
$$\Omega_1 = \left\{ c \;\middle|\; \frac{k-1}{2} \le \mathcal{A}(c) < k+s,\ c \neq 0 \right\},$$
$$\Omega_2 = \left\{ c \;\middle|\; 0 \le \mathcal{A}(c) < \frac{k-1}{2} \quad \text{or} \quad n-s-1 \le \mathcal{A}(c) < n,\ c \neq 0 \right\}.$$

Figure 2 illustrates these regions. We show

$$\gamma_1(c) = \begin{cases} |1-c|, & c \in \Theta_1, \\ |\omega^{2(k-1)} - c|, & c \in \Theta_2, \\ |\omega^{2k_c} - c|, & \text{otherwise}, \end{cases} \qquad \gamma_2(c) = \begin{cases} |\omega^{2(k+s)+1} - c|, & c \in \Omega_1, \\ |\omega^{2(n-s-1)-1} - c|, & c \in \Omega_2, \\ |\omega^{2j_c} - c|, & \text{otherwise}, \end{cases} \tag{12}$$

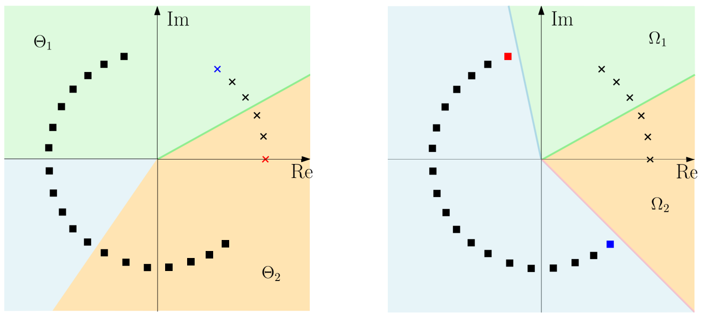
where $k_c = \lfloor \mathcal{A}(c) - \frac{n}{2} \rceil$ and $j_c = \lfloor \mathcal{A}(c) - \frac{1}{2} \rceil + \frac{1}{2}$ (with tie-breaking by rounding up). Let $c = r_0 e^{\mathbf{i}\theta_0}$ for some $r_0 > 0, \theta_0 \in [0, 2\pi)$ and let

$$f(\phi) = |c - e^{\mathbf{i}\phi}|^2 = (r_0\cos(\theta_0) - \cos(\phi))^2 + (r_0\sin(\theta_0) - \sin(\phi))^2 = r_0^2 + 1 - 2r_0\cos(\phi - \theta_0).$$

In order to determine $\gamma_1(c)$ and $\gamma_2(c)$, we just need to find $\max f(\phi)$ and $\min f(\phi)$, respectively. Since $f'(\phi) = 2r_0\sin(\phi - \theta_0)$, $f(\phi)$ has extrema when $\phi = \theta_0 + k\pi$ with an integer k. Now,

$$f(\theta_0 + k\pi) = \begin{cases} (r_0+1)^2, & \text{if } k \text{ is odd}, \\ (r_0-1)^2, & \text{otherwise}. \end{cases}$$

Thus, $f(\phi)$ reaches its maximum, at, say $\theta_0 + \pi$. Note that f is increasing when $\phi - \theta_0 \in [0, \pi]$ and decreasing when $\phi - \theta_0 \in [\pi, 2\pi]$. Hence, $\gamma_1(c) = \max_{x \in \mathbf{s}} |c - x|$ occurs at a point $e^{\mathbf{i}\phi_1} \in \mathbf{s}$ closest to $e^{\mathbf{i}(\theta_0+\pi)}$. Similarly, $f(\phi)$ reaches its minimum at, say, θ_0, so $\gamma_2(c) = \min_{y \in \mathbf{t}} |c - y|$ occurs at a point $e^{\mathbf{i}\phi_2} \in \mathbf{t}$ closest to $e^{\mathbf{i}\theta_0}$. Then, the results in (12) follow immediately for c in different regions.



(a) Regions of definition for γ_1 (b) Regions of definition for γ_2

Figure 2 Illustration of the regions of definition for the piecewise-defined functions γ_1 and γ_2, along with points in the sets $\mathbf{s}$ ($\times$) and $\mathbf{t}$ ($\blacksquare$). On Θ_1 and Θ_2, $\gamma_1(c)$ returns the distance between the input c and the red and blue crosses, respectively. On Ω_1 and Ω_2, $\gamma_2(c)$ returns the distance between c and the red and blue squares, respectively.

Next, we show $g(c) \ge 1$ for any $c \in \mathbb{C} \setminus \{\Omega_1 \cup \Omega_2 \cup \{0\}\}$. Again, write $c = r_0 e^{\mathbf{i}\theta_0}$ with $\theta_0 \in [0, 2\pi)$. With the optimal points $e^{\mathbf{i}\phi_1}$ and $e^{\mathbf{i}\phi_2}$ obtained as above, if we suppose $\phi_1, \phi_2 \in [0, 2\pi)$, then $\frac{2\pi}{n} \le |\phi_1 - \theta_0| \le (n-1)\frac{2\pi}{n}, |\phi_2 - \theta_0| \le \frac{2\pi}{n}$. This yields $\cos(\phi_1 - \theta_0) \le \cos(\phi_2 - \theta_0)$ and then

$$g^2(c) = \frac{f(\phi_1)}{f(\phi_2)} = \frac{r_0^2 + 1 - 2r_0\cos(\phi_1 - \theta_0)}{r_0^2 + 1 - 2r_0\cos(\phi_2 - \theta_0)} \ge 1.$$

Thus,

$$\arg\min(g) \in \Omega_1 \cup \Omega_2.$$

Let $c_0 \in \Omega_1$. If we write $c_0 = \rho\omega^{k-1}e^{\mathbf{i}\theta}$ for some $\rho, \theta \in \mathbb{R}$ with $\rho > 0$, we have $\tilde{c}_0 = \rho\omega^{k-1}e^{-\mathbf{i}\theta} \in \Omega_2 \cup \left\{c \mid \mathcal{A}(c) = \frac{k-1}{2},\ c \neq 0\right\}$. Then $\gamma_1(\tilde{c}_0) = |\omega^{2(k-1)} - \rho\omega^{k-1}e^{-\mathbf{i}\theta}| = |1 - \rho\omega^{-(k-1)}e^{-\mathbf{i}\theta}| = |1 - \bar{c}_0| = |1 - c_0| = \gamma_1(c_0)$. Similarly, we have $\gamma_2(c_0) = \gamma_2(\tilde{c}_0)$. Accordingly,

$$g(c_0) = g(\tilde{c}_0).$$

(See Figure 3(a).) Hence,

$$\min_{c\in\Omega_1\cup\Omega_2} g(c) = \min_{c\in\Omega_1} g(c).$$

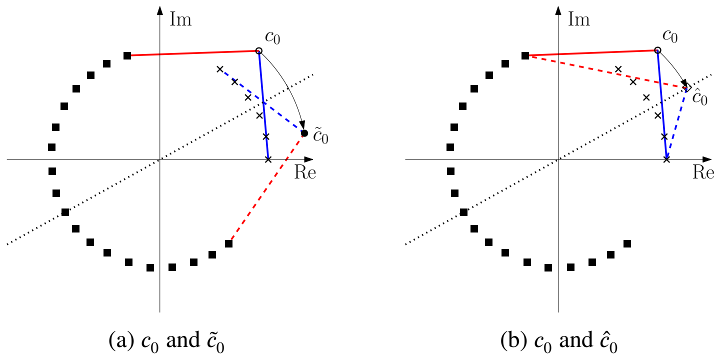


Figure 3 Illustration of the points $c_0 = \rho\omega^{k-1}e^{\mathbf{i}\theta}$, $\tilde{c}_0 = \rho\omega^{k-1}e^{-\mathbf{i}\theta}$, and $\hat{c}_0 = \rho\omega^{k-1}$, along with points in the sets $\mathbf{s}$ (×) and $\mathbf{t}$ (■), where the dotted black line illustrates the relevant symmetry. In (a), $g\left(c_0\right)$ is the length of the solid blue line divided by the length of the solid red line and $g\left(\tilde{c}_0\right)$ is the length of the dashed blue line divided by the length of the dashed red line. In (b), $g\left(\hat{c}_0\right)$ is the length of the dashed blue line divided by the length of the dashed red line.

Then, let $\hat{c}_0 = \rho\omega^{k-1}$ (that satisfies $\mathcal{A}(\hat{c}_0) = \frac{k-1}{2}$). The law of cosines applied to two triangles formed by the origin, the point 1, and either c_0 or $\hat{c}_0$ immediately leads to $\gamma_1(c_0) \geq \gamma_1(\hat{c}_0)$. Similarly, $\gamma_2(c_0) \leq \gamma_2(\hat{c}_0)$ holds. Thus, we have $g(c_0) \geq g(\hat{c}_0)$. Accordingly,

$$\arg\min(g) \in \left\{c \,\middle|\, \mathcal{A}(c) = \frac{k-1}{2},\ c \neq 0\right\}.$$

Then from (12), the positive minima of g occur at the positive minima of the following function:

$$h(\rho) = \frac{\left|1 - \rho\omega^{k-1}\right|}{\left|\omega^{2(k+s)+1} - \rho\omega^{k-1}\right|} = \frac{|1 - \rho e^{\beta\mathbf{i}}|}{|e^{\alpha\mathbf{i}} - \rho e^{\beta\mathbf{i}}|},$$

where $\alpha = \frac{2\pi(k+s)+\pi}{n}$ and $\beta = \frac{\pi(k-1)}{n}$. We look at

$$h^2(\rho) = \frac{(\rho\sin\beta)^2 + (1 - \rho\cos\beta)^2}{(\sin\alpha - \rho\sin\beta)^2 + (\cos\alpha - \rho\cos\beta)^2}.$$

Since the denominator above does not vanish, by the quotient rule, the positive minima of $h^2(\rho)$ occur at the positive zeros of the function

$$\tilde{h}(\rho) = 2(\rho^2 - 1)\left(\cos\beta - \cos\left(\alpha - \beta\right)\right).$$

The only such zero is at $\rho = 1$, so this is the only positive minimum argument of $h^2(\rho)$. Thus, $\arg\min(g) = e^{\frac{\pi(k-1)\mathbf{i}}{n}} = \omega^{k-1}$. With the choice of the center $c = \omega^{k-1}$, we have

$$\gamma_1 = \left|1 - \omega^{k-1}\right| = \sqrt{2(1 - \cos\beta)} = 2\sin\frac{(k-1)\pi}{2n},$$
$$\gamma_2 = \left|\omega^{2(k+s)+1} - \omega^{k-1}\right| = \sqrt{2(1 - \cos(\alpha - \beta)} = 2\sin\frac{(k + 2(s+1))\pi}{2n}.$$

Then $\min_{c\in\mathbb{C}} g(c)$ in (10) is obtained.

Finally, by the definition of the sets **s** and **t**, we have $k < \frac{n}{2}$ and $s+1 \le \frac{n-k}{2}$. Then

$$\frac{(k-1)\pi}{2n} \in \left[0, \frac{\pi}{4}\right), \quad \frac{(k+2(s+1))\pi}{2n} \in \left(0, \frac{\pi}{2}\right).$$

Also, note that $\frac{(k+2(s+1))\pi}{2n} - \frac{(k-1)\pi}{2n} = \frac{(2s+3)\pi}{2n} \in (0, \frac{\pi}{2})$. Then

$$\sin\frac{(k-1)\pi}{2n} = \sin\frac{(k+2(s+1)-(2s+3))\pi}{2n} < \sin\frac{(k+2(s+1))\pi}{2n}\cos\frac{(2s+3)\pi}{2n}.$$

This yields the inequality in (10). □

This theorem shows how to choose the optimal center c in (6) so as to make the error in (8) as small as possible. The optimal choice is actually the "center" of the source subset **s** in terms of the scaled argument. Later for convenience, we sometimes refer to this center as an *argument center*. With the optimal center, we can ensure that **s** and **t** in (5) are well separated so as to obtain δ in (6) as a constant smaller than 1 and independent of n. More specifically, we have the following result.

Corollary 1. For **s** and **t** in (5), suppose $s = \frac{k}{2} = o(n)$. With the optimal c in Theorem 1, we have

$$\frac{\max_{x\in\mathbf{s}}|c-x|}{\min_{y\in\mathbf{t}}|c-y|} \sim \frac{1}{2} \quad \text{as} \quad n\to\infty. \tag{13}$$

Proof. This is true since

$$\sin\frac{(k-1)\pi}{2n} \Big/ \sin\frac{(k+2(s+1))\pi}{2n} \sim \frac{k}{k+2s}.$$

□

Note that the result can be made more general. For convenience, we introduce the following definition.

Definition 1. For subsets $\mathbf{s}\subset\mathbf{x}$ and $\mathbf{t}\subset\mathbf{y}$, with the notation in (11), the argument span of **s** and the argument gap between **s** and **t** are, respectively,

$$\operatorname{span}_{\mathcal{A}}(\mathbf{s}) = \max_{x_1,x_2\in\mathbf{s}}\left(\left|\mathcal{A}(x_1)-\mathcal{A}(x_2)\right| \bmod n\right),$$
$$\operatorname{gap}_{\mathcal{A}}(\mathbf{s},\mathbf{t}) = \min_{x\in\mathbf{s},y\in\mathbf{t}}\left(\left|\mathcal{A}(x)-\mathcal{A}(y)\right| \bmod n\right).$$

In this definition, $\operatorname{span}_{\mathcal{A}}(\mathbf{s})$ and $\operatorname{gap}_{\mathcal{A}}(\mathbf{s},\mathbf{t})$ are given in terms of the scaled argument. For example, for **s** and **t** in (5),

$$\operatorname{span}_{\mathcal{A}}(\mathbf{s}) = k, \quad \operatorname{gap}_{\mathcal{A}}(\mathbf{s},\mathbf{t}) > s.$$

Thus, s controls the argument gap between **s** and **t**. In general, for subsets $\mathbf{s}\subset\mathbf{x}$ and $\mathbf{t}\subset\mathbf{y}$, as long as

$$\operatorname{gap}_{\mathcal{A}}(\mathbf{s},\mathbf{t}) \ge \frac{1}{2}\operatorname{span}_{\mathcal{A}}(\mathbf{s}), \tag{14}$$

then the separation ratio between **s** and **t** is $\frac{1}{2}$ or smaller with center c chosen as above. (With some technicalities, a precise statement can be made and is skipped here.) This will be useful in Section 4 when the points are sparsified.

This discussion indicates that, as long as $\operatorname{gap}_{\mathcal{A}}(\mathbf{s},\mathbf{t})$ is large enough in comparison with $\operatorname{span}_{\mathcal{A}}(\mathbf{s})$, **s** and **t** would be well separated. Then the interaction matrix between **s** and **t** is numerically low rank. Accordingly, with the proxy point method, we can use $r = O(|\log\tau|)$ proxy points to make the error in (9) bounded by any accuracy τ. In fact, for **s** and **t** in (5), following the study in Reference[35], we can select proxy points that are actually uniform quadrature points used in the trapezoidal rule and located on a circle with radius γ, and γ is another parameter with a nearly optimal value $\sqrt{\gamma_1\gamma_2}$. Thus, we choose

$$\gamma = 2\sqrt{\sin\frac{(k-1)\pi}{2n}\sin\frac{(k+2(s+1))\pi}{2n}}.$$

3 | RELATIONSHIPS AMONG OFF-DIAGONAL BLOCKS OF C

Another key idea in our sublinear complexity rank-structured approximation of C is to fully explore the relationships among the off-diagonal blocks of C. For the sake of convenience, we suppose n is a power of 2, so that C can be hierarchically partitioned up to $L = O(\log_2 n)$ times with uniform block sizes at each level. That is, at level $l = 0, 1, \ldots, L$, C is partitioned evenly into

2^l block rows, each with row size $\frac{n}{2^l}$. C is also partitioned into block columns in a similar way. The hierarchical partitioning produces a binary tree T that helps organize the approximation process. We explore the relationships among the off-diagonal block rows and columns at each hierarchical level l so as to save low-rank compression cost in the rank-structured approximation.

Let a node i of T correspond to the kth block row at that level. That is, i corresponds to the subblock of C with index set

$$\mathbf{I}_i = \left\{(k-1)\frac{n}{2^l}+1, (k-1)\frac{n}{2^l}+2, \ldots, k\frac{n}{2^l}\right\}. \tag{15}$$

Following the notation in Reference[29], the *HSS block row* and *column* corresponding to node i are, respectively,

$$C_i^- = C|_{\mathbf{I}_i\times(\mathbf{N}\backslash\mathbf{I}_i)}, \quad C_i^| = C|_{(\mathbf{N}\backslash\mathbf{I}_i)\times\mathbf{I}_i} \tag{16}$$

where $\mathbf{N} = \{1 : n\}$. Suppose j is the node at level l of T that corresponds to the $(k+1)$st block row at that level and has row index set

$$\mathbf{I}_j = \left\{k\frac{n}{2^l}+1, k\frac{n}{2^l}+2, \ldots, (k+1)\frac{n}{2^l}\right\}. \tag{17}$$

Denote the HSS block row corresponding to node j by C_j^-. In Reference[6], it is pointed out that the subblock $C|_{\mathbf{I}_i\times\mathbf{I}_j}$ of C_i^- and the subblock $C|_{\mathbf{I}_j\times((k+1)\frac{n}{2^l}+1:(k+2)\frac{n}{2^l})}$ of C_j^- differ by just the constant scalar $\omega^{-\frac{n}{2^l}}$. This follows directly from the explicit definition of C. Thus, if a low-rank approximation is obtained for $C|_{\mathbf{I}_i\times\mathbf{I}_j}$, it can be reused for $C|_{\mathbf{I}_j\times((k+1)\frac{n}{2^l}+1:(k+2)\frac{n}{2^l})}$.

Here, we would like to systematically generalize such a strategy and also give an intuitive justification. We first look at the block rows at the same level and then the block columns. The following lemma directly follows from the structure of C and provides a convenient tool to study the relationships among the HSS blocks of C.

Lemma 1. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be circulant matrices with the first row

$$\begin{pmatrix} \frac{1}{1-\omega} & \frac{1}{1-\omega^3} & \cdots & \frac{1}{1-\omega^{2n-1}} \end{pmatrix}, \quad \begin{pmatrix} \frac{1}{1-\omega} & \frac{1}{\omega^{2n-2}-\omega} & \cdots & \frac{1}{\omega^2-\omega} \end{pmatrix},$$

respectively. Then

$$C = \Lambda\mathcal{C}_1 = \mathcal{C}_2\Lambda,$$

where $\Lambda = \operatorname{diag}(1, \omega^{-2}, \ldots, \omega^{-2(n-1)})$.

The relationship among the HSS block rows/columns at each level can be shown as follows.

Proposition 1. Let i and j be any two nodes at the same level of T corresponding to HSS block rows C_i^- and C_j^-, respectively. Then there exist a nonzero scalar μ and a permutation matrix P such that

$$C_i^- = \mu C_j^- P, \quad C_i^| = \mu P^T C_j^|. \tag{18}$$

Proof. For the first equality in (18), it suffices to show the result for the two nearby HSS blocks C_i^- and C_j^- corresponding to the row index sets $\mathbf{I}_i$ in (15) and $\mathbf{I}_j$ in (17), respectively. Once this is shown, the result can then be immediately extended to all the HSS blocks at the same level l. Following the notation in (15) and (17), we may consider the sets

$$\mathbf{N}\backslash\mathbf{I}_i = \mathbf{J}_{i,1}\cup\mathbf{J}_{i,2}\cup\mathbf{J}_{i,3}, \quad \mathbf{N}\backslash\mathbf{I}_j = \mathbf{J}_{j,1}\cup\mathbf{J}_{j,2}\cup\mathbf{J}_{j,3},$$

where

$$\mathbf{J}_{i,1} = \left\{1 : (k-1)\frac{n}{2^l}\right\}, \quad \mathbf{J}_{i,2} = \left\{k\frac{n}{2^l}+1 : n-\frac{n}{2^l}\right\}, \quad \mathbf{J}_{i,3} = \left\{n-\frac{n}{2^l}+1 : n\right\},$$
$$\mathbf{J}_{j,1} = \left\{1 : \frac{n}{2^l}\right\}, \quad \mathbf{J}_{j,2} = \left\{\frac{n}{2^l}+1 : k\frac{n}{2^l}\right\}, \quad \mathbf{J}_{j,3} = \left\{k\frac{n}{2^l}+1 : n\right\}.$$

The partitions are illustrated in Figure 4. According to Lemma 1,

$$C_i^- = \begin{pmatrix} C|_{\mathbf{I}_i\times\mathbf{J}_{i,1}} & C|_{\mathbf{I}_i\times\mathbf{J}_{i,2}} & C|_{\mathbf{I}_i\times\mathbf{J}_{i,3}} \end{pmatrix} = \Lambda|_{\mathbf{I}_i\times\mathbf{I}_i}\begin{pmatrix} \mathcal{C}_1|_{\mathbf{I}_i\times\mathbf{J}_{i,1}} & \mathcal{C}_1|_{\mathbf{I}_i\times\mathbf{J}_{i,2}} & \mathcal{C}_1|_{\mathbf{I}_i\times\mathbf{J}_{i,3}} \end{pmatrix}, \tag{19}$$

$$C_j^- = \begin{pmatrix} C|_{\mathbf{I}_j\times\mathbf{J}_{j,1}} & C|_{\mathbf{I}_j\times\mathbf{J}_{j,2}} & C|_{\mathbf{I}_j\times\mathbf{J}_{j,3}} \end{pmatrix} = \Lambda|_{\mathbf{I}_j\times\mathbf{I}_j}\begin{pmatrix} \mathcal{C}_1|_{\mathbf{I}_j\times\mathbf{J}_{j,1}} & \mathcal{C}_1|_{\mathbf{I}_j\times\mathbf{J}_{j,2}} & \mathcal{C}_1|_{\mathbf{I}_j\times\mathbf{J}_{j,3}} \end{pmatrix}. \tag{20}$$

Since $\Lambda|_{\mathbf{I}_i\times\mathbf{I}_i} = \operatorname{diag}(\omega^{-2(k-1)\frac{n}{2^l}}, \ldots, \omega^{-2(k\frac{n}{2^l}-1)})$, $\Lambda|_{\mathbf{I}_j\times\mathbf{I}_j} = \operatorname{diag}(\omega^{-2k\frac{n}{2^l}}, \ldots, \omega^{-2((k+1)\frac{n}{2^l}-1)})$, we have $\Lambda|_{\mathbf{I}_i\times\mathbf{I}_i} = \mu\Lambda|_{\mathbf{I}_j\times\mathbf{I}_j}$ with $\mu = \omega^{\frac{n}{2^{l-1}}}$. Also, the circulant structure of $\mathcal{C}_1$ means

$$\mathcal{C}_1|_{\mathbf{I}_i\times\mathbf{J}_{i,1}} = \mathcal{C}_1|_{\mathbf{I}_j\times\mathbf{J}_{j,2}}, \quad \mathcal{C}_1|_{\mathbf{I}_i\times\mathbf{J}_{i,2}} = \mathcal{C}_1|_{\mathbf{I}_j\times\mathbf{J}_{j,3}}, \quad \mathcal{C}_1|_{\mathbf{I}_i\times\mathbf{J}_{i,3}} = \mathcal{C}_1|_{\mathbf{I}_j\times\mathbf{J}_{j,1}}.$$

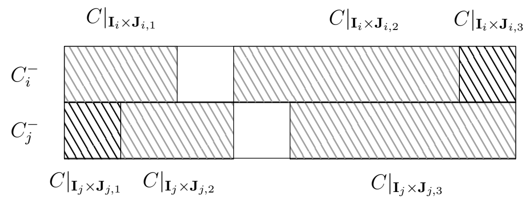

$C|_{\mathbf{I}_i\times\mathbf{J}_{i,1}}$
$C|_{\mathbf{I}_i\times\mathbf{J}_{i,2}}$
$C|_{\mathbf{I}_i\times\mathbf{J}_{i,3}}$
C_i^-
C_j^-
$C|_{\mathbf{I}_j\times\mathbf{J}_{j,1}}$
$C|_{\mathbf{I}_j\times\mathbf{J}_{j,2}}$
$C|_{\mathbf{I}_j\times\mathbf{J}_{j,3}}$

Figure 4 Partitioning of C_i^- and C_j^-.

This is because of the same shift amount in both the row and the column indices. For example, for the first equality above, the shifts from $\mathbf{I}_i$ to $\mathbf{I}_j$ and from $\mathbf{I}_{i,1}$ to $\mathbf{I}_{j,2}$ are both $\frac{n}{2^l}$. Thus,

$$\left(C_1|_{\mathbf{I}_i\times\mathbf{J}_{i,1}} \; C_1|_{\mathbf{I}_i\times\mathbf{J}_{i,2}} \; C_1|_{\mathbf{I}_i\times\mathbf{J}_{i,3}} \right) = \left(C_1|_{\mathbf{I}_j\times\mathbf{J}_{j,2}} \; C_1|_{\mathbf{I}_j\times\mathbf{J}_{j,3}} \; C_1|_{\mathbf{I}_j\times\mathbf{J}_{j,1}} \right) = \left(C_1|_{\mathbf{I}_j\times\mathbf{J}_{j,1}} \; C_1|_{\mathbf{I}_j\times\mathbf{J}_{j,2}} \; C_1|_{\mathbf{I}_j\times\mathbf{J}_{j,3}} \right) P,$$

where P is the permutation matrix that brings $C_1|_{\mathbf{I}_j\times\mathbf{J}_{j,1}}$ from the beginning to the end of the first matrix on the far right-hand side above. (19)–(20) then lead to

$$C_i^- = \mu_1 \Lambda|_{\mathbf{I}_j\times\mathbf{I}_j} \left(C_1|_{\mathbf{I}_j\times\mathbf{J}_{j,1}} \; C_1|_{\mathbf{I}_j\times\mathbf{J}_{j,2}} \; C_1|_{\mathbf{I}_j\times\mathbf{J}_{j,3}} \right) P = \mu_1 C_j^- P.$$

Next, for the second equality in (18), we may consider the HSS block rows of C^T. According to Lemma 1, $C^T = \Lambda C_2^T$. Since C_2^T is still a circulant matrix and it is scaled on the left by the same diagonal matrix Λ, the result can then be shown in the same way as above to get the same μ and P. □

This proposition shows that the HSS block rows (columns) are related by scalar multiples and column (row) permutations. Moreover, we can further relate the HSS block rows to the HSS block columns due to the following proposition.

Proposition 2. Let i be any node at level l of T corresponding to the HSS block row C_i^- and HSS block column $C_i^|$ as in (16). Let m be the row size of $C_i^|$. Then there exists a permutation matrix P such that

$$(P^T C_i^|)|_{\{2:m\}} = -\frac{1}{\omega}\left((C_i^- P)|_{\{:\}\times\{1:m-1\}}\right)^T.$$

Proof. Let i_1 be the leftmost node at the same level l as i. According to Proposition 1, there exist a scalar μ and a permutation matrix P such that

$$C_{i_1}^- = \mu C_i^- P, \quad C_{i_1}^| = \mu P^T C_i^|. \tag{21}$$

Now for $1 \le j < k \le n$, since

$$C_{k,j} = \frac{1}{\omega^{2k-2} - \omega^{2j-1}} = -\frac{1}{\omega}\frac{1}{\omega^{2j-2} - \omega^{2k-3}} = -\frac{1}{\omega} C_{j,k-1},$$

we have

$$C_{i_1}^||_{\{2:m\}} = -\frac{1}{\omega}(C_{i_1}^-|_{\{:\}\times\{1:m-1\}})^T. \tag{22}$$

(See Figure 5 for an illustration.) Then (21) and (22) together yield the result. □

The implication of Proposition 1 is that, once we obtain a low-rank approximation to one HSS block row (or column) at level l, we can reuse its column (row) basis matrix for all the other HSS block rows (or columns) at level l. Proposition 2 further means that the HSS block column $C_i^|$ and row C_i^- are also closely related. With the exception of one row in $C_i^|$ and one column in C_i^-, the remaining subblocks can also share basis information. In the next section, we will take advantage of these relations to design our sublinear complexity HSS approximation for C.

4 | ANALYTICAL HIERARCHICAL COMPRESSION SCHEME WITH SUBLINEAR COMPLEXITY

In this section, we showcase our sublinear complexity HSS approximation of C through a combination of multiple techniques. Again, we assume the method of partitioning in the beginning of Section 3. The binary tree T produced in the hierarchical

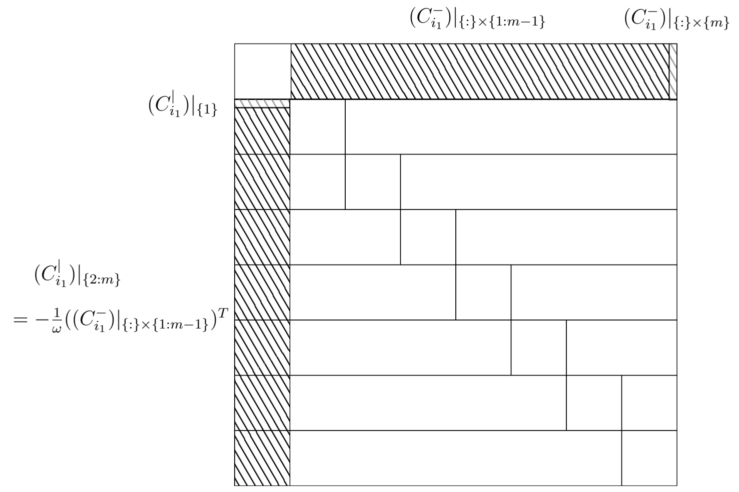

$(C^{-}_{i_1})|_{\{:\}\times\{1:m-1\}}$
$(C^{-}_{i_1})|_{\{:\}\times\{m\}}$
$(C^{|}_{i_1})|_{\{1\}}$
$(C^{|}_{i_1})|_{\{2:m\}}$
$= -\frac{1}{\omega}((C^{-}_{i_1})|_{\{:\}\times\{1:m-1\}})^T$

Figure 5 The relationship between $C^{|}_{i_1}|_{\{2:m\}}$ and $C^{-}_{i_1}|_{\{:\}\times\{1:m-1\}}$, when i_1 is the leftmost node at level l of T.

partitioning of C can essentially serve as a tree (called HSS tree) for representing the resulting HSS approximation, which is recursively defined by[5, 31]

$$D_k = \begin{pmatrix} D_i & U_i B_i V_j^T \\ U_j B_j V_i^T & D_j \end{pmatrix},$$

where k is a node in T with children i and j, and U_i, U_j, V_i, V_j are off-diagonal basis matrices. These last matrices are also defined recursively by

$$U_k = \begin{pmatrix} U_i R_i \\ U_j R_j \end{pmatrix}, \quad V_k = \begin{pmatrix} V_i W_i \\ V_j W_j \end{pmatrix}.$$

Only the leaf-level D, U, and V matrices are stored, along with the R and W matrices of every level except the root level. If k is the root of T, then D_k is just the entire HSS matrix. This gives a generator representation of the HSS form, where the D, U, V, R, W matrices are said to be the HSS generators. Each block row or column without the diagonal block is an HSS block as in (16).

During the construction of the HSS approximation for C, each HSS block row and column are compressed so as to find the generators. In all existing HSS approximation methods, the HSS blocks are compressed individually, leading to the complexity of at least $O(n)$. In the following subsections, we show how to construct an HSS approximation to C in sublinear complexity. The main ingredients include the following.

- Extract a row basis matrix for only one HSS block row at each hierarchical level l via the proxy point compression for its far-field subblock.
- Ensure a near-field subblock of that HSS block row has small row size regardless of the level l.
- Share the row basis matrix across the HSS block rows at the entire level and also extend to the HSS block columns.

We recall our assumption on n that allows us to partition C hierarchically with L levels of block rows, so that the associated HSS tree T is a perfect binary tree. Also, corresponding to a leaf node $i \in T$, each HSS block row $C_i^- = \kappa(\mathbf{x}_i, \mathbf{y}_i)$ is the interaction between subsets $\mathbf{x}_i \subset \mathbf{x}$ and $\mathbf{y}_i \subset \mathbf{y}$ of sizes $\frac{n}{2^L}$ and $n - \frac{n}{2^L}$, respectively. For example, if $i = 1$ which corresponds to the topmost HSS block row at the leaf level, we have $\mathbf{x}_i = \{\omega^{2k-2} \mid 1 \le k \le \frac{n}{2^L}\}$ and $\mathbf{y}_i = \{\omega^{2k-1} \mid \frac{n}{2^L} + 1 \le k \le n\}$.

4.1 | Compression at the leaf level

For a node i at the leaf level or level L, first consider the case where i corresponds to the topmost HSS block row, i.e., $i = 1$. Partition $\mathbf{x}_i$ into

$$\mathbf{x}_i = \mathbf{s}_i \cup \bar{\mathbf{s}}_i \quad \text{with} \tag{23}$$

$$\mathbf{s}_i = \left\{\omega^{2k-2} \mid \frac{n}{2^{L+2}} \le k < \frac{3n}{2^{L+2}}\right\}, \quad \bar{\mathbf{s}}_i = \mathbf{x}_i \setminus \mathbf{s}_i.$$

Since we are considering the interaction between $\mathbf{x}_i$ and $\mathbf{y}_i$, we may consider $\mathbf{s}_i$ as the "far-field" subset of $\mathbf{x}_i$ and $\bar{\mathbf{s}}_i$ the "near-field" subset of $\mathbf{x}_i$. (The reason why we split the source set $\mathbf{x}_i$ instead of the target set $\mathbf{y}_i$ will be explained later in Remark 1.) See Figure 6(a) for an illustration and see Figure 7 for the partitioning of the corresponding HSS block C_i^-. Since it is the leaf level, the sizes of both $\mathbf{s}_i$ and $\bar{\mathbf{s}}_i$ are $\frac{n}{2^{L+1}}$ and are typically set to be small multiples of a desired numerical rank r.

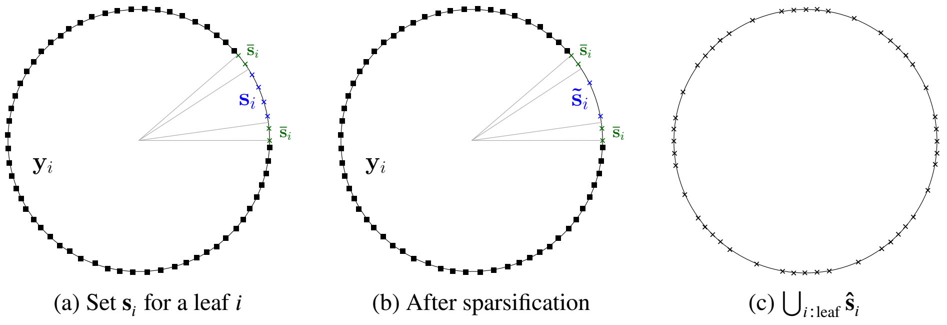


Figure 6 Illustration of the sets under consideration, the sparsification of $\mathbf{s}_i$ into $\tilde{\mathbf{s}}_i$ for a leaf i, and the resulting source point subsets of $\mathbf{x}$ after one level of sparsification.

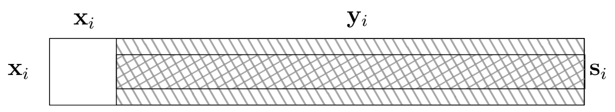


Figure 7 Partitioning of the HSS block C_i^- corresponding to Figure 6(a).

Note that the partitioning in (23) essentially makes $\operatorname{span}_{\mathcal{A}}(\mathbf{s}_i)$ to be at most half of $\operatorname{span}_{\mathcal{A}}(\mathbf{x}_i)$. $\bar{\mathbf{s}}_i$ has two pieces, each has argument span about $1/4$ of $\operatorname{span}_{\mathcal{A}}(\mathbf{x}_i)$. It can be verified that, if we set $\mathbf{s} = \mathbf{s}_i$ and $\mathbf{t} = \mathbf{y}_i$ in Definition 1, then (14) holds and $\mathbf{s}$ and $\mathbf{t}$ are well separated. Thus, the analytical compression strategy in Section 2 is applicable to the far-field interaction $C_{i,2}^- \equiv \kappa(\mathbf{s}_i, \mathbf{y}_i)$. For convenience, let $P^{(L)}$ be a row permutation matrix such that

$$C_i^- = P^{(L)} \begin{pmatrix} C_{i,1}^- \\ C_{i,2}^- \end{pmatrix} \equiv \begin{pmatrix} \kappa(\bar{\mathbf{s}}_i, \mathbf{y}_i) \\ \kappa(\mathbf{s}_i, \mathbf{y}_i) \end{pmatrix}.$$

To apply the proxy point method in Section 2, we follow the choice of the optimal center in Theorem 1. As discussed after Theorem 1, the optimal center c is the argument center of the source subset $\mathbf{s}_i$, which is also the argument center of $\mathbf{x}_i$:

$$c = \omega^{\frac{n}{2^L}-1}.$$

Following the proof of Theorem 1, we can then choose a set $\mathbf{z}_i$ of $r = \mathcal{O}(|\log \tau|)$ proxy points on the circle with center c and radius

$$\gamma = \sqrt{|1-c| \cdot |\omega^{\frac{n}{2^{L+1}}} - c|}.$$

The proxy point method then yields the following approximation with relative accuracy τ (see (7)):

$$C_{i,2}^- \approx G_i H_i^T, \quad \text{with} \quad G_i = \kappa(\mathbf{s}_i, \mathbf{z}_i). \tag{24}$$

Now, apply a rank-revealing factorization to G_i to get a row permutation matrix $\Pi^{(L)}$, which leads to a factorization of the form

$$G_i \approx \Pi^{(L)} \begin{pmatrix} I \\ E^{(L)} \end{pmatrix} G_i|_{\tilde{\mathbf{I}}_i}, \tag{25}$$

where $G_i|_{\tilde{\mathbf{I}}_i}$ consists of r selected rows of G_i corresponding to a row index set $\tilde{\mathbf{I}}_i$. Strong rank-revealing factorizations like the one in Reference[11] may be used so that $E^{(L)}$ has entries satisfying certain bounds to ensure the reliability. (In practice, simpler variations like row pivoted QR factorizations often work reasonably well and have better efficiency.) The factorization (25) is also called an interpolative decomposition[18] or structure preserving rank-revealing (SPRR) factorization[33]. As mentioned in Reference[33], this results in the approximation

$$C_{i,2}^{-}(=\kappa(\mathbf{s}_i,\mathbf{y}_i))\approx\Pi^{(L)}\begin{pmatrix}I\\E^{(L)}\end{pmatrix}G_i|_{\tilde{\mathbf{I}}_i}H_i^T\approx\Pi^{(L)}\begin{pmatrix}I\\E^{(L)}\end{pmatrix}C_{i,2}^{-}|_{\tilde{\mathbf{I}}_i}. \tag{26}$$

Thus, $C_{i,2}^{-}|_{\tilde{\mathbf{I}}_i}$ serves as an approximate row basis matrix for $C_{i,2}^{-}$. The procedure of using the proxy point method and the SPRR factorization to get (24)–(26) is also called a hybrid compression in Reference[35]. We pause to note that we have $C_{i,2}^{-}|_{\tilde{\mathbf{I}}_i}=\kappa(\tilde{\mathbf{s}}_i,\mathbf{t}_i)$ for a subset $\tilde{\mathbf{s}}_i$ of $\mathbf{s}_i$. That is, (26) sparsifies $\mathbf{s}_i$ into $\tilde{\mathbf{s}}_i$. In such circumstances, $\tilde{\mathbf{s}}_i$ is sometimes called a skeleton[7, 15, 25] or representative set[30].

Accordingly, we have

$$C_i^{-}\approx P^{(L)}\begin{pmatrix}I&\\&\Pi^{(L)}\begin{pmatrix}I\\E^{(L)}\end{pmatrix}\end{pmatrix}\begin{pmatrix}C_{i,1}^{-}\\C_{i,2}^{-}|_{\tilde{\mathbf{I}}_i}\end{pmatrix}. \tag{27}$$

(Here, we abuse notation and use I to denote identity matrices of different sizes.) For convenience, let $\bar{\mathbf{I}}_i$ be the row index of $C_{i,1}^{-}$ in C_i^{-} and let $\hat{\mathbf{I}}^{(L)}=\bar{\mathbf{I}}_i\cup\tilde{\mathbf{I}}_i$. Then we may rewrite (27) as

$$C_i^{-}\approx U^{(L)}C_i^{-}|_{\hat{\mathbf{I}}^{(L)}}\quad\text{with}\quad U^{(L)}=\hat{P}^{(L)}\begin{pmatrix}I\\\hat{E}^{(L)}\end{pmatrix}, \tag{28}$$

where $\hat{P}^{(L)}=P^{(L)}\begin{pmatrix}I&\\&\Pi^{(L)}\end{pmatrix}$ is a permutation matrix, $\hat{E}^{(L)}=\begin{pmatrix}0&E^{(L)}\end{pmatrix}$, and the identity block in $\hat{P}^{(L)}$ and the zero block in $\hat{E}^{(L)}$ both have column sizes equal to $|\bar{\mathbf{s}}|$.

(28) essentially means that the entire $\mathbf{x}_i$ set is then sparsified to

$$\hat{\mathbf{s}}_i\equiv\bar{\mathbf{s}}_i\cup\tilde{\mathbf{s}}_i, \tag{29}$$

which corresponds to a row index set $\hat{\mathbf{I}}^{(L)}$ for C_i^{-}. This is illustrated in Figure 6(b). (28) may then also be written as

$$C_i^{-}(=\kappa(\mathbf{x}_i,\mathbf{y}_i))\approx U^{(L)}\kappa(\hat{\mathbf{s}}_i,\mathbf{y}_i)(=U^{(L)}C_i^{-}|_{\hat{\mathbf{I}}^{(L)}}). \tag{30}$$

Note that

$$|\mathbf{s}_i|=|\bar{\mathbf{s}}_i|=\frac{n}{2^{L+1}},\quad|\hat{\mathbf{s}}_i|=\frac{n}{2^{L+1}}+r. \tag{31}$$

At this point, we set the HSS generator $U_i=U^{(L)}$. If i is any other node at the leaf level not corresponding to the topmost HSS block row, we note that by Proposition 1, all of the displays in the previous case are still valid and the same matrix $U^{(L)}$ can serve as a row basis matrix. Hence, we may still set the HSS generator

$$U_i=U^{(L)}.$$

(Note that our notation $\Pi^{(L)}$, $U^{(L)}$, $\hat{E}^{(L)}$, and $\hat{\mathbf{I}}^{(L)}$ above is therefore justified.) Also, when $\mathbf{s}_i$ for each leaf i is sparsified, the entire set $\mathbf{x}$ gets sparsified accordingly, as illustrated in Figure 6(c).

4.2 | Compression at nonleaf levels

For a node i at a nonleaf level $l<L$, by induction, we may assume that we have computed $U^{(l+1)}$. Let a_1 and a_2 be the children of i.

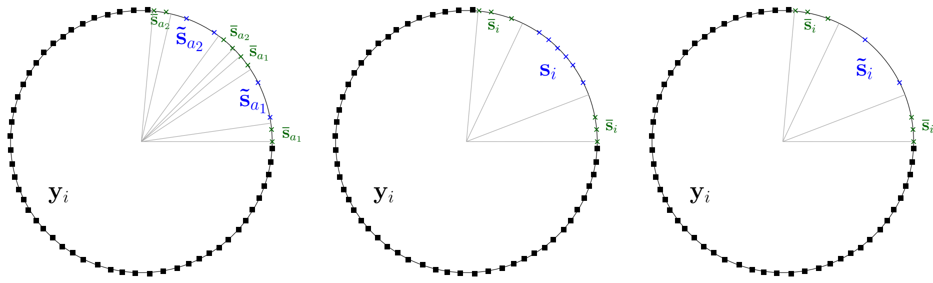
First, suppose that i corresponds to the topmost HSS block row at level l. With i in (30) set to be a_1 and a_2, respectively, we have that $C_{a_1}^{-}\approx U^{(l+1)}\kappa(\hat{\mathbf{s}}_{a_1},\mathbf{y}_{a_1})$ and $C_{a_2}^{-}\approx U^{(l+1)}\kappa(\hat{\mathbf{s}}_{a_2},\mathbf{y}_{a_2})$. Then

$$C_i^{-}(=\kappa(\mathbf{x}_i,\mathbf{y}_i))\approx\begin{pmatrix}U^{(l+1)}&\\&U^{(l+1)}\end{pmatrix}\begin{pmatrix}\kappa(\hat{\mathbf{s}}_{a_1},\mathbf{y}_i)\\\kappa(\hat{\mathbf{s}}_{a_2},\mathbf{y}_i)\end{pmatrix}\equiv\begin{pmatrix}U^{(l+1)}&\\&U^{(l+1)}\end{pmatrix}\kappa(\mathbf{x}_i,\mathbf{y}_i), \tag{32}$$

where $\mathbf{x}_i=\hat{\mathbf{s}}_{a_1}\cup\hat{\mathbf{s}}_{a_2}$. See Figure 8(a). Following the HSS construction procedure in Reference[31], the compression of C_i^{-} can then be done on $\kappa(\mathbf{x}_i,\mathbf{y}_i)$, which corresponds to the rows of C_i^{-} with index set $\hat{\mathbf{I}}^{(l+1)}\cup(\hat{\mathbf{I}}^{(l+1)}+\frac{n}{2^{l+1}})$. Similarly to before, partition $\mathbf{x}_i$ as

$$\mathbf{x}_i=\mathbf{s}_i\cup\bar{\mathbf{s}}_i\quad\text{with}$$
$$\mathbf{s}_i=\mathbf{x}_i\cap\left\{\omega^{2k-2}\mid\frac{n}{2^{l+2}}\le k<\frac{3n}{2^{l+2}}\right\},\quad\bar{\mathbf{s}}_i=\mathbf{x}_i\setminus\mathbf{s}_i. \tag{33}$$

Again, when the interaction between $\mathbf{x}_i$ and $\mathbf{y}_i$ are considered as in $\kappa(\mathbf{x}_i, \mathbf{y}_i)$, $\mathbf{s}_i$ and $\bar{\mathbf{s}}_i$ may be considered as the "far-field" and "near-field" subsets of $\mathbf{x}_i$, respectively. Note that the reason why we take the intersection with $\left\{ \omega^{2k-2} \mid \frac{n}{2^{l+2}} \le k < \frac{3n}{2^{l+2}} \right\}$ in (33) is for the purpose of separation in distance similar to (23). This ensures that, if we set $\mathbf{s} = \mathbf{s}_i$ and $\mathbf{t} = \mathbf{y}_i$ in Definition 1, then $\mathrm{gap}_{\mathcal{A}}(\mathbf{s}, \mathbf{t})$ and $\mathrm{span}_{\mathcal{A}}(\mathbf{s})$ satisfy (14) and $\mathbf{s}$ and $\mathbf{t}$ are well separated. See Figure 8(b). In the meantime, the size of the near-field subset $\bar{\mathbf{s}}_i$ remains reasonably small, which can be seen soon.



(a) $\hat{\mathbf{s}}_{a_1} = \bar{\mathbf{s}}_{a_1} \cup \tilde{\mathbf{s}}_{a_1}, \hat{\mathbf{s}}_{a_2} = \bar{\mathbf{s}}_{a_2} \cup \tilde{\mathbf{s}}_{a_2}$ (b) $\mathbf{s}_i$ (c) After sparsification

Figure 8 Forming $\mathbf{x}_i = \hat{\mathbf{s}}_{a_1} \cup \hat{\mathbf{s}}_{a_2}$ and $\mathbf{s}_i$ and sparsifying $\mathbf{s}_i$ into $\tilde{\mathbf{s}}_i$ for a nonleaf node i.

Then just like in the previous subsection, an analytical compression step can be applied to $\kappa(\mathbf{s}_i, \mathbf{y}_i)$. Choose proxy points on a circle with center c and radius γ, where

$$c = \omega^{\frac{n}{2^l} - 1}, \quad \gamma = \sqrt{|1 - c| \cdot |\omega^{\frac{n}{2^{l+1}}} - c|}.$$

(c is still the argument center of $\mathbf{x}_i$.) The proxy point method followed by an SPRR factorization sparsifies the set $\mathbf{s}_i$ to $\tilde{\mathbf{s}}_i$ of size $|\tilde{\mathbf{s}}_i| = r$. See Figure 8(c). Similarly to (26) and (30), we then obtain

$$\kappa(\mathbf{s}_i, \mathbf{y}_i) \approx \Pi^{(l)} \begin{pmatrix} I \\ E^{(l)} \end{pmatrix} \kappa(\tilde{\mathbf{s}}_i, \mathbf{y}_i), \tag{34}$$

$$\kappa(\mathbf{x}_i, \mathbf{y}_i) \approx U^{(l)} \kappa(\hat{\mathbf{s}}_i, \mathbf{y}_i), \quad \text{with} \quad U^{(l)} = \hat{P}^{(l)} \begin{pmatrix} I \\ \hat{E}^{(l)} \end{pmatrix}, \tag{35}$$

where $\hat{\mathbf{s}}_i$ is given as in (29) and corresponds to a row index set $\hat{\mathbf{I}}^{(l)}$ in $\kappa(\mathbf{x}_i, \mathbf{y}_i)$, $\hat{P}^{(l)}$ is a permutation matrix, and $\hat{E}^{(l)} = \begin{pmatrix} 0 & E^{(l)} \end{pmatrix}$ has bounded entries.

Note that

$$|\mathbf{s}_i| = |\bar{\mathbf{s}}_i| = \frac{n}{2^{L+1}} + r(L - l), \quad |\hat{\mathbf{s}}_i| = \frac{n}{2^{L+1}} + r(L - l + 1). \tag{36}$$

The reason for this is as follows. For i at level L, (31) holds. For i at level $L - 1$ with children a_1 and a_2, $\bar{\mathbf{s}}_i$ is formed by $\frac{n}{2^{l+2}}$ points of $\bar{\mathbf{s}}_{a_1}$, $\frac{n}{2^{l+2}}$ points of $\bar{\mathbf{s}}_{a_2}$, and r points from $\tilde{\mathbf{s}}_{a_1}$ and $\tilde{\mathbf{s}}_{a_2}$. (The points in $\tilde{\mathbf{s}}_{a_1}$ and $\tilde{\mathbf{s}}_{a_2}$ are sampled from $\mathbf{s}_{a_1}$ and $\mathbf{s}_{a_2}$, respectively, in the same way, and together their contributions to $\bar{\mathbf{s}}_i$ has r points.) Then, for i at any level l, (36) can be shown by induction. Thus, the size of the near-field set $\bar{\mathbf{s}}_i$ in (36) is reasonably small. (This also indicates that the resulting numerical ranks of the HSS blocks scale logarithmically, which is consistent with previous results in References[4, 13, 33].)

Remark 1. At this point, it is clear why we split the source set $\mathbf{x}_i$ instead of the target set $\mathbf{y}_i$. It is because $\mathbf{x}_i$ is sparsified after the compression while $\mathbf{y}_i$ is not. The sparsification of $\mathbf{x}_i$ ensures the near-field subset $\bar{\mathbf{s}}_i$ has cardinality as in (36) that only increases slightly when we move to an upper level. On the other hand, if we split $\mathbf{y}_i$, it would need a near-field subset of $\mathbf{y}_i$ to have a potentially large number of points since the argument span of each source set roughly doubles whenever we move to an upper level.

Now, from (32) and the HSS construction process in Reference[31], we can set the HSS generators R_{a_1}, R_{a_2} as

$$\begin{pmatrix} R_{a_1} \\ R_{a_2} \end{pmatrix} = U^{(l)}.$$

Again from Proposition 1, for any other node i at the same level l, the same R_{a_1}, R_{a_2} generators are used.

Also, for the purpose of extracting B generators later, we introduce the following index set (as a column vector):

$$\mathbf{J}^{(l)} = \left. \begin{pmatrix} \mathbf{J}^{(l+1)} \\ \mathbf{J}^{(l+1)} + \frac{n}{2^{l+1}} \end{pmatrix} \right|_{\hat{\mathbf{I}}^{(l)}}, \tag{37}$$

where $\mathbf{J}^{(L)} = \hat{\mathbf{I}}^{(L)}$. $\mathbf{J}^{(l)}$ is used to keep track of the index set of $\hat{\mathbf{s}}_i$ in $\mathbf{x}$ or the row index set of $\kappa(\hat{\mathbf{s}}_i, \mathbf{y}_i)$ in C.

This process is performed until every level $l > 0$ of T is visited. At that point, we get all the U, R generators in the HSS approximation.

4.3 | Other HSS generators

We now look at the V, R, B, D generators. As discussed in Section 3, Proposition 2 means $C_i^|$ and $(C_i^-)^T$ are closely related. In fact, other than one row in $C_i^|$, the remaining subblock of $C_i^|$ after a row permutation is a scalar multiple of $(C_i^-)^T$ with one row excluded. If we take an appropriate column $C|_{\mathbf{I}_i \times \{k\}}$ from $C|_{\mathbf{I}_i \times \mathbf{I}_i}$ (where the index k corresponds to a point of $\mathbf{y}$ right adjacent to $\mathbf{y}_i$) and put it next to C_i^-, then the resulting matrix $\left(C|_{\mathbf{I}_i \times \{k\}} \;\; C_i^- \right)$ has a subblock as a scalar multiple of $(C_i^|)^T$ after a column permutation. Thus indeed, a numerical column basis matrix for $\left(C|_{\mathbf{I}_i \times \{k\}} \;\; C_i^- \right)$ may be reused as a numerical column basis matrix for $(C_i^|)^T$. On the other hand, since $C|_{\mathbf{I}_i \times \{k\}}$ only adds an extra target point to the target set $\mathbf{y}_i$ and a basis matrix like G_i in (24) is only defined based on the source set $\mathbf{s}_i$ and the proxy points $\mathbf{z}_i$, the basis matrix G_i from the proxy point method applied to C_i^- can still serve as a numerical column basis matrix for $\left(C|_{\mathbf{I}_i \times \{k\}} \;\; C_i^- \right)$. The impact on the accuracy is negligible due to the separation between the source and target sets. Accordingly, G_i can still serve as a numerical column basis matrix for $(C_i^|)^T$.

As a result, we can obtain the V, W generators by setting for each leaf i,

$$V_i = U^{(L)}.$$

Similarly, for each nonleaf i at level l with children a_1, a_2, we set

$$\begin{pmatrix} W_{a_1} \\ W_{a_2} \end{pmatrix} = U^{(l)}.$$

These basis matrices are the same for all i at the same level due to Proposition 1.

Next, due to the forms of $U^{(L)}$ in (28) and $U^{(l)}$ in (35), the B_i generator is essentially a submatrix of C. We can use the index sets defined in (37) to pick, for a left node i at level l,

$$B_i = B^{(l,1)} \equiv C|_{\mathbf{J}^{(l)} \times (\mathbf{J}^{(l)} + \frac{n}{2^l})},$$

and for a right node i at level l,

$$B_i = B^{(l,2)} \equiv C|_{(\mathbf{J}^{(l)} + \frac{n}{2^l}) \times \mathbf{J}^{(l)}}.$$

Lastly, for the (leaf-level) D generators, we need only to store $D_1 = \kappa(\mathbf{x}_1, \mathbf{x}_1)$. Lemma 1 means that this can be used to obtain the D_i generators for any other leaf i.

Overall, we need only to obtain the matrices $U^{(l)}$, $B^{(l,1)}$, $B^{(l,2)}$ at each level l as well as D_1 at the leaf level. They are sufficient to write out all the HSS generators of the HSS approximation to C. The basis generators U, V, W, R are provided by $U^{(l)}$ which is defined by a permutation matrix and a matrix $E^{(l)}$ like in (28) or (35). For D_1 and each B generator, it just needs to store the associated index sets.

5 | ALGORITHM ANALYSIS AND PERFORMANCE

Algorithm 1 details the construction process. At each level of the HSS tree T, one HSS block is compressed. The corresponding point set is sparsified. The compression information is then used for the other HSS blocks at the same level. We would like to mention that the assumption that n is a power of 2 and the use of uniform block partitioning are merely for convenience. For general n with nonuniform partitioning, we may just compress the HSS block row (or column) with the largest row (or column) size at each level.

The scheme clearly leads to an overall sublinear complexity with a sublinear storage, as verified by the following proposition.

Proposition 3. Let $L = O(\log n)$ and let r be the number of proxy points used in each far-field compression step of the algorithm. Then the algorithm described above constructs the HSS approximation in $O(r^3 \log^2 n)$ flops with $O(r^2 \log^2 n)$ storage for the

Algorithm 1 Sublinear complexity HSS approximation for C

```
procedure $(D_1, U_1, R^{(l)}, B^{(l)}) = \mathsf{SublinearHSS}(T, \tau)$    ▷ $T$: HSS tree with $L$ levels
    $r \leftarrow O(|\log_2 \tau|)$    ▷ Number of proxy points per level
    $i \leftarrow 1$
    $\mathbf{x}_i \leftarrow \left(\omega^{2k-2}\right)_{0\le k<\frac{n}{2^L}}$    ▷ Indices ordered by increasing $k$
    $D_i \leftarrow C|_{\mathbf{s}_i\times\mathbf{s}_i}$
    for $l = L, L-1, \ldots, 1$ do    ▷ Levelwise traversal of the HSS tree
        if $i$ is a non-leaf node then
            $a_1 \leftarrow$ left child of $i$
            $\mathbf{x}_i \leftarrow \left(\hat{\mathbf{s}}_{a_1}, \omega^{\frac{n}{2^l}}\hat{\mathbf{s}}_{a_1}\right)$    ▷ Assembling $\mathbf{x}_i$ using $\hat{\mathbf{s}}_{a_1}$ from the child level; $\omega^{\frac{n}{2^l}}\hat{\mathbf{s}}_{a_1}$: entrywise product
        end if
        $\mathbf{s}_i \leftarrow \mathbf{x}_i \cap \left\{\omega^{2k-2} \mid \frac{n}{2^{l+2}} \le k < \frac{3n}{2^{l+2}}\right\}$    ▷ Far field
        $\bar{\mathbf{s}}_i \leftarrow \mathbf{x}_i \setminus \mathbf{s}_i$    ▷ Near field
        $c = \omega^{\frac{n}{2^l}-1}$    ▷ Center for proxy points
        $\gamma = \sqrt{|1-c|\cdot|\omega^{\frac{n}{2^{l+1}}} - c|}$    ▷ Radius of proxy circle
        $\mathbf{z} \leftarrow$ uniformly-spaced points on circle with center $c$ and radius $\gamma$
                                                                    ▷ Proxy points
        $\kappa(\mathbf{s}_i, \mathbf{z}) \approx U^{(l)}\kappa(\tilde{\mathbf{s}}_i, \mathbf{z})$    ▷ Rank-$r$ SPRR factorization to get $\tilde{\mathbf{s}}_i \subset \mathbf{s}_i$
        $\hat{\mathbf{s}}_i \leftarrow \bar{\mathbf{s}}_i \cup \tilde{\mathbf{s}}_i$    ▷ Reordered counterclockwise in $\mathbb{C}$ starting at 1
        if $l = L$ then
            $U_i \leftarrow U^{(l)}$
        else
            $\begin{pmatrix} R_{a_1} \\ R_{a_2} \end{pmatrix} \leftarrow U^{(l)}$    ▷ $a_1, a_2$: children of $i$
        end if
        $\mathbf{J}_i \leftarrow$ index set corresponding to $\hat{\mathbf{s}}_i$ in $\mathbf{x}$
        $B_i \leftarrow C|_{\mathbf{J}_i\times(2^{L-l}+\mathbf{J}_i)}$, $B_{i+1} \leftarrow C|_{(2^{L-l}+\mathbf{J}_i)\times\mathbf{J}_i}$
        $i \leftarrow$ parent of $i$ in $T$
    end for
end procedure
```

HSS generators if the B generators are not explicitly formed. An extra $O(r^2\log^3 n)$ cost and $O(r^2\log^3 n)$ storage are needed for the B generators. In addition, the B generators at level l have sizes $O\left(r(L-l)\right)$.

Proof. At each level l, associated with the leftmost node i, the cost to form G_i like in (24) is $O\left(r\left(\frac{n}{2^{L+1}} + r(L-l)\right)\right)$ flops, since G_i has size $|\mathbf{s}_i| \times r$ with $|\mathbf{s}_i|$ in (36). The factorization of G_i in (25) costs $O\left(r^2\left(\frac{n}{2^{L+1}} + r(L-l)\right)\right)$. The total compression cost at all levels is then

$$\sum_{l=1}^{L} O\left(r^2\left(\frac{n}{2^{L+1}} + r(L-l)\right)\right) = O(r^3L^2).$$

The storage is mainly for $E^{(l)}$ in (34) and for some index vectors and looks like

$$\sum_{l=1}^{L} O\left(r\left(\frac{n}{2^{L+1}} + r(L-l)\right)\right) = O(r^2L^2).$$

The B generators at level l have sizes $|\mathbf{J}^{(l)}|$ with $\mathbf{J}^{(l)}$ in (37), which is also $|\hat{\mathbf{I}}^{(l)}|$. Now, $|\hat{\mathbf{I}}^{(l)}| = |\hat{\mathbf{s}}_i| = O\left(r(L-l)\right)$ from (36). If additionally we explicitly form B generators, then the extra cost is

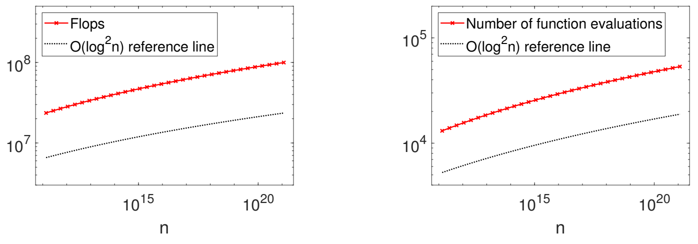
$$\sum_{l=1}^{L} O\left((r(L-l))^2\right) = O(r^2L^3).$$

(Note that the cost for forming D_1 is only $O(r^2)$.) □

From this proposition, we can see that the resulting HSS approximation has B generators at level l with sizes $O(r(L-l)) = O(\log n)$. This reflects the off-diagonal ranks of the HSS form. With the compression of each far-field blocks satisfying the accuracy in (9), it is convenient to apply a result in Reference[27] to obtain a global approximation error, which is roughly $O(\tau r^{\log n})$.

To illustrate the performance of the algorithm, we apply it to C with matrix sizes $n = 2^7, 2^8, \dots, 2^{70}$. The leaf-level diagonal block size is 128. The number of proxy points $r = 25$ is used for all the compression steps, which is also the numerical rank in (25). To get (25), we use a rank-revealing QR factorization with row pivoting in the implementation of our algorithms. Its flop count follows that in Reference[11]. The pivoted QR factorization may be less accurate than the strong rank-revealing factorization. However, it is usually more efficient and works reasonably well in practical tests. The accuracy results in Table 1 vindicate this.

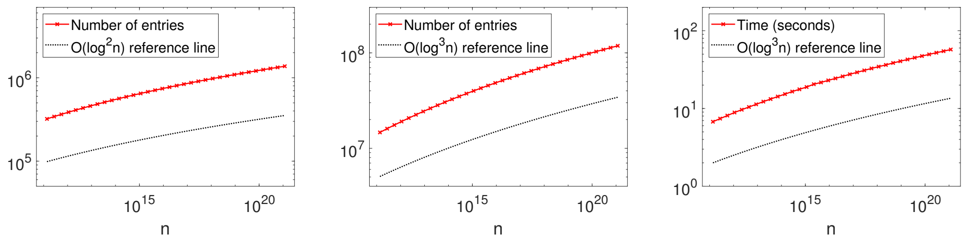
In Figure 9(a), we give flop counts in the construction of the D, U, R, V, W generators using Algorithm 1. For these generators, we also need a number of function evaluations to precompute selected points from (2) on the unit circle. For completeness, we count these function evaluations in Figure 9(b). It is evident that both counts asymptotically follow the $O(\log^2 n)$ pattern, as predicted by Proposition 3.



(a) Cost (excluding function evaluations) (b) Number of function evaluations

Figure 9 Numbers of flops and exponential function evaluations for the construction of the D, U, R, V, W generators for C with varying n, where (a) includes all flop counts not associated with the exponential function evaluations and (b) is for the number of exponential function evaluations used to precompute selected points from (2).

To report the storage of the generators, we look at the "effective storage". That is, the generators are stored in an economic way. The storage needed for the D generators is for only r^2 entries of D_1 and is negligible. For the U, R, V, W generators, only the E matrices like in (34) and some permutation vectors are stored. Such an effective storage count is given in Figure 10(a) and also follows the $O(\log^2 n)$ pattern. The storage for the B generators is shown in Figure 10(b) and follows the $O(\log^3 n)$ pattern, which is consistent with Proposition 3.

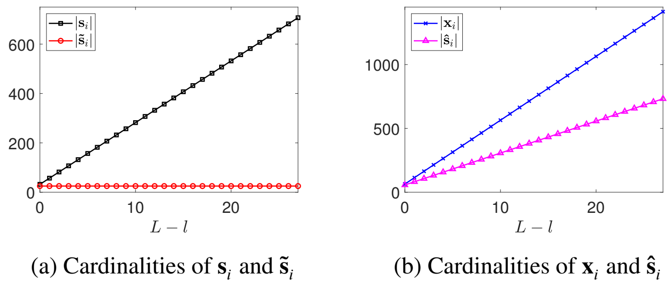


(b) Storage without B generators (b) Storage for B generators (c) Total timing

Figure 10 Effective storage for the generators and total HSS construction timing.

Note that the cost for forming the B generators is from the evaluation of the entries. Thus, it is simply proportional to the storage count so that the plot in Figure 10(b) also reflects the cost to form the B generators. Then the overall cost of the algorithm behaves like $O(\log^3 n)$, which corroborates that the algorithm has sublinear complexity. This can also be observed from the computational time (in Matlab running on a computing cluster node with two 2.6GHz CPUs). See Figure 10(c). Note that even in Matlab, it takes only about 57 seconds for the HSS construction for the matrix size as large as $n = 2^{70} \approx 1.18e21$.

Also, in Figures 11, we plot the cardinalities of $\mathbf{s}_i$, $\tilde{\mathbf{s}}_i$, $\mathbf{x}_i$, and $\hat{\mathbf{s}}_i$ at each hierarchical level for C of size $n = 2^{34} \approx 1.72e10$. (Such cardinalities are the same for all i at each level $l = 1, 2, \ldots, L$, with $L = 28$.) The cardinalities of $\mathbf{s}_i$, $\hat{\mathbf{s}}_i$, and $\mathbf{x}_i$ demonstrate linear growth with respect to $L - l$. $|\tilde{\mathbf{s}}_i|$ remains constant. This illustrates the compactness of the resulting HSS generators and accordingly validates the counts predicted by Proposition 3.



(a) Cardinalities of $\mathbf{s}_i$ and $\tilde{\mathbf{s}}_i$ (b) Cardinalities of $\mathbf{x}_i$ and $\hat{\mathbf{s}}_i$

Figure 11 Cardinalities of $\mathbf{s}_i$, $\tilde{\mathbf{s}}_i$, $\mathbf{x}_i$, and $\hat{\mathbf{s}}_i$ at different levels, where $n = 2^{34}$, $L = 28$.

Next, Table 1 shows the relative error $\frac{\|\tilde{C}-C\|_F}{\|C\|_F}$ for the HSS approximation $\tilde{C}$ to C constructed using Algorithm 1. Due to the high expense for dense matrices, we only report the accuracy for smaller n. The other parameters are the same as above. It is clear that the compact HSS approximations constructed with our sublinear complexity algorithm achieve desirable accuracies.

Table 1 Relative error for the HSS approximation $\tilde{C}$ to the dense matrix C.

n	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}
$\frac{\|\tilde{C}-C\|_F}{\|C\|_F}$	$4.08e-13$	$4.29e-13$	$3.18e-13$	$2.71e-13$	$5.13e-13$	$7.75e-13$

Finally, Table 2 compares the cost of Algorithm 1 for the HSS construction (as in Figure 9(a)) with the cost of matrix-vector multiplications using the resulting HSS forms. As the matrix size doubles, the flops of the matrix-vector multiplication grow nearly linearly, while those of the HSS construction grow far more slowly. Also, the flops in the table required by the matrix-vector multiplication are now greater, showing that the associated HSS construction is no longer the performance bottleneck.

Table 2 Comparison between the flops of Algorithm 1 (as in Figure 9(a)) with the flops of matrix-vector multiplications using the resulting HSS forms.

n	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
HSS construction	$6.71e5$	$9.48e5$	$1.27e6$	$1.65e6$	$2.07e6$	$2.54e6$
HSS matrix-vector multiplication	$3.51e6$	$7.60e6$	$1.60e7$	$3.29e7$	$6.71e7$	$1.36e8$

6 | GENERALIZATIONS

Our discussions above are given in terms of the important class of Cauchy matrices with regularly-spaced points on the unit circle. Going forward, we can extend these ideas to other kernel matrices defined on a set of points with regular geometry as follows. Since the compression step for each HSS row block C_i^- only relies on the factor G_i in (24), it is feasible to apply the ideas here to other problems. For example, reusing the same row basis $U^{(l)}$ as in (35) across all the HSS block rows at the same level is possible whenever the relative geometry of the proxy points and the enclosed points are the same due to symmetry. See Figure 12 for two examples. In our construction algorithm for C, we make use of the circle symmetry of the points in question, shown in Figure 12(a), to reuse generators. This idea also applies to other types of symmetries such as the line symmetry in Figure 12(b), when the kernel in question is evaluated at equally-spaced points on the line.

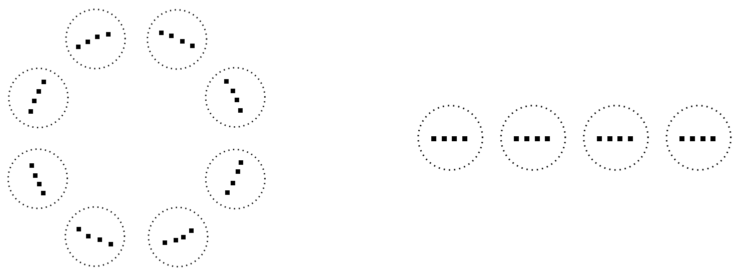

(a) Circle symmetry (b) Line symmetry

Figure 12 Symmetry of the proxy point contours (dotted lines) and their enclosed points $\mathbf{s}_i$ (■).

Moreover, problems involving other kernel functions $\xi(x, y)$ evaluated at more general point sets may also benefit from our ideas. In particular, let $A = \xi(\mathbf{x}, \mathbf{y})$ for some kernel function $\xi(x, y)$ and finite sets $\mathbf{x}, \mathbf{y} \subseteq \mathbb{C}$, and let $\mathbf{x}_i, \mathbf{y}_i$ be the points associated with the ith HSS block row A_i^-. Suppose that for each $y \in \mathbf{y}_i$, $\xi(x, y)$ is complex-analytic in x on a region containing a circular contour around a far-field subset $\mathbf{s}_i$ of $\mathbf{x}_i$. Let $\mathbf{z}_i$ be the associated set of proxy points. Then, using a Cauchy integration strategy like in Reference[35], we may construct the following proxy point approximation:

$$\xi(\mathbf{s}_i, \mathbf{y}_i) \approx \kappa(\mathbf{s}_i, \mathbf{z}_i)\xi_0(\mathbf{z}_i, \mathbf{y}_i), \tag{38}$$

where κ is still the Cauchy kernel in (1) and ξ_0 depends on ξ and i. Here the basis matrix $\kappa(\mathbf{s}_i, \mathbf{z}_i)$ is then similar to that in (7). If this approximation yields an identical $\kappa(\mathbf{s}_i, \mathbf{z}_i)$ for each i, we may apply the basis reuse idea from our earlier discussion. In our forthcoming work[17], we show that this is indeed the case for kernel matrices that have Toeplitz structures. Specifically, an $n \times n$ Toeplitz matrix may be viewed as a kernel matrix given by the evaluation of a certain function $f(x - y)$ on the point set $\{1 : n\}$. That is, $T_{i,j} = f(j - i)$ for $1 \leq i, j \leq n$. For certain such f, it can be shown that we can apply our technique from Section 4 to directly construct HSS approximations in sublinear complexity. Note that it is also possible to apply this technique to other, non-Toeplitz kernel matrices with point symmetry, such as the Hilbert matrix.

Additionally, it may be possible to consider other, non-circular contours for the proxy point approximation. In general, the techniques outlined here do require point symmetries. It may also be possible to extend the techniques to higher dimensions with sufficiently nice point symmetries, which is worth studying in future work. A possible strategy is to integrate some of our strategies into the methods in References[2, 3, 8].

Another way to generalize the work is to design preconditioners. For problems where the point sets are not uniform or it is difficult to find proxy point sets with the desired symmetries, we may choose appropriate uniform point sets or proxy point sets with symmetry so as to construct preconditioners in sublinear complexity. The resulting preconditioners may then be applied quickly via rank-structured solvers.

As mentioned earlier, the ideas in the paper mainly focus on accelerating the structured approximation stage of fast algorithms for some kernel matrices. In general, subsequent matrix-vector multiplications or linear system solutions need at least $O(n)$ complexity since an associated vector or right-hand side b itself has $O(n)$ data. However, it is possible to take advantage of our ideas to further reduce this cost in some applications, where b is also structured. For example, sometimes b may be approximated by low-rank forms after reshaping. Then the shared basis generators produced by our method may be multiplied with the low-rank basis matrices from the reshaped *b*. The results can then be shared across multiple nodes at each level of the tree.

CONCLUSIONS

In this work, we have outlined a new method to construct a rank-structured approximation to a type of important Cauchy matrices. These Cauchy matrices arise from intermediate steps of direct Toeplitz solvers and computations with some kernel matrices. The new method relies on the reuse of a low-rank basis matrix for all HSS blocks at a given HSS level, the fast computation of this basis matrix using a proxy point method, and a analytical hierarchical compression scheme. The feasibility of the basis reuse and an optimal parameter choice in the analytical compression are analyzed. The novel combination of these ideas effectively reduce the number of compression steps from the usual $O(n)$ to $O(\log n)$, leading to sublinear complexity and storage of the resulting algorithm. The new efficiency is confirmed both theoretically and numerically. We also note that the ideas may be further extended to different contexts to obtain similarly quick rank-structured approximation for more general kernel matrices.

ACKNOWLEDGEMENTS

The authors thank Ming Gu at UC Berkeley for some useful discussions, especially on reusing basis matrices on off-diagonal compression in Section 4. Also thank two anonymous reviewers for some helpful suggestions.

Conflict of interest

The authors declare no potential conflict of interests.

References

1. Börm S, Gördes, J. Low-rank approximation of integral operators by using the Green formula and quadrature. Numer. Algor. 2013;**64**:567–592.

2. Cai D, Chow E, Erlandson L, Saad Y, Xi Y. SMASH: Structured matrix approximation by separation and hierarchy. Numer. Linear Algebra Appl. 2018;**25**:e2204.

3. Cai D, Huang H, Chow E, Xi Y. Data-driven construction of hierarchical matrices with nested bases. arXiv 2002;2206.01885.

4. Cai D, Xia J. A stable matrix version of the fast multipole method: stabilization strategies and examples. Electron. Trans. Numer. Anal. 2021;**54**:581–609.

5. Chandrasekaran S, Dewilde P, Gu M, Pals T. A fast ULV decomposition solver for hierarchically semiseparable representations. SIAM J. Matrix Anal. Appl. 2006;**28**:603–622.

6. Chandrasekaran S, Gu M, Sun X, Xia J, Zhu J. A superfast algorithm for Toeplitz systems of linear equations. SIAM J. Matrix Anal. Appl. 2007;**29**:1247–1266.

7. Corona E, Martinsson PG, Zorin D. An $O(N)$ direct solver for integral equations in the plane. Appl. Comput. Harmon. Anal. 2015;**38**:284–317.

8. Erlandson L, Cai D, Xi Y, Chow E. Accelerating parallel hierarchical matrix-vector products via data-driven sampling. 34th IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, 2020.

9. Gohberg I, Kailath T, Olshevsky V. Fast Gaussian elimination with partial pivoting for matrices with displacement structure. Math. Comp. 1995;**64**:1557–1576.

10. Greengard L, Rokhlin V. A fast algorithm for particle simulations. J. Comput. Phys., 1987;**73**:325–348.

11. Gu M, Eisenstat SC. Efficient algorithms for computing a strong-rank revealing QR factorization. SIAM J. Sci. Comput. 1996;**17**:848–869.

12. Hackbusch W. A sparse matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices. Computing. 1999;**62**:89–108.

13. Hackbusch W, Khoromskij BN, Kriemann R. Hierarchical matrices based on a weak admissibility criterion. Computing. 2004;**73**:207–243.

14. Heinig G. Inversion of generalized Cauchy matrices and other classes of structured matrices, in Linear Algebra for Signal Processing, IMA Vol. Math. Appl. 69, Springer, New York, 1995:95–114.

15. Ho KL, Ying L. Hierarchical interpolative factorization for elliptic operators: integral equations. Comm. Pure Appl. Math. 2016;**69**:1314–1353.

16. Kailath T, Kung SY, Morf M. Displacement ranks of matrices and linear equations. J. Math. Anal. Appl. 1979;**68**:395–407.

17. Lepilov M, Xia J. The proxy point method for analytic kernels with applications to some Toeplitz matrices. preprint.

18. Liberty E, Woolfe F, Martinsson PG, Rokhlin V, Tygert M. Randomized algorithms for the low-rank approximation of matrices. Proc. Natl. Acad. Sci. USA. 2007;**104**:20167–20172.

19. Martinsson PG, Rokhlin V. A fast direct solver for boundary integral equations in two dimensions. J. Comput. Phys. 2005;**205**:1–23.

20. Martinsson PG, Rokhlin V, Tygert M. A fast algorithm for the inversion of general Toeplitz matrices. Comput. Math. Appl. 2005;**50**:741–752.

21. Ou X, Xia J. SuperDC: Superfast divide-and-conquer eigenvalue decomposition with improved stability for rank-structured matrices. SIAM J. Sci. Comput. 2022;to appear.

22. Pan VY. On computations with dense structured matrices. Math. Comp., 1990;**55**:179–190.

23. Pan VY. Transformations of matrix structures work again. Linear Algebra Appl. 2015;**465**:107–138.

24. Shen J, Wang Y, Xia J. Fast structured direct spectral methods for differential equations with variable coefficients, I. The one-dimensional case. SIAM J. Sci. Comput. 2016;**38**:A28–A54.

25. Tyrtyshnikov EE. Mosaic-skeleton approximations. Calcolo. 1996;**33**:47–57.

26. Vogel J, Xia J, Cauley S, Balakrishnan V. Superfast divide-and-conquer method and perturbation analysis for structured eigenvalue solutions. SIAM J. Sci. Comput. 2016;**38**:A1358–A1382.

27. Xi Y, Xia J, Cauley S, Balakrishnan V. Superfast and stable structured solvers for Toeplitz least squares via randomized sampling. SIAM J. Matrix Anal. Appl. 2014;**35**:44–72.

28. Xi Y, Xia J, Chan R. A fast randomized eigensolver with structured LDL factorization update. SIAM J. Matrix Anal. Appl. 2014;**35**:974–996.

29. Xia J. On the complexity of some hierarchical structured matrix algorithms. SIAM J. Matrix Anal. Appl. 2012;**33**:388–410.

30. Xia J. Multi-layer hierarchical structures. CSIAM Trans. Appl. Math., 2021;**2**:263–296.

31. Xia J, Chandrasekaran S, Gu M, Li XS. Fast algorithms for hierarchically semiseparable matrices, Numer. Linear Algebra Appl. 2010;**17**:953–976.

32. Xia J, Lepilov M. Why are many circulant matrices rank structured? preprint.

33. Xia J, Xi Y, Gu M. A superfast structured solver for Toeplitz linear systems via randomized sampling. SIAM J. Matrix Anal. Appl. 2012;**33**:837–858.

34. Xing X, Chow E. Interpolative decomposition via proxy points for kernel matrices. SIAM J. Matrix Anal. Appl. 2020;**41**:221–243.

35. Ye X, Xia J, Ying L. Analytical low-rank compression via proxy point selection. SIAM J. Matrix Anal. Appl. 2020;**41**:1059–1085.

36. Ying L. A kernel independent fast multipole algorithm for radial basis functions. J. Comput. Phys., 2006;**213**:451–457.

37. Ying L, Biros G, Zorin D. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. J. Comput. Phys. 2004;**196**:591–626.

38. Zhang Y, Leithead WE, Leith DJ. Time-series Gaussian process regression based on Toeplitz computation of $O(N^2)$ operations and $O(N)$-level storage. Proc. IEEE Conf. Decis. Control. 2005;**44**:3711–3716.

How to cite this article: M. Lepilov and J. Xia (2023), Sublinear complexity rank-structured approximation of some Cauchy matrices, *submitted*.