A HIERARCHICAL RANDOM COMPRESSION METHOD FOR KERNEL MATRICES*

DUAN CHEN† AND WEI CAI‡

Abstract. In this paper, we propose a hierarchical random compression method (HRCM) for kernel matrices in fast kernel summations. The HRCM combines the hierarchical framework of the $\mathscr{H}$-matrix and a randomized sampling technique of column and row spaces for far-field interaction kernel matrices. We show that a uniform column/row sampling of a far-field kernel matrix, thus without the need and associated cost to pre-compute a costly sampling distribution, will give a low-rank compression of such low-rank matrices, independent of the matrix sizes and only dependent on the separation of the source and target locations. This far-field random compression technique is then implemented at each level of the hierarchical decomposition for general kernel matrices, resulting in an $O(N \log N)$ random compression method. Error and complexity analysis for the HRCM are included. Numerical results for electrostatic and Helmholtz wave kernels have validated the efficiency and accuracy of the proposed method with a cross-over matrix size, in comparison of direct $O(N^2)$ summations, in the order of thousands for a 3-4 digits relative accuracy for electrostatic and low frequency wave interaction kernels.

Key words. Fast kernel summation, Randomized algorithm, Matrix approximation, Singular value decomposition, $\mathscr{H}$-matrix, Hierarchical algorithms.

AMS subject classifications. 65F30; Secondary, 68W20, 60B20

1. Introduction.

Kernel matrices arises from many scientific and engineering computation, data analytics, and deep learning algorithms. How to efficiently handle those matrices has been an active research area as the size of the matrices increase dramatically due to large dimension data set in the era of exascale computing and big data. In many fields of applications, kernel matrices are used for summations of the following form

$$E_i = q_i \sum_{j=1}^{N} \mathcal{K}(\mathbf{r}_i, \mathbf{r}_j) q_j, \quad i = 1, 2, ...M, \tag{1}$$

where $\mathcal{K}(\mathbf{r}_i, \mathbf{r}_j)$ is the kernel function representing the interaction betweens M targets $\{q_i\}_{i=1}^{M}$ and N sources $\{q_j\}_{j=1}^{N}$, which have position coordinates $\mathbf{r}_i$ and $\mathbf{r}_j$, respectively. In scientific computing, Eq. (1) could arise from boundary element discretization for partial differential equations such as Poisson-Boltzmann equation, Helmholtz equation, Maxwell equations or fractional differential equations. The kernel summation also plays a fundamental role in non-parametric statistics or machine learning techniques in applications such as Latent semantic indexing (LSI), or analysis in DNA microarray data, or eigenfaces and facial recognition.

When M and N are large, the summation task as Eq. (1) becomes prohibitively expensive for practical computations. To speed up the kernel matric summation and reduce data storage, a low rank approximation usually is sought to reduce the operation from $O(MN)$ to $O(KN)$ in a K-rank approximation where $K \ll N$. Generally, a small subspace, containing the major action of the original matrix, is first

*Submitted to the editors on June 14, 2018.

Funding: This work was funded by the US Army Office of Research (Grant No. W911NF-17-1-0368) and US National Science Foundation (Grant No. Grant No. DMS-1802143).

†Department of Mathematics and Statistics, University of North Carolina at Charlotte, Charlotte, NC 28223, USA.

‡Corresponding author, Department of Mathematics, Southern Methodist University, Dallas, TX75275, USA.

identified, which is then compressed and approximated by a low rank representation. Traditional truncated singular value decomposition (SVD) and rank-revealing QR factorization are popular ways to construct such low rank approximations. But the cost of implementing SVD or QR themselves are much higher than a single matrix-vector multiplication, which is the typical core operation required for an iterative solution of the linear systems related to the kernel matrices. The fast multipole method (FMM) [12] is one of the most important fast algorithms for kernel summation, in which target and source points are hierarchically divided as well-separated sets, and on each set, the kernel function is low-rank approximated by using multipole expansions. In the original FMM, kernel function is approximated by analytical tools (either with addition theorems of special functions or Taylor expansions) [12, 4, 6, 10, 5]. To overcome the difficulties when analytic formulation of kernel functions is not available, various semi-analytic[1, 11, 18] and algebraic FMMs [21, 22, 23] were developed in recent decades. In some other approaches [16, 17], the whole kernel matrix is split into block matrices with various ranks, on each of which the SVD was implemented and then a truncated summation was used. More recently, a random interpolative decomposition method was developed in the framework of FMM [20, 19, 24].

To take advantage of modern computational architecture and especially for high dimensional data, randomized methods are found to provide efficient approach of low-rank approximations to handle transient data sets, where access to the full matrices may not be possible or too expensive as only single pass or constant number of passes of the matrices are preferred or realistic. Some fast Monte-Carlo algorithms for large matrix-matrix multiplications and low-rank approximations have been developed in [7, 8]. Still, those algorithms are generally not more efficient than *a single* implementation of Eq. (1). For example, it was proposed [9][15] that by randomly sampling the column and rows of a matrix, based on a the magnitude of the row/column vector, will give a good low rank approximation in high probability. However, such an approach requires the calculation of the L_2 norm of the column/row vectors, which is already of complexity $O(MN)$ as the direct kernel summation cost. Furthermore, error analysis in [7, 8] is for general matrices and relies on large number of samples. Number of samples will be greatly limited in practice if high efficiency is pursued and the low-rank property of the matrix should give some additional benefits in error analysis.

In this paper, we develop a novel hierarchical random compression algorithm for kernel matrices in order to enhance the efficiency of kernel summation and reduce data storage at a very large scale. This hierarchical approach is different from that of FMM methods; it requires only a one-way top-down pass, and is simple to implement in a recursive manner. More importantly, this algorithm does not use any analytic expansion of the kernel function. Thus it is especially useful when the analytic formula of kernel does not exist, such as Green's functions in complicated inhomogeneous geometries. To achieve this goal, we first apply the randomized sampling of the column and row space technique for the kernel matrix resulting from far-field interactions, i.e. the target and source points set are well-separated. We show that for such a scenery, the uniform sampling distribution will be sufficient to give a good low-rank approximation, thus removing the need and associated cost of computing sampling distributions. The expectation of error depends on the number of sampled column (row) and the diameter-distance ratio of the two sets. Next, for general source-field configurations, we will employ the $\mathscr{H}$-matrix framework [13, 14] to construct a hierarchical multilevel tree structure, and apply the far-field randomized low-rank approximation for admissible interactions at each level, recursively.

The rest of the paper is organized as follows. Section 2 describes the randomized

compression method for far field interaction matrices. Analysis of the algorithm for the far field case is given in Section 3. Section 4 introduces the hierarchical random compression method (HRCM) where $\mathscr{H}$-matrix framework is used with the randomized compression method applied to far field on different hierarchical level of the data set. Numerical tests for the proposed HRCM are provided for kernels from electrostatic and Helmholtz Green's functions in Section 5. Finally, conclusion and discussion are given in Section 6.

2. Basic random algorithms for well-separated sources and targets. For convenience, we list some basic notations in linear algebra. For a vector $\mathbf{x} \in \mathbb{K}^N$, where $\mathbb{K}$ represents either real $\mathbb{R}$ or complex $\mathbb{C}$, denote Euclidean norm by $|\mathbf{x}| = \left(\sum_{i=1}^N |x_i|^2\right)^{1/2}$. For a matrix $\mathbf{A} \in \mathbb{K}^{M\times N}$, let $A^{(j)}$ and $A_{(i)}$ denote its j-th column and i-th row, respectively. The Frobenius norm is $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n A_{ij}^2}$, and the product of $\mathbf{AB}$ can be written as $\mathbf{AB} = \sum_{j=1}^N A^{(j)} B_{(j)}$.

In this section and Section 3 we restrict ourselves to low-rank matrices corresponding to the approximation of kernel function for well-separated target and source points.

Define the diameter of the target set $\{\mathbf{r}_i\}$ as

$$diam(T) := \max_{i,i'} |\mathbf{r}_i - \mathbf{r}_{i'}|, \tag{2}$$

using the Euclidean norm in $\mathbb{R}^d$. Diameter of the source set $diam(S)$ is defined similarly. Additionally, we will need the distance of the two groups as

$$dist(T,S) := |\mathbf{r}_T^* - \mathbf{r}_S^*|, \tag{3}$$

where $\mathbf{r}_T^*$ and $\mathbf{r}_S^*$ are the Chebyshev centers of sets T and S, respectively. Then we say the source and target charges are well-separated if $dist(T,S) \geq \frac{1}{2}(diam(T)+diam(S))$.

2.1. Low-rank characteristics of kernel functions. We say the kernel function $\mathcal{K}(\mathbf{r},\mathbf{r}')$ is a generalized asymptotically smooth function [3, 2] if

$$|\partial_{\mathbf{r}}^{\alpha}\partial_{\mathbf{r}'}^{\beta}\mathcal{K}(\mathbf{r},\mathbf{r}')| \leq c(|\alpha|,|\beta|)(1+k|\mathbf{r}-\mathbf{r}'|)^{|\alpha|+|\beta|}|\mathbf{r}-\mathbf{r}'|^{-|\alpha|-|\beta|-\tau}, \tag{4}$$

where parameters $k, \tau \geq 0$ and α, β are d-dimensional multi-indices. The constant $C(|\alpha|,|\beta|)$ only depends on $|\alpha|$ and $|\beta|$. Condition (4) covers a wide-range of kernel functions. For example, for Green's functions of Laplace equation, $k = 0$, $\tau = 0$ for the 2D case and $k = 0$, $\tau = 1$ for the 3D case, respectively. For 3D Green's functions of Helmholtz equation, one has k as wave number and $\tau = 1$.

Consider the Taylor expansion of $\mathcal{K}(\mathbf{r},\mathbf{r}')$ around $\mathbf{r}^*$, which is the Chebyshev center of T: $\mathcal{K}(\mathbf{r},\mathbf{r}') = \tilde{\mathcal{K}}(\mathbf{r},\mathbf{r}') + R$ with the polynomial

$$\tilde{\mathcal{K}}(\mathbf{r},\mathbf{r}') = \sum_{|\upsilon|=0}^{m-1} \frac{1}{\upsilon!}(\mathbf{r}-\mathbf{r}^*)^{\upsilon} \frac{\partial^{\upsilon}\mathcal{K}(\mathbf{r}^*,\mathbf{r}')}{\partial \mathbf{r}^{\upsilon}}, \tag{5}$$

and the remainder R satisfies

$$|R| = |\mathcal{K}(\mathbf{r},\mathbf{r}') - \tilde{\mathcal{K}}(\mathbf{r},\mathbf{r}')| \leq \frac{1}{m!}|\mathbf{r}-\mathbf{r}^*|^m \max_{\zeta\in T,|\gamma|=m} \left|\frac{\partial^{\gamma}\mathcal{K}(\zeta,\mathbf{r}')}{\partial \zeta^{\gamma}}\right|. \tag{6}$$

If the well-separated target (T) and source (S) charges satisfies

$$\frac{\max\{diam(T), diam(S)\}}{dist(T,S)} < \eta, \tag{7}$$

for a parameter $0 < \eta < 1$, then Eq. (6) has the following estimate,

$$|\mathcal{K}(\mathbf{r},\mathbf{r}') - \tilde{\mathcal{K}}(\mathbf{r},\mathbf{r}')| \le c(m)\eta^m(1+k|\mathbf{r}-\mathbf{r}'|)^m|\mathbf{r}-\mathbf{r}'|^{-\tau}, \quad \mathbf{r}\in T, \mathbf{r}'\in S. \tag{8}$$

Estimate (8) implies that for small wavenumber parameter k , if one replaces $\mathcal{K}(\mathbf{r}_i,\mathbf{r}_j)$ in Eq. (1) by $\tilde{\mathcal{K}}(\mathbf{r}_i,\mathbf{r}_j)$ as defined in Eq. (5) with error $O(\eta^m)$, the corresponding interaction matrix $\tilde{\mathcal{K}}$ is of low rank, i.e.

$$\text{rank}(\tilde{\mathcal{K}}) = K(m) \ll \min\{M,N\} = \text{rank}(\mathcal{K}), \tag{9}$$

where

$$K(m) = \sum_{p=0}^{m-1} \frac{(d-1+p)!}{(d-1)!p!}. \tag{10}$$

In the case of 2D, we have $d = 2$ and $K(m) = m(m+1)/2$.

This property indicates that we can replace matrix $\mathcal{K}$ by the low-rank matrix $\tilde{\mathcal{K}}$ with enough accuracy for well-separated target and source points. Thus, the efficiency of computation could be greatly improved. However, in many situations, there is not easily available explicit formula for $\tilde{\mathcal{K}}$, as for layered Green's function, thus the matrix $\tilde{\mathcal{K}}$ cannot not be explicitly computed by Eq. (5). In our approach, we will take advantage of the fact that $\mathcal{K}$ has redundant information (essential low rank characteristics) and will use randomized sampling methods to select a small amount of its rows or columns and approximate the full matrix with the sampled sub-matrices. It should be noted that the low-rank characteristics depend on parameter k and τ in the decaying condition of the kernel (4).

2.2. Random kernel compression algorithms. For matrix $\mathbf{A} \in \mathbb{K}^{M\times N}$ with a rank $K \ll \min\{M,N\}$, the matrix-vector multiplication with vector $\mathbf{x} \in \mathbb{K}^N$ can be represented as

$$\mathbf{A}\mathbf{x} = \sum_{i=1}^{K} \sigma_A^i \mathbf{U}_A^{(i)} {\mathbf{V}_A^{(i)}}^* \mathbf{x}, \tag{11}$$

where $\mathbf{A} = \mathbf{U}_A\Sigma_A\mathbf{V}_A^*$, $\mathbf{U}_A \in \mathbb{K}^{M\times M}$, $\mathbf{V}_A \in \mathbb{K}^{N\times N}$ is the SVD decomposition of matrix $\mathbf{A}$. Although Eq. (11) includes small amount of summation, performing SVD to obtain σ_A^i, $\mathbf{U}_A^i$ and $\mathbf{V}_A^i$ is much more expensive than the direct multiplication. Instead, we will approximate the singular values and unitary matrices by fast Monte Carlo methods. This process is outlined as follows:

Let $\mathbf{C} \in \mathbb{K}^{M\times c}$ be the matrix made of c columns sampled from matrix $\mathbf{A}$ and denote $\mathbf{C} = \mathbf{U}_c\Sigma_c\mathbf{V}_c^*$, where $\mathbf{U}_c \in \mathbb{K}^{M\times M}$ and $\mathbf{V}_c \in \mathbb{K}^{c\times c}$. Further, let $\mathbf{C}_r \in \mathbb{K}^{r\times c}$ be the matrix made of r rows sampled from $\mathbf{C}$ and denote $\mathbf{C}_r = \mathbf{U}_r\Sigma_r\mathbf{V}_r^*$, where $\mathbf{U}_r \in \mathbb{K}^{r\times r}$ and $\mathbf{V}_r \in \mathbb{K}^{c\times c}$. Then

- SVD is performed for the much smaller matrix $\mathbf{C}_r$, in which $r, c \le K \ll \min\{M,N\}$ and independent of M and N. Due to rapid decay of singular values, only first t_0 columns of $\mathbf{V}_r$ are needed, where t_0 is determined by checking $|\sigma_r^{t_0+1} - \sigma_r^{t_0}| < \epsilon$, where ϵ is a preset accuracy criteria.

- Since $\mathbf{V}_r$ is now available, we can implement $\mathbf{CV}_r$, which will be used to approximate $\mathbf{U}_c\Sigma_c = \mathbf{CV}_c$. Then a QR factorization of $\mathbf{CV}_r$ is computed, i.e., $\mathbf{CV}_r = \tilde{\mathbf{U}}_c\mathbf{R}$, where $\mathbf{R}$ is an upper triangle matrix. The resulting $\tilde{\mathbf{U}}_c$ is considered as the approximation of $\mathbf{U}_c$.

- Finally we take the approximation $\mathbf{U}_A \approx \tilde{\mathbf{U}}_A = \tilde{\mathbf{U}}_c$ and $\mathbf{Ax} \approx \tilde{\mathbf{U}}_A(\tilde{\mathbf{U}}_A^*\mathbf{A})\mathbf{x}$. Note that exact computation of $\tilde{\mathbf{U}}_A^*\mathbf{A}$ still requires $O(t_0MN)$ operations, so the Monte Carlo Basic matrix multiplication algorithm in [7] is adopted, i.e., only c columns from $\tilde{\mathbf{U}}_A^*$ and c rows from $\mathbf{A}$ are sampled and computed, this approximation reduces the complexity of matrix-matrix multiplication to $O(ct_0N)$.

Detailed implementations are given in Algorithm 1, where $\Pi\mathbf{A}$ is represented by orthonormal vectors. Note that overall the random kernel compression will have $O(\max\{M,N\})$ operations.

Algorithm 1 Random kernel compression

Input: matrix $\mathbf{A} \in \mathbb{K}^{M\times N}$, $c, r, \in \mathbb{N}$, such that $1 < c \ll N$ and $1 < r \ll M$. A constant $\epsilon > 0$.

Output: $\sigma_t \in \mathbb{R}^+$, orthonormal vectors $\mathbf{U}_t \in \mathbb{K}^M$, and $\mathbf{V}_t \in \mathbb{K}^N$, for $t = 1, 2..., t_0 \ll \min(M,N)$, such that

$$\mathbf{A} \approx \Pi\mathbf{A} = \sum_{t=1}^{t_0} \sigma_t \mathbf{U}_t \mathbf{V}_t^*. \tag{12}$$

1. Construct matrix $\mathbf{C} \in \mathbb{K}^{M\times c}$:
for t = 1 to c **do**
(a) pick $i_t \in 1, 2, ...N$ randomly using uniform distribution;
(b) set $\mathbf{C}^{(t)} = \sqrt{N/c}\mathbf{A}^{(i_t)}$;
end for
2. Construct matrix $\mathbf{C}_r \in \mathbb{R}^{r\times c}$:
for t = 1 to r **do**
(a) pick $j_t \in 1, 2, ...M$ randomly using uniform distribution;
(b) set $\mathbf{C}_{r(t)} = \sqrt{M/r}\mathbf{C}_{(j_t)}$;
end for
3. Perform SVD on matrix $\mathbf{C}_r$, i.e., $\mathbf{C}_r = \mathbf{U}_r\mathbf{\Sigma_r}\mathbf{V}_r^*$ and denote $\sigma(\mathbf{C}_r)$ as the singular values.
4. Let $t_0 = \min\{r, c, \max\{j, \sigma_j(\mathbf{C}_r)\} > \epsilon\}$:
for t =1 to t_0 **do**
(a) $\sigma_t = \sigma_t(\mathbf{C_r})$;
(b) $\mathbf{U}_t^c = \mathbf{CV}_r$
end for
5. Perform QR decomposition on $\{\mathbf{U}_t^c\}$ to obtain the output orthonormal vectors $\{\mathbf{U}_t\}_{t=1}^{t_0}$.
6. Approximate $\mathbf{V} = \mathbf{A}^*\mathbf{U}_t$ by the Monte Carlo Basic matrix multiplication algorithm in [7], with c columns from $\mathbf{A}^*$ and c rows from $\mathbf{U}_t$;
7. Perform QR decomposition on $\mathbf{V} \in \mathbb{K}^{N\times t_0}$ to obtain the output orthonormal vectors $\{\mathbf{V}_t\}_{t=1}^{t_0}$

Note that in this algorithm, only the small matrix $\mathbf{C}_r$ is stored in memory and

entries of other matrices are calculated on the fly. The idea of Algorithm 1 is similar to the "ConstantTimeSVD algorithm" in [8]. However, in that work, the columns $\mathbf{U}_t$ are directly calculated as $\mathbf{C}\mathbf{V}_r/\sigma_t(\mathbf{C_r})$ and not orthonormal. More importantly, when $\mathbf{V}_r$ is not close enough to $\mathbf{V}_c$ (otherwise efficiency will be compromised), dividing rapidly decaying singular values is not numerically stable. There will be no such issues in Algorithm 1.

3. Analysis of the compression algorithm for well-separated sets. In this section we investigate the error of the random compression operator Π in (12) for $\mathbf{A}$ as generated by Algorithm 1,

$$\|(\mathbf{A}-\Pi\mathbf{A})\mathbf{x}\| \le \|(\mathbf{A}-\Pi\mathbf{A})\|\|\mathbf{x}\|. \tag{13}$$

Unfortunately, a full analysis on $\Pi\mathbf{A}$ with general sampling probability is technically complicated. We would rather consider a theoretically simpler case about

$$\mathbf{A}-\tilde{\Pi}\mathbf{A}, \quad \text{with } \tilde{\Pi}=\mathbf{U}_K\mathbf{U}_K^* \tag{14}$$

where $\mathbf{U}_K$ is the matrix containing only the first K columns of $\mathbf{U}_c$ in $\mathbf{C}=\mathbf{U}_c\Sigma_c\mathbf{V}_c^*$. Matrix $\mathbf{C}$ here is the same as in Algorithm 1, but could be associated with arbitrary sampling probability. Construction of the projector $\tilde{\Pi}$ is equivalent to the "LinearTimeSVD" algorithm in [8]. It is computationally inefficient but theoretically simple. We cite the result from [8]:

THEOREM 3.1 (*Theorem 2 and 3 in [8]*). *Suppose* $\mathbf{A}\in\mathbb{K}^{M\times N}$ *and* $\mathbf{C}\in\mathbb{K}^{M\times c}$ *being the column sampled matrix from* $\boldsymbol{A}$ *as in Algorithm 1. Let* $\tilde{\Pi}$ *be the projector defined in Eq. (14), then*

$$\|\mathbf{A}-\tilde{\Pi}\mathbf{A}\|_F^2 \le \|\mathbf{A}-\mathbf{A}_K\|_F^2 + 2\sqrt{K}\|\mathbf{A}\mathbf{A}^*-\mathbf{C}\mathbf{C}^*\|_F; \tag{15}$$

and

$$\|\mathbf{A}-\tilde{\Pi}\mathbf{A}\|_2^2 \le \|\mathbf{A}-\mathbf{A}_K\|_2^2 + 2\|\mathbf{A}\mathbf{A}^*-\mathbf{C}\mathbf{C}^*\|_2; \tag{16}$$

where $\mathbf{A}_K$ *the best* K*-rank approximation of* $\mathbf{A}$.

Although the theoretical projector $\tilde{\Pi}$ is different from projector Π of (12) in Algorithm 1, we still can obtain meaningful insights about sampling strategies and accuracy of Algorithm 1 by examining Theorem 3.1.

Since $\mathbf{A}$ is already assumed as low-rank, we can only focus on estimating $\|\mathbf{A}\mathbf{A}^*-\mathbf{C}\mathbf{C}^*\|_\xi, \xi=2,F$. We look for a practical sampling probability and derive an error estimate for well-separated source and target points. For simplicity, we assume $a=diam(T)=diam(S)$, $\delta=dist(T,S)$, and $a/\delta\le\eta$ for some $\eta\in(0,1)$.

3.1. Nearly optimal uniform sampling for far field kernel matrices. The low rank property of a matrix $\mathbf{A}$ means most of its columns/rows are linearly dependent, while each column/rows may contribute much differently to the overall matrix property. For example the matrix $A=vv^T$ with $v^T=(1,2,3,....N)$ is only of rank one, but the last column/row is the most important. Therefore, the column/row sampling probability is a critical factor in minimizing the error $\|\mathbf{A}\mathbf{A}^*-\mathbf{C}\mathbf{C}^*\|_\xi$.

The optimal probability of (column) sampling to perform a Monte Carlo matrix-matrix multiplication $\mathbf{A}\mathbf{B}$ is proposed in [7].

DEFINITION 3.2 (Optimal probability). *For* $\mathbf{A}\in\mathbb{K}^{M\times N}, \mathbf{B}\in\mathbb{K}^{N\times P}$, $1\le c\le N$, *and* $\{p_j\}_{j=1}^N$ *such that*

$$p_j=\frac{|A^{(j)}||B_{(j)}|}{\sum_{j'=1}^N|A^{(j')}||B_{(j')}|}, \quad j=1,2,...,N, \tag{17}$$

then for $t = 1$ to c, pick $i_t \in \{1, 2, ..., N\}$ with $\mathbf{Pr}[i_t = j] = p_j, j = 1, 2, ...N$ independently and with replacement. Set $C^{(t)} = A^{(i_t)}/\sqrt{cp_{i_t}}$ and $R_{(t)} = B_{(i_t)}/\sqrt{cp_{i_t}}$, the expectation value $\mathbf{E}\left[\|\mathbf{AB} - \mathbf{CR}\|_F\right]$ is minimized.

Applying this definition to $\mathbf{AA}^*$, and using Lemma 4 in [7], it is easy to conclude that if c columns are sampled, the expectation of error $\|\mathbf{AA}^* - \mathbf{CC}^*\|_F^2$ is minimized as

$$\mathbf{E}\left[\|\mathbf{AA}^* - \mathbf{CC}^*\|_F^2\right] = \frac{1}{c}\left(\|\mathbf{A}\|_F^4 - \|\mathbf{AA}^*\|_F^2\right). \tag{18}$$

However, unless known in advance, utilizing the optimal probability is not practical because its computation is already more expensive than the actual matrix-vector multiplication.

In practice, a nearly optimal probability introduced in [7] will be used for our approach.

DEFINITION 3.3 (Nearly optimal probability). *For the same conditions in Definition 3.2, the probability $\{p_j\}$ is called a nearly optimal probability if*

$$p_j \geq \frac{\beta |A^{(j)}||B_{(j)}|}{\sum_{j'=1}^{N} |A^{(j')}||B_{(j')}|}, \quad j = 1, 2, ..., N, \tag{19}$$

for some $0 < \beta \leq 1$.

With the nearly optimal probability, one has

$$\mathbf{E}\left[\|\mathbf{AA}^* - \mathbf{CC}^*\|_F^2\right] \leq \frac{1}{\beta c}\|\mathbf{A}\|_F^4 - \frac{1}{c}\|\mathbf{AA}^*\|_F^2. \tag{20}$$

Now we will present the following result.

THEOREM 3.4. *For well-separated source and target points, uniform sampling provides a nearly optimal probability.*

Proof. For Eq. (18), the optimal probability is

$$p_j = \frac{|A^{(j)}|^2}{\|A\|_F^2}. \tag{21}$$

Recall entries of matrix $\mathbf{A}$ are $\mathcal{K}(\mathbf{r}_i, \mathbf{r}_j)q_i$ and denote the distance $r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$ for simplicity. Then, for the well-separated source and target points, we have the bound

$$\delta - a \leq r_{ij} \leq \delta + a,$$

since $|\mathcal{K}(\mathbf{r}_i, \mathbf{r}_j)|^2$ is monotonically decreasing with r_{ij}, we have

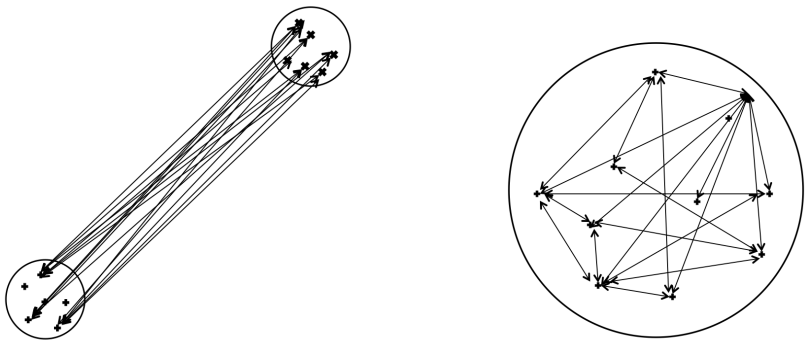
$$p_j = \frac{|A^{(j)}|^2}{\|A\|_F^2} \leq \frac{|\mathcal{K}(\delta - a)|^2 \sum_{i=1}^{M} |q_i|^2}{N|\mathcal{K}(\delta + a)|^2 \sum_{i=1}^{M} |q_i|^2}. \tag{22}$$

Thus, the uniform sampling probability

$$\hat{p}_j = \frac{1}{N} > \frac{|\mathcal{K}(\delta + a)|^2}{|\mathcal{K}(\delta - a)|^2} \frac{|A^{(j)}|^2}{\|A\|_F^2}, \tag{23}$$

is the the nearly optimal probability with $\beta = \dfrac{|\mathcal{K}(\delta + a)|^2}{|\mathcal{K}(\delta - a)|^2} < 1$. ∎

Theorem 3.4 indicates that for a fixed number of samples and without additional computational effort (uniform sampling), we can achieve the nearly optimal accuracy as in estimate (20) in the kernel compression for well-separated source and target points. However, the bound in Eq. (20) depends on parameter β, which is an indicator of how "well" the two sets are separated. In case the two sets "touching" each other, one has $\eta \to 1$ or $\delta - a \to 0$ and hence $\beta \to 0$. As a result, the error bound (20) fails.



(a) (b)

FIG. 1. *(a) well-separated; (b) not well-separated.*

The reason why uniform sampling works can also be illustrated heuristically by Fig.1: When target and source sets are well-separated and have a small diameter-distance ratio η as in Fig. 1(a), if we associate the interaction between target/source points as lines connecting the sources and targets, it can be seen that all interactions are "similar", in terms of direction and magnitude, to each other. Then in this case, columns or row vectors in the matrix have fairly the same contribution, so uniform sampling will perform well with a small error. On the other hand, if target and source points are mixed and belong to the same set as in Fig. 1(b), the interaction lines are rather different from each other. Even the matrix maybe low rank, uniform sampling will yield in uncontrollable error due to the complexity of the "interaction" lines.

In the next subsection we provide a further quantitative analysis of Eqs. (18) or (20) for the separation. We show that even the error (18) or (20) depends on not only sample number c, but also the diameter-distance ratio η.

3.2. Error analysis on target-source separation. Error estimates (18) and (20) provided in [8] are for a general matrix $\mathbf{A}$, stating that the error is small if the samples are large enough. But in practice, the sample number c can not be too large due to efficiency requirement. Here, we give a finer estimate for $\mathbf{A}$ corresponding to well-separated target and source points. We show that for some type of kernels, if the target and source sets are far way enough, the error is small enough regardless of the sample numbers. Since we are more interested in the dependence of error bound on diameter-distance ration, we will consider Eq. (18) for simplicity.

THEOREM 3.5. *Suppose $\mathbf{A} \in \mathbb{K}^{M\times N}$ is the matrix generated from the kernel function $\mathcal{K}(\mathbf{r}, \mathbf{r}')$ satisfying condition (4), and $\mathbf{C} \in \mathbb{K}^{M\times c}$ being the column sampled matrix from $\boldsymbol{A}$. Let $\tilde{\Pi}$ be the orthogonal projector defined in Eq. (14), then*

$$\|\mathbf{A} - \tilde{\Pi}\mathbf{A}\|_F^2 \leq \|\mathbf{A} - \mathbf{A}_K\|_F^2 + 2\sqrt{K}\|\mathbf{A}\mathbf{A}^* - \mathbf{C}\mathbf{C}^*\|_F; \tag{24}$$

or

$$\|\mathbf{A}-\tilde{\Pi}\mathbf{A}\|_2^2 \le \|\mathbf{A}-\mathbf{A}_K\|_2^2 + 2\|\mathbf{A}\mathbf{A}^*-\mathbf{C}\mathbf{C}^*\|_2; \tag{25}$$

and

$$\mathbf{E}\left[\|\mathbf{A}\mathbf{A}^*-\mathbf{C}\mathbf{C}^*\|_F\right] \le C(\delta,a,\tau,k,M,N)\frac{1}{\sqrt{c}}\frac{2\eta}{2-\eta}\|\mathbf{A}\|_F^2, \tag{26}$$

where δ and a are the distance between and diameter for the target and source sets, respectively, $0<\eta<1$ is the diameter-over-distance ratio of target and source sets, and

$$C(\delta,a,\tau,k,M,N) = \left(\frac{\delta}{2}\right)^{-\tau}\frac{(1+2k\delta)}{|\mathcal{K}(\delta+a)|}\frac{\sqrt{M}}{\sqrt{N}}. \tag{27}$$

Proof. Equations (24) and (25) have been stated in Theorem 3.1. To prove (26), write the i-th row of $\mathbf{A}$ as

$$A_{(i)} = q_i(\mathcal{K}(\mathbf{r}_i,\mathbf{r}_1),\mathcal{K}(\mathbf{r}_i,\mathbf{r}_2),...,\mathcal{K}(\mathbf{r}_i,\mathbf{r}_N)), \quad i=1,2,...M. \tag{28}$$

Then, the difference between $q_{i'}A_{(i)}$ and $q_iA_{(i')}$ is, approximated to the first order,

$$\Delta_{ii'} = q_iq_{i'}(\Delta_{ii'}^{(1)},\Delta_{ii'}^{(2)},...\Delta_{ii'}^{(N)}), \tag{29}$$

where

$$\Delta_{ii'}^{(j)} = \mathcal{K}(\mathbf{r}_i,\mathbf{r}_j)-\mathcal{K}(\mathbf{r}_{i'},\mathbf{r}_j) = (\mathbf{r}_i-\mathbf{r}_{i'})\cdot\frac{\partial}{\partial\mathbf{r}}\,\mathcal{K}(\mathbf{r},\mathbf{r}_j)|_{\mathbf{r}=\mathbf{r}_T^*}. \tag{30}$$

Recall assumption in Eq. (4) and condition (7), we have estimate

$$|\Delta_{ii'}^{(j)}| \le \frac{a}{\delta-\frac{a}{2}}\left(1+k\left(\delta+\frac{a}{2}\right)\right)\left(\delta-\frac{a}{2}\right)^{-\tau} \le \frac{2\eta}{2-\eta}(1+2k\delta)\left(\frac{\delta}{2}\right)^{-\tau}. \tag{31}$$

Rewrite Eq. (18) as

$$\begin{aligned}
&\mathbf{E}\left[\|\mathbf{A}\mathbf{A}^*-\mathbf{C}\mathbf{C}^*\|_F^2\right] = \frac{1}{c}\left(\|\mathbf{A}\|_F^4-\|\mathbf{A}\mathbf{A}^*\|_F^2\right)\\
&= \frac{1}{c}\left[\left(\sum_{i=1}^M|A_{(i)}|^2\right)^2-\sum_{i=1}^M\sum_{i'=1}^M\langle A_{(i)},A_{(i')}\rangle^2\right]\\
&= \frac{1}{c}\sum_{i=1}^M\sum_{i'=1}^M\frac{1}{q_i^2q_{i'}^2}\left[\langle q_{i'}A_{(i)},q_{i'}A_{(i)}\rangle\langle q_iA_{(i')},q_iA_{(i')}\rangle-\langle q_{i'}A_{(i)},q_iA_{(i')}\rangle^2\right],
\end{aligned} \tag{32}$$

where $\langle,\rangle$ represents inner product.

Since $q_iA_{(i')} = q_{i'}A_{(i)}+\Delta_{ii'}$, using the linearity of inner product, we have

$$\begin{aligned}
&\langle q_{i'}A_{(i)},q_{i'}A_{(i)}\rangle\langle q_iA_{(i')},q_iA_{(i')}\rangle-\langle q_{i'}A_{(i)},q_iA_{(i')}\rangle^2\\
&= \langle q_{i'}A_{(i)},q_{i'}A_{(i)}\rangle\langle\Delta_{ii'},\Delta_{ii'}\rangle-\langle q_{i'}A_{(i)},\Delta_{ii'}\rangle^2\\
&\le \langle q_{i'}A_{(i)},q_{i'}A_{(i)}\rangle\langle\Delta_{ii'},\Delta_{ii'}\rangle.
\end{aligned} \tag{33}$$

Plugging (33) in (32) and using estimate (31), we arrive at

$$\mathbf{E}\left[\|\mathbf{A}\mathbf{A}^* - \mathbf{C}\mathbf{C}^*\|_F^2\right] \leq \frac{1}{c}\sum_{i=1}^{M}\sum_{i'=1}^{M}\langle A_{(i)}, A_{(i)}\rangle\langle \Delta_{ii'}\Delta_{ii'}\rangle$$

$$\leq \frac{M\|Q\|_2^2}{c}\left(\frac{2\eta}{2-\eta}\right)^2(1+2k\delta)^2\left(\frac{\delta}{2}\right)^{-2\tau}\|\mathbf{A}\|_F^2, \tag{34}$$

where $Q = (q_1, q_2, ...q_M)$ is the density of target points. By Jensen's inequality

$$\mathbf{E}\left[\|\mathbf{A}\mathbf{A}^* - \mathbf{C}\mathbf{C}^*\|_F\right] \leq \frac{\sqrt{M}\|Q\|_2}{\sqrt{c}\|\mathbf{A}\|_F}(1+2k\delta)\left(\frac{\delta}{2}\right)^{-\tau}\frac{2\eta}{2-\eta}\|\mathbf{A}\|_F^2.$$

$$\leq \frac{\sqrt{M}\|Q\|_2(1+2k\delta)\left(\frac{\delta}{2}\right)^{-\tau}}{\sqrt{N}\|Q\|_2\left|\mathcal{K}(\delta+a)\right|}\frac{1}{\sqrt{c}}\frac{2\eta}{2-\eta}\|\mathbf{A}\|_F^2$$

$$= C(\delta, a, \tau, k, M, N)\frac{1}{\sqrt{c}}\frac{2\eta}{2-\eta}\|\mathbf{A}\|_F^2, \tag{35}$$

where $C(\delta, a, \tau, k, M, N) = \left(\frac{\delta}{2}\right)^{-\tau}\frac{(1+2k\delta)}{|\mathcal{K}(\delta+a)|}\frac{\sqrt{M}}{\sqrt{N}}$. ∎

This result indicates that the error of kernel compression also depends on the diameter-distance ratio of the well-separated target and source points. It is important to point out that in the constant $C(\delta, a, \tau, k, M, N)$, wavenumber parameter k is critical to the compression error. A typical example is the Green's function for Helmholtz equation, for which the high frequency problem is still a challenge for any kernel compression algorithm.

4. Hierarchical matrix ($\mathscr{H}$-matrix) structure for general data sets.

For general cases when target and source are not well-separated, in fact typically they belong to the same set, we will partition the whole matrix into blocks, each of which corresponds to a far-field interaction sub-matrix and will have a low-rank approximation, thus the kernel compression algorithm can be applied hierarchically at different scales. The resulting method will be termed "hierarchical random compression method (HRCM)".

4.1. Review of $\mathscr{H}$-matrix.

DEFINITION 4.1 (Hierarchical matrices ($\mathscr{H}$-matrix)). *Let I be a finite index set and P_2 be a (disjoint) block partitioning (tensor or non-tensor) of $I \times I$ and $K \in \mathbb{N}$. The underlying field of the vector space of matrices is $\mathbb{K} \in \{\mathbb{R}, \mathbb{Z}\}$. The set of $\mathscr{H}$-matrix induced by P_2 is*

$$\mathscr{M}_{\mathscr{H},K} := \{\mathbf{M} \in \mathbb{K}^{I\times I} : \textit{each block } \mathbf{M}^b, b \in P_2, \textit{satisfies } rank(\mathbf{M}^b) \leq K\}. \tag{36}$$

Remarks: (1) The index set I can be the physical coordinates of target/source points; we denote a matrix $\mathbf{A}$ as R-K matrix if $rank(\mathbf{A}) \leq K$; (2) A specific $\mathscr{H}$-matrix is defined through 4.1 recursively. Full definition, description, and construction of $\mathscr{H}$-matrices are given in details in [13, 14]. Two simple examples are given as follows; (3) Since $\mathscr{H}$-matrix is recursive, we always assume $\mathbf{A} \in \mathbb{K}^{N\times N}$ and $N = 2^p$ for the following context.

Example One: $\mathbf{A} \in \mathscr{M}_{\mathscr{H},K}$ if either $2^p = K$ or it has the structure

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_{11} & \mathbf{A}_{12}\\ \mathbf{A}_{21} & \mathbf{A}_{22}\end{bmatrix} \text{ with } \mathbf{A}_{11}, \mathbf{A}_{22} \in \mathscr{M}_{\mathscr{H},K} \text{ and R-}K \text{ matrices } \mathbf{A}_{12}, \mathbf{A}_{21}.$$

This is the simplest $\mathscr{H}$-matrix. Such a matrix with three levels of division is visualized in the left of Fig. 2. All blocks are R-K matrices, except the smallest ones.

The next example is more complicated and it includes another two recursive concepts of neighborhood matrix($\mathcal{N}$-type and $\mathcal{N}^*$-type). Construction of this $\mathscr{H}$-matrix includes three steps.

Example Two: (i) Neighborhood matrix of $\mathcal{N}$-type, or $\mathscr{M}_{\mathcal{N},K}$: if either $2^p = K$ or it has the structure

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \text{ with } \mathbf{A}_{21} \in \mathscr{M}_{\mathcal{N},k} \text{ and R-}K \text{ matrices } \mathbf{A}_{11}, \mathbf{A}_{12}, \mathbf{A}_{22}.$$

(ii) Neighborhood matrix of $\mathcal{N}^*$-type, or $\mathscr{M}_{\mathcal{N}^*,K}$: $\mathbf{A} \in \mathscr{M}_{\mathcal{N}^*,k}$ if $\mathbf{A}^* \in \mathscr{M}_{\mathcal{N},K}$.

(iii) $\mathbf{A} \in \mathscr{M}_{\mathscr{H},K}$ if either $2^p = K$ or it has the structure

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \text{ with } \mathbf{A}_{11}, \mathbf{A}_{22} \in \mathscr{M}_{\mathscr{H},K}, \mathbf{A}_{12} \in \mathscr{M}_{\mathcal{N},K}, \text{ and } \mathbf{A}_{21} \in \mathscr{M}_{\mathcal{N}^*,K}.$$

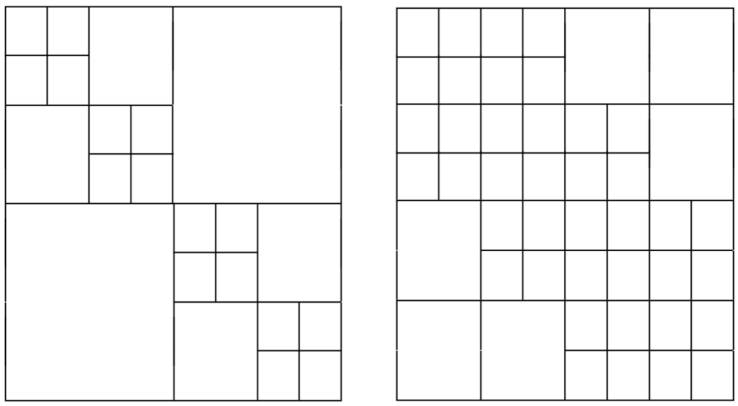


FIG. 2. *Two examples of $\mathscr{H}$-matrix. Left: example one; right: example two.*

Visualization of the second example with three levels of division is given in the right subfigure of Fig. 2. Similarly, only the larger blocks are R-K matrices.

With such a decomposition, matrix-vector product will be performed as

$$\mathbf{A}\mathbf{x} = \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 + \mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2. \tag{37}$$

where $\mathbf{x}^T = (\mathbf{x}_1^T, \mathbf{x}_2^T)$. For $\mathbf{A}$ in Example One, random compression can be immediately applied to $\mathbf{A}_{12}\mathbf{x}_2$ and $\mathbf{A}_{21}\mathbf{x}_1$, while recursive division and random compression need to be implemented on $\mathbf{A}_{11}\mathbf{x}_1$ and $\mathbf{A}_{22}\mathbf{x}_2$, until a preset minimum block is reached where direct matrix-vector multiplication is used.

But in Example Two, none of the four terms in Eq. (37) is R-K matrix ready for compression. Each $\mathbf{A}_{ij}$ needs to be further divided and investigated. For instance, $\mathbf{A}_{12} \in \mathscr{M}_{\mathcal{N},K}$ by definition (iii), then by (i), $\mathbf{A}_{12_{11}}$, $\mathbf{A}_{12_{12}}$, and $\mathbf{A}_{12_{22}}$ are R-K matrices and the compression algorithm can be applied. In contrast, $\mathbf{A}_{12_{21}}$ needs to be further divided.

Matrices in Example One and Two can be understood as interactions of target/-source points along a 1D geometry. Example One indicates that interactions of all

subset of I can be approximated as low-rank compressions except self-interactions of the subsets, which requires further division. While the blocks in Example Two require further division for both self-interacting and immediate neighboring subsets of I.

Structures of $\mathscr{H}$-matrices for target/source points in high-dimensional geometry are much more complicated. It is difficult to partition them into blocks as shown in Fig 2. Instead, we construct the $\mathscr{H}$-matrix logically through a partition tree of I and the concept of admissible clusters.

4.2. Tree structure of $\mathscr{H}$-matrix for two dimensional data. We illustrate algorithms in two-dimensional (2D) case. Here the dimension refers to the geometry where the target and source points are located instead of the dimension of the kernel function. For simplicity, let $\Omega = [0, L] \times [0, L]$ and consider a regular grid index

$$I = \{(i, j) : 1 \le i, j \le N_1\}, \quad N_1 = 2^p. \tag{38}$$

Each index $(i, j) \in I$ is associated with the square

$$X_{ij} : \{(x, y) : (i-1)h \le x \le ih, (j-1)h \le y \le jh\}, \quad h = L/N_1, \tag{39}$$

where a certain amount of target/source points are located. The partitioning $T(I)$ of I uses a quadtree, with children (or leaves):

$$t^l_{\alpha,\beta} := \{(i, j) : 2^{p-l}\alpha + 1 \le i \le 2^{p-l}(\alpha + 1), 2^{p-l}\beta + 1 \le j \le 2^{p-l}(\beta + 1)\}, \tag{40}$$

with $0 \le \alpha, \beta \le 2^l - 1$ belong to level $0 \le l \le p$.

We consider a target quadtree I_t and a source quadtree I_s, which could be same or different. Then blocks of interaction matrix are defined as a block $b = (t_1, t_2) \in T(I_t \times I_s)$, where $t_1, t_2 \in I_t \times I_s$ belong to the the same level l. Then, following Eq. (2)-(3), we define the diameters and distances of t_1 and t_2, and the admissibility condition

$$\max\{diam(t_1), diam(t_2)\} \le \eta dist(t_1, t_2), \quad 0 < \eta < 1, \tag{41}$$

for the block $b = (t_1, t_2)$. A block b is called admissible or an admissible cluster if either b is a leaf or the admissibility condition holds. If b is admissible, no matter how many points are in t_1 and t_2, the block matrix $\mathbf{M}^b$, consisting of entries from the original matrix $\mathbf{M}$ with row indices t_1 and column indices t_2, has rank up to K, thus low rank approximation algorithms are used. Otherwise, both t_1 and t_2 will be further partitioned into children to be further investigated. And the above process is implemented, recursively.

Figure 3(a) shows the index set I, where black solid, gray solid and gray lines are for partition at level $l = 1, 2, 3$, respectively. For different values of η, admissible clusters are different for a given child. For the square marked with star in Fig 3(a), if $\eta = \sqrt{2}/2$, the non-admissible clusters are itself and the eight immediate surrounding squares. While for $\eta = 1/2$, any squares within the the red lines are non-admissible.

Figure 3(b) displays the quadtree that divides each square. Four children of each branch are labeled as $0, 1, 2, 3$ and ordered counter-clock wisely. If there are $N = 4^p$ target (source) points, the depth of the target (source) tree is $p - p_0$, where p_0 is the number of points in each leaf, or direct multiplication is performed when matrix size is down to $4^{p_0} \times 4^{p_0}$.

Figure 4 illustrates the admissible and non-admissible clusters. Initially the 2D set is partitioned into four subdomain A, B, C and D. Any two of the subdomains

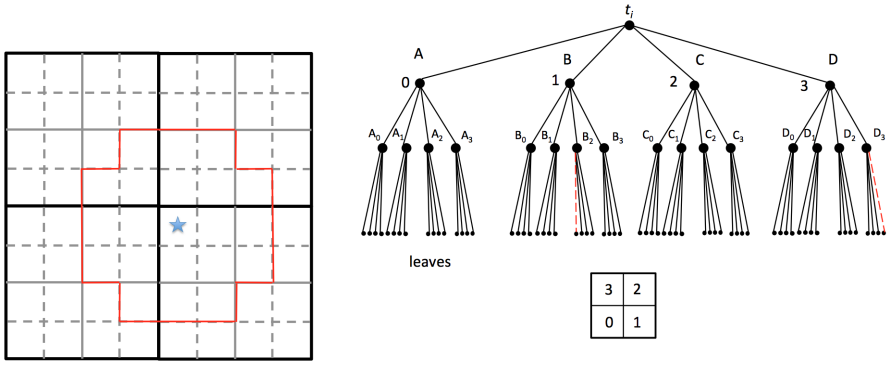


FIG. 3. *Illustration of the index set I: (a) admissible cluster for the starred square; (b) quadtree structure.*

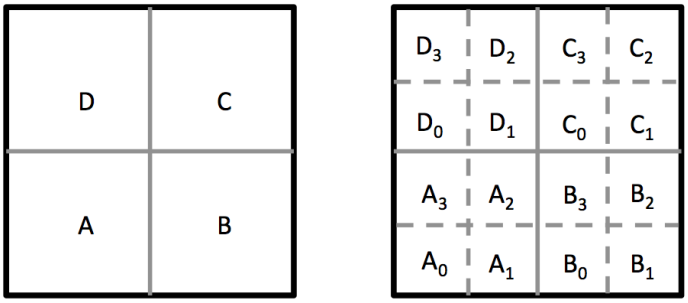


FIG. 4. *Illustration of the partitioning of admissible clusters*

are non-admissible for $\eta = \sqrt{2}/2$. Then each of them are further divided into four children domain, as label on the right of Fig 4, among which the interactions are examined. For example, the interactions of A and B can be viewed as the sum of interactions of A_i and B_j, $i, j = 0, 1, 2, 3$. If we take $\eta = 1/2$, only A_3 and B_1, and A_0 and B_2 are admissible clusters. But if $\eta = \sqrt{2}/2$, only A_2 and B_0, and A_1 and B_3 are non-admissible pairs. For both values of η, only A_2 and C_0 are non-admissible clusters in the interactions of A and C. Self-interactions, such as interaction between A and A, can be viewed as the same process of interactions among A, B, C and D, but for A_0, A_1, A_2, and A_3.

Direct and low rank approximation of matrix-vector multiplications are implemented on the quadtree. To perform the algorithms, all the index in I is ordered as $i = \{0, 1, ...N-1\}$ with $N = 4^p$. Note that there are totally 4^{p-l} points in each child/leaf at level l. Matrix column (row) sampling is achieved through sampling of children from level $l+1$ to level p. For example, cluster $b = (B_2, D_3)$ in Fig. 3 (b) is admissible and assume it is at level l. If we generate $s_i \in \{0, 1, 2, 3\}, i = l+1, ..p$ randomly and choose only the s_i-th child of B_2 (red dash line) at i-th level, then one row sampling of the corresponding (block) kernel matrix is accomplished assuming $B_2 \in I_t$. Similarly column sampling is the child-picking process on $D_3 \in I_s$.

The full HRCM algorithms are summarized as in the following:

Algorithm 2 HRCM: Direct product on source and target quadtrees

subroutine name: DirectProduct(∗target, ∗source, int level)

Input: Root pointers of source and target quadtree information for $\mathbf{A}$ and $\mathbf{x}$. Current level l and maximum level p.

Output: Product $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{y}$ is stored in the target quadtree.

```
if level == maxlevel then
    perform and scalar product, and return
else
    for j = 0; j < 4; j++ do
        for i = 0; i < 4; i++ do
            DirectProduct(target->child[j], source->child[i], level +1)
        end for
    end for
end if
```

Algorithm 3 HRCM: Low-rank product on source and target quadtrees

subroutine name: LowRankProduct(∗target, ∗source, int level)

Input: Root pointers of source and target quadtree information for $\mathbf{A}$ and $\mathbf{x}$. Current level l and maximum level p.

Output: $\mathbf{y} \approx \sum_{t=1}^{t_0} \sigma_t \mathbf{U}_t \mathbf{V}_t^* \mathbf{x}$, where $\mathbf{y}$ is stored in the target quadtree.

1. Column sampling: On the source tree, pick the "random path" from level l to maxlevel p by only randomly choosing one child from each level;
2. Row sampling: On the target tree, pick the "random path" from level l to maxlevel p by only randomly choosing one child from each level;
3. Extract matrix entries from the source and target tree by the column/row sampling; perform Algorithm 1
4. Instore $\mathbf{y}$ into the target tree with root ∗target.

4.3. Efficiency analysis. It is easy to perform efficiency analysis of the hierarchical kernel compression method by constructing an interaction pattern tree. With $\eta = \sqrt{2}/2$, all the non-admissible clusters can be classified into three interaction patterns: the self-, edge-contact, and vertex-contact interactions. And these interactions are labeled as S, E and V as shown in Fig. 5. Assume it is currently level l and those target and source boxes need to be further divided into four children in level $l+1$. In level $l+1$, those children form 16 interactions that fall into the S, E, V patterns. It is easy to check that from level l to level $l+1$, as displayed in Fig 5, S-interaction forms 4 S-, 8 E- and 4 V-interactions at level $l+1$. On the other hand, E-interaction forms 2 E-interaction, 2 V-interactions and 12 admissible clusters for which low-rank approximation (LR) applies. Additionally, V-interaction forms 1 V-interactions and 15 LR approximations.

Complexity of all direct calculations. We assume the direct computation is implemented when the matrix scale is down to $4^{p_0} \times 4^{p_0}$. So we start from S-interaction as the root at level l and just need to count how many E- and V-interactions at level $p-p_0-1$ are generated. From Fig. 5 we can see that at level $l+1$, S generates eight Es and four Vs. And afterwards, from level $l+1$ to level $p-p_0-1$, each E generates two Es and two Vs, and each V generates just one E. Thus, the total number of generated

Algorithm 4 HRCM: $\mathscr{H}$-matrix product on source and target quadtrees

subroutine name: HmatrixProduct(∗target, ∗source, int level)

Input: Root pointers of source and target quadtree for $\mathbf{A}$ and $\mathbf{x}$. Current level l and maximum level p.

Output: $\tilde{\mathbf{y}} \approx \mathbf{y} = \mathbf{A}\mathbf{x}$, where $\tilde{\mathbf{y}}$ is stored in the target quadtree.

```
if matrix small enough then
    DirectProduct(∗target, ∗source, level)
else
    if clusters rooted from ∗target, ∗source are admissible then
        LowRankProduct(∗target, ∗source, level)
    else
        for j = 0; j < 4; j++ do
            for i = 0; i < 4; i++ do
                HmatrixProduct(target->child[j], source->child[i], level +1)
            end for
        end for
    end if
end if
```

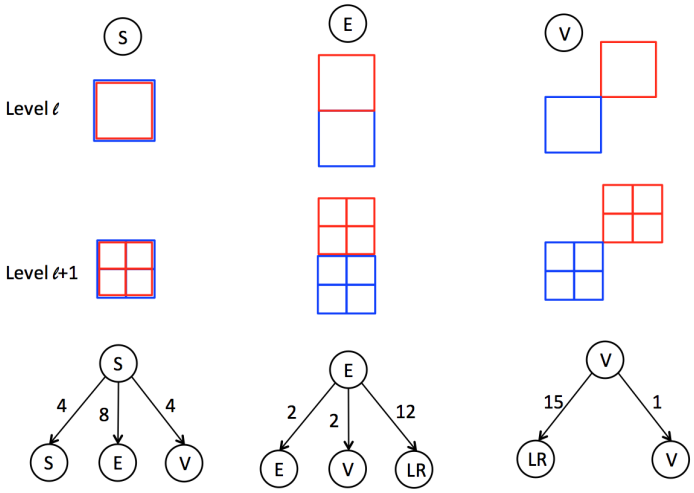


FIG. 5. *Evolution of non-admissible clusters. S: self-interaction clusters; E: clusters touch by edge; V: clusters touch by vertex; LR: admissible clusters with low-rank approximation.*

E and V and level $p - p_0 - 1$ from the S at level l is

$$\underbrace{8 \cdot 2^{p-p_0-1-l-1}}_{\text{generated Es}} + \underbrace{8 \cdot 2 \cdot 1^{p-p_0-1-l-2} + 4 \cdot 1^{p-p_0-1-l-1}}_{\text{generated Vs}}. \tag{42}$$

Since there are 4^l S-interactions at level l, the total complexity for performing direct computation is

$$16 \cdot O((4^{p_0})^2) \left[\sum_{l=0}^{p-p_0-3} 4^l (8 \cdot 2^{p-p_0-l-2} + 8 \cdot 2 + 4) + 4^{p-p_0-2} \cdot 16 \right] = O(4^p) = O(N). \tag{43}$$

Complexity of all low-rank compressions. We follow the similar strategy and count the low rank interactions (LR) generated from S at level l to level $p-p_0-1$. Again we start from S as the root at level l and according to Fig. 5, the resulting E and V start to generate LR at the $(l+2)$-th level and continue to the last level. At level $l+1$, eight Es are generated, each of which generates 12 LRs at level $l+2$; at the same time, four Vs are generated and each of them generates 15 LRs at level $l+2$. Then at level $l+2$, totally $8\times 12+4\times 15$ LRs are generated. Recall at level l the complexity of performing LR is $O(4^{p-l})$, so the complexity of performing LR starting from S at level l is

$$\underbrace{(8\cdot 12+4\cdot 15)O(4^{p-l-2})}_{\text{from E and V at l+2 level}}+\underbrace{\sum_{l'=l+3}^{p-p_0-1} 8\cdot 2^{l'-l-2}\cdot 12\cdot O(4^{p-l'})}_{\text{from all the E to the bottom}}$$

$$+\underbrace{\sum_{l'=l+3}^{p-p_0-1} 8\cdot 2\cdot 1^{l'-l-3}\cdot 15\cdot O(4^{p-l'})+\sum_{l'=l+3}^{p-p_0-1} 4\cdot 1^{l'-l-2}\cdot 15\cdot O(4^{p-l'})}_{\text{from all the V to the bottom}}, \tag{44}$$

where the latter three terms in (44) account for LR generated by E and V from level $l+3$ all down to level $p-p_0-1$. Note that the first item in (44) dominates and there are 4^l S-interactions in level l. The total complexity of low-rank approximation in the HRCM is in the order of

$$\sum_{l=0}^{p-p_0-1} 4^l O(4^{p-l-2}) = O(p\times 4^p) = O(N\log N). \tag{45}$$

Combining Eqs. (43) and (45), the complexity of the HRCM is $O(N\log N)$.

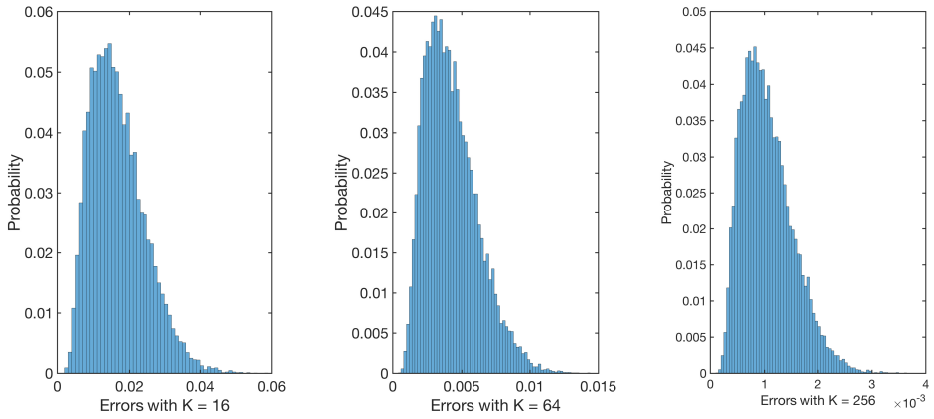
5. Numerical results. In this section, we present the statistical analysis, accuracy and efficiency of the proposed HRCM in 2D computations. For all the following simulations, we take $N=4^p$ target/source points uniformly distributed in square domains of length L. In the random kernel compression Algorithm 1, the total numbers of sampled columns and rows are denoted as $c=r=K=4^{p_c}$, respectively.

5.1. Uncertainty quantification. In this section, we study the statistical properties of the single-realization and multiple-realization of HRCM in kernel summations. In order to perform the algorithm with large number of replications, we take a matrix with moderate size $N=4^7$ ($p=7$), and pick $K=16$, 64, and 256, or $p_c=2,3,4$. The 16384 target and source points are distributed in two squares with length $L=8$ that are separated 16 apart. The kernel function is taken as $\mathcal{K}(\mathbf{r}_i,\mathbf{r}_j)=e^{-ikR}/R$ with $k=0.5$.

Single-realization of HRCM: Given sample size K, one can apply the HRCM algorithm by one realization to obtain the compressed matrix and continue to next implementations such as kernel summation. This treatment gives the best algorithm efficiency but we concern the reliability. The algorithm has been replicated by $N_s=20,000$ times and the corresponding relative errors of single realization are displayed as histograms in Fig. 6 (a) for $K=16$, Fig. 6 (b) for $K=64$, and Fig. 6 (c) for $K=256$.

Define the sample mean error of single realization (sMESR) as

$$\text{sMESR}=\frac{1}{N_s}\sum_{s=1}^{N_s}\frac{\|(\mathbf{A}-\Pi_s(K)\mathbf{A})\mathbf{x}\|_2}{\|\mathbf{A}\mathbf{x}\|_2}. \tag{46}$$



(a) (b) (c)

FIG. 6. *Histograms of relative errors of* 20,000 *single-realizations of matrix compression. (a) With* $K = 16$*, sample mean* 1.699×10^{-2}*, variance* 6.002×10^{-5}*,* 95^{th} *percentile* 3.118×10^{-2}*; (b) With* $K = 64$*, sample mean* 4.213×10^{-3}*, variance* 3.694×10^{-6}*,* 95^{th} *percentile* 7.84×10^{-3}*; (c) With* $K = 256$*, sample mean* 1.056×10^{-3}*, variance* 2.250×10^{-7}*,* 95^{th} *percentile* 1.928×10^{-3}*.*

where $\Pi_s(K)$ is the s-th realization of the compression projector. The quantitative results corresponding to Fig. 6 are summarized in Table 1. We conclude that a single

TABLE 1
Statistics of single realization of HRCM with 20,000 replications.

	sMESR	Variance	95^{th} percentile
$K = 16$	1.699E-2	6.002E-5	3.118E-2
$K = 64$	4.213E-3	3.694E-6	7.840E-3
$K = 256$	1.056E-3	2.250E-7	1.928E-3

realization of HRCM can provide reliable results: for very small sample size K, it offers two-digits relative errors with high confidence. And when K is increased, the HRCM shows convergence with respect to K at the same percentile of confidence.

Random versus Deterministic sampling: Secondly, we illustrate the importance of *random sampling*, although a single-realization of HRCM can be trusted. In Algorithm 4.2, sampling is achieved by randomly choosing one child from the target/source tree. For comparison, we also perform the HRCM by deterministically picking a specific child each time. This is equivalent to sample columns/rows from the matrix evenly according to indices, e.g., the iN/K-th columns/rows ($i = 0, 1, 2, ..K - 1$) are picked. The corresponding relative errors of single-realization of HRCM with such sampling strategies are listed in Table 2. Patterns I, II, III, and IV correspond to the situations that each time the j-th ($j = 0, 1, 2, 3$) child is picked. It clearly indicates that the errors are larger than the sMESR and even above the 95-th percentile of the HRCM with random samples. So we can conclude that it is statistically significant that deterministic sampling will yield larger errors in the HRCM.

Multiple-realization of HRCM: To improve the accuracy of HRCM with a given K, one can also apply it multiple times, take average of the compressed matrix and then implement the kernel summation. At last, we present the efficiency of the

TABLE 2
Errors of the HRCM with some deterministic samples

Patterns	I	II	III	IV
$K = 16$	3.080E-2	9.515E-2	4.320E-2	9.883E-2
$K = 64$	1.600E-2	4.218E-2	1.905E-2	4.140E-2
$K = 256$	7.323E-3	1.904E-2	8.362E-3	1.956E-2

multiple-realization of HRCM. Define the error of multiple realization (EMR):

$$\text{EMR} = \frac{\|(\mathbf{A} - \frac{1}{N_s}\sum_{s=1}^{N_s} \Pi_s(K)\mathbf{A})\mathbf{x}\|_2}{\|\mathbf{A}\mathbf{x}\|_2}, \tag{47}$$

we calculate the EMR with different number of N_s (from $N_s = 5$ to $10,000$) and display the results in Fig. 7. It can be concluded that the EMR converges in the scale of $1/\sqrt{N_s}$ and reaches an equilibrium state (expectation value) depending on K.

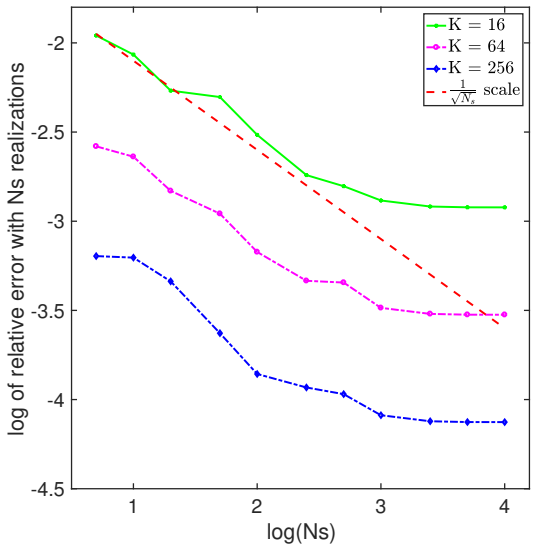


FIG. 7. *Convergence of EMR with respect to N_s.*

By comparing statistics of the sMESR and EMR we can see that one can trust the one-time realization of HRCM, which is especially useful for some situations such as the positions of target/source points are dynamic so everything needs to be computed on the fly. On the other hand, averaged matrix compressions with multiple realizations of HRCM will improve accuracy with convergence order roughly as $1/\sqrt{N_s}$. Since the HRCM is applied multiple times, extra computational efforts are committed. But this treatment is useful if the compressed matrix can be stored and used repeatedly, such as iterative method for solving linear systems.

In the following numerical results regarding accuracy and efficiency of HRCM, we will only consider the single realization.

5.2. Accuracy and efficiency for well-separated sets. First, we investigate the decay of singular values for the kernel matrix formed by the well-separated target

and source points. The kernel function is taken as $\mathcal{K}(\mathbf{r}_i, \mathbf{r}_j) = e^{-0.01R}/R$ with $L = 8$, and the SVD of relatively small matrices with $N = 1024$ are calculated, with diameter/distance ratios being $\eta = 0.5, 0.36$, and 0.25. Logarithmic values of the first 18 singular values for each case are displayed in Fig. 8 (a). It clearly shows that singular values of the matrix decay faster as the corresponding target and source sets are further away. For the fixed $\eta = 0.5$, approximated singular values from randomly sampled matrix $\mathbf{C}_r \in \mathbb{R}^{K\times K}$, with $K = 4^2, 4^3$ and 4^4 are presented in Fig. 8 (b). The relative error with respect to the largest singular value is small enough after several singular values even for a very small amount of samples.

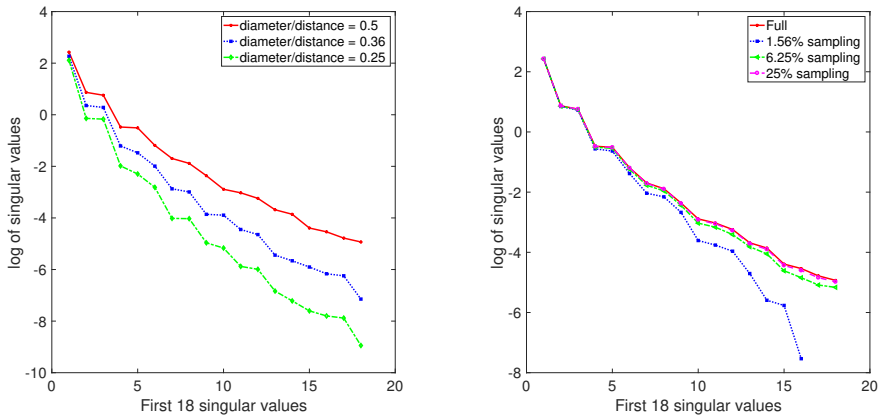


FIG. 8. *(a) Singular values for matrices from well-separated points (1024 targets and 1024 source points) with various cluster diameter/distance ratios; (b) singular values comparison between different amounts of matrix samplings:* $K = 16, 64, 256$ *against* $N = 1024$.

Next, we check the the algorithm accuracy. A total of N target points and source points are uniformly assigned in two 8×8 boxes with centers 16 units apart. Then, the direct multiplication (1) and Algorithm 1 are performed with parameter $c = r = K$ and $\epsilon = 1.0\times 10^{-8}$. For each comparison, sMESR and variance of errors are calculated $N_s = 20$. Errors and variances for kernels $\mathcal{K}(\mathbf{r}, \mathbf{r}') = \log\left(\sqrt{(x-x')^2 + (y+y')^2}\right) - \log\left(\sqrt{(x-x')^2 + (y-y')^2}\right)$ and $\mathcal{K}(\mathbf{r}_i, \mathbf{r}_j) = \exp(-0.01R)/R$ are displayed in Tables 3-4, respectively, with various N and K and $\eta = 0.5$. We can clearly observe the convergence of the mean errors against K in these tables.

Based on the numerical results from Tables 3-4, we conclude that that, given the fixed diameter/distance ratio of the target/source point sets, the accuracy of the low-rank compression algorithm does not depend significantly on the total number N but on the sample number K. It suggests that in computational applications, as long as two boxes are admissible clusters, it does not matter how many target/source points are in them, the algorithm accuracy is mainly controlled by the diameter/ration distance and number of row/column samples.

Table 5 summarizes the corresponding computational time in seconds for the matrix-vector product, for direct computation and the low-rank compression method. If the target and source points are well-separated, the algorithm is very efficient and the computational time is linear both in sample size K and matrix size N. CPU times for the two kernel functions are similar so only one of them is presented.

TABLE 3
Errors and variances for a pair of well-separated target/source point sets. Kernel function $\mathcal{K}(\mathbf{r},\mathbf{r}') = \log(\sqrt{(x-x')^2+(y+y')^2}) - \log(\sqrt{(x-x')^2+(y-y')^2})$

	$N = 1,024$	$N = 4096$	$N = 16,384$	$N = 65,536$	$N = 262,144$
$K = 16$					
sMESR	2.79E-2	3.07E-2	3.51E-2	3.78E-2	4.01E-2
Variance	3.58E-4	4.13E-4	5.38E-4	6.32E-4	6.95E-4
$K = 64$					
sMESR	8.06E-3	8.54E-3	9.70E-3	9.84E-3	1.01E-2
Variance	5.46E-6	5.89E-6	4.92E-6	6.27E-6	6.18E-6
$K = 256$					
sMESR	2.25E-3	2.39E-3	2.52E-3	2.90E-3	2.75E-3
Variance	4.76E-6	4.71E-6	5.37E-6	5.63E-6	5.86E-6

TABLE 4
Errors and variances for a pair of well-separated target/source point sets. Kernel function $\mathcal{K}(\mathbf{r}_i,\mathbf{r}_j) = \exp(-0.01R)/R$

	$N = 1,024$	$N = 4096$	$N = 16,384$	$N = 65,536$	$N = 262,144$
$K = 16$					
sMESR	2.67E-2	3.39E-2	3.07E-2	3.02E-2	3.51E-2
Variance	7.51E-4	4.44E-4	6.28E-4	6.62E-4	9.95E-4
$K = 64$					
sMESR	7.46E-3	7.58E-3	6.70E-3	8.51E-3	8.40E-3
Variance	1.41E-5	2.78E-5	3.89E-5	4.17E-5	4.86E-5
$K = 256$					
sMESR	1.62E-3	1.85E-3	1.92E-3	2.10E-3	2.30E-3
Variance	1.76E-6	1.47E-6	1.37E-6	3.53E-6	3.53E-6

Figure 9 shows the algorithm error against the diameter-distance ratios with $N = 262,144$ and different values of K. As expected, the relative error decays as η increases. This graph is for kernel $\mathcal{K}(\mathbf{r}_i,\mathbf{r}_j) = \exp(-0.01R)/R$, the one for kernel $\mathcal{K}(\mathbf{r}_i,\mathbf{r}_j) = \log(R)$ is similar.

TABLE 5
CPU time (second) comparison for well-separated target and source points.

	$N = 1,024$	$N = 4096$	$N = 16,384$	$N = 65,536$	$N = 262,144$
Direct	0.047	0.75	12	204	3,264
$K = 16$	0.01	0.039	0.16	0.625	2.5
$K = 64$	0.04	0.16	0.66	2.6	12.0
$K = 256$	0.18	0.8	2.7	12.5	47.0

5.3. Accuracy and efficiency for a common source and target set . Next, we test the accuracy and efficiency of the HRCM for target and source points in the same set. In total $N = 4^p$ points with $p = 6, 7, 8, 9, 10, 11$ are uniformly distributed in the domain $[0, 8] \times [0, 8]$. For best computation efficiency, we only present the results with $K = 16$ and 64. Note that it has been found that once sample size K is fixed, the accuracy of the low-rank compression algorithm for a pair of admissible cluster

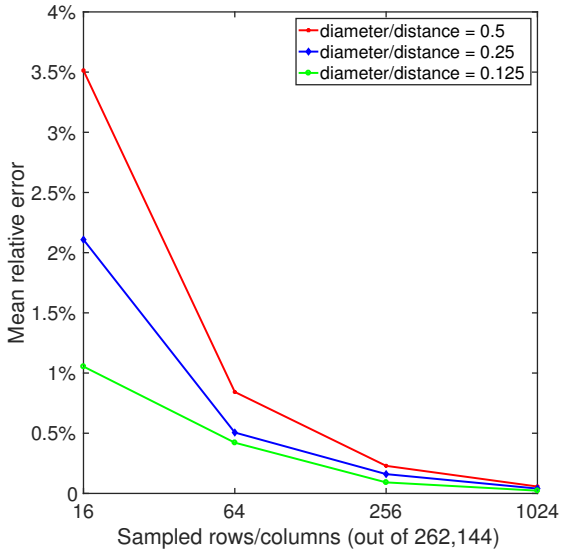


FIG. 9. *Error of low-rank compression algorithm against diameter-distance ratio η.*

does not change too much regardless of number of points in them.

The error for kernel $\mathcal{K}(\mathbf{r}_i, \mathbf{r}_j) = \exp(-0.01R)/R$ with different K and N are summarized in Table 6. Note these values are generally smaller than those in Table 4. Because Table 4 is for a single pair of well-separated target/source sets with diameter-distance $\eta = 0.5$. But in the HRCM there exist a mixture of η with $\eta = 0.5$ as the largest value. Similar convergence of error with respect to K is shown in the table.

TABLE 6
Accuracy of the HRCM for kernel summation with $\mathcal{K}(\mathbf{r}_i, \mathbf{r}_j) = \exp(-0.01R)/R$

Matrix size	$N = 16,384$	$N = 65,536$	$N = 262,144$	$N = 1,048,576$
$k = 16$				
sMESR	2.87E-3	3.32E-3	3.46E-3	3.53E-3
Variance	6.82E-7	7.32E-7	7.65E-7	8.30E-7
$k = 64$				
sMESR	6.09E-4	7.43E-4	6.26E-4	7.32E-4
Variance	7.03E-8	6.49E-8	6.63E-8	7.56E-8

The efficiency of the HRCM with such a kernel function is presented in Fig. 10 as log-log CPU time and matrix size N. For better comparison, the curves of CPU time for the direct method and an ideal $O(N \log N)$ scale are also displayed. For HRCM with $K = 16$ and $K = 64$, the curves are almost parallel to the ideal $O(N \log N)$ scale. Combining Fig. 10 and Table 6, we can conclude that the break-even point of the HRCM comparing to the direct method with three or four digits in relative error is N slightly larger than 10^4 for this screened Coulomb potential. If higher accuracy is desired, one needs to increase number of K and the break-even point will be larger.

5.4. Wave number k dependence. In this section, we will study the performance of the HRCM for compressing the kernel matrices for wave interactions. We

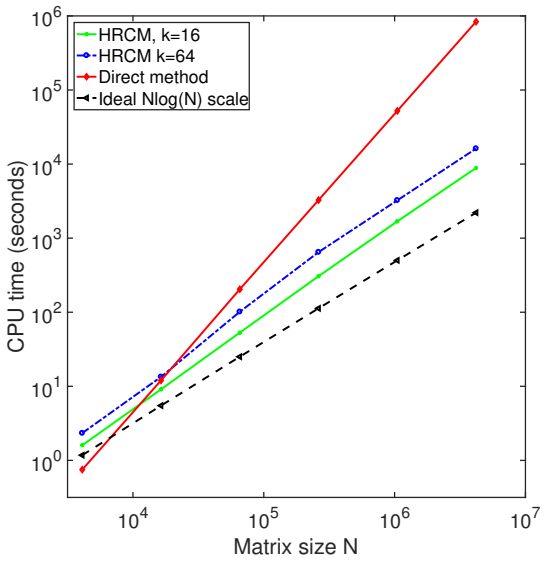


FIG. 10. *CPU costs of HRCM with $K = 16$ and $K = 64$, for various matrix sizes. For comparison, the CPU time for direct method is shown in red and the ideal $N \log(N)$ curve is in black.*

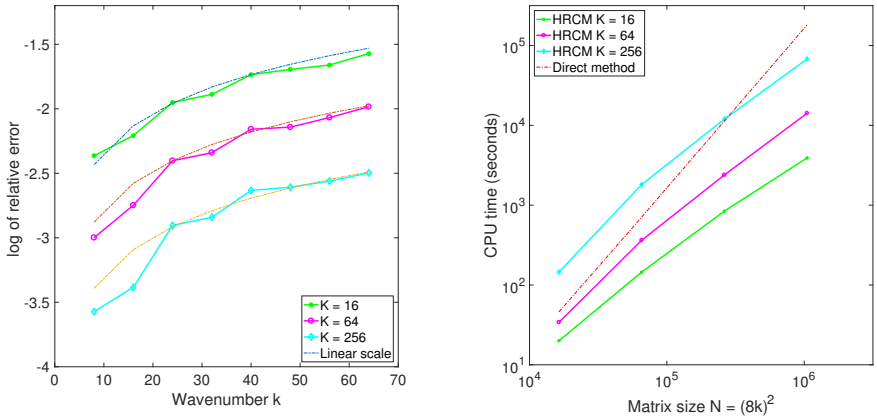
consider the Green's function for the 3-D Helmholtz equation, $\mathcal{K}(\mathbf{r}_i, \mathbf{r}_i) = \exp(-ikR)/R$, when the wavenumber k appears in the error estimate constant C in (27). In the following results, we take N points in the domain $[0, 2\pi]^2$, so the wavenumber k is the same as the number of wavelengths along each direction of the domain. For small value of k, the performance of HRCM is similar to the cases of non-oscillatory screened Coulomb kernels. Errors and variances of the HRCM for this kernel with $k = 0.25, k = 0.5$ and $k = 1$ are presented in Tables 7, where the relative errors are all below 0.2% for small sample size $K = 16$ or 64.

For wave problems with larger wavenumber k, one needs N large enough and proportional to k in order to resolve the wave structure of the numerical solution of the Helmholtz equation. In the following simulations, we use 8 points per wavelength in each directions of the domain and implement the HRCM for wave number k up to 64. The relation of errors and wavenumber k is revealed in Fig. 11 (a). For each fixed sample size K, the relation is roughly linear as indicated by the error analysis in Eq. (35). As a consequences, large sample size K is required to maintain a desired accuracy. For example, with relative error 1%, the sample size $K = 16$ can only handle the wavenumber k up to 24, and if the relative error 0.3% is desired, one has to take the sample size $K = 256$ for wavenumber $k = 64$. On the other hand, large sample size will undermine the efficiency of the HRCM. For sample size $K = 16$ and 64, the HRCM is more efficient than the direct method, while for $K = 256$, the HRCM is only superior when N exceeds 10^6, as shown in Fig. 11 (b).

6. Conclusion and discussion. Kernel summation in large scale is a common challenge in a wide range of applications, from problems in computational sciences and engineering to statistical learning. In this work, we have developed a novel hierarchical random compression method (HRCM) to tackle this difficulty. The HRCM is a fast Monte-Carlo method that can reduce computation complexity from $O(N^2)$

TABLE 7
Accuracy of the HRCM for $\mathcal{K}(\mathbf{r}_i, \mathbf{r}_j) = \dfrac{\exp(-ikR)}{R}$ *with small* $k = 0.25, 0.5, 1$.

Matrix size	$N = 16,384$	$N = 65,536$	$N = 262,144$	$N = 1,048,576$
$K = 16, k = 0.25$				
sMESR	2.56E-3	2.68E-3	2.71E-3	2.89E-3
Variance	6.11E-7	3.56E-7	1.35E-7	1.01E-7
$K = 64, k = 0.25$				
sMESR	5.42E-4	5.51E-4	5.57E-4	5.89E-4
Variance	1.43E-8	7.69E-8	2.18E-9	1.32E-9
$K = 16, k = 0.5$				
sMESR	2.86E-3	2.98E-3	3.17E-3	3.65E-3
Variance	5.97E-7	6.26E-7	6.54E-7	7.83E-7
$K = 64, k = 0.5$				
sMESR	6.91E-4	6.99E-4	7.95E-4	9.01E-4
Variance	2.55E-8	3.61E-8	6.21E-8	7.53E-8
$K = 16, k = 1$				
sMESR	5.36E-3	5.64E-3	6.23E-3	7.47E-3
Variance	2.12E-6	3.26E-6	4.25E-6	6.37E-6
$K = 64, k = 1$				
sMESR	1.12E-3	1.37E-3	1.51E-3	1.86E-3
Variance	3.48E-7	4.16E-7	6.09E-7	1.11E-6



(a) (b)

FIG. 11. *(a) Relative errors and (b) Efficiency of HRCM agains wavenumber k.*

to $O(N \log N)$ for a given accuracy. The method can be readily applied to iterative solver of linear systems resulting from discretizing surface/volume integral equations of Poisson equation, Helmholtz equation or Maxwell equations, as well as fractional differential equations. It also applies to machine learning methods such as regression or classification for massive volume and high dimensional data.

In designing HRCM, we first developed a random compression algorithm for kernel matrices resulting from far-field interactions, based on the fact that the interaction

matrix from well-separated target and source points is of low-rank. Therefore, we could sample a small number of columns and rows from the large-scale matrix, independent of matrix sizes and only dependent on the separation distance between source and target locations, and then perform SVD on the small matrix, resulting in a low-rank approximation to the original matrix. A key factor in the HRCM is that a uniform sampling, implemented without cost of computing the usual sampling distribution based on the magnitude of sampled columns/rows, can yield a nearly optimal error in the low-rank approximation algorithm. The HRCM is kernel-independent without the need for analytic expansions of the kernels. Furthermore, an error bound of the algorithm with some assumption on kernel function was also provided in terms of the smoothness of the kernel, the number of samples and diameter-distance ratio of the well-separated sets.

For general source and target configurations, we utilized the concept of $\mathscr{H}$-matrix to hierarchically divide the whole matrix into logical block matrices, for which the developed low-rank compression algorithm can be applied if blocks correspond to low-rank far field interactions at an appropriate scale, or they are divided further until direct summation is needed. Different from analytic or algebraic FMMs, the recursive structure nature of HRCM only executes an one-time, one way top-to-down path along the hierarchical tree structure: once a low-rank matrix is compressed, the whole block is removed from further consideration, and has no communications with the remaining entries of the whole kernel matrix. As the HRCM combines the $\mathscr{H}$-matrix structure and low-rank compression algorithms, it has an $O(N \log N)$ computational complexity.

Numerical simulations are provided for source and targets in two-dimensional (2D) geometry for several kernel functions, including 2D and 3D Green's function for Laplace equation, Poisson-Boltzmann equation and Helmholtz equation. As a Monte Carlo method, its reliability was analyzed in terms of single-realization and multiple-realizations. We concluded that, a single-realization of HRCM is statistically reliable. Multi-realization of the algorithm is more accurate but efficiency needs to be balanced. In various cases, the mean relative errors of the HRCM against direct kernel summation show convergence in terms of number of samples and diameter-distance ratios. The computational cost was shown numerically as $O(N \log N)$. Additionally, the break-even point with direct method is in the order of thousands, with three or four digit relative error. At last, the HRCM is implemented for high frequency wave problems for Helmholtz equations. As shown by our simulations, the mean error is linearly proportional wave number k, thus it could be significantly large in high frequency region. Increasing numbers of sampled columns or rows in the low-compression algorithm is one of the ways to reduce the error, but may not be the best way. The HRCM algorithm needs to be improved to handle high frequency problem.

One direction in the future work is to extend the HRCM for higher dimensional data. The next work could be solving volume integral equation (VIE) of Helmholtz or Maxwell's equations in 3-D using the HRCM. In this case, target and source points are distributed in 3-D geometries so an octree will be used to construct the $\mathscr{H}$-matrix, but the random compression algorithm for low rank matrix can be applied exactly the same way. Efficiency and accuracy of HRCM of the VIE method will be analyzed. For even higher dimensional data with dimension $d > 3$, the d-d tree is a more suitable space partitioning data structure. However, it is one type of binary trees different from quadtrees or octrees. The overall efficiency of the hierarchical structure and possible changes in the low rank compression algorithm for ultra-high dimensional data require careful investigations.

674 REFERENCES

675 [1] C. R. Anderson, *An implementation of the fast multipole method without multipoles*, SIAM
676 Journal on Scientific and Statistical Computing, 13 (1992), pp. 923–947.
677 [2] L. Banjai and W. Hackbusch, *Hierarchical matrix techniques for low-and high-frequency*
678 *helmholtz problems*, IMA journal of numerical analysis, 28 (2008), pp. 46–79.
679 [3] M. Bebendorf, *Approximation of boundary element matrices*, Numerische Mathematik, 86
680 (2000), pp. 565–589.
681 [4] W. C. Chew, E. Michielssen, J. Song, and J.-M. Jin, *Fast and efficient algorithms in*
682 *computational electromagnetics*, Artech House, Inc., 2001.
683 [5] M. H. Cho, J. Huang, D. Chen, and W. Cai, *A Heterogeneous FMM for Layered Media*
684 *Helmholtz Equation I: Two Layers in* $\mathbb{R}^2$, Journal of Computational Physics, in press
685 (2018).
686 [6] E. Darve, *The fast multipole method i: error analysis and asymptotic complexity*, SIAM
687 Journal on Numerical Analysis, 38 (2000), pp. 98–128.
688 [7] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast monte carlo algorithms for matrices i:*
689 *Approximating matrix multiplication*, SIAM Journal on Computing, 36 (2006), pp. 132–
690 157.
691 [8] P. Drineas, R. Kannan, and M. W. Mahoney, *Fast monte carlo algorithms for matrices ii:*
692 *Computing a low-rank approximation to a matrix*, SIAM Journal on computing, 36 (2006),
693 pp. 158–183.
694 [9] A. Frieze, R. Kannan, and S. Vempala, *Fastmonte carlo algorithms for finding low-rank*
695 *approximations*, J. ACM, 51 (2004), pp. 1025–1041.
696 [10] Z. Gimbutas and V. Rokhlin, *A generalized fast multipole method for nonoscillatory kernels*,
697 SIAM Journal on Scientific Computing, 24 (2003), pp. 796–817.
698 [11] A. G. Gray and A. W. Moore, *N-body'problems in statistical learning*, in Advances in neural
699 information processing systems, 2001, pp. 521–527.
700 [12] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, Journal of Com-
701 putational Physics, 135 (1997), pp. 280–292.
702 [13] W. Hackbusch, *A sparse matrix arithmetic based on* $\mathscr{H}$*-matrices. Part I: Introduction to*
703 $\mathscr{H}$*-matrices*, Computing, 62 (1999), pp. 89–108.
704 [14] W. Hackbusch and B. Khoromskij, *A sparse* $\mathscr{H}$*-matrix Arithmetic. Part II: Application to*
705 *Multi-Dimensional Problems*, Computing, 64 (2000), pp. 21–47.
706 [15] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Proba-*
707 *bilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53
708 (2011), pp. 217–288.
709 [16] S. Kapur and D. E. Long, *N-body problems: Ies 3: Efficient electrostatic and electromagnetic*
710 *simulation*, IEEE Computational Science and Engineering, 5 (1998), pp. 60–67.
711 [17] S. Kapur and J. Zhao, *A fast method of moments solver for efficient parameter extraction*
712 *of mcms*, in Proceedings of the 34th annual Design Automation Conference, ACM, 1997,
713 pp. 141–146.
714 [18] D. Lee, P. Sao, R. Vuduc, and A. G. Gray, *A distributed kernel summation framework*
715 *for general-dimension machine learning*, Statistical Analysis and Data Mining: The ASA
716 Data Science Journal, 7 (2014), pp. 1–13.
717 [19] W. B. March and G. Biros, *Far-field compression for fast kernel summation methods in high*
718 *dimensions*, Applied and Computational Harmonic Analysis, 43 (2017), pp. 39–75.
719 [20] W. B. March, B. Xiao, and G. Biros, *Askit: Approximate skeletonization kernel-independent*
720 *treecode in high dimensions*, SIAM Journal on Scientific Computing, 37 (2015), pp. A1089–
721 A1110.
722 [21] P.-G. Martinsson, V. Rokhlin, and M. Tygert, *A randomized algorithm for the decompo-*
723 *sition of matrices*, Applied and Computational Harmonic Analysis, 30 (2011), pp. 47–68.
724 [22] N. Yarvin and V. Rokhlin, *Generalized gaussian quadratures and singular value decomposi-*
725 *tions of integral operators*, SIAM Journal on Scientific Computing, 20 (1998), pp. 699–718.
726 [23] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in*
727 *two and three dimensions*, Journal of Computational Physics, 196 (2004), pp. 591–626.
728 [24] C. D. Yu, J. Levitt, S. Reiz, and G. Biros, *Geometry-oblivious fmm for compressing dense*
729 *spd matrices*, in Proceedings of the International Conference for High Performance Com-
730 puting, Networking, Storage and Analysis, ACM, 2017, p. 53.