Journal of Computational Physics 397 (2019) 108828

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

# An $O(N \log N)$ hierarchical random compression method for kernel matrices by sampling partial matrix entries

Duan Chen [a], Wei Cai [b,*]

[a] Department of Mathematics and Statistics, University of North Carolina at Charlotte, Charlotte, NC 28223, USA
[b] Department of Mathematics, Southern Methodist University, Dallas, TX 75275, USA

ARTICLE INFO

*Article history:*
Received 13 December 2018
Received in revised form 9 July 2019
Accepted 12 July 2019
Available online 17 July 2019

*Keywords:*
Fast kernel summation
Randomized algorithm
Matrix approximation
Singular value decomposition
$\mathscr{H}$-matrix
Hierarchical algorithms

ABSTRACT

In this paper, we propose an $O(N \log N)$ hierarchical random compression method (HRCM) for kernel matrix compressing, which only requires sampling $O(N \log N)$ entries of a matrix. 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 matrix. 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 matrix, independent of the matrix size 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 low frequency Helmholtz wave kernels have validated the efficiency and accuracy of the proposed method in comparison of direct $O(N^2)$ summations.

## 1. Introduction

Kernel matrices arise from many scientific and engineering computation, data analytics, and machine learning algorithms. A fundamental operation of kernel matrices is the summation:

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

where $\mathcal{K}(\mathbf{r}_i, \mathbf{r}_j)$ is the kernel function representing the interactions between $M$ 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. Simple operation as (1) poses great challenges in the era of big data even with current computing power, because the summation task becomes prohibitively expensive in both time and storage for practical computations when $M$ and $N$ are large. A general approach of the problem is to approximate the original kernel function in a degenerate form. i.e.

* Corresponding author.
*E-mail address:* cai@smu.edu (W. Cai).

https://doi.org/10.1016/j.jcp.2019.07.027
0021-9991/© 2019 Elsevier Inc. All rights reserved.

$$\mathcal{K}(\mathbf{r},\mathbf{r}') \approx \sum_{i=1}^{k} \sigma_i \phi_i(\mathbf{r})\psi_i(\mathbf{r}'), \tag{2}$$

where $\phi_i$ and $\psi_i$ are some basis functions and $\sigma_i$ are coefficients independent of $\mathbf{r}$ and $\mathbf{r}'$. Equivalently, in matrix form, we seek

$$\mathbf{A}^{M\times N} \approx \mathbf{B}^{M\times k}\mathbf{C}^{k\times N}, \tag{3}$$

where $\mathbf{A}$ is the matrix with entries corresponding to the kernel function $\mathcal{K}(\mathbf{r},\mathbf{r}')$ evaluated at $M$ targets from $N$ sources and $k \ll \min\{M,N\}$ is called the numerical rank of $\mathbf{A}$. With low-rank approximation (2) or (3), summation operation (1) and many matrix-related operations can be implemented rapidly, and the storage of data is more efficient. Applications include matrix-vector multiplication, iterative methods, matrix inversion, or kernel based optimizations.

Various fast algorithms have been studied over past decades. They share two common components: an essential low-rank approximation algorithm, and a hierarchical structure which breaks the whole summation into blocks where the low-rank algorithm can be applied. Hierarchical structures include the fast multipole method (FMM) [27], the tree code method [4], $\mathscr{H}$-matrices [28,30,25], $\mathscr{H}^2$-matrices [7,29], and hierarchically semiseparable (HSS) matrices [44,8,9], etc. On each compressible block, low-rank approximation can be roughly categorized as rapid-evaluation and matrix-decomposition types. If the kernel function is simple, fast evaluation can be implemented by analytical tools such as addition theorems of special function expansions or even simple Taylor expansions [27,12,15,24,23,33,13]. Otherwise, kernel-independent methods, in which no analytic expansion of the kernel function is required, were developed under various hierarchical structures. Some pioneering methods include semi-analytic and algebraic FMMs [2,46,48,22,49], kernel-independent treecode [42]. More recently, a proxy point selection method [47] and a unified framework for oscillatory kernels [45] were developed. There are also some work [26,35] in machine learning community that aiming at certain specific kernel functions. These rapid-evaluation are one-phase methods [1], but have to be implemented for every different vector $q_j$ in (1). In contrast, matrix-decomposition methods are two-phase. They work on the kernel matrix $\mathcal{K}(\mathbf{r}_i,\mathbf{r}_j)$ *only*, first decompose it into forms (2) or (3), then the low-rank approximation can be repeatedly used to evaluate (1) for any vector $q_j$ in a fast fashion. Some fast matrix-decomposition methods include the Adaptive Cross Approximation (ACA) algorithm [34,6,51], HSS approximation [8,9,43,44], and a recently developed nearest-neighbor based interpolative decomposition (ID) algorithms for high dimension summation [37,36,50]. Matrix decomposition has potential advantages for direct methods [1,8,44] or preconditioning [14] in solving large linear systems.

Recently, randomized numerical linear algebra (RNLA) methods thrived in speeding up low-rank decomposition of matrix data [40,44,38,39,16–19]. As illustrated in the comprehensive review [31], large data in current informational sciences are subject to random lost, inaccuracy or fluctuation. So it seems profligate to pursue highly accurate results by expensive algorithms, when the imprecision of the original data essentially limits the inherent solution quality. Additionally, computational cost now depends heavily on the transfer of large data from storage to CPU memory. Traditional deterministic methods usually require more passes over data, thus they are substantially slower in practice. Further, it is important to develop numerical algorithms to adapt to modern structure of computer hardware architectures, such as graphic processing units. Therefore, randomized algorithms serve as a powerful tool for matrix low-rank approximation. Indeed, contemporary Monte Carlo methods of matrix low-rank approximations are relatively insensitive to the quality of randomness and offer highly accurate results.

RNLA for low-rank approximation includes random projection and random sampling methods. Based on Johnson-Lindenstrauss lemma [32], random projection methods project the original large data matrix to a suitable low-dimensional space by multiplying a random matrix, such that the distances between column (row) vectors are approximately preserved. This idea was originally applied in fast computation of least square problems, while was recently used in fast matrix factorizations [40,38]. On the other hand, the idea of random sampling methods is only choosing partial information of the original data to approximate its overall behavior. Based on this methodology, randomly sampled algorithms for SVD [16,17], matrix-vector multiplication [16], or other decompositions have been developed [18,19]. Both methods are efficient in data transfer, because usually at most two passes of data are needed.

In this work, we develop a RNLA-based hierarchical random compression method (HRCM) to handle the kernel summation challenges in numerical solutions of the volume integral equations (VIEs)

$$\mathbf{C}\cdot\mathbf{E}(\mathbf{r}) = \mathbf{E}^{\mathrm{src}}(\mathbf{r}) + \mu\omega^2\ \mathrm{p.v.}\int_{\Omega} \Delta\epsilon(\mathbf{r}')\mathbf{E}(\mathbf{r}')\cdot\overline{\mathbf{G}}_{\mathbf{E}}(\mathbf{r},\mathbf{r}')d\mathbf{r}' \tag{4}$$

for studying electromagnetic fields with the Maxwell equations in random meta-materials (MMs) [10,11]. In this application, a large number of variables are needed in order to take into account the many degrees of freedom of the meta-atoms in the MMs in a layered material environment. Discretization of Eq. (4) results in a very large linear system with a dense matrix. Besides large size, problem (4) encounters two other challenges: (i) Evaluation of kernel matrix entries is by complicated Sommerfeld integrals [10], so it is time consuming to obtain all the matrix entries; (ii) for optimal device design of solar cells [41], the large linear system needs to be solved repeatedly with random sample of kernel matrices corresponding to stochastic variations of MMs microstructures. Aiming at the two challenges, our objective is to develop a fast kernel-independent decomposition method that only uses partial information and can be implemented repeatedly in high efficiency.

It is very important to point out the difference of our objective from existing methods. The goal of all current RNLA methods is to enhance computational efficiency of matrix decomposition from their conventional deterministic counter parts. They generally reduce complexity from $O(N^3)$ to at least $O(N^2)$. This is because: (a) in random projection methods, all matrix entries need to be visited to implement the multiplication with the random matrix; (b) random sampling methods only use partial information, but unless *a prior* knowledge of original data is available, all matrix data need to be involved to compute the optimal sampling probability. In this sense, most of current decomposition methods are not cheaper than the complexity $O(N^2)$ of direct kernel summation. In [40,38], an $O(N)$ random projection method was developed for HSS decomposition, but it is conditioned with the assumption that the other fast algorithms (such as FMM) can apply to the target matrix. Based on random sampling, an $O(N)$ CUR decomposition method was proposed in [18] but it is based on known sampling probability. A similar technique was used in [21], while the sampling issue was not addressed sufficiently.

In this work, we propose an $O(N)$ data-oblivious compression algorithm for low-rank matrix and address the sampling issues. Specifically, the novelty of the developed method in our work include:

(i) No need to access all matrix entries. The hierarchal low-rank approximation is developed with sampling *small amount* of original data. This property is especially important because evaluating each entry $\bar{\mathbf{G}}_{\mathbf{E}}(\mathbf{r}_i, \mathbf{r}_j)$ in Eq. (4) by Sommerfeld representation is expensive;

(ii) Low cost in data sampling. In traditional RNLA methods, it costs $O(N^2)$ complexity to calculate the $l_2$ norm of the column/row of original data in order to get optimal probability or statistical leverage. In our approach, the uniform sampling distribution (no cost) is shown to be $\beta$-optimal on SVD sampling and only an $O(N)$ computation for $\beta$-optimal probability on matrix-vector projection. This strategy greatly enhances algorithm efficiency and maintains a balanced accuracy;

(iii) The compressing process itself is faster than the direct matrix-vector production for large scale problems. The complexity of storage and computing for the compressing algorithm is $O(N)$ for low-rank matrices while $O(N\log N)$ for general cases. This property guarantees the algorithm efficiency for handling the stochastic fluctuation of the kernel matrix.

Additionally, the expectation of error of the proposed algorithm converges with increasing number of sampled columns (rows) and decreasing diameter-distance ratio of the source-target sets. Meanwhile, the accuracy convergence and statistical robustness of the method will be demonstrated.

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 of a matrix 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 (T) set $\{\mathbf{r}_i\}$ as

$$diam(T) := \max_{i,i'} |\mathbf{r}_i - \mathbf{r}_{i'}|, \tag{5}$$

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

$$dist(T,S) := |\mathbf{r}_T^* - \mathbf{r}_S^*|, \tag{6}$$

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 $dist(T,S) > \frac{1}{2}(diam(T)+diam(S))$.

### 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 [5,3] if

$$|\partial_{\mathbf{r}}^{\alpha}\partial_{\mathbf{r}'}^{\beta}\mathcal{K}(\mathbf{r},\mathbf{r}')| \le c(|\alpha|,|\beta|)(1+k|\mathbf{r}-\mathbf{r}'|)^{|\alpha|+|\beta|}|\mathbf{r}-\mathbf{r}'|^{-|\alpha|-|\beta|-\tau}, \tag{7}$$

where parameters $k,\tau \ge 0$ and $\alpha,\beta$ are $d$-dimensional multi-indices. The constant $C(|\alpha|,|\beta|)$ only depends on $|\alpha|$ and $|\beta|$. Condition (7) 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 the wave number and $\tau=1$.

Assume points $\mathbf{r}$ and $\mathbf{r}'$ belong to target ($T$) and source ($S$) sets, respectively. Consider the Taylor expansion of $\mathcal{K}(\mathbf{r},\mathbf{r}')$ around $\mathbf{r}^*$, which is the Chebyshev center of $T$, i.e., $\mathcal{K}(\mathbf{r},\mathbf{r}')=\tilde{\mathcal{K}}(\mathbf{r},\mathbf{r}')+R$ with the polynomial

$$\tilde{\mathcal{K}}(\mathbf{r},\mathbf{r}') = \sum_{|\upsilon|=0}^{m-1}\frac{1}{\upsilon!}(\mathbf{r}-\mathbf{r}^*)^{\upsilon}\frac{\partial^{\upsilon}\mathcal{K}(\mathbf{r}^*,\mathbf{r}')}{\partial\mathbf{r}^{\upsilon}}, \tag{8}$$

and the remainder $R$ satisfies

$$|R| = |\mathcal{K}(\mathbf{r},\mathbf{r}')-\tilde{\mathcal{K}}(\mathbf{r},\mathbf{r}')| \le \frac{1}{m!}|\mathbf{r}-\mathbf{r}^*|^m \max_{\zeta\in T,|\gamma|=m}\left|\frac{\partial^{\gamma}\mathcal{K}(\zeta,\mathbf{r}')}{\partial\zeta^{\gamma}}\right|. \tag{9}$$

If the well-separated target and source charges satisfies

$$\max\{diam(T), diam(S)\} < \eta \times dist(T,S), \tag{10}$$

for a parameter $0<\eta<1$, then Eq. (9) has the following estimate,

$$|\mathcal{K}(\mathbf{r},\mathbf{r}')-\tilde{\mathcal{K}}(\mathbf{r},\mathbf{r}')| \le c(m)\eta^m(1+k|\mathbf{r}-\mathbf{r}'|)^m|\mathbf{r}-\mathbf{r}'|^{-\tau}, \quad \mathbf{r}\in T, \mathbf{r}'\in S. \tag{11}$$

Estimate (11) 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. (8) with error $O(\eta^m)$, the corresponding interaction matrix $\tilde{\mathcal{K}}$ is of low rank, i.e.

$$rank(\tilde{\mathcal{K}}) = K(m) \ll \min\{M,N\} = rank(\mathcal{K}), \tag{12}$$

where

$$K(m) = \sum_{p=0}^{m-1}\frac{(d-1+p)!}{(d-1)!p!}. \tag{13}$$

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. (8). 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 $\tau$ in the decaying condition of the kernel (7).

### 2.2. Random kernel compression algorithms

The best 2-norm low-rank approximation of a matrix is the truncated SVD, according to the Echhart-Young theorem [20]. But it is extremely expensive to obtain a SVD approximation

$$\mathbf{A} \approx \sum_{i=1}^{K}\sigma_A^i\mathbf{U}_A^{(i)}\mathbf{V}_A^{(i)*}, \tag{14}$$

for a matrix $\mathbf{A}\in\mathbb{K}^{M\times N}$ with a rank $K \ll \min\{M,N\}$. Motivated by the Monte Carlo methods described in [16,17], we develop an $O(N)$ fast algorithm to approximate the singular values $\sigma_A^i$, left singular vector $\mathbf{U}_A^{(i)}$ and right singular vector $\mathbf{V}_A^{(i)}$ with desired accuracy. So the compression process, plus the consequential matrix vector multiplication, are more efficient than the direct summation.

In [17], an $O(1)$ implementation (the "ConstantTimeSVD") was developed to obtain the approximated singular value $\sigma_A^i$ and a "description" of the left singular vector of $\mathbf{A}$. However, it will take an additional $O(\max\{M,N\})$ complexity to explicitly approximate $\mathbf{U}_A^{(i)}$ and still $O(MN)$ complexity to project $\mathbf{A}$ on $\mathbf{U}_A^{(i)}$ to get the approximated right singular vectors

$\mathbf{V}_A^{(i)}$. To overcome the $O(MN)$ complexity, we combine the idea of constant time SVD with Monte Carlo matrix multiplication algorithms [16] and proper sampling strategies.

Let $\mathbf{C} \in \mathbb{K}^{M\times c}$ be a 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 $\mathbf{V}_c \in \mathbb{K}^{c\times c}$, and the singular value diagonal matrix $\Sigma_c \in \mathbb{K}^{M\times c}$. 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 $\mathbf{V}_r \in \mathbb{K}^{c\times c}$, and the singular value diagonal matrix $\Sigma_r \in \mathbb{K}^{r\times c}$. Then

- SVD is only performed to obtain $\mathbf{C}_r = \mathbf{U}_r \Sigma_r \mathbf{V}_r^*$, in which $r, c \le K \ll \min\{M, N\}$ and independent of $M$ and $N$. Due to rapid decay of singular values, only the first $t_0$ columns of $\mathbf{V}_r$ are needed, where $t_0$ is determined by checking $|\sigma_r^{t_0+1} - \sigma_r^{t_0}| < \epsilon$, and $\epsilon$ is a preset accuracy criteria. Then $\{\sigma_r^t\}_{t=1}^{t_0}$ are considered as the approximated singular values of $\mathbf{A}$.
- The columns in $\mathbf{V}_r$ are similar to the "description" of the left singular vectors of $\mathbf{A}$ in [17]. But we do need an explicit orthonormal set to approximate $\mathbf{U}_A$. To achieve this goal, we compute $\mathbf{U}_c\Sigma_c \approx \mathbf{C}\mathbf{V}_r$ with only $t_0$ columns of $\mathbf{V}_r$. Then a QR factorization of $\mathbf{C}\mathbf{V}_r$ is computed, i.e., $\mathbf{C}\mathbf{V}_r = \tilde{\mathbf{U}}_c\mathbf{R}$, with $\mathbf{R}$ being the upper triangle matrix. We use the resulting $\tilde{\mathbf{U}}_c$ as an approximation of $\mathbf{U}_c$, and hence an approximation of $\mathbf{U}_A$, since $\mathbf{A}$ and $\mathbf{C}$ share "a similar" column space.
- To approximate the right singular vectors of $\mathbf{A}$, one could project $\mathbf{A}$ on $\tilde{\mathbf{U}}_c$. Our experiments show that $\tilde{\mathbf{U}}_c\tilde{\mathbf{U}}_c^*\mathbf{A}$ offers an approximation of $\mathbf{A}$ with extremely high accuracy, but exact computation of $\tilde{\mathbf{U}}_c^*\mathbf{A}$ requires $O(t_0MN)$ operations and thus exceeds our budget. Instead, we compute QR factorization of $\mathbf{A}^*\tilde{\mathbf{U}}_c$, i.e., $\mathbf{A}^*\tilde{\mathbf{U}}_c = \tilde{\mathbf{V}}_A\mathbf{R}$ and use $\tilde{\mathbf{V}}_A$ as the approximated right singular vectors of $\mathbf{A}$. When computing $\mathbf{A}^*\tilde{\mathbf{U}}_c$, the Monte Carlo Basic matrix multiplication algorithm in [16] is adopted, i.e., only $c$ columns from $\mathbf{A}^*$ and $c$ rows from $\tilde{\mathbf{U}}$ are sampled and computed, this approximation reduces the complexity to $O(ct_0N)$.

**Remark.** (i) It is easy to verify that the overall random kernel compression approximating singular values, left and right singular vectors of $\mathbf{A}$ through above three steps have $O(\max\{M, N\})$ operations; (ii) one could implement the projection $\tilde{\mathbf{U}}_c^*\mathbf{A}$ by the Monte Carlo Basic matrix multiplication algorithm and use $\tilde{\mathbf{U}}_c\tilde{\mathbf{U}}_c^*\mathbf{A}$ as the decomposition. However, our practices suggest that orthonormalization of $\mathbf{A}^*\tilde{\mathbf{U}}_c$ as right singular vectors gives much higher accuracy; (iii) sampling strategy is an important issue. As discussed later, we take the $\beta$-optimal probabilities, so there is no cost in sampling to perform randomized SVD and only a $O(N)$ cost in the random matrix-vector multiplication.

Detailed implementations are given in Algorithm 1, where $\Pi\mathbf{A}$ represents the compressed matrix.

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

$$\mathbf{A} \approx \Pi\mathbf{A} = \sum_{t=1}^{t_0} \sigma_t \mathbf{U}_t \mathbf{V}_t^*. \tag{15}$$

```
1. Construct matrix $\mathbf{C} \in \mathbb{K}^{M\times c}$, by randomly picking $c$ columns from $\mathbf{A}$ with a uniform distribution (as $\beta$-optimal probability) without replacement and rescaling each entry by $\sqrt{N/c}$, i.e., $\mathbf{C}^{(t)} = \sqrt{N/c}\mathbf{A}^{(t)}$, $t = 1, 2, ..., c$.
2. Construct matrix $\mathbf{C}_r \in \mathbb{R}^{r\times c}$, by randomly picking $r$ rows from $\mathbf{C}$ with a uniform distribution without replacement and rescaling each entry by $\sqrt{N/r}$, i.e., $\mathbf{C}_{r(t)} = \sqrt{N/r}\mathbf{C}_{(t)}$, $t = 1, 2, ..., c$.
3. Perform SVD on matrix $\mathbf{C}_r$, i.e., $\mathbf{C}_r = \mathbf{U}_r \boldsymbol{\Sigma}_r \mathbf{V}_r^*$ and denote $\sigma(\mathbf{C}_r)$ as the set of singular values.
4. Let $t_0 = \min\{r, c, \arg\max_j \sigma_j(\mathbf{C}_r)\} > \epsilon\}$, set $\sigma_t = \sigma_t(\mathbf{C_r})$ and compute $\{\mathbf{U}_t^c\}_{t=1}^{t_0} = \{\mathbf{C}\mathbf{V}_t\}_{t=1}^{t_0}$;
5. Perform QR decomposition on $\{\mathbf{U}_t^c\}_{t=1}^{t_0}$ to obtain the output orthonormal vectors $\{\mathbf{U}_t\}_{t=1}^{t_0}$.
6. Compute $\{\tilde{\mathbf{V}}_t\}_{t=1}^{t_0} = \{\mathbf{A}^*\mathbf{U}_t\}_{t=1}^{t_0}$ by the MonteCarlo Basic matrix multiplication algorithm in [16], with sample size $c$ and probability $\bar{p}_i = \mathbf{U}_t^{(i)} / \sum_{i'=1}^{M} \mathbf{U}_t^{(i')}$;
7. Perform QR decomposition on $\{\tilde{\mathbf{V}}_t\}_{t=1}^{t_0}$ to obtain the output orthonormal vectors $\{\mathbf{V}_t\}_{t=1}^{t_0}$
```

## 3. Analysis of the compression algorithm for well-separated sets

In this section we investigate some characteristics of the random compression algorithm. Algorithm 1 includes a two-level sampling of a matrix to perform its approximated SVD and another sampling for projection. A full analysis on $\Pi\mathbf{A}$ in (15) with general sampling probability is technically complicated and still ongoing. Instead, we consider a prototype model in the rest of this section. This prototype model is computationally inefficient but theoretically simple. It is different from Algorithm 1 but still provides meaningful insights about sampling strategies and accuracy for our methods.

*A prototype model:* Construct the matrix $\mathbf{C}$ as in Algorithm 1, but with an arbitrary sampling probability, and let $\mathbf{U}_K$ be the matrix containing only the first $K$ columns of $\mathbf{U}_c$ in $\mathbf{C} = \mathbf{U}_c\Sigma_c\mathbf{V}_c^*$. Define the projection operator $\tilde{\Pi} = \mathbf{U}_K\mathbf{U}_K^*$ and investigate the compression error

$$\mathbf{A} - \tilde{\Pi}\mathbf{A}, \quad \text{with } \tilde{\Pi} = \mathbf{U}_K\mathbf{U}_K^*. \tag{16}$$

Construction of the projector $\tilde{\Pi}$ is equivalent to the "LinearTimeSVD" algorithm in [17]. It is accurate but the complexity is $O(MN)$. Nevertheless, we can exam sampling strategies and perform error analysis by the following results from [17]:

**Theorem 1** *(Theorems 2 and 3 in [17]). Suppose* $\mathbf{A} \in \mathbb{K}^{M\times N}$ *and* $\mathbf{C} \in \mathbb{K}^{M\times c}$ *being the column sampled matrix from* $\mathbf{A}$ *as in Algorithm 1 with an arbitrary sampling distribution. Let* $\tilde{\Pi}$ *be the projector defined in Eq. (16), then*

$$\|\mathbf{A} - \tilde{\Pi}\mathbf{A}\|_F^2 \le \|\mathbf{A} - \mathbf{A}_K\|_F^2 + 2\sqrt{K}\|\mathbf{AA}^* - \mathbf{CC}^*\|_F; \tag{17}$$

*and*

$$\|\mathbf{A} - \tilde{\Pi}\mathbf{A}\|_2^2 \le \|\mathbf{A} - \mathbf{A}_K\|_2^2 + 2\|\mathbf{AA}^* - \mathbf{CC}^*\|_2; \tag{18}$$

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

Since $\mathbf{A}$ is assumed of low-rank, we only need to focus on estimating $\|\mathbf{AA}^* - \mathbf{CC}^*\|_\xi, \xi = 2, F$. 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. $\beta$-optimal sampling for far field kernel matrices

Sampling probability is critical for accuracy of randomized algorithms. For example, what is the best probability such that the error $\|\mathbf{AA}^* - \mathbf{CC}^*\|_\xi$ in (17) or (18) is minimized? What is the best probability for the matrix vector multiplication in the projection process in Algorithm 1? The optimal probability of (column) sampling to perform a Monte Carlo matrix-matrix multiplication $\mathbf{AB}$ is proposed in [16].

**Definition 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

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

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 $\mathbf{E}[\|\mathbf{AB} - \mathbf{CR}\|_F]$ is minimized.

Applying this definition to $\mathbf{AA}^*$, and using Lemma 4 in [16], it is easy to conclude that if $c$ columns are sampled, the expectation of error $\|\mathbf{AA}^* - \mathbf{CC}^*\|_F^2$ is minimized as

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

However, unless known in advance, utilizing the optimal probability is not practical, because computing all columns (rows) of a matrix is already more expensive than the actual matrix-vector multiplication. In our approach, a nearly optimal probability introduced in [16] will be used, instead.

**Definition 3** *(Nearly optimal probability).* For the same conditions in Definition 2, the probability $\{p_j\}$ is called a nearly optimal probability if

$$p_j \ge \frac{\beta|A^{(j)}||B_{(j)}|}{\sum_{j'=1}^N |A^{(j')}||B_{(j')}|}, \quad j = 1, 2, ..., N, \tag{21}$$

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

With the nearly optimal probability, one has

$$\mathbf{E}\left[\|\mathbf{AA}^* - \mathbf{CC}^*\|_F^2\right] \le \frac{1}{\beta c}\|\mathbf{A}\|_F^4 - \frac{1}{c}\|\mathbf{AA}^*\|_F^2. \tag{22}$$

Similarly, if the second matrix $\mathbf{B}$ is a vector, i.e. $P = 1$, one has

$$\mathbf{E}\left[|\mathbf{Ax} - \mathbf{C}\bar{\mathbf{x}}|^2\right] \le \frac{1}{\beta c}\left(\sum_{j=1}^N |\mathbf{A}^{(j)}||x_j|\right)^2 - \frac{1}{c}|\mathbf{Ax}|^2. \tag{23}$$

The positivity of the right-hand sides of Eqs. (22) and (22) are guaranteed by the Cauchy-Schwarz inequality and the fact $\beta < 1$.

Now we will present the following results, to claim that the sampling strategies in Algorithms 1 are $\beta$-optimal sampling–the nearly optimal sampling depending on $\beta$.

**Theorem 4.** *For well-separated source and target points, uniform sampling provides a nearly optimal probability for sampling matrix in the prototype model. Meanwhile, the probability $\bar{p}_j = \frac{|x_j|}{\sum_{j'=1}^{N} |x_{j'}|}$ serves as the nearly optimal probability for projecting $\mathbf{A}$ on a vector $\mathbf{x}$.*

**Proof.** To prove the first half of the conclusion, we look at Eqs. (17)-(20). According to (19), the optimal probability to minimize $\mathbf{E}\left[\|\mathbf{AA}^* - \mathbf{CC}^*\|_F^2\right]$ is

$$p_j = \frac{|A^{(j)}|^2}{\|A\|_F^2}. \tag{24}$$

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

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

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

$$p_j = \frac{|A^{(j)}|^2}{\|A\|_F^2} \le \frac{|\mathcal{K}(\delta - a)|^2}{N|\mathcal{K}(\delta + a)|^2}. \tag{25}$$

Thus, the uniform sampling probability

$$\hat{p}_j = \frac{1}{N} > \frac{|\mathcal{K}(\delta + a)|^2}{|\mathcal{K}(\delta - a)|^2} \frac{|A^{(j)}|^2}{\|A\|_F^2}, \tag{26}$$

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

Similarly, for the second half of the theorem, the optimal probability for fast computing $\mathbf{Ax}$ is

$$p_j = \frac{|A^{(j)}||x_j|}{\sum_{j'=1}^{N} |A^{(j')}||x_{j'}|} \le \frac{\sqrt{M}|\mathcal{K}(\delta - a)||x_j|}{\sqrt{M}|\mathcal{K}(\delta + a)| \sum_{j'=1}^{N} |x_{j'}|}, \tag{27}$$

hence the probability

$$\bar{p}_j = \frac{|x_j|}{\sum_{j'=1}^{N} |x_{j'}|} \ge \frac{|\mathcal{K}(\delta + a)|}{|\mathcal{K}(\delta - a)|} \frac{|A^{(j)}||x_j|}{\sum_{j'=1}^{N} |A^{(j')}||x_{j'}|} \tag{28}$$

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

Theorem 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 (22) in the kernel compression for well-separated source and target points. We also use this idea in the more complicated Algorithm 1. However, the bounds in Eqs. (22)-(23) depend on parameter $\beta$, 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 bounds (22)-(23) fail.

The reason why uniform sampling works can be illustrated heuristically by Fig. 1: When target and source sets are well-separated and have a small diameter-distance ratio $\eta$ 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. (20) or (22) for the separation. We show that even the error (20) or (22) depends on not only sample number $c$, but also the diameter-distance ratio $\eta$.

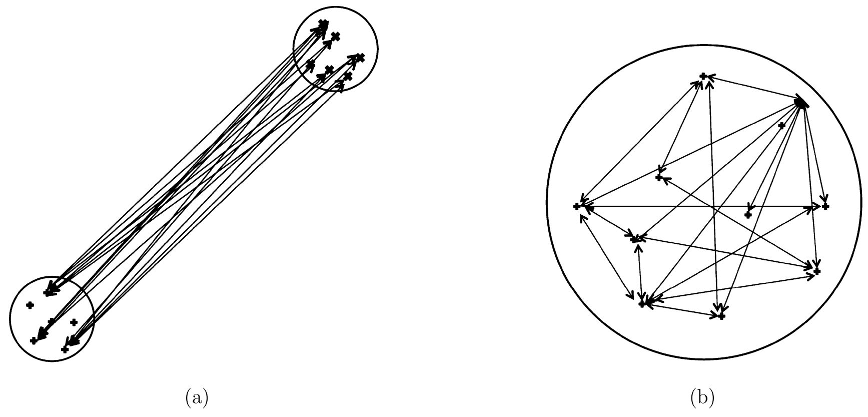


**Fig. 1.** (a) well-separated; (b) not well-separated.

### 3.2. Error analysis on target-source separation

Error estimates (20) and (22) provided in [17] are for a general matrix **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 **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 ratio, we will consider Eq. (20) for simplicity.

**Theorem 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 (7), and $\mathbf{C} \in \mathbb{K}^{M\times c}$ being the column sampled matrix from $\mathbf{A}$. Let $\tilde{\Pi}$ be the orthogonal projector defined in Eq. (16), then in addition to the error analysis in Eqs. (17)-(18), we have the sampling error*

$$\mathbf{E}\left[\|\mathbf{AA}^* - \mathbf{CC}^*\|_F\right] \le C(\delta, a, \tau, k)\frac{1}{\sqrt{c}}\frac{2\eta}{2-\eta}\|\mathbf{A}\|_F^2, \tag{29}$$

*where $\delta$ 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*

$$C(\delta, a, \tau, k) = \left(\frac{\delta}{2}\right)^{-\tau}\frac{(1+2k\delta)}{|\mathcal{K}(\delta+a)|}. \tag{30}$$

**Proof.** To prove (29), write the $i$-th row of **A** as

$$A_{(i)} = (\mathcal{K}(\mathbf{r}_i,\mathbf{r}_1), \mathcal{K}(\mathbf{r}_i,\mathbf{r}_2), ..., \mathcal{K}(\mathbf{r}_i,\mathbf{r}_N)), \quad i = 1, 2, ...M. \tag{31}$$

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

$$\Delta_{ii'} = (\Delta_{ii'}^{(1)}, \Delta_{ii'}^{(2)}, ...\Delta_{ii'}^{(N)}), \tag{32}$$

where

$$\Delta_{ii'}^{(j)} = \mathcal{K}(\mathbf{r}_i,\mathbf{r}_j) - \mathcal{K}(\mathbf{r}_{i'},\mathbf{r}_j) = (\mathbf{r}_i - \mathbf{r}_{i'}) \cdot \frac{\partial}{\partial \mathbf{r}}\mathcal{K}(\mathbf{r},\mathbf{r}_j)\big|_{\mathbf{r}=\mathbf{r}_T^*}. \tag{33}$$

Recall the assumption in Eq. (7) and condition (10), we have an estimate

$$|\Delta_{ii'}^{(j)}| \le \frac{a}{\delta - \frac{a}{2}}\left(1 + k\left(\delta + \frac{a}{2}\right)\right)\left(\delta - \frac{a}{2}\right)^{-\tau} \le \frac{2\eta}{2-\eta}(1+2k\delta)\left(\frac{\delta}{2}\right)^{-\tau}. \tag{34}$$

Rewrite Eq. (20) as

$$\mathbf{E}\left[\|\mathbf{AA}^* - \mathbf{CC}^*\|_F^2\right] = \frac{1}{c}\left(\|\mathbf{A}\|_F^4 - \|\mathbf{AA}^*\|_F^2\right)$$
$$= \frac{1}{c}\left[\left(\sum_{i=1}^M |A_{(i)}|^2\right)^2 - \sum_{i=1}^M\sum_{i'=1}^M \langle A_{(i)}, A_{(i')}\rangle^2\right]$$

$$= \frac{1}{c}\sum_{i=1}^{M}\sum_{i'=1}^{M}\left[\langle A_{(i)}, A_{(i)}\rangle\langle A_{(i')}, A_{(i')}\rangle - \langle A_{(i)}, A_{(i')}\rangle^2\right], \tag{35}$$

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

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

$$\begin{aligned}
&\langle A_{(i)}, A_{(i)}\rangle\langle A_{(i')}, A_{(i')}\rangle - \langle A_{(i)}, A_{(i')}\rangle^2 \\
&= \langle A_{(i)}, A_{(i)}\rangle\langle \Delta_{ii'}, \Delta_{ii'}\rangle - \langle A_{(i)}, \Delta_{ii'}\rangle^2 \\
&\le \langle A_{(i)}, A_{(i)}\rangle\langle \Delta_{ii'}, \Delta_{ii'}\rangle.
\end{aligned} \tag{36}$$

Plugging (36) in (35) and using estimate (34), we arrive at

$$\begin{aligned}
\mathbf{E}\left[\|\mathbf{AA}^* - \mathbf{CC}^*\|_F^2\right] &\le \frac{1}{c}\sum_{i=1}^{M}\sum_{i'=1}^{M}\langle A_{(i)}, A_{(i)}\rangle\langle \Delta_{ii'}, \Delta_{ii'}\rangle \\
&\le \frac{MN}{c}\left(\frac{2\eta}{2-\eta}\right)^2 (1+2k\delta)^2\left(\frac{\delta}{2}\right)^{-2\tau}\|\mathbf{A}\|_F^2.
\end{aligned} \tag{37}$$

By Jensen's inequality

$$\begin{aligned}
\mathbf{E}\left[\|\mathbf{AA}^* - \mathbf{CC}^*\|_F\right] &\le \frac{\sqrt{MN}}{\sqrt{c}\|\mathbf{A}\|_F}(1+2k\delta)\left(\frac{\delta}{2}\right)^{-\tau}\frac{2\eta}{2-\eta}\|\mathbf{A}\|_F^2. \\
&\le \frac{\sqrt{MN}(1+2k\delta)\left(\frac{\delta}{2}\right)^{-\tau}}{\sqrt{MN}\,|\mathcal{K}(\delta+a)|}\frac{1}{\sqrt{c}}\frac{2\eta}{2-\eta}\|\mathbf{A}\|_F^2 \\
&= C(\delta, a, \tau, k)\frac{1}{\sqrt{c}}\frac{2\eta}{2-\eta}\|\mathbf{A}\|_F^2,
\end{aligned} \tag{38}$$

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

This result indicates that if the matrix $\mathbf{A}$ is approximately low-rank ($\|\mathbf{A}-\mathbf{A}_K\|_F$ or $\|\mathbf{A}-\mathbf{A}_K\|_2$ is negligible), then the error of kernel compression majorly comes from the sampling error. In addition to sample size, this relative sampling error 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 $C(\delta, a, \tau, k)$, 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 kernel compression algorithms.

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

For general situation, target and source are not well-separated. In fact, they typically belong to the same set. In this case, we will partition the whole matrix into a sequence of blocks at different levels. Each of these blocks 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*

Full definition, description, and construction of $\mathscr{H}$-matrices are given in details in [28,30], and a good introduction can be found in [25]. For convenience, we briefly summarized the definition and two simple examples in the following. The idea of constructing a $\mathscr{H}$-matrix is to approximate a dense, full-rank matrix by partitioning it into a set of (recursively structured) "data sparse" or low-rank matrices. The matrix structure is based on a finite index set $I$ (e.g. $I = \{1, 2, 3, \ldots n\}$) and a partition $P$ which divides $I$ into disjoint subsets, i.e., $P = \{I_j : 1 \le j \le k\}$ with $I = \cup_{j=1}^{k} I_j$. Based on the partition $P$ of $I$, the block partition of a matrix is defined by the product $P_2 = P \times P = \{I' \times I'' : I', I'' \in P\}$. Then,

**Definition 6** *(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

$$\mathscr{M}_{\mathscr{H},K} := \{\mathbf{M} \in \mathbb{K}^{I\times I} : \text{each block } \mathbf{M}^b, b \in P_2, \text{satisfies } rank(\mathbf{M}^b) \le K\}. \tag{39}$$

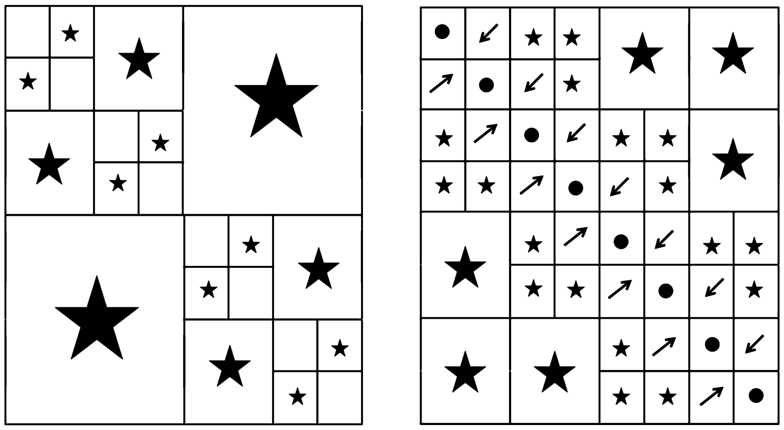


**Fig. 2.** Two examples of $\mathscr{H}$-matrix. Left: example one; right: example two.

**Remarks:** (1) $I$, $P$, and $P_2$ in the definition are very general, but in practice, the partition is achieved by a binary tree ($Tree$). For example, in the depth 0, $Tree = \{I\}$ and $P_2(I, Tree) = \{I \times I\}$. If the depth is 1, then $Tree = \{I_1, I_2\}$ and $P_2 = \{I_1 \times I_1, I_1 \times I_2, I_2 \times I_1, I_2 \times I_2\}$. Note that not all subsets in $P_2$ have to be partitioned in further depth, so the partitioning is not necessary to be in tensor form; (2) The index set $I$ can be the counting of physical coordinates of target/source points. 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; (3) A specific $\mathscr{H}$-matrix is defined through 6 recursively, and we denote a matrix $\mathbf{A}$ as R-$K$ matrix if $rank(\mathbf{A}) \leq K$. To help understanding the concept, two simple examples are given as follows:

**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 the diagonal blocks are $\mathscr{H}$-matrix (subject to further partition) in the next level and off diagonal blocks are R-$K$ matrices, as stared in the graph.

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

**Example Two:** (i) concept one: 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) concept two: 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) The $\mathscr{H}$-matrix: $\mathbf{A} \in \mathscr{M}_{\mathscr{H},K}$ if either $2^p = K$ or it has the structure

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

Visualization of the second example with three levels of division is given in the right subfigure of Fig. 2. In this case, $R$-K matrices, as stared, start to appear from the second level of division. All the diagonal blocks, labeled by solid dots, are $\mathscr{H}$-matrices in further level, while blocks with arrow pointing up and down are of $\mathcal{N}^*$-type and $\mathcal{N}$-type, respectively.

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

$$\mathbf{A}\mathbf{x} = \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 + \mathbf{A}_{21}\mathbf{x}_1 + \mathbf{A}_{22}\mathbf{x}_2, \tag{40}$$

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 $\mathbf{A}_{12}$ and $\mathbf{A}_{21}$, while recursive division need to be implemented on $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ until random compression can apply, or a preset minimum block is reached where direct matrix-vector multiplication is used. But in Example Two, none of the four terms in Eq. (40) 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 Fig. 2 can be understood as interactions of target/source points along a 1D geometry (imaging $I$ as an interval). Example One indicates that interactions from different subinterval of $I$ (off diagonal blocks) can be approximated by low-rank compressions, while points with self-interactions or from the same subinterval require further division. In

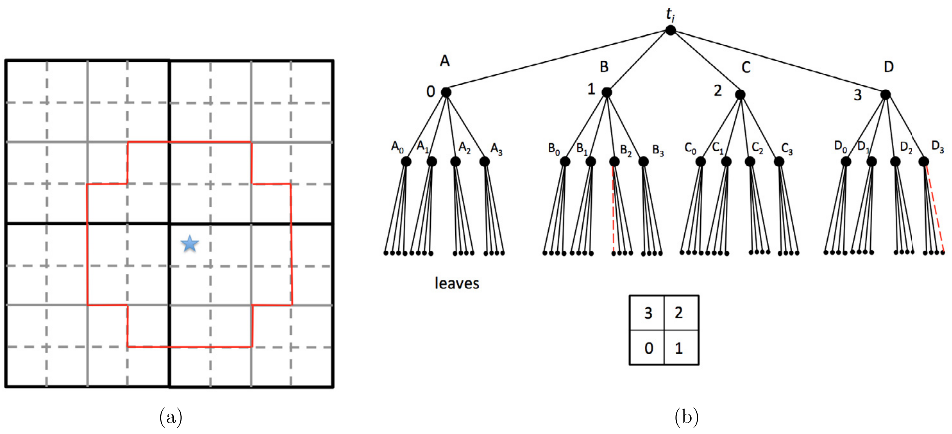


**Fig. 3.** Illustration of the index set $I$: (a) admissible cluster for the starred square; (b) quadtree structure. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

contrast, Example Two illustrates that matrices generated from points in the same subintervals, and from immediate neighboring subintervals are not low-rank yet; further partitioning is needed. Only the points from two subintervals "with some distance" can generate $R$-K matrices.

## 4.2. Tree structure of $\mathscr{H}$-matrix for two dimensional data

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 $P_2$ and the concept of admissible clusters.

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 as $I$, i.e.

$$I = \{(i, j) : 1 \le i, j \le N_1\}, \quad N_1 = 2^p. \tag{41}$$

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

$$X_{ij} : \{(x, y) : (i-1)h \le x \le ih, (j-1)h \le y \le jh\}, \quad h = L/N_1, \tag{42}$$

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

$$t^l_{\alpha,\beta} := \{(i, j) : 2^{p-l}\alpha + 1 \le i \le 2^{p-l}(\alpha+1), 2^{p-l}\beta + 1 \le j \le 2^{p-l}(\beta+1)\}, \tag{43}$$

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 $t_1 \in I_t$ and $t_2 \in I_s$ belong to the same level $l$. They represent general case so we drop the indices $\alpha$, $\beta$ and $l$. Then, following Eq. (5)-(6), we define the diameters and distances of $t_1$ and $t_2$, and the admissibility condition

$$\max\{diam(t_1), diam(t_2)\} \le \eta dist(t_1, t_2), \quad 0 < \eta < 1, \tag{44}$$

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. The above process is implemented recursively.

Fig. 3(a) shows the index set $I$, where black solid, gray solid and gray dashed lines are for partition at level $l = 1, 2, 3$, respectively. For different values of $\eta$, 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 red lines are non-admissible.

Fig. 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 $4^{p_0} \times 4^{p_0}$.

Fig. 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 are non-admissible for $\eta = \sqrt{2}/2$. Then each of them are further divided into four

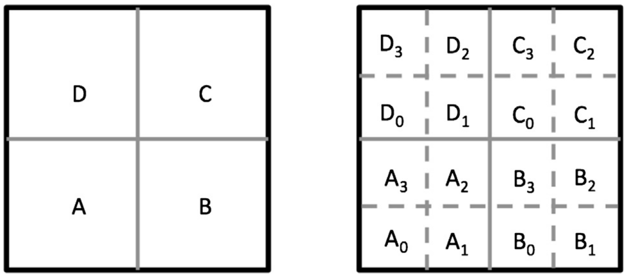


**Fig. 4.** Illustration of the partitioning of admissible clusters.

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 $\eta$, 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 A and x. Current level l and maximum level p.
Output: Product y = Ax, where 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 A and 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 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 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 on childless leaves. 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-

**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 A and x. Current level l and maximum level p.
Output: ỹ ≈ y = Ax, where ỹ 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
```

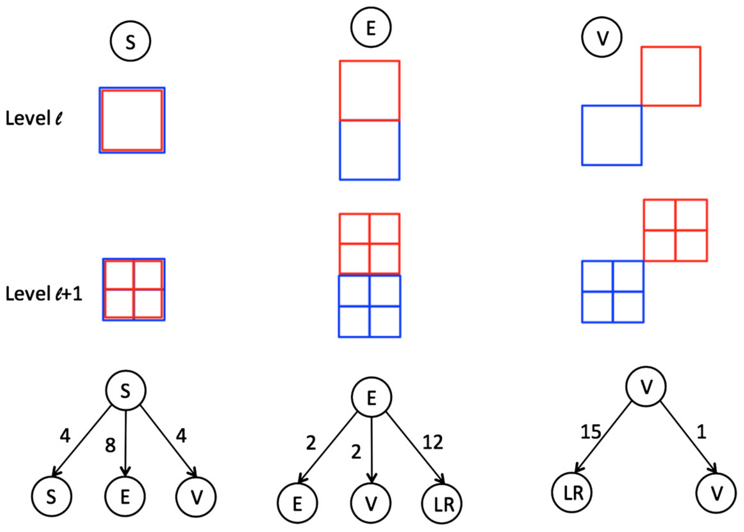


**Fig. 5.** Evolution of non-admissible clusters. S: self-interaction clusters; E: clusters touch by edge; V: clusters touch by vertex; LR: admissible clusters with low-rank approximation.

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

$$\underbrace{8 \cdot 2^{p-p_0-1-l-1}}_{\text{generated Es}} + \underbrace{8 \cdot 2 \cdot 1^{p-p_0-1-l-2} + 4 \cdot 1^{p-p_0-1-l-1}}_{\text{generated Vs}}. \tag{45}$$

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

$$16 \cdot O((4^{p_0})^2) \left[ \sum_{l=0}^{p-p_0-3} 4^l (8 \cdot 2^{p-p_0-l-2} + 8 \cdot 2 + 4) + 4^{p-p_0-2} \cdot 16 \right] = O(4^p) = O(N). \tag{46}$$

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

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

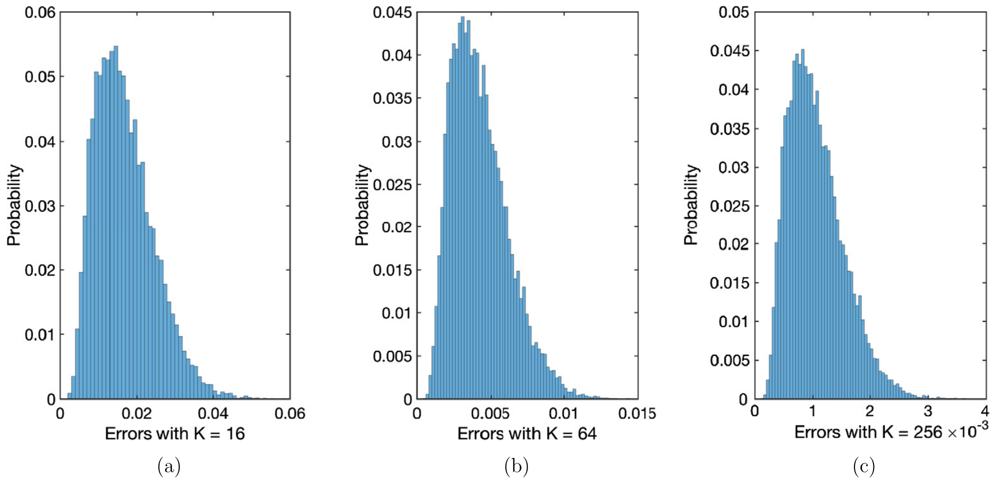


**Fig. 6.** Histograms of relative errors of 20, 000 single-realizations of matrix compression. (a) With $K = 16$, sample mean $1.699 \times 10^{-2}$, variance $6.002 \times 10^{-5}$, 95th percentile $3.118 \times 10^{-2}$; (b) With $K = 64$, sample mean $4.213 \times 10^{-3}$, variance $3.694 \times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

$$+\underbrace{\sum_{l'=l+3}^{p-p_0-1} 8 \cdot 2 \cdot 1^{l'-l-3} \cdot 15 \cdot O(4^{p-l'}) + \sum_{l'=l+3}^{p-p_0-1} 4 \cdot 1^{l'-l-2} \cdot 15 \cdot O(4^{p-l'})}_{\text{from all the V to the bottom}}, \tag{47}$$

where the latter three terms in (47) 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 (47) 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

$$\sum_{l=0}^{p-p_0-1} 4^l O(4^{p-l-2}) = O(p \times 4^p) = O(N \log N). \tag{48}$$

Combining Eqs. (46) and (48), 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 apart by double of their length. 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

$$\text{sMESR} = \frac{1}{N_s} \sum_{s=1}^{N_s} \frac{\|\mathbf{A} - \Pi_s(K)\mathbf{A}\|_F}{\|\mathbf{A}\|_F}, \tag{49}$$

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 realization of HRCM can provide reliable results: for very small sample size

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

| | sMESR | Variance | 95th 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 |

**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 |

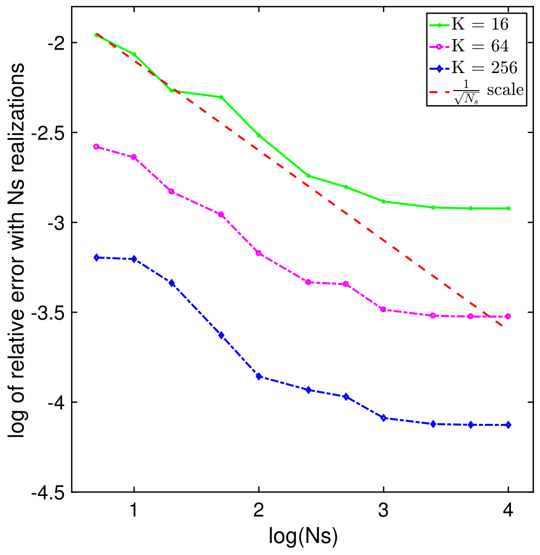


**Fig. 7.** Convergence of EMR with respect to $N_s$.

$K$, it offers two-digits relative errors with high confidence. When $K$ is increased, the HRCM shows convergence with respect to $K$ at the same percentile of confidence.

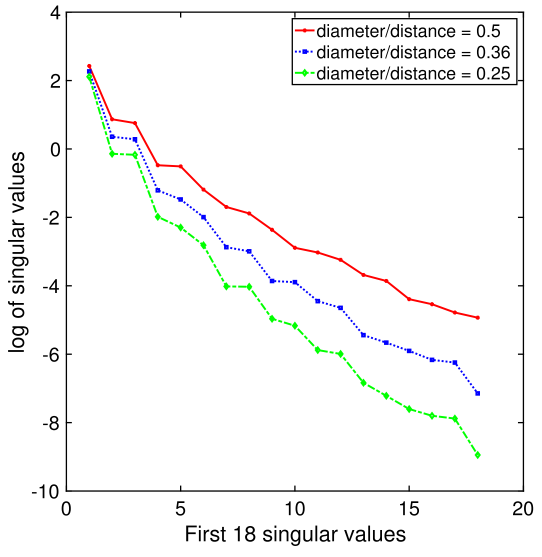
Random versus deterministic sampling: Secondly, we illustrate the importance of *random sampling*, although a single-realization of HRCM can be trusted. In Algorithm 2, sampling is achieved by randomly choosing one child from the target/source tree. For comparison, we also perform the HRCM by deterministically 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 multiple-realization of HRCM. Define the error of multiple realization (EMR):

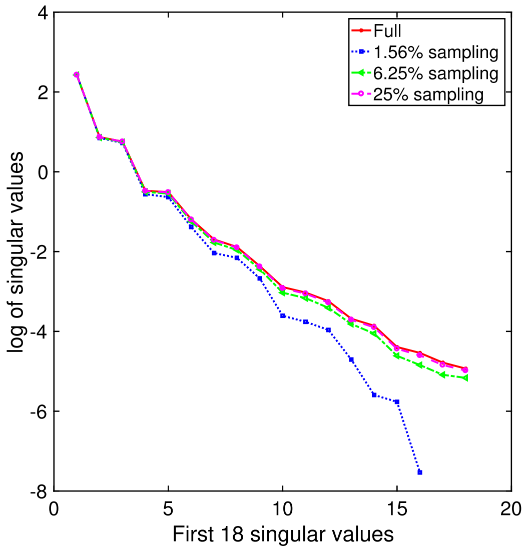
$$\text{EMR} = \frac{\|\mathbf{A} - \frac{1}{N_s}\sum_{s=1}^{N_s}\Pi_s(K)\mathbf{A}\|_F}{\|\mathbf{A}\|_F}, \tag{50}$$

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 $K$.

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



(a)



(b)

**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 $N = 1024$.

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

| | $N = 1{,}024$ | $N = 4096$ | $N = 16{,}384$ | $N = 65{,}536$ | $N = 262{,}144$ |
|---|---|---|---|---|---|
| $K = 16$ | | | | | |
| 1E sMESR | 2.79E−2 | 3.07E−2 | 3.5-2 | 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−7 | 4.71E−7 | 5.37E−7 | 5.63E−7 | 5.86E−7 |

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.

Next, we check the algorithm accuracy. A total of $N$ target points and source points are uniformly assigned in two $8\times 8$ 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 $\mathcal{K}(\mathbf{r},\mathbf{r}') = \log(\sqrt{(x-x')^2+(y+y')^2}) - \log(\sqrt{(x-x')^2+(y-y')^2})$ 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.

**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 |

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

| Direct | $N = 1{,}024$ | $N = 4096$ | $N = 16{,}384$ | $N = 65{,}536$ | $N = 262{,}144$ |
|---|---|---|---|---|---|
| | 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 |

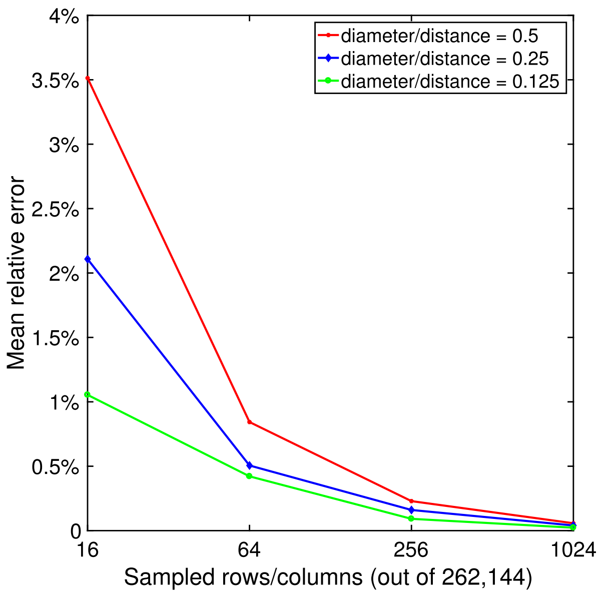


**Fig. 9.** Error of low-rank compression algorithm against diameter-distance ratio $\eta$.

Based on the numerical results from Tables 3–4, we conclude 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/ratio distance and number of row/column sampled.

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.

Fig. 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 $\eta$ 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 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 |

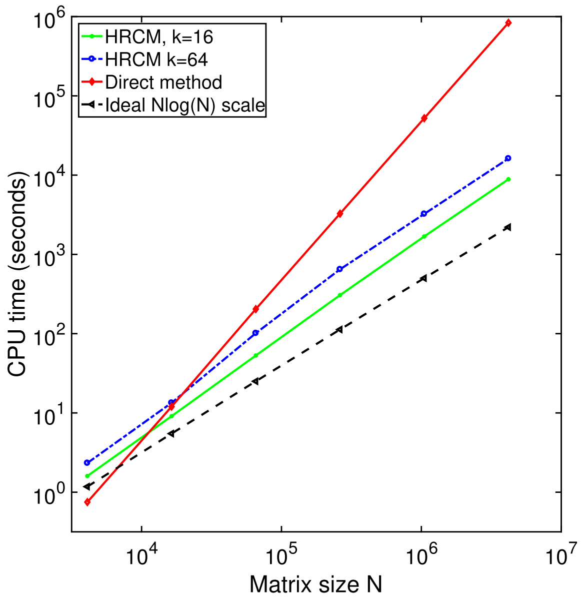


**Fig. 10.** CPU costs of HRCM with $K = 16$ and $K = 64$, for various matrix sizes. For comparison, the CPU time for direct method is shown in red and the ideal $N \log(N)$ curve is in black.

## 5.3. *Accuracy and efficiency for the same set of source and target points*

In this section, 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 does not change 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 $\eta$ with $\eta = 0.5$ as the largest value. Similar convergence with respect to $K$ is shown in the table. Note that simply ignoring the far-field reaction will end up with a relative error between $3 \sim 5 \times 10^{-1}$ in various cases.

The efficiency of the HRCM with such a kernel function is presented in Fig. 10 as log-log CPU time against 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 wave interactions. We consider the Green's function for the 3-D Helmholtz equation, $\mathcal{K}(\mathbf{r}_i, \mathbf{r}_j) = \exp(-ikR)/R$, when the wavenumber $k$ appears in the error estimate constant $C$ in (30). 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

**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 |

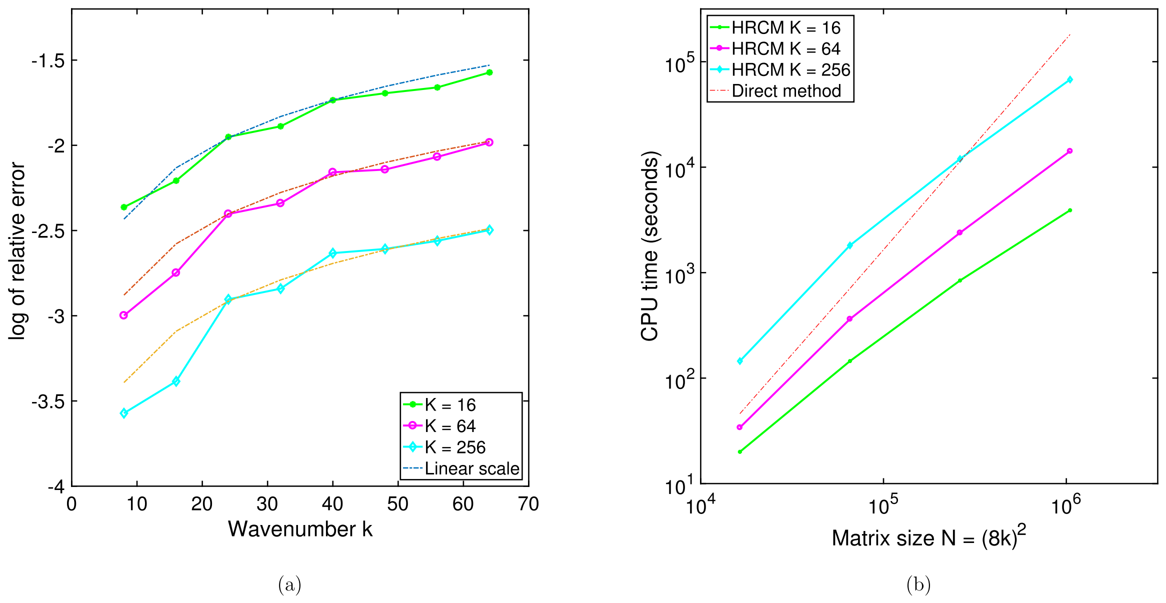


**Fig. 11.** (a) Relative errors and (b) efficiency of HRCM against wavenumber $k$.

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 Table 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 direction 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. (38). As a consequence, 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, for which the computation complexity of computing the matrix-vector product is $O(N \log N)$ for a given accuracy. Additionally, the HRCM is a purely date-based algorithm and only a small portion of the full data matrix is sampled. The two properties make HRCM specially suitable for large system with complicated date generating mechanism and stochastic fluctuation. 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, regardless of matrix sizes and only depending 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 each block, the developed low-rank compression algorithm can be applied if it corresponds to low-rank far field interactions at an appropriate scale, or it is 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 target points 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 at a higher cost. In various cases, the mean relative errors of the HRCM against direct kernel summation show convergence in terms of sample sizes 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 tested 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. 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 with more complicated kernel functions. 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 in 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.

## Acknowledgement

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

## References

[1] Sivaram Ambikasaran, Eric Darve, An $O(NlogN)$ fast direct solver for partial hierarchically semi-separable matrices, J. Sci. Comput. 57 (3) (2013) 477–501.
[2] Christopher R. Anderson, An implementation of the fast multipole method without multipoles, SIAM J. Sci. Stat. Comput. 13 (4) (1992) 923–947.
[3] Lehel Banjai, Wolfgang Hackbusch, Hierarchical matrix techniques for low- and high-frequency Helmholtz problems, IMA J. Numer. Anal. 28 (1) (2008) 46–79.
[4] Josh Barnes, Piet Hut, A hierarchical $O(N \log N)$ force-calculation algorithm, Nature 324 (6096) (1986) 446.
[5] Mario Bebendorf, Approximation of boundary element matrices, Numer. Math. 86 (4) (2000) 565–589.

[6] Mario Bebendorf, Sergej Rjasanow, Adaptive low-rank approximation of collocation matrices, Computing 70 (1) (2003) 1–24.
[7] Steffen Börm, Efficient Numerical Methods for Non-local Operators: $\mathscr{H}^2$-Matrix Compression, Algorithms and Analysis, vol. 14, European Mathematical Society, 2010.
[8] S. Chandrasekaran, M. Gu, W. Lyons, A fast adaptive solver for hierarchically semiseparable representations, Calcolo 42 (3–4) (2005) 171–185.
[9] Shiv Chandrasekaran, Ming Gu, Timothy Pals, A fast ULV decomposition solver for hierarchically semiseparable representations, SIAM J. Matrix Anal. Appl. 28 (3) (2006) 603–622.
[10] Duan Chen, Min Hyung Cho, Wei Cai, Accurate and efficient Nystrom volume integral equation method for electromagnetic scattering of 3-D metamaterials in layered media, SIAM J. Sci. Comput. 40 (1) (2018) B259–B282.
[11] Duan Chen, Brian Zinser, Wei Cai, Min Hyung Cho, Accurate and efficient Nyström volume integral equation method for the Maxwell equations for multiple 3-D scatterers, J. Comput. Phys. (2016) 303–320.
[12] Weng Cho Chew, Eric Michielssen, J.M. Song, Jian-Ming Jin, Fast and Efficient Algorithms in Computational Electromagnetics, Artech House, Inc., 2001.
[13] Min Hyung Cho, Jingfang Huang, Dangxing Chen, Wei Cai, A heterogeneous FMM for layered media Helmholtz equation I: two layers in $\mathbb{R}^2$, J. Comput. Phys. (2018) 237–251.
[14] Pieter Coulier, Hadi Pouransari, Eric Darve, The inverse fast multipole method: using a fast approximate direct solver as a preconditioner for dense linear systems, SIAM J. Sci. Comput. 39 (3) (2017) A761–A796.
[15] Eric Darve, The fast multipole method I: error analysis and asymptotic complexity, SIAM J. Numer. Anal. 38 (1) (2000) 98–128.
[16] Petros Drineas, Ravi Kannan, Michael W. Mahoney, Fast Monte Carlo algorithms for matrices I: approximating matrix multiplication, SIAM J. Comput. 36 (1) (2006) 132–157.
[17] Petros Drineas, Ravi Kannan, Michael W. Mahoney, Fast Monte Carlo algorithms for matrices II: computing a low-rank approximation to a matrix, SIAM J. Comput. 36 (1) (2006) 158–183.
[18] Petros Drineas, Ravi Kannan, Michael W. Mahoney, Fast Monte Carlo algorithms for matrices iii: computing a compressed approximate matrix decomposition, SIAM J. Comput. 36 (1) (2006) 184–206.
[19] Petros Drineas, Michael W. Mahoney, Shan Muthukrishnan, Relative-error cur matrix decompositions, SIAM J. Matrix Anal. Appl. 30 (2) (2008) 844–881.
[20] Carl Eckart, Gale Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.
[21] Björn Engquist, Lexing Ying, et al., A fast directional algorithm for high frequency acoustic scattering in two dimensions, Commun. Math. Sci. 7 (2) (2009) 327–345.
[22] William Fong, Eric Darve, The black-box fast multipole method, J. Comput. Phys. 228 (23) (2009) 8712–8725.
[23] Weihua Geng, Robert Krasny, A treecode-accelerated boundary integral Poisson–Boltzmann solver for electrostatics of solvated biomolecules, J. Comput. Phys. 247 (2013) 62–78.
[24] Zydrunas Gimbutas, Vladimir Rokhlin, A generalized fast multipole method for nonoscillatory kernels, SIAM J. Sci. Comput. 24 (3) (2003) 796–817.
[25] Lars Grasedyck, Wolfgang Hackbusch, Construction and arithmetics of $\mathscr{H}$-matrices, Computing 70 (4) (2003) 295–334.
[26] Alexander G. Gray, Andrew W. Moore, 'N-body' problems in statistical learning, in: Advances in Neural Information Processing Systems, 2001, pp. 521–527.
[27] Leslie Greengard, Vladimir Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 135 (2) (1997) 280–292.
[28] Wolfgang Hackbusch, A sparse matrix arithmetic based on $\mathscr{H}$-matrices. Part I: introduction to $\mathscr{H}$-matrices, Computing 62 (2) (1999) 89–108.
[29] Wolfgang Hackbusch, Steffen Börm, Data-sparse approximation by adaptive $\mathscr{H}^2$-matrices, Computing 69 (1) (2002) 1–35.
[30] Wolfgang Hackbusch, B.N. Khoromskij, A sparse $\mathscr{H}$-matrix arithmetic. Part II: application to multi-dimensional problems, Computing 64 (2) (2000) 21–47.
[31] Nathan Halko, Per-Gunnar Martinsson, Joel A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.
[32] William B. Johnson, Joram Lindenstrauss, Extensions of Lipschitz mappings into a Hilbert space, Contemp. Math. 26 (189–206) (1984) 1.
[33] Robert Krasny, Lei Wang, Fast evaluation of multiquadric RBF sums by a Cartesian treecode, SIAM J. Sci. Comput. 33 (5) (2011) 2341–2355.
[34] Stefan Kurz, Oliver Rain, Sergej Rjasanow, The adaptive cross-approximation technique for the 3D boundary-element method, IEEE Trans. Magn. 38 (2) (2002) 421–424.
[35] Dongryeol Lee, Piyush Sao, Richard Vuduc, Alexander G. Gray, A distributed kernel summation framework for general-dimension machine learning, Stat. Anal. Data Min. ASA Data Sci. J. 7 (1) (2014) 1–13.
[36] William B. March, George Biros, Far-field compression for fast kernel summation methods in high dimensions, Appl. Comput. Harmon. Anal. 43 (1) (2017) 39–75.
[37] William B. March, Bo Xiao, George Biros, ASKIT: approximate skeletonization kernel-independent treecode in high dimensions, SIAM J. Sci. Comput. 37 (2) (2015) A1089–A1110.
[38] Per-Gunnar Martinsson, A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix, SIAM J. Matrix Anal. Appl. 32 (4) (2011) 1251–1274.
[39] Per-Gunnar Martinsson, Compressing rank-structured matrices via randomized sampling, SIAM J. Sci. Comput. 38 (4) (2016) A1959–A1986.
[40] Per-Gunnar Martinsson, Vladimir Rokhlin, Mark Tygert, A randomized algorithm for the decomposition of matrices, Appl. Comput. Harmon. Anal. 30 (1) (2011) 47–68.
[41] Ivi C. Tsantili, Min Hyung Cho, Wei Cai, George Em Karniadakis, A computational stochastic methodology for the design of random meta-materials under geometric constraints, SIAM J. Sci. Comput. 40 (2) (2018) B353–B378.
[42] Lei Wang, Robert Krasny, Svetalana Tlupova, A kernel-independent treecode based on barycentric Lagrange interpolation, arXiv:1902.02250, 2019.
[43] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, Xiaoye S. Li, Superfast multifrontal method for large structured linear systems of equations, SIAM J. Matrix Anal. Appl. 31 (3) (2009) 1382–1411.
[44] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, Xiaoye S. Li, Fast algorithms for hierarchically semiseparable matrices, Numer. Linear Algebra Appl. 17 (6) (2010) 953–976.
[45] Haizhao Yang, A unified framework for oscillatory integral transforms: when to use NUFFT or butterfly factorization?, J. Comput. Phys. (2019).
[46] Norman Yarvin, Vladimir Rokhlin, Generalized Gaussian quadratures and singular value decompositions of integral operators, SIAM J. Sci. Comput. 20 (2) (1998) 699–718.
[47] Xin Ye, Jianlin Xia, Lexing Ying, Analytical low-rank compression via proxy point selection, arXiv:1903.08821, 2019.
[48] Lexing Ying, George Biros, Denis Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, J. Comput. Phys. 196 (2) (2004) 591–626.
[49] Rio Yokota, Huda Ibeid, David Keyes, Fast multipole method as a matrix-free hierarchical low-rank approximation, arXiv:1602.02244, 2016.
[50] Chenhan D. Yu, James Levitt, Severin Reiz, George Biros, Geometry-oblivious FMM for compressing dense SPD matrices, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, 2017, p. 53.
[51] Kezhong Zhao, Marinos N. Vouvakis, Jin-Fa Lee, The adaptive cross approximation algorithm for accelerated method of moments computations of emc problems, IEEE Trans. Electromagn. Compat. 47 (4) (2005) 763–773.