Linear Algebra and its Applications 755 (2027) 216–251

Contents lists available at ScienceDirect

Linear Algebra and its Applications

journal homepage: www.elsevier.com/locate/laa

Hierarchical matrices with Tucker blocks: Construction and forward evaluation

Yingzhou Li [a,b], Jingyu Liu [a,*]

[a] School of Mathematical Sciences, Fudan University, 220 Handan Road, Shanghai, 200433, China
[b] Shanghai Key Laboratory for Contemporary Applied Mathematics, Fudan University, 220 Handan Road, Shanghai, 200433, China

ARTICLE INFO

Article history:
Received 22 January 2026
Received in revised form 21 October 2026
Accepted 28 October 2026
Available online 3 November 2026
Submitted by V. Nick

MSC:
65R20

Keywords:
Fast algorithms
$\mathcal{H}$-matrix
Integral equations
Tucker decomposition

ABSTRACT

In this paper, we introduce hierarchical matrices with Tucker blocks (HMTB) for data-sparse approximation of matrices arising from nonoscillatory and mildly oscillatory kernels. Building on the hierarchical matrix framework, HMTB matrices replace conventional low-rank blocks with Tucker representations. Using high-dimensional interpolation and tensor compression and contractions, we develop algorithms for constructing HMTB matrices and performing matrix-vector multiplication. For an $N \times N$ matrix, these algorithms have linear and quasi-linear complexities in N, respectively. Numerical comparisons with an interpolation-based $\mathcal{H}$-matrix implementation demonstrate storage savings and, in several cases, reduced construction times, while maintaining competitive forward-evaluation costs. The framework is also applicable to quasi-uniform grids.

© 2026 Elsevier Inc. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

* Corresponding author.
E-mail addresses: yingzhouli@fudan.edu.cn (Y. Li), jyliu22@m.fudan.edu.cn (J. Liu).

https://doi.org/10.1016/j.laa.2026.10.041
0024-3795/© 2026 Elsevier Inc. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

1. Introduction

This paper considers the integral equation (IE)

$$a(\boldsymbol{x})u(\boldsymbol{x}) + \int_{\Omega} k(\boldsymbol{x},\boldsymbol{y})u(\boldsymbol{y})\mathrm{d}\boldsymbol{y} = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{1}$$

where $\Omega = [0,1]^d \subset \mathbb{R}^d$. The kernel $k(\boldsymbol{x},\boldsymbol{y})$ and the function $a(\boldsymbol{x})$ are given. We consider kernels arising from Green's functions associated with partial differential equations (PDEs), such as the Laplace and Helmholtz equations, as well as Gaussian kernels. Given $u(\boldsymbol{x})$, computing $f(\boldsymbol{x})$ through (1) is called forward evaluation; given $f(\boldsymbol{x})$, solving (1) for $u(\boldsymbol{x})$ is called inversion. Both problems are important in practice. Although we focus on rectangular domains, the proposed approach also applies to simply-structured non-tensor domains that can be partitioned into tensor subdomains, such as L-shaped domains.

Discretizing (1) by standard methods yields a dense linear system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}, \tag{2}$$

where $\boldsymbol{A} \in \mathbb{C}^{N\times N}$ and N is the number of discretization points. For example, applying the Nyström method on a uniform grid with n points in each direction yields a system of size $N = n^d$:

$$a(\boldsymbol{x}_i)u_i + \sum_{j=1}^{N} K_{i,j}h^d u_j = f(\boldsymbol{x}_i), \quad i = 1,\dots,N, \tag{3}$$

where $u_i \approx u(\boldsymbol{x}_i)$. The entries of $\boldsymbol{K}$ are given by

$$K_{i,j} \coloneqq \begin{cases} k(\boldsymbol{x}_i,\boldsymbol{x}_j), & j \neq i, \\ \dfrac{1}{h^d}\displaystyle\int_{\Omega_i} k(\boldsymbol{x}_i,\boldsymbol{y})\mathrm{d}\boldsymbol{y}, & j = i, \end{cases} \tag{4}$$

where Ω_j is the box of side length $h = 1/n$ centered at $\boldsymbol{x}_j$. Thus, $K_{i,j}h^d$ approximates the integral of $k(\boldsymbol{x}_i,\cdot)$ over Ω_j, with the diagonal integral evaluated separately by numerical quadrature. Although other discretization schemes can be used, this simple Nyström method suffices to demonstrate our approach. It can also be viewed as a collocation method with piecewise-constant basis functions supported on $\{\Omega_i\}$, where the off-diagonal integrals are approximated by point evaluations.

1.1. Related work

Dense direct solvers for (2), such as LU factorization, require $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ storage and thus become prohibitively expensive for large N.

Fast algorithms exploit the fact that off-diagonal blocks of $\boldsymbol{A}$ in (3) are numerically low-rank when the kernel k is smooth away from the diagonal and not highly oscillatory. This property enables efficient storage and matrix operations. Early examples include the *Barnes–Hut algorithm* [3], also known as the tree code, and the *fast multipole method* (FMM) [8,17,22,53], which accelerate matrix-vector multiplication for kernel matrices. Hackbusch and collaborators subsequently introduced *hierarchical matrices* [24,27], including $\mathcal{H}$- and $\mathcal{H}^2$-matrices. They support many matrix operations, such as matrix-vector multiplication, matrix-matrix multiplication, and LU factorization, with quasi-linear or linear complexity [6,24–27,40]. The $\mathcal{H}$- and $\mathcal{H}^2$-matrix frameworks can be viewed as algebraic counterparts of the Barnes–Hut algorithm and FMM, respectively.

Hierarchical block-separable (HBS) matrices, also known as *hierarchically semiseparable* (HSS) matrices [42], form another class of data-sparse matrices for accelerating kernel matrix operations. They are closely related to $\mathcal{H}^2$-matrices under weak admissibility. Various HBS factorization algorithms achieve quasi-linear or linear complexity for matrix-vector multiplication and linear system solution [9,12,19,23,52]. Another class constructs structured block-sparse factorizations of $\boldsymbol{A}$ that can be applied or inverted efficiently. Examples include *recursive skeletonization factorization* (RS) [33,44] and *hierarchical interpolative factorization* (HIF) [34]. These algorithms rely primarily on *interpolative decomposition* [10] and *proxy surfaces* [41]. For smooth, nonsingular kernels, such as Gaussian kernels, low-rank approximation can be applied more aggressively, even to diagonal blocks [50,51].

Although these fast algorithms have low complexity in the matrix size, their costs may grow rapidly, often exponentially, with the dimension d, leading to the *curse of dimensionality* (CoD). Tensor low-rank decompositions are commonly used to mitigate this issue. Kolda and Bader [36] provide a comprehensive introduction to the *CANDECOMP/PARAFAC* (CP) and *Tucker* decompositions, which generalize the matrix singular value decomposition (SVD) to tensors. These decompositions can be computed using algorithms such as the *higher-order SVD* (HOSVD) [14], *higher-order orthogonal iteration* (HOOI) [15], *sequentially truncated HOSVD* (STHOSVD) [49], and their randomized variants [32,45]. In addition, *tensor train*, *tensor ring*, and, more generally, *tensor-network decompositions* are widely used in practice, particularly in computational physics and chemistry [11,46].

Combining hierarchical and tensor decompositions can reduce both asymptotic complexity and prefactors, potentially yielding linear or quasi-linear algorithms. The *hierarchical Kronecker tensor-product* (HKT) approximation [18,28–31] combines these techniques to provide quasi-linear representations of certain high-dimensional integral and elliptic operators. In [13], the quantized tensor train (QTT) decomposition was used to approximate integral operators, and matrix-vector multiplication and inversion algorithms were developed for the resulting representation. Ehrlacher, Grigori, Lombardi, and Song [16] constructed a hierarchical partition to identify numerically low-rank subtensors, which were then compressed using standard tensor decomposition algorithms such as HOSVD [14]. Their problem differs slightly from ours, as they primarily consider

function approximation rather than operator approximation. Other related work can be found in [47]. More recently, the *tensor butterfly algorithm* [35] combines *butterfly factorization* [37–39] with Tucker-type tensor compression for high-dimensional oscillatory integral operators. It extends the complementary low-rank property of conventional butterfly algorithms to a tensor setting, partitions the discretized operator into multilevel subtensors, and compresses the middle-level interactions using Tucker-type interpolative decompositions. By exploiting this tensor structure across levels, the method achieves $\mathcal{O}(n^d)$ computational and storage complexities for a d-dimensional operator discretized as a $2d$-mode tensor with n^{2d} entries.

1.2. Contributions

In this paper, we define *hierarchical matrices with Tucker blocks* (HMTB) by replacing the low-rank blocks of an $\mathcal{H}$-matrix with *Tucker low-rank* (TLR) representations. On tensor grids, the Tucker format mitigates the curse of dimensionality (CoD), reducing storage and computational costs compared with conventional low-rank representations. We show that HMTB matrices require storage that is linear in the number N of discretization points. We also develop construction and matrix-vector multiplication algorithms with linear and quasi-linear complexities in N, respectively. The construction uses multidimensional interpolation, with an error bound established under suitable assumptions on the kernel. For representations generated by the same tensor-product interpolation, the matrix-vector multiplication algorithm uses tensor contractions and has a smaller leading prefactor than its $\mathcal{H}$-matrix counterpart. The approach is also applicable to quasi-uniform grids. Numerical experiments support the complexity analysis and demonstrate the efficiency of HMTB matrices.

Although developed here within the $\mathcal{H}$-matrix framework, the underlying idea can also be incorporated into the kernel-independent FMM [53], $\mathcal{H}^2$-matrices [26], HBS matrices [41], HIF [34], RS [33], and HODLR matrices [1], potentially yielding similar memory and runtime savings. A similar idea was independently incorporated into butterfly factorization for oscillatory kernels in [35]. Algorithms for solving linear systems with HMTB matrices, such as LU factorization, may also be developed analogously to those for $\mathcal{H}$-matrices, but are considerably more involved and left for future work.

1.3. Organization

The remainder of this paper is organized as follows. Section 2 introduces the notation and briefly reviews $\mathcal{H}$-matrices. Section 3 introduces HMTB matrices on uniform grids, establishes their linear storage complexity, and discusses their application to quasi-uniform grids. Section 4 presents the construction and forward evaluation algorithms for HMTB matrices, together with their complexity analysis. Section 5 presents numerical experiments in two and three dimensions to demonstrate the efficiency of HMTB matrices. Finally, Section 6 concludes the paper and discusses future work.

2. Preliminaries

2.1. Notation

For a positive integer N, we denote the index set $\{1, \dots, N\}$ by $[N]$. The notation $|\cdot|$ denotes either the cardinality of a set or the measure of a domain. Vectors are denoted by boldface lowercase letters, e.g., $\boldsymbol{a}$, matrices by boldface uppercase letters, e.g., $\boldsymbol{A}$, and tensors of order three or higher by boldface Euler script letters, e.g., $\boldsymbol{\mathcal{A}}$. We use MATLAB notation throughout the paper. For example, the i-th entry of a vector $\boldsymbol{a}$ is denoted by $\boldsymbol{a}_i$ or $\boldsymbol{a}(i)$. The submatrix of $\boldsymbol{A}$ corresponding to the row and column index sets τ and σ is denoted by $\boldsymbol{A}_{\tau,\sigma}$ or $\boldsymbol{A}(\tau, \sigma)$. A colon indicates all indices along a dimension; for example, the j-th column of $\boldsymbol{A}$ is denoted by $\boldsymbol{A}_{:,j}$ or $\boldsymbol{A}(:, j)$. The conjugate transpose of $\boldsymbol{A}$ is denoted by $\boldsymbol{A}^*$. A matrix $\boldsymbol{U}$ is *orthonormal* if its columns are orthonormal, i.e., $\boldsymbol{U}^*\boldsymbol{U} = \boldsymbol{I}$.

We use multi-indices to describe tensor structures. Consider the two-dimensional case as an example. A tensor grid in two dimensions is a set of discretization points of the form $\{(x_{1;i_1}, x_{2;i_2})\}_{i_1,i_2=1}^{n}$, that is, the Cartesian product of the discretization points in the two coordinate directions. The subscripts 1 and 2 indicate that the discretization points may differ between the two directions. We use the multi-index $\boldsymbol{i} = (i_1, i_2)$ to index each grid point and write $\boldsymbol{x_i} = (x_{1;i_1}, x_{2;i_2})$. Thus, the grid can also be denoted by $\{\boldsymbol{x_i}\}_{\boldsymbol{i}\in[n]^2}$. To arrange the multi-indices as row or column indices of a matrix, we use the one-to-one correspondence

$$\boldsymbol{i} = (i_1, i_2) \longleftrightarrow i = i_1 + (i_2 - 1)n.$$

We further define $|\boldsymbol{i}| \coloneqq i_1 + i_2$ and $\boldsymbol{i}! \coloneqq i_1!i_2!$. These notations extend analogously to the d-dimensional case.

Our tensor notation and terminology are consistent with [36]. The mode-ℓ product of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{n_1\times\cdots\times n_d}$ and a matrix $\boldsymbol{U} \in \mathbb{C}^{m\times n_\ell}$ is denoted by $\boldsymbol{\mathcal{A}} \times_\ell \boldsymbol{U} \in \mathbb{C}^{n_1\times\cdots\times n_{\ell-1}\times m\times n_{\ell+1}\times\cdots\times n_d}$. We denote the contraction of two tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ by $\boldsymbol{\mathcal{A}} \times_{\boldsymbol{a},\boldsymbol{b}} \boldsymbol{\mathcal{B}}$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ specify the corresponding modes of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ to be contracted. For example, let $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{3\times4\times5}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{C}^{4\times5\times7}$. Then $\boldsymbol{C} \coloneqq \boldsymbol{\mathcal{A}} \times_{[2,3],[1,2]} \boldsymbol{\mathcal{B}} \in \mathbb{C}^{3\times7}$, where $\boldsymbol{C}(i, j) = \sum_{\alpha=1}^{4}\sum_{\beta=1}^{5} \boldsymbol{\mathcal{A}}(i, \alpha, \beta)\boldsymbol{\mathcal{B}}(\alpha, \beta, j)$.

A *Tucker decomposition* of a tensor $\boldsymbol{\mathcal{A}}$ with *core tensor* $\boldsymbol{\mathcal{G}} \in \mathbb{C}^{p_1\times\cdots\times p_d}$ and *factor matrices* $\{\boldsymbol{U}_\ell \in \mathbb{C}^{n_\ell\times p_\ell}\}_{\ell=1}^{d}$ is given by

$$\boldsymbol{\mathcal{A}} = \mathrm{Tucker}(\boldsymbol{\mathcal{G}}, \{\boldsymbol{U}_\ell\}_{\ell=1}^{d}) \coloneqq \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}_1 \times_2 \cdots \times_d \boldsymbol{U}_d. \tag{5}$$

The entries of $\boldsymbol{p} = (p_1, \dots, p_d)$ are called the *Tucker bond dimensions* of this representation. They specify the numbers of columns of the factor matrices and the corresponding mode sizes of the core tensor, and need not be minimal. When $p_1 = \cdots = p_d = p$, we call p the *uniform Tucker bond dimension*.

A Tucker decomposition can be compressed without explicitly forming the full tensor. Given a decomposition (5), we first compute QR factorizations of the factor matrices and absorb the resulting triangular factors into the core tensor. We then compute a compressed Tucker decomposition of the updated core tensor and combine its factor matrices with the orthonormal factors obtained from the QR factorizations. For a prescribed relative tolerance ε, the resulting decomposition $\widetilde{\boldsymbol{\mathcal{A}}}$ satisfies $\|\boldsymbol{\mathcal{A}} - \widetilde{\boldsymbol{\mathcal{A}}}\|_{\mathrm{F}} = \mathscr{O}(\varepsilon)\|\boldsymbol{\mathcal{A}}\|_{\mathrm{F}}$ and has factor matrices with orthonormal columns. Assuming that $p_\ell = p$ and $n_\ell = n$ for all ℓ, and that a standard HOSVD- or STHOSVD-type algorithm is used to compress the updated core tensor, the total compression cost is $\mathscr{O}(dnp^2 + dp^{d+1})$. The procedure is summarized in Algorithm 1.

Algorithm 1 TuckerCompress$(\boldsymbol{\mathcal{G}}, \{\boldsymbol{U}_\ell\}_{\ell=1}^d, \varepsilon)$.

Input: A Tucker decomposition $\boldsymbol{\mathcal{A}} = \mathrm{Tucker}(\boldsymbol{\mathcal{G}}, \{\boldsymbol{U}_\ell\}_{\ell=1}^d)$ and a relative tolerance ε
Output: A compressed Tucker decomposition $\widetilde{\boldsymbol{\mathcal{A}}} = \mathrm{Tucker}(\widetilde{\boldsymbol{\mathcal{G}}}, \{\widetilde{\boldsymbol{U}}_\ell\}_{\ell=1}^d)$ satisfying $\|\boldsymbol{\mathcal{A}} - \widetilde{\boldsymbol{\mathcal{A}}}\|_{\mathrm{F}} = \mathscr{O}(\varepsilon)\|\boldsymbol{\mathcal{A}}\|_{\mathrm{F}}$

$\widehat{\boldsymbol{\mathcal{G}}} \leftarrow \boldsymbol{\mathcal{G}}$
for $\ell = 1, \dots, d$ **do**
  Compute the QR factorization $\boldsymbol{U}_\ell = \boldsymbol{Q}_\ell \boldsymbol{R}_\ell$
  $\widehat{\boldsymbol{\mathcal{G}}} \leftarrow \widehat{\boldsymbol{\mathcal{G}}} \times_\ell \boldsymbol{R}_\ell$
end for
Compute a compressed Tucker decomposition $\mathrm{Tucker}(\widetilde{\boldsymbol{\mathcal{G}}}, \{\boldsymbol{W}_\ell\}_{\ell=1}^d)$ of $\widehat{\boldsymbol{\mathcal{G}}}$ with relative tolerance ε
for $\ell = 1, \dots, d$ **do**
  $\widetilde{\boldsymbol{U}}_\ell \leftarrow \boldsymbol{Q}_\ell \boldsymbol{W}_\ell$
end for
return $\widetilde{\boldsymbol{\mathcal{G}}}, \{\widetilde{\boldsymbol{U}}_\ell\}_{\ell=1}^d$

Order-$2d$ tensors are often used to represent interaction matrices between two tensor grids. For example, let $\{\boldsymbol{x_i} = (x_{1;i_1}, x_{2;i_2})\}_{\boldsymbol{i}\in[n]^2}$ and $\{\boldsymbol{y_j} = (y_{1;j_1}, y_{2;j_2})\}_{\boldsymbol{j}\in[n]^2}$ be two tensor grids, and let k be a kernel function. The corresponding interaction matrix $\boldsymbol{A} \in \mathbb{C}^{n^2\times n^2}$ is defined by $\boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j}) = k(\boldsymbol{x_i}, \boldsymbol{y_j})$. It is convenient to view $\boldsymbol{A}$ as an order-4 tensor $\boldsymbol{\mathcal{A}} \in \mathbb{C}^{n\times n\times n\times n}$, with $\boldsymbol{\mathcal{A}}(i_1, i_2, j_1, j_2) = \boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j})$. For such tensors, we use the notation

$$\mathrm{Tucker}\big(\boldsymbol{\mathcal{G}}, \{\boldsymbol{U}_\ell\}_{\ell=1}^d, \{\boldsymbol{V}_\ell\}_{\ell=1}^d\big) \coloneqq \boldsymbol{\mathcal{G}} \times_1 \boldsymbol{U}_1 \times_2 \cdots \times_d \boldsymbol{U}_d \times_{d+1} \boldsymbol{V}_1 \times_{d+2} \cdots \times_{2d} \boldsymbol{V}_d$$

for their Tucker decompositions.

2.2. Hierarchical matrices

The definition of $\mathcal{H}$-matrices is based on the concept of a cluster tree.

Definition 1 *(Cluster tree).* Let I be the index set associated with the grid points arising from the discretization of IE (1). A tree T_I is called a *cluster tree* corresponding to I if the following conditions hold:

1. Every node in T_I is a subset of I.
2. The root of T_I is I.

3. Every node $\tau \in \mathsf{T}_I$ is nonempty.
4. For every $\tau \in \mathsf{T}_I$, let children(τ) denote the set of its children. Then either children$(\tau) = \emptyset$, in which case τ is a leaf node, or children(τ) forms a partition of τ, i.e., $\tau = \sqcup_{\alpha \in \text{children}(\tau)} \alpha$, where $\sqcup$ denotes the union of pairwise disjoint sets.

The cluster tree corresponds to a hierarchical partition of the set of points associated with the matrix. Typically, each point $\boldsymbol{x}_i$ is associated with a computational domain Ω_i, and these domains satisfy $|\Omega_i \cap \Omega_j| = 0$ for $i \neq j$. In this sense, we say that the computational domains are non-overlapping. The computational domains may be chosen as the Voronoi cells [2] associated with the grid points. In particular, for a uniform grid, each Ω_i can be chosen as a box centered at $\boldsymbol{x}_i$. The cluster tree can then be constructed by recursively partitioning the computational domain uniformly into 2^d subdomains. For an index set τ associated with the points $\{\boldsymbol{x}_i\}_{i\in\tau}$, we define $\Omega_\tau := \bigcup_{i\in\tau} \Omega_i$.

In the construction of a cluster tree, a stopping criterion is typically imposed to prevent excessive partitioning by prescribing a maximum leaf size N_0. Specifically, if $|\tau| \leq N_0$, then τ is designated as a leaf node and is not partitioned further.

Example 1 *(A cluster tree on a 2D uniform grid).* Suppose that $\Omega = [0,1]^2$ and that I is associated with the uniform grid points

$$\boldsymbol{x_i} = \left((i_1 - 1)h + \frac{h}{2}, (i_2 - 1)h + \frac{h}{2}\right), \quad \boldsymbol{i} = (i_1, i_2), \quad 1 \leq i_1, i_2 \leq n,$$

where $n \in \mathbb{N}_+$ and $h = 1/n$. The computational domain associated with each grid point is defined by $\Omega_{\boldsymbol{i}} = [(i_1-1)h, i_1 h] \times [(i_2-1)h, i_2 h]$. By recursively partitioning Ω uniformly into four subdomains, we obtain a quadtree cluster tree T_I, whose leaf nodes are shown in Fig. 1.

The structure of $\mathcal{H}$-matrices is determined by an admissibility condition, which specifies whether the interaction matrix between two domains can be approximated by a low-rank matrix. Two types of admissibility conditions are commonly used: weak admissibility and strong admissibility. Fig. 2 illustrates these two admissibility conditions.

Definition 2 *(Weak admissibility).* Two nodes τ and σ in T_I are weakly admissible if Ω_τ and Ω_σ are non-overlapping.

Definition 3 *(Strong admissibility).* Two nodes τ and σ in T_I are strongly admissible if

$$\operatorname{dist}(\Omega_\tau, \Omega_\sigma) \geq \eta \max\big\{\operatorname{diam}(\Omega_\tau), \operatorname{diam}(\Omega_\sigma)\big\},$$

where $\eta > 0$ is a prescribed admissibility parameter,

$$\operatorname{diam}(\Omega_\tau) := \max_{\boldsymbol{x},\boldsymbol{y}\in\Omega_\tau} \|\boldsymbol{x} - \boldsymbol{y}\|$$

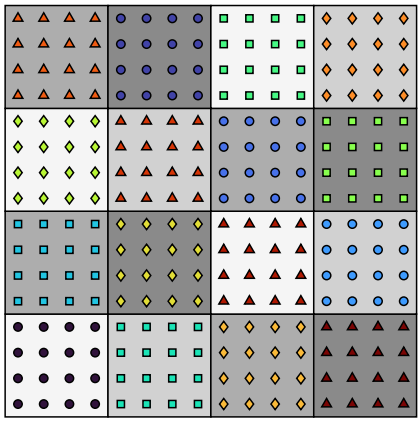


Fig. 1. Leaf nodes in the cluster tree in Example 1. Here $n = 16$, $N = n^2 = 256$ and $N_0 = 16$. Different shapes or colors represent different leaf nodes. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

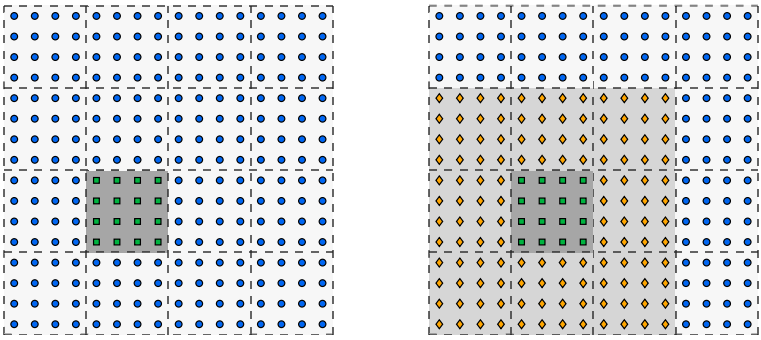


Fig. 2. Weak admissibility (left) and strong admissibility with parameter $\eta = 1/\sqrt{2}$ (right) in Example 1. The grid points we consider are colored by green and marked by squares. The admissible grid points with respect to them are colored by blue and marked by circles, while the inadmissible ones are colored by yellow and marked by diamonds. The dashed domain is the computational domain Ω_τ for each index set τ.

is the diameter of Ω_τ, and

$$\operatorname{dist}(\Omega_\tau, \Omega_\sigma) \coloneqq \min_{\substack{\boldsymbol{x}\in\Omega_\tau \\ \boldsymbol{y}\in\Omega_\sigma}} \|\boldsymbol{x} - \boldsymbol{y}\|$$

is the distance between Ω_τ and Ω_σ. In this case, Ω_τ and Ω_σ are said to be well-separated.

Given a cluster tree T_I and an admissibility condition, the block cluster tree defined below specifies which blocks admit low-rank representations, while its leaves form a partition of $I \times I$. A node of the block cluster tree represents a block $\tau \times \sigma$ and is denoted by (τ, σ), where $\tau, \sigma \in \mathsf{T}_I$. It is called admissible if the pair (τ, σ) satisfies the chosen admissibility condition, and inadmissible otherwise. A block cluster tree may contain both admissible and inadmissible nodes. Moreover, it can be directly verified

that a block cluster tree is itself a cluster tree. An $\mathcal{H}$-matrix is then defined based on the block cluster tree by approximating the submatrices corresponding to admissible leaves by low-rank matrices. The structures of the $\mathcal{H}$-matrices corresponding to weak and strong admissibility with $\eta = 1/\sqrt{2}$ for Example 1 are shown in Fig. 3.

Definition 4 *(Block cluster tree).* Let T_I be a cluster tree associated with the index set I. The *block cluster tree* $\mathsf{T}_{I\times I}$, associated with T_I and an admissibility condition, is a tree whose nodes are subsets of $I \times I$. Starting from the root node $(\tau, \sigma) = (I, I)$, it is constructed recursively as follows:

- If the pair (τ, σ) is admissible, then (τ, σ) is an admissible leaf node.
- If the pair (τ, σ) is inadmissible and at least one of the clusters τ and σ has no children, then (τ, σ) is an inadmissible leaf node.
- Otherwise, (τ, σ) is a non-leaf node whose children are

$$\text{children}\big((\tau, \sigma)\big) = \text{children}(\tau) \times \text{children}(\sigma).$$

Definition 5 *(Hierarchical matrix, $\mathcal{H}$-matrix, [5]).* Let I be the index set associated with the grid points arising from the discretization of IE (1), and let $\mathsf{T}_{I\times I}$ be a block cluster tree. An $\mathcal{H}$-matrix of rank r, denoted by $\boldsymbol{A}^{\mathcal{H}}$, is a matrix defined on $\mathsf{T}_{I\times I}$ such that, for every node $(\tau, \sigma) \in \mathsf{T}_{I\times I}$, the following conditions hold:

- If (τ, σ) is an admissible leaf node, then $\boldsymbol{A}^{\mathcal{H}}_{\tau,\sigma}$ is a matrix of rank at most r, represented as

$$\boldsymbol{A}^{\mathcal{H}}_{\tau,\sigma} = \boldsymbol{U}_{\tau,\sigma}\boldsymbol{G}_{\tau,\sigma}\boldsymbol{V}^{*}_{\tau,\sigma},$$

  where $\boldsymbol{U}_{\tau,\sigma} \in \mathbb{C}^{|\tau|\times r}$, $\boldsymbol{V}_{\tau,\sigma} \in \mathbb{C}^{|\sigma|\times r}$ are orthonormal matrices, and $\boldsymbol{G}_{\tau,\sigma} \in \mathbb{C}^{r\times r}$.
- If (τ, σ) is an inadmissible leaf node, then $\boldsymbol{A}^{\mathcal{H}}_{\tau,\sigma} \in \mathbb{C}^{|\tau|\times|\sigma|}$ is stored as a dense matrix.
- If (τ, σ) is a non-leaf node, then $\boldsymbol{A}^{\mathcal{H}}_{\tau,\sigma}$ is a block matrix whose blocks are $\boldsymbol{A}^{\mathcal{H}}_{\alpha,\beta}$, where $(\alpha, \beta) \in \text{children}\big((\tau, \sigma)\big)$.

When the kernel function is singular only on the diagonal, smooth everywhere else, and not highly oscillatory, as is the case for many commonly used kernels, including the Laplace and low-frequency Helmholtz kernels, the main distinction between weak and strong admissibility lies in whether the numerical rank of the interaction matrix associated with a pair of admissible nodes can be bounded independently of the matrix size. $\mathcal{H}$-matrices based on weak admissibility are sometimes referred to as *hierarchical off-diagonal low-rank* (HODLR) matrices [1], since all off-diagonal blocks are approximated by low-rank matrices. However, the numerical ranks of these blocks may increase mildly with the matrix size [23,33]. In contrast, under strong admissibility, the numerical ranks can be bounded independently of the matrix size for a prescribed accuracy, at the cost of a more complex $\mathcal{H}$-matrix structure [6,27].

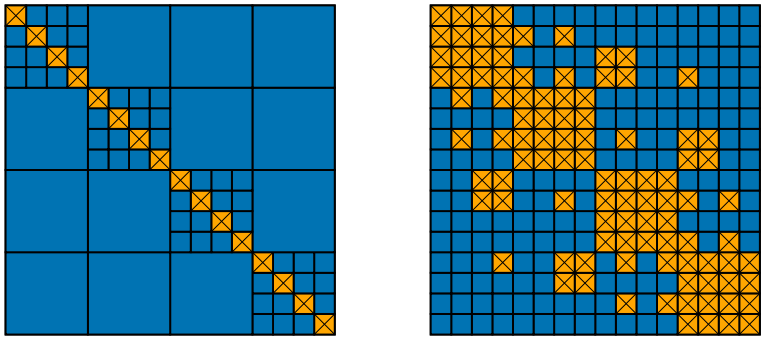


Fig. 3. $\mathcal{H}$-matrices corresponding to Example 1 under weak admissibility (left) and strong admissibility with parameter $\eta = 1/\sqrt{2}$ (right) where $n = 16$, $N = n^2 = 256$ and $N_0 = 4^2$. The yellow and crossed submatrices are dense while the blue ones are low-rank. The weak admissibility implies that there is only one dense submatrix (the diagonal block) in each row block. On the other hand, under strong admissibility, each row block has at most 9 dense submatrices.

Compared with the $\mathcal{O}(N^2)$ storage required for a dense matrix, an $\mathcal{H}$-matrix of rank r has a significantly lower storage cost of $\mathcal{O}(rN \log N)$ [20,24]. However, as we will demonstrate in Section 3.1, for a prescribed accuracy, the rank bound obtained from tensor-product interpolation generally depends exponentially on the dimension d, taking the form $r = p^d$, as shown below in Theorem 2; see also Theorem 4.22 in [25]. This exponential dependence can lead to substantial storage requirements even for moderately large d.

3. Hierarchical matrices with Tucker blocks

When considering the interaction between two distinct tensor-product grids, the Tucker decomposition provides an effective approach to low-rank compression of the corresponding interaction tensor. By combining this idea with the $\mathcal{H}$-matrix framework, we introduce the HMTB matrix in this section, which achieves linear storage complexity. We also demonstrate how the HMTB matrix can be effectively applied to quasi-uniform grids.

3.1. Tucker decomposition from high-dimensional interpolation

Interpolation is widely used to construct low-rank approximations of kernel matrices [7,17]. We first review high-dimensional interpolation on a regular domain and then apply it to construct Tucker decompositions of the interaction matrices between two tensor-product grids.

We begin with the 1D case. Let $f\colon [a,b] \to \mathbb{C}$ be interpolated at the p Chebyshev points

$$\xi_t = \frac{b-a}{2} \cos\left(\frac{(2t-1)\pi}{2p}\right) + \frac{b+a}{2}, \quad 1 \le t \le p.$$

Its interpolant is $\mathscr{J}_{[a,b]}[f](x) \coloneqq \sum_{t=1}^{p} f(\xi_t) L_{[a,b];t}(x)$, where $L_{[a,b];t}(x) \coloneqq \prod_{\substack{j=1 \\ j\neq t}}^{p} (x - \xi_j)/(\xi_t - \xi_j)$ is the tth Lagrange basis polynomial.

In the d-dimensional case, for $B \coloneqq \prod_{\ell=1}^{d} [a_\ell, b_\ell]$ and $f\colon B \to \mathbb{C}$, the Chebyshev interpolant is

$$\mathscr{J}_B[f](\boldsymbol{x}) \coloneqq \sum_{\boldsymbol{t}\in[p]^d} f(\boldsymbol{\xi}_{\boldsymbol{t}}) L_{B;\boldsymbol{t}}(\boldsymbol{x}), \tag{6}$$

where $\boldsymbol{t} = (t_1, \ldots, t_d)$, $\boldsymbol{\xi}_{\boldsymbol{t}} \coloneqq (\xi_{1;t_1}, \ldots, \xi_{d;t_d})$ is the tensor-product Chebyshev point in B corresponding to $\boldsymbol{t}$, and $L_{B;\boldsymbol{t}}(\boldsymbol{x}) \coloneqq \prod_{\ell=1}^{d} L_{[a_\ell,b_\ell];t_\ell}(x_\ell)$ is the associated Lagrange basis polynomial. The following lemma bounds the interpolation error.

Lemma 1 *(Interpolation error in d dimensions, Lemma B.7 in [25]). Let $B = \prod_{\ell=1}^{d} [a_\ell, b_\ell]$ and suppose that $f \in C^p(B;\mathbb{C})$. Then the Chebyshev interpolant (6) satisfies*

$$\|\mathscr{J}_B[f] - f\|_{\infty,B} \leq 2\Lambda_p^{d-1} \sum_{\ell=1}^{d} \left(\frac{b_\ell - a_\ell}{4}\right)^p \frac{\|\partial_\ell^p f\|_{\infty,B}}{p!}, \tag{7}$$

where Λ_p is the Lebesgue constant and satisfies $\Lambda_p \sim 2\log(p)/\pi$; see Theorem 15.2 in [48].

When this interpolation estimate is applied to kernel functions on two distinct boxes, the convergence rate also depends on the geometric separation between the boxes. In particular, the strong admissibility parameter η controls the ratio between the box diameters and their separation distance, and therefore enters the interpolation error estimate derived in Theorem 2. Let $k(\boldsymbol{x}, \boldsymbol{y})$ be a kernel function, and let $B_\tau = \prod_{\ell=1}^{d} [a_\ell, b_\ell]$ and $B_\sigma = \prod_{\ell=1}^{d} [c_\ell, d_\ell]$ be two disjoint boxes in $\mathbb{R}^d$. Applying (6) successively with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$ gives

$$k(\boldsymbol{x}, \boldsymbol{y}) \approx \sum_{\boldsymbol{t}\in[p]^d} \sum_{\boldsymbol{s}\in[p]^d} k(\boldsymbol{\xi}_{\boldsymbol{t}}, \boldsymbol{\eta}_{\boldsymbol{s}}) L_{B_\tau;\boldsymbol{t}}(\boldsymbol{x}) L_{B_\sigma;\boldsymbol{s}}(\boldsymbol{y}). \tag{8}$$

Suppose that $\{\boldsymbol{x}_{\boldsymbol{i}} = (x_{1;i_1}, \ldots, x_{d;i_d}) : \boldsymbol{i} \in [n]^d\} \subset B_\tau$ and $\{\boldsymbol{y}_{\boldsymbol{j}} = (y_{1;j_1}, \ldots, y_{d;j_d}) : \boldsymbol{j} \in [n]^d\} \subset B_\sigma$ are two tensor-product grids, each containing n^d points. Their interaction matrix $\boldsymbol{K} \in \mathbb{C}^{n^d \times n^d}$ is defined by $\boldsymbol{K}(\boldsymbol{i}, \boldsymbol{j}) = k(\boldsymbol{x}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{j}})$. By (8), its entries satisfy

$$\boldsymbol{K}(\boldsymbol{i}, \boldsymbol{j}) \approx \sum_{\boldsymbol{t}\in[p]^d} \sum_{\boldsymbol{s}\in[p]^d} k(\boldsymbol{\xi}_{\boldsymbol{t}}, \boldsymbol{\eta}_{\boldsymbol{s}}) L_{B_\tau;\boldsymbol{t}}(\boldsymbol{x}_{\boldsymbol{i}}) L_{B_\sigma;\boldsymbol{s}}(\boldsymbol{y}_{\boldsymbol{j}}). \tag{9}$$

Define

$$\begin{aligned} \boldsymbol{U}_\ell(i_\ell, t_\ell) &= L_{[a_\ell,b_\ell];t_\ell}(x_{\ell;i_\ell}), \quad \boldsymbol{U}_\ell \in \mathbb{C}^{n\times p}, \quad 1 \leq \ell \leq d, \\ \boldsymbol{V}_\ell(j_\ell, s_\ell) &= L_{[c_\ell,d_\ell];s_\ell}(y_{\ell;j_\ell}), \quad \boldsymbol{V}_\ell \in \mathbb{C}^{n\times p}, \quad 1 \leq \ell \leq d, \\ \boldsymbol{G}(\boldsymbol{t}, \boldsymbol{s}) &= k(\boldsymbol{\xi}_{\boldsymbol{t}}, \boldsymbol{\eta}_{\boldsymbol{s}}), \quad \boldsymbol{G} \in \mathbb{C}^{p^d \times p^d}. \end{aligned} \tag{10}$$

Then (9) can be written as

$$\boldsymbol{K} \approx \boldsymbol{U}\boldsymbol{G}\boldsymbol{V}^{\top}, \tag{11}$$

where $\boldsymbol{U} = \boldsymbol{U}_d \otimes \cdots \otimes \boldsymbol{U}_1 \in \mathbb{C}^{n^d \times p^d}$ and $\boldsymbol{V} = \boldsymbol{V}_d \otimes \cdots \otimes \boldsymbol{V}_1 \in \mathbb{C}^{n^d \times p^d}$. The matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are called the *basis matrices*, while $\boldsymbol{G}$ is called the *core matrix*.

The decomposition (11) has rank at most p^d and provides a low-rank representation of admissible leaf blocks in $\mathcal{H}$-matrices [24,25]. Explicitly storing $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{G}$ requires $2n^d p^d + p^{2d}$ entries. This representation can be further compressed by orthogonalizing $\boldsymbol{U}$ and $\boldsymbol{V}$ and truncating the modified core matrix $\boldsymbol{G}$ according to the prescribed accuracy.

However, owing to the inherent tensor-product structure of the grids, it is more advantageous to represent (9) in Tucker format rather than explicitly constructing $\boldsymbol{U}$ and $\boldsymbol{V}$. Let $\boldsymbol{\mathcal{K}} \in \mathbb{C}^{n\times\cdots\times n}$ and $\boldsymbol{\mathcal{G}} \in \mathbb{C}^{p\times\cdots\times p}$ be the order-$2d$ tensorizations of $\boldsymbol{K}$ and $\boldsymbol{G}$, respectively, defined by

$$\boldsymbol{\mathcal{K}}(i_1,\ldots,i_d,j_1,\ldots,j_d) = \boldsymbol{K}(\boldsymbol{i},\boldsymbol{j}), \quad \boldsymbol{\mathcal{G}}(t_1,\ldots,t_d,s_1,\ldots,s_d) = \boldsymbol{G}(\boldsymbol{t},\boldsymbol{s}).$$

Using (10), we can express the approximation (9) in Tucker format as

$$\boldsymbol{\mathcal{K}} \approx \mathrm{Tucker}\big(\boldsymbol{\mathcal{G}}, \{\boldsymbol{U}_\ell\}_{\ell=1}^d, \{\boldsymbol{V}_\ell\}_{\ell=1}^d\big). \tag{12}$$

The matrix approximation (11) is the matricization of the Tucker approximation (12), obtained by grouping the first d modes into the row index and the last d modes into the column index. The two forms therefore represent the same interpolant. In the Tucker representation, $\boldsymbol{U}$ and $\boldsymbol{V}$ are retained implicitly through their one-dimensional factors, without explicitly forming the Kronecker products. The core matrix and the core tensor both contain p^{2d} entries. Explicitly storing the two basis matrices of size $n^d \times p^d$ requires $2n^d p^d$ entries, whereas storing their $2d$ one-dimensional factors of size $n \times p$ requires only $2dnp$ entries. Therefore, for fixed n and p, the storage required for the Tucker factors grows linearly with d, whereas that required for the conventional basis matrices grows exponentially with d. We emphasize that this reduction does not eliminate the curse of dimensionality. The core tensor still contains p^{2d} entries, so an exponential dimension dependence remains. Rather, the Tucker representation mitigates the curse of dimensionality by reducing the storage and computational costs associated with the basis factors from exponential to linear dependence on d.

After the interpolation, we apply Algorithm 1 to obtain orthonormal factor matrices and reduce the Tucker bond dimensions according to the prescribed accuracy. The resulting computational complexity is $\mathcal{O}\big(dp^2 n + dp^{2d+1}\big)$. For fixed d and p and sufficiently large n, the leading term is $dp^2 n$, which depends linearly on d. The complete construction procedure is presented in Section 4.1.

We conclude this section with an error estimate for the interpolation approximation (8), which relates the Tucker bond dimensions to the approximation accuracy for kernel functions satisfying the so-called *asymptotic smoothness* property.

Definition 6 *(Asymptotic smoothness, [5,25]).* A kernel function $k\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ is said to be asymptotically smooth if there exist constants $C_{\mathrm{as}}, \gamma > 0$ and $s \geq 0$, depending only on k, such that

$$\left|\partial_{\boldsymbol{x}}^{\boldsymbol{\alpha}} \partial_{\boldsymbol{y}}^{\boldsymbol{\beta}} k(\boldsymbol{x}, \boldsymbol{y})\right| \leq C_{\mathrm{as}} (\boldsymbol{\alpha} + \boldsymbol{\beta})! \gamma^{|\boldsymbol{\alpha}+\boldsymbol{\beta}|} \left\|\boldsymbol{x} - \boldsymbol{y}\right\|^{-s-|\boldsymbol{\alpha}|-|\boldsymbol{\beta}|} \tag{13}$$

for all $\boldsymbol{x} \neq \boldsymbol{y}$ and $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}^d$ satisfying $|\boldsymbol{\alpha}| + |\boldsymbol{\beta}| > 0$. Here, $\partial_{\boldsymbol{x}}^{\boldsymbol{\alpha}} \coloneqq \partial_{x_1}^{\alpha_1} \cdots \partial_{x_d}^{\alpha_d}$ and $\partial_{\boldsymbol{y}}^{\boldsymbol{\beta}} \coloneqq \partial_{y_1}^{\beta_1} \cdots \partial_{y_d}^{\beta_d}$ denote the corresponding multidimensional partial differential operators. For a multi-index $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_d)$, we use the convention $\boldsymbol{\nu}! \coloneqq \prod_{\ell=1}^d \nu_\ell!$; in particular, $(\boldsymbol{\alpha} + \boldsymbol{\beta})! = \prod_{\ell=1}^d (\alpha_\ell + \beta_\ell)!$.

A related one-sided interpolation estimate for asymptotically smooth kernels is given in [5, Theorem 3.18]. When such estimates are available in both variables, successive interpolation in $\boldsymbol{x}$ and $\boldsymbol{y}$, together with the stability bounds of the interpolation operators, gives a two-sided estimate. The following theorem provides such an estimate under the asymptotic smoothness assumption and admissibility convention used here.

Theorem 2 *(Interpolation error for asymptotically smooth kernels). Suppose that k is an asymptotically smooth kernel on $B_\tau \times B_\sigma$, where $B_\tau = \prod_{\ell=1}^d [a_\ell, b_\ell]$ and $B_\sigma = \prod_{\ell=1}^d [c_\ell, d_\ell]$ are two boxes in $\mathbb{R}^d$. If*

$$\operatorname{dist}(B_\tau, B_\sigma) \geq \eta \max\bigl\{\operatorname{diam}(B_\tau), \operatorname{diam}(B_\sigma)\bigr\},$$

then the error $r(\boldsymbol{x}, \boldsymbol{y})$ of the Chebyshev interpolation (8), defined by

$$r(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{t}, \boldsymbol{s} \in [p]^d} k(\boldsymbol{\xi}_{\boldsymbol{t}}, \boldsymbol{\eta}_{\boldsymbol{s}}) L_{B_\tau; \boldsymbol{t}}(\boldsymbol{x}) L_{B_\sigma; \boldsymbol{s}}(\boldsymbol{y}) - k(\boldsymbol{x}, \boldsymbol{y}),$$

satisfies

$$\left\|r\right\|_{\infty, B_\tau \times B_\sigma} \leq \frac{4 C_{\mathrm{as}} \gamma^p \Lambda_p^{2d-1} d}{(4\eta)^p} \operatorname{dist}(B_\tau, B_\sigma)^{-s}, \tag{14}$$

where s, C_{as}, and γ are the constants in (13), and Λ_p is the Lebesgue constant in (7).

Proof. Let $\mathcal{J}_\tau^{\boldsymbol{x}}$ and $\mathcal{J}_\sigma^{\boldsymbol{y}}$ denote the tensor-product Chebyshev interpolation operators with respect to $\boldsymbol{x}$ on B_τ and $\boldsymbol{y}$ on B_σ, respectively. Then

$$\begin{aligned} r &= \mathcal{J}_\tau^{\boldsymbol{x}} \mathcal{J}_\sigma^{\boldsymbol{y}} k - k \\ &= \mathcal{J}_\tau^{\boldsymbol{x}} \bigl(\mathcal{J}_\sigma^{\boldsymbol{y}} k - k\bigr) + \bigl(\mathcal{J}_\tau^{\boldsymbol{x}} k - k\bigr). \end{aligned}$$

The operator norm of a d-dimensional tensor-product interpolation operator with respect to the uniform norm is bounded by Λ_p^d. Therefore, applying Lemma 1 separately with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$, and using $\Lambda_p \geq 1$, we obtain

$$\|r\|_{\infty,B_\tau\times B_\sigma} \le 2\Lambda_p^{2d-1}\left[\sum_{\ell=1}^{d}\left(\frac{b_\ell-a_\ell}{4}\right)^p \frac{\|\partial_{x_\ell}^p k\|_{\infty,B_\tau\times B_\sigma}}{p!} + \sum_{\ell=1}^{d}\left(\frac{d_\ell-c_\ell}{4}\right)^p \frac{\|\partial_{y_\ell}^p k\|_{\infty,B_\tau\times B_\sigma}}{p!}\right].$$

By the asymptotic smoothness property (13), for every $1 \le \ell \le d$,

$$\|\partial_{x_\ell}^p k\|_{\infty,B_\tau\times B_\sigma} \le C_{\mathrm{as}} p!\gamma^p \operatorname{dist}(B_\tau,B_\sigma)^{-s-p},$$
$$\|\partial_{y_\ell}^p k\|_{\infty,B_\tau\times B_\sigma} \le C_{\mathrm{as}} p!\gamma^p \operatorname{dist}(B_\tau,B_\sigma)^{-s-p}.$$

Moreover,

$$b_\ell - a_\ell \le \operatorname{diam}(B_\tau), \qquad d_\ell - c_\ell \le \operatorname{diam}(B_\sigma).$$

Substituting these estimates into the preceding inequality gives

$$\|r\|_{\infty,B_\tau\times B_\sigma} \le 2C_{\mathrm{as}}\gamma^p\Lambda_p^{2d-1} d \operatorname{dist}(B_\tau,B_\sigma)^{-s} \times\left[\left(\frac{\operatorname{diam}(B_\tau)}{4\operatorname{dist}(B_\tau,B_\sigma)}\right)^p + \left(\frac{\operatorname{diam}(B_\sigma)}{4\operatorname{dist}(B_\tau,B_\sigma)}\right)^p\right].$$

Finally, the strong admissibility condition implies

$$\frac{\operatorname{diam}(B_\tau)}{\operatorname{dist}(B_\tau,B_\sigma)} \le \frac{1}{\eta}, \qquad \frac{\operatorname{diam}(B_\sigma)}{\operatorname{dist}(B_\tau,B_\sigma)} \le \frac{1}{\eta}.$$

Hence,

$$\|r\|_{\infty,B_\tau\times B_\sigma} \le \frac{4C_{\mathrm{as}}\gamma^p\Lambda_p^{2d-1}d}{(4\eta)^p}\operatorname{dist}(B_\tau,B_\sigma)^{-s},$$

which proves (14). □

For fixed d, the Lebesgue constant satisfies $\Lambda_p = \mathcal{O}(\log p)$. Hence, if $4\eta > \gamma$, the polylogarithmic factor Λ_p^{2d-1} is dominated by the exponential decay of $\big(\gamma/(4\eta)\big)^p$. Since a larger η imposes stronger separation, it also improves the convergence rate in (14). Thus, for sufficiently well-separated boxes B_τ and B_σ, the absolute interpolation error decreases nearly exponentially with p. Moreover, for fixed d, η, and a prescribed accuracy relative to the scale $\operatorname{dist}(B_\tau,B_\sigma)^{-s}$, the resulting rank bound p^d is independent of the cardinalities of the corresponding index clusters.

The constant γ in the general asymptotic smoothness estimate may be conservative because it accounts for arbitrary mixed derivatives, whereas the proof of Theorem 2 requires only single-coordinate derivatives. Let $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{y}$ and $r = \|\boldsymbol{z}\|$. For $m \ge 1$, the two-dimensional logarithmic kernel and the three-dimensional Newton kernel satisfy

$$\left|\partial_{z_\ell}^m \log r\right| \le m! r^{-m}, \qquad \left|\partial_{z_\ell}^m \frac{1}{r}\right| \le m! r^{-m-1},$$

respectively, with the same bounds for derivatives with respect to x_ℓ and y_ℓ. Thus, one may take $\gamma = 1$ for the derivatives required in the interpolation analysis. Since $4\eta = 4/\sqrt{d} > 1$ for $\eta = 1/\sqrt{d}$ and $d \in \{2, 3\}$, the adopted admissibility parameter guarantees the nearly exponential convergence predicted by the interpolation estimate for these two kernels. For the Helmholtz kernel, the constants in the asymptotic-smoothness estimate depend on the wavenumber. Therefore, the above estimate should not be interpreted as a frequency-uniform rank bound; in this work we consider only a fixed mildly oscillatory regime.

3.2. Hierarchical matrices with Tucker blocks

Section 3.1 shows how tensor-product interpolation naturally yields Tucker approximations of interaction matrices on block domains where the kernel is sufficiently smooth. The matrix representation (11) and the Tucker representation (12) describe the same interpolant. We now combine these local Tucker approximations of admissible leaf blocks with a hierarchical block partition to define HMTB matrices. To apply this construction blockwise, we require the cluster tree and block cluster tree to inherit the tensor-product structure of the grid.

Definition 7 *(Tensor node cluster tree).* Let $I = \{\boldsymbol{i} : \boldsymbol{i} \in [n]^d\}$ be the index set associated with the tensor-product grid arising from the discretization (3). A cluster tree T_I is called a *tensor node cluster tree* if every node $\tau \in \mathsf{T}_I$ can be expressed as $\tau = \tau_1 \times \cdots \times \tau_d$, where $\tau_\ell \subseteq [n]$ is a contiguous index set associated with the ℓ-th dimension for each $\ell \in [d]$.

Definition 8 *(Tensor node block cluster tree).* Let T_I be a tensor node cluster tree with root index set I. The block cluster tree $\mathsf{T}_{I\times I}$ constructed from T_I and an admissibility condition is called a *tensor node block cluster tree*.

A key property of the tensor node cluster tree is that the grid points associated with each node retain a tensor-product structure, while their computational domain forms a d-dimensional box. On a tensor node block cluster tree, the following definition retains the hierarchical block structure of Definition 5. For each admissible leaf block, the matrix low-rank representation is replaced by a Tucker representation of its order-$2d$ tensorization. Inadmissible leaf blocks remain dense, and non-leaf blocks retain the same recursive structure.

Definition 9 *(Hierarchical matrices with Tucker blocks, HMTB matrices).* Suppose $\mathsf{T}_{I\times I}$ is the tensor node block cluster tree with root index set I. We define a matrix $\boldsymbol{A}^{\mathrm{HMTB}}$ as an HMTB matrix on $\mathsf{T}_{I\times I}$ with Tucker bond dimensions bounded by p if the following conditions hold:

- If (τ, σ) is an admissible leaf, denoting the order-$2d$ tensorization of $\boldsymbol{A}_{\tau,\sigma}^{\mathrm{HMTB}}$ by $\boldsymbol{\mathcal{A}}_{\tau,\sigma}^{\mathrm{HMTB}}$, we have

$$\boldsymbol{\mathcal{A}}_{\tau,\sigma}^{\mathrm{HMTB}} = \mathrm{Tucker}\big(\boldsymbol{\mathcal{G}}_{\tau,\sigma}, \{\boldsymbol{U}_{\tau,\sigma;\ell}\}_{\ell=1}^{d}, \{\boldsymbol{V}_{\tau,\sigma;\ell}\}_{\ell=1}^{d}\big),$$

  for some $\boldsymbol{U}_{\tau,\sigma;\ell} \in \mathbb{C}^{|\tau_\ell| \times p}$, $\boldsymbol{V}_{\tau,\sigma;\ell} \in \mathbb{C}^{|\sigma_\ell| \times p}$, and $\boldsymbol{\mathcal{G}}_{\tau,\sigma} \in \mathbb{C}^{p \times \cdots \times p}$.
- If (τ, σ) is an inadmissible leaf, then $\boldsymbol{A}_{\tau,\sigma}^{\mathrm{HMTB}}$ is a dense matrix.
- If (τ, σ) is a non-leaf node, then $\boldsymbol{A}_{\tau,\sigma}^{\mathrm{HMTB}}$ is a block matrix with blocks $\boldsymbol{A}_{\alpha,\beta}^{\mathrm{HMTB}}$, where $(\alpha, \beta) \in \mathrm{children}\big((\tau, \sigma)\big)$.

It follows from (11) and (12) that, when both representations are derived from the same high-dimensional interpolation, the matrix representation used in $\mathcal{H}$-matrices, with rank at most p^d, and the TLR representation with uniform Tucker bond dimension p achieve the same approximation accuracy. However, the tensor structure of the latter leads to a lower storage cost. It should be emphasized that p^d is the rank bound obtained directly from the tensor-product interpolation. A subsequent algebraic recompression may reduce the numerical rank of the corresponding $\mathcal{H}$-matrix substantially. Therefore, the following complexity comparison concerns the interpolation-generated representations and should not be interpreted as a lower bound on the rank or complexity of a general algebraically compressed $\mathcal{H}$-matrix. In particular, the storage complexity of an HMTB matrix is linear in the matrix dimension N, as stated in the following proposition.

Proposition 3 *(Storage of HMTB matrices under weak admissibility). For fixed $d \geq 2$, let $N_0 = n_0^d$ and $I = \{\boldsymbol{i} : \boldsymbol{i} \in [n]^d\}$ be the index set associated with the tensor-product grid arising from the discretization (3), where $N = n^d$. Suppose that the tensor node cluster tree T_I is constructed by recursive 2^d-way partitioning until every leaf node contains at most N_0 points, and let $\mathsf{T}_{I\times I}$ be the corresponding tensor node block cluster tree under the weak admissibility condition. If $\boldsymbol{A}^{\mathrm{HMTB}}$ is an HMTB matrix with Tucker bond dimensions bounded by p and $p \leq n_0 \leq 2p$, then its storage complexity is $\mathcal{O}(p^d N)$, where the prefactor depends only on d.*[1]

Proof. For simplicity, assume that $N = n^d = 2^{dL} n_0^d$, where L is the maximum level of T_I. The general case follows from the same argument with modified constant factors. At level ℓ, the number of clusters is $2^{d\ell}$, and each cluster contains $N/2^{d\ell}$ indices. Under weak admissibility, each cluster is associated with at most one inadmissible block and $2^d - 1$ admissible blocks.

Using the storage cost of TLR matrices discussed in Section 3.1, the storage of $\boldsymbol{A}^{\mathrm{HMTB}}$, denoted by $S^{\mathrm{HMTB}}(N)$, satisfies

[1] Hereafter, the prefactors in all complexity estimates are assumed to depend only on d.

$$
\begin{aligned}
S^{\mathrm{HMTB}}(N) &\le \sum_{\ell=1}^{L} 2^{d\ell}(2^d-1)\big(2dn_0 2^{L-\ell}p + p^{2d}\big) + 2^{dL}n_0^{2d} \\
&\le \left(\frac{2^d d(2^d-1)}{2^{d-1}-1}\frac{p}{n_0^{d-1}} + 2^d\frac{p^{2d}}{n_0^d} + n_0^d\right) N \\
&\le \left(\frac{2^d d(2^d-1)}{2^{d-1}-1}p^{2-d} + 2^d p^d + 2^d p^d\right) N \\
&= \mathcal{O}\big(p^d N\big).
\end{aligned}
\tag{15}
$$

This completes the proof. □

An argument analogous to that in the proof of Proposition 3 gives the following bound on the storage cost $S^{\mathcal{H}}(N)$ of the corresponding $\mathcal{H}$-matrix under weak admissibility:

$$
\begin{aligned}
S^{\mathcal{H}}(N) &\le \sum_{\ell=1}^{L} 2^{d\ell}(2^d-1)\left(2\frac{N}{2^{d\ell}}p^d + p^{2d}\right) + 2^{dL}n_0^{2d} \\
&\le \left(2(2^d-1)Lp^d + 2^d\frac{p^{2d}}{n_0^d} + n_0^d\right) N \\
&\le \left(\frac{2(2^d-1)}{d}p^d\log_2\big(N/N_0\big) + 2^d p^d + 2^d p^d\right) N \\
&= \mathcal{O}\big(p^d N\log_2(N)\big).
\end{aligned}
\tag{16}
$$

For the interpolation-generated representations considered here, explicitly storing the basis matrices of admissible $\mathcal{H}$-matrix blocks requires $\mathcal{O}(p^d N\log_2 N)$ entries. For HMTB matrices, summing the factor-matrix contributions in (15) gives $\mathcal{O}(pN/n_0^{d-1}) = \mathcal{O}(p^{2-d}N)$, using $d \ge 2$ and $p \le n_0 \le 2p$. This dependence on p includes the choice of leaf size $N_0 = n_0^d$ and the corresponding block partition. The Tucker cores and dense leaf blocks together require $\mathcal{O}(p^d N)$ storage, so the total storage remains $\mathcal{O}(p^d N)$, which is linear in N for fixed d and p.

Similar results can be obtained under strong admissibility, with only the prefactor in the storage estimate being different.

Proposition 4 *(Storage of HMTB matrices under strong admissibility). Under the same conditions as in Proposition 3, except that $\mathsf{T}_{I\times I}$ is constructed under the strong admissibility condition with $\eta = 1/\sqrt{d}$, the storage complexity of an HMTB matrix $\boldsymbol{A}^{\mathrm{HMTB}}$ with Tucker bond dimensions bounded by p is $\mathcal{O}(p^d N)$.*

Proof. The proof is analogous to that of Proposition 3. The only difference is that, under strong admissibility, the numbers of inadmissible and admissible blocks associated with each cluster are bounded by 3^d and $6^d - 3^d$, respectively. □

Another limitation arises from the size of the leaf clusters. Under the present tensor-grid construction, a leaf contains $N_0 = n_0^d$ points, with n_0 chosen to be comparable to the interpolation order p. Hence, even for moderate p, the leaf size grows exponentially with the dimension. This limits the polynomial orders that can be used in genuinely high-dimensional problems and makes the present construction most suitable for low-to-moderate spatial dimensions. Therefore, the complexity estimates in this work should be interpreted for fixed d, rather than as dimension-uniform bounds.

3.3. HMTB matrices on a quasi-uniform grid

The HMTB matrix proposed in Section 3.2 requires a uniform tensor grid. However, nonuniform grids may arise from perturbed data or finite element discretizations on nonuniform meshes. Fully nonuniform discretizations can be challenging for HMTB matrices and may also produce deep and unbalanced cluster trees for $\mathcal{H}$-matrices, potentially destroying their quasi-linear scaling. We therefore consider quasi-uniform grids, which can be viewed as perturbations of uniform grids. In this section, we apply HMTB matrices to the IE (1) discretized on a quasi-uniform grid by interpolating between the tensor grid and the quasi-uniform grid, as in the nonuniform fast Fourier transform [4,21].

Suppose that the quasi-uniform grid points $\{\boldsymbol{x}_i\}_{i=1}^N$ are associated with computational cells $\{\Omega_i^{\mathrm{qu}}\}_{i=1}^N$ having mutually disjoint interiors and satisfying $\Omega = \bigcup_{i=1}^N \Omega_i^{\mathrm{qu}}$. For example, the cells may be the triangular elements of a quasi-uniform finite element mesh, with each $\boldsymbol{x}_i$ chosen as the centroid of Ω_i^{qu}, as illustrated in Fig. 4. We discretize the IE (1) as

$$a(\boldsymbol{x}_i)u_i + \sum_{j=1}^{N} K_{i,j}\left|\Omega_j^{\mathrm{qu}}\right|u_j = f(\boldsymbol{x}_i), \qquad i = 1, \ldots, N,$$

where

$$K_{i,j} \coloneqq \begin{cases} k(\boldsymbol{x}_i, \boldsymbol{x}_j), & i \neq j, \\ \dfrac{1}{\left|\Omega_i^{\mathrm{qu}}\right|} \displaystyle\int_{\Omega_i^{\mathrm{qu}}} k(\boldsymbol{x}_i, \boldsymbol{y})\mathrm{d}\boldsymbol{y}, & i = j. \end{cases}$$

Throughout this section, we assume that the input vector $\boldsymbol{u}$ is sampled from a smooth function u, i.e., $u_i = u(\boldsymbol{x}_i)$. We introduce an auxiliary uniform tensor grid $\{\hat{\boldsymbol{x}}_t\}_{t=1}^M$, where $M = m^d$ and m is the number of grid points in each direction, together with its computational cells $\{\Omega_t^{\mathrm{uni}}\}_{t=1}^M$. Let $\hat{\boldsymbol{A}}$ be the matrix defined by (3) on this grid. We further introduce grid-transfer matrices $\boldsymbol{S}\colon \mathbb{C}^N \to \mathbb{C}^M$ and $\boldsymbol{T}\colon \mathbb{C}^M \to \mathbb{C}^N$, which transfer data from the quasi-uniform grid to the uniform grid and vice versa. We construct these matrices using the local piecewise-constant L^2-projections described below. Their entries satisfy $\boldsymbol{S}_{t,i} = \boldsymbol{T}_{i,t} = 0$ whenever $|\Omega_t^{\mathrm{uni}} \cap \Omega_i^{\mathrm{qu}}| = 0$. We assume that each cell has

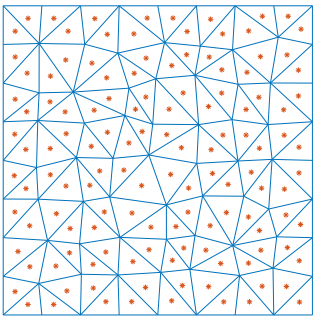


Fig. 4. Triangulation of $[0,1]^2$. The triangle edges are shown in blue, and their centroids are marked by orange stars.

positive-volume overlap with at most a fixed number of cells of the other mesh under grid refinement. Then

$$\boldsymbol{A}\boldsymbol{u} \approx \boldsymbol{T}\hat{\boldsymbol{A}}\boldsymbol{S}\boldsymbol{u}. \tag{17}$$

Specifically, $\boldsymbol{S}\boldsymbol{u}$ transfers the input to the uniform grid, $\hat{\boldsymbol{A}}$ performs the matrix-vector multiplication there, and $\boldsymbol{T}$ transfers the result back to the quasi-uniform grid. Since $\hat{\boldsymbol{A}}$ admits an HMTB approximation and $\boldsymbol{S}$ and $\boldsymbol{T}$ are sparse, this computation can be performed efficiently. Moreover, replacing $\hat{\boldsymbol{A}}$ by its HMTB approximation $\hat{\boldsymbol{A}}^{\mathrm{HMTB}}$ yields a data-sparse approximation to the action of $\boldsymbol{A}$ on vectors sampled from smooth functions.

We next discuss the choice of M and the construction of $\boldsymbol{T}$ and $\boldsymbol{S}$. The choice $M \approx \rho^d N$, where $\rho \geq 1$ is a small constant, typically provides a suitable balance between accuracy and efficiency; see Section 5.3 for numerical results.

We construct the grid-transfer matrices using an L^2-projection. A vector $\boldsymbol{u} \in \mathbb{C}^N$ on the quasi-uniform grid is identified with the piecewise-constant function $u^{\mathrm{qu}}(\boldsymbol{x}) = \sum_{i=1}^{N} u_i \mathbf{1}_{\Omega_i^{\mathrm{qu}}}(\boldsymbol{x})$. Here, $\mathbf{1}_D$ denotes the indicator function of a set D, equal to 1 on D and 0 outside D. Its approximation on the uniform grid is written as $u^{\mathrm{uni}}(\boldsymbol{x}) = \sum_{t=1}^{M} \hat{u}_t \mathbf{1}_{\Omega_t^{\mathrm{uni}}}(\boldsymbol{x})$. Projecting u^{qu} onto $\mathbf{1}_{\Omega_t^{\mathrm{uni}}}$ gives

$$\hat{u}_t = \sum_{i=1}^{N} \frac{\left|\Omega_t^{\mathrm{uni}} \cap \Omega_i^{\mathrm{qu}}\right|}{\left|\Omega_t^{\mathrm{uni}}\right|} u_i.$$

Hence, the transfer matrices from the quasi-uniform grid to the uniform grid and vice versa are $\boldsymbol{S}_{t,i} = |\Omega_t^{\mathrm{uni}} \cap \Omega_i^{\mathrm{qu}}|/|\Omega_t^{\mathrm{uni}}|$ and $\boldsymbol{T}_{i,t} = |\Omega_i^{\mathrm{qu}} \cap \Omega_t^{\mathrm{uni}}|/|\Omega_i^{\mathrm{qu}}|$, respectively. Under the stated bounded-overlap assumption, each row and column of $\boldsymbol{S}$ and $\boldsymbol{T}$ contains only $\mathscr{O}(1)$ nonzero entries. Thus, both matrices are sparse, with $\mathscr{O}(\max\{M,N\}) = \mathscr{O}(M)$ nonzero entries. Higher-order grid-transfer schemes may also be used; the present scheme is chosen for simplicity.

4. Construction and forward evaluation of HMTB matrices

This section presents the construction and forward evaluation of HMTB matrices. Both follow a framework similar to that for $\mathcal{H}$-matrices, except that admissible leaf blocks are represented in Tucker format. Throughout this section, we assume that a tensor node block cluster tree $\mathsf{T}_{I\times I}$ associated with a uniform grid is given.

Although Algorithms 2 and 3 are written recursively following the block cluster tree, the computation need not be implemented recursively. After the block cluster tree is constructed, its leaf blocks can be stored in a linear list and processed sequentially or in parallel. In particular, the blockwise computations can be scheduled independently, with the resulting contributions accumulated into the corresponding output subvectors, making this linearized representation convenient for parallel implementations.

4.1. Construction of HMTB matrices

The construction of $\boldsymbol{A}^{\mathrm{HMTB}}$ follows Definition 9. For each node (τ,σ), we distinguish the following three cases:

- If (τ,σ) is an admissible leaf node, then Ω_τ and Ω_σ are disjoint, so the corresponding block contains no diagonal term. We first approximate the tensorization of the kernel matrix block by

$$\boldsymbol{\mathcal{K}}_{\tau,\sigma} \approx \mathrm{Tucker}\big(\boldsymbol{\mathcal{G}}, \{\boldsymbol{U}_\ell\}_{\ell=1}^d, \{\boldsymbol{V}_\ell\}_{\ell=1}^d\big)$$

  using the d-dimensional interpolation in (10). Other suitable interpolation schemes may also be used to construct the Tucker approximation. We then absorb h^d into the core tensor and compress the resulting Tucker representation using Algorithm 1, yielding the TLR representation of $\boldsymbol{A}_{\tau,\sigma}^{\mathrm{HMTB}}$ with the prescribed accuracy.
- If (τ,σ) is an inadmissible leaf node, we store the block densely as

$$\boldsymbol{A}_{\tau,\sigma}^{\mathrm{HMTB}} = \big(a(\boldsymbol{x}_i)\delta_{i,j} + K_{i,j}h^d\big)_{i\in\tau,\, j\in\sigma}, \tag{18}$$

  where $\delta_{i,j}$ denotes the Kronecker delta.
- If (τ,σ) is a non-leaf node, the construction is applied recursively to every $(\alpha,\beta) \in \mathrm{children}(\tau,\sigma)$.

Algorithm 2 summarizes the complete procedure, starting from the root (I,I) of $\mathsf{T}_{I\times I}$.

Remark 1. The interpolation factor matrices in (10) depend only on the normalized local grids and interpolation orders. Factors corresponding to identical normalized grids and orders, together with their QR factorizations, can therefore be precomputed and reused. For the regular uniform-grid hierarchy considered here, this applies to clusters

Algorithm 2 Construct$\big(a(\boldsymbol{x}), k(\boldsymbol{x},\boldsymbol{y}), (\tau,\sigma), p, \varepsilon\big)$.

Input: Functions $a(\boldsymbol{x})$ and $k(\boldsymbol{x},\boldsymbol{y})$, current node (τ,σ), interpolation order p, and compression tolerance ε
Output: The HMTB representation of the block $\boldsymbol{A}_{\tau,\sigma}^{\mathrm{HMTB}}$

```
if (τ, σ) is an admissible leaf then
    Compute 𝒢, {U_ℓ}_{ℓ=1}^d, and {V_ℓ}_{ℓ=1}^d by Chebyshev interpolation as in (10)
    Update 𝒢 ← h^d 𝒢
    Compress Tucker(𝒢, {U_ℓ}_{ℓ=1}^d, {V_ℓ}_{ℓ=1}^d) to tolerance ε using Algorithm 1
    Set A_{τ,σ}^{HMTB} to the resulting compressed TLR representation
else if (τ, σ) is an inadmissible leaf then
    Form A_{τ,σ}^{HMTB} by (18)
else
    for (α, β) ∈ children(τ, σ) do
        A_{α,β}^{HMTB} ← Construct(a(x), k(x, y), (α, β), p, ε)
    end for
end if
```

at the same level. The same reuse is available for the auxiliary uniform-grid matrix $\hat{\boldsymbol{A}}$ in Section 3.3.

More generally, factors can be grouped according to their normalized local grids and interpolation orders and reused within each group. This accommodates different interpolation orders without restricting the applicability of the construction. The subsequent compression of the transformed core remains specific to each admissible block, so the final compressed factor matrices need not coincide.

Algorithm 2 and the implementation used in the reported experiments compute the interpolation factors and their QR factorizations independently for each admissible block. The construction-cost analysis and timings therefore refer to this implementation without reuse. Reusing these quantities can eliminate repeated interpolation and QR computations, but the resulting practical speedup has not been quantified in the present experiments.

The leading complexity in Algorithm 2 arises from the Tucker compression of the admissible leaf blocks. Under the assumptions of Proposition 3, in particular, $p \le n_0 \le 2p$, the total compression cost is

$$\sum_{\ell=1}^{L} \mathscr{O}\big(2^{d\ell}(2dn_0 2^{L-\ell}p^2 + 2dp^{2d+1})\big) = \mathscr{O}\big(p^{3-d}N + p^{d+1}N\big) = \mathscr{O}\big(p^{d+1}N\big).$$

The dominant contribution comes from the compression of the core tensors.

As in the storage comparison in Section 3.2, the following comparison with $\mathcal{H}$-matrices concerns the low-rank representations generated by the same tensor-product interpolation before possible further algebraic recompression. For $\mathcal{H}$-matrices, the leading construction cost similarly arises from the compression of the low-rank representations. At level ℓ, the compression of each admissible block requires $\mathscr{O}(2^{d(L-\ell)}n_0^d p^{2d})$ operations for the QR factorizations of the basis matrices and $\mathscr{O}(p^{3d})$ operations for the subsequent compression. Consequently, the total compression cost for all admissible leaf blocks of the $\mathcal{H}$-matrix is

$$\sum_{\ell=1}^{L} \mathcal{O}\big(2^{d\ell}(2^d-1)(2^{d(L-\ell)}n_0^d p^{2d} + p^{3d})\big)$$
$$= \mathcal{O}\big(p^{2d}N\log_2 N + p^{2d}N\big) = \mathcal{O}\big(p^{2d}N\log_2 N\big).$$

In this case, the dominant contribution comes from the QR factorizations of the basis matrices. Therefore, the construction complexities of HMTB matrices and $\mathcal{H}$-matrices are $\mathcal{O}(p^{d+1}N)$ and $\mathcal{O}(p^{2d}N\log_2 N)$, respectively. This difference arises because an admissible HMTB block has $2d$ factor matrices of size $n \times p$, whereas the corresponding $\mathcal{H}$-matrix block has two basis matrices of size $n^d \times p^d$. We summarize this comparison in the following proposition.

Proposition 5 *(Complexity of HMTB construction). Under both weak and strong admissibility conditions, the complexity of constructing an HMTB matrix with interpolation order p and compression tolerance ε is $\mathcal{O}(p^{d+1}N)$.*

4.2. Forward evaluation of HMTB matrices

Using the hierarchical structure, the forward evaluation of an HMTB matrix, i.e., its multiplication by a vector, can also be performed efficiently. Starting from the root (I, I), let (τ, σ) be the current node and consider the update

$$\boldsymbol{f}_\tau \leftarrow \boldsymbol{f}_\tau + \boldsymbol{A}^{\mathrm{HMTB}}_{\tau,\sigma}\boldsymbol{u}_\sigma. \tag{19}$$

We distinguish the following three cases:

- If (τ, σ) is an admissible leaf, suppose that the TLR representation of $\boldsymbol{A}^{\mathrm{HMTB}}_{\tau,\sigma}$ is

$$\boldsymbol{A}^{\mathrm{HMTB}}_{\tau,\sigma} = \mathrm{Tucker}\big(\boldsymbol{\mathcal{G}}, \{\boldsymbol{U}_\ell\}_{\ell=1}^d, \{\boldsymbol{V}_\ell\}_{\ell=1}^d\big).$$

  The update is then computed as

$$\boldsymbol{f}_\tau \leftarrow \boldsymbol{f}_\tau + (\boldsymbol{U}_d \otimes \cdots \otimes \boldsymbol{U}_1)\boldsymbol{G}(\boldsymbol{V}_d \otimes \cdots \otimes \boldsymbol{V}_1)^\top \boldsymbol{u}_\sigma, \tag{20}$$

  where $\boldsymbol{G}$ is obtained by reshaping $\boldsymbol{\mathcal{G}}$ into a matrix.
  The Kronecker products in (20) need not be formed explicitly. Let $\boldsymbol{\mathcal{U}}$ be the tensorization of $\boldsymbol{u}_\sigma$. The contribution to $\boldsymbol{f}_\tau$ can be computed through mode products and a tensor contraction:

$$\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{U}} \times_1 \boldsymbol{V}_1^\top \times_2 \cdots \times_d \boldsymbol{V}_d^\top,$$
$$\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{G}} \times_{[d+1,\ldots,2d],[1,\ldots,d]} \boldsymbol{\mathcal{W}},$$
$$\boldsymbol{\mathcal{F}} = \boldsymbol{\mathcal{H}} \times_1 \boldsymbol{U}_1 \times_2 \cdots \times_d \boldsymbol{U}_d.$$

If all factor matrices are of size $n \times p$ and the matricized core $\boldsymbol{G}$ is of size $p^d \times p^d$, this computation requires $\mathcal{O}(dpn^d + p^{2d})$ operations. By comparison, evaluating the corresponding tensor-product representation as a conventional rank-p^d representation requires $\mathcal{O}(p^d n^d + p^{2d})$ operations.

- If (τ, σ) is an inadmissible leaf, the update (19) is computed directly because $\boldsymbol{A}^{\mathrm{HMTB}}_{\tau,\sigma}$ is stored as a small dense matrix.
- If (τ, σ) is a non-leaf node, the update is performed recursively for every child node:

$$\boldsymbol{f}_\alpha \leftarrow \boldsymbol{f}_\alpha + \boldsymbol{A}^{\mathrm{HMTB}}_{\alpha,\beta} \boldsymbol{u}_\beta, \qquad (\alpha, \beta) \in \mathrm{children}(\tau, \sigma).$$

Algorithm 3 summarizes the forward evaluation procedure, and Proposition 6 gives its computational complexity.

Algorithm 3 $\mathrm{HMultVec}\big(\boldsymbol{A}^{\mathrm{HMTB}}, (\tau, \sigma), \boldsymbol{u}, \boldsymbol{f}\big)$.

Input: HMTB matrix $\boldsymbol{A}^{\mathrm{HMTB}}$, current node (τ, σ), and vectors $\boldsymbol{u}$ and $\boldsymbol{f}$
Output: $\boldsymbol{f}_\tau \leftarrow \boldsymbol{f}_\tau + \boldsymbol{A}^{\mathrm{HMTB}}_{\tau,\sigma} \boldsymbol{u}_\sigma$

if (τ, σ) is an admissible leaf **then**
  Let $\boldsymbol{A}^{\mathrm{HMTB}}_{\tau,\sigma} = \mathrm{Tucker}(\boldsymbol{\mathcal{G}}, \{\boldsymbol{U}_\ell\}_{\ell=1}^d, \{\boldsymbol{V}_\ell\}_{\ell=1}^d)$
  Reshape $\boldsymbol{u}_\sigma$ into a d-dimensional tensor $\boldsymbol{\mathcal{U}}_\sigma$
  $\boldsymbol{\mathcal{W}}_\sigma = \boldsymbol{\mathcal{U}}_\sigma \times_1 \boldsymbol{V}_1^\top \times_2 \cdots \times_d \boldsymbol{V}_d^\top$
  $\boldsymbol{\mathcal{H}}_\tau = \boldsymbol{\mathcal{G}} \times_{[d+1,\ldots,2d],[1,\ldots,d]} \boldsymbol{\mathcal{W}}_\sigma$
  $\boldsymbol{\mathcal{F}}_\tau = \boldsymbol{\mathcal{H}}_\tau \times_1 \boldsymbol{U}_1 \times_2 \cdots \times_d \boldsymbol{U}_d$
  Reshape $\boldsymbol{\mathcal{F}}_\tau$ into a vector and add it to $\boldsymbol{f}_\tau$
else if (τ, σ) is an inadmissible leaf **then**
  Compute $\boldsymbol{f}_\tau \leftarrow \boldsymbol{f}_\tau + \boldsymbol{A}^{\mathrm{HMTB}}_{\tau,\sigma} \boldsymbol{u}_\sigma$
else
  for $(\tau', \sigma') \in \mathrm{children}(\tau, \sigma)$ **do**
    $\mathrm{HMultVec}(\boldsymbol{A}^{\mathrm{HMTB}}, (\tau', \sigma'), \boldsymbol{u}, \boldsymbol{f})$
  end for
end if

Proposition 6 *(Complexity of HMTB forward evaluation). Under weak or strong admissibility and the corresponding assumptions of Proposition 3 or Proposition 4, respectively, the forward evaluation of an HMTB matrix with Tucker bond dimensions bounded by p requires $\mathcal{O}(pN \log N + p^d N)$ operations.*

As in the preceding storage and construction comparisons, the following comparison with $\mathcal{H}$-matrices concerns the representations generated by the same tensor-product interpolation before possible further algebraic recompression. For a rank-p^d $\mathcal{H}$-matrix of size N, the complexity of forward evaluation is $\mathcal{O}(p^d N \log N + p^d N)$. For both HMTB matrices and $\mathcal{H}$-matrices, the computations involving the dense matrices stored at inadmissible leaves require $\mathcal{O}(p^d N)$ operations. Similarly, the computations involving the core matrices or tensors at admissible leaves also require $\mathcal{O}(p^d N)$ operations. The difference lies in the computations involving the factor or basis matrices: their complexity is $\mathcal{O}(pN \log N)$ for HMTB matrices, compared with $\mathcal{O}(p^d N \log N)$ for $\mathcal{H}$-matrices.

Therefore, for these interpolation-generated representations, HMTB matrices have a significantly smaller prefactor in the leading $N \log N$ term.

5. Numerical results

We compare HMTB matrices with $\mathcal{H}$-matrices through a series of numerical experiments. Three types of kernels are considered:

- Gaussian kernel:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2\sigma^2}\right).$$

  The Gaussian kernel is smooth and nonsingular. In our experiments, the bandwidth is set to $\sigma = \sqrt{d}$, where d denotes the spatial dimension.
- Single-layer potential (SLP) kernel:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} -\dfrac{1}{2\pi} \log\big(\|\boldsymbol{x} - \boldsymbol{y}\|\big), & d = 2, \\ \dfrac{1}{4\pi\|\boldsymbol{x} - \boldsymbol{y}\|}, & d = 3. \end{cases}$$

  The SLP kernel is singular on the diagonal and smooth away from it.
- Helmholtz kernel:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \dfrac{\mathrm{i}}{4} H_0^{(1)}\big(\kappa\|\boldsymbol{x} - \boldsymbol{y}\|\big), & d = 2, \\ \dfrac{\exp\big(\mathrm{i}\kappa\|\boldsymbol{x} - \boldsymbol{y}\|\big)}{4\pi\|\boldsymbol{x} - \boldsymbol{y}\|}, & d = 3, \end{cases}$$

  where $H_0^{(1)}$ denotes the Hankel function of the first kind of order zero and κ is the wavenumber. The Helmholtz kernel is singular on the diagonal and oscillatory away from it. In our experiments, we set $\kappa = 10$.

All algorithms are implemented in MATLAB R2023b. The experiments were performed on a server equipped with two Intel Xeon Gold 6226R processors running at 2.90 GHz and 1000.6 GB of RAM. The code is available at https://github.com/JingyuLiuMath/HMTB.

For each experiment, t_{c} and t_{a} denote the construction and forward-evaluation times, respectively, measured in seconds, and m_{h} denotes the memory usage in GB. The relative error of the forward evaluation is estimated on a randomly sampled index set $I_r \subset [N]$ by

$$e_{\mathrm{a;r}} = \frac{\|\widetilde{\boldsymbol{f}}(I_r, :) - \boldsymbol{f}(I_r, :)\|_{\mathrm{F}}}{\|\boldsymbol{f}(I_r, :)\|_{\mathrm{F}}},$$

where $\boldsymbol{f} = \boldsymbol{A}\boldsymbol{u}$ is the exact result and $\widetilde{\boldsymbol{f}}$ is the corresponding approximate forward evaluation.

In Sections 5.1 and 5.2, $\boldsymbol{u} \in \mathbb{C}^{N\times 10}$ contains ten randomly generated test vectors. In Section 5.3, $\boldsymbol{u} \in \mathbb{C}^{N}$ is obtained by sampling a prescribed smooth function, so that the Frobenius norm in the above definition reduces to the Euclidean norm. For all experiments, I_r is sampled uniformly at random from $[N]$ with $|I_r| = 256$. Since the number of test vectors is fixed, evaluating the exact product only on I_r requires $\mathcal{O}(|I_r|N)$ operations and avoids explicitly forming the dense matrix $\boldsymbol{A}$, which would require $\mathcal{O}(N^2)$ storage.

Unless otherwise specified, we set the leaf-size threshold and interpolation order to $N_0 = 16^2$ and $p = 8$, respectively, for the 2D problems, and to $N_0 = 4^3$ and $p = 4$, respectively, for the 3D problems. The compression tolerance is set to $\varepsilon = 10^{-3}$ throughout. Based on the numerical investigation in Appendix A, we use weak admissibility for the Gaussian kernel and strong admissibility for the SLP and Helmholtz kernels.

For a consistent comparison, whenever a ratio is reported in the following tables, it is defined as the corresponding $\mathcal{H}$-matrix quantity divided by the HMTB quantity:

$$r_{\mathrm{c}} = \frac{t_{\mathrm{c}}^{\mathrm{H}}}{t_{\mathrm{c}}^{\mathrm{HMTB}}}, \qquad r_{\mathrm{m}} = \frac{m_{\mathrm{h}}^{\mathrm{H}}}{m_{\mathrm{h}}^{\mathrm{HMTB}}}, \qquad r_{\mathrm{a}} = \frac{t_{\mathrm{a}}^{\mathrm{H}}}{t_{\mathrm{a}}^{\mathrm{HMTB}}}.$$

Thus, a ratio greater than one indicates an advantage of HMTB matrices for the corresponding quantity, whereas a ratio below one indicates that the $\mathcal{H}$-matrix is less expensive.

Our numerical experiments are designed to compare the two representations at comparable levels of accuracy. For both HMTB matrices and $\mathcal{H}$-matrices, admissible blocks are initially approximated using the same interpolation order p, and the same compression tolerance is used in the subsequent compression step. We focus on three aspects: construction time, memory usage, and forward-evaluation time. The dependence of the approximation accuracy on the interpolation order is further investigated in Appendix A.

Sections 5.1 and 5.2 present results on uniform grids in two and three dimensions, respectively. Section 5.3 then illustrates the application of HMTB matrices to quasi-uniform grids.

Remark 2. The construction-time comparison is specific to the interpolation-based implementations used here. The initial rank bound p^d for an admissible $\mathcal{H}$-matrix block can substantially exceed its numerical rank at the prescribed accuracy. In our implementation, recompression requires QR factorizations of the initial basis matrices and an SVD of the resulting $p^d \times p^d$ core matrix. The latter costs $\mathcal{O}(p^{3d})$, in addition to the basis orthogonalization and update costs. Although truncation reduces the final storage requirement, it does not eliminate the work already incurred using the initial matrix rank bound p^d.

Alternative construction strategies include adaptive cross approximation (ACA) [5], randomized hierarchical compression using applications of the operator and its adjoint [43], and proxy-surface methods for suitable kernels [41]. These methods can

construct approximations whose ranks reflect the numerical rank more closely, without first forming the full tensor-product interpolation representation.

Consequently, the observed construction-time differences reflect both the representations and the chosen construction procedures. They should not be interpreted as an intrinsic advantage of HMTB matrices over all $\mathcal{H}$-matrix construction methods. The memory savings reported here are measured against the recompressed $\mathcal{H}$-matrix implementation used in these experiments; comparisons with the alternative construction strategies above are outside the scope of the present numerical study.

5.1. Uniform grid in two dimensions

This section presents numerical experiments on two-dimensional uniform grids. Let $\Omega = [0,1]^2$ be the unit square and $a(\boldsymbol{x}) \equiv 0$ in (1).

We first consider the Gaussian kernel under weak admissibility. The problem is discretized using n points in each coordinate direction, with n ranging from 256 to 16384. Fig. 5 shows the construction time, memory usage, and forward-evaluation time of HMTB matrices and $\mathcal{H}$-matrices. Both the construction time t_{c} and the memory usage m_{h} exhibit the predicted $\mathcal{O}(N)$ scaling, while the forward-evaluation time is consistent with the $\mathcal{O}(N \log N)$ complexity for sufficiently large problem sizes. Detailed results are reported in Table 1.

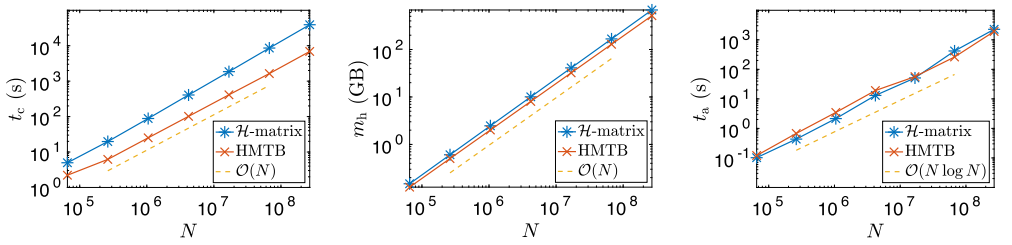
Both representations achieve comparable accuracy, with $e_{\mathrm{a;r}}$ on the order of 10^{-4}. Compared with $\mathcal{H}$-matrices, HMTB matrices reduce the construction time by a factor of approximately 2.2 to 5.7 and the memory usage by a factor of approximately 1.2 to 1.3. For smaller problem sizes, the forward evaluation of HMTB matrices is somewhat more expensive than that of $\mathcal{H}$-matrices. As the problem size increases, however, the difference decreases, and HMTB becomes faster for the two largest problem sizes considered.

Next, we consider the Helmholtz kernel under strong admissibility. In this experiment, n ranges from 256 to 4096. Fig. 6 compares the construction time, memory usage, and forward-evaluation time of $\mathcal{H}$-matrices and HMTB matrices, while detailed results are reported in Table 2. Both representations achieve comparable accuracy, with $e_{\mathrm{a;r}}$ on the order of 10^{-4}. The observed scaling of the construction time, memory usage, and forward-evaluation time is consistent with the theoretical complexity estimates.

Compared with $\mathcal{H}$-matrices, HMTB matrices reduce the construction time by a factor of approximately 1.8 to 3.2 and the memory usage by a factor of approximately 1.1 to 1.4. For smaller problem sizes, the forward evaluation of HMTB matrices is more expensive than that of $\mathcal{H}$-matrices. As N increases, this difference diminishes, and HMTB becomes faster for the two largest problem sizes considered.

5.2. Uniform grid in three dimensions

We next consider the three-dimensional analogue of the experiments in Section 5.1. The computational domain is $\Omega = [0,1]^3$, with $a(\boldsymbol{x}) \equiv 0$ in (1).

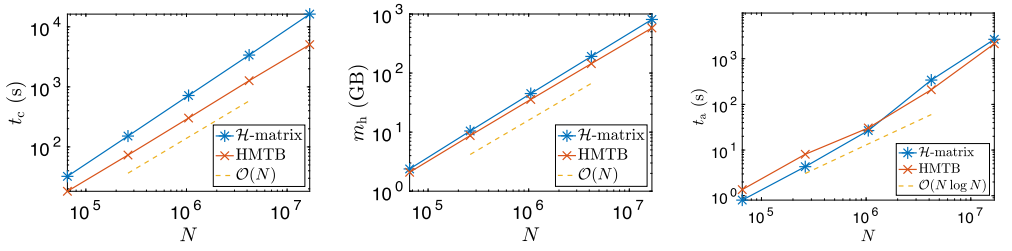


(a) Construction time. (b) Memory. (c) Forward time.

Fig. 5. Time and memory costs of $\mathcal{H}$-matrices and HMTB matrices for the Gaussian kernel under weak admissibility in 2D.

Table 1
Numerical results of $\mathcal{H}$-matrices and HMTB for the Gaussian kernel under weak admissibility in 2D.

N		t_c (s)	m_h (GB)	t_a (s)	$e_{a;r}$
	$\mathcal{H}$-matrix	5.0e+00	1.5e-01	1.0e-01	3.8e-04
256^2	HMTB	2.2e+00	1.3e-01	1.2e-01	3.5e-04
	Ratio	2.2×	1.2×	0.8×	-
	$\mathcal{H}$-matrix	2.0e+01	6.1e-01	4.3e-01	3.9e-04
512^2	HMTB	6.2e+00	5.0e-01	6.8e-01	3.6e-04
	Ratio	3.2×	1.2×	0.6×	-
	$\mathcal{H}$-matrix	8.8e+01	2.5e+00	2.1e+00	3.6e-04
1024^2	HMTB	2.5e+01	2.0e+00	3.5e+00	3.2e-04
	Ratio	3.5×	1.2×	0.6×	-
	$\mathcal{H}$-matrix	4.1e+02	1.0e+01	1.3e+01	2.6e-04
2048^2	HMTB	1.0e+02	8.0e+00	1.9e+01	2.3e-04
	Ratio	4.0×	1.3×	0.7×	-
	$\mathcal{H}$-matrix	1.9e+03	4.1e+01	5.1e+01	2.7e-04
4096^2	HMTB	4.1e+02	3.2e+01	5.7e+01	2.6e-04
	Ratio	4.5×	1.3×	0.9×	-
	$\mathcal{H}$-matrix	8.6e+03	1.7e+02	4.2e+02	2.7e-04
8192^2	HMTB	1.6e+03	1.3e+02	2.6e+02	2.4e-04
	Ratio	5.2×	1.3×	1.6×	-
	$\mathcal{H}$-matrix	3.9e+04	6.8e+02	2.3e+03	2.5e-04
16384^2	HMTB	6.9e+03	5.2e+02	1.9e+03	2.3e-04
	Ratio	5.7×	1.3×	1.2×	-

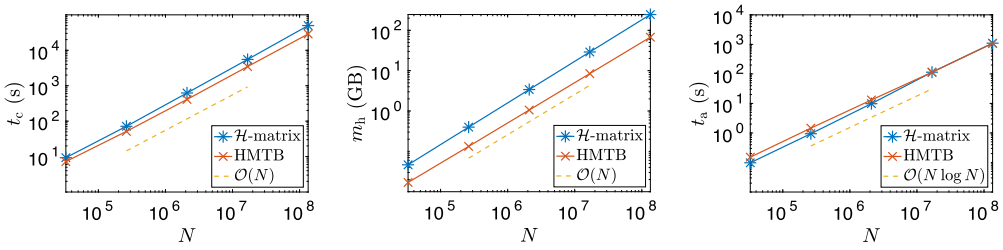


(a) Construction time. (b) Memory. (c) Forward time.

Fig. 6. Time and memory costs of $\mathcal{H}$-matrices and HMTB matrices for the Helmholtz kernel under strong admissibility in 2D.

Table 2
Numerical results of $\mathcal{H}$-matrices and HMTB for the Helmholtz kernel under strong admissibility in 2D.

N		t_c (s)	m_h (GB)	t_a (s)	$e_{a;r}$
256^2	$\mathcal{H}$-matrix	3.2e+01	2.4e+00	8.0e-01	4.8e-04
	HMTB	1.8e+01	2.1e+00	1.4e+00	6.7e-04
	Ratio	1.8×	1.1×	0.6×	-
512^2	$\mathcal{H}$-matrix	1.5e+02	1.0e+01	4.3e+00	5.0e-04
	HMTB	7.3e+01	8.7e+00	8.1e+00	7.1e-04
	Ratio	2.1×	1.2×	0.5×	-
1024^2	$\mathcal{H}$-matrix	7.2e+02	4.5e+01	2.7e+01	5.1e-04
	HMTB	3.0e+02	3.6e+01	3.0e+01	7.4e-04
	Ratio	2.4×	1.3×	0.9×	-
2048^2	$\mathcal{H}$-matrix	3.4e+03	1.9e+02	3.4e+02	4.9e-04
	HMTB	1.3e+03	1.4e+02	2.1e+02	7.3e-04
	Ratio	2.7×	1.3×	1.6×	-
4096^2	$\mathcal{H}$-matrix	1.6e+04	8.1e+02	2.7e+03	5.8e-04
	HMTB	5.1e+03	5.8e+02	2.1e+03	8.6e-04
	Ratio	3.2×	1.4×	1.3×	-

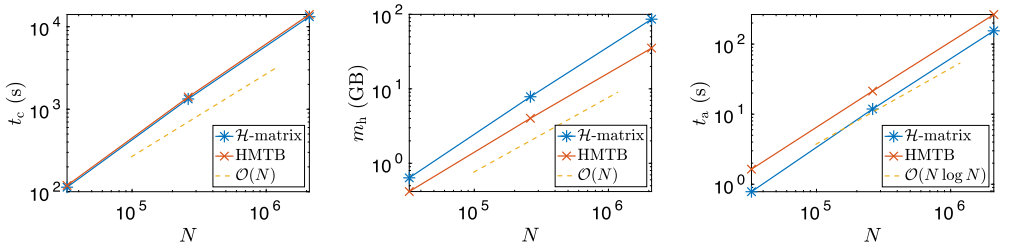


(a) Construction time. (b) Memory. (c) Forward time.

Fig. 7. Time and memory costs of $\mathcal{H}$-matrices and HMTB for the Gaussian kernel under weak admissibility in 3D.

Table 3
Numerical results of $\mathcal{H}$-matrices and HMTB for the Gaussian kernel under weak admissibility in 3D.

N		t_c (s)	m_h (GB)	t_a (s)	$e_{a;r}$
32^3	$\mathcal{H}$-matrix	9.3e+00	4.6e-02	9.8e-02	1.6e-04
	HMTB	7.2e+00	1.7e-02	1.5e-01	1.3e-04
	Ratio	1.3×	2.7×	0.6×	-
64^3	$\mathcal{H}$-matrix	7.2e+01	4.0e-01	9.4e-01	1.7e-04
	HMTB	5.1e+01	1.3e-01	1.5e+00	1.4e-04
	Ratio	1.4×	3.0×	0.6×	-
128^3	$\mathcal{H}$-matrix	6.2e+02	3.4e+00	9.8e+00	1.4e-04
	HMTB	4.1e+02	1.1e+00	1.3e+01	1.1e-04
	Ratio	1.5×	3.2×	0.7×	-
256^3	$\mathcal{H}$-matrix	5.5e+03	2.9e+01	1.1e+02	2.1e-04
	HMTB	3.5e+03	8.5e+00	1.1e+02	1.6e-04
	Ratio	1.6×	3.4×	1.0×	-
512^3	$\mathcal{H}$-matrix	5.0e+04	2.5e+02	1.1e+03	1.4e-04
	HMTB	2.9e+04	6.8e+01	1.1e+03	1.1e-04
	Ratio	1.7×	3.6×	1.0×	-



(a) Construction time. (b) Memory. (c) Forward time.

Fig. 8. Time and memory costs of $\mathcal{H}$-matrices and HMTB matrices for the SLP kernel under strong admissibility in 3D.

Table 4
Numerical results of $\mathcal{H}$-matrices and HMTB for the single-layer potential kernel under strong admissibility in 3D.

N		t_c (s)	m_h (GB)	t_a (s)	$e_{a;r}$
32^3	$\mathcal{H}$-matrix	1.1e+02	6.4e-01	7.8e-01	8.6e-04
	HMTB	1.2e+02	4.2e-01	1.6e+00	9.5e-04
	Ratio	1.0×	1.5×	0.5×	-
64^3	$\mathcal{H}$-matrix	1.3e+03	7.8e+00	1.2e+01	1.1e-03
	HMTB	1.4e+03	4.0e+00	2.1e+01	1.1e-03
	Ratio	0.9×	1.9×	0.6×	-
128^3	$\mathcal{H}$-matrix	1.3e+04	8.6e+01	1.5e+02	9.8e-04
	HMTB	1.4e+04	3.5e+01	2.6e+02	1.0e-03
	Ratio	0.9×	2.4×	0.6×	-

For the Gaussian kernel, n ranges from 32 to 512. Fig. 7 shows the construction time, memory usage, and forward-evaluation time, with detailed results reported in Table 3. The observed construction time and memory usage of both $\mathcal{H}$-matrices and HMTB matrices scale linearly with N, while the forward-evaluation time is consistent with the $\mathcal{O}(N \log N)$ complexity.

Compared with $\mathcal{H}$-matrices, HMTB matrices reduce the construction time by a factor of approximately 1.3 to 1.7 and the memory usage by a factor of approximately 2.7 to 3.6. In particular, for $N = 512^3$, the HMTB representation requires approximately 68 GB of memory, compared with approximately 250 GB for the corresponding $\mathcal{H}$-matrix. For smaller problem sizes, the forward evaluation of HMTB matrices is somewhat more expensive, but the difference decreases as N increases, and the two methods have comparable forward-evaluation times for the largest problem sizes considered. Both representations achieve comparable accuracy, with $e_{a;r}$ on the order of 10^{-4}.

For the SLP kernel, we discretize the problem using $n = 32$, 64, and 128 points in each coordinate direction. The corresponding results are summarized in Fig. 8 and Table 4. The observed scaling of the construction time, memory usage, and forward-evaluation time is consistent with the theoretical complexity estimates. Both $\mathcal{H}$-matrices and HMTB matrices achieve comparable accuracy, with $e_{a;r}$ on the order of 10^{-3}.

In this case, the construction costs of the two representations are comparable, with HMTB matrices being slightly more expensive for the larger problem sizes. The forward evaluation of HMTB matrices is also more expensive than that of $\mathcal{H}$-matrices, by a factor of approximately 1.7 to 2.1. This behavior is associated with the strong admissibility condition, under which a larger fraction of the matrix is represented by dense inadmissible blocks, reducing the computational advantage obtained from the Tucker representation of admissible blocks. Nevertheless, HMTB matrices reduce the memory usage by a factor of approximately 1.5 to 2.4, with the memory advantage becoming more pronounced as the problem size increases.

5.3. Quasi-uniform grid in two dimensions

In this section, we present two examples to illustrate the application of HMTB matrices on quasi-uniform grids, considering the Gaussian and SLP kernels on the two-dimensional domain $\Omega = [0,1]^2$. The quasi-uniform grid is obtained from a triangulation of Ω, with each discretization point $\boldsymbol{x}_i$ chosen as the centroid of the corresponding triangular cell; see Fig. 4. We consider matrix sizes

$$N = 8192,\ 32768,\ 131072,\ \text{and } 524288.$$

Let M denote the number of points on the auxiliary uniform grid, and let $\boldsymbol{A}$ and $\hat{\boldsymbol{A}}$ denote the matrices associated with the quasi-uniform and uniform grids, respectively, as described in Section 3.3. The matrix $\hat{\boldsymbol{A}}$ is represented by its HMTB approximation. We define the oversampling ratio by

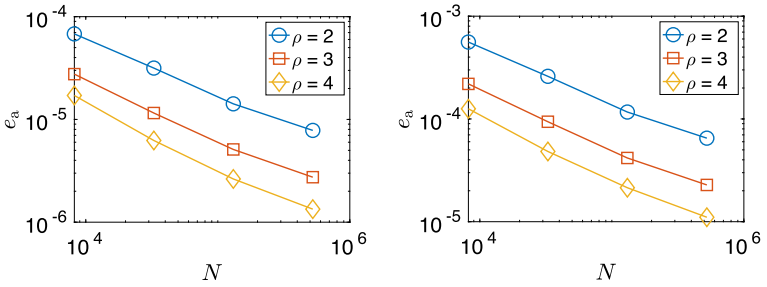
$$\rho = \sqrt{\frac{2M}{N}},$$

where the factor 2 accounts for the fact that each square cell of the uniform grid corresponds to two triangular cells in the quasi-uniform triangulation. The input vector $\boldsymbol{u}$ is obtained by sampling the smooth function

$$u(\boldsymbol{x}) = u(x_1, x_2) = 1 + 0.5\mathrm{e}^{-(x_1-0.3)^2-(x_2-0.6)^2} + \sin(5x_1x_2)$$

at the quasi-uniform grid points.

Fig. 9 shows the relative error of $\boldsymbol{T}\hat{\boldsymbol{A}}^{\mathrm{HMTB}}\boldsymbol{S}\boldsymbol{u}$ as an approximation to $\boldsymbol{A}\boldsymbol{u}$ for different problem sizes and oversampling ratios. Weak admissibility is used for the Gaussian kernel, while strong admissibility is used for the SLP kernel. For each fixed N, increasing ρ generally reduces the error, as the auxiliary uniform grid becomes finer. For a fixed ρ, the error also decreases as N increases, since both the quasi-uniform grid and the auxiliary uniform grid are refined simultaneously. In particular, when $N = 524288$ and $\rho = 2$, the relative errors for the Gaussian and SLP kernels are approximately 10^{-5} and 10^{-4}, respectively. These results indicate that a modest oversampling ratio, such as



(a) Gaussian kernel. (b) SLP kernel.

Fig. 9. Relative forward-evaluation errors on the quasi-uniform grid for the Gaussian and SLP kernels in 2D. Left: Gaussian kernel. Right: SLP kernel.

$\rho = 2$, already provides a reasonable balance between accuracy and computational cost in the present experiments.

6. Conclusion and future work

In this paper, we introduce HMTB matrices and develop corresponding algorithms for their construction and forward evaluation. When the underlying discretization has a tensor-product structure, HMTB matrices can exploit this structure to reduce the computational and storage costs of conventional $\mathcal{H}$-matrices. For fixed d and p, we show that HMTB matrices require $\mathcal{O}(N)$ storage and can be constructed in $\mathcal{O}(N)$ complexity, while their forward evaluation has complexity $\mathcal{O}(N \log N)$. Compared with $\mathcal{H}$-matrices, HMTB matrices reduce the dimension-dependent costs associated with the basis factors by exploiting their tensor structure, replacing their exponential dependence on the dimension by a linear dependence. The core-tensor contribution, however, retains an exponential dependence on the dimension. We further extend the framework to quasi-uniform grids through sparse grid-transfer operators, thereby broadening its applicability. Numerical experiments in two and three dimensions support the predicted complexity scaling. For the interpolation-based constructions tested, HMTB matrices offer memory savings and, in several cases, shorter construction times, while their forward-evaluation costs depend on the kernel and admissibility condition.

The implications of these results are twofold. First, from a methodological perspective, HMTB delivers a principled way to exploit separability and tensor structure in large-scale operator compression, contributing to the theory of hierarchical tensor approximations with provable complexity guarantees and favorable dimension dependence. Second, from a practical standpoint, the reduced memory footprint and competitive forward-evaluation cost make HMTB attractive for applications where repeated operator evaluations or limited memory are critical.

There are several future directions to consider. One potential avenue is to develop additional algebraic operations for HMTB matrices, including matrix addition, multiplication, and LU decomposition. A second direction is to develop adaptive and algebraic

construction strategies for settings without explicit tensor grids, for example using randomized sampling. Such approaches could enable HMTB compression of black-box operators or data-driven interaction matrices. Furthermore, exploring the parallelization of these algorithms could enhance their efficiency and scalability, enabling them to accommodate larger problem sizes effectively.

Declaration of competing interest

The authors declare that they have no competing interests.

Acknowledgments

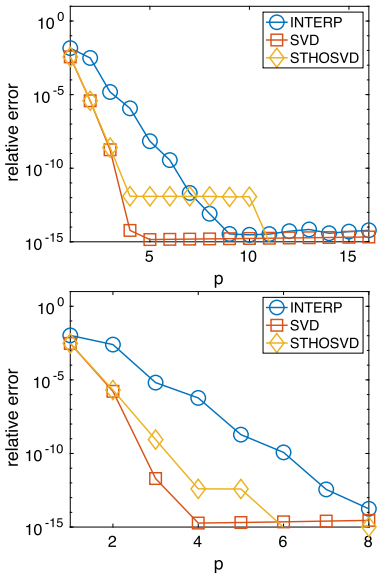
This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 12271109 and 12526211; by the Shanghai Pilot Program for Basic Research-Fudan University under Grant No. 21TQ1400100 (22TQ017); by the Scientific Research Innovation Capability Support Project for Young Faculty under Grant No. SRICSPYF-ZY2025159; and by the Xuemin Institute of Advanced Studies, Fudan University. We thank the two anonymous reviewers for their valuable comments and suggestions.

Appendix A. Exploring tensor low-rank representations of kernel functions

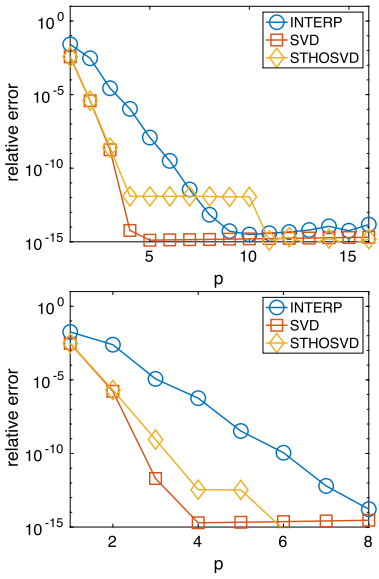
We numerically explore the Tucker low-rankness of interaction matrices, with a comparison to conventional low-rank matrices. Two configurations of domains for the interaction matrices are examined: neighbor domains, which consist of two adjacent subdomains, and well-separated domains, which are characterized by two strongly admissible subdomains. To construct a low-rank decomposition, we use the following approaches: For both types of low-rank structures, the interpolation (INTERP) method as discussed in Section 3.1 is used to create a low-rank decomposition. In addition, we apply SVD to compute a conventional low-rank decomposition and STHOSVD [49] to compute a Tucker decomposition. The accuracy of the compression is measured using the relative error with respect to the Frobenius norm.

In the 2D case, we consider the following domains: $\Omega_1 = [0, h] \times [0, h]$, $\Omega_2 = [h, 2h] \times [0, h]$ and $\Omega_3 = [2h, 3h] \times [0, h]$ where $h = 0.25$. The domains Ω_1 and Ω_2 are neighbors, while Ω_1 and Ω_3 are well-separated. Each domain is discretized with 32 points in each direction. The uniform Tucker bond dimension p ranges from 1 to 16, and the SVD rank is set to p^2. Similarly, the domains in 3D are given by $\Omega_1 = [0, h] \times [0, h] \times [0, h]$, $\Omega_2 = [h, 2h] \times [0, h] \times [0, h]$ and $\Omega_3 = [2h, 3h] \times [0, h] \times [0, h]$ for $h = 0.25$. Each domain is discretized with 16 points in each direction, and p ranges from 1 to 8, with the SVD rank set to p^3.

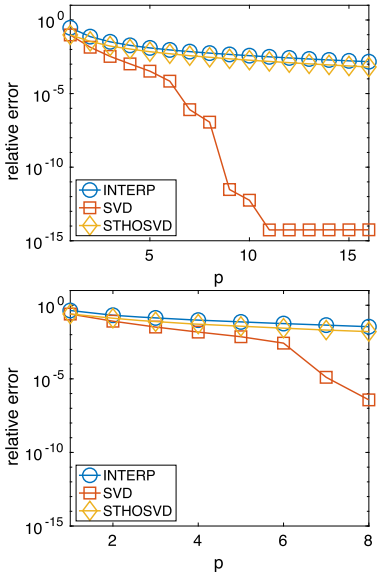
The results of different low-rank decompositions are plotted in Fig. A.10. For well-separated domains (Fig. A.10b, A.10d and A.10f), all three methods exhibit a rapid



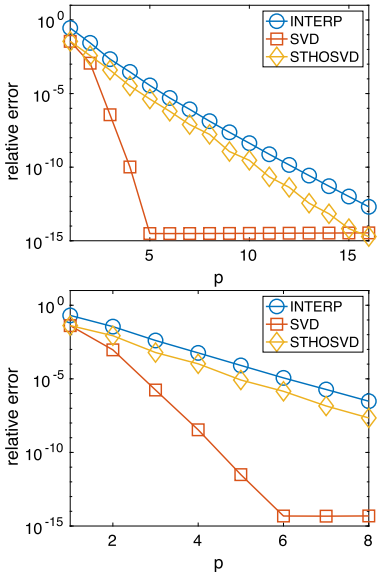
(a) Gaussian kernel and neighbor domains.



(b) Gaussian kernel and well-separated domains.



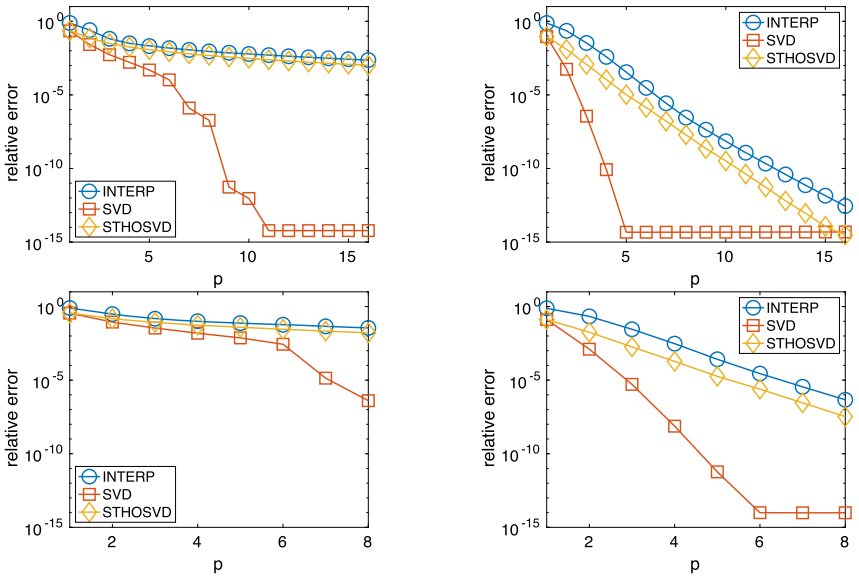
(c) SLP kernel and neighbor domains.



(d) SLP kernel and well-separated domains.

Fig. A.10. Relative approximation errors of the Gaussian, SLP, and Helmholtz kernels. The size of the interaction matrix is $32^2 \times 32^2$ for 2D problems (top) and $16^3 \times 16^3$ for 3D problems (bottom).

decay in error as p increases, regardless of the kernel. In this scenario, the interpolation method is sufficient to achieve an accurate low-rank decomposition. However, when two domains are neighbors (as depicted in Fig. A.10a, A.10c and A.10e), the decay rates of the relative error exhibit dependence on the kernels. For the Gaussian kernel, both



(e) Helmholtz kernel and neighbor domains.

(f) Helmholtz kernel and well-separated domains.

Fig. A.10. Relative approximation errors of the Gaussian, SLP, and Helmholtz kernels (continued).

low-rank structures still work well, with the interpolation-based construction yielding satisfactory results. In contrast, for the SLP and Helmholtz kernels, the Tucker low-rank format proves unsuitable. Even with the application of STHOSVD, the relative error decreases very slowly as the uniform Tucker bond dimension p increases. Accordingly, in Section 5 we use strong admissibility for the SLP and Helmholtz kernels and weak admissibility for the Gaussian kernel.

Data availability

Data will be made available on request.

References

[1] A. Aminfar, S. Ambikasaran, E. Darve, A fast block low-rank dense solver with applications to finite-element matrices, J. Comput. Phys. 304 (C) (2016) 170–188.
[2] F. Aurenhammer, Voronoi diagrams—a survey of a fundamental geometric data structure, ACM Comput. Surv. 23 (3) (1991) 345–405.
[3] J. Barnes, P. Hut, A hierarchical $O(N \log N)$ force-calculation algorithm, Nature 324 (6096) (1986) 446–449.
[4] A.H. Barnett, J. Magland, L. af Klinteberg, A parallel nonuniform fast Fourier transform library based on an "exponential of semicircle" kernel, SIAM J. Sci. Comput. 41 (5) (2019) C479–C504.
[5] M. Bebendorf, Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems, 1st edition, Lect. Notes Comput. Sci. Eng., vol. 63, Springer, Berlin, 2008.
[6] S. Börm, L. Grasedyck, W. Hackbusch, Introduction to hierarchical matrices with applications, Eng. Anal. Bound. Elem. 27 (5) (2003) 405–422.

[7] E. Candès, L. Demanet, L. Ying, A fast butterfly algorithm for the computation of Fourier integral operators, Multiscale Model. Simul. 7 (4) (2009) 1727–1750.
[8] J. Carrier, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm for particle simulations, SIAM J. Sci. Stat. Comput. 9 (4) (1988) 669–686.
[9] S. Chandrasekaran, M. Gu, T. Pals, A fast ULV decomposition solver for hierarchically semiseparable representations, SIAM J. Matrix Anal. Appl. 28 (3) (2006) 603–622.
[10] H. Cheng, Z. Gimbutas, P.-G. Martinsson, V. Rokhlin, On the compression of low rank matrices, SIAM J. Sci. Comput. 26 (4) (2005) 1389–1404.
[11] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D.P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 1 Low-rank tensor decompositions, Found. Trends Mach. Learn. 9 (4–5) (2016) 249–429.
[12] E. Corona, P.-G. Martinsson, D. Zorin, An $O(N)$ direct solver for integral equations on the plane, Appl. Comput. Harmon. Anal. 38 (2) (2015) 284–317.
[13] E. Corona, A. Rahimian, D. Zorin, A tensor-train accelerated solver for integral equations in complex geometries, J. Comput. Phys. 334 (2017) 145–169.
[14] L. De Lathauwer, B. De Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl. 21 (4) (2000) 1253–1278.
[15] L. De Lathauwer, B. De Moor, J. Vandewalle, On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensors, SIAM J. Matrix Anal. Appl. 21 (4) (2000) 1324–1342.
[16] V. Ehrlacher, L. Grigori, D. Lombardi, H. Song, Adaptive hierarchical subtensor partitioning for tensor compression, SIAM J. Sci. Comput. 43 (1) (2021) A139–A163.
[17] W. Fong, E. Darve, The black-box fast multipole method, J. Comput. Phys. 228 (23) (2009) 8712–8725.
[18] I.P. Gavrilyuk, W. Hackbusch, B.N. Khoromskij, Hierarchical tensor-product approximation to the inverse and related operators for high-dimensional elliptic problems, Computing 74 (2) (2005) 131–157.
[19] A. Gillman, P.M. Young, P.-G. Martinsson, A direct solver with $O(N)$ complexity for integral equations on one-dimensional domains, Front. Math. China 7 (2) (2012) 217–247.
[20] L. Grasedyck, W. Hackbusch, Construction and arithmetics of $\mathcal{H}$-matrices, Computing 70 (4) (2003) 295–334.
[21] L. Greengard, J.-Y. Lee, Accelerating the nonuniform fast Fourier transform, SIAM Rev. 46 (3) (2004) 443–454.
[22] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 135 (2) (1997) 280–292.
[23] L. Greengard, D. Gueyffier, P.-G. Martinsson, V. Rokhlin, Fast direct solvers for integral equations in complex three-dimensional domains, Acta Numer. 18 (2009) 243–275.
[24] W. Hackbusch, A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices, Computing 62 (2) (1999) 89–108.
[25] W. Hackbusch, Hierarchical Matrices: Algorithms and Analysis, 1st edition, Springer Ser. Comput. Math., vol. 49, Springer, Berlin, 2015.
[26] W. Hackbusch, S. Börm, Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices, Computing 69 (1) (2002) 1–35.
[27] W. Hackbusch, B.N. Khoromskij, A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems, Computing 64 (1) (2000) 21–47.
[28] W. Hackbusch, B.N. Khoromskij, Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. Part I. Separable approximation of multi-variate functions, Computing 76 (3) (2006) 177–202.
[29] W. Hackbusch, B.N. Khoromskij, Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. Part II. HKT representation of certain operators, Computing 76 (3) (2006) 203–225.
[30] W. Hackbusch, B.N. Khoromskij, Tensor-product approximation to multidimensional integral operators and Green's functions, SIAM J. Matrix Anal. Appl. 30 (3) (2008) 1233–1253.
[31] W. Hackbusch, B.N. Khoromskij, E.E. Tyrtyshnikov, Hierarchical Kronecker tensor-product approximations, J. Numer. Math. 13 (2) (2005) 119–156.
[32] B. Hashemi, Y. Nakatsukasa, RTSMS: Randomized Tucker with single-mode sketching, Electron. Trans. Numer. Anal. 63 (2025) 247–280.
[33] K.L. Ho, L. Greengard, A fast direct solver for structured linear systems by recursive skeletonization, SIAM J. Sci. Comput. 34 (5) (2012) A2507–A2532.

[34] K.L. Ho, L. Ying, Hierarchical interpolative factorization for elliptic operators: integral equations, Commun. Pure Appl. Math. 69 (7) (2016) 1314–1353.
[35] P.M. Kielstra, T. Shi, H. Luo, J. Qian, Y. Liu, A linear-complexity tensor butterfly algorithm for compressing high-dimensional oscillatory integral operators, Multiscale Model. Simul. 23 (2) (2025) 864–893.
[36] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500.
[37] Y. Li, H. Yang, Interpolative butterfly factorization, SIAM J. Sci. Comput. 39 (2) (2017) A503–A531.
[38] Y. Li, H. Yang, E.R. Martin, K.L. Ho, L. Ying, Butterfly factorization, Multiscale Model. Simul. 13 (2) (2015) 714–732.
[39] Y. Li, H. Yang, L. Ying, Multidimensional butterfly factorization, Appl. Comput. Harmon. Anal. 44 (3) (2018) 737–758.
[40] Y. Li, J. Poulson, L. Ying, Distributed-memory $\mathcal{H}$-matrix algebra I: Data distribution and matrix-vector multiplication, CSIAM Trans. Appl. Math. 2 (3) (2021) 431–459.
[41] P.-G. Martinsson, Fast Direct Solvers for Elliptic PDEs, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2019.
[42] P.-G. Martinsson, V. Rokhlin, A fast direct solver for boundary integral equations in two dimensions, J. Comput. Phys. 205 (1) (2005) 1–23.
[43] P.-G. Martinsson, J.A. Tropp, Randomized numerical linear algebra: foundations and algorithms, Acta Numer. 29 (2020) 403–572.
[44] V. Minden, K.L. Ho, A. Damle, L. Ying, A recursive skeletonization factorization based on strong admissibility, Multiscale Model. Simul. 15 (2) (2017) 768–796.
[45] R. Minster, A.K. Saibaba, M.E. Kilmer, Randomized algorithms for low-rank tensor decompositions in the Tucker format, SIAM J. Math. Data Sci. 2 (1) (2020) 189–215.
[46] I.V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. 33 (5) (2011) 2295–2317.
[47] C. Qian, A.C. Yucel, On the compression of translation operator tensors in FMM-FFT-accelerated SIE simulators via tensor decompositions, IEEE Trans. Antennas Propag. 69 (6) (2021) 3359–3370.
[48] L.N. Trefethen, Approximation Theory and Approximation Practice, SIAM, Philadelphia, PA, 2019.
[49] N. Vannieuwenhoven, R. Vandebril, K. Meerbergen, A new truncation strategy for the higher-order singular value decomposition, SIAM J. Sci. Comput. 34 (2) (2012) A1027–A1052.
[50] R. Wang, Y. Li, E.F. Darve, On the numerical rank of radial basis function kernels in high dimensions, SIAM J. Matrix Anal. Appl. 39 (4) (2018) 1810–1835.
[51] R. Wang, Y. Li, M.W. Mahoney, E. Darve, Block basis factorization for scalable kernel evaluation, SIAM J. Matrix Anal. Appl. 40 (4) (2019) 1497–1526.
[52] J. Xia, S. Chandrasekaran, M. Gu, X.S. Li, Fast algorithms for hierarchically semiseparable matrices, Numer. Linear Algebra Appl. 17 (6) (2010) 953–976.
[53] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, J. Comput. Phys. 196 (2) (2004) 591–626.