Journal of Computational Physics 448 (2022) 110761

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A hierarchical matrix approach for computing hydrodynamic interactions [☆]

Xin Xing [a], Hua Huang [b], Edmond Chow [b,*]

[a] Department of Mathematics, University of California, Berkeley, CA, United States of America
[b] School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA, United States of America

ARTICLE INFO

Article history:
Available online 6 October 2021

Keywords:
Rotne–Prager–Yamakawa tensor
$\mathcal{H}^2$ matrix
Interpolative decomposition
Proxy surface method

ABSTRACT

For simulations of large numbers of small, spherical particles in a Stokes flow, the long-range hydrodynamic interactions approximated by the Rotne–Prager–Yamakawa (RPY) kernel can be summed rapidly using, for example, the fast multipole method (FMM) or the particle-mesh Ewald (PME) method. In this paper, we develop new fast methods for computing these sums using the $\mathcal{H}^2$ hierarchical matrix representation, for open and for periodic boundary conditions. To the best of our knowledge, the method for infinite periodic sums using the $\mathcal{H}^2$ hierarchical matrix representation is the first such method developed. We also consider a more general RPY kernel that handles polydisperse particle radii, and show analytically and experimentally that the proxy surface method for efficiently constructing the $\mathcal{H}^2$ hierarchical matrix representation remains effective in this case. Numerical tests demonstrate the well-controlled accuracy of the $\mathcal{H}^2$ summation methods and their linear-scaling computation and storage cost. We find that the $\mathcal{H}^2$ matrix approach has lower cost for computing the summations compared to FMM and PME, but higher precomputation cost (required for each particle configuration). This precomputation cost can be amortized over several summations when computing Brownian displacements or forces in Brownian and Stokesian dynamics simulations with very large numbers of particles.

1. Introduction

1.1. Hydrodynamic interactions

Particles immersed in a fluid will experience forces and move in response to fluid motion. In turn, particle motions will create fluid motions that affect the motion of other particles. These interactions between particles, mediated by the fluid, are called *hydrodynamic interactions*. These interactions are particularly significant for small particles in a slowly-moving, viscous fluid, e.g., suspensions of particles such as proteins and other macromolecules in a living cell. In these scenarios, hydrodynamic interactions may play a large role in diffusion, rheological properties, ordered structures, and collective motions [1–5].

Physical simulation has been a primary tool to explore the complex dynamics of particle suspensions. To model the hydrodynamic interactions between pairs of spherical particles of finite radius in Stokes flows, the Rotne–Prager–Yamakawa

☆ Supported by National Science Foundation under grants ACI-1609842 and ACI-2003683.
* Corresponding author.
E-mail addresses: xxing@berkeley.edu (X. Xing), huangh223@gatech.edu (H. Huang), echow@cc.gatech.edu (E. Chow).

https://doi.org/10.1016/j.jcp.2021.110761
0021-9991/© 2021 Elsevier Inc. All rights reserved.

(RPY) tensor kernel [6,7] is the most common approximation that is used. The resulting hydrodynamic effect on a particle is then taken to be the *summation* of the particle's pairwise interactions with all other particles. Hydrodynamic interactions are long range—the RPY kernel falls off as $1/|r|$ where $|r|$ is the distance between particles—and thus short-range truncations of the interactions generally cannot be used. The topic of this paper is the acceleration of the above summations when computing the hydrodynamic interactions for large numbers of particles.

For a system of N particles in the above setting, and ignoring the rotation of the particles, the resulting velocity v_i of the ith particle due to a set of forces f_j on the particles $j = 1, \dots, N$ is

$$v_i = \sum_{j=1}^{N} K(x_i, x_j) f_j,$$

where x_j is the position of particle j, and $K(x_i, x_j)$ is the RPY kernel,

$$K(x,y) = \begin{cases} \dfrac{1}{6\pi\eta a} I, & x = y \\ \dfrac{1}{8\pi\eta|r|}\left[\left(I + \dfrac{rr^T}{|r|^2}\right) + \dfrac{2a^2}{|r|^2}\left(\dfrac{1}{3}I - \dfrac{rr^T}{|r|^2}\right)\right], & x \neq y \end{cases}$$

where $r = x - y$. Here, we have assumed nonoverlapping particles only for brevity of the introduction (see (3) for the definition in the overlapping case), a is the radius of all the particles, η is the fluid viscosity, and I is the 3×3 identity matrix. A particle's "interaction" with itself corresponds to hydrodynamic drag. Computing the $3N \times 1$ vector v of all velocities has the form of a matrix-vector multiplication

$$v = Kf, \tag{1}$$

where f is the vector of forces and K is the $3N \times 3N$ symmetric positive definite *kernel matrix* associated with the RPY kernel and the particle positions. The set of sums represented by v in (1) can be computed efficiently via the fast multipole method (FMM) or the particle-mesh Ewald (PME) method, among others. In FMM and PME, the kernel matrix K is not formed explicitly.

1.2. Brownian simulations

Brownian forces on particles due to random collisions with fluid molecules are correlated hydrodynamically, and it is the computation of these forces or displacements that makes hydrodynamic interactions the bottleneck in large Brownian simulations when such interactions are modeled. Their calculation also involves RPY summation of the form (1) as we discuss now.

In the Ermak-McCammon algorithm [8] for Brownian dynamics (BD), the $3N \times 1$ vector x of particle positions evolves in time t as

$$\begin{aligned} x(t+\Delta t) &= x(t) + K\,f\,\Delta t + k_B T(\nabla \cdot K)\Delta t + g, \\ \langle g \rangle &= 0, \quad \langle g\,g^T \rangle = 2k_B T K \Delta t, \end{aligned} \tag{2}$$

where K, f, and g are quantities at the current time step, Δt is the time step size, and $k_B T$ is the Boltzmann constant times the temperature. For the RPY kernel, $\nabla \cdot K$ vanishes. The quantity g is a *Brownian displacement* sampled from a Gaussian distribution with covariance proportional to K. This covariance condition is a consequence of the fluctuation-dissipation theorem that relates fluctuations in Brownian forces with friction. Thus the Brownian displacements of the particles are correlated through hydrodynamic interactions.

The standard way to compute a Brownian displacement g is to first compute the lower triangular Cholesky factor or principal square root S of K and then compute $g = (2k_B T \Delta t)^{1/2} S z$ where z is a standard Gaussian random vector. However, in large-scale cases where FMM or PME is used, the matrix K, as mentioned, is not formed and thus not available to be factored. In these cases, the Brownian displacements are computed with $p(K)z$ where $p(K)$ is a matrix polynomial approximating the square root of K in some interval. The polynomial could be a Chebyshev polynomial [9] or one generated by a Krylov subspace method [10–14]. Note that $p(K)$ itself is not computed, but only its action on z, which only requires d matrix-vector multiplications with the matrix K, where d is the degree of the polynomial, with larger d giving a less biased sample. In other words, we only need operations of the form (1).

In Stokesian dynamics (SD) simulations [15–18], the Brownian *forces* are Gaussian and correlated with covariance matrix $R = K^{-1} + R_{\text{lub}}$ where R_{lub} is a sparse "lubrication" matrix which improves the modeling of short-range hydrodynamic interactions compared to using the RPY kernel alone. In SD codes, solves with R or, equivalently, $I + KR_{\text{lub}}$ are required, and an iterative solver used for large-scale models requires repeated matrix-vector multiplications with a given RPY kernel matrix K [19–22].

1.3. FMM, PME, and hierarchical matrix representations

FMM has been applied to compute the RPY summation (1) in $O(N)$ time. Two approaches using the FMM have been taken: writing the RPY summation in terms of Coulombic summations [23,24], and using the kernel-independent FMM [25]. The above approaches were applied for particle systems with open boundary conditions. For periodic boundary conditions, PME has been used [26,19,27–29], giving the RPY summation in $O(N \log N)$ time. (In principle, a periodic FMM could also be applied for the RPY summation in the case of periodic boundary conditions.) In this paper, we develop alternative fast methods for the RPY summation by using the $\mathcal{H}^2$ hierarchical matrix representation. We develop $O(N)$ algorithms for both open and periodic boundary conditions.

Hierarchical matrix representations, also called *rank-structured* matrix representations, exploit hierarchical block low-rank structure in matrices in order to accelerate operations with these matrices. In particular, this structure can be exploited to compute the kernel summation (1) in $O(N)$ time and space with a specific hierarchical matrix representation called $\mathcal{H}^2$ [30,31]. Kernel summation using this representation is intimately related to the FMM [32–34]. The main difference is that FMM requires a degenerate approximation of the kernel function (e.g., a multipole expansion), while forming the $\mathcal{H}^2$ matrix representation requires algebraic or hybrid analytic-algebraic techniques, e.g., [35–37], which can give more compact representations (lower ranks of the blocks) but with higher precomputation cost to “construct” or “build” the representation.

More precisely, $\mathcal{H}^2$ matrix representations are based the property that, for two *well separated* (to be defined in Section 3) sets of particles, X and Y, and for smooth kernel functions, the kernel matrix block $K(X, Y)$ containing interactions between the two sets of particles has small $O(1)$ numerical rank. Compressing all such blocks of maximum size in the kernel matrix into low-rank form leads to the $\mathcal{H}^2$ matrix representation. Compression using algebraic techniques such as SVD and rank-revealing QR decomposition, however, can lead to expensive, quadratic $\mathcal{H}^2$ construction cost. For kernel functions from potential theory, compression using a hybrid analytic-algebraic technique called the *proxy surface method* [38–41] can efficiently compress $K(X, Y)$ via algebraically compressing an intermediate, small matrix $K(X, Y_p)$. The artificial *proxy particles* Y_p, replacing Y in its interactions with X, lie uniformly on a surface separating X from Y. This paper will focus on using the proxy surface method. An extension of the proxy surface method to more general kernel functions, such as Gaussians, has also been developed [42].

Another class of hierarchical matrix representations, referred to as fast direct solvers, which includes recursive skeletonization [41] and hierarchically semiseparable matrices [43,44], may also be used for kernel summation and additionally allow for possible efficient symmetric decomposition of a kernel matrix K for computing Brownian displacements directly. However, compared to the linear complexity of using the $\mathcal{H}^2$ representation, these special representations generally require quadratic precomputation (even with the proxy surface method) and storage costs for a kernel matrix K defined by particles in 3-dimensional space. Experimentally, we have found these high costs to outweigh the benefit of having a symmetric decomposition of K, but further development of fast direct solvers may make them viable for Brownian simulations in the future.

1.4. Contributions of this paper

In this paper, we consider a more general version of the RPY kernel, where the radii of the particles may differ. This is important in simulations of, for example, proteins in the cytoplasm, where different proteins are modeled by particles of different sizes corresponding to their hydrodynamic radii [45]. This general RPY kernel, defined in (3) later in this paper, is a function of two particle radii as well as two particle positions, making it unclear whether the proxy surface method can be used in this case. In this paper, we show analytically that the proxy surface method remains effective for the general RPY kernel when we choose the radii of the proxy particles to be zero.

In this paper, we also address the use of $\mathcal{H}^2$ matrix representations with periodic boundary conditions. More precisely, the central cell (simulation box) is repeated in all directions and tiles all of space. We show that the RPY interactions between the central cell and each adjacent image cell can also be represented in the $\mathcal{H}^2$ matrix representation. We further show that the remaining interactions, i.e., the summation of the RPY interactions between the central cell and all nonadjacent image cells, can be represented in low-rank form and thus also has linear multiplication cost. To the best of our knowledge, our algorithm for such infinite periodic sums using the $\mathcal{H}^2$ hierarchical matrix representation is the first such algorithm developed. It does, however, bear similarities to periodic FMM algorithms, e.g., [46–48].

Compared to FMM and PME, the $\mathcal{H}^2$ matrix representation helps compute RPY summations more rapidly, but there is a high *precomputation cost* to construct the representation itself. In large-scale BD simulations, this precomputation cost can be amortized over the number of matrix-vector multiplications needed to compute $p(K)z$, i.e., a sample of the Brownian displacement. Further, in some BD simulations, the same K using the particle configuration at one time step can be used for multiple time steps without significantly altering the resulting macroscopic properties [13], allowing further amortization of the precomputation cost. In large-scale SD simulations, the precomputation cost can be amortized over the large number of matrix-vector multiplications required for a given K at a single time step.

Our methods are implemented in an open-source, general-purpose, high-performance $\mathcal{H}^2$ matrix package called H2Pack [49].

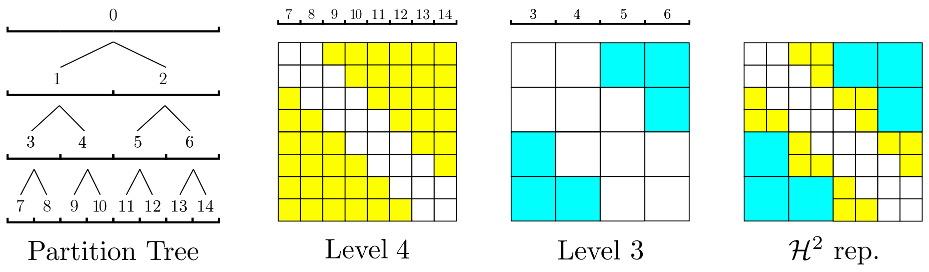

Fig. 1. Example of an $\mathcal{H}^2$ matrix representation with a binary partition tree for points in 1D space and open boundary conditions. Each colored block $K(X_i, X_j)$ (at different levels) is associated with two nonadjacent boxes i and j at the same level, and is represented in low-rank form. Each white block is associated with two adjacent or identical boxes at the same level. White blocks at the leaf level are represented in dense form in the $\mathcal{H}^2$ matrix representation.

2. Background

2.1. General RPY kernel and notation

Consider a set of N particles located at $x_1, x_2, \ldots, x_N$ with radii $a_1, a_2, \ldots, a_N$, respectively. Let $\boldsymbol{x_i}$ denote the pair $\{x_i, a_i\}$ and let $X = \{\boldsymbol{x}_i\}$ denote the set of all particles. For two particles $\boldsymbol{x} = \{x, a\}$ and $\boldsymbol{y} = \{y, b\}$, the RPY kernel $K(\boldsymbol{x}, \boldsymbol{y}) : \mathbb{R}^4 \times \mathbb{R}^4 \to \mathbb{R}^{3\times 3}$ for the case of polydisperse or nonuniform particle radii (see Ref. [50]) is defined as

$$K(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} \dfrac{1}{8\pi\eta|r|}\left[\left(1+\dfrac{a^2+b^2}{3|r|^2}\right)I + \left(1-\dfrac{a^2+b^2}{|r|^2}\right)\dfrac{rr^T}{|r|^2}\right], & r > a+b \\ \dfrac{1}{6\pi\eta ab}\left[\dfrac{16|r|^3(a+b)-((a-b)^2+3|r|^2)^2}{32|r|^3}I + \cdots \right. & \\ \qquad \left. \dfrac{3((a-b)^2-|r|^2)^2}{32|r|^3}\dfrac{rr^T}{|r|^2}\right], & |a-b| < r < a+b \\ \dfrac{1}{6\pi\eta\max(a,b)}I, & r < |a-b| \end{cases} \tag{3}$$

with $r = x - y$ and fluid viscosity η. Above, we have included the case where particles may overlap. For two sets of particles X_* and Y_*, $K(X_*, Y_*)$ denotes the RPY interaction block consisting of all $K(\boldsymbol{x}_i, \boldsymbol{y}_j)$ with $\boldsymbol{x}_i \in X_*$ and $\boldsymbol{y}_j \in Y_*$. We use a_i and b_j to denote the radii of such two particles $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ in $K(X_*, Y_*)$, respectively. Given a set of particles X, the kernel matrix in the nonperiodic case is defined as $K(X, X)$ and in the periodic case is formally defined as $\sum_s K(X, X+s)$ where $\{X+s\}$ denotes all the images of X translated by lattice vectors s.

2.2. $\mathcal{H}^2$ matrix representation

Given a set of particles X, an $\mathcal{H}^2$ matrix representation of the kernel matrix $K(X, X)$ starts with a hierarchical partitioning of X. A cubic box enclosing all particles is adaptively and recursively bisected into smaller cubic boxes until the number of particles in each finest box is less than a prescribed constant. This hierarchical partitioning of X can be described by a *partition tree* $\mathcal{T}$, which is an octree in the 3-dimensional (3D) case. Each node of $\mathcal{T}$ corresponds to a cubic box and to the set of particles with centers in this box. For simplicity, we assume $\mathcal{T}$ to be perfect (fully populated in each level). The case of nonperfect partition trees is discussed in [49].

For each node $i \in \mathcal{T}$, let X_i denote the set of particles centered in box i and let Y_i denote the set of particles centered in all boxes that are separated from box i by at least one box of the same size as box i. Such a set of particles Y_i and boxes that contain these particles are said to be *well separated* from X_i or box i. See Fig. 2 later in this paper for an example of X_i and Y_i (denoted as X_* and Y_* in the figure). Using this notation, block $K(X_i, X_j)$ denotes the interactions between particles in box i and in box j, and $K(X_i, Y_i)$ denotes the interactions between particles in box i and all particles well separated from box i (assuming Y_i is not empty). An $\mathcal{H}^2$ matrix representation of $K(X, X)$ is built upon the property that $K(X_i, Y_i)$ for each node i always has small $O(1)$ numerical rank.

An $\mathcal{H}^2$ matrix representation consists of (i) dense blocks $K(X_i, X_j)$ with leaf nodes i and j whose boxes are adjacent or identical to each other ($i = j$), and (ii) low-rank approximations of blocks $K(X_i, X_j)$ with well separated nodes i and j at the same level that are not contained in larger low-rank blocks at upper levels. Fig. 1 illustrates an $\mathcal{H}^2$ matrix representation for particles in 1-dimensional (1D) space hierarchically partitioned into boxes. The root node in the partition tree is defined to be at level 1. Note in the figure that only levels 3 and 4 are interesting; for a node i in levels 1 or 2, all Y_i are empty.

Using an $\mathcal{H}^2$ matrix representation, the matrix-vector multiplication of $K(X, X)$ can be computed with linear cost [30,31, 49] by appropriately traversing the above dense and low-rank approximated blocks and accumulating the partial products.

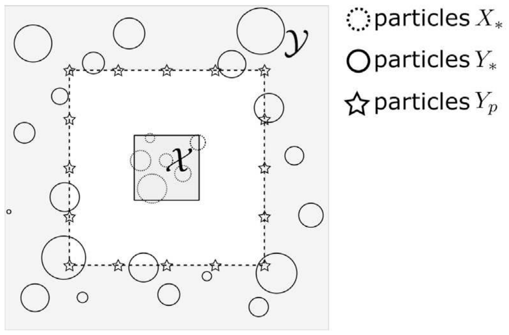

Fig. 2. 2-dimensional (2D) illustration of the proxy surface method.

2.3. *Interpolative decomposition*

In $\mathcal{H}^2$ matrix representations, we use a particular form for the low rank approximations called interpolative decomposition (ID). Given a matrix $A \in \mathbb{R}^{m\times n}$, a rank-$k$ ID approximation of A has the form UA_J where $U \in \mathbb{R}^{m\times k}$ has bounded entries and $A_J \in \mathbb{R}^{k\times n}$ contains k rows of A. For a fixed approximation rank or an error threshold, an ID approximation of A can be computed *algebraically* by using the QR decomposition of A^T with column pivoting.

To construct an $\mathcal{H}^2$ matrix representation, first compute an ID approximation of $K(X_i, Y_i)$ for each node i in the partition tree with nonempty Y_i,

$$K(X_i, Y_i) \approx U_i K(X_i^{\mathrm{id}}, Y_i), \quad X_i^{\mathrm{id}} \subset X_i. \tag{4}$$

Then, for two nonadjacent boxes i and j at the same level of the partition tree, it holds by definition that $X_j \subset Y_i$ and $X_i \subset Y_j$, and thus block $K(X_i, X_j)$ can be approximated by

$$K(X_i, X_j) \approx U_i K(X_i^{\mathrm{id}}, X_j^{\mathrm{id}}) U_j^T, \tag{5}$$

using the ID approximations (4) associated with nodes i and j.

In constructing an $\mathcal{H}^2$ matrix representation of $K(X, X)$, the ID approximation (4) of $K(X_i, Y_i)$ is computed explicitly for each leaf node i, and implicitly for each nonleaf node i using a nested approach [49], i.e., U_i for a nonleaf node i is expressed in terms of the U matrices for its children. This is called the *nested basis property* of $\mathcal{H}^2$ matrix representations and is necessary for its linear-scaling matrix-vector multiplication.

2.4. *Proxy surface method*

Calculating an ID decomposition efficiently is the key behind the efficient construction of many kinds of rank-structured matrix representations. When the block to be approximated is large, the algebraic method of using the pivoted QR decomposition to compute the ID approximation is very costly. We now discuss the proxy surface method [38–41] which can compute the ID approximation efficiently in many cases.

First define a set of particles X_* with centers in a box denoted as $\mathcal{X}$, and another set of particles Y_* centered in the union of boxes, denoted as $\mathcal{Y}$, that are well separated from $\mathcal{X}$. For a kernel function K from potential theory, the proxy surface method computes the ID approximation

$$(X_*, Y_*) \approx UK(X_*^{\mathrm{id}}, Y_*),$$

where U and X_*^{id} are the result of an algebraic calculation of the ID approximation

$$K(X_*, Y_p) \approx UK(X_*^{\mathrm{id}}, Y_p),$$

where Y_p is a small set of *proxy particles* that are uniformly sampled on the boundary of the far field $\partial\mathcal{Y}$ that encloses $\mathcal{X}$, i.e., the proxy surface, as illustrated in Fig. 2. Since Y_p is a small set, the matrix $K(X_*, Y_p)$ is small and the algebraic calculation of its ID approximation is inexpensive. For kernel functions from potential theory, the effectiveness of the proxy surface method can be justified by Green's theorem [51].

3. $\mathcal{H}^2$ RPY summation: nonperiodic case

The $\mathcal{H}^2$ matrix representation for the RPY kernel matrix $K(X, X)$ for a set of points X can be constructed in a straightforward fashion, using the proxy surface method to calculate the low rank ID approximations as described in the previous section. However, the proxy surface method was originally designed [38] as a fast method for computing an ID approximation for kernel matrix blocks associated with *point sources* and kernel functions from potential theory. Since our general RPY kernel (3) involves finite and nonuniform particle radii, rather than just particle locations, it is not clear that the proxy surface method calculates accurate ID approximations in this case.

We provide a theoretical error analysis of the proxy surface method for the RPY kernel with nonuniform particle radii in Appendix A. The analysis shows that the ID components U and X_*^{id} computed by the proxy surface method in this case define a good *vector function* approximation,

$$K(X_*, \boldsymbol{y}) \approx U K(X_*^{\mathrm{id}}, \boldsymbol{y}), \tag{6}$$

for any particles $\boldsymbol{y} = \{y, b\}$ with centers in domain $\mathcal{Y}$, i.e., $y \in \mathcal{Y}$. Plugging Y_* into this function approximation exactly gives the ID approximation of $K(X_*, Y_*)$. In the analysis, we choose the radius of the proxy particles to be 0. In practice, a small radius relative to the width of the box containing X_* can also be used.

It is worth noting that the proxy surface method and its analysis in Appendix A only work for the non-overlapping RPY interaction in (3). To make sure the particle interactions between two nonadjacent boxes do not involve overlapping particles, we assure that all finest boxes have edge length greater than twice the maximum radius of all the particles in X when partitioning X hierarchically into boxes. In other words, all overlapping particle interactions are always performed directly and described explicitly in dense blocks $K(X_i, X_j)$ with adjacent or identical leaf boxes.

The proxy surface method for computing an ID approximation of $K(X_*, Y_*)$ with $O(\varepsilon)$ relative error is shown in Algorithm 1.

Algorithm 1 Proxy surface method for the RPY kernel.

```
Input: $X_*$, $Y_*$, $\partial\mathcal{Y}$, relative error threshold $\varepsilon$
Output: $U$ and $X_*^{\mathrm{id}}$ for an ID approximation $K(X_*, Y_*) \approx U K(X_*^{\mathrm{id}}, Y_*)$
  Step 1: sample a set of particles $Y_p$ with zero radii uniformly on $\partial\mathcal{Y}$
  Step 2: calculate $U$ and $X_*^{\mathrm{id}}$ by an algebraic ID approximation of $K(X_*, Y_p)$
          $K(X_*, Y_p) \approx U K(X_*^{\mathrm{id}}, Y_p),$        (7)
  with relative error threshold $\varepsilon$
```

In Step 2 of the algorithm, since the RPY kernel is a tensor kernel, we use a "grouped" version of the ID approximation. In the grouped version, the row subset $K(X_*^{\mathrm{id}}, Y_p)$ in (7) always contains either all or none of the three rows associated with each particle in X_*. This grouping is necessary to efficiently satisfy the nested basis requirement of $\mathcal{H}^2$ matrix representations.

The grouped ID approximation can be computed algebraically by the QR decomposition of $K(X_*, Y_p)^T = K(Y_p, X_*)$ using "group-column" pivoting. Every three columns of $K(Y_p, X_*)$ associated with a particle in X_* are grouped together. The column pivoting step in QR is applied over the groups and is followed by three consecutive Householder transformations on the three columns in the pivot group. Such an ID approximation to $K(X_*, Y_*)$ is said to have error below a threshold ε_0 if the approximation to every grouped three rows has error in the Frobenius norm bounded by ε_0. A relative error threshold can be defined similarly as in the general pivoted QR decomposition.

Using Algorithm 1 for the general RPY kernel, we have found experimentally that the number of proxy particles required in the algorithm depends on the desired accuracy of the ID approximation, but not on the absolute size of $\mathcal{X}$ or $\mathcal{Y}$ or the number of points in X_* or Y_*. This observation also applies to the original proxy surface method for the 3D Laplace kernel [51]. More specifically, to obtain an ID approximation of $K(X_*, Y_*)$ with relative error up to 10^{-k}, numerical results show that it is sufficient to select $6(k+1)^2$ proxy particles corresponding to $(k+1) \times (k+1)$ regular grid points on each face of the proxy surface.

Using a numerical computation, Fig. 3 plots the relative approximation errors of the proxy surface method, showing it has accuracy similar to that of algebraic compression methods. From the results, we note that when a relative error threshold ε is used for the algebraic ID approximation of $K(X_*, Y_p)$, the defined ID approximation of $K(X_*, Y_*)$ has relative error comparable to or even smaller than ε.

Finally, Algorithm 2 shows the $\mathcal{H}^2$-based RPY summation algorithm for a nonperiodic system of particles (open boundary conditions). The first step constructs an $\mathcal{H}^2$ matrix representation of $K(X, X)$ for a given set of particles X. By using the proxy surface method, the cost of this step is linear in the number of particles. The second step performs the $\mathcal{H}^2$ matrix-vector multiplication with a multiplicand f, which also has linear cost. The second step can be repeated for a new multiplicand (without repeating the first step) for the same particle configuration.

In the implementation of Algorithm 2, the dense $K(X_i, X_j)$ and the intermediate blocks $K(X_i^{\mathrm{id}}, X_j^{\mathrm{id}})$ in the approximation (5) are not stored and are dynamically computed when needed. In the precomputation step, U_i and X_i^{id} for all nodes i at the same level can be constructed in parallel, and the construction proceeds from the leaf level to the root level. In the $\mathcal{H}^2$ matrix-vector multiplication, the dominant computation is the multiplication with all the dense blocks $K(X_i, X_j)$ and the intermediate blocks $K(X_i^{\mathrm{id}}, X_j^{\mathrm{id}})$. All these block multiplications are independent and can be computed in parallel. See [49] for a detailed discussion on the parallel implementation of $\mathcal{H}^2$ matrix construction and matrix-vector multiplication.

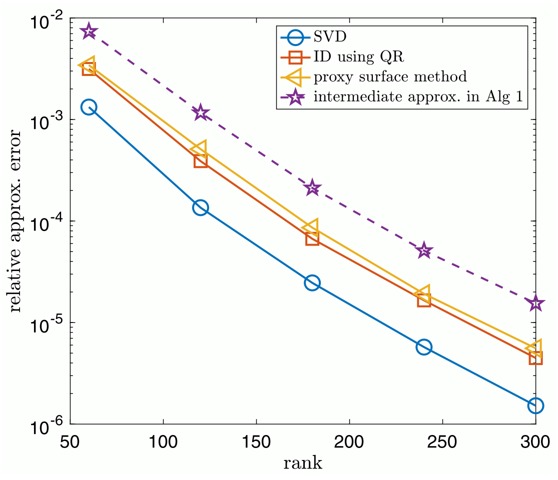

Fig. 3. Relative errors of the low-rank approximations of $K(X_*, Y_*)$ in the Frobenius norm with different ranks. Three methods are tested: SVD, ID using pivoted QR, and ID using the proxy surface method. The dashed line denotes the relative errors of the intermediate ID approximation of $K(X_*, Y_p)$ in Algorithm 1. The test RPY block $K(X_*, Y_*)$ is constructed by 400 particles X_* randomly distributed in the box $\mathcal{X} = [-l/2,\ l/2]^3$ of edge length $l = 30$ and 39200 particles Y_* in the annulus $\mathcal{Y} = [-5l/2,\ 5l/2]^3 \setminus [-3l/2,\ 3l/2]^3$. The proxy surface method has $6 \times (7 \times 7) = 294$ proxy particles Y_p on the cubical surface $\partial[-3l/2,\ 3l/2]^3$. Particles in X_* and Y_* have radii randomly sampled from a uniform distribution in $[1, 10]$.

Algorithm 2 $\mathcal{H}^2$-based RPY summation for nonperiodic systems.

Input: the set of particles X, relative error threshold ε, multiplicand vector f
Output: the product $K(X, X)f$ with relative error $O(\varepsilon)$

Step 1: Precomputation

```
• generate a hierarchical partition of $X$, denoted by a $L$-level tree $\mathcal{T}$
for $l = L, L-1, \dots, 1$ do
    for node $i$ in level $l$ do
        if $Y_i$ is nonempty then
            • compute $U_i$ and $X_i^{\text{id}}$ from an ID approximation of $K(X_i, Y_i)$ with relative error threshold $\varepsilon$ using Algorithm 1 and the nested ID approximation approach in $\mathcal{H}^2$ matrix construction (see [49, Sec. 2.2] for detailed steps)
        end if
    end for
end for
```

Step 2: Multiplication

```
• apply $\mathcal{H}^2$ matrix-vector multiplication (see [49, Alg. 3] for detailed steps) to compute $K(X, X)f$
```

4. $\mathcal{H}^2$ RPY summation: periodic case

We now consider the RPY kernel and particle systems with 3D periodicity. The following discussion can be easily generalized to other kernel functions and to quasi-1D or quasi-2D periodic systems in 3D space.

Consider a cubic unit cell $[-l/2, l/2]^3$ of edge length l containing N particles, $X = \{\boldsymbol{x}_i\}$. The set of lattice vectors for this periodic system is denoted by $\mathcal{L} = \{(k_1 l,\ k_2 l,\ k_3 l) \mid k_1, k_2, k_3 \in \mathbb{Z}\}$. The RPY summation for this system is

$$v_i = \sum_{s \in \mathcal{L}} \sum_{j=1}^{N} K(\boldsymbol{x}_i, \boldsymbol{x}_j + s) f_j, \quad i = 1, \dots, N, \tag{8}$$

where $\{v_i\}$ and $\{f_i\}$ are 3-dimensional vectors associated with the particles in X, the notation $\boldsymbol{x}_j + s$ refers to the translation of particle $\boldsymbol{x}_j$ by s without modifying its radius, and $K(\boldsymbol{x}, \boldsymbol{y})$ is the RPY kernel. Define the *periodic RPY kernel* as

$$K_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{s \in \mathcal{L}} K(\boldsymbol{x}, \boldsymbol{y} + s),$$

such that the summation in (8) corresponds to the matrix-vector multiplication $v = K_{\mathcal{L}}(X, X)f$. It is worth noting that this definition of $K_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y})$ is formal since the summation in (8) is conditionally convergent and is computed in practice by an Ewald summation [52] under the assumption $\sum_j f_j = 0$.

To construct the $\mathcal{H}^2$ matrix representation of $K_{\mathcal{L}}(X, X)$, we first partition X hierarchically, as in the nonperiodic case. This partitioning, which defines the partition tree $\mathcal{T}$, applies to each image of X as well. Next, we use the proxy surface method to compute an ID approximation (using the *nonperiodic* kernel),

$$K(X_i, Y_p^{(i)}) \approx U_i K(X_i^{\text{id}}, Y_p^{(i)}), \quad X_i^{\text{id}} \subset X_i,$$

for *every* node $i \in \mathcal{T}$ (including the root node). Here, $Y_p^{(i)}$ denotes the set of proxy particles selected in Algorithm 1 for X_i. According to the analysis of the proxy surface method in Appendix A, it holds that the computed U_i and X_i^{id} define a good ID approximation $K(X_i, Y_*) \approx U_i K(X_i^{\text{id}}, Y_*)$ for any set of particles Y_* that are well separated from the particles in X_i. Due to the translational invariance of $K(\boldsymbol{x}, \boldsymbol{y})$, it further holds that for any image of X_i, i.e., $X_i + s$ with $s \in \mathcal{L}$, the quantities U_i and $X_i^{\text{id}} + s$ define a good ID approximation

$$K(X_i + s, Y_*) \approx U_i K(X_i^{\text{id}} + s, Y_*), \tag{9}$$

for any Y_* that is well separated from $X_i + s$.

Let $\mathcal{N}$ be the set of 27 lattice vectors that define the $3 \times 3 \times 3$ cells around and including the central cell. The matrix-vector multiplication in (8) can be split as

$$K_{\mathcal{L}}(X, X) f = \sum_{s \in \mathcal{N}} K(X, X + s) f + \sum_{s \in \mathcal{L} \setminus \mathcal{N}} K(X, X + s) f. \tag{10}$$

For each $s \in \mathcal{N}$, $K(X, X + s)$ can be represented in $\mathcal{H}^2$ format using the components $\{U_i\}$ and $\{X_i^{\text{id}}\}$ just like in the nonperiodic case. Specifically, the $\mathcal{H}^2$ matrix representation of $K(X, X + s)$ consists of (i) the dense blocks $K(X_i, X_j + s)$ with leaf nodes i and j such that X_i is adjacent or identical to $X_j + s$, and (ii) the low-rank approximations of blocks $K(X_i, X_j + s)$ given as

$$K(X_i, X_j + s) \approx U_i K(X_i^{\text{id}}, X_j^{\text{id}} + s) U_j^T,$$

with nodes i and j on the same level such that X_i and $X_j + s$ are well separated, and $K(X_i, X_j + s)$ is not contained in a larger low-rank block at an upper level. This low-rank approximation is derived from the computed ID approximations for X_i and $X_j + s$. Each multiplication $K(X, X + s) f$ in the first term of (10) can thus be efficiently computed using $\mathcal{H}^2$ matrix-vector multiplication.

For the second term in (10), note that $X+s$ is well separated from X for any $s \in \mathcal{L} \setminus \mathcal{N}$. Thus, according to (9), $K(X, X+s)$ can be well approximated by $U_0 K(X_0^{\text{id}}, X_0^{\text{id}} + s) U_0^T$, where 0 indicates the root node ($X_0 = X$). Summing all these approximations of $K(X, X + s)$ together gives

$$\begin{aligned}
\sum_{s \in \mathcal{L} \setminus \mathcal{N}} K(X, X + s) &\approx \sum_{s \in \mathcal{L} \setminus \mathcal{N}} U_0 K(X_0^{\text{id}}, X_0^{\text{id}} + s) U_0^T \\
&= U_0 \left(\sum_{s \in \mathcal{L} \setminus \mathcal{N}} K(X_0^{\text{id}}, X_0^{\text{id}} + s) \right) U_0^T \\
&= U_0 \left(K_{\mathcal{L}}(X_0^{\text{id}}, X_0^{\text{id}}) - \sum_{s \in \mathcal{N}} K(X_0^{\text{id}}, X_0^{\text{id}} + s) \right) U_0^T \\
&= U_0 B_{00} U_0^T,
\end{aligned} \tag{11}$$

where B_{00} denotes the bracketed term in the third equation and $U_0 B_{00} U_0^T$ gives a low-rank approximation to the overall matrix summation. The second term in (10) can thus be approximately and efficiently computed as $U_0(B_{00}(U_0^T f))$ with $O(|X||X_0^{\text{id}}|)$ computation cost.

Overall, the $\mathcal{H}^2$-based multiplication algorithm for the periodic case is shown in Algorithm 3. The intermediate block B_{00} is precomputed by plain Ewald summation and stored because the evaluation of $K_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y})$ is usually expensive. In comparison with the nonperiodic case (Algorithm 2), the periodic case additionally computes $\{U_i\}$ and $\{X_i^{\text{id}}\}$ for nodes i with empty Y_i, i.e., the nodes in levels 1 and 2 of the partition tree, and the intermediate block B_{00}. The total precomputation and storage costs of Algorithm 3 both scale linearly and are only slightly more expensive than those of the nonperiodic case.

After the precomputation, the multiplication step involves 27 $\mathcal{H}^2$ matrix-vector multiplications and one multiplication by the low-rank approximation $U_0 B_{00} U_0^T$, and has linear computation cost in total. We note that many calculations in the 27 $\mathcal{H}^2$ matrix-vector multiplications and the low-rank multiplication are shared and only need to be computed once, such as the matrix-vector multiplications by each U_i^T and U_i. All dense blocks $K(X_i, X_j + s)$ and intermediate blocks $K(X_i^{\text{id}}, X_j^{\text{id}} + s)$ are dynamically computed in the multiplication step.

5. Numerical experiments

We refer to the $\mathcal{H}^2$ matrix-vector multiplication algorithm for the RPY kernel, i.e., RPY summation, with either nonperiodic or periodic systems, as $\mathcal{H}^2$-RPY. To demonstrate its accuracy and efficiency, we test the method on particle systems

Algorithm 3 $\mathcal{H}^2$-based RPY summation for periodic systems.

```
Input: the set of particles $X$, unit cell vectors (that define the lattice vector set $\mathcal{L}$ and the 27 neighboring lattice vectors $\mathcal{N}$ for the splitting in (10)), relative error threshold $\varepsilon$, multiplicand vector $f$
Output: the product $K_{\mathcal{L}}(X, X)f$ with relative error $O(\varepsilon)$

Step 1: Precomputation
• generate a hierarchical partition of $X$, denoted by a $L$-level tree $\mathcal{T}$
for $l = L, L-1, \ldots, 1$ do
    for node $i$ in level $l$ do
        • compute $U_i$ and $X_i^{\text{id}}$ from an ID approximation of $K(X_i, Y_p^{(i)})$ with relative error threshold $\varepsilon$ using Algorithm 1 and the nested ID approximation approach in $\mathcal{H}^2$ matrix construction (see [49, Sec. 2.2] for detailed steps)
    end for
end for
• compute and store $B_{00} = K_{\mathcal{L}}(X_0^{\text{id}}, X_0^{\text{id}}) - \sum_{s\in\mathcal{N}} K(X_0^{\text{id}}, X_0^{\text{id}} + s)$, with the first term computed using plain Ewald summation

Step 2: Multiplication
• apply $\mathcal{H}^2$ matrix-vector multiplication (see [49, Alg. 3] for detailed steps) to compute $K(X, X+s)f$ for $s \in \mathcal{N}$
• compute $U_0(B_{00}(U_0^T f))$
• accumulate the above multiplication results:
    $$K_{\mathcal{L}}(X, X)f = \sum_{s\in\mathcal{N}} K(X, X+s)f + U_0 B_{00} U_0^T f$$
```

with different volume fractions, 0.01, 0.1, 0.2, and 0.3. For context, the volume fraction of macromolecules in the cytoplasm of living cells is believed to be between 0.2 and 0.4 [53], while the volume fraction of DNA in the nucleoid of bacterial cells is believed to be between 0.1 and 0.2 [4]. For each test system, particles are randomly distributed inside a cubic box with their radii randomly and uniformly sampled from the interval [1, 10]. Unless otherwise indicated, timings and relative errors for $\mathcal{H}^2$-based matrix-vector multiplications are averaged over five trials (with the same kernel matrix).

The $\mathcal{H}^2$-RPY method is implemented within an existing $\mathcal{H}^2$ matrix package called H2Pack [49]. Comparisons are made with existing codes, RPYFMM [24] and StokesDT [28]. The tests are executed using multithreaded parallelism on a dual Intel Xeon Gold 6226 CPU computer with 180 GB of main memory, using all 24 cores and one hyperthread per core.

The $\mathcal{H}^2$-RPY method only takes one parameter, ε, the threshold for controlling the relative error. For $\varepsilon = 10^{-k}$, Algorithm 1 for computing the ID approximations in $\mathcal{H}^2$ matrix construction uses $6(k+1)^2$ proxy particles (see Section 3). In Algorithm 3 for periodic systems, the Ewald summation for constructing $K_{\mathcal{L}}(X_0^{\text{id}}, X_0^{\text{id}})$ sets the Ewald parameter $\alpha = \pi^{\frac{1}{2}}|V|^{-\frac{1}{3}}$ for a cubic unit cell of volume V and sets the range of both the real-space and the reciprocal-space summations to 4 image cells in all directions. This is a very accurate setting, suitable for ε as small as 10^{-11}, but this Ewald summation is not a significant cost since $K_{\mathcal{L}}(X_0^{\text{id}}, X_0^{\text{id}})$ is a small matrix. In the hierarchical partitioning of the given particles, a box is not further bisected when its edge length is smaller than four times the maximum particle radius (to avoid overlapping RPY interactions between non-adjacent boxes, see Section 3), or when it contains fewer than 300 particles. The value of 300 is commonly chosen in FMM codes for computational efficiency but may vary for different computer architectures.

5.1. Accuracy

Recall that a relative error threshold ε is input to the proxy surface method to control the accuracy and cost of an $\mathcal{H}^2$ matrix representation. The actual relative error of the representation can be estimated by the accuracy of matrix-vector multiplication using the representation. More precisely, if we denote the computed result of the multiplication as $\bar{b}$ and the numerically exact result as b (directly computed using the RPY kernel), we compute the relative error

$$\text{relerr} = \frac{\sqrt{\sum_{i\in S}\left(b_i - \bar{b}_i\right)^2}}{\sqrt{\sum_{i\in S} b_i^2}}, \tag{12}$$

where the multiplicand vector is selected randomly from a standard multivariate Gaussian distribution. Since the exact multiplication can be expensive to compute for large cases, we only measure the accuracy for a set S of 2000 indices randomly chosen from the particle indices.

To test the accuracy of the $\mathcal{H}^2$ matrix representation of the RPY kernel matrix, we use four nonperiodic and four periodic systems of 1.28×10^6 particles with various volume fractions. For different input relative error thresholds, Table 1 shows the resulting relative error for $\mathcal{H}^2$-based matrix-vector multiplications (averaged over five trials). From the results, the relative errors of our $\mathcal{H}^2$-based matrix-vector multiplications are all well controlled by the specified relative error thresholds. Given the same error threshold, the multiplication in the periodic case tends to have larger actual relative errors than in the nonperiodic case, which is expected since more matrix blocks are approximated in the periodic case.

Table 1
Relative error of $\mathcal{H}^2$-based matrix-vector multiplications with different prescribed relative error thresholds for constructing the $\mathcal{H}^2$ matrix representations.

Rel. err. threshold ε	1.0e-2	1.0e-4	1.0e-6	1.0e-8	1.0e-10
Nonperiodic systems					
Volume fraction 0.01	5.2e-3	6.0e-5	7.4e-7	1.5e-8	3.4e-10
Volume fraction 0.1	4.9e-3	6.6e-5	9.5e-7	1.4e-8	3.3e-10
Volume fraction 0.2	5.0e-3	8.7e-5	9.1e-7	1.7e-8	3.6e-10
Volume fraction 0.3	4.7e-3	6.8e-5	1.2e-6	1.8e-8	2.8e-10
Periodic systems					
Volume fraction 0.01	1.3e-2	1.2e-4	1.6e-6	3.1e-8	3.3e-10
Volume fraction 0.1	2.0e-2	1.6e-4	1.6e-6	3.0e-8	6.7e-10
Volume fraction 0.2	3.4e-2	2.8e-4	2.7e-6	2.8e-8	7.0e-10
Volume fraction 0.3	1.9e-2	2.0e-4	2.3e-6	4.1e-8	8.8e-10

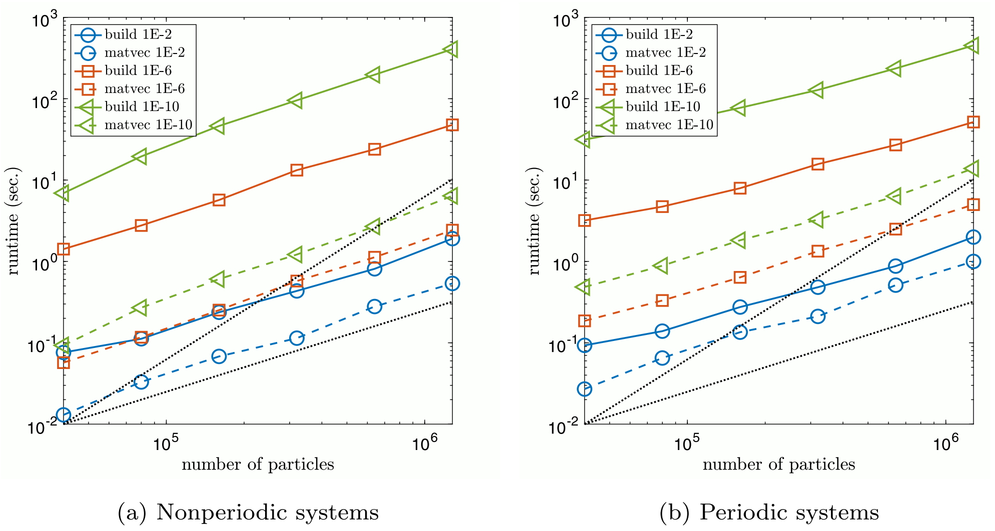

Fig. 4. Execution time for the precomputation ("build") and matrix-vector multiplication ("matvec") of $\mathcal{H}^2$-RPY with different relative error thresholds. The two dotted lines correspond to linear and quadratic scaling.

5.2. Computation and storage costs

We first consider a set of particle systems with fixed volume fraction 0.1. Fig. 4 plots the execution time for the precomputation and matrix-vector multiplication of $\mathcal{H}^2$-RPY with different relative error thresholds. The precomputation refers to the construction of the $\mathcal{H}^2$ matrix representation, i.e., the computation of ID components $\{U_i\}$ and $\{X_i^{\text{id}}\}$ in the nonperiodic case, and the computation of $\{U_i\}$, $\{X_i^{\text{id}}\}$, and B_{00} in the periodic case. Fig. 5 plots the corresponding storage costs.

From the results, both the precomputation and matrix-vector multiplication have linear computation and storage costs. For smaller relative error thresholds, the computation and storage costs become more expensive due to the larger approximation ranks required for ID approximations of RPY kernel blocks to meet the threshold.

We observe that the precomputation costs with relative error thresholds 10^{-2}, 10^{-6}, 10^{-10} are around 3, 20, and 70 times the corresponding matrix-vector multiplication cost, respectively, in the nonperiodic case, and 2, 10, and 35 times in the periodic case. Thus, the overhead of precomputation is relatively much smaller when lower accuracy is desired. The overhead of precomputation can be amortized if many matrix-vector multiplications are required for the same particle configuration, for example, in the iterative calculation of Brownian forces or displacements.

Table 2 further lists the execution time and storage cost for several systems with 1.28×10^6 particles but with different volume fractions. The computation and storage costs both have negligible differences for systems with different volume fractions. The additional $\mathcal{H}^2$ construction time for periodic systems compared to nonperiodic systems is mainly due to computing U_i and X_i^{id} components for nodes near the root level. Computation of B_{00} in Table 2 requires approximately 0.3 seconds. The $\mathcal{H}^2$ matrix-vector multiplication is approximately twice as expensive for periodic systems than for nonperiodic systems, mainly due to the multiplications with intermediate blocks $K(X_i^{\text{id}}, X_j^{\text{id}} + s)$, which are more numerous in the periodic case. In comparison, the overhead of periodic FMM over nonperiodic FMM is much smaller [46,54,55]. The reason for this is that the multipole-to-local operators in FMM (the analogue of intermediate blocks in $\mathcal{H}^2$ representations) are the

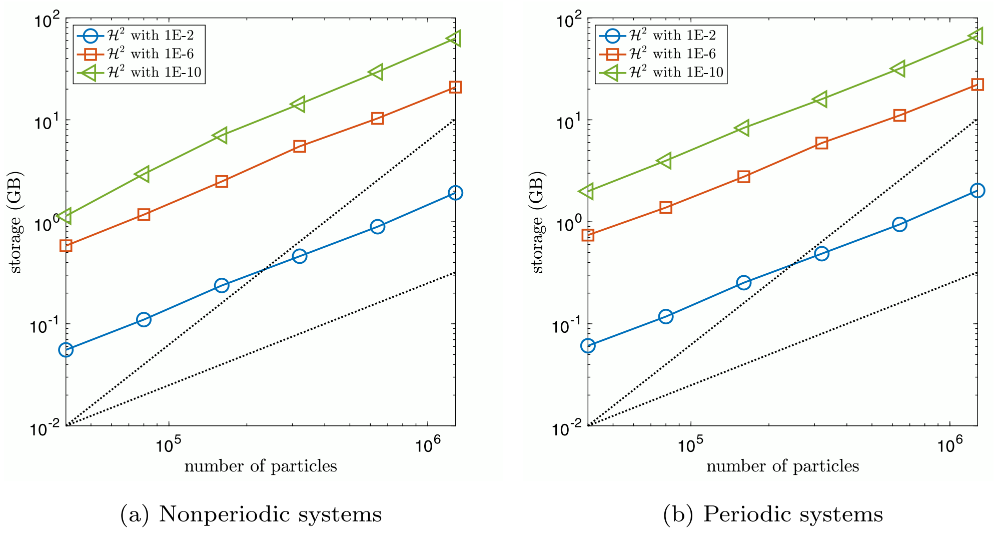

Fig. 5. Storage costs of $\mathcal{H}^2$-RPY with different relative error thresholds. The two dotted lines correspond to linear and quadratic scaling.

Table 2
Execution time for the precomputation and matrix-vector multiplication of $\mathcal{H}^2$-RPY and the corresponding storage costs for 1.28×10^6 particles with different volume fractions. The relative error threshold 10^{-6} is used for all tests.

Volume fraction	0.01	0.1	0.2	0.3
Nonperiodic systems				
$\mathcal{H}^2$ construction (sec.)	49.09	47.65	47.02	46.68
$\mathcal{H}^2$ matvec (sec.)	2.41	2.41	2.42	2.41
$\mathcal{H}^2$ storage (GB)	20.92	20.92	20.97	20.96
Periodic systems				
$\mathcal{H}^2$ construction (sec.)	53.26	51.49	51.00	50.83
$\mathcal{H}^2$ matvec (sec.)	5.04	5.05	5.03	5.02
$\mathcal{H}^2$ storage (GB)	22.19	22.19	22.23	22.24

same for all cluster pairs with the same relative position, which can reduce the computational cost of the multipole-to-local transformations in the periodic case.

5.3. Comparison with FMM for nonperiodic systems

To our best knowledge, no efficient FMM package is available for computing RPY interactions in systems of different-sized particles. Therefore, we compare $\mathcal{H}^2$-RPY with FMM only on systems of same-sized particles. RPYFMM [24] is a state-of-the-art package for FMM with the RPY kernel, and provides parallel implementations for both shared and distributed memory computations. Table 3 compares the numerical results of RPYFMM and $\mathcal{H}^2$-RPY method for several systems with different relative error thresholds while fixing the volume fraction to 0.1 and the particle radii to 1.0. For systems of different volume fractions, the numerical performance of both methods do not vary significantly.

As can be noted, the $\mathcal{H}^2$-RPY method has faster matrix-vector multiplications than RPYFMM but has relatively expensive precomputation cost. The better multiplication efficiency is due to the fact that $\mathcal{H}^2$-RPY compresses RPY interaction blocks more effectively (i.e., with smaller ranks) using the proxy surface method than FMM using multipole expansions. This advantage of $\mathcal{H}^2$-RPY further increases when particles lie on low-dimensional manifolds [49].

5.4. Comparison with SPME for periodic systems

Smooth particle-mesh Ewald (SPME) summation can be used straightforwardly to compute periodic RPY interactions, and can be generalized to work with systems of different-sized particles. For this method, let α denote the Ewald parameter that balances between real-space and reciprocal-space calculations, let $r_{\max}$ denote the cutoff for interactions computed in real-space, and let p denote the order of the B-spline interpolation used on the $n_{\text{fft}} \times n_{\text{fft}} \times n_{\text{fft}}$ reciprocal-space grid. The performance of SPME heavily relies on a careful selection of parameter values to balance computation cost and accuracy. In comparison, $\mathcal{H}^2$-RPY takes a single parameter that can directly control the multiplication accuracy.

We use the StokesDT package [28] for SPME calculations with systems of different-sized particles. StokesDT provides parallel implementations for both shared and distributed memory computations. For a given system of particles, all the

Table 3
Execution time (in sec.) and average relative multiplication error of RPYFMM and $\mathcal{H}^2$-RPY for different-sized systems and different relative error thresholds ε. For all the test systems, the volume fractions are set to 0.1 and the particle radii are set to 1.0. "Build" refers to the precomputation of $\mathcal{H}^2$-RPY.

Number of particles ($\times 10^4$)	RPYFMM		$\mathcal{H}^2$-RPY		
	matvec	error	build	matvec	error
$\varepsilon = 10^{-3}$					
16	0.34	2.38e-03	0.47	0.10	4.58e-04
32	0.88	2.37e-03	0.87	0.18	5.29e-04
64	1.44	2.06e-03	1.60	0.37	5.17e-04
128	2.34	2.53e-03	3.61	0.95	4.98e-04
$\varepsilon = 10^{-6}$					
16	1.00	9.50e-07	6.20	0.26	3.89e-07
32	2.12	1.39e-07	14.17	0.59	4.13e-07
64	5.88	1.23e-07	25.81	1.17	4.68e-07
128	7.61	1.74e-07	51.59	2.50	4.60e-07
$\varepsilon = 10^{-9}$					
16	1.64	2.71e-09	31.49	0.51	6.78e-10
32	3.79	3.29e-09	67.41	1.04	8.36e-10
64	8.94	3.49e-09	139.10	2.36	9.32e-10
128	11.89	3.15e-09	276.77	5.28	9.93e-10

Table 4
Execution time (in sec.) and average relative multiplication error of SPME and $\mathcal{H}^2$-RPY for periodic systems with different volume fractions. Parameters for SPME are listed and tuned to have multiplication errors around 10^{-4} while minimizing the computation cost. "Build" for SPME refers to the construction of the real-space term as a sparse matrix. For all tests with $\mathcal{H}^2$-RPY, a relative error threshold $\varepsilon = 10^{-4}$ is used.

Number of particles ($\times 10^4$)	SPME				$\mathcal{H}^2$-RPY		
	build	matvec	error	α, $r_{\max}$, p, n_{fft}	build	matvec	error
volume fraction 0.01							
8	0.19	0.25	2.39e-04	0.03,100,6,256	0.94	0.14	8.44e-05
32	0.47	0.48	2.47e-04	0.03,100,6,256	3.10	0.62	1.15e-04
128	2.24	2.79	1.94e-04	0.03,100,6,512	11.91	2.67	1.16e-04
512	11.95	16.06	1.77e-04	0.03,100,6,1024	40.68	8.60	1.46e-04
volume fraction 0.1							
8	0.23	0.29	5.86e-05	0.07,50,6,256	0.91	0.14	1.43e-04
32	0.52	0.49	4.96e-04	0.07,50,6,512	3.02	0.62	1.37e-04
128	2.53	2.80	7.43e-05	0.07,50,6,512	11.64	2.68	1.73e-04
512	13.28	16.15	3.23e-05	0.07,50,6,1024	39.70	8.61	1.93e-04
volume fraction 0.3							
8	0.35	0.16	7.77e-05	0.07,50,8,128	0.90	0.14	1.58e-04
32	1.34	0.82	5.99e-05	0.07,50,8,256	3.03	0.62	1.69e-04
128	5.68	2.53	3.16e-04	0.07,50,8,256	11.72	2.71	1.68e-04
512	26.29	11.95	4.20e-05	0.07,50,8,512	39.63	8.60	1.67e-04

parameters of SPME mentioned above, i.e., α, $r_{\max}$, p, and n_{fft}, are manually tuned to give minimal execution time while having relative multiplication errors around 10^{-4}. Table 4 compares the execution time and relative multiplication error of SPME and $\mathcal{H}^2$-RPY for several systems of different volume fractions. $\mathcal{H}^2$-RPY is asymptotically faster than SPME, i.e., $O(N)$ vs. $O(N\log(N))$ which makes $\mathcal{H}^2$-RPY matrix-vector multiplication faster for large problems, as observed in the table, but $\mathcal{H}^2$-RPY has a significant precomputation cost.

5.5. Brownian dynamics simulations with $\mathcal{H}^2$-RPY

In this section, we compare the use of $\mathcal{H}^2$-RPY and SPME for computing hydrodynamic interactions in BD simulations. BD simulations are performed for suspensions of particles in a periodic box. Besides hydrodynamic interactions modeled by the RPY tensor (3), the particles are also subjected to a harmonic steric repulsive potential like in our earlier studies [3,4]. We used the Ermak-McCammon BD algorithm (2) with the refinement that the RPY kernel matrix K is only updated every $\lambda_{\mathrm{RPY}} = 25$ time steps [13]. When K is updated, the Brownian displacements for λ_{RPY} time steps can be computed simultaneously using the block Lanczos method. The velocities $Kf(t)$ due to the external forces $f(t)$ at a time step are updated every time step, with matrix K not older than λ_{RPY} time steps. This choice of $\lambda_{\mathrm{RPY}} = 25$ is a conservative value that saves computational cost and does not noticeably alter the macroscopic properties of the simulation [13].

Table 5
Average timings (in sec.) per $\lambda_{\mathrm{RPY}} = 25$ steps for the precomputation ('build'), computing the Brownian displacements ('$K^{\frac{1}{2}}z$'), and computing the velocity components 'Kf'. The precomputation and Brownian displacement computation are done once every λ_{RPY} time steps, while Kf is computed λ_{RPY} times.

	N $\times 10^4$	SPME			$\mathcal{H}^2$-RPY		
		build	$K^{\frac{1}{2}}z$	Kf	build	$K^{\frac{1}{2}}z$	Kf
Monodisp.	8	0.12	12.0	0.81	0.27	9.3	2.24
	16	0.24	22.4	1.33	0.54	21.4	5.81
	32	0.51	80.9	4.57	0.95	39.7	9.37
Polydisp.	8	0.14	31.0	2.69	0.27	9.3	2.26
	16	0.25	51.7	3.53	0.54	21.3	5.89
	32	0.52	193.0	12.50	0.98	40.1	9.45

For computing the Brownian displacements with the block Lanczos method, we used a relative error threshold of 10^{-2}, following [56,28]. Both $\mathcal{H}^2$-RPY and SPME were set to obtain a relative accuracy of 10^{-3}. The BD simulations were validated by comparing the estimated translational diffusion coefficients of particles from BD simulations with $\mathcal{H}^2$-RPY and with SPME for monodisperse particle suspensions of 5000 randomly distributed particles (after suitable equilibration) with volume fractions ranging from 0.1 to 0.5, and matching these values with those reported in the literature [56,28].

To compare the performance of $\mathcal{H}^2$-RPY and SPME, we simulated large suspensions of 80,000, 160,000, and 320,000 particles with volume fraction of approximately 0.2 in a periodic box. Both monodisperse and polydisperse particle systems were used. In the latter case, particle radii were chosen from a uniform random distribution on the interval [0.9, 1.1]. Table 5 lists the average timings per $\lambda_{\mathrm{RPY}} = 25$ time steps of BD for the K matrix precomputation, for generating λ_{RPY} samples of Brownian displacement vectors, and for computing the velocities Kf at each of the λ_{RPY} time steps. The average is over 250 time steps.

First, we note that with $\lambda_{\mathrm{RPY}} = 25$, the precomputation timings are small compared to that of the other computational components. Next, consider the timings for Kf using SPME. SPME with polydisperse systems requires about twice the computation as monodisperse systems [52]. The 80,000 and 160,000 particle systems use $n_{\mathrm{fft}} = 128$, but the 320,000 particle system requires $n_{\mathrm{fft}} = 256$ to achieve the target SPME relative accuracy of 10^{-3}. This explains the nonsmooth increase in timings with problem size for SPME.

In the polydisperse case, computing Kf is faster using $\mathcal{H}^2$-RPY than with SPME for the 320,000 particle case, but slower in the other cases. $\mathcal{H}^2$-RPY is expected to be preferred for very large particle systems.

Now consider the timings for simultaneously computing λ_{RPY} Brownian displacement vectors, denoted by $K^{\frac{1}{2}}z$. Here, we observe that the timings for $\mathcal{H}^2$-RPY are usually much smaller than for SPME. The reason is because $\mathcal{H}^2$-RPY can compute matrix-vector multiplications with a block of $\lambda_{\mathrm{RPY}} > 1$ vectors much faster than λ_{RPY} times the time required for one matrix-vector multiplication. This, in turn, is mainly because temporary blocks $K(X_i, X_j)$ and $K(X_i^{\mathrm{id}}, X_j^{\mathrm{id}})$ in $\mathcal{H}^2$-RPY can be reused immediately for multiple vectors, rather than recomputed for each vector. Further, data read once from main memory into cache memory can be applied to multiple vectors rather than have to be read multiple times from main memory, once for each vector. SPME does not have similar gains when simultaneously computing with a block of vectors. Although some efficiencies can be gained in the real-space sparse matrix-vector multiplication with a block of vectors, the FFT used in the reciprocal-space portion of the SPME sum does not benefit, since FFT twiddle factors are already stored in cache memory, and only the vectors themselves need to be read from main memory.

6. Conclusion

We presented an $\mathcal{H}^2$-based fast summation algorithm for RPY kernel matrices for both periodic and nonperiodic systems of different-sized particles. The proxy surface method is the key for efficiently constructing $\mathcal{H}^2$ matrix representations of corresponding RPY interaction matrices. A brief error analysis of this compression method is provided to theoretically justify its effectiveness. The overall summation algorithm is implemented in H2Pack [49] and could be easily adopted in existing simulation codes.

The $\mathcal{H}^2$ matrix approach has lower cost for computing the summations compared to FMM and PME, as shown in the numerical experiments, but higher precomputation cost. This precomputation cost can be amortized over the several summations needed when computing Brownian displacements or forces in an iterative fashion in Brownian and Stokesian dynamics simulations with very large numbers of particles.

CRediT authorship contribution statement

Xin Xing: Conceptualization, Methodology, Software, Writing – original draft. **Hua Huang:** Methodology, Software. **Edmond Chow:** Conceptualization, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Effectiveness of the proxy surface method for the general RPY kernel

Following [51] for the 3D Laplace kernel, the effectiveness of the proxy surface method, Algorithm 1, for the general RPY kernel is justified as follows. In Algorithm 1, the two ID approximations of $K(X_*, Y_p)$ and $K(X_*, Y_*)$ have their rows written as

$$K(\boldsymbol{x}_i, Y_p) \approx u_i^T K(X_*^{\text{id}}, Y_p) \qquad \boldsymbol{x}_i \in X_*, \tag{A.1}$$

$$K(\boldsymbol{x}_i, Y_*) \approx u_i^T K(X_*^{\text{id}}, Y_*) \qquad \boldsymbol{x}_i \in X_*, \tag{A.2}$$

with u_i^T being the three rows of U associated with the ith particle. These two row approximations are exactly the evaluation of the function approximation

$$K(\boldsymbol{x}_i, \boldsymbol{y}) \approx u_i^T K(X_*^{\text{id}}, \boldsymbol{y}), \quad \boldsymbol{y} = \{y, b\} \text{ with } b \geqslant 0,\, y \in \mathcal{Y}, \tag{A.3}$$

at Y_p and Y_*, respectively. Denote the error of this function approximation as $e_i(\boldsymbol{y}) = K(\boldsymbol{x}_i, \boldsymbol{y}) - u_i^T K(X_*^{\text{id}}, \boldsymbol{y}) \in \mathbb{R}^{3\times 3}$. The proxy surface method first computes u_i^T and X_*^{id} via an algebraic ID approximation of $K(X_*, Y_p)$ which defines a function approximation (A.3) that is accurate at Y_p, i.e.,

$$\|e_i(Y_p)\|_F = \|K(\boldsymbol{x}_i, Y_p) - u_i^T K(X_*^{\text{id}}, Y_p)\|_F \leqslant \varepsilon_{\text{id}},$$

with a pre-specified error threshold ε_{id}. This defined function approximation turns out to be accurate for particles in the whole domain $\mathcal{Y}$ (to be shown next). Plugging Y_* into (A.3) thus gives a good approximation (A.1) to $K(\boldsymbol{x}_i, Y_*)$ and overall gives a good ID approximation of $K(X_*, Y_*)$.

The boundedness of $e_i(\boldsymbol{y})$ can be proved as follows. With a sufficient number of proxy particles on $\partial\mathcal{Y}$, it is reasonable to assume that, when $e_i(Y_p)$ is bounded, $e_i(\{y, 0\})$ (note that $\{y, 0\}$ denotes a particle of radius 0 centered at y) is also well bounded on $\partial\mathcal{Y}$, i.e.,

$$\max_{y \in \partial\mathcal{Y}} \|e_i(\{y,0\})\|_* \lesssim \max_{\{y,0\} \in Y_p} \|e_i(\{y,0\})\|_* \leqslant O(1)\|e_i(Y_p)\|_F, \tag{A.4}$$

where $\|\cdot\|_*$ gives the maximum absolute value of all entries in a matrix (note that $e_i(\boldsymbol{y})$ is of dimension 3×3). On the other hand, based on properties of the RPY kernel, the error function $e_i(\boldsymbol{y})$ with any $y \in \mathcal{Y}$ can actually always be bounded by its values on $\partial\mathcal{Y}$ as described by the following proposition.

Proposition 1. *For any ID components u_i^T and X_*^{id}, the function approximation defined in (A.3) at any $\boldsymbol{y} = \{y, b\}$ with $y \in \mathcal{Y}$ has its error bounded as*

$$\max_{y \in \mathcal{Y}} \|e_i(\{y,b\})\|_* \leqslant (1 + O(1)b^2) \max_{y \in \partial\mathcal{Y}} \|e_i(\{y,0\})\|_*.$$

Proof. Note that each $e_i(\boldsymbol{y})$ is a linear combination of functions in $\{K(\boldsymbol{x}_j, \boldsymbol{y})\}_{\boldsymbol{x}_j \in X_*}$. In the following, we prove a more general proposition that any linear combination of $\{K(\boldsymbol{x}_i, \boldsymbol{y})\}_{\boldsymbol{x}_i \in X_*}$, say $f(\boldsymbol{y}) = \sum_i w_i K(\boldsymbol{x}_i, \boldsymbol{y})$, always satisfies

$$\max_{y \in \mathcal{Y}} \|f(\{y,b\})\|_* \leqslant (1 + O(1)b^2) \max_{y \in \partial\mathcal{Y}} \|f(\{y,0\})\|_*, \quad b \geqslant 0.$$

Two basic formulas are used. First, if particles $\{x_0, a_0\}$ and $\{y_0, b_0\}$ are separated, i.e., $|x_0 - y_0| > a_0 + b_0$, the RPY kernel $K(\boldsymbol{x}_0, \boldsymbol{y}_0)$ is derived from the Oseen kernel $T(r) = \frac{1}{8\pi\mu}\left(\frac{1}{|r|}I + \frac{rr^T}{|r|^3}\right)$ as

$$\begin{aligned} K(\boldsymbol{x}_0, \boldsymbol{y}_0) &= \int\limits_{\partial B(y_0, b_0)} \mathrm{d}\sigma(y) \int\limits_{\partial B(x_0, a_0)} \mathrm{d}\sigma(x) T(x - y) \\ &= \int\limits_{\partial B(y_0, b_0)} \mathrm{d}\sigma(y)(1 + \frac{a_0^2}{6}\Delta)T(x_0 - y) \qquad (A.5) \\ &= (1 + \frac{a_0^2 + b_0^2}{6}\Delta)T(x_0 - y_0), \qquad (A.6) \end{aligned}$$

where $B(x,a)$ denotes a ball of radius a centered at x. Second, $f(\{y,b\})$ can be split based on (A.6) as

$$f(\{y,b\}) = f(\{y,0\}) + \frac{b^2}{6}\sum_i w_i \Delta T(x_i - y). \tag{A.7}$$

First note that each entry of $T(r)$ as a scalar function of $r \neq 0$ meets the condition for the Hopf maximum principle. For example, the $(1,1)$ diagonal entry $T_{11}(r)$ and the $(1,2)$ off-diagonal entry $T_{12}(r)$ satisfy the elliptic PDEs

$$\Delta T_{11}(r) - \frac{2}{r_1}\frac{\partial}{\partial r_1}T_{11}(r) = 0, \quad r \neq 0,$$
$$\Delta T_{12}(r) - \frac{2}{r_3}\frac{\partial}{\partial r_3}T_{12}(r) = 0, \quad r \neq 0,$$

where r_k is the kth coordinate of r. Meanwhile, $f(\{y,b\})$ depends linearly on $\{T(x_i - y)\}_{\boldsymbol{x}_i \in X_*}$ as

$$f(\{y,b\}) = \sum_i w_i K(\boldsymbol{x}_i, \{y,b\}) = \sum_i w_i (1 + \frac{a_i^2 + b^2}{6}\Delta) T(x_i - y),$$

and thus also has its entries satisfying the condition for the Hopf maximum principle when $y \in \mathcal{Y}$. Therefore, the Hopf maximum principle gives that

$$\max_{y\in\mathcal{Y}} \|f(\{y,b\})\|_* \leqslant \max_{y\in\partial\mathcal{Y}} \|f(\{y,b\})\|_*, \quad b \geqslant 0. \tag{A.8}$$

Based on the splitting of $f(\{y,b\})$ in (A.7), it remains to show that

$$\|\sum_i w_i \Delta T(x_i - y)\|_* \leqslant O(1)\|f(\{y,0\})\|_*, \quad y \in \partial\mathcal{Y}.$$

Consider a point $y_0 \in \mathcal{Y} \setminus \partial\mathcal{Y}$. Let $\varepsilon > 0$ be a scalar such that $B(y_0, 2\varepsilon)$ is within $\mathcal{Y}$. Using (A.5), $f(\{y_0, 2\varepsilon\})$ can be bounded as

$$\begin{aligned}
\|f(\{y_0,2\varepsilon\})\|_* &= \|\sum_i w_i \int_{\partial B(y_0,2\varepsilon)} \mathrm{d}\sigma(y)(1+\frac{a_i^2}{6}\Delta)T(x_i - y)\|_* \\
&= \|\int_{\partial B(y_0,2\varepsilon)} \mathrm{d}\sigma(y)\left(\sum_i w_i(1+\frac{a_i^2}{6}\Delta)T(x_i-y)\right)\|_* \\
&= \|\int_{\partial B(y_0,2\varepsilon)} \mathrm{d}\sigma(y) f(\{y,0\})\|_* \\
&\leqslant 4\pi(2\varepsilon)^2 \max_{y\in\mathcal{Y}} \|f(\{y,0\})\|_* \\
&\leqslant 4\pi(2\varepsilon)^2 \max_{y\in\partial\mathcal{Y}} \|f(\{y,0\})\|_*,
\end{aligned}$$

where the last inequality is from (A.8). Similarly, $\|f(\{y_0,\varepsilon\})\|_*$ can be bounded by $4\pi\varepsilon^2 \max_{y\in\partial\mathcal{Y}} \|f(\{y,0\})\|_*$. Combining the two estimations, we have

$$\|f(\{y_0,2\varepsilon\}) - f(\{y_0,\varepsilon\})\|_* = \|\frac{\varepsilon^2}{2}\sum_i w_i \Delta T(x_i - y_0)\|_* \leqslant 20\pi\varepsilon^2 \max_{y\in\partial\mathcal{Y}} \|f(\{y,0\})\|_*,$$

which leads to

$$\|\sum_i w_i \Delta T(x_i, y_0)\|_* \leqslant 40\pi \max_{y\in\partial\mathcal{Y}} \|f(\{y,0\})\|_*, \quad y_0 \in \mathcal{Y} \setminus \partial\mathcal{Y}. \tag{A.9}$$

Due to the continuity of $f(\{y,0\})$ and $T(x_i - y)$ in $\mathcal{Y}$, this inequality also holds on $\partial\mathcal{Y}$. Based on (A.9), (A.8), and (A.7), we have the final error bound

$$\max_{y\in\mathcal{Y}} \|f(\{y,b\})\|_* \leqslant \max_{y\in\partial\mathcal{Y}} \|f(\{y,b\})\|_* \leqslant (1 + \frac{20\pi}{3}b^2) \max_{y\in\partial\mathcal{Y}} \|f(\{y,0\})\|_*. \quad \square$$

Combining the assumption (A.4) and Proposition 1, we obtain an error estimate for the proxy surface method: each three rows of the computed ID approximation of $K(X_*, Y_*)$ are bounded as

$$\begin{aligned}
\|K(\boldsymbol{x}_i, Y_*) - u_i^T K(X_*^{\mathrm{id}}, Y_*)\|_F &\leqslant \sqrt{3|Y_*|} \max_{\boldsymbol{y}\in Y_*} \|e_i(\boldsymbol{y})\|_* \\
&\leqslant \sqrt{3|Y_*|}(1 + O(1)A_{Y_*}^2) \max_{y\in\partial\mathcal{Y}} \|e_i(\{y, 0\})\|_* \\
&\leqslant \sqrt{3|Y_*|}\, O(1 + A_{Y_*}^2)\|e_i(Y_p)\|_F \\
&\leqslant \sqrt{3|Y_*|}\, O(1 + A_{Y_*}^2)\varepsilon_{\mathrm{id}}
\end{aligned}$$

where A_{Y_*} denotes the maximum radius of all particles in Y_*. This primitive error bound shows the effectiveness of the proxy surface method.

References

[1] J.F. Brady, G. Bossis, The rheology of concentrated suspensions of spheres in simple shear flow by numerical simulation, J. Fluid Mech. 155 (1985) 105–129.
[2] T. Frembgen-Kesner, A.H. Elcock, Striking effects of hydrodynamic interactions on the simulated diffusion and folding of proteins, J. Chem. Theory Comput. 5 (2) (2009) 242–256.
[3] E. Chow, J. Skolnick, Effects of confinement on models of intracellular macromolecular dynamics, Proc. Natl. Acad. Sci. USA 112 (48) (2015) 14846–14851.
[4] E. Chow, J. Skolnick, DNA internal motion likely accelerates protein target search in a packed nucleoid, Biophys. J. 112 (11) (2017) 2261–2270.
[5] J. Skolnick, Perspective: on the importance of hydrodynamic interactions in the subcellular dynamics of macromolecules, J. Chem. Phys. 145 (10) (2016) 100901.
[6] J. Rotne, S. Prager, Variational treatment of hydrodynamic interaction in polymers, J. Chem. Phys. 50 (11) (1969) 4831–4837.
[7] H. Yamakawa, Transport properties of polymer chains in dilute solution: hydrodynamic interaction, J. Chem. Phys. 53 (1) (1970) 436–443.
[8] D.L. Ermak, J.A. McCammon, Brownian dynamics with hydrodynamic interactions, J. Chem. Phys. 69 (4) (1978) 1352–1360.
[9] M. Fixman, Construction of Langevin forces in the simulation of hydrodynamic interaction, Macromolecules 19 (4) (1986) 1204–1207.
[10] L.A. Knizhnerman, Calculation of functions of unsymmetric matrices using Arnoldi's method, USSR Comput. Math. Math. Phys. 31 (1991) 1–9.
[11] E. Gallopoulos, Y. Saad, Efficient solution of parabolic equations by polynomial approximation methods, SIAM J. Sci. Stat. Comput. 13 (1992) 1236–1264.
[12] Y. Saad, Analysis of some Krylov subspace approximations to the matrix exponential operator, SIAM J. Numer. Anal. 29 (1992) 209–228.
[13] T. Ando, E. Chow, Y. Saad, J. Skolnick, Krylov subspace methods for computing hydrodynamic interactions in Brownian dynamics simulations, J. Chem. Phys. 137 (6) (2012) 064106.
[14] E. Chow, Y. Saad, Preconditioned Krylov subspace methods for sampling multivariate Gaussian distributions, SIAM J. Sci. Comput. 36 (2) (2014) A588–A608.
[15] G. Bossis, J.F. Brady, Dynamic simulation of sheared suspensions. I. General method, J. Chem. Phys. 80 (10) (1984) 5141–5154.
[16] J.F. Brady, G. Bossis, Stokesian dynamics, Annu. Rev. Fluid Mech. 20 (1) (1988) 111–157.
[17] L. Durlofsky, J.F. Brady, G. Bossis, Dynamic simulation of hydrodynamically interacting particles, J. Fluid Mech. 180 (1987) 21–49.
[18] A.J. Banchio, J.F. Brady, Accelerated Stokesian dynamics: Brownian motion, J. Chem. Phys. 118 (22) (2003) 10323–10332.
[19] A. Sierou, J.F. Brady, Accelerated Stokesian dynamics simulations, J. Fluid Mech. 448 (2001) 115–146.
[20] M.N. Viera, Large scale simulation of Brownian suspensions, PhD thesis, University of Illinois at Urbana-Champaign, 2002.
[21] Q. Meng, J.J.L. Higdon, Large scale dynamic simulation of plate-like particle suspensions. Part I: non-Brownian simulation, J. Rheol. 52 (1) (2008) 1–36.
[22] Q. Meng, J.J.L. Higdon, Large scale dynamic simulation of plate-like particle suspensions. Part II: Brownian simulation, J. Rheol. 52 (1) (2008) 37–65.
[23] Z. Liang, Z. Gimbutas, L. Greengard, J. Huang, S. Jiang, A fast multipole method for the Rotne–Prager–Yamakawa tensor and its applications, J. Comput. Phys. 234 (2013) 133–139.
[24] W. Guan, X. Cheng, J. Huang, G. Huber, W. Li, J. McCammon, B. Zhang, RPYFMM: parallel adaptive fast multipole method for Rotne–Prager–Yamakawa tensor in biomolecular hydrodynamics simulations, Comput. Phys. Commun. 227 (2018) 99–108.
[25] S. Jiang, Z. Liang, J. Huang, A fast algorithm for Brownian dynamics simulation with hydrodynamic interactions, Math. Comput. 82 (283) (2013) 1631–1645.
[26] E.K. Guckel, Large scale simulation of particulate systems using the PME method, PhD thesis, University of Illinois at Urbana-Champaign, 1999.
[27] D. Saintillan, E. Darve, E.S.G. Shaqfeh, A smooth particle-mesh Ewald algorithm for Stokes suspension simulations: the sedimentation of fibers, Phys. Fluids 17 (3) (2005) 033301.
[28] X. Liu, E. Chow, Large-scale hydrodynamic Brownian simulations on multicore and manycore architectures, in: 2014 IEEE 28th International Parallel and Distributed Processing Symposium, IEEE, 2014, pp. 563–572.
[29] A. Saadat, B. Khomami, Matrix-free Brownian dynamics simulation technique for semidilute polymeric solutions, Phys. Rev. E 92 (2015) 033307.
[30] W. Hackbusch, S. Börm, Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices, Computing 69 (1) (2002) 1–35.
[31] W. Hackbusch, B. Khoromskij, S.A. Sauter, On $\mathcal{H}^2$-matrices, Lect. Appl. Math. (2000) 9–29.
[32] L. Greengard, V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, Acta Numer. 6 (1997) 229–269.
[33] W. Fong, E. Darve, The black-box fast multipole method, J. Comput. Phys. 228 (23) (2009) 8712–8725.
[34] Z. Gimbutas, V. Rokhlin, A generalized fast multipole method for nonoscillatory kernels, SIAM J. Sci. Comput. 24 (3) (2003) 796–817.
[35] M. Bebendorf, S. Rjasanow, Adaptive low-rank approximation of collocation matrices, Computing 70 (1) (2003) 1–24.
[36] 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.
[37] L. Ying, A kernel independent fast multipole algorithm for radial basis functions, J. Comput. Phys. 213 (2) (2006) 451–457.
[38] P.G. Martinsson, V. Rokhlin, A fast direct solver for boundary integral equations in two dimensions, J. Comput. Phys. 205 (1) (2005) 1–23.
[39] W.Y. Kong, J. Bremer, V. Rokhlin, An adaptive fast direct solver for boundary integral equations in two dimensions, Appl. Comput. Harmon. Anal. 31 (3) (2011) 346–369.
[40] 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.
[41] K. Ho, L. Greengard, A fast direct solver for structured linear systems by recursive skeletonization, SIAM J. Sci. Comput. 34 (5) (2012) A2507–A2532.
[42] X. Xing, E. Chow, Interpolative decomposition via proxy points for kernel matrices, SIAM J. Matrix Anal. Appl. 41 (2020) 221–243.

[43] 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.
[44] J. Xia, S. Chandrasekaran, M. Gu, X.S. Li, Fast algorithms for hierarchically semiseparable matrices, Numer. Linear Algebra Appl. 17 (6) (2010) 953–976.
[45] T. Ando, J. Skolnick, Crowding and hydrodynamic interactions likely dominate in vivo macromolecular motion, Proc. Natl. Acad. Sci. 107 (43) (2010) 18457–18462.
[46] M. Challacombe, C. White, M. Head-Gordon, Periodic boundary conditions and the fast multipole method, J. Chem. Phys. 107 (1997) 10131.
[47] K.N. Kudin, G.E. Scuseria, Revisiting infinite lattice sums with the periodic fast multipole method, J. Chem. Phys. 121 (7) (2004) 2886–2890.
[48] W. Yan, M. Shelley, Flexibly imposing periodicity in kernel independent FMM: a multipole-to-local operator approach, J. Comput. Phys. 355 (2018) 214–232.
[49] H. Huang, X. Xing, E. Chow, H2Pack: high-performance $\mathcal{H}^2$ matrix package for kernel matrices using the proxy point method, ACM Trans. Math. Softw. 47 (1) (2020) 3.
[50] P.J. Zuk, E. Wajnryb, K.A. Mizerski, P. Szymczak, Rotne–Prager–Yamakawa approximation for different-sized particles in application to macromolecular bead models, J. Fluid Mech. 741 (2014) R5.
[51] X. Xing, E. Chow, Error analysis of an accelerated interpolative decomposition for 3D Laplace problems, Appl. Comput. Harmon. Anal. 49 (2020) 316–327.
[52] C. Beenakker, Ewald sum of the Rotne–Prager tensor, J. Chem. Phys. 85 (3) (1986) 1581–1582.
[53] K. Luby-Phelps, Cytoarchitecture and physical properties of cytoplasm: volume, viscosity, diffusion, intracellular surface area, Int. Rev. Cytol. 192 (2000) 189–221.
[54] F. Ethridge, L. Greengard, A new fast-multipole accelerated Poisson solver in two dimensions, SIAM J. Sci. Comput. 23 (3) (2001) 741–760.
[55] H. Cheng, J. Huang, T.J. Leiterman, An adaptive fast solver for the modified Helmholtz equation in two dimensions, J. Comput. Phys. 211 (2) (2006) 616–637.
[56] T. Ando, E. Chow, J. Skolnick, Dynamic simulation of concentrated macromolecular solutions with screened long-range hydrodynamic interactions: algorithm and limitations, J. Chem. Phys. 139 (2013) 121922.