Journal of Computational Physics 231 (2012) 6827–6845

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

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

Multibody multipole methods

Dongryeol Lee [a,*], Arkadas Ozakin [b], Alexander G. Gray [a]

[a] Computational Science and Engineering, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, United States
[b] Georgia Tech Research Institute, Georgia Institute of Technology, 266 Ferst Drive, Atlanta, GA 30332, United States

ARTICLE INFO

Article history:
Received 17 May 2011
Received in revised form 11 June 2012
Accepted 18 June 2012
Available online 6 July 2012

Keywords:
Fast multipole methods
Data structures
kd-trees
Axilrod–Teller potential
Multi-tree algorithms

ABSTRACT

A three-body potential function can account for interactions among triples of particles which are uncaptured by pairwise interaction functions such as Coulombic or Lennard-Jones potentials. Likewise, a multibody potential of order n can account for interactions among n-tuples of particles uncaptured by interaction functions of lower orders. To date, the computation of multibody potential functions for a large number of particles has not been possible due to its $O(N^n)$ scaling cost. In this paper we describe a fast tree-code for efficiently approximating multibody potentials that can be factorized as products of functions of pairwise distances. For the first time, we show how to derive a Barnes–Hut type algorithm for handling interactions among more than two particles. Our algorithm uses two approximation schemes: (1) a deterministic series expansion-based method; (2) a Monte Carlo-based approximation based on the central limit theorem. Our approach guarantees a user-specified bound on the absolute or relative error in the computed potential with an asymptotic probability guarantee. We provide speedup results on a three-body dispersion potential, the Axilrod–Teller potential.

1. Introduction

In this paper, we generalize previous algorithmic frameworks for rapidly computing pair-wise summations to include higher-order summations. Suppose we are given a set of particles $X = \{x_0, \ldots, x_{N-1}\}$ in D-dimensional space.

For $x \in X$ and a n-tuple function $\phi : \underbrace{\mathbb{R}^D \times \cdots \times \mathbb{R}^D}_{n\text{copies}} \to \mathbb{R}$, we are interested in computing the following form[1]:

$$\Phi\left(x; \underbrace{X \times \cdots \times X}_{(n-1)\text{copies}}\right) = \sum_{x_{i_2} \in X \setminus \{x\}} \sum_{\substack{x_{i_3} \in X \setminus \{x\} \\ i_2 < i_3}} \cdots \sum_{\substack{x_{i_n} \in X \setminus \{x\} \\ i_{n_1} < i_n}} \phi(x, x_{i_2} \ldots, x_{i_n}) \tag{1}$$

Sums of the form Eq. (1) occur in molecular dynamics, protein structure prediction, and other similar contexts. Biomolecular simulations usually break down the interactions in complex chemical systems into balls-and-springs mechanical models augmented by torsional terms, pairwise point charge electrostatic terms, and simple pairwise dispersion (van der Waals) interactions, etc. However, such pairwise ($n = 2$) interactions often fail to capture important, complex non-additive interactions found in real systems. Though many researchers have argued that multibody potentials enable more accurate and realistic molecular modeling, the evaluation of n-body forces for $n \geqslant 3$ in systems beyond tiny sizes (less than 10,000 particles) has not been possible due to the unavailability of an efficient way to realize the computation.

* Corresponding author.
E-mail address: drselee@gmail.com (D. Lee).

[1] In computing $\Phi(x)$, we fix one of the arguments of ϕ as x and choose a $(n-1)$-subset from $X^{(n-1)}$ which does not contain x.

0021-9991/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2012.06.027

In this paper we focus on computing multibody potentials of the third order ($n = 3$), but frame our presentation so that the methods can easily be generalized to handle higher-order potentials. For concreteness, we consider the Axilrod–Teller potential (dispersion potential):

$$\phi(x_i, x_j, x_k) = \frac{1 + 3\cos\theta_i \cos\theta_j \cos\theta_k}{||x_i - x_j||^3 ||x_i - x_k||^3 ||x_j - x_k||^3} \tag{2}$$

where $\theta_i, \theta_j, \theta_k$ are the angles at the vertices of the triangle $x_i x_j x_k$ and $||\cdot||$ is the Euclidean distance metric. This potential [1] describes induced dipole interactions between triples of atoms, and is known to be important for the accurate computation of the physical properties of certain noble gases.

This Paper. For the first time, we introduce a fast algorithm for efficiently computing multibody potentials for a large number of particles. We restrict the class of multibody potentials to those that can be factorized as products of functions of pairwise Euclidean distances. That is,

$$\phi(x_{i_1}, \ldots, x_{i_n}) = \prod_{1 \leqslant p < q \leqslant n} \phi_{p,q}(x_{i_p}, x_{i_q}) = \prod_{1 \leqslant p < q \leqslant n} \phi_{p,q}(x_{i_p} - x_{i_q}) \tag{3}$$

Our algorithm achieves speedup by utilizing two approximation methods: a deterministic and a probabilistic one. The deterministic approximation is based on the analytic series-expansion-based approach in [2–4] to handle potential functions that describe n-body interactions with $n > 2$. The probabilistic approach uses a Monte Carlo-based approximation based on the central limit theorem. Our algorithm can compute multibody potentials within user-specified bounds for relative or absolute error with an asymptotic probability guarantee.

However, we would like to point out the following limitations in our algorithm. First of all, we do not present a full-fledged derivation of all three translation operators (namely the far-to-far, the far-to-local, and the local-to-local translation operators) for the general multibody case. While we define the far-field expansion for a restricted class of multibody potentials, defining the local expansion for this same class is harder (see Section 3.4). We would also like to point out that the hybrid deterministic/probabilistic approximation heuristic works under some partial distributions but not all. Indeed, there are configurations for which the speedup factor over the naive brute-force method is minimal. The Monte-Carlo based approximation relies on two theorems: (1) the central limit theorem from which we determine the number of required samples; (2) the Berry–Esseen theorem which characterizes the rate at which the sample average converges to the true average. Both theorems provide only asymptotic guarantees.

Our work utilizes and extends a framework for efficient algorithms for so-called *generalized N-body problems* [5], which introduced *multi-tree methods*. The framework was originally developed to accelerate common bottleneck statistical computations based on distances; it utilizes multiple *kd*-trees and other spatial data structures to reduce computation times both asymptotically and practically by multiple orders of magnitude. This work extends the framework with higher-order hierarchical series approximation techniques, demonstrating a fast multipole-type method for higher-order interactions for the first time, effectively creating a *multibody multipole method*.

Section 3 introduces the *generalized N-body framework* and describes a partial extension of fast multipole-type methods to handle higher-order interactions; we will discuss the technical difficulties for deriving all of the necessary tools for the general multibody case. As a result, we utilize only a simple but effective approximation using the center-of-mass approximations. Section 4 focuses on three-body interactions and introduces methods to do potential computations under both deterministic and probabilistic error criteria; the section also provides a description of the fast algorithm for the three-body case. Section 5 proves that our proposed algorithms can approximate potentials within user-specified error bounds. Section 6 shows experimental scalability results for our proposed algorithms against the naive algorithm under different error parameter settings.

Notations. Throughout this paper, we use these common sets of notations:

- (**Normal distribution**). This is denoted by $\mathcal{N}(\mu, \Sigma)$ where μ and Σ are the mean and the covariance respectively.
- (**Vector component**). For a given vector $v \in \mathbb{R}^k$, we access its dth component by $v[d]$ where $1 \leqslant d \leqslant k$ (i.e. 1-based index).
- (**Multi-index notation**). Throughout this paper, we will be using the multi-index notation. A D-dimensional multi-index $\boldsymbol{\alpha}$ is a D-tuple of non-negative integers and will be denoted using a bold lowercase Greek alphabet. For any D-dimensional multi-indices $\boldsymbol{\alpha}, \boldsymbol{\beta}$ and any $x \in \mathbb{R}^D$,

$$\begin{aligned}
&|\boldsymbol{\alpha}| = \boldsymbol{\alpha}[1] + \boldsymbol{\alpha}[2] + \cdots + \boldsymbol{\alpha}[D] \\
&\boldsymbol{\alpha}! = (\boldsymbol{\alpha}[1])!(\boldsymbol{\alpha}[2])! \cdots (\boldsymbol{\alpha}[D])! \\
&x^{\boldsymbol{\alpha}} = (x[1])^{\boldsymbol{\alpha}[2]} (x[1])^{\boldsymbol{\alpha}[2]} \cdots (x[D])^{\boldsymbol{\alpha}[D]} \\
&D^{\boldsymbol{\alpha}} = \partial_1^{\boldsymbol{\alpha}[1]} \partial_2^{\boldsymbol{\alpha}[1]} \ldots \partial_D^{\boldsymbol{\alpha}}[D] \\
&\boldsymbol{\alpha} + \boldsymbol{\beta} = (\boldsymbol{\alpha}[1] + \boldsymbol{\beta}[1], \ldots, \boldsymbol{\alpha}[D] + \boldsymbol{\beta}[D]) \\
&\boldsymbol{\alpha} - \boldsymbol{\beta} = (\boldsymbol{\alpha}[1] - \boldsymbol{\beta}[1], \ldots, \boldsymbol{\alpha}[D] - \boldsymbol{\beta}[D]) \quad \text{for} \quad \boldsymbol{\alpha} \geqslant \boldsymbol{\beta}
\end{aligned}$$

where ∂_i is a ith directional partial derivative. Define $\boldsymbol{\alpha} > \boldsymbol{\beta}$ if $\boldsymbol{\alpha}[d] > \boldsymbol{\beta}[d]$, and $\boldsymbol{\alpha} \geqslant p$ for $p \in \mathbb{Z}^+ \cup \{0\}$ if $\boldsymbol{\alpha}[d] \geqslant p$ for $1 \leqslant d \leqslant D$ (and similarly for $\boldsymbol{\alpha} \leqslant p$).

- (**Size of a point set**). Given a set S, it size is denoted by $|S|$.
- (**Probability guarantee**). We use the unbold Greek alphabet α.
- (**A tree node**). A tree node represents a subset of a point set represented by the root node. Hence, we use the same notation as the previous.
- (**Representative point of a tree node**). Usually a geometric center is used but any point inside the bounding primitive of a tree node is chosen as well. For the tree node P, this is denoted as c_P.
- (**Child nodes of an internal tree node**). Given a node N, denote its left and right child nodes by N^L and N^R, respectively.

2. Related work

2.1. Error bounds

Due to its expensive computational cost, many algorithms approximate sums at the expense of reduced precision. The following error bounding criteria are used in the literature:

Definition 2.1 (τ *absolute error bound*). For each $\Phi(x)$ for $x \in X$, it computes $\widetilde{\Phi}(x)$ such that $\left|\widetilde{\Phi}(x) - \Phi(x)\right| \leqslant \tau$.

Definition 2.2 (ϵ *relative error bound*). For each $\Phi(x)$ for $x \in X$, compute $\widetilde{\Phi}(x)$ such that $\left|\widetilde{\Phi}(x) - \Phi(x)\right| \leqslant \epsilon|\Phi(x)|$.

Bounding the relative error is much harder because the error bound criterion is in terms of the initially unknown exact quantity. As a result, many previous methods [4,6] have focused on bounding the absolute error. The relative error bound criterion is preferred to the absolute error bound criterion in statistical applications in which high accuracy is desired. Our framework can enforce the following error form:

Definition 2.3 ($(1-\alpha)$ *probabilistic* ϵ *relative/*τ *absolute error*). For each $\Phi(x)$ for $x \in X$, compute $\widetilde{\Phi}(x)$, such that with at least probability $0 < 1 - \alpha \leqslant 1$, $\left|\widetilde{\Phi}(x) - \Phi(x)\right| \leqslant \epsilon|\Phi(x)| + \tau$.

2.2. Series expansion

A series of papers first laid the foundations for efficiently computing sums of pairwise potentials such as Coulombic and Yukawa potentials [2–4]. The common approach in these papers is to derive analytical series expansions of the given potential function in either Cartesian or spherical coordinate systems. The series expansion is then truncated after taking a fixed number of terms. The associated error bounds are derived from summing the truncated terms in an appropriate infinite geometric sum or bounding the remainder term using Taylor's theorem. A recent line of work on efficient computation of pairwise function has focused on developing numerical representations of the potential matrix $[\phi(x_m, x_n)]_{m,n=1}^{N}$, rather than relying on analytical expansion of the potential function. [7,8] use singular value decomposition and the QR decomposition to compute the

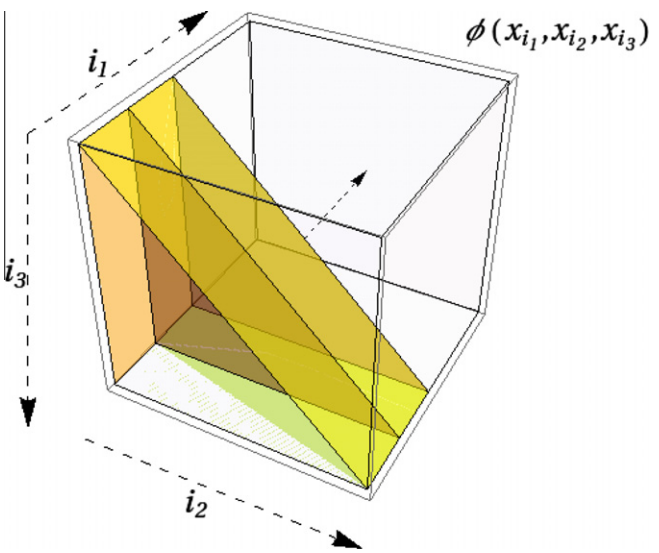

Fig. 1. An example multibody computation ($n = 3$). For each fixed argument x_{i_1}, $\Phi(x_{i_1})$ equals the summation of the entries $\phi(x_{i_1}, x_{i_2}, x_{i_3})$ in the shaded region corresponding to x_{i_1}.

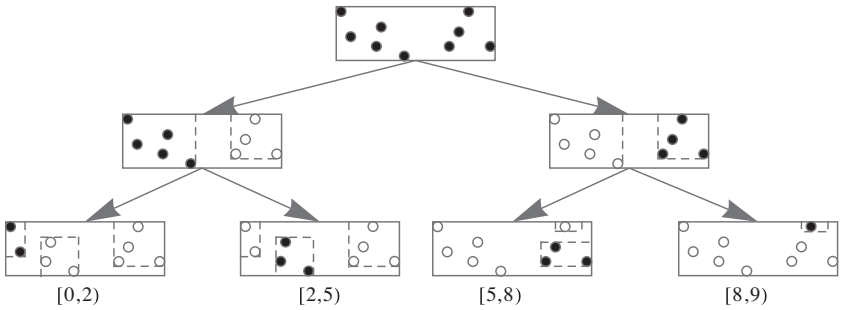

Fig. 2. *kd*-tree of a two-dimensional point set. At each level, the bounding box is split in half along the widest dimension. The solid points denote the points owned by each node. At each leaf node, we can enumerate each point with its depth-first rank. The minimum depth-first rank (inclusive) and the maximum depth-first rank (exclusive) is shown for each node.

compressed forms of the potential function and the three translation operators. [9,10] take the "pseudo-particle" approach by placing equivalent artificial charges on the bounding surface of the actual particles by solving appropriate integral equations. All of these works have been limited to pairwise potential functions, and the approach does not naturally suggest a generalization to n-body potentials with $n > 2$. To our knowledge, no research has been performed on the problem of evaluating multibody potentials using a method more sophisticated than the $O(N^n)$ brute-force algorithm with an ad hoc cut-off distance [11,12].

3. Generalized *N*-body framework

We use a variant of *kd*-trees [13] to form hierarchical groupings of points based on their locations using the recursive procedure shown in Algorithm 1. Initially, the algorithm starts with $P = X$ (the entire point set). We split a given set of points along the widest dimension of the bounding hyper-rectangle into two equal halves at the splitting coordinate. We continue splitting until the number of points is below some user-defined threshold called the *leaf threshold*. If the number of points owned by a node exceeds the leaf threshold, then it is called an *internal node*. Otherwise it is called a *leaf node*. Assuming that each split on a level results in the equal number of points on the left subset and the right subset P^L and P^R, respectively, the runtime cost is $\mathcal{O}(|X| \log |X|)$. We note that the cost of building a *kd*-tree is negligible compared to the actual multibody potential computation (see Section 6). See Fig. 2.

Algorithm 1: (BuildKdTree(P)

f$|P|$ is above the leaf threshold **then**
  Find the widest dimension d of the bounding box of P.
  Choose an axis-aligned split s along d.
  Split $P = P^L \cup P^R$ where $P^L = \{x \in P \mid x[d] \leqslant s\}$ and $P^R = P \setminus P^L$.
  BuildKdTree(P^L), BuildKdTree(P^R)
  Form far-field moments of P by translating far-field moments of P^L and P^R.
else
  Form far-field moments of P.
Initialize summary statistics of P.

The general framework for computing Eq. (1) is formalized in [5,14–16].
This approach consists of the following steps:

1. Build a spatial tree (such as *kd*-trees) for the set of particles X and build far-field moments on each node of the tree (**Bottom-up phase**).
2. Perform a *multi-tree traversal* over n-tuples of nodes (**Approximation phase**).

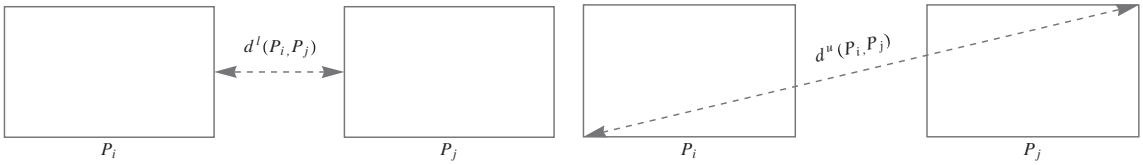

Fig. 3. The lower and upper bound on pairwise distances between the points contained in a pair of nodes.

3. Pre-order traverse the tree and propagate unincorporated bound changes downward (**Top-down phase**).

Step 2 utilizes the procedure shown in Algorithm 2 (called by setting each $P_i = X$ for $1 \leqslant i \leqslant n$), a recursive function that allows us to consider the n-tuples formed by choosing each x_i from P_i; we can gain efficiency over the naive enumeration of the n-tuples by using the bounding box and the moment information stored in each P_i. One such information is the distance bound computed using the bounding box (see Fig. 3).

Algorithm 2: MTPotentialCanonical($\{P_i\}_{i=1}^n$)

if CanSummarize($\{P_i\}_{i=1}^n$) (Try approximation.)
  Summarize($\{P_i\}_{i=1}^n, \epsilon, \tau, \alpha$)
else
  if all of S_i are leaves
    MTPotentialBase($\{P_i\}_{i=1}^n$) (Base case.)
  else
    Find an internal node P_k to split among $\{P_i\}_{i=1}^n$.
    Propagate bounds of P_k to P_k^L and P_k^R.
    MTPotentialCanonical($\{P_1, \cdots, P_{k-1}, P_k^L, P_{k+1}, \ldots, P_n\}$)
    MTPotentialCanonical($\{P_1, \cdots, P_{k-1}, P_k^R, P_{k+1}, \ldots, P_n\}$)
    Refine summary statistics based on the two recursive calls.

CanSummarize function first eliminates redundant recursive calls for the list of node tuples that satisfy the following condition: if there exists a pair of nodes P_i and P_j ($i < j$) among the node list $P_1, \cdots, P_n$, such that the maximum depth-first rank of P_i is less than the minimum depth-first rank of P_j. In this case, the function returns true. See Fig. 2 and [17]. In addition, if any one of the nodes in the list includes one of the other nodes (i.e. there exists nodes P_i and P_j such that the minimum depth-first rank of $P_i <$ the minimum depth-first rank of $P_j <$ the maximum depth-first rank of $P_j <$ the maximum depth-first rank of P_i), CanSummarize returns false. We do this because it is a bit tricky to count the number of tuples for each point in this case (see Fig. 4).

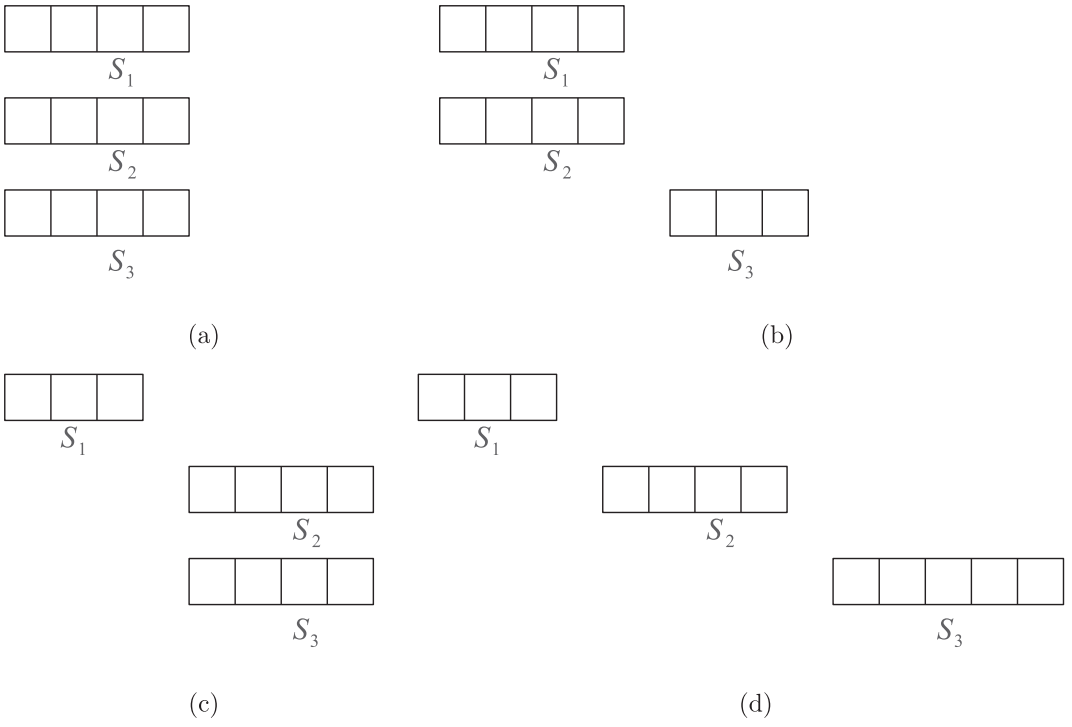

Fig. 4. For $n = 3$, four canonical cases of the three "valid" (i.e.depth-first order) node tuples encountered during the algorithm: (a) All three nodes are equal; (b) S_1 and S_2 are equal, and S_3 comes later in the depth-first order; (c) S_2 and S_3 are equal and come later in the depth-first order; (d) All three nodes are different.

Otherwise, CanSummarize function tests whether each potential sum for $x \in \bigcup_{1\leqslant m\leqslant n} P_m$ can be approximated within the error tolerance determined by the algorithm. For example, if $n = 4$, we test for each $x_1 \in P_1, x_2 \in P_2, x_3 \in P_3, x_4 \in P_4$, the following exact quantities can be approximated:

$$\Phi(x_1; P_2 \times P_3 \times P_4) = \sum_{x_{i_2} \in P_2 \setminus \{x_1\}} \sum_{\substack{x_{i_3} \in P_3 \setminus \{x_1\} x_{i_4} \\ i_2 < i_3}} \sum_{\substack{\in P_4 \setminus \{x_1\} \\ i_3 < i_4}} \phi(x_1, x_{i_2}, x_{i_3}, x_{i_4})$$

$$\Phi(x_2; P_1 \times P_3 \times P_4) = \sum_{x_{i_1} \in P_1 \setminus \{x_2\} x_{i_3}} \sum_{\substack{\in P_3 \setminus \{x_2\} x_{i_4} \\ i_1 < i_3}} \sum_{\substack{\in P_4 \setminus \{x_2\} \\ i_3 < i_4}} \phi(x_2, x_{i_1}, x_{i_3}, x_{i_4})$$

$$\Phi(x_3; P_1 \times P_2 \times P_4) = \sum_{x_{i_1} \in P_1 \setminus \{x_3\} x_{i_2}} \sum_{\substack{\in P_2 \setminus \{x_3\} x_{i_4} \\ i_1 < i_2}} \sum_{\substack{\in P_4 \setminus \{x_3\} \\ i_2 < i_4}} \phi(x_3, x_{i_1}, x_{i_2}, x_{i_4})$$

$$\Phi(x_4; P_1 \times P_2 \times P_3) = \sum_{x_{i_1} \in P_1 \setminus \{x_4\} x_{i_2}} \sum_{\substack{\in P_2 \setminus \{x_4\} x_{i_3} \\ i_1 < i_2}} \sum_{\substack{\in P_3 \setminus \{x_4\} \\ i_2 < i_3}} \phi(x_4, x_{i_1}, x_{i_2}, x_{i_3})$$

If the approximation is not possible, then the algorithm continues to consider the data at a finer granularity; it chooses an internal node P_k (typically the one with the largest diameter) to split among $\{P_i\}_{i=1}^n$. Before recursing to two sub-calls in Line 9 and Line 10 of Algorithm 2, the algorithm can optionally push quantities from a node that is being split to its child nodes (Line 8). After returning from the recursive calls, the node that was just split can refine *summary statistics* based on the results accumulated on its child nodes. The details of these operations are available in earlier papers [5,14–16,18].

The basic idea is to terminate the recursion as soon as possible, i.e. by considering a tuple of large subsets and avoiding the number of exhaustive leaf-leaf-leaf computations. We note that the CanSummarize and Summarize functions effectively replace unwieldy interaction lists used in FMM algorithms. Interaction lists in n-tuple interaction, if naively enumerated, can be large depending on the potential function ϕ and the dimensionality D of the problem, whereas the generalized N-body approach can handle a wide spectrum of problems without this drawback.

3.1. Algorithm for pairwise potentials ($n = 2$)

The general algorithmic strategy for pairwise potentials $\phi(\cdot,\cdot)$ is described in [5,14–16], and consists of the following three main phases (see Fig. 5). Suppose we are given a set of "source" points (denoted as *reference points*) and a set of "target" points (denoted as *query points*).[2]

1. **Bottom-up phase:** Compute far-field moments of order p in every leaf node of the reference tree. The resulting far-field expansion of each reference node P_2 is given by:

$$\Phi(x; P_2) = \sum_{\boldsymbol{\alpha}\geqslant 0} \left[\sum_{x_{i_2} \in P_2} \frac{(-1)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} (x_{i_2} - c_{P_2})^{\boldsymbol{\alpha}} \right] D^{\boldsymbol{\alpha}} \phi(x - c_{P_2}) = \sum_{\boldsymbol{\alpha}\geqslant 0} M_{\boldsymbol{\alpha}}(P_2, c_{P_2}) D^{\boldsymbol{\alpha}} \phi(x - c_{P_2})$$

$\Phi(x; P_2)$ reads as "the potential sum on x due to the contribution of P_2" and $M_{\boldsymbol{\alpha}}(P_2, c_{P_2})$ as "the $\boldsymbol{\alpha}$th far-field coefficient of P_2 centered at c_{P_2}." Because it is impossible to store an infinite number of far-field moments $M_{\boldsymbol{\alpha}}(P_2, c_{P_2})$, we truncate the Taylor expansion up to the order p (determined either arbitrarily or by an appropriate error criterion): (Fig. 5)

$$\widetilde{\Phi}(x; P_2; F(c_{P_2}, p)) = \sum_{|\boldsymbol{\alpha}|\leqslant p} M_{\boldsymbol{\alpha}}(P_2, c_{P_2}) D^{\boldsymbol{\alpha}} \phi(x - c_{P_2}) \tag{4}$$

such that $\left| \widetilde{\Phi}(x; P_2) - \Phi(x; P_2) \right|$ is sufficiently small. $\widetilde{\Phi}(x; P_2; F(c_{P_2}, p))$ reads as "the approximated potential sum on x due to the points owned by P_2 using up to the pth order far-field expansion of P_2 centered at c_{P_2}".

For internal reference nodes, perform the far-to-far (F2F) translation to convert the far-field moments owned by the child nodes to form the far-field moments for their common parent node P_2. For example, the far-field moments of P_2^L centered at $c_{P_2^L}$ is shifted to c_{P_2} by:

$$\widetilde{\Phi}(x; P_2^L; F(c_{P_2}, p)) = \sum_{\boldsymbol{\gamma}\leqslant p} M_{\boldsymbol{\gamma}}(P_2^L, c_{P_2}) (-1)^{\boldsymbol{\gamma}} D^{\boldsymbol{\gamma}} \phi(x - c_{P_2}) \tag{5}$$

where

$$M_{\boldsymbol{\gamma}}(P_2^L, c_{P_2}) = \sum_{\boldsymbol{\alpha}\leqslant\boldsymbol{\gamma}} \frac{M_{\boldsymbol{\alpha}}(P_2^L, c_{P_2^L}) (c_{P_2^L} - c_{P_2})^{\boldsymbol{\gamma}-\boldsymbol{\alpha}}}{(\boldsymbol{\gamma} - \boldsymbol{\alpha})!} \tag{6}$$

Note that there is no error incurred in each F2F translation, i.e. $\widetilde{\Phi}(x; P_2^L; F(c_{P_2^L}, p)) = \widetilde{\Phi}(x; P_2^L; F(c_{P_2}, p))$ for any query point y from the intersection of the domains of x for $\widetilde{\Phi}(x; P_2^L; F(c_{P_2^L}, p))$ and $\widetilde{\Phi}(x; P_2^L; F(c_{P_2}, p))$; the domain for which the far-field

[2] The terms "reference/query" have been used in our general framework.

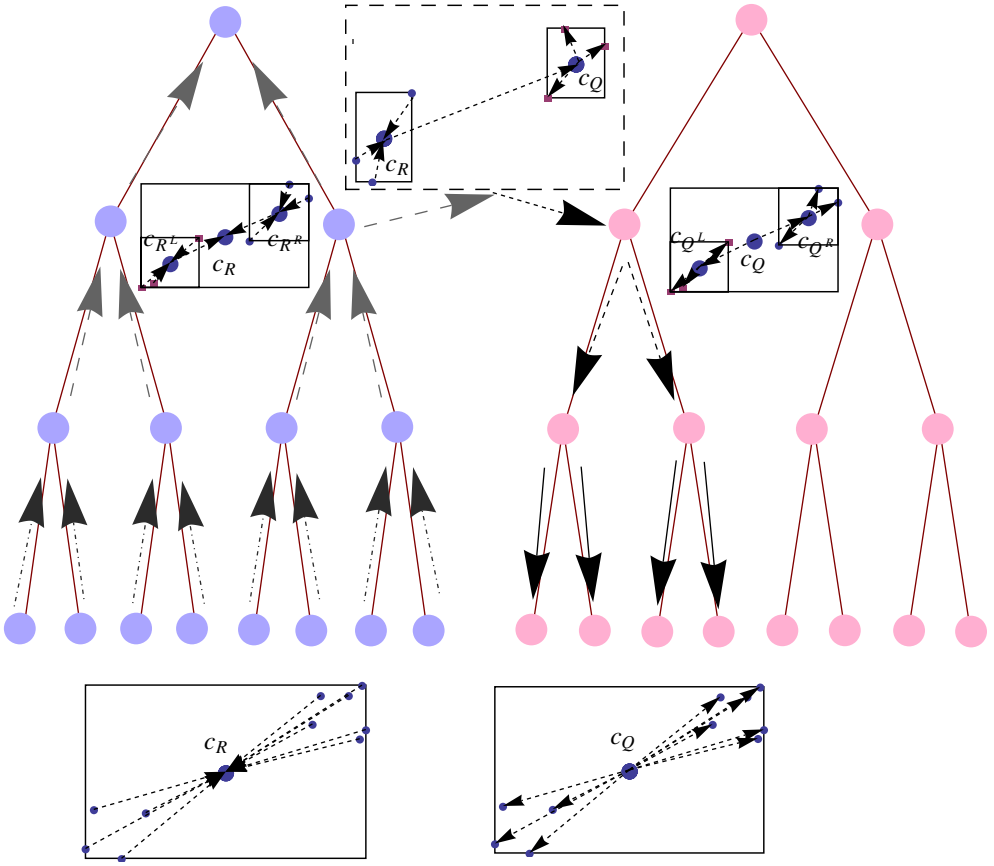

Fig. 5. The reference points (the left tree) are hierarchically compressed and uncompressed when a pair of query (from the right tree)/reference nodes is approximated within an error tolerance.

expansion remains valid depends on the error bound criterion for each potential. The far-field moments of the parent node P_2 is the sum of the translated moments of its child nodes: $M_\gamma(P_2, c_{P_2}) = \sum_{\alpha \leqslant \gamma} \frac{M_\alpha(P_2^L, c_{P_2^L})(c_{P_2^L} - c_{P_2})^{\gamma-\alpha}}{(\gamma-\alpha)!} + \frac{M_\alpha(P_2^R, c_{P_2^R})(c_{P_2^R} - c_{P_2})^{\gamma-\alpha}}{(\gamma-\alpha)!}$

2. **Approximation phase:** For a given pair of the query and the reference nodes, determine the order of approximation and either (1) translate the far-field moments of the reference node to the local moments of the query node (2) or recurse to their subsets, if the F2L translation is more costly than the direct exhaustive method.
   Let us re-write the exact contribution of P_2 to a point $x \in P_1$:

$$\begin{aligned}\Phi(x; P_2) &= \sum_{\beta \geqslant 0} \frac{1}{\beta!} \sum_{\alpha \geqslant 0} M_\alpha(P_2, c_{P_2}) D^{\alpha+\beta} \phi(c_{P_1} - c_{P_2})(x - c_{P_1})^\beta = \sum_{\beta \geqslant 0} \left[\sum_{x_{i_2} \in P_2} \frac{1}{\beta!} D^\beta \phi(c_{P_1} - x_{i_2}) \right] (x - c_{P_1})^\beta \\ &= \sum_{\beta \geqslant 0} N_\beta(P_2, c_{P_1})(x - c_{P_1})^\beta \end{aligned} \tag{7}$$

where $N_\beta(P_2, c_{P_1})$ reads as "the exact local moments[3] contributed by the points in P_2 centered at c_{P_1}." Truncating Eq. (7) at $|\beta| \leqslant p'$ for some $p' \leqslant p$ yields a direct local accumulation of order p.
From the bottom-up phase, we know that $|\alpha| \leqslant p$. Similarly, we can store only a finite number of local moments up to the order $p' \leqslant p$ and thus $|\beta| \leqslant p'$. We get the local expansion for P_1 formed due to translated far-field moments of P_2:

$$\widetilde{\Phi}(x; P_2; \widetilde{N}(c_{P_1}, p')) = \sum_{|\beta| \leqslant p'} \left[\frac{1}{\beta!} \sum_{|\alpha| \leqslant p'} M_\alpha(P_2, c_{P_2}) D^{\alpha+\beta} \phi_{1,2}(c_{P_1} - c_{P_2}) \right] (x - c_{P_1})^\beta = \sum_{|\beta| \leqslant p'} \widetilde{N}_\beta(P_2, c_{P_1})(x - c_{P_1})^\beta \tag{8}$$

where $\widetilde{N}_\beta(P_2, c_{P_1})$ reads as "approximation to the exact local moments $N_\beta(P_2, c_{P_1})$" and $\widetilde{\Phi}(x; P_2; \widetilde{N}(c_{P_1}, p'))$ as "the approximated potential sum on x due to the points in P_2 using up to the p-th order inexact local moments centered at c_{P_1}". The F2L translation is applied only if $\left| \widetilde{\Phi}(x; P_2; \widetilde{N}(c_{P_1}, p')) - \Phi(x; P_2) \right|$ is sufficiently small.

[3] We use N to denote the local moments because a "near-field" expansion is another widely used term for a local expansion. It avoids the potential notational confusion in the later parts of the paper.

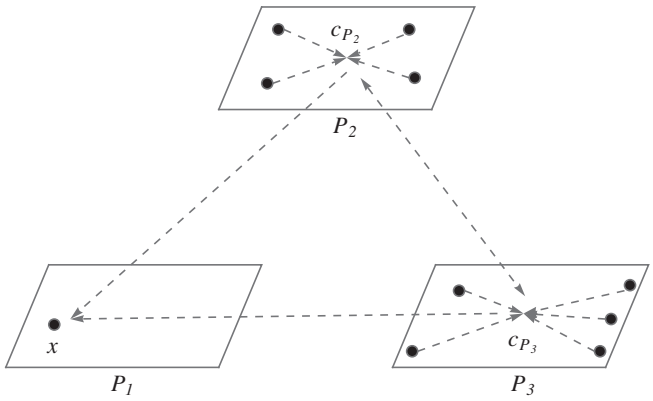

Fig. 6. A far-field expansion at x_{i_1} created by the moments of P_2 and P_3. Note the double-arrow between the nodes P_2 and P_3 corresponding to the basis functions $D^{\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}-\boldsymbol{\alpha}_{1,3}}\phi_{2,3}(P_2^c - P_3^c)$ (see Eq. 13).

3. **Top-down phase:** Propagate the local moments of each query node (i.e. pruned quantities) to its child nodes using the local-to-local (L2L) operator. Suppose we have the following local expansion for $x \in P_1$:

$$\widetilde{\Phi}(x; F2L(P_1) \cup DL(P_1); \widetilde{N}(c_{P_1}, p^u_{P_1})) = \sum_{|\boldsymbol{\alpha}| \leqslant p^u_{P_1}} \widetilde{N}_{\boldsymbol{\alpha}}(F2L(P_1) \cup DL(P_1), c_{P_1})(x_{i_1} - c_{P_1})^{\boldsymbol{\alpha}}$$

where $p^u_{P_1}$ is the maximum approximation order among (1) the F2L translations performed for P_1 and all of the ancestor nodes of P_1 (denoted by $F2L(P_1)$); and (2) the direct local accumulations of P_1 and those passed down from all of the ancestors of P_1 (denoted by $DL(P_1)$). Shifting the expansion to another center $c^*_{P_1} \in P_1$ is given by:

$$\widetilde{\Phi}(x; F2L(P_1) \cup DL(P_1); \widetilde{N}(c^*_{P_1}, p^u_{P_1})) \tag{9}$$

$$= \sum_{|\boldsymbol{\alpha}| \leqslant p^u_{P_1}} \left[\sum_{\boldsymbol{\beta} \geqslant \boldsymbol{\alpha}} \boldsymbol{\beta}\boldsymbol{\alpha} \widetilde{N}_{\boldsymbol{\beta}}(F2L(P_1) \cup DL(P_1), c_{P_1})(c^*_{P_1} - c_{P_1})^{\boldsymbol{\beta}-\boldsymbol{\alpha}} \right] (x - c^*_{P_1})^{\boldsymbol{\alpha}}$$

$$= \sum_{|\boldsymbol{\alpha}| \leqslant p^u_{P_1}} \widetilde{N}_{\boldsymbol{\alpha}}(F2L(P_1) \cup DL(P_1), c^*_{P_1})(x - c^*_{P_1})^{\boldsymbol{\alpha}} \tag{10}$$

This shifted moments are added to the local moments of each child of P_1, in effect transmitting the pruned contributions downward. At each query leaf, we evaluate the resulting local expansion at each query point.

3.2. Far-field expansion for three-body potentials ($n = 3$)

In this section, we define far-field expansions for a three-body potential that is a product of functions of pairwise distances (see Eq. (1)):

$$\phi(x_{i_1}, x_{i_2}, x_{i_3}) = \phi_{1,2}(x_{i_1}, x_{i_2}) \cdot \phi_{1,3}(x_{i_1}, x_{i_3}) \cdot \phi_{2,3}(x_{i_2}, x_{i_3}) \tag{11}$$

We define the far-field moments of a node the same way defined for the pairwise potential case. Suppose we are given three nodes $P_1 \neq P_2 \neq P_3$ from the tree. The following $(n-1)$-nested sum expresses the contribution for $x \in P_1$ due to the other nodes P_2 and P_3:

$$\Phi(x; P_2 \times P_3) = \sum_{x_{i_2} \in P_2} \sum_{x_{i_3} \in P_3} \phi(x, x_{i_2}, x_{i_3}) \tag{12}$$

The basic goal here is to decompose Eq. (12) into sums of products of the far-field moments of each node. A far-field expansion for $x_{i_1} \in P_1$ induced by the far-field moments of P_2 and P_3 is given by (see Fig. 6):

$$\Phi(x; P_2 \times P_3) = \sum_{x_{i_2} \in P_2} \sum_{x_{i_3} \in P_3} \sum_{\boldsymbol{\alpha}_{1,2} \geqslant 0} \frac{(x_{i_2} - c_{P_2})^{\boldsymbol{\alpha}_{1,2}}}{\boldsymbol{\alpha}_{1,2}!} (-1)^{\boldsymbol{\alpha}_{1,2}} D^{\boldsymbol{\alpha}_{1,2}} \phi_{1,2}(x - c_{P_2}) \sum_{\boldsymbol{\alpha}_{1,3} \geqslant 0} \frac{(x_{i_3} - c_{P_3})^{\boldsymbol{\alpha}_{1,3}}}{\boldsymbol{\alpha}_{1,3}!} (-1)^{\boldsymbol{\alpha}_{1,3}} D^{\boldsymbol{\alpha}_{1,3}} \phi_{1,3}(x$$

$$- c_{P_3}) \sum_{\boldsymbol{\alpha}_{2,3} \geqslant 0} \sum_{\boldsymbol{\beta}_{2,3} \leqslant \boldsymbol{\alpha}_{2,3}} \frac{(x_{i_2} - c_{P_2})^{\boldsymbol{\beta}_{2,3}}}{\boldsymbol{\beta}_{2,3}!} \frac{(x_{i_3} - c_{P_3})^{\boldsymbol{\alpha}_{2,3} - \boldsymbol{\beta}_{2,3}}}{(\boldsymbol{\alpha}_{2,3} - \boldsymbol{\beta}_{2,3})!} (-1)^{\boldsymbol{\alpha}_{2,3} - \boldsymbol{\beta}_{2,3}} D^{\boldsymbol{\alpha}_{2,3}} \phi_{2,3}(c_{P_2} - c_{P_3})$$

By setting $\boldsymbol{\alpha} = \boldsymbol{\alpha}_{1,2} + \boldsymbol{\alpha}_{1,3} + \boldsymbol{\alpha}_{2,3}$ and pushing the summations over $x_{i_2} \in P_2$ and $x_{i_3} \in P_3$ inside, we get:

$$\Phi(x;P_2\times P_3)=\sum_{\boldsymbol{\alpha}\geqslant 0}\sum_{\boldsymbol{\alpha}_{1,2}\leqslant\boldsymbol{\alpha}\boldsymbol{\alpha}_{1,3}}\sum_{\leqslant\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}\boldsymbol{\beta}_{2,3}\leqslant\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}-\boldsymbol{\alpha}_{1,3}}\binom{\boldsymbol{\alpha}_{1,2}+\boldsymbol{\beta}_{2,3}}{\boldsymbol{\alpha}_{1,2}}\binom{\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}-\boldsymbol{\beta}_{2,3}}{\boldsymbol{\alpha}_{1,3}}M_{\boldsymbol{\alpha}_{1,2}+\boldsymbol{\beta}_{2,3}}(P_2,c_{P_2})M_{\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}-\boldsymbol{\beta}_{2,3}}(P_3,c_{P_3})$$
$$\times(-1)^{\boldsymbol{\beta}_{2,3}}D^{\boldsymbol{\alpha}_{1,2}}\phi_{1,2}(x_{i_1}-c_{P_2})D^{\boldsymbol{\alpha}_{1,3}}\phi_{1,3}(x_{i_1}-c_{P_3})D^{\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}-\boldsymbol{\alpha}_{1,3}}\phi_{2,3}(c_{P_2}-c_{P_3}) \tag{13}$$

Truncating $\boldsymbol{\alpha}$ at pth order yields:

$$\widetilde{\Phi}(x;P_2\times P_3;F(c_{P_2}\times c_{P_3},p))=\sum_{|\boldsymbol{\alpha}|\leqslant p}\sum_{\boldsymbol{\alpha}_{1,2}\leqslant\boldsymbol{\alpha}\boldsymbol{\alpha}_{1,3}}\sum_{\leqslant\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}\boldsymbol{\beta}_{2,3}\leqslant\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}-\boldsymbol{\alpha}_{1,3}}\binom{\boldsymbol{\alpha}_{1,2}+\boldsymbol{\beta}_{2,3}}{\boldsymbol{\alpha}_{1,2}}\binom{\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}-\boldsymbol{\beta}_{2,3}}{\boldsymbol{\alpha}_{1,3}}M_{\boldsymbol{\alpha}_{1,2}+\boldsymbol{\beta}_{2,3}}(P_2,c_{P_2})M_{\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}}$$
$$-\boldsymbol{\beta}_{2,3}(P_3,c_{P_3})(-1)^{\boldsymbol{\beta}_{2,3}}D^{\boldsymbol{\alpha}_{1,2}}\phi_{1,2}(x_{i_1}-c_{P_2})D^{\boldsymbol{\alpha}_{1,3}}\phi_{1,3}(x_{i_1}-c_{P_3})D^{\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}-\boldsymbol{\alpha}_{1,3}}\phi_{2,3}(c_{P_2}-c_{P_3}) \tag{14}$$

where $\widetilde{\Phi}(x;P_2\times P_3;F(c_{P_2}\times c_{P_3},p))$ reads as "the pth order far-field expansion at x due to the moments of P_2 centered at c_{P_2} and the moments of P_3 centered at c_{P_3}."

Computational Cost of Evaluating the Far-field Expansion. The first three summations over $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}_{1,2}$, $\boldsymbol{\alpha}_{1,3}$ collectively contribute $\mathcal{O}(p^3)$ terms, and the inner summation contributing at most $\mathcal{O}(p^3)$ terms. Thus, evaluating the pth order far-field expansion for a three-body potential on a single point takes $\mathcal{O}(p^6)$ time.

3.3. Far-field expansion for general multibody potentials ($n\geqslant 2$)

For a general multibody potential that can be expressed as products of pairwise functions (see Eq. (3)), the far-field expansion induced by the points in $P_2,\ldots,P_n$ for $x\in P_1$ is:

$$\Phi(x;P_2\times\cdots\times P_n)=\prod_{2\leqslant k\leqslant n}\sum_{x_{i_k}\in P_k}\sum_{\boldsymbol{\alpha}_{1,k}\geqslant 0}\frac{(x_{i_k}-c_{P_k})^{\boldsymbol{\alpha}_{1,k}}}{\boldsymbol{\alpha}_{1,k}!}(-1)^{\boldsymbol{\alpha}_{1,k}}D^{\boldsymbol{\alpha}_{1,k}}\phi_{1,k}(x-c_{P_k})\prod_{2\leqslant s<t\leqslant n}\sum_{\boldsymbol{\alpha}_{s,t}\geqslant 0}\sum_{\boldsymbol{\beta}_{s,t}\leqslant\boldsymbol{\alpha}_{s,t}}\frac{(x_{i_s}-c_{P_s})^{\boldsymbol{\beta}_{s,t}}}{\boldsymbol{\beta}_{s,t}!}\frac{(x_{i_t}-c_{P_t})^{\boldsymbol{\alpha}_{s,t}-\boldsymbol{\beta}_{s,t}}}{(\boldsymbol{\alpha}_{s,t}-\boldsymbol{\beta}_{s,t})!}$$
$$\times(-1)^{\boldsymbol{\alpha}_{s,t}-\boldsymbol{\beta}_{s,t}}D^{\boldsymbol{\alpha}_{s,t}}\phi_{s,t}(c_{P_s}-c_{P_t})$$

Focus on grouping and multiplying monomial powers of $(x_{i_k}-c_{P_k})$ for each $2\leqslant k\leqslant n$:

$$\frac{(x_{i_k}-c_{P_k})^{\boldsymbol{\alpha}_{1,k}+\sum_{u=2}^{k-1}(\boldsymbol{\alpha}_{u,k}-\boldsymbol{\beta}_{u,k})+\sum_{v=k+1}^{n}\boldsymbol{\beta}_{k,v}}}{\boldsymbol{\alpha}_{1,k}!\prod_{u=2}^{k-1}(\boldsymbol{\alpha}_{u,k}-\boldsymbol{\beta}_{u,k})!\prod_{v=k+1}^{n}\boldsymbol{\beta}_{k,v}!}$$

Let $\xi_k=\boldsymbol{\alpha}_{1,k}+\sum_{u=2}^{k-1}(\boldsymbol{\alpha}_{u,k}-\boldsymbol{\beta}_{u,k})+\sum_{v=k+1}^{n}\boldsymbol{\beta}_{k,v}$ and $b_k=\frac{\xi_k!}{\boldsymbol{\alpha}_{1,k}!\prod_{u=2}^{k-1}(\boldsymbol{\alpha}_{u,k}-\boldsymbol{\beta}_{u,k})!\prod_{v=k+1}^{n}\boldsymbol{\beta}_{k,v}!}$. Then,

$$\Phi(x;P_2\times\cdots\times P_n)=\prod_{2\leqslant s<t\leqslant n}\prod_{2\leqslant k\leqslant n}\sum_{\boldsymbol{\alpha}_{1,k}\geqslant 0}\sum_{\boldsymbol{\alpha}_{s,t}\geqslant 0}\sum_{\boldsymbol{\beta}_{s,t}\leqslant\boldsymbol{\alpha}_{s,t}}b_k\,M_{\xi_k}(P_k,c_{P_k})(-1)^{\boldsymbol{\beta}_{s,t}}D^{\boldsymbol{\alpha}_{1,k}}\phi_{1,k}(x-c_{P_k})D^{\boldsymbol{\alpha}_{s,t}}\phi_{s,t}(c_{P_s}-c_{P_t}) \tag{15}$$

Eq. (15) is a convolution of far-field moments of $P_2,\ldots,P_n$. We can truncate the expansion above for terms for $|\boldsymbol{\alpha}|=|\sum_{1\leqslant r<s\leqslant n}\alpha_{r,s}|>p$ for some $p>0$. Note that Eq. (15) includes the $n=2$ and $n=3$ cases.

$$\widetilde{\Phi}(x;P_2\times\cdots\times P_n;F(c_{P_2}\times\cdots\times c_{P_n},p))=\prod_{2\leqslant s<t\leqslant n}\prod_{2\leqslant k\leqslant n}\sum_{|\boldsymbol{\alpha}|\leqslant p}\sum_{\boldsymbol{\alpha}_{1,k}\geqslant 0}\sum_{\boldsymbol{\alpha}_{s,t}\geqslant 0}\sum_{\boldsymbol{\beta}_{s,t}\leqslant\boldsymbol{\alpha}_{s,t}}b_kM_{\xi_k}(P_k,c_{P_k})(-1)^{\boldsymbol{\beta}_{s,t}}D^{\boldsymbol{\alpha}_{1,k}}\phi_{1,k}(x$$
$$-c_{P_k})D^{\boldsymbol{\alpha}_{s,t}}\phi_{s,t}(c_{P_s}-c_{P_t}) \tag{16}$$

Computational cost of evaluating the far-field expansion. The summations over $\boldsymbol{\alpha}_{r,s}$ for $1\leqslant r<s\leqslant n$ collectively contribute $\mathcal{O}(p^3)$ terms, and each inner summation over $\boldsymbol{\beta}_{s,t}$ contributing at most $\mathcal{O}(p^3)$ terms. Thus, evaluating the p-th order far-field expansion for a general multibody potential of the form Eq. (3) on a single point takes $\mathcal{O}\left(p^{3(n-12+1)}\right)$ time. In practice, we are forced to use $p=0$ for $n>2$ unless most $\phi_{p,q}(x_{i_p},x_{i_q})$'s in Eq. (3) are constant functions.

3.4. Local expansion for three-body potentials ($n=3$)

Unlike the far-field expansion case, we are presented a fundamental difficulty. In order to derive a local expansion, we need to express the influence of each non-evaluation point x_{i_j} on the evaluation point x at a center near x. However, breaking up the interaction among the non-evaluation points (i.e. x_{i_j}'s in the arguments of $\phi(x,x_{i_1},\cdots,x_{i_{n-1}})$) without loss of information is hard. To see this: take a three-body potential expressible in products of pairwise functions (see Fig. 7). Expanding near c_{P_1} inside the node P_1 yields an expansion valid for $x\in P_1$:

$$\Phi(x;P_2\times P_3)=\sum_{x_{i_2}\in P_2}\sum_{x_{i_3}\in P_3}\sum_{\boldsymbol{\alpha}_{1,2}\geqslant 0}\frac{D^{\boldsymbol{\alpha}_{1,2}}\phi_{1,2}(c_{P_1}-x_{i_2})}{\boldsymbol{\alpha}_{1,2}!}(x-c_{P_1})^{\boldsymbol{\alpha}_{1,2}}\sum_{\boldsymbol{\alpha}_{1,3}\geqslant 0}\frac{D^{\boldsymbol{\alpha}_{1,3}}\phi_{1,3}(c_{P_1}-x_{i_3})}{\boldsymbol{\alpha}_{1,3}!}$$
$$\times(x-c_{P_1})^{\boldsymbol{\alpha}_{1,3}}\sum_{\boldsymbol{\alpha}_{2,3}\geqslant 0}\frac{D^{\boldsymbol{\alpha}_{2,3}}\phi_{2,3}(c_{P_1}-x_{i_3})}{\boldsymbol{\alpha}_{2,3}!}(x_{i_2}-c_{P_1})^{\boldsymbol{\alpha}_{2,3}}$$

Again, let $\boldsymbol{\alpha}=\boldsymbol{\alpha}_{1,2}+\boldsymbol{\alpha}_{1,3}+\boldsymbol{\alpha}_{2,3}$. Switching the orders of summations results:

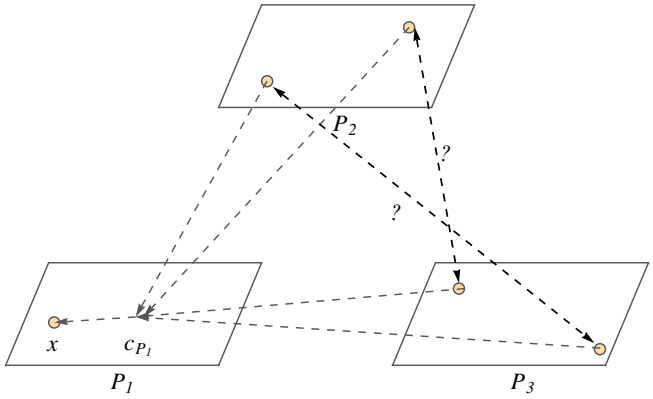

Fig. 7. A local expansion created inside the node P_1 at x by directly accumulating each point in P_2 and P_3 (see Eq. (17)). We are not aware of a technique to express an interaction between a particle in P_2 and a particle in P_3 (marked by the? symbol) for $p > 0$.

$$\Phi(x; P_2 \times P_3) = \sum_{\boldsymbol{\alpha}\geqslant 0}\sum_{\boldsymbol{\alpha}_{1,2}\leqslant\boldsymbol{\alpha}}\sum_{\boldsymbol{\alpha}_{1,3}\leqslant\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}}\left[\sum_{x_{i_2}\in P}\frac{D^{\boldsymbol{\alpha}_{1,2}}\phi_{1,2}(c_{P_1}-x_{i_2})}{\boldsymbol{\alpha}_{1,2}!}(x_{i_2}-c_{P_1})^{\boldsymbol{\alpha}_{2,3}}\right]\left[\sum_{x_{i_3}\in P_3}\frac{D^{\boldsymbol{\alpha}_{1,3}}\phi_{1,3}(c_{P_1}-x_{i_3})}{\boldsymbol{\alpha}_{1,3}!}\frac{D^{\boldsymbol{\alpha}_{2,3}}\phi_{2,3}(c_{P_1}-x_{i_3})}{\boldsymbol{\alpha}_{2,3}!}\right]$$
$$\times (x-c_{P_1})^{\boldsymbol{\alpha}_{1,2}+\boldsymbol{\alpha}_{1,3}} = \sum_{\boldsymbol{\alpha}\geqslant 0}\left[\sum_{\boldsymbol{\alpha}_{1,2}\leqslant\boldsymbol{\alpha}}\sum_{\boldsymbol{\alpha}_{1,3}\leqslant\boldsymbol{\alpha}-\boldsymbol{\alpha}_{1,2}}\overline{N}_{\boldsymbol{\alpha}}(P_2,c_{P_1})\overline{N}_{\boldsymbol{\alpha}}(P_3,c_{P_1})\right](x-c_{P_1})^{\boldsymbol{\alpha}_{1,2}+\boldsymbol{\alpha}_{1,3}} \quad (17)$$

We need the exponent of $(x-c_{P_1})$ to match α to be able to define the local moments inside P_1. Unless $\alpha_{2,3}=0$ (i.e. ignore the interaction between a particle in the second set and a particle in the third set), this is not possible. Since we encounter a similar problem in the general case, we will skip its discussion.

4. Simpler algorithm for general multibody potentials

Instead of trying to derive the full-fledged tools for general multibody potentials, we focus on deriving something simpler. Let us focus on the $n=3$ case (see Fig. 8 for the overall algorithm for a three-body potential). For a given set of three pairwise disjoint nodes: P_1, P_2, P_3 and a monotonically decreasing[4] three-body potentials such as $\phi(x_1,x_2,x_3)=\frac{1}{||x_1-x_2||^{\nu_{1,2}}||x_1-x_3||^{\nu_{1,3}}||x_2-x_3||^{\nu_{2,3}}}$,

$$\forall x_i\in P_1,\quad \widetilde{\Phi}(x_i;P_2\times P_3)=|P_2||P_3|\phi(c_{P_1},c_{P_2},c_{P_3})$$
$$\forall x_j\in P_2,\quad \widetilde{\Phi}(x_j;P_1\times P_3)=|P_1||P_3|\phi(c_{P_1},c_{P_2},c_{P_3})$$
$$\forall x_k\in P_3,\quad \widetilde{\Phi}(x_k;P_1\times P_2)=|P_1||P_2|\phi(c_{P_1},c_{P_2},c_{P_3})$$

which can be obtained by setting $p=0$ in Eq. (14). This means that we can get a cheaper approximation using the number of points owned by each node. Using the pairwise minimum and maximum node distances yields:

$$\phi(d^u(P_1,P_2),d^u(P_1,P_3),d^u(P_2,P_3))\leqslant\phi(c_{P_1},c_{P_2},c_{P_3})\leqslant\phi(d^l(P_1,P_2),d^l(P_1,P_3),d^l(P_2,P_3))$$

It is straightforward to generalize this for the $n\geqslant 2$ case.

Non-monotonic potentials: For non-monotonic potentials such as the Lennard-Jones potential $\phi(x_1,x_2)=\frac{a}{r^{12}}-\frac{b}{r^6}$, we can compute the critical points of ϕ and determine the intervals of monotonicity of ϕ and consider how ϕ behaves in the distance bound range between $d^l(P_1,P_2)$ and $d^u(P_1,P_2)$. We take a simpler approach that results in an algorithm that is easier to code; we break up the potential into two parts such that $\phi(x_1,x_2,\ldots,x_n)=\phi^+(x_1,x_2,\ldots,x_n)-\phi^-(x_1,x_2,\ldots,x_n)$, and get a lower and upper bound (though a looser bound) on the contributions from the positive potential ϕ^+ and negative potential ϕ^-.

4.1. Specifying the approximation rules

The overall algorithm which also subsumes the pairwise potential case ($n=2$) was shown in Algorithm 2. We can now specify the CanSummarize function for the general multibody case. For guaranteeing τ absolute error bound criterion (Definition 2.1), the CanSummarize function returns true if:

$$\left|\phi(d^u(P_1,P_2),\cdots,d^u(P_{n-1},P_n))-\phi(d^l(P_1,P_2),\ldots,d^l(P_{n-1},P_n))\right|\leqslant\frac{\tau}{T^{root}}$$

[4] "Monotonic" multibody potentials decrease in value if one of the Euclidean distance arguments is increased while the other two are held constant.

where $T^{root} = N - 1n - 1$ (i.e. the total number of tuples in each slice in Fig. 1). Let us also define T_i to be the number of tuples containing a fixed particle in P_i (see Fig. 4). For example, for $n = 3$, the corresponding SUMMARIZE function would accumulate for each node: for $P_1 : |P_2||P_3|\phi(c_{P_1}, c_{P_2}, c_{P_3})$, for $P_2 : |P_1||P_3|\phi(c_{P_1}, c_{P_2}, c_{P_3})$, and for $P_3 : |P_1||P_2|\phi(c_{P_1}, c_{P_2}, c_{P_3})$.

Hybrid absolute/relative error guarantee. The algorithm for guaranteeing the hybrid absolute/relative error bound (Definition 2.2) deterministically ($\alpha = 0$) is not so much different from that for guaranteeing the absolute error bound. In each node P, we maintain the lower bound on the accumulated potentials for the particles in P (denoted as $\Phi^l(P)$, a *summary statistic* stored in P). The function CANSUMMARIZE returns true if,

$$\left|\phi(d^u(P_1,P_2),\ldots,d^u(P_{n-1},P_n)) - \phi(d^l(P_1,P_2),\ldots,d^l(P_{n-1},P_n))\right| \leqslant \frac{\epsilon \min_{1\leqslant i\leqslant n}(\Phi^l(P_i) + \delta^l(P_i; P_1\times\cdots\times P_{i-1}\times P_{i+1}\times\cdots\times P_n)) + \tau}{T^{root}} \tag{18}$$

where each $\delta^l(P_i; P_1\times\ldots\times P_{i-1}\times P_{i+1}\times\ldots\times P_n) = \prod_{1\leqslant j\leqslant n j\neq i}|P_j|\phi(d^u(P_1,P_2),\ldots,d^u(P_{n-1},P_n))$ (which is computed just using the contribution of the other nodes on the ith node) is added to the currently running lower bound on each node $\Phi^l(P_i)$ to reflect the most recently available information on the lower bound. $\Phi^l(P_i)$ can be incremented and tightened as the computation progresses, either in the base case or when the recursive sub-calls in Algorithm 2 are completed (Line 11). (Fig. 8)

Monte Carlo-based approximations. The error bounds provided by the bounding boxes (see Fig. 3) assume that all pairs of points selected between the two nodes are collapsed to two positions that achieve the minimum distance (and vice versa for the maximum distance); therefore, these bounds are very pessimistic and loose. Here we introduce a method for approximating the potential sums with a probabilistic bound satisfying Definition 2.3. We can trade determinism for further gain in efficiency. We have an additional parameter α that controls the probability level at which the deviation between each approximation and its corresponding exact values holds. This was introduced first in [19,20] for probabilistic approximations of aggregate sums and later extended in [21] to handle per-particle quantities. The theorem that we rely on for probabilistic approximation is the following:

Theorem 4.1. *A widely accepted statistical rule of thumb asserts that 30 or more samples are usually enough to put a sample mean into the asymptotic regime. Berry–Esseen theorem characterizes the rate at which this convergence to normality takes place more precisely:*

Algorithm 3: CANSUMMARIZE $(\{P_i\}_{i=1}^n)$: the Monte Carlo-based approximation.

```
if ζ · m_limit ⩽ min{T_1, T_2, T_3}
  for each P_i ∈ {P_i}_{i=1}^n
    if i == 1 or P_i ≠ P_{i-1}
      for x_i ∈ P_i
        if CANSUMMARIZEMCPOINT(x_i, i, {P_i}_{i=1}^n) == false
          return false
  return true
else
  return false
```

Theorem 4.2 (*Berry–Esseen theorem*). *Let $\widetilde{\mu}$ be the sample mean of m samples drawn from the distribution F, and let μ, σ^2, and ρ be the mean, variance, and third central moment of F. Let $F_m(x)$ be the cumulative distribution function of $\widetilde{\mu}$, and $\Psi(x;\mu,\sigma^2)$ be the cdf of the Gaussian with mean μ and variance σ^2. Then there exists a positive constant C such that for all values of $\widetilde{\mu}$ and m:*

$$\left|F_m(\widetilde{\mu}) - \Psi(\widetilde{\mu};\mu,\sigma^2)\right| \leqslant \frac{C\rho}{\sigma^3\sqrt{m}}$$

which roughly says that the discrepancy between the normal distribution and the sample mean distribution goes down as $\frac{1}{\sqrt{m}}$. For three-body potentials, suppose we are given the set of three nodes, P_1, P_2, and P_3. Let us consider $x \in P_1$ (similar approximations can be made for each point in P_2 and P_3), and the contribution of P_2 and P_3 to its potential sum:

$$\Phi(x; P_2\times P_3) = \sum_{x_{i_2}\in X\setminus\{x\}}\sum_{\substack{x_{i_3}\in X\setminus\{x\}\\ i_2<i_3}}\phi(x, x_{i_2}, x_{i_3})$$

We can sample m potential values $\phi(x_{i_1}, x_{i_2}, x_{i_3})$ from the empirical distribution F formed by the 3-tuples formed among S_1, S_2, and S_3 that contain x in the list. From the m samples, we get the empirical distribution F_m^x, from which we form an approximate $\widetilde{\Phi}(x; P_2\times P_3)$:

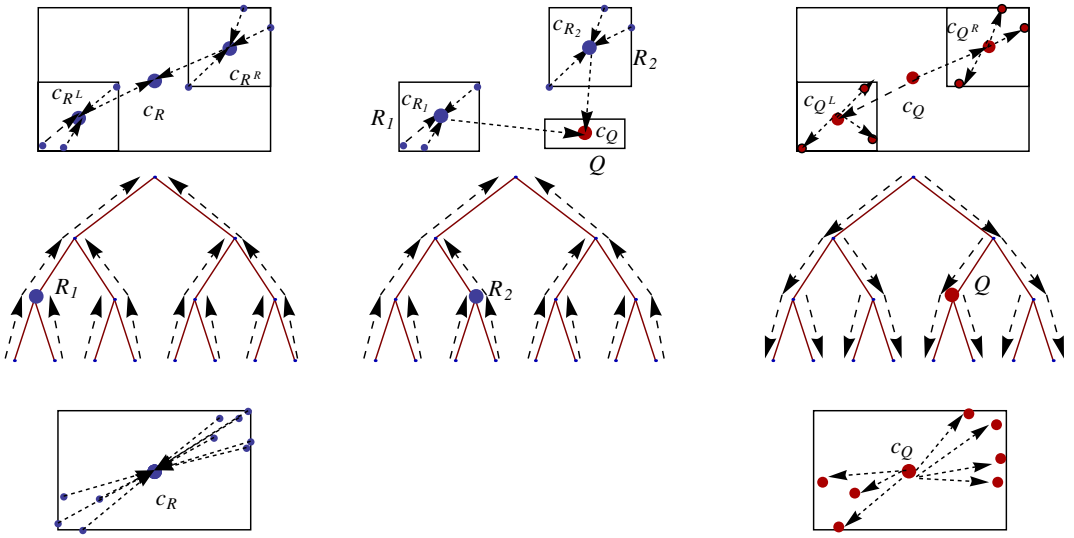

Fig. 8. Three-body multipole methods for $p = 0$ in a nutshell.

$$\widetilde{\Phi}(x; P_2 \times P_3; F_m^x) = T_1 \widetilde{\mu}_{F_m^{x_i}} = \frac{T_1}{m} \sum_{s=1}^{m} \phi(x_{i_1^s}, x_{i_2^s}, x_{i_3^s})$$

where $x_{i_1^s} = x$ for all $1 \leqslant s \leqslant m$. For sufficiently large values of m, we can assume that the discrepancy provided by the Berry–Esseen theorem is small and concentrate on the sample variance of the sample mean distribution. The sample variance of the sample mean distribution $\widetilde{\sigma}_{\mu_{F_m^{x_i}}}$ is given by:

Algorithm 4: CanSummarizeMCPoint$(x, i, \{S_i\}_{i=1}^n)$: Pruning function for the Monte Carlo based approximation per each point.

$F^x \leftarrow \emptyset$
repeat
  Get a random n-tuple $(x_1, \cdots, x_{i-1}, x, x_{i+1}, \cdots, x_n)$ where $x_j \in S_j$
  $F^x \leftarrow F^x \cup \{\phi(x_j, \cdots, x_{j-1}, x, x_{j+1}, \cdots, x_n)\}$
until $(z_{\alpha/2}\widetilde{\sigma}_{\mu_{F^x}} \leqslant \frac{\tau}{T^{root}}$ and $|F^x| \geqslant 30)$ or $|F^x| \geqslant m_{limit}$
return $z_{\alpha/2}\widetilde{\sigma}_{\mu_{F^x}} \leqslant \frac{\tau}{T^{root}}$

$$\widetilde{\sigma}_{\mu_{F_m^{x_i}}} = \frac{\widetilde{\sigma}_{F_m^{x_i}}}{\sqrt{m}} = \frac{1}{\sqrt{m}} \sqrt{\frac{1}{m-1} \sum_{s=1}^{m} (\phi(x_{i_1^s}, x_{i_2^s}, x_{i_3^s}) - \widetilde{\mu}_{F_m^{x_i}})^2}$$

where $\widetilde{\sigma}$ is the sample variance. Given m i.i.d. samples, with probability of at least $(1 - \alpha)$,

$$\left| \widetilde{\Phi}(x; P_2 \times P_3; F_m^x) - \Phi(x; P_2 \times P_3) \right| \leqslant T_1 z_{\alpha/2} \widetilde{\sigma}_{\mu_{F_m^{x_i}}}$$

where $z_{\alpha/2}$ is the number of standard deviations on either side of $\widetilde{\mu}_{F_m^{x_i}}$ to give at least $(1 - \alpha)$ coverage under the normal distribution.

Modifications to the algorithm. A Monte Carlo sampling based routine is shown in Algorithm 3. The function CanSummarize determines whether performing Monte Carlo approximations (which involves iterating over each unique point $x \in \bigcup_{i=1}^n P_i$) with at least m_{limit} samples is computationally cheaper than the brute-force computation. ζ is a global variable that dictates the desired amount of speedup needed for applying Monte Carlo approximations, rather than recursing to smaller subsets of the three nodes. If a desired speedup could be achieved, it loops for each unique point in $x \in \bigcup_{i=1}^n P_i$ and computes the sample mean of the potential values of the tuples that contain x, and the corresponding variance of the sample mean until (1) the desired error is achieved; or (2) exceeds the number of trial samples m_{limit}. Algorithm 3 is the form used for bounding the

absolute error of each potential sum error by τ with at least probability of $(1-\alpha)$. For bounding the hybrid absolute/relative error with at least probability of $(1-\alpha)$ (Definition 2.3), we replace the termination condition in the loop: $z_{\alpha/2}\widetilde{\sigma}_{\mu_{F^x}} \leqslant \frac{\tau}{T^{root}}$ with:

$$T_i \cdot z_{\alpha/2}\widetilde{\sigma}_{\mu_{F^{x_i}}} \leqslant \frac{\epsilon(\Phi^l(P_i) + T_i(\widetilde{\mu}_{F^{x_i}} - z_{\alpha/2}\widetilde{\sigma}_{\mu_{F^{x_i}}})) + \tau T_i}{T^{root}} \tag{19}$$

Algorithm 5: SUMMARIZEMC($\{S_i\}_{i=1}^n, \{T_i\}_{i=1}^n, \beta$): Monte Carlo based approximation

for each $S_i \in \{S_i\}_{i=1}^n$
if $i == 1$ or $S_i \neq S_{i-1}$
for $x_i \in S_i$
$\widetilde{\Phi}(x_i) \leftarrow \widetilde{\Phi}(x_i) + T_i \cdot \widetilde{\mu}_{F^{x_i}}, \Phi^l(x_i) \leftarrow \Phi^l(x_i) + T_i \cdot \left(\widetilde{\mu}_{F^{x_i}} - z_{\beta/2}\widetilde{\sigma}_{\mu_{F^{x_i}}}\right)$

5. Correctness of the algorithm

The correctness of our algorithm for the deterministic hybrid absolute/relative error criterion is given by:

Theorem 5.1. *Algorithm 2 with the function* CANSUMMARIZE *with the relative error bound guarantee (Eq. 18) produces approximation* $\widetilde{\Phi}(x_{i_1})$ *for* $x_{i_1} \in X$ *such that*

$$|\widetilde{\Phi}(x_{i_1}) - \Phi(x_{i_1})| \leqslant \epsilon\Phi(x_{i_1}) + \tau \tag{20}$$

Proof. (By mathematical induction) For simplicity, let us focus on $n = 3$. We induct on the number of points $|P_1 \cup P_2 \cup P_3|$ encountered during the recursion of the algorithm.

Base case: There are two parts to this part of the proof.

- Line 1 of the function MTPOTENTIALCANONICAL in Algorithm 2: any set of nodes P_1, P_2, P_3 for which the function CANSUMMARIZE returns true satisfies the error bounds for $x_{i_u} \in S_u$ for $u = 1, 2, 3$:

$$\begin{aligned}
&\forall x_{i_1} \in P_1, \left|\widetilde{\Phi}(x_{i_1}; P_2 \times P_3) - \Phi(x_{i_1}; P_2 \times P_3)\right| \\
&\leqslant \frac{T_{x_{i_u} \times P_2 \times P_3}}{T^{root}}\left(\epsilon\Phi^l(P_1) + \tau\right) \leqslant \frac{T_{x_{i_u} \times P_2 \times P_3}}{T^{root}}\left(\epsilon\Phi(x_{i_1}) + \tau\right) \\
&\forall x_{i_2} \in P_2, \left|\widetilde{\Phi}(x_{i_2}; P_1 \times P_3) - \Phi(x_{i_2}; P_1 \times P_3)\right| \\
&\leqslant \frac{T_{x_{i_u} \times P_2 \times P_3}}{T^{root}}\left(\epsilon\Phi^l(P_2) + \tau\right) \leqslant \frac{T_{x_{i_u} \times P_2 \times P_3}}{T^{root}}\left(\epsilon\Phi(x_{i_2}) + \tau\right) \\
&\forall x_{i_3} \in P_3, \left|\widetilde{\Phi}(x_{i_3}; P_1 \times P_2) - \Phi(x_{i_3}; P_1 \times P_2)\right| \\
&\leqslant \frac{T_{x_{i_u} \times P_2 \times P_3}}{T^{root}}\left(\epsilon\Phi^l(P_3) + \tau\right) \leqslant \frac{T_{x_{i_u} \times P_2 \times P_3}}{T^{root}}\left(\epsilon\Phi(x_{i_3}) + \tau\right)
\end{aligned} \tag{21}$$

where $T_{x_{i_u} \times P_2 \times P_3}$ denotes the number of tuples chosen by fixing x_{i_u} and selecting the other two from P_2 and P_3 and so on.
- The function call MTPOTENTIALBASE in Algorithm 2: each $x_{i_1} \in P_1$ and $x_{i_2} \in P_2$ and $x_{i_3} \in P_3$ exchange contributions exactly and incur no approximation error.

Inductive step: Suppose we are given the set of three nodes P_1, P_2, and P_3 (at least one of which is an internal node) in the function MTPOTENTIALCsc anonical. Suppose the three tuples P_1, P_2, P_3 could not be pruned, and that we need to recurse on each child of P_1, P_2, and P_3.

By assumption, CANSUMMARIZE returns false if any one of the nodes P_1, P_2, P_3 includes one of the other nodes (see Section 3). For $n = 3$, we can assume that the possible node tuple cases that could be considered for pruning are shown in Fig. 4. Let $\{\{P_s^k\}_{s=1}^3\}_{k=1}^t$ be the set of set of three nodes considered during the recursive sub-computations using the child nodes of each P_1, P_2, and P_3; note that the maximum value of t is 8 for three-body interactions. Note that for each k, P_s^k is either (1) the node

P_s itself (2) the left child node of P_s (3) the right child node of P_s. Therefore, for each $k = 1, 2, \cdots, t$, $|P_1^k \cup P_2^k \cup P_3^k| \leqslant |P_1 \cup P_2 \cup P_3|$. The equality holds when all of P_1, P_2, and P_3 are leaf nodes for which the error criterion is satisfied by the base case function (no error incurred).

If any one of P_1, P_2, and P_3 is an internal node, then we are guaranteed that $|P_1^k \cup P_2^k \cup P_3^k| < |P_1 \cup P_2 \cup P_3|$ for all $k = 1, \cdots, t$. We invoke the inductive hypothesis to conclude that for each k and for each $x_{i_u} \in P_u^k$ for $u = 1, 2, 3$:

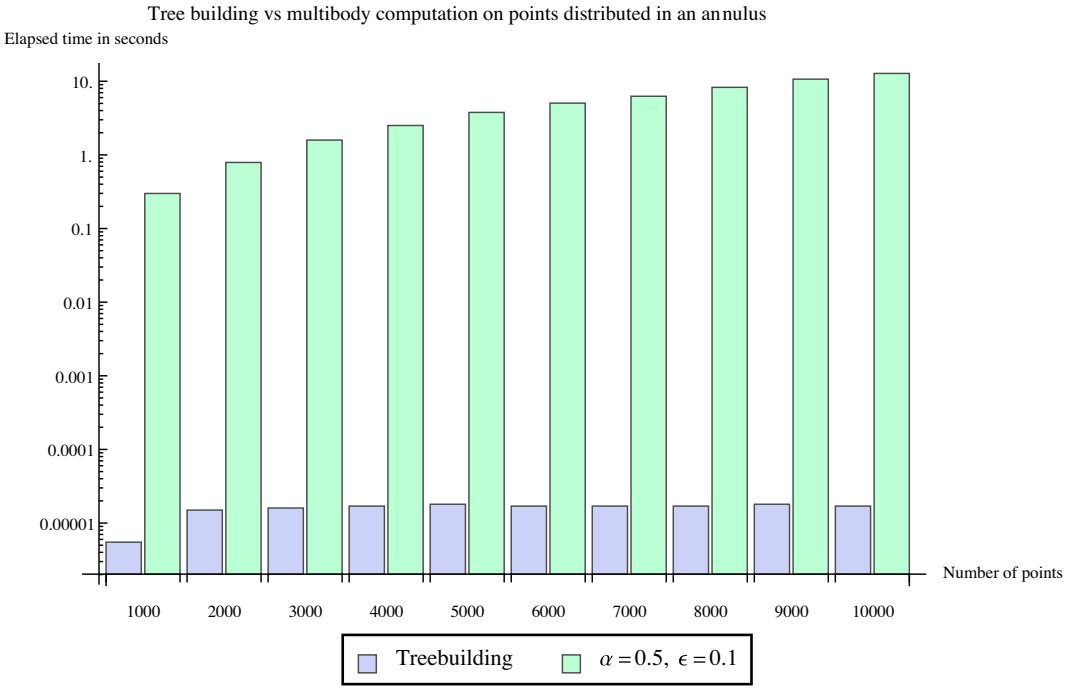

Fig. 9. Building the *kd*-tree takes negligible amount of time compared to the time it takes for the actual multibody computation.

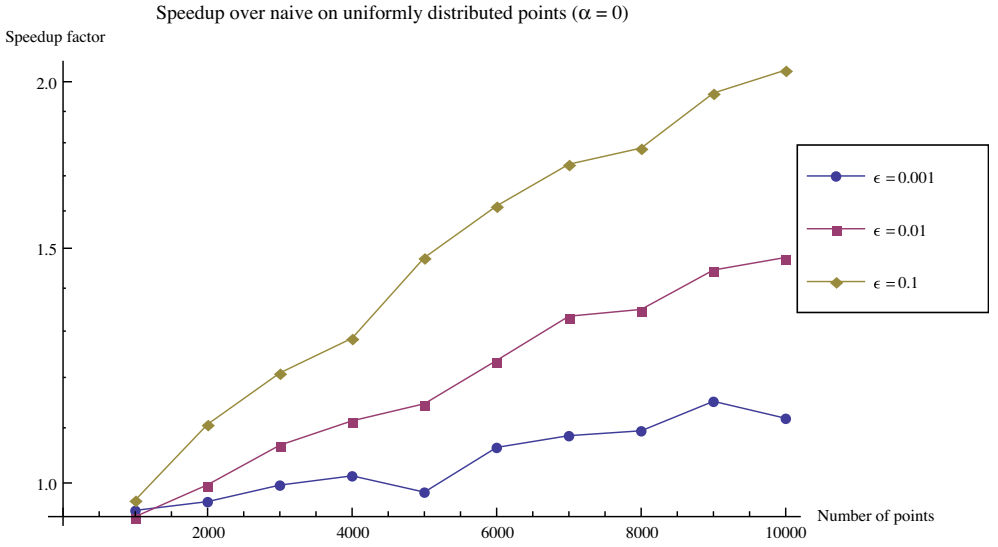

Fig. 10. Speedup result on uniformly distributed points using the deterministic algorithm ($\alpha = 0$). The base timings for the naive algorithm on each point set are: 1.91×10^1 s, 1.54×10^2 s, 5.17×10^2 s, 1.23×10^3 s, 2.39×10^3 s, 4.16×10^3 s, 6.64×10^3 s, 9.76×10^3 s, 1.43×10^4 s, and 1.92×10^4 s.

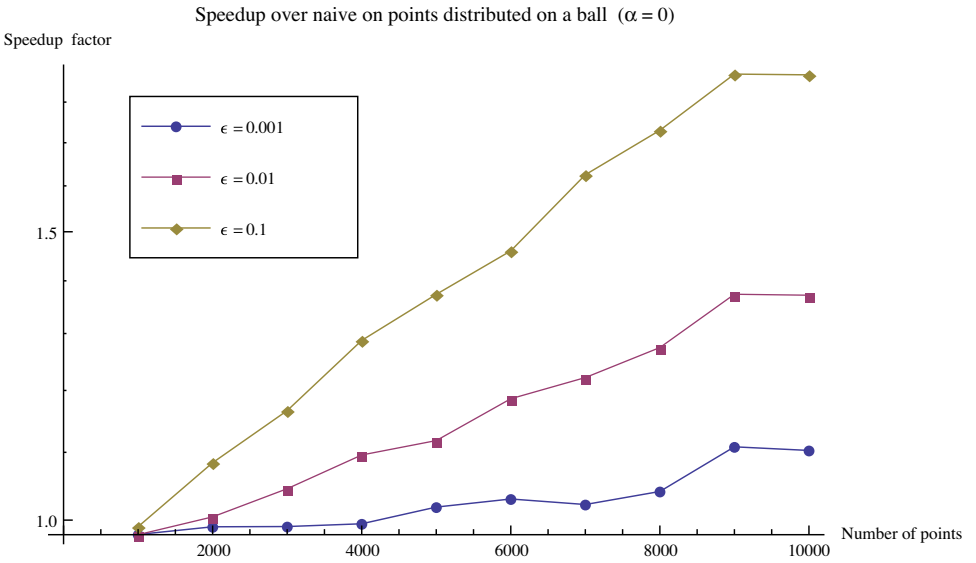

Fig. 11. Speedup result on points distributed inside a sphere using the deterministic algorithm ($\alpha = 0$). The base timings for the naive algorithms are listed in Fig. 10.

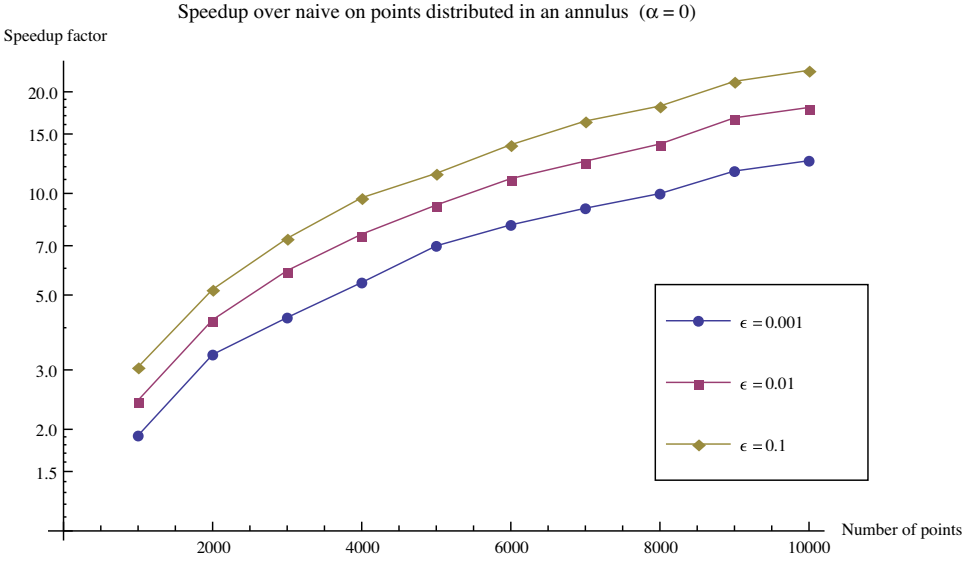

Fig. 12. Speedup result on points distributed on an annulus using the deterministic algorithm ($\alpha = 0$). The base timings for the naive algorithms are listed in Fig. 10.

$$\forall x_{i_1} \in P_1^k, \left|\widetilde{\Phi}(x_{i_1}; P_2^k \times P_3^k) - \Phi(x_{i_1}; P_2^k \times P_3^k)\right| \leqslant \frac{T_{x_{i_1} \times P_2^k \times P_3^k}}{T^{root}} \left(\epsilon \Phi^l(P_1^k) + \tau\right) \leqslant \frac{T_{x_{i_1} \times P_2^k \times P_3^k}}{T^{root}} \left(\epsilon \Phi(x_{i_1}) + \tau\right)$$

$$\forall x_{i_2} \in P_2^k, \left|\widetilde{\Phi}(x_{i_2}; P_1^k \times P_3^k) - \Phi(x_{i_2}; P_1^k \times P_3^k)\right| \leqslant \frac{T_{x_{i_2} \times P_1^k \times P_3^k}}{T^{root}} \left(\epsilon Phi^l(P_2^k) + \tau\right) \leqslant \frac{T_{x_{i_2} \times P_1^k \times P_3^k}}{T^{root}} \left(\epsilon \Phi(x_{i_2}) + \tau\right)$$

$$\forall x_{i_3} \in P_3^k, \left|\widetilde{\Phi}(x_{i_3}; P_1^k \times P_2^k) - \Phi(x_{i_3}; P_1^k \times P_2^k)\right| \leqslant \frac{T_{x_{i_3} \times P_1^k \times P_2^k}}{T^{root}} \left(\epsilon \Phi^l(P_3^k) + \tau\right) \leqslant \frac{T_{x_{i_3} \times P_1^k \times P_2^k}}{T^{root}} \left(\epsilon \Phi(x_{i_3}) + \tau\right)$$

where T_s^k is the number of 3-tuples formed among P_1^k, P_2^k, P_3^k that contain a fixed point in P_s^k. By the triangle inequality, Eq. 20 holds by extending to $P_1 = P_2 = P_3 = X$ since the number of encountered tuples for each particle add up to T^{root}. □

We are now ready to prove the correctness of our algorithm for bounding the relative error probabilistically.

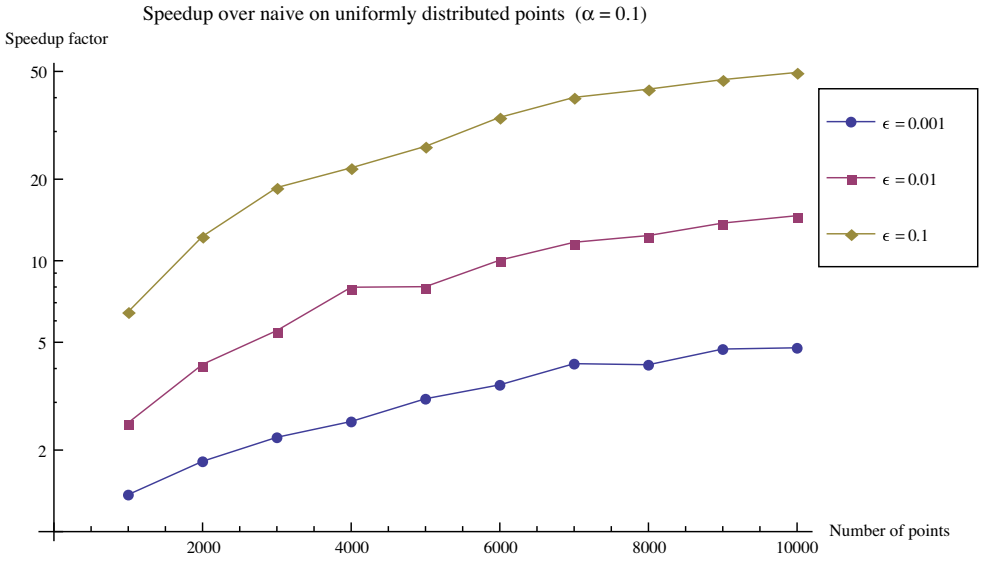

Fig. 13. Speedup result on uniformly distributed points using the Monte Carlo-based algorithm ($\alpha = 0.1$). The base timings for the naive algorithms are listed in Fig. 13.

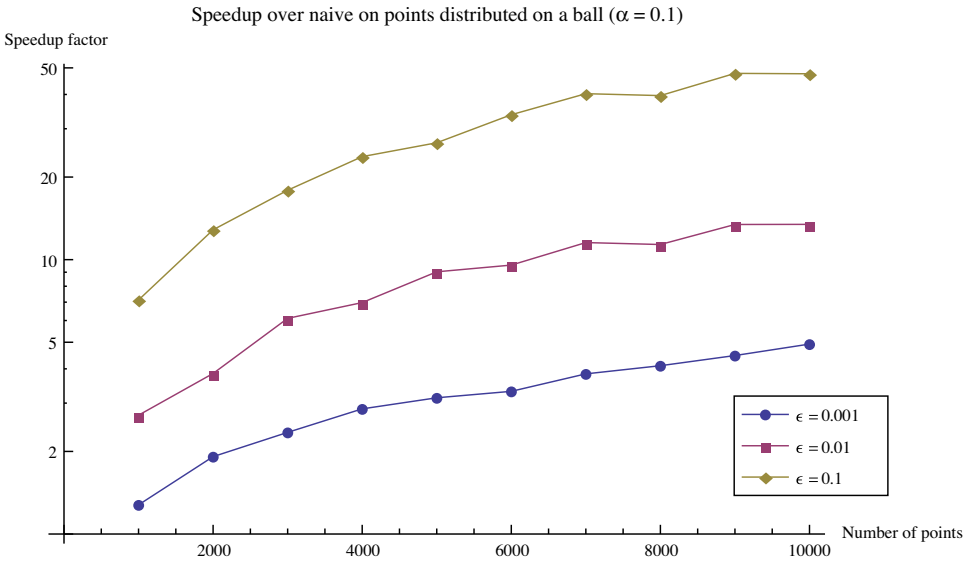

Fig. 14. Speedup result on points distributed inside a sphere using the Monte Carlo-based algorithm ($\alpha = 0.1$). The base timings for the naive algorithms are listed in Fig. 10.

Table 1
The distribution of relative error on the uniform distribution using $\alpha = 0.1$ and $\epsilon = 0.001$.

Number of points	% Achieving	Average relative error	Variance	Maximum relative error
1000	98.3	1.11×10^{-4}	8.89×10^{-7}	2.86×10^{-2}
2000	97.9	1.28×10^{-4}	7.71×10^{-7}	2.78×10^{-2}
3000	98.6	1.51×10^{-4}	2.64×10^{-6}	6.47×10^{-2}
4000	98.3	1.44×10^{-4}	3.37×10^{-6}	1.01×10^{-1}
5000	98.7	2.65×10^{-4}	1.09×10^{-4}	7.36×10^{-1}
6000	98.3	1.29×10^{-4}	1.39×10^{-6}	3.62×10^{-2}
7000	98.4	1.86×10^{-4}	9.29×10^{-6}	1.96×10^{-1}
8000	98.8	9.89×10^{-5}	1.21×10^{-6}	6.50×10^{-2}
9000	98.8	9.94×10^{-5}	1.39×10^{-6}	6.69×10^{-2}
10,000	98.9	1.02×10^{-4}	1.95×10^{-6}	1.06×10^{-1}

Table 2
The distribution of relative error on the ball distribution using $\alpha = 0.1$ and $\epsilon = 0.001$.

Number of points	% Achieving	Average relative error	Variance	Maximum relative error
1000	98.6	8.21×10^{-5}	1.17×10^{-7}	7.22×10^{-3}
2000	98.7	1.35×10^{-4}	1.26×10^{-6}	2.78×10^{-2}
3000	98.7	1.11×10^{-4}	7.58×10^{-7}	3.23×10^{-2}
4000	97.0	1.36×10^{-3}	1.21×10^{-3}	1.81×10^{0}
5000	98.2	1.19×10^{-4}	1.18×10^{-6}	4.85×10^{-2}
6000	98.9	1.20×10^{-4}	3.70×10^{-6}	1.27×10^{-1}
7000	98.8	1.22×10^{-4}	3.32×10^{-6}	1.11×10^{-1}
8000	98.5	1.31×10^{-4}	3.67×10^{-6}	1.12×10^{-1}
9000	97.9	6.24×10^{-4}	3.89×10^{-4}	1.14×10^{0}
10,000	97.6	5.09×10^{-4}	2.40×10^{-4}	1.28×10^{0}

Table 3
The distribution of relative error on the annulus distribution using $\alpha = 0.1$ and $\epsilon = 0.001$.

Number of points	% Achieving	Average relative error	Variance	Maximum relative error
1000	98.4	9.33×10^{-5}	3.42×10^{-7}	1.38×10^{-2}
2000	97.2	9.21×10^{-4}	2.69×10^{-4}	5.15×10^{-1}
3000	98.7	8.52×10^{-5}	1.16×10^{-6}	5.09×10^{-2}
4000	91.8	2.53×10^{-2}	6.10×10^{-1}	4.80×10^{1}
5000	96.9	1.28×10^{-3}	1.09×10^{-3}	1.27×10^{0}
6000	92.8	6.28×10^{-3}	1.38×10^{-2}	6.43×10^{0}
7000	95.2	2.13×10^{-3}	1.36×10^{-3}	6.66×10^{-4}
8000	91.2	1.45×10^{-2}	3.77×10^{-1}	5.36×10^{1}
9000	94.6	5.17×10^{-3}	6.56×10^{-3}	3.94×10^{0}
10,000	91.6	2.72×10^{-2}	8.06×10^{-1}	8.29×10^{1}

Theorem 5.2. *Algorithm 2 with the function* CanSummarize *with the modification described in Eq. 19 produces approximations* $\widetilde{\Phi}(x_i)$ *for* $x_i \in X$ *such that*

$$|\widetilde{\Phi}(x_i) - \Phi(x_i)| \leqslant \epsilon\Phi(x_i) + \tau \tag{22}$$

with the probability of at least $1 - \alpha$ *for* $0 < \alpha < 1$, *as the number of samples in the Monte Carlo approximation tends to infinity.*

Proof. We extend the proof in Theorem 5.1. For simplicity, we again focus on the $n = 3$ case.

Base case: Given the set of three nodes with the desired failure probability α, the base case MTPotentialBase is easily shown to satisfy Eq. 21 with 100 % probability ($> 1 - \alpha$). Similarly, each Monte Carlo prune satisfies Eq. 21 with probability of $1 - \alpha$ asymptotically.

Inductive case: For a non-prunable set of three nodes $\{P_k\}_{k=1}^{3}$ for the required failure probability β. Note that MTPotentialCanonical results in a maximum of four (i.e. $2^{3-1} = 4$) sub-calls for a set of non-prunable P_1, P_2, P_3 nodes. For example, suppose P_1 is an internal node, and consider its left child, P_1^L. The contribution of P_2 and P_3 on P_1^L can be computed by considering the node combinations: $(P_1^L, P_2^L, P_3^L), (P_1^L, P_2^L, P_3^R)$, $(P_1^L, P_2^R, P_3^L), (P_1^L, P_2^R, P_3^R)$, resulting in a maximum of four combinations if P_1, P_2, P_3 satisfy the case 4a in Fig. 4. Each recursive sub-call is equivalent to a stratum in a stratified sampling, and satisfies the following:

$$\left|\widetilde{\Phi}(x_{i_u}; P_2^L \times P_3^L) - \Phi(x_{i_u}; P_2^L \times P_3^L)\right| \leqslant \frac{\epsilon T_{x_{i_u} \times P_2^L \times P_3^L}}{T^{root}} \Phi^l(P_1^L) + \frac{\tau T_{x_{i_u} \times P_2^L \times P_3^L}}{T^{root}}$$

$$\left|\widetilde{\Phi}(x_{i_u}; P_2^L \times P_3^R) - \Phi(x_{i_u}; P_2^L \times P_3^R)\right| \leqslant \frac{\epsilon T_{x_{i_u} \times P_2^L \times P_3^R}}{T^{root}} \Phi^l(P_1^L) + \frac{\tau T_{x_{i_u} \times P_2^R \times P_3^L}}{T^{root}}$$

$$\left|\widetilde{\Phi}(x_{i_u}; P_2^R \times P_3^L) - \Phi(x_{i_u}; P_2^R \times P_3^L)\right| \leqslant \frac{\epsilon T_{x_{i_u} \times P_2^R \times P_3^L}}{T^{root}} \Phi^l(P_1^L) + \frac{\tau T_{x_{i_u} \times P_2^R \times P_3^R}}{T^{root}}$$

$$\left|\widetilde{\Phi}(x_{i_u}; P_2^R \times P_3^R) - \Phi(x_{i_u}; P_2^R \times P_3^R)\right| \leqslant \frac{\epsilon T_{x_{i_u} \times P_2^R \times P_3^R}}{T^{root}} \Phi^l(P_1^L) + \frac{\tau T_{x_{i_u} \times P_2^R \times P_3^R}}{T^{root}}$$

Collectively, the results from these strata add up to potential estimates that satisfy the error bound with at least $1 - \alpha$ probability for each $x_{i_u} \in P_1^L$ and the following holds:

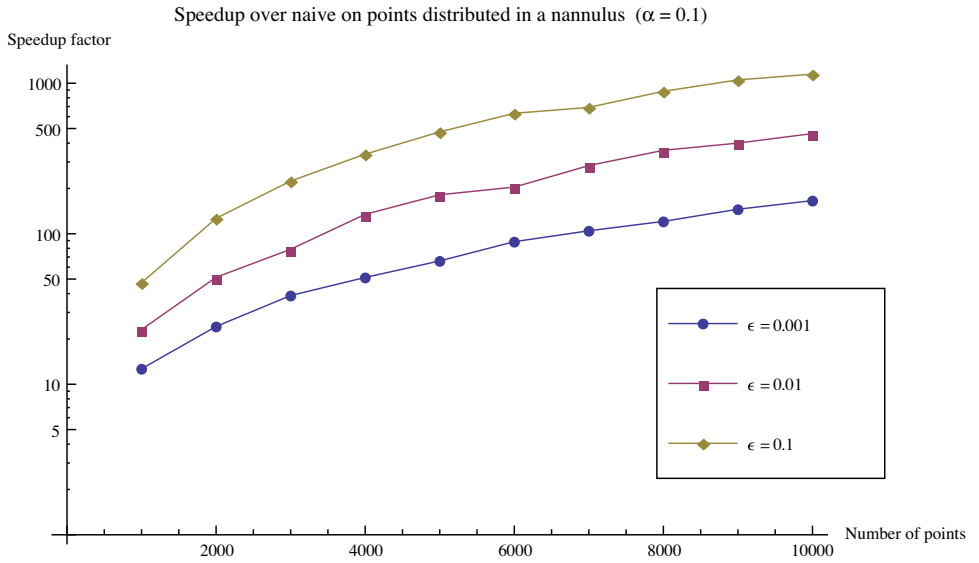

Fig. 15. Speedup result on points distributed on an annulus using the Monte Carlo-based algorithm ($\alpha = 0.1$). The base timings for the naive algorithms are listed in Fig. 13.

$$\left|\widetilde{\Phi}(x_{i_u}; P_2 \times P_3) - \Phi(x_{i_u}; P_2 \times P_3)\right| \leqslant \frac{\epsilon T_{x_{i_u} \times P_2 \times P_3}}{T^{root}} \Phi^l(x_{i_u}) + \frac{\tau T_{x_{i_u} \times P_2 \times P_3}}{T^{root}}$$

where $T_{x_{i_u} \times P_2 \times P_3} = T_{x_{i_u} \times P_2^L \times P_3^L} + T_{x_{i_u} \times P_2^L \times P_3^R} + T_{x_{i_u} \times P_2^R \times P_3^L} + T_{x_{i_u} \times P_2^R \times P_3^R}$. The similar bounds hold for each $x \in P_1^R$, and the same reasoning can be extended to the bounds for P_2 and P_3. Because $\Phi^l(P_1) = \min\{\Phi^l(P_1^L), \Phi^l(P_1^R)\}$ throughout the execution of the algorithm, we can extend the argument to the case where $P_1 = P_2 = P_3 = X$.

6. Experiment results

All of our algorithms were based on an open-source C++ library called MLPACK [22,23]. The experiments were performed on a desktop with AMD Phenom II X6 1100T Processors utilizing only one core with 8 GB of RAM.

6.1. Tree building

The cost of tree-building is negligible compared to the actual multibody computation. Compared to complex, irregular memory access patterns encountered in the multibody computation (as do most recursive algorithms in general), the tree-building phase requires mostly sequential scanning of contiguous blocks of memory and thus requires shorter amount of time. See Fig. 9, where the tree building is compared to the multibody computation with the relative error criterion $\epsilon = 0.1$ and the 50 % probability guarantee ($\alpha = 0.5$). The annulus distribution was chosen deliberately to show that even under the distribution for which the multibody computation is relatively fast (see Section 6.2), the tree building requires a tiny fraction of time compared to the computation time.

6.2. Multibody computation

We demonstrate speedup results of our approximate algorithms guaranteeing the $(1 - \alpha)$ probabilistic ϵ relative error criterion (Definition 2.3). For this paper, we focus strictly on the relative error criterion ($\tau = 0$) and test on three relative error parameter values ($\epsilon = 0.001, \epsilon = 0.01$, and $\epsilon = 0.1$). We test on three different types of distribution: uniform within the unit hypercube $[0, 1]^3$ (denoted as the "uniform" distribution), the annulus distribution (denoted as the "annulus" distribution) in three dimensions, and uniform within the unit three-dimensional sphere (denoted as the "ball" distribution). These three distributions were also used in [24]). For the deterministic and probabilistic algorithms, the order of local expansion is fixed at $p = 0$ and only 0th order multipole expansions are used for the results.

Deterministic approximations. Figs. 10, 11 and 12 show speedup results against the naive algorithm using only the deterministic approximation (i.e. $\alpha = 0$). On the uniform distribution and the ball distribution, the speedup is almost non-existent; the speedup factor is a little bit more than two on the dataset containing 10,000 points using the lowest parameter setting of $\epsilon = 0.1$. On the annulus distribution, our deterministic algorithm achieves a little bit better speedup against the

naive algorithm; a factor of more than 20 times speedup on $10,000$ points is encountered on $\epsilon = 0.1$. A tree-based hierarchical method generally works better for clustered point sets, and this is reflected in our results.

Monte-Carlo approximations. In this section, we show whether adding indeterminism by sampling can reduce the computation time while guaranteeing a slightly relaxed error criterion (but with a high probability guarantee for each potential sum). We first relax the probability guarantee to be 90% (i.e. $\alpha = 0.1$). Like the results shown using the deterministic algorithm, our Monte Carlo-based algorithm achieves the most speedup on points distributed in an annulus (1000 times speedup on $10,000$ points using $\epsilon = 0.1$). See Figs. 13–15.

We also list the percentage of the points actually achieving the ϵ relative error bound along with the mean and the variance in Tables 1–3. The relative error level of 0.001 and the probability guarantee of 90% was used. Under all three distributions, the percentage of points whose potential sum achieved the desired relative error of 0.001 was well above 90%. We list the average relative error, the variance, and the maximum relative error. Note that the maximum relative error can exceed 100% if the true potential sum and its approximation have opposite signs. For a particle with a small potential sum, we have observed that this is indeed the case due to numerical inaccuracies accumulated during the summation (Fig. 15).

7. Conclusion

In this paper, we have introduced the framework for extending the pairwise series expansion to potentials that involve more than two points. Through this process, we have formally defined an analogue to the far-field expansion for approximating the multibody potentials in a hierarchical fashion as done in traditional FMM algorithms and have derived algorithms for guaranteeing (1) absolute error bound (2) relative error bound (3) probabilistic absolute/relative error on each particle potential sum and proved the correctness of our algorithms formally. However, we do not present a full-fledged derivation of all three translation operators and the analogue to the local expansion due to a technical difficulty. Instead, we propose to use only a monopole approximation ($p = 0$) in a simpler alternative algorithm. Our experiment demonstrates that the algorithm using the hybrid deterministic/probabilistic approximation heuristic achieves speedup under points lying on an annulus of a sphere (i.e. lower-dimensional manifold). For our future work, we are working on parallelization as done in [25,26].

References

[1] B. Axilrod, E. Teller, Interaction of the van der Waals type between three atoms, The Journal of Chemical Physics 11 (1943) 299.
[2] H. Cheng, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm in three dimensions, Journal of Computational Physics 155 (2) (1999) 468–498.
[3] L. Greengard, J. Huang, A new version of the fast multipole method for screened Coulomb interactions in three dimensions, Journal of Computational Physics 180 (2) (2002) 642–658.
[4] L. Greengard, J. Strain, The fast Gauss transform, Siam Journal of Scientific and Statistical Computing 12 (1) (1991) 79–94.
[5] A. Gray, A.W. Moore, N-Body Problems in Statistical Learning, in: T.K. Leen, T.G. Dietterich, V. Tresp (Eds.), Advances in Neural Information Processing Systems 13 (December 2000), Mit Press, 2001.
[6] C. Yang, R. Duraiswami, N.A. Gumerov, L. Davis, Improved fast gauss transform and efficient kernel density estimation, in: International Conference on Computer Vision.
[7] P. Martinsson, V. Rokhlin, Y.U.D. of Computer Science, An accelerated kernel-independent fast multipole method in one dimension, Siam Journal on Scientific Computing 29 (3) (2008) 1160.
[8] S. Kapur, D. Long, Ies 3: efficient electrostatic and electromagneticsimulation, IEEE Computational Science & Engineering 5 (4) (1998) 60–67.
[9] C. Anderson, An implementation of the fast multipole method without multipoles, Siam Journal on Scientific and Statistical Computing 13 (1992) 923.
[10] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, Journal of Computational Physics 196 (2) (2004) 591–626.
[11] G. Marcelli, The Role of Three-Body Interactions on the Equilibrium and Non-Equilibrium Properties of Fluids from Molecular Simulation, PhD. Thesis Swinburne University of Technology Department of Computer Science, 2001.
[12] G. Marcelli, B. Todd, R. Sadus, Beyond traditional effective intermolecular potentials and pairwise interactions in molecular simulation, in: Computational Science ICCS 2002, 2009, pp. 932–941.
[13] J.L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (1975) 509–517.
[14] A.G. Gray, A.W. Moore, Nonparametric Density Estimation: Toward Computational Tractability, in: Siam International Conference on Data Mining, 2003.
[15] A. Gray, A. Moore, Very fast multivariate kernel density estimation via computational geometry, in: Joint Stat. Meeting, 2003.
[16] A. Gray, A. Moore, Rapid evaluation of multiple density models, 2003.
[17] A. Moore, A. Connolly, C. Genovese, A. Gray, L. Grone, N. Kanidoris, R. Nichol, J. Schneider, A. Szalay, I. Szapudi, L. Wasserman, Fast algorithms and efficient statistics: N-point correlation functions, in: Proceedings of Mpa/Mpe/Eso Conference Mining the Sky, July 31–August 4, Garching, Germany, 2000.
[18] D. Lee, A. G. Gray, A. W. Moore, Dual-tree fast gauss transforms, 2011. <arXiv:1102.2878>.
[19] M. Holmes, A. Gray, C. Isbell Jr, Fast kernel conditional density estimation: a dual-tree Monte Carlo approach, Computational Statistics & Data Analysis 54 (7) (2010) 1707–1718.
[20] M. Holmes, A. Gray, C. Isbell Jr, Ultrafast Monte Carlo for kernel estimators and generalized statistical summations, Advances in Neural Information Processing Systems (Nips) 21.
[21] D. Lee, A. Gray, Fast high-dimensional kernel summations using the monte carlo multipole method, Advances in Neural Information Processing Systems 21 (2009).
[22] G. Boyer, R. Riegel, N. Vasiloglou, D. Lee, L. Poorman, C. Mappus, N. Mehta, H. Ouyang, P. Ram, L. Tran, W.C. Wong, A. Gray, Mlpack, 2009. <http://mloss.org/software/view/152>.
[23] R. Curtin, J. Cline, N. Slagle, M. Amidon, A. Kale, B. March, N. Mehta, P. Ram, D. Lee, A. Gray, libmlpack, 2011. <http://mloss.org/software/view/364/>.
[24] J. Bentley, K-d trees for semidynamic point sets, in: Proceedings of the Sixth Annual Symposium on Computational Geometry, ACM, 1990, pp. 187–197.
[25] J. Li, Z. Zhou, R. Sadus, Modified force decomposition algorithms for calculating three-body interactions via molecular dynamics, Computer Physics Communications 175 (11-12) (2006) 683–691.
[26] R. Sampath, H. Sundar, S. Veerapaneni, Parallel Fast Gauss Transform, in: Supercomputing, 2010.