An efficient linked list for molecular simulations on a spherical surface

OPEN ACCESS
IOP Publishing
Journal of Physics A: Mathematical and Theoretical
J. Phys. A: Math. Theor. **55** (2022) 385001 (26pp)
https://doi.org/10.1088/1751-8121/ac852f

An efficient linked list for molecular simulations on a spherical surface

Esteban Vélez Ramírez[1,2] **and Christer Elvingson**[2,*]

[1] Department of Mathematics, Uppsala University, Box 480, 751 06 Uppsala, Sweden
[2] Department of Chemistry—Ångström Laboratory, Physical Chemistry, Uppsala University, Box 523, 751 20 Uppsala, Sweden

E-mail: Christer.Elvingson@kemi.uu.se

Received 10 February 2022, revised 30 June 2022
Accepted for publication 28 July 2022
Published 23 August 2022

Abstract

The main part of the cpu time in molecular simulations is usually spent calculating the non-bonded interactions. To improve the efficiency, a neighbour list or a cell linked list is normally used, where the linked list is usually more efficient and requires less memory for a large number of particles or dense systems. The linked list, however, still suffers from including many pairs which will not be within the interaction distance, and this overhead becomes even more significant for higher dimensions. In this work, we consider specifically simulations of particles confined to move on a spherical surface, where the overhead using a cubic grid to form a linked list becomes even larger by also including many cells which do not intersect the sphere. We address this by setting up a linked list directly on the spherical surface, thus reducing the dimensionality of the resulting neighbour search. We show that one obtains not only a substantial reduction in cpu time, but also a significant decrease in memory requirement. We further show how to extend this procedure to the 3-sphere (hypersphere), which can be used to simulate bulk systems avoiding periodic boundaries. Also in this case, using a linked list directly on the 3-sphere, the reduction in cpu time is significant, and the decrease in memory requirement, compared with a regular grid in $\mathbb{R}^4$, is even more pronounced. We finally comment on how the efficiency of the described method can be further improved.

Keywords: linked list, spherical surface, Brownian dynamics, molecular dynamics, Monte Carlo simulation, non-bonded interactions

*Author to whom any correspondence should be addressed.

S Supplementary material for this article is available online

(Some figures may appear in colour only in the online journal)

1. Introduction

Many important processes on the molecular and colloidal scale occur on curved surfaces. These cover a broad range of examples, not least in biology and chemistry, e.g., diffusion of molecules in membranes, cell dynamics, or nanoparticles confined to a spherical surface [1–6]. Important progress has also been made in the modelling of active particles on curved surfaces on the microscale [7–10], even up to literally global motion processes [11].

Using computer simulations to model both structure and dynamical processes on the microscopic level is paramount to improving our understanding also for macroscopic properties. In such simulations, the calculation of the non-bonded interactions will take the bulk of the cpu time. The short-range interactions are normally truncated at a certain cut-off distance, but also the real part in an Ewald summation of the electrostatic interactions involves a finite interaction distance [12, 13]. The calculation of the short-range forces are usually performed using a Verlet neighbour list or a cell linked list to reduce the computational cost [12–16]. For large systems, the linked list is normally used, showing both higher efficiency and requiring less memory [12, 13, 17]. Although reducing the cpu time substantially compared to $\mathcal{O}(n^2)$ if calculating the interaction between all pairs (for n number of particles), several improved methods for both the neighbour list and the cell linked list or combinations of these have been described [18–25], and for many systems, one can also estimate the optimal parameters for updating a neighbour list or the size of a cell in a linked list [26–28]. The use of neighbour lists and linked lists are important also in other types of simulations [29–31].

For particles confined to move on a spherical surface (the 2-sphere, denoted $\mathbb{S}^2$), one normally views the system as living in a three-dimensional Euclidean space. Using a linked list would then, in principle, mean that every cell would have 26 neighbouring cells, where some of the cells, however, are not intersecting the spherical surface.

For simulations of bulk systems with long-range interactions, such as electrostatic forces, an alternative to Ewald summation and other methods used to approximate the resulting infinite sum [12, 13], is to use spherical boundary conditions by embedding the system to the surface of a ball in $\mathbb{R}^4$ (the 3-sphere, denoted $\mathbb{S}^3$). This method, initially introduced for LJ-systems [32, 33], was extensively developed by Caillol *et al* for electrostatic interactions [34–37], and has also been used to model dipolar fluids [38], charged colloids [39], diffusion processes using Brownian dynamics (BD) [40], and for simulations of polymer chain molecules [41], as well as for constructing a closed, non-periodic network in simulations of gel-like materials [42]. The use of periodic boundaries for systems with long-range interactions in $\mathbb{R}^3$ can also lead to artificial periodicities or other artifacts [43–45], and a system size dependence, not least for dynamic properties [12, 45]. For simulations of electrolytes in $\mathbb{S}^3$, however, no system size dependence can be observed [46].

Setting up an ordinary cell linked list in $\mathbb{R}^4$, for simulations on $\mathbb{S}^3$, means, in principle, including 80 neighbour cells, for which an even larger search volume will not intersect the 3-sphere. Since $\mathbb{S}^2$ is a two-dimensional surface and $\mathbb{S}^3$ a three-dimensional volume, the number of neighbour cells within the interaction range can, however, be reduced by constructing a linked list directly on the respective surface, being one dimension smaller compared with using a regular grid in $\mathbb{R}^3$ or $\mathbb{R}^4$. In this work, we describe the implementation of a cell linked list directly on the respective spherical surface, comparing both the memory requirement and the

cpu time with using a cubic grid in the corresponding Euclidean space. The construction of the list of neighbour cells to each 'central' cell is more involved compared to the implementation using a cubic grid, but since this is only done once before the actual simulation begins, it will only have a negligible effect on the cpu time. Instead, we show that the time for evaluating the short-range interactions is considerably reduced, even if the time to sort the particles into the different cells increases. An equally important aspect is that the memory requirement decreases substantially, since the dimensionality of the list of neighbour cells is reduced.

In sections 2 and 3, we briefly review the properties of the unit sphere in d dimensions and also summarize an algorithm due to Leopardi, for dividing a d-dimensional sphere into regions of equal area [47]. This is followed by a general description of how to determine the neighbour cells (which we from now on, following Leopardi, will also call regions) for each central cell. This procedure is then described in an algorithmic way for $\mathbb{S}^1$ and $\mathbb{S}^2$, and we also briefly comment on how this is done for $\mathbb{S}^3$, which is described in more detail in the supplementary material (https://stacks.iop.org/JPA/55/385001/mmedia). Many of the details for $\mathbb{S}^1$ and $\mathbb{S}^2$ are also given in the supplementary material, and we have tried below to only keep the principle steps. We then present the results of simulations performed to test both the memory requirement and efficiency of this new method. The paper ends with some concluding remarks.

2. The geometry of the unit sphere

To introduce the notation used in the following sections, we briefly summarize the properties of the d-dimensional unit sphere, $\mathbb{S}^d$, in the Euclidean space $\mathbb{R}^{d+1}$, which is defined by

$$\mathbb{S}^d = \left\{ x \in \mathbb{R}^{d+1} \;\middle|\; \sum_{i=1}^{d+1} x_i^2 = 1 \right\}. \tag{1}$$

A point $\mathbf{a} \in \mathbb{S}^d$ can be characterized either by its Cartesian coordinates $(x_1, \ldots, x_{d+1})$ or by the spherical polar coordinates $(\theta_{d-1}, \ldots, \theta_1, \phi)$, which include $d-1$ colatitude angles $\theta_1, \ldots, \theta_{d-1} \in [0, \pi]$ and one longitude angle $\phi \in [0, 2\pi)$. The Cartesian coordinates are related to the spherical polar coordinates by [48]

$$\begin{aligned} x_1 &= \cos\,\phi \prod_{j=1}^{d-1} \sin\,\theta_j, \quad x_2 = \sin\,\phi \prod_{j=1}^{d-1} \sin\,\theta_j, \\ x_k &= \cos\,\theta_{k-2} \prod_{j=k-1}^{d-1} \sin\,\theta_j, \quad k \in \{3, \ldots, d+1\}. \end{aligned} \tag{2}$$

The angle θ_{d-1} is chosen to be the major colatitude, which means that the north and south poles of $\mathbb{S}^d$ correspond to $\theta_{d-1} = 0$ and $\theta_{d-1} = \pi$ respectively. We will also be using a subset of $\mathbb{S}^d$, produced by setting c number of colatitude angles constant, $\theta_j = \alpha_j$ for $j \in \{d-c, \ldots, d-1\}$. This generates a sphere of dimension $d-c$ and radius $r = \prod_{j=d-c}^{d-1} \sin\,\alpha_j$, denoted $\mathbb{S}^{d-c}(\alpha_{d-1}, \ldots, \alpha_{d-c})$.

An example of this is the generalization of the concept of a parallel through a point on $\mathbb{S}^2$, which would generate a circle, $\mathbb{S}^1(\alpha)$. Thus, for $\mathbf{a} = \mathbf{x}(\alpha_{d-1}, \theta_{d-2}, \ldots, \theta_2, \theta_1, \phi) \in \mathbb{S}^d$, a *parallel* through $\mathbf{a}$ is given by

$$\ominus(\mathbf{a}) = \mathbb{S}^{d-1}(\alpha_{d-1}), \tag{3}$$

which is a sphere of dimension $d-1$ and radius $r = \sin\,\alpha_{d-1}$.

In section 3, we will summarize an equal area partition of $\mathbb{S}^d$ introduced by Leopardi [47], which will be the starting point when constructing a linked list on $\mathbb{S}^{\mathrm{d}}$ (a 'spherical linked list'). To that end, and following Leopardi's notation, we introduce the following subsets of $\mathbb{S}^d$. For $d > 1$, a *zone*, $\mathbb{Z}(\alpha, \beta) \subset \mathbb{S}^d$, is described by

$$\mathbb{Z}(\alpha, \beta) = \left\{(\theta_{d-1}, \ldots, \theta_1, \phi) \in \mathbb{S}^d \quad | \quad \theta_{d-1} \in [\alpha, \beta]\right\}, \tag{4}$$

where $0 \leqslant \alpha < \beta \leqslant \pi$. Specifically, $\mathbb{Z}(0, \alpha)$ is the north polar cap, $\mathbb{Z}(\alpha, \pi)$ is the south polar cap, and $\mathbb{Z}(\alpha, \beta)$ defines a *collar* (an annular region) for $0 < \alpha < \beta < \pi$, which is bounded by the parallels through $(\alpha, \theta_{d-2}, \ldots, \theta_1, \phi)$ and $(\beta, \theta_{d-2}, \ldots, \theta_1, \phi)$. If $\mathbb{Z}(\alpha, \beta)$ is a collar and the angles, $0 < \nu < \tau < \pi$, represent fixed values for the colatitude angle θ_{d-2}, we introduce the subset $\mathbb{W}(\nu, \tau) \subset \mathbb{Z}(\alpha, \beta)$ as

$$\mathbb{W}(\nu, \tau) = \left\{(\theta_{d-1}, \ldots, \theta_1, \phi) \in \mathbb{S}^d \quad | \quad (\theta_{d-1}, \theta_{d-2}) \in [\alpha, \beta] \times [\nu, \tau]\right\}, \tag{5}$$

called a *sub-zone* bounded by the spheres $\mathbb{S}^{d-1}(\alpha)$ and $\mathbb{S}^{d-1}(\beta)$ in the θ_{d-1} direction, and the spheres $\mathbb{S}^{d-1}(\theta_{d-2} = \nu)$ and $\mathbb{S}^{d-1}(\theta_{d-2} = \tau)$ in the θ_{d-2} direction. If $\nu = 0$ and $\tau \neq 0$, we call the set $\mathbb{W}(0, \tau)$ a *pseudo north polar cap* belonging to the collar $\mathbb{Z}(\alpha, \beta)$, and if $\nu \neq 0$ and $\tau = \pi$, the set $\mathbb{W}(\nu, \pi)$ is a *pseudo south polar cap* belonging to the same collar. If $\nu \neq 0$ and $\nu \neq \pi$, and also $\tau \neq 0$ and $\tau \neq \pi$, $\mathbb{W}(\nu, \tau)$ is a *sub-collar* belonging to the collar $\mathbb{Z}(\alpha, \beta)$.

We further recall that the distance $\gamma(\mathbf{a}, \mathbf{b})$ between two points $\mathbf{a}, \mathbf{b} \in \mathbb{S}^d$ is given by $\gamma(\mathbf{a}, \mathbf{b}) = \cos^{-1}(\mathbf{a} \cdot \mathbf{b})$, which is part of the great circle which passes through the points $\mathbf{a}$ and $\mathbf{b}$. A north polar cap can then be written

$$\mathbb{Z}(0, \alpha) = \{\mathbf{b} \in \mathbb{S}^d \quad | \quad \gamma((0, \theta_{d-2} \ldots, \theta_1, \phi), \mathbf{b}) \leqslant \alpha\}, \tag{6}$$

with $(0, \theta_{d-2}, \ldots, \theta_1, \phi)$ being the north pole. An equivalent expression applies for the south polar cap, with $(0, \theta_{d-2}, \ldots, \theta_1, \phi)$ being replaced by $(\pi, \theta_{d-2}, \ldots, \theta_1, \phi)$. To construct a linked list on a sphere, we further introduce the following lemma (a proof is given in appendix A).

Lemma 2.1. *Let* $\mathbf{a} = \left(\theta^a_{d-1}, \ldots, \theta^a_1, \phi^a\right) \in \mathbb{S}^d$ *and* $s \in [0, \pi]$ *such that* $s = \gamma(\mathbf{a}, \mathbf{b})$*, where* $\mathbf{b} \in \mathbb{S}^d$ *is a point with fixed colatitude angles* $\theta^b_1, \theta^b_2, \ldots, \theta^b_{d-1}$*. The longitude angle,* ϕ^b *for the point* $\mathbf{b}$ *can then take the following two values*

$$\phi^b = \begin{cases} \phi^a + \cos^{-1}\left(\Upsilon\left(s, \theta^a_{d-1}, \ldots, \theta^a_1, \theta^b_{d-1}, \ldots, \theta^b_1\right)\right), \\ \phi^a - \cos^{-1}\left(\Upsilon\left(s, \theta^a_{d-1}, \ldots, \theta^a_1, \theta^b_{d-1}, \ldots, \theta^b_1\right)\right), \end{cases} \tag{7}$$

where

$$\Upsilon\left(s, \theta^a_{d-1}, \ldots, \theta^a_1, \theta^b_{d-1}, \ldots, \theta^b_1\right) = \frac{\cos s - \sum_{k=1}^{d-1} \cos\, \theta^a_k \cos\, \theta^b_k \prod_{j=k+1}^{d-1} \sin\, \theta^a_j \sin\, \theta^b_j}{\prod_{k=1}^{d-1} \sin\, \theta^a_k \sin\, \theta^b_k}. \tag{8}$$

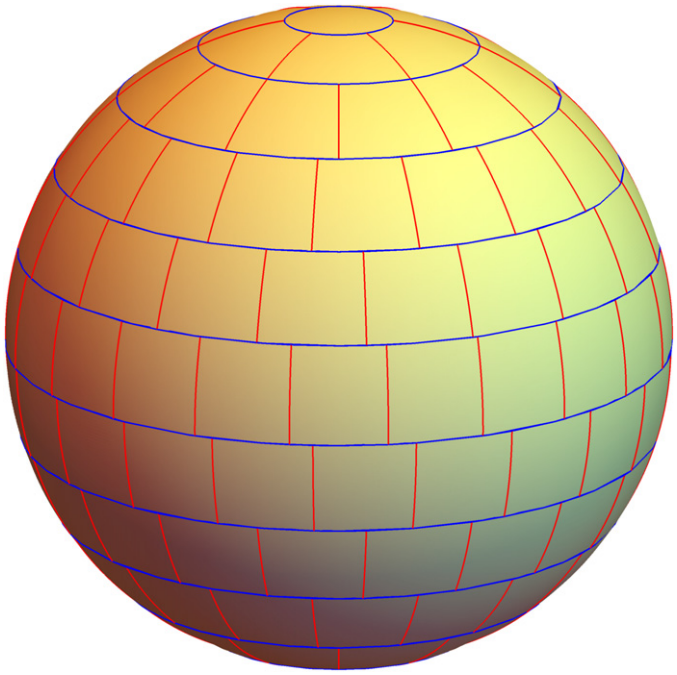


Figure 1. An equal area partition of $\mathbb{S}^2$ with $N = 153$ regions.

In the following sections, s will be the maximum interaction distance (the cut-off distance) between particles moving on $\mathbb{S}^{\mathrm{d}}$.

3. Equal area partition algorithm

To have approximately the same number of particles in each cell when setting up a linked list on $\mathbb{S}^d$ for a homogeneous system, we will use the equal area partition algorithm EQ(d, N), introduced by Leopardi [47] to divide the unit sphere, $\mathbb{S}^d$, into N equal area regions, $\mathcal{R}_i$.

For convenience, we enumerate the regions southwards, beginning with 1 for the north polar cap to N for the south polar cap. Figure 1 illustrates the partition of $\mathbb{S}^2$ into 153 equal area regions. The partition consists of two polar caps, while the remaining regions are spherical rectangles arranged in zonal collars. One should note that: (i) the sides of the regions (which are circular segments) have in general different lengths; (ii) the vertexes of the spherical rectangles lie on edges and not corners of neighbouring regions. These two properties illustrate a difference compared to a regular cubic grid. Another observation is that the colatitude intervals defining the different collars are not constant. Thus, for two different collars $\mathbb{Z}(\alpha_j, \alpha_{j+1})$ and $\mathbb{Z}(\alpha_k, \alpha_{k+1})$, in general $\Delta\alpha_j \neq \Delta\alpha_k$. It is not difficult to show that although the EQ(d, N) algorithm generates regions on the unit sphere, $\mathbb{S}^d$, the corresponding angles are independent of the radius, R, of the sphere.

To illustrate the partition also for $\mathbb{S}^3$, we will change the notation $(\theta_2, \theta_1, \phi)$ to (ξ, θ, ϕ). Although we can not visualize this partition directly, figure 2 shows a geometric construction which helps to illustrate how an arbitrary region, $\mathcal{R}_i \in \mathbb{S}^3$, can be understood. Notice that $\mathcal{R}_i$ has an associated collar $\mathbb{Z}(\xi_j, \xi_{j+1})$ (or zone if $\mathcal{R}_i$ is a polar cap), to which it belongs, and, assuming

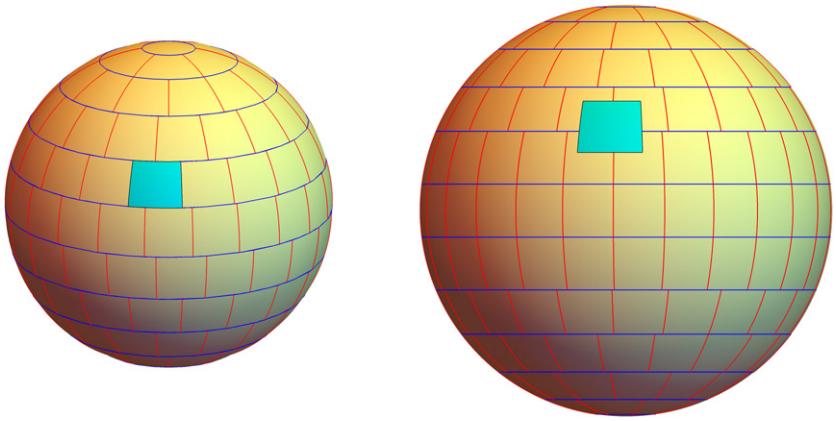


Figure 2. Representation of an arbitrary region, $\mathcal{R}_i$, belonging to a collar $\mathbb{Z}(\xi_j, \xi_{j+1}) \subset \mathbb{S}^3$. The sphere to the left represents the parallel through the point (ξ_j, θ, ϕ) together with the corresponding equal area subsets generated by the EQ(3, N) algorithm acting on collar $\mathbb{Z}(\xi_j, \xi_{j+1})$. The sphere to the right represents the parallel through the point $(\xi_{j+1}, \theta, \phi)$ together with the equal area subsets generated by EQ(3, N) acting on the collar $\mathbb{Z}(\xi_{j+1}, \xi_{j+2})$. The light blue subsets represent the upper and lower parts, respectively of the region $\mathcal{R}_i$. Assuming the collars are on the northern hemisphere, the parallel through $(\xi_{j+1}, \theta, \phi)$ will have a larger radius compared to the parallel through (ξ_j, θ, ϕ), which is also illustrated in the figure.

$\mathcal{R}_i$ is neither a polar cap nor a pseudo polar cap, it looks like a spherical 'brick'. Figure 2 (left) shows a 2-sphere, $\mathbb{S}^2(\xi_j)$, which represents the parallel through the point (ξ_j, θ, ϕ) (the upper bound of the collar which contains the region), subdivided into spherical subsets of equal area, which characterize the effect of the EQ(3, N) algorithm on the collar $\mathbb{Z}(\xi_j, \xi_{j+1})$. The light blue subset describes the upper part (or 'ceiling') of region $\mathcal{R}_i$. Figure 2 (right) represents a parallel through the point $(\xi_{j+1}, \theta, \phi)$ (lower bound of the collar which contains the region), subdivided into spherical subsets of equal area on the collar $\mathbb{Z}(\xi_{j+1}, \xi_{j+2})$. The light blue subset is in this case the lower part (or 'floor') of region $\mathcal{R}_i$.

To illustrate the EQ(d, N) algorithm, we describe the output of EQ(3, N) acting on the 3-sphere. When describing how to set up the linked list for $\mathbb{S}^1$ and $\mathbb{S}^2$ in sections 4.1 and 4.2, we will keep the more extensive notation for $\mathbb{S}^3$ introduced in the present section, but setting relevant angles as constants. For $\mathbb{S}^3$, we first introduce the list, $\boldsymbol{\Xi}$, describing the division of the major colatitude angle (the ξ direction) into zonal intervals $[\xi_j, \xi_{j+1}]$, which is given by

$$\Xi = (\xi_1, \ldots, \xi_j, \ldots, \xi_{Z+1}), \quad \xi_1 = 0, \quad \xi_{Z+1} = \pi, \tag{9}$$

with Z being the total number of zones (i.e. the collars and the two polar caps). For the second colatitude angle, θ, we define the two-dimensional list, $\boldsymbol{\Theta}$, given by

$$\Theta = (0, \Theta_2, \ldots, \Theta_j, \ldots, \Theta_{Z-1}, 0), \tag{10}$$

where each element is a (sub)list

$$\Theta_j = (\theta_{j1}, \ldots, \theta_{jk}, \ldots, \theta_{jw_j}), \quad \theta_{j1} = 0, \quad \theta_{jw_j} = \pi, \tag{11}$$

which represents the total number of sub-zones within the collar $\mathbb{Z}(\xi_j, \xi_{j+1})$, with the index w_j being the jth element of the list $\mathbf{W}$ (the sub-zone list, see equation (13)). Equation (5) allows

us to write an expression for each specific $\mathbb{W}(\theta_{jk}, \theta_{jk+1})$

$$\mathbb{W}(\theta_{jk}, \theta_{jk+1}) = \left\{(\xi, \theta, \phi) \in \mathbb{S}^3 \quad | \quad (\xi, \theta) \in [\xi_j, \xi_{j+1}] \times [\theta_{jk}, \theta_{jk+1}]\right\}. \tag{12}$$

The list, **W**, representing the different sub-zones, is then given by

$$\mathbf{W} = (0, w_2, \ldots, w_j, \ldots, w_{Z-1}, 0). \tag{13}$$

Notice that the first and last element in **W** must be zero since the polar caps do not contain sub-zones.

To describe the partition in the longitude direction, we introduce **Φ**, which is a three-dimensional list

$$\begin{aligned}\mathbf{\Phi} = (0, (\Phi^{21}, \ldots, \Phi^{2k}, \ldots, \Phi^{2w_2}), \ldots, (\Phi^{j1}, \ldots, \Phi^{jk}, \ldots, \Phi^{jw_j}),\\ \ldots, (\Phi^{Z-11}, \ldots, \Phi^{Z-1k}, \ldots, \Phi^{Z-1w_{Z-1}}), 0),\end{aligned} \tag{14}$$

where each element, Φ^{jk}, is a list

$$\begin{aligned}&\Phi^{j1} = (0, 2\pi), \quad \Phi^{jk} = (\phi_1^{jk}, \ldots, \phi_l^{jk}, \ldots, \phi_{q_{jk}+1}^{jk}), \quad \Phi^{jw_j} = (0, 2\pi),\\ &\phi_1^{jk} = 0 \quad \text{and} \quad \phi_{q_{jk}+1}^{jk} = 2\pi,\end{aligned} \tag{15}$$

where the q_{jk} sub-index is the element jk in the two-dimensional list, **q** (see equation (16)) which represents the total number of regions within the sub-zone $\mathbb{W}(\theta_{jk}, \theta_{jk+1})$. One should note that there are $q_{jk} + 1$ longitudinal angles per sub-zone, and that Φ^{j1} and Φ^{jw_j} are the longitude-lists corresponding to the pseudo north and pseudo south polar caps associated with collar $\mathbb{Z}(\xi_j, \xi_{j+1})$, and therefore spanning the interval $[0, 2\pi)$. When writing equations (14) and (15), we have introduced the convention in which longitude variables use two super-indexes for the colatitude directions and one sub-index for the longitude direction. Thus, the first super-index, j, refers to the collar $\mathbb{Z}(\xi_j, \xi_{j+1})$ associated with the angle ξ, and the second super-index, k, refers to the sub-zone $\mathbb{W}(\theta_{jk}, \theta_{jk+1})$ associated with the angle θ. The sub-index l in equation (15) represents the division of the longitude interval into equal angular segments of length $\Delta\phi^{jk} = \phi_{l+1}^{jk} - \phi_l^{jk}$ for each sub-zone. The list, **q**, which contains the number of regions per sub-collar is now given by

$$\begin{aligned}\mathbf{q} = (0, (q_{21}, \ldots, q_{2k}, \ldots, q_{2w_2}), \ldots, (q_{j1}, \ldots, q_{jk}, \ldots, q_{jw_j}),\\ \ldots, (q_{Z-11}, \ldots, q_{Z-1k}, \ldots, q_{Z-1w_{Z-1}}), 0),\\ q_{j1} = q_{jw_j} = 1.\end{aligned} \tag{16}$$

The first and last entry in the list **q** are not lists, but are equal to zero as they represent the polar caps which do not have sub-zones.

We further define the one-dimensional list, **p**, where the element, p_j, is the total number of regions in the zone $\mathbb{Z}(\xi_j, \xi_{j+1})$, given by

$$\mathbf{p} = (p_1, \ldots, p_j, \ldots, p_Z), \tag{17}$$

where $p_1 = p_Z = 1$, being the north and south polar caps, respectively. The elements p_j and q_{jk} further satisfy the following relations

$$\begin{aligned} p_j &= \sum_{k=1}^{w_j} q_{jk} \quad \text{where} \quad j \in \{2, Z-1\}, \\ N &= \sum_{j=1}^{Z} p_j = 2 + \sum_{j=2}^{Z-1} \sum_{k=1}^{w_j} q_{jk}. \end{aligned} \tag{18}$$

The lists **p** and **q**, do not, however, provide all the necessary information to construct a linked list on $\mathbb{S}^3$, and we need to introduce two additional lists. Since the regions are enumerated in increasing order starting from the north polar cap, we first introduce the list, $\mathbf{p}^+$,

$$\mathbf{p}^+ = (p_1^+, \ldots, p_j^+, \ldots, p_Z^+), \quad p_1^+ = 0, \quad p_{j>1}^+ = \sum_{l=1}^{j} p_l, \tag{19}$$

where p_j^+ is the total number of regions contained in the set $\mathbb{Z}(0, \xi_j)$. Since $\mathbb{Z}(0, \xi_1) = (0, \theta, \phi)$ is the north pole, $p_1^+ = 0$, as the north pole is a point not containing any regions. We also define the list $\mathbf{q}^+$ by

$$\begin{aligned} \mathbf{q}^+ &= (0, (q_{21}^+, \ldots, q_{2k}^+, \ldots, q_{2w_2}^+), \ldots, (q_{j1}^+ \ldots, q_{jk}^+, \ldots, q_{jw_j}^+), \\ &\quad \ldots, (q_{Z-11}^+, \ldots, q_{Z-1k}^+, \ldots, q_{Z-1w_{Z-1}}^+), 0), \\ q_{j1}^+ &= \sum_{l=1}^{j} p_l, \quad \text{and} \quad q_{jk}^+ = p_{j+1}^+ + \sum_{t=1}^{k-1} q_{jt}, \end{aligned} \tag{20}$$

where q_{jk}^+ represents the number of regions contained in the set $\mathbb{Z}(0, \xi_j) \cup \mathbb{W}(0, \theta_k)$. The list $\mathbf{p}^+$ is thus the accumulated number of regions, starting from the north polar cap, including collar $\mathbb{Z}(\xi_{j-1}, \xi_j)$, while the list $\mathbf{q}^+$, which goes deeper into the partition, provides us with the accumulated number of regions starting from the north polar cap including collar $\mathbb{Z}(\xi_{j-1}, \xi_j)$ plus the accumulated number of regions belonging to the collar $\mathbb{Z}(\xi_j, \xi_{j+1})$ starting from the pseudo-north polar cap, $\mathbb{W}(0, \theta_{j2})$, also including the sub collar $\mathbb{W}(\theta_{jk-1}, \theta_{jk})$. We can note that $q_{j1}^+ = p_j^+$ as it corresponds to the accumulated number of regions up to the pseudo north polar cap belonging to collar $\mathbb{Z}(\xi_j, \xi_{j+1})$.

Based on the information provided by the partition algorithm described above, an arbitrary region, $\mathcal{R}_i$, on $\mathbb{S}^3$ can now be introduced

$$\mathcal{R}_i = ([\xi_j, \xi_{j+1}] \times [\theta_{jk}, \theta_{jk+1}] \times [\phi_l^{jk}, \phi_{l+1}^{jk}]). \tag{21}$$

We also note that any given index, i, is characterized by a unique combination of indexes $\{j, k, l\}$, where $1 \leqslant j \leqslant Z$, $1 \leqslant k \leqslant w_j$ and $1 \leqslant l \leqslant q_{jw_j}$.

4. The structure of a spherical linked list

To illustrate the construction of a linked list on a spherical surface, we first discuss the general procedure and then show in some detail how to do this for $\mathbb{S}^1$ and $\mathbb{S}^2$ before also briefly describing the procedure for $\mathbb{S}^3$. We begin by defining a list, M, which identifies all the neighbour cells

Algorithm 1. FCN algorithm.

procedure FCN($s, \mathcal{R}_i, j, k, v, u$)▷ List of neighbour cells to a cap in $\mathbb{Z}(\xi_v, \xi_{v+1})$ or in $\mathbb{W}(\theta_{vu}, \theta_{vu+1})$

if $j = 1$ **then**

$\mathcal{R}_i$ is the north polar cap. Construct the list M_1^v of neighbour cells in the collar; $\mathbb{Z}(\xi_v, \xi_{v+1})$

end if

if $j \neq 1$ and $k = 1$ **then**

$\mathcal{R}_i$ is a pseudo north polar cap. Construct the list M_{j1}^{vu} of neighbour cells within the sub-collar $\mathbb{W}(\theta_{vu}, \theta_{vu+1})$;

end if

end procedure

to every region. The number of neighbour cells for each $\mathcal{R}_i$ will depend on the particular colatitude and longitude angle for each region, as well as on the interaction distance, s. The list, **M**, begins with the neighbour cells to the first region, the north polar cap, followed by all neighbour cells to the second region, which is the pseudo north polar cap in the first collar, which is then followed by the neighbour cells of the third region, which is a region in the first sub-collar within the first collar, until it reaches the last region, the south polar cap. As the number of neighbour cells for different regions is not constant (unlike for a regular grid in $\mathbb{R}^{\mathrm{d}}$), we also need to know at which position in the list, **M**, a change of region occurs. A pointer list, **B**, is then constructed where the value of the entry i tells us the position in the list, **M**, where the neighbour cells of region i begins.

In the present section, we show how the concepts introduced in sections 2 and 3 can be used to construct the lists **M** and **B**, beginning by introducing two algorithms which we repeatedly will be referring to. The first algorithm, find cap neighbours, FCN($s, \mathcal{R}_i, j, k, v, u$), constructs a list of neighbour cells within the zone $\mathbb{Z}(\xi_v, \xi_{v+1})$ to the north polar cap or within the sub-zone $\mathbb{W}(\theta_{vu}, \theta_{vu+1})$ to a pseudo north polar cap (see algorithm 1). The second algorithm, find neighbours in sub-collar, FNSC($s, \mathcal{R}_i, j, k, l, v, u$), constructs a list of neighbour cells within the sub-collar $\mathbb{W}(\theta_{vu}, \theta_{vu+1})$ to a general region belonging to a particular sub-collar $\mathbb{W}(\theta_{jk}, \theta_{jk+1})$, with $j \neq 1$ and $k \neq 1$ (neither a polar cap nor a pseudo-polar cap), assuming that the sub-collar $\mathbb{W}(\theta_{vu}, \theta_{vu+1})$ is within the interaction distance (see algorithm 2). A more detailed description is given in the supplementary material.

For a given colatitude angle, ξ or θ, we need to introduce a procedure to determine the index of the zone which has position j in the list $\boldsymbol{\Xi}$ such that $\xi_j < \xi < \xi_{j+1}$ (see equation (9)), or the index of the sub-zone which has position k in the list Θ_j such that $\theta_{jk} < \theta < \theta_{jk+1}$ (see equation (11)). To that end, we introduce a general colatitude angle ϑ ($\vartheta = \xi$ if associated with a zone or $\vartheta = \theta$ if associated with a sub-zone). Let H denote the index of the zone $\mathbb{Z}(\xi_H, \xi_{H+1})$ or the sub-zone $\mathbb{W}(\theta_{jH}, \theta_{jH+1})$ which contains ϑ ($\xi_H < \vartheta < \xi_{H+1}$ or $\theta_{jH} < \vartheta < \theta_{jH+1}$). We define the map $\chi : (0, \pi) \times (0, \pi) \to \{1, \ldots, T\}$:

$$\chi(\vartheta, \vartheta_c) = H, \tag{22}$$

with ϑ_c being the angle which defines the north polar cap ($\vartheta_c = \xi_2$) and $T = Z$ (total number of zones) if ϑ is associated with a zone, or if ϑ_c is the angle which defines the pseudo north polar cap ($\vartheta_c = \theta_{j2}$), and $T = w_j$ if ϑ is associated with a sub-zone. The index H is calculated

Algorithm 2. FNSC algorithm.

procedure FNSC($s, \mathcal{R}_i, j, k, l, v, u$) ▷ List of neighbour cells to $\mathcal{R}_i$ in the sub-collar $\mathbb{W}(\theta_{vu}, \theta_{vu+1})$
  if $v < Z$ **then**
    if $u = 1$ **then**
      Construct the list M^{v1}_{jkl} which has just one element, the pseudo north polar cap $\mathbb{W}(0, \theta_{v2})$;
    else if $1 < u < w_v$ **then**
      Determine the range of the longitude interval in the sub-collar
      $\mathbb{W}(\theta_{vu}, \theta_{vu+1})$, which will include neighbour cells for a given interaction distance s;
      Determine the number of neighbour cells in this longitude interval;
      Construct the list M^{vu}_{jkl} of neighbour cells in this longitude interval;
    else
      Construct the list $M^{vw_v}_{jkl}$ which has just one element, the pseudo south polar cap $\mathbb{W}(\theta_{vw_v}, \pi)$;
    end if
      Construct the list M^{Z}_{jkl} which has just one element, the south polar cap;
  end if
end procedure

using the following consecutive steps which complete the definition of the map χ

$$
\begin{aligned}
H &= \left\lfloor \frac{\vartheta - \vartheta_c}{\vartheta_c} \right\rfloor + 1 + \frac{1 + \lfloor \mathrm{sign}\left(1, \frac{\vartheta - \vartheta_c}{\vartheta_c}\right) \rceil}{2}, \\
H &= H + \left\lfloor \frac{\vartheta}{\vartheta_H} \right\rfloor, \\
H &= H - \left\lfloor \frac{\vartheta_{H-1}}{\vartheta} \right\rfloor - 1,
\end{aligned}
\tag{23}
$$

where $\vartheta_H = \xi_H$ or $\vartheta_H = \theta_{jH}$, and where $\lfloor x \rceil$ is the round of x defined as $\lfloor x \rceil = \lfloor x + 0.5 \rfloor$, with $\lfloor x \rfloor$ being the floor of x.

We will below introduce the 1-sphere ($\mathbb{S}^1(\xi_j, \theta_{jk})$) and the 2-sphere ($\mathbb{S}^2(\xi_j)$) as sets produced by slicing the more general $\mathbb{S}^3$ at given values of ξ and θ. The algorithm which finds neighbour cells on $\mathbb{S}^3$ will then use the algorithm finding neighbour cells on $\mathbb{S}^2(\xi_j)$, and this in turn, will use the algorithm which finds neighbour cells on $\mathbb{S}^1(\xi_j, \theta_{jk})$. If we are only interested in finding neighbours on $\mathbb{S}^1$ or $\mathbb{S}^2$, the notation can be simplified by setting $\xi = \pi/2$ for $\mathbb{S}^2$ and $\xi = \theta = \pi/2$ for $\mathbb{S}^1$ in the respective algorithm.

4.1. Neighbour mapping algorithm on $\mathbb{S}^1$

Without loss of generality, and for $2 \leqslant j \leqslant Z - 1$ and $2 \leqslant k \leqslant w_j - 1$, we introduce a circle of radius $r = \sin\theta_{jk} \sin\xi_j$ as a subset of $\mathbb{S}^3$, where θ_{jk} and ξ_j are chosen to be the lower bound angles of the sub-collar $\mathbb{W}(\theta_{jk}, \theta_{jk+1})$ and the collar $\mathbb{Z}(\xi_j, \xi_{j+1})$ generated by EQ$(3, N)$ (see

Algorithm 3. NM_1 algorithm.

procedure NM_1$(s, \mathcal{R}_i, j, k, l)$ ▷ List of neighbour cells M^{jk}_{jkl} for $\mathcal{R}_i$

Apply the FNSC$(s, \mathcal{R}_i, j, k, l, j, k)$ algorithm to find the neighbour cells for each region within the set $\mathbb{S}^1(\xi_j, \theta_{jk})$;

end procedure

section 3). The induced partition on $\mathbb{S}^1(\xi_j, \theta_{jk})$ is then given by equation (15)

$$\Phi^{jk} = (\phi^{jk}_1, \ldots, \phi^{jk}_l, \ldots, \phi^{jk}_{q_{jk}+1}), \quad \phi^{jk}_1 = 0 \quad \text{and} \quad \phi^{jk}_{q_{jk}+1} = 2\pi. \tag{24}$$

The circle $\mathbb{S}^1(\xi_j, \theta_{jk})$ is divided into q_{jk} regions of length $\Delta\phi^{jk} = \phi^{jk}_{l+1} - \phi^{jk}_l$. For this trivial case, the lists $\mathbf{M}$ and $\mathbf{B}$ can easily be constructed. Let $s \in [0, \pi]$ be the maximum interaction distance between particles on $\mathbb{S}^1(\xi_j, \theta_{jk})$. The neighbour mapping algorithm which finds the neighbour cells to an arbitrary region $\mathcal{R}_i \in \mathbb{S}^1(\xi_j, \theta_{jk})$ within the interaction distance, s, is denoted by $\mathrm{NM}_1(s, \mathbb{R}_i, j, k, l)$ and is given in algorithm 3.

4.2. Neighbour mapping algorithm on $\mathbb{S}^2$

We next introduce the 2-sphere of radius $r = \sin\xi_j$, as a subset of $\mathbb{S}^3$, where ξ_j is again chosen to be the lower bound angle of collar $\mathbb{Z}(\xi_j, \xi_{j+1})$ generated by EQ$(3, N)$, and therefore, the induced partition on $\mathbb{S}^2(\xi_j)$ is given by a combination of equations (11) and (15)

$$\begin{aligned} \Theta_j &= (\theta_{j1}, \ldots, \theta_{jk}, \ldots, \theta_{jw_j}), \quad \theta_{j1} = 0, \quad \theta_{jw_j} = \pi, \\ &\quad ((0, 2\pi), \ldots, \Phi^{jk}, \ldots, (0, 2\pi)), \\ \Phi^{jk} &= (\phi^{jk}_1, \ldots, \phi^{jk}_l, \ldots, \phi^{jk}_{q_{jk}+1}), \quad \Delta\phi^{jk} = \phi^{jk}_{l+1} - \phi^{jk}_l \\ &\quad \phi^{jk}_1 = 0 \quad \text{and} \quad \phi^{jk}_{q_{jk}+1} = 2\pi. \end{aligned} \tag{25}$$

The 2-sphere, $\mathbb{S}^2(\xi_j)$, is divided into p_j regions of equal area, with p_j being the total number of regions in collar $\mathbb{Z}(\xi_j, \xi_{j+1})$. The $\mathrm{NM}_2(s, \mathbb{R}_i, j, k, l)$ algorithm given in algorithm 4, with index j fixed, now provides the list of neighbours $M^{jk}_{jk}, M^{jk+1}_{jk}, \ldots, M^{jK}_{jk}$ (index K defined in algorithm 4) in the different collars which include neighbour cells to region $\mathcal{R}_i$.

We can then construct the lists M and B as illustrated in algorithm 5.

4.3. Neighbour mapping algorithm on $\mathbb{S}^3$

We can generalize the neighbour mapping algorithm to include another colatitude degree of freedom for $\mathbb{S}^3$ ($\xi \in [\xi_j, \xi_{j+1}]$) by calling the algorithm $\mathrm{NM}_3(s, \mathbb{R}_i, j, k, l)$, and a description of this algorithm is given in the supplementary material. This will provide us with the list of neighbours cells, $M^{jk}_{jkl}, M^{jk+1}_{jkl}, \ldots, M^{jK}_{jkl}, M^{j+1K^-_{j+1}}_{jkl}, \ldots, M^{j+1K^+_{j+1}}_{jkl}, \ldots, M^{JK^-_J}_{jkl}, \ldots, M^{JK^+_J}_{jkl}$, within the different collars and the respective sub-collars, which include all neighbour cells associated with a region $\mathcal{R}_i$ for a given interaction distance. With this information, we can also construct the lists M and B according to algorithm 8, as shown in the supplementary material.

The NM_d ($d = 1, 2, 3$) algorithms are designed to include more than only nearest neighbour cells. This is a necessary procedure as the geometry of $\mathbb{S}^d$ is different compared to the flat $\mathbb{R}^{d+1}$. For a fixed distance, s, if, e.g., $\mathbf{a}, \mathbf{b} \in \mathbb{S}^2$, such that $\mathbf{a} = (\theta_{jk}, \phi^{jk+1}_l)$ and $\mathbf{b} = (\theta_{jk+1}, \phi^{jk}_l)$,

Algorithm 4. NM_2 algorithm.

procedure NM_2($s, \mathcal{R}_i, j, k, l$) ▷ List of neighbour cells $M_{jkl}^{jk}, M_{jkl}^{jk+1}, \ldots, M_{jkl}^{jK}$ for $\mathcal{R}_i$
 Find the index, K, of the southern most collar $\mathbb{W}(\theta_{jK}, \theta_{jK+1}) \cap \mathbb{S}^2(\xi_j)$ which includes neighbour cells within the interaction distance, s(see supplementary material);
 if $k = 1$ **then**
  for $u = 2$ **to** K **do**
   Apply the FCN($s, \mathcal{R}_i, j, 1, j, u$) algorithm to find the neighbour cells in the collar $\mathbb{W}(\theta_{ju}, \theta_{ju+1}) \cap \mathbb{S}^2(\xi_j)$;
  end for
 else
  Apply the NM_1($s, \mathcal{R}_i, j, k, l$) algorithm to find the neighbour cells in the collar $\mathbb{W}(\theta_{jk}, \theta_{jk+1}) \cap \mathbb{S}^2(\xi_j)$;
  for $u = k + 1$ **to** K **do**
   Apply the FNSC($s, \mathcal{R}_i, j, k, l, j, u$) algorithm to find the neighbour cells in the collar $\mathbb{W}(\theta_{ju}, \theta_{ju+1}) \cap \mathbb{S}^2(\xi_j)$;
  end for
 end if
end procedure

Algorithm 5. Construction of the lists M and B on $\mathbb{S}^2$.

Initialize M as an empty list, set $B = (b_1)$ with $b_1 = 1$ and set $h = 1$;
Calculate the neighbour cells to the north polar cap by applying NM_2($s, \mathcal{R}_i, j, 1, l$);
Compute $b_{h+1} = b_h + \|M_{j1}^{j2}\| + \cdots + \|M_{j1}^{jK}\|$[a] and update $h = h + 1$;
Append b_2 to B and the lists $M_{j1}^{j2}, \ldots, M_{j1}^{jK}$ to M;
for $k = 2$ **to** $w_j - 1$ **do**
 for $l = 1$ **to** q_{jk} **do**
  Apply NM_2($s, \mathcal{R}_i, j, k, l$) to calculate the lists $M_{jkl}^{jk}, M_{jkl}^{jk+1}, \ldots, M_{jkl}^{jK}$ of neighbour cells to region $\mathcal{R}_i$;
  Compute $b_{h+1} = b_h + \|M_{jkl}^{jk}\| + \|M_{jkl}^{jk+1}\| + \cdots + \|M_{jkl}^{jK}\|$ and update $h = h + 1$;
  Append b_{h+1} to B and the lists $M_{jkl}^{jk}, M_{jkl}^{jk+1}, \ldots, M_{jkl}^{jK}$ to M;
 end for
end for

[a]Note that only two sub-indexes are included because the north polar cap does not have an associated sub index l (see equation (25)).

where $0 < \theta_{jk} < \theta_{jk+1} < \pi/2$, it is more likely that regions which belong to collars which are closer to the polar caps must have more neighbour cells in the longitude direction compared with regions which belong to collars close to the equator. The algorithm also includes more than first neighbour collars in the different colatitude directions. For this reason the index, K, was introduced in the NM_2 algorithm (algorithm 4), and the indexes J, K_v^- and K_v^+ are introduced in the NM_3 algorithm (algorithm 7 in the supplementary material).

4.4. Sorting particles into a spherical linked list

Setting up the lists M and B, as described above, is done once, before the actual simulation begins. During a simulation, the particles are then sorted into the different regions before calculating the interactions. This is easily performed in $\mathbb{R}^d$ for a regular grid of boxes with periodic boundaries [12]. For simulations on a finite object like a sphere, one can still use a cubic network of boxes in $\mathbb{R}^d$, but one has to take into account that this grid will also be finite (see

appendix B). For a spherical linked list, however, when sorting the particles into the different regions, one must consider that the colatitude intervals defining the major collars and sub-collars are not constant (see section 3). One can therefore not use a sorting procedure based on a regularity of the grid. Instead the map, χ, described in section 4, can be used to sort the particles into their corresponding regions, and we illustrate this for the general case of a particle on $\mathbb{S}^3$.

Let $\mathbf{a} = (\xi_a, \theta_a, \phi_a) \in \mathbb{S}^3$ represent the spherical polar coordinates of an arbitrary particle at $\mathbf{x}_a$ moving on $\mathbb{S}^3$. The index, J, of the collar, $\mathbb{Z}(\xi_J, \xi_{J+1})$, which contains $\mathbf{x}_a$ can be calculated using equation (22), $J = \chi(\xi_a, \xi_2)$, where ξ_2 is the major colatitude angle which defines the north polar cap and is provided by the partition algorithm. The index, K, of the sub-collar $\mathbb{W}(\theta_{JK}, \theta_{JK+1})$ which contains $\mathbf{x}_a$, can be computed using the index, J, by $K = \chi(\theta_a, \theta_{J2})$ with θ_{J2} being the colatitude angle defining the pseudo north polar cap within the collar $\mathbb{Z}(\xi_J, \xi_{J+1})$. The index, L, which identifies the location of the particle in the longitude direction is given by

$$L = \left\lceil \frac{\phi_a}{\Delta\phi_L^{JK}} \right\rceil, \tag{26}$$

with $\lceil x \rceil$ being the ceiling of x. Thus, $\mathbf{x}_a$, belongs to the region defined by

$$\mathcal{R}_i = ([\xi_J, \xi_{J+1}] \times [\theta_{JK}, \theta_{JK+1}] \times [\phi_L^{JK}, \phi_{L+1}^{JK}]), \tag{27}$$

with the index $i \in \{1, \ldots, N\}$ given by

$$i = q_{JK}^{+} + L. \tag{28}$$

When all particles have been sorted into their corresponding regions, the spherical linked list is complete.

5. Computational details

We illustrate the use of spherical linked lists both on $\mathbb{S}^2$ and $\mathbb{S}^3$ by performing BD simulations. On $\mathbb{S}^3$, this can be done using the algorithm described by Nissfolk *et al* [40]. If $\mathbf{a}_i \in \mathbb{S}^3$ is the initial position of particle i, the new position, $\mathbf{a}_{\text{new}}$, is given by

$$\mathbf{a}_{\text{new}} = \cos\left(\frac{|\Delta\mathbf{r}_i|}{R}\right)\mathbf{a}_i + \frac{R\,\sin\left(|\Delta\mathbf{r}_i|/R\right)}{|\Delta\mathbf{r}_i|}\Delta\mathbf{r}_i, \tag{29}$$

where

$$\Delta\mathbf{r}_i = \frac{D_0\Delta t}{kT}\mathbf{F}_i + \mathbf{A}_i, \tag{30}$$

with Δt being the time step, $\mathbf{F}_i$ is the total direct force acting on particle i, R is the radius of $\mathbb{S}^3$, D_0 is the diffusion coefficient given by the Stokes–Einstein relation, $D_0 = kT/6\pi\eta b$, where η is the viscosity, b is the radius of the particle, and $\mathbf{A}_i$ is a random walk describing a Brownian motion on $\mathbb{S}^3$. In the present simulations, we chose for simplicity a soft repulsive interaction between particles [49]

$$\mathbf{F}_i = \frac{\epsilon kT}{(2b)^2}\sum_j \left(\frac{2b}{r_{ij}} - 1\right) r_{ij}\hat{\mathbf{t}}_{ij}, \tag{31}$$

where ϵ is a dimensionless force constant, r_{ij} is the distance between particles i and j along the geodesic, and $\hat{\mathbf{t}}_{ij}$ is the tangent vector to the geodesic connecting particles i and j.

As described by Carlsson *et al* [50], a BD simulation on $\mathbb{S}^2$ can be implemented from a Brownian motion on $\mathbb{S}^3$. If $\mathbf{a}_i \in \mathbb{S}^3$, such that $\mathbf{a}_i = (\mathbf{a}_1, a_4)$, where $\mathbf{a}_1 \in \mathbb{S}^2$, the new position, $\mathbf{a}_{\text{new}} \in \mathbb{S}^3$, of a Brownian particle constrained to move on $\mathbb{S}^2$ of radius r is given by

$$\mathbf{a}_{\text{new}} = \left(\frac{r}{|\tilde{\mathbf{y}}_1|} \tilde{\mathbf{y}}_1, a_4 \right), \tag{32}$$

where $\tilde{\mathbf{y}}_1$ is the result of mapping $\mathbf{a}_i$ onto the equator of $\mathbb{S}^3$ to the point $\tilde{\mathbf{a}}_1$ using

$$\tilde{\mathbf{a}}_1 = R \frac{\mathbf{a}_i - (\mathbf{a}_i \cdot \hat{\mathbf{x}}_4)\hat{\mathbf{x}}_4}{|\mathbf{a}_i - (\mathbf{a}_i \cdot \hat{\mathbf{x}}_4)\hat{\mathbf{x}}_4|}, \tag{33}$$

where $\hat{\mathbf{x}}_4$ is the unit vector in the x_4 direction, and then applying equation (29) to $\tilde{\mathbf{a}}_1$ resulting in $(\tilde{\mathbf{y}}_1, y_4)$. We can note [51] that a pure random walk can also be implemented directly on $\mathbb{S}^2$ or on a sphere of arbitrary dimension [52], as well as on more general manifolds [53].

In the present simulations, we use $T = 293.15$ K, $\epsilon = 200$, $\eta = 1.002$ mPas, $\Delta t = 10$ fs, $b = 0.5$ nm, and for the simulations on $\mathbb{S}^2$, $r = R$ (i.e. the equatorial $\mathbb{S}^3$). We have investigated two different interaction distances $\lambda = 1$ nm and $\lambda = 2$ nm, and three different volume fractions $\rho = 0.05, 0.10, 0.20$. The interaction distance is further converted to an angular distance, $s = \lambda/R$. The radius, R, is calculated from the volume fraction

$$\rho = \frac{nV(\mathbb{S}^{\mathrm{d}-1}(b))}{A(\mathbb{S}^{\mathrm{d}}(R))} = \frac{nb^d\, V(\mathbb{S}^{\mathrm{d}-1})}{R^d A(\mathbb{S}^{\mathrm{d}})}, \tag{34}$$

where n is the total number of particles, $V(\mathbb{S}^{\mathrm{d}}(b))$ is the volume of a ball of radius b in $\mathbb{R}^{\mathrm{d}+1}$ given by [54]

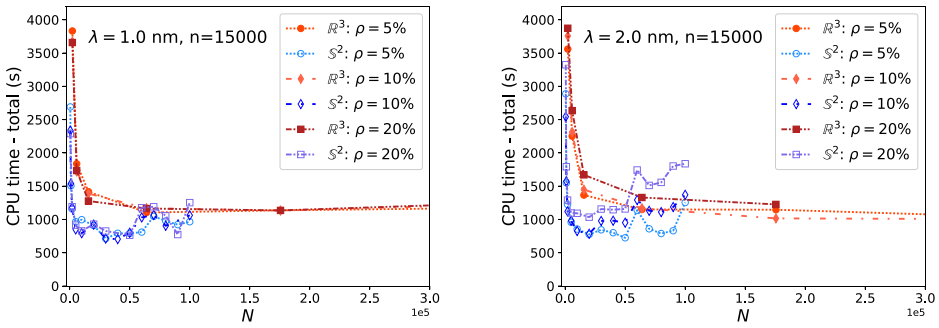
$$V(\mathbb{S}^{\mathrm{d}}(b)) = b^{d+1} V(\mathbb{S}^{\mathrm{d}}) = b^{d+1} \frac{\pi^{\frac{d+1}{2}}}{\Gamma\left(\frac{d+3}{2}\right)} \tag{35}$$

and $A(\mathbb{S}^{\mathrm{d}}(R))$ being the area of the d-sphere with radius R

$$A(\mathbb{S}^d(R)) = R^d A(\mathbb{S}^d) = R^d \frac{2\pi^{\frac{d+1}{2}}}{\Gamma\left(\frac{d+1}{2}\right)}, \tag{36}$$

with Γ being the gamma function.

In section 6, we present the results comparing the cpu time and memory requirement for a spherical linked list on $\mathbb{S}^2$ and $\mathbb{S}^3$ with a linked list in $\mathbb{R}^3$ and $\mathbb{R}^4$ respectively, using $n = 15\,000$ particles. Each simulation was run for 250 000 time steps and the results are averaged over ten trajectories. We also ran simulations with $n = 10\,000$ and $n = 20\,000$ particles which showed the same qualitative results as these presented in section 6. To test the implementation of the different algorithms, we also performed test simulations comparing the total number of inter-particle interactions for $\mathbb{S}^2$ and $\mathbb{R}^3$, and $\mathbb{S}^3$ and $\mathbb{R}^4$, respectively, also comparing with the total number of interactions when including all pairs of particles as a reference. The implementation of setting up the linked list was done using Python, while the BD simulation program was written in Fortran. All simulations were run on one core of an Intel Xeon V4 node. We further determined the diffusion coefficient and the radial distribution function, $g(r)$, for each set of parameters, comparing the results using a spherical linked list with those for $\mathbb{R}^3$ and $\mathbb{R}^4$ respectively, verifying the implementation of the algorithms. In the supplementary material, we illustrate the results showing some examples for both quantities.



(a) (b)

Figure 3. The total cpu time as a function of number of regions for $n = 15\,000$ particles moving on a 2-sphere for different volume fractions using a linked list for a cubic grid (denoted $\mathbb{R}^3$) and a spherical cell linked list (denoted $\mathbb{S}^2$) for an interaction distance (a) $\lambda = 1$ nm and (b) $\lambda = 2$ nm.

Table 1. The cpu time for sorting the particles into the corresponding regions and for the calculation of the inter-particle forces, using the corresponding linked list, at the minimum of the respective curve in figure 3. The total number of neighbour cells in each case, $\|\mathbf{M}\|$, is also shown.

System	N	ρ	λ (nm)	Sorting (s)	Interactions (s)	$\|\mathbf{M}\|$
$\mathbb{S}^2$	30 000	0.05	1.0	94.2	149.0	111 869
$\mathbb{S}^2$	40 000	0.1	1.0	95.4	140.8	166 520
$\mathbb{S}^2$	50 000	0.2	1.0	102.7	158.0	239 327
$\mathbb{S}^2$	50 000	0.05	2.0	95.2	160.9	238 907
$\mathbb{S}^2$	20 000	0.1	2.0	74.3	250.7	169 494
$\mathbb{S}^2$	20 000	0.2	2.0	75.0	383.9	219 585
$\mathbb{R}^3$	64 001	0.05	1.0	19.9	580.5	789 517
$\mathbb{R}^3$	175 617	0.1	1.0	31.3	573.2	2199 341
$\mathbb{R}^3$	175 617	0.2	1.0	31.0	575.0	2199 341
$\mathbb{R}^3$	512 001	0.05	2.0	39.6	447.6	6484 637
$\mathbb{R}^3$	512 001	0.1	2.0	39.4	480.9	6484 637
$\mathbb{R}^3$	175 617	0.2	2.0	26.6	689.6	2199 341

6. Results and discussion

6.1. Linked lists on $\mathbb{S}^2$ and in $\mathbb{R}^3$

In figure 3, the cpu time for 15 000 particles confined to move on a 2-sphere is shown as a function of the total number of regions for different volume fractions and for two different interaction distances, using both a spherical linked list on $\mathbb{S}^2$ as well as for a cubic grid in $\mathbb{R}^3$. There is a significant reduction in cpu time using a spherical linked list compared to using a three-dimensional network of cubic boxes. This is also seen in table 1 where the cpu time for sorting the particles into the cells and for calculating the inter-particle interactions at the minimum of the respective curve in figure 3, are shown. For short range interactions (figure 3(a)), the total cpu time is approximately constant after reaching the optimal number of

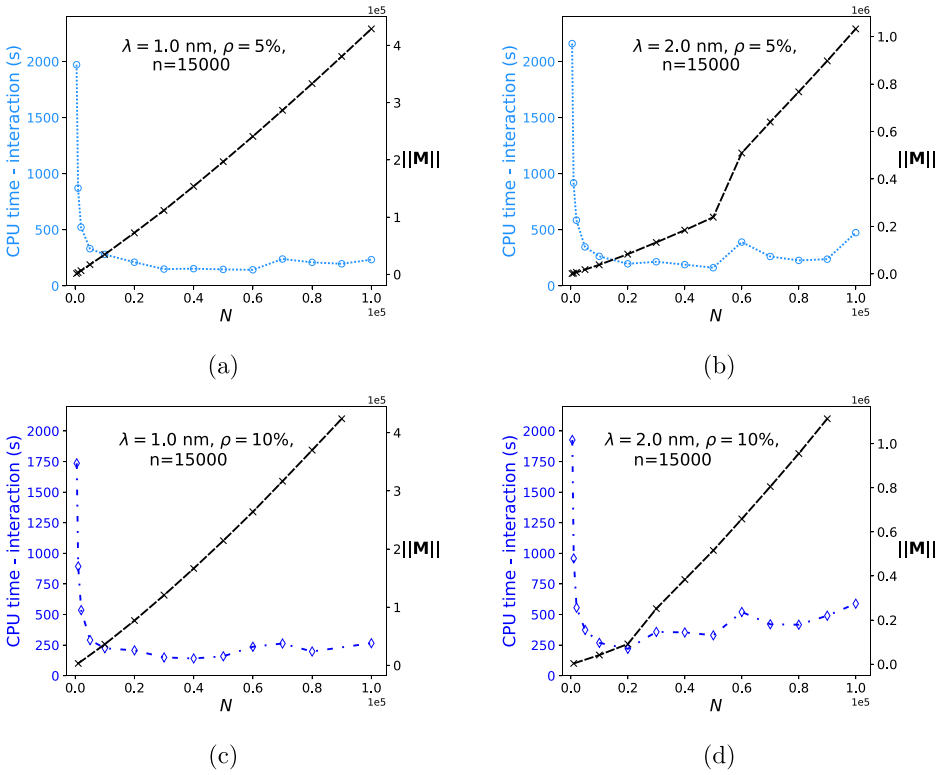


Figure 4. The cpu time for the inter-particle interactions (blue) and the total number of neighbour cells (black) as a function of number of regions for $\mathbb{S}^2$. Note the different scales on the vertical axes for $\|\mathbf{M}\|$, for $\lambda = 1$ nm and $\lambda = 2$ nm.

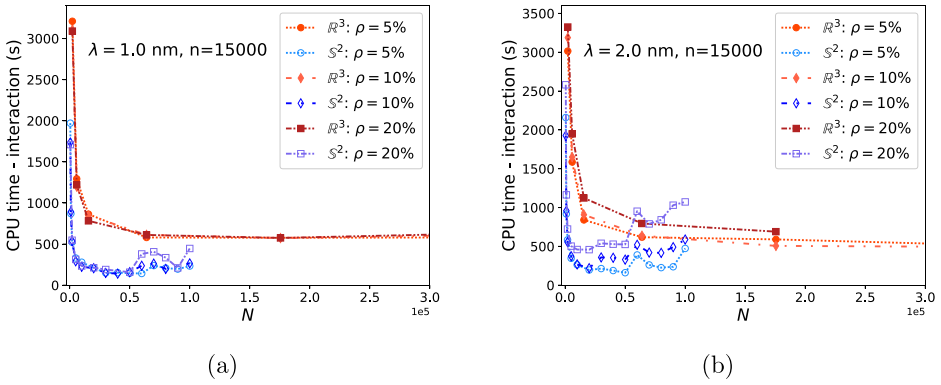
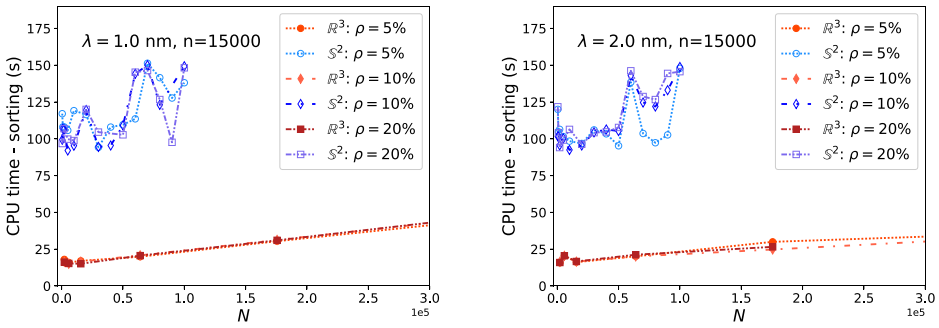


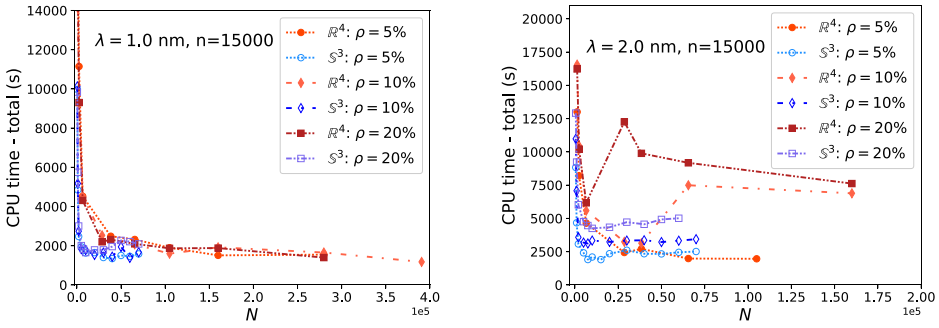
Figure 5. The cpu time for calculating the inter-particle interactions as a function of number of regions for 15 000 particles using a spherical linked list and a cubic grid for different volume fractions and (a) $\lambda = 1$ nm and (b) $\lambda = 2$ nm.

regions for a cubic grid, and the minimum is broad also for the spherical linked list, although there is a slight increase in cpu time in the latter case with increasing number of regions,



(a) (b)

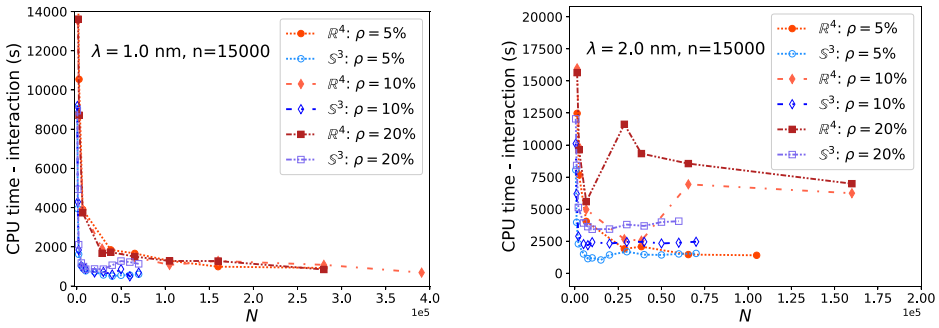
Figure 6. The cpu time for sorting the particles into the corresponding cells, as a function of number of regions. The notation is the same as in figure 5.



(a) (b)

Figure 7. The total cpu time as a function of number of regions for $n = 15\,000$ particles moving on a 3-sphere for different volume fractions using a linked list for a regular grid (denoted $\mathbb{R}^4$) and a spherical linked list (denoted $\mathbb{S}^3$) for (a) $\lambda = 1$ nm and (b) $\lambda = 2$ nm.

showing that more regions will be included as neighbours, when the size of the regions decreases. For a larger interaction distance (figure 3(b)), using a linked list directly on the spherical surface still results in a substantial reduction in cpu time, and in this case, we can also observe a jump in the corresponding curves. This is the result of also including a second neighbour collar when the size of the regions decreases with increasing N. The reason for this can be more clearly seen in figure 4 which shows the cpu time for calculating the inter-particle interactions and also the total number of neighbour cells for the spherical linked list as a function of the number of regions. For $\lambda = 1$ nm (effectively a soft excluded volume), the jump is smaller, while for a larger interaction distance, we can see the effect of also including a second neighbour collar. This, however, always occurs after the optimum number of regions and will thus not affect the cpu time in practical simulations. The same effect would eventually have been observed also for the cubic grid, if the size of the cells would decrease even further (see section 6.2).



(a) (b)

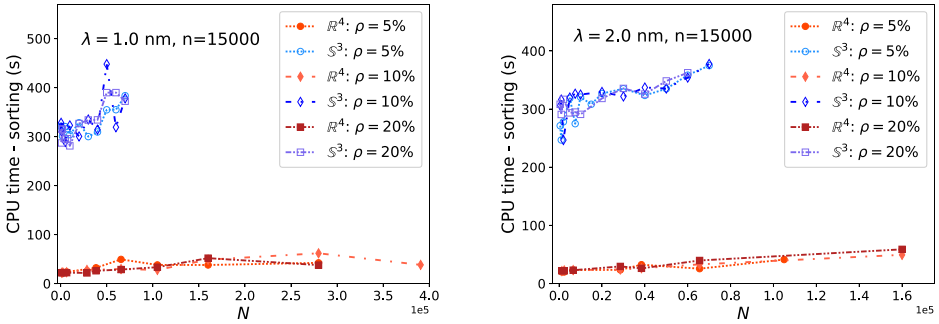
Figure 8. The cpu time for calculating the inter-particle interactions as a function of the number of regions for 15 000 particles using a spherical linked list and a regular grid for different volume fractions and (a) $\lambda = 1$ nm and (b) $\lambda = 2$ nm.

Table 2. Comparison of the cpu time for sorting the particles into the corresponding cells for a linked list on $\mathbb{S}^2$ and for a cubic grid in $\mathbb{R}^3$.

System	λ (nm)	ρ	Coordinate transf (s)	Sorting (s)	Total sorting (s)
$\mathbb{S}^2$	1.0	0.05	3647.0	3649.6	7296.6
$\mathbb{S}^2$	1.0	0.1	3799.2	3833.1	7632.3
$\mathbb{S}^2$	1.0	0.2	3814.2	3814.7	7629.0
$\mathbb{S}^2$	2.0	0.05	4080.0	4143.8	8223.8
$\mathbb{S}^2$	2.0	0.1	3972.4	3989.3	7961.7
$\mathbb{S}^2$	2.0	0.2	3662.5	3673.0	7335.5
$\mathbb{R}^3$	1.0	0.05		4013.0	4013.0
$\mathbb{R}^3$	1.0	0.1		3913.1	3913.1
$\mathbb{R}^3$	1.0	0.2		4117.9	4117.9
$\mathbb{R}^3$	2.0	0.05		3610.8	3610.8
$\mathbb{R}^3$	2.0	0.1		3735.1	3735.1
$\mathbb{R}^3$	2.0	0.2		3976.5	3976.5

The main contributions to the total cpu time is shown in more detail in figures 5 and 6. The time for calculating the inter-particle forces is shown in figure 5 and in table 1, and looking only at the time for evaluating the forces, using a spherical linked list is at least twice as fast or usually more, compared with the cubic grid at the respective optimum number of regions.

The other important contribution to the total cpu time is the sorting of particles into the appropriate regions before calculating the interactions. In figure 6 and in table 1, we can see that this appears to be faster for a cubic grid compared to a linked list on $\mathbb{S}^2$. Since the sorting procedure for a regular grid only contains modulus-operations, this result could be anticipated, even if one has to include an extra dimension. Looking at the numbers in a little more detail in table 2, however, shows that half of the cpu time for sorting the particles into the different regions for a spherical linked list is spent on converting the Cartesian coordinates of each particle to spherical polar coordinates, which are used to find the appropriate region for the particles (see section 4.4). Thus, implementing a simulation algorithm for $\mathbb{S}^2$ which uses



(a) (b)

Figure 9. The cpu time sorting the particles into the corresponding cells in the linked lists, as a function of the number of regions for $n = 15\,000$ particles. The notation is the same as in figure 8.

Table 3. The cpu time for sorting the particles into the corresponding regions and for the calculation of the inter-particle forces, using a spherical linked list on $\mathbb{S}^3$ and for a cubic grid in $\mathbb{R}^4$, at the minimum of the respective curves in figure 7. The total number of neighbour cells, $\|\mathbf{M}\|$, in each case is also shown.

System	N	ρ	λ (nm)	Sorting (s)	Interactions (s)	$\|\mathbf{M}\|$
$\mathbb{S}^3$	40 000	0.05	1.0	309.4	505.4	583 702
$\mathbb{S}^3$	60 000	0.1	1.0	319.2	493.2	1127 851
$\mathbb{S}^3$	10 000	0.2	1.0	281.1	861.4	144 566
$\mathbb{S}^3$	15 000	0.05	2.0	308.2	1034.5	291 557
$\mathbb{S}^3$	7500	0.1	2.0	326.4	2187.7	141 189
$\mathbb{S}^3$	10 000	0.2	2.0	290.9	3431.8	413 536
$\mathbb{R}^4$	160 000	0.05	1.0	37.9	988.4	5578 248
$\mathbb{R}^4$	390 625	0.1	1.0	38.1	684.5	14 003 808
$\mathbb{R}^4$	279 841	0.2	1.0	37.7	844.0	9935 640
$\mathbb{R}^4$	104 976	0.05	2.0	41.3	1397.7	3603 320
$\mathbb{R}^4$	38 416	0.1	2.0	26.4	2541.1	1260 792
$\mathbb{R}^4$	6561	0.2	2.0	23.0	5584.5	192 032

spherical polar coordinates for the particle positions would result in a slightly faster sorting for a spherical linked list, even compared to a regular grid in $\mathbb{R}^3$. We have not implemented this in the present version, since we wanted to compare, in detail, the different contributions to the total cpu time, and also, for comparison, to use the algorithm previously described [50] for diffusion on $\mathbb{S}^2$. (The reason for the cpu times being larger in table 2 compared with those in table 1 is due to that the timing routines in the former case are inside the loop used to calculate the forces.)

We can also note that the reason for the curves for the spherical linked list in figures 3–6 having more points compared to the graphs showing the results for $\mathbb{R}^3$ is that when constructing the linked list for a cubic grid, the number of cells per dimension, n_l, determines the total number of cells (see appendix B), while the number of regions for $\mathbb{S}^2$ can be chosen arbitrarily when partitioning the sphere into equal area regions.

Table 4. Comparison of the cpu time for sorting the particles into the corresponding cells for a linked list on $\mathbb{S}^3$ and a cubic grid in $\mathbb{R}^4$.

System	λ (nm)	ρ	Coordinate transf (s)	Sorting (s)	Total sorting (s)
$\mathbb{S}^3$	1.01	0.05	4118.3	4144.1	8262.4
$\mathbb{S}^3$	1.01	0.1	3493.1	3489.6	6982.7
$\mathbb{S}^3$	1.01	0.2	3446.1	3414.3	6860.4
$\mathbb{S}^3$	2.01	0.05	3329.0	3315.2	6644.2
$\mathbb{S}^3$	2.01	0.1	3654.0	3622.6	7276.6
$\mathbb{S}^3$	2.01	0.2	3632.8	3602.0	7234.8
$\mathbb{R}^4$	1.01	0.05		3396.6	3396.6
$\mathbb{R}^4$	1.01	0.1		3584.2	3584.2
$\mathbb{R}^4$	1.01	0.2		3617.0	3617.0
$\mathbb{R}^4$	2.01	0.05		3411.4	3411.4
$\mathbb{R}^4$	2.01	0.1		3480.6	3480.6

Another, at least as important feature when using a spherical link list, is the memory allocation. The list of neighbour cells, M, (algorithm 5) requires considerably less memory compared to a cubic grid in $\mathbb{R}^3$ (see table 1 and appendix B). The difference in memory requirement becomes even more relevant for simulations of larger systems, as the optimal point for the cpu time will require the use of larger values of N, and consequently, a larger value for the total number of neighbour cells.

6.2. *Linked lists on $\mathbb{S}^3$ and in $\mathbb{R}^4$*

We also compared the cpu time and memory requirement for simulations of bulk systems using spherical boundary conditions with a linked list directly on $\mathbb{S}^3$, compared with a cubic grid in $\mathbb{R}^4$. In figure 7 and table 3, it is seen that using a linked list directly on $\mathbb{S}^3$ is in general more efficient, but the improvement is not as large as for particles moving on $\mathbb{S}^2$. Making a similar analysis as for $\mathbb{S}^2$, figure 8 shows the cpu time for the inter-particle interactions as a function of number of regions, and for $\lambda = 1$ nm in figure 8(a), we can see that the spherical linked list is still approximately twice as fast compared to a cubic grid for $\rho = 0.05$, while being about the same for $\rho = 0.2$, at the optimal number of regions (see table 3). For $\lambda = 2$ nm, the spherical linked list is still in general considerably faster for the calculation of the inter-particle forces.

We further observe in figures 7 and 8 that for $\mathbb{S}^3$, the cpu time associated with both types of linked lists shows jumps which are now also more pronounced for the cubic grid in $\mathbb{R}^4$. We have already seen for $\mathbb{S}^2$ that the jumps are associated with including a second neighbour collar for the spherical linked list and here, a second layer of neighbour cells for the regular grid is also needed for a larger number of cells.

In figure 9, the time for sorting the particles into the appropriate regions is shown, and the difference is in this case larger compared to figure 6 and table 2. This can partly be explained by the map χ being used twice, calculating the indexes K and J (see section 4.4). A similar analysis as for the 2-sphere shows, however, that also in this case, the time to sort the particles into the different regions is about the same for both $\mathbb{S}^3$ and $\mathbb{R}^4$, (see table 4). The time for converting the Cartesian coordinates to spherical polar coordinates is, however, almost the same as for the actual sorting of the particles. Thus, using spherical polar coordinates in the simulations would again amplify the efficiency of using a linked list directly on $\mathbb{S}^3$ compared with a cubic network in $\mathbb{R}^4$. An important feature for the spherical linked list, which is even more pronounced for

$\mathbb{S}^3$ compared to particles on the 2-sphere, is also the difference in memory requirement. Using a spherical linked list requires in most cases significantly less memory compared to a regular cubic grid as seen in table 3, which can be a limiting factor for large systems. The special case for $\lambda = 2$ nm and $\rho = 0.2$ for $\mathbb{R}^4$ (see also figure 8), which also shows a large jump in the cpu time, is due to including a second layer of neighbours cells for a small value of N.

7. Conclusions

To improve both the efficiency and reducing the memory requirement when using a cell linked list to evaluate the inter-particle interactions in simulations of particles constrained to move on a spherical surface (the 2-sphere), we show how to set up a linked list directly on the spherical surface, instead of using a cubic grid in $\mathbb{R}^3$. The result is a significant decrease both in cpu time and, not least, the memory requirement, compared with using a cubic simulation box.

We also describe the extension of this algorithm to the 3-sphere, using spherical boundary conditions for bulk simulations, which for many systems can be a useful alternative to periodic boundary conditions. In this case, the decrease in memory requirement was, in general, even larger, and there is still a significant improvement in cpu time, compared with a cubic grid in $\mathbb{R}^4$, even if the decrease is somewhat smaller, compared with particles moving on the 2-sphere.

Analysing the contribution to the total cpu time in more detail for both $\mathbb{S}^2$ and $\mathbb{S}^3$, an even larger improvement in cpu time for the spherical linked list could be obtained by performing the simulations using spherical polar coordinates, avoiding the coordinate transformations from Cartesian coordinates when sorting the particles into the respective regions.

A spherical linked list can also be used when particles are constrained to move on part of $\mathbb{S}^2$ or $\mathbb{S}^3$, as it can be adapted to only include the corresponding area (or volume). This feature will be used in future simulations of confined electrolyte solutions.

Acknowledgments

CE acknowledges computer resources in the Project SNIC 2020/15-344 provided by the Swedish National Infrastructure for Computing (SNIC) at UPPMAX. EVR acknowledges financial support from the Centre for Interdisciplinary Mathematics at Uppsala University. T Ekholm is acknowledged for fruitful discussions.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Appendix A. Proof of lemma

Proof of lemma. 2.1. We consider two points $\mathbf{a} = \left(\theta^a_{d-1}, \ldots, \theta^a_1, \phi^a\right)$ and $\mathbf{b} = \left(\theta^b_{d-1}, \ldots, \theta^b_1, \phi^b\right) \in \mathbb{S}^d$. Employing the definition of a spherical distance, we have $s = \gamma(\mathbf{a}, \mathbf{b}) = \cos^{-1}(\mathbf{a}, \mathbf{b})$

$$
\begin{aligned}
\cos\, s &= \mathbf{a}\cdot\mathbf{b} \\
&= \cos\,\phi^a\,\cos\,\phi^b\prod_{j=1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_j^b + \sin\,\phi^a\,\sin\,\phi^b\prod_{j=1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_j^b \\
&\quad + \sum_{k=1}^{d-1}\cos\,\theta_k^a\,\cos\,\theta_k^b\prod_{j=k+1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_j^b, \\
&= (\cos\,\phi^a\,\cos\,\phi^b + \sin\,\phi^a\,\sin\,\phi^b)\prod_{j=1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_j^b \\
&\quad + \sum_{k=1}^{d-1}\cos\,\theta_k^a\,\cos\,\theta_k^b\prod_{j=k+1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_j^b, \\
&= \cos(\phi^a-\phi^b)\prod_{j=1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_j^b + \sum_{k=1}^{d-1}\cos\,\theta_k^a\,\cos\,\theta_k^b\prod_{j=k+1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_j^b.
\end{aligned}
$$

Using that $\cos(\phi^a-\phi^b)=\cos(\phi^b-\phi^a)$ and solving for ϕ^b we obtain

$$
\phi^b = \begin{cases} \phi^a + \cos^{-1}\left(\dfrac{\cos\,s - \sum_{k=1}^{d-1}\cos\,\theta_k^a\,\cos\,\theta_k^b\prod_{j=k+1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_j^b}{\prod_{k=1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_k^b}\right), \\ \phi^a - \cos^{-1}\left(\dfrac{\cos\,s - \sum_{k=1}^{d-1}\cos\,\theta_k^a\,\cos\,\theta_k^b\prod_{j=k+1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_j^b}{\prod_{k=1}^{d-1}\sin\,\theta_j^a\,\sin\,\theta_k^b}\right). \end{cases} \tag{A.1}
$$

Appendix B. A linked list in $\mathbb{R}^d$ without periodic boundaries

When constructing a linked list using an cubic grid in $\mathbb{R}^d$ consisting of only first neighbour cells and assuming periodic boundary conditions, it is a rather straightforward procedure to estimate the memory requirement by calculating the total number of neighbour cells. The number of neighbour cells, m', to every region is constant, where a primed quantity denotes a system with periodic boundaries, and is given by

$$
m' = \frac{3^d-1}{2}, \tag{B.1}
$$

where the term 3 is the number of first neighbour cells in each dimension, and the factor $1/2$ is introduced from Newton's third law in order not to count interactions twice. Therefore, if n_l' represents the number of cells per dimension, the total number of neighbour cells, M', can be calculated using

$$
M' = m'(n_l')^d = \frac{(n_l')^d(3^d-1)}{2}. \tag{B.2}
$$

Setting up a corresponding linked list as a cubic grid in $\mathbb{R}^{\mathrm{d}}$ for particles confined to move on $\mathbb{S}^{\mathrm{d}-1}$, periodic boundaries need, however, not be used. To estimate the memory requirement, the total number of (first) neighbour cells can still be computed. Equation (B.1) can in this case

Table B1. Number of neighbour cells for *interior regions* in the hyper-cubic grid.

d	a	b	h_1	h_2	h_3	h_4	m
2			1	3			4
3	0	2	1	3			13
4	0	2	1	3	3		40
5	0	2	1	3	3	3	121

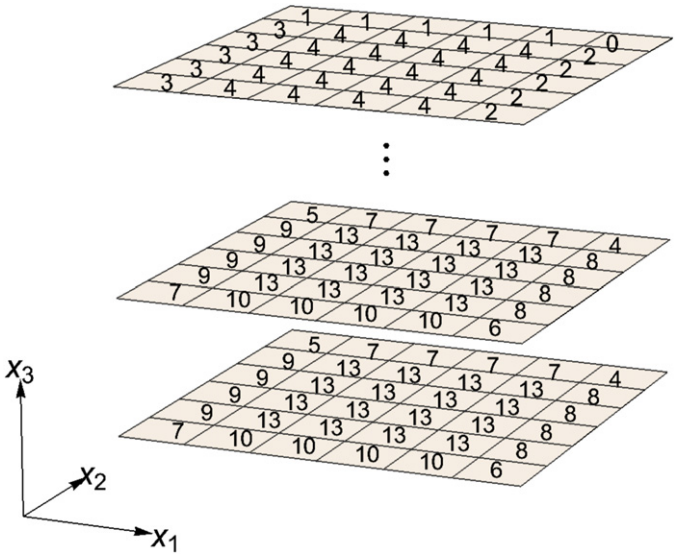


Figure B.1. Illustration of how the number of neighbour cells per region depends on its location in the simulation grid for $\mathbb{R}^3$.

be generalized to read

$$m_i = h_1 + h_2 + 2^a 3^b + h_3 2^a 3^b + \cdots + h_{d-1}^{d-3} 2^a 3^b \tag{B.3}$$

with m_i being the number of neighbour cells for the region $\mathcal{R}_i$. The terms 2 and 3 are introduced since one needs to include two or three cells per dimension, depending on where the cell is located in the (hyper)cubic grid. The exponents $a, b \in \{0, 1, 2\}$ are the number of times two and three cells are considered per dimension. The terms h_j refer to the number of neighbour cells that must be considered along direction x_j, in general $h_1 \in \{0, 1\}$ and $h_{j>1} \in \{2, 3\}$. The number of neighbour cells, m_i, can now be computed also for regions which belong to the boundary of the grid. Table B1 shows how the number of neighbour cells for *interior regions* of the grid can be computed using equation (B.3). It is easy to verify that the same values would have been obtained if equation (B.1) had been used instead.

As an example, figure B.1 shows the result of using equation (B.3), with an appropriate set of numbers $\{a, b, h_1, h_2\}$, to find the number of neighbour cells per region for a non-periodic cubic grid in $\mathbb{R}^3$ with number of cells per dimension $n_l = 6$. For a general n_l, an expression for the total number of neighbour cells, M, can be calculated by adding all the contributions, m_i,

from all the regions in the grid, with the result

$$M_{\mathbb{R}^3} = 13(n_l - 2)^2(n_l - 1) + 38(n_l - 2)(n_l - 1) + 28(n_l - 1). \tag{B.4}$$

By following a similar procedure, an expression for the total number of neighbour cells, M, in $\mathbb{R}^4$ is given by

$$M_{\mathbb{R}^4} = 40(n_l - 2)^3(n_l - 1) + 172(n_l - 2)^2(n_l - 1) + 248(n_l - 2)(n_l - 1) + 120(n_l - 1). \tag{B.5}$$

ORCID iDs

Christer Elvingson https://orcid.org/0000-0001-5115-5481

References

[1] Krishna M M G, Das R, Periasamy N and Nityananda R 2000 Translational diffusion of fluorescent probes on a sphere: Monte Carlo simulations, theory, and fluorescence anisotropy experiment *J. Chem. Phys.* **112** 8502–14
[2] Weiss M, Hashimoto H and Nilsson T 2003 Anomalous protein diffusion in living cells as seen by fluorescence correlation spectroscopy *Biophys. J.* **84** 4043–52
[3] Baptista D, Teixeira L, van Blitterswijk C, Giselbrecht S and Truckenmüller R 2019 Overlooked? Underestimated? Effects of substrate curvature on cell behavior *Trends Biotechnol.* **37** 838–54
[4] Jaskolski F and Henley J M 2009 Synaptic receptor trafficking: the lateral point of view *Neuroscience* **158** 19–24
[5] Li G, Tam L-K and Tang J X 2008 Amplified effect of Brownian motion in bacterial near-surface swimming *Proc. Natl Acad. Sci. USA* **105** 18355–9
[6] Fei W, Driscoll M M, Chaikin P M and Bishop K J M 2018 Magneto-capillary dynamics of amphiphilic Janus particles at curved liquid interfaces *Soft Matter* **14** 4661–5
[7] Bruss I R and Glotzer S C 2017 Curvature-induced microswarming *Soft Matter* **13** 5117–21
[8] Keber F C, Loiseau E, Sanchez T, DeCamp S J, Giomi L, Bowick M J, Marchetti M C, Dogic Z and Bausch A R 2014 Topology and dynamics of active nematic vesicles *Science* **345** 1135–9
[9] Ai B-q, Zhou B-y and Zhang X-m 2020 Binary mixtures of active and passive particles on a sphere *Soft Matter* **16** 4710–7
[10] Großmann R, Peruani F and Bär M 2015 A geometric approach to self-propelled motion in isotropic & anisotropic environments *Eur. Phys. J. Spec. Top.* **224** 1377–94
[11] Brillinger D R 1997 A particle migrating randomly on a sphere *J. Theor. Probab.* **10** 429–43
[12] Allen M P and Tildesley D J 1987 *Computer Simulation of Liquids* (Oxford: Oxford University Press)
[13] Frenkel D and Smit B 2002 *Understanding Molecular Simulation* 2nd edn (New York: Academic)
[14] Verlet L 1967 Computer 'experiments' on classical fluids: I. Thermodynamical properties of Lennard–Jones molecules *Phys. Rev.* **159** 98–103
[15] Quentrec B and Brot C 1973 New method for searching for neighbors in molecular dynamics computations *J. Comput. Phys.* **13** 430–2
[16] Hockney R W and Eastwood J W 1988 *Computer Simulation Using Particles* (Bristol: IOP Publishing)
[17] Fomin E S 2011 Consideration of data load time on modern processors for the Verlet table and linked-cell algorithms *J. Comput. Chem.* **32** 1386–99
[18] Auerbach D J, Paul W, Bakker A F, Lutz C, Rudge W E and Abraham F F 1987 A special purpose parallel computer for molecular dynamics: motivation, design, implementation, and application *J. Phys. Chem.* **91** 4881–90
[19] Mazzeo M D, Ricci M and Zannoni C 2010 The linked neighbour list (LNL) method for fast off-lattice Monte Carlo simulations of fluids *Comput. Phys. Commun.* **181** 569–81

[20] Gonnet P 2013 Pseudo-Verlet lists: a new, compact neighbour list representation *Mol. Simul.* **39** 721–7
[21] Mattson W and Rice B M 1999 Near-neighbor calculations using a modified cell-linked list method *Comput. Phys. Commun.* **119** 135–48
[22] Sutmann G and Stegailov V 2006 Optimization of neighbor list techniques in liquid matter simulations *J. Mol. Liq.* **125** 197–203
[23] Yao Z, Wang J-S, Liu G-R and Cheng M 2004 Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method *Comput. Phys. Commun.* **161** 27–35
[24] Meloni S, Rosati M and Colombo L 2007 Efficient particle labeling in atomistic simulations *J. Chem. Phys.* **126** 121102
[25] Luo J and Liu L 2017 Optimisation of data locality in energy calculations for large-scale molecular dynamics simulations *Mol. Simul.* **43** 284–90
[26] Wang D-B, Hsiao F-B, Chuang C-H and Lee Y-C 2007 Algorithm optimization in molecular dynamics simulation *Comput. Phys. Commun.* **177** 551–9
[27] Welling U and Germano G 2011 Efficiency of linked cell algorithms *Comput. Phys. Commun.* **182** 611–5
[28] Li W-Q, Ying T, Jian W and Yu D-J 2010 Comparison research on the neighbor list algorithms: Verlet table and linked-cell *Comput. Phys. Commun.* **181** 1682–6
[29] Awile O, Büyükkeçeci F, Reboux S and Sbalzarini I F 2012 Fast neighbor lists for adaptive-resolution particle simulations *Comput. Phys. Commun.* **183** 1073–81
[30] Domínguez J M, Crespo A J C, Gómez-Gesteira M and Marongiu J C 2011 Neighbour lists in smoothed particle hydrodynamics *Int. J. Numer. Methods Fluids* **67** 2026–42
[31] Khorasanizade S and Sousa J M M 2019 Improving linked-lists using tree search algorithms for neighbor finding in variable-resolution smoothed particle hydrodynamics *Commun. Comput. Phys.* **26** 57–86
[32] Kratky K W 1980 New boundary conditions for computer experiments of thermodynamic systems *J. Comput. Phys.* **37** 205–17
[33] Fanti L A and Glandt E D 1989 Monte Carlo simulation of fluids in curved three-dimensional space *Mol. Simul.* **2** 163–76
[34] Caillol J M and Levesque D 1991 Numerical simulations of homogeneous and inhomogeneous ionic systems: an efficient alternative to the Ewald method *J. Chem. Phys.* **94** 597–607
[35] Caillol J M 1993 A new potential for the numerical simulations of electrolyte solutions on a hypersphere *J. Chem. Phys.* **99** 8953–63
[36] Caillol J M 1999 Numerical simulations of Coulomb systems: a comparison between hyperspherical and periodic boundary conditions *J. Chem. Phys.* **111** 6528–37
[37] Delville A, Pellenq R J-M and Caillol J M 1997 A Monte Carlo (N, V, T) study of the stability of charged interfaces: a simulation on a hypersphere *J. Chem. Phys.* **106** 7275–85
[38] Trulsson M 2010 Simulations of high-dielectric Stockmayer fluids in hyperspherical geometry *J. Chem. Phys.* **133** 174105
[39] Allahyarov E, D'Amico I and Löwen H 1999 Effect of geometrical confinement on the interaction between charged colloidal suspensions *Phys. Rev.* E **60** 3199–210
[40] Nissfolk J, Ekholm T and Elvingson C 2003 Brownian dynamics simulations on a hypersphere in four-space *J. Chem. Phys.* **119** 6423–32
[41] Råsmark P J, Ekholm T and Elvingson C 2005 Computer simulations of polymer chain structure and dynamics on a hypersphere in four-space *J. Chem. Phys.* **122** 184110
[42] Kamerlin N, Ekholm T, Carlsson T and Elvingson C 2014 Construction of a closed polymer network for computer simulations *J. Chem. Phys.* **141** 154113
[43] Hünenberger P H and McCammon J A 1999 Effect of artificial periodicity in simulations of biomolecules under Ewald boundary conditions: a continuum electrostatics study *Biophys. Chem.* **78** 69–88
[44] Hub J S, de Groot B L, Grubmüller H and Groenhof G 2014 Quantifying artifacts in Ewald simulations of inhomogeneous systems with a net charge *J. Chem. Theory Comput.* **10** 381–90
[45] Yeh I-C and Hummer G 2004 System-size dependence of diffusion coefficients and viscosities from molecular dynamics simulations with periodic boundary conditions *J. Phys. Chem.* B **108** 15873–9
[46] Hanassab S and VanderNoot T J 2003 Spherical boundary conditions: a finite and system size independent geometry for simulations of electrolytic liquids *Mol. Simul.* **29** 527–33

[47] Leopardi P 2006 A partition of the unit sphere into regions of equal area and small diameter *Electron. Trans. Numer. Anal.* **25** 309–27
[48] Blumenson L E 1960 A derivation of *n*-dimensional spherical coordinates *Am. Math. Mon.* **67** 63–6
[49] Elvingson C 1991 A general Brownian dynamics simulation program for biopolymer dynamics and its implementation on a vector computer *J. Comput. Chem.* **12** 71–7
[50] Carlsson T, Ekholm T and Elvingson C 2010 Algorithm for generating a Brownian motion on a sphere *J. Phys. A: Math. Theor.* **43** 505001
[51] Ghosh A, Samuel J and Sinha S 2012 A 'Gaussian' for diffusion on the sphere *Europhys. Lett.* **98** 30003
[52] Mijatović A, Mramor V and Uribe Bravo G 2020 A note on the exact simulation of spherical Brownian motion *Stat. Probab. Lett.* **165** 108836
[53] Castro-Villarreal P, Villada-Balbuena A, Méndez-Alcaraz J M, Castañeda-Priego R and Estrada-Jiménez S 2014 A Brownian dynamics algorithm for colloids in curved manifolds *J. Chem. Phys.* **140** 214115
[54] Müller C 1998 *Analysis of Spherical Symmetries in Euclidean Spaces* (Berlin: Springer)