Journal of Molecular Graphics and Modelling 68 (2016) 197–205

Contents lists available at ScienceDirect

Journal of Molecular Graphics and Modelling

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

Overcoming the minimum image constraint using the closest point search

David M. Rogers

University of South Florida, 4202 E. Fowler Ave., CHE 205, Tampa, FL 33620, United States

ARTICLE INFO

Article history:
Received 15 April 2016
Received in revised form 20 June 2016
Accepted 17 July 2016
Available online 21 July 2016

Keywords:
Neighbor list
Lattice summation
Molecular dynamics
Finite size effects

ABSTRACT

Finding the set of nearest images of a point in a simulation cell with periodic (torus) boundary conditions is of central importance for molecular dynamics algorithms. To compute all pairwise distances closer than a given cutoff in linear time requires region-based neighbor-listing algorithms. Available algorithms encounter increasing difficulties when the cutoff distance exceeds half the shortest cell length. This work provides details on two ways to directly and efficiently generate region–region interaction lists in n-dimensional space, free from the minimum image restriction. The solution is based on a refined version of existing algorithms solving the closest vector problem. A self-contained discussion of lattice reduction methods for efficient higher-dimensional searches is also provided. In the MD setting, these reduction criteria provide useful guidelines for lattice compaction.

1. Introduction

The use of periodic boundary conditions is common in molecular dynamics simulations of sets of 3D coordinates. Most often, a central molecule of interest is surrounded by a buffer region composed of solvent or vacuum. As an optimization, simulations often use non-rectangular cell shapes. As pointed out in Ref. [1], the smallest triclinic unit cell shape is a rhombic dodecahedron, which provides the same buffer distance between periodic images but reduces the volume and hence the simulation work by 29%. Even using a hexagonal prism unit cell reduces the work by 14%. General unit cell shapes are also commonly realized as output from molecular crystallography and used in simulations of material properties.

One of the key operations used repeatedly throughout setup, simulation, and analysis is finding all periodic images of a given atom within a certain distance, R, from a central atom of interest.[1] All recent, efficient MD codes use a 2-step method for solving this problem in $O(N)$ time, where N is the number of atoms in the simulation. In the first step, atoms are assigned to unique regions tiling the simulation volume. This step is usually referred to as the linked cell method [3]. In the second step, distance tests are performed between atoms within pairs of regions, either to directly compute forces [4] or to construct Verlet, atom-atom neighborlists for force computation [5,6].

The linear time performance relies on checking only pairs of regions that contain points separated by less than a cutoff, R. This work refers to such pairs as 'interacting' regions. When the linked cell method was originally introduced [3], Verlet atom–atom neighborlists were briefly discarded because the interacting cells ensured $O(N)$ performance. However, the significant difference in volume between a cell-shaped cube and an inscribed sphere motivated combined methods [7]. As message-passing parallel methods were implemented, the cube-shaped cells began to take on greater importance for spatial distribution of atoms across processors [5,8]. There, finding effective opportunities for parallelization depends critically on decreasing the size of these regions.

The early literature deals almost exclusively with the case where the regions are larger than the cutoff distance, R. The reason is that smaller regions would extend the range of interaction from a central cell further than its 26 nearest neighbors in 3D space. In a message-passing parallel implementation where each processor is responsible for atoms in a single region, the interacting cells make up the potential 'import' volume of the cell. Recently, it has been noted how shifting the assignment of pair-calculations to different processors can decrease the range of the import volume to a cube extended by only $R/2$ [9]. In another work, a vector-parallelization was shown that greatly improved parallel efficiency using very small regions, containing only 4-atoms each [4]. These small regions have to solve the new and challenging problem of how to efficiently find interacting region pairs. Ref. [6] implemented domain decomposition for general cells with dynamically changing cell and region sizes. They noted the difficulties with implementing these methods in non-cubic geometries, stating that

E-mail address: davidrogers@usf.edu

[1] The set of periodic images of the position $r_i \in \mathbb{R}^3$ is defined as the set of all translations of r_i by lattice vectors (Eq. (2)).

http://dx.doi.org/10.1016/j.jmgm.2016.07.004
1093-3263/

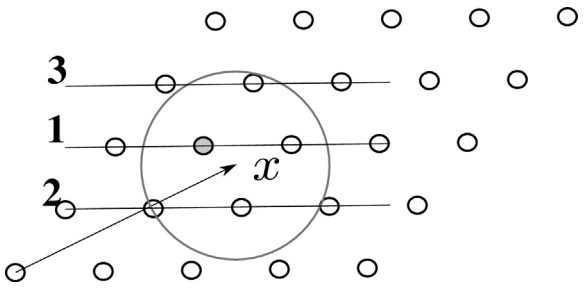

Fig. 1. Illustration of the search for the minimal cell containing position $x \in \mathbb{R}^2$ (shown by the vector). Every small circle indicates a periodic image of the origin. The large circle (radius R) contains all relevant images of the origin, while the minimal cell can be identified with the nearest lattice point to x (shaded). The minimum image problem is thus solved by enumerating all the lattice points inside a sphere of radius R. The numbered rows show the order of trial searches.

despite extensive bookkeeping, "GROMACS 4 does not (yet) take full advantage of the reduction in the communication due to rounding of the zone."

This work contributes toward more robust and efficient methods for domain decomposition by presenting an algorithm proven to find all pairs of interacting regions for any general R and convex region shape. It is not limited in any way by the ratio of domain size to the cutoff distance, R. This is significant, since the historical algorithms have all been formulated with reference to the minimum image constraint. For this constraint, the regions cannot have a lattice vector smaller than R, and the overall simulation cell cannot have a lattice vector smaller than $2R$.

In its place, we simply use a robust algorithm for finding all pairs of interacting regions. Because the region sizes can be very small, these interacting cell-lists can be efficiently used for direct force calculation [4,7]. This procedure also removes the barrier imposed by the minimum image constraint, and can enable study of a broad class of numerical applications involving large cutoffs, R, relative to the lattice size. This work will show how the interacting regions problem can be solved by carefully considering the closest vector problem.

1.1. Formulation and background

This section introduces the notation and formulation of the closest vector problem. Our final algorithms (Fig. 2) for finding pairs of interacting regions are all based on the solution of this simpler problem.

Six degrees of freedom specify a general unit cell shape in 3D through specifying the directions of three lattice vectors ($\vec{a}$, $\vec{b}$, and $\vec{c}$). Collecting those vectors into a column matrix,

$$L^T = \left[\begin{array}{c|c|c} a_x & b_x & c_x \\ & b_y & c_y \\ & & c_z \end{array}\right] \tag{1}$$

we can define the set of lattice points (small circles in Fig. 1) as,

$$\mathcal{V} = \{L^T n : n \in \mathbb{Z}^3\}. \tag{2}$$

For each cell shape, there exists a minimal unit cell composed of all the points closest to the origin, modulo translation symmetries:

$$\mathcal{C} = \{x : |x| \leq |x - v| \quad \forall v \in \mathcal{V}\}. \tag{3}$$

Mathematically, this says that for x to be in $\mathcal{C}$, no lattice translation can bring x closer to the origin than it already is. This is the specification for the Voronoi region [10] or the Wigner–Seitz primitive cell (or Brillouin zone if the lattice vectors represent the reciprocal space) [11].

It is not trivial to determine whether a given x lies within the minimal cell, $\mathcal{C}$. The obvious method for checking is to try all lattice vectors and determine which will bring x closest to the origin. Or, identically, to determine which lattice point lies nearest to x (Fig. 1). Since the set of relevant lattice vectors are those nearest x, the search can stop once the lattice points get too far away. Our final algorithms can all be visualized using this geometrical picture of trying all lattice points inside a sphere centered at x.

A general problem in periodic simulations is wrapping 3D atomic coordinates into the Voronoi region, $\mathcal{C}$. Wrapping is formally accomplished by finding the closest lattice point. Formally, this is a set of 3 integers, n, which minimize

$$R^2 = \min_{n \in \mathbb{Z}^3} |x - L^T n|^2 \tag{4}$$

A solution to the wrapping problem will also solve the decision problem above. If x is in the minimal cell, then the distance from x to the origin is equal to the distance from x to its nearest lattice point (here, R).

The answer to the wrapping problem also gives the closest periodic distance between two points, r_i and r_j, in a unit cell. To see this, set $x = r_i - r_j$, and solve for the smallest $|x - L^T n|$. The same n then gives the periodic image of j, $r_j + L^T n$ that minimizes the distance, $|r_i - (r_j + L^T n)|$.

Aside from molecular visualization, the problem of wrapping a coordinate into the Voronoi region has not attracted much attention in the MD community. For example, Gromacs [12] [2] and VMD [14][3] currently implement minimal cell wrapping using a search over a few of the nearest translation vectors.

One brute force solution would be to find the faces of the minimal cell analytically from a Voronoi procedure [16,10,11], and repeatedly move a point until it lies on the interior of all (14) faces. This type of approach is used by some codes developed for solving molecular crystal structures [17].[4] While repeated reflection is not guaranteed to converge after a fixed number of attempts, it is helpful in that context because crystal structures often contain additional symmetry operations besides lattice translation.

A re-phrasing of the same problem is encountered in the signal processing literature as,

$$\min_{n \in \mathbb{Z}^d} |y - n|_Q^2 \equiv (y - n)^T Q (y - n), \tag{5}$$

where y is termed the received vector and Q is a positive-definite metric matrix. This problem can be cast into the form of Eq. (4) using the Cholesky decomposition of $Q = LL^T$,

$$\min_{n \in \mathbb{Z}^d} (y - n)^T L L^T (y - n) = |x - L^T n|^2 \tag{6}$$

where $x = L^T y$.

Eq. (4) is itself a special case of integer quadratic programming, and has been extensively studied as the closest point problem [10]. Defining the problem size to be the dimensionality of each vector, even the decision problem is known to be NP-hard [19]. Its solution has applications across many areas, including high-accuracy GPS geolocation [20,21], communication theory, and integer factorization [19]. In communication theory it is known as sphere decoding [22,23]. There, a message is encoded as a vector of integers, in $\mathbb{Z}^n$. It acquires random noise when sent over a real communication line,

[2] In Gromacs version 5.0.5, compact cell wrapping uses `pbc_dx` from pbc.c:672. This routine covers multiple special cases, including rhombic dodecahedron and truncated octahedron by testing translations to a set of up to 14 nearest neighbors.

[3] The pbctools package only supports wrapping into a parallelepiped or 'brick' produced by carrying out one shift per lattice vector.

[4] Clipper, distributed in CCP4 (released 2015-09-30) implements this method in `Coord_frac::symmetry_copy_near` in coords.cpp, line 331.