A Simple Algorithm to Accelerate the Computation of Non-Bonded Interactions in Cell-Based Molecular Dynamics Simulations

PEDRO GONNET
Institute of Computational Science, ETH Zürich, 8092 Zürich, Switzerland

Received 17 January 2006; Revised 1 March 2006; Accepted 9 March 2006
DOI 10.1002/jcc.20563
Published online 21 December 2006 in Wiley InterScience (www.interscience.wiley.com).

Abstract: Cell lists are ubiquitous in molecular dynamics simulations—be it for the direct computation of short-range inter-atomic potentials, the short-range direct part of a long-range interaction or for the periodic construction of Verlet lists. The conventional approach to computing pairwise interactions using cell lists leads to a large number of unnecessary interparticle distance calculations. In this paper, an algorithm is presented which reduces the number of spurious distance calculations by first sorting the particles along the cell pair axis and then only interacting two particles if their distance along the axis is smaller than the cutoff distance of the interaction. This approach is shown to be more efficient than the conventional approach and similar approaches using smaller cells.

J Comput Chem 28: 570–573, 2007

Key words: molecular dynamics; cell list; neighbor list

Cell lists (or cell linked lists)[1] are ubiquitous in molecular dynamics or N-body simulations—be it for the direct computation of short-range interatomic potentials, the short-range direct part of a long-range interaction (i.e. for Particle Mesh Ewald[2]) or for the periodic construction of neighbor lists for a Verlet list.[3]

The conventional approach to computing pairwise interactions using cell lists is to partition the simulation domain into n_c cells C_α, $\alpha = 1 \ldots n_c$ of edge length greater or equal to the cutoff radius r_c of the interaction to be computed (or $r_c + r_s$ in the case of Verlet lists with skin radius r_s). The interaction potential for each particle $p_\alpha \in C_\alpha$ is then computed as the sum of the pairwise interactions between p_α and all other particles $\in C_\alpha$ and all other particles in the 26 neighboring cells C_β (in the following we will only consider simulations in three spatial dimensions). The interaction with each neighboring cell is computed according to Algorithm 1.

Algorithm 1. Compute interactions between all $p_\alpha \in C_\alpha$ with all $p_\beta \in C_\beta$

```
1: for all $p_\alpha \in C_\alpha$ do
2:     for all $p_\beta \in C_\beta$ do
3:         $r^2 = \left|\mathbf{x}[p_\alpha] - \mathbf{x}[p_\beta]\right|^2$
4:         if $r^2 \leq r_c^2$ then
5:             Compute and store the interaction between $p_\alpha$ and $p_\beta$.
6:         end if
7:     end for
8: end for
```

Due to the rather coarse partitioning—we are discretizing a sphere with a radius of r_c with a cube with an edge length of $3r_c$—the test $r^2 \leq r_c^2$ in line 4 will fail for approximately 84% of all particle pairs tested. Although the potential is only computed for the remaining 16% of particle pairs within the cutoff distance of each other, the interparticle distance r^2 is evaluated for every pair (line 3), causing significant overhead.

One way of overcoming this inefficiency is to partition the domain into cells of edge length smaller than r_c. The pairwise interactions are then not just computed between neighboring cells, but between all cells within r_c of each other (first suggested in Ref. 1 and implemented and analyzed in refs. 4–6). This approach can be taken to the limit wherein each cell holds at most one single particle, therefore reducing the number of spurious pairwise distance evaluations to zero. This gain in efficiency, however, is quickly offset by the number of cells C_β that need to be inspected for every interaction with a cell C_α, which grows cubically with the inverse of the cell edge length.

We return to the simpler case of cells with edge length r_c. Let us consider the case of two neighboring cells C_α and C_β sharing a common face (Fig. 1a). If we compute the pairwise interactions as in Algorithm 1, we need to compute $|C_\alpha| \times |C_\beta|$ pairwise distances, or, if we assume the particles are distributed homogeneously throughout the system, n_p^2, where $n_p = N/n_c$ is the average number of particles per cell.

Suppose now that we subdivide both cells into equal halves as shown in Figure 1b. The halves of each cell interact (arrows in Fig. 1) with the halves of the other cell with the exception of the two

Correspondence to: P. Gonnet; e-mail: gonnetp@inf.ethz.ch

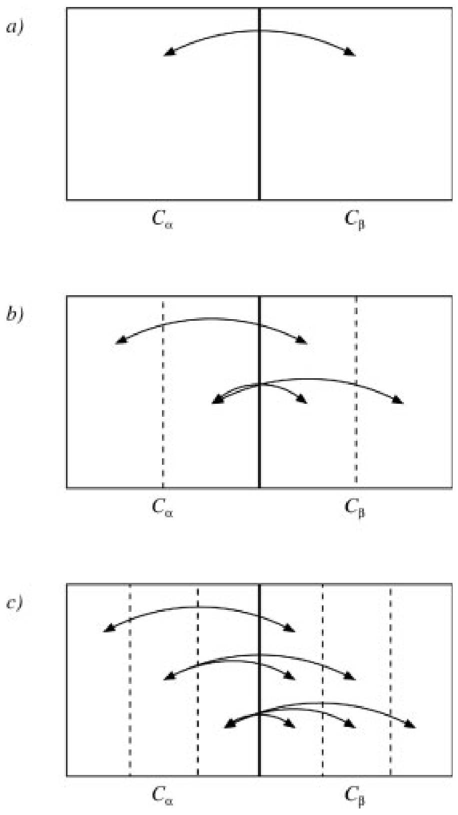

Figure 1. Subdivision of each cell into two or three fractions reduces the number of pairwise distances to be checked.

outer cell halves, which are separated by r_c and hence contain only particles too far apart to be considered. The total number of pairwise distance computations for each cell-half pair is then $(n_p/2)^2$, and thus for all three interacting cell-half pairs $\frac{3}{4}n_p^2$.

If we proceed in the same fashion and subdivide each cell into three fractions (Fig. 1c), each cell fraction pair interacts with an average of $(n_p/3)^2$ pairwise distance computations and since only six cell fraction pairs are separated by less than r_c, the total number of distance computations is further reduced to $\frac{6}{9}n_p^2$.

In general, if we partition each cell into k fractions, interacting each fraction pair involves computing $(n_p/k)^2$ pairwise distances. The number of fractions within r_c of each other and thus have to be interacted is $\frac{1}{2}k(k+1)$. Therefore, the total number of pairwise distance computations is $(k(k+1)/2k^2)n_p^2$, which, for $k \to \infty$, is $\frac{1}{2}n_p^2$.

Instead of splitting the cells into an infinite number of fractions, we sort the particles according to their position along the normalized vector $\vec{r}$ relaying the centers of both cells C_α and C_β (Fig. 2). Using the definition of the dot-product

$$\vec{a}\cdot\vec{b} = \cos\vartheta\,|\vec{a}|\,|\vec{b}|$$

and replacing $\vec{a}$ with $(\mathbf{x}[p_\alpha]-\mathbf{x}[p_\beta])$, where $p_\alpha \in C_\alpha$ and $p_\beta \in C_\beta$, and $\vec{b}$ with the normalized $\vec{r}/|\vec{r}|$, we obtain

$$\left(\mathbf{x}[p_\alpha]-\mathbf{x}[p_\beta]\right)\cdot\frac{\vec{r}}{\|\vec{r}\|} = \cos\vartheta\,\left|\mathbf{x}[p_\alpha]-\mathbf{x}[p_\beta]\right|\underbrace{\left|\frac{\vec{r}}{\|\vec{r}\|}\right|}_{=1}$$

$$\mathbf{x}[p_\alpha]\cdot\frac{\vec{r}}{\|\vec{r}\|} - \mathbf{x}[p_\beta]\cdot\frac{\vec{r}}{\|\vec{r}\|} = \cos\vartheta\,\left|\mathbf{x}[p_\alpha]-\mathbf{x}[p_\beta]\right|$$

$$\left|\mathbf{x}[p_\alpha]\cdot\frac{\vec{r}}{\|\vec{r}\|} - \mathbf{x}[p_\beta]\cdot\frac{\vec{r}}{\|\vec{r}\|}\right| = \underbrace{|\cos\vartheta|}_{\leq 1}\,\left|\mathbf{x}[p_\alpha]-\mathbf{x}[p_\beta]\right|$$

$$\left|\mathbf{x}[p_\alpha]\cdot\frac{\vec{r}}{\|\vec{r}\|} - \mathbf{x}[p_\beta]\cdot\frac{\vec{r}}{\|\vec{r}\|}\right| \leq \left|\mathbf{x}[p_\alpha]-\mathbf{x}[p_\beta]\right| \tag{1}$$

where $\mathbf{x}[p]\cdot(\vec{r}/\|\vec{r}\|)$ is the position of the particle p projected onto $\vec{r}$. We can see that particles that have a pairwise distance larger than r_c along $\vec{r}$ are guaranteed to be farther than r_c apart. In order to avoid repeatedly evaluating eq. (1), we precompute the dot products $\mathbf{x}[p_\alpha]\cdot(\vec{r}/||\vec{r}||)$ for all $p_\alpha \in C_\alpha$ and $\mathbf{x}[p_\beta]\cdot(\vec{r}/||\vec{r}||) - r_c$ for all $p_\beta \in C_\beta$ and store them with the corresponding particle in a temporary array. We then sort this array according to the distances along $\vec{r}$. Each particle $p_\alpha \in C_\alpha$ is then interacted with every particle $p_\beta \in C_\beta$ to the left of its position in the sorted array. For such particle pairs, the sorting ensures that

$$\mathbf{x}[p_\beta]\cdot\frac{\vec{r}}{||\vec{r}||} - r_c \leq \mathbf{x}[p_\alpha]\cdot\frac{\vec{r}}{|\vec{r}|}$$

$$\mathbf{x}[p_\beta]\cdot\frac{\vec{r}}{||\vec{r}||} - \mathbf{x}[p_\alpha]\cdot\frac{\vec{r}}{|\vec{r}|} \leq r_c$$

and thus the particles may be within r_c of each other. All particle pairs separated by more than r_c along $\vec{r}$ are ignored. The algorithm is described in Figure 2 and Algorithm 2.

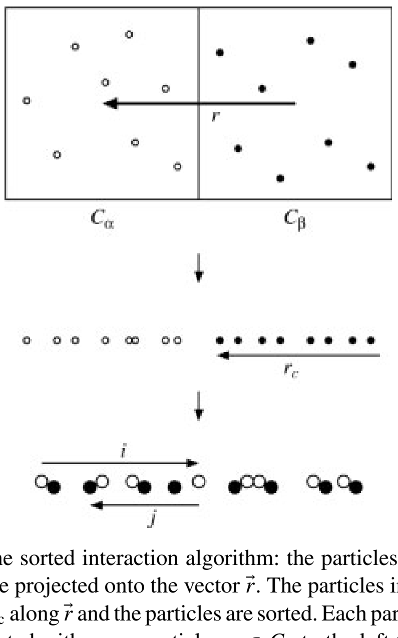

Figure 2. The sorted interaction algorithm: the particles in both cells C_α and C_β are projected onto the vector $\vec{r}$. The particles in C_β are then shifted by $-r_c$ along $\vec{r}$ and the particles are sorted. Each particle $p_\alpha \in C_\alpha$ is then interacted with every particle $p_\beta \in C_\beta$ to the left thereof in the sorted array.

Algorithm 2. Compute sorted interactions between all $p_\alpha \in C_\alpha$ with all $p_\beta \in C_\beta$

```
1: $d \leftarrow []$
2: $\mathsf{parts} \leftarrow []$
3: $n_{\mathsf{parts}} \leftarrow 0$
4: $\vec{r} \leftarrow C_\alpha - C_\beta$
5: $\vec{r} \leftarrow \vec{r}\frac{1}{||\vec{r}||}$
6: for $p_\alpha \in C_\alpha$ do
7:     $n_{\mathsf{parts}} \leftarrow n_{\mathsf{parts}} + 1$
8:     $d[n_{\mathsf{parts}}] \leftarrow \mathbf{x}[p_\alpha] \cdot \vec{r}$
9:     $\mathsf{parts}[n_{\mathsf{parts}}] \leftarrow p_\alpha$
10: end for
11: for $p_\beta \in C_\beta$ do
12:     $n_{\mathsf{parts}} \leftarrow n_{\mathsf{parts}} + 1$
13:     $d[n_{\mathsf{parts}}] \leftarrow \mathbf{x}[p_\beta] \cdot \vec{r} - r_c$
14:     $\mathsf{parts}[n_{\mathsf{parts}}] \leftarrow p_\beta$
15: end for
16: Sort $d$ and parts according to $d$.
17: $\mathsf{left} \leftarrow []$
18: $n_{\mathsf{left}} \leftarrow 0$
19: for $i$ from 1 to $n_{\mathsf{parts}}$ do
20:     if $\mathsf{parts}[i] \in C_\beta$ then
21:         $n_{\mathsf{left}} \leftarrow n_{\mathsf{left}} + 1$
22:         $\mathsf{left}[n_{\mathsf{left}}] \leftarrow \mathsf{parts}[i]$
23:     else
24:         for $j$ from 1 to $n_{\mathsf{left}}$ do
25:             $r^2 = \left|\mathbf{x}[\mathsf{parts}[i]] - \mathbf{x}[\mathsf{left}[j]]\right|^2$
26:             if $r^2 \le r_c^2$ then
27:                 Compute and store the interaction between $\mathsf{parts}[i]$ and $\mathsf{left}[j]$.
28:             end if
29:         end for
30:     end if
31: end for
```

We will now try to assess the efficiency of this approach. The probability of two particles in neighboring cells sharing a common face in three dimensions actually being within r_c of each other if their pairwise distance along $\vec{r}$ less than r_c, is approximately 67.1% (computed using Monte-Carlo integration), as opposed to approximately 33.5% for particles anywhere in C_α and C_β.

For diagonally positioned cells C_α and C_β sharing only one common edge (Fig. 3), only approximately 16.2% of the particle pairs $p_\alpha \in C_\alpha$, $p_\beta \in C_\beta$ are within at least r_c of each other along $\vec{r}$. Of these particle pairs, approximately 56.7% are within r_c of each other. For diagonally positioned cells C_α and C_β sharing only a single common corner, only about 3.6% of particle pairs spanning both cells are within r_c of each other along $\vec{r}$. Of these, approximately 57.6% are within r_c of each other.

In three dimensions, every cell has 6 direct neighbors, 12 neighbors that share only an edge, and 8 neighbors that share only a single corner. This means that if we compute only the pairwise distance of particles within r_c of each other along the respective vectors $\vec{r}$, in total, $\frac{1}{26}(6 \times 67.1\% + 12 \times 56.7\% + 8 \times 57.6\%) \approx 59.4\%$ of the particles pairs for which the pairwise distance is computed are actually within r_c of each other. This is almost four times better than the 16% which can be expected from the naive Algorithm 1 and more than double the 27% using cells with an edge length of $\frac{1}{2}r_c$ as suggested in ref. 5.

The sorted interaction algorithm (Algorithm 2) was implemented in the FASTTUBE simulation software[7] using the QuickSort algorithm[8] to sort the particles and was compared with the naive implementation in Algorithm 1 for the cell edge lengths r_c, $\frac{1}{2}r_c$, $\frac{1}{3}r_c$, and $\frac{1}{4}r_c$. All simulations were run on an IBM ThinkPad T40p computer with a 1.6 GHz Pentium M CPU.

A simulation consisting of 1000 SPC\E water molecules in a $3.166 \times 3.166 \times 3.166$ nm simulation box at 300 K with all interactions truncated at 1 nm was used to obtain the times in Table 1. Although simulations containing water are usually run using the Particle Mesh Ewald algorithm[2] or similar algorithms for the computation of the long-range electrostatic potential, these algorithms, all need to compute a short-range direct part, which is computed using a cutoff and, therefore, cell lists. The sorted interaction algorithm is approximately 28% faster than the naive Algorithm 1 with a cell edge length of r_c and 14% faster than the naive Algorithm 1 with an optimal edge length of $\frac{1}{2}r_c$.

The speed-up is due to the reduced number of spurious pairwise distance calculations. The results do not scale as would be expected with the average number of spurious pairwise distance evaluations

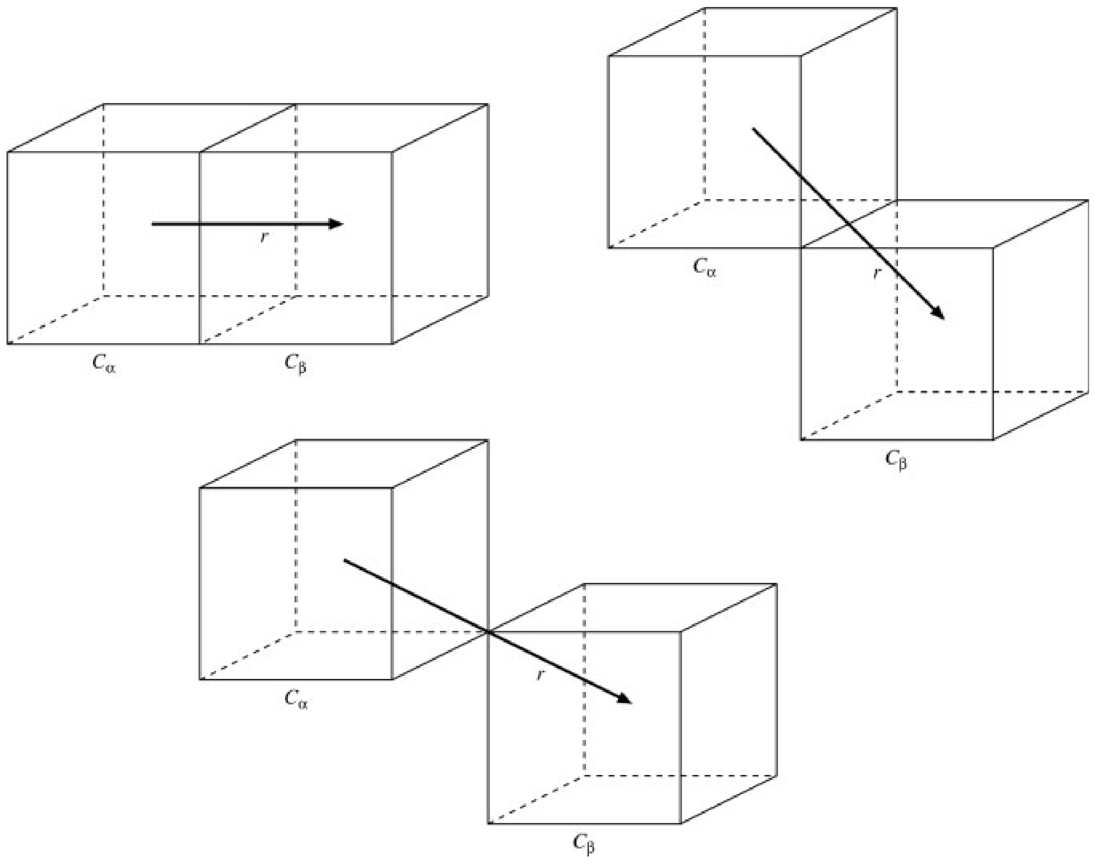

Figure 3. Three possible configurations for neighboring cells C_α and C_β with the corresponding vectors $\vec{r}$.

computed earlier due to the overhead incurred by the sorting of the particles. The sorting, however, scales with $\mathcal{O}(n \log n)$, which is below the $\mathcal{O}(n^2)$ required for computing the pairwise interactions and therefore should not affect the asymptotic behavior of the algorithm.

Table 1. Average Times per Time Step and Relative Time Per Time Step Spent Computing the Pairwise Interactions for the Different Algorithms for a Simulation of 1000 SPC\E Water Molecules in a $3.166 \times 3.166 \times 3.166$ nm at 300 K with $r_c = 1.0$.

Method	Cell edge length	Per time step (μs)	Rel.
Sorted	r_c	143.1	1.000
Naive	r_c	199.2	1.392
Naive	$\frac{1}{2}r_c$	167.0	1.167
Naive	$\frac{1}{3}r_c$	235.3	1.644
Naive	$\frac{1}{4}r_c$	337.9	2.361

References

1. Allen, M. P.; Tildesley, D. J. Computer Simulation of Liquids. Clarendon Press: Oxford, 1987.
2. Darden, T.; York, D.; Pedersen, L. Particle mesh Ewald: An $N \cdot \log N$ method for Ewald sums in large systems. J Chem Phys 1993, 98, 10089.
3. Verlet, L. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. Phys Rev 1967, 159, 98.
4. Mattson, W.; Rice, B. M. Near-neighbor calculations using a modified cell-linked list method. Comput Phys Commun 1999, 119, 135.
5. Yao, Z.; Wang, J.-S.; Liu, G.-R.; Cheng, M. Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method. Comput Phys Commun 2004, 161, 27.
6. Heinz, T. N.; Hünenberger, P. H. A fast pairlist-construction algorithm for molecular simulations under periodic boundary conditions. J Comput Chem 2004, 25, 1474.
7. Werder, T.; Walther, J. H.; Jaffe, R. L.; Halicioglu, T.; Koumoutsakos, P. On the water-graphite interaction for use in MD simulations of graphite and carbon nanotubes. J Phys Chem B 2003, 107, 1345.
8. Hoare, C. A. R. Algorithm 63, Partition: Algorithm 64, Quicksort. Commun ACM 1961, 4, 321.