Constructing a Near-Minimal-Volume Computational Box for Molecular Dynamics Simulations with Periodic Boundary Conditions

Henk Bekker[1], Jur P. van den Berg[1], and Tsjerk A. Wassenaar[2]

[1]Institute for Mathematics and Computing Science, University of Groningen, P.O.B. 800 9700 AV Groningen, The Netherlands. [2]Groningen Biomolecular Sciences and Biotechnology Institute (GBB), Department of Biophysical Chemistry, University of Groningen, Nijenborgh 4, 9747 AG, Groningen, The Netherlands.
bekker@cs.rug.nl, j.p.van.den.berg@wing.rug.nl, T.A.Wassenaar@chem.rug.nl

Abstract. In many M.D. simulations the simulated system consists of a single macromolecule in a solvent. Usually, one is not interested in the behaviour of the solvent, so, the CPU time may be minimized by minimizing the amount of solvent. For a given molecule and cut-off radius this may be done by constructing a computational box with near minimal volume. In this article a method is presented to construct such a box, and the method is tested on a significant number of macromolecules. The volume of the resulting boxes proves to be typically 40% of the volume of usual boxes, and as a result the simulation time decreases with typically 60%.

1 Introduction

Much CPU time is spent nowadays on the molecular dynamics (M.D) simulation of bio-macromolecules, notably proteins, with the goal to gain insight in the functioning of biophysical processes. Of these simulations a considerable part consists of the simulation of a single macromolecule m surrounded by a solvent, in most cases water. To prevent finite system effects, most M.D. simulations are done under periodic boundary conditions (PBC) which means that the computational box B is surrounded by an infinite number of replica boxes in a regular, space filling manner. In 3D there are five convex space filling shapes namely the triclinic box, the hexagonal prism, the dodecahedron, the elongated dodecahedron and the truncated octahedron, see figure 1.

Let s be the system formed by a computational box containing m and the water surrounding m, and let S be the infinite system formed by tessellating space with an infinite number of replica's of s, see figure 2. In M.D. simulations, interactions over a distance r_{co} are truncated. Moreover, in M.D. simulations replica's of m should not interact with each other. That means that in S no two replica's of m should be closer than r_{co}. This may be reformulated by introducing a shape M, see figure 2, defined as m dilated by a layer of width $\frac{1}{2}r_{co}$. Then, stating that in S no two replica's of m should be closer than r_{co} is equivalent with stating that in S no two replica's of M should overlap.

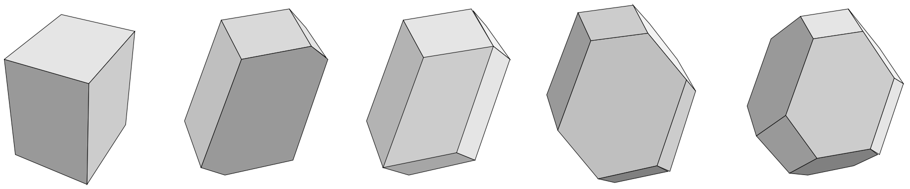

Fig. 1. The five box types used as computational box in current M.D. simulations with periodic boundary conditions. With each type, space may be tessellated in a space-filling way. From left to right: the triclinic box, the hexagonal prism, the dodecahedron, the elongated dodecahedron and the truncated octahedron

During an M.D. simulation m rotates in an erratic way, resulting in a number of full rotations during a typical simulation. Besides rotational motion m may also show conformational changes. Often these changes are of major interest and should not be restrained in any way. In current M.D. practice the computational box B is constructed by enclosing M by one of the five regular spacefillers such that m may rotate freely in B and may show some conformational change without leaving B. Often the additional space in B allowing for conformational changes of m is created by taking the width of the layer around m a bit larger than $\frac{1}{2}r_{co}$.

Let the space outside m and inside B be called C, so, $C = B - m$, and let the space outside M and inside B be called D, so, $D = B - M$, see figure 2c. After B has been constructed m is placed in B, and C is filled with water molecules. From the foregoing it will be clear that the water in D does not contribute to the simulation. Yet, per unit volume, simulating it takes approximately the same CPU effort as simulating the atoms in M, so, denoting the volume of D by $vol(D)$, we spend $\approx \frac{vol(D)}{vol(B)}$ of our CPU time on the simulation of irrelevant water.

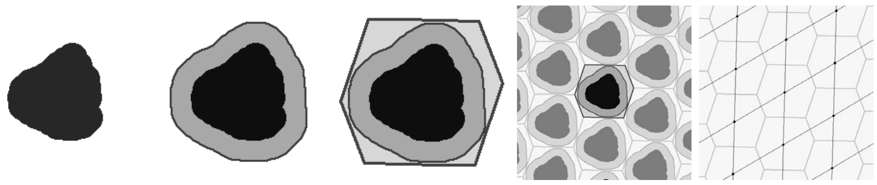

Fig. 2. a: A molecule m. b: m surrounded by a layer of width $\frac{1}{2}r_{co}$, giving M. c: A PBC box containing M. d: Part of an infinite M.D. system S, formed by tessellating space with a PBC box s. e: The lattice defined by the boxes in d.

In this article we present a technique to construct a near-minimal-volume PBC box around M, resulting in a significant decrease of $vol(D)$, so, resulting in a significant decrease of simulation time. For a near-minimal-volume M.D. simulation two ingredients are essential:

1. A method to restrain the rotational motion of m during the simulation without affecting the conformational changes of m.
2. A method to construct a PBC box of which $vol(D)$ is nearly minimal.

Fortunately, we do not have to take care of the first requirement. Some time ago such a method has been developed [1], amongst others with the goal to enable minimal volume simulations. The second requirement will be taken care of in this article.

The structure of this article is as follows. In section two we show that, instead of focussing on the shape of the computational box, we should focus on the lattice defined by box positions in S. In this way the problem of finding a box with minimal volume is reformulated as the problem of finding the densest lattice packing of M. We introduce a method to calculate this packing. In section three implementation issues are treated, and in section four our implementation is tested on some bio-macromolecules.

2 Boxes, Their Related Lattices and Calculating Lattice Packings

As mentioned earlier, there are five types of convex space-filling boxes. When s is such a box and S is formed by tessellating space with s, a lattice L is defined [2]. Here, a lattice L is the set of points $L = i * \mathbf{a} + j * \mathbf{b} + k * \mathbf{c} \;\; i, j, k$ integer, and $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are the lattice vectors. So, in S at every point of L a box s is situated. Let the triclinic box spanned by the lattice vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ be called s_T. Now, instead of forming S by tessellating space with s, S may be formed just as well by tessellating space with s_T. So, a minimal volume simulation may be set up by devising a densest lattice packing of M. Let s_T^* be the triclinic box spanned by the lattice vectors of the densest lattice packing of M. Then S may be formed by tiling space with s_T^*. This observation is essential for our method.

Having reformulated the minimal-volume box problem as a densest lattice packing problem we have to look for a method which determines for a given body M the densest lattice packing. For polyhedral convex M such a method exists [3]. However, for most bio-macromolecules m and typical layer widths around m, the shape of M is non-convex. For non-convex M there exists no densest lattice packing algorithm, and in the computational geometry community this problem is considered as hard, so, in the foreseeable future, very probably, no such algorithm will be devised. For that reason we have to work with a heuristic method to find an approximation of the densest lattice packing of M, which we will call the near-densest lattice packing (NDLP) of M. The NDLP heuristic is based on the incorrect assumption that M is convex, and a check is added to filter out incorrect packings due to the non-convexity of M.

In principle, the NDLP method works for 3D bodies in general. However, to avoid discussions about degenerate problem instances, in this article we assume that the volume of M is non-zero and that M has no points at infinity. The first step in the NDLP algorithm is to construct the *contact body* of M, designated by N. It is constructed using the Minkowski sum [4]. The Minkowski sum $R = P \oplus Q$ of two bodies P and Q is another body R defined as $R \equiv \{\mathbf{a}+\mathbf{b} : \mathbf{a} \in P, \mathbf{b} \in Q\}$. Defining $-M$ as the body M inverted in the origin, N is given by

$$N \equiv M \oplus -M. \tag{1}$$

It can be shown easily that N is symmetric, and centered at the origin. Denoting by $M_{\mathbf{a}}$ the body M translated over the vector $\mathbf{a}$, N has the following property. *The boundary of N consists of all points* $\mathbf{a}$ *for which holds that M and $M_{\mathbf{a}}$ touch without overlapping.* See figure 3.

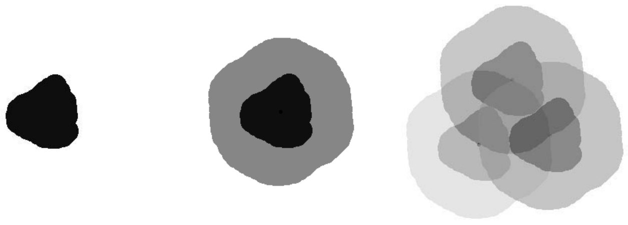

Fig. 3. 2D example of a body M (a), its contact body N (b), and N used to construct a situation where three copies of M touch each other (c). Note that N is point-symmetric.

Let us now explain how the NDLP heuristic works. We want to position M and three of its translates $M_{\mathbf{a}}$, $M_{\mathbf{b}}$ and $M_{\mathbf{c}}$ in such a way that *the volume of the triclinic cell spanned by* $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ *is minimal, without overlap between M, $M_{\mathbf{a}}$, $M_{\mathbf{b}}$ and $M_{\mathbf{c}}$* . In principle, for this we have to search through all combinations of $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, so, in principle we are dealing with a nine dimensional minimization problem. The key property of our NDLP method is that we reduce this nine dimensional problem to a three dimensional problem by making the following choice. *We only search through those combinations of* $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ *for which holds that every body of the set $\{M, M_{\mathbf{a}}, M_{\mathbf{b}}, M_{\mathbf{c}}\}$ is touched by the three other ones.* That this choice leads to a three dimensional minimization problem can be seen as follows. M is placed at an arbitrary location. For $M_{\mathbf{a}}$ $\mathbf{a}$ is chosen on the boundary of N, so, M and $M_{\mathbf{a}}$ touch. Because $\mathbf{a}$ is chosen on the boundary of N there are two degrees of freedom in choosing $\mathbf{a}$. Now we calculate the intersection of N and $N_{\mathbf{a}}$, which is a curve in 3D. On this curve we choose $\mathbf{b}$. So, $M_{\mathbf{b}}$ touches M and $M_{\mathbf{a}}$. Because $\mathbf{b}$ is chosen on a curve there is one degree of freedom in choosing $\mathbf{b}$. Finally we choose $\mathbf{c}$ on the intersection of N, $N_{\mathbf{a}}$ and $N_{\mathbf{b}}$, which is a small set of points, for typical M ranging from 2 to 10. So, the number of degrees of freedom in choosing $\mathbf{c}$ is zero. Herewith the number of degrees of freedom of the search problem proves to be $2 + 1 + 0 = 3$.

Now let us assume that we are searching through all combinations of $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$, according to our NDLP heuristic, with an appropriate search granularity to be discussed later. For every combination of $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ we have to calculate $|det(\mathbf{a}, \mathbf{b}, \mathbf{c})|$ and store the $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ that give minimal $|det(\mathbf{a}, \mathbf{b}, \mathbf{c})|$. Obviously, $M_{\mathbf{a}}$, $M_{\mathbf{b}}$ and $M_{\mathbf{c}}$ do not overlap M. However, for non-convex M, possibly there exist one or more lattice points $\mathbf{d} = i * \mathbf{a} + j * \mathbf{b} + k * \mathbf{c}$ i, j, k integer, for which M and $M_{\mathbf{d}}$ overlap. That this may happen is not obvious to understand, there is no 2D analogy. To filter out these cases, for every $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ with minimal volume $|det(\mathbf{a}, \mathbf{b}, \mathbf{c})|$ we have to perform an additional test. That M and $M_{\mathbf{d}}$ overlap means that there are i, j, k not all 0, such that the lattice point $\mathbf{d} = i * \mathbf{a} + j * \mathbf{b} + k * \mathbf{c}$ $\quad i, j, k$ integer, is in the interior of N. So we have to test for all lattice points within a range $1/2$ $diam(N)$ of the origin whether they fall in the interior of N, where $diam(N)$ is the diameter of N.

Having found a minimal volume box spanned by $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ that also passes the test that no lattice point lies in N, we have to put m in the triclinic box B_T spanned by $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$. The location of m in B_T is completely free but the obvious choice is to locate m in the middle of B_T. Sometimes m will not fit entirely in B_T, it sticks out no matter where it is located in B_T. That does not matter, we simply locate m somewhere in the middle of B_T. See figure 4. Now for every atom of m protruding B_T it holds that it can be shifted over some lattice vector $\mathbf{d}$ such that it falls in B_T. For every protruding atom such a vector is calculated and the atom is translated over this vector. Now all atoms of m are in B_T but m is possibly fragmented. That is no problem because, when B_T containing a fragmented m is used to tessellate space, giving the infinite M.D. system S, in S complete molecules are formed from these fragments. Finally, all voids in B_T are filled with water molecules. Herewith we have constructed the near-minimum-volume triclinic system s^*_T. Summarizing, the complete NDLP algorithm outline is as follows.

```
from m and r_co construct M;
from M construct N;
forall a on boundary of N  do
  forall b on intersection of N, N_a  do
    forall  c on intersection of N, N_a, N_b do
      if |det(a,b,c)| < old_det_abc and not
      point_of_L_inside_N then store(a,b,c);
    end // c loop
  end // b  loop
end // a  loop
put m in box a,b,c
```

Let us briefly comment on our choice only to search through those combinations of $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ for which holds that every body of the set $\{M, M_{\mathbf{a}}, M_{\mathbf{b}}, M_{\mathbf{c}}\}$ is touched by the three other ones. It is shown that, for some convex bodies, there are other contact situations giving minimal volume [3]. Very probably this also holds for non-convex bodies, but little is known about that. However, searching through these situations would take much more CPU time than searching

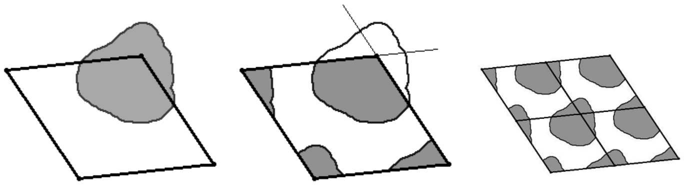

Fig. 4. a: 2D example of a minimal volume triclinic box B_T containing a protruding molecule m. b: The protruding parts of m have been reset in B_T by shifting them over lattice vectors, resulting in a fragmented molecule in B_T. c: Part of the infinite M.D. system formed by tessellating space with B_T with fragmented m. In the infinite system, whole molecules are formed by fragments from various boxes.

through the situations of our choice, probably without finding a considerably denser packing.

3 Implementation

In this section we explain how we transformed the NDLP algorithm into an efficient and robust program. In the previous section we did not specify how m, M, and N are represented. In the implementation m and M are point sets, and N is polyhedral. We start with a macro-molecule m from some library, for example from the Proteine Data Bank (PDB) [5]. For us, m is simply a point set, where every point represents an atom. In m we include all atoms of the macromolecule, so hydrogen atoms are included. From m we have to construct M. Recall that m dilated with a layer of width $\frac{1}{2}r_{co}$ gives M. We assume that r_{co} has been chosen big enough to allow for conformational changes. Now we construct a spherical point set *ball* by distributing ≈ 50 points more or less evenly on the boundary of a sphere with radius r_{co}. The point set M is obtained by taking the Minkowski sum of m and *ball*, so, $M \equiv m \oplus ball$. Obviously, the number of points in M is fifty times the number of points in m. Of M we only need boundary points, so we delete interior points. This strongly reduces the number of points in M.

From M we construct the contact body N by taking the Minkowski sum of M and $-M$, so, $N \equiv M \oplus -M$. The number of points in N is the square of the number of points in M. We use only part of these points by deleting all interior points of N and part of the boundary points. We want to have control over the number of remaining boundary points, which is done by using a grid-based selection method, that is, we construct a rectangular grid of $32 * 32 * 32$ cells, covering N, and determine of each point of N the cell it falls in. All points in interior cells are deleted, and of the points in a boundary cell only the one nearest to the boundary is kept. In this way the boundary is defined by typically 3000 points. Now we switch from the point set representation of N to a polyhedral representation of N. For this we use the α-*hull* surface reconstruction algorithm [6]. The α-hull algorithm takes as input a set of points and constructs an outer hull around these points. Whether a point is considered as a boundary point or an interior point depends on the value of the parameter α. For $\alpha = \infty$ the outer hull is the convex hull of the point set, for $\alpha = 0$ every point is considered as an interior point. We choose α so that the overall shape of the polyhedral approximation is practically identical to the shape of the point set. Besides constructing a polyhedral hull, the α-hull algorithm also generates a Delaunay triangulation of this hull, so, the result of the α-hull algorithm is a collection of triangles defining the hull of N.

In the NDLP method not only N but also translates of N are used. Translating N over some vector $\mathbf{a}$ is done by adding $\mathbf{a}$ to every coordinate of N, giving $N_{\mathbf{a}}$. To calculate the intersection curve of N and $N_{\mathbf{a}}$, represented by $N \cap N_{\mathbf{a}}$, we use the OBBTree algorithm [7]. This algorithm calculates of two sets of triangles in 3D which pairs of triangles intersect. For each pair of intersecting triangles we calculate the line segment that is in both triangles, so, the resulting set of intersection segments forms the intersection curve $N \cap N_{\mathbf{a}}$. Subsequently we calculate the intersection of $N \cap N_{\mathbf{a}}$ and $N_{\mathbf{b}}$.

As explained before, the NDLP method is in essence a search problem in three continuous parameters. To make the method practicable the parameter space has to be transformed into a finite set of discrete points, i.e. the granularity of the search process has to be determined. In our implementation the search granularity is dictated by the granularity of the triangulation of N. More precise, the vector $\mathbf{a}$ runs through all of the centers of the triangles of N. In the same way, $\mathbf{b}$ runs through all of the centers of the line segments of $N \cap N_{\mathbf{a}}$. As the triangulation of N depends on the number of points returned by the grid-based selection method, the search granularity may be controlled by the number of grid-cells.

We implemented the NDLP method in C++ using the computational geometry library CGAL[8], the α-hull algorithm and the OBBTree algorithm.

4 Results

We tested the NDLP method on seventeen macromolecules from [5]. The shape of these molecules ranges from almost spherical to complex, see figure 5. Every molecule was packed in two ways: with GROMACS [9] in the conventional way in a dodecahedron, and with the NDLP method. Subsequently, every molecule was simulated for 25000 timesteps of $2fs$ in two different triclinic boxes. To keep the comparison fair we did not do the simulation in the octahedron but in the triclinic box defined by the lattice of the octahedron, i.e. every molecule was simulated in two different triclinic boxes; one calculated via the truncated octahedron and one calculated by the NDLP method. We used the GROMACS M.D. simulation package, using rotational restraining for the simulation in the NDLP box. The simulations were done on one AMD Athlon 600 Mhz. In table 1 the molecules are given, the volume of their simulation boxes, the number of water molecules in the box, and the simulation time. From this table it is clear that on average *boxes calculated with the NDLP method have a volume of ≈40% of the corresponding GROMACS dodecahedron, and the speedup of the simulation is ≈ 2.2.*

Macro-molecules			Dodecahedron			NDLP triclinic box			
nr	PDB code	nr of atoms	volume $[nm^3]$	nr of water molecules	simulation time [hr:min]	volume $[nm^3]$	nr of water molecules	simulation time [hr:min]	speedup factor
1	1A32	1102	577.82	18610	06:51	118.93	3474	01:25	**4.83**
2	1A6S	805	142.43	4366	01:45	80.08	2254	00:58	**1.81**
3	1ADR	763	167.25	5137	01:59	80.73	2266	00:55	**2.16**
4	1AKI	1321	233.54	7027	02:48	93.99	2454	01:07	**2.50**
5	1BW6	595	130.27	3904	01:29	66.32	1874	00:45	**1.97**
6	1HNR	485	124.31	3865	01:29	59.30	1717	00:41	**2.17**
7	1HP8	686	177.10	5522	02:09	77.57	2203	00:53	**2.43**
8	1HQI	982	218.77	6764	02:38	103.71	2947	01:12	**2.19**
9	1NER	768	147.91	4498	01:45	85.35	2403	00:58	**1.81**
10	1OLG	1808	468.93	14537	05:37	203.44	5781	02:25	**2.32**
11	1PRH	11676	1337.80	38715	16:27	611.67	14554	07:47	**2.11**
12	1STU	668	190.32	5973	02:16	73.41	2074	00:50	**2.72**
13	1VCC	833	152.69	4612	01:48	69.77	1905	00:49	**2.20**
14	1VII	389	99.74	3093	01:08	46.96	1348	00:32	**2.12**
15	2BBY	767	159.26	4868	01:53	80.78	2271	00:56	**2.01**
16	1D0G*	1052	645.92	20805	07:42	112.78	3168	01:19	**5.84**
17	1D4V*	3192	1319.21	42202	15:51	451.23	13215	05:29	**2.89**

Table 1. Seventeen macro-molecules packed in simulation boxes, and simulated with GROMACS. Every molecule is packed in two ways: by the standard method of the M.D. simulation program GROMACS using the the dodecahedron, and using the NDLP method. In the NDLP method the width of the layer around m is $10\AA$. Subsequently, every molecule is simulated in two different triclinic boxes, namely the one defined by the lattice vectors of the space tessellation with the dodecahedron and the one calculated with the NDLP method. In both simulations $r_{co} = 14\AA$. Every box and molecule is simulated for 25000 timesteps of $2fs$ using the simulation package GROMACS running on a single 600 Mhz. AMD Athlon. For every box and molecule the box volume is given, the number of water molecules surrounding the macro-molecule and the simulation time. In the last column the speedup of the simulation time is given. The main result of this article is that on average the volume of the simulation box calculated with the NDLP method is ≈ 40% of the volume of the dodecahedron calculated with the current method of GROMACS, and that the average speedup of the simulation using the NDLP box is ≈ 2.2.

5 Discussion and Conclusion

– In the NDLP method, calculating N takes typically $\frac{1}{3}$ of the total time, and the actual search process takes typically $\frac{2}{3}$ of the total time. On our system

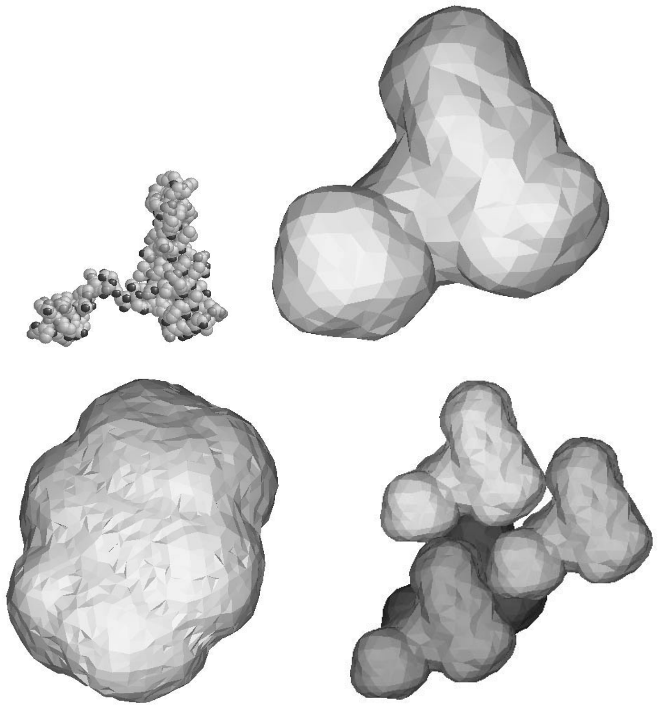

Fig. 5. The NDLP method applied to molecule $1a32$ from the PDB, consisting of 1102 atoms. The molecule is shown as a set of atoms, and M and N are shown as polyhedrals. The scaling of the figures varies. Top left: The molecule m. Top right: The body M, obtained by dilating m with a layer of width $10\mathring{A}$. Bottom left: The contact body N, defined as $N \equiv M \oplus -M$. Bottom right: The NDLP of M. The four copies of M in the NDLP configuration touch each other. Calculating N took 6 minutes, calculating the the NDLP took 9 minutes, done on a Pentium III 500 mhz. with 128 MB.

it takes $15 - 45$ minutes to calculate the NDLP of a molecule, depending on the complexity of its shape.

- The NDLP method is only useful when combined with rotational restraining.
- The overhead in CPU time introduced by the rotational restraining is negligible w.r.t. the speedup.
- The NDLP method works for a single macromolecule and for multiple macromolecules with more or less fixed relative positions. In table 1 the molecules 16 and 17 are of the latter type.
- We only compared the NDLP method with the GROMACS packing method, not with other M.D. simulation packages. However, because other packages use the same methods as GROMACS to calculate the computational box we expect that for other packages a similar gain in efficiency can be achieved.

- We will make the NDLP method available as an internet service on a page of the M.D. group in Groningen.
- The main conclusion of this article is that the speed of M.D. simulations using boxes constructed with the NDLP method is on average 2.2 times the speed of simulations using boxes constructed with conventional methods.

References

[1]A. Amadei, G. Chillemi, M. A. Ceruso, A. Grottesi, A. Di Nola,
Molecular dynamics simulation with constrained roto-translational motions: Theoretical basis and statistical mechanical consistency. Journal of Chemical Physics, vol. 112, nr. 1, 1 jan. 2000.
[2]H. Bekker, Unification of Box Shapes in Molecular Simulations.
Journal of Computational Chemistry, Vol. 18, No. 15, 1930-1942 (1997).
[3] U. Betke, M. Henk Densest lattice packings of 3-polytopes. Comput. Geom. 16, 3,pp. 157-186 (2000)
[4]M. de Berg, M. van Krefeld, M. Overmars, O. Schwarzkopf, Computational Geometry, Algorithms and Applications, Springer Verlag, Berlin (2000)
[5] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne: The Protein Data Bank.
http://www3.oup.co.uk:80/nar/Volume_28/Issue_01/html/gkd090_gml.html
[6] H. Edelsbrunner, E. P. Muecke, Three dimensional alpha shapes. ACM Transactions on Graphics, Vol. 13, No. 1, pp. 43-72 (1994).
[7] S. Gottschalk, M. C. Lin, D. Manocha, OBBTree: A hierarchical structure for rapid interference detection. Computer Graphics, Vol. 30, Annual Conference Series, pp. 171-180 (1996)
[8] Computational Geometry Algorithms Library,
http://www.cgal.org/Manual/doc_html/index.html
[9]GROMACS: The GROningen MAchine for Chemical Simulations,
http://www.gromacs.org/