PACKMOL: A package for building initial configurations for molecular dynamics simulations

L. Martínez * R. Andrade † E. G. Birgin † J. M. Martínez ‡

December 10, 2008

Abstract

Adequate initial configurations for molecular dynamics simulations consist of arrangements of molecules distributed in space in such a way to approximately represent the system's overall structure. In order that the simulations are not disrupted by large Van der Waals repulsive interactions, atoms from different molecules must keep safe pairwise distances. Obtaining such a molecular arrangement can be considered a packing problem: Each type molecule must satisfy spatial constraints related to the geometry of the system, and the distance between atoms of different molecules must be greater than some specified tolerance. We have developed a code able to pack millions of atoms, grouped in arbitrarily complex molecules, inside a variety of three-dimensional regions. The regions may be intersections of spheres, ellipses, cylinders, planes or boxes. The user must provide only the structure of one molecule of each type and the geometrical constraints that each type of molecule must satisfy. Building complex mixtures, interfaces, solvating biomolecules in water, other solvents, or mixtures of solvents, is straightforward. In addition, different atoms belonging to the same molecule may also be restricted to different spatial regions, in such a way that more ordered molecular arrangements can be built, as micelles, lipid double-layers, etc. The packing time for state-of-the-art molecular dynamics systems varies from a few seconds to a few minutes in a personal computer. The input files are simple and currently compatible with PDB, Tinker, Molden or Moldy coordinate files. The package is distributed as free software and can be downloaded from http://www.ime.unicamp.br/∼martinez/packmol/.

Keywords: Initial configuration, molecular dynamics, packing, large-scale optimization, PACKMOL.

*Department of Physical Chemistry, IQ-UNICAMP, University of Campinas. leandromartinez98@gmail.com

†Department of Computer Science IME-USP, University of São Paulo, Rua do Matão 1010, Cidade Universitária, 05508-090, São Paulo SP, Brazil.

‡Department of Applied Mathematics, IMECC-UNICAMP, University of Campinas, CP 6065, 13081-970 Campinas SP, Brazil.

1 Introduction

The first step in a molecular dynamics simulation consists of obtaining initial coordinates for all the atoms of the system. For example, in order to run a simple simulation consisting of 300 water molecules with experimental density, we need the positions of the 300 molecules inside an adequately sized box. Furthermore, since molecular dynamics force-fields contain repulsive terms that increase abruptly for short atom-to-atom distances, the distances between atoms from different molecules must be large enough so that repulsive potentials do not disrupt the simulations. Frequently, the instability and non-differentiability of the potential energy resulting from overlapping atoms is hard to overcome [1].

For a simple system such as a water box, we can obtain an adequate configuration simply by ordering the molecules in a regular lattice. However, for slightly more complex systems such as a solvated peptide, regular configurations would almost certainly contain overlapping atoms. Many times, this inconvenience is overcome in the following way: a box of water molecules regularly distributed is constructed (or a previously equilibrated solvent box is used, when available). Then, the "big" molecule is added to the system and the solvent molecules containing overlapping atoms are removed. Finally, the energy of the system is minimized.

However, when the complexity of the system increases, the work for building a starting configuration may be very tedious. For example, if the protein is added to a box containing water, ions and urea (a common denaturant), and the overlapping solvent molecules are removed, the charge of the system and molar fraction of urea must be further adjusted. Building ordered molecular systems, such as micelles, double layers, or interfaces, require lots of trials, manipulation of files, small ad-hoc codes, etc., and, so, the very first steps of the simulation turn out to be quite cumbersome. For this reason, ready-to-use configurations and specific programs have been developed for the construction of some commonly studied systems, particularly membranes [2, 3]. As the variety of interesting systems being studied increases, more general approaches will be of great utility.

Recently, we proposed that the initial configuration problem can be treated as a packing problem [1]. The molecules are packed within spatial regions with the desired characteristics, in such a way that atoms from different molecules keep a safe pairwise distance. Small systems composed by interfaces, mixtures of various components and solvated proteins were successfully built and used in actual molecular dynamics simulations [4, 5, 6]. In that work, we have shown that random sets of appropriately packed molecules, with no intermolecular clashes, can be rapidly equilibrated to the thermodynamic energy using standard MD integration algorithms and energy minimization [1], thus validating the approach.

In this paper we show how this idea gives rise to an efficient code for public use. The construction of large systems with increasing structural complexity is now envisaged. With this in mind, we define a general set of spatial constraints, so that molecules can be packed in complex regions formed by the intersections of planes, spheres, ellipsoids, cylinders and boxes. We have also implemented tools that allow the user to allocate different parts of the molecules to different regions, in such a way that ordered structures, such as micelles, may

be built. To achieve the needs of the increasingly large systems currently being simulated, we have modified the function and gradient evaluation methods in order that the practical algorithm scales linearly with the number of atoms of the system, and we have developed a parallel version. As its predecessor, the computational software is called PACKMOL, and is free software. Successive versions of PACKMOL are employed by many members of the Molecular Dynamics community since 2003. Briefly speaking, the present paper reports the improvements of PACKMOL since its introduction in 2003 as a proof-of-concept package.

This paper is organized as follows. Section 2 presents the mathematical model. Section 3 deals with the efficient evaluation of the objective function and its derivatives. The optimization method used to solve the packing problem and the parallelization of the objective function and its derivatives are described in Section 4. In Section 5 we describe the usage of the software and we present some examples. Conclusions and perspectives are stated in Section 6.

2 Mathematical model

Let us call nmol the total number of molecules that we want to place in a region of the three-dimensional space. For each $i = 1, \ldots, \text{nmol}$, let $\text{natom}(i)$ be the number of atoms of the $i-$th molecule. Molecules can be grouped in different types (water, protein, urea, and so on) but this classification is irrelevant for the model description. Each molecule is represented by the cartesian coordinates of its atoms. The point whose coordinates are the arithmetic averages of the coordinates of the atoms will be called barycenter. To facilitate visualization, assume that the origin is the barycenter of all the molecules. For all $i = 1, \ldots, \text{nmol}$, $j = 1, \ldots, \text{natom}(i)$, let

$$a^{ij} = (a_1^{ij}, a_2^{ij}, a_3^{ij})$$

be the coordinates of the $j-$th atom of the $i-$th molecule.

Suppose that the $i - th$ molecule is sequentially rotated around the axes x_1, x_2 and x_3, being $\theta^i = (\theta_1^i, \theta_2^i, \theta_3^i)$ the angles that define such rotations. Moreover, suppose that after these rotations, the whole molecule is displaced so that its barycenter becomes $c^i = (c_1^i, c_2^i, c_3^i)$. These movements transform the coordinates a^{ij} to

$$p^{ij} = (p_1^{ij}, p_2^{ij}, p_3^{ij}).$$

Observe that p^{ij} is always a function of (c^i, θ^i), the relation being

$$p^{ij} = c^i + R(\theta^i)a^{ij}, \tag{1}$$

where $R(\theta^i)$ is the rotation matrix given by

$$R(\theta^i) = \begin{pmatrix} c_1^i c_2^i c_3^i - s_1^i s_3^i & s_1^i c_2^i c_3^i + c_1^i s_3^i & -s_2^i c_3^i \\ -c_1^i c_2^i s_3^i - s_1^i c_3^i & -s_1^i c_2^i s_3^i + c_1^i c_3^i & -s_2^i s_3^i \\ c_1^i s_2^i & s_1^i s_2^i & c_2^i \end{pmatrix}, \tag{2}$$

in which $s_k^i \equiv \sin\theta_k^i$ and $c_k^i \equiv \cos\theta_k^i$, for $k = 1, 2, 3$.

Our objective is to find angles θ^i and displacements c^i, $i = 1, \ldots, \text{nmol}$, in such a way that, for all $i, = 1, \ldots, \text{nmol}$, $j = 1, \ldots, \text{natom}(i)$, the point whose coordinates are $(p_1^{ij}, p_2^{ij}, p_3^{ij})$ satisfy the constraints imposed to the atom j of the molecule i. In addition, we wish that for all $i \neq i'$, $j = 1, \ldots, \text{natom}(i)$, $j' = 1, \ldots, \text{natom}(i')$,

$$\|p^{ij} - p^{i'j'}\| \geq d_{\text{tol}}, \tag{3}$$

where $d_{\text{tol}} > 0$ is a user-specified tolerance. The symbol $\|\cdot\|$ stands for the usual Euclidian distance. In other words, the rotated and displaced molecules must remain in the desired region and the distance between any pair of atoms of different molecules must not be less than d_{tol}.

A large variety of positioning constraints may be required individually to the atoms. Let r^{ij} be the number of constraints that apply to the $j-$th atom of the $i-$th molecule. In practice, the constraints may be applied to a subset of atoms of all the molecules of the same type, or to all the atoms of a molecule, or to any desired subset of atoms or molecules, but, again, this is irrelevant for the model description.

These constraints can be represented as

$$g_\ell^{ij}(p^{ij}) \leq 0, \ell = 1, \ldots, r^{ij}. \tag{4}$$

Examples of the positioning constraints that may be required for the atoms will be shown in Section 5.

In our approach, the constraints are incorporated into the objective function. This means that, for each atom, the objective function contains a part associated with the distance to other atoms and a part associated with the fulfillment of the geometrical constraints that are imposed to it. Given the position of the atom in space, the quantity $g_\ell^{ij}(p^{ij})$ is positive if the constraint is not satisfied and negative otherwise. This justifies the addition of the terms of the form $[\max\{0, g_\ell^{ij}(p^{ij})\}]^2$ to the objective function.

The objectives (3–4) lead us to define the following merit function f:

$$\begin{aligned} f(c,\theta) &= \sum_{i=1}^{\text{nmol}} \sum_{j=1}^{\text{natom}(i)} \left(\sum_{i'=i+1}^{\text{nmol}} \sum_{j'=1}^{\text{natom}(i')} \max\{0, d_{\text{tol}}^2 - \|p^{ij} - p^{i'j'}\|^2\}^2 \right) \\ &+ \sum_{i=1}^{\text{nmol}} \sum_{j=1}^{\text{natom}(i)} \left(\sum_{\ell=1}^{r^{ij}} \max\{0, g_\ell^{ij}(p^{ij})\}^2 \right), \end{aligned} \tag{5}$$

where $c = (c^1, \ldots, c^{\text{nmol}}) \in I\!R^{3\times\text{nmol}}$ and $\theta = (\theta^1, \ldots, \theta^{\text{nmol}}) \in I\!R^{3\times\text{nmol}}$. (Remember the dependence of p^{ij} on the variables (c^i, θ^i) and the constants a^{ij} given by (1-2).) Note that $f(c, \theta)$ is non-negative for all angles and displacements. Moreover, f vanishes if, and only if, the objectives (3–4) are fulfilled. This means that, if we find displacements and angles where $f = 0$, the atoms of the resulting molecules fit the desired region and are sufficiently separated. This leads us to define the following unconstrained minimization problem:

$$\text{Minimize } f(c, \theta). \tag{6}$$

The number of variables is $6 \times$ nmol (three angles and a displacement in the three-dimensional space per molecule). The analytical expression of the derivatives of f is cumbersome but straightforward.

The definition (5) possesses two nice features. One one hand, the functional value vanishes at the solutions of the packing problem, where no overlaps are present and all constraints are satisfied. Therefore, the global minimizers of (5) are recognized trivially. On the other hand, the function f is continuous and first-order differentiable. These are important advantages over the explicit minimization of the potential energy of the system. In fact, most force-fields of molecular mechanics contain non-differentiable terms (both the Van-der-Waals and Electrostatics at $d = 0$), which do not represent problems for a stable simulation, but introduce instabilities that hamper the process of obtaining a suitable initial configuration by direct energy minimization. For example, a system may have an overall acceptable energy, but a single unsatisfactory interatomic distance can lead to simulation instabilities. On the other hand, by minimizing the packing function and recognizing a global solution with $f = 0$ one is sure that there are no clashes, and energy minimization and/or simulations may run smoothly (even if the total energy of the system is not satisfactory and needs to be equilibrated to the thermodynamic energy at the desired temperature) [1].

The smoothness of the objective function facilitates the minimization procedure to the point that global solutions are obtained frequently. At the same time, the objective function can be evaluated using very fast procedures due to its local character (there are no long-range interactions), as will be described below.

3 Function and gradient evaluation

The expression (5) has two terms: the first term reflects the minimum-distance requirement between atoms of different molecules; and the second one corresponds to the fulfillment of constraints. The second term can be computed in linear time (with respect to the number of atoms), whereas the first one involves all the atom-to-atom distances. Therefore, in principle, the computational cost of the first term calculation could increase as the square of the number of atoms, being impractical for large problems.

We implemented the fast evaluation of the first term using a Linked Cell technique [7]. This technique is related to well-known multipole ideas [8] but it is even simpler since our approach does not involve long-range interactions. Here, the system is partitioned into small boxes (bins) of side d_{tol}. Each atom is assigned to each bin using simple arithmetic operations. Then, since the objective function vanishes for distances greater than d_{tol}, for each atom only the distances to atoms belonging to the same bin or to adjacent bins is necessary. If the atoms are reasonably well distributed in space, the time required to evaluate these distances only depends on the number of atoms per bin, scaling linearly with the total number of atoms [7]. Since d_{tol} is of the order of a few (usually 2.0) angstroms, we observe that only about 10 distances must be computed for each atom, providing the algorithm with great efficiency.

A good external bounding box not exceeding the real size of the system is necessary

for an efficient partition of the space into bins. The number of bins must be small in order to avoid looping over empty boxes. We note that a "bounding box" in which all molecules are included cannot be deduced from the desired constraints. For example, for building spherical micelles, one usually imposes that the polar head of the lipids must be *outside a sphere* of a given radius, whereas the tail must be inside some other sphere. The presence of the *outer sphere* constraint makes the definition of an external bounding box quite cumbersome in general, and the structure of the molecules that are going to be packed should be taken into account.

In order to define a suitably practical bounding box, we decided to solve first an auxiliary packing problem, as follows. The position of the molecules are randomly generated within a very large region and the problem of packing all the molecules into the regions defined by the geometrical constraints *ignoring* the distances between their atoms is solved. From the solution of this easy problem, we obtain maximum and minimum coordinates for each atom of the system, and these coordinates define the external bounding box. Since the initial positions of the molecules are generated in a very large box, the solution will contain the molecules generally close to the boundaries of the constraints, and a good estimate of the total size of the system is obtained.

This procedure is also useful in a different sense: Sometimes the user imposes a spatial constraint for some of the molecules of the system which cannot be satisfied (the molecule does not fit into the region). If the algorithm fails to put the molecules into the desired regions defined by the constraints, it is almost certain that the constraints are not well defined, and the desired packing will not be successful. Therefore, by solving this initial problem the software recognizes inconsistencies in the input information and stops with an appropriate error message.

4 Optimization and Parallelization

4.1 Optimization method

For solving the molecular packing problem it is necessary to minimize the function $f(c, \theta)$. In order to achieve this purpose, an efficient local optimization algorithm is required. The first version of Packmol used Box-Quacan [9] whereas, in the present version, we decided to use Gencan [10], which turned out to be more efficient. As most optimization algorithms [11, 12], Gencan is an iterative method that, starting from an initial approximation for (c, θ), computes, at each step, an approximation with smaller function value. In the limit, a local minimizer is obtained. At each iteration, Gencan calculates a direction in the (c, θ) space along which the objective function f decreases. The new approximation to the solution is computed in the half-line determined by the above mentioned *descent direction* with origin in the current approximation. The descent direction is defined according to the Newtonian paradigm, which means that it is the best possible direction in the case that the objective function possesses a simple (quadratic) structure. The procedure for finding a new point along this direction uses sophisticated interpolation and extrapolation techniques that allows one to obtain efficiency and to guarantee ultimate convergence. Gencan is available at the Tango Project web page

(see http://www.ime.usp.br/∼egbirgin/tango/).

We have used GENCAN in combination with heuristics devised to enhance the convergence to global (i. e. $f = 0$) minimizers. One of the obstacles associated to the global minimization of $f(c, \theta)$ (also causing slow convergence) was observed to be the formation of clusters of atoms which, by the local nature of the objective function, are hard to disperse. This is because the algorithm cannot recognize the presence of "empty spaces" far from the clusters. In order to avoid the formation of these clusters and enhance the convergence to global solutions, two strategies were adopted. First, the packing process starts with a larger minimum-distance requirement than the one specified by the user. Thus, the packing method begins aiming a larger separation of the atoms than the required by the tolerance d_{tol}. The effect of this is to expand possible clusters and to occupy remaining empty spaces. In the current version of PACKMOL the initial tolerance is 10% larger than d_{tol}. Using this initial tolerance, ten GENCAN iterations are performed. The procedure is repeated while the objective function with the actual d_{tol} decreases at the end of each ten-iteration loop. Then, the actual d_{tol} is restored and the final optimization steps are performed. Usually, if the density of a system is close to the water density at room temperature, one requires a target tolerance $d_{\text{tol}} = 2$Å, but most times a solution is found before, with a larger minimum interatomic distance. As a second heuristic procedure, the function value is evaluated for each molecule separately, and the 5% worst molecules are moved to the vicinity of the 5% best molecules, aiming a more homogeneous distribution. The classification of the function value for the molecules is done by the also linear-scaling algorithm Flashsort [13]. This procedure is applied to each type of molecule independently.

Some other procedures were employed to improve the reliability and efficiency of the method: When many types of molecules are present, the packing problem is solved for each type independently, and only as a final step all the molecules are packed together. In this way, we take advantage of the fact that many times each type of molecule occupies a different region in space (as the two components of an interface, or the lipids and the solvent in micelles, for example). The packing process runs faster for each type of molecule and, if the regions do not overlap, the independent solutions are very close to an actual solution of the whole packing problem. Finally, when large fixed structures are present, such as proteins, our initial approximations automatically exclude solvent molecules that clash with the big molecules. With these strategies, global solutions are obtained for all kinds of mixtures of small molecules and for the solvation of large fixed molecules. When larger molecules (as lipids) are considered as variables, the whole packing process may not succeed, but generally the best configuration found is satisfactory. The improvement of the heuristics in order to obtain global solutions for a greater number of systems is a subject of continuous research.

4.2 Parallelization

Experiments show that 85% of the CPU time used to solve the optimization problem (6) is employed in the evaluation of the objective function (5) and its derivatives. Motivated by this observation, we decided to develop a parallel code for these computations. We note

that PACKMOL, even without parallelization, is generally quite fast, being able to solve a problem with about a hundred thousand atoms in some minutes in a typical personal computer. Therefore, the aim of the parallel version is to take advantage of the multiple-core architectures available in current personal computers. The parallelization to massive parallel computations is not a priority.

The strategy for the parallelization of the objective function evaluation consists on assigning the task of computing the distances of an atom to its neighbors to individual processes. Each process computes a *partial sum* and then all these terms are added in order to obtain the final *sum*.

Since function and gradient evaluations represent, in the sequential version of the code, 85% of the CPU time, the theoretical speedup of the parallel implementation of PACKMOL with ρ processors is given by

$$S = \frac{1}{0.15 + 0.85/\rho}.$$

For example, running the parallel version of PACKMOL in a machine with $\rho = 4$ cores, it is expected to obtain a 2.759 times speedup relative to using a single core.

The parallel Fortran 77 version of PACKMOL was implemented using the OpenMP Application Program Interface (API) [14] that supports multi-platform shared-memory parallel programming in C/C++ and Fortran. Codes were compiled with gfortran (GNU Fortran compiler, version 4.2) and the optimization compiler options "-O4 -ffast-math -mcmodel=medium" for the serial version, and the same options plus "-fopenmp" for the parallel version. All the experiments were run in a computer with two 2.83GHz Intel(R) Xeon(R) Quad-Core processors and 4GB of RAM memory.

We illustrate the behaviour for two problems:

Problem A: A proof-of-concept 1.4 million atom mixture of water and urea. This example illustrates the behavior of PACKMOL for very large systems, for which the use of the parallel version will be mostly recommended.

Problem B: A protein solvated by water and ions. The system was actually used in practical MD simulations [6] and bears 53,769 atoms.

The details of these systems can be seen in the Examples section. Figure 1 illustrates the empirical speedup compared with the theoretical one for these two examples.

The parallel version is 2.75 times faster with four cores than with one core, in agreement with the predicted scalability. On the other side, running in a single core, it is $\sim$1.6 times slower than the serial version, due to the overhead associated to the usage of the application programming interface that enables the multi-platform shared memory multiprocessing programming.

5 Packmol usage

With PACKMOL the user can pack molecules (or other objects defined as groups of points) restricted to regions of the space whose shapes are defined by geometrical constraints. Full instructions on the usage of the software can be obtained in the PACKMOL site.

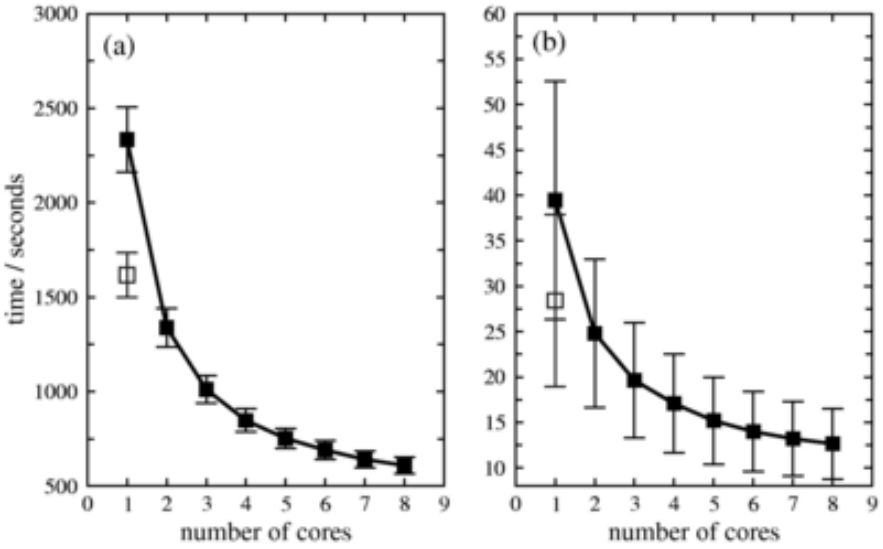

Figure 1: Scalability of the parallel version of PACKMOL in two examples. (a) A 1.4 million atom mixture of water and urea and (b) protein solvated in water and ions. The open square indicates the time of the serial version. The bars display standard deviations of ten runs with different random initial points.

PACKMOL is distributed as free software under the GNU General Public License. It can be downloaded from

http://www.ime.unicamp.br/∼martinez/packmol

A user-guide and some auxiliary applications are also maintained in this page.

5.1 The input data and basic input syntax

The user must provide the coordinates of one molecule of each type. For example, in order to solvate a protein with water and urea, the user must provide a file containing the structure of a single water molecule, a file containing the structure of a single urea molecule, and the protein structure. These structures may be given in Tinker, Molden's XYZ, Moldy or PDB formats.

The current version of PACKMOL uses a user-friendly input syntax. For example, a minimum input file could be:

```
filetype pdb
output waterbox.pdb
tolerance 2.0
structure water.pdb
  number 1000
  inside cube 0.  0.  0.  31.
end structure
```

With this input, PACKMOL will fill a cube of side 31Å with 1000 water molecules (density 1 g/ml). Every pair of atoms of different molecules will be separated by, at least, 2.0 Å, and the molecules will be, very likely, well distributed inside the cube. More types of molecules could be added by additional sections of the form

```
structure ...  end structure
```

and, within each structure, additional constraints may be set.

5.2 Atom selections

An additional functionality of the current PACKMOL version is the ability to restrict some atoms of a structure to spatial regions independently of the other atoms. This may be useful for building vesicles, where the hydrophilic part of the surfactants must be pointing to the aqueous environment, or lipid layers, in which all lipids are approximately aligned, for example. Suppose that the coordinate file of a single molecule contains, say, 10 atoms. One could restrict atoms 9 and 10 to a particular region using the keyword `atoms`, for example:

```
structure molecule.pdb
  inside cube 0.  0.  0.  20.
  atoms 9 10
    inside box 0.  0.  15.  20.  20.  20.
  end atoms
end structure
```

In this case, all the atoms of the molecule will be placed inside the defined cube and, additionally, atoms 9 and 10 will be inside the box.

5.3 Types of constraints

Six types of geometrical constraints may be imposed to whole molecules or to some atoms: *cubes*, *boxes*, *spheres*, *ellipsoids*, *cylinders*, and *planes*. Of course, cubes are particular cases of boxes and spheres are particular cases of ellipsoids but, in order to simplify the usage, they have their independent input. For closed regions (cubes, boxes, spheres, ellipsoids and cylinders) there are two possibilities: *inside* and *outside*. For open regions (planes) the possibilities are *over* and *below*. In addition, individual molecules (a protein, for instance) can be *fixed* in some desired position. Therefore, a total of 13 different combinations of

keywords can be used to order the system. The details on how to use each of these options can be obtained in the PACKMOL software documentation.

6 Examples

Figure 2 presents some examples of systems built with PACKMOL, that illustrate some of the capabilities of the package. The corresponding input files can be obtained in the PACKMOL site or upon request to authors.

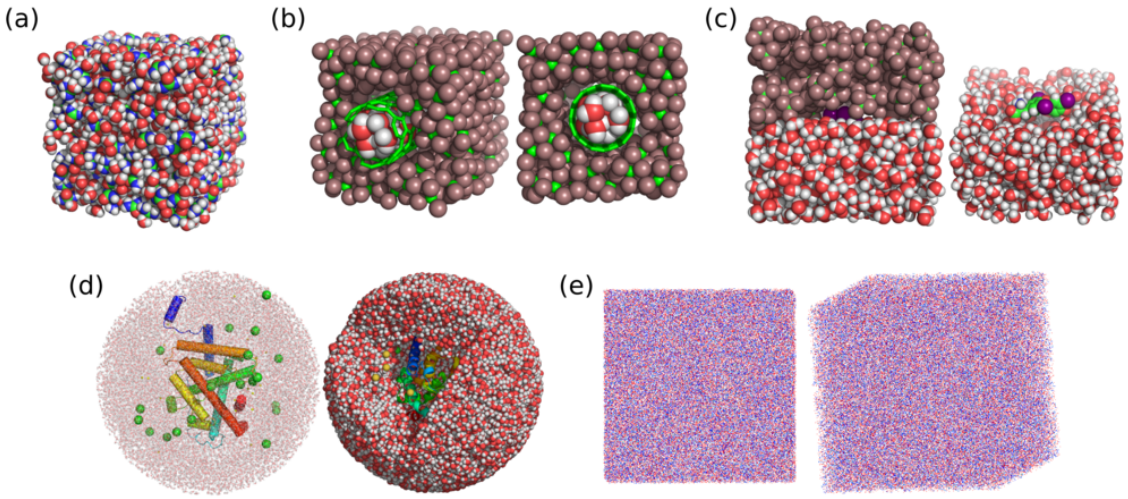

Figure 2: Examples of simulations boxes built with PACKMOL: (a) A mixture of water and urea. (b) Carbon nanotube containing water inside and carbon tetrachloride outside. (c) Water and carbon tetracloride with a hormone in the interface. (d) Thyroid hormone receptor solvated spherically by water and ions. (e) Proof-of-concept million-atom example: a mixture of water and urea. More details of these examples are described in Table 1.

Table 1 summarizes the components of the systems and the computational time required to solve them. All these examples were run in a Sony Vaio VGN-NR11Z/S laptop with Intel Core 2 Duo T7250 processor and 2 Gb of RAM, running Ubuntu 8.04 Linux. The computational times correspond to the serial version of the package compiled with the GNU Fortran Compiler (g77) version 3.4.6 with the "-ffast-math -O3" flags.

In order to illustrate a complete PACKMOL input file, the one corresponding to the example (b) is given in Figure 3.

Lines 1 to 3 define general aspects of the system: the type of structure files provided will be `pdb`, the minimum distance between atoms of different molecules at the solution will be `2.0` Å and the output file will be called `solvtube.pdb`. The carbon nanotube will be placed according to its coordinates in the structure file provided (no rotations, no translations). Then, 45 water molecules will be put inside the nanotube, by restricting them to be inside a cylinder that starts at the minimum coordinates of the tube (`0. 0. -11.7`), is oriented in the z-axis (`0. 0. 1.`), has a radius of `4.0` Å and a length of

System	Composition	N_{atoms}	N_{var}	t
(a)	400 urea molecules 1,000 water molecules	6,200	8,400	5 s
(b)	45 water molecules 150 CCl_4 molecules One carbon nanotube	1,125	1,170	16 s
(c)	1,019 water molecules 199 CCl_4 molecules One T3 (thyroid hormone)	4,087	7,308	3 s
(d)	16,500 water molecules 20 Cl^- ions 30 Na^+ ions Thyroid Hormone Receptor LBD	53,796	99,300	37 s
(e)	216,000 water molecules 86,400 urea molecules Memory requirement: $\sim$400 Mb	1,339,200	1,814,400	37 min

Table 1: Properties of the examples of Figure 2. N_{atoms} is the number of atoms of the system, N_{var} is the number of variables of the optimization problem, and t is the running time.

```
filetype pdb
tolerance 2.0
output solvtube.pdb
structure nanotube.pdb
  fixed 0.  0.  0.  0.  0.  0.
end structure
structure water.pdb
  number 45
  inside cylinder 0.  0.  -11.7 0.  0.  1.  4.0 23.4
end structure
structure ccl4.pdb
  number 150
  outside cylinder 0.  0.  -11.7 0.  0.  1.  4.0 23.4
  inside box -10.  -10.  -11.7 10.  10.  11.7
end structure
```

Figure 3: Input file for example (b).

`23.4` Å. The carbon tetrachloride molecules, then, are put outside this same cylinder, and inside a box with minimum coordinates (`-10. -10. -11.7`) and maximum coordinates (`10. 10. 11.7`), therefore surrounding the whole system.

7 Conclusions

We have developed a software for building initial configurations for molecular dynamics simulations based on the concept of packing optimization. The software is called PACKMOL and allows the user to define the molecular system to be simulated by packing different types of molecules into regions defined by geometric constraints. Function and gradient evaluations are optimized to the point that millions of atoms can be packed in reasonable time, and initial configurations for state-of-the-art simulations can be built in few minutes or seconds. The user must provide only the structures of one molecule of each type and the geometrical constraints that must be fulfilled. It is possible to build mixtures of several components, solvate proteins, and build ordered arrangements as double layers or micelles. The package is currently compatible with Tinker, Moldy, Molden's XYZ and PDB file formats. It is free software and is available online at

`http://www.ime.unicamp.br/~martinez/packmol`

PACKMOL has already been used for building initial configurations for different applications in different research groups, such as protein solvation with different solvents [6, 16], multiple component mixtures or uncommon liquids [17, 18], ionic liquids [19], polymer solutions [20], interfaces [4, 5], and others [21, 22, 23, 24, 25, 26, 27, 28].

Further improvements of the package may include macro-input keywords to build the most common systems and more complex unit cells, the parallelization of tasks other than function and gradient evaluation, the improvement of global convergence heuristics, and the development of a graphical user interface, or its incorporation as a plugin into some graphical molecular viewer.

Acknowledgements

The authors thank the Brazilian national funding agencies CAPES and CNPq and the state of São Paulo agency FAPESP for financial support. We also acknowledge two anonymous referees whose suggestions improved the quality of the paper.

References

[1] Martínez, J. M.; Martínez, L. J Comp Chem 2003, 24, 819-825.

[2] Balabin, I. Membrane Plugin, Version 1.1; 2008.

[3] Jo, S.; Kim, T.; Im W. PLoS ONE 2007, 2, e880.

[4] Martins, L. R.; Skaf, M. S.; Ladanyi, B. M. J Phys Chem B 2004, 108, 19687-19697.

[5] Moreira, N. H.; Skaf, M. S. Prog Coll Pol Sci 2004, 128, 81-85.

[6] Martínez, L.; Webb, P.; Polikarpov, I.; Skaf, M. S. J Med Chem 2006, 49, 23-26.

[7] Griebel, M.; Knapek, S.; Zumbush, G. Numerical Simulation in Molecular Dynamics; Springer, Berlin-Heidelberg, 2007.

[8] Greengard, L.; Rokhlin, A. J Comp Phys 1987, 73, 325-348.

[9] Friedlander, A.; Martínez, J. M.; Santos, S. A. Appl Math Opt 1994, 30, 235-266.

[10] Birgin, E. G.; Martínez, J. M. Comp Optim Appl 2002, 23, 101-125.

[11] Dennis Jr. J. E.; Schnabel, R. B. Numerical Methods for Unconstrained Optimization and Nonlinear Equations; Prentice-Hall, Englewoods Cliffs, 1983.

[12] Luenberger, D. G. Linear and Nonlinear Programming; Addison-Wesley, California, 1984.

[13] Neubert, K-D. Dr Dobbs J 1998, 23, 123-129.

[14] http://www.openmp.org

[15] J. T. Frey, D. J. Doren, TubeGen 3.3; University of Delaware, Newark, DE, 2005.

[16] Ander, M.; Luzhkov, V. B.; Aqvist, J. Biophys J 2008, 94, 820-831.

[17] Gardner, A.; Vasquez, V. R.; Clifton, A.; Graeve, O. A. Fluid Phase Eq 2007, 264-270.

[18] Aparicio S.; Alcalde, R.; Davila, M. J.; Garcia, B.; Leal, J. M. J Phys Chem B 2007, 111, 4417-4431.

[19] Davilla, M. J.; Aparicio, S.; Alcalde, R.; et al. Green Chem 2007, 9, 221-232.

[20] Costa, L. T.; Ribeiro, M. C. C. J Chem Phys 2006, 124, 184902.

[21] Siqueira, L. J. A.; Ribeiro, M. C. C. J Phys Chem B 2007, 111, 11776-11785.

[22] Riihimaki, E. S.; Martínez, J. M.; Kloo, L. J Phys Chem B 2007, 111, 10529-10537.

[23] Waibel, B.; Scheiber, J.; Meier C.; et al. Europ J Org Chem 2007, 18, 2921-2930.

[24] Aparicio, S. J Phys Chem A 2007, 111, 4671-4683.

[25] Luzhkov, V. B.; Almlof, B.; Nervall, M.; et al. Biochemsitry 2006, 45, 10807-10814.

[26] Mudi, A.; Chakravarty, C. J Phys Chem B 2006, 110, 8422-8431.

[27] Riihimaki, E. S.; Martínez, J. M.; Kloo, L. J Mol Struct 2006, 760, 91-98.

[28] Preto, M. A. C.; Melo, A.; Maia, H. L. S.; et al. J Phys Chem B 2005, 109, 17743-17751.