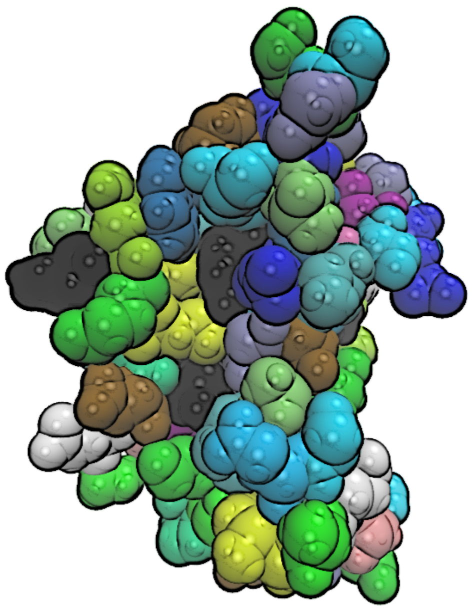
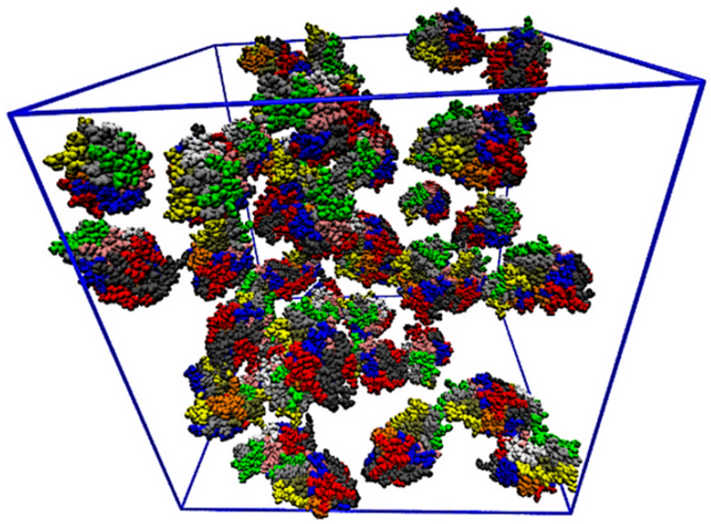
BD_BOX
version 2.2
user's manual – bd_rigid

Paweł Zieliński

Maciej Długosz





Contact authors:

Paweł Zieliński: pzielins@cent.uw.edu.pl

Maciej Długosz: mdlugosz@cent.uw.edu.pl

Work on the BD_BOX package is supported by National Science Centre, grant number N N519 646640

BD_BOX is a scalable Brownian dynamics package for UNIX/LINUX platforms. BD_BOX is written in C and uses modern computer architectures and technologies:

- MPI technology for distributed-memory architectures
- OpenMP technology for shared-memory systems
- NVIDIA CUDA framework for GPGPU
- SSE vectorization for CPU

Within the BD_BOX framework macromolecules can be represented either with flexible bead models (the *bd_flex* module of the BD_BOX package) or as rigid bodies (the *bd_rigid* module of the BD_BOX package).

When flexible models are employed, each molecule consists of a various number of spherical subunits (beads) connected with deformable bonds [1, 2]. Bonded interactions that result in deformations of planar and dihedral angles can be also modeled. Direct, nonbonded interactions between molecules are evaluated using pairwise functions describing screened electrostatics in dielectric media [3] with effective charges assigned to spherical subunits [4] and Lennard-Jones potential types. The far-field hydrodynamic effects are modeled using the configuration dependent Rotne-Prager-Yamakawa mobility tensor [5, 6, 7] or its Ewald-summed form in case of periodic systems [8, 9]. Equations of motions are propagated using either the Ermak-McCammon [10] scheme or the predictor-corrector IG-T algorithm by Iniesta and Garcia de la Torre [11]. Hydrodynamically correlated random displacements are generated either via the Cholesky factorization of the configuration-dependent mobility tensor matrix [10], using the TEA-HI approach proposed by Geyer and Winter [12, 13], or with an approach described by Ando and Skolnick [14] that utilizes Krylov subspaces. With BD_BOX one can simulate flexible molecules in homogeneous flows [2] or external electric fields (direct, alternate or rotating fields).

With BD_BOX one can also simulate rigid bodies that are described with fully anisotropic diffusion tensors [15]. Molecules, treated as rigid bodies can be described either using a coarse-grained representation or with fully atomistic details. In the latter case, intermolecular interactions may include electrostatic, hydrophobic and Lennard-Jones potentials. External electric fields can also be applied to the simulated system. Hydrodynamic interactions between molecules modeled within the rigid-body framework are currently not supported.

BD_BOX simulations can be performed without or with boundaries; in the latter case the containing or periodic boundary conditions can be used. With BD_BOX one can effectively simulate either single molecules or multimolecular systems composed of large numbers of different species. For efficient simulations of dense systems we implemented algorithms preventing the overlapping of diffusing molecules [16, 17, 18].

This manual describes the usage of the *bd_rigid* module of the BD_BOX package. The *bd_flex* module for BD simulations of flexible molecular models, is described separately.

Contents

1 Requirements

The BD_BOX package is distributed as source code. A UNIX/LINUX *make* tool is needed to build a working binary from the source code (see below). CPU versions of BD_BOX binaries can be run either in a serial mode or in parallel, either on shared-memory machines using OpenMP and MPI libraries or on architectures with distributed memory using the MPI library. GPU versions of BD_BOX binaries require the CUDA Toolkit (obtainable freely at *http://developer.nvidia.com*). Both CPU and GPU versions can be compiled with the support for either single or double-precision floating-point arithmetic.

For the generation of random Gaussian numbers, BD_BOX uses functions implemented in the GNU Scientific Library (GSL) (*http://www.gnu.org/software/gsl*) that are based on the Mersenne Twister algorithm of Matsumoto and Nishimura [19]. We also implemented a function that uses the standard system routine *drand48()* and the polar Box-Muller transformation [20].

A generic, sequential algorithm implemented in BD_BOX for the Cholesky factorization, follows the *choldc()* subroutine described in [21]. However, it is better to compile BD_BOX with the implemented support for the LAPACK (or SCALAPACK) library (*http://www.netlib.org/lapack*) that offers efficient, highly memory-optimized routines for the Cholesky factorization. LAPACK performs the vast amount of operations by exploiting BLAS. Machine-specific implementations of BLAS are available and such implementations can be crucial for the performance of BD_BOX. The Cholesky factorization on GPU relays on the implemented support for the MAGMA dense linear algebra library (*http://icl.cs.utk.edu/magma*), which provides functionality of LAPACK.

2 Installation

The *configure* shell script, written by GNU Autoconf is used to build the BD_BOX binaries. After downloading and unpacking the compressed BD_BOX archive:

```
gzip -d bd_box-ver.tar.gz
tar -xvf bd_box-ver.tar
```

the user should execute the *configure* script from within the BD_BOX directory:

cd ./bd_box-ver

./configure

configure takes instructions from *Makefile.in* and builds *Makefile* and some other files that work on the user's system. We provide the INSTALL file and some other files (consult the README file in the bd_box-ver directory) with flags and options that can be passed to the *configure* script to tune the compilation parameters on a particular system. Next, the:

make

command followed (optionally) by the:

make install

should be executed. As a result of the compilation process, two binaries: *bd_flex* and *bd_rigid* should be created, either within the *bd_box-ver/src* directory, or in the directory passed by the user to the *configure* script (with the *–prefix* option). It is possible to prevent compilation of either one of binaries using *–disable–flex* or *–disable-rigid* flags.

Tools distributed with the BD_BOX package should be compiled separately. Their sources along with appropriate *Makefiles* ale located in *bd_box-ver/tools/tool_name* directories. Each tool can be compiled using the:

make all

command.

Units

charge: e

length: Å

time: ps

temperature: K

energy: $\frac{kcal}{mol}$

3 Rigid-body Brownian dynamics

3.1 Molecular Models

Within the BD_BOX rigid-body framework molecules can be described either using a coarse-grained representation or with fully atomistic details. In both cases molecules are described as conglomerates of spherical subunits (i.e. atoms). Each kind of interaction that the molecule is involved in is assigned its own representation - for example for a given molecule a set of charged spheres can be used to evaluate electrostatic interactions and a different set (considering both positions and radii of spheres) can be used to evaluate excluded volume interactions with other molecules in the studied system (consult Figure 1 for the overall concept).

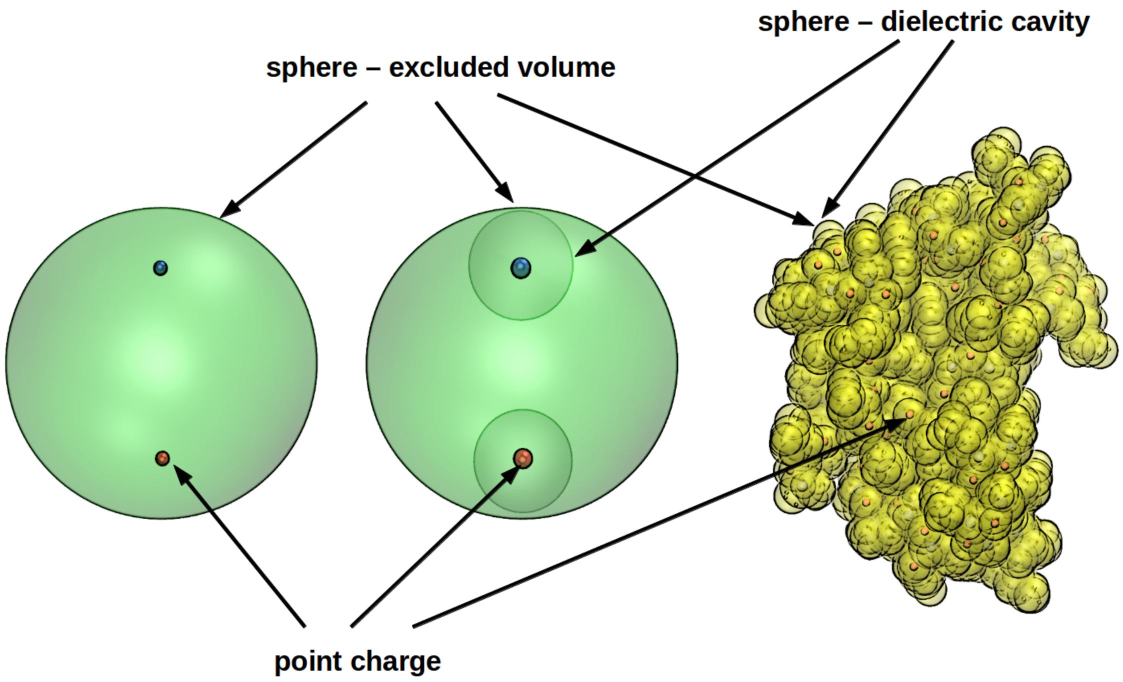


Figure 1: Exemplary molecular models that are tractable within the BD_BOX rigid-body framework. Models consist of spheres with radii defining excluded volume interactions (left). Point charges can be positioned inside molecules (their positions do not necessarily coincide with centers of spheres used to define excluded volume interactions (left)). Additionally, each charge can be assigned a non-zero radius (center). Charge radii are used to define the dielectric cavity occupied by the molecule. Positions and radii of spheres in the dielectric representation and in the excluded volume representation can be the same as for example in case of atomistically detailed molecular models - a molecule is then represented by a set of van der Waals spheres (atoms) with central point charges (right).

3.2 Rigid-body Diffusion Tensor Matrix

In dilute solutions, diffusional properties of molecules are determined by their size and shape, as well as by the temperature and viscosity of the solvent. The information required to characterize translational and rotational movements of quasi-rigid Brownian particles in dilute solutions is contained in their diffusion tensors. Single-particle diffusion tensor, $\boldsymbol{D}$, is represented by a symmetric, 6×6 matrix that consists of four 3×3 blocks. These blocks are related to translational (*tt*) and rotational diffusivities (*rr*) and their couplings (*rt,tr*) [15]:

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{D}^{tt} & \boldsymbol{D}^{tr} \\ \boldsymbol{D}^{rt} & \boldsymbol{D}^{rr} \end{pmatrix} \tag{1}$$

Diffusion tensors of molecules can be obtained theoretically from rigid-body hydrodynamic calculations performed for detailed molecular models [22]. For a spherical particle the above matrix takes a diagonal form (isotropic diffusion) in which identical elements D_{ii}^{tt} correspond to the translational diffusion coefficient of a sphere. Similarly, identical elements D_{ii}^{rr} give its rotational diffusion coefficient. For an arbitrarily-shaped object its average translational and rotational diffusion coefficients are defined, respectively, as: $D^t = \frac{1}{3}Tr\left(\boldsymbol{D}^{tt}\right)$ and $D^r = \frac{1}{3}Tr\left(\boldsymbol{D}^{rr}\right)$, where the diffusion tensor (Equation 1) is evaluated in the molecule-fixed coordinate system with the origin located at the particle's diffusion center, and whose axes coincide with the principal axes of the rotational diffusion tensor.

3.3 Equations of Motions

The propagation scheme for an arbitrarily-shaped rigid body (that is described with a fully-anisotropic diffusion tensor) can be written in the molecule-fixed frame (again, by molecule-fixed frame we mean that the origin of the frame is assumed to be located at the diffusion center of the molecule and its axes coincide with the principal axes of diffusional rotations) as [10, 15]:

$$\Delta\vec{x}(\Delta t) = \frac{\Delta t}{k_B T}\boldsymbol{D}\vec{M} + \vec{R}(\Delta t) \tag{2}$$

where:

$\boldsymbol{D}$ - the precomputed (and constant in the simulation) 6×6 diffusion tensor of the molecule evaluated in the molecule-fixed coordinate system with the origin located at the molecule's diffusion center and axes coinciding with the principal axes of particle's rotations (so that the submatrix $\boldsymbol{D}^{rr}$ in Equation 1 is diagonal)

Δt - the time step

$\vec{x}$ - the vector describing the position of the diffusion center ($\vec{r}$) and orientation ($\vec{\phi}$) of the molecule, $\vec{x} = (\vec{r}, \vec{\phi})^T$

$\vec{M}$ - the generalized force vector, resulting, for example, from interactions with other molecules or external fields and having two components: the total force ($\vec{F}$) and the total torque ($\vec{T}$) referred to the molecule's diffusion center, $\vec{M} = (\vec{F}, \vec{T})^T$.

$\vec{R}(\Delta t)$ - random displacement vector arising from the Brownian noise, with zero mean, and the variance-covariance given with:

$$< \vec{R}(\Delta t)\vec{R}(\Delta t) >= 2\boldsymbol{D}\Delta t \tag{3}$$

Random displacements of the particle can be computed via Cholesky decomposition of its diffusion tensor [10]. Brownian dynamics trajectory of an object in the laboratory frame is obtained by applying at each BD simulation step properly defined transformations - translations and rotations [15] - between the molecule-fixed and the laboratory coordinate frame.

3.4 Rotations

The following matrix is utilized to perform rotational motions [23]:

$$M = \begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{pmatrix}$$

where:

$$M_{11} = \frac{(\Omega_y^2 + \Omega_z^2) cos\Omega + \Omega_x^2}{\Omega^2}$$

$$M_{12} = \frac{\Omega_x \Omega_y}{\Omega^2}(1 - cos\Omega) - \frac{\Omega_z}{\Omega} sin\Omega$$

$$M_{13} = \frac{\Omega_x \Omega_z}{\Omega^2}(1 - cos\Omega) + \frac{\Omega_y}{\Omega} sin\Omega$$

$$M_{21} = \frac{\Omega_x \Omega_y}{\Omega^2}(1 - cos\Omega) + \frac{\Omega_z}{\Omega} sin\Omega$$

$$M_{22} = \frac{(\Omega_x^2 + \Omega_z^2) cos\Omega + \Omega_y^2}{\Omega^2}$$

$$M_{23} = \frac{\Omega_y \Omega_z}{\Omega^2}(1 - cos\Omega) - \frac{\Omega_x}{\Omega} sin\Omega$$

$$M_{31} = \frac{\Omega_x \Omega_z}{\Omega^2}(1 - cos\Omega) - \frac{\Omega_y}{\Omega} sin\Omega$$

$$M_{32} = \frac{\Omega_y \Omega_z}{\Omega^2}(1 - cos\Omega) + \frac{\Omega_x}{\Omega} sin\Omega$$

$$M_{33} = \frac{(\Omega_x^2 + \Omega_y^2) cos\Omega + \Omega_z^2}{\Omega^2}$$

where the vector:

$$\vec{\Omega} = \{\Omega_x, \Omega_y, \Omega_z\}$$

describes rotations around axes x, y, z of the molecule-fixed coordinate system, and:

$$\Omega = \sqrt{\Omega_x^2 + \Omega_y^2 + \Omega_z^2}$$

4 Intermolecular Interactions

Forces and torques acting on molecules ($\vec{M}$ in Equation 2) are evaluated analytically based on derivatives of different potential functions that are described below. During simulations, intermolecular forces are evaluated within atom-based, spherical cutoffs, defined by the user. Minimum image convention is utilized in case of periodic systems.

4.1 Excluded Volume Interactions

Excluded volume interactions are modeled using pairwise terms of form:

$$E_{ij}^{exc} = \alpha\sqrt{\epsilon_i^{LJ}\epsilon_j^{LJ}}\left(B\left(\frac{\sigma_i+\sigma_j}{r_{ij}}\right)^n - \left(\frac{\sigma_i+\sigma_j}{r_{ij}}\right)^m\right) \tag{4}$$

where:

n and m - non-negative integers

i and j - indexes that run over atoms from different molecules

σ - denotes an atomic radius

ϵ^{LJ} - the interaction well depth assigned to a given atom

α and B - user-defined constants

r_{ij} - the distance between the two interacting atoms

values of n/m can be set to: $12/6$, $12/0$, $8/6$, $8/0$.

4.2 Hydrophobic Interactions

Hydrophobic (i.e. nonpolar interactions) can be evaluated using one of the two solvent accessible surface based approaches implemented in BD_BOX: either the FACTS model [24] or the model (that we will further refer to using the therm *primitive*) similar in the spirit to the approach described by Gabdoulline and Wade [25].

4.2.1 FACTS Approach

In the FACTS model, the nonpolar contribution to the free energy of solvation of a molecule is modeled in relation to the amount of its surface exposed to the solvent (SASA) [24]:

$$E^{nonpolar} = \mu\sum_{i=1}^{N} SASA_i \tag{5}$$

where summation runs over all (N) atoms of a molecule (or molecules), μ denotes the empirical surface tension parameter [26], and atomic SASAs are given with [24]:

$$SASA_i = c_o + \frac{c_1}{1 + e^{-c_2(D_i - c_3)}} \tag{6}$$

$$D_i = \sum_{j \neq i}^{N} V_j \Theta_{ij} + d_1 \left| \frac{\sum_{j\neq i}^{N} \frac{V_j}{r_{ij}} \Theta_{ij} \hat{x}_{ij}}{1 + \sum_{j \neq i}^{N} \frac{V_j}{r_{ij}} \Theta_{ij}} \right| + d_2 \sum_{j \neq i}^{N} V_j \Theta_{ij} \left| \frac{\sum_{j\neq i}^{N} \frac{V_j}{r_{ij}} \Theta_{ij} \hat{x}_{ij}}{1 + \sum_{j \neq i}^{N} \frac{V_j}{r_{ij}} \Theta_{ij}} \right| \tag{7}$$

In the above equations V_j denotes the van der Waals volume of the j^{th} atom, r_{ij} is the distance between atoms i and j, $\hat{x}_{ij}$ is a unit vector pointing from the i^{th} atom to the neighboring j^{th} atom and Θ_{ij} is a sigmoidal weighting function:

$$\Theta_{ij} = \begin{cases} \left(1 - \left(\frac{r_{ij}}{R_i}\right)^2\right)^2, & \text{if } r_{ij} \leq R_i \\ 0, & \text{if } r_{ij} > R_i \end{cases} \tag{8}$$

where R_i is a predefined radius of a sphere (i.e. cutoff) enclosing the i^{th} atom.

Parameters c_o, c_1, c_2, c_3, d_1 and d_2 that are derived by fitting to exact values of SASA [24] are defined in the PARAM22 CHARMM parameter set [27]. For all atoms values of R_is are similar following the original parametrization of the FACTS model [24].

4.2.2 Primitive Approach

Each atom of a given molecule is assigned a constant SASA value computed in the absence of other molecules, $SASA_i^o$. The change in the SASA of a particular surface atom (i) of one molecule due to the proximity of surface atoms (j) of other molecules, $SASA_i$, is computed as:

$$SASA_i = \phi SASA_i^o \sum_j f_{ij}(r_{ij}) \tag{9}$$

where

$$f_{ij}(r_{ij}) = \frac{b - (r_{ij} - \sigma_j)}{b - a} \quad for \quad a < r_{ij} - \sigma_j < b \tag{10}$$

$$f_{ij}(r_{ij}) = 1 \quad for \quad r_{ij} - \sigma_j < a \tag{11}$$

$$f_{ij}(r_{ij}) = 0 \quad for \quad r_{ij} - \sigma_j > b \tag{12}$$

and ϕ is a user defined parameter, and σ_j is the van der Waals radius of the i^{th} atom.

Parameters a and b are also user defined (see Figure 2). Energy is computed as:

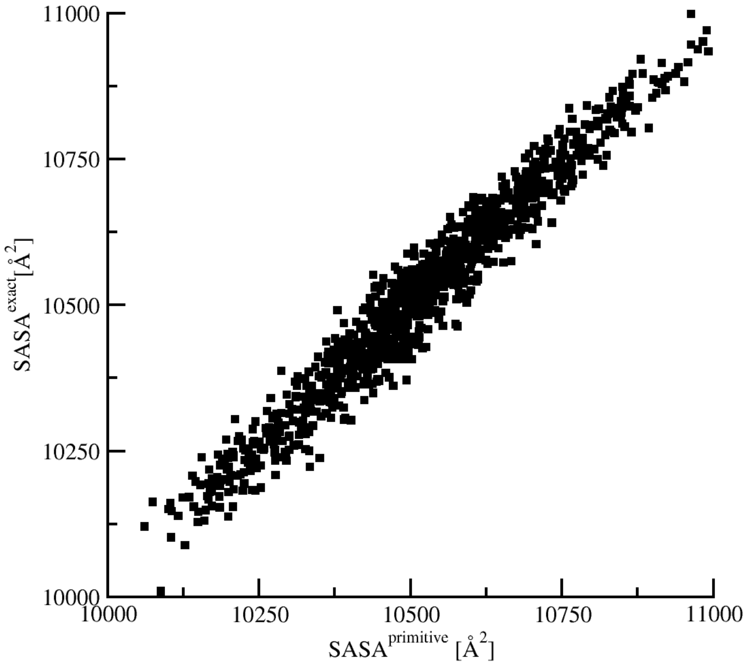


Figure 2: Performance of the primitive approach. Total values of solvent accessible surfaces calculated for 1500 different relative configurations of two small proteins, with $\phi = 0.5$, $a = 1.3$Å, and $b = 6.4$Å. All-atom protein models were used in calculations with atomic radii taken from the PARAM22 CHARMM parameter set [27]. Exact SASA values were calculated using APBS [28].

$$E^{nonpolar} = \mu \sum_{i=1}^{N} (SASA_i^o - SASA_i) \tag{13}$$

4.3 Electrostatic Interactions

4.3.1 Screened Coulombic Interactions

The energy of the electrostatic interaction (screened by dissolved ions) between two charges q_i and q_j that belong to different molecules is computed as:

$$E_{ij}^{Coulombic} = \gamma \frac{q_i q_j}{\epsilon} \frac{e^{-\kappa r_{ij}}}{r_{ij}} \tag{14}$$

where:

κ - the inverse of the Debye length

r_{ij} - the distance between the two charges

γ - the unit conversion factor (an user-defined parameter)

ϵ - the dielectric constant of the solvent.

4.3.2 Polarization terms

Polarization (desolvation) terms are evaluated between the charges of one molecule and the dielectric cavity of another molecule. Dielectric body of a molecule is treated as a conglomerate of spheres (i.e. atoms) [29]. Energy of the interaction between the i^{th} charge (q_i) of one molecule with the j^{th} dielectric sphere (radius σ_j) of another molecule is computed as [30]:

$$E_{ij}^{cavity} = \gamma\beta \frac{\epsilon - \epsilon_p}{\epsilon(2\epsilon + \epsilon_p)} \frac{(1+\kappa r_{ij})^2}{r_{ij}^4} \exp\left(-2\kappa r_{ij}\right)(\sigma_j)^3 (q_i)^2 \tag{15}$$

where:

γ - the unit conversion factor (an input parameter)

β - the user-defined (an input parameter) constant that depends on ionic strength - it also takes into account the fact that atoms in the molecule overlap

ϵ - the dielectric constant of the solvent

ϵ_p - the dielectric constant of a molecule

r_{ij} - the distance between the i^{th} charge of one molecule and j^{th} atom of another molecule

κ - the inverse of the Debye length.

5 External Fields

5.1 Bounding Sphere

A spherical surface, impenetrable for molecules, that encloses simulated system can be included in *bd_rigid* simulations. The influence of such a containing sphere on the studied system is modeled by applying a central force of form:

$$\vec{F} = -\frac{A}{(R_{sphere} - r)^n}\hat{e}_r \tag{16}$$

where:

R_{sphere} - the radius of a spherical surface

n - a non-negative integer

A - the amplitude of the force

r - the distance between a diffusion center of a particular molecule and the system's center of geometry

The above force is applied to each of the studied molecules outside a predefined cutoff.

5.2 Electric Field

It is possible to simulate with *bd_rigid* the interactions of charged molecules with external electric fields:

DC – direct current field

$$\vec{E} = E_o\hat{e}_E \tag{17}$$

AC – alternate current field

$$\vec{E} = E_o\cos{(\omega t)}\hat{e}_E \tag{18}$$

RF – rotate field

$$\vec{E} = E_o(\cos{(\omega t)}, \sin{(\omega t)}, 0)\hat{e}_E \tag{19}$$

where $\hat{e}_E = \hat{e}_x$ or $\hat{e}_E = \hat{e}_y$ or $\hat{e}_E = \hat{e}_z$, E_o is the magnitude of the external field and ω its frequency. Except for the rotate field, all other fields act along the space fixed direction.

6 Overlaps

The presence of the Gaussian random displacement vector in the BD integration scheme (Equation 2) may lead to nonphysical overlaps between molecules. Overlaps occur when the distance between two atoms that belong to different molecules is smaller than the sum of excluded volume radii of these atoms. *bd_rigid* can check whether a particular simulation step leads to an overlap. Next, positions of molecules are corrected to remove all cases of overlaps. This can be done in one of the four ways.

One option is that the current step of the simulation is rejected, original positions of all molecules are restored and a new step is attempted with a different random vector, until there are no cases of overlaps.

The second option is similar to the first, however, each of the subsequent attempts is made with a smaller time step. After successful removal of all overlaps, the time step is restored to its initial value.

As a third option, we also implemented an iterative procedure that removes overlaps between atoms via translations and rotations around diffusion centers of colliding molecules ($\Delta\vec{x}^{corr}$) resulting from pseudo-forces acting along the line connecting centers of overlapping atoms ($\vec{M}^{corr}$):

$$\Delta\vec{x}^{corr}(\Delta t) = \xi \boldsymbol{D} M^{\vec{corr}} \tag{20}$$

where:

ξ - the scaling factor

$\boldsymbol{D}$ - the diffusion tensor.

Each pair of overlapping atoms is treated separately. Translations and rotations of molecules are performed until there are no overlaps in the system.

The fourth option is similar in the spirit to the third one but overlaps between two different molecules are evaluated simultaneously and all pairs of overlapping atoms from a given molecule pair are considered in order to determine the directions of correcting forces.

7 Running Simulations with bd_rigid

Various input files are needed to perform BD simulations with the *bd_rigid* module of the BD_BOX package (Figure 3).

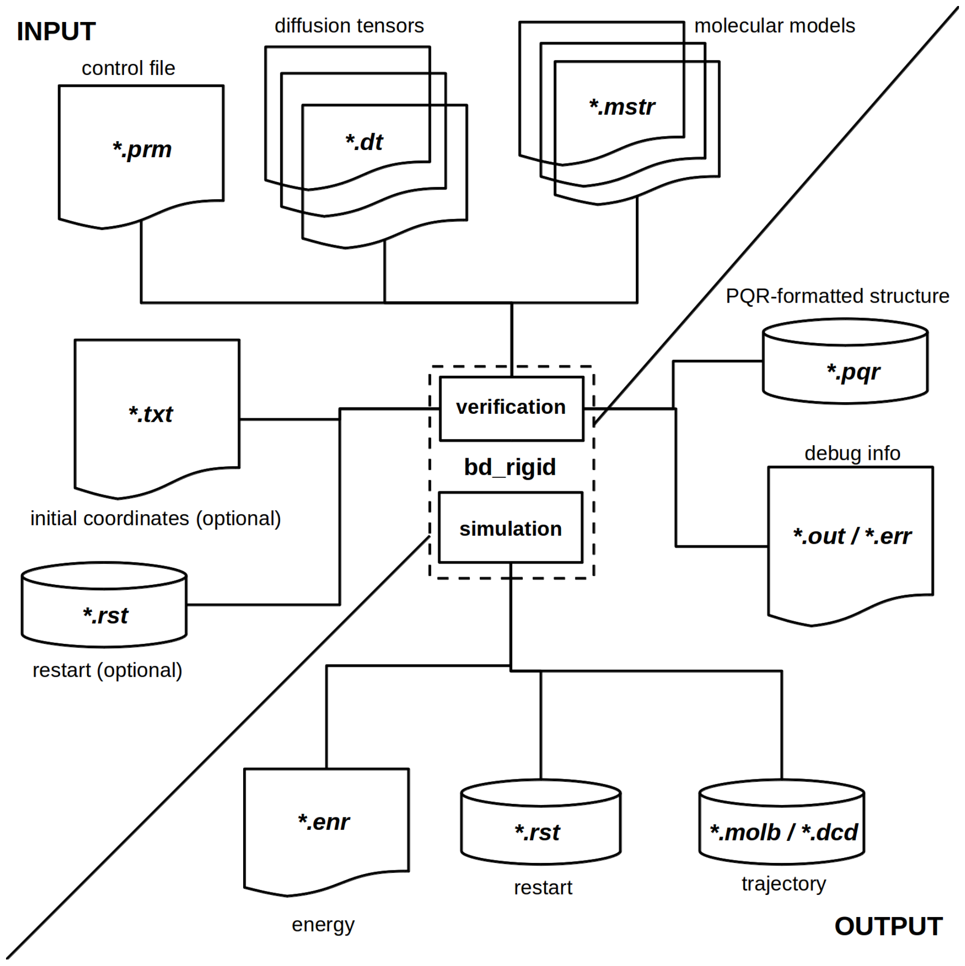


Figure 3: Schematics of the BD_BOX bd_rigid module

Structure files (**.mstr*) contain definitions of molecular models that are used to represent different types of molecules to be simulated. A separate **.mstr* file is required for each kind of a molecule that is present in the simulated system (for example, when monodisperse system is to be simulated, a single structure file will suffice); the number of molecules of each kind in the system is specified by the user in the **.prm* file.

Diffusion tensor files (**.dt*) contain definition of diffusion tensors of simulated molecules. A separate **.dt* file is required for each kind of a molecule that is present in the simulated system.

Simulation parameters are stored in the text control file (**.prm*) (Figure 3). These include for example the type of the propagation algorithm and its time step, temperature, ionic strength, definition of boundaries, values of cutoffs for intermolecular interactions, specifications of the output (names and types of files, write frequency), path to the input structure and diffusion tensor files, numbers of molecules of different types in the simulated system, and many others.

While molecules are usually placed by the program at random positions and with random orientations inside the primary simulation cell, it is also possible to define initial positions and orientations of all objects in the simulated system using a text-formatted file (Figure 3).

The output from *bd_rigid* consists of a few files (Figure 3).

The output file (**.out/*.err*) contains the track of the BD run, error messages (if any) and debugging information.

Another file, in the PQR format (**.pqr*), contains the definition of the molecular system under study - it can be used to verify the starting configuration of molecules and also (together with a trajectory file) to visualize results.

The trajectory file, either in the binary DCD or binary MOLB format (**.dcd/*.molb*) describes the time-dependent behavior of the studied system. MOLB trajectory contains only translations and rotations that are needed to perform transformations from the molecule-fixed to the laboratory frame, while DCD files contain Cartesian coordinates of atoms of all molecules in the simulated system in the laboratory frame.

The energy file (**.enr*) registers different contributions to the total energy of the system during a simulation. The binary restart file (**.rst*) contains all the information needed to continue an interrupted BD run. All files are periodically updated during a BD run with frequencies defined by the user.

Having prepared all the necessary files, the user may run the simulation with the command:

```
bd_rigid file.prm
```

It is also possible not to use the control file at all or to override all options specified in the control file by using command line parameters that correspond to appropriate control file keywords:

```
bd_rigid --parameter=value
```

or

```
bd_rigid file.prm --parameter=value
```

7.1 Structure File

Structure file contains representations of a molecule that are used in evaluation of intermolecular interactions. In general, the structure file adheres to the following format:

TAG list of properties

Currently, two tags are recognized: Q and LJ. Lines that begin with the tag Q define dielectric/electrostatic model of a molecule. Molecule is composed of dielectric spheres with central discrete charges. Lines that begin with the tag Q should be formatted as follows:

$Q\ x_1\ y_1\ z_1\ charge_1\ radius_1$

....

$Q\ x_i\ y_i\ z_i\ charge_i\ radius_i$

....

$Q\ x_N\ y_N\ z_N\ charge_N\ radius_N$

where *x*, *y*, *z* denote Cartesian coordinates of a dielectric sphere in the molecular frame, *charge* denotes its central charge and *radius* denotes its radius. Molecular representation specified with Q tags is used in evaluation of electrostatic interactions (Coulombic and desolvation terms, equations 14 and 15).

Molecular representation specified with LJ tags is used in evaluation of collisions, excluded volume and hydrophobic interactions. Lines that begin with the LJ tag should be formatted as follows:

$LJ\ x_1\ y_1\ z_1\ radius_1\ \epsilon_1\ d1_1\ d2_1\ c0_1\ c1_1\ c2_1\ c3_1\ \mu_1\ SASA^o_1$

....

$LJ\ x_j\ y_j\ z_j\ radius_j\ \epsilon_j\ d1_j\ d2_j\ c0_j\ c1_j\ c2_j\ c3_j\ \mu_j\ SASA^o_j$

....

$LJ\ x_M\ y_M\ z_M\ radius_M\ \epsilon_M\ d1_M\ d2_M\ c0_M\ c1_M\ c2_M\ c3_M\ \mu_M\ SASA^o_M$

where *x*, *y*, *z* denote Cartesian coordinates of spheres in the molecular frame, *radius* denotes radius used in evaluations of excluded volume and hydrophobic interactions, ϵ is the well depth of excluded volume interactions, parameters d and c are for the FACTS model (equations 6, 7), $SASA^o$ denotes the solvent accessible surface area of a given sphere in the model (Equation 9) and μ (Equations 5, 13) denotes the surface tension parameter.

Molecular representations given with tags Q and LJ can differ both in the number of spheres (atoms) and their positions.

We supply a tool (*orient_mstr*, see below) that can be used to transform (rotate and translate) molecular coordinates (given either as a **.mstr*, **.pdb* or **.pqr* files to the molecular frame that coincides with the main axes of diffusional rotations with the origin located at the diffusion center of a molecule.

7.2 Tensor File

Diffusion tensor files (**.dt*) (a separate file is needed for each type of a molecule that is present in the simulated system) contain elements of the diffusion tensor matrix in the plain text format. Elements of $\boldsymbol{D}$ are given in **.dt* files in the following order:

$$\begin{array}{lll@{\qquad}lll}
D^{tt}_{11} & D^{tt}_{12} & D^{tt}_{13} & D^{tr}_{11} & D^{tr}_{12} & D^{tr}_{13} \\
D^{tt}_{21} & D^{tt}_{22} & D^{tt}_{23} & D^{tr}_{21} & D^{tr}_{22} & D^{tr}_{23} \\
D^{tt}_{31} & D^{tt}_{32} & D^{tt}_{33} & D^{tr}_{31} & D^{tr}_{32} & D^{tr}_{33} \\
\\
D^{rt}_{11} & D^{rt}_{12} & D^{rt}_{13} & D^{rr}_{11} & D^{rr}_{12} & D^{rr}_{13} \\
D^{rt}_{21} & D^{rt}_{22} & D^{rt}_{23} & D^{rr}_{21} & D^{rr}_{22} & D^{rr}_{23} \\
D^{rt}_{31} & D^{rt}_{32} & D^{rt}_{33} & D^{rr}_{31} & D^{rr}_{32} & D^{rr}_{33}
\end{array}$$

where units of D^{tt}_{ij} are Å$^2 \cdot ps^{-1}$, units of $D^{tr/rt}_{ij}$ are Å$\cdot ps^{-1}$ and units of D^{rr}_{ij} are ps^{-1}. Elements of $\boldsymbol{D}$ should be given in the molecule-fixed frame with origin located at the diffusion center of a molecule and axes coinciding with principal axes of rotations. We provide a tool (*dmatrix*, see below) that can be used to compute the diffusion tensor matrix of a molecule either based on the appropriate *PDB* file or a plain text file containing its coarse-grained representation.

7.3 External Coordinates File

Initial coordinates of objects in the studied system (in the laboratory frame) can be specified by the user in an external text file, which should adhere to the following format:

$v_x^{(1)}\ v_y^{(1)}\ v_z^{(1)}\ \Omega_x^{(1)}\ \Omega_y^{(1)}\ \Omega_z^{(1)}$

....

$v_x^{(i)}\ v_y^{(i)}\ v_z^{(i)}\ \Omega_x^{(i)}\ \Omega_y^{(i)}\ \Omega_z^{(i)}$

....

$v_x^{(N)}\ v_y^{(N)}\ v_z^{(N)}\ \Omega_x^{(N)}\ \Omega_y^{(N)}\ \Omega_z^{(N)}$

where i runs over N molecules (ordering of molecules results from parameters defined in the **.prm* file - see below); the vector:

$$\vec{v} = \{v_x, v_y, v_z\}$$

describes translations that are needed to translate the diffusion center of a molecule to its initial position in the laboratory frame, and the vector:

$$\vec{\Omega} = \{\Omega_x, \Omega_y, \Omega_z\}$$

describes rotations that are needed to orient the molecule in the laboratory frame.

$$\Omega = \sqrt{\Omega_x^2 + \Omega_y^2 + \Omega_z^2}$$

The rotation matrix is defined as [23]:

$$M = \begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{pmatrix}$$

where:

$$M_{11} = \frac{(\Omega_y^2 + \Omega_z^2)cos\Omega + \Omega_x^2}{\Omega^2}$$

$$M_{12} = \frac{\Omega_x\Omega_y}{\Omega^2}(1 - cos\Omega) - \frac{\Omega_z}{\Omega}sin\Omega$$

$$M_{13} = \frac{\Omega_x\Omega_z}{\Omega^2}(1 - cos\Omega) + \frac{\Omega_y}{\Omega}sin\Omega$$

$$M_{21} = \frac{\Omega_x\Omega_y}{\Omega^2}(1 - cos\Omega) + \frac{\Omega_z}{\Omega}sin\Omega$$

$$M_{22} = \frac{(\Omega_x^2 + \Omega_z^2)cos\Omega + \Omega_y^2}{\Omega^2}$$

$$M_{23} = \frac{\Omega_y\Omega_z}{\Omega^2}(1 - cos\Omega) - \frac{\Omega_x}{\Omega}sin\Omega$$

$$M_{31} = \frac{\Omega_x\Omega_z}{\Omega^2}(1 - cos\Omega) - \frac{\Omega_y}{\Omega}sin\Omega$$

$$M_{32} = \frac{\Omega_y\Omega_z}{\Omega^2}(1 - cos\Omega) + \frac{\Omega_x}{\Omega}sin\Omega$$

$$M_{33} = \frac{(\Omega_x^2 + \Omega_y^2)cos\Omega + \Omega_z^2}{\Omega^2}$$

7.4 Simulation Control File

Control file is based on a number of keywords that allow the user to specify simulations conditions. Only these lines in the control file that begin with a particular keyword are recognized by the program.

Each line of the **.prm* file should contain only one keyword and its value:

keyword value

The complete list of available keywords is given in the next section.

8 Control File Keywords and Command Line Parameters

Below, a complete list of recognized keywords and their possible values, is given. Values of all keywords relevant to a simulation must be specified by the user - default values that are given below are these at which program variables are initialized and they are not always meaningful.

8.1 Input/Output Control

out_filename *string* - text file containing diagnostic informations regarding the current *bd_rigid* run (output)

dcd_filename *string* - *DCD*-formatted binary trajectory file, default: *NULL string* (output)

save_dcd_freq *integer* - frequency (number of steps) for writing to the *DCD* trajectory, default: *1*

pqr_filename *string* - *PQR*-formatted file, containing the startup configuration of the simulated system, default: *NULL string* (output)

rep *string* - this keyword is used to define which molecular representation from the **.mstr* file (tags *Q* and *LJ*) should be written to the *PQR*-formatted file, possible options are *lj* (Lennard-Jones spheres defined with the *LJ* tags) or *elec* (charges, defined with the *Q* tags); default *lj*

molb_filename *string* - binary trajectory file (**.molb*), default: *NULL string* (output)

save_molb_freq *integer* - frequency (number of steps) for writing to the **.molb* trajectory, default: *1*

enr_filename *string* - plain text energy file, default: *NULL string* (output)

save_enr_freq *integer* - frequency (number of steps) for writing to the energy file, default: *1*

rst_filename *string* - binary restart file, default: *NULL string* (output)

save_rst_freq *integer* - frequency (number of steps) for writing to the binary restart file, default: *1*

ext_coor_file *string* - an optional plain text file describing an user-defined configuration of the simulated system, default: *NULL string* (input) *restart* *string* - this keyword indicates that a continuation run should be performed, using the *string* restart file, default: *NULL string* (input); restarts should be specified using command line parameters

object *string* *integer* - the name of the object (molecule) to be simulated and the number of objects of this kind in the system - the program will look for appropriate *string.mstr* and *string.dt* files; multiple *object* lines can be present in the parameter file - their order determines the ordering of molecules in all output files as well as the order of objects in the external coordinate file defined with the *ext_coor_file* key

benchmark *string* - whether to benchmark evaluation of forces (yes no), default: *no*,

8.2 Nonbonded Interactions

elec string - whether to evaluate electrostatic (screened Coulomb) interactions, *yes/no*, default: *yes*

elec_desolv string - whether to evaluate desolvation (cavity) forces, *yes/no*, default: *yes*

hdb_surf string - whether to evaluate non-polar (hydrophobic) interactions, *none/primitive/facts*, default: *none*

elec_cutoff float - cutoff for electrostatic interactions (also cavity terms), default: *0*

hdb_cutoff float - cutoff for nonpolar interactions evaluated with the FACTS model (R_i in Equation 8),default: *10.0*

hdb_a float - inner cutoff for nonpolar interactions evaluated with the primitive model (a in Equation 10), default: *3.1*

hdb_b float - outer cutoff for nonpolar interactions evaluated with the primitive model (a in Equation 10), default: *4.35*

hdb_phi float - scaling factor for nopolar interactions evaluated with the primitive approach, default: *0.5* *lj_cutoff* float - cutoff for Lennard-Jones interactions, default: *10.0*; setting this parameter to *-1* results in cutoffs defined for each pair of atoms as $2^{\frac{1}{6}}(R_i + R_j)$ (useful in case of the *12/6* form of the L-J potential)

lj_repul float - repulsive term in the Lennard-Jones potential, (Equation 4, r^{-lj_repul}), possible values *8*, *12*, default: *12*

lj_attract float - attractive term in the Lennard-Jones potential, (Equation 4, $r^{-lj\_attract}$), possible values: *0*, *6*, default: *6*

repul_B float - the magnitude of the repulsive part in the Lennard-Jones potential (Equation 4), default: *1*

probe_radius float - radius of the spherical probe used to determine surface atoms - during simulations collisions are evaluated between surface atoms, default: *1.4*

srf_ratio float - value used to determine whether an atom is buried or belong to the surface of a molecule, i.e. if for a given atom the ratio $surface^{buried}/surface^{total}$ is smaller than *srf_ratio* the atom belongs to the surface of the molecule, default: *0.9*

points_per_sphere integer - number of points used to describe the surface of an atom, default: *1000*

gamma float - unit conversion factor for electrostatic interactions (Equation 14 and 15), default (conversion to $\frac{kcal}{mol}$): *331.842*

alpha float - scaling factor for Lennard-Jones interactions (Equation 4), default: *4.0*

beta float - scaling factor for desolvation (cavity) forces (Equation 14 and 15), default: *1.0*

8.3 Boundaries

bc *string* - specifies boundary conditions, possible values are *none* - an infinite system, *pbc* - periodic boundary conditions, *sphere* - an impenetrable sphere

xbox *float* - the primary simulation cell, size in the x direction, default: *0.0*

ybox *float* - the primary simulation cell, size in the y direction, default: *0.0*

zbox *float* - the primary simulation cell, size in the z direction, default: *0.0*

sphere_radius *float* - the radius of an impenetrable sphere enclosing the simulated system, default: *0.0*

sboundary *string* - whether to use the bounding sphere potential, default: *no* (another possibility *yes*). Note that this keyword has nothing to do with the *bc* keyword; if *bc sphere* is used molecules are not allow to cross a spherical surface around the studied system but no potential is used for that, rather BD moves leading outside the spherical surface are simply rejected an repeated with different random vectors

sboundary_A *float* - the magnitude of the bounding sphere force (Equation 16), default: *0.0*

sboundary_n *integer* - the power of the radial distance dependence of the bounding sphere force (Equation 16), default: *0*

sboundary_cutoff *float* - the bounding sphere force is applied outside this cutoff radius (Equation 16), default: *0.0*

8.4 External Electric Fields

E_ext *string* - whether to switch on/off the external electric field, *yes/no*, default: *no*

E_magn *float* - the magnitude of the external electric field, default: *0.0*

E_type *string* - a choice between different types of the electric field, possible values are *AC*, *DC* or *RF*, default: *DC*

E_freq *float* - the frequency (where applicable) of the external electric field (units $\left[\frac{1}{ps}\right]$), default: *0.0*

E_dir1 *string* - the direction of the external electric field, *x*, *y* or *z*, default: *x*

E_dir2 *string* - the second direction of the external electric field (RF), *x*, *y* or *z*, default: *y*

E_factor *float* - the unit conversion factor for the external electric field (to $\frac{kcal}{mol} \cdot e^{-1} \mathring{A}^{-1}$), default: *1.0*

8.5 Algorithms

bd_algorithm *string* - this keyword defines the algorithm that is used to generate the BD trajectory of the studied system, possible values are *ermak_const* (the time step is constant during the simulation) or *ermak_var* (variable time step is used in order to prevent overlaps between simulated objects)

bd_steps *integer* - the number of BD simulation steps

dt *float* - time step

min_dt *float* - minimal possible value of the time step when the variable time step algorithm is used

rand_seed *integer* - seed for the random number generator

move_attempts *integer* - the maximal number of attempts that are made by the program to remove overlaps in the system, in each attempt a BD step that leads to an overlap is repeated with a different random vector (Equation 3) *pos_iter* *integer* - maximal number of iterations that are performed by the program to position and orient objects in the simulation cell at the begining of the BD run

pos_min *integer* - the minimal allowed surface-to-surface distance between objects in the simulation cell at the begining of the BD run

check_overlaps *string* - whether to check if there are overlapping objects in the simulated system whose positions need to be corrected *yes*/*no*

overlaps *string* - defines the algorithm that is applied to look for the overlaps in the system, either *trees* (for a hierarchical algorithm) or *direct* (for a brute force algorithm)

overlaps_removal *string* - defines the method that is applied to remove overlaps in the system: *none* (if the *check_overlaps* keyword is set to *yes* the program will repeat all steps that lead to overlaps with different random vectors), *atom* - overlaps will be corrected iteratively by applying appropriate forces and torques - each pair of overlapping atoms is evaluated separately, *molecule* - overlaps will be corrected iteratively by applying forces and torques and evaluating all pairs of overlapping atoms within a given molecules pair simultaneously

nb_list *string* - defines a method that is used to evaluate nonbonded interactions; *brute* - brute force method, *spatial* - a method similar in the spirit to the cell-linked lists algorithm, however, without explicit creation of nonbonded lists, *verlet* - Verlet lists algorithm with a twin-range cutoff

verlet_roff float - defines the value of the outer cutoff in the twin-range cutoff Verlet algorithm, i.e. the outer cutoff is computed as a cutoff increased by *verlet_roff*

verlet_count integer - Verlet lists will be refreshed during a simulation every *verlet_count* steps

dm_alg float - defines the method used to perform thread decomposition (in case of parallel runs), either *atomdec* (decomposition is based on the indexes of atoms) or *spatialdec* (decomposition is performed based on Cartesian coordinates of atoms)

8.6 Physical Conditions

T *float* - temperature, default: *298.15*

kappa *float* - the inverse of the Debye length, default: *0.1*

eps *float* - solvent dielectric constant, default: *78.54*

epsin *float* - solute dielectric constant, default: *1*

8.7 Devices Control

cuda_devices *integer integer* - CUDA devices to be used, their numbers followed by system identifiers

*cuda_block**integer* - dimension of the thread block - relevant in case of CUDA devices with the *compute* capability smaller than 2.0

9 Tools

We provide a set of tools that can be used to prepare input files necessary to perform BD simulations with the BD_BOX bd_rigid module. These are described below.

9.1 dmatrix

This utility can be used to compute the rigid-body diffusion tensor of a molecule. In case of an atomistic model of a molecule coordinates are read from a *PDB*-formatted file. However, if the user needs to compute the diffusion tensor of a coarse-grained molecular model, its coordinates should be specified directly in the text input file. *dmatrix* requires an input file *input.txt* (the name of the file should not be changed) that adheres to the following format:

input.txt - computations are to be performed for a molecule whose coordinates are stored in a *PDB*-formatted file:

---

comment - single line containing comments (can be left empty)

1 - coordinates of a molecule are stored in the PDB file

name of the appropriate PDB file - name of the PDB file with coordinates of a molecule

viscosity temperature - viscosity of the solvent (Poisse) and temperature (K)

solvent radius - radius of solvent molecules (typically set to 1.4Å - water)

---

input.txt - computations are to be performed for a coarse-grained model:

---

comment - single line containing comments (can be left empty)

0 - computations are to be performed for a coarse-grained model

number of beads in a model - number of spherical subunits (atoms) in a model

x y z radius - coordinates of a spherical subunit and its radius

x y z radius

x y z radius

...

viscosity temperature - viscosity of the solvent (Poisse) and temperature (K)

solvent radius - radius of solvent molecules (typically set to 0.0Å in case of coarse-grained models)

To run computations, the user should simply execute the *dmatrix* command from within the directory where the *input.txt* file is located. As a result, three text files should be created:

output.txt - details of computations and debugging information

tensor.txt - 6×6 diffusion tensor matrix

transformations.txt - transformations (translations and rotations) that should be performed to orient the molecule along main axes of the rotational diffusion tensor, with the origin located at the diffusion centre. This file can be fed to the *orient_mstr* utility.

NOTE: when computations are performed for a coarse-grained model, overlaps between beads defined in the input.txt file are possible only if all beads in a model are of the same size

9.2 orient_mstr

Tool for orienting molecule in the molecular frame (with axes coinciding with the principal axes of diffusional rotations).

Usage:

```
orient_mstr transformations structure
```

where:

transformations - file resulting from the execution of the *dmatrix* tool

structure - file containing coordinates of a molecule to be transformed (*PDB/PQR/MSTR* files are supported)

Results of the *orient_mstr* execution - transformed coordinates of a molecule formatted according to the format of the input structure file - are printed to the standard output.

9.3 parsemolb

Tool for converting binary **.molb* files to text

Usage:

parsemolb file index precision

where:

file - name of the **.molb* file to be converted

index - index of a molecule whose coordinates are to be extracted (numbering of molecules starts with 0)

precision - either *float* or *double*

Output is printed to the standard output using the following format:

time v_x v_y v_z *o(1) .... o(9)*

where *time* denotes current time, v_x, v_y, v_z are components of the translation vector (from the molecule-fixed to the laboratory frame) and *o(1) ... o(9)* are components of the rotation matrix. Current position of a given molecule in the laboratory frame can be obtained by applying rotations and then translations to the input structure of a molecule.

9.4 molb2dcd

Tool for converting binary **.molb* trajectory files to binary **.dcd* files.

Usage:

molb2dcd input type precision name

where: *input* - text input file

type - either *all* or *separate* - coordinates of molecules are either stored in a single file or separate **.dcd* files are created for each molecule

precision - either *double* or *float*

name - input **.molb* file

The text input file is used to specify molecules for which DCD trajectory will be created, it contains names of PDB files and indexes of molecules of interest in the source **.molb* file. Indexes can be given as a list or as a range. An exemplary text input file is given below.

exemplary text input file for the *molb2dcd* utility

```
name0.pdb 0
name1.pdb 1-10
name2.pdb 11 12 13 14 15
name3.pdb 16
```

9.5 sasa_tool

This small utility program can be used to compute atomic SASA's using atomic coordinates and radii of atoms provided by the user in the PQR-formatted input file.

Usage:

```
sasa_tool input output N radius
```

where:

input - input *.pqr file

output - output file, plain text

N - number of points used to describe the surface of an atom

radius - radius of the solvent probe

9.6 charge_optimizer

Utility program that can be used to derive a set of effective charges. Effective charges positioned initially at heavy atoms of the molecule are derived by fitting the electrostatic potential resulting from the Debye-Hückel approximation to the external molecular potential obtained as a numerical solution of the Poisson-Boltzmann equation (see [31] for details). The total number of effective charges is optimized upon the execution of *charge_optimizer*. Fitting of effective charges can be performed in various regions around the molecule: the whole space around the molecule, spherical layer centered on the molecule, or the molecular skin defined by inflating van der Waals radii of molecule's atoms (see Figure 4). Execution of the program is controlled via keywords given in the plain text parameter file.

Usage:

charge_optimizer file.prm

and the following keywords can be specified in the parameter file (in the *keyword value* manner):

gamma float - unit conversion factor, the default value of 331.842 results in electrostatic potentials evaluated in units of $\frac{kcal}{mol \cdot e}$

eps float - dielectric constant of the solvent

kappa float - the inverse of the Debye length, default: 0.1 Å^{-1}

dxfile string - DX-formatted file with the electrostatic potential of the molecule obtained by solving the linear Poisson-Boltzmann equation, input

outsourcedx string - DX-formatted file with the input electrostic potential (scaled by *ext_phi_scale*, see below) that can be used for debuging and visualization purposes, output

ext_phi_scale float - factor for scaling (multiplication) values of the input electrostatic potential (can be used to convert units), default value: 0.5922 (scaling from kT_{298K} to $\frac{kcal}{mol}$)

skip_points integer - every *skip_points* point of the input grid (in each direction, x, y, z) will be taken into account upon evaluation of electrostatic potential, default: 1 - all points are taken into account

pqrfile string - input, PQR-formatted file containing the definition of the evaluated molecule

outdx string - DX-formatted file with the electrostatic potential generated by the derived effective charges, output *outpqr* - PQR-formatted output file with effective charges

tol float - goodness of electrostatic charges fit, this keyword is used to define the stop condition for the program - value of 1.0 means perfect fit (sensible values of effective charges are typically obtained for *tol* in the range of 0.85 to 1.0), default value: 0.9995

penalty float - this variable is used to prevent the effective charges to have unphysical (extremely large) magnitudes, typically values below 0.01 should used, default:0.001

step_attempts integer - number of attempts made by the prgram to reduce the number of effective charges, default: 1000; initially effective charges are positioned at heavy atoms - if possible this number is subsequently reduced while preserving the accuracy specified with the *tol* parameter

seed integer - seed for the random number generator, 31415

omega string - defines the region where the fitting is performed, possible values are *full, sphere, skin*

omega_low float - lower boundary of the fitting region, relevant for all values of the *omega* keyword

omega_high float - upper boundary of the fitting region, relevant only for *omega* values *skin* and *sphere*

out_filename string - plain text log file, default: err.log

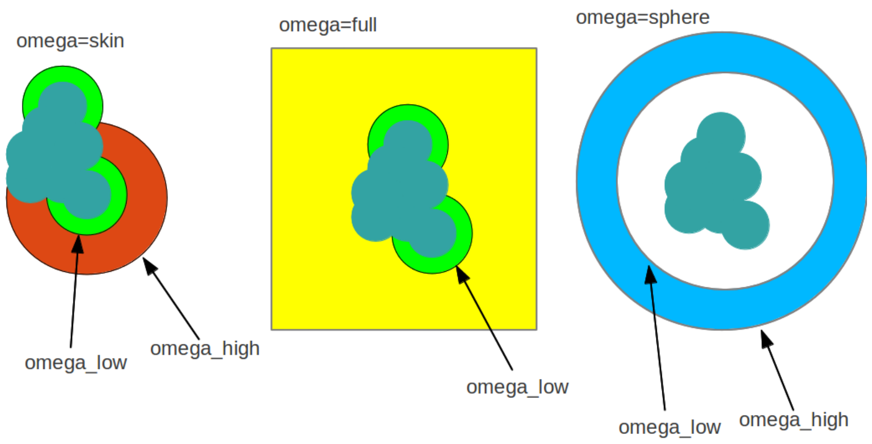


Figure 4: Possible definitions of the region outside the molecule where its electrostatic potential is fitted upon evaluation of effective charges. When the *omega* parameter is assigned value *skin*, electrostatic potential is fitted in the volume bounded by surfaces obtained by inflating van der Waals surface of the molecule by *omega_low* and *omega_high* (left). When *omega* is set to *full* the upper bondary of the volume results from the definition of the input electrostatic grid (center). Setting *omega* to *sphere* results in a volume of spherical layer between surfaces of spheres of radii *omega_low* and *omega_high*, respectively (right).

10 Examples distributed with BD_BOX

Below, we present a list of examples that are distributed along with the BD_BOX source code. Examples are located in *examples/rigid/name* directories.

HEWL

A single hen egg white lysozyme molecule. All-atom model.

SPHERES

Periodic box with hard spheres at volume fraction of 30%.

QSPHERES

Periodic box with oppositely charged spheres.

HEWL_pH3_100mM

420 molecules of hen egg white lysozyme at volume fraction of 10%. Charges assigned at pH 3 and 100mM ionic strength.

11 Final Notes

We did our best to ensure that the BD_BOX code is bug-free. We have tested implemented features using rather simple and thus predictable models (such as single spheres, dumbbells, chains) but also more elaborate models of proteins. Properties of these models derived from BD simulations (such as for example translational and rotational diffusion coefficients at different conditions of temperature and viscosity, chains end-to-end distances and their radii of gyration) were validated using theoretical/analytical predictions and available literature data. Additionally, we have also cross-examined BD_BOX using various hardware platforms.

We invite users to send comments and questions regarding their own applications of BD_BOX. Reports on possible bugs are also welcomed.

References

[1] K Kremer and G S Grest. Dynamics of entangled linear polymer melts: A molecular-dynamics simulation. *J Chem Phys*, 92:5057–5087, 1990.

[2] J G de la Torre, J G H Cifre, A Ortega, R R Schmidt, M X Fernandes, H E P Sánchez, and R Pamies. Simuflex: algorithms and tools for simulation of the conformation and dynamics of flexible molecules and nanoparticles in dilute solution. *J Chem Theory Comput*, 5:2606–2618, 2009.

[3] G M Bell, S Levine, and McCartney L N. Approximate methods of determining the double-layer free energy of interaction between two charged colloidal spheres. *J of Colloid Int Sci*, 33:335–359, 1970.

[4] M Aubouy, E Trizac, and L Bocquet. Effective charge versus bare charge: an analytical estimate for colloids in the infinite dilution limit. *J Phys A: Math Gen*, 36:58355840, 2003.

[5] J Rotne and S Prager. Variational treatment of hydrodynamic interaction in polymers. *J Chem Phys*, 50:4831–4838, 1969.

[6] H Yamakawa. Transport properties of polymer chains in dilute solution: hydrodynamic interaction. *J Chem Phys*, 53:436–444, 1970.

[7] J G de la Torre and V A Bloomfield. Hydrodynamic properties of macromolecular complexes. i. translation. *Biopolymers*, 16:1747 – 1763, 1977.

[8] E R Smith, I K Snook, and W van Megen. Hydrodynamic interactions in brownian dynamics simulations. i. periodic boundary conditions for computer simulations. *Physica A*, 143:441–467, 1987.

[9] C W J Beenakker. Ewald sum of the rotne-prager tensor. *J Chem Phys*, 85:1581–1582, 1986.

[10] D L Ermak and J A McCammon. Brownian dynamics with hydrodynamic interactions. *J Chem Phys*, 69:1352–1360, 1978.

[11] A Iniesta and J G de la Torre. A second order algorithm for the simulation of the brownian dynamics of macromolecular models. *J. Chem. Phys.*, 92:2015–2019, 1990.

[12] U Winter and T Geyer. Coarse grained simulations of a small peptide: Effects of finite damping and hydrodynamic interactions. *J Chem Phys*, 131:104102–104107, 2009.

[13] T Geyer and U Winter. An o(n2) approximation for hydrodynamic interactions in brownian dynamics simulations. *J Chem Phys*, 130:114905–114913, 2009.

[14] T Ando and J Skolnick. Krylov subspace mthods for computing hydrodynamic interactions in Brownian dynamics simulations. *J. Chem. Phys.*, 137:064106, 2012.

[15] M X Fernandes and J G de la Torre. Brownian dynamics simulations of rigid particles of arbitrary shape in external fields. *Biophys J*, 83:3039–3048, 2002.

[16] P Strating. Brownian dynamics simulations of hard-sphere suspension. *Phys Rev E*, 59:2157–2187, 1999.

[17] J Sun and H Weinstein. Toward realistic modeling of dynamic processes in cell signaling: Quantification of macromolecular crowding effects. *J Chem Phys*, 127:155105–155115, 2007.

[18] S R McGuffee and A H Elcock. Atomically detailed simulations of concentrated protein solutions: the effects of salt, ph, point mutations and protein concentration in simulations of 1000-molecule systems. *J Am Chem Soc*, 128:12098, 2006.

[19] M Matsumoto and T Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8:330, 1998.

[20] G E P Box and M E Muller. A note on the generation of random normal deviates. *Annals Math Stat*, 29:610–611, 1958.

[21] W Press, B Flannery, S Teukolsky, and W Vetterling. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, 1992.

[22] B Carrasco and J G de la Torre. Hydrodynamic properties of rigid particles: comparison of different modeling and computational procedures. *Biophys J*, 76:3044–3057, 1999.

[23] D A Beard and Schlick T. Unbiased rotational moves for rigid-body dynamics. *Biophys. J.*, 85:2973–2976, 2003.

[24] U Haberthür and A C Caflisch. Facts: Fast analytical continuum treatment of solvation. *J. Comput. Chem.*, 29:701–715, 2008.

[25] R R Gabdoulline and R C Wade. On the contributions of diffusion and thermal activation to electron transfer between phormidium laminosum plastocyanin and cytochrome f: Brownian dynamics simulations with explicit modeling of nonpolar desolvation interactions and electron transfer events. *J Am Chem Soc*, 131:9230–9238, 2009.

[26] B Roux and T Simonson. Implicit solvent models. *Biophys. Chem.*, 78:1–20, 1999.

[27] A D MacKerell Jr., D Bashford, M Bellott, R L Dunbrack Jr., J D Evanseck, M J Field, S Fischer, J Gao, H Guo, S Ha, D Joseph-McCarthy, L Kuchnir, K Kuczera, F T K Lau, C Mattos, S Michnick, T Ngo, D T Nguyen, B Prodhom, W E Reiher III, B Roux, M Schlenkrich, J C Smith, R Stote, J Straub, M Watanabe, J Wiórkiewicz-Kuczera, D Yin, and M Karplus. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem. B*, 102:3586–3616, 1998.

[28] N A Baker, D Sept, S Joseph, M J Holst, and J A McCammon. Electrostatics of nanosystems: application to microtubules and the ribosome. *Proc Natl. Acad Sci USA*, 98:10037–10041, 2001.

[29] R R Gabdoulline and R C Wade. Effective charges for macromolecules in solvent. *J Phys Chem*, 100:3868–3878, 1996.

[30] X Li, Y Levin, and M E Fisher. Cavity forces and criticality in electrolytes. *Europhys. Lett.*, 26:683–688, 1994.

[31] M Długosz and Antosiewicz J. Anisotropic diffusion effects on the barnase-barstar encounter kinetics. *J. Chem. Theory Comput.*, 9:1667–1677, 2013.