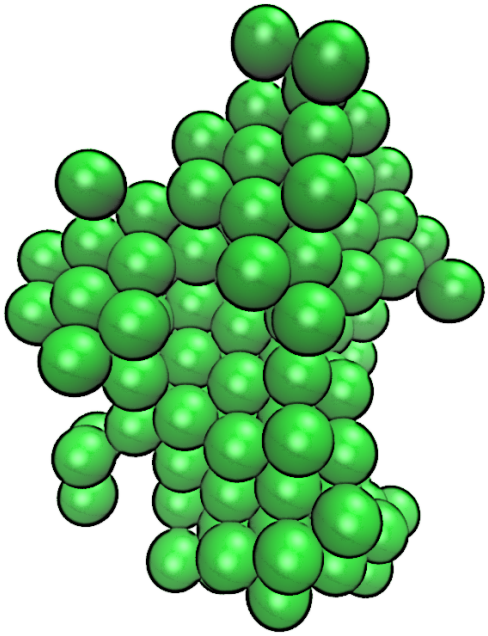
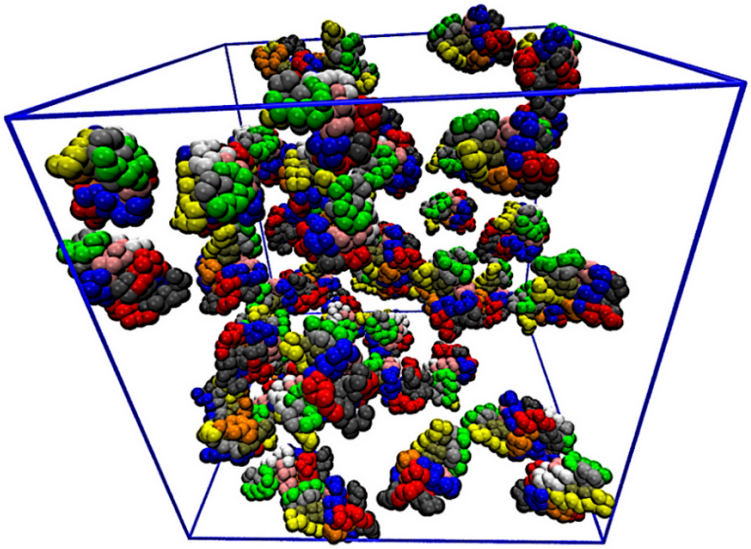
BD_BOX
version 2.2
user's manual – bd_flex

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, 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 simulated systems. 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 containing or periodic boundary conditions can be used. With BD_BOX one can effectively simulate either single molecules or multi-

molecular 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_flex* module of the BD_BOX package. The *bd_rigid* module for rigid-body BD simulations, 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 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 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 on particular systems. 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.

Units

charge: e

length: Å

time: ps

temperature: K

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

viscosity Poise

3 Bead Models

The *bd_flex* module of BD_BOX utilizes coarse-grained models of molecules (see for example Figure 1). Each model consists of a number of spherical subunits (beads). Beads can be placed for example on repeating units of biomolecules, such as amino or nucleic acids [22, 23]. Various levels of resolution are possible and different modeling approaches can be applied [24]. BD_BOX does not offer a separate tool dedicated to build and parametrize molecular models based on atomic structures of molecules. However, most of the coarse-grained models share the concept of a reduced representation with pseudo-atoms (beads) that represent groups of atoms and such models can be easily incorporated in BD_BOX (*bd_flex*) simulations [25, 2, 26]. Spherical subunits are assigned hydrodynamic radii so that the overall diffusive properties of a coarse-grained model correspond to the measured or computed properties of real molecules. Subunits' hydrodynamic radii values can be parametrized for example using the rigid-body hydrodynamic calculations [32, 29].

Additionally, each subunit is assigned a hard-core radius (Figure 1) used to evaluate Lennard-Jones interactions. The hydrodynamic and the hard-core radius of a given subunit can be different.

4 Potentials and Fields

Below we present bonded and nonbonded interactions potentials, currently implemented in the *bd_flex* module of the BD_BOX package.

4.1 Bonded Potentials

4.1.1 Bonds

The following potential [26] is considered to model connections between spherical subunits (beads) *i* and *j*:

$$V_{ij} = -\frac{1}{2} H r_{max}^2 \ln\left(\frac{r_{max}^2 - r_{ij}^2}{r_{max}^2 - r_o^2}\right) - \frac{1}{2} H r_{max} r_o \ln\left(\frac{(r_{max} + r_{ij})(r_{max} - r_o)}{(r_{max} - r_{ij})(r_{max} + r_o)}\right) \quad (1)$$

where:

r_o – the equilibrium bond length

r_{max} – the maximum bond length

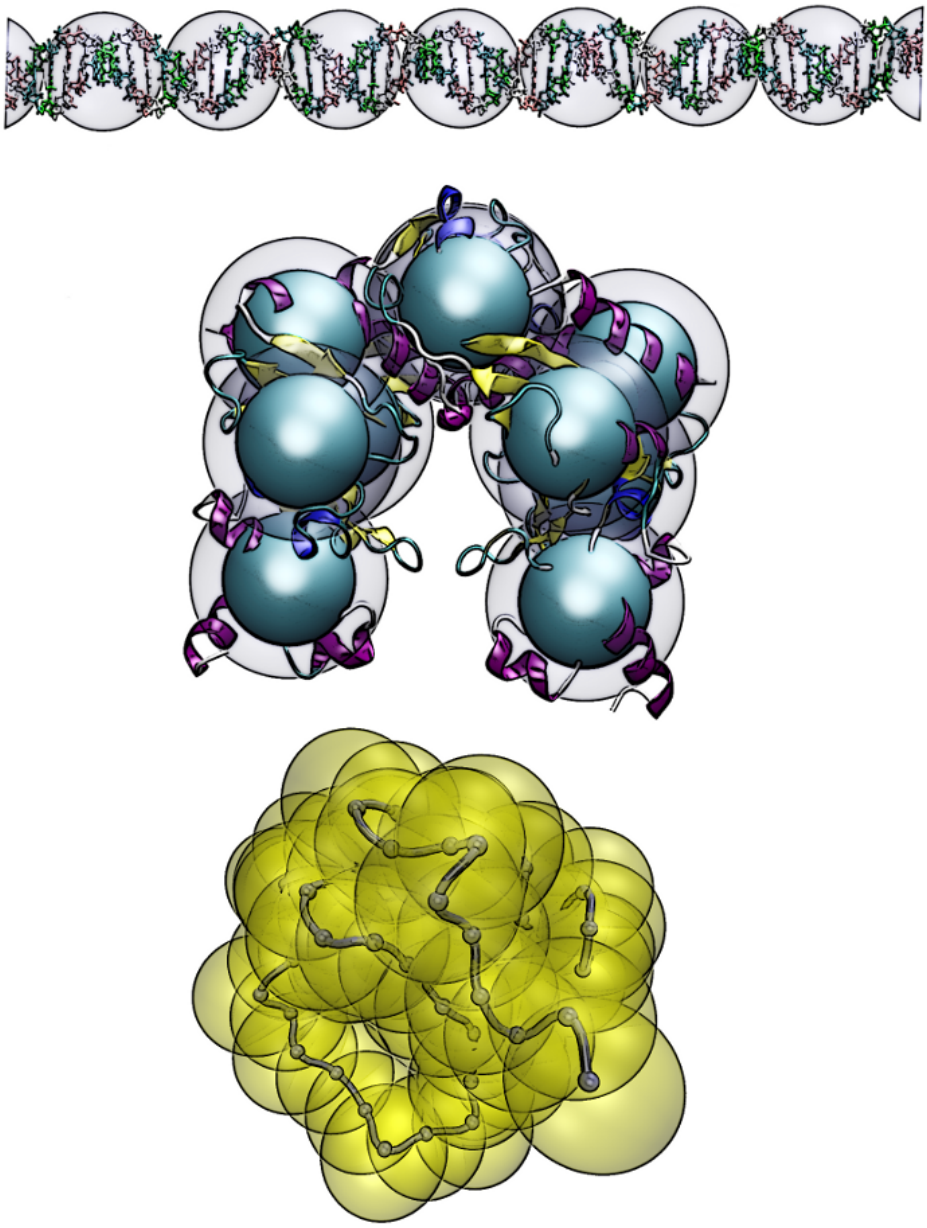


Figure 1: Exemplary bead models (hydrodynamic subunits are shown as transparent spheres). Top: DNA double helix taken from[2]. Middle: EcoRV endonuclease (PDB ID 1EOO[27]), radii of opaque beads define excluded volume interactions in multicomponent systems. The presented coarse-grained model of the EcoRV molecule was created using the CG Builder module from the VMD[28] package. Hydrodynamic radii of beads were computed using the HYDROPRO suite[29]. Bottom: Chymotrypsin inhibitor 2 (PDB ID 3CI2[30]). Hydrodynamic radii of beads were parametrized based on BD simulations with experimental data[31] and HYDROPRO[29] calculations as a reference.

H – the force constant

r_{ij} – the distance between beads centers

With an appropriate choice of parameters H and r_{max}[26, 2] this potential can describe either harmonic or FENE (finite extensible nonlinear elastic) [1] bonds. The number of bonds originating from a given subunit is unlimited.

4.1.2 Angles

Potential associated with the deformation of a planar angle ϕ_n can be either of form:

$$V_{\phi_n} = \frac{1}{2} k_{\phi_n} (\cos \phi_n - \cos \phi^o{}_n)^2 \tag{2}$$

or

$$V_{\phi_n} = \frac{1}{2} k_{\phi_n} (\phi_n - \phi^o{}_n)^2 \tag{3}$$

with k_ϕ being the force constant and ϕ^o the value of a given angle at the equilibrium.

4.1.3 Dihedrals

Rotation around a bond connecting beads (the deformation of a dihedral angle θ_n) is currently modeled either using a simple harmonic function:

$$V_{\theta_n} = \frac{1}{2} k_{\theta_n} (\theta_n - \theta^o{}_n)^2 \tag{4}$$

with k_θ being the force constant and θ^o the equilibrium angle, or more elaborate:

$$V_{\theta_n} = k_{\theta_n} \left(1 + \cos\left(\delta_n\right) \cos\left(m_n \theta_n\right)\right) \tag{5}$$

where δ =0 or $\delta = \pi$ and m= $1, 2..., 6$.

4.2 Nonbonded Potentials

Nonbonded interactions implemented in BD_BOX (*bd_flex*) include electrostatics and repulsive-attractive Lennard-Jones interactions.

4.2.1 Electrostatics

The Debye - Hückel approximation) [3] is used to model screened electrostatic interactions. Two spherical subunits with central charges Q_i and Q_j interact via a pairwise additive potential of a form:

$$V_{ij} = \gamma \frac{Q_i Q_j}{\epsilon_s r_{ij}} \exp\left(-\kappa r_{ij}\right) \tag{6}$$

where ϵ_s is the dielectric constant of the immersing medium, κ is the inverse of the Debye screening length, and r_{ij} is the separation of charges.

4.2.2 Lennard-Jones Potentials

Nonspecific interactions between subunits, repulsive at small separations and attractive at large separations are evaluated using standard 6/12 Lennard-Jones potentials:

$$V_{ij} = \alpha \epsilon_{LJ} \left(\left(\frac{\sigma_{LJ}^{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{LJ}^{ij}}{r_{ij}} \right)^{6} \right) \tag{7}$$

where $\epsilon_{ij}^{LJ} = \sqrt{\epsilon_i^{LJ} \epsilon_j^{LJ}}$ is the well depth and $\sigma_{LJ}^{ij} = R_i + R_j$ where R_i and R_j are hard-core radii of interacting subunits. It is also possible to model only purely repulsive interactions, without the long-range ($\sim \frac{1}{r^6}$) term.

4.3 Electric Fields

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

DC – direct current field

$$\vec{E} = E_o \hat{e}_E \tag{8}$$

AC – alternate current field

$$\vec{E} = E_o \cos(\omega t)\hat{e}_E \tag{9}$$

RF – rotate field

$$\vec{E} = E_o(\cos(\omega t), \sin(\omega t), 0)\hat{e}_E \tag{10}$$

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.

4.4 Bounding Sphere

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

$$\vec{F} = -\frac{A}{(R_{sphere} - r)^n}\hat{e}_r \tag{11}$$

where R_{sphere} is the radius of a spherical surface, n is a nonnegative integer, A is the amplitude of the force and r is the distance between a particular subunit and the system's centre of geometry. The above force is applied to each of the studied subunits outside a predefined cutoff.

5 Hydrodynamics and Brownian motion

Brownian dynamics trajectories are generated using either the Ermak-McCammon algorithm (E-M) [10], or the two step predictor-corrector algorithm developed by Iniesta and de la Torre (IG-T) [11].

5.1 Equations of Motions - E-M and I-GT Algorithms

An initial position of the i^{th} bead is described by $\vec{r}_i^{\,o}$. The position vector $\vec{r}_i$ after a time step Δt is computed using the following scheme:

predictor step:

$$\vec{r}_i' = \vec{r}_i^o + \frac{\Delta t}{k_B T} \sum_{j=1}^{N} \boldsymbol{D}_{ij}^o \vec{F}_j^o + \vec{R}_i \quad i,j = 1,...,N \tag{12}$$

with:

$$< \vec{R}_i >= 0 \quad i = 1,...,N \tag{13}$$

and

$$< \vec{R}_i (\vec{R}_j)^T >= 2\boldsymbol{D}_{ij}^o \Delta t \quad i,j = 1,...,N \tag{14}$$

The above equations describe the E–M algorithm [10], which is equivalent to the first-order Euler algorithm (except for the Brownian drift term). In the IG–T algorithm [11] an additional corrector step is taken:

corrector step:

$$\vec{r}_i = \vec{r}_i^o + \frac{1}{2}\frac{\Delta t}{k_B T} \sum_{j=1}^{N} \left(\boldsymbol{D}_{ij}^o \vec{F}_j^o + \boldsymbol{D}_{ij}' \vec{F}_j' \right) + \vec{R}_i' \quad i,j = 1,...,N \tag{15}$$

with:

$$< \vec{R}_i' >= 0 \quad i = 1,...,N \tag{16}$$

and

$$< \vec{R}_i' (\vec{R}_j')^T >= 2 \left(\frac{1}{2} \left(\boldsymbol{D}_{ij}^o + \boldsymbol{D}_{ij}' \right) \right) \Delta t \quad i,j = 1,...,N \tag{17}$$

where prime denotes that forces and diffusion tensors are evaluated for subunits in a configuration given with $\vec{r}_i'$.

in equations given above:

k_B - the Boltzmann constant

T - temperature

$\boldsymbol{D}_{ij}$ - 3×3 element of the 3N×3N configuration–dependent diffusion tensor

$\vec{R}_i$ - random vector

$\vec{F}$ - total force acting on the ith subunit

The vector $\vec{R}$ representing random movement of all beads can be obtained from:

$$\vec{R} = \boldsymbol{S}\vec{X} \tag{18}$$

where, the 3N×3N tensor $\boldsymbol{S}$ is derived from tensor $\boldsymbol{D}$ by the relation:

$$\boldsymbol{D} = \boldsymbol{S}\boldsymbol{S}^T \tag{19}$$

and $\vec{X}$ is a 3N×1 column vector, the elements of which are random Gaussian numbers with zero mean and 2Δt variance. *bd_flex* uses Cholesky decomposition of $\boldsymbol{D}$ to obtain matrices $\boldsymbol{S}$.

5.2 Diffusion Tensors

For a system consisting of N spherical subunits with hydrodynamic radii a_i, the following form of the 3N×3N diffusion tensor is used [5, 6, 7, 33]:

$$\boldsymbol{D_{ii}} = \frac{k_B T}{6\pi\eta a_i}\boldsymbol{I} \tag{20}$$

and:

$$\boldsymbol{D}_{ij} = \frac{k_B T}{8\pi\eta r_{ij}}\left[\left(1+\frac{a_i^2+a_j^2}{3r_{ij}^2}\right)\boldsymbol{I} + \left(1-\frac{a_i^2+a_j^2}{r_{ij}^2}\right)\frac{\vec{r}_{ij}\vec{r}_{ij}^T}{r_{ij}^2}\right] \qquad r_{ij} > (a_i + a_j)$$

$$\boldsymbol{D}_{ij} = \frac{k_B T}{6\pi\eta a_i a_j}\left[\frac{16r_{ij}^3(a_i+a_j) - ((a_i-a_j)^2+3r_{ij}^2)^2}{32r_{ij}^3}\boldsymbol{I} + \frac{3((a_i-a_j)^2-r_{ij}^2)^2}{32r_{ij}^3}\frac{\vec{r}_{ij}\vec{r}_{ij}^T}{r_{ij}^2}\right] \quad (a_> - a_<) < r_{ij} \le (a_i + a_j)$$

$$\boldsymbol{D}_{ij} = \frac{k_B T}{6\pi\eta a_>} \qquad r_{ij} \le (a_> - a_<)$$

where:

k_B - the Boltzmann constant

T - temperature

$\boldsymbol{I}$ - unit matrix 3×3

$\boldsymbol{D}_{ij}$ - 3×3 element of the 3N×3N configuration–dependent diffusion tensor

a_i - the hydrodynamic radius of a bead

$a_>$ - the hydrodynamic radius of the larger bead

$a_<$ - the hydrodynamic radius of the smaller bead

$\vec{r}_{ij}$, r_{ij} - the distance between beads

η - solvent viscosity

In case of finite-size simulation cells containing the studied system, when periodic boundary conditions are being used, the form of the diffusion tensor presented above is not directly applicable. We implemented in the *bd_flex* module of BD_BOX the direct Ewald summation procedure that leads to a periodic form of the diffusion tensor as proposed by Smith [8]. Let's consider a cubic box with side L, containing N spherical particles (some of them may be connected with bonds) with radii *a*. Particle i has position $\vec{r}_i$. With the center of the primary box at $(0,0,0)$, we construct an array of its copies having centers at $L\vec{m}$ with $\vec{m}$ being a vector of integer components:

$$\vec{m} = (m_1, m_2, m_3) \qquad m_i \in (-\infty, +\infty)$$
$$|\vec{m}| = \sqrt{m_1^2 + m_2^2 + m_3^2} \tag{21}$$

The magnitude of $\vec{m}$ is defined as:

$$m = |m_1| + |m_2| + |m_3| \tag{22}$$

In a given copy of the primary simulation box there are N particles with particle i at $L\vec{m} + \vec{r}_i$. The array consists of those copies for which $|L\vec{m}| \leq R$, with $R \to \infty$. In the system defined above, the diffusion tensor is given by [8]:

$$D_{ij} = \frac{k_B T}{6\pi\eta a}\left(\delta_{ij} I + \left(\frac{3a}{4L} O_{PBC}\left(\frac{\vec{r}_{ij}}{L}\right) + \frac{a^3}{2L^3} Q_{PBC}\left(\frac{\vec{r}_{ij}}{L}\right)\right)\right) \tag{23}$$

where I is 3×3 identity matrix and δ_{ij} is the Kronecker's delta. Following the notation introduced in the work of Smith et al. [8] we introduce:

$$O(\vec{r}) = \frac{1}{r}\left(I + \hat{r}\hat{r}^T\right) \qquad Q(\vec{r}) = \frac{1}{r^3}\left(I - 3\hat{r}\hat{r}^T\right) \qquad \hat{r} = \frac{\vec{r}}{r} \tag{24}$$

and

$$\vec{\sigma} = \frac{\vec{r}_{ij}}{L} \tag{25}$$

Now, for $\vec{\sigma} \neq \vec{m}'$ (interactions between different particles within the primary cell or between a particular particle in the primary cell with an image of other particle) we have the following Ewald formulas[8]:

$$O_{PBC}(\vec{\sigma}) = \sum_{\vec{m}} \left[erfc(\alpha|\vec{m}+\vec{\sigma}|)O(\vec{m}+\vec{\sigma}) + \frac{2\alpha}{\sqrt{\pi}} e^{-\alpha^2(\vec{m}+\vec{\sigma})^2} \frac{(\vec{m}+\vec{\sigma})(\vec{m}+\vec{\sigma})^T}{|\vec{m}+\vec{\sigma}|^2} \right] + \sum_{\vec{n}\neq 0} \frac{2}{\pi \vec{n}^2} e^{-\frac{\pi^2\vec{n}^2}{\alpha^2}} e^{2\pi i \vec{n}\vec{\sigma}} \left[I - \left(1 + \frac{\pi^2\vec{n}^2}{\alpha^2}\right) \hat{n}\hat{n}^T \right] \tag{26}$$

$$Q_{PBC}(\vec{\sigma}) = \sum_{\vec{m}} \left[erfc(\alpha|\vec{m}+\vec{\sigma}|) + \frac{2\alpha}{\sqrt{\pi}} |\vec{m}+\vec{\sigma}| e^{-\alpha^2(\vec{m}+\vec{\sigma})^2} \right] Q(\vec{m}+\vec{\sigma}) - \sum_{\vec{m}} \frac{4\alpha^3}{\sqrt{\pi}} e^{-\alpha^2(\vec{m}+\vec{\sigma})^2} \frac{(\vec{m}+\vec{\sigma})(\vec{m}+\vec{\sigma})^T}{|\vec{m}+\vec{\sigma}|^2} + 4\pi \sum_{\vec{n}\neq 0} e^{-\frac{\pi^2\vec{n}^2}{\alpha^2}} e^{2\pi i \vec{n}\vec{\sigma}} \hat{n}\hat{n}^T \tag{27}$$

Additionally, for $\vec{\sigma} = \vec{m}'$ (interactions between a particular particle in the primary cell with its self-images across the array) we have [8]:

$$O_{PBC}(\vec{m'}) = \sum_{\vec{m}\neq 0} \left[erfc(\alpha|\vec{m}|)O(\vec{m}) + \frac{2\alpha}{\sqrt{\pi}} e^{-\alpha^2\vec{m}^2} \hat{m}\hat{m}^T \right] - \frac{3\alpha a}{2\sqrt{\pi}L} I + \sum_{\vec{n}\neq 0} \frac{2}{\pi\vec{n}^2} e^{-\frac{\pi^2\vec{n}^2}{\alpha^2}} \left[I - \left(1 + \frac{\pi^2\vec{n}^2}{\alpha^2}\right) \hat{n}\hat{n}^T \right] \tag{28}$$

$$Q_{PBC}(\vec{m'}) = \sum_{\vec{m}\neq 0} \left[erfc(\alpha|\vec{m}|) + \frac{2\alpha}{\sqrt{\pi}} |\vec{m}| e^{-\alpha^2\vec{m}^2} \right] Q(\vec{m}) - \sum_{\vec{m}\neq 0} \frac{4\alpha^3}{\sqrt{\pi}} e^{-\alpha^2\vec{m}^2} \hat{m}\hat{m}^T - \frac{1}{3\sqrt{\pi}} \left(\frac{\alpha a}{L}\right)^3 I + 4\pi \sum_{\vec{n}\neq 0} e^{-\frac{\pi^2\vec{n}^2}{\alpha^2}} \hat{n}\hat{n}^T \tag{29}$$

In the equations given above, $erfc()$ is the complementary error function. Vectors $\vec{m}$ and $\vec{n}$ have integer components. Summations over the vectors $\vec{n}$ and $\vec{m}$ give reciprocal and real-space contributions to the diffusion tensor. The above equations can be extended to a system of spherical particles with different radii a_i. One need

only replace in each of the above equations the radius a to the first power by a_i and the radius a to the third power by $\frac{1}{2}a_i\left(a_i^2+a_j^2\right)$ [9].

An additional modification of the original formulation of Smith et al. [8] is a correction to the final form of the periodic form of the diffusion tensor that allows for overlaps between spherical particles [34] :

$$
\begin{aligned}
D_{ij}^{overlap} &= D_{ij} + A_{ij} - B_{ij} \\
A_{ij} = \frac{k_B T}{6\pi\eta a_i a_j}\left[\frac{16r_{ij}^3(a_i+a_j)-((a_i-a_j)^2+3r_{ij}^2)^2}{32r_{ij}^3}\boldsymbol{I} + \frac{3((a_i-a_j)^2-r_{ij}^2)^2}{32r_{ij}^3}\frac{\vec{r}_{ij}\vec{r}_{ij}^T}{r_{ij}^2}\right] &\quad (a_> - a_<) < r_{ij} \le (a_i+a_j) \\
A_{ij} = \frac{k_B T}{6\pi\eta a_>} &\quad r_{ij} \le (a_> - a_<) \\
B_i j = \frac{k_B T}{8\pi\eta r_{ij}}\left[\left(1+\frac{a_i^2+a_j^2}{3r_{ij}^2}\right)\boldsymbol{I} + \left(1-\frac{a_i^2+a_j^2}{r_{ij}^2}\right)\frac{\vec{r}_{ij}\vec{r}_{ij}^T}{r_{ij}^2}\right] &\quad r_{ij} > (a_i+a_j)
\end{aligned}
\tag{30}
$$

where D_{ij} is the Ewald-summed 3×3 diffusion tensor for the two overlapping particles (or particle and an image of another particle) i and j given with Equation 23.

6 Overlaps

The presence of a Gaussian random displacement vector in the BD integration scheme (Equations 12) may lead to nonphysical overlaps between beads. Overlaps occur when the distance between any two subunits is smaller than the sum of their hard-core radii. *bd_flex* can check whether a particular simulation step leads to an overlap between subunits. Next, positions of subunits are corrected to remove all cases of overlaps. This can be done in one of the three ways. One option is that the current step of a simulation is rejected, positions of all subunits are reseted to initial ones and a new step is attempted with different random vector, until there are no cases of overlap. 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 overlaps, the time step is restored to its initial value. As a third option, we also implemented the elastic collision method [16], in which all the collisions between subunits are considered explicitly as the elastic collision. A numerical procedure applied at each step of a simulation locates time, collision partners and parameters for every collision occurring in the system in chronological order and corrects positions of spherical subunits accordingly, using laws of classical mechanics.

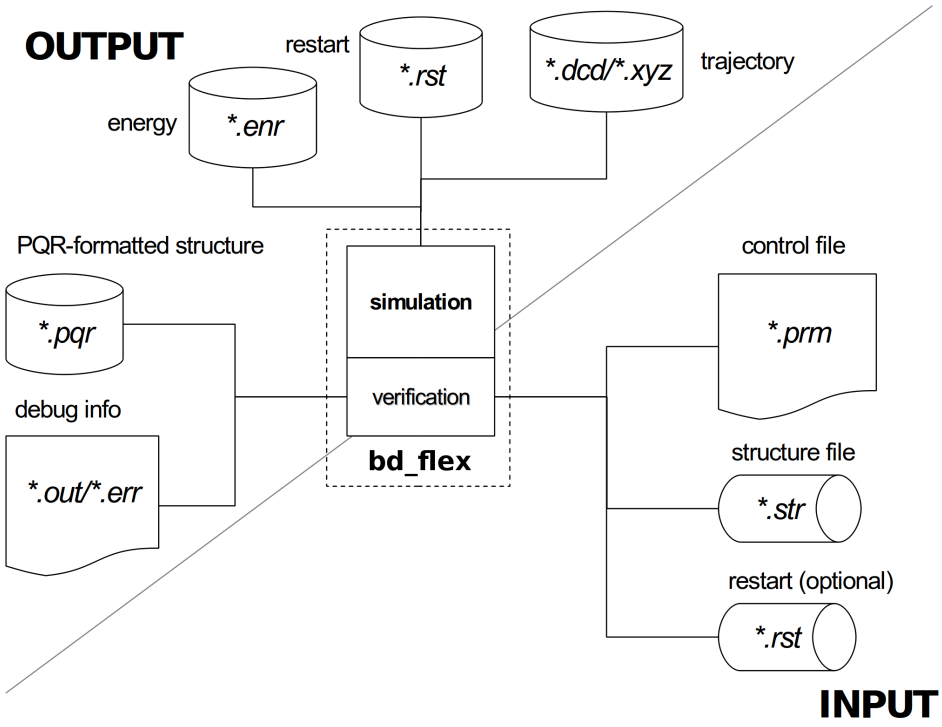


Figure 2: *bd_flex* schematics

7 Running Simulations with bd_flex

Two input files are needed to perform BD simulations with the *bd_flex* module of BD_BOX (Figure 2). The structure file (*.str) contains the definition of the molecular system to be simulated: initial positions of beads (Cartesian coordinates), their identifiers (name and index), their parameters such as radii, charges, masses, well depths of Lennard-Jones interactions, and their connectivities (bonds, planar angles and dihedrals). Simulation parameters are stored in the text control file (*.prm) (Figure 2). These include for example the type of the propagation algorithm and its time step, temperature, viscosity, ionic strength, definition of boundaries, the treatment of hydrodynamic interactions, values of cutoffs for nonbonded interactions, specifications of the output (names and types of files, write frequency), path to the input structure file and many others.

The output from *bd_flex* consists of a few files (Figure 2). 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 (names, coordinates, charges and radii of beads) 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 text XYZ format (*.dcd/*.xyz) defines the behavior of

the studied system along time. 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 control and structure files, the user may run the simulation with the command:

bd_flex file.prm

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

bd_flex −−parameter=value

or

bd_flex file.prm −−parameter=value

7.1 Input Files

Both input (*.str, *.prm) files are loosely formatted text files. Below we describe their structure and some rules that must be followed upon their construction.

7.1.1 Structure File

The BD_BOX structure file contains the definition of the studied system: initial positions of subunits (beads), their parameters (such as charges and radii), their bonded and nonbonded interactions parameters. Meaningful lines in the structure file are these that begin with one of the words: *sub*, *bond*, *angle* or *dihe*.

- *sub* lines - each line describes a single spherical subunit and its parameters:

  sub name id x y z σ Q 2·R ϵ^{LJ} *m*

  where:

$name$ - the name of the subunit

id - an unique identifier assigned to the subunit (integer)

x y z - Cartesian coordinates of the subunit

σ - the hydrodynamic radius of the subunit

Q - the central charge

R - the hard-core (Lennard-Jones) radius (note that the value of R should be doubled in the structure file)

ϵ^{LJ} - Lennard-Jones well depth

m - mass (mass is only needed when the elastic collision method is to be used during the simulation. However, this column is obligatory. As only the ratio of masses of different subunits is important during evaluation of collisions within pairs of particles, one does not need to care about units and particular values.)

- *bond* lines - each line define a single bond and its parameters (Equation 1)

  bond *id(1)* *id(2)* r_o r_{max} *H*

  where:

  id(1) and *id(2)* - identifiers of subunits to be connected

  r_o - the equilibrium bond length

  r_{max} - the maximum bond length

  H - the force constant

- *angle* lines - each line defines a single planar angle (between bonds connecting subunits id(1), id(2) and id(2), id(3)). Currently, two types of the planar angle potential are supported:

  angle *angle* *id(1)* *id(2)* *id(3)* ϕ^o k_ϕ

  angle *cos* *id(1)* *id(2)* *id(3)* ϕ^o k_ϕ

  where:

  angle - defines the form of the potential to be used - Equation 3

  cos - defines the form of the potential to be used - Equation 2

  id(1), id(2), id(3) - identifiers of subunits in the planar angle

  ϕ^o - the equilibrium angle value

  k_ϕ - the force constant

- *dihe* lines - each line defines a single dihedral angle (subunits id(1),id(2),id(3),id(4) - rotation around the id(2)-id(3) bond). Two types of the dihedral angle potential are supported:

  dihe angle id(1) id(2) id(3) id(4) θ^o k_θ

  dihe cos id(1) id(2) id(3) id(4) k_θ *m cos(*δ*)*

  where:

  angle - defines the form of the potential to be used - Equation 4

  cos - defines the form of the potential to be used - Equation 5

  k_θ - the force constant

  θ^o - the equilibrium angle value

  cos(δ*)* - the phase

  m - multiplicity

7.1.2 Simulation Control File

The control file contains 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.

8.1 Input/Output Control

restart string - this keyword indicates that a continuation run should be performed, using the *string* restart file (input); restarts should be specified using command line

out_filename string - the name of the plain text output file

save_xyz_freq integer value - the frequency for writing to the XYZ trajectory file, default *1*

save_dcd_freq integer value - the frequency for writing to the DCD trajectory file, default *1*

save_rst_freq integer value - the frequency for writing to the restart file, default *1*

save_enr_freq integer value - the frequency for writing to the energy file, default *1*

str_filename string - the name of the input structure file

xyz_filename string - the name of the XYZ trajectory file (output)

dcd_filename string - the name of the DCD trajectory file (output)

enr_filename string - the name of the energy file (output)

rst_filename string - the name of the restart file to be written

pqr_filename string - the name of the PQR structure file (output)

8.2 Nonbonded Interactions

alpha_lj float - the α (Equation 7) parameter for Lennard-Jones interactions scaling, typically *4.0*, default: *4.0*

cutoff_lj float - the cutoff radius for Lennard-Jones interactions, default: *0.0*. Setting the *cutoff_lj* to *-1* results in cutoffs defined for each pair of the subunits as $2^{\frac{1}{6}}(R_i + R_j)$

lj_6_term yes/no - whether to use the $\left(\frac{1}{r}\right)^6$term in the Lennard-Jones potential (Equation 7), default: *yes*

epsilon_c float - the dielectric constant of the immersing medium (water), default: *78.54*

kappa_c float - the inverse of the Debye screening length, the κ parameter in Equation 6, default: *0.1*

cutoff_c float - the cutoff radius for electrostatic interactions, default: *0.0*.

gamma_c float - unit conversion factor, the γ parameter in the Equation 6, default (conversion to $\frac{kcal}{mol}$) *331.842*

elec yes/no - whether electrostatic interactions should be evaluated, default *yes*

bond_lj_scale float - the scaling factor for Lennard-Jones interactions between bonded (1-2) pseudoatoms, default: *1.0*

bond_c_scale float - the scaling factor for electrostatic interactions between bonded (1-2) pseudoatoms, default: *1.0*

nb_list string - this keyword specifies the algorithm used to create nonbonded interactions lists, *brute* (default, brute force), *spatial* (grid-based), *verlet* (explicit creation of Verlet lists)

verlet_count integer - frequency of re-creation of nonbonded interactions (Verlet) lists, default: *1*

verlet_roff *float* - outer cutoff for creation of nonbonded interactions (Verlet) lists, default: *10.0*

8.3 Boundaries

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*

NOTE: Currently, only cubical boxes are supported in case of Ewald-summed diffusion tensors

bc *string* - boundary conditions to be used, either *none*, *pbc* (in case of periodic boundary conditions) or *sphere*, default: *none*

sboundary *yes/no* - whether to use the bounding sphere potential (Equation 11), default: *no*. 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

sphere_radius *float* - the radius of the bounding sphere, default: *0.0*. This keyword applies either when *bc sphere* or *sboundary yes* is specified

sboundary_A *float* - the magnitude of the bounding sphere force, the *A* parameter in Equation 11, default: *0.0*

sboundary_n *integer* - the power of the radial distance dependence of the bounding sphere force, the *n* parameter in Equation 11, default: *0.0*

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

ewald_real *integer* - the magnitude of the real lattice vectors (Equation 22, Equations 26-29), default: *0*

ewald_recip *integer* - the magnitude of the reciprocal lattice vectors (Equation 22, Equations 26-29), default: *0*

ewald_alpha *float* - this parameter controls the convergence of the Ewald summation (α in Equations 26-29), default: $\sqrt{\pi}$

ewald_method *string* - the method used to build the Ewald-summed diffusion tensor, default: *smith* as in [8]

8.4 External Electric Fields

E_ext *yes/no* - whether to switch on/off the external electric field, 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 (Equations 9, 8 and 10), 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 Flows

vel_grad_tensor float float float float float float float float float - the velocity gradient tensor *G(i,j)* where *i* runs over rows and *j* runs over columns, *G(1,1) G(1,2) G(1,3) G(2,1) G(2,2) ... G(3,3)*, default: 9×*0.0*

8.6 Devices Control

MPI_nprow integer - the number of rows in the processor grid, default: *0*

MPI_npcol integer - the number of columns in the processor grid, default: *0*

MPI_block integer - the size of a block in the processor grid, default: *0*, this value should be less than the rank of the diffusion tensor matrix

cuda_devices integer integers - this keyword specifies CUDA devices to use, their number followed by their system identifiers

cuda_block integer - this is the dimension of the thread block, possible values are 32, 64, 128, 256, 512, 1024. The value of the *cuda_block* parameter depends on the hardware device and the number of beads in the simulated system, default: *256*; this keyword is relevant in case of devices with compute capability less than 2.0

NOTE:

MPI_nprow×MPI_npcol should be equal or lesser than the number of computational nodes used

Keywords MPI_nprow, MPI_npcol and MPI_block are relevant only when SCALAPACK is being used.

8.7 Algorithms

dt *float* - the time step, in [ps], default: *0.0*

bdsteps *integer* - the total number of simulation's steps, default: *0*

hydro *string* - the method used to evaluate hydrodynamic interactions, can be set to *none* (a diagonal form of the diffusion tensor will be used throughout the simulation), *cholesky* (Cholesky decomposition of the diffusion tensor matrix [10]), *geyer* (the TEA-HI method of Geyer and Winter [13]), *lanczos* (the Krylov subspace approach [14]), or *blocklanczos* (the block variant of the Krylov subspace approach [14])

algorithm *string* - algorithm for the trajectory generation. Possible values are: *igt_const*, *igt_var*, *ermak_const*, *ermak_var*, *ermak_newton*; *igt(ermak)* means that either the Ermak algorithm or the IG-T algorithm will be used, either with a constant or variable time step (*const(var)*). The elastic collision method can be switched on by setting *algorithm* to *ermak_newton*.

rand_seed *integer* - a seed for the random number generator, default: *12345*

check_overlap *yes/no* - whether to check for overlaps in the studied system after each simulation's step, default: *yes*.

move_attempts *integer* - the number of attempts of repeating a particular simulation step (with a different random vector) when overlaps between beads are detected in the system. When this number is exhausted and there still are overlaps, the program will stop, default: *10000*

geyer_on_the_fly *yes/no* - whether to compute hydrodynamically correlated random displacements (within the TEA-HI framework) using the whole diffusion tensor matrix (*no*) or, to save memory, using 3×3 submatrices (*yes*), default: *yes*

lanczos_m - number of steps in both of the Lanczos approaches [14], default: 4

lanczos_s - size of the normal vector block in the block variant of the Lanczos approach [14], default: 4

e_collision float - the restitution parameter used in the elastic collision approach, falls between *0.0* (perfectly inelastic collisions) and *1.0* (perfectly elastic collisions), default: *1.0*

8.8 Physical Conditions

T float - temperature [K], default: *298.15*

visc float - viscosity [Poise], default: *0.01*

vfactor float - factor for conversion of viscosity units, default: *14.4*, (to convert from Poise to $\frac{kcal}{mol}$ ps $\AA^3$ should be set to 14.4)

9 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/flex/name* directories

DNA: a single DNA-like polymer, composed of touching beads (52 beads), with excluded volume interactions modeled via Lennard-Jones potentials and hydrodynamic interactions evaluated based on the Cholesky decomposition of the diffusion tensor. Bonded interactions include flexible bonds and deformable planar angles.

DNA_BSPHERE: 20 DNA-like polymers (see above) enclosed inside of a bounding sphere. Hydrodynamic interactions are evaluated based on the TEA-HI approach.

HARD_SPHERES, HARD_SPHERES_CUDA: the periodic system of hard spheres (512 beads). Overlaps are corrected using the elastic collision method. Hydrodynamic interactions are evaluated based on the Cholesky

decomposition of the Ewald-summed diffusion tensor. GPU and CPU simulation setup files are included.

DBELLS_EXT_E : a periodic system containing 256 spring-bead dimers with oppositely charged beads. Non-bonded interactions between dimers are evaluated using screened electrostatics and Lennard-Jones potentials. Dimers move under the influence of an external electric (rotate) field. Hydrodynamic interactions are evaluated based on the Cholesky decomposition of the Ewald-summed diffusion tensor. A GPU simulation setup file is included.

CI2_CHOLESKY, CI2_TEA_HI : a hydrodynamic model of the chymotrypsin inhibitor 2 protein (see Figure 1). Reduced representation of the protein with beads positioned on atoms C_α. Beads within the distance of 15Å are connected with harmonic springs.

10 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 mathods 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] V Tozzini. Multiscale modeling of proteins. *Acc Chem Res*, 43:220–230, 2010.

[23] V Tozzini, J Trylska, C E Chang, and J A McCammon. Flap opening dynamics in hiv-1 protease explored with a coarse grained model. *J Struct Biol*, 157:606–615, 2007.

[24] A Arkhipov, P L Freddolino, and K Schulten. Stability and dynamics of virus capsids described by coarse-grained modeling. *Structure*, 14:1767–1777, 2006.

[25] T Frembgen-Kesner and A H Elcock. Striking effects of hydrodynamic interactions on the simulated diffusion and folding of proteins. *J Chem Theory Comput*, 5:242–256, 2009.

[26] G D R Echenique, R R Schmidt, J J Freire, J G H Cifre, and J G de la Torre. A multiscale scheme for the simulation of conformational and solution properties of different dendrimer molecules. *J Am Chem Soc*, 131:8548–8556, 2009.

[27] N C Horton and J J Perona. Crystallographic snapshots along a protein-induced dna-bending pathway. *Proc Natl Acad Sci USA*, 97:5729–5734, 2000.

[28] W Humphrey, A Dalke, and K Schulten. VMD – Visual Molecular Dynamics. *J Mol Graphics*, 14:33–38, 1996.

[29] J G de la Torre, M L Huertas, and B Carrasco. Calculation of hydrodynamic properties of globular proteins from their atomic-level structure. *Biophys J*, 78:719–730, 2000.

[30] S Ludvigsen, H Y Shen, M Kjaer, J C Madsen, and F M Poulsen. Refinement of the three-dimensional solution structure of barley serine proteinase inhibitor 2 and comparison with the structures in crystals. *J Mol Biol*, 222:621, 1991.

[31] Y Wang, C Li, and G J Pielak. Effects of proteins on protein diffusion. *J Am Chem Soc*, 132:9392, 2010.

[32] 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.

[33] P J Zuk, E Wajnryb, K A Mizerski, and P Szymczak. Rotne-Prager-Yamakawa approximation for different-sized particles in application to macromolecular bead models. *J. Fluid. Mech.*, 741:R5, 2014.

[34] T Zhou and B Chen. Computer simulations of diffusion and dynamics of short-chain polyelectrolytes. *J Chem Phys*, 124:034904, 2006.