Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

Browndye: A software package for Brownian dynamics ☆

Gary A. Huber [a,*], J. Andrew McCammon [a,c,d,b]

[a] Howard Hughes Medical Institute, University of California San Diego, La Jolla, CA 92093-0365, United States
[b] Center for Theoretical and Biological Physics, University of California San Diego, La Jolla, CA 92093-0365, United States
[c] Department of Chemistry and Biochemistry, University of California San Diego, La Jolla, CA 92093-0365, United States
[d] Department of Pharmacology, University of California San Diego, La Jolla, CA 92093-0365, United States

A R T I C L E   I N F O

Article history:
Received 2 June 2010
Received in revised form 2 July 2010
Accepted 7 July 2010
Available online 16 July 2010

Keywords:
Brownian dynamics
Protein association
Software
Diffusion
Kinetics

A B S T R A C T

A new software package, Browndye, is presented for simulating the diffusional encounter of two large biological molecules. It can be used to estimate second-order rate constants and encounter probabilities, and to explore reaction trajectories. Browndye builds upon previous knowledge and algorithms from software packages such as UHBD, SDA, and Macrodox, while implementing algorithms that scale to larger systems.

Program summary

Program title: Browndye
Catalogue identifier: AEGT_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AEGT_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: MIT license, included in distribution
No. of lines in distributed program, including test data, etc.: 143 618
No. of bytes in distributed program, including test data, etc.: 1 067 861
Distribution format: tar.gz
Programming language: C++, OCaml (http://caml.inria.fr/)
Computer: PC, Workstation, Cluster
Operating system: Linux
Has the code been vectorised or parallelized?: Yes. Runs on multiple processors with shared memory using pthreads
RAM: Depends linearly on size of physical system
Classification: 3
External routines: uses the output of APBS [1] (http://www.poissonboltzmann.org/apbs/) as input. APBS must be obtained and installed separately. Expat 2.0.1, CLAPACK, ocaml-expat, Mersenne Twister. These are included in the Browndye distribution.
Nature of problem: Exploration and determination of rate constants of bimolecular interactions involving large biological molecules.
Solution method: Brownian dynamics with electrostatic, excluded volume, van der Waals, and desolvation forces.
Running time: Depends linearly on size of physical system and quadratically on precision of results. The included example executes in a few minutes.
References:
[1] N. Baker, D. Sept, S. Joseph, M. Holst, J. McCammon, Electrostatics of nanosystems: Application to microtubules and the ribosome, Proc. Natl. Acad. Sci. USA 98 (18) (2001) 10037–10041.

1. Introduction

One of the main determinants of biological structure and function is the interaction of two or more molecules, especially protein molecules. Understanding the dynamics of these bi-molecular interactions is important for the understanding of such cellular

☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.
E-mail address: gahuber95@gmail.com (G.A. Huber).

0010-4655/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cpc.2010.07.022

structures as the cytoskeleton (actin and tubulin, for example), ribosomes, chromosomes, and polymerases, as well as processes such as cell signalling and cell motility [1,2]. Furthermore, the encounter stages of such reactions, which are often the rate-limiting steps, are diffusion-limited. Therefore, use of Brownian dynamics is appropriate for such systems (see Ref. [3] for an excellent review). For several decades, Brownian dynamics has found use in polymer and peptide simulations [4–6] and simulations of enzyme-substrate reactions [7–10]. More recently, it has found use in studying protein–protein association reactions [11,12].

The Browndye software package consists of two simulation programs and about 25 auxiliary programs for processing data. It can compute the 2nd-order rate constant of the encounter of two rigid bodies moving according to Brownian dynamics (BD), and it can also compute the probabilities of the two bodies moving from one binding mode to another, as well as outputting the molecules' trajectories. It has functionality very similar to the packages SDA [11] and MacroDox [13], and like those two other packages, is intended primarily for simulations of large biological molecules. Its current limitations arise mainly from the approximation of flexible molecules as rigid bodies and the approximate nature of the force computations between the molecules. It also draws ideas from an earlier package, UHBD [8].

2. Model description

Given unlimited computational resources, one would want to include every atom in the solvent and the large molecules in molecular simulations. Unfortunately, such detailed models are still impractical for simulations of macromolecular encounters because of the large number of individual objects and the disparity in time scales. Therefore, common practice is to model the solvent using continuum models, along with a simplified description of the macromolecules. A detailed model of the solvent is replaced with two simplified models. First, the thermal motion and hydrodynamic drag are replaced by a suitable stochastic force on the macromolecules. Second, the ion concentration and dielectric properties of the water are used in a continuum-based models for computation of the electrostatic forces between the macromolecules [14,15]. A final assumption of the Brownian model is that the solvent damping is very large compared to inertial effects, turning the macromolecules' trajectories into a random walk with no defined linear or angular velocities. The general equation describing the motion is [4,16–22,3]:

$$d\begin{pmatrix}\mathbf{x}_1\\ \boldsymbol{\phi}_1\\ \mathbf{x}_2\\ \boldsymbol{\phi}_2\end{pmatrix} = \frac{dt}{k_bT}\mathbf{D}\cdot\begin{pmatrix}\mathbf{F}_1\\ \mathbf{T}_1\\ \mathbf{F}_2\\ \mathbf{T}_2\end{pmatrix} + \sqrt{2\,dt}\,\mathbf{S}\cdot\mathbf{w} + \nabla\cdot\mathbf{D}\,dt \qquad (1)$$

where the vectors $\mathbf{x}_1$, $\mathbf{x}_2$, $\boldsymbol{\phi}_1$, and $\boldsymbol{\phi}_2$ are the positions and orientations of the molecules, dt is the time step size, the matrix $\mathbf{D}$ is a generalization of the diffusivity which can also depend on positions and orientations, and the matrix $\mathbf{S}$ is the matrix square root of $\mathbf{D}$ (often computed using the Cholesky decomposition). The vector $\mathbf{w}$ comprises 12 independent random Gaussian variables, each with zero mean and unit variance. Computing the terms in this equation and updating the molecules' states with it is the main computational task of Browndye.

The model for the rigid-body molecule is an assemblage of possibly overlapping charged spheres, grouped into "residues". Each sphere and residue can also have a "type". Although a common use of this model is to equate each sphere with an atom, and, in the case of proteins, to equate each residue with an amino acid, one can generalize this model to other molecular models. For example, coarse-grained models of proteins often treat each amino acid residue as a sphere.

In computing an association rate constant, it is necessary to decide when a reaction has taken place. A reaction criterion is specified by pairs of spheres, with each sphere on a different molecule. Each pair is also assigned a distance. The reaction criterion itself is further characterized by a number n. A pair is said to be "satisfied" if its sphere centers are closer than the assigned distance. If n pairs are satisfied, then the reaction is said to occur. For example, a criterion might be composed of all pairs of atoms forming hydrogen bonds in a protein–protein complex, and if any three of the pairs are within 5.1 Å. In addition to containing pairs, a reaction criterion can contain other reaction criteria; these inner criteria are treated like pairs to be satisfied. An example would be an enzyme with two active sites. The binding of the substrate to each site would be represented by a separate reaction criterion, but the two reaction criteria could be lumped into another criterion to represent the overall reaction.

Entire reaction pathways can be specified by named **states** and named **reactions**. As an example, suppose the system comprises a substrate that can react first at one active site on an enzyme, and then can react at a second active site. When the molecules are initialized for a trajectory, the system is considered to be in the *reactants* state, and if they react, the system moves to the *product-1* state. The reaction itself is called *reaction-1*. The first reaction can then be followed by binding to the other site. The following state is called *product-2*, and the reaction is called *reaction-2*. An example of the input file specifying such a system is given in the documentation. The output of the Browndye software gives rate constants or probabilities for each named reaction.

The forces appearing in Eq. (1) fall into two categories: electrostatic and short-ranged. The electrostatic forces are computed from precomputed solutions to the Poisson–Boltzmann equation describing the electric field around each molecule in isolation

$$\nabla\cdot[\epsilon\nabla\Phi] = -\rho - \lambda\sum_i c_i z_i \exp\left[-\frac{z_i\Phi}{k_bT}\right] \qquad (2)$$

where Φ is the electric potential, ϵ is the dielectric of the material (either solvent or macromolecule), ρ is the permanent charge density in the macromolecule, and c_i and z_i are the bulk concentration and charge of the ith ion species in the solvent. The factor λ depends on position and describes where mobile ions can penetrate; it typically takes a value of 1 in the solvent and 0 in the macromolecular interior. The electric potential around each molecule is typically computed by using the APBS software package [15] and stored in one or more rectilinear grids. Given the electric field around one molecule, Coulombic forces can be computed from its interaction with the sphere charges on the other molecule. This model alone is not sufficient, because as the molecules approach, their electric fields cause a build-up of induced charge at the dielectric boundaries between the macromolecules and solvent. This effect is partially included by computing so-called *desolvation forces* described below.

The short-ranged forces are of two types. The default short-ranged force is simply a hard-sphere exclusion force; if two spheres on different molecules overlap after a time step, the step is rejected. Browndye also includes an option to use Lennard–Jones forces between spheres; this is described by the pairwise potential energy between two spheres:

$$V(r) = \epsilon\left[\left(\frac{r_m}{r}\right)^{12} - 2\left(\frac{r_m}{r}\right)^{6}\right] \qquad (3)$$

where r is the intersphere distance, ϵ is energy well depth, and r_m is the sum of the two radii. (This formula differs slightly from that seen commonly in the literature, in that the energy minimum occurs at the radii sum.) The radius $r_{m,i}$ and energy ϵ_i are specified by the user for each sphere and residue type, with ϵ for a sphere pair computed from the mixing rule

$$\epsilon = \sqrt{\epsilon_i \epsilon_j} \tag{4}$$

It is possible to have both hard-sphere and Lennard–Jones forces in the system. Finally, Browndye includes an option for a "softer" Lennard–Jones-type interaction, the "8–6" force [23]:

$$V(r) = 4\epsilon \left[\frac{3}{4} \left(\frac{r_m}{r} \right)^8 - \left(\frac{r_m}{r} \right)^6 \right] \tag{5}$$

Finally, the rate constant must be extracted from trajectories following Eq. (1) and the reaction criteria. This is done by running many trajectories, usually several thousand or more. A trajectory is started by positioning the molecule centers a distance b from each other, with randomly-chosen orientations. The molecules are stepped forward until they either react, in which case the trajectory ends, or they attain a distance q slightly larger than b. A random number is then used to decide whether the molecules escape from each other, thereby ending the trajectory. If they do not escape, they are again placed a distance b from each other, but with the orientations drawn from a specific probability distribution, and the trajectory continues. After the desired number of completed trajectories has been generated, the rate constant is computed from the ratio of reacted versus escaped trajectories [24]. In the case of complex reaction pathways, the movement of the molecules from state to state is recorded, until either the molecules escape or a final state with no reactions leading out of it is reached. Probabilities and rate constants are reported with 95% confidence intervals. The more trajectories are run, the more narrow are the intervals. An alternative method for computing the rate constant is also provided by the Weighted-Ensemble Brownian Dynamics method [25]; this can be useful in cases where the probability of reaction is very small compared to that of escape.

3. Software description

The inputs into Browndye are the following. First of all, files describing the spheres' relative positions, radii, and charges, as well as their types and grouping into residues are required for each molecule. These files are in the PQR format, which is a variation on the Protein Data Bank file format. Next, files containing the electric field around each molecule are required; as mentioned above, these can be obtained from APBS and are in the DX format [26]. The reaction criteria are provided using an XML file which describes the pairs of spheres, their necessary distances, and number required for each reaction. Finally, an input file, also in XML format, is included, which gives information on the previously-mentioned files, physical properties of the solvent, and details regarding the simulation itself, such as various time-step size and force parameters, number of trajectories, information for outputting trajectory information, as well as many others.

The inputs are not fed directly into the simulation programs, but are processed by several auxiliary programs. This is done for several reasons. First of all, the preprocessing is done only once, but can still take some time; if it gets interrupted, it is much easier to restart and continue if the intermediate results are stored in files. For the software developer, it is much easier to maintain and change a collection of programs than one large, "black-box" program. Perhaps most importantly, the user can also replace any of the intermediate files with his own information and continue the preprocessing; this gives considerable flexibility. The user can also run any of the auxiliary programs on his data. For example, several users have opted to use a collection of "effective charges" (described below) which is more robust than what is currently provided by Browndye.

With the exception of the DX files containing the electric field data, the intermediate files are in XML format; even the PQR files are converted to an equivalent XML format. Using XML files has two main advantages. For the user, the tags provide a human-readable commentary on the data. For the programmer, changing the format of the file by adding or removing information in the form of XML tags can be done without being required to rewrite the software that parses the file. It is also possible to automatically check an XML file against an expected format and to quickly flag any inconsistencies, although Browndye does not currently make use of this feature. The programs of Browndye use the freely available and light-weight XML parser Expat [27].

Almost all of the auxiliary programs are written in the Objective Caml (or OCaml) language [28], which is a variant of the ML computer language. OCaml is a statically-typed, garbage-collected functional language which combines the conciseness, safety, and ease of programming offered by functional and scripting languages with the performance of procedural languages such as C++ and Fortran. Each auxiliary program is small enough that it can be treated as a black box, so the general unfamiliarity of scientists with OCaml should not be an issue.

On the other hand, the two simulation programs **nam_simulation** and **we_simulation**, which perform the trajectory and weighted-ensemble simulations, are written in C++. First of all, Ocaml might not be available on the computers at the various supercomputer centers. While the user can almost always preprocess the data on his own computer where he has some control, he might want to run the actual time-consuming simulation elsewhere. Also, a future goal of this project is to provide various generic algorithmic building blocks for building other software packages, and the templating facility of C++ is ideal for this. Finally, at this time, the Ocaml language does not offer support for concurrent multi-threaded execution on several processors.

Because the execution of Browndye consists of a moderately complex network of files and programs, another program, called **bd_top**, is used to keep track of which files are up-to-date and which auxiliary programs still need to run, and calls the necessary programs with appropriate arguments. This typically is the first program in the Browndye suite to be run, and it takes the XML input file mentioned above as its input. The program bd_top is written using an Ocaml module called the **orchestrator**, which is similar to the "make" utility on many Unix-based systems [29]. Like the "make" utility, it can compare the last-updated times of files and decide which programs to run. In addition, it can also compare tagged quantities between two XML files and extract arguments to programs from XML files. As Browndye grows and evolves, this allows the bd_top program to be updated accordingly. After the program bd_top is run, which in turn generates many more XML and DX files as well as an input file into one of the simulation programs, one of the simulation programs is run (Fig. 1). Because of the computationally intensive nature of the simulations, the two simulation programs have been written to make use of systems with several processors and shared memory. They make use of *pthreads*, the standard interface for systems-level multithreaded programming [30]. In the single-trajectory nam_simulation program, each trajectory can be run independently, so a thread is created for each processor. When a thread finishes with a trajectory, it pulls another future trajectory off of a queue, thus ensuring that each thread is fully occupied until the very end of the simulation. In the weighted-ensemble we_simulation program, all of the two-molecule system copies are advanced simultaneously at each time step, and the system time step advancements are distributed among the threads using the same queuing method. The simulation programs are structured so that most of the data associated with the molecules is shared among the threads. In particular, the data grids, which hold the electrostatic potentials and other quan-

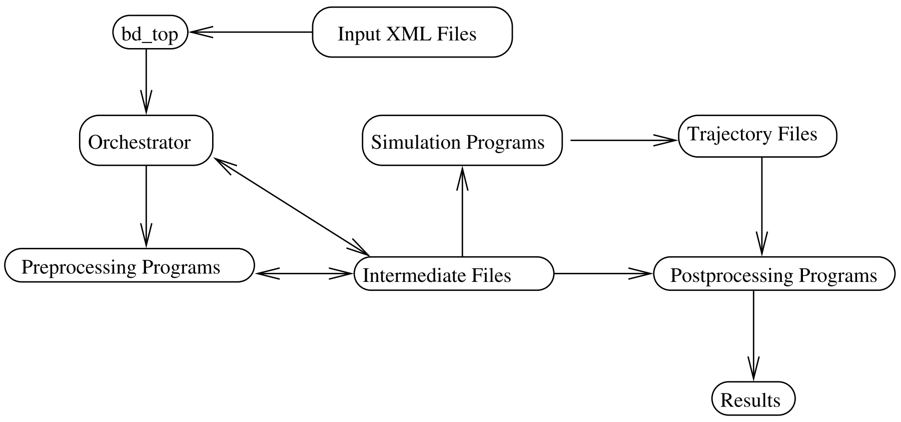

Fig. 1. Software structure.

tities, use the bulk of the total memory required. Fortunately, each large data grid can be read by any of the threads; no grid needs to be duplicated.

At present, there are two types of outputs from the simulation programs. As they run, they update an XML file containing raw information on which trajectories have escaped or completed given reactions. This information, at any time, may be given to another program, **compute_rate_constant**, which then computes second-order rate constants, along with their confidence intervals, and the relative probabilities of following the various reaction pathways. The other type of output, from nam_simulation, consists of the simulation trajectories themselves and their states. The output state consists of the relative positions and orientations, various components of the interaction energy, and the current reaction state. Because these files can be quite large, the numerical data for twenty adjacent states is encoded into a MIME format and embedded in XML in order to reduce the number of memory-consuming XML tags. In addition to trajectory files, an "index" file is output, which allows subsequent post-processing programs to quickly find certain trajectories with desired properties. The main post-processing program, **process_trajectories**, can be used to extract out portions of trajectories that correspond to desired reaction paths. The resulting sub-trajectories can be processed further by the program **xyz_trajectory** to generate files for visualization with the molecular visualization package VMD [31].

4. Algorithms and optimizations

Although Browndye has functionality similar to other packages mentioned above, it includes several algorithms and optimizations to help its performance scale well to larger systems. Most of these techniques are concerned with computing long-ranged forces, resolving collisions, and propagating the trajectories. Because each molecule is made up of a large number of spheres (N), any algorithm whose execution time would scale as N^2 is avoided, while algorithms that scale as N or $\log(N)$ are included, even if their benefit is not realized as strongly on smaller systems. Algorithms that help to reduce the number of distinct steps, along with their required force calculations, are also included.

As mentioned above, most of the required memory is consumed by the field grids; these contain the electric field around the larger molecule, and desolvation fields (see below) around both molecules. For points near the surface, a higher resolution is required, which in turn requires smaller spacing between the grid points. Further away, however, a coarser grid can be used. Browndye allows the use of such nested grids in order to save memory; any number of such grids can be imported from the output of APBS. Outside of the outermost grid, the electric potential data in the outer points is fit, using least-squares, to a Cartesian multipole expansion [32], out to quadrupole level, of the screened Coulombic interaction. Any point outside of the grids that needs a value of the potential or its gradient uses the multipole expression.

Perhaps the most prominent force is the Coulombic interaction among the two molecules; this can be computed from the charge at each sphere on the smaller molecule and the gradient of the electric potential at that point, usually by trilinear interpolation from the nearest eight grid points. Often, the "smaller molecule" can itself be quite large, and the charges on the large molecule are also used in the desolvation forces below. Looping over large numbers of charges at each time step can be time-consuming, so several approaches are used to reduce the effort.

The first step is to reduce the number of charges to a smaller set of "effective charges". One commonly-used approach, used in the SDA package, is to fix a charge position on each charged residue and to perform a least-squares fit of the charges to the surrounding electric field [33]. The main drawbacks are that the effort to compute the charge values scales unfavorably with large systems, and a certain amount of experimenting with a regularization parameter is required in order to avoid unphysical charge values. Currently, Browndye uses a simpler, "test charge" approach by placing one point charge at the center-of-charge at each charged residue, with the point charge value equal to the total residue charge. As mentioned above, however, it is possible and desirable to import effective charges from SDA or other methods by replacing one of the intermediate XML files and running bd_top again to update the other files. Future improvements to Browndye will include more robust and quickly-computed effective charges.

Given a set of effective charges, it is possible to lump point charges together dynamically as the simulation proceeds. If the electric field is relatively smooth in the region of a set of charges, a coarse-grained description can be used in order to avoid looping over all of the charges and computing the field gradient at each point. In general, the further away from a charge source one is, the smoother is the field, so a decision to use a coarse-grained model can be based on distance from the other molecule during the simulation.

Browndye surrounds the molecule with a box, and divides the box into two smaller boxes along its longest axis so that equal numbers of charges fall into the two child boxes. This process is recursively repeated until a small number of charges remains in each bottom-level box. Each box has 64 quadrature points at which the electric potential is evaluated, and the obtained values are used to compute the force and torque on the box (Appendix A). The model used, which is based on Chebyshev interpolation of the potential, assumes that the field is fairly smooth on the scale of the box. If the field is not smooth, then the child boxes need to be used for the force and torque evaluation. At each time step, the algorithm starts at the top box and recurses down the hierarchy of boxes. If the distance from the box center to the nearest point charge on the other molecule becomes less than a certain factor times the diagonal length of the box, then the children are used. An efficient algorithm, described below, is used to find the distance. If bottom-level box is reached, then the force and torque are computed from the point charges themselves and the field gradient.

The desolvation penalty force is computed by using a formula very similar to the one used in the SDA package [11], but with a prefactor inspired by the generalized Born model [34]. The contribution due to one effective charge interacting with the other molecule is:

$$\mathcal{E} = \frac{q^2\alpha(\epsilon_p^{-1} - \epsilon_s^{-1})}{32\pi^2\epsilon_0} \int (1+\kappa r)^2 \exp(-2\kappa r) r^{-4}\, dr \tag{6}$$

where ϵ_0 is the vacuum permittivity, ϵ_p and ϵ_s are the dimensionless dielectrics of the molecule interior and the solvent, κ is the inverse Debye length of the solvent, q is the value of the effective charge, r is the distance from the charge, and the integral is over the interior of the other molecule. The factor α, whose default is 1, may be included for a better match to the energy. The factor without the q^2 term is precomputed for each molecule and is stored on a grid, with r corresponding to the exterior grid points. The integral is performed using a Cartesian-based fast multipole method [35], allowing this preprocessing step to scale linearly with the size of the molecule. The desolvation grids match the shape and number of the electric potential grids. The squared effective charges are lumped in a recursive box structure as described above, and during the simulation, they interact with the precomputed desolvation in the manner as the charges interact with the electric field. The interior of the molecule is represented by a grid of integers (1 for inside, 0 for outside) of the same geometry as the smallest electrostatic grid enclosing the whole molecule. The generation of this grid is described below.

Hydrodynamic interactions are handled by using the variation of the Rotne–Prager tensor [36] developed by Garcia de la Torre and Bloomfield [37] for two spheres:

$$\mathbf{D}_{ij} = \frac{kT}{6\pi\mu\sigma_i}\mathbf{I}\delta_{ij} + \frac{kT}{6\pi\mu r_{ij}}\left[\left(\mathbf{I} + \frac{\mathbf{r}_{ij}\mathbf{r}_{ij}}{r_{ij}}\right) + \frac{\sigma_i^2+\sigma_j^2}{r_{ij}^2}\left(\frac{1}{3}\mathbf{I} - \frac{\mathbf{r}_{ij}\mathbf{r}_{ij}}{r_{ij}^2}\right)\right](1-\delta_{ij}) \tag{7}$$

where μ is the solvent viscosity, $\mathbf{r}_{ij}$ is the vector from sphere i to sphere j, and σ_i is the radius of sphere i. The diffusion matrix in Eq. (1) is given by

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_{11} & 0 & \mathbf{D}_{12} & 0 \\ 0 & \mathbf{D}_{r1} & 0 & 0 \\ \mathbf{D}_{21} & 0 & \mathbf{D}_{22} & 0 \\ 0 & 0 & 0 & \mathbf{D}_{r2} \end{pmatrix} \tag{8}$$

where the rotation matrices are given by [20]

$$D_{ri} = \frac{kT}{8\pi\mu\sigma_i^3} \tag{9}$$

This model neglects the hydrodynamic rotation–rotation and rotation–translation coupling. One significant advantage of the Rotne–Prager model is that the divergence of the diffusion matrix in Eq. (1) is zero.

Because the molecules, in general, are not perfect spheres, their effective radii are computed using the Monte Carlo surface integral method of Hansen [38]. One of the intermediate files is a grid of points with an indicator of whether the point is inside or outside the molecule (see below). Pieces of the molecule's surface, for this purpose, are defined within each cube whose vertices are made up of both inside and outside points. A plane is assumed to cut through each edge shared by an inside and outside point, and the areas and centers for each bit of surface are computed for each surface cube.

The effective gap between the molecules is computed by assuming that the molecules are ellipsoids. To find the equivalent ellipsoid of a molecule, the moment of inertia matrix is computed for the molecule interior. This matrix is inverted to obtain a transform that maps the points of the molecule onto a solid with an isotropic moment matrix. Then, the smallest enclosing sphere (see below) of the transformed molecule subspheres is found. This sphere is deformed back into an ellipsoid using the moment matrix. During the simulation, the intersection of the points on the two ellipsoids with the line segment connecting their centers is computed, and the gap is approximated as the distance between the two intersection points. The distance r_{12} in Eq. (7) is the sum of the effective radii and the effective gap.

One computational challenge in BD simulations is figuring out when a collision takes place between two rigid assemblages of spheres. One method would be to compute the distance between every pair of spheres, but the required time would scale unfavorably for large systems. The SDA and MacroDox packages address this by precomputing an exclusion grid for one molecule, and pretending that all the spheres on the other molecule have the same size. The solution used by Browndye is to enclose each molecule in a hierarchy of spheres which enclose the various parts of the molecules, and test the pairs of fewer and larger spheres, recursing downward if necessary. This is a simple version from the class of Hierarchical Bounding Volume methods [39]. First, a sphere is generated that closely approximates the minimal enclosing sphere of the molecule's spheres. The minimal enclosing sphere is approximated by first computing the minimal enclosing sphere of the centers using the method of Welzl [40]. Then, the sphere radius is increased until all of the spheres are inside. After this, the moment of inertia matrix is computed for the sphere centers, and sphere set is divided into equal halves by a plane perpendicular to the principal axis corresponding to the largest eigenvalue. Enclosing spheres are constructed for the sphere subgroups, which are divided again, until just a small number of molecule spheres remain in the group. During the simulation, the outer-most spheres are tested for overlap. If there is no overlap, then there is no collision. Otherwise, the two spheres on each molecule which are one level down are tested for collisions with each other (four different tests), and the algorithm recurses down until two atom-spheres are found which collide.

In effect, Browndye keeps the large molecule fixed, while letting the smaller molecule move. Of course, both molecules move, but after each move, the system is translated and rotated so that the larger molecule is back in its original position and orientation. So, the positions of the atom-spheres of the larger molecule are readily available. The situation is more complex with the smaller molecule; the positions of the enclosing spheres and the atom-spheres must be computed by transforming from the original positions. Because not all spheres are tested, it does not make sense to transform all of them, so only those that are being tested should be transformed. So, the first time a sphere is tested, its transformed position is computed and stored, since the sphere may be tested several times. When the transformed position is stored, a flag is set to indicate that this has happened and to keep the work from being duplicated later. After the collision testing is complete, the algorithm recursively descends from the top sphere and clears the flags for the next test. In using multiple threads, then, the flags and the transformed positions of the spheres of the smaller molecule must be duplicated for each thread, while all of the threads can use the same copy of the positions of the larger molecule.

The short-ranged Lennard–Jones forces are computed using the same idea. An effective radius is computed for each sphere, with the edge of the sphere marking the point where the potential energy is negligible. During the downward recursion, all in-range interactions are computed and added to the total force and torque. The main difference between this and the collision test is that all in-range pairs are tested and processed, rather than stopping once a collision is found.

A related algorithm, which is used in several places in Browndye, is the determination of the nearest atom-sphere to a given point outside of the sphere collection. A similar class of algorithms has been used for ray-tracing applications in computer graphics [41]. First, the distance between the point and the surface of the top sphere is computed, and the top sphere is put on a priority queue. This priority queue is ordered by distance from the point, so the item that is pulled off the queue is the closest sphere to the point. The algorithm proceeds by pulling a sphere off the queue, and checking to see if the nearest possible atom-sphere is greater than an upper bound. If it is, then the sphere is discarded; otherwise, its two child spheres are enqueued. If the distance from the test point to the furthest point in the dequeued sphere is less than the upper bound, the upper bound is set to that distance. This prevents the need to check spheres that are known with certainty not to contain a nearest atom-sphere. Spheres are pulled off the queue until one atom-sphere remains that satisfies the upper bound. Because the priority queue is implemented using the heap data structure, adding and pulling off items can be done very efficiently.

Some computer time can be saved by eliminating ahead of time those spheres that will not participate in a collision. First, a set of surface spheres are defined by rolling a probe ball across the surface using the algorithm of Connolly [42,43]. The neighbors of a surface sphere are those which are touched at the same time as itself by the probe sphere, and the neighboring sphere centers define a closed mesh of triangles. All of the spheres that have not been touched by the probe sphere are tested to see if their centers are inside the mesh or any of the surface spheres. This is done by picking a point well outside the mesh, and computing the number of intersections between a line connecting the two points and the mesh triangles (more details below). Those that are inside are discarded, and the ones not inside are called "danglers". The outside spheres are removed and saved elsewhere, while the probe sphere is rolled across the set of danglers. Since the set of danglers is usually divided into isolated clusters and the probe sphere can roll across only one cluster, the process is repeated until the clusters have been processed and divided into inside and outside spheres, and no more spheres remain.

Once the closed meshes of triangles and set of dangling outside spheres are generated, the grid of interior points is generated. A point is considered to be "inside" if it is overlapped by one of the spheres expanded by an additional exclusion distance, or if it is inside one of the closed triangular meshes generated above. Because there is a large number of grid points and often a large number of spheres and triangles, it is important to have an efficient algorithm that avoids testing every pair of grid point and sphere or triangle. The spheres are grouped into a tree-of-spheres data structure as used for collision detection above, and the triangles are likewise grouped into such a data structure. In order to test a point, the nearest sphere is found; if it overlaps, then we know that it is inside. Otherwise, another point is selected which is distant enough from the grid center to guarantee that it is outside the molecule.

We then have to determine how many triangles intersect the line segment connecting the test point with the outer point. If the number of intersections is odd, then the test point is inside; otherwise, it is outside. This algorithm proceeds recursively by finding the distance from the line segment to the outermost sphere enclosing the triangles. If the line segment does not intersect the sphere, then there are no intersections. Otherwise, the two child spheres are likewise tested, with no further work required for any non-intersecting spheres. Eventually, individual triangles are tested, and the total number of intersections are added up. This method eliminates the need to test most of the triangles. In the rare case that a point of intersection is on an edge between two triangles, the outside point is perturbed very slightly and another attempt is made.

Another gain in efficiency can be made by using the fact that if an outside point is a more than certain distance from the surface of the nearest sphere, then all other points within that distance of the first point are also outside. Conversely, if an interior point is known to be a certain distance from the surface of the nearest triangle, all other points within that distance of the first point are also inside. So, rather than stepping through the grid points sequentially, a self-avoiding sequence of grid points is generated by incrementing an integer index, reversing the bits, and using every third bit, starting with the first bit, to generate the index in the x-direction. The y-index and z-index are likewise generated by starting at the second and third bits. If the status (inside or outside) of the selected grid point is unknown, it is tested, and the distance to the nearest sphere or triangle is found. Then, all the points within that distance are set to have the same status. In this way, large chunks of points are determined without having to perform a test for each one.

Many of the existing packages for BD use the Northrup–Allison–McCammon (NAM) algorithm [44]. This algorithm starts the smaller molecule on the b-radius described above, and steps the system forward until a final reaction has been reached, or a larger q-radius is reached. In the traditional algorithm, the trajectory is terminated upon reaching the q-radius, and the q-radius must be considerably larger, in many cases, than the b-radius. Also, the required q-radius is difficult to compute, leading most users to use a large value in order to be safe. A newer version by Luty, McCammon, and Zhou (LMZ) [24] allows one to use a smaller q-radius, but then either places the smaller molecule back on the b-sphere according to an analytically computed probability distribution without yet terminating the trajectory, or assuming an escape and terminating the trajectory (Fig. 2). The original Luty–Zhou–McCammon algorithm does not take into account the change in orientation of the smaller molecule during its presumed journey from the q-sphere back to the b-sphere. The implementation in Browndye does take this into account, using an enhanced probability distribution that includes not only the new starting position but also the time taken to get there. Given a travel time, the algorithm then generates a change in orientation from a probability distribution, using a series formula that is precomputed and stored (unpublished results).

When using the weighted-ensemble method with the we_simulation program, the system copies (molecule pairs) are started with a separation of b, and the probabilities of reaction versus escape are computed from the relative probability fluxes. As the preprocessing step, the bins are generated by starting the copies with a separation of q, and running the copies until they react, setting up the bin partitions as described in Huber and Kim [25]. During the bin generation phase, the system copies "bounce back" if they go past a separation of q. Rather than using the simple bootstrap method as in the original paper, the confidence intervals on the probabilities are derived using the Stationary Bootstrap method for time series [45].

Although there are no restrictions on the size of the q-sphere in the LMZ algorithm, the b-radius must be large enough so that the force between the molecules depends only on the center-to-center distance of the molecules and not their orientation. Browndye

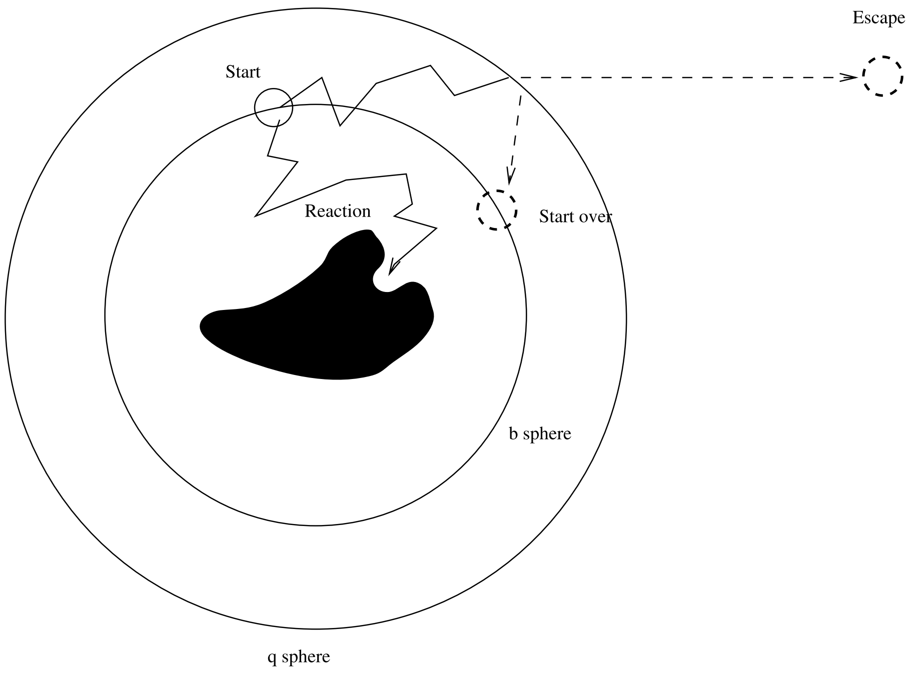

Fig. 2. Generating trajectories which react or escape.

can automatically determine the smallest radius that satisfies this requirement. One of the auxiliary programs first determines the smallest sphere about each molecule where the electric field is uniform, within a certain tolerance, on the surface of the sphere; these radii are r_{f1} and r_{f2}. Also, the maximum distance from the center among the point charges is computed for each molecule (r_{c1} and r_{c2}). The b-radius is taken to be

$$b = \max(r_{f1} + r_{c2}, r_{f2} + r_{c1}) \tag{10}$$

which ensures that each charge on one molecule interacts only with a radially symmetric field from the other molecule. Currently, the q-radius is taken to be 1.1 times the b-radius.

One important aspect of BD simulations is the selection of the time step size. Not only can the choice of time step size significantly affect performance, the computed reaction rates can be quite sensitive to the size if it is too large. Browndye attempts to determine an appropriate time step based on the geometry of the two molecules, and then corrects and refines the time step, if necessary, if the change in the forces and torques are too large. The following expression is used:

$$r_{mid} = \frac{1}{2}(q + b) \tag{11}$$

$$g_o = q - r + \mathcal{H}(r - r_{mid})(r - r_{mid}) \tag{12}$$

$$\mathcal{D} = (\mathbf{D}_{11} + \mathbf{D}_{22}) \cdot \frac{\mathbf{r}}{r} \tag{13}$$

$$\Delta t_o = \frac{\alpha_o g_o^2}{\mathcal{D}} \tag{14}$$

$$\Delta t_g = \frac{\alpha_g g^2}{\mathcal{D}} \tag{15}$$

$$\Delta t_r = \frac{\alpha_r \rho_{min}^2}{\mathcal{D}} \tag{16}$$

$$\mathcal{D}_r = \max(\mathcal{D}_{r1}, \mathcal{D}_{r2}) \tag{17}$$

$$\Delta t_{max} = \frac{\pi^2}{\mathcal{D}_r} \tag{18}$$

$$\Delta t_{min} = \frac{\Delta x^2}{2\mathcal{D}} \tag{19}$$

$$\Delta t_{r,min} = \frac{\Delta x_r^2}{2\mathcal{D}} \tag{20}$$

$$\Delta t = \min\bigl(\Delta t_{max}, \min\bigl(\max\bigl(\Delta t_{min}, \min(\Delta t_g, \Delta t_o)\bigr), \max(\Delta t_{r,min}, \Delta t_r)\bigr)\bigr) \tag{21}$$

where q and b are the q- and b-radii, r is the center-to-center distance, $\mathcal{H}$ is the Heaviside step function, α_o, α_g, and α_r are various tolerances, g is the effective gap, $\mathcal{D}$ is the relative diffusivity of the two molecules along the axis through both centers, $\mathcal{D}_{r1}$ and $\mathcal{D}_{r2}$ are the rotational diffusivities of the two molecules, ρ_{min} is the minimum of all reaction coordinates that are relevant for the current state, Δx_{min} is an average minimum distance per time step. The parameter $\Delta x_{r,min}$ is like Δx_{min} but can be made even smaller; this allows very small time steps near reactive configurations. The various time step components have physical interpretations. The time step Δt_o makes sure that too big of a time step is not taken near the q-sphere. The time step Δt_g ensures that small steps are taken when the molecules collide, and Δt_r ensures that small steps are taken near a reactive configuration. The time step Δt_{max} ensures that the rotations are small enough (see below), and the steps Δt_{min} and $\Delta t_{r,min}$ put limits on how small the time steps can become. The dimensionless α-parameters and the parameters Δx_{min} and $\Delta x_{r,min}$ are given reasonable defaults which can be overridden by the user.

In addition to using the geometrically determined time step Δt, Browndye bounds the time steps based on changes in $\mathcal{D}$ and in the force and torques between the molecules. Eq. (1) makes the assumption that these quantities do not change much over the course of a time step, so if that assumption is violated, the time step must be reduced. One simple scheme would be to reject the time step and just take another,smaller one. This, however, introduces a bias, so Browndye uses a more elaborate scheme. One can imagine a random walk with some vector W driven by the Gaussian variables w in Eq. (1):

$$\mathbf{W}_i = \sum_{j=1}^{i} \sqrt{2\,dt}\,\mathbf{w}_j \tag{22}$$

Suppose we took too large a time step. We would still want to keep the random walk up to this point to avoid biasing the results, but we would subdivide it by adding an extra W according to the formula

$$\mathbf{W}_{i-1/2} = \frac{1}{2}(\mathbf{W}_{i-1} + \mathbf{W}_i) + \sqrt{\frac{\Delta t}{2}}\mathbf{N} \tag{23}$$

where $\mathbf{N}$ is a vector of independent Gaussian variables with zero mean and unit variance. Then, the $\mathbf{w}$ variables from this refined random walk are

$$\mathbf{w}_{i-1/2} = \mathbf{W}_{i-1/2} - \mathbf{W}_{i-1} \tag{24}$$

$$\mathbf{w}_i = \mathbf{W}_i - \mathbf{W}_{i-1/2} \tag{25}$$

and both are used with half the original time step. If, upon using the reduced time step, the conditions are still violated, the algorithm backs up, and subdivides the random walk again. All values of $\mathbf{W}$ in the future that have been calculated from subdividing the random walk are used, and repeated subdivisions can lead to a chain of $\mathbf{W}$'s that must be followed. Eventually, the algorithm reaches the end of the chain of $\mathbf{W}$'s, and the values of $\mathbf{w}$ are generated from a Gaussian distribution again. The time step must be subdivided if any of the following inequalities is satisfied, for either molecule:

$$|\mathbf{F}_i - \mathbf{F}_{i-1}||\mathbf{x}_i - \mathbf{x}_{i-1}| > 0.02kT \tag{26}$$

$$|\mathbf{T}_i - \mathbf{T}_{i-1}||\phi| > 0.02kT \tag{27}$$

where ϕ is the rotation angle between the two successive orientations. In addition, the subdivision is triggered if there is a collision, or if this inequality is satisfied:

$$|\mathcal{D}_i - \mathcal{D}_{i-1}| > 0.02|\mathcal{D}_{i-1}| \tag{28}$$

One difficulty that is sometimes encountered with collisions between groups of hard spheres is that they can get stuck, and no reasonable number of small times steps can get them apart. Browndye detects this occurrence by keeping track of how many collisions have occurred since the last successful step. If a certain number (currently 10) of collisions occur in a row, then the two molecules are moved apart, along the line connecting their centers, by amount of overlap between the colliding atom-spheres. If there is still a collision, then the molecules are further moved apart in intervals of Δx_{min} until there is no more collision.

Another enhancement removes the need to take small steps near the absorbing q-sphere by using the method of Lamm and Schulten [46] to solve the unsteady-state diffusion equation near the boundary. Although the boundary is curved, the center of the smaller molecule is close enough so that the boundary can be considered flat, resulting in a one-dimensional diffusion equation. When the center of the smaller molecule is outside the b-sphere and inside the q-sphere, and is closer to the q-sphere, the molecules are stepped forward in time as usual. If their separation goes beyond the q-radius, then the smaller molecule either escapes or is put back onto the q-sphere, as described above. If not, then a survival probability is computed:

$$P_{sur} = 1 - \exp\left(-\frac{(q - r_{old})(q - r_{new})}{\mathcal{D}\,dt}\right) \tag{29}$$

where r_{old} and r_{new} are the old and new separations. A uniform random number between 0 and 1 is generated; if it falls above P_{sur}, the smaller molecule is assumed to be absorbed by the q-sphere and is processed as described above. This additional treatment takes into account the possibility that, during a large time step, the separation might become greater than q, even if the final separation is not. Without it, Browndye would have to take very small time steps in order not to underestimate the absorbtion at the q-sphere.

When using Eq. (1) to step the system forward, adding together the results of the changes in orientation (ϕ_1 and ϕ_2) is not as straightforward as it is for the positions. Numerically, it is most convenient to convert the incremental rotations, as well as the rotation matrix of the smaller molecule, and combine the rotations using quaternion multiplication. The resulting quaternion is then converted back to a rotation matrix. The time-stepping algorithm above includes a provision to limit the size of the rotations, since the rotational part of Eq. (1) is not valid for large changes in orientation.

5. Limitations

Currently, Browndye's model has many limitations. The molecules are treated as rigid bodies, ruling out realistic simulations of systems where molecular flexibility is significant for function. Inclusion of flexibility is currently in progress. The desolvation energy term is essentially an approximation of an approximation, and use must be made of the fudge factor in Eq. (6). The reaction criteria, especially the distances, can be used as another adjustable parameter, although one might conjecture that, given a good interaction model, the reaction rates would not be very sensitive to the reaction distances. There are no built-in hydrophobic forces, although one can approximate them using the Lennard–Jones forces. Nevertheless, Browndye is potentially useful for computing relative rates among different experimental conditions and mutants, and for exploring the trajectories to gain insight into receptor-ligand encounters. Future work on the model will include a better desolvation and effective charges model, flexibility and mobile side chains. Future work on the software itself will include modularization and documentation of the components to enable other developers to use the above algorithms in their own software.

6. Example

An example of a diffusion-limited, electrostatically-driven protein–protein interaction is the binding of thrombin and thrombomodulin, two proteins in the blood clotting cascade. After the PDB structures of the two molecules were suitably preprocessed (e.g., hydrogens added), the electric fields around each were generated using APBS, assuming salt concentration of 0.15 mM. The pairs of spheres defining the reaction were taken from possible hydrogen-bonding pairs. Using a criterion of three required pairs at a separation of 4.45 Å, and a million trajectories, a rate of $7.0 \times 10^6 \pm 0.5 \times 10^6\ \mathrm{M}^{-1}\,\mathrm{s}^{-1}$ was obtained, closely matching the experimental rate constant of 6.7×10^6 [47]. Using the same conditions but changing the salt concentration to 0.1 mM gave the rate constant $12.0 \times 10^6 \pm 0.8 \times 10^6\ \mathrm{M}^{-1}\,\mathrm{s}^{-1}$, which is within the experimental error bars of the experimental value of $15.1 \times 10^6\ \mathrm{M}^{-1}\,\mathrm{s}^{-1}$. Each run, which took place on 4 Intel Xeon processors, took about a day. (This is the system used in the example used in the software distribution, except that the reaction criteria in the example are changed to unrealistically large values in order to allow the user to compute "rates" after just a few minutes.)

7. Availability

The Browndye software package is freely available at the website browndye.ucsd.edu. The package and its components are distributed under an MIT-style license, and there are no components that are restricted by a GNU license. It runs under Linux, and can run on both 32-bit and 64-bit processors.

Acknowledgements

We are grateful to Maciej Dlugosz for patiently enduring a first draft of this software and providing valuable feedback. We also thank Barry Grant for being the first user of many of Browndye's features and providing more valuable feedback. We thank Adam van Wynsberghe for providing the preprocessed thrombin and thrombomodulin PDB files. This work has been supported by the National Institutes of Health, National Science Foundation, the Howard Hughes Medical Institute, the Center for Theoretical Biological Physics at UCSD, and the National Biomedical Computational Resource.

Appendix A. Charge lumping

A useful method for representing a function of one variable on the interval $[-1:1]$ is Chebyshev approximation:

$$f(x) \approx \sum_{i=0}^{N-1} c_i T_i(x) \tag{30}$$

with coefficients

$$c_i = \left(1 - \frac{1}{2}\delta_{0i}\right) \sum_{m=1}^{N} T_i(x_m) f(x_m) \tag{31}$$

where T_i is the ith Chebyshev polynomial and the x_i are the N zeros of polynomial T_N, and δ_{0i} is the Kronecker delta. This is a useful approximation, because it tends to spread the error out over the interval.

The same method can be extended to three dimensions, where x, y, and z all range from -1 to 1:

$$f(x, y, z) \approx \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} c_{ijk} T_i(x) T_j(x) T_k(x) \tag{32}$$

with coefficients

$$c_{ijk} = \frac{8}{N^3}\left(1 - \frac{1}{2}\delta_{0i}\right)\left(1 - \frac{1}{2}\delta_{0j}\right)\left(1 - \frac{1}{2}\delta_{0k}\right) \times \sum_{m=1}^{N} \sum_{n=1}^{N} \sum_{p=1}^{N} T_i(x_m) T_i(y_n) T_i(z_p) f(x_m, y_n, z_p) \tag{33}$$

For example, if one wanted to approximate a function to fourth order, corresponding to $N = 4$, one would evaluate the function at the 64 points of the Cartesian product of the zeros of T_4 and use that information to compute the coefficients.

Browndye approximates the electric potential using a 3-D Chebyshev interpolation, but using a different order of calculation and storage. Given a collection of point charges, a box is constructed about it. We scale and shift the positions so that the x, y, and z coordinates of the box range from -1 to 1; the 'hat' on the positions, force, and torque denote the scaled quantities. Browndye uses $N = 4$ as described above, resulting in the 64 points. If there are more than 64 point charges inside the box, we would like to evaluate the electric potential (in this case, interpolating from an APBS grid) at just the 64 points and obtain the total force and torque on the collection of charges. This, of course, can only be done if the potential field is smooth enough in that region to be accurately represented by the Chebyshev approximation.

Given an approximate potential V, the potential energy due to N_c charges is

$$E = \sum_{s=1}^{N_c} q_s V(\hat{\mathbf{x}}_s) \tag{34}$$

where q_s and $\hat{\mathbf{x}}_s$ are the sth charge and position. Likewise, the force and torque are

$$\hat{\mathbf{F}} = -\sum_{s=1}^{N_c} q_s \nabla V(\hat{\mathbf{x}}_s) \tag{35}$$

$$\hat{\mathbf{T}} = -\sum_{s=1}^{N_c} q_s (\hat{\mathbf{x}}_s - \hat{\mathbf{X}}_0) \times \nabla V(\hat{\mathbf{x}}_s) \tag{36}$$

where $\hat{\mathbf{X}}_0$ is the point about which the torque is measured. From the preceding three equations, it can be shown that a linear relation exists between the potential values at the quadrature points on one hand, and the energy, force, and torque on the other:

$$E = \hat{\mathbf{K}}_E \cdot \mathbf{V} \tag{37}$$

$$\hat{\mathbf{F}} = \hat{\mathbf{K}}_F \cdot \mathbf{V} \tag{38}$$

$$\hat{\mathbf{M}} = \hat{\mathbf{K}}_M \cdot \mathbf{V} \tag{39}$$

where $\mathbf{V}$ is the vector of potential values, $\hat{\mathbf{K}}_E$ is a vector of length N^3, and $\hat{\mathbf{K}}_F$ and $\hat{\mathbf{K}}_M$ are N^3 by 3 matrices. We define the coefficients

$$c_{mnp,ijk} = \frac{8}{N^3}\left(1 - \frac{1}{2}\delta_{0i}\right)\left(1 - \frac{1}{2}\delta_{0j}\right)\left(1 - \frac{1}{2}\delta_{0k}\right) \times T_i(\hat{x}_m) T_j(\hat{y}_n) T_k(\hat{z}_p) \tag{40}$$

and the Green's functions

$$V_{mnp}(\hat{x}, \hat{y}, \hat{z}) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} T_i(\hat{x}) T_j(\hat{y}) T_k(\hat{z}) \tag{41}$$

The Green's function indexed by *mnp* is the result of plugging a one into the potential value at the quadrature point indexed by *mnp* and zero into the all the other values. The energy vector is then

$$\hat{K}^E_{mnp} = \sum_{s=1}^{N_c} q_s V_{mnp}(\hat{\mathbf{x}}) \tag{42}$$

where the quadrature points are indexes using *mnp*. The force and torque matrices are

$$\hat{K}^F_{r,mnp} = -\sum_{s=1}^{N_c} q_s \frac{\partial V_{mnp}(\hat{\mathbf{x}}_s)}{\partial \hat{x}_r} \tag{43}$$

$$\hat{K}^M_{r,mnp} = \sum_{s=1}^{N_c} q_s \sum_{i=1}^{3} \sum_{j=1}^{3} \epsilon_{rij} \frac{\partial V_{mnp}(\hat{\mathbf{x}}_s)}{\partial \hat{x}_i} (\hat{x}_{s,j} - \hat{X}_{0,j}) \tag{44}$$

where ϵ_{rij} is the Levi-Civita symbol. These matrices can be constructed before the simulation.

The transformation from the scaled coordinates to the actual coordinates is

$$x = \frac{1}{2}(H_x + L_x) + \frac{1}{2}(H_x - L_x)\hat{x} \tag{45}$$

$$y = \frac{1}{2}(H_y + L_y) + \frac{1}{2}(H_y - L_y)\hat{y} \tag{46}$$

$$z = \frac{1}{2}(H_z + L_z) + \frac{1}{2}(H_z - L_z)\hat{z} \tag{47}$$

where L and H are the lower and higher bounds on the actual box. During the preprocessing stage, the coordinates of the charges are transformed into the $[-1:1]$ range using the inverse of the above transform, and the matrices of Eqs. (43) and (44) are computed. The scaled matrix

$$\mathbf{K}_F = \mathbf{J}^{-1} \cdot \hat{\mathbf{K}}^F \tag{48}$$

is computed, where $\mathbf{J}$ is the Jacobian of the transformation. Due to transformation properties of the torque, the torque matrix is unchanged by the transformation, as is the energy vector:

$$\mathbf{K}^E = \hat{\mathbf{K}}^E \tag{49}$$

$$\mathbf{K}^M = \hat{\mathbf{K}}^M \tag{50}$$

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, Molecular Biology of the Cell, 4th edition, Garland Science, New York, 2002.

[2] T. Pollard, W. Earnshaw, Cell Biology, W.B. Saunders, New York, 2007.

[3] A. Elcock, Molecular simulations of diffusion and association in multimacromolecular systems, in: Numerical Computer Methods, Part D, in: Methods in Enzymology, vol. 383, Academic Press Inc, 525 B Street, Suite 1900, San Diego, CA 92101-4495, USA, 2004, pp. 166–198.

[4] D. Ermak, J. McCammon, Brownian dynamics with hydrodynamic interactions, J. Chem. Phys. 69 (4) (1978) 1352–1360.

[5] R. Jendrejack, M. Graham, J. de Pablo, Hydrodynamic interactions in long chain polymers: Application of the Chebyshev polynomial approximation in stochastic simulations, J. Chem. Phys. 113 (7) (2000) 2894–2900.

[6] T. Shen, C. Wong, J. McCammon, Atomistic Brownian dynamics simulation of peptide phosphorylation, J. Am. Chem. Soc. 123 (37) (2001) 9107–9111.

[7] S. Allison, J. McCammon, Dynamics of substrate binding to copper–zinc superoxide-dismutase, J. Phys. Chem. 89 (7) (1985) 1072–1074.

[8] J. Madura, J. Briggs, R. Wade, M. Davis, B. Luty, A. Ilin, J. Antosiewicz, M. Gilson, B. Bagheri, L. Scott, J. McCammon, Electrostatics and diffusion of molecules in solution – simulations with the University-of-Houston Brownian dynamics program, Comput. Phys. Commun. 91 (1–3) (1995) 57–95.

[9] R. Wade, B. Luty, E. Demchuk, J. Madura, M. Davis, J. Briggs, J. McCammon, Simulation of enzyme-substrate encounder with gated active sites, Nat. Struct. Biol. 1 (1) (1994) 65–69.

[10] C. Chang, T. Shen, J. Trylska, V. Tozzini, J. McCammon, Gated binding of ligands to HIV-1 protease: Brownian dynamics simulations in a coarse-grained model, Biophys. J. 90 (11) (2006) 3880–3885, doi:10.1529/biophysj.105.074575.

[11] R. Gabdoulline, R. Wade, Simulation of the diffusional association of Barnase and Barstar, Biophys. J. 72 (5) (1997) 1917–1929.

[12] A. Elcock, D. Sept, J. McCammon, Computer simulation of protein–protein interactions, J. Phys. Chem. B 105 (8) (2001) 1504–1518, doi:10.1021/jp003602d.

[13] S. Northrup, T. Laughner, G. Stevenson, Macrodox macromolecular simulation program, 1997.

[14] B. Honig, A. Nicholls, Classical electrostatics in biology and chemistry, Science 268 (5214) (1995) 1144–1149.

[15] N. Baker, D. Sept, S. Joseph, M. Holst, J. McCammon, Electrostatics of nanosystems: Application to microtubules and the ribosome, Proc. Natl. Acad. Sci. USA 98 (18) (2001) 10037–10041.

[16] M. Fixman, Simulation of polymer dynamics 1. General theory, J. Chem. Phys. 69 (4) (1978) 1527–1537.

[17] C.W. Gardiner, Handbook of Stochastic, 2nd edition, Methods, Springer, Berlin, 1985.

[18] E. Dickinson, S. Allison, J. McCammon, Brownian dynamics with rotation translation coupling, J. Chem. Soc. Faraday Trans. II 81 (4) (1985) 591–601.

[19] S. Allison, A Brownian dynamics algorithm for arbitrary rigid bodies – application to polarized dynamic light-scattering, Macromolecules 24 (2) (1991) 530–536.

[20] S. Kim, S. Karrila, Microhydrodynamics: Principles and Selected Applications, Butterworth-Heinemann, Boston, 1991.

[21] M. Fernandes, J. Garcia de la Torre, Brownian dynamics simulation of rigid particles of arbitrary shape in external fields, Biophys. J. 83 (6) (2002) 3039–3048.

[22] D. Beard, T. Schlick, Unbiased rotational moves for rigid-body dynamics, Biophys. J. 85 (5) (2003) 2973–2976.

[23] M. Levitt, Simplified representation of protein conformations for rapid simulation of protein folding, J. Mol. Biol. 104 (1) (1976) 59–107.

[24] B. Luty, J. McCammon, H. Zhou, Diffusive reaction-rates from Brownian dynamics simulations – replacing the outer cutoff surface by an analytical treatment, J. Chem. Phys. 97 (8) (1992) 5682–5686.

[25] G. Huber, S. Kim, Weighted-ensemble Brownian dynamics simulations for protein association reactions, Biophys. J. 70 (1) (1996) 97–110.

[26] Opendx, http://www.opendx.org, 2006.

[27] C. Cooper, Using expat, http://www.libexpat.org/, 1999.

[28] X. Leroy, D. Doligez, J. Garrigue, D. Rémy, J. Vouillon, The objective Caml system: release 3.11, http://caml.inria.fr/pub/docs/misc-ocaml/index.html, 2008.

[29] I. Free Software Foundation, GNU make manual, http://www.gnu.org/software/make/, 2006.

[30] B. Barney, Posix threads programming, https://computing.llnl.gov/tutorials/pthreads/, 2009.

[31] W. Humphrey, A. Dalke, K. Schulten, VMD – visual molecular dynamics, J. Mol. Graph. 14 (1996) 33–38.

[32] H. Ding, N. Karasawa, W. Goddard, Atomic level simulations on a million particles – the cell multipole method for Coulomb and London nonbond interactions, J. Chem. Phys. 97 (6) (1992) 4309–4315.

[33] R. Gabdoulline, R. Wade, Effective charges for macromolecules in solvent, J. Phys. Chem. 100 (9) (1996) 3868–3878.

[34] W. Still, A. Tempczyk, R. Hawley, T. Hendrickson, Semianalytical treatment of solvation for molecular mechanics and dynamics, J. Am. Chem. Soc. 112 (16) (1990) 6127–6129.

[35] P.B. Visscher, D.M. Apalkov, Simple recursive implementation of fast multipole method, J. Magn. Magn. Mater. 322 (2) (2010) 275–281, doi:10.1016/j.jmmm.2009.09.033.

[36] J. Rotne, S. Prager, Variational treatment of hydrodynamic interaction in polymers, J. Chem. Phys. 50 (11) (1969) 4831.

[37] J. Garcia de la Torre, V. Bloomfield, Hydrodynamic properties of macromolecular complexes. 1. Translation, Biopolymers 16 (8) (1977) 1747–1763.

[38] S. Hansen, Translational friction coefficients for cylinders of arbitrary axial ratios estimated by Monte Carlo simulation, J. Chem. Phys. 121 (18) (2004) 9111–9115, doi:10.1063/1.1803533.

[39] J. Klosowski, M. Held, J. Mitchell, H. Sowizral, K. Zikan, Efficient collision detection using bounding volume hierarchies of k-DOPS, IEEE Trans. Vis. Comput. Graph. 4 (1) (1998) 21–36.

[40] E. Welzl, Smallest enclosing disks (balls and ellipsoids), Lect. Notes Comput. Sci. 555 (1991) 359–370.

[41] J. Goldsmith, J. Salmon, Automatic creation of object hierarchies for ray tracing, IEEE Comput. Graph. Appl. 7 (5) (1987) 14–20.

[42] M. Connolly, Analytical molecular-surface calculation, J. Appl. Crystallogr. 16 (10) (1983) 548–558.

[43] M. Sanner, A. Olson, J. Spehner, Reduced surface: An efficient way to compute molecular surfaces, Biopolymers 38 (3) (1996) 305–320.

[44] S. Northrup, S. Allison, J. McCammon, Brownian dynamics simulation of diffusion-influenced bimolecular reactions, J. Chem. Phys. 80 (4) (1984) 1517–1526.

[45] D. Politis, J. Romano, The stationary bootstrap, J. Am. Stat. Assoc. 89 (428) (1994) 1303–1313.

[46] G. Lamm, K. Schulten, Extended Brownian dynamics approach to diffusion-controlled processes, J. Chem. Phys. 75 (1) (1981) 365–371.

[47] A. Baerga-Ortiz, A. Rezaie, E. Komives, Electrostatic dependence of the thrombin–thrombomodulin interaction, J. Mol. Biol. 296 (2) (2000) 651–658.