Algorithm for rigid-body Brownian dynamics

Dan Gordon,* Matthew Hoyles, and Shin-Ho Chung†
Computational Biophysics Group, Research School of Biology, The Australian National University, Acton, ACT 0200, Australia
(Received 30 July 2009; published 4 December 2009)

We present an algorithm for performing rigid-body Brownian dynamics that can take into account the hydrodynamic properties (translational and rotational friction tensors and the coupling between them) of each rigid body. In the zero temperature limit, the error term scales as Δ^4 for time step Δ, while at nonzero temperatures the error scaling is $\Delta^{5/2}$. We test the algorithm by applying it to a molecule of four-aminopyridine in water. We intend to use the algorithm to model the interaction between biological ion channels and other channel blocker molecules, but it may also have applicability to modeling other small particles such as colloids or nanoparticles.

DOI: 10.1103/PhysRevE.80.066703 PACS number(s): 02.60.−x, 31.15.xv, 05.40.Jc, 87.15.km

I. INTRODUCTION

In this paper, we present an algorithm for performing rigid-body Brownian dynamics. No spatial symmetries are assumed in either the rotational or frictional specification of the body. In addition, hydrodynamic terms that link the six rotational and translational degrees of freedom of each body may be taken into account through the appropriate specification of the friction tensor for that body.

In the zero-temperature limit, the algorithm shows the same $O(\Delta^4)$ error propagation and time reversibility as commonly used algorithms such as the velocity Verlet algorithm [1], to which our algorithm is closely related. At nonzero temperature, the addition to the algorithm of stochastic terms increases the local error at each step, which now scales as $O(\Delta^{5/2})$ [in Appendix B, we show how an error $O(\Delta^{7/2})$ algorithm could be developed]. This still compares favorably with the $O(\Delta^{3/2})$ scaling of the commonly used Euler algorithm [2]. Note that such stochastic error terms might be expected to exhibit better weak error scaling (i.e. scaling of the probability distribution as a function of time) than they do strong scaling (i.e. scaling of individual paths for a given realization of Brownian motion, as treated here) [3]. See also Greiner *et al.* [4] for a detailed treatment of error scaling.

We demonstrate the viability of the algorithm as well as the error scaling by simulating the motion of a small ion-channel-blocker molecule [four-aminopyridine (4-AP)] under various test conditions.

Fernandes and de la Torre [5] have previously published a rigid-body Brownian dynamics algorithm that works in the diffusive limit, where the momentum terms are neglected. This algorithm was later improved by Beard and Schlick [6], who showed how biases introduced in the rotational motion could be removed. In the diffusive limit, the bodies undergo a directed random walk in which the position at the next step in the algorithm is independent of velocity. This limit is valid on timescales that are longer than those of the velocity autocorrelation functions; typically between 10 and 100 fs in the kinds of molecular-scale systems that we are interested in here. Although these timescales are appropriate for many types of Brownian-dynamics simulations, if the forces in the system fluctuate on shorter time scales, then it may be useful to go beyond the diffusive limit [7]. For example, in our experience with modeling biological ion channels, we have found that employing Langevin dynamics, rather than diffusive Brownian dynamics, can lead to a more realistic simulation within the confined spaces of the ion-channel vestibules and pores [8]. (In fact, there is some suggestion [9,10] that, in cases involving narrow barrier crossings, memory effects may play a role, and that therefore it might be appropriate to go further and use the generalized Langevin equation). This use of shorter timescale Langevin dynamics in ion channels and other similar cases can also help to avoid the problems caused by unphysical collisions, or by the system moving into regions of extremely high potential due to a large time step. Therefore, it also seems desirable to have available rigid-body Langevin algorithms that go beyond the diffusive limit.

Recently, Sun *et al.* [11] have presented such a Langevin dynamics algorithm for rigid bodies that makes use of the operator splitting techniques used in the nonstochastic algorithm of Dullweber *et al.* [12] The nondissipative parts of the motion are treated in a time-symmetric manner that acts to reduce the error by an order, as per the velocity Verlet algorithm. The treatment of the rotation automatically preserves the determinant of the rotation matrix, which may be advantageous [12]. As well as employing a somewhat different mathematical framework, our algorithm employs an implicit, iterative scheme, under which the frictional terms gain an extra order of accuracy. Likewise, we consider higher order terms in the random forces.

Our work is closely related to the rotational algorithm of Omelyan [13], but with the addition of Brownian motion. In the frictionless limit, the algorithm essentially reduces to the velocity-Verlet equivalent of Omelyan's rotational leapfrog algorithm, plus the usual velocity Verlet algorithm for the translational motion. Adding frictional and random forces complicates the analysis, firstly by the fact that additional frictional and stochastic terms must be considered, and secondly by the fact that hydrodynamic interactions can couple translational and rotational degrees of freedom.

Also relevant is the algorithm of Ermak and McCammon [14], who have considered an arbitrary collection of interact-

*Corresponding author; dan.gordon@anu.edu.au
†http://langevin.anu.edu.au/

ing spherical particles, with a hydrodynamic coupling applied between the translational coordinates of the particles. Typically, composite bodies are modeled as a collection of spheres held together by a potential or using a constraint algorithm. This algorithm has been used to simulate ion-channel-blocker systems similar to the ones we are interested [15]. Dickinson *et al.* [16] have extended the theory to include the rotational motion of the spherically symmetric particles. Our algorithm, in contrast, models rigid bodies as monolithic objects having three translational and three rotational degrees of freedom. The hydrodynamic frictional and random tensors in our model couple together these six degrees of freedom for each rigid body; we do not consider the hydrodynamic coupling between two or more rigid bodies [17]. Of course, the bodies will typically be coupled together using an ordinary potential. These differences, between our model and the interacting spherical particle hydrodynamics models [14,16], mean that they will have somewhat different domains of applicability. Our model is less general, in that it would be poorly suited to modeling highly flexible molecules or tight complexes of molecules, or to predicting the hydrodynamic properties of composite bodies based on atom-scale parameters. It is, however, very well suited to modeling the motion of possibly large, quite rigid molecules at concentrations that are not too great, where the hydrodynamic properties of the molecule as a whole (coupled rigid-body hydrodynamic friction) are known or calculated in advance. This very well suits our aim, which is to model the interactions between channel blockers and biological ion channels.

Finally, our algorithm stands in contrast to the constraint algorithms commonly used in molecular dynamics [18], which enforce bond-length and bond-angle constraints through an iterative procedure. While this is most appropriate for treating small molecules (such as water) in molecular dynamics, we choose to use a geometric treatment of the rigid-body motion, firstly to avoid complications brought about by the use of internal coordinates and the need to avoid performing multiple redundant calculations, and secondly to provide a more direct link to rigid-body hydrodynamic friction tensors.

While we have developed our algorithm with a view to modeling the interactions between biological ion channels and "channel blocker" molecules, it is possible that it could be applied to modeling other small particles in a fluid, such as colloidal particles, liquid crystals or nanotechnology, although this would require demonstration.

The geometric algebra paradigm (see Appendix A) is used throughout the derivations, providing a comprehensive and appropriate framework within which to work. Much of the following notation and discussion makes use of material in the book "Geometric Algebra for Physicists" by Doran and Lasenby [19], to which the reader is referred.

II. SPATIAL FRAMEWORK FOR RIGID-BODY COMPUTATIONAL DYNAMICS

In the following section, we derive an equation of motion for a rigid body moving under the influence of external forces. In order to fully orient a rigid body in space, we need to specify the location of a point in the body (say, the center of mass), and the rotational orientation R of the body relative to this point. We define a "body reference frame" as the frame whose origin is the center of mass of the body, and whose axes are fixed relative to the body. We will generally prefer to use this body reference frame, so points in the laboratory frame will be distinguished by the addition of an apostrophe, e.g. $\mathbf{y}'$. The body's location and orientation in the laboratory frame are then specified by the center of mass, $\mathbf{x}'$, and the rotor R (see Appendix A) that rotates the body from the body reference to the laboratory frame. If $\mathbf{y}'$ is a point in the body in the body reference frame, then

$$\mathbf{y}' = R\mathbf{y}R^{\dagger} + \mathbf{x}',$$

$$\mathbf{y} = R^{\dagger}(\mathbf{y}' - \mathbf{x}')R.$$

In this scheme, there are seven parameters and six degrees of freedom. There are three parameters for the center of mass of the body, and four for the rotation R. There are only three degrees of freedom for the four parameter object R because of the normalization condition on R.

A. Rotational dynamics

Our starting point for deriving the rotational dynamics is the *rotor equation*, which relates the rate of change of R to the instantaneous angular velocity Ω'. In the context of geometric algebra, angular velocity is usually viewed as a bivector, related to the familiar (axial vector) angular velocity ω by $\Omega = I\omega$ [20].

The rotor equation, which can be derived by taking the time derivative of the equation for the rotation of an arbitrary vector $(\mathbf{y}(t) = R\mathbf{y}(0)R^{\dagger})$, is

$$\dot{R} = -\frac{1}{2}\Omega' R, \tag{1}$$

or, rotating Ω' to the body reference frame, we have

$$\dot{R} = -\frac{1}{2}R\Omega. \tag{2}$$

Note that Ω is some function of t, which needs to be specified in order to derive any dynamics. The rotor equation is analogous to the equation $\dot{\mathbf{x}} = \mathbf{v}$, but differs in that the derivative $\dot{R}$ explicitly depends on the position variable R as well as the velocity variable Ω.

From basic mechanics, we know that the rate of change of angular momentum is equal to the torque, analogous to $\mathbf{f} = \dot{\mathbf{v}}$. In the language of geometric algebra, we have

$$N' = \dot{L}'. \tag{3}$$

where $N'(\mathbf{x}', R)$ is a bivector representing the torque about the instantaneous center of mass $\mathbf{x}'$ as a function of the position and orientation of the body, and L' is a bivector representing the angular momentum of the body about the instantaneous center of mass. If the constituent particles of the body have coordinates $\mathbf{y}_i'$, with corresponding body refer-

ence coordinates $\mathbf{y}_i$ and corresponding masses m_i, then

$$\begin{aligned} L' &= \sum m_i(\mathbf{y}_i' - \mathbf{x}') \wedge \dot{\mathbf{y}}_i' \\ &= \sum m_i(\mathbf{y}_i' - \mathbf{x}') \wedge \{(\mathbf{y}_i' - \mathbf{x}') \cdot \Omega' + \dot{\mathbf{x}}'\} \\ &= R\Big[\sum m_i \mathbf{y}_i \wedge (\mathbf{y}_i \cdot \Omega)\Big]R^\dagger. \end{aligned} \tag{4}$$

In deriving the final equality, the term involving $\mathbf{x}'$ disappears because it is proportional to the center of mass in the body reference frame, which is the origin. The term in brackets is a bivector that depends only on Ω and on the $\mathbf{y}_i$s, which are fixed. It can, therefore, be represented as a constant tensor i.e. a linear map from bivectors to bivectors:

$$L = \mathcal{I}(\Omega). \tag{5}$$

$\mathcal{I}(\Omega)$ gives the angular momentum for a given angular velocity of the reference body.

If the reference body is chosen so that its principal axes correspond to the basis vectors $\mathbf{e}_i$, then $\mathcal{I}$ will be diagonal; that is, its basis expansion will be of the form

$$\mathcal{I}(I\mathbf{e}_i) = \mathcal{I}_i I\mathbf{e}_i, \tag{6}$$

and we can define an inverse $\mathcal{I}^{-1}$ as

$$\mathcal{I}^{-1}(I\mathbf{e}_i) = \frac{1}{\mathcal{I}_i} I\mathbf{e}_i. \tag{7}$$

From Eqs. (3) and (5) we therefore have

$$\begin{aligned} N'(\mathbf{x}', R) &= \dot{L}' \\ &= R[\mathcal{I}(\dot{\Omega})R^\dagger + \dot{R}\mathcal{I}(\Omega)R^\dagger + R\mathcal{I}(\Omega)]\dot{R}^\dagger \\ &= R\left\{\mathcal{I}(\dot{\Omega}) - \frac{1}{2}[\Omega, \mathcal{I}(\Omega)]\right\}R^\dagger, \end{aligned}$$

$$N(\mathbf{x}', R) = \mathcal{I}(\dot{\Omega}) - \frac{1}{2}[\Omega, \mathcal{I}(\Omega)], \tag{8}$$

where the third line follows from the rotor equation, Eq. (2), and we have introduced the commutator product $[A,B] := AB - BA$. Therefore,

$$\dot{\Omega} = \mathcal{I}^{-1}\left\{N(\mathbf{x}', R) + \frac{1}{2}[\Omega, \mathcal{I}(\Omega)]\right\}, \tag{9}$$

which defines the body reference angular acceleration $\dot{\Omega}$ as a function of R and Ω. The first term is a force divided by mass term, and the second is a geometric term, like the Coriolis force.

In components, the equations of motion can, therefore, be written as

$$\dot{\Omega}_n = a_n(\mathbf{x}', R) + b_{nml}\Omega_m\Omega_l, \tag{10}$$

$$\dot{R}_n = c_{nml}R_m\Omega_l, \tag{11}$$

where

$$a_n(\mathbf{x}', R) := \frac{1}{\mathcal{I}_n} N(\mathbf{x}', R), \tag{12}$$

$$b_{nml} := \frac{\mathcal{I}_m}{\mathcal{I}_n}\epsilon_{nml}, \tag{13}$$

$$c_{nml}R_m\Omega_l = -\frac{1}{2}(R\Omega)_n, \tag{14}$$

define the angular acceleration a_n and the tensors b_{nml} and c_{nml}.

B. Center of mass dynamics

It can be shown that the center of mass of the body obeys Newton's laws for a point particle:

$$\dot{\mathbf{v}}' = \mathbf{a}'(\mathbf{x}', R) = \frac{1}{m}\mathbf{f}'(\mathbf{x}', R), \tag{15}$$

$$\dot{\mathbf{x}}' = \mathbf{v}', \tag{16}$$

where m is the total mass of the body and $\mathbf{f}'(\mathbf{x}', R)$ is the total external force:

$$\mathbf{f}'(\mathbf{x}', R) = \sum_i \mathbf{f}_i'[\mathbf{y}_i'(\mathbf{x}', R)], \tag{17}$$

with $\mathbf{f}_i'$ being the external force on the ith particle as a function of its position $\mathbf{y}_i'$. Obviously, we do not consider forces between the individual particles in the body, firstly because the body is rigid, and secondly because such forces do not, in any case, affect the center of mass motion.

When we later come to add frictional and random forces to this equation, we will find it easiest to transform $\mathbf{v}'$ into the body reference frame. Recalling that $\mathbf{v} = R^\dagger \mathbf{v}' R$, and using the rotor Eq. (2), we find

$$\begin{aligned} \dot{\mathbf{v}} &= \dot{R}^\dagger \mathbf{v}' R + R^\dagger \dot{\mathbf{v}}' R + R^\dagger \mathbf{v}' \dot{R} \\ &= \frac{1}{2}\Omega\mathbf{v} + \mathbf{a}(\mathbf{x}', R) - \frac{1}{2}\mathbf{v}\Omega \\ &= \mathbf{a}(\mathbf{x}', R) + \frac{1}{2}[\Omega, \mathbf{v}], \end{aligned} \tag{18}$$

in complete analogy to Eq. (9).

The equation for $\dot{x}'$ is accomplished by a simple rotation:

$$\dot{\mathbf{x}}' = R\mathbf{v}R^\dagger. \tag{19}$$

In components, these equations are:

$$\dot{v}_i = a_i(\mathbf{x}', R) + b_{ijn}v_j\Omega_n, \tag{20}$$

$$\dot{x}_i' = d_{inmj}R_nR_mv_j, \tag{21}$$

where

$$b_{ijn} := \epsilon_{ijn}, \tag{22}$$

$$d_{inmj}R_nR_mv_j = (R\mathbf{v}R^\dagger)_i, \tag{23}$$

define the tensor b_{ijn} and the rotation tensor d_{inmj}.

III. RANDOM AND FRICTIONAL FORCES

The effects of a surrounding, implicit medium can be taken into account by introducing random and frictional

forces to Eqs. (16) and (9). Consider an irregularly shaped object. It will experience the usual translational frictional force as it moves through the medium, and rotational friction as it spins. In addition, translational motion can exert a torque, and conversely, rotation of the object can exert a translational force, as would be the case for a propeller-shaped object. Other coupling forces can be present, for example lift forces where a translational velocity in one direction can give rise to translational forces in a different direction. Finally, an angular velocity about one axis can give rise to a torque about a different axis. These forces can be encapsulated in a hydrodynamic frictional acceleration tensor, which resembles a tensor version of the usual scalar friction coefficient used for spherically symmetric particles, except that the six degrees of freedom may be coupled together. In order to develop a general algorithm, we will not make any assumptions about the frictional acceleration tensor.

At this point, the reason for writing the translational velocity in the reference frame of the object becomes apparent. For an irregularly shaped object moving in a given direction, the friction will depend on the orientation of the object relative to that direction. However, if we measure the friction in the reference frame of the body, the friction can no longer depend on the orientation, as the orientation of the reference object is fixed. This simplifies the analysis, but at the cost of adding additional geometric terms, namely, the commutator terms in Eqs. (9) and (18) that need to be added to the equations of motion to account for the continuous change of basis that occurs as the particles move.

We begin by writing combined generalized position and velocity variables:

$$X = (\mathbf{x}', R), \tag{24}$$

$$V = (\mathbf{v}, \Omega), . \tag{25}$$

A. Rotational stochastic equation (the Euler-Langevin equation)

The rotational equations Eqs. (10) and (11) can be modified by adding frictional and random torques. The equation we arrive at for Ω is essentially the Euler-Langevin equation, with some additional coupling to the translational degrees of freedom. In order to employ the tools of stochastic calculus [21], we work in differential form for the equations of motion.

$$d\Omega_n = \{a_n(X) + b_{nml}\Omega_m\Omega_l - \gamma_{n\alpha}V_\alpha\}dt + s_{n\alpha}dW_\alpha \tag{26}$$

where $\gamma_n\alpha$ is the frictional acceleration tensor, $s_{n\alpha}$ is the random acceleration tensor, and the $W_\alpha(t)$'s are uncorrelated Wiener processes i.e. $W_\alpha(0)=0$ and $[W_\alpha(t)-W_\alpha(t')]$ obeys the normal distribution with mean 0 and variance $(t-s)$.

The equation for R, Eq. (11), is still the unchanged rotor equation:

$$dR_n = c_{nml}R_m\Omega_l dt. \tag{27}$$

B. Center of mass stochastic equation (the Langevin equation)

To Eq. (20) we add frictional and random forces:

$$dv_i = \{a_i(X) + b_{ijn}v_j\Omega_n - \gamma_{i\alpha}V_\alpha\}dt + s_{i\alpha}dW_\alpha. \tag{28}$$

This is essentially the Langevin equation, but with coupling to the rotational degrees of freedom via the frictional and random forces. $\gamma_{i\alpha}$ is a frictional acceleration tensor. Note that the translational frictional force can depend on both the translational and rotational velocities. $s_{i\alpha}$ is a tensor that scales the strength of the random force. We shall see below that γ and s are related by the fluctuation-dissipation theorem.

The equation for $d\mathbf{x}'$, Eq. (21), is unchanged:

$$dx_i' = d_{inmj}R_nR_mv_j dt. \tag{29}$$

C. Combined equations

Comparing Eq. (26) with Eq. (28) and Eq. (27) with Eq. (29), we see that there are many similarities. In order to simplify the analysis, we can write combined equations of motion:

$$dV_\alpha = \{a_\alpha(X) + b_{\alpha\beta n}V_\beta V_n - \gamma_{\alpha\beta}V_\beta\}dt + s_{\alpha\beta}dW_\beta, \tag{30}$$

$$dX_\alpha = \{c_{\alpha nm}X_nV_m + d_{\alpha nmj}X_nX_mV_j\}dt. \tag{31}$$

Remember that the indices i,j,k range over the translational degrees of freedom, the indices n,m,l range over rotational degrees of freedom, and the indices α,β,γ range over both. The externally applied acceleration and angular acceleration is a, b is defined by Eqs. (13) and (22) for rotational and translational variables respectively, γ and s are the generalized friction and random force tensors, c is defined by Eq. (14), and d is defined by Eq. (23).

This is a convenient point, at which to introduce the frictional force tensor, ζ,

$$\zeta_{\alpha\beta} = M_\alpha\gamma_{\alpha\beta}, \tag{32}$$

and the random force tensor, u,

$$u_{\alpha\beta} = M_\alpha s_{\alpha\beta}. \tag{33}$$

M_α are the generalized mass/moment of inertia components, equal to the mass of the body for translational coordinates and to the moments of inertia $\mathcal{I}_\alpha$ for rotational coordinates. Note that, unlike the corresponding acceleration tensors γ and s, ζ and u are symmetric tensors.

D. Measuring the frictional force tensor

In theory, it should be possible to use simulation, such as molecular dynamics, to calculate the friction tensor [22]. However, in reality, the multiple degrees of freedom for molecules of arbitrary shape make this computationally prohibitive. Instead, hydrodynamic calculations may be employed. For example, a program, HYDROPRO [23], is available for free download [24]. It uses hydrodynamics to calculate the friction on a body specified that is specified by atomic coordinates. It outputs a diffusion tensor D, which can easily be

transformed into the frictional force tensor ζ by applying Einstein's relation,

$$\zeta_{\alpha\beta} = kT(D^{-1})_{\alpha\beta}. \tag{34}$$

We use results obtained from HYDROPRO, applied to the four-aminopyridine molecule, in our numerical example.

E. Fluctuation dissipation relations

The second fluctuation-dissipation [22] theorem allows us to relate the frictional force tensor ζ to the random force tensor u. We find

$$u_{\alpha\gamma}u_{\beta\gamma} = 2kT\zeta_{\alpha\beta}. \tag{35}$$

In practice, we will usually know the frictional force tensor ζ but not the random force tensor u. In order to obtain the latter, we need to numerically solve Eq. (35), which is a matrix equation of the form $AA^{\mathrm{T}}=B$. The solution is not unique, and a triangular matrix for u can be obtained by Cholesky decomposition [25].

IV. A STOCHASTIC ALGORITHM FOR RIGID-BODY BROWNIAN DYNAMICS

Below, we give a derivation of an algorithm for rigid-body Brownian dynamics. The algorithm is most closely related to the popular velocity Verlet algorithm [1]. Stochastic forces are included, and are treated using stochastic calculus. The inclusion of velocity-dependent frictional and rotational terms makes it necessary to solve an additional system of equations at each step, similar to Omelyan [13]. This is done relatively simply by a process of iteration.

In Appendix B, we give a sketch of a slightly more complicated algorithm, which provides a better error scaling for the stochastic terms. However, we have not implemented the more complicated algorithm in our numerical tests.

In the derivations below, we use the following notation in order to make our equations more compact: For a function $f(V(t),X(t),t)$ we write $[f]_{t'} := f(V(t'),X(t'),t')$ and $[f]_{t''}^{t'} := [f]_{t'} - [f]_{t''}$ (the latter form being the same notation often used in the evaluation of definite integrals).

A. Preliminary theory: Stratonovich stochastic Taylor series

Our algorithm will be developed by using Taylor expansions of the stochastic differential equations of motion Eqs. (30) and (31). As stochastic calculus is most easily presented in integral form, we use the Stratonovich Taylor expansion [26], which is summarized below in a form relevant to our algorithm.

Consider a system of equations of the form:

$$dV_\alpha = g_\alpha(V,X)dt + s_{\alpha\beta}dW_\beta, \tag{36}$$

$$dX = h_\alpha(V,X)dt, \tag{37}$$

where V and X are functions of t. We define operators L_1 and $L_{2\beta}$ such that

$$L_1 = g_\alpha(V,X)\frac{\partial}{\partial V_\alpha} + h_\alpha(V,X)\frac{\partial}{\partial X_\alpha}, \tag{38}$$

$$L_{2\beta} = s_{\alpha\beta}\frac{\partial}{\partial V_\alpha}. \tag{39}$$

Then, by applying the chain rule of calculus (which is also valid for the Stratonovich stochastic integral), we find that, for any function $F(V,X)$, we have

$$dF = L_1 F dt + L_{2\beta}F dW_\beta, \tag{40}$$

Or, in integral form (using the notation described above):

$$[F]_b = [F]_a + D_1 + D_2, \tag{41}$$

where we define the remainder terms D_1 and D_2 by

$$D_1 = \int_a^b dt L_1 F,$$

$$D_2 = \int_a^b dW_\beta L_{2\beta} F.$$

This expansion can be iterated. So for example, by expanding the integrands in the expression above in the same manner, we arrive at a second-order series:

$$[F]_b = [F]_a + C_1 + C_2 + D_{11} + D_{12} + D_{21} + D_{22}, \tag{42}$$

where

$$C_1 = [L_1 F]_a \int_a^b dt,$$

$$C_2 = [L_{2\beta}F]_a \int_a^b dW_\beta,$$

$$D_{11} = \int_a^b dt \int_a^t dt' L_1^2 F,$$

$$D_{12} = \int_a^b dt \int_a^t dW'_\beta L_{2\beta} L_1 F,$$

$$D_{21} = \int_a^b dW_\beta \int_a^t dt' L_1 L_{2\beta} F,$$

$$D_{22} = \int_a^b dW_\beta \int_a^t dW'_\gamma L_{2\gamma} L_{2\beta} F.$$

The C's are defined to be those terms where the final integrand is a constant, due to the first term on the RHS of Eq. (41) that can be moved out of the integral. The D's are remainder terms, where the final integrand is an implicit non-constant function of time, resulting from the second and third terms on the RHS of Eq. (41). The subscript 1 indicates an integration over time, and the subscript 2 indicates an integral over Brownian motion.

By systematically repeating such expansions, we can easily write down the expression for the Taylor series of any order. The order of error of the remainder terms can be determined by noting that each integral over t contributes a full order and each integral over W contributes half an order. So for example, the D_{12} term is $O(\Delta^{3/2})$.

B. Algorithm

Finally, we are ready to describe the algorithm itself. Our starting point for the algorithm is the combined translational and rotational equations of motion, Eqs. (30) and (31) derived in the previous sections. In order to simplify the analysis, we write these equations of motion for a single rigid body in general form as

$$dV_\alpha = g_\alpha(V,X)dt + s_{\alpha\beta}dW_\beta, \tag{43}$$

$$dX = h_\alpha(V,X)dt, \tag{44}$$

where X and V are the generalized position and velocity variables, defined in Eqs. (24) and (25), and we have defined

$$g_\alpha(V,X) := a_\alpha(X) + b_{\alpha\beta n}V_\beta V_n - \gamma_{\alpha\beta}V_\beta, \tag{45}$$

$$h_\alpha(V,X) := c_{\alpha nm}X_n V_m + d_{\alpha nmj}X_n X_m V_j. \tag{46}$$

We will also find it useful to know the following partial derivatives of g and h:

$$\frac{\partial g_\alpha}{\partial V_\beta} = (b_{\alpha\beta n} + b_{\alpha n\beta})V_n - \gamma_{\alpha\beta}, \tag{47}$$

$$\frac{\partial h_\alpha}{\partial V_\beta} = c_{\alpha n\beta}X_n + d_{\alpha nm\beta}X_n X_m, \tag{48}$$

$$\frac{\partial h_\alpha}{\partial X_n} = c_{\alpha nm}V_m + (d_{\alpha nmj} + d_{\alpha mnj})X_m V_j. \tag{49}$$

Recall our rule that the indices i,j,k range over the translational degrees of freedom, the indices n,m,l range over the rotational variables freedom, and the indices α,β,γ range over both. There are three translational and three rotational degrees of freedom for the generalized velocity variables V, and three translational and four rotational degrees of freedom for generalized position variables X, since one of the rotational degrees of freedom is redundant. Finally, recall the definitions of the various tensors: $a_\alpha(X)$ is the generalized force-torque term, $b_{\alpha\beta n}$ describes geometric effects resulting from the change of basis as the particle moves [see Eqs. (13) and (22)], γ is the frictional acceleration tensor, which may couple together the six degrees of freedom of the generalized velocity, s is the random acceleration tensor, related to γ as described in Sec. III, c parametrizes the rotor equation, in tensor form, see Eq. (14), and d parametrizes a rotation described by a rotor R, see Eq. (23).

Note that we can easily write down expressions for any derivative of g that involves V, and for any derivative of h, because these functions are polynomials in X and V. We wish to avoid calculating X only derivatives of g, because this would involve multiple force evaluations of a, which has proved to be inefficient in typical simulations [22]. In practice, such evaluations turn out not to be required in the algorithm.

We assume that at the current time step, $t=0$, we know X and V. We derive the new values for these quantities at time $t=\Delta$. The usual velocity Verlet algorithm is accurate to an error tolerance of $O(\Delta^4)$ in position. (In fact, the story is a bit more complicated than that: the error at each time step is $O(\Delta^3)$, but the Δ^3 terms cancel at alternate time steps.) Due to the presence of the stochastic terms, we are unable to maintain this accuracy without increased complexity in the algorithm. However, we work to an accuracy such that, in the limit where the frictional and random forces approach zero, this $O(\Delta^4)$ accuracy is regained. In order to keep track of the error for both stochastic and nonstochastic versions of the algorithm, we use the special notation $O(\Delta^n,\Delta^m)$ where $O(\Delta^n)$ is the error for the stochastic algorithm and $O(\Delta^m)$ is the error for the nonstochastic algorithm.

The steps in the algorithm are summarized below:

(1) Begin with initial generalized positions and velocities, $[X]_0$ and $[V]_0$.

(2) Generate random variables W and Y, as explained in Sec. IV B 4.

(3) Evaluate g and h, using Eqs. (45) and (46), and the relevant partial derivatives, using Eqs. (47)–(49).

(4) Use the values from steps 1–3 to propagate X to the next time step, $[X]_\Delta$, as explained in Sec. IV B 1.

(5) Evaluate the new potential, $[a(X)]_\Delta$.

(6) Use the values from steps 1–5 to define the coefficients of the nonlinear equation for V, as explained in Sec. IV B 2.

(7) Solve this equation for V by iteration as explained in Sec. IV B 3.

(8) Update variables to the new time step and repeat from step 1.

1. Equation for X

We first calculate $[X]_\Delta$ from $[X]_0$ and $[V]_0$. We work to an error tolerance of $O(\Delta^{5/2},\Delta^3)$ (see above). In fact, we can relatively easily carry the expansion further and derive terms, involving stochastic processes such as twice integrated Brownian motion, to achieve an error of $O(\Delta^{7/2},\Delta^3)$ (see Appendix B), but in practice we prefer to utilize the simpler, lower order expressions given below.

Performing a stochastic Taylor expansion of X, Eq. (44), as described in Sec. IV A, we find that

$$[X_\alpha]_\Delta = [X_\alpha]_0 + C_1 + C_{11} + C_{12} + O(\Delta^{5/2},\Delta^3), \tag{50}$$

where

$$C_1 = [h_\alpha]_0\Delta, \tag{51}$$

$$C_{11} = \left[g_\beta\frac{\partial h_\alpha}{\partial V_\beta} + h_n\frac{\partial h_\alpha}{\partial X_n}\right]_0\frac{\Delta^2}{2}, \tag{52}$$

$$C_{12} = s_{\beta\gamma}\left[\frac{\partial h_\alpha}{\partial V_\beta}\right]_0 [Y_\gamma]_0^\Delta, \tag{53}$$

where these expressions can be evaluated by substituting from Eqs. (45)–(49). We define Y as integrated Brownian motion:

$$[Y]_a^b := \int_a^b dt [W]_a^t, \tag{54}$$

for Brownian motion defined as

$$[W]_a^b := \int_a^b dW. \tag{55}$$

We have not indicated terms in the expansion that turn out to be zero: $C_{2\ldots}$ are all zero because there is no stochastic term in Eq. (44). Also, the stochastic term C_{122}, which would have error $O(\Delta^2)$, is zero because $\partial^2 h_\alpha / dV_\beta dV_\gamma = 0$. All other nonzero terms in the expansion that are not explicitly indicated are part of the error term. Combining the terms in the expansion gives:

$$[X_\alpha]_\Delta = [h_\alpha]_0 \Delta + \left[g_\beta \frac{\partial h_\alpha}{\partial V_\beta} + h_n \frac{\partial h_\alpha}{\partial X_n}\right]_0 \frac{\Delta^2}{2} + s_{\beta\gamma}\left[\frac{\partial h_\alpha}{\partial V_\beta}\right]_0 [Y_\gamma]_0^\Delta + O(\Delta^{5/2}, \Delta^3). \tag{56}$$

2. Equation for V

If the substitutions Eqs. (45)–(49) are made in Eq. (56), then it will be seen that in order to emulate the accuracy of the usual Verlet algorithm, we require values for $[V]_0$ up to error $O(\Delta^{3/2}, \Delta^3)$. Therefore, having derived $[X]_\Delta$, we now derive $[V]_\Delta$ to this accuracy.

We perform an integration that is reminiscent of the velocity Verlet algorithm. Performing stochastic Taylor expansions from $[V]_{\Delta/2}$ to $[V]_\Delta$ and from $[V]_{\Delta/2}$ to $[V]_0$, and subtracting the two series, we arrive at the following expression:

$$[V_\alpha]_\Delta = [V_\alpha]_0 + C_1 + D_2 + O(\Delta^{3/2}, \Delta^3), \tag{57}$$

where,

$$C_1 = [g_\alpha]_{\Delta/2} \Delta, \tag{58}$$

$$D_2 = s_{\alpha\beta}[W_\beta]_0^\Delta, \tag{59}$$

where Eq. (45) defines g. Combining these terms and substituting from Eq. (45), we have

$$[V_\alpha]_\Delta = [a_\alpha]_{\Delta/2} + b_{\alpha\beta n}[V_\beta V_n]_{\Delta/2} - \gamma_{\alpha\beta}[V_\beta]_{\Delta/2} + s_{\alpha\beta}[W_\beta]_0^\Delta + O(\Delta^{3/2}, \Delta^3). \tag{60}$$

3. Solving the nonlinear equation for V

There is an added complication here, as Eq. (60) requires the half time step quantities $[a_\alpha]_{\Delta/2}$ and $[V_\beta]_{\Delta/2}$ to $O(\Delta^{1/2}, \Delta^2)$ in order to satisfy these conditions. We find that

$$[a_\alpha]_{\Delta/2} = \frac{1}{2}([a_\alpha]_0 + [a_\alpha]_\Delta) + O(\Delta^{3/2}, \Delta^2), \tag{61}$$

and also that

$$[V_\beta]_{\Delta/2} = \frac{1}{2}([V_\beta]_0 + [V_\beta]_\Delta) + O(\Delta^{1/2}, \Delta^2), \tag{62}$$

which leads to the following expression for V:

$$[V_\alpha]_\Delta = \left\{\frac{1}{2}[a_\alpha]_0 + \frac{1}{2}[a_\alpha]_\Delta + \frac{1}{4}b_{\alpha\beta n}[V_\beta V_n]_0 - \frac{1}{2}\gamma_{\alpha\beta}[V_\beta]_0\right\}\Delta + s_{\alpha\beta}[W_\beta]_0^\Delta + \left\{\frac{1}{4}(b_{\alpha\beta\gamma} + b_{\alpha\gamma\beta})[V_\gamma]_0 - \frac{1}{2}\gamma_{\alpha\beta}\right\}\Delta[V_\beta]_\Delta + \frac{1}{4}b_{\alpha\beta n}\Delta[V_\beta V_n]_\Delta + O(\Delta^{3/2}, \Delta^3). \tag{63}$$

Note that the RHS of this expression contains terms linear and quadratic in the unknown, $[V]_\Delta$. We must perform an extra step to solve this equation for $[V]_\Delta$. Following the method used for the leapfrog algorithm in [13], the equation is solved by iteration, using the first order approximation defined by setting $[V]_\Delta \to [V]_0$ as an initial guess. So long as Δ is not too large, the solution is found to converge rapidly, usually within around 2–11 iterations.

4. Random variables

In the expressions above, we encounter the following random variables:

$$[W]_0^\Delta := \int_0^\Delta dW \quad \text{Brownian motion}, \tag{64}$$

$$[Y]_0^\Delta := \int_0^\Delta [W]_0^t dt \quad \text{Integrated Brownian motion}. \tag{65}$$

$[W]_0^\Delta$ is a Gaussian random variable with mean zero and variance Δ. $[Y]_0^\Delta$ is also a Gaussian random variable, with mean zero and variance $\Delta^3/3$. The two are not independent. Their covariance is

$$\langle [W]_0^\Delta [Y]_0^\Delta \rangle = \frac{\Delta^2}{2}. \tag{66}$$

We can generate these variables (see Kloeden and Platen [27]) by first generating two independent random variables U_1 and U_2 from $N(0;1)$ and then setting

$$[W]_0^\Delta = U_1 \sqrt{\Delta},$$

$$[Y]_0^\Delta = \frac{1}{2}\Delta^{3/2}\left(U_1 + \frac{1}{\sqrt{3}}U_2\right). \tag{67}$$

C. Multiple bodies

The interaction via a potential between multiple rigid bodies, or a set of rigid bodies and a set of point particles, can be handled simply treating each body separately, except that the force $a(X)$ will now depend on the coordinates of all the bodies rather than just a single body (typically, such interactions might be simplified by considering only two-body in-

TABLE I. The diffusion tensor, D, of a molecule of four-aminopyridine.

$D_{tt}(\times 10^{-10}\ \mathrm{m^2\ s^{-1}})$			$D_{tr}(\times 10^{-3}\ \mathrm{m\ s^{-1}})$		
4.787	−0.048	0.047	1.632	1.701	0.314
−0.048	4.594	0.034	1.701	1.174	−1.822
0.047	0.034	4.390	0.314	−1.822	−2.154
$D_{rt}(\times 10^{-3}\ \mathrm{m\ s^{-1}})$			$D_{rr}(\times 10^{9}\ \mathrm{s^{-1}})$		
1.632	1.701	0.314	1.844	−0.063	0.049
1.701	1.174	−1.822	−0.063	1.503	0.010
0.314	−1.822	−2.154	0.049	0.010	1.370

teractions). One can see this by considering the fact that, if the bodies only interact via a potential, the only coupling between bodies will be due to the term $a(X)$. However, the position at the completion of a time step depends only on $a(X)$ at the beginning of the time step, and does not involve any derivatives of a. In other words, all we need is the multibody force evaluated at the beginning of each time step, and then we can calculate the positions of all the particles at the end of the time step, which will allow us to evaluate the new force for the next time step and so on.

V. COMPUTATIONAL EXAMPLE

We have tested the algorithm by simulating the motion of a molecule of 4-AP ($H_2NC_5H_4N$), which is a small ion channel blocker used to characterize potassium channels and in the treatment of multiple sclerosis. The molecular coordinates of 4-AP are extracted from the Research Collaboratory for Stuctural Bioinformatics (RCSB) protein data bank entry 1AEG [28]. The molecule is then rotated to its principal axes. The principal moments of inertia of the molecule are $(I_x, I_y, I_z) = (89, 195, 284)$ Å^{2}u. The diffusion tensor is determined using the HYDROPRO program [23], and is shown in Table I above. Atomic coordinates and masses, and the diffusion tensor, are used as input to the simulation, which is performed at a temperature of 300 K.

There is actually a separation in timescales inherent in this particular diffusion tensor. The average fitted decay time of the autocorrelation function for the three translational coordinates is 58.8 fs, whereas for the three rotational coordinates it is 4.5 fs. The mean free rotational time is therefore about an order of magnitude smaller than the mean free translational time, and therefore we must work at the rotation timescale. We have verified that the algorithm functions under these conditions, but they do not provide a good test case, as the translational motion will be simulated on a shorter timescale than would normally be the case. In order to provide a better test of both the translational and rotational parts of the algorithm working together, as well as to improve the efficiency of the algorithm under these conditions, while still essentially simulating the real 4-AP molecule, we have done a "temperature rescaling" of the rotational degrees of freedom. Referring to Eq. (34), the frictional acceleration tensor ζ is divided into blocks ζ_{tt}, ζ_{tr}, ζ_{rt}, and ζ_{rr} (with t indexing the translational and r the rotational degrees of freedom), which are modified by setting $\zeta_{tr} \to \sqrt{\sigma}\zeta_{tr}$, $\zeta_{rt} \to \sqrt{\sigma}\zeta_{rt}$ and $\zeta_{rr} \to \sigma\zeta_{rr}$, where σ is a scaling factor. Similarly, referring to Eq. (35), the frictional acceleration u is modified by setting $u_{tr} \to \sqrt{\sigma}u_{tr}$, $u_{rt} \to \sqrt{\sigma}u_{rt}$ and $u_{rr} \to \sigma u_{rr}$. These transformations can be seen, referring to Eqs. (34) and (35) to amount to a decrease in T for the rotational degrees of freedom only, while keeping D (and, therefore, the RMS translational and angular distance diffused in a given time) constant. For the example considered here, we use $\sigma = 0.1$. Under the rescaled parameters, we use a time step of 5 fs, which seems to give good performance.

The results of simulating the free evolution of the molecule are shown in Fig. 1. The center of mass and rotational orientation of the molecule diffuse randomly from their starting values.

We check that the equipartition theorem holds by plotting the time averaged translational and rotational kinetic energies as the simulation progresses, see Fig. 2. The average translational and rotational kinetic energies do indeed appear to converge to $(3/2)kT$, as predicted by the equipartition theorem.

As a further test, we check that the velocity autocorrelation functions for the various degrees of freedom are consis-

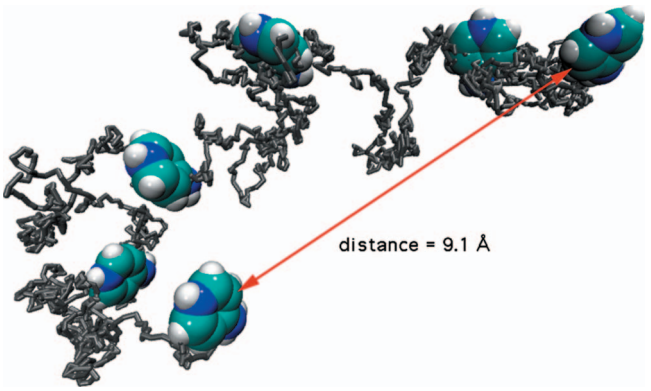

FIG. 1. (Color) The motion of a molecule of four-aminopyridine (4-AP) under the influence of Brownian frictional and random forces. A total of 250 ps of simulation time is shown, and the simulation time step is 5 fs. The trajectory traced out by the center of mass in three-dimensional space is depicted as a line. The molecule itself is depicted at 50 ps intervals in order to show the rotational orientation. For clarity, the size of the molecule is depicted as 25% of its actual size.

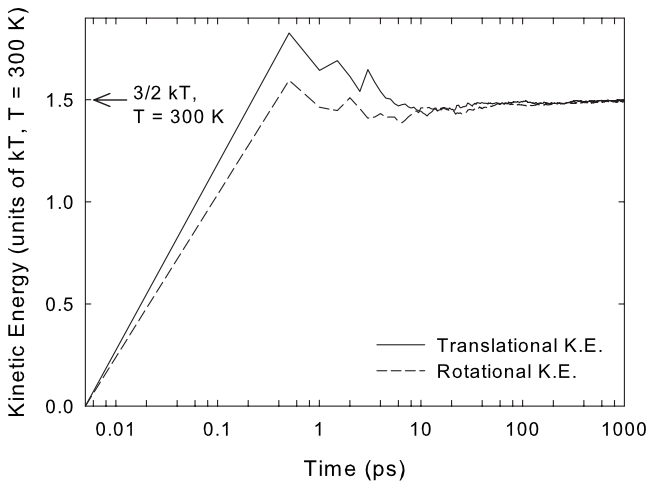

FIG. 2. The time averaged translational and rotational kinetic energies for a molecule of four-aminopyridine. Both quantities appear to converge to a value of $3/2kT$, therefore satisfying the equipartition theorem.

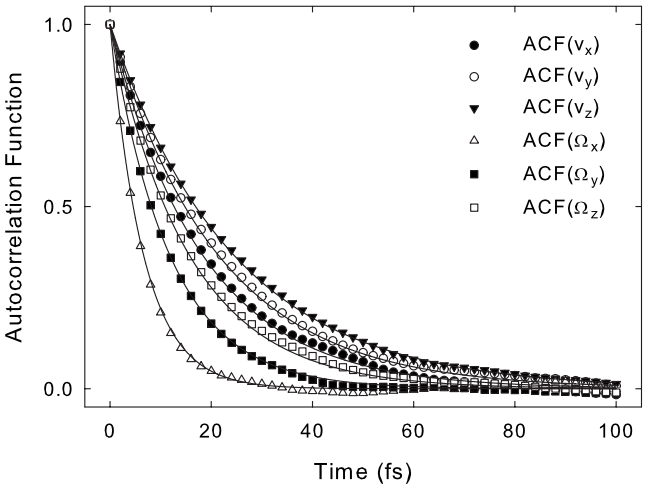

FIG. 3. The velocity autocorrelation and angular velocity autocorrelation functions. Computational results are shown as points, and the theoretical curves derived from the frictional acceleration tensor are shown as lines. There is a good match between computational results and theory.

tent with the results predicted by the frictional acceleration tensor. In order to do so, we modify the diffusion tensor by setting all off-diagonal elements to zero, and by changing the diagonal elements so that the frictional decay timescales in the system can be easily distinguished. The diagonal elements of the modified diffusion tensor are $(D_{xx}\ D_{yy}\ D_{zz}) = (4.78, 5.59, 6.39) \times 10^{-10}$ m^2 s^{-1} for the translational degrees of freedom and $(D_{\Omega_x\Omega_x}\ D_{\Omega_y\Omega_y}\ D_{\Omega_z\Omega_z}) = (1.84, 1.50, 1.37) \times 10^9$ s^{-1} for the rotational degrees of freedom. The system is simulated for 1 ns, and the velocity autocorrelation and angular velocity autocorrelation functions are calculated. Theoretically, we expect that the velocity autocorrelations will be exponentially decaying functions, with decay constants given by the diagonal components of the diagonal frictional acceleration tensor, γ. In Fig. 3, we plot the computational results against the theoretical curves. There is a good match, meaning that the algorithm shows the correct diffusive behavior.

In order to test the scaling of the error per time step as a function of the time step, we perform two parallel simulations. The first simulation lasts for a single time step, and the second simulation breaks this large time step down into many smaller time steps. The state at the end of the second simulation is then used as a benchmark for judging the error in the first simulation. Some care needs to be taken for the stochastic terms, since the underlying random process needs to be equivalent for both simulations. We ensure that this is the case by using a special procedure to calculate the large time step random variables: instead of directly sampling the random variables, the series of small-time step random variables is used to deduce corresponding large-time step random variables.

We apply a force and torque by applying a harmonic potential located at the origin and with spring constant 0.1 kg s^{-2} to the H2 hydrogen atom (see RCSB data bank, 1AEG.pdb). An initial velocity of $(10, 50, -60)$ m s^{-1} and an initial angular velocity of $(5, 2, 10) \times 10^9$ s^{-1} is applied. The molecule is initially located with its center of mass at the origin.

We test the error scaling both with and without the presence of random forces by running two sets of simulations, one at $T=300$ K and the other at $T=0$ K. Results are shown in Fig. 4, and are close to the predicted error scaling. At zero temperature, where no random forces are present, the error for both generalized position and generalized velocity variables scales as approximately $O(\Delta^3)$. Running at finite temperature causes the scaling of error in generalized position to go to approximately $O(\Delta^{5/2})$, whereas the error in generalized velocity goes to approximately $O(\Delta^{3/2})$. We conclude that the algorithm performs more or less as predicted with regard to per-time step error.

VI. DISCUSSION AND CONCLUSION

We have derived an algorithm for rigid-body Brownian dynamics that allows molecules of arbitrary shapes and hy-

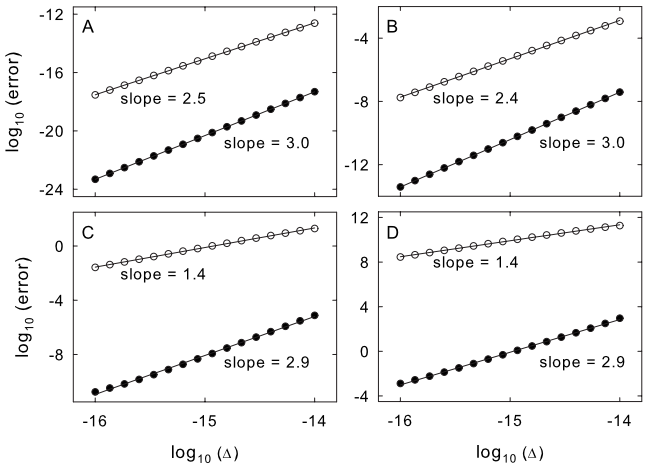

FIG. 4. Log-Log plots showing the scaling of the error per time step as a function of the timestep. The black circles were obtained at $T=0$ K, and the white circles were obtained at $T=300$ K. (A) Error in **x**. (B) Error in R. (C) Error in **v**. (D) Error in Ω.

drodynamic friction tensors to be simulated. In the zero-temperature limit, the algorithm performs like the velocity Verlet algorithm with regard to error propagation. In the stochastic case, the scaling of the local error is a factor Δ better than the Euler algorithm. We have demonstrated the feasibility and error scaling of our algorithm in sample numerical calculations.

Consider simulating a group of solute molecules (for example, globular proteins) interacting in a solvent. In normal molecular dynamics, each of the solvent molecules is simulated explicitly, and, in addition, the internal dynamics of the atoms in the solute molecules are taken into account. This can be quite computationally expensive. For many applications, it may be preferable to treat the solvent as an implicit, bulk medium, and to treat the solute molecules as being rigid bodies. For example, implicit solvent and rigid-body approximations are employed in Brownian dynamics models of biological ion channels [29]. They allow simulation times of tens of microseconds, as opposed to the tens of nanoseconds typically seen in molecular dynamics studies.

In a typical Brownian dynamics simulation of an ion channel, the channel and the lipid membrane within which it is embedded are treated as a stationary, rigid structure containing fixed charges and with a given dielectric constant. Dissolved ions are modeled as charged spheres. The water that surrounds and permeates the channel is treated as an implicit bulk solvent. It has two effects: firstly, it modifies the electrostatic interactions in the system through the presence of a dielectric constant, and secondly, it causes the ions to undergo Brownian motion by giving rise to frictional and random forces, which are normally modeled using the Langevin equation. The ions interact with the channel through macroscopic electrostatics.

We are currently incorporating the algorithm presented here into a similar Brownian dynamics model of the interaction between biological cell membrane ion channels (protein pores that span the cell membrane and provide a conduction pathway for ions) and *channel blockers* (polypeptides or other molecules that may affect the function of the ion channel by physically occluding the pore). To do so, we will need to model the motion of the blocker molecules with respect to the channel, including a realistic treatment of the translational and rotational diffusion. The ion channel will be treated as a fixed, rigid structure, and the blocker molecules as rigid bodies undergoing rotational and translation Brownian motion, simulated using our algorithm. The blocker molecules will be present in concentrations low enough that hydrodynamic interactions between two or more blocker molecules are not expected to play an important role; the molecules will experience independent Brownian motion, and will interact via an ordinary potential. We will combine Brownian dynamics techniques [29], developed for studying the conduction of ions through ion channels, with the rotational algorithm presented here. This will allow us to model the interaction between the channel, the blocker molecules, and the ions in solution. However, the algorithm may have wider applicability to modeling the motion of any fairly rigid molecules, colloidal particles, nanoparticles, liquid crystals, or other small particles in fluids.

APPENDIX A: REVIEW OF GEOMETRIC ALGEBRA OF 3D EUCLIDEAN SPACE

The geometric algebra of three–dimensional (3D) Euclidean space can be constructed by starting with real scalars and the usual vectors of Euclidean 3D space. A geometric product, denoted *ab*, is introduced, which is associative and distributive over addition, and with the additional requirement that the square of any vector under this geometric product is real (and equal to the dot product square). A set of *multivectors* (itself a vector space) is then constructed as the space spanned by linear combinations of arbitrary products of vectors and scalars. It can be shown that the multivectors in 3D Euclidean space admit the following basis:

Object	Grade	Basis Multivectors
Scalars	Grade 0	1
Vectors	Grade 1	$\{\mathbf{e}_1,\mathbf{e}_2,\mathbf{e}_3\}$
Bivectors	Grade 2	$\{\mathbf{e}_2\mathbf{e}_3,\mathbf{e}_3\mathbf{e}_1,\mathbf{e}_1\mathbf{e}_2\}$
Pseudoscalar	Grade 3	$I=\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$

For vectors, the symmetric component of the product is the usual dot product $(\cdot)$, and the antisymmetric component is the wedge product of Clifford algebra $(\wedge)$, so $\mathbf{ab}=\mathbf{a}\cdot\mathbf{b}+\mathbf{a}\wedge\mathbf{b}$.

The full algebra of the geometric product can be deduced from the axioms of geometric algebra. Here, we define it directly by its action on the basis vectors. We employ the Einstein summation convention, where a repeated index implies summation over that index.

$$1a = a1 = a \quad a \text{ is any multivector}, \tag{A1}$$

$$Ia = aI \quad a \text{ is any multivector}, \tag{A2}$$

$$\mathbf{e}_i\mathbf{e}_j = \delta_{ij} + \epsilon_{ijk}I\mathbf{e}_k, \tag{A3}$$

$$I^2 = -1. \tag{A4}$$

This table is sufficient to define any multivector multiplication. Note that, as the bivector basis vectors can be written as a vector times the pseudoscalar, e.g. $\mathbf{e}_1\mathbf{e}_2=I\mathbf{e}_3$, there exists a natural mapping between the vectors and bivectors. In fact, the axial vectors commonly used to represent cross product quantities such as angular momentum and the like are best viewed as bivectors, and the wedge product is used in place of the more complicated and less general cross product.

1. Reversion

If a is written as a sum of products of vectors, then $a^\dagger$ reverses the order of all vector factors. For example, if $a=1+\mathbf{e}_1+4\mathbf{e}_2\mathbf{e}_3$, then $a^\dagger=1+\mathbf{e}_1+4\mathbf{e}_3\mathbf{e}_2=1+\mathbf{e}_1-4\mathbf{e}_2\mathbf{e}_3$.

2. Quaternions

In the context of three-dimensional Euclidean geometric algebra, Hamilton's quaternions (a four component object consisting of a scalar plus three anticommuting imaginary components) have the same algebraic properties as mixed

grade objects consisting of scalars plus bivectors. A quaternion $a+bi+cj+dk$ is equivalent to the multivector $a+bI\mathbf{e}_1+cI\mathbf{e}_2+dI\mathbf{e}_3$, which we represent here in component form as (a,b,c,d). Note that the reverse of (a,b,c,d) is $(a,b,c,d)^\dagger=(a,-b,-c,-d)$. In what follows, we shall make no distinction between Hamilton's quaternions and these objects, which we shall simply refer to as quaternions.

3. Rotors

A rotor is a unit quaternion that can be used to represent rotations. If a is any multivector and R a rotor, then $RaR^\dagger$ rotates a with a direction and magnitude specified by R. The inverse of the rotation is $R^\dagger$, and we necessarily have $RR^\dagger=1$.

Given the components of a rotor, it is possible to write down a rotation matrix which acts on the column vector consisting of the components of a vector. This is probably the most efficient way to actually implement rotations numerically [30].

APPENDIX B: SKETCH OF A HIGHER ORDER ALGORITHM

In deriving stochastic motion algorithms, experience teaches us to avoid derivatives of $a(X)$, because these would necessitate multiple force evaluations at each step. Given this restriction, we can still derive a stochastic algorithm that is more accurate (but also more complicated) than the algorithm given above. We provide a rough sketch of such an algorithm below:

In the equation for X, we can carry the stochastic Taylor expansion further to include the terms C_{112} and C_{121}:

$$C_{112}=s_{\gamma\delta}\left[\frac{\partial g_\beta}{\partial V_\gamma}\frac{\partial h_\alpha}{\partial V_\beta}+\frac{\partial h_n}{\partial V_\gamma}\frac{\partial h_\alpha}{\partial X_n}+h_n\frac{\partial^2 h_\alpha}{\partial V_\gamma\partial X_n}\right]_0\times\int_0^\Delta dt\int_0^t dt' W'_\delta(t'), \tag{B1}$$

$$C_{121}=s_{\beta\gamma}\left[h_n\frac{\partial^2 h_\alpha}{\partial X_n\partial V_\beta}\right]_0\int_0^\Delta dt\int_0^t dW'_k t'. \tag{B2}$$

The expression for X is now accurate to $O(\Delta^{7/2},\Delta^3)$ (with the $O(\Delta^3)$ error canceling, as usual, each alternate time step, to give an error of $O(\Delta^{7/2},\Delta^4)$). Note that there are additional random variable terms (the integrals in the expressions above) that are correlated with the existing random variables W and Y. Deriving the additional variances and covariances variables is left as an exercise for the reader.

In order to maintain this degree of accuracy in X, we must now derive V to an accuracy $O(\Delta^{5/2},\Delta^3)$. This necessitates adding a term for C_{12} to Eq. (57):

$$C_{12}=s_{\beta\gamma}\left[\frac{\partial g_\alpha}{\partial V_\beta}\right]_{\Delta/2}\int_0^\Delta dt\int_{\Delta/2}^t dW'_\gamma. \tag{B3}$$

This term contains yet another stochastic integral, but by splitting all stochastic integrals into half time step pieces and performing some simple manipulations, we can express it in terms of Brownian motion W and integrated Brownian motion Y.

Note also that $[\partial g_\alpha/\partial V_\beta]_{\Delta/2}$ is evaluated at a half time step. This term, as well as the term $[g_\alpha]_{\Delta/2}$ seen in the original algorithm, are treated in exactly the same manner as previously, but in order to maintain accuracy, we must add a term to Eq. (62), which now becomes

$$[V_\beta]_{\Delta/2}=\frac{1}{2}([V_\beta]_0+[V_\beta]_\Delta)+s_{\beta\gamma}([W_\gamma]_0^{\Delta/2}-[W_\gamma]_{\Delta/2}^\Delta)+O(\Delta^{3/2},\Delta^2). \tag{B4}$$

As was the case before, the equation for $[V]_\Delta$ contains terms linear and quadratic in the unknown, $[V]_\Delta$ on the RHS, and an additional step must be performed to solve it.

[1] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson, J. Chem. Phys. **76**, 637 (1982).

[2] We also provide a sketch of a more sophisticated algorithm that allows the error to be reduced to $O(\Delta^{7/2})$.

[3] B. Øksendal, *Stochastic Differential Equations: An Introduction with Applications* (Springer-Verlag, Berlin, 1989).

[4] A. Greiner, W. Strittmatter, and J. Honerkamp, J. Stat. Phys. **51**, 95 (1988).

[5] M. X. Fernandes and J. G. de la Torre, Biophys. J. **83**, 3039 (2002).

[6] D. A. Beard and T. Schlick, Biophys. J. **85**, 2973 (2003).

[7] W. van Gunsteren and H. J. C. Berendsen, Mol. Phys. **45**, 637 (1982).

[8] S. H. Chung, T. Allen, M. Hoyles, and S. Kuyucak, Biophys. J. **75**, 793 (1998).

[9] T. Baştuğ and S. Kuyucak, Chem. Phys. Lett. **401**, 175 (2005).

[10] D. Gordon, V. Krishnamurthy, and S. H. Chung, Mol. Phys. **106**, 1353 (2008).

[11] X. Sun, T. Lin, and J. D. Gezelter, J. Chem. Phys. **128**, 234107 (2008).

[12] A. Dullweber, B. Leimkuhler, and R. McLachlan, J. Chem. Phys. **107**, 5840 (1997).

[13] I. P. Omelyan, Phys. Rev. E **58**, 1169 (1998).

[14] D. L. Ermak and J. A. McCammon, J. Chem. Phys. **69**, 1352 (1978).

[15] K. Yu, W. Fu, H. Liu, X. Luo, K. X. Chen, J. Ding, and J. Shen, Biophys. J. **86**, 3542 (2004).

[16] E. Dickinson, S. A. Allison, and A. J. McCammon, J. Chem. Soc., Faraday Trans. 2 **81**, 591 (1985).

[17] It seems likely that the algorithm could be extended to this case: this would involve adding a particle index to the random and frictional tensors, and would not seem to change the mathematics of the derivation. However, dealing with the complicated structure of the new larger friction tensors would no doubt present considerable practical difficulty.

[18] J. P. Ryckaert, G. Ciccotti, and H. J. C. Berendsen, J. Comput.

Phys. **23**, 327 (1977).
[19] C. Doran and A. Lasenby, *Geometric Algebra for Physicists* (Cambridge, Cambridge, England, 2003).
[20] Angular velocity Ω can be defined by its action on unit vectors: if $\mathbf{e}_i$ is a frame rotating with angular velocity Ω, then $\dot{\mathbf{e}}_i = \mathbf{e}_i \cdot \Omega$.
[21] We assume the Stratonovich rather than Ito stochastic integral, which means that the usual transformation laws of calculus apply.
[22] H. J. C. Berendsen, *Simulating the Physical World* (Cambridge, Cambridge, England, 2007).
[23] J. G. de la Torre, M. L. Huertas, and B. Carrasco, Biophys. J. **78**, 719 (2000).
[24] http://leonardo.fcu.um.es/macromol/programs/hydropro/hydropro.htm.
[25] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing* (Cambridge, Cambridge, England, 2007).
[26] P. E. Kloeden and E. Platen, Math. Nachr. **151**, 33 (1991).
[27] P. E. Kloeden and P. Eckhard, *Numerical Solution of Stochastic Differential Equations* (Springer–Verlag, Berlin, 1992).
[28] R. A. Musah, G. M. Jensen, S. W. Bunte, R. J. Rosenfeld, and D. B. Goodin, J. Mol. Biol. **315**, 845 (2002).
[29] S. H. Chung, T. W. Allen, M. Hoyles, and S. Kuyucak, Biophys. J. **77**, 2517 (1999).
[30] L. Dorst, D. Fontijne, and S. Mann, *Geometric Algebra for Computer Scientists* (Morgan Kaufmann, San Francisco, 2007).