Extended Brownian dynamics. II. Reactive, nonlinear diffusion

Gene Lamm[a)] and Klaus Schulten

Physik Department, Technische Universität München, 8046 Garching and Max Planck Institut für Biophysikalische Chemie, 3400 Göttingen, Federal Republic of Germany

(Received 10 September 1982; accepted 26 October 1982)

An improved version of the "extended Brownian dynamics" algorithm recently proposed by the authors [J. Chem. Phys. 75, 365 (1981)] is given. This Monte Carlo procedure for solving the one-dimensional Smoluchowski diffusion equation is statistically exact near a boundary for a constant force and approximately correct for a linear force. The improved algorithm is both more accurate and simpler than the earlier version. In addition, the algorithm is extended to include diffusion near a reactive boundary or in a reactive optical potential. The treatment of diffusion for nonlinear forces is conveniently handled by choosing the time for a single diffusive jump locally. The algorithm converges as this jump time approaches zero. The appropriate modifications necessary to treat diffusion between two (possibly reactive) boundaries or diffusion with a spatially varying diffusion coefficient are also given. Finally, it is shown how multidimensional diffusion in a spherically symmetric force field may be treated by the one-dimensional algorithm described here. As in the earlier paper, numerical results are presented and compared with analytical and numerical descriptions of the diffusion process to demonstrate the validity of the algorithm.

I. INTRODUCTION

In recent years, several Monte Carlo algorithms for solving the Smoluchowski diffusion equation (SDE) have been proposed.[1-3] The need for such an algorithm is evident when one realizes that other numerical techniques (chiefly those based on finite-difference schemes[4,5]) are limited to essentially one-dimensional problems. Monte Carlo (MC) approaches, on the other hand, are capable of treating asymmetric three-dimensional diffusion (in the presence of boundaries), and so, offer a way of investigating certain diffusion-controlled processes that have previously been untreatable.

The success of a numerical algorithm for solving the diffusion equation may be judged by its performance in four categories:

(i) The algorithm should be applicable to the general three-dimensional diffusion equation, including force and reaction terms and general boundary conditions. Although, in principle, finite-difference schemes are capable of treating the general three-dimensional equation, the requirement of a convergent step size or the imposition of certain boundary conditions makes the approach too time consuming [category (3) below] and/or computer code too tedious to write [category (4)]. As for three MC schemes recently proposed,[1-3] all are capable of handling three-dimensional diffusion, although only the work of Ermak[1] has explicitly done so. Furthermore, only the Schulten–Epstein algorithm[2] (an extension of the Ermak algorithm) has dealt with reactive diffusion.

(ii) The second category of interest is that of *convergence*. All algorithms should yield results that converge uniformly to the correct results as some internal parameter(s) is varied. For finite-difference schemes, the convergence parameter is the mesh size of the diffusion space. MC algorithms converge as the size of the individual jumps taken by the "Brownian dynamics" particle is decreased and in the limit of increasing sample size (total number of trajectories). The second limit is statistical in nature and is inherent in all MC procedures, the convergence rate being proportional to $1/\sqrt{N}$, where N is the number of trajectories sampled. The size of the diffusive jumps is dependent on the duration time chosen for these jumps. This jump time Δt is usually taken to be as large as possible subject to the assumed constancy or linearity of the local force. Furthermore restrictions on the jump time may be required if a boundary or optical potential is present.

(iii) The above category of the convergence of an algorithm is closely connected with its *feasibility*. As mentioned above, finite-difference schemes are, in principle, applicable to the general three-dimensional diffusion equation, however, computational space and/or time limitations make this approach impractical for all but the simplest problems. Computational requirements of MC algorithms limit the statistical accuracy of the results. Letting T denote the diffusion time simulated by one trajectory, the total computer time for a MC program is then proportional to $NT/\Delta t$. With T fixed by the specific application and Δt determined by local force requirements of the particular algorithm used, limitations in available computer time set the statistical accuracy of the results. It is, therefore, important to use an algorithm that allows Δt to be chosen as large as possible for a given problem. The Ermak algorithm[1] requires a local constant force far from any boundary and no force at a boundary. The "extended Brownian dynamics" algorithm of Lamm and Schulten[3] requires a local linear force in the absence of boundaries and is rigorously correct near a boundary only when no force is present (although the results of that algorithm indicate that even for a local linear force, simulation near a boundary is accurate).

(iv) Finally, we must consider the *accessibility* of the numerical routine. As the application of any numerical algorithm necessitates the acquisition of the computer code implementing the algorithm, either the code must be readily available, easily convertible to the machine and problem at hand, and foolproof in use, or the algorithm must be easily programmed by the user. As the difficulties involved in the former procedure are obvious, the second method of acquisition is preferred. Multidimensional finite-difference schemes involving general boundary conditions are notoriously difficult to program. MC algorithms, on the other hand, are quickly programmed for the one-dimensional case and obviously and easily extended to higher-dimensional diffusion problems. Of those MC methods already mentioned, the Ermak[1] and Schulten–Epstein[2] algorithms are essentially equivalent, only the applications being different. The "extended" algorithm of Lamm and Schulten[3] is slightly more difficult to program owing to the method of treatment of the boundary, but the savings in computer time offered by the local linear force approximation should be advantageous.

In the following, another MC diffusion algorithm is presented which was developed specifically with the above comments in mind. As a first demonstration, we investigate the one-dimensional diffusion equation, including nonlinear force and reaction terms and boundary effects. The extension to three-dimensional situations will be discussed in a later publication. To maximize the jump time for diffusive jumps, and to minimize computational effort, both the analytic solution for constant force diffusion near a reactive boundary (due to Smoluchowski, Secs. II and III), and that for linear diffusion in the absence of boundaries (the Ornstein-Uhlenbeck process, Sec. IV) will be incorporated. Analytic inversion of the cumulative distribution function to obtain the diffusion jump endpoint provides a fast algorithm, avoiding the need for separately stored inversion tables (which is impractical for the algorithm proposed here, see Appendix A). The proposed algorithm is then statistically exact for the above-mentioned processes and should allow local jump times in realistic systems (e.g., those involving ionic interactions) that are significantly longer than with other algorithms. To treat diffusion in a spatially varying optical potential, an analytic correction to the local linear distribution is derived that accounts for the reactivity of the process and is valid to first order in the jump time (Sec. V). For nonlinear diffusion or diffusion in a spatially varying optical potential, the algorithm is not exact and convergence depends on the value chosen for the jump time. This time is chosen locally according to local force and/or reaction conditions (Sec. VI). In Sec. VII, numerical results of the algorithm are presented to illustrate certain features and applications. For diffusion in the presence of two boundaries (possibly reactive), a convenient solution to the SDE is needed that can be treated by the algorithm. In Sec. VIII, we derive a suitable approximation to the exact solution that is convergent in the limit of small jump times. In Sec. IX, the appropriate extension of the algorithm to handle diffusion with a spatially varying diffusion coefficient is given. Finally, in Sec. X we show how the SDE for multidimensional diffusion in a spherically symmetric force field may be expressed as a one-dimensional SDE, thus allowing treatment by the algorithm developed here. Sections XI and XII contain final comments and a summary of the important points of the algorithm. Appendices discuss the specific random number generator used and present a brief analysis of the statistical error inherent in the algorithm.

II. NONREACTIVE, CONSTANT FORCE DIFFUSION

We begin with the one-dimensional Smoluchowski diffusion equation (SDE) for diffusion in a constant force

$$\frac{\partial}{\partial t} p(x,t|x_0) = \left(\frac{\partial^2}{\partial x^2} + b\,\frac{\partial}{\partial x}\right) p(x,t|x_0)\,, \tag{2.1}$$

where $b = -\,\mathrm{force}$[6] and $t = D \cdot \mathrm{time}$, D denoting the (constant) diffusion coefficient. The solution to Eq. (2.1), subject to the initial condition that the particle be at $x = x_0$ at time $t=0$, i.e.,

$$p(x, t=0|x_0) = \delta(x-x_0)\,, \tag{2.2}$$

and the reflective boundary condition

$$\left(\frac{\partial}{\partial x} + b\right) p(x,t|x_0) = 0\,, \quad \text{at } x=0\,, \tag{2.3}$$

was obtained analytically by Smoluchowski.[7] It may be written as

$$p(x,t|x_0) = \sum_{i=0,1,2} p_i(x,t|x_0)\,, \tag{2.4}$$

where

$$p_0(x,t|x_0) = (4\pi t)^{-1/2} \exp[-(x-x_0+bt)^2/4t]\,, \tag{2.5}$$

$$p_1(x,t|x_0) = (4\pi t)^{-1/2} \exp[bx_0 - (x+x_0+bt)^2/4t]\,, \tag{2.6}$$

$$p_2(x,t|x_0) = \tfrac{1}{2} b \exp(-bx)\, \mathrm{erfc}[(x+x_0-bt)/\sqrt{4t}]\,, \tag{2.7}$$

where erfc(z) denotes the complementary error function.[8] $p_0(x,t|x_0)$ describes the diffusion process in the absence of the boundary while the remaining two terms account for the boundary. Since the above solution is exact for all values of the force parameter b, a MC algorithm should *at least* correctly describe diffusion in a constant force near a reflective boundary. This is done as follows:

We first require the area (over x) of the partial distribution curves (2.5), (2.6), and (2.7). Direct integration yields

$$N_i(t|x_0) = \int_0^\infty dx\, p_i(x,t|x_0)\,, \quad i=0,1,2\,, \tag{2.8}$$

or explicitly,

$$N_0(t|x_0) = \tfrac{1}{2} \mathrm{erfc}\,[(-x_0+bt)/\sqrt{4t}]\,, \tag{2.9}$$

$$N_1(t|x_0) = \tfrac{1}{2} \exp(bx_0)\, \mathrm{erfc}[(x_0+bt)/\sqrt{4t}]\,, \tag{2.10}$$

$$N_2(t|x_0) = 1 - N_0(t|x_0) - N_1(t|x_0)\,. \tag{2.11}$$

We follow the suggestion made in paper I and partition individual diffusive jumps into one of the three distributions $p_i(x,t|x_0)$. To reproduce the total distribution $p(x,t|x_0)$, we must first insure that the relative number of jumps assigned to a particular p_i is statistically correct. The probability that a jump will be distributed according to p_i is proportional to $N_i(t|x_0)$, the area under the curve. We thus choose a random number r uniformly distributed on the interval $[0, 1)$ (see Appendix A). Then, if $0 \le r < N_0$, the jump endpoint is distributed according to p_0. If, however, $N_0 \le r < N_0 + N_1$, the jump endpoint is chosen from distribution p_1. Finally, if $N_0 + N_1 \le r < 1$, the jump endpoint is distributed according to p_2.[9]

The second, and final, step in the procedure is to distribute the jump endpoint (from x_0 to x_f in time t) according to the above chosen distribution. We now select another uniformly distributed random number $0 \le r' < 1$ and obtain the jump endpoint x_f by inverting the integral for the cumulative distribution function[10]

$$r' = \int_0^{x_f} dx\, p_i(x, t|x_0) \; . \tag{2.12}$$

The lower limit of the integral is 0 (rather than $-\infty$ as in paper I) because the diffusive jump is confined to the diffusion domain $[0, \infty)$. For distributions (2.5) and (2.6), the integral can be inverted analytically to give either

$$x_f = x_0 - bt + \sqrt{4t}\, \mathrm{erfc}^{-1}\left[r' \,\mathrm{erfc}\left(\frac{-x_0 + bt}{\sqrt{4t}}\right)\right] , \tag{2.13}$$

from Eq. (2.5), or

$$x_f = -x_0 - bt + \sqrt{4t}\, \mathrm{erfc}^{-1}\left[r' \,\mathrm{erfc}\left(\frac{x_0 + bt}{\sqrt{4t}}\right)\right] , \tag{2.14}$$

from Eq. (2.6). For jumps made according to p_2, analytic inversion of integral (2.12) is not possible. One alternative is simply to iterate Eq. (2.12) using a Newton–Raphson scheme.[11] A slightly less accurate, but faster, procedure is to approximate $p_2(x, t|x_0)$ by a function that allows analytic inverson of the cumulative distribution function. Either an exponential or Gaussian approximation to p_2 will do and we have chosen the form

$$p_2(x, t|x_0) \approx a \exp(-b^* x) \; . \tag{2.15}$$

Requiring the exact value of $p_2(x, t|x_0)$ at the boundary and equal areas for the approximate and exact distributions gives the constants

$$a = p_2(0, t|x_0) = b[1 - N_0(t|x_0)] , \quad b^* = b\, \frac{1 - N_0(t|x_0)}{N_2(t|x_0)} \; . \tag{2.16}$$

The jump endpoint is then given by

$$x_f = |(\ln r')/b^*| \; . \tag{2.17}$$

We have found representation (2.15) to be accurate for a wide variety of forces tested and clearly superior to that used previously in paper I.[12] If, however, diffusion occurs in the presence of a reactive optical potential near the boundary (Sec. IV), the slight shift in particle density implied by Eq. (2.15) may noticeably affect the resulting distribution. A more accurate form for p_2 may be obtained by using the erf(z) representation

$$\mathrm{erf}(z) = 1 - \exp[-(0.722\,181\,610\, z^2 + 2z/\sqrt{\pi})] , \quad z \ge 0 , \tag{2.18}$$

accurate to 1% for positive values of z. This form also allows analytic inversion of Eq. (2.12) and has been tested. The algorithm is 10%–20% slower than that using Eq. (2.15) and requires additional partitioning, but may be necessary if greater accuracy is required.

The above algorithm for generating jump endpoints according to $p(x, t|x_0)$ of Eq. (2.4) is statistically exact (except for the insignificant approximation made in using Eq. (2.15) to represent p_2). Typical results are shown in Fig. 1(a) below and in Figs. 1 and 2 of paper I, where a brief discussion may be found (see also Appendix C).[13] The advantages of the above algorithm over previous methods are threefold. First, diffusive jumps far from the boundary are made according to $p_0(x, t|x_0)$ since in this case $N_0(t|x_0) \approx 1$. This allows jump times to be chosen assuming a local constant force and will be extended to include a local linear force in the following section. The algorithm correctly biases diffusive drift motion in the direction of the force.[14] Second, diffusion behavior near a boundary is rigorously correct for a constant force. Since an attractive force towards a boundary accumulates particles there, a considerable savings in computer time should result with this new algorithm, implying better statistics through an increased number of trajectories sampled. Third, only as much information as is contained in the SDE is used *and no more*. Thus, in contrast to earlier algorithms, the particles *never strike the boundary* so artificial deterministic boundary conditions are avoided.

As a technical note, the time-consuming part of the algorithm is the evaluation of the erfc(z) and erfc^{-1}(z) functions used to partition and displace each jump (see Appendix A). As the $N_i(t|x_0)$ are known from the first part of the algorithm involved in partitioning events into the correct $p_i(x, t|x_0)$, the erfc(z) calls in the endpoint determination in Eqs. (2.13) and (2.14) need not be made.

III. CONSTANT FORCE DIFFUSION NEAR A REACTIVE BOUNDARY

Consider now diffusion in a constant force near a reactive boundary. The SDE (2.1) is solved subject to initial condition (2.2) and the reactive boundary condition

$$\left(\frac{\partial}{\partial x} + b\right) p_k(x, t|x_0) = k p_k(x, t|x_0) , \quad \text{at } x = 0 , \tag{3.1}$$

where the constant k is a measure of the reaction at the boundary: $k = 0$ implies that no reaction occurs, $k \to \infty$ implies that reaction occurs upon every encounter with the boundary. The analytic solution for this process is

$$p_k(x, t|x_0) = \sum_{i=0,1,2} p_{ki}(x, t|x_0) , \tag{3.2}$$

where, as before,

$$p_{k0}(x, t|x_0) = (4\pi t)^{-1/2} \exp[-(x - x_0 + bt)^2/4t] , \tag{3.3}$$

$$p_{k1}(x, t|x_0) = (4\pi t)^{-1/2} \exp[b x_0 - (x + x_0 + bt)^2/4t] , \tag{3.4}$$

$$p_{k2}(x, t|x_0) = \tfrac{1}{2}(b - 2k) \exp[-bx + k(x + x_0 + (k - b)t)] \times \mathrm{erfc}[(x + x_0 + (2k - b)t)\sqrt{4t}] \; . \tag{3.5}$$

This solution is most easily obtained by Laplace transforming Eqs. (2.1) and (3.1). One method of simulating the reactive diffusion process is to develop an algorithm based on Eq. (3.2) by generalizing the results given in Sec. II.[15] However, there is a simpler method. The

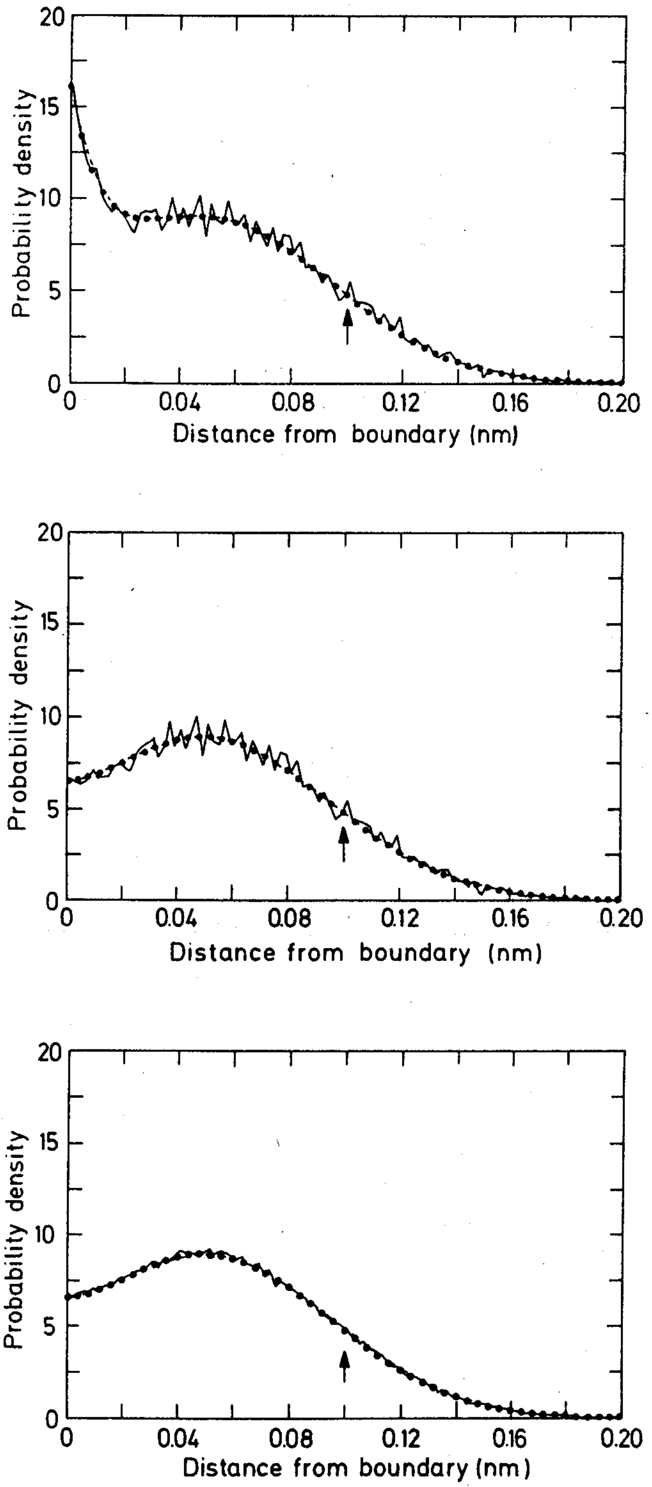


FIG. 1. (a) Results for the probability distribution $p(x, t = 0.001 \text{ nm}^2 | x_0 = 0.1 \text{ nm})$ for diffusion in the constant force $F(x) = -50 \text{ nm}^{-1}$ near a reflective boundary (at $x = 0$). The MC algorithm of Sec. II averaged over 10^4 one-jump trajectories (—) is compared with the analytic expression [- - -, Eq. (2.4)] and the results from a difference equation approach (• • •, see paper I for details). The arrow indicates the starting position $x_0 = 0.1$ nm; the numerical solution assumes an artificial reflective boundary at 0.2 nm; all trajectories with endpoints beyond 0.2 nm in the MC algorithm have been collected in the last endpoint bin at $x = 0.2$ nm. (b) The process of (a) but with a reactive boundary of reactivity $k = 50 \text{ nm}^{-1}$; the MC algorithm is described in Sec. III. (c) The process of (b) but with 10^5 MC trajectories.

survival probability for a jump from x_0 to x in time t (i.e., the probability that a particle does not react during the specified time interval) is $p_k(x, t | x_0)/p_{k=0}(x, t | x_0)$. Since both p_k and $p_{k=0}$ are known analytically, the survival probability is easily calculated. The algorithm proceeds exactly as in the nonreactive case, with individual diffusive jumps made according to the nonreactive distribution $p_{k=0}(x, t | x_0)$. After each jump, a "weight" is multiplied by the survival probability for that jump. This weight then measures, in the sense of the Schulten–Epstein[2] algorithm, that fraction of particles in an ensemble which have not yet reacted. For the data displayed in Fig. 1(b), the weight assigned to the particle at the end of its trajectory is accumulated in the appropriate endpoint bin [in Fig. 1(a) this weight was always unity]. This algorithm is both fast and simple and requires only a slight modification of that for the nonreactive case.

To demonstrate the validity of the algorithm, we followed the same procedure as in paper I. The endpoints of 10^4 one-jump trajectories were recorded and accumulated in the appropriate endpoint bin. To provide a reasonably accurate solution, the diffusion domain (0, 0.2 nm) was subdivided into 100 bins of equal size (0.002 nm). The number of endpoints in the bins was normalized as described in Appendix C to provide a MC discrete approximation to the probability distribution. As seen from Fig. 1(a), which shows results for diffusion in the constant force $F(x) = 50 \text{ nm}^{-1}$ near a reflective boundary, the statistical fluctuations at the peak are about $\pm 1 \text{ nm}^{-1}$, in agreement with Eq. (C6). Comparison of the MC results is made with the analytic Smoluchowski distribution (2.3) and with the numerical solution to the SDE obtained using a difference equation approach. The difference approach is essentially exact and has been presented in paper I.[16] A typical run of 10^4 trajectories took about 3 s CPU time on a CDC Cyber installation (single precision Fortran). Additional comments may be found in paper I.

For diffusion near a reactive boundary, we chose a reactivity of $k = 50 \text{ nm}^{-1}$. For this value of k, the boundary condition requires that the slope of the distribution at the boundary be zero; this is correctly reproduced in Fig. 1(b) by all three methods. The analytic distribution corresponds to Eq. (3.2). The influence of the boundary is slight, as a comparison of Figs. 1(a) and 1(b) shows. As the same sequence of pseudorandom numbers was used for both processes, the individual stochastic peaks in the two figures can be matched. The quantity of main interest is the number of reacting particles, that is, the reaction yield, in this case just one minus the area under the curve (Appendix D). For the three methods, we obtain yields of 0.1053 (MC), 0.106 97 (analytic), and 0.107 04 (difference method). Identical MC runs with different random number sequences will lead to reaction yields that fluctuate about the analytic value. For runs with a larger number of trajectories, the MC results will be more accurate, as Fig. 1(c) demonstrates (yield = 0.1075). The slight difference between the analytic and difference equation approaches is due to approximations inherent in the latter. For more physical reaction descriptions, the reactive boundary might be replaced by a reactive optical potential spreading out the reaction process over a finite region of space. The modifications of the algorithm necessary to treat this problem are derived in Sec. V with results presented in Sec. VII.

IV. LINEAR FORCE DIFFUSION NEAR A REACTIVE BOUNDARY

To extend the results of the previous two sections to diffusion in a linear force, we consider the solution to the corresponding SDE in the diffusion domain $(-\infty, \infty)$

$$\frac{\partial}{\partial t} p(x, t|x_0) = \left[\frac{\partial^2}{\partial x^2} + (b+cx)\frac{\partial}{\partial x} + c\right] p(x, t|x_0) , \qquad (4.1)$$

subject to initial condition (2.2). The boundary-free solution (the Ornstein-Uhlenbeck process[17]) may be written as

$$p_0(x, t|x_0) = \left[\frac{c}{2\pi(1-\theta^2)}\right]^{1/2} \times \exp\left\{\frac{-c}{2(1-\theta^2)}\left[x - x_0\theta + \frac{b}{c}(1-\theta)\right]^2\right\}, \qquad (4.2)$$

where $\theta = e^{-ct}$. We note that Eq. (4.2) may be obtained from Eq. (2.5) through the formal substitutions

$$x_0 \to x_0' = x_0\theta , \quad b \to b' = 2b/(1+\theta) , \quad t \to t' = (1-\theta^2)/2c . \qquad (4.3)$$

An approximate solution to Eq. (4.1) subject to the (reflective) boundary condition (2.3) may be found by using Eqs. (4.3) in Eqs. (2.4)–(2.7). The resulting modified Ornstein–Uhlenbeck distribution obeys Eqs. (2.2), (2.3), and (4.1) in the limit of small jump times t. Furthermore, it is exact far from the boundary ($x_0 \gg 0$) *and* yields the correct image solution when b vanishes. We have not investigated the full range of conditions for which this approximation adequately represents the exact solution.[18] The important point here is that an algorithm based on this approximate solution converges more rapidly than previously suggested MC algorithms. Furthermore, as substitutions (4.3) depend only on the jump time t, the $N_i(t|x_0)$ obtained in Eqs. (2.9)–(2.11) are still valid, provided Eqs. (4.3) are used for the various parameters. Thus, the jump endpoint formulas (2.13), (2.14), and (2.18) remain unchanged and the identical algorithm described for the constant force case may be applied to linear force diffusion.

For diffusion near a reactive boundary, Eq. (4.1) is solved subject to Eq. (3.1). An approximate solution may be obtained from the constant force solution (3.2) by supplementing substitutions (4.3) with some appropriate choice for the reactivity $k \to k'$. The boundary condition then satisfied by the approximate solution is

$$\frac{\partial}{\partial x} p(x, t|x_0) = (k' - b') p(x, t|x_0) , \quad \text{at } x = 0. \qquad (4.4)$$

This result suggests three choices for k':

$$k' = k , \qquad (4.5a)$$

$$k' = 2k/(1+\theta) , \qquad (4.5b)$$

$$k' = k + b' - b . \qquad (4.5c)$$

The first choice argues for preserving the correct boundary reactivity in the approximate solution; the second forces the correct boundary condition when $k = b$ while retaining for k' the same time dependence as b'; the third choice allows the correct boundary condition to be satisfied for all time and for all k and b. Unfortunately, the third choice leads to an unnormalized distribution in the unreactive case, since now $k' = b' - b \neq 0$. This may be amended by obtaining the reactive analog of Eq. (2.8). We find

$$N(b, k) \equiv N_k(t|x_0) = N_0(t|x_0) + N_1(t|x_0) + N_{k2}(t|x_0) , \qquad (4.6)$$

where $N_0(t|x_0)$ and $N_1(t|x_0)$ are as given in Eqs. (2.9) and (2.10) and

$$N_{k2}(t|x_0) = \frac{k - b/2}{k - b} \exp[-(x_0 - bt)^2/4t] \times \left[E\left(\frac{x_0 + (2k-b)t}{\sqrt{4t}}\right) - E\left(\frac{x_0 + bt}{\sqrt{4t}}\right)\right] , \qquad (4.7)$$

where we have defined

$$E(x) = e^{x^2} \operatorname{erfc}(x) . \qquad (4.8)$$

We can then renormalize the approximate solution by dividing it by $N(b', b' - b)$.

To determine which choice for k' to use, we need only compare the three analytic solutions with the results obtained using the difference equation method, since the MC results will necessarily converge to whichever analytic distribution we base the algorithm on. Figure 2(a) shows the results for diffusion in the linear force $F(x) = -20 - 400x$ near a reflective boundary. It is seen that choice (4.5c) is a significant improvement over the other choices. Unfortunately, this improvement does not carry over to reactive diffusion, as Fig. 2(b) demonstrates ($k = 20$ nm^{-1}). All three choices fail in giving a reasonable estimate of the number of surviving particles. What this means, in relation to the MC algorithm, is that the diffusive jump time of a particle is limited to times for which our choice for k' leads to a reasonably accurate solution. For unreactive boundaries, we could use choice (4.5c) for k' and choose jump times up to 5 ps (for the force parameters chosen above). However, for reactive boundaries Eq. (4.5) with jump times less than 1 ps would be needed. *In view of the simplicity of choice* (4.5b), *we have used this result throughout the remainder of the paper.* To demonstrate the algorithm, MC results for the reactive boundary process based on Eq. (4.5b) for k' and with a constant jump time of 0.1 ps are also displayed in Fig. 2(b). Note that, for this process, the zero slope of the distribution at the boundary is correctly (and stochastically) reproduced.

V. REACTIVE, NONLINEAR DIFFUSION

The algorithm developed in the preceding sections is applicable to diffusion in a linear force near a reactive boundary. The treatment of nonlinear diffusion is easily handled by limiting the duration of individual diffusive jumps to values such that the local force is approximately linear. Force and jump time parameters are then locally determined (Sec. VI) and many individual jumps strung together to give an accurate, long-

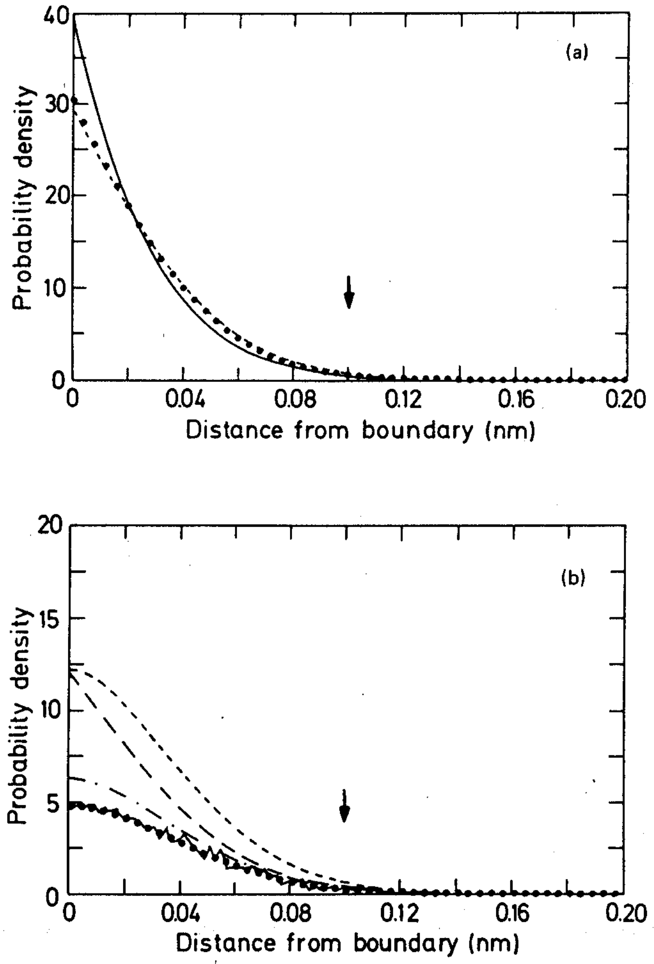


FIG. 2. (a) Results for the probability distribution $p(x, t = 0.005\ \text{nm}^2 | x_0 = 0.1\ \text{nm})$ for diffusion in the linear force $F(x) = -20 - 400x\ \text{nm}^{-1}$ near a reflective boundary (at $x = 0$). The analytical expression (3.2) with substitutions (4.3) and (4.5b) (—) or (4.5c) (– –) is compared with the difference method results (•••). (b) The process of (a) but with a reactive boundary of reactivity $k = 20\ \text{nm}^{-1}$. The analytical expression (3.2) with substitutions (4.3) and (4.5a) (— —), (4.5b) (–· –), or (4.5c) (– – –) is compared with the results of the difference method (•••). The MC distribution consisting of 10^4 50-jump trajectories based on the analytic expression using Eq. (4.5b) is also given (—). For further details refer to the caption for Fig. 1(a); the MC algorithm is described in Sec. IV.

time nonlinear diffusion trajectory. Such multijump trajectories may also be necessary for linear diffusion near a reactive boundary, as Fig. 2(b) implies.

When an optical potential is used to describe reaction conditions, the algorithm must be modified to account for the finite region over which reaction can take place. Unfortunately, only for an optical potential of uniform strength and infinite extent is an analytic solution available. For more realistic potentials, some approximation must be made. In this section, a suitable correction to the modified Ornstein–Uhlenbeck distribution of Sec. IV is derived that not only accounts for the finiteness of the optical potential, but also includes a correction term describing the nonlinearity of the applied force. This correction term vanishes in the limit of small jump times but could improve the convergence rate of the algorithm when applied to nonlinear processes. The actual utility of this correction will be discussed in Sec. VII where results for nonlinear, reactive diffusion are presented.

The algorithm developed so far is based on the exact analytic solution to the SDE[19]

$$\frac{\partial}{\partial t} p_0(x, t | x_0) = \frac{\partial}{\partial x}\left[\frac{\partial}{\partial x} - F_0(x)\right] p_0(x, t | x_0) , \tag{5.1}$$

where the nonlinear force $F(x)$ has the local linear form

$$F_0(x) = F(x_0) + (x - x_0)F'(x_0) . \tag{5.2}$$

The general SDE including a reactive optical potential we want to solve is

$$\frac{\partial}{\partial t} p(x, t | x_0) = \frac{\partial}{\partial x}\left[\frac{\partial}{\partial x} - F(x)\right] p(x, t | x_0) - k(x) p(x, t | x_0) , \tag{5.3}$$

where $k(x)$ describes the reaction rate throughout the diffusion domain. $p_0(x, t | x_0)$ and $p(x, t | x_0)$ are assumed to obey initial condition (2.2). To simplify discussion, we assume no boundary is present, although the derivation does not rely on this. We may obtain a relation between $p(x, t | x_0)$ and $p_0(x, t | x_0)$ by substituting

$$p(x, t | x_0) = q(x, t | x_0) \exp\left[\frac{1}{2} \int_{x_0}^{x} dx' F(x')\right] \tag{5.4}$$

into Eq. (5.3) to get

$$\frac{\partial q(x, t | x_0)}{\partial t} = \left[\frac{\partial^2}{\partial x^2} - \phi(x) - k(x)\right] q(x, t | x_0) , \tag{5.5}$$

where

$$\phi(x) = \tfrac{1}{4} [F(x)]^2 + \tfrac{1}{2} F'(x) . \tag{5.6}$$

The adjoint SDE corresponding to Eq. (5.2) may be written as[20]

$$\frac{\partial p_0(x_f, T | x, t)}{\partial T} = \left[\frac{\partial}{\partial x} + F_0(x)\right] \frac{\partial}{\partial x} p_0(x_f, T | x, t) . \tag{5.7}$$

Substituting into this equation the analog of Eq. (5.4)

$$p_0(x_f, T | x, t) = q_0(x_f, T | x, t) \exp\left[\frac{1}{2} \int_{x}^{x_f} dx' F_0(x')\right] \tag{5.8}$$

gives

$$-\frac{\partial q_0(x_f, T | x, t)}{\partial t} = \left[\frac{\partial^2}{\partial x^2} - \phi_0(x)\right] q_0(x_f, T | x, t) , \tag{5.9}$$

where

$$\phi_0(x) = \tfrac{1}{4} [F_0(x)]^2 + \tfrac{1}{2} F_0'(x) . \tag{5.10}$$

Finally, multiplying Eq. (5.5) by $q_0(x_f, T | x, t)$, Eq. (5.9) by $q(x, t | x_0)$, subtracting the resulting equations, and integrating over x and t leads to the integral equation[21]

$$\begin{aligned} p(x, t | x_0) = {} & \exp[\zeta(x, x_0)] p_0(x, t | x_0) \\ & - \int_0^t dt' \int_0^\infty dx' \exp[\zeta(x, x')] \\ & \times p_0(x, t | x', t') \phi_k(x', x_0) p(x', t' | x_0) , \end{aligned} \tag{5.11}$$

where

$$\zeta(x, x_0) = \frac{1}{2} \int_{x_0}^{x} dx' [F(x') - F_0(x')] \qquad (5.12)$$

and

$$\phi_k(x', x_0) = \phi(x') - \phi_0(x') + k(x') . \qquad (5.13)$$

Equation (5.11) may be viewed as the diffusion theory equivalent to the Lippmann–Schwinger equation in scattering theory.[22] Both $p(x, t|x_0)$ and $p_0(x, t|x_0)$ obey the same initial and boundary conditions. Equation (5.11) is exact if p_0 is exact but is too difficult to solve for p unless $\phi_k(x, x_0)$ is constant everywhere. However, we can obtain a first-order correction by evaluating the time integral in Eq. (5.11) using the trapezoidal rule. The resulting integrations reduce to integrals over delta functions and can be integrated exactly [as in the derivation of Eq. (5.11)]. We obtain the approximation

$$p(x, t|x_0) = \exp[\xi(x, t|x_0)] p_0(x, t|x_0) , \qquad (5.14)$$

where

$$\xi(x, t|x_0) = \zeta(x, x_0) - \frac{t}{2} [\phi(x) - \phi_0(x) + k(x) + k(x_0)] . \qquad (5.15)$$

The first term in Eq. (5.15) corrects p_0 for the large spatial jumps that can occur even for short jump times.

To implement the correction factor in Eq. (5.14) in the algorithm, we first note that $\xi(x, t|x_0)$ is at least cubic in x and so cannot be combined with p_0 to modify the jump endpoint formulas of Sec. II. For a linear force but with a spatially varying optical potential $k(x)$, the correction factor takes the form of a survival probability as discussed in the previous section. Indeed, Eqs. (5.14) and (5.15) lead to the modified, nonlocal survival probability

$$p_k(x, t|x_0)/p_{k=0}(x, t|x_0) = \exp[-\frac{t}{2}(k(x_0) + k(x))] . \qquad (5.16)$$

For an optical potential of uniform strength k and infinite extent, Eq. (5.16) becomes

$$p_k(x, t|x_0) = \exp(-kt) p_{k=0}(x, t|x_0) \qquad (5.17)$$

and is the exact solution to Eq. (5.11). Such a solution agrees with the results of first-order rate theory in which the number of particles obtained by integrating Eq. (5.17) over the diffusion domain decreases exponentially in time. The Schulten–Epstein[2] algorithm was based on the purely local approximation to the reaction rate

$$p_k(x, t|x_0) = \exp[-k(x_0)t] p_{k=0}(x, t|x_0) . \qquad (5.18)$$

In fact, Eq. (5.16) might have been suggested as the logical first-order improvement over the purely local form of Eq. (5.18), but any such argument based on a constant velocity diffusive jump is not strictly valid. Equation (5.16) has been found to be a significant improvement over Eq. (5.18) and for slowly varying optical potentials is surprisingly accurate (see Sec. VII for a discussion of results).

For a nonlinear force (with or without reaction) we use the correction factor in Eq. (5.14) as a "weight" for each jump, exactly as the survival probability of Sec. III was used.[23] If both a reactive boundary and nonlinear force are present, the correction factor in Eq. (5.14) is multiplied by the survival probability $p_k/p_{k=0}$ of Sec. III to yield the weight for the individual jump. A slight complication arises because distribution (5.14) is no longer normalized. For nonreactive diffusion, renormalization is easily performed after all trajectories are run. For reactive diffusion, it is necessary to accumulate the nonlinear correction terms in Eq. (5.15) separately from the reactive correction terms during a trajectory simulation. This then allows renormalization as in the nonreactive case. In Sec. VII, we discuss whether the time spent in evaluating the nonlinear corrections justifies their inclusion in the algorithm.

So far, we have avoided mention of the specific boundary conditions satisfied by $p(x, t|x_0)$ and $p_0(x, t|x_0)$. If $p_0(x, t|x_0)$ is chosen to satisfy Eq. (4.4) then, from the discussion of Sec. IV, it is clear that, in general, $p(x, t|x_0)$ will not obey Eq. (3.1) with b replaced by $-F(0)$. The same argument, as presented there, again leads to three choices for an effective reactivity k', but with Eq. (4.5c) replaced by

$$k' = k + b' + F(0) - \left.\frac{\partial \xi(x, x_0)}{\partial x}\right|_{x=0} . \qquad (5.19)$$

Although this last choice should improve the solution for a nonreactive boundary in the same manner as the linear solution was improved, its extension to a reactive boundary ($k \neq 0$) will also fail. We have thus used Eq. (4.5b) for the effective boundary reactivity in all cases.

The entire algorithm is an importance sampling scheme[24] based on the linear distribution $p_0(x, t|x_0)$. If the force is exactly linear, then each path contributes the same unit weight (for the nonreactive case) and the algorithm is as efficient as possible. If the force is nonlinear, each path will have a different final weight depending on the size of the nonlinear deviations along it. This weight gives the relative contribution of the path to the total nonlinear distribution. The formal solution to Eq. (5.11) may be written in the suggestive path integral form[25]

$$p(x, t|x_0) = \exp[\zeta(x, x_0)] \int_{x_0}^{x(t)} d[p_0(x, t|x_0)]$$

$$\times \exp\left[-\int_0^t dt' \phi_k([x(t')], x_0)\right] , \qquad (5.20)$$

where the stochastic measure $d[p_0(x, t|x_0)]$ indicates that paths are chosen from distribution p_0. The path-dependent argument of the exponential is evaluated by constructing long-time trajectories from many short-time segments. This is equivalent to repeatedly applying the Chapman–Kolmorogov equation[26]

$$p(x, t|x_0) = \int_0^\infty dx' p(x, t|x', t') p(x', t'|x_0) \qquad (5.21)$$

$$= \prod_{j=1}^{J} \int_0^\infty dx_j p(x_{j+1}, t_{j+1}|x_j, t_j) p(x_1, t_1|x_0) , \qquad (5.22)$$

where $(x_J, t_J) = (x, t)$, until all $p(x_{j+1}, t_{j+1}|x_j, t_j)$ may be accurately approximated by $p_0(x_{j+1}, t_{j+1}|x_j, t_j)$ in Eq. (5.14). The specific time duration of these segments is discussed in the following section and results of the algorithm for nonlinear diffusion are presented in Sec. VII.

VI. CHOICE OF LOCAL JUMP TIMES FOR NONLINEAR FORCES

As discussed in the introduction, the convergence rate of the algorithm depends on the number of trajectories sampled N and on the size of the individual particle jump times. The statistical fluctuation of the results depends only on N and converges as $N^{-1/2}$ (see Appendix C). In the limit of large N these statistical results converge to results that depend in part on the jump times chosen: these may *not* be accurate compared to the exact results. For a fixed jump time t, the particle trajectories sample the applied force and reaction rate at a discrimination level of $\langle x^2\rangle^{1/2} \sim 2t$. To observe fine structure in the force or reaction rate, one must decrease the jump time accordingly. When computed results do not change appreciably with a large change in the jump time, the algorithm has converged (for a fixed N). To automatically insure convergence of the algorithm, either a sufficiently small, constant jump time can be set prior to trajectory calculations, or the jump time can be determined by local force and reaction rate conditions. The latter method takes advantage of regions in which the force is nearly linear and the reaction rate nearly constant and may be necessary for calculations in three dimensions where the available diffusion space is large. In this section, we relate our choice of local jump times to limitations in the algorithm placed upon it by the use of exact, analytic solutions. We also present results for the reaction yield for linear diffusion near a boundary to demonstrate convergence properties of the algorithm.

Three assumptions based on the use of analytic solutions of the SDE lead to errors in the algorithm: (1) a local constant force near the boundary; (2) a local linear force far from the boundary; and (3) a local constant reaction rate. We choose a local jump time by expressing the errors involved in the above assumptions in terms of an "average" jump distance Δ, which is then related to the jump time. Actually, *the* error in the algorithm will depend on what quantity is being calculated, so we cannot treat this problem exactly in the mathematical sense. This is, however, not necessary. What is actually required is *some* measure of the error that monotonically decreases to zero as the jump time decreases, even if the error itself is unknown. The measure we have used may not be optimal but it does provide a convenient way of assessing and controlling the error made in calculations.

For jumps near the boundary, we can decrease the error in the algorithm due to assumption (1) by requiring that most particles be distributed (or partitioned) into $p_0(x,t|x_0)$. Thus, for a jump from x_0 to $x_0 \pm \Delta$, we choose an average jump distance Δ to be less than the distance to the boundary, that is, we set

$$\Delta < x_0 . \tag{6.1}$$

The specific relation of Δ to the jump time will be given below. For assumption (2), a possible choice for the measure of the error incurred by the algorithm could be the quantity

$$\epsilon = \left|\frac{\Delta^2 F''(x_0)}{F(x_0)}\right| . \tag{6.2}$$

For a preset value of ϵ, the maximum allowed jump distance will be restricted by the local nonlinearity of the force. If the force is locally linear, no constraint is placed upon Δ. Similarly, assumption (3) yields the restriction

$$\epsilon = \left|\frac{\Delta k'(x_0)}{k(x_0)}\right| \tag{6.3}$$

for a chosen ϵ. For simplicity we have introduced the same parameter ϵ in both Eqs. (6.2) and (6.3). ϵ is used only to determine an appropriate jump distance and not as a measure of convergence (to be discussed below). We will set the value $\epsilon = 0.01$ with the understanding that smaller nonlinear deviations of the force are unmeasurable. The desired jump distance can then be selected to be the minimum value obtained from Eqs. (6.1)–(6.3). We have used the slightly more stringent result

$$\frac{1}{\Delta} = \frac{1}{\Delta_M} + \left|\frac{F''(x_0)}{\epsilon F(x_0)}\right|^{1/2} + \left|\frac{k'(x_0)}{\epsilon k(x_0)}\right| , \tag{6.4}$$

where $\Delta_M \leq x_0$ (see below).

The stochasticity of the process prevents us from assigning a jump time t that will result in a fixed jump distance Δ. We can, however, insure that the probability that the jump distance will be larger than Δ is small. For jumps made according to the boundary-free distribution $p_0(x,t|x_0)$, we choose t such that for some fixed $\epsilon' < 1$,

$$1 - \epsilon' = \int_{x_0-\Delta}^{x_0+\Delta} dx\, p_0(x,t|x_0) . \tag{6.5}$$

In the absence of forces, Eq. (6.5) gives the desired jump time

$$t_0 = \left(\frac{\Delta}{2E}\right)^2 , \tag{6.6}$$

where we have defined the *convergence parameter* $E = \mathrm{erf}^{-1}(1-\epsilon')$. E thus serves to limit the number of jumps larger than Δ and, hence, to limit the error incurred in the algorithm and it does so monotonically (as a function of t). For a constant force $F(x) = -b$, the diffusive drift makes the integrand asymmetric about x_0 so the time t can only be estimated. A tighter restriction than Eq. (6.5) leads to

$$1 - \epsilon' = \mathrm{erf}\left(\frac{\Delta - |bt|}{\sqrt{4t}}\right) , \tag{6.7}$$

which gives the jump time

$$t_b = t_0 \left(\frac{\sqrt{1+2\sigma} - 1}{\delta}\right)^2 , \tag{6.8}$$

where $\delta = \Delta|b|/(2E^2)$, valid for all b. An expression for the jump time in a linear force has proved more difficult to obtain. We have finally settled on

$$t_c = |2c|^{-1} \ln(1 + |2ct_b|) \tag{6.9}$$

TABLE I. Convergence of the MC reaction yield for reactive linear diffusion using fixed-time diffusive jumps.[a]

Δt (ps)	Yield	Error[b]	CPU time (s)
1.00	0.7484	−0.0076	120
0.50	0.7498	−0.0062	230
0.20[c]	0.7513	−0.0047	560
	0.7531	−0.0029	
	0.7523	−0.0037	
	0.7516	−0.0044	
0.10[c]	0.7517	−0.0041	1090
	0.7551	−0.0007	
	0.7532	−0.0028	
	0.7515	−0.0043	

[a]The process is diffusion in the linear force $F(x) = -20-400x$ nm^{-1} near a boundary (at $x=0$) of reactivity $k=20$ nm^{-1}; each run consists of 10^5 trajectories of total time duration 1 ps; the particle is initially placed at $x_0 = 0.1$ nm.
[b]Relative to the difference method yield of 0.7558 extrapolated beyond 200 subdivisions of the diffusion domain (0, 0.2 nm) (see Appendix C of paper I).
[c]Four runs with different random number seeds were performed.

based on Eq. (4.3), where t_b is given by Eq. (6.8) but with

$$\delta = \frac{\Delta\,|b+cx_0|}{2E^2} . \tag{6.10}$$

A few comments need to be made concerning the use of Eq. (6.4) in Eqs. (6.6)–(6.10). Since Eq. (6.4) is intended for use in all applications of the algorithm, with $\epsilon = 0.01$ fixed, regions of the diffusion space where nonlinear deviations of the force are large could lead to extremely small values for Δ. This, in turn, would yield small jump times and the algorithm would stagnate, i.e., the trajectory would make repeated short-time jumps in the same region. To avoid this, one could decrease the value chosen for E but this must be done *a posteriori*. It is more convenient to restrict values of Δ:

$$\Delta_m \le \Delta \le \Delta_M . \tag{6.11}$$

The fact that this will make the error larger is of little consequence since the error is to be estimated from successive runs with different E values. Furthermore, as Δ and E are coupled in the determination of the jump time [see Eqs. (6.6) and (6.10)], a universal value for Δ_m can be assigned. The maximum allowable jump distance Δ_M would normally be assigned the value x_0 to decrease the error near the boundary. It may also be used to account for the presence of sharp features in optical potentials (Sec. VII).

In Tables I–V, we present different aspects of the convergence of the algorithm. The process we have chosen is diffusion in the linear force $F(x) = -20-400x$ nm^{-1} near a boundary of reactivity $k = 20$ nm^{-1}. The distribution after 5 ps is illustrated in Fig. 2(b), however, we will focus attention on the reaction yield. As discussed in Appendix C, the yield is a quantity of direct physical interest and converges faster stochastically than the distribution. We compare the MC yield values with those obtained from the difference equation method. To obtain sufficiently accurate values for the latter, we

TABLE II. Convergence of the MC reaction yield for reactive linear diffusion using variable-time diffusive jumps with the average jump distance Δ fixed.[a]

Δ (nm)	Yield	Error[b]	CPU time (s)
0.100	0.7407	−0.0151	120
0.05	0.7454	−0.0104	170
0.02	0.7481	−0.0077	320
0.01[c]	0.7505	−0.0053	560
	0.7501	−0.0057	
0.005[c]	0.7531	−0.0027	1030
	0.7523	−0.0035	

[a]Additional parameters: $E=0.01$; see also Table I, footnote a.
[b]Relative to the difference method yield of 0.7558 extrapolated beyond 200 subdivisions of the diffusion domain (0, 0.2 nm) (see Appendix C of paper I).
[c]Two runs with different random number seeds were performed.

TABLE III. Effect of the minimum jump distance Δ_m on the accuracy of the MC reaction yield.[a]

Δ_m (nm)	Yield[b]	Error[c]	CPU time (s)
0.01	0.7537	−0.0021	230
	0.7538	−0.0020	
0.001	0.7547	−0.0011	260
	0.7550	−0.0008	
0.0001	0.7548	−0.0010	260
	0.7552	−0.0006	

[a]Additional parameters: $E=0.01$, $\Delta_m < \Delta < 0.1$ nm; see also Table I, footnote a.
[b]Two runs with different random number seeds were performed for each value of Δ_m.
[c]Relative to the difference method yield of 0.7558.

TABLE IV. Effect of the maximum jump distance Δ_M on the accuracy of the MC reaction yield.[a]

Δ_M (nm)	Yield[b]	Error[c]	CPU time (s)
1.00	0.7546	−0.0012	260
	0.7549	−0.0009	
0.10	0.7547	−0.0011	260
	0.7550	−0.0008	
0.05	0.7542	−0.0016	280
	0.7532	−0.0026	
0.01	0.7533	−0.0025	620
	0.7524	−0.0034	

[a]Additional parameters: $E=0.01$, 0.001 nm $< \Delta < \Delta_M$; see also Table I, footnote a.
[b]Two runs with different random number seeds were performed for each value of Δ_M.
[c]Relative to the difference method yield of 0.7558.

TABLE V. MC reaction yield for reactive linear diffusion as a function of the convergence parameter E.[a]

E[b]	Yield	Error[c]	CPU time (s)
0.001	0.7544	−0.0014	250
	0.7545	−0.0013	
0.010	0.7547	−0.0011	260
	0.7550	−0.0008	
0.100	0.7545	−0.0013	330
	0.7543	−0.0014	
0.200	0.7568	0.0010	460
	0.7560	0.0002	
0.500	0.7579	0.0021	1650
	0.7560	0.0002	

[a]Additional parameters: 0.001 nm$<\Delta<$0.1 nm: see also Table I, footnote a.
[b]Two runs with different random number seeds were performed for each value of E.
[c]Relative to the difference method yield of 0.7558 extrapolated beyond 200 subdivisions of the diffusion domain (0, 0.2 nm) (see Appendix C of paper I).

have noticed that as the number of subdivisions M of the diffusive domain (0, 0.2 nm) becomes large, the difference method yield varies linearly with $1/M$. A least squares fit for M up to 200 for the above process gives a yield of 0.7558.

In Table I, we show the MC yield obtained with the algorithm employing fixed time jumps. This would be equivalent with Ermak's algorithm[1] except that, in our simulation, the diffusion particle never strikes the boundary. The error in the yield is attributed to the approximation made in accounting for the boundary, i.e., the analytic linear distribution used to distribute the jumps is not an adequate solution to the process near the boundary. We have chosen a strongly reactive process to emphasize boundary effects. It is seen from Table I that by choosing smaller jump steps, the error decreases [from below, as expected from the discussion pertaining to Fig. 2(b)] and the CPU time accordingly increases. For a jump time of 0.1 ps, the convergence rate is slower because stochastic fluctuations due to the finite sample size (10^5 trajectories) are about as large as the error.

The algorithm employing variable-time diffusive jumps produces the yield values given in Table II. We have fixed the average jump distance Δ and allowed the jump time to vary. For the force parameters chosen, Eqs. (6.6)–(6.10) become approximately

$$t_c(\text{ps}) = \tfrac{5}{4}\ln\left(1+\frac{2\Delta}{x+0.05\ \text{nm}}\right) \tag{6.12}$$

if E is chosen much less than 1. Setting $E=0.01$ then leaves Δ as the single variable affecting the jump time. A comparison of Tables I and II shows that the two variations of the algorithm are roughly equivalent in regards to accuracy and to speed. This is, of course, to be expected but this latter (variable-time) algorithm is readily extended to allow jump times to be locally determined.

In Tables III–V, we present results for the MC yield using the full variable-time algorithm. The jump distance Δ is now locally determined from Eq. (6.4); i.e., from

$$\Delta = x_0\ . \tag{6.13}$$

It has been found necessary to further restrict Δ to a finite range (Δ_m, Δ_M). For Table III, we have kept Δ within the range $(\Delta_m, 0.1\ \text{nm})$. (In Table IV, we will investigate the effect of imposing an upper bound on Δ.) We notice first that as Δ_m is decreased, the error decreases and there is a slight increase in the CPU time required. Beyond $\Delta_m = 0.001$ nm, neither the accuracy nor the speed of the algorithm is affected. For the process chosen, all jump distances are larger than this minimum value. It is, however, not necessary to determine an appropriate value for Δ_m for each process investigated. In setting some finite Δ_m we are, in effect, smearing out fine structure in the force or reaction field at scales smaller than this. It is clearly inappropriate to investigate a diffusion process at a level smaller than say 0.01 nm. Of course, if the SDE to be solved is only mathematically related to a diffusion process, then some other minimum value for Δ_m may be appropriate. In the following, we have set Δ_m to 0.001 nm because we are investigating the SDE in a pure mathematical sense.

In Table IV, we have varied the upper limit of the jump displacements Δ_M, where the jump distance is restricted to the range $(0.001\ \text{nm}, \Delta_M)$. The effect of Δ_M on the algorithm becomes noticeable below 0.05 nm, where both the speed and the accuracy of the algorithm begin to change. For the process considered, there is no need to impose a restriction on the maximum jump distance, but in Sec. VII, in connection with reactive optical potentials, we will see that such a restriction can be quite useful.

From the results of Tables III and IV, limiting the jump distance Δ to the range (0.001, 0.1 nm) has essentially no effect on either the accuracy or the speed of the algorithm. With these fixed choices for Δ_m and Δ_M the jump distance is entirely locally determined and is coupled to the jump time determination only through the choice for the (convergence) parameter E. In Table V, we present results of the full MC algorithm as a function of this parameter. For values of E much less than unity, the algorithm is roughly independent of E, as already noted. Near $E=0.2$, the increase in CPU time indicates that this parameter now has some effect, although for the sample size chosen, an increase in accuracy is difficult to detect. Essentially, the algorithm has converged at the smallest value of E and an increase in E merely increases the running time of the algorithm. For the linear process investigated, the error in the algorithm is satisfactorily accounted for in the jump time assignment Eq. (6.12), which is independent of E. In Sec. VII, we will investigate a nonlinear process in which this is not so and the effect of E on the distribution will be significant.

Finally, we discuss briefly how the force parameters b and c are determined for a nonlinear force. Since the local force is assumed to have the form $-b-cx$, one requirement is clearly

$$F(x_0)=-b-cx_0 \tag{6.14}$$

for a particle jumping from x_0. Perhaps the most obvious additional condition is to demand that the approximation to the force be quadratically correct about x_0. This was implicitly assumed in Sec. V and implies that we set

$$c=-F'(x_0)\ , \quad b=-F(x_0)+x_0F'(x_0)\ . \tag{6.15}$$

These are, however, not the only logical choices one could make. The analytic solution for a constant force assumes that the value of the force at x_0 is the same as that at the boundary. By choosing b locally, we necessarily obtain a solution that does not, in general, obey the correct boundary condition, as discussed in the previous section. (It is, in fact, not even correct at short times, however, the error is insignificant, as most particles will be partitioned into the boundary-free part of the distribution.) Requiring, instead of Eq. (6.15), $b=-F(0)$ leads to $c=[F(x_0)-F(0)]/x_0$, the nonlocal difference approximation to Eq. (6.15). For a quadratic force, this choice for b and c may actually improve the distribution of jumps toward the boundary. For simplicity, and because any error inherent in the use of Eq. (6.15) is likely to be insignificant, we have always chosen the force parameters locally.

VII. RESULTS FOR REACTIVE, NONLINEAR DIFFUSION

In this section, we present and discuss results for linear diffusion in a variety of reactive optical potentials, and for nonreactive diffusion in a quartic (double-well) potential.

For linear diffusion near a reactive boundary, we found in Sec. IV that the process at short times may be analytically described by a modified Ornstein–Uhlenbeck distribution. The use of a reactive boundary implies that reaction can take place only when a particle impinges on the boundary. Particles do not have to reach the boundary explicitly at the end of a jump as the algorithm accounts for boundary contacts during a diffusion jump. More realistic reaction conditions may require that reaction extends over a finite region of space. For this purpose, one introduces an optical potential; the strength of this potential is chosen on the basis of available experimental data. The specific shape of the potential, i.e., the extension of the reaction domain, is generally not known but some reasonable form can be used with the assumption that the reaction yield is relatively insensitive to the extent of the reaction domain.[27] We can test this assumption by means of the algorithm prescribed here.

Consider some function $k(x)$ for the reaction rate constant (the optical potential) at each point in the diffusion domain. Kinetic data will normally provide a value for the "total reaction strength"

$$k_r=\int_0^\infty dx\, k(x)\ , \tag{7.1}$$

where the integral is evaluated over the entire diffusion domain. We may thus compare the yields obtained using different functional forms for $k(x)$ by requiring that each form give the same total strength (7.1). The various forms chosen for comparison are:

(i) a delta function

$$k(x)=\begin{cases}k_r\ , & x=0\\ 0\ , & x\neq 0\ ;\end{cases} \tag{7.2}$$

(ii) a finite step function

$$k(x)=\begin{cases}k_r/\lambda\ , & 0\le x\le\lambda\\ 0\ , & x>\lambda\ ;\end{cases} \tag{7.3}$$

(iii) an exponential function

$$k(x)=\frac{k_r}{\lambda}\exp(-x/\lambda)\ ; \tag{7.4}$$

(iv) a Gaussian function

$$k(x)=\frac{2k_r}{\sqrt{\pi}\lambda}\exp(-x^2/\lambda^2)\ ; \tag{7.5}$$

and (v) a finite linear function

$$k(x)=\begin{cases}\dfrac{2k_r}{\lambda}\left(1-\dfrac{x}{\lambda}\right), & 0\le x\le\lambda\\ 0\ , & x>\lambda\ .\end{cases} \tag{7.6}$$

Functions (7.3)–(7.6) require the further assignment of the length parameter λ. For $k_r=20\ \text{nm}^{-1}$, Table VI lists the values of λ used. For the finite step function (7.3), we have investigated two values for λ and for the exponential and Gaussian functions of Eqs. (7.4) and (7.5) have chosen λ such that 92% of the total reactive strength lies in the region (0, 0.05 nm). For the underlying diffusion process we assume the linear force $F(x)=-20-400x\ \text{nm}^{-1}$.

The delta function potential was investigated by the method described in Sec. IV. To improve the accuracy of the algorithm near the sharp border of the reaction domain at $x=\lambda$ for forms (7.3) and (7.6), the maximum jump distance Δ_M discussed in the preceding section was set equal to the minimum of $x_0|x_0-\lambda|$, and 0.1 nm. We have not varied the convergence parameter E to obtain convergence in the yield values. We have assigned E the value 0.01 in order to show the improvement obtained by using Eq. (5.16) instead of Eq. (5.18) for the survival probability. The different forms for the optical potentials are presented in Fig. 3(a).

In Fig. 3(b), we present the difference method distributions obtained for the various forms for the optical potentials. Differences in the distributions are most noticeable near the boundary where the influence of the optical potential is most strongly felt. The area under the distribution curves gives the respective yields for the processes and these values are compared in Table VI. Comparison of columns 3 and 4 shows the improvement obtained by using the nonlocal expression (5.16) for the survival probability over the local form. This improvement is most significant for the finite step function potentials. The differing forms for the optical potentials lead to yields that vary by only 2%–5%. It

TABLE VI. Results for the MC reaction yield for linear diffusion in various reactive optical potentials. For details refer to the caption for Fig. 3 and to the text.[g]

Potential parameters		MC yield				
k_r (nm^{-1})	λ (nm)	Eq. (5.18)	Eq. (5.16)	Difference yield[h]	Error	CPU time (s)
20[a]	...	...	0.7586[i]	0.7558	0.0028	25
20[b]	0.02[j]	0.7012	0.7582	0.7573	0.0009	40
20[c]	0.05[j]	0.7035	0.7207	0.7154	0.0053	35
20[d]	0.02	0.7455	0.7442	0.7419	0.0023	430
20[e]	0.04	0.7347	0.7342	0.7305	0.0037	420
20[f]	0.05[j]	0.7316	0.7532	0.7473	0.0059	220

[a]Reactive boundary.
[b]Finite step function, Eq. (7.3).
[c]Finite step function, Eq. (7.3).
[d]Exponential function, Eq. (7.4).
[e]Gaussianfunction, Eq. (7.5).
[f]Linear function, Eq. (7.6).
[g]Additional parameters: 10^4 trajectories per run, $E=0.01$, $0.001\ \text{nm}<\Delta<0.1\ \text{nm}$, $F(x)=-20-400x\ \text{cm}^{-1}$.
[h]Extrapolated beyond 200 subdivisions of the diffusion domain (0, 0.2 nm).
[i]Calculated using the survival probability discussed in Secs. III and IV.
[j]The maximum jump distance Δ_M depends on x_0 and λ as discussed in Sec. VII.

would thus appear that, unless the total yield is accurately known, it is not necessary to treat the form of the potential in detail. On the basis of speed, we recommend that the delta function potential be used to describe reactive diffusion when this MC algorithm is applied. More sophisticated forms merely slow down the algorithm (by requiring shorter jump times for a given E) without significantly affecting the yield.

To illustrate the application of the algorithm to nonlinear diffusion, we have investigated diffusion in the quartic potential

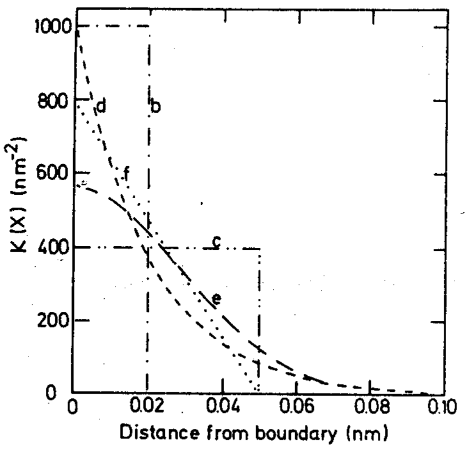
$$V(x)=-\frac{a_1}{2}(x-0.1)^2+\frac{a_2}{4}(x-0.1)^4 . \tag{7.7}$$

This potential has a local maximum at 0.1 nm and two minima at $x=0.1\ \text{nm}\pm\sqrt{a_1/a_2}$. The force is given by $F(x)=-V'(x)$. To sufficiently confine the distribution within the domain (0, 0.2 nm), we have chosen the parameter values

$$a_1=2\times10^3\ \text{nm}^{-2} ,\qquad a_2=4\times10^4\ \text{nm}^{-4} . \tag{7.8}$$

The particle is initially placed at $x_0=0.1$ nm at $t=0$. We would expect to observe a double-peaked distribution develop in time. In Figs. 4(a)–4(c), we show the distribution after 1 ps as a function of the convergence parameter E. Convergence occurs throughout the diffusive domain and for $E=0.1$ the MC distribution is sufficiently accurate. In Fig. 5, we show the time development of the distribution. The nearly Gaussian distribution at short times broadens as the particles sample more of the diffusion space and gradually feel the effects of the nonlinear force. At $t=0.6$ ps the onset of two peaks occurs and at longer times the particles tend to remain in one of the two valleys.

We have used this process to test the effectiveness of the nonlinear correction term in Eqs. (5.14) and (5.15) by including the correction term and repeating the above run with $E=0.02$ and $E=0.05$. The corrected algorithm



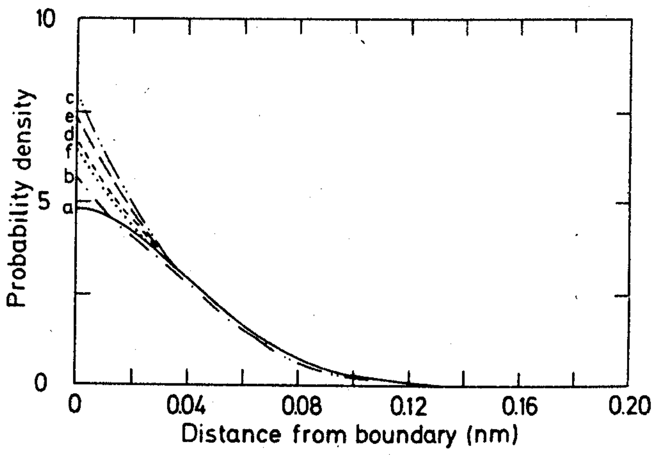
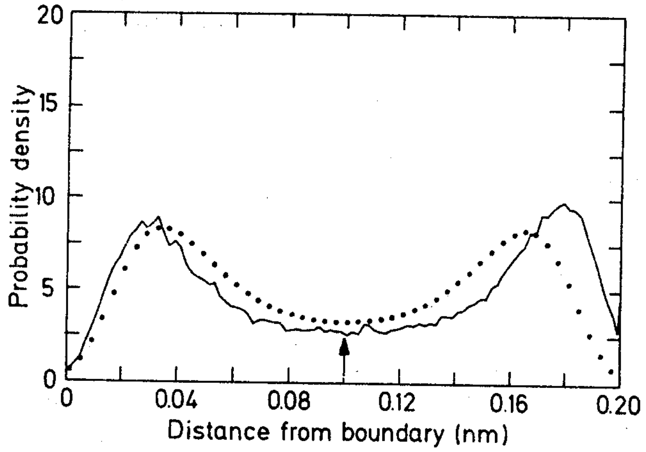
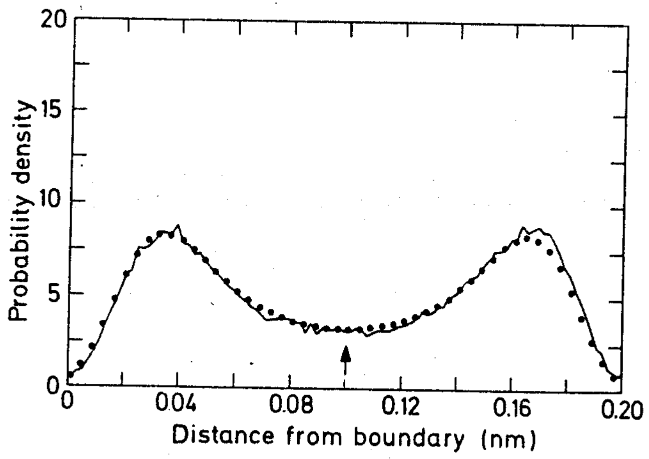


FIG. 3. (a) The various optical potentials of Eqs. (7.3)–(7.6) using the parameter values of Table VI (b–f). Refer to Table VI and to the text for further details. (b) Results for the probability distribution $p(x,t=0.005\ \text{nm}^2\,|\,x_0=0.1\ \text{nm})$ for diffusion in the linear force $F(x)=-20-400x\ \text{nm}^{-1}$ in the reactive optical potentials displayed in (a) (curves b–f) and near a reactive boundary (curve a). The distributions were obtained using the difference method approach discussed in paper I.

is about 10% slower than the uncorrected one with no noticeable improvement. In view of the added difficulties in programming the correction factor, its use is, therefore, not recommended or required in the algorithm.





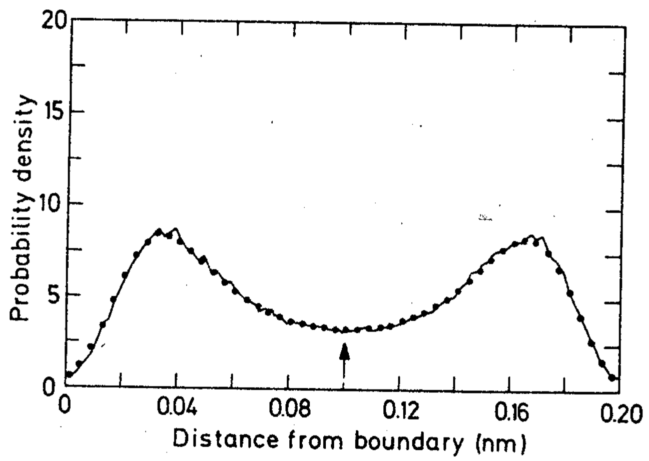


FIG. 4. (a) Results for the probability distribution $p(x, t = 0.001\ \mathrm{nm}^2 | x_0 = 0.10\ \mathrm{nm})$ for diffusion in the bistable potential (7.7). The MC algorithm with 10^5 trajectories and convergence parameter $E = 0.02$ (—) is compared with the difference equation approach (• • •). (b) Same as (a) except with $E = 0.05$. (c) Same as (a) except with $E = 0.10$.

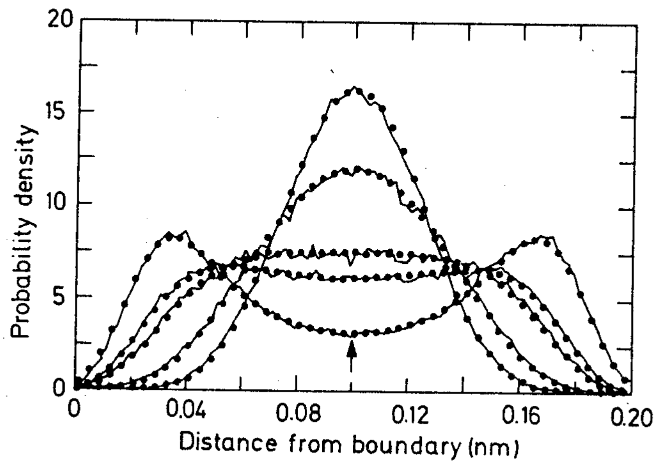


FIG. 5. Results for the probability distribution $p(x, t | x_0 = 0.1\ \mathrm{nm})$ for diffusion in the bistable potential (7.7) at the times t(ps) $= 0.2, 0.3, 0.5, 0.6, 1.0$. The MC algorithm with 10^5 trajectories and convergence parameter $E = 0.10$ (—) is compared with the difference equation approach (• • •).

We now consider the above process with the particle initially placed at $x_0 = 0.09$ nm, slightly to the left of the central maximum. The distribution after 1 ps is shown in Fig. 6(a). In such a barrier crossing one would be interested in the total yield (fraction) of particles reaching the right-hand well. Figure 6(a) demonstrates that roughly one-third of the particles have crossed the barrier. In other realistic applications, this number may be very small. Such situations would render the algorithm inefficient as crossing processes are then extremely rare. A way to improve the algorithm is to generalize the method introduced in Sec. III (and Appendix B) employing the survival probability. To obtain distribution $p_k(x, t | x_0)$, we distribute particles according to distribution $p'(x, t | x_0)$ and multiply the (accumulated) particle weight by the factor p_k/p', i.e.,

$$p_k(x, t | x_0) = [p_k(x, t | x_0)/p'(x, t | x_0)]\, p'(x, t | x_0), \tag{7.9}$$

where the factor in brackets is a "survival probability" for the jump and $p'(x, t | x_0)$ has been used to determine the jump endpoint. In Sec. III $p'(x, t | x_0)$ was taken to be the nonreactive distribution for the reactive process considered there. For Fig. 6(a) one can choose $p'(x, t | x_0)$ as the actual local linear distribution and, since the process is assumed to be nonreactive, the survival probability is unity. We could, however, choose some distribution other than the local linear one to distribute the jumps. We could, for example, distribute those jumps initially in the left-hand well ($x_0 < 0.1$ nm) according to a distribution that biases jumps toward the right-hand well; i.e., one derived from the constant force $F(x) = 20\ \mathrm{nm}^{-1}$. Then, for these jumps, we must multiply the weight by the above survival probability. Such a modification of the algorithm leads to the distribution displayed in Fig. 6(b). As can be seen, the distribution in the right-hand well (all that we are interested in) is correctly reproduced. The distribution to the left of the maximum is not accurate because particles that would normally have endpoints in this

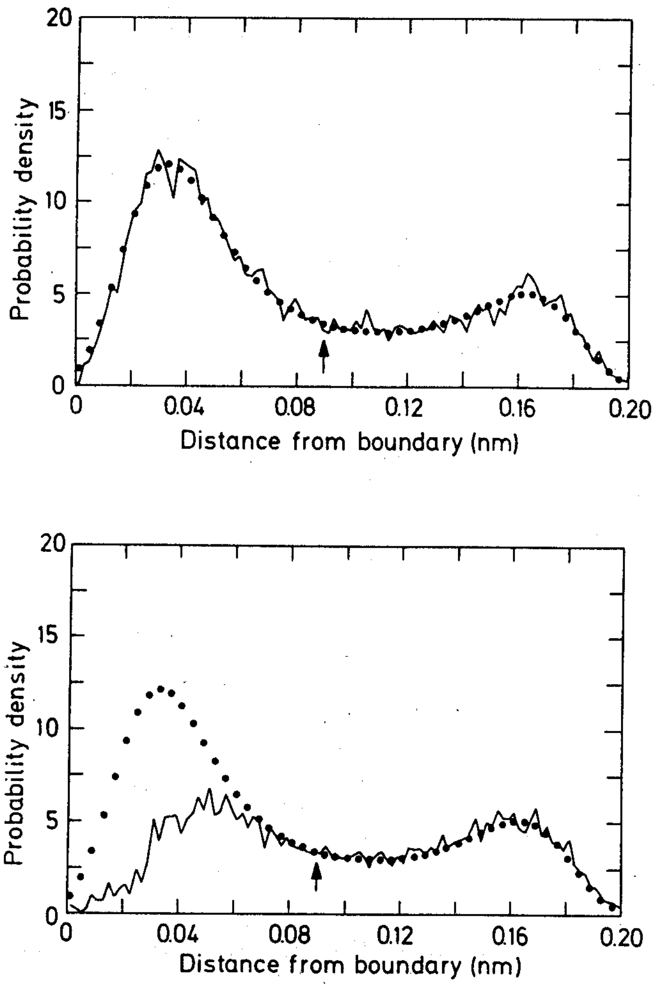


FIG. 6. (a) Results for the probability distribution $p(x, t=0.001\ \text{nm}^2 | x_0 = 0.09\ \text{nm})$ for diffusion in the bistable potential (7.7). The MC algorithm with 10^5 trajectories and convergence parameter $E=0.10$ is compared with the difference equation approach (• • •). (b) Same as (a) except that the biased MC algorithm discussed in Sec. VII was used.

region have been artificially sent across the barrier. Letting N denote the number of trajectories sampled and N_r the number of particles ending up in the right-hand well after time t, the above two alternative procedures may be approximately compared as follows. For Fig. 6(a), we have used an algorithm that gives N_r/N percent of the particles with weight one in the right-hand well. For Fig. 6(b), the "biased" algorithm gives (roughly) all of the particles with weight N_r/N in the right-hand well. As is seen by comparing Figs. 6(a) and 6(b), the stochastic fluctuations of the distributions of the right-hand well are about the same. In the example chosen, we have not significantly altered the number of particles crossing the barrier but, in a more realistic application, one might noticeably improve the calculated yield. Also important is that, in the biased algorithm, every particle can be made to cross the barrier and contribute to the yield and since particle distribution in the left-hand well is made according to the simple constant force distribution (without the need for considering the presence of the boundary at $x=0$), the biased algorithm is also faster: 86 CPU s for Fig. 6(a) compared to 72 CPU s for Fig. 6(b). This saving in computer time will clearly increase as the yield decreases. It should be noted that one cannot speed up the algorithm by basing local jump times on the biased distribution $p'(x, t|x_0)$ since the survival probability factor in Eq. (7.9) must be accurate. The biased algorithm based on Eq. (7.9) can be obviously generalized to the case of reactive diffusion.

VIII. DIFFUSION IN THE PRESENCE OF TWO BOUNDARIES

The solution to the SDE for diffusion between two boundaries is usually expressed in the form of an infinite series of image solutions.[28] Such a form is clearly unsuitable for treatment by a MC algorithm as developed here. We have, therefore, used the analytic solution for diffusion in a constant force near one boundary to obtain an approximate solution for nonlinear diffusion between two boundaries (either reflecting or reactive) that can be treated by the algorithm suggested in the preceding sections.

We write the one-boundary solution given by Eq. (2.4) in the form

$$p(x, t|x_0) = p_0(x, t|x_0) + p_a(x, t|x_0) , \tag{8.1}$$

where $p_a(x, t|x_0)$ denotes the boundary contributions (2.6) and (2.7) for a boundary located at x_a. This solution satisfies the (nonreactive) boundary condition

$$\left[\frac{\partial}{\partial x} - F(x)\right] [p_0(x, t|x_0) + p_a(x, t|x_0)] = 0 , \quad \text{at } x = x_a . \tag{8.2}$$

In the presence of a second boundary at x_b, we generalize Eq. (8.1) to read

$$p(x, t|x_0) \approx p_0(x, t|x_0) + \lambda_a p_a(x, t|x_0) + \lambda_b p_b(x, t|x_0) , \tag{8.3}$$

where coefficients $\lambda_a(t)$ and $\lambda_b(t)$ are assumed independent of x. The boundary condition satisfied by $p_b(x, t|x_0)$ is

$$\left[\frac{\partial}{\partial x} - F(x)\right] [p_0(x, t|x_0) + p_b(x, t|x_0)] = 0 , \quad \text{at } x = x_b . \tag{8.4}$$

p_0, p_a, and p_b each satisfy the SDE in the diffusion region between x_a and x_b. λ_a and λ_b are determined by requiring solution equation (8.3) to satisfy the actual boundary conditions at x_a and x_b

$$\left[\frac{\partial}{\partial x} - F(x)\right] p(x, t|x_0) = 0 , \quad \text{at } x = x_a, x_b . \tag{8.5}$$

Combining Eqs. (8.2), (8.4), and (8.5), and introducing the fluxes

$$j_m(x_n) \equiv j_m(x_n, t|x_0) = -\left[\frac{\partial}{\partial x} - F(x)\right] p_m(x, t|x_0) , \quad x = x_n ;$$
$$m = 0, a, b , \quad n = a, b , \tag{8.6}$$

we can derive at the following simultaneous equations for λ_a and λ_b:

$$\begin{aligned} j_0(x_a)\lambda_a - j_b(x_a)\lambda_b &= j_a(x_a) , \\ j_a(x_b)\lambda_a - j_0(x_b)\lambda_b &= j_0(x_b) , \end{aligned} \tag{8.7}$$

where all quantities depend on the jump time t. The solution to Eqs. (8.7) is

$$\lambda_a(t)=\frac{j_0(x_a)j_0(x_b)+j_0(x_b)j_b(x_a)}{j_0(x_a)j_0(x_b)-j_a(x_b)j_b(x_a)}\,. \tag{8.8}$$

λ_b is obtained from Eq. (8.8) by interchanging indices a and b. With these values for $\lambda_{a,b}$, distribution (8.3) must be renormalized. Although this solution could be used in a MC procedure, evaluation of the fluxes after each jump would be quite time consuming. A better method would be to express the fluxes in terms of the integrals of the distributions, as these must be calculated to provide the correct partitioning of jumps into p_0, p_a, and p_b.

Integrating the SDE for $p_0(x,t|x_0)$ from $x=x_b$ to infinity gives

$$j_0(x_b,t|x_0)=\frac{d}{dt}N_0(x_b,t|x_0)\ , \tag{8.9}$$

where

$$N_0(x_b)\equiv N_0(x_b,t|x_0)=\int_{x_b}^{\infty}dx\,p_0(x,t|x_0)\ . \tag{8.10}$$

Repeating this for $p_a(x,t|x_0)$ gives

$$j_a(x_b,t|x_0)=\frac{d}{dt}N_a(x_b,t|x_0)\ , \tag{8.11}$$

where

$$N_a(x_b)\equiv N_a(x_b,t|x_0)=\int_{x_b}^{\infty}dx\,p_a(x,t|x_0)\ . \tag{8.12}$$

These results simply express the flux at x_b due to the two distributions p_0 and p_a in terms of the rate of change of the number of particles that lie in these distributions beyond x_b. Note that the boundary condition at x_a is not used. The ratio of fluxes (8.9) and (8.11), after an integration by parts, can be expressed as

$$\frac{j_a(x_b)}{j_0(x_b)}=\frac{N_a(x_b)}{N_0(x_b)}+\frac{1}{N_0(x_b)}\int_0^t dt'N_0(x_b,t'|x_0)\times\frac{d}{dt'}\left[\frac{j_a(x_b,t'|x_0)}{j_0(x_b,t'|x_0)}\right]\,. \tag{8.13}$$

The approximation made is to neglect the second term in Eq. (8.13) either on the grounds that it vanishes for small time, or that the ratio of fluxes is nearly constant. The same procedure follows for the flux ratio at the boundary at x_a. We can then write Eq. (8.8) as

$$\lambda_a(t)=\frac{N_0(x_a)N_0(x_b)+N_0(x_b)N_b(x_a)}{N_0(x_a)N_0(x_b)-N_a(x_b)N_b(x_a)}\,. \tag{8.14}$$

Again, λ_b is obtained from Eq. (8.14) by interchanging a and b. The solution (8.1) will be close to the actual solution when λ_a and λ_b are nearly time independent. Note also that for all t, for these values of $\lambda_{a,b}$, distribution equation (8.3) is normalized.[29] When the overlap of p_a with the boundary at x_b is negligible, and similarly for p_b at x_a, Eqs. (8.8) and (8.14) reduce to $\lambda_a\approx\lambda_b\approx 1$. In most situations, where either the boundaries are relatively far apart compared to the range of the force or when the jump time is small (essentially two ways of looking at the same thing), this result simplifies the algorithm slightly. The N_m in Eq. (8.14), given by Eqs. (8.10) and (8.12), are readily calculated from Eqs. (2.5)–(2.7) to give

$$N_0(x_a)=E(b,0,x_0,-x_a)\ , \tag{8.15a}$$

$$N_0(x_b)=E(-b,0,-x_0,x_b)\ , \tag{8.15b}$$

$$N_a(x_b)=E(b,x_b-x_a,x_0-x_a,x_b-x_a)\ , \tag{8.15c}$$

$$N_b(x_a)=E(-b,x_b-x_a,x_b-x_0,x_b-x_a)\ , \tag{8.15d}$$

where we have defined

$$E(\beta,x_1,x_2,x_3)=\tfrac{1}{2}\exp(-\beta x_1)\,\mathrm{erfc}[(x_2+x_3-\beta t)/\sqrt{4t}]\ . \tag{8.16}$$

For a linear force, the parameters of Eqs. (8.16) transform as

$$\begin{aligned}&x_1\to x_1'=x_1\ ,\quad x_2\to x_2'=x_2\theta\ ,\quad x_3\to x_3'=x_3\ ,\\&\beta\to\beta'=2\beta/(1+\theta)\ ,\quad t\to t'=(1-\theta^2)/2c\ ,\end{aligned} \tag{8.17}$$

where $\theta=\exp(-ct)$. Also, in Eqs. (8.15a)–(8.15c), b must be replaced by $b-cx_a$ and in Eq. (8.15d) b must be replaced by $b+c(x_b-x_a)$.

The basic procedure of first partitioning jumps according to p_0, $\lambda_a p_a$, and $\lambda_b p_b$ remains as before. Defining the areas

$$N_m=\int_{x_a}^{x_b}dx\,p_m(x,t|x_0)\ ,\quad m=0,a,b\ , \tag{8.18}$$

one again chooses a uniformly distributed random number on the interval $[0,1)$. Then for $0\le r<N_0$, the endpoint is distributed according to p_0; for $N_0\le r<N_0+\lambda_aN_a$, endpoint distribution is made according to p_a; otherwise, p_b is used.[30] For jump endpoints distributed according to p_a or p_b, the two contributions to each distribution [Eqs. (2.6) and (2.7)] mean that an additional partitioning must be performed, or that one simply partitions the original $p(x,t|x_0)$ in Eq. (8.3) into five terms instead of three. Whatever the method, endpoints are chosen from the inversion of

$$r'=\frac{1}{N_m}\int_{x_a}^{x_f}dx\,p_m(x,t|x_0)\ . \tag{8.19}$$

If the boundaries are reactive, jump endpoints are still determined by Eq. (8.19) using the unreactive distributions, but now a survival probability must be calculated. To obtain the required reactive distribution $p_k(x,t|x_0)$ corresponding to Eq. (8.3), we could try to repeat the derivation that led to Eq. (8.14). One would find that Eq. (8.8) is still valid if the reactive fluxes are used

$$j_m(x_n)=-\left[\frac{\partial}{\partial x}-F(x)-k_n\right]p_{km}(x,t|x_0)\ ,\qquad x=x_n\ ;\ m=0,a,b\ ;\ n=a,b\ . \tag{8.20}$$

Unfortunately, the simple result (8.13) is no longer available for the ratio of the fluxes at the boundaries. Furthermore, any approximation of this kind would relate the λ's to the areas under the reactive distributions. These are not required in the partitioning phase of the algorithm and would be time consuming to calculate. As a first approximation to a more accurate solution, we assume that the λ's may adequately be given by Eq. (8.14) where the $N_m(x_n)$ are determined by the unreactive distributions, that is, as given by Eqs. (8.15). The $p_m(x,t|x_0)$, however, are given by their

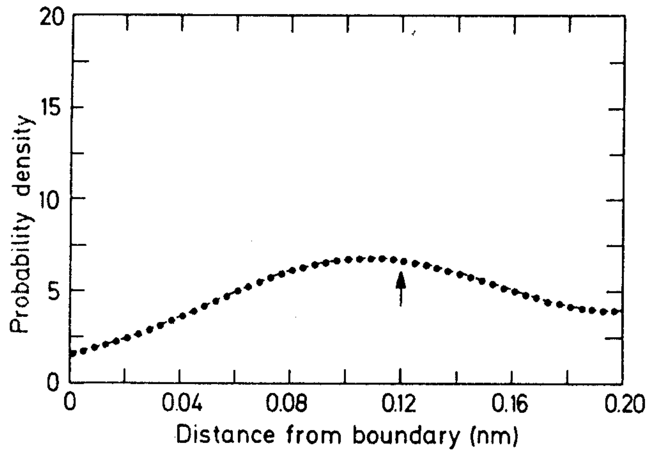


FIG. 7. Results for the probability distribution $p(x, t=0.0015\ \mathrm{nm}^2 | x_0=0.12\ \mathrm{nm})$ for diffusion in the linear force $F(x)=-20+100x\ \mathrm{nm}^{-1}$ between a reactive boundary ($k=40\ \mathrm{nm}^{-1}$, at $x=0$) and a reflective boundary ($k=0$, at $x=0.2$ nm). The analytic expression (8.21) (- - -) is compared with the difference equation approach (• • •).

reactive forms [Eqs. (4.2)–(4.5)]. This approximate solution is clearly valid for small jump times when p_0 accounts for most of the jumps and is perhaps correct to first order in the reaction rate as the λ's involve only the ratio of fluxes.

In summary, the solution to the SDE for diffusion between two reactive boundaries at x_a and x_b ($x_a<x_b$) is approximately given by

$$p_k(x,t|x_0) \approx p_0(x,t|x_0)+\lambda_a p_a(x,t|x_0)+\lambda_b p_b(x,t|x_0)\ , \quad (8.21)$$

where

$$p_0(x,t|x_0)=p_{k0}(x,t|x_0)\ , \quad (8.22)$$

$$p_a(x,t|x_0)=p_{k_a1}(x-x_a,t|x_0-x_a) + p_{k_a2}(x-x_a,t|x_0-x_a)\ , \quad (8.23)$$

$$p_b(x,t|x_0)=p_{k_b1}(x_b-x,t|x_b-x_0) + p_{k_b2}(x_b-x,t|x_b-x_0)\ , \quad (8.24)$$

with the right-hand sides denoting the distributions in Eqs. (4.3)–(4.5). The coefficients λ_a and λ_b are given by Eqs. (8.14)–(8.17). Endpoints are chosen according to the unreactive distribution equation (8.3) and the survival probability is given by the ratio of distribution equation (8.21) to distribution equation (8.3).[31] Finally, if a reactive optical potential is present, the survival probability is given by Eq. (5.16).

To test solution (8.21), we need only investigate the analytic expression as the MC algorithm will converge to it for short time steps. We have placed the boundaries at $x_a=0$ and $x_b=0.2$ nm. Diffusion between the boundaries occurs in the linear force $F(x)=-20+100x\ \mathrm{nm}^{-1}$ with the boundary at x_a having a reactivity of $k_a=40\ \mathrm{nm}^{-1}$ and that at x_b being totally reflective ($k_b=0$). The particle is initially at $x_0=0.12$ nm and the distribution was found after $t=1.5$ ps. In Fig. 7, the analytic distribution based on Eq. (8.21) is compared with the results of the difference method. Agreement is excellent over the entire diffusion domain. The reaction yields obtained by the two methods are 0.0286 (analytic) and 0.0293 (difference).

A number of other diffusion processes were also investigated to see if the error in Eq. (8.21) could be illustrated. It was found that if one chooses a process such that, say, λ_b is large compared to unity, then distribution p_b is exceedingly small and the corresponding term makes no contribution to Eq. (8.21). This will probably be the case in most (if not all) applications of the algorithm. Thus, at any given point in the diffusion space, one only need consider one (or neither) of the two boundaries in order to correctly partition the particle jump.

IX. DIFFUSION WITH A VARIABLE DIFFUSION COEFFICIENT

In most applications of the diffusion equation, a constant diffusion coefficient is assumed to allow a reasonable, if not highly accurate, solution to be obtained. In some situations, however, a variable diffusion coefficient may be necessary for correct modeling. In this section, we show how the algorithm may be extended to account for spatial variations in the diffusion coefficient.

The SDE for a variable diffusion coefficient $D(x)=D(x_0)B(x)$ is given by

$$\frac{\partial}{\partial t}p(x,t|x_0)=\frac{\partial}{\partial x}B(x)\left[\frac{\partial}{\partial x}-F(x)\right]p(x,t|x_0)\ , \quad (9.1)$$

where $t=D(x_0)\cdot$ time. We make the coordinate transformation

$$\frac{dz}{dx}=\frac{1}{\sqrt{B(x)}}\ ,\ z(x)=x_0+\int_{x_0}^{x}\frac{dx'}{\sqrt{B(x')}}\ , \quad (9.2)$$

with

$$q(z,t|x_0)=\sqrt{B(x)}\,p(x,t|x_0)\ . \quad (9.3)$$

This puts Eq. (9.1) in the form

$$\frac{\partial}{\partial t}q(z,t|x_0)=\frac{\partial}{\partial z}\left[\frac{\partial}{\partial z}F(z)\right]q(z,t|x_0)\ , \quad (9.4)$$

where the effective force $F(z)$ is

$$F(z)=\sqrt{B(x)}\,F(x)+\frac{1}{2\sqrt{B(x)}}\left.\frac{dB(x)}{dx}\right|_{x=x(z)}\ . \quad (9.5)$$

From Eq. (9.5) it is seen that a variable diffusion coefficient leads to a nonlinear effective force. The boundary condition satisfied by $q(z,t|x_0)$ is

$$\left[\frac{\partial}{\partial z}-F(z)\right]q(z,t|x_0)=0\ ,\quad \text{at } z=z(x(0))\ . \quad (9.6)$$

Equation (9.4) subject to Eq. (9.6) constitutes the nonlinear force problem described in Sec. V. The force parameters b and c representing the local force are chosen according to Eq. (6.13). From Eq. (9.5), we find

$$c=-F'(x_0)-\tfrac{1}{2}B'(x_0)F(x_0)+\tfrac{1}{4}[B'(x_0)]^2-\tfrac{1}{2}B''(x_0)\ ,$$
$$b=-F(x_0)-\tfrac{1}{2}B'(x_0)+cx_0\ . \quad (9.7)$$

The algorithm is then to choose some initial point x_0

and calculate the force parameters from Eqs. (9.7). A jump is made according to the appropriate partitioning and endpoint determination (from $x_0 = z_0$ to z_f in time t) on the basis of local linear diffusion *in z space*. If a reaction is present, we use the survival probability based on $q_k(z, t|x_0)$ with $k(x)$ as given in x space (no change) to decrease the particle weight as described in Secs. IV and V. The endpoint z_f must then be transformed to x_f using Eq. (9.2). If $B(x)$ varies approximately linearly with x near x_0, Eq. (9.2) yields the inversion

$$x_f = z_f + \tfrac{1}{4}(z_f - x_0)^2 B'(x_0) . \qquad (9.8)$$

A new jump is then performed starting from x_f and with parameters b and c redetermined locally. After the final trajectory jump to some x_F, the final weight is multiplied by $[B(x_F)]^{-1/2}$ to convert $q(z_F, t|x_0)$ to $p(x_F, t|x_0)$. The jump time t can be chosen locally according to Sec. VI, with the effective force $F(z)$ replacing the actual force $F(x)$ in the formulas.

X. THREE-DIMENSIONAL, SPHERICALLY SYMMETRIC DIFFUSION

Although the main reason for developing the one-dimensional algorithm is to extend it to treat higher-dimensional diffusion processes which cannot be handled by other numerical techniques, we note that diffusion in a spherically symmetric force can be described by a SDE in one variable (the radial coordinate) and should be treatable by the procedure already outlined. This is easily done if we start from the general d-dimensional radial SDE

$$\frac{\partial p(r,t|r_0)}{\partial t} = r^{1-d}\frac{\partial}{\partial r} r^{d-1}\left[\frac{\partial}{\partial r} - F(r)\right] p(r,t|r_0) - k(r)p(r,t|r_0) , \qquad (10.1)$$

subject to the initial condition

$$p(r,0|r_0) = \frac{1}{(2r_0)^{d-1}\pi}\delta(r-r_0) , \quad d = 2, 3 , \qquad (10.2)$$

and the boundary condition

$$\left[\frac{\partial}{\partial r} - F(r)\right] p(r,t|r_0) = k_1 p(r,t|r_0) , \quad \text{at } r = r_1 . \qquad (10.3)$$

The numerical factor in Eq. (10.2) is unity for the one-dimensional case. We have included both a reactive optical potential as well as a reactive boundary for generality, although only one would normally be assumed in a specific application. Equation (10.1) is easily transformed into the nonlinear SDE (5.3) through the substitution

$$p(r,t|r_0) = \frac{1}{(2r)^{d-1}\pi} q(x,t|r_0) , \qquad (10.4)$$

where $x = r - r_1$. This gives the one-dimensional SDE

$$\frac{\partial q(x,t|r_0)}{\partial t} = \frac{\partial}{\partial x}\left[\frac{\partial}{\partial x} - F(x+r_1) - \frac{d-1}{x+r_1}\right] q(x,t|r_0) - k(x+r_1)\,q(x,t|r_0) , \qquad (10.5)$$

subject to the initial and boundary conditions

$$q(x,0|r_0) = \delta(x + r_1 - r_0) \qquad (10.6)$$

and

$$\left[\frac{\partial}{\partial x} - F(r_1) - \frac{d-1}{r_1}\right] q(x,t|r_0) = k_1 q(x,t|r_0) , \quad \text{at } x = 0 . \qquad (10.7)$$

Equations (10.5)–(10.7) are readily solved using the nonlinear algorithm as described in Sec. V. Equation (10.5) shows that higher-dimensional diffusion processes in a spherically symmetric force act like one-dimensional processes in the presence of an additional nonlinear force. This force acts to keep the particle away from the origin and, in higher dimensions, essentially scales the radial coordinate to account for the increased phase space and the smaller chance of finding the particle near the origin.

The radial distribution $p(r,t|r_0)$ is obtained by applying the algorithm to Eqs. (10.5)–(10.7) to find $q(x,t|r_0)$. The final trajectory weight is then multiplied by the numerical factor in Eq. (10.4) to yield values for $p(r,t|r_0)$. In most applications, however, this is not necessary since contained in $q(x,t|r_0)$ is the higher-dimensional Jacobian factor, that appears in all radial integrals. A typical quantity of interest is the number of particles surviving after time t. This is the area under the radial distribution $p(r,t|r_0)$ and is given by (see Appendix D).

$$N(t|r_0) = 2^{d-1}\pi \int_{r_1}^{\infty} dr\, r^{d-1} p(r,t|r_0) \qquad (10.8)$$

$$= \int_0^{\infty} dx\, q(x,t|r_0) . \qquad (10.9)$$

Thus, it is often more expedient to save $q(x,t|r_0)$ rather than $p(r,t|r_0)$. Similar results hold when one is calculating the reaction rate for diffusion in a reactive optical potential or near a reactive boundary (Appendix D).

XI. FINAL COMMENTS ON THE USE OF THE ALGORITHM

We consider here, three additional modifications of the algorithm that need to be incorporated for certain applications. The first concerns our assumption and use of a delta function initial distribution

$$p(x, t=0|x_0) = \delta(x - x_0) . \qquad (11.1)$$

This functional form was chosen for simplicity but many applications may require that the SDE be solved subject to some other normalized initial distribution

$$p(x, t=0|x_0) = f(x_0) . \qquad (11.2)$$

To treat this case, the procedure of Sec. II need only be prefaced by a step that chooses a (different) starting point x_0 for each trajectory. One selects a uniformly distributed random number $r'' \epsilon [0, 1)$ and inverts the cumulative distribution function corresponding to the initial distribution

$$r'' = \int_0^{x_0} dx f(x) \qquad (11.3)$$

to obtain x_0. Once this initial point is chosen, the algorithm proceeds as before. If Eq. (11.3) cannot be inverted analytically, a similar distribution along with an initial nonunit trajectory weight may be required (Appendix B).

The second application of the algorithm we discuss concerns the calculation of time-dependent quantities, such as the reaction rate or yield in reactive diffusion. Suppose we want the yield at specified times $t_1, t_2, t_3, \ldots$. We modify the local jump time procedure of Sec. IV by requiring that the jump time Δt for a local jump during trajectory time t_0 to $t_0+\Delta t$ be such that the entire jump lies within one of the specified time intervals, e.g., $t_2 \le t_0 < t_0 + \Delta t \le t_3$. When the final time $t_0 + \Delta t$ coincides with one of the fixed time values (t_3 say), then the appropriate information is stored just as at the end of a trajectory. If the local jump times are generally smaller than the time intervals for which the yields are required, then the algorithm is not significantly slower. However, if the local jump times are usually longer than the selected time intervals, the computational time will be proportional to the number of intervals, i.e., to the amount of information desired. In any event, the algorithm is as efficient as possible. This modification has been programmed to obtain reaction rates and yields, as described in Appendix D, and found to work extremely well.

The final point to be considered is the application of the algorithm to processes in which either the force (or diffusion coefficient) or reaction rate is time dependent. The procedure as described assumes that both are constant in time but as the algorithm follows diffusing particles in time as well as in space, it is a simple matter to introduce the necessary modifications. Generalizing the ideas behind the jump distance determination of Sec. VI [Eq. (6.4)], we need only append a final step to the jump time procedure by setting

$$\frac{1}{\Delta t} = \frac{1}{t_c} + \left|\frac{\dot F(x_0,t_0)}{\epsilon F(x_0,t_0)}\right| + \left|\frac{\dot k(x_0,t_0)}{\epsilon k(x_0,t_0)}\right| + \left|\frac{\dot D(x_0,t_0)}{E D(x_0,t_0)}\right| , \quad (11.4)$$

where t_c is the jump time as given by Eq. (6.9). This new jump time Δt accounts for time-dependent changes in either the force or reaction rate when the local trajectory time t_0 is considered as a parameter in the force and reaction rate determination.

XII. SUMMARY OF THE ALGORITHM AND CONCLUSIONS

The most effective way to summarize the algorithm is to present a brief outline of the approach as a programmer might view it. For completeness, we include the additional points discussed in Sec. XI:

(i) Run initialization;

(ii) Trajectory initialization (a) starting point x_0 determination—Eq. (11.3); (b) initialize trajectory weight: Sec. XI, Appendix B;

(iii) Force parameter assignment—Secs. VI, IX, X, and XI (a) set $b(x_0,t_0)$, $c(x_0,t_0)$—Eqs. (6.15) and (9.7); (b) for reaction, set $k(x_0,t_0)$—Eqs. (7.2)–(7.6);

(iv) Local jump time determination—Eqs. (6.9) and (11.4);

(v) Pseudoconstant force parameter assignment—Eqs. (4.3) and (4.5b);

(vi) Jump endpoint partitioning and determination—Eqs. (2.8)–(2.11) and Eqs. (2.13), (2.14), and (2.17);

(vii) Modify trajectory weight for—(a) nonlinear correction (if included)—Eqs. (5.14) and (5.15); (b) a biased local jump—Eq. (7.9), Appendix B; (c) a reaction—Eqs. (3.2)–(3.5) ff, or Eq. (5.16);

(viii) Store time-dependent quantities (reaction rate or yield): Sec. XI, Appendix D;

(ix) For another trajectory, return to step (ii);

(x) Otherwise, store final quantities and normalize data—Appendix D.

We have presented here a rather complete description of a new Monte Carlo approach to solving the Smoluchowski diffusion equation, including time-dependent, nonlinear force, diffusion coefficient, and reaction terms, and subject to general initial and boundary conditions. It has been constructed with the advantages and disadvantages of previous MC and numerical approaches in mind and with the extension and application to the full three-dimensional (or higher) SDE in view. The algorithm is based as much as possible on known analytical results in diffusion theory and several Monte Carlo "tricks" have been discussed to further extend its usefulness. The major points of the procedure have been illustrated to show its convergence properties. In addition to the basic algorithm summarized above, a number of alternative modifications or procedures have been discussed that place the entire algorithm in a more meaningful setting.

The procedure described here is believed to be the only method available for investigating the SDE in full generality and that, by this method, a comprehensive study of many important and interesting diffusion systems in the biological, chemical, and physical regimes can now be undertaken.

ACKNOWLEDGMENTS

G. L. is very grateful to Wolfgang Elberfeld for numerous helpful discussions and suggestions during the course of this work. The authors would like to thank A. Weller for support of this work. This work has been supported in part by the Deutsche Forschungsgemeinschaft through the Sonderforschungsbereich SFB 123 "Primary Processes in Bacterial Photosynthesis".

APPENDIX A: THE RANDOM NUMBER GENERATOR AND erfc(x) EVALUATION

The success of any MC algorithm depends crucially on the randomness of the "pseudorandom" numbers generated. While no given sequence of numbers can be proven to be random, the sequence can be subjected to certain statistical tests that measure its nonrandomness.[32] As we require only a sequence of numbers uniformly distributed on the unit interval, it is easiest to use a method that has already been shown to yield pseudorandom number sequences that meet certain requirements for randomness. We have, therefore, chosen the multiplicative congruence method defined by the iterative equation

$$x_{i+1} = Ax_i(\mathrm{mod}\, T)\ , \tag{A1}$$

with $A = 7^5$ and $T = 2^{31} - 1$.[33] The Fortran implementation of Eq. (A1) is given below[24]:

```
FUNCTION RANDM (ISEED)
DATA IA, IT, TINV/16807, 2147483647, 0.4656612873E-09/
ISEED = MOD(IA*ISEED, IT)
RANDM = ISEED*TINV
RETURN
END
```

ISEED is initially assigned some starting value (7777777) and is subsequently updated by the subroutine. The ISEED values define a pseudorandom integer sequence on the interval (0, IT) and RANDM defines a pseudorandom real sequence on the interval [0, 1). The above routine has been found to be efficient and results of the algorithm would indicate that the sequence of numbers generated is sufficiently random for the applications intended here.[35]

The time-consuming part of the algorithm is the large number of calls that must be made to subroutine evaluations of erfc(x) and erfc^{-1}(x), where[8]

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty dz \exp(-z^2) \tag{A2}$$

is the complementary error function. Sufficiently accurate rational approximations to these functions have been given by Hastings.[36] We have found little significant difference in either the accuracy or the speed of the various forms available and have used the approximations[36]

$$e^{x^2}\mathrm{erfc}(x) = ((((a_5 y + a_4)\, y + a_3)\, y + a_2)\, y + a_1)\, y\ , \quad x \geq 0\ , \tag{A3}$$

where

$$y = (1 + a_6 x)^{-1}$$

and

$$a_1 = 0.254\,829\,593\ , \quad a_4 = -1.453\,152\,027,$$

$$a_2 = -0.284\,496\,736\ , \quad a_5 = 1.061\,405\,429,$$

$$a_3 = 1.421\,413\,741\ , \quad a_6 = 0.327\,591\,1\ ,$$

to evaluate erfc(x) and[37]

$$\mathrm{erfc}^{-1}(x) = y - \frac{(b_3 y + b_2)\, y + b_1}{((b_6 y + b_5)\, y + b_4)\, y + 1}\ , \quad x \geq 0\ . \tag{A4}$$

where

$$y = [b_7 - \ln(1 - |1 - x|)]^{1/2}$$

and

$$b_1 = 1.778\,739\ , \quad b_5 = 0.378\,538,$$

$$b_2 = 0.802\,853\ , \quad b_6 = 0.003\,699\,6,$$

$$b_3 = 0.014\,606\ , \quad b_7 = \ln 2 = 0.693\,147\,181\ ,$$

$$b_4 = 2.026\,268$$

to evaluate erfc^{-1}(x).

As the slow part of the algorithm is the evaluation of the erfc(x) and erfc^{-1}(x) functions, it might be asked whether they could in some way be dispensed with. In the Ermak[1] algorithm, the required random number sequence should be normally distributed and derives from the strict Gaussian distribution used to distribute the particle. Muller[38] has made an extensive comparison of several ways of obtaining such sequences and has offered some useful conclusions. However, in the algorithm presented here, the particle never strikes the boundary and the random sequence is consequently *not* derived from a Gaussian function [the lower limit of integration in Eq. (2.12) is 0 and not $-\infty$ yielding a cutoff Gaussian]. To our knowledge, there exists no alternative method of obtaining the required random sequences other than that given here.

APPENDIX B: SAMPLING FROM A NONRECTANGULAR DISTRIBUTION

This MC algorithm is based on sampling from a given normalized distribution $f(x)$ defined on the domain $[0, \infty)$. To do this we obtain a random number r uniformly distributed on the unit interval [0, 1) (see Appendix A) and invert the cumulative distribution function for $f(x)$,[10]

$$r = \int_0^x dx' f(x')\ . \tag{B1}$$

If $f(x)$ consists of a sum of terms, e.g.,

$$f(x) = f_1(x) + f_2(x)\ , \tag{B2}$$

then it is necessary to determine which term is to be used for the inversion. This is done by first determining the areas (which may be negative)

$$N_i = \int_0^\infty dx' f_i(x')\ , \quad i = 1, 2\ . \tag{B3}$$

Then, given a second random number $r' \in [0, 1)$, if $0 \leq r' < |N_1|$, the inversion

$$r = \int_0^x dx' f_1(x') \tag{B4}$$

provides the appropriate value for x. If $|N_1| \leq r' < |N_1| + |N_2|$, then inversion based on $f_2(x)$ is used. If both distributions are positive, then $|N_1| + |N_2| = 1$ for normalized $f(x)$. This procedure is easily generalized to any number of terms.

Several methods are available for inverting a given distribution. If $f(x)$ in Eq. (B1) [or $f_1(x)$ in Eq. (B4)] is either exponential or Gaussian, then the inversion may be carried out analytically. Certain other forms for $f(x)$ also allow analytic inversion. If $f(x)$ does not allow analytic inversion, then other methods such as stored tables of values, rejection techniques, or numerical procedures may be required. One "trick" for obtaining x analytically is to introduce into the inversion integral another distribution $g(x)$ whose shape is usually similar to $f(x)$,

$$r = \int_0^x dx' g(x') \left[\frac{f(x')}{g(x')}\right]\ . \tag{B5}$$

Inversion is then performed based on

$$r = \int_0^x dx' g(x') \qquad \text{(B6)}$$

and the sample (value of x) is assigned the weight $f(x)/g(x)$, which presumably can be evaluated analytically. This nonunit weighting corrects for the bias in selecting x values from $g(x)$ instead of $f(x)$. In the text, when $f(x)$ denotes a reactive distribution and $g(x)$ denotes the corresponding unreactive distribution, this weight is termed the survival probability (for the jump).

If $f(x)$ can be used for inversion, then most of the samples will be chosen near where $f(x)$ has a maximum, and convergence of quantities related to $f(x)$ will be most rapid. This is termed importance sampling.[24] If some other function $g(x)$ is required for inversion, convergence will be slower. There are instances, however, when it is actually better to sample from some approximate distribution rather than from the actual one, even when the latter can be analytically inverted (see Sec. VII).

APPENDIX C: STATISTICAL EVALUATIONS OF THE ALGORITHM

Here, we present a brief discussion of the statistical fluctuations inherent in all MC algorithms.

The results shown in Fig. 1(a) for the probability distribution were obtained by selecting a trajectory endpoint x from the distribution $p(x, t|x_0)$ as described in Sec. II. [It is assumed below that $p(x, t|x_0)$ corresponds to the correct distribution.] The x axis was partitioned into bins of equal size Δ and the final endpoint was accumulated into the corresponding endpoint bin. This procedure provides a discrete approximation to $p(x, t|x_0)$. If the mth bin accumulates n_m endpoints (of weight unity), then

$$p(x_m, t|x_0) \approx \frac{n_m}{\Delta N} \qquad \text{(C1)}$$

is the approximate distribution at the bin center (x_m) for N trajectories. From standard probability theory, the probability that n_m endpoints will be found in the mth bin is[39]

$$\text{prob}(n_m) = \frac{N!}{n_m!(N-n_m)!}(p_m)^{n_m}(1-p_m)^{N-n_m} , \qquad \text{(C2)}$$

where p_m is the probability that an endpoint will be put into the mth bin. This probability is

$$p_m = \int_{x_m-\Delta/2}^{x_m+\Delta/2} dx\, p(x, t|x_0) . \qquad \text{(C3)}$$

From Eq. (C2), we easily obtain the average values

$$\langle n_m\rangle = Np_m, \quad \sigma_m = \langle n_m^2\rangle - \langle n_m\rangle^2 = Np_m(1-p_m) . \qquad \text{(C4)}$$

Thus, the discrete distribution (C1) becomes

$$p(x_m, t|x_0) \approx \frac{1}{\Delta N}(\langle n\rangle \pm \sigma_m^{1/2}) \qquad \text{(C5)}$$

$$\approx \frac{p_m}{\Delta} \pm \frac{1}{\Delta}\sqrt{\frac{p_m(1-p_m)}{N}} . \qquad \text{(C6)}$$

Equation (C6) shows that the error decreases as $N^{-1/2}$ as expected. It also shows that this error is largest where $p(x_m, t|x_0)$ is largest, as is evident from Fig. 1(a). Alternative forms for Eq. (C6) show that the relative error decreases as $[p(x_m, t|x_0)]^{-1/2}$ and is proportional to $n_m^{-1/2}$. Thus, to obtain a more accurate discrete representation of the distribution function by decreasing the bin size, one must also increase the trajectory number to keep the relative error constant (ΔN = constant).

Similar results hold for other calculated quantities. If, e.g., we require

$$p(x, t) = \int dx_0\, p(x, t|x_0) f(x_0) \qquad \text{(C7)}$$

for a given normalized starting distribution $f(x_0)$, Eq. (C6) is replaced with

$$p_m = \int_{x_m-\Delta/2}^{x_m+\Delta/2} dx \int dx_0\, p(x, t|x_0) f(x_0) . \qquad \text{(C8)}$$

For quantities such as

$$G(t|x_0) = \int dx\, g(x)\, p(x, t|x_0) , \qquad \text{(C9)}$$

the error will in general be much smaller than that for $p(x_m, t|x_0)$. For the special case $g(x) = 1$, we have $G(t|x_0) = N(t|x_0)$, the number of particles surviving at time t. Since the algorithm was constructed to reproduce this number exactly (except for the error involved in approximating the reactive optical potential, which vanishes with decreasing jump time), $G(t|x_0)$ will converge extremely rapidly.

APPENDIX D: CALCULATION OF THE REACTION YIELD AND THE REACTION RATE FOR REACTIVE DIFFUSION

In diffusion processes, one rarely requires all the information that is available from the distribution. It is usually sufficient to obtain just a few quantities related to the moments of the distribution. One such is the total particle number

$$N(t|x_0) = \int_0^\infty dx\, p(x, t|x_0) , \qquad \text{(D1)}$$

i.e., the area under the distribution curve. As discussed in Appendix C, such integral quantities converge faster stochastically than the distribution and, for a fixed number of trajectories sampled, are thus more accurate. Moreover, the large amount of computer core needed to obtain accurate, discretized MC distributions may pose a serious problem, especially when one is working in higher dimensions. It often becomes necessary to judiciously select those quantities which do not require a lot of computer memory, such as $N(t|x_0)$.

In reactive diffusion processes, a quantity of interest is the reaction yield related to $N(t|x_0)$ by

$$\phi(t|x_0) = 1 - N(t|x_0) . \qquad \text{(D2)}$$

This quantity thus has the same computational advantages as $N(t|x_0)$: it can be conveniently and accurately computed. The yield is obtained by accumulating the final trajectory endpoint weight, represented by $p(x, t|x_0)$ in Eq. (D1) and discussed in Sec. III, and normalizing the accumulated result after all trajectories are run. Another quantity of interest is the reaction rate $-dN(t|x_0)/dt$. This quantity could be calculated by employing a difference approximation to Eq. (D2),

$$-\frac{dN(t|x_0)}{dt}=\frac{\phi(t-\Delta t|x_0)-\phi(t|x_0)}{\Delta t}\ , \qquad (D3)$$

but this method is inaccurate unless the yield is fairly constant during the time step.[40] It would clearly be advantageous to express the rate as an integral over the distribution just as the yield in Eq. (D2) is expressed. This may be done as follows:

Consider the SDE for diffusion in a reactive optical potential $k(x)$,

$$\frac{\partial p(x,t|x_0)}{\partial t}=\frac{\partial}{\partial x}\left[\frac{\partial}{\partial x}-F(x)\right]p(x,t|x_0)-k(x)p(x,t|x_0) \qquad (D4)$$

subject to the reflective boundary condition

$$\left[\frac{\partial}{\partial x}-F(x)\right]p(x,t|x_0)=0\ ,\quad \text{at } x=0\ . \qquad (D5)$$

Integrating Eq. (D4) over the diffusion domain gives

$$-\frac{\partial N(t|x_0)}{\partial t}=\int_0^\infty dx\, k(x)\,p(x,t|x_0)\ , \qquad (D6)$$

where Eqs. (D1) and (D5) have been used. This result is analogous to that obtained for the yield and is computed similarly. The final trajectory endpoint weight is multiplied by the reactive potential value $k(x)$ corresponding to the trajectory endpoint and the result is accumulated and normalized after all trajectories are run. This procedure is an MC evaluation of integral equation (D6). Note that the jump biasing technique of Sec. VII may be useful if the integrand of Eq. (D6) is small.

For diffusion near a reactive boundary, the procedure is slightly different. We can either repeat the above derivation using, instead, the nonreactive SDE supplemented by the reactive boundary condition, or we can simply insert the expression $k(x)=k\delta(x)$ into Eq. (D6). Both yield the rate

$$-\frac{\partial N(t|x_0)}{\partial t}=kp(0,t|x_0)\ . \qquad (D7)$$

As it stands, this result is not useful as we do not have an accurate value for $p(0,t|x_0)$. However, if we iterate the right-hand side of Eq. (D7) once using the Chapman-Kolomorogov equation (5.21), we find

$$-\frac{\partial N(t|x_0)}{\partial t}=k\int_0^\infty dx'\,p(0,t|x',t')\,p(x',t'|x_0)\ . \qquad (D8)$$

To evaluate this expression, we multiply the weight corresponding to the jump endpoint *before the last*, represented by $p(x',t'|x_0)$, by the quantity $kp(0,t|x',t')$. This latter factor is known analytically, since the jump time determination of Sec. VI assumes that during an individual diffusive jump, specifically from t' to t, the locally determined linear distribution $p(0,t|x',t')$ is suitably accurate. Loosely speaking, $p(0,t|x',t')$ is the "probability" that the particle will jump to the reactive boundary during the last diffusive jump and react (whether it actually does so or not).

Equations (D6) and (D8) allow determination of reaction rates for reactive diffusion in a way that is rapidly convergent and computationally convenient, simple, and fast.

APPENDIX E: ERRATA FOR PAPER I [J. CHEM. PHYS. 75, 365 (1981)]

Equation (7) should read

$$p_1(x,t|x_0)=(4\pi t)^{-1/2}\exp[bx_0-(x+x_0+bt)^2/4t]\ . \qquad (7)$$

Equation (13) should read

$$n(\tau|x_0)=[x_0^2 t/(4\pi\tau^4)]^{1/2}\exp[-(x_0-b\tau)^2 t/4\tau^2]\ . \qquad (13)$$

Equation (19) should read

$$p_0(x,t|x_0)=[c/2\pi\theta]^{1/2}\times\exp\{-c[x+b/c-(x_0+b/c)\exp(-ct)]^2/2\theta\}\ . \qquad (19)$$

In Figs. 3–6, the coefficient of the linear component of the force should be doubled, e.g., Fig. 4 should read "... $\beta F(x)=(-20-400x)\ \mathrm{nm}^{-1}$ ...".

Finally, we mention that the MC results were slightly inaccurately plotted. The endpoints accumulated in the bins were plotted at the x coordinate of the left-hand side of the bin, and not in the center where they should have been plotted. Thus, the distribution at the boundary shown in Figs. 2(b) and 5(b) is more accurate than appears from the graph. This has been corrected for the present paper.

[a]Present address: Laboratory of Chemical Physics, National Institute of Arthritis, Diabetes, and Digestive and Kidney Diseases, Building 2, Room B1-28, National Institutes of Health, Bethesda, MD 20205.

[1]D. L. Ermak, J. Chem. Phys. 62, 4189, 4197 (1975); D. L. Ermak and J. A. McCammon, *ibid.* 69, 1352 (1978).
[2]K. Schulten and I. R. Epstein, J. Chem. Phys. 71, 309 (1979).
[3]G. Lamm and K. Schulten, J. Chem. Phys. 75, 365 (1981); this paper is referred to as I throughout the present article and errata are listed in Appendix E of the present paper.
[4]J. Crank and P. Nicolson, Proc. Cambridge Philos. Soc. 43, 50 (1947); J. Crank, *The Mathematics of Diffusion*, 2nd ed. (Clarendon, Oxford, 1975), p. 141; see also Ref. 5 below.
[5]Z. Schulten and K. Schulten, J. Chem. Phys. 66, 4616 (1977).
[6]Throughout this paper the force will be expressed in units of k_BT.
[7]M. V. Smoluchowski, Phys. Z. 17, 557, 585 (1916); see Eqs. (52)–(54).
[8]M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions* (Dover, New York, 1972), p. 297.
[9]If $b<0$, distribution p_2 becomes negative and two modifications of the procedure are necessary. First, in the partitioning phase, the random number r must be chosen from the interval $[0,N)$, where $N=N_0+N_1+|N_2|$. Second, the contribution due to endpoints distributed according to p_2 (which will be small) must be subtracted from the total distribution of endpoints. These complications are minor and are easily accounted for. This case occurs when dealing with a nonlinear force (Sec. VII) or with two boundaries (Sec. VIII). Appendix B gives a more general analysis of the problem of nonrectangular sampling.

[10]J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods* (Wiley, New York, 1964), p. 36. This is the standard reference in Monte Carlo work; for a more recent summary see Ref. 35 below.

[11]We have programmed this algorithm and the results agree in all details with those presented here. Although it is slightly easier to program and more accurate (though not noticeably so), the algorithm is about 30% slower than the procedure described below.

[12]We have not observed any error near the boundary, even for extremely large b, as was the case with the algorithm in paper I. It is also possible to correct for the approximation by assigning a weight $p_2(x,t\,|\,x_0)/[a\exp(-b^*x)]$ to the jump (see Appendix B), but the error is so small that this is unnecessary.

[13]Note the corrections to the figure of paper I given in Appendix E of the present paper.

[14]An application of the algorithm in which it is advantageous not to bias motion in the direction of the force is given in Sec. VII.

[15]We have also programmed this algorithm and have obtained results identical to those shown here. We regard this procedure as unsatisfactory, since for $k > b/2$ distribution Eq. (3.5) becomes negative and the modifications given in Ref. 9 above become nesessary. When k is relatively large, p_2 must be accurately described in order to avoid negative distribution values for large x owing to the stochastic nature of the algorithm. In such cases, we have found Eq. (2.15) inadequate and have had to use Eq. (2.18) to represent the error function. This, in turn, requires additional partitioning with the result that the algorithm is about 30% slower and more difficult to program than that described here. An alternative procedure is to include the weighting factor mentioned in Ref. 12 but this then suggests what follows.

[16]In Ref. 5 the difference approach is applied to reactive diffusion.

[17]G. E. Uhlenbeck and L. S. Ornstein, Phys. Rev. **36**, 823 (1930). [This paper is reprinted in *Selected Papers on Noise and Stochastic Processes*, edited by N. Wax (Dover, New York, 1954).] See also Ref. 7, Eq. (51).

[18]A rough, semiempirical restriction on the jump time is $ct < \ln(1+cx_0/b)$, for $b, c > 0$; see also Sec. V.

[19]In this section only, $p_0(x,t\,|\,x_0)$ denotes some exactly known (unperturbed) distribution, and not the boundary-free distribution of Eq. (2.5).

[20]A. Szabo, K. Schulten, and Z. Schulten, J. Chem. Phys. **72**, 4350 (1980); see also P. M. Morse and H. Feshbach, *Methods of Theoretical Physics* (McGraw-Hill, New York, 1953), Vol. I, p. 870.

[21]The diffusion domain is assumed to be $(0,\infty)$. The surface terms at $x=0$ vanish if p and p_0 obey identical boundary conditions.

[22]L. I. Schiff, *Quantum Mechanics*, 3rd ed. (McGraw-Hill, New York, 1968), p. 304, Eq. (36.18).

[23]Note that, although the spatial factor $\zeta(x,x_0)$ depends on the potential difference between the initial point x_0 and the trajectory endpoint x, it must be computed at all intermediate jump points since the local force $F_0(x)$ is position dependent.

[24]Reference 10, pp. 57–59.

[25]A good introduction to path integrals is D. Falkoff, Ann. Phys. N. Y. **4**, 325 (1958).

[26]W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd ed. (Wiley, New York, 1970), Vol. I, p. 470, Vol. II, p. 322, Eq. (1.3).

[27]In fact previous MC algorithms were unable to treat a reactive boundary correctly and an optical potential was required to describe reactive diffusion (Ref. 2).

[28]N. S. Goel and N. Richter-Dyn, *Stochastic Models in Biology* (Academic, New York, 1974), pp. 251.

[29]Equation (8.14) can also be derived assuming multiple reflections of the diffusing particle off the boundaries. This explains why Eq. (8.13) "just happens to be" normalized with these λ values.

[30]When a force is present b at one of the boundaries will probably be negative. This requires the slight modifications to the algorithm given previously in Ref. 9.

[31]Although this algorithm has not been programmed, we estimate the additional complications would slow it down by 10%–30%.

[32]Reference 10, pp. 29–31. A general discussion of random number generation may be found in T. E. Hull and A. R. Dobell, SIAM Rev. **4**, 230 (1962) and in B. Jansson, *Random Number Generators* (Victor Pettersons Bokindustri Aktiebolag, Stockholm, 1966).

[33]P. A. W. Lewis, A. S. Goodman, and J. M. Miller, IBM Syst. J. **2**, 136 (1969). Additional pseudorandom number generators may be found in Ref. 8, p. 950.

[34]This particular algorithm may be machine dependent.

[35]F. James, Rep. Prog. Phys. **43**, 1145 (1980) gives a recent and particularly relevant discussion; see pp. 1173–1175.

[36]C. Hastings, *Approximations for Digital Computers* (Princeton University, Princeton, 1955), pp. 169; this approximation is also quoted in Ref. 8, p. 299, formula (7.1.26).

[37]Reference 36, p. 192.

[38]M. E. Muller, J. Assoc. Comp. Mach. **6**, 376 (1959).

[39]Reference 26, Vol. I, p. 35, 148.

[40]In Ref. 2, the rates given in Table I were calculated in this way for both the analytical and the Monte Carlo entries, i.e., the analytical entries were determined by employing in Eq. (D.3) the analytical yields from Eq. (4.13) of Ref. 5 rather than from the analytical expression (4.14) of Ref. 5 for $-dN/dt$ as stated.