Simulating mesoscopic reaction-diffusion systems using the Gillespie algorithm

David Bernstein*
1717 Hopkins Street, Berkeley, California 94707, USA
(Received 12 October 2004; revised manuscript received 21 January 2005; published 8 April 2005)

We examine an application of the Gillespie algorithm to simulating spatially inhomogeneous reaction-diffusion systems in mesoscopic volumes such as cells and microchambers. The method involves discretizing the chamber into elements and modeling the diffusion of chemical species by the movement of molecules between neighboring elements. These transitions are expressed in the form of a set of reactions which are added to the chemical system. The derivation of the rates of these diffusion reactions is by comparison with a finite volume discretization of the heat equation on an unevenly spaced grid. The diffusion coefficient of each species is allowed to be inhomogeneous in space, including discontinuities. The resulting system is solved by the Gillespie algorithm using the fast direct method. We show that in an appropriate limit the method reproduces exact solutions of the heat equation for a purely diffusive system and the nonlinear reaction-rate equation describing the cubic autocatalytic reaction.

DOI: 10.1103/PhysRevE.71.041103 PACS number(s): 82.20.Wt, 05.40.Fb

I. INTRODUCTION

The Gillespie algorithm [1,2] is a stochastic method that is frequently used to simulate spatially homogeneous chemical systems with small reactant populations. Under ordinary circumstances such systems are naturally characterized by small length scales as well. Since diffusion is fast on such scales (e.g., [3]) reaction-diffusion systems are natural models of small scale, spatially inhomogeneous chemical systems. Unless diffusion coefficients are artificially decreased (as in macroscopic experiments on the formation of Turing instabilities, e.g., [4]) the range of length scale of such systems is on the order of a typical cell, roughly 0.1 to 100 μm. In systems of this size the number of reactant molecules may only be in the thousands or millions.

Recently interest has turned toward using the Gillespie algorithm to simulate such mesoscopic, spatially inhomogeneous systems [5,6]. The natural extension of the method is to discretize the reaction chamber into subvolumes (usually squares or cubes) and consider them to be separate chambers that are coupled by the addition of a set of reactions which model diffusion. Some form of the Gillespie algorithm is then applied to the entire system consisting of all the reactions for each subvolume plus all the diffusion events which couple them.

The purpose of this paper is to examine the numerical and computational issues involved in adding diffusion to the Gillespie algorithm in this way in more detail than has been done in the past. In particular, we show how the diffusive rate constants can be derived from a finite volume discretization of the heat equation. We then show how the results of a stochastic simulation can be compared against exact solutions of the corresponding reaction-rate equations. This simultaneously provides a method for code testing and enables us to determine the rate of convergence of the error introduced by the spatial discretization. This puts the algorithm on a somewhat more firm computational foundation than previous works. We also investigate the regime in which the simulation contains only a single molecule and show that, again in an appropriate limit, Brownian motion is accurately simulated. The algorithm displays similar numerical properties when chemical reactions are added to the system. Although the method trivially generalizes to Cartesian meshes in any dimension, to make the analysis and numerical examples as simple and clear cut as possible we restrict our attention to systems with one space dimension. The use of the finite volume method makes likely the extension of the method to unstructured meshes in higher dimensions and various possibilities along these lines are discussed in the conclusion.

Work in this area began with the suggestion of such an application by Gillespie himself [1]. Elf *et al.* [5], and Fricke and Schnakenburg [6] have implemented algorithms similar to that described in this paper but on higher dimensional Cartesian meshes. In both cases the manner in which diffusion is handled by the Gillespie algorithm differs from what is presented here. These methods are also limited to uniform meshes and systems for which the diffusion coefficient is constant. In addition, no substantial code tests are given in either case. Lukkien *et al.* [7] describe an application of what is essentially the Gillespie algorithm applied to a spatially inhomogeneous chemical system which does not strictly speaking include diffusion.

The master equation for reaction-diffusion systems was studied by Baras and Mansour [8]. They compare predictions obtained by the master equation to that obtained by simulation of mesoscopic systems using Bird's algorithm. In their work the domain was also one dimensional, although the boundary conditions were periodic. Their simulations were also restricted to uniform meshes and constant diffusion coefficients.

Slightly further afield but still related we find the cellular automata simulations of Weimar [9] and the pattern formation studies of Turk on triangulated surfaces [10]. Similar

*Current address: Lawrence Berkeley National Laboratory, Mail Stop 50A-1148, One Cyclotron Road, Berkeley, CA 94720. Electronic address: dhbernstein@earthlink.net

ideas on the relationship between continuum equations and stochastic models based on discrete nearest neighbor interactions can be found in [11–13]. The Monte Carlo part of the Gillespie algorithm appears to have been reinvented a number of times and used in a wide variety of applications; see for instance [14–16].

The underlying model assumed in what follows is that we have a set of reactant molecules immersed in a solvent and that the reactants undergo collisions with solvent molecules much more frequently than they do with each other. This is in addition to the assumption, as per the original algorithm, that nonreacting collisions between reactants greatly outnumber reacting collisions. The explicit addition of a solvent to the model is required to justify the Brownian motion of the reactants in the limit in which their populations are very small. Note that these assumptions imply that the mean free path of the reactant molecules is negligibly small, much smaller than any other length scale in the system.

The outline of this paper is as follows. Section II contains a brief review of the Gillespie algorithm while Sec. III shows how the diffusion part of the reaction-diffusion system is incorporated into the algorithm. Section IV gives an overview of the algorithms and data structures that make up the code. Section V details tests of the code on a purely diffusive system. In Sec. VI we examine a nonlinear reaction-diffusion system and test the code on an exact traveling wave solution. We summarize the work in Sec. VII.

II. THE GILLESPIE ALGORITHM

The Gillespie algorithm is a well-known stochastic method in which the number of each chemical species is considered the independent variable and each reaction the system undergoes is executed explicitly. Hence the time evolution consists of a number of steps with each step being the execution of a specific reaction at a specific time. After a reaction is executed the number of molecules of each of the affected species is updated according to the reaction formula and the evolution goes on to the next step. Each reaction is assumed to be independent of the preceding one so that the evolution of the system is a Markov process. This section contains a review of only those elements of the method that will be necessary for what follows. The reader is referred to [1,2] for more detailed discussions.

The algorithm makes the following assumptions. We have a chemical system consisting of S species whose state at a given time t can be characterized by an integer valued tuple $\{S_i\}$, $1 \leqslant i \leqslant S$, where S_i is the population of species S_i. We use the convention that italicized symbols represent numbers (i.e., populations) and their nonitalicized counterparts represent names. The dynamics of the system are represented by a set of reactions of the form

$$\alpha_1^r \mathrm{S}_1 + \alpha_2^r \mathrm{S}_2 + \cdots + \alpha_S^r \mathrm{S}_S \xrightarrow{k_r} \beta_1^r \mathrm{S}_1 + \beta_2^r \mathrm{S}_2 + \cdots + \beta_S^r \mathrm{S}_S \tag{1}$$

where α_i^r and β_i^r are non-negative integer constants (the reactant and product coefficients, respectively) and r is an index which runs over the number of reactions, $1 \leqslant r \leqslant R$. Crucially, we assume that the events are exponentially distributed, i.e., the probability of reaction j occurring in the time interval $t+\delta t$ to first order in δt is $a_j \delta t$ where the a_j depend only on the state of the system at time t. As pointed out in [17] this is only strictly valid for dimolecular reactions although it can be argued for monomolecular reactions (as will be necessary later) and holds approximately for trimolecular reactions. From these assumptions it can be shown that the reaction probability density function is given by

$$P(\tau, j) = a_j \exp(-a\tau) \tag{2}$$

where

$$a = \sum_{j=1}^{R} a_j. \tag{3}$$

$P(\tau, j)d\tau$ is the probability that at time t the next reaction will occur in the interval $(t+\tau, t+\tau+d\tau)$ and will be the jth reaction of the system. Given the state of the system at time t the core of the algorithm is a procedure for selecting the next reaction and the time it is executed in such a way that over many simulations (2) is reproduced exactly.

There exist several equivalent procedures for selecting a pair (τ, j). The merits of each of these will be discussed in Sec. IV. Here we give the basic algorithm, called the direct method, a version of which will be used in the final code. Let $u_1, u_2 \in (0,1)$ be independent, uniformly distributed random numbers. Then the jth reaction will be executed if

$$\sum_{r=1}^{j-1} a_r < a u_1 \leqslant \sum_{r=1}^{j} a_r \tag{4}$$

and the time at which it is executed is $t+\tau$ where

$$\tau = -\frac{\ln u_2}{a}. \tag{5}$$

The quantity a_j is often referred to as the propensity for reaction j. Its relation to the familiar rate constant k_j is given in [1,2] but it will be convenient to summarize the results here for mono- through trimolecular reactions (reactants: propensity)

$$\begin{gathered} \varnothing : kV, \\ \mathrm{A} : kA, \\ \mathrm{A} + \mathrm{B} : kABV^{-1}, \\ 2\mathrm{A} : kA(A-1)V^{-1}, \\ \mathrm{A} + \mathrm{B} + \mathrm{C} : kABCV^{-2}, \\ 2\mathrm{A} + \mathrm{B} : kA(A-1)BV^{-2}, \\ 3\mathrm{A} : kA(A-1)(A-2)V^{-2}. \end{gathered} \tag{6}$$

Here k is the appropriate rate constant and V is the volume of the reaction chamber or element.

III. DIFFUSION

In this section we show how diffusion on an irregular grid in one dimension can be simulated by the Gillespie algorithm. As in previous work [5,6,8], the basic idea of the method is to divide the domain into a number of subvolumes, which we will call elements, and to consider the elements to be separate reaction chambers which are coupled together by allowing them to exchange molecules in a way designed to simulate diffusion. The coupling is formulated as an additional set of reactions which are appended to the list of chemical reactions taking place in each element. This master list is then processed by the Gillespie algorithm resulting in a coupled reaction-diffusion simulation.

In order that the original Gillespie algorithm be applicable to the chemical reactions occurring in each element we require that the concentrations there be considered uniform. This is equivalent to saying that each molecule in a given element will have an equal chance of interacting with any other molecule in the same element in a typical time interval between chemical reactions in that element. In order for this to be true the diffusion time across each element for every species should be much less than the typical reaction time. If h is the length scale of an element, D a typical diffusion coefficient, and d the dimension of the domain, then this requirement is

$$\tau_D \approx \frac{h^2}{2dD} \ll \tau_C \tag{7}$$

where τ_C is a typical time interval between chemical reactions. However, τ_D will also be the approximate diffusion time between elements, so this requirement is equivalent to saying that the typical rate of diffusion events, i.e., transfers of molecules between elements, is much greater than that of the chemical reactions inside the elements themselves.

Note that τ_C will be approximately the inverse of the propensity given in (6). These are of the form of a rate constant multiplied by zero or more concentration values multiplied by a single population value. Hence the propensity scales as the element population which is proportional to the element volume which in turn is proportional to h^d. Thus τ_C scales as h^{-d} and so the validity of the above inequality improves as h is reduced.

Whether or not (7) holds in a given simulation can easily be checked by comparing the number of diffusion events involving an element to the number of chemical reactions. In the example simulation in Sec. VI diffusion events typically outnumber chemical reactions by a hundred or more to one.

A. The master system

It will be convenient to think of the reaction-diffusion system not as an interaction between S chemical species in a spatially inhomogeneous domain which has been subdivided into E elements but as the interaction of SE species in a homogeneous domain of unit volume. This larger chemical system, which we call the master system, operating in a fictitious unit volume domain enables us to use the Gillespie algorithm in its original form. The relationship between the actual chemical system and that in the master domain is the obvious one: species A is represented by E species labeled A_i with the index indicating which element is occupied. The results of a simulation in the master domain can be transformed back into the problem domain when convenient, e.g., for purposes of visualization. Note that the assumption of uniform concentration in each element implies homogeneity of the master system. The use of the Gillespie algorithm on the master system differs from its ordinary usage only in the calculation of the propensity (6), in which the factor V^n varies according to the volume of the element the reaction is taking place in.

Let R be the number of reactions in the chemical system and n the number of neighbors of each element. Since each element contributes R chemical reactions and Sn diffusion reactions to the master system the total number of reactions in the master system is approximately $R'=E(R+Sn)$ (elements with faces on a boundary may contribute more or less than Sn reactions depending on the boundary conditions imposed there). For instance, if the chemical system contains a reaction of the form $\mathrm{A}+\mathrm{B}\rightarrow\mathrm{C}$ then the master system will contain E reactions of the form $\mathrm{A}_i+\mathrm{B}_i\rightarrow\mathrm{C}_i$. Similarly, the ESn diffusion reactions in the master system have the form $\mathrm{A}_i\rightarrow\mathrm{A}_j$. Note that molecules in separate elements are prevented from reacting in the master system because no reactions on the master list allow them to do so.

B. Diffusion in one dimension

Consider the case of the diffusion of a single species U on a nonuniform grid in one dimension whose elements are labeled by an index i. The master system will then have E species labeled U_i. We consider a model of diffusion whereby each element exchanges molecules with its nearest neighbors only. In the master system the exchange is represented by the transformation of a molecule of species U_i to one of species U_j. This suggests a set of reactions for U_i of the form

$$\mathrm{U}_i \xrightarrow{k_{i,i+1}} \mathrm{U}_{i+1}, \quad \mathrm{U}_i \xrightarrow{k_{i,i-1}} \mathrm{U}_{i-1},$$

$$\mathrm{U}_{i+1} \xrightarrow{k_{i+1,i}} \mathrm{U}_i, \quad \mathrm{U}_{i-1} \xrightarrow{k_{i-1,i}} \mathrm{U}_i \tag{8}$$

where $k_{i,j}$ denotes the rate governing the reaction that transforms U_i into U_j. Using the standard procedure, this set of reactions results in the reaction-rate equation for U_i

$$\frac{dU_i}{dt} = -(k_{i,i+1}+k_{i,i-1})U_i + k_{i+1,i}U_{i+1} + k_{i-1,i}U_{i-1} \tag{9}$$

where we have used the homogeneity of the master domain.

The reader familiar with finite difference or finite element methods will recognize that the resulting set of coupled ordinary differential equations for the time evolution of the variables U_i is in the same form as that resulting from a spatial discretization of a first order in time partial differential equation (PDE) using the method of lines, where the form of the spatial operator and the nature of the discretization are undetermined at this point. We next show that the

rate constants can be set so that (9) is an approximation to the heat equation.

C. Finite volume approximation

Let $u(x,t)$ be the concentration of species U. The two fundamental continuum equations governing diffusion are the conservation of mass

$$\frac{\partial u}{\partial t} = -\nabla \cdot \mathbf{J} \tag{10}$$

and Fick's law for the flux $\mathbf{J}$

$$\mathbf{J} = -D(x)\nabla u. \tag{11}$$

In this paper we consider the case in which D is allowed to vary in space but not in time.

We start by integrating (10) over element i and use the divergence theorem to evaluate the volume integral on the right hand side,

$$\frac{\partial U_i}{\partial t} = -\int_i \nabla \cdot \mathbf{J}\, dx = \mathbf{J}(c_i - h_i/2) - \mathbf{J}(c_i + h_i/2) \tag{12}$$

where c_i denotes the center of element i. Central to the finite volume method is the approximation used for the flux on the boundaries of each element. Here we approximate the gradient of u at the left boundary by

$$\nabla u(c_i - h_i/2) \approx \frac{u(c_i) - u(c_{i-1})}{c_i - c_{i-1}} = \frac{1}{c_i - c_{i-1}}\left(\frac{U_i}{h_i} - \frac{U_{i-1}}{h_{i-1}}\right). \tag{13}$$

Performing the same procedure at the right boundary, employing (11), and inserting into (12) yields

$$\frac{\partial U_i}{\partial t} = -U_i\left(\frac{D_{i,i-1}}{h_i|c_i - c_{i-1}|} + \frac{D_{i,i+1}}{h_i|c_i - c_{i+1}|}\right) + U_{i-1}\frac{D_{i-1,i}}{h_{i-1}|c_i - c_{i-1}|} + U_{i+1}\frac{D_{i+1,i}}{h_{i+1}|c_i - c_{i+1}|} \tag{14}$$

where $D_{i,j}$ is the diffusion coefficient evaluated at the interface between elements i and j. This suggests setting the rate constants in (9) to

$$k_{i,j} = \begin{cases} \dfrac{D_{i,j}}{h_i|c_i - c_j|}, & |i-j| = 1, \\ 0, & \text{otherwise}. \end{cases} \tag{15}$$

The diffusion coefficient at the faces of the mesh is evaluated using the well-known weighted harmonic average (e.g., [18,19])

$$D_{i,j} = \left(\frac{1}{|c_i - c_j|}\int_{c_i}^{c_j} \frac{ds}{D(s)}\right)^{-1} \tag{16}$$

where the integral is along the straight line joining the centers of the two elements. For $D(x)$ piecewise constant in one dimension this is

$$D_{i,i+1} = \left[\frac{1}{h_i + h_{i+1}}\left(\frac{h_i}{D_i} + \frac{h_{i+1}}{D_{i+1}}\right)\right]^{-1} \tag{17}$$

which in the case of a uniform grid reduces to the usual harmonic average $D_{i,i+1} = (1/D_i + 1/D_{i+1})^{-1}$.

If we consider $u_i = U_i/h_i$ to be element-centered variables and write (14) in terms of u_i then the truncation error term on the right hand side is $O(h)$ where h indicates the size of the largest element in the mesh. However we will see later that the L_∞ norm of the error of the overall scheme converges as h^2. This increase in the rate of convergence over that of the truncation error is known as supraconvergence and is a common feature of finite volume approximations to systems based on conservation laws [20].

Thus a solution to the system (9) will generate a set of element-centered concentration values u_i having an error compared to the exact solution of the heat equation which will decrease as h^2 assuming the time integration is exact. The Gillespie algorithm is such an exact time integrator in the sense that the only error associated with it is a sampling error.

Note that on a uniform grid with a constant diffusion coefficient (15) reduces to $k_{i,j} = Dh^{-2}$ which is the expression used in [5,6,8]. In addition, (9) reduces to

$$\frac{dU_i}{dt} = D\frac{U_{i+1} - 2U_i + U_{i-1}}{h^2}, \tag{18}$$

which is the familiar second order centered rule for finite differences.

D. Boundary conditions

The natural condition for a closed microchamber is that no molecules move through the boundary. Since the rate constants in (15) are associated with faces of the mesh this condition is enforced by setting the rate constants associated with faces on the boundary to zero. That this condition yields an approximation to a Neumann condition can be seen by rewriting the flux approximation (13) in terms of the rate constants and noticing that the flux across the boundaries is zero.

Dirichlet conditions can be modeled as follows. Let the leftmost interval have index 1 and let a fictitious element to its left have index 0. The rate equation for U_1 is

$$\frac{dU_1}{dt} = -(k_{1,2} + k_{1,0})U_1 + k_{2,1}U_2 + k_{0,1}U_0. \tag{19}$$

The second and fourth terms represent molecules leaving and entering the domain through the left boundary which can be represented by the reactions

$$\mathrm{U}_1 \xrightarrow{k_{1,0}} \varnothing, \quad \varnothing \xrightarrow{k_{0,1}} \mathrm{U}_1. \tag{20}$$

The two rate constants can be set by choosing a reasonable value for h_0, one that is comparable to other values in the system. For instance, choosing $h_0 = h_1$ gives

$$k_{1,0} = \frac{D_{1,0}}{h_1^2}, \quad k_{0,1} = D_{0,1}\frac{U_0}{h_0 h_1} = \frac{D_{0,1}u_0}{h_1} \tag{21}$$

where u_0 is the boundary value of the concentration. The choice of h_0, and consequently $k_{1,0}$ and $k_{0,1}$, effectively determines how rapidly the Dirichlet condition is able to respond to changes in the concentration near the boundary.

IV. ALGORITHMS AND DATA STRUCTURES

The number of reactions in the master system can easily be in the millions for even a modest mesh and a small set of chemical reactions. Thus efficient data storage and fast algorithms are required to make the simulation feasible. Fortunately this work has already been done in order to address the needs of simulating large chemical systems with many reactions. In this section we give a brief overview of these methods and their implementation in the code that is used in the rest of the paper.

A. The fast direct method

As is well known both of Gillespie's original algorithms for computing the next reaction and its execution time, called the direct method and the first reaction method, are not efficient when the number of reactions is very large. As a result a series of fast versions of both algorithms have been developed. The fast version of the direct method involves a binary tree while the fast version of the first reaction method involves a priority queue. Both methods require time to produce a pair (τ, j). They differ primarily in two respects: (i) the constant in front of the term, and (ii) the number of uniform random numbers needed.

In our experience, (ii) is not a concern since the speed of the random number generator is not a bottleneck in the simulation. In the examples below we have used the 32-bit numerical recipes routines RAN2 and RAN3 [21] and the 64-bit routine LSFR258 [22]. The last is roughly three times slower than RAN3 but the resulting simulation time is only a few percent longer. Most of the simulation time is spent recomputing the rates of affected reactions and updating the corresponding data structures.

We have chosen to use the fast direct method over the next reaction method of Gibson and Bruck [23] even though it is likely to be inferior regarding point (i). The reason for this is that we are looking ahead to the addition of automatic mesh refinement (AMR). AMR will involve the dynamical addition and subtraction of reactions from the system and it is unclear how to maintain the priority queue under these conditions. The maintenance of the binary tree in the direct method, however, is conceptually straightforward since reactions appear as leaves of the tree, which can easily be added and removed.

Since details of the binary tree used in the method are given in [24,23] we will not repeat them here but will merely make the following observations. Because of the nearest neighbor model of diffusion the dependency graph of the master system remains sparse so that updating all of the reactions dependent on the outcome of a given reaction is a constant time operation. We also note that the current implementation in the code described below is that of a straightforward binary tree. This means that the depth of the tree can be as high as 20 or 25, i.e., the tree may have tens of millions of leaves. At this depth, these types of trees often suffer in performance due to cache related problems. Although we have not observed this in the examples described below it seems prudent to warn the reader that they may occur in higher dimensional simulations with larger meshes and more complex chemical systems.

B. The code

The code used in the examples below is written in ANSI C++ and relies heavily on the standard template library. The code was built using CODE WARRIOR V9, optimized for speed, and run on a single processor of a dual 2 GHz, 1.5 gigabyte random access memory RAM, Macintosh G5 running operating system 10.3.5. In a typical simulation with a moderately large tree (10^5 leaves) the code executes about 3×10^5 reactions per second.

C. Complexity

The running time of the code is a function of E, R, S, and n, as well as the total number of molecules in the simulation. However, as stated above the model assumes that the slowest diffusion process in a given element is much faster than the fastest chemical process. This means that in any given simulation the number of diffusion events will greatly outnumber the number of chemical reactions. Hence the running time of the code will be determined primarily by the diffusion part of the simulation and this in turn will be determined by the species which undergoes the most diffusion events. Thus the complexity of the algorithm will depend primarily on two quantities: the number of molecules of this species and the number of elements in the mesh. In the case where all species have similar diffusion coefficients this number is just the total number of all species. Hence in this section we will examine the scaling of the running time of the code in the case of the pure diffusion of a single species. Since the formulas here are easily generalized to more than one space dimension we do so; the reader should keep in mind that the examples in the sections which follow are all performed with $d=1$.

Consider a purely diffusive system involving N molecules of one species A. From (5) we see that a typical time step has size

$$\tau \approx \left(\sum_{r=1}^{R'} a_r \right)^{-1}. \tag{22}$$

If h is a typical element length scale then according to (6) and (15) $a_r \propto A_i h^{-2}$ where A_i is the number of molecules in element i. There are n diffusion reactions per element, with n the number of neighbors of each element, so that $R \approx nE$ which gives

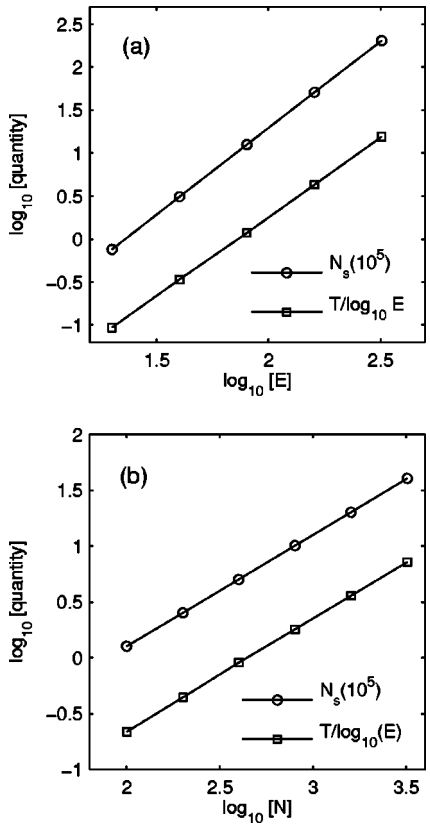

FIG. 1. Illustration of relations (24) and (25) in one dimension. (a) shows $\log_{10} N_S$ and $\log_{10}(T_r/\log_{10} E)$ vs $\log_{10} E$ for $N=1000$. (b) shows the same quantities vs $\log_{10} N$ for $E=80$. Least squares fits to the data in (a) give slopes of 2.02 and 1.84 for N_S and $T_r/\log_{10} E$, respectively, while fits to both curves in (b) give unit slope. The run time is in seconds and the number of steps is in units of 10^5.

$$\tau \propto \left(\sum_{i=1}^{E} A_i h^{-2} \right)^{-1} \propto \frac{h^2}{N}. \tag{23}$$

For a uniformly refined mesh the number of elements scales as h^{-d}; hence a run to time t has number of steps

$$N_s \approx \frac{t}{\tau} \propto \frac{N}{h^2} \propto NE^{2/d} \tag{24}$$

and since each step requires time proportional to the run time T_r scales as

$$T_r = O(NE^{2/d}\, E). \tag{25}$$

The scaling of N_s and T_r is shown in Fig. 1 for a series of runs on the unit interval. There is one species with $N=1000$ and $D=1$. The initial distribution is uniform and the code is run to $T=0.1$. The run time is in seconds and and the number of steps is in units of 10^5. The data are averaged over five runs.

It is instructive to compare (25) with the complexity of a simple finite element or finite difference code for approximating solutions of the heat equation. For the purposes of this comparison we ignore the important issues of stiffness, automatic time stepping, mesh refinement, etc. We assume that the PDE method uses an implicit scheme with update time per step of $O(E\,E)$. In order that the comparison be meaningful we require the PDE method to be second order in space and time, i.e., that it has a truncation error which scales as $O(h^2,(\Delta t)^2)$. Hence in order that this error scale as h^2 we should have $\Delta t \propto h$ which leads to $N_S \propto h^{-1} \propto E^{1/d}$ so that the run time should scale as

$$T_{\mathrm{PDE}} = O(E^{1+1/d}\, E). \tag{26}$$

The ratio of this time to (25) is

$$\frac{T_r}{T_{\mathrm{PDE}}} = O(NE^{-1+1/d}) \tag{27}$$

so that for small N the stochastic method is favored in all dimensions $d>1$. On the other hand, for fixed E the scaling favors the PDE method in the limit of large N, which is precisely the limit in which the continuum approximation becomes valid. This is not surprising since the magnitude of a concentration appears as an amplitude in a system of PDEs so it does not affect the complexity. Note that here we are comparing the simulation times for a single run of both methods. In some cases, as in the sections which follow, it will be useful to compute an average over a very large number of runs. In this case the average solution may be meaningfully compared to the result of the continuum approximation method and it would be appropriate to append a factor of M^2 to T_r where M is the number of runs. In this case the stochastic method suffers considerably because of the slow convergence of the sampling error.

V. TESTING DIFFUSION

The purpose of this section is to determine whether or not the simulation of diffusion by the Gillespie algorithm in the manner outlined above works. We do this by performing two tests: (i) comparison of the simulation to an exact solution of the heat equation and (ii) comparison of the simulation to the statistics of Brownian motion. While these tests are essentially straightforward, there are one or two fine points that need to be mentioned before we begin.

First, note that the result of a single simulation is a randomly generated piecewise integer function over the grid. Hence in order that a meaningful comparison be made to a smoothly varying exact solution the results must be averaged over many simulations. It is important to understand that this average solution is not guaranteed to be in some sense representative of the final state of the system. This will happen for instance in the case where the system undergoes a bifurcation due to an instability (as in pattern formation via a Turing instability). In general, we should anticipate that this situation will arise whenever we have a system that is sensitive to initial conditions. However, there are other systems for which the average solution does converge in a sensible way and this makes them ideal candidates for code testing.

For these cases, averaging over simulations makes sense and for the type of test described below the fluctations about the average solution can be regarded as a type of sampling error which can be decreased by increasing the number of

simulations. In this case the only property of the fluctuations we will be interested in is their mean. In general though other statistical properties of the fluctuations will be more important. Indeed, one of the attractive properties of the Gillespie algorithm is the possibility that it can reproduce in a natural way the statistical properties of real systems [17].

A. Initial data

To begin a simulation the code reads a data file containing among other things a set of user specified concentration functions which determine the average initial conditions. For a single species A let this function be $a(x)$. The starting point of a simulation is the initial distribution of molecules among the elements, which, as noted above, is a piecewise integer function over the domain. This function cannot be identical for each simulation because on average it would not represent $a(x)$ very well. In this section we show how the initial distribution is generated from a probability distribution so that when averaged over many instances it does converge to $a(x)$.

Let $A_{i,\mu}$ be the number of molecules initially placed in element i for simulation μ. The procedure for generating $A_{i,\mu}$ is as follows. First the total number of molecules in the simulation is computed from $a(x)$. In some cases the total number of molecules can be computed exactly but in general an approximation

$$\overline{A} = \int a(x)dx = \sum_{i=1}^{E} \int_i a(x)dx \approx \sum_{i=1}^{E} \overline{A_i} \tag{28}$$

can be used, where $\overline{A_i}$ is a suitable quadrature of $a(x)$ over element i. Even though an exact expression is available for the test systems below, in what follows the trapezoidal rule is used to compute $\overline{A_i}$ in order that the results be more representative of what might be encountered in practice. $\overline{A}$ is of course noninteger in general. Let the total number of molecules for simulation μ be generated randomly from $\overline{A}$ by

$$A_\mu = \lfloor \overline{A} \rfloor + \begin{cases} 1, & u < \mathrm{frac}(\overline{A}), \\ 0, & u \geqslant \mathrm{frac}(\overline{A}), \end{cases} \tag{29}$$

where $u \in (0,1)$ is a uniform random number and $\lfloor \overline{A} \rfloor$ and $\mathrm{frac}(\overline{A})$ are the integer and fractional parts of $\overline{A}$, respectively. The function $P_i = \overline{A_i}/\overline{A}$ is treated as a discrete probability distribution which is sampled A_μ times, the outcome of each sample being used to increment $A_{i,\mu}$ (hence $A_\mu = \Sigma_i A_{i,\mu}$).

It is clear that this method of producing initial data will converge to the continuous function $a(x)$ in the limit $M \to \infty$ in the sense that the averages

$$A_M = \frac{1}{M}\sum_{\mu=1}^{M} A_\mu, \quad A_{i,M} = \frac{1}{M}\sum_{\mu=1}^{M} A_{i,\mu} \tag{30}$$

will obey

$$|A_M - \overline{A}| \propto M^{-1/2}, \quad |A_{i,M} - \overline{A_i}| \propto M^{-1/2}. \tag{31}$$

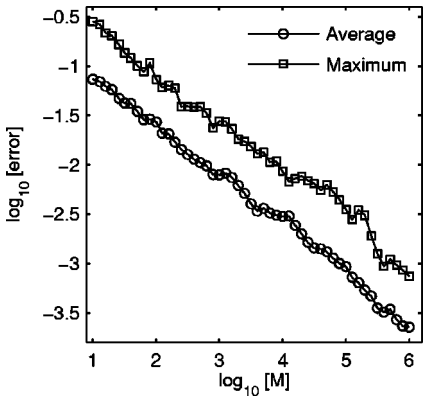

FIG. 2. Convergence of the error measures (33) for initial data on the unit interval. In this example $E=50$, $n=3$, and $\tilde{A}=500$. Least squares fits to the data give slopes of -0.5 and -0.48 for the average and maximum error, respectively.

To see this in practice consider the error in $A_{i,\mu}$ and denote by $e_{i,M}$ its average over M runs,

$$e_{i,M} = \frac{1}{M}\sum_{\mu=1}^{M}\left(1 - \frac{A_{i,\mu}}{\overline{A_i}}\right). \tag{32}$$

We look at the average and maximum of $e_{i,M}$ over the mesh,

$$\overline{e_M} = \frac{1}{E}\sum_{i=1}^{E}|e_{i,M}|, \quad \hat{e}_M = \max_{1\leqslant i\leqslant E}|e_{i,M}|. \tag{33}$$

Figure 2 shows the convergence of these two quantities for initial data given by (35) with $n=3$ and $\tilde{A}=500$. The mesh consists of 50 evenly spaced elements.

B. Time evolution with constant diffusion coefficient

In this case the concentration $a(x,t)$ obeys

$$\frac{\partial a}{\partial t} = D\frac{\partial^2 a}{\partial x^2}, \quad \frac{\partial a}{\partial x}(0) = \frac{\partial a}{\partial x}(1) = 0. \tag{34}$$

Setting $D=1$, an exact solution on the unit interval $[0, 1]$ satisfying the boundary conditions is

$$a(x,t) = \tilde{A}[1 + 1/2e^{-\pi^2 n^2 t}\cos(n\pi x)] \tag{35}$$

where n is a non-negative integer. Note that the total number of A is $\tilde{A}$.

In this section we look at the convergence of the error measures (33) for the time evolution. To do this we replace $\overline{A_i}$ in (32) with $\overline{A_i}(t)$ computed from (35). From the exact solution we see that the difference between the maximum and minimum values of $a(x,t)$ will decrease from $\tilde{A}$ to $\tilde{A}/2$ in time

$$T = \frac{\ln 2}{n^2\pi^2} \tag{36}$$

which gives us a reasonable time at which to end the simulation. Note that since the time increments τ are randomly

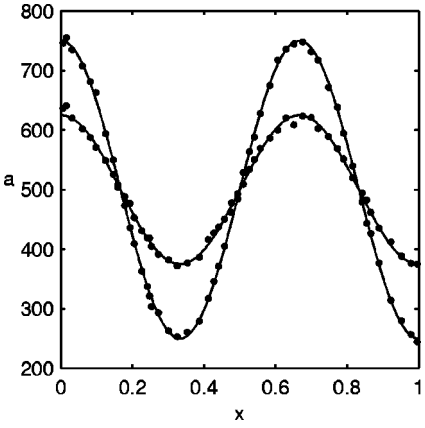

FIG. 3. Average solution of the initial data (top curve) and evolution to $T=0.0078$ (bottom curve) on a randomly spaced grid. Here $n=3$, $E=50$, $D=1$, $\tilde{A}=500$, and $M=1000$. The solid circles are the average concentration while the solid line is the exact solution (35).

generated it is very unlikely that the cumulative simulation time will reach T exactly. Nevertheless, each simulation can be stopped precisely at T. Suppose that a reaction takes the system from time $T_1 \leqslant T$ to time $T_2 > T$. Thus we may infer that the state of the system at T must have been the state it was in at time T_1 since by definition the system does not change state between reactions. Hence each simulation proceeds until a reaction is chosen such that the total simulation time exceeds T, at which point the simulation is stopped without executing the reaction.

Recall that each simulation is begun with a randomly generated set of initial data using the algorithm in the previous section. We can define an average concentration over M runs as

$$a_{i,M} = \frac{1}{M}\sum_{\mu=1}^{M} A_{i,\mu} h_i^{-1}. \tag{37}$$

A typical plot of $a_{i,M}$ for 1000 simulations is shown in Fig. 3. In this figure $n=3$, $\tilde{A}=500$, and the mesh consists of 50 randomly sized elements created by randomly generating $0.1 \leqslant h_i \leqslant 1$ and scaling to the unit interval.

Next we inspect the error measures (33). We set $n=1$ which gives $T \approx 0.07$. Figure 4 shows the two error measures for a set of 10^4 runs to $T=0.07$ on a mesh with 50 equal size elements and with 50 randomly sized elements. The number of molecules is set at $\tilde{A}=500$ in both cases.

Again the slope is about $-1/2$ in both cases. As in Fig. 2, it appears that the curves in Fig. 4 are converging to zero error in the limit $M \to \infty$. However, unlike the results for the initial data, we show in Fig. 5 that this is not the case. Here the time evolution is performed on a coarser grid and we use $n=3$ in (35). In Fig. 5(a) we see that $\overline{e_M}$ for $E=20$ equal size elements levels off at around 0.002 at $M \approx 10^4$ and does not decrease further. Figure 5(b) shows that this asymptotic sampling error is related to the mesh spacing h and scales as h^2.

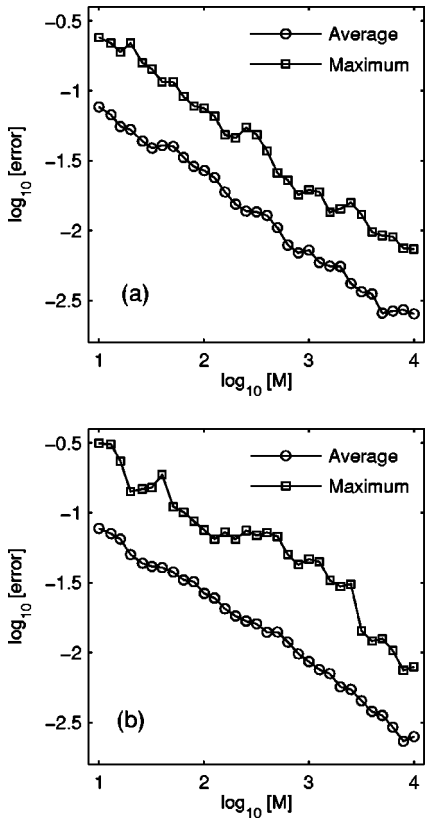

FIG. 4. Convergence to the exact solution of the heat equation on an evenly (a) and randomly (b) spaced grid. Fits to $\overline{e_M}$ yield slopes of -0.52 and -0.5, respectively. The diffusion coefficient is set to unity in both cases.

C. Time evolution with discontinuous diffusion coefficient

In this section we repeat the tests on the unit interval above but with the inhomogeneous diffusion coefficient

$$D(x) = \begin{cases} 1, & 0 \leqslant x \leqslant 0.5, \\ 5, & 0.5 < x \leqslant 1. \end{cases} \tag{38}$$

$D(x)$ is chosen to be discontinuous rather than smoothly varying because we anticipate that this case will be the one most likely encountered in practice.

The eigenfunctions and eigenvalues of the weak form of

$$\frac{\partial}{\partial x}\left(D(x)\frac{\partial f}{\partial x}\right) - \lambda^2 f = 0 \tag{39}$$

with Neumann boundary conditions are

$$f_n(x) = \begin{cases} \rho_n \cos(\lambda_n x), & 0 \leqslant x \leqslant 0.5, \\ \cos[\lambda_n (1-x)/\sqrt{5}], & 0.5 < x \leqslant 1, \end{cases} \tag{40}$$

with λ_n given by the roots of

$$(1/\sqrt{5})\tan(\lambda/2) + \tan(\lambda/2\sqrt{5}) = 0 \tag{41}$$

and ρ_n given by

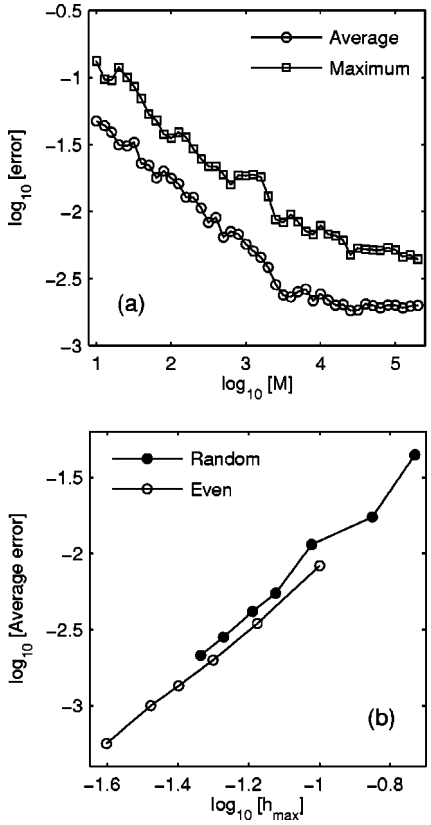

FIG. 5. Demonstration of the spatial discretization error. In these examples $n=3$ and $T=\ln 2/(9\pi^2)\approx 0.0078$. In (a) we show the error measures for $E=20$ on an evenly spaced grid. The asymptotic value of $\overline{e_M}$ is about 0.002. In (b) we plot the asymptotic value of $\overline{e_M}$ vs the maximum element size h for a series of evenly and randomly spaced grids with $10\leqslant E\leqslant 40$. Fits to the data give slopes of 1.92 and 2.11, respectively.

$$\rho_n = \frac{\cos(\lambda_n/2\sqrt{5})}{\cos(\lambda_n/2)}. \tag{42}$$

Similar to the above, we choose the following linear combination of the zeroth and second eigenfunctions:

$$a(x,t) = \tilde{A}[1 + 1/4e^{-\lambda_2^2 t} f_2(x)], \tag{43}$$

where $\lambda_2\approx 9.0065$ and $\rho_2\approx 2.0651$, and set the stopping time to $T=\ln 2/\lambda_2^2\approx 0.0085$.

Figure 6 shows the average solution over 10^4 runs against the exact solution (43) on an evenly spaced 40-element mesh. Measuring the spatial error with the method outlined above shows that the convergence is $O(h^2)$ in this case as well.

D. Random walks

The relationship between diffusion, Brownian motion, and random walks is well known. In this section we look at the motion of a single molecule propagating on a mesh by the algorithm above. If the molecule undergoes a random walk starting at the origin at time $t=0$ the distribution of its position at time t is given by

$$P(\mathbf{r},t) = \frac{1}{(4\pi Dt)^{d/2}} \exp\left(-\frac{r^2}{4Dt}\right) \tag{44}$$

which has the well-known expectation value

$$\langle r^2\rangle = 2dDt. \tag{45}$$

Let the molecule begin the simulation at $t=0$ in an element with center c_0 and end the simulation at time t in an element with center c_t. A measure of (45) over M runs is

$$m_M = \frac{1}{M}\sum_{\mu=1}^{M} \frac{|c_t - c_0|_\mu^2}{2dDt}. \tag{46}$$

m_M measures only the mean value of the distribution. To see the whole distribution note that in one dimension the probability of the molecule ending up in an interval of width Δ centered at x is

$$p(x,t,\Delta) = \left.\operatorname{erf}\left(\frac{x}{\sqrt{4Dt}}\right)\right|_{x-\Delta/2}^{x+\Delta/2}. \tag{47}$$

With $D=1$ the typical time to diffuse from the center of the unit interval to the boundary is $T_b=1/8$. We set the stopping time to a small fraction of this, $T=0.005$, in order to make it very unlikely that the molecule reaches one of the boundaries by the end of the run. The results for (46) and (47) of the random walk of a single molecule in one dimension for a series of 5×10^4 runs are shown in Fig. 7. The spatial discretization error discussed above should be kept in mind when considering this figure. We expect the error in the random walk to have the same $O(h^2)$ dependency as the previous results.

VI. NONLINEAR WAVE PROPAGATION

The cubic autocatalytic reaction (see, e.g., [25])

$$\mathrm{A} + 2\mathrm{B} \xrightarrow{k} 3\mathrm{B} \tag{48}$$

is at the heart of several interesting and well-studied nonlinear chemical systems, the archetypal example being the Brusselator [26,27]. In one dimension it becomes a good candidate for algorithm testing as a reaction-diffusion system because of the existence of a traveling wave solution whose wave front has a constant velocity and shape. In addition we will see that the averaging procedure outlined in the previous section appears to make sense for this system with the initial conditions described below.

A. Wave propagation

Let the two species A and B have the same diffusion coefficient D and concentrations $a(x,t)$ and $b(x,t)$. If the initial value of $a+b$ is constant in space then $a+b$ will be constant in time as well and the reaction-rate equation governing b reduces to

$$\frac{\partial b}{\partial t} = D\frac{\partial^2 b}{\partial r^2} + kb^2(a_0 - b), \tag{49}$$

where a_0 is the initial value of $a+b$. This can be put in dimensionless form by the scaling

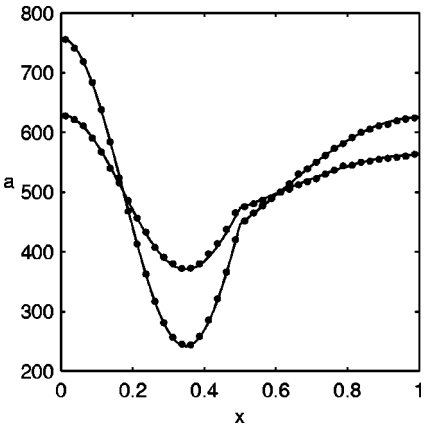

FIG. 6. The average solution (solid circles) is shown against the exact solution (solid line) for a series of runs with the discontinuous diffusion coefficient (38). The top curve is the initial data while the bottom curve is the data at $T=0.0085$. In this case $E=40$, $M=10^4$, $\tilde{A}=500$, and the mesh is evenly spaced.

$$\beta = \frac{b}{a_0}, \quad \tau = ka_0^2 t, \quad x = r\sqrt{\frac{ka_0^2}{D}}. \tag{50}$$

An exact traveling wave solution for β of (49) is

$$\beta(z) = \frac{\exp[-(1/\sqrt{2})(z-z_0)]}{1+\exp[-(1/\sqrt{2})(z-z_0)]} \tag{51}$$

where

$$z = x - c\tau, \quad c = \frac{1}{\sqrt{2}}, \tag{52}$$

and z_0 is a constant of integration. The boundary conditions satisfied by (51) are

$$\lim_{z\to+\infty} \beta(z) = 0, \quad \lim_{z\to-\infty} \beta(z) = 1. \tag{53}$$

The location of the wave front is given by $\beta=1/2$ or $x-c\tau - z_0=0$, which at $\tau=0$ implies $x=z_0$.

Our aim in this section is to compare the output of the code to the exact solution (51). However, as discussed in Sec. III D. the simulation requires boundary conditions and (51) on a finite domain $[0,x_b]$ does not satisfy either a Dirichlet or Neumann condition (or a linear combination of them) at either boundary. Nevertheless, suppose we choose the Neumann condition. At $x=\tau=0$ we have

$$\frac{\partial\beta}{\partial x}(0,0) = -\frac{1}{\sqrt{2}}\frac{\exp(z_0/\sqrt{2})}{[1+\exp(z_0/\sqrt{2})]^2} \tag{54}$$

$$\approx -\frac{1}{\sqrt{2}}\exp(-z_0/\sqrt{2}) \tag{55}$$

where the last expression holds when $z_0 \gg 0$. Hence $\beta'(0,0)$ can be made arbitrarily small by choosing z_0 large enough. By choosing the wave to start far enough away from the left boundary we can enforce the Neumann condition to a certain degree of accuracy there, e.g., $|\beta'(0,0)| \approx 10^{-3}$ gives $z_0 \approx 9.3$.

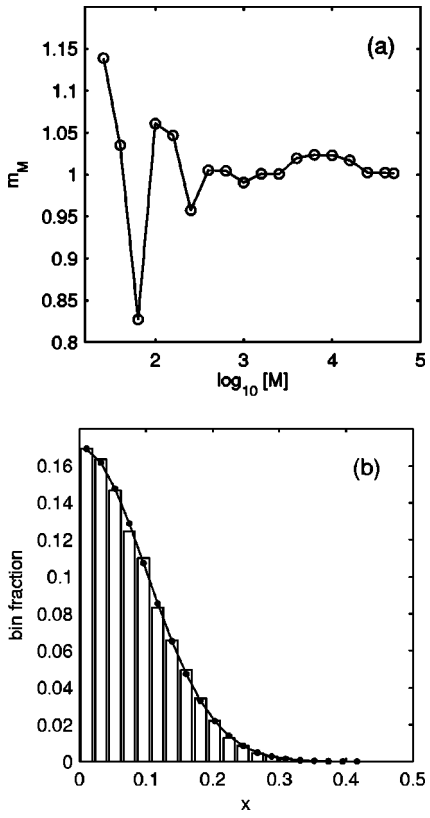

FIG. 7. Random walk data for a series of 5×10^4 runs. The mesh consists of 1000 randomly sized elements, $D=1$, and the run time is $T=0.005$. (a) shows m_M vs M while (b) shows the distribution of $x=|c_t-c_0|$ at $T=0.005$. The solid circles are the exact distribution (47).

Similarly, we choose the time of the simulation such that the wave stops before it comes close to the right boundary. Hence we choose

$$z_0 = 10, \quad x_b = 30, \quad T = \sqrt{2}(x_b - 2z_0) = 14.1. \tag{56}$$

In addition we set $D=1$, $k=2.5\times10^{-5}$, and $a_0=200$, giving a total population of 6000 molecules. The wave starts off at $x=10$ at $t=0$ and travels to $x=20$ at $t=14.1$. The distribution of molecules for a typical run at time $T=14.1$ for $E=60$ is shown in Fig. 8. Note the large amount of variation in the data. The average solution over 1000 runs is shown versus the exact solution (51) also in Fig. 8. The reaction count in these simulations is about 3.3×10^5 diffusion events for both A and B and about 2×10^3 occurrences of the autocatalytic reaction. Each simulation required about 1.5 s of CPU time. We can estimate the inequality (7) on this mesh by taking the values of A and B at the center of the wave front, where the majority of the reactions take place. At this point the populations are roughly 50 molecules per element so that the left hand side is $h^2/2D=1/8$ while the right hand side is approximately $1/(kA^2Bh^{-2}) \approx 1.3$.

To test the sampling and spatial discretization error we use the same domain, set $a_0=100$, $k=10^{-4}$, $D=1$, and

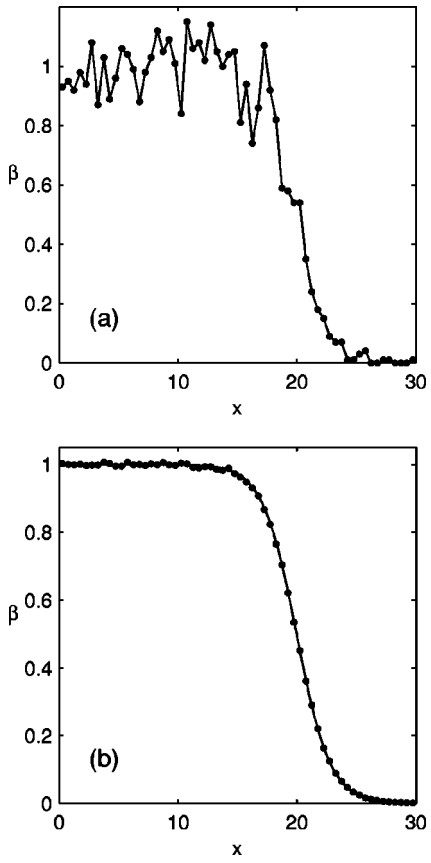

FIG. 8. A typical distribution of molecules at time $T=14.1$ is shown in (a) on a 60 element evenly space mesh. The average distribution over 1000 runs is shown in (b) (solid circles) along with the exact solution (solid line).

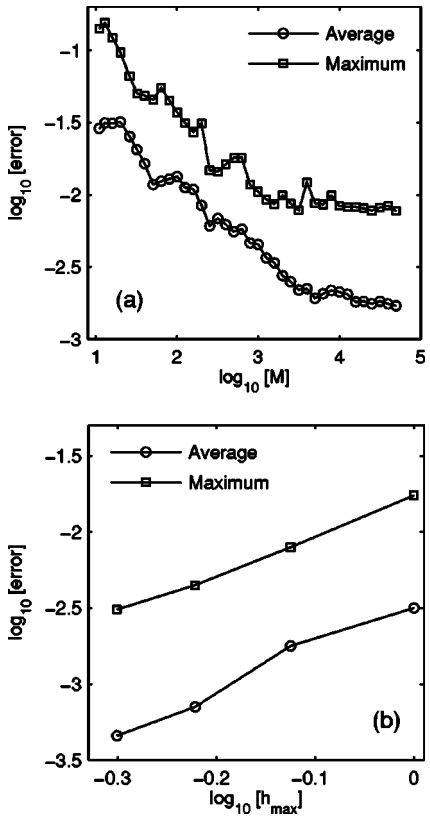

FIG. 9. Convergence of the error for the cubic autocalytic reaction. In (a) we show the error measures (33) on a 40 element evenly spaced mesh for $M \leqslant 50\,000$. In (b) the asymptotic value of $\overline{e_M}$ and $\hat{e}_M$ are plotted versus the element size for $30 \leqslant E \leqslant 60$. Fits to the data have slopes 2.9 and 2.5, respectively.

$$z_0 = 14, \quad x_b = 30, \quad T = \sqrt{2}(x_b - 2z_0) = 2.83. \qquad (57)$$

The wave travels a distance of only $x=2$, from one side of the center of the grid to the other. This gives a smaller value of β' at the boundary, about 5×10^{-5}, which ensures that this error will be smaller than the one which is being measured. The error measures (33) are shown in Fig. 9 on a 40 element evenly spaced mesh for $M \leqslant 5\times10^5$. The error measures are taken with respect to the concentration of B over the interval $[0, 18.5]$ rather than the entire domain in order to avoid the region of very small concentration at the right end. The convergence of the spatial error in this case is not as clear as in the pure diffusion case but it appears to be at a somewhat higher rate, although this could be an artifact of the small meshes we are restricted to due to the slow convergence of the sampling error.

B. Dilute reaction times

The previous examples were computed on meshes for which the number of molecules per element was fairly large. In this situation all of the elements along the wave front will contain enough reactant molecules to execute the autocatalytic reaction. One possible concern is that on very dense meshes there will be few elements which satisfy this criterion and the more complex reactions in a system will never be executed. In this final example we show that this is unlikely to be a problem. Consider the autocatalytic system on the unit interval and let the initial population consist of one molecule of A and two of B. The molecules are randomly placed in elements with equal probability (i.e., the algorithm of Sec. V A is used where the concentration functions are constant). The molecules are allowed to randomly walk over the mesh until all three of them occupy the same element and the algorithm determines that the autocatalytic reaction is executed. Using (2) and (6) the average time until the reaction occurs is $\tau=(2k)^{-1}$. In Table I we show τ when $k=1/2$ on a series of meshes with E ranging from 1 to 100. The diffusion coefficient has been set to $D=10$ so that the average diffusion time across the mesh is much smaller than the expected reaction time. The results are averaged over 1000 runs.

TABLE I. Average time for a single execution of the autocatalytic reaction on a series of evenly spaced meshes of varying size. The expected time is $\tau=1$.

E	1	2	5	10	25	50	100
τ	1.06	1.07	1.07	1.04	1.03	1.09	0.99

VII. CONCLUSION

We have examined an extension of the Gillespie algorithm for simulating reaction-diffusion systems on irregularly spaced Cartesian meshes. The systems may have inhomogeneous diffusion coefficients, including those with discontinuities. We have shown numerical examples in one dimension which demonstrate that the algorithm is capable of simulating both pure diffusion and the nonlinear cubic autocatalytic reaction-diffusion system accurately in an appropriate limit. We have also shown that the algorithm reproduces the statistics of Brownian motion for the random walk of a single molecule, at least on a sufficiently fine grid. We show that the spatial error caused by the discretization of the domain scales as h^2 for the diffusion problem and at least this fast for the cubic autocatalytic reaction. It should be stressed that we have not derived any general formulas for the scaling of this error for an arbitrary reaction-diffusion system. Currently this must be determined empirically and the method used in this paper for determining this rate is not guaranteed to work in a general setting.

Although the examples shown in this paper are in one space dimension the method itself trivially generalizes to higher dimensional Cartesian meshes. However, in a general situation such as an irregularly shaped cell or a microchamber of complex geometry one would like to use an unstructured grid. The derivation of the diffusive rate constants by the finite volume method opens up the possibility that one can borrow more results from the finite volume literature to generalize the method to unstructured meshes. Possibilities along these lines include Voronoi cells [19,28] and irregular polygons based on underlying simplicial meshes [29,30]. Such meshes have many advantages over regular Cartesian meshes including the fact that they can be graded in element size, locally refined and unrefined, and that they approximate the domain boundaries much more accurately.

In addition, the use of automatic mesh refinement would seem to be very useful in future versions of the algorithm. For instance, in the simulations of the autocatalytic reaction the variation in the concentrations of A and B are greatest in the neighborhood of the wave front which would seem to be a reasonable place to concentrate the elements. Elsewhere the ratio of A to B is either very large or very small so the number of reactions is small and in these regions it would be preferable to have fewer elements. In addition, in three dimensions the scaling of the diffusion rate is such that a large element will experience fewer diffusion events per unit time than a large number of smaller elements with the same total volume and population. Controlling the element size to take advantage of this may result in a substantial overall savings in simulation time.

There are other reasons for controlling the element size as well. For instance, up to this point we have ignored the finite size of the reactants. If an element becomes too small, it may become "overcrowded" and violate the basic assumption that the reactants collide much more frequently with the solvent than they do with each other. An extreme case would be an element too small to contain even a single reactant molecule. And of course elements cannot be too large either since the inequality (7) will be violated.

We must also be aware, however, that these restrictions on the element size may in some cases be mutually exclusive. This may happen for instance in systems characterized by a large separation of time scales, i.e., rate constants and diffusion coefficients which differ by many orders of magnitude. To be fair, these conditions arise in a wide variety of physical systems and their accurate simulation is an active area of work in many fields.

To summarize, this work can be considered an early step in the simulation of reaction-diffusion systems using the Gillespie algorithm. Many important issues remain to be resolved such as the extension to unstructured meshes, the effectiveness of automatic mesh refinement, and the addition of more realistic features to the basic model such as thermodynamics and time varying domain geometry.

ACKNOWLEDGMENTS

Thanks to Mayya Tokman for many helpful discussions and to Naida Lacevic for a useful conversation.

[1] D. T. Gillespie, J. Comput. Phys. **22**, 403 (1976).
[2] D. T. Gillespie, J. Phys. Chem. **81**, 2340 (1977).
[3] J. A. Pelesko and D. Bernstein, *Modeling MEMS and NEMS* (CRC Press, Baton Rouge, FL, 2002).
[4] Q. Ouyang, R. Li, G. Li, and H. L. Swinnery, J. Chem. Phys. **102**, 2551 (1995).
[5] J. Elf, A. Doncic, and M. Ehrenberg, SPIE First International Symposium on Fluctuations and Noise, 2003 (unpublished), pp. 114–124.
[6] T. Fricke and J. Schnakenberg, Z. Phys. B: Condens. Matter **83**, 277 (1991).
[7] J. J. Lukkien, J. P. L. Segers, P. A. J. Hilbers, R. J. Gelten, and A. P. J. Jansen, Phys. Rev. E **58**, 2598 (1998).
[8] F. Baras and M. M. Mansour, Phys. Rev. E **54**, 6139 (1996).
[9] J. Weimar, Fund. Inform. **52**, 277 (2002).
[10] G. Turk, Comput. Graph. **25**, 289 (1991).
[11] D. Givon, R. Kupferman, and A. M. Stuart, Nonlinearity **17**, R55 (2004).
[12] C. Haselwandter and D. Vvedensky, J. Phys. A **35**, L579 (2002).
[13] D. Vvedensky, Phys. Rev. E **67**, 025102(R) (2003).
[14] P. A. Maksym, Solid State Commun. **3**, 594 (1988).
[15] K. A. Fichthorn and W. H. Weinberg, J. Chem. Phys. **95**, 1090 (1991).
[16] J. L. Blue, I. Beichl, and F. Sullivan, Phys. Rev. E **51**, R867 (1995).
[17] D. T. Gillespie, Physica A **188**, 401 (1992).
[18] Z. Cai, J. Douglas, and M. Park, Adv. Comput. Math. **19**, 3 (2003).
[19] I. D. Mishev, Numer. Methods Partial Differ. Equ. **14**, 193

(1998).
[20] B. Cockburn and P. A. Gremaud, Math. Comput. **66**, 547 (1997).
[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C* (Cambridge University Press, Cambridge, U.K., 1993).
[22] P. L'Ecuyer, Math. Comput. **68**, 261 (1999).
[23] M. A. Gibson and J. Bruck, J. Phys. Chem. A **104**, 1876 (2000).
[24] S. Rajasekaran and K. W. Ross, ACM Trans. Model. Comput. Simul. **3**, 1 (1993).
[25] P. Gray and S. K. Scott, *Chemical Oscillations and Instabilities* (Clarendon Press, Oxford, 1990).
[26] I. Prigogine and R. Lefever, J. Chem. Phys. **48**, 1695 (1968).
[27] G. Nicolis and I. Prigogine, *Self-Organization in Nonequilibrium Systems* (Wiley, New York, 1977).
[28] N. Sukumar, Int. J. Numer. Methods Eng. **57**, 1 (2003).
[29] I. Aavatsmark, T. Barkve, O. Boe, and T. Mannseth, SIAM J. Sci. Comput. (USA) **19**, 1700 (1998).
[30] R. D. Lazarov and S. Z. Tomov Comput. Geosci. **6**, 483 (2002).