Published in final edited form as:
J Comput Chem. 2013 January 30; 34(3): 245–255. doi:10.1002/jcc.23130.

Lattice Microbes: high-performance stochastic simulation method for the reaction-diffusion master equation

Elijah Roberts[1,2,*], **John E. Stone**[3], and **Zaida Luthey-Schulten**[1,2,3,†]

[1]Department of Chemistry, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

[2]Center for the Physics of Living Cells, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

[3]Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

Abstract

Spatial stochastic simulation is a valuable technique for studying reactions in biological systems. With the availability of high-performance computing, the method is poised to allow integration of data from structural, single-molecule, and biochemical studies into coherent computational models of cells. Here we introduce the Lattice Microbes software package for simulating such cell models on high-performance computing systems. The software performs either well-stirred or spatially resolved stochastic simulations with approximated cytoplasmic crowding in a fast and efficient manner. Our new algorithm efficiently samples the reaction-diffusion master equation using NVIDIA GPUs and is shown to be two orders of magnitude faster than exact sampling for large systems while maintaining an accuracy of ~0.1%. Display of cell models and animation of reaction trajectories involving millions of molecules is facilitated using a plug-in to the popular VMD visualization platform. The Lattice Microbes software is open source and available for download at http://www.scs.illinois.edu/schulten/lm.

Keywords

chemical master equation; reaction-diffusion master equation; Gillespie algorithm; stochastic simulation; GPU computing

1 Introduction

Many cellular processes involve low copy number proteins and/or nucleic acids and consequently exhibit stochastic effects, which has been demonstrated by a series of pioneering experiments as reviewed in [1, 2, 3, 4]. The spatially homogenous chemical master equation (CME) and its inhomogeneous counterpart, the reaction-diffusion master equation (RDME), are frequently used to model such stochastic biochemical systems, *e.g.*, by McAdams and Arkin for gene expression [5]. Since the CME and RDME are analytically intractable for systems of significant complexity, biological stochastic systems are generally studied using large ensembles of computationally generated trajectories (realizations) of the underlying equations' time evolution [6].

The CME, while accounting for stochasticity, assumes the system is well-stirred such that reactions are equally likely between *any* molecules of reactants in the entire volume. For *in*

†Correspondence to: Zan Luthey-Schulten, Department of Chemistry, University of Illinois at Urbana-Champaign, A544 Chemical & Life Sciences Lab, 600 South Mathews Avenue, Urbana, IL 61801, Ph: 217-333-3518, Fax: 217-244-3186, zan@illinois.edu.
*Current address: Department of Biophysics, Johns Hopkins University, Baltimore, MD, 21218, USA

vitro biochemical systems the well-stirred approximation proves reasonable, but spatial organization and molecular crowding inside the cell bring this assumption into question for *in vivo* systems [7]. The RDME extends the master equation formalism of the CME to account for spatial degrees of freedom by dividing the system volume into discrete subvolumes with molecules diffusing between the subvolumes and reacting *only* with other molecules in the local subvolume. RDME theory [8, 9, 10, 11, 12] and numerics [13, 14, 15, 16, 17, 18, 19] have been the subject of much recent research. Our multiparticle diffusion (MPD) method allowed us to parallelize the diffusion operator of the RDME for efficient calculation of *in vivo* diffusion on graphics processing units (GPUs) [20].

In addition to the spatial degrees of freedom accounted for by the RDME, reactions occurring in a living cell are also subject to *in vivo* crowding. Cryoelectron tomography studies of single cells have revealed a crowded cytoplasm with decidedly non-uniform distributions of macromolecules [7, 21, 22, 23, 24, 25]. Molecular crowding and non-specific interactions have been theoretically predicted to give rise to anomalous subdiffusion in the cytoplasm [26, 27, 28, 7] and to have an effect on reaction kinetics [7]. The full extent to which these two effects impact the function of the cell is the subject of active investigation.

Here we introduce the “Lattice Microbes” software package for efficiently sampling trajectories from the CME and RDME on high performance computing (HPC) infrastructure using both exact and approximate methods. Particularly, the software takes advantage of any attached GPUs or other many-core processors to increase performance. The focus of the software is on the simulation of cell models with approximated *in vivo* crowding, such as shown in Figure 1. We also present a new approximate method for sampling the RDME using our GPU-optimized multiparticle diffusion operator (MPD-RDME), as first applied in simula-tions of *lac* genetic switch at the whole cell level [7]. Models and trajectories can be loaded and visualized using VMD [29], which allows for easy simulation setup and analysis.

2 Methods

2.1 Master equations for modeling stochastic chemical systems

To probabilistically study chemico-physical processes, one often uses a master equation formalism, which describes the time evolution of the probability for the system to be in a given state [30, 31]. Specifically, the CME [32] is widely used to stochastically model reactions in a well-stirred volume. Under the well-stirred assumption, each reaction occurs with a probability per unit time (propensity) proportional to its rate constant and the number of reacting molecules. The time derivative of the probability distribution P for the system to be in a given state x is then:

$$\frac{dP(x,t)}{dt}=\sum_{r}^{R}[-a_r(x)P(x,t)+a_r(x-S_r)P(x-S_r,t)]. \quad (1)$$

Here, x is a vector containing the number of molecules for each of the N species in the system and $a_r(x)$ is the reaction propensity for reaction r of R given a state vector. For a first order reaction involving species α: $a(x) = kx_\alpha$ (with k in units of s^{-1}), for a second order reaction involving species α and β: $a(x)=\frac{kx_\alpha x_\beta}{N_A V}$ (with k in units of $M^{-1}\ s^{-1}$ and V in L), and so forth. S is the $N \times R$ stoichiometric matrix describing the net change in molecule number when a reaction occurs.

The RDME is a less well-known method for modeling chemical reactions under conditions of slow diffusion [8, 9, 10]. In the formalism of the RDME, the system's volume is divided into a set of uniform subvolumes with spacing λ and with the molecules in the system being distributed amongst the subvolumes. Reactions occur only between molecules within a subvolume and each subvolume is considered to be well-stirred such that reactions within it follow standard kinetic theory and can be described by the CME. Diffusion is accounted for by random transitions of species between neighboring subvolumes, also with constant probability per unit time.

The time evolution of the probability for the system to be in a specific state x (where x_ν contains the number of molecules of each species in the $\nu \in V$ subvolume) is then the sum of the rates of change due to reaction and diffusion, as described by the operators $\mathcal{R}$ and $\mathcal{D}$, respectively:

$$\frac{dP(x,t)}{dt} = \mathrm{R}P(x,t) + \mathrm{D}P(x,t),$$

$$\frac{dP(x,t)}{dt} = \sum_{\nu}^{V}\sum_{r}^{R}[-a_r(x_\nu)P(x_\nu,t) + a_r(x_\nu - S_r)P(x_\nu - S_r,t)]$$

$$+ \sum_{\nu}^{V}\sum_{\xi}^{\pm\widehat{i},\widehat{j},\widehat{k}}\sum_{\alpha}^{N}[-d^\alpha x_\nu^\alpha P(x,t) + d^\alpha (x_{\nu+\xi}^\alpha + 1)P(x + 1_{\nu+\xi}^\alpha - 1_\nu^\alpha, t)]$$

The reaction operator is simply the CME applied to each subvolume independently. The diffusion operator describes the rate of change of the probability due to the molecules' propensity to diffuse between the subvolumes. x_ν^α is the number of molecules of species $\alpha \in N$ in subvolume ν and d^α is the diffusive propensity for a molecule of species α to jump from subvolume ν to neighboring subvolume $\nu + \xi$, which is related to its macroscopic diffusion coefficient by $d = \frac{D}{\lambda^2}$. The first part of the diffusion operator then is probability flux out of the current state due to molecules diffusing from subvolume ν to subvolume $\nu + \xi$, where ξ is a neighboring subvolume in the $\pm x$, $\pm y$, or $\pm z$ direction as indicated by the $\hat{\imath}$, $\hat{\jmath}$ and $\hat{k}$ units vectors. The second part of the diffusion operator describes probability flux into the current state due to molecules diffusing into the current subvolume from a neighboring subvolume. The 1_ν^α syntax represents a single molecule of type α in subvolume ν.

The RDME has been shown to asymptotically approximate the Smoluchowski diffusion-limited reaction model when the reaction radius of the molecules is small compared to the length of the subvolume [9, 10, 11, 13]. This condition is satisfied when the average time until the next reaction event in a subvolume is much longer than the average time until the next diffusion event, $\tau_R \gg \tau_D$. For example, the reaction of a single molecule of A with a single molecule of B according to $A + B \xrightarrow{k} C$ requires (if k is in units of $M^{-1}\,s^{-1}$ and assuming pseudo-first-order kinetics such that $A \xrightarrow{k[B]} C$):

$$\tau_R \gg \tau_D,$$

$$\frac{1}{k[B]} \gg \frac{\lambda^2}{6D},$$

$$\frac{N_A \cdot \lambda^3 \cdot 1000}{k} \gg \frac{\lambda^2}{6D},$$

$$\lambda \gg \frac{k}{D \cdot N_A \cdot 6000},$$

where $D = D_A + D_B$ is the relative diffusion coefficient between molecules of A and B, N_A is the Avogadro constant, and the factor of 1000 converts from L to m^3.

The CME and RDME are difficult to analytically study for even simple systems, and instead are most often sampled using a Monte Carlo approach. Many independent realizations of the system's path through the probability space are computationally calculated and then combined to reconstruct the system's time-dependent probability density function (PDF). Typically, approaches for exactly sampling the CME are variants of Gillespie's stochastic simulation algorithm (SSA) [33]. Gillespie's direct method involves straightforward generation of a trajectory by sampling an exponentially distributed time until a reaction occurs and then choosing from the possible reactions based on the weighting of their propensities. Gibson and Bruck developed the more efficient next-reaction method [34] for generating trajectories for systems with many reactions by using a priority queue to track upcoming reaction times. Several approximate sampling methods, such as τ-leaping [35, 36] and R-leaping [37], have also been developed that can decrease simulation time under certain simulation conditions.

The next-subvolume method for exactly sampling the RDME was introduced by Elf and Ehrenberg [13, 14]. This method uses a next-reaction-like priority queue for organizing the list of subvolumes by the time of their next diffusion or reaction event. Once a subvolume has been selected for an event, the standard Gillespie direct method is used to determine the specific reaction or diffusion event that occurred in the subvolume. Marquez-Lago and Burrage devised the binomial τ-leaping variant of the next-subvolume method to speed up RDME calculations in a manner analogous to the τ-leaping algorithm for the CME [38]. Similarly, Lampoudi, Gillespie, and Petzold published the multinomial simulation algorithm to speed up the diffusion operator by efficiently calculating only the net flux between subvolumes [17]. An alternate approach was taken in development of the Gillespie multi-particle (GMP) method by Rodríguez *et al.* [15]. This approach is the most similar to our algorithm, as discussed below, and uses operator splitting to calculate the reaction and diffusion operations separately. Diffusion is accounted for by a jump process [39] in which molecules jump to a neighboring subvolume at a predetermined time according to their macroscopic diffusion coefficient. In between diffusion jumps, reactions are processed in continuous time using the standard Gillespie method on a per subvolume basis. Drawert and colleagues have recently taken a different route and used the finite state projection method during the calculation the diffusion operator [18]. Instead of jumping molecules from subvolume to subvolume, they instead calculate the probability distribution of the diffusion operator using the finite state projection method and then randomly update the lattice each time step using the statistics of the distribution. There are numerous other variations on these algorithms including methods that intermix deterministic and stochastic solutions according to molecule concentration [19, 40], methods that use direct compilation of models into code [41], and methods that use the GPU to accelerate, *e.g.*, the GMP algorithm in two dimensions [42] or large reaction networks [43].

2.2 Exact master equation sampling using GPUs

Lattice Microbes provides both the direct [33] and next-reaction [34] methods for exact sampling of the CME along with the next-subvolume method [13] for exact sampling of the RDME. These algorithms are not generally parallelizable, except for the generation of large quantities of independent pseudorandom numbers. Lattice Microbes will use any attached GPUs to generate pseudorandom numbers in parallel with sampling, which can boost the speed of the exact methods. Generation of uniformly, exponentially, and normally distributed pseudorandom numbers on the GPU is done using the XORWOW algorithm [44] with the appropriate transform.

2.3 Accelerated sampling of the RDME using the MPD-RDME method

Taking full advantage of GPUs for RDME sampling requires an algorithm with fine-grained parallelism. In a typical RDME simulation, diffusion is the most frequent event, with many independent diffusion events occurring across the volume in a short time window. To take advantage of this independence we developed the MPD-RDME method, which relaxes the constraint of exact time evolution and instead takes an approximate time-stepping approach. Unlike τ-leaping methods, our time steps are chosen to be very short such that the probability of any individual molecule being involved in a diffusion or reaction event is small. The short time step duration leads to many extra calculations, but the subvolumes are rendered independent and can be calculated in parallel on the GPU.

The basic principle behind our approach starts with the discretization of space into a three-dimensional cubic lattice with spacing λ and time into time steps of length τ. Then, starting from an initial state at t_0, the state of the system at times t_τ, $t_{2\tau}$, $t_{3\tau}$… is sequentially calculated. The calculation involves integrating the propensity for reaction and diffusion events to occur over the time step in each subvolume and then updating the subvolume using random firings of the events according to the calculated probabilities. Compared to the standard Gillespie algorithm, which jumps from event to event along a systems' history, the MPD-RDME can be viewed as a brute force method of marching through the history step by step (see Figure 2). In the limit of infinitesimal τ($\tau = dt$), the history generated by pure time stepping would correspond exactly to that generated by the standard Gillespie algorithm. In practice, the finite size of the time steps introduces error into the generated history. In the Results section we discuss how to chose a time step and its effect on the error.

For efficiency of calculation, the MPD-RDME algorithm takes an operator splitting approach such that the reaction and diffusion operators are sequentially applied at each time step. Algorithm 1 shows the MPD-RDME algorithm. The diffusion operator calculates any diffusion events of the molecules to and from neighboring subvolumes over the time step. Due to the nature of the GPU architecture, the optimal implementation of the diffusion operator uses dimensional splitting to further split the diffusion operator into three suboperators, one each for the x, y, and z dimensions [20]. Within each suboperator each subvolume is processed in parallel and each molecule in a subvolume is considered to have two possible diffusion events, one for diffusion to the subvolume in the plus direction and one for diffusion to the subvolume in the minus direction. The propensity for either of these events is $\frac{D}{\lambda^2}$ and the probability that the molecule will diffuse away during the time step is then $P(D^{\pm})=\int_0^{\tau}\frac{2D}{\lambda^2}e^{-\frac{2D}{\lambda^2}t}dt \approx \frac{2D\tau}{\lambda^2}$, where for efficiency we have only kept the first two terms of the Taylor expansion of the exponential. Note that $P(D^+) = P(D^-) = P(D^+)=P(D^-)=\frac{1}{2}P(D^{\pm})$. We generate a uniformly distributed random number from (0, 1] and determine which of the ranges (0, $P(D^+)$], ($P(D^+)$, $P(D^-)$], or ($P(D^+)$, 1] the value falls into to decide whether the molecule diffuses in the plus direction, diffuses in the minus direction, or stays in the subvolume, respectively.

During calculation of the diffusion operator, the probability that a molecule moves should never exceed 1, which limits $\tau \leq \frac{\lambda^2}{2D}$. In the case that the molecule jumps every time step ($P(D^{\pm})$ = 1) the time steps have duration $\tau=\frac{\lambda^2}{2D}$, and we recover the diffusion operator of Chopard and Droz [39] as used in the GMP method [15]. In the MPD-RDME method, τ is

usually significantly shorterthan this limit. (Especially, one should always limit $P(D^{\pm}) \leq \frac{1}{2}$ to avoid an issue where molecules initially in even and odd subvolumes can never interact for $P(D^{\pm}) = 1$ [39].) The advantage of the MPD-RDME diffusion operator is that molecules do not move deterministically at their mean characteristic diffusion time as in GMP, but rather move probabilistically in time in a manner more similarto exact stochastic sampling of the RDME.

After the application of the diffusion operator during a time step, the reaction operator is applied to calculate any changes in the chemical species present in each subvolume. First, the total propensity a_{tot} of all possible reactions given the state of the subvolume is calculated using the standard procedure [33]. The probability that at least one reaction occurs during the time step is then calculated as $P(R)=\int_0^{\tau} a_{tot} e^{-a_{tot}t} dt$. Here we do not use a Taylor approximation but compute the probability directly as $1 - e^{-a_{tot}\tau}$ using the GPU's native transcendental function. A uniformly distributed random number is drawn from (0,1] and if the value is $\leq P(R)$, a reaction is assumed to occur. In this case, a second uniformly distributed random number is drawn to determine the reaction that occurred according to their relative propensities following the standard procedure [33]. A limitation of the algorithm is that only a single reaction can occur in each subvolume during a time step. To limit the error that arises from missed reaction events, we typically limit the time step such that there is a low probability (typically ≤2%) for any particular reaction to occur in a subvolume during a given time step.

2.4 Using the RDME with *in vivo* Models

To model cellular systems, including approximated cytoplasmic crowding, we introduce a subvolume type and make the reaction and diffusion propensities of the RDME dependent on the subvolume type(s). Addition of the subvolume type allows reactions to occur only in subvolumes of a specific type and diffusion of molecules to be limited to, or different in, specific subvolumes. The site dependent diffusion and reaction propensities can then be configured in such a way as to approximate the spatial organization of a cell (see Figure 1).

To construct a lattice representation of a cell, one first builds a three-dimensional cell model in continuous space using geometric primitives, such as cylinders, spheres, etc. Space is partitioned into regions were diffusion coefficients and reaction rates are uniform. Users are free to build any desired geometries. For example, a simple model for *Escherichia coli* can be constructed using two coincident capsule shapes with radii r_1 and r_2, where $r_2 > r_1$ and $r_2 - r_1$ is the thickness of the cytoplasmic membrane. The inner volume is used to model the cytoplasm, the boundary volume models the membrane, and the volume outside of the capsules is the extracellular space. Within the cytoplasm, proteins and metabolites diffuse with their *in vivo* diffusion coefficients. Within the membrane region, membrane proteins diffuse more slowly with the appropriate two-dimensional diffusion coefficient, with no probability to transition into either the cytoplasmic or extracellular volume. In the extracellular space, metabolites and signaling molecules diffuse with their *in vitro* diffusion coefficients. Small molecules that are membrane permeable have the appropriate transition rates between the three volume types such that they can diffuse from the extracellular space across membrane into the cytoplasm. The system volume is most often bounded with a constant concentration boundary such that molecules are created and destroyed to maintain the extracellular concentration, but the boundary may also be periodic, reflective, or absorbing depending on the simulation requirements.

All boundary conditions are implemented by controlling the behavior of molecules in a virtual apron of subvolumes that surround the simulation space. This apron is reloaded at each time step and then used in the diffusion operator to process molecules diffusing into and out of the edge subvolumes. For periodic boundary conditions, each apron subvolume is loaded with the state of its periodic partner. For reflective boundary conditions the apron is loaded as a subvolume into which diffusion is not allowed by any molecule. For absorbing boundary conditions the apron is loaded as a subvolume that molecules can diffuse into but not out of. Finally, for constant concentration boundary conditions the apron is loaded with a random statistical sample of molecules at the desired concentration. Molecules can diffuse into and out of the apron during the time step, but the apron is destroyed and randomly loaded again at the next time step maintaining the concentration.

To model cytoplasmic crowding, a series of immobile obstacles are placed in the cytoplasm. Such obstacles are modeled as reflective volumes that molecules cannot diffuse into. We use the size distribution of *in vivo* crowders from Ridgway *et al.* [27] based on proteomics data. For simulations incorporating structural data, such as from cryoelectron tomography experiments, exact locations for known obstacles are used directly. The remaining obstacles are then placed randomly from largest to smallest filling the cytoplasmic volume to a specific fraction of its total volume.

Once the continuous-space model is constructed, it is discretized onto a lattice (see Figure 3) using a coarse-graining procedure. First, each subvolume is mapped to a simulation region by finding the geometric primitive that contains the subvolume. Care must be taken when assigning subvolume contained in multiple primitives to ensure the connectivity of the subvolumes reproduces that of the original primitive, *e.g.*, the subvolumes representing the cytoplasmic membrane must be freely traversable using a series of nondiagonal jumps. The type of each subvolume is assigned based on the simulation regions to which it is assigned. Second, a list tracking the total occupied volume for each individual subvolume is created. The geometric volume of each continuous-space obstacle is divided amongst all the subvolumes with which it overlaps. The subvolume list is then sorted in order of decreasing occupancy. The subvolumes are marked as reflective in sorted order until the total volume of reflective subvolumes equals the original occupancy fraction. Third, molecules are randomly placed into the appropriate subvolume. The full process may be repeated with different subvolume spacings to obtain discretizations of the same simulation system with differing spatial resolution.

2.5 Simulation execution, output, and analysis

To accurately sample the statistics of a CME or RDME model, one needs to generate many independent trajectories. To facilitate this process on large compute clusters, Lattice Microbes can be executed as a parallel program using MPI. In this mode, Lattice Microbes assigns trajectories to CPU cores and GPUs, starting new trajectories as those running finish. One MPI controller process is launched per computer node and is responsible for launching simulation threads for the trajectories using the assigned resources. At present, only assignment of a single CPU core and GPU is supported per trajectory. In a future work we plan to extend Lattice Microbes to allow multiple CPU cores and GPUs, both intra- and inter-node, to cooperate in order to decrease the wall time required to calculate an individual trajectory.

When sampling the CME and RDME, complete individual trajectories are often needed to reconstruct the full distribution of the system's time evolution or to compare with experimentally observed behavior; the first few moments of the distribution may not be sufficient. When generating 10,000-100,000+ trajectories in parallel on a large cluster, though, organizing them as individual files could overload a network file system and thus

data output would become a bottleneck. Instead, Lattice Microbes streams each trajectory's state along with other statistical data over low-latency interconnects (if available) and outputs them into a single HDF5 formatted file using a dedicated output thread. The HDF5 format supports efficient, independent storage of many large data sets in a single file.

Lattice Microbes allows trajectory state to be output at either every reaction event or at a specified time interval. For both CME and RDME trajectories the total count of each species is saved in one data set and for RDME trajectories the complete lattice state is saved in another (optionally using a different output interval). The first passage time for a species to reach a given count can also be exactly tracked to facilitate analysis of rare events. The HDF5 files can be directly read by Matlab, Python, and other tools for later numerical analysis. Furthermore, a single-process scripted version of Lattice Microbes allows Python scripts to be written using application primitives to programmatically construct and analyze the simulation files. Reaction models contained in SBML [45] files can also be imported for initial setup of simulations. COPASI [46] and Virtual Cell [47] are two popular biochemical modeling programs that provide GUI tools for constructing SBML files that are suitable for use with Lattice Microbes.

2.6 Visualization

Lattice Microbes includes a plug-in for VMD [29] that enables VMD to read simulation trajectories for visualization and analysis. Contemporary with development of the Lattice Microbes software, we have modified and in some cases redesigned the data structures and visualization algorithms in VMD so that it can visualize cellular models and animate cellular simulation trajectories (see Figure 3) using its built-in graphical representations. The combination of Lattice Microbes with VMD provides a powerful system for model preparation and verification, simulation, visualization, and analysis. VMD can display both the continuous-space cellular model and its coarse-grained lattice representation. Superimposing both of the models allows direct comparison of the two to ensure that a coarse-grained model captures the essential features of the full model prior to beginning a simulation with the Lattice Microbes software.

Cellular models typically contain tens-to-hundreds of millions of particles, and cellular simulation trajectories often store thousands of frames, collectively requiring hundreds of gigabytes to several terabytes of storage. As an example, a previously published simulation of an *E. coli* cell [7], involved 20,000 active particles and 600,000 obstacles and required 500 simulation runs, each containing roughly 5000 timesteps. The total storage required for each simulation trajectory was roughly 1.5 GB, for a total of 750 GB in all.

In order to allow large models to be loaded into the limited amount of host and GPU memory for simulation, visualization, and analysis, the Lattice Microbes and VMD software packages use highly compact memory representations that currently support up to 256 particle types. A larger number of particle types can be represented with a commensurate increase in per-particle memory use by changing the internal particle data structures in Lattice Microbes and VMD to use larger word sizes. Increasing the data structures to use a two-byte word size size would allow 65,536 particle types to be represented, but it would reduce simulation performance significantly and decrease the maximum cell size that could be simulated with a single-GPU algorithm. Future multi-GPU implementations of Lattice Microbes will make it feasible to increase these limits since the simulations will no longer be constrained to the memory capacity and performance of a single GPU. When animating trajectories that are larger than the available host memory in a graphics workstation, VMD allows the user to load a subset of trajectory frames, by selecting starting and ending frames, and by optionally skipping frames. Recently, VMD has been adapted to allow interactive trajectory animations using out-of-core data access in concert with solid state disks (SSDs),

which allows the user to view arbitrarily long trajectories, limited only by SSD storage capacity [48].

VMD allows particles or subvolumes to be selected for display with an easy-to-use text-based selection language that is used within its graphical interfaces and in user-written analysis scripts or plug-ins. The selection language allows particles to be selected through combinations of criteria such as particle names, types, various physical properties, and by their location or their spatial relationship with other particles. For example, in a simulation of a dividing *E. coli* cell one might select all of the FtsZ molecules within a certain distance of the septum, which would facilitate potential analysis of any ring-like structures. Once selected for display, the individual particles or subvolumes can be represented using glyphs such as points, discs, or space filling spheres.

In systems with tens-to-hundreds of millions of particles aggregated representations will be an absolute necessity. It is clear that viewing of such large models as discrete particles has limited value given that the particles would outnumber the pixels on a typical display by a factor of one hundred or more. VMD allows groups of selected particles to be collectively visualized using a fast Gaussian density isosurface representation that encloses selected particles within smooth surfaces [49]. Such surfaces can be used to represent cellular structures with a level of structural detail that is both appropriate for interactive visual exploration, and reduces the graphics workload, thereby increasing the interactive display performance. Cellular boundaries are represented with interior and exterior triangle meshes, and are typically rendered with a high degree of transparency so that the interior of the cell can be seen clearly.

VMD supports a wide array of 3-D display and input technologies [50], and uses programmable shading and GPU computing [48] to allow the user to interactively explore Lattice Microbes models containing tens-to-hundreds of millions of particles with stereoscopic 3-D displays. Lattice Microbes models can also be rendered (non-interactively) using photorealistic lighting and shading techniques. The images shown in Figure 3 demonstrate the use of ambient occlusion lighting and shadowing [51] to improve perception of channels and pockets, and were rendered using the Tachyon parallel ray tracing engine built-into VMD [52]. VMD also allows Lattice Microbes models to be exported to many different geometric file formats used by professional rendering and animation packages, 3-D solid model printers, and web-based 3-D model viewers.

3 Results and Discussion

3.1 Accuracy of the MPD-RDME method

To study the accuracy of our MPD-RDME method, we compared the results of sampling the RDME using the approximate MPD-RDME against 100,000 trajectories sampled using the exact next-subvolume method. The reversible bimolecular reaction $A+B\underset{k_2}{\overset{k_1}{\rightleftharpoons}}C$ is a simple reaction system with deviations from the well-stirred approximation of the CME and served as a test case for our comparison. The simulation parameters used were arbitrarily chosen to be biologically representative and to maintain consistency with the results presented in the Supporting Information. Using a lattice of size $\mathcal{L} = 32 \times 32 \times 32$, a lattice spacing of $\lambda = 31.25 \times 10^{-9} m$, and a diffusion coefficient for all molecules of $D = 8.15 \times 10^{-14} m^2 s^{-1}$, one can calculate the mean time until a molecule experiences a diffusion event to any neighboring subvolume as $\tau D = \frac{\lambda^2}{6D} = 2.0 \times 10^{-3} s$. Likewise, using $k_1 = 1.07 \times 10^5\ M^{-1}\ s^{-1}$ and $k_2 = 0.351\ s^{-1}$, one can calculate the mean time until an $A + B \rightarrow C$ reaction (assuming

one A and one B in a subvolume) as $\tau_{R1}=\frac{1}{k_1\cdot[B]}=\frac{N_A\cdot\lambda^3\cdot 1000}{k_1}=1.7\times10^{-1}s$ and the mean time for a $C\rightarrow A+B$ reaction (assuming one C molecule in a subvolume) as $\tau_{R2}=\frac{1}{k_2}=2.8s$. So the condition $\tau_D\ll min(\tau_{R1},\tau_{R2})$ is satisfied.

As described above, the MPD-RDME algorithm requires time steps to be chosen such that there is a low probability of the fastest reaction happening during each time step. For the reaction rate constants used, the fastest reaction is the bimolecular reaction with a rate in a single subvolume (assuming one A and B pair) of $k=\frac{k_1}{N_A\cdot\lambda^3\cdot 1000}=5.82s^{-1}$. The probability of a reaction occurring in a time step of duration τ is then given by $\int_0^\tau ke^{-kt}dt=1-e^{-k_1\tau}$. Using 2% as the threshold for a reaction to occur, one obtains the maximum acceptable time step $\tau=\frac{ln0.98}{-5.82}=3.5\times10^{-3}s$. Similarly, the MPD-RDME method imposes restrictions on the maximum time step according to the diffusion coefficient of the fastest diffusing species: $\tau\leq\frac{1}{2}\frac{\lambda^2}{2D}=3.0\times10^{-3}s$. We choose the lower of the two values and set our maximum time step to be $3.0\times10^{-3}s$. To study the effect of the time step on the error we also tested the shorter time steps 1.5×10^{-3} *s*, 6.0×10^{-4} *s*, and 3.0×10^{-4} *s*.

Figure 4(a+b) shows the accuracy of the time-dependent PDF reconstructed by sampling using the MPD-RDME with the different time steps. For the longest time step, the error in the expected value $Err(E\{A(t)\})=\frac{\left|E\{A_{MPD}(t)\}-E\{A_{NSM}(t)\}\right|}{E\{A_{NSM}(t)\}}$ is always below 2×10^{-3}. As the time step decreases, the error in $E\{A(t)\}$ also decreases. For the shortest time step, the error in $E\{A(t)\}$ falls to $\sim1\times10^{-4}$. Figure 4(d+e) shows that the error in the variance is $\sim1\times10^{-2}$ for all of the time steps, suggesting that time stepping in the algorithm is not the primary source of error for $Var\{A(t)\}$.

Since we are reconstructing the PDF using a number of trajectories, there is also error due to the finite sample size. To study this effect, we calculated the mean error in $E\{A(t)\}$ and $Var\{A(t)\}$ as a function of the number of trajectories used (see Figure 4[c+f]). For fewer than ~250 trajectories the sample size error dominates the error in $E\{A(t)\}$ and all of the time steps show equivalent error as, in fact, does the next-subvolume method. At ≥500 trajectories the time step error begins to dominate in the longest time step simulations. Similarly, for each time of the next two time steps the error levels off at a finite value once the number of trajectories reaches a certain threshold. The shortest time step, however, is indistinguishable from the next-subvolume method through the range of sampling performed. For $Var\{A(t)\}$ the sample size error dominates the time step error through 100,000 trajectories.

Overall, the approximate MPD-RDME method, with careful selection of an appropriate time step, compares favorably with the next-subvolume method in terms of accuracy. When using more than a few hundred trajectories to reconstruct the PDF a shorter time step must be chosen, with the ensuing performance slowdown, to ensure that the error introduced by the time stepping algorithm is not greater than the error due to finite sampling. Further studies on the accuracy of both the CME and RDME sampling methods implemented in the Lattice Microbes software are detailed in the Supporting Information.

3.2 Simulation performance

To asses the performance of our implementations of the various CME and RDME sampling methods in Lattice Microbes, we constructed three test cases. The first case was the low-complexity reversible bimolecular reaction systems described earlier. There were 2 reactions and 3 species with approximately 1700 total molecules. For the RDME simulations the volume was divided into a lattice of size $32 \times 32 \times 32$ with $\lambda = 31.25 \times 10^{-9}$ m. For the MPD-RDME simulations $\tau = 3 \times 10^{-3}$ s.

The second test case was the *lac* genetic switch in a modeled *E. coli* cell, as described in [7]. For this test case the external inducer concentration was 5 μM and the system was in the uninduced state. There were 23 reactions and 12 species with ~5,000 active molecules and ~600,000 fixed obstacles representing molecular crowding. RDME simulation were performed on a lattice of size $64 \times 64 \times 128$ with $\lambda = 16 \times 10^{-9}$ m containing a cell of length 2 μm and diameter 0.8 μm. For the MPD-RDME simulations $\tau = 50 \times 10^{-6}$ s.

The third test case was the *lac* genetic switch with an external inducer concentration of 40 μM, which switched the system to the induced state. The reaction scheme was the same as in test case two, but there were 1.1 million active molecules in the system. To accommodate the increased molecule count, the RDME simulations were run on a lattice of size $128 \times 128 \times 256$ and $\lambda = 8 \times 10^{-9}$ m. For the MPD-RDME simulations $\tau = 12.5 \times 10^{-6}$ s.

For each test case, we collected performance data and determined both the average rate at which reactions were being processed by the system and the total simulation time that could be computed by a single CPU core (and a single GPU for MPD-RDME simulations) in one hour of wall time. Simulations were performed on a local cluster consisting of four nodes, each containing 2 × 6-core 2.66 GHz Xeon X5650 CPUs and 4 Tesla C2050 GPUs. During Lattice Microbes performance tests, 48 simultaneous trajectories were started for the exact sampling methods and 16 simultaneous trajectories were started for the MPD-RDME method to fully occupy the cluster. Since each trajectory is calculated independently in the current version, essentially perfect scaling is achieved and we therefore only report the performance per core. The other software tested has varying support for parallel execution so performance tests for these packages were run on only a single core of the cluster. Each test was performed ten times and here the mean execution time is reported.

The upper half of Table 1 shows the performance data for our CME sampling implementations as well as a comparison with two other simulation packages capable of performing Gillespie simulations. For the low complexity reversible bimolecular simulations, a 2.5× speedup in the direct method was achieved by using the GPU for random number generation. This speedup implies that more than half of the calculation time in the CPU-only code is spent generating random numbers. The remaining code for processing the reaction events and updating the propensities takes ~110 clock cycles per reaction event. For the high complexity lac simulations, using the GPU gives a smaller 1.5× speedup, as each reaction event requires more updates to the propensity tables with less relative time spent generating random numbers. Interestingly, even for the lac system with 23 reactions the next-reaction method (which is more efficient at updating propensities) is not able to outpace the direct method. For our implementation the cross-over point appears to be somewhat greater than 23 reactions.

The performance data for our RDME sampling implementations are shown in the lower half of Table 1. In the next-subvolume method, compared to exact CME samplers, much less time is spent generating random numbers; using the GPU for random number generation provides only a 1.1× speedup. In the next-subvolume simulations there are many diffusion events per reaction event. For the reversible bimolecular reaction scheme there are ~4500

diffusion events for each reaction event. The overall rate of processing events is then 1.7×10^6 events/sec, corresponding to ~1500 clock cycles per event or more than 13× longer than the direct CME sampling method. There is likely still room for performance improvement in the next-subvolume solver, but random number generation is much less constraining for the RDME.

For low-molecule-count simulations, the MPD-RDME method provides a 4.5×–7.5× performance improvement compared to exact next-subvolume method. For high-molecule-count simulations the speedup provided by exploiting the GPU and the fine-grained parallelism in the MPD-RDME method increase this performance advantage to 300×. Unlike the simulation method using the GPU only for random number generation, the MPD-RDME runs the entire simulation algorithm on the GPU and is therefore able to achieve a dramatic speedup. However, the number of trajectories that can be simultaneously generated is limited by the number of GPUs.

Lattice Microbes is a high-performance code for sampling the CME and RDME using both exact and approximate methods and provides a significant performance advantage compared to other simulation packages tested. Configuration and SBML files and basic instructions to run the performance tests on all tested software are located in the Supporting Information and the User's Guide (http://www.scs.illinois.edu/schulten/lm).

3.3 Rebinding of transcription factors under *in vivo* conditions

As an illustration of a biochemical research problem for which the Lattice Microbes software is intended, we present a problem in modeling stochastic gene expression. A transcription factor acting as a repressor generally inhibits expression of a target gene by binding to a DNA sequence upstream of the gene's promoter thus blocking RNA polymerase (RNAP) from binding. Each time the repressor unbinds from its DNA binding site one or more RNAP molecules are able to bind to the promoter creating a burst of mRNA and consequent proteins. The duration of the unbinding event determines the size of the burst. In well-stirred models of repressor unbinding and binding, the burst duration is determined solely by the first time for any free repressor to bind from bulk and is therefore exponentially distributed with a mean determined by the binding rate constant and number of free repressors. In models accounting for diffusion, however, there is a short term memory due to the localization of the newly unbound repressor until it diffuses into the bulk. This memory enhances the probability of quick rebinding events and modifies the overall distribution of binding times [53].

An additional memory effect immediately following repressor unbinding is caused by *in vivo* crowding, which introduces local trapping of the repressor on short timescales. This trapping introduces an anomalous component in the repressor's diffusion at a certain timescale dictated by the size of the crowders [7]. If this timescale coincides with the timescale of the binding kinetics, the rebinding statistics can be further modified from those expected from a well-stirred model. Here we studied the effect of crowding on rebinding by simulating escape to bulk versus rebinding of a repressor following unbinding from its DNA binding site.

The simulated system consisted of a spherical reaction volume 200 nm in radius with an absorbing boundary condition and a DNA binding site located at the center (see Figure 5[a]). The DNA binding site was considered to be a single point such that we ignored the role of 1D sliding in transcription factor dynamics. The reaction volume was filled to a variable packing fraction (fraction of the total volume occupied) with stationary obstacles distributed according to an *in vivo* approximation described previously [20, 27].

At the beginning of a simulation, a single repressor was located in the same subvolume as the DNA binding site as if it had unbound at $t = 0$. The simulation was then run until the repressor either rebound with the DNA or escaped into the bulk (reached the absorbing boundary). The repressor diffusion coefficient was $D = 1 \times 10^{-11}$ m^{2}s^{-1} and the binding rate for the repressor was $K = 2.43 \times 10^6$ M^{-1}s^{-1}. 52,500 RDME trajectories were generated for each packing fraction using the next-subvolume solver with a 2 nm subvolume length. As discussed earlier, the RDME is valid under the condition that each subvolume can be considered well-stirred, *i.e.*, $\tau_D \ll \tau_R$. For this system, $\tau_D = \frac{\lambda^2}{D} = 4 \times 10^{-7} s$ and $\tau_R = \frac{1}{K \cdot [R]} = \frac{N_A \cdot \lambda^2 \cdot 1000}{K} = 1.98 \times 10^{-6}$, which satisfies the condition.

Results from the simulations indicate that *in vivo* crowding affects both the rebinding probability and the rebinding time distribution. As shown in Figure 5(b), as the packing fraction increases so too does the probability that a newly unbound repressor will rebind to the DNA before escaping to the bulk. As binding from the bulk generally takes much longer than rebinding, systems with *in vivo* crowding will have more extremely short bursts and proportionately fewer long bursts. At the same time, the short bursts due to rebinding become longer in duration (see Figure 5[c]). Without packing, virtually all of the rebinding events occur by 1 μs while at 60% packing, rebinding events can last up to 100 μs. Whether the elongated burst duration has a noticeable effect on bursts statistics would depend on the rate of RNAP binding, which is system dependent. But the loss of some fraction of long binding events certainly modifies the burst frequency statistics.

Additional insight into the memory induced by *in vivo* crowding can be obtained by measuring how far a repressor must diffuse before the memory effects are lost. In the simulations with no packing, the timescale for rebinding is 1 μs, which means that once the repressor has diffused ~8 nm away (using $\langle r^2 \rangle = 6Dt$) there is little probability that it will rebind. At 60% packing the effective D becomes 3.7×10^{-12} m^{2}s^{-1} and the corresponding distance before correlation is lost ~50 nm. *In vivo* packing thus introduces correlations across a much larger volume of reaction space than in freely diffusing systems.

4 Conclusions

The Lattice Microbes software package enables efficient sampling of the CME and RDME on HPC plat-forms. Using GPU accelerators, it is from 1.75×-4× faster than other tested codes for exact sampling of the CME. Using GPU acceleration one can perform longer CME simulations in the same wall clock compared to non-GPU accelerated code. Importantly, one CPU core does not saturate a single GPU and multiple CPU cores can therefore share a GPU for random number generation with no performance impact. This represent a different way to use GPUs to accelerate CME sampling than has been presented previously [54] and is optimized for accelerating individual trajectories to sample rare events ratherthan for increasing the number of trajectories generated.

The Lattice Microbes package provides high performance exact and approximate methods for sampling the RDME. Both the exact and approximate methods are significantly faster than other tested software. Our MPD-RDME method can take advantage of attached GPUs or other many-core processors to accelerate calculation of *in vivo* cell models. Although the MPD-RDME algorithm was implemented in the NVIDIA CUDA programming language and targets the current generation of GPU devices, it leverages features of GPU hardware architecture that are common to devices made by several hardware vendors. Due to GPUs' performance and energy efficiency advantages for many scientific applications [55], they are now being incorporated into state-of-the-art supercomputers. We expect that the key

attributes of today's massively parallel GPU hardware architecture will continue in future designs, and that over the next ten years, some of these features will be incorporated into CPUs and into other many-core processors, such as the Intel MIC architecture, and other microprocessors targeting high performance computing workloads. The fine-grained parallelism provided by the MPD-RDME method will continue to make it ideally suited to GPUs and next-generation many-core microprocessor designs.

Code and binaries for Linux and Mac OS X are available at the project web page http://www.scs.illinois.edu/schulten/lm. Sample submission scripts are available for national supercomputing resources and local compute clusters. Virtual machine builds are also available for deployment on the Amazon EC2 compute cloud under the name “LatticeMicrobes”.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

Contract grant sponsor: DOE (Office of Science BER); Contract grant number: DE-FG02-10ER6510; Contract grant sponsor: NIH (Center for Macromolecular Modeling and Bioinformatics); Contract grant number: NIH-RR005969; Contract grant sponsor: NSF; Contract grant number: MCB-0844670; Contract grant sponsor: NSF (Center for the Physics of Living Cells); Contract grant number: PHY-0822613

References

1. Eldar A, Elowitz MB. Nature. 2010; 467:167–73. [PubMed: 20829787]
2. Li GW, Xie XS. Nature. 2011; 475:308–15. [PubMed: 21776076]
3. Golding I. Annu Rev Biophys. 2011; 40:63–80. [PubMed: 21545284]
4. Munsky B, Neuert G, van Oudenaarden A. Science. 2012; 336:183–7. [PubMed: 22499939]
5. McAdams HH, Arkin A. Proc Natl Acad Sci USA. 1997; 94:814–9. [PubMed: 9023339]
6. Gillespie DT. Annu Rev Phys Chem. 2007; 58:35–55. [PubMed: 17037977]
7. Roberts E, Magis A, Ortiz JO, Baumeister W, Luthey-Schulten Z. PLoS Comput Biol. 2011; 7:e1002010. [PubMed: 21423716]
8. Gardiner CW, McNeil K, Walls D, Matheson I. J Stat Phys. 1976; 14:307–31.
9. Isaacson S. SIAM J Appl Math. 2009; 70:77–111.
10. Isaacson SA, Isaacson D. Phys Rev E. 2009; 80:066106.
11. Erban R, Chapman SJ. Phys Biol. 2009; 6:046001. [PubMed: 19700812]
12. Fange D, Berg OG, SjÖberg P, Elf J. Proc Natl Acad Sci USA. 2010; 107:19820–5. [PubMed: 21041672]
13. Elf J, Ehrenberg M. IEE Syst Biol. 2004; 1:230–6.
14. Hattne J, Fange D, Elf J. Bioinformatics. 2005; 21:2923–4. [PubMed: 15817692]
15. Rodríguez JV, Kaandorp JA, Dobrzyński M, Blom JG. Bioinformatics. 2006; 22:1895–901. [PubMed: 16731694]
16. Dobrzynski M, Rodríguez JV, Kaandorp JA, Blom JG. Bioinformatics. 2007; 23:1969–77. [PubMed: 17537752]
17. Lampoudi S, Gillespie DT, Petzold LR. J Chem Phys. 2009; 130:094104. [PubMed: 19275393]
18. Drawert B, Lawson MJ, Petzold L, Khammash M. J Chem Phys. 2010; 132:074101. [PubMed: 20170209]
19. Ferm L, Hellander A, LÖtstedt P. J Comput Phys. 2010; 229:343–60.
20. Roberts, E.; Stone, JE.; Sepulveda, L.; M, W.; Hwu, W.; Luthey-Schulten, Z. Long time-scale simulations of in vivo diffusion using GPU hardware. IEEE; 2009.

21. Ortiz JO, FÖrster F, Kürner J, Linaroudis AA, Baumeister W. J Struct Biol. 2006; 156:334–41. [PubMed: 16857386]
22. Comolli LR, Baker BJ, Downing KH, Siegerist CE, Banfield JF. ISME J. 2009; 3:159–67. [PubMed: 18946497]
23. Briegel A, Ortega DR, Tocheva EI, Wuichet K, Li Z, Chen S, Müller A, Iancu CV, Murphy GE, Dobro MJ, Zhulin IB, Jensen GJ. Proc Natl Acad Sci USA. 2009; 106:17181–6. [PubMed: 19805102]
24. Beck M, MalmstrÖm JA, Lange V, Schmidt A, Deutsch EW, Aebersold R. Nat Meth. 2009; 6:817–23.
25. Kühner S, van Noort V, Betts MJ, Leo-Macias A, Batisse C, Rode M, Yamada T, Maier T, Bader S, Beltran-Alvarez P, Castaño-Diez D, Chen WH, Devos D, Güell M, Norambuena T, Racke I, Rybin V, Schmidt A, Yus E, Aebersold R, Herrmann R, Bo¨ttcher B, Frangakis AS, Russell RB, Serrano L, Bork P, Gavin AC. Science. 2009; 326:1235–40. [PubMed: 19965468]
26. Banks DS, Fradin C. Biophys J. 2005; 89:2960–71. [PubMed: 16113107]
27. Ridgway D, Broderick G, Lopez-Campistrous A, Ru'aini M, Winter P, Hamilton M, Boulanger P, Kovalenko A, Ellison MJ. Biophys J. 2008; 94:3748–59. [PubMed: 18234819]
28. McGuffee SR, Elcock AH. PLoS Comput Biol. 2010; 6:e1000694. [PubMed: 20221255]
29. Humphrey W, Dalke A, Schulten K. J Mol Graph. 1996; 14:33–8. [PubMed: 8744570]
30. van Kampen, NG. Stochastic Processes in Physics and Chemistry. Elsevier; Oxford, UK: 2007.
31. Gardiner, CW. Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences. Springer; New York, NY: 2004.
32. McQuarrie D. J Appl Probab. 1967; 4:413–78.
33. Gillespie DT. J Phys Chem. 1977; 81:2340–61.
34. Gibson M, Bruck J. J Phys Chem. 2000; 104:1876–89.
35. Cao Y, Gillespie DT, Petzold LR. J Chem Phys. 2006; 124:044109. [PubMed: 16460151]
36. Tian T, Burrage K. J Chem Phys. 2004; 121:10356–64. [PubMed: 15549913]
37. Auger A, Chatelain P, Koumoutsakos P. J Chem Phys. 2006; 125:084103. [PubMed: 16964997]
38. Marquez-Lago TT, Burrage K. J Chem Phys. 2007; 127:104101. [PubMed: 17867731]
39. Chopard, B.; Droz, M. Cellular Automata Modeling Of Physical Systems. Cambridge University Press; Cambridge, UK: 1998.
40. Arjunan SNV, Tomita M. Syst Synth Biol. 2010; 4:35–53. [PubMed: 20012222]
41. Lis M, Artyomov MN, Devadas S, Chakraborty AK. Bioinformatics. 2009; 25:2289–91. [PubMed: 19578038]
42. Vigelius M, Lane A, Meyer B. Bioinformatics. 2011; 27:288–90. [PubMed: 21062761]
43. Komarov I, D'Souza RM, Tapia JJ. PLoS One. 2012; 7:e37370. [PubMed: 22715366]
44. Marsaglia G, Stat J. Software. 2003; 8:1–6.
45. Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, Cuellar AA, Dronov S, Gilles ED, Ginkel M, Gor V, Goryanin II, Hedley WJ, Hodgman TC, Hofmeyr JH, Hunter PJ, Juty NS, Kasberger JL, Kremling A, Kummer U, Le Novère N, Loew LM, Lucio D, Mendes P, Minch E, Mjolsness ED, Nakayama Y, Nelson MR, Nielsen PF, Sakurada T, Schaff JC, Shapiro BE, Shimizu TS, Spence HD, Stelling J, Takahashi K, Tomita M, Wagner J, Wang J, Forum S. Bioinformatics. 2003; 19:524–31. [PubMed: 12611808]
46. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, Singhal M, Xu L, Mendes P, Kummer U. Bioinformatics. 2006; 22:3067–74. [PubMed: 17032683]
47. Moraru II, Schaff JC, Slepchenko BM, Blinov ML, Morgan F, Lakshminarayana A, Gao F, Li Y, Loew LM. IET Syst Biol. 2008; 2:352–62. [PubMed: 19045830]
48. Stone JE, Vandivort KL, Schulten K. Lect Notes in Comp Sci. 2011; 6939:1–12.
49. Krone M, Stone JE, Ertl T, Schulten K. EuroVis 2012 – Short Papers. 2012:67–71.
50. Stone JE, Kohlmeyer A, Vandivort KL, Schulten K. Lect Notes in Comp Sci. 2010; 6454:382–93.
51. Zhukov S, Iones A, Kronin G. Rendering Techniques 98 (Proceedings of EuroGraphics Workshop on Rendering). 1998:45–55.

52. Stone, JE. An Efficient Library for Parallel Ray Tracing and Animation Master's thesis. Computer Science Department; University of Missouri at Rolla: 1998.

53. van Zon JS, Morelli MJ, Tãnase-Nicola S, ten Wolde PR. Biophys J. 2006; 91:4350–67. [PubMed: 17012327]

54. Li H, Petzold L. Int J High Perform Comput Appl. 2010; 24:107–16.

55. Enos J, Steffen C, Fullop J, Showerman M, Shi G, Esler K, Kindratenko V, Stone JE, Phillips JC. International Conference on Green Computing. 2010:317–324.

56. Sanft KR, Wu S, Roh M, Fu J, Lim RK, Petzold LR. Bioinformatics. 2011; 27:2457–8. [PubMed: 21727139]

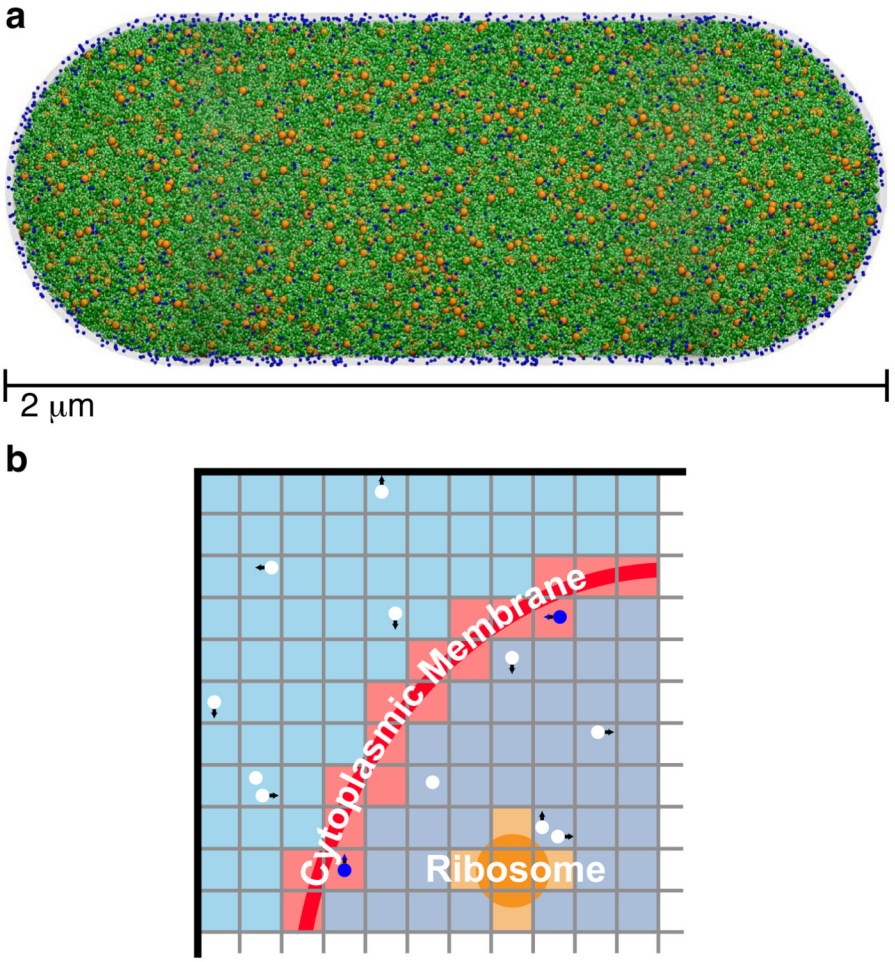

Figure 1.
(a) Model of a crowded *E. coli* cell with *in vivo* packing. (b) Schematic diagram of the *in vivo* RDME method.

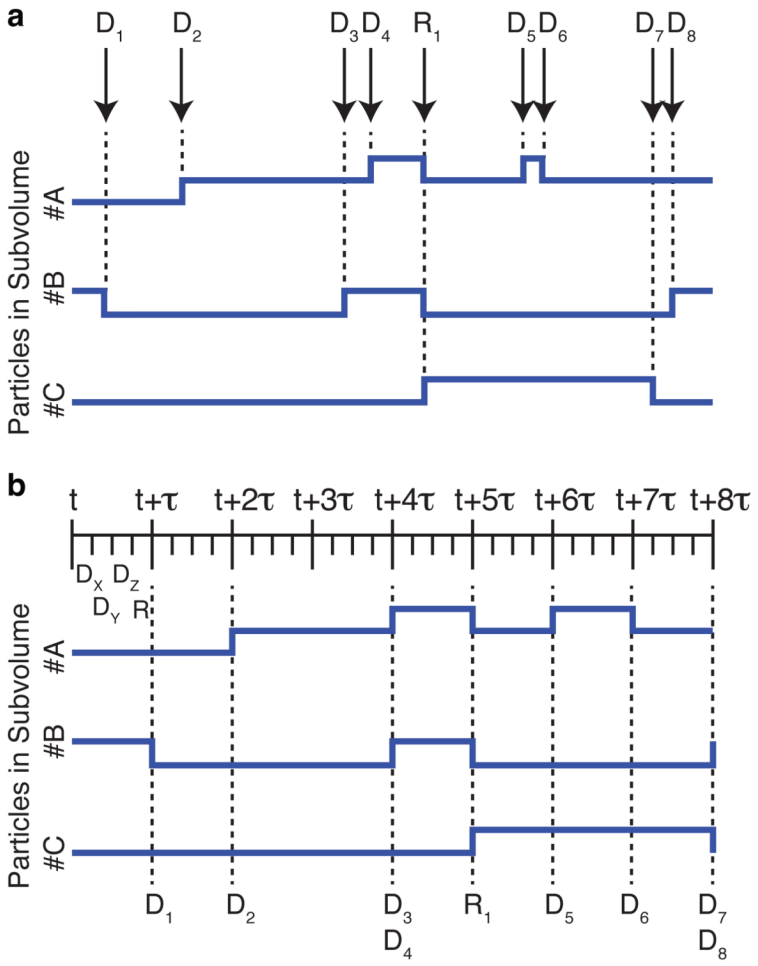

Figure 2.

Example history of a single subvolume from an RDME simulation with molecule types *A*, *B*, and *C* and the reversible bimolecular reaction $A + B \leftrightharpoons C$. Shown are (a) a Gillespie sampling of the RDME and (b) an MPD-RDME sampling. Diffusion events are annotated as D_N and reaction events as R_N.

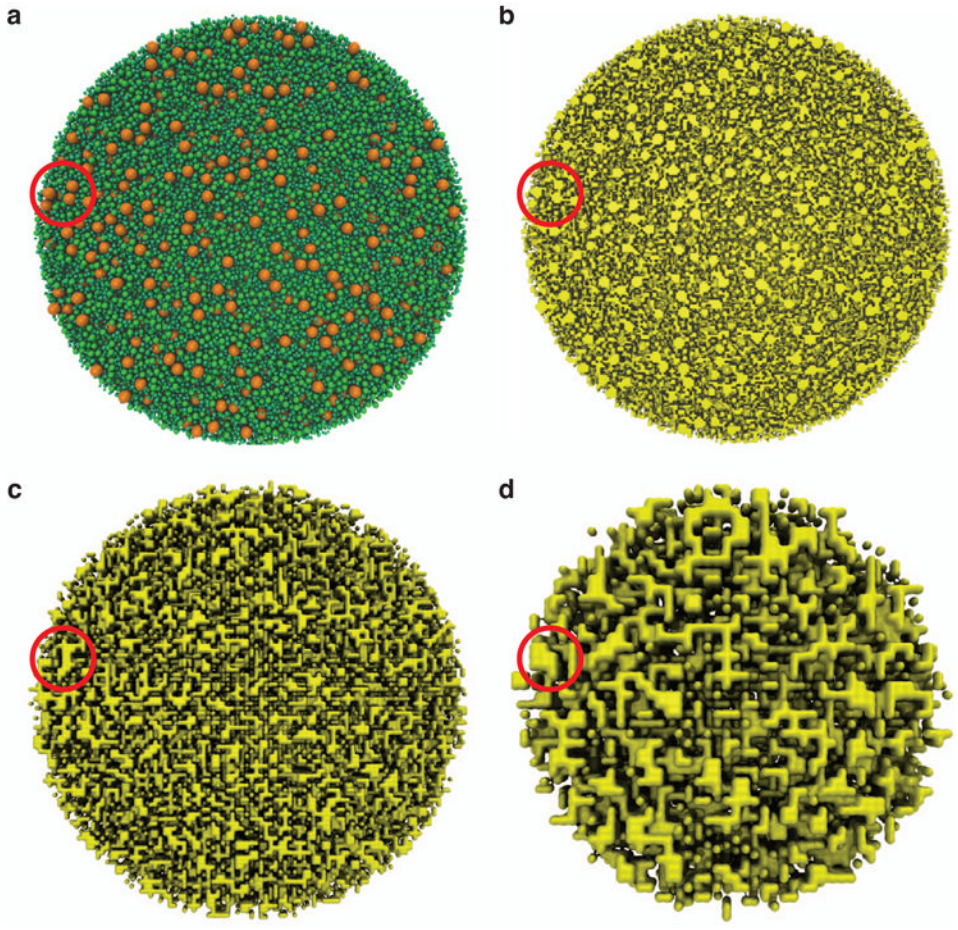

Figure 3.

Visualization of a slice through an *E. coli* cell model with *in vivo* crowding occupying 40% of the total volume. (a) Continuous-space model using spheres to represent the obstacles. Orange spheres represent ribosomes and green spheres represent proteins and protein complexes. (b) Coarse-grained model of the same slice shown in (a) using 4nm subvolumes. Occupied subvolumes are shown in yellow using a surface representation. (c+d) As in (b) for 8nm and 16 nm, respectively. Per-pixel shading, depth cueing, shadows, and ambient occlusion lighting techniques are used to enhance spatial perception in these crowded renderings.

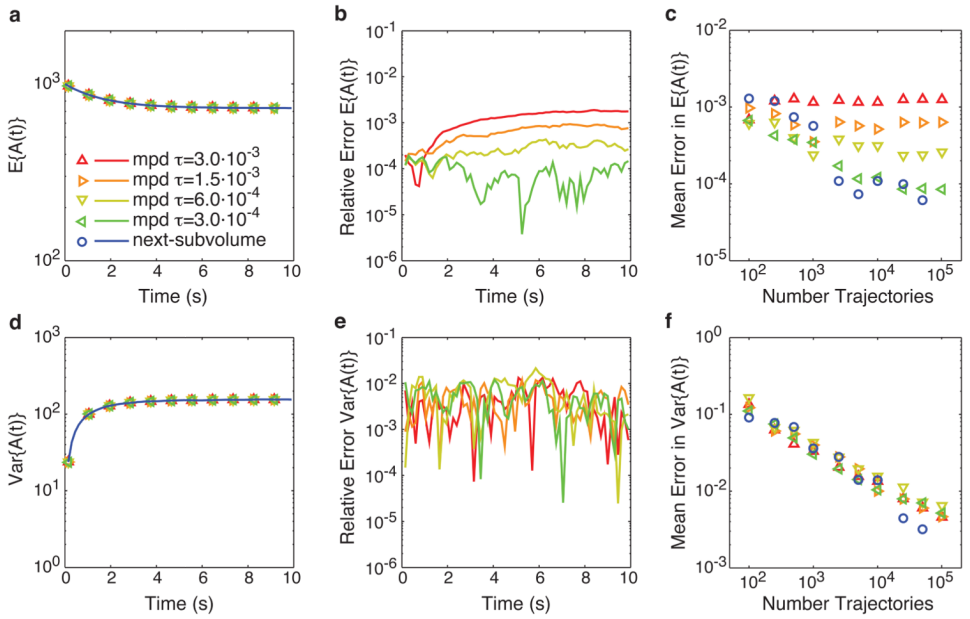

Figure 4.

Accuracy of the MPD-RDME method for sampling the RDME. (a) Expectation value of $A(t)$ calculated from 100,000 MPD-RDME trajectories using time steps of (red Δ) 3.0×10^{-3} *s*, (orange ▸) 1.5×10^{-3} *s*,(yellow ▽) 6.0×10^{-4} *s*, and (green ◂) 3×10^{-4} *s* compared to the expected value obtained from (blue ○) 100,000 next-subvolume trajectories. (b) The relative error in $E\{A(t)\}$ as calculated from the MPD-RDME trajectories. Errors were calculated relative to the distribution from 100,000 next-subvolume trajectories. (c) The mean relative error in $E\{A(t)\}$ versus the number of trajectories used to reconstruct the probability distribution. (d,e,f) As in (a,b,c) for the variance of $A(t)$. Reaction parameters are $A_0 = 1000$, $B_0 = 1000$, $C_0 = 0$, $k_1 = 1.07 \times 10^5\ M^{-1}\ s^{-1}$ and $k_2 = 0.351\ s^{-1}$. Diffusion parameters are $\mathcal{L}$, $= 32 \times 32 \times 32$, $\lambda = 31.25 \times 10^{-9}\ m$, and $D = 8.15 \times 10^{-14}\ m^2\ s^{-1}$.

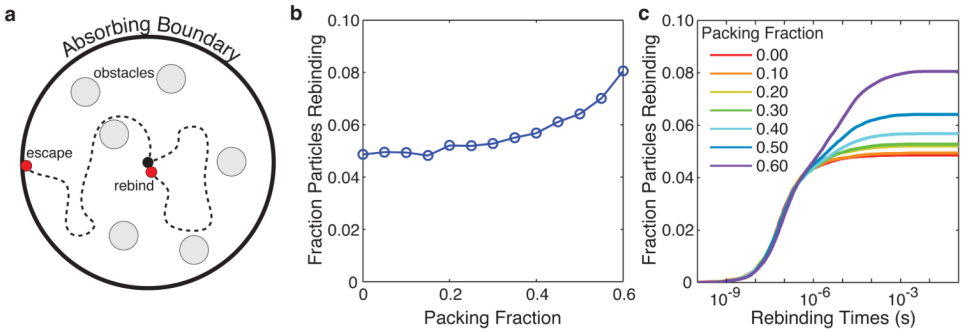

Figure 5.

Results of *in vivo* transcription factor rebinding simulations. (a) Diagram of the simulation system. (b) Fraction of repressors that rebind versus escape to bulk as a function of the reaction volume occupied by *in vivo* obstacles. (c) The cumulative fraction of particles that rebind by a given time.

Table 1

Performance comparison of CME and RDME sampling methods*

	Reversible Bimolecular		Lac Switch, [I]=5μM		Lac Switch, [I]=40μM	
Method	**R×ns/sec**	**sim/clock·hr**	**r×ns/sec**	**sim/clock·hr**	**r×ns/sec**	**sim/clock·hr**
CME						
Direct GPU	24.1×10^6	4.6×10^8 sec	7.3×10^6	7.3×10^8 sec	7.4×10^6	3.2×10^6 sec
Direct No-GPU	9.5×10^6	1.8×10^8 sec	5.0×10^6	5.0×10^8 sec	5.0×10^6	2.2×10^6 sec
Next React GPU	14.5×10^6	2.8×10^8 sec	5.5×10^6	5.5×10^8 sec	6.2×10^6	2.7×10^6 sec
Next React No-GPU	7.7×10^6	1.5×10^8 sec	3.9×10^6	4.0×10^8 sec	4.5×10^6	2.0×10^6 sec
Stochkit SSA [56]	6.0×10^6	1.1×10^8 sec	4.2×10^6	4.2×10^8 sec	4.2×10^6	1.8×10^6 sec
COPASI [46]	6.2×10^6	1.2×10^8 sec	2.7×10^6	2.7×10^8 sec	2.8×10^6	1.2×10^6 sec
RDME						
MPD-RDME GPU	2.92×10^3	55.4×10^3 sec	4.4×10^{-1}	79.5×10^0 sec	1.200×10^{-1}	2.160×10^0 sec
Next Subvol GPU	0.38×10^3	7.2×10^3 sec	1.0×10^{-1}	18.0×10^0 sec	0.004×10^{-1}	0.007×10^0 sec
Next Subvol No-GPU	0.34×10^3	6.5×10^3 sec	1.0×10^{-1}	17.3×10^0 sec	0.004×10^{-1}	0.007×10^0 sec
MesoRD [14]	0.05×10^3	0.9×10^3 sec	0.1×10^{-1}	1.8×10^0 sec	0.003×10^{-1}	0.006×10^0 sec

*per CPU core/GPU using a system with 2 × 6-core 2.66 GHz Xeon X5650 and 4 Tesla C2050

J Comput Chem. Author manuscript; available in PMC 2013 September 04.

Algorithm 1

the MPD-RDME algorithm

while $t < t_{end}$ **do**
  # Diffusion operator.
  for $dim \in \{x, y, z\}$ **do**
    parfor $\nu \in V$ **do**
      for *particle* $\in x_\nu$ **do**
        n = *uniformRand*()
        if $n \leq P(particle, D^{+dim})$ **then**
          move *particle* from subvolume x_ν to neighboring subvolume in the $+dim$ direction
        else if$(n - P(particle, D^{+dim})) \leq P(particle, D^{-dim})$ **then**
          move *particle* from subvolume x_ν to neighboring subvolume in the $-dim$ direction
        end
      end
    end
  end
  # Reaction operator.
  parfor $\nu \in V$ **do**
    for $r \in R$ **do**
      $a_{tot} = a_{tot} + a_r(x_\nu)$
    end
    n_1 = *uniformRand*()
    If $n_1 \leq \int_0^\tau a_{tot} e^{-a_{tot} t} dt$ **then**
      n_2 = *uniformRand*()
      for $r \in R$ **do**
        if $a_{r-1}(x\nu) < n_2 \cdot a_{tot} \leq a_r(x_\nu)$ **then**
          perform reaction r in subvolume x_ν
        end
      end
    end
  end
  $t = t + \tau$
end