Accurate reaction-diffusion operator splitting on tetrahedral meshes for parallel stochastic molecular simulations

I. Hepburn,1,2 W. Chen,1 and E. De Schutter1,2,a)
1Computational Neuroscience Unit, Okinawa Institute of Science and Technology Graduate University, Onna, Okinawa 904 0495, Japan
2Theoretical Neurobiology & Neuroengineering, University of Antwerp, Antwerp 2610, Belgium
(Received 8 December 2015; accepted 18 July 2016; published online 5 August 2016)

Spatial stochastic molecular simulations in biology are limited by the intense computation required to track molecules in space either in a discrete time or discrete space framework, which has led to the development of parallel methods that can take advantage of the power of modern supercomputers in recent years. We systematically test suggested components of stochastic reaction-diffusion operator splitting in the literature and discuss their effects on accuracy. We introduce an operator splitting implementation for irregular meshes that enhances accuracy with minimal performance cost. We test a range of models in small-scale MPI simulations from simple diffusion models to realistic biological models and find that multi-dimensional geometry partitioning is an important consideration for optimum performance. We demonstrate performance gains of 1-3 orders of magnitude in the parallel implementation, with peak performance strongly dependent on model specification. Published by AIP Publishing. [http://dx.doi.org/10.1063/1.4960034]

I. INTRODUCTION

Recent years have seen the rise of detailed spatial stochastic modeling, by voxel-based methods (e.g., STEPS,1 URDME,2 MesoRD,3 NeuroRD4) and particle tracking methods (e.g., Smoldyn,5 MCell6). Although the field is still in its infancy, the severe computational demands of tracking molecules individually or in small spatial discretizations whilst accurately simulating their movement and interactions means that models that run serially are very limited in their reach. Serial models can take many weeks or even months to run,7 placing restrictions on the scale of the model and results achievable in practical timeframes. Although the embarrassingly parallel solution of distributing different realizations is a useful form of parallelization, it is only a partial solution, and often a speed-up to the serial simulation is also necessary to make runtimes practical.

Based on Gillespie’s original Stochastic Simulation Algorithm8 (SSA) and extended for diffusion, the Inhomogeneous Stochastic Simulation Algorithm (ISSA) or similar exact methods9 are inherently serial and very little speedup is achieved by direct parallelization.10,11 Therefore, it is also broadly recognized that modifications to the exact algorithms of randomized event times are necessary in order to achieve a scalable solution. This paper will describe an approximate alternative to the ISSA for irregular meshes that is inherently parallel and includes novel modifications on previous methods to improve accuracy, and we will describe some important considerations for parallel performance in an initial parallel Message Passing Interface (MPI) implementation.

Attempts at applying approximations to ISSA methods for spatial reaction-diffusion simulation date back to 2006 with the “Gillespie Multi-Particle” (GMP) method,12 but are not often specifically tailored towards parallel computing per se. The GMP is a rather simple algorithm based on regular meshes that has been shown to be inaccurate under some conditions.13 However, the idea of applying operator splitting to ISSA methods (that is to separate reaction and diffusion algorithms from each other) for biological systems was introduced in this paper and one could say that the GMP forms the basis of all present operator splitting applications today, including the method introduced in this paper and implemented in STEPS.

The idea was enhanced by Lampoudi et al. in 2009 with the “Multinomial Simulation Algorithm” (MSA),14 which allows multi-step multinomial diffusion. They make the observation of an error in operator splitting methods of diffusion distance by restricting molecules to nearest neighbor, and go to great efforts to derive an algorithm in 1D that allows molecules to travel within a prescribed distance per diffusion application and distributed binomially. This means that molecules can travel further than nearest neighbor per diffusion application, in effect “hopping over” regions of space. Although the algorithm includes important concepts such as multinomial distribution and is the first to apply net diffusion transfer, it is unfortunately, in their words, “considerably more complicated” and unimplemented in 2D, and there is no mention at all of 3D systems. The MSA uses the ISSA for reactions, but only allows one reaction anywhere in the system per diffusion update. Although highly accurate, this is very restrictive in terms of performance gain.

Also in 2009 came the first application of operator splitting to irregular subvolumes15 which was further developed in 2010.16 This approach taken by Ferm et al. is to allow 3 different solutions to diffusion: continuous
deterministic, tau-leap or SSA-based, adapted locally based on an error estimate and separated by Strang-splitting. Their algorithm is based on excellent theoretical work and the implementation is shown to be fast and accurate under some conditions. However, there appears to be a cost to error estimation and it is shown that systems must be stiff for there to be any performance gain, with the algorithm in fact performing slower than the ISSA implementation known as the Next-Subvolume Method (NSM) for non-stiff, low molecule systems. Since the “SSA” operation also contained a diffusive term, making the initial algorithm inherently serial, the algorithm was further developed in 2014\(^{17}\) for parallelization. This algorithm is based on a two step Lie-Trotter splitting implementation with an adaptive time step controlled by local error estimates based on the method, and an initial parallel version applied up to 4 cores shows good scaling promise. Of all approaches to date, this algorithm is the most similar to the method we introduce in this paper, but with some important differences as we will describe.

Tau-leaping is an approximate method initially applied to well-mixed reaction systems that reduces computational cost by calculating a small time step “\(\tau\)” over which the event propensities may be approximated as constant and the simulation advanced before updating. Marquez-Lago and Burrage\(^{18}\) introduced spatial tau-leaping and shortly afterwards Rosinelli et al. published an alternative spatial tau-leaping implementation,\(^{19}\) combined with a hybrid system whereby diffusion proceeds deterministically. The Marquez-Lago-Burrage algorithm was modified by Lyengar et al. to improve numerical accuracy,\(^{20}\) and Koh and Blackwell later developed another spatial tau-leaping implementation where diffusion is based on net transfer between subvolumes.\(^{21}\) The tau-leaping approach and operator splitting differ in that operator splitting methods are only approximate in terms of restricting diffusion to predetermined update times and there is no approximation to reaction or diffusion propensities.

The literature suggests many alternatives for a potential approximate method implementation in STEPS and in Secs. II–IV, where necessary, we will investigate mathematically and practically their accuracy and performance with a view to justifying inclusion or exclusion in our algorithm.

II. THEORY

We consider the excellent work in the past decade on ISSA approximation and incorporate some of the ideas in our work, but we take a slightly different approach to many before us by tailoring for the specific goal of scalable parallel computation with maximum performance gain at acceptable accuracy: In order to achieve this, we set out to satisfy three objectives:

(1) Recover reaction and diffusion speeds, expected analytically and simulated by the ISSA. Mathematical theory lays the groundwork for this, but we also place a strong emphasis on validation in practical tests and we will introduce some new models for this purpose as well as using some that we have introduced previously,\(^{1}\) and test a realistic biological model from the literature.

(2) To preserve noise in the system, including spatial variability. This will come at a cost, but in our view is an essential feature of stochastic systems that we do not want to lose.

(3) Allow good scaling by maximizing computation per communication and minimizing the frequency of communication across processes in a parallel implementation. We want to go to the upper-limit of communication period that our requirements in (1) and (2) allow in order to achieve the best performance we can in the parallel implementation.

We use some ideas from the literature that contribute to our goals and reject others that do not, and introduce a novel method to enhance accuracy in the stochastic operator splitting framework.

A. Multi-molecule diffusion

It is well known that a parallelizable solution requires randomized event times in the ISSA to be replaced by predetermined update times where multiple diffusion events are aligned (Figure 1(a)). This is obviously an approximation to the exact algorithm, yet allows reaction and diffusion algorithms to be distributed and computed in parallel, with regular cross-process communication consisting of diffusive transfer. We will refer to this period as “\(\tau\),” though it is important to emphasize that we are not describing a “tau-leaping” implementation.

A common choice for \(\tau\) is to base it on local diffusion rates with restriction of movement to nearest neighbor per update

\[
\tau = \min_m (1/d_m)
\]

where \(d_m\) is the local diffusion rate for a molecule in sub-volume \(m\).

For regular cubic meshes and for any particular diffusing species \(d_m\) is constant throughout the volume, but for irregular tetrahedral meshes \(d_m\) varies throughout the geometry. Defining a local value \(\zeta_m\) for each subvolume as such,

\[
\zeta_m = d_m\tau,
\]

i.e., the ratio of local diffusion rate to the maximum diffusion rate, then it can be seen that \(\zeta_m \leq 1\) and will vary throughout the geometry (Figure 1(c)). \(\zeta_m\) then gives the probability for the molecule to diffuse from subvolume \(m\) at \(\tau\). For multi-species systems, \(\tau\) is simply chosen by the fastest diffusing species and \(\zeta\) varies both per subvolume and per species, but is always less than or equal to 1.

It is well known that reducing \(\tau\) reduces error, and an important concept is that tetrahedral meshes intrinsically reduce error due to the small \(\tau\) compared to regular meshes. In terms of sampling the exact probability density for diffusion, \(\zeta = 0.1\) captures the distribution accurately (Figure 1(d)). For \(\zeta = 1\), as for regular meshes with no artificial \(\tau\) reduction, only the mean diffusion time is sampled and not the underlying probability density function (PDF). The drawback for irregular meshes is that the intrinsically small \(\tau\) necessitates
more computation compared to regular meshes, which may negatively impact performance.

In the example mesh we have presented in previous studies\(^1\) and shown in Figure 1(b), \(\zeta_m\) has a mean value of 0.06 for the fastest diffusing species (and will be lower for all other chemical species) and so the diffusion PDF will be well-sampled (Figure 1(d) shows \(\zeta = 0.1\)). Indeed 43 260 out of 45 661 tetrahedrons in the sample mesh--approximately 95\%--produce a \(\zeta\) value less than 0.1 (Figure 1(c)), so without any special treatment this mesh can intrinsically be assumed to produce accurate diffusion within the operator splitting framework with good sampling of the exact PDFs. As long as the exact PDFs are closely sampled and reaction rates are slower than diffusive changes, it can be assumed that molecules experience similar distributions of occupancy with their reactants counterparts compared to the exact case, which may be important for simulating reaction rates with high accuracy. There is no guarantee that every tetrahedral mesh will produce acceptable values of \(\zeta\). For some meshes, \(\zeta\) may be too small for most tetrahedrons (too irregular), affecting performance (but not accuracy). For others, \(\zeta\) may be too large (too regular). The latter case is easily fixed by reducing \(\tau\) everywhere by some factor (reducing the update period); the former case may be dealt with by implementing an approximation that ignores outliers (effectively allowing \(\zeta > 1\) in those outliers). The number of outliers may be small so the cost to accuracy may be minimal, unless they are in crucial regions in which case they should be left unaltered. The mesh presented in Figure 1, for example, contains only 10 outliers (of 45 661 tetrahedrons overall) where \(\zeta\) is greater than 0.4.

When simulating diffusion in the operator splitting framework, two important concepts must be considered: (1) the number of molecules to diffuse to neighbors per subvolume at \(\tau\), and (2) how they are distributed to the neighboring subvolumes. For every species, the probability of diffusion in the operator splitting framework is closely sampled and reaction rates are slower than di

\[P(\{N\}) = N\zeta - \lfloor N\zeta \rfloor,\]

\[P(\{N\}) = \lfloor N\zeta \rfloor - N\zeta,\]

where of course \(P(\{N\}) + P(\{N\}) = 1\).

This method is demonstrated in Figure 1(e) (gray “Mean” bars). That is, if \(N = 1\), \(\zeta = 0.06\), and so

\[n_{\text{mean}} = N\zeta,\]

which for discrete systems requires rounding to an integer. The most accurate way to do this is to allow the choice of either the floor or the ceiling based on the following probabilities:

\[P(\{0\}) = N\zeta - \lfloor N\zeta \rfloor,\]

\[P(\{0\}) = \lfloor N\zeta \rfloor - N\zeta,\]

FIG. 1. (a) Schematic of the difference between the ISSA and operator splitting for simulating diffusive transfer; in this example the departure of 3 “A” molecules from a tetrahedron to its neighbors. In the ISSA, single molecule events are simulated at unique times, whereas in operator splitting several events are aligned to a predetermined update period, \(\tau\). (b) The example spiny dendrite mesh that is analyzed. (c) The distribution of \(\zeta\) for all tetrahedrons in the mesh shown in (b). (d) The probability density distribution of the diffusive transfer period (the time each molecule resides within a tetrahedron before diffusing out) for the exact ISSA (black) compared to operator splitting (white bars) for a subvolume where \(\zeta = 0.1\), that is, the local diffusion rate of a molecule in that tetrahedron is 10\% of the largest local rate across all tetrahedrons in the mesh (see text). Note: although in operator splitting, transfer takes place at discrete times (right edges of bars), the displayed bars are of full width for visual comparison. (e) Comparison of the number of molecules transferred at each step if chosen binomially (black) or by the discrete mean (gray) for 3 different example molecule numbers, \(N\), of (left to right) 1, 5, and 100, with \(\zeta = 0.06\) in each case.

Another approximation that has been considered before is to base \( n \) on simulating only the net (gradient-based) diffusive transfer between subvolumes. Net diffusion between subvolumes means that transfer only takes place from the higher concentrated subvolume to the lower concentrated subvolume (one way) and no diffusive transfer place at all if concentrations are equal, and is well described elsewhere. The intuitive notion that net diffusion reduces spatial noise has been demonstrated before in regular meshes but may be an accepted loss of accuracy due to the gain in performance allowing simulation of larger systems. We include a brief analysis of how net diffusion may affect a spatial stochastic simulation in irregular meshes in an attempt to quantify this effect. Figure 2(c) shows one example subvolume at one example molecule population, and Figure 2(d) (yellow line) shows the simulated spatial noise compared to the spatial

\[ N \zeta = 0.06 \] then \( n_{\text{mean}} \) equals 0 with probability 0.94 or 1 with probability 0.06, as shown in Figure 1(e) left panel.

The distributions of \( n \) resulting from samples of \( n_{\text{binomial}} \) and \( n_{\text{mean}} \) are only exactly equivalent when \( N = 1 \) but quite different for large \( N \) (Figure 1(e)); so the mean \( n \) may be a good assumption for low concentration systems, but differs significantly from \( n_{\text{binomial}} \) for high density systems. Therefore we tested the effect on spatial noise from \( n_{\text{mean}} \) (Figures 2(b) and 2(d)) measured as simulated coefficient of variation of the stationary distribution averaged across subvolumes and compared to the expected average coefficient of variation of the binomial distribution. We conclude that the binomial distribution should be used so as to avoid under-sampling of noise (Figure 2(d): compare “Gross d, Binomial n” to “Gross d, Mean n”).

**FIG. 2.** Spatial noise simulated by different operator splitting methods. (a)–(c) For a diffusion simulation in a tetrahedral mesh with 1 \( \mu m \times 1 \mu m \times 1 \mu m \) cubic boundaries and mean \( \zeta \) for tetrahedrons = 0.5, for an example tetrahedron the simulated molecule distribution (gray bars) compared to the correct binomial distribution (black bars) if the number of molecules by diffusive transfer, \( n \), is (a) binomial and based on gross diffusive flux, (b) the discrete mean calculated by gross diffusive flux, or (c) the number of molecules is binomial but based on net diffusive flux. (d) Comparison to the analytical coefficient of variation of the three simulated methods shown in (a)–(c) (shapes) and the ISSA (black line) in addition to a “Gross d, Mean n” implementation with 50% \( \tau \) reduction (dashed red line and red squares) with varying system size. The coefficient of variation for a subvolume is the standard deviation of the data relative to its mean, and the coefficient of variations across subvolumes is averaged for the results shown in (d) and compared to the averaged coefficient of variation of the correct binomial distributions.
noise in exact simulations (as a percentage of the coefficient of variation, averaged across subvolumes) for varying molecule densities, and demonstrates the drastic loss of spatial noise by net diffusive transfer for all systems except those of very low molecule density (fewer than 1 molecule per subvolume on average). With only 1 molecule on average per tetrahedron the spatial variance is reduced to approximately 60% of the ISSA, for 10 molecules per subvolume on average the spatial variance is reduced to just 20% of the ISSA, and reduces even more dramatically for higher densities (Figure 2(d)). These results complement previous studies in demonstrating that simulating net diffusive transfer is not desirable if one wants to capture spatial noise in the stochastic system.

The results in Figure 2 are based on a mesh with mean \( \zeta = 0.5 \), and the amplitude of the errors may vary with different values of \( \zeta \). However, Figure 2(d) includes an \( n_{\text{mean}} \) implementation with a 50% \( \tau \) reduction, which still shows significant errors, and in general we assume the results are valid with reasonable ranges of \( \zeta \). For some approximations and models, convergence may occur but perhaps only with very low \( \zeta \) in high molecule density systems at a severe cost to simulation performance. The simulations for net diffusion are based on an exact implementation and are the best possible case for ordinary net diffusion, and so the data shown cannot be improved by \( \tau \) reduction.

B. Molecule distribution

Molecules should be distributed to neighboring subvolumes multinomially.\(^{14}\) To quantify the importance of multinomial distribution we compared to a uniform implementation within the same simple model system. Figure 3(a) demonstrates the importance of multinomial distribution in capturing spatial variance. The multinomial algorithm in the operator splitting framework closely reproduces the output of the ISSA and agrees with the analytical solution to within 1%, even when large \( n \) is transferred per subvolume per update as in the high molecule density simulations. Therefore, diffusion within the operator splitting framework is a very accurate approximation to exact diffusion provided that net transfer is not used, \( n \) is binomial, and distribution is multinomial.

Multinomial distribution can be captured either by randomly selecting a direction for each molecule one by one (\( n \) random numbers required), or applying a multinomial function that distributes \( n \) over the (maximum) 4 neighbors of each tetrahedron (4 or fewer random numbers required). For tetrahedrons, diffusion rate varies for each direction\(^4\) and so the multinomial function should correct weighting for each direction as such: the number of molecules out of a total number of molecules to transfer, \( n \), from the subvolume calculated for direction \( i \),

\[
n_i = \text{binomial}(n - \sum_{j=0}^{i-1} n_j, p_i),
\]

where \( p_i \), the probability for direction \( i \) is

\[
p_i = \frac{w_i}{\sum_{j=0}^{i-1} w_j}.
\]

FIG. 3. Accuracy and performance of the multinomial operator splitting algorithm. (a) For the same model system as shown in Figure 2, a comparison to the mean expected coefficient of variation (CV) across all subvolumes of the ISSA (black line) and operator splitting implementations where molecules are distributed multinomially (circles) or uniformly (squares). (b) Runtimes when the multinomial distribution of molecules is performed individually for each molecule (light dashed line) or by a multinomial function (light solid line) and compared to the ISSA (black line).

where \( w_i \) is the “weight” for the direction in the irregular subvolume and the sum of all weights over the (maximum) 4 directions is equal to 1.

There is a threshold at which the function will perform faster than distributing molecules one by one (Figure 3(b)), which suggests the implementation should incorporate a dynamic switch between the two methods based on local conditions. This threshold depends on the intricacies of computational code but is tested to be approximately \( n = 10 \) in our implementation (Figure 3(b)). Figure 3(b) also demonstrates that there is already a performance benefit of the multinomial algorithm in the operator splitting
C. Reaction-diffusion operator splitting

Reaction-diffusion operator splitting entails alternating some algorithm for diffusion and another for reactions, and there have been several approaches to this in the past. One approach is to apply a diffusion update after every single reaction chosen by the SSA, effectively reducing $\tau$ to the inverse of the zero reaction propensity. The method then removes the reactants before applying the diffusion update, and inputs them upon its conclusion. In tests, in realistic models, however, we find that $\tau$ calculated on diffusion rates is typically orders of magnitude larger than the inverse of the zero propensity and so one may expect a severe performance penalty by reducing $\tau$ so drastically, and this does not fit in with our goal of efficient parallelization. For example, in a published model we calculate approximately a 250x reduction in $\tau$ if it is based on individual reaction events; in other words, if $\tau$ is based on diffusion rates, one might expect ~250 reactions to occur anywhere in the system between diffusion updates, so clearly it is not desirable to base $\tau$ on the inverse of the zero reaction propensity in terms of performance for this model. Some previous methods do allow for multiple reactions per diffusion update, such as in the GMP, but without any special steps taken to consider how diffusion and reactions affect each other during $\tau$ within the operator splitting framework. In our implementation we keep $\tau$ at the upper limit calculated for diffusion to minimize communication in a parallel implementation, therefore allowing multiple reactions during every $\tau$ period. As such we are implementing a first order splitting solution (Figure 4(a)), but with an important enhancement as we will describe in Sec. II D.

D. Mean-corrected diffusive transfer based on $\tau$-occupancy

In our particular implementation, molecule number does not change during $\tau$ by diffusion, which is necessary for a parallelizable solution (see Sec. III). With this in mind, and when allowing multiple reactions to occur between diffusion updates, the problem becomes how to calculate the number of molecules to diffuse to a neighbor at $\tau$, given that the population may have changed during $\tau$ by reactions.

Starting with a hypothetical continuous case, some population $N$ of a molecule present in a subvolume for small time $dt$ with local diffusion rate per molecule of $d$ should contribute $N(t) \cdot dt \cdot d$ to a diffusive transfer applied at $\tau$. The total number of molecules, $n$, for diffusive transfer applied at $\tau$ is therefore

$$n = \int_0^\tau N(t) \cdot dt \cdot d = \overline{N} \cdot \tau \cdot d.$$  

(2)

The value $\tau \cdot d \leq 1$ and is equivalent to the value defined as $\xi$ in Sec. II A.

The schematics in Figure 4(b) demonstrate the behavior for stochastic systems: $N(t)$ is simply a discretization of the continuous case where the diffusive transfer may be written as

$$n = \sum_0^{\tau} N_j \Delta t_j \cdot d$$

$$= d(N_1 \Delta t_1 + N_2 \Delta t_2 + ...)$$

$$= \overline{N} \tau d,$$

as expected.

FIG. 4. Schematic of the reaction-diffusion behavior during operator splitting. (a-i) With 1st order splitting, the reaction operator proceeds for time $\tau$, then the diffusion operator applies sequentially (with initial state including reaction changes to $t+\tau$) for time $\tau$. (a-ii) With 2nd order splitting, the reaction operator proceeds for time $\tau/2$, the diffusion operator proceeds for time $\tau$ (with initial state including reaction changes to $t+\tau/2$), then the reaction operator proceeds for a further time $\tau/2$ to complete the step. (b-i) Within a subvolume by 1st order splitting the diffusion algorithm is applied at $t+\tau$ and so $\overline{N}$, the number of molecules in the subvolume, changes only by reactions during $\tau$. The number of molecules to transfer at $t+\tau$ depends on the mean number of molecules present during period $\tau$ (“$\overline{N}$-mean” in figure), which can only be calculated by keeping track of the changes. (b-ii) shows an example where the median assumption to approximate the mean $N$ during $\tau$ (as is effectively calculated with 2nd order splitting) fails.
And so a correction to the transfer when reactions are present, compared to that calculated in the diffusion-only scenario previously (Theory A Multi-molecule diffusion), is that the transfer is based on $\overline{N}$, that is, the mean $N$ during $\tau$. This simple result has not been applied to previous operator splitting methods to the best of our knowledge. When $N$ does not change during $\tau$, the transfer is simply $n = N \zeta$, or more correctly, $n = \text{binomial}(N, \zeta)$ as derived earlier for diffusion-only systems (Equation (1)). When $N$ changes during $\tau$ the transfer is

$$n = \text{binomial}(\overline{N}, \zeta).$$

This “mean–corrected binomial distribution” will approximate the variance well in the case that the fluctuations around the mean-occupancy $N$ are small (Sam Yates: personal communication). It is possible that $\overline{N}$ is larger than the available $N$ at $\tau$. In such a case it is only possible to base transfer on $N$ at $\tau$, not $\overline{N}$, and this is an unavoidable error.

Our method, as implemented in STEPS, stores for each subvolume a sequence of $N_t \Delta t$ that we call the “occupancy” and requires, computationally, just two extra floating-point numbers per subvolume: the time since the last change and the cumulative occupancy. Just these two numbers allow STEPS to calculate the occupancy during $\tau$, inexpensively, at every diffusion update. The algorithm implemented in STEPS is as follows.

First order splitting (Figure 4(a-i)) is commonly applied in operator splitting implementations$^{17,23}$ without any special steps taken to modify diffusion transfer of $N$ at $\tau$. Higher order splitting methods may effectively calculate diffusion transfer based on approximations such as a linear assumption $(N(t)$ at $\tau/2$-effectively what is calculated with half-step splitting methods such as second order Strang splitting) yet Figure 4(b-ii) demonstrates an occasion where, for stochastic simulation, this approximation differs significantly from the true value of $\overline{N}$ and we can never expect $\overline{N}$ and $N_{\text{median}}$ to be exactly equal whenever there are stochastic changes, and so we expect our method to be at least as accurate as such methods.

**ALGORITHM 1. Descriptive Reaction-Diffusion Operator Splitting algorithm in STEPS.**

Precompute $\tau$ based on model and geometry properties:

$\tau = \min(1.0/d_{m,rec})$ over all local diffusion coefficients $d$, for all tetrahedrons $tet$, and diffusing species $m$.

while ($t < t_{\text{end}}$)

Step 1: Adjust $\tau$ to align to $t_{\text{end}}$ if necessary:

$$t + \tau > t_{\text{end}}?: \tau = t_{\text{end}} - t$$

Step 2: Run SSA to previous reaction before $\tau$:

Step 2.1: When each SSA reaction occurs, for all products and reactants update local occupancy.

Step 3: Run diffusion algorithm at $\tau$:

For each diffusing species:

Step 3.1: calculate $\overline{N}$ based on occupancy

Step 3.2: calculate the number to diffuse to neighbors:

$$n = \text{binomial}(\overline{N}, \zeta)$$

Step 3.3: distribute $n$ multinomially

Step 4: Increase biological time, $t = t + \tau$

We next systematically test our operator splitting implementation for accuracy and performance and compare to other methods in a variety of models designed to capture a wide range of possible scenarios for realistic biological simulation.

**III. RESULTS**

**A. Diffusion validation**

The first results we present are based on our validation set, which largely focus on testing the speed of diffusion within different geometrical settings both for volume diffusion (transfer between tetrahedral subvolumes) and surface diffusion (transfer between surface triangles). We have already presented high accuracy with respect to capturing the spatial variance (Figures 2 and 3). Figure 5 shows an example output for each volume diffusion model as we have described previously, with additional models for surface diffusion as supported in STEPS since version 2.0. In stochastic systems any validation has a chance to fail. However, no increased probability of failure was detected for our operator splitting method, which was always less than 1% as for the exact ISSA implementation.

**B. Reaction-diffusion validation**

One of our reaction-diffusion models allows comparison to an analytical solution to the reaction-diffusion master equation (Ref. 1, Appendix B) and merits deeper investigation, where the expected distribution can be compared to the simulated distribution and the error faithfully measured, both for the ISSA and operator splitting method presented in this paper. The model importantly contains a 2nd order reaction, which is sensitive to the specific reaction-diffusion method and underlying algorithm. We varied system size by varying the production rate constant from a sparse system with less than 1 molecule per tetrahedron on average to a system with multiple molecules on average for every tetrahedron. In each case the analytical solution was fitted to the resulting distribution and an apparent 2nd order reaction rate recovered (Figures 6(a) and 6(b) and see Ref. 1). Figures 6(c) and 6(d) show how the correct reaction rate was simulated to within an error of 1% for all system sizes, and the error in the operator splitting implementation was comparable to the small error in the ISSA.

Clearly, the main advantage of our “mean–corrected binomial distribution” enhancement to 1st order splitting is in capturing molecule distribution well when many molecules are created or destroyed by reactions during $\tau$. The ideal model for exploring this effect is a model “1D diffusion in a finite tube with constant influx at both ends” (Ref. 1, Appendix C), which consists of a stochastic reaction that is restricted to the ends of the cylinder and provides a fast constant influx of molecules (Figure 7(a)). The STEPS model is simulated on a cylindrical mesh and, as the molecules diffuse, after time a spatial gradient occurs (Figure 7(b)) for which there is an analytical solution.$^{1,24}$ This model contains a strong interplay between reaction and diffusion with diffusion speed strongly dependent on the production rate and so is a good test of...
our operator splitting method. Figure 7 compares output from the exact ISSA implementation in STEPS with the operator splitting method introduced in this paper, a first order splitting method (often termed “Lie-Trotter” splitting (“L-T” in figure)) which differs from our method in that $N$ at time $\tau$ is used to calculate the diffusion transfer (Figure 4(a-i)), the same first order splitting method with a 50% reduction in $\tau$, and a second order splitting implementation in which reactions proceed for half a step, diffusion proceeds for the whole step, and reactions proceed for another half step (Figure 4(a-ii)). To add an extra dimension to the tests, we also tested two meshes describing the same overall geometry: a “coarse” mesh of 459 tetrahedrons (mean $\zeta = 0.35$) and a “fine” mesh of 4838 tetrahedrons (mean $\zeta = 0.40$). Since tetrahedron size varied but all other components of the model remained the same, the ratio of the local reaction rate to the diffusion rate (per molecule) is different for the two meshes: $\sim 13$ for the coarse mesh and $\sim 0.5$ for the fine mesh. The simulation was advanced 12 ms to the state shown in Figure 7(b) and concentrations recorded in spatial bins of 0.2 $\mu$m width in all cases and compared to the deterministic concentration. Figures 7(c) and 7(d) show that, although in this model overall the error is quite low for all methods, there is a noticeable advantage for the STEPS method over 1st order splitting, which is somewhat comparable in accuracy to a 50% reduction in $\tau$ or 2nd order splitting. An improvement in accuracy is still clear for the fine mesh, although overall the error is lower for each method. At this point the error in our operator splitting method itself is comparable to the error in the ISSA, the small error which is presumably due to the finite number of iterations that were averaged (10 000).

C. Recovery of reaction probability density functions

An important consideration for any operator splitting implementation is how well it captures the noise compared to...
exact stochastic methods. The underlying goal of stochastic simulation is to faithfully capture reactive noise in the biological system and any approach that fails to do this should be rejected. In order to investigate the intuitive belief that our implementation does indeed accurately capture reactive noise, we implemented a reaction-diffusion model modified from a previous study\(^1\) that contains 10 distinct chemical species diffusing between the range of 10 and 100 μM²/s and reacting by 8 different channels (consisting of 4 reversible reactions) with a broad range of reaction constants from 1 to 1000 μM⁻¹ s⁻¹ for the 4 bimolecular, second-order reactions and 1-100 s⁻¹ for the unimolecular, first-order reactions. The model is described in more detail in Appendix D. The goal of this model is to test a broad range of realistic kinetics that form the building blocks of more complex biological models.

The simulation was run with the operator splitting algorithm in STEPS for 500 s with the period between every single reaction event recorded for the 8 different types of reaction. After approximately 10 s the system reached a steady state (Figure 8(a)) and so the analytical PDF for each reaction based on the mean molecule number could be calculated and compared to the recorded probability density. In every case the simulation captured the expected probability density very closely—Figure 8(b) shows one example: Reaction \#1 A + B > C bimolecular reaction. The simulated probability density could be fit to an exponential distribution with a free rate parameter, and the rate from the best fit compared to the expected rate. Figure 8(c) shows an example least squares fit, again for Reaction \#1, and Figure 8(d) shows the comparison of the fits to the expected rates for all 8 reactions. In each case the simulated rate was recovered to within a 0.5% error and the one standard deviation errors are small indicating a good fit to the data and therefore full recovery of the expected probability density, proving that our operator splitting implementation captures reaction noise accurately over the tested range.
FIG. 7. Reaction-diffusion “1D diffusion in a finite tube with constant influx at both ends” validation model of the operator splitting implementation in STEPS. (a) A stochastic production reaction of constant rate is restricted to the tetrahedrons that border the two faces of the cylinder, and the molecules diffuse towards the center. (b) Example concentration after time 0.1 s along the axis of the cylinder by five methods: the ISSA (black line), operator splitting method in STEPS as described in this paper (cyan dashed line), a first order “Lie-Trotter” implementation with the same $\tau$ as for the STEPS method (green dashed line) and with a 50% reduction in $\tau$ (blue dashed line), and a second order splitting implementation (red dashed line). (c) The percentage error along the axis at the time shown in (b) for the five methods as compared to the deterministic simulation, for a “coarse” mesh of 459 tetrahedrons and “fine” mesh of 4838 tetrahedrons. The error was measured from each boundary to a distance of 1.4 $\mu$m along the axis in 0.2 $\mu$m spatial bins. (d) The box plots showing the range of errors for all spatial bins of the data shown in (c) for both meshes.

D. Realistic biological model simulation

To test the performance of our operator splitting method compared to other splitting methods for a realistic biological model, we chose the Min phenotypes model of Fange and Elf$^{25}$ that has also been used to test other operator splitting implementations.$^{17}$ The model is described in Appendix E. The system was implemented in a tetrahedral mesh describing the bacterial morphology (Figure 9(a)). The system was allowed to evolve for some time in an ISSA implementation, then a snapshot taken of the system state. The system was then evolved for varying periods of time and for varying methods always starting from the same initial system state, and simulation behavior compared for accuracy and performance. Figure 9(b) (top panel) demonstrates the mean and standard deviation of the MinD_ATP_cytoplasm species per tetrahedron for $1 \times 10^6$ iterations with different random number seeds (only 30 example tetrahedrons are shown for clarity), showing visually close agreement between the ISSA and the STEPS operator splitting implementation described in this paper. This agreement was analyzed further in terms of the error in the mean and standard deviation of the STEPS operator splitting implementation compared to the ISSA for every tetrahedron (Figure 9(b), bottom panel). As well as analyzing the STEPS method, a Lie-Trotter first order split implementation with the same $\tau$ as for the STEPS operator splitting implementation
FIG. 8. Operator splitting simulation of a reaction-diffusion model consisting of 10 chemical species and 8 reactions as described further in the text. (a) Small time window of the full 400 s simulation showing that the 10 molecular counts have reached steady state after approximately 10 s. (b) The probability density of simulated reaction periods for the A + B > C reaction, in bins of width 0.003 ms (bars) compared to the expected exponential distribution (black line). (c) For the same data as shown in (b), a least squares fit to the exponential distribution (light dashed line) compared to the analytical distribution (solid black line). (d) For all 8 reactions, the normalized fitted reaction rate (bars) with one standard deviation of the fit (black error bars) (note: displayed range is 90%-110%). The reaction rate was found by performing a least squares fit of the data to the exponential distribution (as shown in (c)) in SciPy.

with a 50% reduction in \( \tau \) were also tested. Figure 9(c) shows the error in comparison to the ISSA for all three methods in terms of mean and standard deviation of molecule number per tetrahedron of the \( 1 \times 10^6 \) runs, with box plots showing the range across all tetrahedrons. There was very little separation between the different methods with very low error overall for this model, though there was a small noticeable improvement with 50% \( \tau \) reduction. The three different methods were similarly difficult to separate statistically for all other species in the model (not shown) suggesting perhaps that sampling error is still present despite the large number of runs sampled. The marginal improvement with 50% reduction in \( \tau \) comes at a cost to performance as shown in Figure 9(d). In this case, although there is only a small or unnoticeable benefit to the STEPS method it comes at very little cost with similar runtime to the simpler Lie-Trotter 1st order splitting method.

Figure 10(a) shows the time evolution of the system in terms of the averaged percentage error in the mean and standard deviation for all five species for the STEPS method, showing small errors initially but gradual increase to up to 3% after 10 s. So, in the long time limit, operator splitting implementations eventually diverge from the ISSA solutionsomewhat akin to different solutions with slightly different meshes or with very small changes in parameters. But in the long time limit it is vital that the approximate method captures the important features of the system. In this case it is the oscillations in minD between different regions of
FIG. 9. The Min phenotypes model implemented with the STEPS operator splitting method and compared to 1st order Lie-Trotter splitting. (a) The wt E. coli geometry represented by a tetrahedral mesh. (b) Comparison between the minD_ATP chemical species number in each cytosolic tetrahedron after 0.1 s evolution from a saved system state for $1 \times 10^6$ iterations with different random number seeds. Top panel shows comparison between the mean (thick lines) and standard deviation (thin lines) by the ISSA (black) and STEPS operator splitting method (cyan dashed line) for 30 tetrahedrons, and the bottom panel shows the percentage error of this comparison. (c) The box plots of percentage error across all tetrahedrons for the minD_ATP chemical species at 0.1 s by 3 tested methods in comparison to the ISSA: Lie-Trotter 1st order splitting, Lie-Trotter 1st order splitting with a 50% reduction in $\tau$ and the STEPS operator splitting method. (d) Computational runtime for 1 biological second by the 3 different methods shown in (c).

the cell (Figure 9(a) shows the regions and see Ref. 25). Figure 10(b) shows the oscillations of minD in region 1 of the cell up to 500s of ISSA and STEPS operator splitting simulation. For 1000s of simulation, the interval between peak and peak number of molecules was analyzed in both regions for both simulation methods. Figure 10(c) shows very close agreement between the ISSA and the STEPS operator splitting method both in the means and standard deviations (in all cases the error in the mean is less than 0.1%). This demonstrates that the operator splitting methods are suitable for accurate simulation of this realistic biological model capturing the key behavior of the model with very high accuracy.

E. Parallel implementation

The operator splitting solution presented in this paper is, by design, friendly to parallelization. Parallel implementations for stochastic reaction-diffusion simulations have been described before, with GPU implementations dominating the particle-tracking field.26,27 and GPU solutions have also been described for voxel-based operator splitting methods on regular meshes.28-30 High Performance Computer (HPC) parallel implementations, for which the de facto standard protocol is the Message Passing Interface (MPI), have gained less attention to date but offer several advantages over GPU implementations such as ease of deployment across different
FIG. 10. Long time limit analysis of the MinD model. (a) Time evolution of the error in mean (solid lines) and standard deviation (dashed lines) of all 5 chemical species from 0.1 s to 10 s. (b) Min D oscillations in cell region 1 captured both by the ISSA (top panel, black) and STEPS operator splitting method (bottom panel, cyan). (c) Top panel: Mean and standard deviation of the interval between MinD peak number for the ISSA (black) and STEPS operator splitting method (cyan) in both region 1 and region 2. Bottom panel: Mean and standard deviation of the maximum number of molecules in each peak by both methods and for both regions.

HPC architectures, to which most researchers have access to in the modern age. There is a report of modest performance gain (~20 speedup for 32 cores) for one tested MPI implementation on regular meshes and for irregular meshes, such as those used by STEPS, to date there is only report of a 4-core “naïve” MPI implementation which achieved a speedup of ~4 for the same model system as analyzed in Sec. III D in this paper. So there are many remaining issues to explore with regards to parallelism specifically in an HPC-MPI setting and for irregular meshes, as well as for the field of stochastic reaction-diffusion operator splitting parallelism in general.

We start by performing some systematic tests of simple models up to realistic biological models in an MPI implementation on compute nodes consisting of two 12-core 2.50 GHz Intel Xeon E5-2680v3 processors sharing 128 GB of system memory and interconnected using 56 Gbit/s InfiniBand FDR. In our parallel implementation, each MPI process simulates a fraction of the complete tetrahedral geometry. At the initial stage, $\tau$ (the period between diffusion applications) is decided globally, as described in Algorithm 1. At the start of an MPI cycle, each process firstly performs reactions by the SSA independently until $\tau$ is reached. When $\tau$ is reached, the diffusion algorithm described in Algorithm 1 step 3 is applied in each process, whilst remote molecule changes and propensity updates are stored as local buffers. These buffers are then synchronized and applied across affected neighboring processes. This completes the update cycle.

First we use a simple uniform diffusion model to perform some initial tests on the performance of the method under different settings. The model consists of 1 chemical species distributed uniformly throughout the volume with varying molecule number and diffusion with coefficient 200 $\mu$m$^2$/s. In each case, the model was run to 0.1 biological seconds and performance averaged over 10 runs. We used different geometrical boundaries describing the same geometrical volume of 100 $\mu$m$^3$ but with different dimensionality: a “1D”
cuboidal boundary $1 \mu m \times 1 \mu m \times 100 \mu m$, a “2D” cuboidal boundary $1 \mu m \times 10 \mu m \times 10 \mu m$, and a “3D” cuboidal boundary $4.64 \mu m \times 4.64 \mu m \times 4.64 \mu m$. All 3 meshes contained approximately the same number of tetrahedrons for comparisons, although two different mesh sizes of $\sim 13000$ tetrahedrons (“coarse mesh”) and $\sim 50000$ tetrahedrons (“fine mesh”) were tested for each. We define “speedup” as the wall-clock runtime for the 1 process simulation divided by the wall-clock runtime for the same simulation over multiple processes. Figure 11(a) shows the MPI performance for a uniform diffusion model with $10^6$ molecules in both the coarse and fine meshes. It is apparent that, whilst significant performance gain is achieved in all models, the number of subvolumes significantly affects the degree of performance gain with the best speedup approximately 100 with 64 cores in the 1D fine mesh model compared to approximately 50 for the coarse mesh. It is also apparent that the dimensionality of the volume influences the parallel performance, which is most noticeable in the fine mesh simulations where the 1D system outperforms the 2D system which outperforms the 3D system. This effect is most likely due to the decreased MPI communication with lower dimensional systems where each partition needs to only communicate with 2 neighbors in the 1D case compared to a maximum of 6 neighbors in the 3D case. Figure 11(b) shows the same simulation in the fine mesh but with 3 different molecule numbers: $10^4$ (“low density”), $10^5$ (“medium density”), and $10^6$ (“high density”). For the low density system, where there is less

FIG. 11. Uniform diffusion model MPI implementation on HPC. For the 1D simulations we performed linear partitioning of the tetrahedral mesh along the long z-axis. For the 2D simulations we performed linear partitioning on the long y and z axes by the following number of x-y-z partitions: 1-2-2 (4 processes), 1-3-3 (9 processes), 1-4-4 (16 processes), 1-5-5 (25 processes), 1-6-6 (36 processes), 1-7-7 (49 processes), and 1-8-8 (64 processes). For the 3D simulations we performed linear partitioning on all 3 axes as such: 2-2-2 (8 processes), 3-3-3 (27 processes), and 4-4-4 (64 processes). (a) Simulation speedup relative to 1 process performance for meshes of $\sim 12000$ tetrahedrons (left panel) and $\sim 50000$ tetrahedrons (right panel) with varying dimensionality. (b) Parallel performance with varying molecules density on the same meshes of $\sim 50000$ tetrahedrons and varying dimensionality. (c) Best parallel performance in terms of speedup to serial ISSA for uniform diffusion model in 1D for $30000$, $300000$, and $3000000$ molecules.
than 1 molecule on average per tetrahedron, performance gain saturates at around 32 processes. For the medium density (~2 molecules per tetrahedron on average) and high density (~20 molecules per tetrahedron on average) systems performance gain is much more impressive and actually maximum in the medium density system. This is likely due to reduced MPI communication in the medium density system where many bordering tetrahedrons will be unoccupied per update, whereas MPI communication is higher for the high density system.

Figures 11(a) and 11(b) demonstrate occasions where the speedup for p processes is greater than p; that is parallel efficiency is greater than 1.31 This somewhat counter-intuitive phenomenon is termed “superlinear speedup,” for which one cause is the improvement of memory caching effects in parallel computing.32 In the parallel STEPS implementation, reaction and diffusion rules as well as their update dependencies are stored only in the MPI process where they are simulated. As the number of processes increases, the memory consumption of the simulation for each individual process decreases, meaning that a larger proportion of the overall model can be stored in cache. This allows the simulation to achieve superlinear speedup in comparison to the 1 process simulation.

So there are many factors at play influencing the performance gain of the parallel system including geometric dimensionality, mesh size, and molecule density, with the parallel distribution of the reaction-diffusion system, caching effects, and cross-process MPI communication underlying the performance. Generally speaking, by increasing the number of processes more diffusion events are executed simultaneously, which improves performance; however, there is a decrease of efficiency because using more processes inevitably increases the amount of cross-process diffusion events. All models will be susceptible to such considerations and the peak parallel performance will depend on these factors for the particular model at hand.

Figure 11(c) displays the peak parallel performance with varying number of molecules, as similarly reported by Gladkov and Andrews for a GPU implementation of Smoldyn.27 The model was run on the 1D “coarse mesh” and the number of processes where maximum speedup occurs varies with number of molecules in our MPI implementation, from 48 processes for 30 000 molecules to 240 processes for 3000 000 molecules. This speedup is similar in order of magnitude to the report by Gladkov and Andrews where a speedup of ~300× was reported for the 3 000 000 molecule case on GPU.

To investigate the performance of realistic reaction-diffusion models, we created MPI versions of two published voxel-based models: the minD phenotypes model25 also used in Sec. III D, and the cAMP model of Oliveira and colleagues1 described in Appendix F. The cAMP model was also used to benchmark the STEPS operator splitting method against experimental data, as described in Appendix F. Figure 12(a) shows close agreement between simulation and experiment.

For the MPI simulations, the minD model was run on mesh geometry of ~14 000 tetrahedrons describing filament geometry and the cAMP model was run on two meshes of the same geometry; one with ~26 000 tetrahedrons (“fine mesh”) and one with ~1000 tetrahedrons (“coarse mesh”). Figure 12(b) shows the performance of each model up to 96 MPI processes on 96 cores with each geometry partitioned linearly on its principal axis. Each model shows peak performance at different stages: the minD model speedup peaks at 36 processes with a speedup of ~24. The peak performance of the cAMP model differs for the two meshes: a speedup of ~32 for the coarse mesh at 72 processes, and a speedup of ~52 at 96 processes for the fine mesh which is clearly approaching performance peak.

The cAMP geometry consists of a cubic boundary that is significantly longer on the x and y axis than the z axis with dimensions 7.5 µm × 3.72 µm × 0.93 µm³ and merits deeper investigation. The highlighted square on Figure 12(b) shows performance with linear partitioning along the longest x axis, and this partitioning was investigated further with two-dimensional partitioning on the x and y axis (Figure 12(c)). Different degrees of x, y partitioning for the same partition number of 48 were investigated and the peak performance found at a partitioning of 24 on the x axis and 2 on the y axis. All other partitions showed some performance cost from approximately 5% up to almost 30% in the case of a partition of 1 on the x axis and 48 on the y axis.

These results demonstrate that partitioning is another factor that affects MPI performance, and different partitions should be investigated for the particular problem at hand for best performance, which could avoid a significant performance cost compared to a naïve choice of partition.

IV. DISCUSSION

A. ISSA approximation methods

Based on the literature and our own investigations presented in this paper, we briefly discuss the possibilities for ISSA approximation in tetrahedral meshes with particular regard to their suitability of parallel implementation, their strengths and weaknesses, and our justification for inclusion or rejection in our solution for the STEPS software.

1. Tau-leaping

Clearly the first step to achieving a scalable parallel solution is to replace the randomized event times in the ISSA (or similar method) with predetermined execution times. For the reaction-diffusion system the simulation can advance to without the necessity of communication for diffusion, which could be cross-process in a parallel implementation. The question becomes rather how to select a tau and which algorithm to implement during tau. The term “tau-leaping” has become synonymous with a method of advancing a simulation time by an amount “tau” where propensities are not allowed to change and event frequency is calculated on starting propensities. Clearly this fits on the umbrella of predetermined execution times, but comes at the
FIG. 12. MPI implementation of two realistic biological models on HPC. (a) The cAMP model of Oliveira et al. is implemented in the STEPS operator splitting solver and benchmarked against experimental data, both in terms of (a-i) the steady state dose-response FRET signal as a function of cAMP concentration and (a-ii) the time course FRET signal for delivery of 30 µM cAMP. One stochastic run of the operator splitting algorithm was used for each simulation. (b) In MPI simulations, speedup of the cAMP model and minD model relative to the 1 process simulation. The cAMP model is run on two different meshes with ~1000 tetrahedrons (triangles) and ~26,000 tetrahedrons (squares) describing the same geometry. (c) The 48 process implementation of the cAMP model on the fine mesh (highlighted square in (b)) is run with different x,y linear partitions always with a total partition number of 48. Performance is shown as percentage wall-clock increase compared to the best case partitioning that was found to be 24x, 2y.

expense of the extra computation required to calculate tau and broadcast at every step in the parallel process. There is also, by design, an error introduced, which is controlled but for low-molecule systems means that tau must be very small, basically reducing the algorithm to the serial ISSA for low molecule systems with worse performance because of the
2. Operator splitting

A scalable solution requires maximizing the computation that takes place on each process between cross-process updates. For arguments that we have presented, the best way to achieve this is to separate reaction from diffusion and deal with them separately, similarly to previous approaches.\textsuperscript{12,17,23}

3. Net diffusive transfer

Although expected to significantly reduce communication, we reject a net diffusion approach because of the cost to spatial noise (Ref. 21 and Figures 2(c) and 2(d)), the capture of which is one of the important features of a our spatial stochastic simulations. To reduce the spatial variance error associated with net diffusion one may introduce a factor that controls the degree to which gradient-based diffusion is applied,\textsuperscript{23} but capture of full spatial variance is only possible with no degree of net diffusion at all.

4. Multinomial distribution

Multinomial distribution of molecules is essential to maintain spatial noise in comparison to the exact ISSA (Figure 3(a)). For this reason we choose to implement multinomial diffusion. Our diffusion is to nearest neighbor, in part to avoid “hopping over” regions of space that may be important. Our preference is to track a molecule’s exact path through the discretized space, even if at altered leap times in comparison to the exact ISSA, which is unavoidable to achieve a scalable parallel solution. We have shown that our algorithm captures both spatial noise caused by stochastic diffusion (Figures 2 and 3) and diffusion speeds (Figures 5 and 7) accurately.

5. Adaptive \( \tau \)

We believe that operator splitting in irregular meshes operates with high accuracy because \( \tau \) is intrinsically kept small in irregular meshes (Figure 1), and can also be enhanced in some settings by considering how stochastic reactions affect local diffusion propensities (Figures 4 and 7). Accuracy can be further improved by reducing \( \tau \), although this comes at a runtime cost (Figure 9(d)). Tau-leaping and error-estimation approaches allow for adaptive \( \tau \), although this comes at a performance cost (in one example error estimation is reported to perform \( \sim \)20 times slower than the exact ISSA in non-stiff systems\textsuperscript{36}). In a parallel setting, adaptive \( \tau \) will only improve performance if the cost of calculating the \( \tau \) and broadcasting it is less than gain in performance from potentially calculating a larger \( \tau \) on average. In our implementation, the \( \tau \) depends only on the properties of the mesh and the model, and is therefore fixed, reducing the cost of calculating the \( \tau \) adaptively.

To look at the potential cost for calculating \( \tau \) in realistic models we investigated the cost of the simple \( \tau \) calculation by the STEPS method. In brief, this method loops over all local diffusion rules in tetrahedrons and triangles to find the fastest local diffusion rate. Table I shows the cost of this simple \( \tau \) calculation on a 2.8 GHz Intel Core i7 processor of \( 1 \times 10^6 \) clock ticks/s. The processor time to recalculate \( \tau \) is a minimum of 0.24 s in a small mesh for a relatively small model (5 species, 5 reactions) and increases with mesh size to cost 22 s in a mesh of 49 339 tetrahedrons even for a simple diffusion model. The two important points are: (1) that it takes significant computational time to calculate \( \tau \) even by our simple method up to 10 s for the larger meshes tested, and (2) \( \tau \) itself is small in biological time in the range \( \sim 1 \) \( \mu \)s-1 ms. These two considerations mean that frequent recalculations of \( \tau \) are impractical for irregular meshes and, for all but the smallest of meshes and simplest of models, will completely dominate computation over the reaction-diffusion calculation. For example, for the cAMP model on the mesh of 26 369 tetrahedrons if \( \tau \) is recalculated at the highest frequency possible, that is after a period of \( \tau \) itself, then the total cost to recalculate \( \tau \) for a simulation of 1 biological second would take \( 13 \) s/\( 1.4 \times 10^{-6} \approx 10^7 \) s \( \approx 100 \) days! This mesh size is not unrealistically large and some models may in fact employ

\begin{table}[h]
\centering
\caption{The time cost for one \( \tau \) calculation for the STEPS simulations in this paper.}
\begin{tabular}{llll}
\hline
Model (Figure #) & Number of tetrahedrons in mesh & Processor clock ticks to calculate \( \tau \) & Processor time to calculate \( \tau \) (s) & \( \tau \) (biological time) \\
\hline
1D diffusion with constant influx model (7) & 12 033 & 1 488 522 & 1.5 & 1.8 \( \mu \)s \\
Min D phenotypes model (9, 10, 12) & 175 & 236 755 & 0.24 & 2.1 ms \\
& 14 530 & 3 222 223 & 3.2 & 0.13 ms \\
cAMP model (12) & 1 122 & 619 836 & 0.62 & 16 \( \mu \)s \\
& 26 369 & 13 031 201 & 13 & 1.4 \( \mu \)s \\
10 species, 8 reaction model (8) & 447 & 246 601 & 0.25 & 75 \( \mu \)s \\
Uniform diffusion model (11) & 13 983 & 1 993 889 & 2.0 & 8.9 \( \mu \)s \\
& 49 339 & 2 227 9943 & 22 & 3.4 \( \mu \)s \\
\hline
\end{tabular}
\end{table}
larger meshes of 100 s of thousands of tetrahedrons,\textsuperscript{7} and it is completely impractical to recalculate $\tau$ frequently for such meshes.

Although an adaptive $\tau$ could be calculated in parallel, which would help to reduce the cost of calculating it, it would also negatively impact performance by the necessity of broadcasting $\tau$ at every step on a parallel implementation. Therefore, we believe there is a big advantage in computation efficiency by our method of implementing a static $\tau$. Note, however, that in STEPS reaction and diffusion rates may be altered during the simulation, which can affect $\tau$. In such cases only, $\tau$ is recalculated as necessary.

**B. Our operator splitting method**

1. **Accuracy**

We have developed an operator splitting method based on the practicalities of biological reaction-diffusion systems and considering stochastic features that are not present in deterministic systems. Instead of employing half time step methods, we tailor our method to capture discrete events accurately which can affect diffusion speed (Figure 7). In capturing molecule distribution based on reactive changes, our method performs at least as accurately as 2nd order splitting or 50\% reduction in $\tau$. The usefulness of the approach depends on factors such as mesh size, $\zeta$ and model specification. Where errors are already low the effect will also be low (Figure 9c)), but since performance cost is minimal (Figure 9d)) it can be employed safely to enhance accuracy compared to 1st order splitting, however significantly or insignificantly. In addition we have quantified the necessity to distribute diffusion transfers multinomially and to avoid net diffusion if one wishes to capture spatial noise faithfully (Figures 2 and 3). Capturing the correct, stochastic molecule distribution is by far the most important concept for simulating spatial reaction rates correctly and our method is proven to recover reaction rates (Figure 6) and probability densities (Figure 8) accurately under a wide range of realistic biological conditions.

2. **Parallel performance**

Parallel computing enhances the performance gains already evident in voxel-based modeling, which can be orders of magnitude faster than particle-based solutions\textsuperscript{1,4} and so perhaps offer the best opportunity of maximizing the reach of stochastic reaction-diffusion models. Although peak performance gain of operator splitting implementations in a parallel setting depends on model specification, in our tests a gain of at least 1-2 orders of magnitude can be expected for realistic biological models (Figure 12), which easily outperforms direct parallelization of exact methods where the necessary system of checkpointing and rollbacks typically keep performance gain to less than an order of magnitude.\textsuperscript{10,11,34}

We confirm the intuitive notion that there are many factors that can affect parallel performance, including dimensionality and the choice of mesh partitioning (Figures 11 and 12). For parallel implementations of realistic models, therefore, it is vital that performance is tested for optimal cluster use. Peak performance should be found and the maximum number of processes set accordingly for any simulation, when further parallelism can be exploited by the embarrassingly parallel multiple iteration solution. This is an advantage for MPI implementations compared to those on the GPU: MPI simulations can be run on any number of nodes that are available and that make sense for the particular problem at hand. Due to the wide range of different reaction-diffusion models and complex MPI implementation considerations, there are many more questions regarding the scalability to explore. For example, for some models of low concentration and with spatial gradients dynamic load balancing may be beneficial, which will however come at a cost in the load balancing calculation. And how well does the implementation adapt to more complex models at larger scale with massive number of processes? We plan to address these questions systematically in a future study.

**3. Comparison to other parallel implementations**

There are an increasing number of parallel reaction-diffusion implementations, but often for different hardware and methods and with limited reports of parallel performance making direct comparisons between methods difficult. We complement the literature on parallel performance with more detailed reports of accuracy and efficiency than are usually reported, and we expect many of the important conclusions to be consistent for voxel-based methods regardless of the fine details of the implementation. To our knowledge, we present the first results of significant speedup in MPI implementation of reaction-diffusion systems up to 2-3 orders of magnitude. The MPI implementation of Hellander et al.,\textsuperscript{17} which is most similar to the STEPS method, also shows promise for good scaling but only with a report up to 4 cores so far and to our knowledge has not yet been reported for more complex models and in a larger scale parallel setting.

GPU implementations have been described for the particle-based software Smoldyn—the Gladkov\textsuperscript{27} and Dematte\textsuperscript{26} implementations—and also for voxel based methods for regular meshes by a parallel implementation of the GMP\textsuperscript{30} and by an algorithm termed the MPD-RDME,\textsuperscript{28} and so we can draw some comparisons to our MPI method. Firstly, we believe MPI implementations are more easily ported between different architectures and the parallel STEPS code is ready to run on any HPC cluster where certain common dependencies such as Python and MPI are available. Portability is not necessarily a major problem for GPU, but older code may need to be modified for optimal performance. A disadvantage of the Gladkov and Dematte GPU Smoldyn implementations is that neither are merged into the main Smoldyn branch nor maintained, and they would benefit from further development and updates. Neither implementation supports all features of Smoldyn yet: for example, the Gladkov implementation only operates on a cubic domain with no complex geometry and the Dematte implementation only has limited surface and molecule-surface interaction. Our STEPS implementation contains all features of the serial reaction-diffusion solver, including complex morphology support. The usefulness of
the GPU Smol'dyn implementations depends strongly on the molecule number,\textsuperscript{27} with good speedup of 2 orders of magnitude for systems with millions of molecules. This seems to be a common feature of GPU implementations since the MPD-RDME implementation reports a speedup of only $4.5 \times 7.5 \times$ for a low molecule count simulation but $300 \times$ for a high molecule count simulation.\textsuperscript{28} and similarly the GPU implementation of the GMP reports a 2 order of magnitude speedup for a similar homogeneous diffusion problem.\textsuperscript{30} Our MPI implementation performance also strongly depends on molecule number, and we also find other factors are significant such as mesh size, which is already known to affect parallel performance, but also the dimensionality of the system. The minimum peak speedup we found in all our tests was $\sim 20 \times$. In a simple diffusion model and in comparison to the serial ISSA we see maximum speedup of $\sim 30 \times$ for the 30 000 molecule simulation and $\sim 900 \times$ for the 3 000 000 molecule simulation (Figure 11), which compares similarly but favorably to the Gladkov report on a similar model with parallel particle-based methods. An MPI implementation is able to set an arbitrary number of processes for the calculation, which the researcher may choose to set at peak speedup, then parallel realization exploited by assigning more runs of the same model with different random number seeds over the other available nodes.

V. CONCLUSIONS

Recent years have seen the advance of techniques of spatial stochastic simulation, adding ever more realism to biological models. The ISSA and its variants have become established methods within this field and, although more work needs to be done to verify performance of spatial stochastic methods in general against real biological data, this is outside the realm of this study. Instead, we focus on gauging performance of operator splitting methods in terms of accuracy (relative to the ISSA) and parallel performance—a necessity that has arisen due to the intense computation and restricted scale of serial methods.

In this study we have presented an operator splitting implementation for simulating reaction-diffusion systems on irregular meshes, with a novel method to enhance accuracy. The method has proven to be accurate with respect to analytical, deterministic, and exact stochastic simulations under a wide range of tested conditions, including for realistic biological models. An MPI implementation shows good promise for speeding up such models in a parallel setting, although peak speedup is model and partition-dependent. As such, the method and results we present are another step in advancing the scope of spatial stochastic simulators by helping to overcome the speed bottleneck within the field.

ACKNOWLEDGMENTS

This work was funded by the Okinawa Institute of Science and Technology Graduate University. All MPI simulations were run on the “Sango” cluster at the Okinawa Institute of Science and Technology.

We are very grateful to Sam Yates of the Blue Brain Project, EPFL, Geneva, for his critical review of the initial draft of this manuscript, which led to several improvements.

APPENDIX A: SOFTWARE AVAILABILITY


APPENDIX B: “PRODUCTION-DEGRADATION” MODEL OF SEC. III B

The model was first described by Erban and Chapman\textsuperscript{35} and contains a degradation of A reaction

$$A + B \xrightarrow{k_1} B.$$ 

And a production of A reaction

$$A \xrightarrow{k_2}.$$ 

The expected stationary distribution of A is found from the steady-state version of the chemical master equation\textsuperscript{35}

$$\Phi(n) = \frac{1}{n!} \left( \frac{k_2 e^{2/n}}{k_1 B_0} \right) \exp \left( -\frac{k_2 e^{2/n}}{k_1 B_0} \right).$$

The degradation rate, $k_1$, was fixed at a value of $k_1 = 100$ $\mu$M$^{-1}$ s$^{-1}$, but the production rate, $k_2$, was varied in order to vary the number of molecules in the system. The values of $k_2$ were 2 nM/s, 10 nM/s, 20 nM/s, and 40 nM/s, which resulted in a mode (peak) number of molecules of 7, 36, 72, and 145, respectively, as shown in Figure 6. All molecules diffused with a diffusion coefficient of 20 $\mu$m$^2$/s. The model was run in a mesh comprised of 73 tetrahedrons and contained within cubic boundaries of 1 $\mu$m $\times$ 1 $\mu$m $\times$ 1 $\mu$m for 10 000 s to negate sampling error in the molecule distributions shown in Figure 6.

APPENDIX C: “1D DIFFUSION IN A FINITE TUBE WITH CONSTANT INFUX AT BOTH ENDS” MODEL OF SEC. III B, FIGURE 7

The model was run on a tetrahedral mesh describing cylindrical geometry of length 10 $\mu$m and diameter 1 $\mu$m. A production reaction was restricted to either face of the cylinder

$$A \xrightarrow{k} A + X.$$ 

The reaction rate, $k$, was set so as to result in a total flux of X molecules at each boundary of 300 000 s$^{-1}$ with one molecule of A present in each boundary tetrahedron. Molecules of X diffused freely by diffusion coefficient 50 $\mu$m/s. As stated in the text, 2 different meshes were used of 459 tetrahedrons and 4838 tetrahedrons so as to vary local production rate to local (per tetrahedron) diffusion rate. For the 459 tet mesh, $k$ was set to 50 000 s$^{-1}$, giving a local reaction to diffusion ratio of $\sim$13 and for the 4838 tet mesh, $k$ was set to 11 111 s$^{-1}$, giving a local reaction to diffusion ratio of $\sim$0.5 (note that local diffusion rate also varies with the different sized meshes).
The simulations were run for 12 ms for the results shown in Figure 7.

**APPENDIX D: REACTION-DIFFUSION MODEL OF SEC. III C**

There are 10 different chemical species in the model, A-J. The initial number of molecules and diffusion coefficients for each species are as follows.

A: 100 molecules: 100 µm²/s
B: 200 molecules: 90 µm²/s
C: 300 molecules: 80 µm²/s
D: 400 molecules: 70 µm²/s
E: 500 molecules: 60 µm²/s
F: 600 molecules: 50 µm²/s
G: 700 molecules: 40 µm²/s
H: 800 molecules: 30 µm²/s
I: 900 molecules: 20 µm²/s
J: 1000 molecules: 10 µm²/s

And they interact by the following reaction channels and constants:

1. A + B → C: 1000 µM⁻¹ s⁻¹
2. C → A + B: 100 s⁻¹
3. C + D → E: 100 µM⁻¹ s⁻¹
4. E → C + D: 10 s⁻¹
5. F + G → H: 10 µM⁻¹ s⁻¹
6. H → F + G: 1 s⁻¹
7. H + I → J: 1 µM⁻¹ s⁻¹
8. J → H + I: 1 s⁻¹

The model was run in space within a 3×3×3 µm cubic boundary comprised of 447 tetrahedrons.

**APPENDIX E: MIN PHENOTYPES MODEL OF SEC. III D AND SEC. III E**

The model comprises 5 chemical species that interact in the cytosol and cell membrane:

1. MinD_ATP_cytosol → MinD_membrane: 0.0125 µm⁻¹ s⁻¹
2. MinD_ATP_cytosol + MinD_membrane → 2MinD_membrane: 9 × 10⁶ M⁻¹ s⁻¹
3. MinE_cytosol + MinD_membrane → MinD_membrane: 5.56 × 10⁷ M⁻¹ s⁻¹
4. MinD_membrane → MinD_ADp_cytosol + MinE_cytosol: 0.7 s⁻¹
5. MinD_ADp_cytosol → MinD_ATP_cytosol: 0.0125 µm⁻¹ s⁻¹

The model was run in a tetrahedral mesh describing the “wild-type” bacterial morphology of length 3.5 µm and radius of 0.5 µm and comprised of 175 tetrahedrons for the results shown in Figures 9 and 10, and in “filamentous” cells of length 10 µm and radius 0.5 µm comprised of 14 530 tetrahedrons for the results shown in Figure 12. The model is described further in Ref. 25.

**APPENDIX F: HEK 293 CELL MODEL OF SEC. III E (“cAMP model”)**

The model contained the reactions shown in Table II. The original morphology of Oliveira et al. was a grid of 0.93 µm × 0.93 µm × 0.5 µm cuboidal subvolumes arranged in a spatial grid of 4×15. The equivalent geometry was represented by tetrahedral meshes for the STEPS simulations and similarly compartmentalized into a cytosolic compartment and a submembrane compartment. Initial concentrations were set as specified by Oliveira et al. and 4 chemical species diffused with the following rates:

ATP: 255.34 µm²/s, cAMP: 294.91 µm²/s, AMP: 289.72 µm²/s, PKAc: 59.54 µm²/s.

Meshes with varying numbers of tetrahedrons were used for the MPI simulations, as described in the text.

**Table II. Reactions and rate constants of the HEK 293 cell model (“cAMP model”).**

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Kf (µM⁻¹ s⁻¹)</th>
<th>Kb (s⁻¹)</th>
<th>Kcat (s⁻¹)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G4ATP + AC ⇌ E</td>
<td>38.5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>E + ATP ⇌ EATP</td>
<td>0.128</td>
<td>0.261</td>
<td></td>
</tr>
<tr>
<td>E + cAMP ⇌ EATP</td>
<td>0.259</td>
<td>28.46</td>
<td></td>
</tr>
<tr>
<td>2(cAMP) + PKA ⇌ PKARcGcAMP2</td>
<td>0.087</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>2(cAMP) + PKARcGcAMP2 ⇌ PKARcGcAMP4</td>
<td>0.15</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>PKARcGcAMP4 + 2(PKAc) ⇌ PKARcGcAMP4</td>
<td>1.7</td>
<td>0.016</td>
<td></td>
</tr>
<tr>
<td>cAMP + PDE4B ⇌ PDE4BcAMP ⇌ AMP + PDE4B</td>
<td>30</td>
<td>77.44</td>
<td>19.36</td>
</tr>
<tr>
<td>PKAc + PDE4B ⇌ PKAcPDE4B → pPDE4B + PKAc</td>
<td>0.3375</td>
<td>0.408</td>
<td>0.417</td>
</tr>
<tr>
<td>PKAc + PDE4BcAMP ⇌ PKAcPDE4BcAMP → pPDE4BcAMP + PKAc</td>
<td>0.3375</td>
<td>0.408</td>
<td>0.417</td>
</tr>
<tr>
<td>cAMP + pPDE4B ⇌ pPDE4BcAMP ⇌ AMP + pPDE4B</td>
<td>30</td>
<td>77.44</td>
<td>27.10</td>
</tr>
<tr>
<td>pPDE4B → PDE4B</td>
<td>0.01088</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cAMP + PDE4D ⇌ PDE4DaAMP ⇌ AMP + PDE4D</td>
<td>12</td>
<td>58.82</td>
<td>14.70</td>
</tr>
<tr>
<td>PKAc + PDE4D ⇌ PKAcPDE4D → pPDE4D + PKAc</td>
<td>0.625</td>
<td>0.00544</td>
<td>0.00556</td>
</tr>
<tr>
<td>PKAc + PDE4DaAMP ⇌ PKAcPDE4DaAMP → pPDE4DaAMP + PKAc</td>
<td>0.3375</td>
<td>0.408</td>
<td>0.417</td>
</tr>
<tr>
<td>cAMP + pPDE4D ⇌ pPDE4DaAMP ⇌ AMP + pPDE4D</td>
<td>24</td>
<td>58.82</td>
<td>92.58</td>
</tr>
<tr>
<td>pPDE4D → PDE4D</td>
<td>0.01088</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMP → ATP</td>
<td>1.085</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H30 + cAMP ⇌ H30cAMP</td>
<td>0.001221</td>
<td>0.0217</td>
<td></td>
</tr>
</tbody>
</table>
To benchmark the STEPS operator splitting performance against experimental data, we compared a simulated Fluorescence Resonance Energy Transfer (FRET) signal, calculated from simulated concentrations of cAMP-bound-H30 and free H30 sensor, to the original experimental data of Oliveira et al.\(^1\) The interaction was included by adding the reversible reaction to the model (Table II).

Following the same protocol as Oliveira et al., we calculated the simulated FRET ratio, R, as such

\[
R = \text{Cyan Signal}/\text{Yellow Signal},
\]

\[
\text{Cyan Signal (CS)} = (1 - 0.35)[\text{H30}] + [\text{H30cAMP}],
\]

\[
\text{Yellow Signal (YS)} = 0.35[\text{H30}] + 0.12([\text{H30}]
\]

\[
+ [\text{H30cAMP}]) + 0.67(\text{cyan signal}).
\]

The simulated change in R relative to basal R = 0.82 was calculated for 0.266 \(\mu\)M cytosolic H30 with varying doses of cAMP and compared to the experimental data of Oliveira et al. (Figure 12(a-i)). There was very close agreement between the STEPS operator splitting performance and experiment, and essentially identical performance to the original simulations of Oliveira et al. (compare to Figure 2(A) in Ref. 4). In addition, the time course of simulated and experimental FRET ratio relative to the pre-stimulus ratio (R\(_0\)) by delivery of 30 \(\mu\)M cAMP were compared, also showing excellent agreement (Figure 12(a-ii)). Note that for the STEPS simulations, one stochastic run of the operator splitting algorithm is displayed to mimic experimental conditions closely whereas a deterministic solution was used for benchmarking by Oliveira et al. (Figure 2(B) in Ref. 4).


